ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA IDENTIFICACIÓN DE PRODUCTOS EN UN SUPERMERCADO CON TECNOLOGÍA RFID (IDENTIFICACIÓN POR RADIO FRECUENCIA)

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

CÓRDOVA TORRES JUAN CRISTÓBAL
jcct_81@hotmail.com
VELASTEGUI SÁNCHEZ KARLA VALERIA
vale904@gmail.com

DIRECTOR: ING. PABLO HIDALGO
phidalgo@ieee.org

Quito, Diciembre 2008

DECLARACIÓN

Nosotros, Karla Valeria Velastegui Sánchez y Juan Cristóbal Córdova Torres, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Karla Valeria Velastegui Sánchez	**Juan Cristóbal Córdova Torres**

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Karla Valeria Velastegui Sánchez y Juan Cristóbal Córdova Torres, bajo mi supervisión.

Ing. Pablo Hidalgo

DEDICATORIA

Este Proyecto de Titulación va dedicado a Dios, a ese ser maravilloso que me puso siempre donde debí estar. Por nunca dejarme sola, brindarme su amor incondicional y por darme la mejor madre del mundo.

Le dedico de forma muy especial a mi madre quien siempre ha luchado por sacarnos adelante sobre todas las cosas. Te agradezco porque si soy quien soy es gracias a ti. Por estar pendiente siempre de hasta lo más pequeño. Gracias por abrazos, besos y por ser la que está al pie de mi cama cuando estoy enferma o triste. Te adoro mamita.

A mi padre gracias por estar pendiente de mí. Por brindarme su apoyo en el estudio. Gracias por confiar en mí. Gracias por tantos cursos. Gracias por ser mi padre. Te adoro papi.

A mis hermanos. A Picolin (mi Nico querido) que siempre me apoyó aún sin saber qué reto era el que iba a atravesar, él confió en mí. A Tali por todas esas noches de estudios juntas, por las risas en las noches para no dejar dormir al resto de la casa.

A mis abuelitos que aunque a la distancia me apoyaban, gracias por confiar en mí.

Y no puedo olvidarme de Miguelito, gracias por ser mi amigo más que mi tío, por todas esas charlas llenas de sinceridad y confianza.

A Mario por todas esas historias que saben transportarme como una protagonista a ellas, por brindarme la magia y envolverme en ella.

Al Ing. Naranjo por ser mi amigo, por ayudarme cuando nadie encontraba salida, por sentirme de lejos y enviarme consejos a la distancia.

AGRADECIMIENTO

Agradezco a todas aquellas personas que me han apoyado de una u otra forma.

A Marquito y al primo de Juan por habernos ayudado desinteresadamente.
A Viche por no solo en este trabajo sino siempre me ha ayudado y apoyado mucho. Por explicarme como a una niña pero con mucho respeto todo lo que necesitaba.

A Lobito, David y Mau por estar en esos primeros años de carrera junto a mí. Siempre compartiendo trabajos y tiempo. Lobito por ayudarnos a todos siempre, por ser un gran amigo. Para David, hoy mi mejor amigo, eres muy especial, gracias por escuchar cosas que nadie más ha escuchado. Mau por estar ahí en toda la carrera apoyándome y queriendo lo mejor para mí.

Al Ing. Pablo Hidalgo por ser un apoyo y darnos ánimos cuando ya no encontrábamos salida y reprendernos si era necesario.

A aquel ingeniero que en los primeros años no creyó en mí, ¡pronto seré ingeniera, esta profesión si es para mí!

A todos aquellos que pusieron trabas en mi camino por qué me dieron más ánimos para seguir.

Gracias a todos ustedes los quiero mucho.
Siempre los llevaré en mi corazón.

Karla Valeria Velastegui Sánchez

DEDICATORIA

“ Este proyecto quiero dedicar a las siguientes personas:

En primer lugar a mis padres Cristóbal Córdova y Elvira Torres que siempre han estado a mi lado dándome mucho amor y comprensión para seguir adelante .

A mis hermanos David y Oswaldo que son los mejores amigos que puedo tener, y a mi preciosa hermanita Náyelhi que con su amor y ternura me llena cada día de felicidad.

A mis Abuelitos Juan Córdova y Emma Espín que me han colaborado mucho para poder realizar mis estudios universitarios.

A mis compañeros de Universidad que siempre han estado en la espera de ayudarme en lo que necesitaba, en especial a los +kθ`s.

A mi primo Patricio que siempre me ha brindado su ayuda, sin importar el momento o como hacerlo, pero siempre ha estado ahí.

A todos ellos gracias por haberme ayudado a conseguir este objetivo.”

Juan Cristóbal Córdova Torres

RESUMEN

El objetivo de este proyecto fue el diseño e implementación de un prototipo que identifique los productos de un supermercado empleando tecnología RFID.

Para resolver este problema se investigó sobre la tecnología RFID para determinar las características del lector y etiquetas que finalmente fueron adosadas a los productos. Luego se procedió a diseñar, implementar y probar el prototipo, pegando las etiquetas de diferentes formas, para identificar problemas en el reconocimiento de los productos. También se recurrió al pesaje de los productos para chequear que el lector identifique no solo los que están en la canastilla sino aquellos que podrían estar fuera con propósitos de fraude.

Las pruebas que se realizaron permitieron obtener los siguientes resultados. Los productos se debieron clasificar en dos tipos: de difícil lectura (líquidos, químicos, metálicos y aquellos que emiten señales) y de fácil lectura (plástico, papel y cartón).

Con los de fácil lectura, las etiquetas fueron identificadas independientemente de cómo estas estaban fijadas al producto. Pero, para los productos de difícil lectura, denominados así debido a que las señales enviadas por el lector hacia las etiquetas son absorbidas o reflejadas parcialmente o totalmente reduciendo su distancia de identificación o evitando la lectura de la información de la etiqueta, se tuvo que recurrir a colocar un elemento que evite el total contacto con los productos de difícil lectura. Cuando las etiquetas se juntan dentro de la canastilla, solo en este caso el lector falla en la lectura de las mismas.

PRESENTACIÓN

La razón principal del proyecto es la de disminuir el tiempo de adquirir los productos de los supermercados, con esto se logra que la aglomeración de personas en las cajas de los supermercados disminuya creando satisfacción en los clientes, que es uno de los puntos principales en toda empresa.

Otra razón del proyecto es evitar los robos que están sujetos los supermercados; con este propósito el sistema consta de una balanza para evitar que cuando una etiqueta sea alejada del producto y encontrarse éste en el carro de compras no ser identificado, evitando pérdidas económicas a la empresa y la corrupción que es uno de los problemas más relevantes en nuestra sociedad.

El objetivo principal del proyecto es construir un prototipo para la detección de productos en un carro de supermercado utilizando la tecnología de Identificación por Radiofrecuencia, permitiendo una rápida detección de los productos en los cajeros.

Para ello se desarrolla un Software para la autenticación de cada producto, para que permita su identificación en los cajeros de los supermercados y para la comunicación del lector con la computadora por el puerto serial.

Se acopla una balanza al sistema RFID para realizar una comparación entre la suma de pesos obtenidos por la lectura de cada producto y la suma de todos los productos puestos en la balanza, tratando de evitar así el robo de productos.

El prototipo implementado demuestra las amplias posibilidades de aplicación de la tecnología RFID y se espera que constituya una guía para aquellas personas interesadas en aprender y desarrollar nuevas aplicaciones.

CAPÍTULO 1. LA TECNOLOGÍA RFID

1.1 DEFINICIÓN

La tecnología RFID, responde a las siglas de *Radio Frequency IDentification* (Identificación por Radiofrecuencia), permite que una etiqueta sea leída por un lector a una distancia determinada, mediante el uso de radiofrecuencia. La tecnología permite al lector identificar las etiquetas que proporcionarán datos de manera rápida en tiempo real.

En los sistemas RFID es importante tener en cuenta algunas características como:

- La frecuencia de operación
- El rango de alcance (hasta qué distancia se puede mantener la comunicación entre el lector y las etiquetas).
- La alimentación eléctrica del lector
- La cantidad de información que las etiquetas pueden almacenar (que van desde un bit hasta los KBytes)
- La velocidad de transferencia de datos entre el lector y las etiquetas.
- El tamaño que poseerán las etiquetas
- La posibilidad de que un lector detecte varias etiquetas a la vez (anticolisión), o que sólo pueda realizar el reconocimiento de una etiqueta a la vez
- El nivel de emisión que presentará el sistema para que esté acorde con los valores permitidos en el país, etc

1.2 RESEÑA HISTÓRICA

La tecnología RFID apareció en 1940 para permitir la identificación de los aviones, durante la Segunda Guerra Mundial. Posteriormente se realizaron algunas mejoras para utilizarla en la industria ferroviaria (para seguir a los coches del

ferrocarril), en el área agrícola, en la administración de la flora y la fauna, para el rastreo del ganado y los animales, como un sistema antirrobo, como un proceso para la automatización y seguimiento de coches. Sin embargo, la tecnología no posee un descubridor definido, puesto que se ha desarrollado por numerosas contribuciones.

En la década de los 60 se crea el EAS (*Electronic Article Surveillance*), que es el primer sistema usado para la detección de robos en almacenes; éste poseía un único bit de información, que determinaba si era detectada una etiqueta y posteriormente hacía sonar una alarma si no se desactivaba la etiqueta que se encontraba en el producto.

En los 70 se producen grandes avances, los que permiten aplicaciones como el rastreo de automóviles, seguimiento del ganado y automatización industrial. Se observó la creación de algunas empresas enfocadas en el estudio de los sistemas RFID.

Durante los 80 se procede a la implementación de algunos de los estudios anteriores. En los Estados Unidos se ven aplicaciones para accesos masivos y aplicaciones referentes al ganado y a los automóviles. Por otro lado España, Francia, Portugal e Italia se concentran en aplicaciones referentes a los animales y también en aplicaciones industriales.

En los 90 en Estados Unidos y en Europa hace uso de esta tecnología en los peajes, para controlar el paso de los automóviles por una autopista; pero en Europa se usan sistemas de microondas e inductivos para el control de acceso de los vehículos.

Posteriormente la empresa *Texas Instruments* crea un sistema para el encendido de los automóviles y la *Philips* crea además de un sistema de encendido, un control de combustible. Consecuentemente se extendió esta tecnología por algunos continentes.

Debido al éxito obtenido por esta tecnología, la Comisión Federal de las Comunicaciones FCC (*Federal Communications Commission*)[1] le concede la banda de los 5,9 GHz a los sistemas inteligentes de transporte y nuevas aplicaciones creadas.

En la tabla 1.1 se muestra la evolución de la Tecnología RFID.

Década	**Avances Tecnológicos**
1940-1950	Se rediseña el radar para uso militar tomando gran relevancia en la II Guerra Mundial. RFID aparece en 1948
1950-1960	Primeros experimentos con RFID en laboratorios.
1960-1970	Desarrollo de la tecnología RFID, primeros ensayos en algunos campos de la tecnología
1970-1980	Explosión de la tecnología. Se realizan más *tests*. Primeras aplicaciones
1980-1990	Aparecen más aplicaciones para la tecnología
1990-2000	RFID toma relevancia en el mundo cotidiano. Aparecen los estándares

Tabla 1.1: Evolución de la Tecnología RFID [1]

Uno de los avances en los que se está trabajando es en la miniaturización de las etiquetas, que permitirá que se les pueda utilizar en objetos más pequeños, pasando casi desapercibidos por ejemplo en las monedas, en los billetes, en prendas de vestir, etc.

Se está aplicando la tecnología RFID en algunos supermercados alrededor del mundo, como por ejemplo los supermercados Metro, *Wal Mart*, los fabricantes de *Gillette* o *Procter & Gamble*; como también en algunos desarrolladores de *hardware* y *software* como *Microsoft*, Intel o SAP.

1 ***FCC*** es una agencia estatal independiente de Estados Unidos, bajo responsabilidad directa del Congreso. Es la encargada de la regulación de telecomunicaciones interestatales e internacionales por radio, televisión, redes inalámbricas, satélite y cable. [17]

Otro de los aspectos a mencionar de la tecnología RFID es que por no ser una tecnología óptica posee una capacidad de rastrear, detectar, gestionar la información y tener flexibilidad.

1.3 COMPONENTES DEL SISTEMA RFID

La tecnología RFID consta de un lector y de etiquetas o *tags*.

El lector enviará ondas de radiofrecuencia hacia la etiqueta que lleva asociado unos datos identificativos que hacen único al objeto, estas ondas serán detectadas por la antena que poseen las etiquetas. Dichas etiquetas enviarán la información hacia el lector, el cual a su vez adquiere y transfiere a través de un medio de conexión a una aplicación *software,* que utiliza los datos para identificar el objeto y realizar las acciones pertinentes según sean los requerimientos. Como por ejemplo, la localización de un objeto en una base de datos.

Las señales de radiofrecuencia del lector y de las etiquetas deben trabajar en el mismo rango de frecuencia, para que se produzca la comunicación entre éstos.

1.3.1 LECTOR RFID

El lector principalmente lee e incluso en algunos casos escribe en la etiqueta.

Otras de sus funciones son:

- Energizar la etiqueta, si la etiqueta es pasiva.
- Enviar comandos para la interrogación de la etiqueta en el campo electromagnético.
- Escuchar la respuesta de la etiqueta.
- Comunicar el resultado de la lectura al *software* aplicativo.

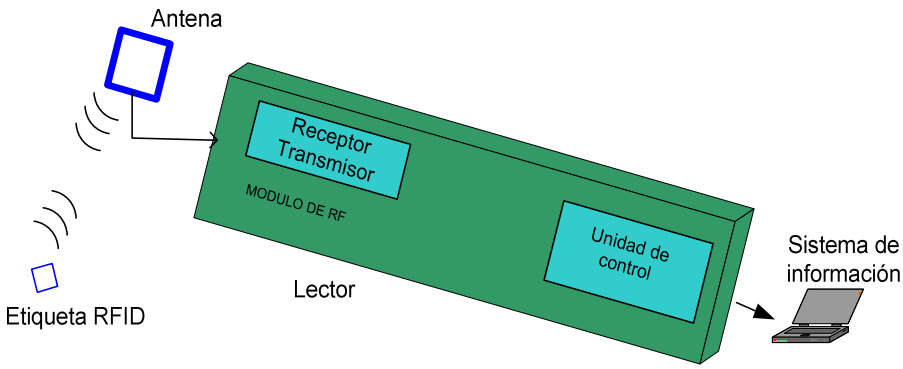


Figura 1.1: Interacción del lector con las etiquetas [2]

En la figura 1.1 se observa que el lector se conecta al computador, esta conexión podría ser por medio de interfaces RS232, RS485, Ethernet, USB o por puntos de accesos inalámbricos. Así se puede conocer la información de cada etiqueta recibida por el lector.

1.3.1.1 Partes del lector

El lector está compuesto por:

- Transmisor
- Receptor
- Microprocesador
- Memoria
- Interfaz de comunicación
- Energía

1.3.1.1.1 Transmisor

El transmisor se utiliza para enviar una serie de ondas de radiofrecuencia a las etiquetas, las mismas que activarán el microchip ubicado en la etiqueta.

1.3.1.1.2 Receptor

El receptor como su nombre lo indica recibirá las señales enviadas por las antenas de la etiquetas. Además el receptor demodulará las señales de radiofrecuencia enviadas por las etiquetas.

1.3.1.1.3 Microprocesador

El microprocesador revisará la integridad de la información recibida. Negociará en caso de que permita está función el proceso de multilectura. Transmitirá la información de las etiquetas al computador.

1.3.1.1.4 Memoria

En la memoria se guardarán los datos necesarios para el correcto funcionamiento del lector.

1.3.1.1.5 Interfaz de comunicación

El interfaz de comunicación es la herramienta que proporcionará las instrucciones para que exista la comunicación entre el lector y las etiquetas, para enviar la información obtenida de la etiqueta y recibir comandos que se traducen en acciones.

1.3.1.1.6 Energía

La energía es el elemento que proporcionará la corriente eléctrica a los componentes del lector.

1.3.1.2 Características

El lector puede tener diferentes características como:

1.3.1.2.1 Número de puertos RF

Son utilizados para conectar las antenas, que típicamente son cuatro. Entre más antenas conectadas se tenga, se podrá disponer de un mayor rango de cobertura y leer múltiples etiquetas simultáneamente dentro del rango.

1.3.1.2.2 Estándares de comunicación

Permite la comunicación entre las etiquetas utilizadas con cualquier lector. Dependiendo del protocolo y la configuración se podrán leer hasta 30 etiquetas por segundo.

1.3.1.2.3 Frecuencia de trabajo

Las bandas en las que trabaja son UHF, HF o LF.

1.3.1.2.4 Potencia máxima

Esta potencia es la referente a la de cada puerto RF.

1.3.1.3 Clases

Se disponen de dos tipos de lectores:

1.3.1.3.1 Sistemas con bobina simple

La bobina servirá tanto para la transmisión de datos y de energía. Estos sistemas no tienen gran alcance pero son simples y baratos.

1.3.1.3.2 Sistemas interrogadores con dos bobinas

En este caso la una bobina servirá para la transmisión de datos y la otra bobina para la transmisión de energía. Estos sistemas a pesar de que son caros brindan mayores servicios.

1.3.2 ETIQUETA RFID

La etiqueta (*tag* o *transponder*) denominado también como etiqueta electrónica, es un elemento de silicio formado por un microchip incorporado o circuito integrado; en éste se almacena un código único (CU), una antena de hilo de cobre y componentes electrónicos como *buffer*, etc. que procesarán la señal de la antena y los datos.

En la figura 1.2 se muestra la etiqueta RFID señalando las partes de la que está compuesta.

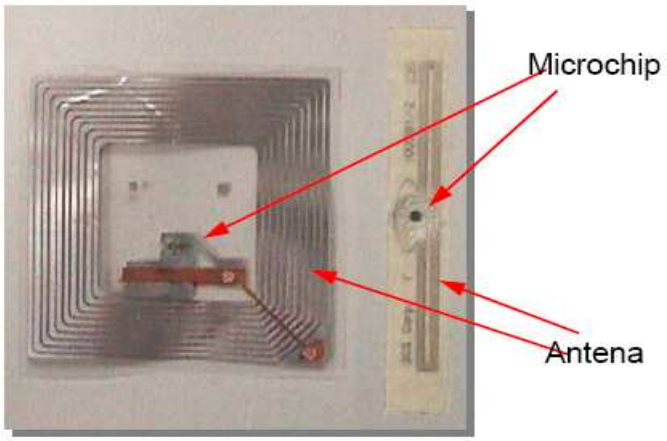


Figura 1.2: Etiqueta RFID [3]

Cada una de las etiquetas poseerá un número único que lo diferenciará de las demás etiquetas; existen varios tipos de esquemas para asignar los números y uno de ellos es el EPC[2] que es diseñado por *Auto-ID Center.*

2 (*Electronic Product Code*) es un número diseñado para identificar de manera individual un producto; esto es, todo producto tendrá su propia identidad exclusiva. El EPC es la única información almacenada en el microchip de la etiqueta RFID. [17]

La dimensión de la etiqueta depende del tamaño de la antena que esté integrada en ésta y además de parámetros como la sustancia que sea utilizada para la elaboración, la frecuencia y el rango de lectura que se desea para la aplicación.

1.3.2.1 Partes de la etiqueta

La etiqueta pasiva está compuesta por una antena y un microchip, y la etiqueta activa está compuesta por una antena, un microchip y una fuente de alimentación.

1.3.2.1.1 Antena

Dependerá de la frecuencia a la que se trabajará; permite recoger la energía procedente del lector y utilizarla para transmitir los datos almacenados en el microchip. Puede tratarse de un elemento inductivo (por ejemplo, una bobina) o bien un dipolo.

Según como esté formada la antena habrá diferentes diagramas de radiación, presentando una mayor potencia de lectura dependiendo cómo se encuentren combinados con la antena del lector.

1.3.2.1.2 Microchip

El chip o circuito integrado tiene un elevado impacto en el comportamiento de la etiqueta. El chip o circuito integrado es el responsable de transformar la energía RF en alimentación eléctrica, de almacenar y recuperar la información, y de modular la señal “de vuelta”.

1.3.2.1.3 Fuente de alimentación

Sirve para proveer de energía a sus sistemas eléctricos de las etiquetas activas, la fuente de alimentación puede ser una batería.

1.3.2.2 Características

El comportamiento de la etiqueta dependerá de varias características, como por ejemplo los requerimientos mínimos que necesita y otros que sólo se encuentran según el modelo o la etiqueta.

1.3.2.2.1 Lectura de la etiqueta

Toda etiqueta debe poder comunicar la información al lector, mediante radiofrecuencia.

1.3.2.2.2 Kill /Disable

Algunas etiquetas permiten al lector enviar un comando (orden) para que deje de funcionar permanentemente. Esto provoca que no responda nunca más.

1.3.2.2.3 Write Once

A muchas etiquetas se les introducen el código de identificación en la propia fabricación. Pero los que contienen la característica *write-once* (una sola escritura) permiten al usuario configurar o escribir su valor una sola vez; después de escribirlo, es imposible cambiarlo.

1.3.2.2.4 Write many

Algunas etiquetas tienen la capacidad de ser escritas y reescritas por el lector tantas veces como se desee (hay un límite de ciclos muy elevado, como por ejemplo 100.000 escrituras) el campo de datos del identificador.

1.3.2.2.5 Anticolisión

Cuando hay muchas etiquetas próximas a un lector, éste puede tener la dificultad de comunicarse con ellos a la vez. Esta característica se realiza mediante protocolos que permiten controlar las comunicaciones entre etiqueta y lector.

1.3.2.2.6 Seguridad y encriptación

Algunas etiquetas permiten encriptar la información en la comunicación; además hay la posibilidad, en varios tipos de estas etiquetas, de responder solo a lectores que les proporcionan un *password* secreto.

1.3.2.2.7 Alimentación

Se la clasifica dependiendo de la energía que utilizan para la comunicación (etiqueta pasiva y activa). La alimentación está en el orden de los mW.
En la tabla 1.2 se observan diferencias entre etiquetas pasivas y etiquetas activas.

	Etiqueta Pasiva	**Etiqueta activa**
Alcance	0.1 - 10 m	10 - 100 m
Alimentación	Campo RF	Batería o mixta
Sensores (temperatura etc.)	Típicamente no	Típicamente si
Tiempo de vida	Ilimitado	Limitado por la batería
Coste	0.04€ ~ 0.8€	0.8€ ~ 17€
Dimensiones	Pequeño	Grande
Tipo de comunicación	Señalización pasiva	Señalización activa o pasiva
Ejemplos típicos	Etiqueta EPC, rastreo de animales, tarjetas inteligentes	Etiquetas para contenedores de transportes

Tabla 1.2: Diferencias entre etiquetas pasivas y etiquetas activas [4]

1.3.2.2.8 Estándares soportados

Las etiquetas pueden cumplir con uno o más estándares, permitiendo comunicarse con los lectores que los cumplen.

1.4 DESCRIPCIÓN DEL SISTEMA RFID

Los sistemas RFID se pueden clasificar siguiendo varios criterios, como pueden ser la frecuencia a la que trabajan los sistemas (LF, HF, UHF o microondas), la alimentación de las etiquetas (activas o pasivas) o según el principio de funcionamiento en el que se basan, que puede ser acoplamiento inductivo, magnético y *backscatter* o *microwave*.

1.4.1 SEGÚN LA FRECUENCIA

De acuerdo a la banda de frecuencia a la que pueden operar, los sistemas RFID se clasifican en:

1.4.1.1 Baja frecuencia (100 KHz-500 KHz)

- Baja velocidad de lectura
- Tamaño grande de la etiqueta
- Identificación unitaria (no existe la multilectura)
- Corta distancia de lectura

1.4.1.2 Media frecuencia (10 MHz –15 MHz)

- Velocidad media de lectura
- Tamaño medio de la etiqueta
- Estándar internacional a 13.56 MHz
- Multilectura

1.4.1.3 Alta frecuencia (850 MHz-950 MHz / 2.4 GHz-5 GHz)

- Conflicto de regulaciones por zonas
- Más velocidad de lectura
- Tamaño menor de la etiqueta / más distancia
- Más sensible a interferencias

1.4.2 SEGÚN EL ACOPLAMIENTO

En la figura 1.3 se muestran las distancias de los acoplamientos.

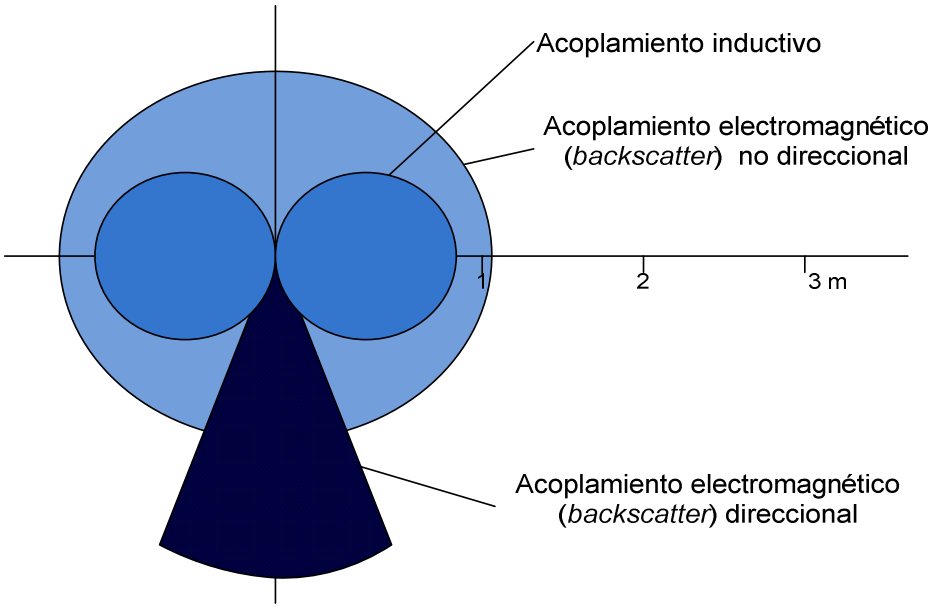


Figura 1.3: Distancias aproximadas según el tipo de acoplamiento [1]

El mecanismo de acoplamiento de la etiqueta determina cómo los circuitos de la etiqueta y el lector se influencian y reciben la información o energía. El tipo de acoplamiento que la etiqueta utiliza afecta directamente al rango de lectura entre los dos dispositivos (etiqueta y lector).

Se pueden agrupar los diferentes rangos de lectura en diferentes sistemas: rango de lectura cerrado en distancias menores a 1 cm, remotas entre 1 cm. y 1 m. o de largo alcance (rango de distancia) para más de 1 m.

1.4.2.1 Acoplamiento Inductivo

La frecuencia que emite el lector es la misma que utilizan las etiquetas para transmitir sus datos. Mientras mayores son las distancias, la potencia que se utiliza debe ser también elevada.

En la figura 1.4 se observa que el lector posee una bobina que será la antena que generará un fuerte campo electromagnético, que ingresará en la antena de las etiquetas que son también bobinas. La bobina tanto del lector como de las etiquetas es de gran tamaño debido a que la longitud de onda es elevada.

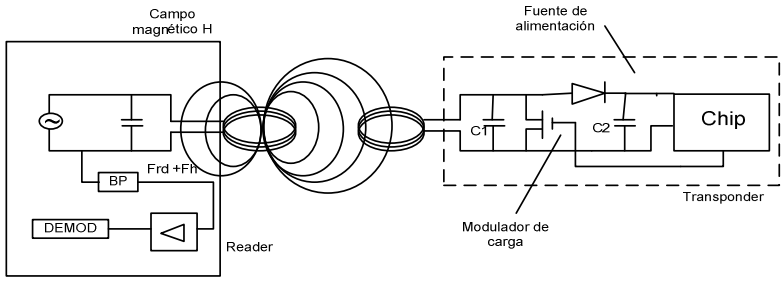


Figura 1.4: Esquema de acoplamiento inductivo [1]

1.4.2.2 Acoplamiento magnético

El acoplamiento electromagnético es similar al acoplamiento inductivo cuando nos referimos que la etiqueta y el lector forman un par de transformadores mediante bobinas.

En la figura 1.5 se aprecia la diferencia entre el acoplamiento inductivo y magnético en la antena del lector, que consiste en una bobina enrollada en una pieza de ferrita con los dos extremos al aire.

La frecuencia con la que transmite el lector es la misma frecuencia con la que transmite la etiqueta.

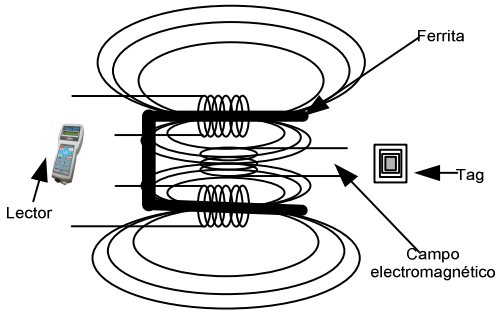


Figura 1.5: Esquema de acoplamiento magnético [5]

1.4.2.3 Acoplamiento *Backscatter* o de microonda

El término *backscatter* se utiliza actualmente para describir que las etiquetas reflectan la señal con la misma frecuencia emitida por el lector, pero cambiando la información contenida en ella. El acoplamiento consiste en reflejar la señal para enviarla al origen.

En la figura 1.6 se observa el camino de las ondas de radiofrecuencia transmitidas por el lector y que son devueltas por la etiqueta mediante dispersión.

Como el lector y la etiqueta usan la misma frecuencia para comunicarse, utilizan turnos para hablar. Este tipo de comunicación es conocida como *Half-Duplex*. El lector continúa proporcionando energía a la etiqueta mientras espera recibir la respuesta de la etiqueta.

La ventaja con estos acoplamientos es que utilizando antenas de pequeño tamaño se consigue una gran eficiencia.

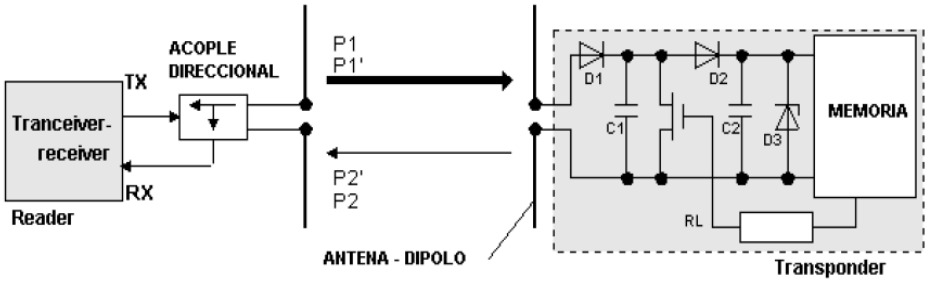


Figura 1.6: Esquema de acoplamiento de microonda [1]

1.4.3 SEGÚN LAS CARACTERÍSTICAS QUE OFRECE LA ETIQUETA

Son sistemas RFID que dependen de las características de las etiquetas tales como: capacidad de almacenamiento en su memoria, rango de información y la capacidad de procesamiento; presentan un amplio espectro de variantes que se dividen en sistemas *Low-end, Mid-range* y *High-end*.

1.4.3.1 Sistemas *Low-end*

Las etiquetas de solo lectura pertenecen a los sistemas EAS (*Electronic Article Surveillance*). Se puede tener el problema de que varias etiquetas quieran comunicarse con el lector y exista una colisión, produciendo que el lector no reconozca a ninguna etiqueta. Estos sistemas son para etiquetas que no almacenan mucha información y son capaces de trabajar en todo el rango de frecuencias que opera RFID.

1.4.3.2 Sistemas *Mid-range*

En estos sistemas se utilizan etiquetas de lectura/escritura; no presentan problemas con las colisiones, el lector puede diferenciar cuál es la etiqueta que envía esa información. Presentan encriptación y autenticación de la información entre las etiquetas y el lector.

Para los sistemas *Mid-range* se necesita memoria EEPROM (para etiquetas pasivas) o SRAM (para etiquetas activas).

1.4.3.3 Sistemas *High-end*

Tienen un sistema de funcionamiento de tarjeta inteligente y microprocesadores, además trabajan a una frecuencia de 13,56 MHz. Cuando se usan los microprocesadores se facilita el emplear encriptación y autenticación más complejos.

1.4.4 SEGÚN LA COMUNICACIÓN ENTRE EL LECTOR Y LAS ETIQUETAS

El lector genera una comunicación con una etiqueta enviando radiofrecuencia, la etiqueta detecta este campo de RF y responde en *broadcast*. El lector debe poseer un sistema para detectar cuál es la señal que él envió y cuál es la recibida por la etiqueta; esto se distingue por que la señal que envía la etiqueta al lector es más débil que la que genera el lector.

Se tienen 3 procesos en la comunicación entre el lector y las etiquetas.

1.4.4.1 Sistemas *half duplex*

En un sistema *half duplex* existe una transferencia de datos que se va alternando entre el lector y las etiquetas.

1.4.4.2 Sistemas *full duplex*

En un sistema *full duplex* existe una comunicación simultánea entre el lector y las etiquetas.

1.4.4.3 Sistemas secuenciales

El campo del lector se enciende y apaga en intervalos regulares. Por esto las etiquetas se alimentan de forma intermitente, es decir por medio de pulsos. La desventaja es que las etiquetas pierden energía cuando se interrumpe la comunicación, pero se puede contrarrestar con alimentación externa.

En la figura 1.7 se muestran los esquemas de los 3 procesos de la comunicación entre el lector y las etiquetas.

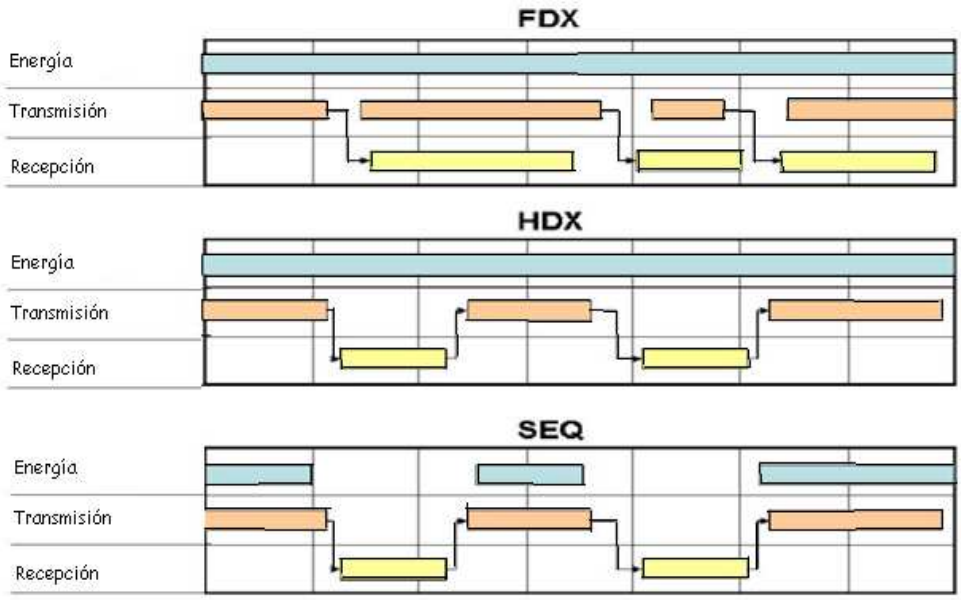


Figura 1.7: Esquema de los 3 procesos de la comunicación entre el lector y las etiquetas [6]

1.4.5 SEGÚN EL TIPO DE MEMORIA DE LA ETIQUETA

Se pueden clasificar los sistemas RFID dependiendo de la memoria que poseen las etiquetas; estas memorias pueden permitir ser escritas más de una vez o solo una vez.

1.4.5.1 EEPROMs (*Electrically Erasable Programmable Read-Only memo*ry)

Las memorias EEPROMs poseen un límite en los ciclos de escritura de 100.000 y 1´000.000. La desventaja de esta memoria se basa en el alto consumo de energía que se da en el momento de la escritura. Es la memoria que más se utiliza en el acoplamiento inductivo y tienen una capacidad de 16 bytes a 8 Kbytes. Estas memorias son las comúnmente utilizadas. Las etiquetas que posee una memoria EEPROM son de menor precio. Cuando la etiqueta posee esta memoria se tiene un código fijo dado por el fabricante.

1.4.5.2 FRAMs (*Ferromagnetic Random Access Memory*)

El tiempo de escritura de las memorias FRAMs es de 1.000 veces menor y poseen un consumo de energía 100 veces menor a la que las memorias EEPROMs.

1.4.5.3 SRAMs (*Static Random Access Memory*)

La memoria SRAMs se utiliza más en los sistemas de microondas, con una capacidad de 256 bytes y 64 Kbytes. Poseen un rápido acceso durante los ciclos de escritura. Este sistema no puede tener interrupción de energía, por esta razón poseen baterías.

1.4.6 SEGÚN LA ALIMENTACIÓN

Se la clasifica dependiendo de la energía que utilizan para la comunicación, la cual está en el orden de los mW.
Las etiquetas según la alimentación se las clasifica en pasivas y activas.

1.4.6.1 Etiquetas Pasivas

Son aquellas que no requieren batería, estas etiquetas poseen un microchip y una antena para recibir el campo electromagnético para su alimentación y para enviar el código al lector.

1.4.6.2 Etiquetas Activas

La alimentación necesaria para la etiqueta se obtendrá de una batería interna. Una de las ventajas es que la comunicación puede ser inicializada por el lector o por la etiqueta.
Debido a que las etiquetas activas poseen una fuente de alimentación que le proporciona mayor potencia y también debido al tamaño de la antena, el alcance es mayor que el de las etiquetas pasivas.

1.5 FUNCIONAMIENTO DE RFID

RFID es un sistema de identificación que puede ser leído a una determinada distancia sin ser necesario tener línea de vista.

Como se puede observar en la figura 1.8, el lector envía una serie de ondas de radiofrecuencia a la etiqueta, que son captadas por la microantena de éste. Dichas ondas activan el microchip, el cual, a través de la microantena y mediante ondas de radiofrecuencia, transmiten al lector la información almacenada en su memoria (CU del producto). Finalmente, el lector recibe la información que tiene la etiqueta y lo envía a través de un medio de comunicación a una base de datos, en la que previamente se han registrado las características del producto según el código único o puede procesarlo según convenga a cada aplicación.

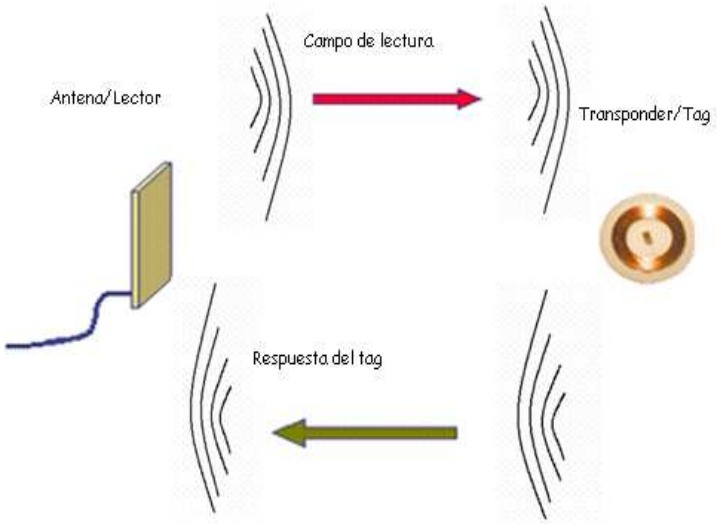


Figura 1.8: Interacción entre la etiquetas y el lector [6]

Un problema que aparece con la tecnología RFID es que la señal de un lector puede interferir con la de otro lector; es lo que se llama “colisión de lecturas”. Una forma de evitar el problema es utilizar una técnica llamada acceso múltiple por división temporal (TDMA por sus siglas en inglés).

El sistema se le denomina sistema anticolisión; éste funciona de la siguiente manera: si varias etiquetas están en el rango de alcance del lector, éste las detectará y se aplicará un algoritmo complejo que hará que las etiquetas transmitan en tiempos distintos evitando así las colisiones. Los algoritmos utilizados dependen de las etiquetas y de los lectores utilizados.

En lo referente a la velocidad que poseen las etiquetas con respecto al lector, se define el tiempo en que debe permanecer la etiqueta en la zona de operación para poder transmitir la información.

1.6 VENTAJAS COMPARATIVAS DEL RFID SOBRE EL CÓDIGO DE BARRAS

La tecnología RFID supera muchas de las limitaciones del código de barras, el sistema de identificación de objetos más utilizado hasta ahora.

Algunas de las ventajas de las etiquetas RFID sobre el código de barras son las siguientes:

- Las etiquetas RFID pueden ser reprogramables por lo que se puede cambiar cualquier dato en cualquier momento.
- En la tecnología RFID el lector no necesita tener línea de vista con la etiqueta para poder identificarlas; los códigos de barras si no poseen línea de vista no pueden ser leídos.
- A diferencia del código de barras, las etiquetas pueden ser identificadas hasta una distancia de 10 metros.
- El sistema RFID identifica a cada producto con un código único, por lo que no todos los productos son iguales; en cambio para el código de barras si lo es, ya que es el mismo código en todas las etiquetas de un producto.
- El lector RFID puede identificar varios productos a la vez, el código de barras no puede realizar esto.

- El código de barras no puede almacenar tanta información, solo puede almacenar un código y un precio; en cambio las etiquetas RFID pueden almacenar mucha más información.
- El código de barras no permite escribir más de una vez, el sistema RFID permite escribir tantas veces como sea necesario dependiendo de la etiqueta.
- El código de barras puede ser reproducido nada más fotocopiando el código, por otro lado una etiqueta RFID necesita un equipo RFID y un cierto grado de conocimientos para su reproducción.
- El código de barras es más vulnerable a daños, que una etiqueta electrónica.

En conclusión los beneficios de la tecnología RFID sobre la de código de barras se ven reflejados en la productividad, la simplificación de los procesos administrativos y en la realización de inventarios en donde se aplique.

En la figura 1.9 se puede observar que al utilizar la tecnología RFID le permitirá a las empresas tener beneficios al momento de entregar los productos, ahorrar grandes cantidades de dinero, saber si ya se ha terminado un producto en el almacén o supermercado y abastecer al mismo de los productos necesarios, lo que brindará a la empresa competitividad frente a otras empresas.

En la tabla 1.3 se resumen las ventajas de la tecnología RFID sobre la de código de barras.

Frente a estas ventajas aparecen los siguientes inconvenientes:

- Poseen un alto coste, a pesar de que a futuro se prevé un considerable descenso en el mismo.
- Existe una carencia de regulación y de estándares comerciales que faciliten su difusión.

Características	Tecnología RFID	Código de Barras
Capacidad	Mayor capacidad de almacenamiento de datos	Espacio limitado
Identificación	Permite la identificación unívoca de un producto	Estandarizada
Actualización	Puede ser reutilizado ya que permite ser regrabado	Solo lectura
Flexibilidad	No se requiere una línea directa entre el lector y la etiqueta	Requiere línea de visión para su lectura
Lectura	Se pueden realizar miles de lecturas en forma simultánea	Una lectura por vez
Tipo de Lectura	Puede leerse a través de varios tipos de materiales	Lee sólo en superficie
Precisión	Es totalmente automático, eliminando errores	Requiere la intervención humana
Durabilidad	Soporta ambientes agresivos (humedad, ácidos, etc.)	Puede dañarse fácilmente

Tabla 1.3: Resumen de ventajas comparativas del RFID sobre el código de barras.

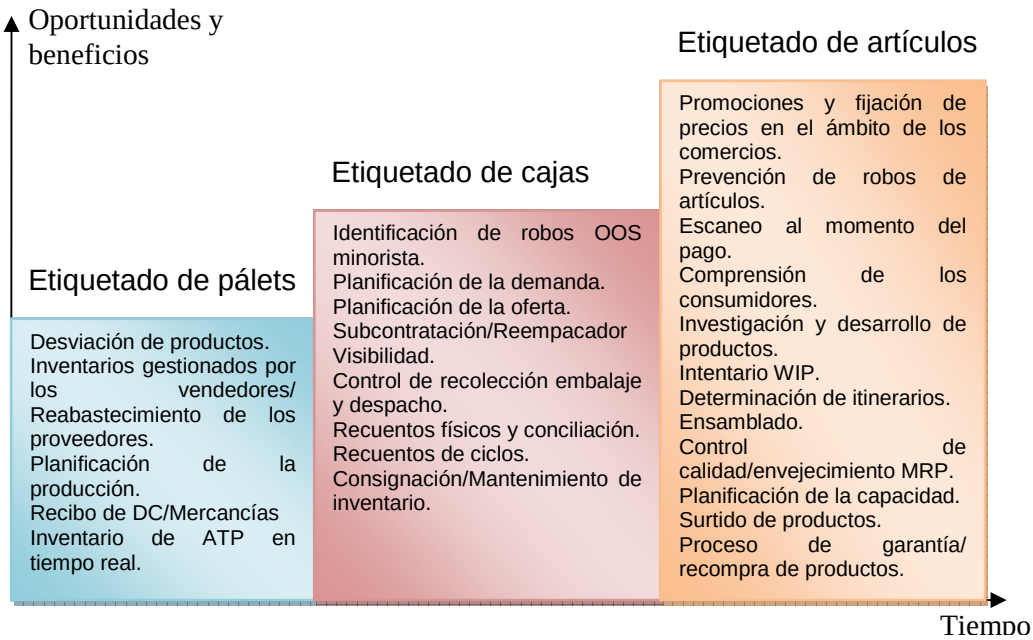


Figura 1.9: Ventajas de la utilización de RFID [7]

- Presenta vulnerabilidades al metal y otros materiales conductivos, así como a líquidos y a interferencias electromagnéticas de baja frecuencia.
- Puede presentar problemas de seguridad si no se toman medidas que eviten lecturas y modificaciones fraudulentas de la información.
- Una de las desventajas y una polémica entorno a esta tecnología, es que puede quitar la privacidad, puesto que piensan aplicar esta tecnología en los pasaportes o en las cédulas de identidad. Pero una de las posibles soluciones que ya se está examinando, es poner estos documentos en una funda protectora que contiene en su interior una jaula de faraday, que son unos finos hilos metálicos entrecruzados; esta funda cuando se cierra bloquea la emisión de la antena del RFID.
- Uno de los problemas en RFID es la interferencia de campos electromagnéticos, puesto que es muy susceptible a ésta, pero este problema se acentúa más en los sistemas con acoplamiento inductivo. Por esto es muy importante asegurarse de que los sistemas RFID no interfieran en la comunicación con radio, televisión, los servicios de radio móvil, con el uso de celulares y las comunicaciones marítimas o aéreas. Por esta razón

se utilizan las frecuencias que están en la banda ISM (*Industrial-Scientific-Medical*)[3].

1.7 APLICACIONES DE LA TECNOLOGÍA RFID

Son muchos los sectores industriales que pueden beneficiarse de las ventajas de la tecnología de auto-identificación por radiofrecuencia.

Algunas de sus aplicaciones son las siguientes:

- El chip para la piel son cápsulas muy pequeñas de vidrio. A los niños se les implantaría un chip y en caso de secuestro o desastre podría ayudar a localizarlos, se tendrían lectores en varias partes y cuando sea identificado el del niño buscado, se activará una alarma.
- En el supermercado, facilitará la obtención de inventarios y se podrá hacer el cobro de los productos de una manera más rápida.
- En las gasolineras se dará a los clientes un llavero, en el cual se podrá descargar el costo del combustible.
- Se tendrá información de la ubicación de las personas dentro de un edificio, teniendo cada persona una tarjeta RFID y así conocer si existe alguien no identificado en el edificio.
- En una cárcel se conocerá la ubicación de los presos y de las visitas, para así evitar la fuga de presos.
- Implantando pequeños chips en los animales se podría saber qué vacunas poseen, la edad del animal. Si una mascota se pierde podría ser identificada por su dueño.
- En los hospitales ya se ha utilizado esta tecnología, con esto se agiliza las urgencias hospitalarias, se da un brazalete en las muñecas para conocer la ubicación de un paciente y conocer datos actuales sobre cada paciente.
- Control de calidad, fechas de caducidad, producción y distribución.

3 Son bandas reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnética en áreas industriales, científica y médica [17]

- Localización y seguimiento de objetos.
- Control de accesos.
- Detección de falsificaciones.
- Automatización de los procesos de fabricación.
- Reducción de tiempo y coste de fabricación.
- Identificación y control de equipajes en los aeropuertos.
- Inventario automático.
- Etc.

CAPÍTULO 2 PRUEBAS DE IDENTIFICACIÓN

2.1 INTRODUCCIÓN

El objetivo de las pruebas que se realizarán en este capítulo es verificar que la etiqueta RFID tenga una buena interacción con el lector cuando están colocadas en los productos.

Los pasos para la realización de las pruebas serán:

- Escoger el voltaje de alimentación del lector para determinar el rango de identificación de las etiquetas.
- Verificar si existe comunicación entre el lector y las etiquetas con el programa de comunicaciones *HyperTerminal*; y tomar la medida de la distancia a la que son detectadas las etiquetas sin estar adheridas a ningún producto.
- Para determinar las distancias de identificación de las etiquetas adheridas a los productos por parte del lector también se lo realizará con el programa de comunicación *HyperTerminal*; cuando se realice la identificación aparecerá el código de la etiqueta detectada en la pantalla del *HyperTerminal*.
- Las medidas de distancia serán tomadas cuando el lector identifique a las etiquetas; estas medidas se las realizará colocando las etiquetas detrás y delante del producto.

Al terminar las pruebas mencionadas anteriormente se realizará una clasificación de los productos, según el comportamiento que presente la señal de radiofrecuencia que es enviada por el lector para identificar las etiquetas.

Esta clasificación se debe a que las señales enviadas por el lector pueden ser absorbidas y reflejadas por los materiales que están cerca de las etiquetas a ser identificadas.

Para finalizar las pruebas se analizará la colocación de las etiquetas en los productos para tratar de evitar que exista interferencia en la comunicación con el lector.

2.2 COMPROBACIÓN DEL FUNCIONAMIENTO DEL LECTOR Y DE LAS ETIQUETAS

La asignación de códigos únicos en las etiquetas no será necesaria puesto que ya vienen con un código diferente asignado de fábrica para cada etiqueta.

Para energizar al lector RFID se utiliza una fuente *switching*; la utilización de la fuente *switching* en lugar de una fuente normal es porque posee gran flexibilidad para tolerar grandes rangos de voltaje de entrada, funcionando indistintamente con voltajes que varían. Esto es imposible con la fuente normal, donde una variación de un 10% en la entrada puede provocar un sobre-calentamiento del regulador lineal.

La fuente *switching* utilizada para el proyecto permite escoger voltajes entre 3 / 4.5 / 6 / 7.5 / 9 / 12 VDC; estos voltajes medidos con el multímetro en el vacio (sin carga), dan las medidas de 3.66 / 4.92 / 6.11 / 7.55 / 9.23 y 12.40 VDC respectivamente. Para la realización del proyecto se utiliza 12 voltios, y debido a este voltaje escogido se obtendrá una distancia determinada, con las variantes correspondientes por la absorción y reflejo que presentan los productos.

Para verificar el funcionamiento del lector y de las etiquetas se realiza con el programa de comunicaciones *HyperTerminal*; al leer las etiquetas con el lector, éste envía su código, el mismo que se lo puede observar en la pantalla del *HyperTerminal*, concluyendo que el lector está realizando la lectura y que las etiquetas están operando correctamente.

Se realizan pruebas con 10 etiquetas para observar las distancias a las que eran identificadas por el lector. Estas pruebas se hicieron con las etiquetas sin estar

adheridas a los productos. De estos datos se obtuvo el promedio de la distancia de identificación entre las etiquetas y el lector como se indica en la tabla 2.1.

Código de la etiqueta	**Distancia para la lectura**
F43DF	18.8 cm
F43DA	19.0 cm
F43D3	18.7 cm
F43E2	18.9 cm
F43D5	18.5 cm
F444E	18.8 cm
F444A	18.5 cm
F43E0	18.7 cm
F43DD	18.6 cm
F444D	19.2 cm
Promedio	**18.8 cm**

Tabla 2.1: Distancia de identificación entre las etiquetas y el lector

La distancia promedio de identificación entre el lector y las etiquetas para el prototipo es de 18.8 cm, alimentando el lector con un voltaje de 12 V.

En una aplicación real para un supermercado, se utilizará un lector con capacidad de identificar las etiquetas a una distancia mayor, aproximadamente a tres metros, para que cubra el área entre un coche de compras existentes en los supermercados y el cajero.

Para realizar la lectura de identificación de las etiquetas adheridas a los productos se utiliza el programa de comunicaciones *HyperTerminal*; este programa realizará la comunicación serial entre el lector y la computadora, y muestra en la pantalla del *HyperTerminal* el código de la etiqueta al ser detectada.

Verificar que cada etiqueta esté en perfectas condiciones es necesario para asegurar que éstas puedan ser detectadas por el lector cuando se encuentran adheridas a los diferentes productos, siempre y cuando las señales de radiofrecuencia enviadas por el lector no se anulen por completo debido al reflejo y/o absorción de los materiales o composición de los productos, ya que estos componentes de los productos influyen en las señales de radiofrecuencia.

El comportamiento de la señal que envía el lector para identificar las etiquetas varían dependiendo del material y composición del producto; se tendrán productos que contienen líquidos que absorben parte o la totalidad de la señal como las botellas de agua, productos de envase metálico que reflejan la señal o parte de la misma como las latas de cola, productos que emiten señales que causan interferencia como los celulares y productos que no contienen líquido ni metal que no ocasionan ningún problema en su detección como las fundas de fideo.

Se dará una explicación mas detallada de la clasificación de estos productos más adelante.

2.3 COLOCACIÓN DE LAS ETIQUETAS EN CADA PRODUCTO

Luego de colocar las etiquetas en los productos se verificará en cada uno de ellos el comportamiento de la señal enviada por el lector para la identificación, debido al comportamiento que sufren con los distintos materiales y composiciones del producto.

En la tabla 2.2 se muestran los resultados de las pruebas. Constará de seis columnas, donde irán los datos del código de la etiqueta, la foto del producto con la etiqueta, el material del envase del producto, la composición del producto, el dato de la distancia tomada cuando el lector identifica la etiqueta al estar adherida delante del producto y el dato de la distancia tomada cuando el lector identifica la etiqueta al estar adherida detrás del producto.

Producto / Código	Figura	Material	Composi-ción	Distancia (cm)	
				Etiqueta delante del producto	Etiqueta detrás del producto
Fideos F444A		Plástico	Harina	17,4	14.3
Avena F43DD		Plástico	Avena	15	8,5
Sopa F43D5		Plástico	Harina	17	16
Cerveza F444D		Vidrio	Líquido	18,5	12
Yogurt F43D3		Plástico	Líquido cremoso	17,5	10.4
Queso F444E		Plástico	Cremoso	17	13,5

Gel F43E0		Plástico	Gel	18	8
Leche F43DA		Cartón y polietileno	Líquido	10	No se lee
Leche de chocolate F43DF		75% cartón, 20% polietileno y 5% aluminio	Líquido	7,8	No se lee
Red Bull F43E2		Aluminio	Líquido	16	11
Cerveza F43E2		Aluminio	Líquido	18	13
Desinfectante F43E0		Plástico	Líquido	17.3	13.4

Aceite F43D3		Plástico	Aceite	16.2	8.3
Arroz F444E		Plástico	Arroz	17.2	9.8
Desodorante F43DD		Plástico	Crema sólida	17	14.8
Celular F444A		Plástico	Metal y plástico	12	9.2
Gatorade F43D5		Plástico	Agua	14.7	8

Tabla 2.2: Distancia de identificación entre las etiquetas adheridas al producto y el lector

Al tener productos que interfieren en la señal para la identificación de las etiquetas, las distancias varían.

Cuando existe mayor absorción y mayor reflejo de la señal de radiofrecuencia causada por los productos a las que van adheridas las etiquetas, la distancia necesaria para que sean identificadas será menor y en otros casos no será identificada.

En el caso de la botella de cerveza existe una apreciable diferencia de distancia de lectura cuando la etiqueta está por detrás; esto se debe a que las señales de radiofrecuencia enviadas por el lector para identificar la etiqueta deben pasar el líquido que hay en la botella, causando una parcial absorción de la señal, teniendo que acercar más el lector al producto para poder detectar la etiqueta.

En los productos como la funda de leche y el cartón de leche, la distancia para detectar la etiqueta cuando está por delante es menor con relación a los otros productos, esto se debe al aluminio que contienen estos productos causando que la señal de radiofrecuencia sea reflejada parcialmente. No son detectadas las etiquetas por el lector cuando están detrás de estos productos, porque la señal de radiofrecuencia enviada por el lector es reflejada y absorbida ya que debe atravesar el aluminio del envase y el contenido líquido.

En otros productos que son de fácil lectura ya que no contienen líquidos y que no están hechos de material metálico, se observa que hay una diferencia apreciable de distancia cuando la etiqueta está por detrás de estos productos como son la funda de avena y arroz; la diferencia de distancia es porque la señal enviada por el lector para detectar la etiqueta debe viajar más distancia debido a que el contenido de estos productos tienen un mayor volumen en relación a los otros.

En la tabla 2.2 se observa que los datos de la distancia de lectura varían en relación de un producto a otro, por lo que a los productos se les dividirá en 2 grupos: Productos de fácil lectura y Productos de difícil lectura.

2.3.1 PRODUCTOS DE FÁCIL LECTURA

Son los productos compuestos de plástico, papel o cartón que no producen ningún tipo de problema como la absorción y reflejo de las señales enviadas por el lector para la detección de los productos, el único inconveniente que se puede presentar en estos productos es saber donde colocar las etiquetas para evitar su deterioro; algunos de estos productos se muestran en la figura 2.1.

Figura 2.1: Productos de fácil lectura

2.3.2 PRODUCTOS DE DIFÍCIL LECTURA

Son los productos que presentan inconvenientes en las señales enviadas por el lector para realizar la identificación. Se los clasifica en cuatro grupos: Productos metálicos, Productos líquidos, Productos químicos y Productos que emiten señales.

2.3.2.1 Productos metálicos

Los productos que están compuestos por estaño o metal reflejan las señales de radiofrecuencia, limitando así la distancia a la que se identificará la etiqueta o, en el peor de los casos, no permitiendo la identificación de la etiqueta. Al tener un producto que sea compuesto totalmente de metal, la situación se complicaría puesto que la etiqueta no trabajará a la frecuencia deseada, por lo que no podrá transmitir ni recibir información. Este problema se solucionará separando la

etiqueta del metal con algún aislante adherido en la parte donde esté colocada la etiqueta.

Por ejemplo se puede apreciar que al estar toda la etiqueta en contacto con la superficie metálica del producto no permite la identificación de la etiqueta; algunos de estos productos se muestran en la figura 2.2.

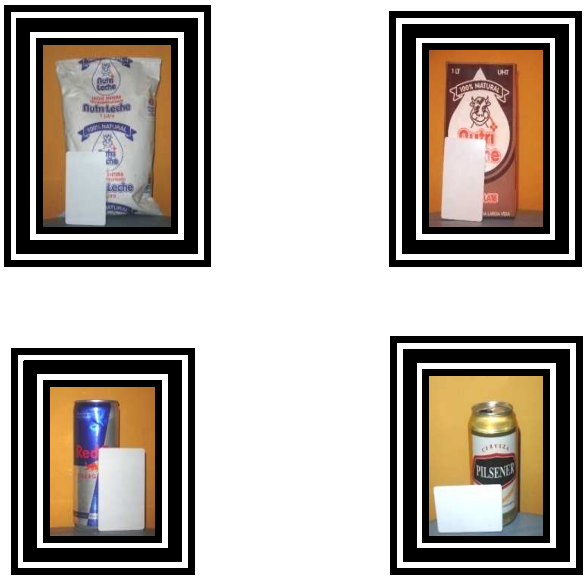


Figura 2.2: Productos metálicos

Al cartón y funda de leche se le ha tratado como producto metálico, porque en su interior tiene un recubrimiento de aluminio y el cartón de su exterior no es lo suficientemente grueso como para actuar como un aislante con la etiqueta.

En el caso de la lata de cerveza se colocó la etiqueta en dos lugares diferentes, para observar que según cómo se coloque la etiqueta afecta en la distancia de la lectura y ver cómo el aluminio de la lata refleja las ondas de radiofrecuencia, más cuando toda la etiqueta está haciendo contacto con la lata.

En la tabla 2.3 se observa cómo se coloca la etiqueta en el producto y la distancia determinada.

Producto / Código	Figura	Material	Composición	Distancia (cm)	
				Etiqueta delante del producto	Etiqueta detrás del producto
Cerveza F43E2		Aluminio	Líquido	18	13
Cerveza F43E2		Aluminio	Líquido	9.6	2.2

Tabla 2.3: Distancia de identificación entre las etiquetas y el lector en un producto metálico con diferente posición de la etiqueta

2.3.2.2 Productos líquidos

Los productos que contengan líquidos pueden absorber las señales de radiofrecuencia limitando la distancia de identificación o no permitiendo la identificación del producto. Estos productos con líquidos absorberán la señal provocando que la etiqueta no tenga la suficiente potencia para responder. La cantidad de líquido que posee el envase también influye, pues las ondas de radiofrecuencia deberán atravesarlo para poder comunicarse; algunos de estos productos se muestran en la figura 2.3.

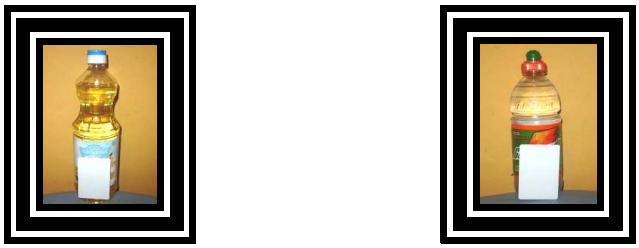


Figura 2.3: Productos líquidos

2.3.2.3 Productos químicos

Los productos químicos corresponden a los de una droguería o productos de limpieza, porque son productos que presentan mayor interferencia que los líquidos, debido a sus compuestos.

Éstos se los clasifica como productos de fácil lectura y de difícil lectura, porque su composición hace que algunos sean transparentes y otros absorbentes a las señales de radio frecuencia. Sus envases serán de metal, plásticos, cartón o papel y su contenido también será variado: polvos, líquidos y líquidos viscosos; en la figura 2.4 se muestra un producto químico con envase de plástico y de contenido líquido viscoso.

Figura 2.4: Producto químico

2.3.2.4 Productos que emiten señales

Se tendrá problemas de interferencia en el momento de la lectura al tener en su proximidad equipos o aparatos que emiten señales o que hagan uso del mismo ancho de banda que el lector y las etiquetas correspondientes.

Producto / Código	Figura	Material	Composición	Distancia (cm)	
				Etiqueta delante del producto	Etiqueta detrás del producto
Celular F444A		Plástico	Metal y plástico	12	9.2
Celular F444A		Plástico	Metal y plástico	9.7	6.7

Tabla 2.4: Distancia de identificación entre las etiquetas y el lector en un producto que emiten señales con diferente posición de la etiqueta

En la tabla 2.4 se muestran los datos tomados de la distancia de lectura para observar cómo los aparatos que emiten señales interfieren en la distancia de la lectura entre el lector y la etiqueta, y cómo la colocación de la etiqueta en el producto afecta a la distancia de la lectura.

2.4 CONCLUSIONES DE LAS PRUEBAS

Las conclusiones de las pruebas son:

- El rango de lectura del lector no será el mismo cuando las etiquetas están solas, que cuando están adheridas a los productos; porque el material y composición de los productos reflejan y absorben las señales de radiofrecuencia enviada por el lector.
- Se tendrán varias distancias de identificación de una etiqueta en un mismo producto cuando ésta se coloque en distintas posiciones. Se recomienda

colocar la etiqueta en el lugar más adecuado para la identificación por parte del lector.

- El lugar para colocar la etiqueta en los productos no será la misma a otros productos, porque no todos los productos tienen el mismo material de envase y de composición.
- Las pruebas realizadas ayudan a clasificar qué productos son los que no se detectarían y cuales necesitarán que se acerque más el lector para identificarlos; y así tomar las respectivas precauciones, como por ejemplo colocar un aislante entre el producto y la etiqueta para disminuir el reflejo y absorción de las señales de radiofrecuencia.
- Es importante tomar en cuenta la presentación del producto con la etiqueta, sin perder de vista la capacidad de identificación de las etiquetas adheridas a los productos.
- Según el lector y las etiquetas que se utilice se tendrá una distancia de identificación, esto dependerá de las propiedades de cada uno de los equipos. En este proyecto por ser un prototipo se utiliza el lector GP20 que tiene un rango de lectura bajo los 20 cm.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO

En este capítulo se mencionarán las características de los equipos utilizados para la implementación del prototipo como son el lector, las etiquetas RFID, la balanza de mesa, la fuente *switching* y los cables de conexión. También se realizará una breve descripción del lenguaje de programación JAVA y se explicará el desarrollo del *software* del prototipo. Finalmente se irá detallando paso a paso la construcción del prototipo para su implementación.

El prototipo que se armará servirá para la detección de productos en un supermercado y para tratar de evitar el robo de productos por parte de los clientes.

El prototipo consistirá en:

- Un lector RFID para la identificación de los productos por medio de las etiquetas.
- Etiquetas RFID, en donde el código único de cada una de ellas servirá para diferenciar un producto de otro con el nombre, peso y precio.
- Una fuente *switching* para la alimentación eléctrica del lector.
- Una canasta para colocar los productos simulando que están en un coche de compras.
- Una balanza de mesa para pesar los productos que están en la canasta simulando que es una balanza de suelo.
- Una computadora para cargar el programa que se realizará en JAVA, que procesará la información que es recibida del lector y de la balanza.
- El *software* que hará la comunicación serial entre el lector y la computadora, realizará la suma total de los precios y de los pesos de los productos que son identificados por el lector, y mostrará el peso obtenido de la balanza para observar si los pesos de la suma total de los pesos y el peso obtenido de la balanza son iguales o no.

- Un cable con conector DB9 hembra a conector DB9 hembra con conexión directa para conectar la balanza, y un cable convertidor de USB a RS232 para enlazar la computadora con el cable con conectores DB9 hembra.

3.1 CARACTERÍSTICAS DEL HARDWARE DEL PROTOTIPO

3.1.1 LECTOR RFID

En la figura 3.1 se muestra el lector RFID ProxID GP20 utilizado en el proyecto, el mismo que es solo de lectura.

Figura 3.1: Lector RFID

Los lectores ProxID GP20 funcionan desde 5 a 13,5 voltios con una distancia de lectura de hasta 20 cm, lo que les hace ideales para una gran variedad de aplicaciones. La característica principal es que el lector básico puede ser configurado para las salidas más usuales de interfaz, incluyendo *Wiegand*, banda magnética, *Clock/Data* y salida RS-232 serial ASCII, haciéndolo muy fácil de aplicar en instalaciones ya existentes.

En la figura 3.2 se muestra la estructura de la trama de datos que envía el lector a la computadora.

Para RS232 serial ASCII se tienen las siguientes propiedades: 9600 Baudios, sin paridad, 8 bits de datos, 1 bit de parada.

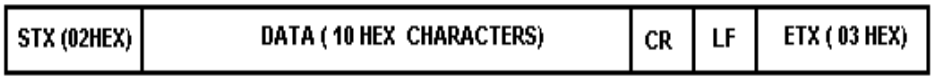


Figura 3.2: Estructura de la trama

En la tabla 3.1 se describen las características del lector RFID utilizado en el proyecto.

Modelo GP20	
Frecuencia	125 Khz
Requerimientos de energía	5 – 13.5 VDC @ 65 mA típico con 12 V
Interfaz de salida	*Wiegand*, *Magstripe*, 9.6 Kbaudios RS232 serial ASCII
Rango máximo de lectura (En condiciones ideales)	20 cm a 13.5 VDC 13 cm a 5 VDC
Dimensiones	7.8 x 4.3 x 1.5 cm
Rango de temperatura	-10 a 60 °C
Asignación de salida para RS232 serial ASCII	Rojo: energía +V Negro: tierra Verde: transmisión de datos Amarillo: sin conexión Blanco: sin conexión Anaranjado: sin conexión

Tabla 3.1: Características del lector RFID

En el ANEXO A.1 se muestran las características de los lectores de proximidad ProxID.

3.1.2 ETIQUETA RFID

En la figura 3.3 se muestran las etiquetas utilizadas en el proyecto que son las Tarjetas de proximidad de 125 Khz de sólo lectura modelo EM H4102.

Figura 3.3: Etiqueta RFID

Es una tarjeta plástica PVC laminada por ambas caras con tamaño ISO estándar de 85,7 x 54 mm y grosor inferior a 0,9 mm.

En la tabla 3.2 se describen las características de las etiquetas RFID utilizada en el proyecto.

Modelo EM H4102	
Frecuencia	125 Khz
Chip	Sólo lectura EM4102
Transferencia de datos	Sin contacto
Velocidad transferencia lectura	20 us.
Velocidad transferencia escritura	Es sólo lectura
Tiempo transacción	Aproximadamente 150 ms.
Capacidad memoria total	64 bytes x 8 bit EEPROM

Tabla 3.2: Características de la etiqueta RFID

En el ANEXO A.3 se muestran las características de las tarjetas de proximidad 125 Khz.

3.1.3 FUENTE DE ALIMENTACIÓN *SWITCHING*

En la figura 3.4 se muestra la fuente de alimentación del lector utilizada en el proyecto que es el adaptador de alimentación *Plug-in* de conmutación, modelo LLAS500.

Figura 3.4: Fuente *switching*

Las fuentes *switching* son cada vez mas utilizadas para diversas aplicaciones, desde las fuentes para computadoras personales hasta las fuentes para centrales telefónicas, pasando por las más diversas aplicaciones. El incremento en su uso, se debe a que posee un buen manejo de la energía y a su reducido tamaño.

En la tabla 3.3 se describen las características de la fuente *switching* LLAS500.

Modelo LLAS500	
Tensión de entrada	90 - 240V CA 50/60 Hz
Voltaje de salida	3 / 4.5 / 6 / 7.5 / 9 / 12 VDC
Carga máxima	500 mA

Tabla 3.3: Características de la fuente *switching*

En el ANEXO A.4 se muestran las características de la fuente *switching* LLAS500.

3.1.4 BALANZA

La balanza utilizada para el pesaje de los productos en el proyecto es la Báscula Torrey modelo EQ-5/10, que se muestra en la figura 3.5.

Figura 3.5: Balanza

Es una balanza de alta precisión, contiene un gabinete y plato en acero inoxidable, capacidad 5 kg/ 10 lb/ 160 onzas; funciona con batería recargable (cargador de baterías de 120 VAC/9 VDC) o con corriente eléctrica. Tiene función de tara progresiva, división mínima 1 gr/0.002 Lb/0.05 oz, 5000 divisiones y puede pesar máximo 10 Kg.

En la tabla 3.4 se describen las características de la balanza.

Modelo EQ – 5/10	
Capacidad	5 Kg / 10 lb / 160 oz
Display	Cuarzo líquido
Corriente eléctrica	110 V / 60 Hz
Conector serial	RS – 232
Tara máxima	2.5 Kg / 5 lb
Plato	20 x 27.9 cm
Temperatura de operación	-10 a 40 °C
Temperatura de almacenaje	-20 a 50 °C
Peso neto	8 Kg / 17.6 lb

Tabla 3.4: Características de la balanza

En el ANEXO A.5 se muestran las características de la báscula EQ-5/10.

3.1.5 CABLES

3.1.5.1 Conector DB9 hembra

Para el prototipo se utilizó un cable con conector DB9 hembra a DB9 hembra con conexión directa para conectar la balanza con el cable convertidor de USB a RS232.

La distribución de pines del conector DB9 hembra se muestra en la figura 3.6

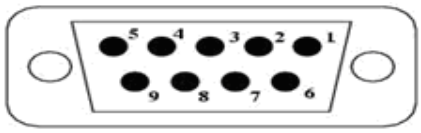


Figura 3.6: Conector DB9 hembra [8]

La información asociada a cada uno de los pines del conector DB9 hembra se describen en la tabla 3.5.

Número de pin	Señal
1	DCD (*Data Carrier Detect*): Detector de portadora
2	RX: Recibir datos
3	TX: Transmitir datos
4	DTR (*Data Terminal Ready*): Terminal de datos lista
5	GND: Señal de tierra
6	DSR (*Data Sheet Ready*): Ajuste de datos lista
7	RTS (*Request To Send*): Permiso para transmitir
8	CTS (*Clear To Send*): listo para enviar
9	RI (*Ring Indicator*): Indicador de llamada

Tabla 3.5: Características de los pines del conector hembra DB9 [9]

El cable con conector DB9 hembra a DB9 hembra con conexión directa utilizado en el proyecto se muestra en la figura 3.7.

Figura 3.7: Cable con conector DB9 hembra a DB9 hembra

3.1.5.2 Convertidor de USB a RS232

En la figura 3.8 se muestra el cable convertidor de USB a RS232 utilizado en el proyecto, que le permite conectar el cable con conectores DB9 hembra al puerto USB en la computadora.

Figura 3.8: Convertidor de USB a RS232 TU-S9

Se instala como un puerto COM de *Windows* estándar, con señales de control de módem Full RS-232, señales de datos RS-232; TxD, RxD, RTS, CTS, DSR, DTR, DCD, RI, GND y admite BUS-*Power*, no requiere de adaptador eléctrico externo.

En la tabla 3.6 se muestran las características del cable convertidor de USB a RS232 utilizado en este proyecto.

Modelo TU-S9	
Interfaz	Tipo A USB 1.1 Macho RS-232 (9-pin)
Compatible con sistemas operativos	*Windows* 98SE/ME/2000/2003 *Server*/XP/Vista, Mac 10.x
Longitud del cable	66 cm (26 pulgadas)
Indicador de fusil	28/24 AWG
Gama de datos	500 kbps
Consumo eléctrico	500 mA (máx)
Peso	75g. (0,2 lb)
Temperatura	Operación: 0°~ 40°C Almacenamiento: -10°~ 45°C
Humedad	85% (sin condensación)

Tabla 3.6: Características del cable convertidor de USB a RS232 TU-S9

3.2 SOFTWARE DEL PROTOTIPO

Antes de realizar el *software* se analizó qué lenguaje de programación será utilizado; de acuerdo a la aplicación del prototipo el *software* deberá tener la facilidad de poder ser reconocido en cualquier sistema operativo, sin necesidad de realizar cambios en el *software*, porque no todos los supermercados utilizan el mismo sistema operativo en las computadoras de las cajas.

El lenguaje de programación utilizado para este proyecto que puede ser llevado a cualquier sistema operativo es JAVA.

Se realizará una breve descripción del lenguaje de programación JAVA y se describirá de manera general el código fuente desarrollado.

3.2.1 INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN JAVA [9, 10, 11]

JAVA es un lenguaje de programación orientada a objetos, desarrollado por *James Gosling* y sus compañeros de *Sun Microsystems* al principio de la década de los 90.

Sun Microsystems, pensó en crear un lenguaje que se basara en lenguajes de implementación que más se utilizaban en el mundo como son C y C++.

El lenguaje de programación JAVA no debe ser confundido con JavaScript ya que éste es un lenguaje interpretado, es decir, que no requiere compilación. La sintaxis de JAVA es similar a la utilizada en los lenguajes C y C++, diferenciándose por la sencillez de JAVA porque elimina herramientas de bajo nivel como es el caso de los punteros.

Java es un lenguaje que ha sido diseñado para producir *software*; presenta las siguientes características:

- Confiable: Minimiza los errores que se escapan a la fase de prueba.

- Multiplataforma: Una vez ya compilado el código JAVA puede llevarse a cualquier sistema operativo sin ser modificado y ejecutarlo allí sin problemas. Esto es porque el código se compila en un lenguaje intermedio llamado *bytecodes* que podrá ser leído independientemente de la máquina. Este lenguaje intermedio es interpretado por la Máquina Virtual Java (JVM) que será necesaria en la plataforma en la que se quiera ejecutar el código.
- Seguro: *Applets* recuperados por medio de la red no pueden causar daño a los usuarios.
- Orientado a objetos: Beneficioso tanto para el proveedor de bibliotecas de clases como para el programador de aplicaciones. Cuando se escriben programas en lenguajes orientados a objetos, no se definen objetos verdaderos sino se definen clases de objetos.
- Robusto: Los errores se detectan en el momento de producirse, lo que facilita la depuración.

Java es un lenguaje provisto de interfaces gráficos con el usuario, que permitirá aprovechar capacidades de multimedia de gráficos, imágenes, animación, audio e incluso video. Otra ventaja de Java es que posee muchas clases de los paquetes de la Java API (del inglés *Application Programming Interface* - Interfaz de Programación de Aplicaciones) que pueden ser reutilizados.

Con todo esto Java tiene un potencial para convertirse en un lenguaje de programación de aplicación general más importante del mundo.

3.2.1.1 Fundamentos de un Entorno Java Típico

Antes de la ejecución de un programa de Java se deberá pasar por cinco fases que son editar, compilar, cargar, verificar y ejecutar.

- En la primera fase, editar, consiste en realizar correcciones a los programas, los cuales posteriormente se almacenarán. En Java los programas terminan con la extensión .java.

- En la fase segunda se compilará. El comando javac se utilizará para compilar el programa. Se compilará para traducir el programa Java a códigos de bytes que será comprendido por el intérprete de Java. Si esta compilación se hace correctamente se generará un archivo con una extensión .class, el cual poseerá los códigos de bytes.
- El paso siguiente a la compilación del programa es crear un archivo HTML (*Hypertext Markup Language*) para poder cargar el *applet* en el navegador (éste leerá los archivos) y procederá a ejecutarlo.
- La tercera fase es la denominada carga. En esta fase se almacena en memoria ese archivo .class.
- En la cuarta fase los códigos de bytes son comprobados por el verificador de códigos de bytes. Este procedimiento se da para comprobar que los códigos de bytes sean válidos y no violen restricciones de seguridad de Java. Estas reglas de seguridad existen para evitar que los programas de Java que llegan a través de una red no causen daños a los archivos y al sistema.
- En la quinta fase se interpreta cada uno de los códigos de bytes.

La portabilidad existente en Java se dificulta debido a las diferencias entre los compiladores, intérpretes y computadoras.

Los *applets* de Java se ejecutan y se despliegan dentro de una página *Web* con otros elementos de la página y debido a esto poseen reglas especiales.

Las aplicaciones de Java son programas que serán ejecutados por el intérprete de Java y son programas autónomos que no necesitan un navegador ni un *appletviewer*[4] para ejecutarse. Una aplicación de Java comienza con el método *main*. Por esto se dice que es más difícil crear un *applet* que una aplicación Java.

Los sistemas Java se forman de las siguientes partes:

4 El *appletviewer* es un programa que prueba *applets* de Java que solo tomará en cuenta a la etiqueta HTML *applet*. [17]

3.2.1.1.1 Un entorno

El entorno utilizado para el proyecto es el *NetBeans* que es un entorno gratuito de código abierto para la generación de código en diversos lenguajes (especialmente pensado para Java). La descarga de este entorno se lo puede hacer desde la página web www.netbeans.org.

3.2.1.1.2 El lenguaje

El compilador Java y el intérprete Java son diferentes. El compilador se utiliza para los archivos fuente, a fin de crear archivos .class y el intérprete se usa para ejecutar los archivos de clase.

Todo el código fuente Java se escribe en documentos de texto con extensión .java. Al ser un lenguaje para Internet, la codificación de texto debe permitir a todos los programadores de cualquier idioma escribir ese código. Eso significa que Java es compatible con la codificación *Unicode*.

En la práctica significa que los programadores que usen lenguajes distintos del inglés no tendrán problemas para escribir símbolos en su idioma, esto se puede extender para nombres de clase, variables, etc.

El compilador de Java no hace caso a los espacios, tabuladores y demás; en Java éstos solo sirven para mejorar la comprensión del programa.

El compilador detectará los errores de sintaxis, en cambio los errores de lógica se perciben el momento de la ejecución; al momento de existir un error de lógica fatal produce que el programa termine antes de lo esperado. El error de lógica no fatal hará que el programa continúe su ejecución pero sin dar los resultados esperados.

El compilador de Java solo podrá evaluar expresiones en donde los tipos de datos sean idénticos.

3.2.1.1.3 La interfaz de programación de Java API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones)

El API (*Application Programming Interface*-Interfaz de Programación de Aplicaciones) de Java o comúnmente conocido como la interfaz de programación de aplicaciones provee de un conjunto de clases utilitarias para efectuar toda clase de tareas necesarias dentro de un programa.

La API (*Application Programming Interface*-Interfaz de Programación de Aplicaciones) Java está organizada en paquetes lógicos, donde cada paquete contiene un conjunto de clases relacionadas semánticamente.

3.2.1.1.4 Bibliotecas de clases

Java está formado por varias clases y métodos. Los programadores hacen uso de esta diversidad de clases y métodos que existen en las bibliotecas de clases de Java para formar sus programas. Existen ventajas cuando se crean las clases y los métodos debido a que se conoce bien su funcionamiento, pero la desventaja es que se invierte tiempo en ello.

Al utilizarse clases y métodos de las bibliotecas de Java se mejora la transportabilidad, pues estas clases y métodos se incluyen en todas las implementaciones de Java.

Toda implementación de Java debe tener las siguientes bibliotecas de clases:

- Manejo de archivos
- Comunicación de datos
- Acceso a la red Internet
- Acceso a bases de datos
- Interfaces gráficas

La interfaz de programación de estas clases es estándar, es decir en todas ellas las operaciones se invocan con el mismo nombre y los mismos argumentos.

3.2.1.2 API (*Application Programming Interface*-Interfaz de Programación de Aplicaciones) de Comunicaciones

El API (*Application Programming Interface*-Interfaz de Programación de Aplicaciones) de comunicaciones es una extensión estándar que permite establecer comunicaciones con los puertos serial RS-232 y paralelo IEEE-1284, esto permitirá realizar aplicaciones de comunicaciones que utilizan los puertos de comunicaciones (tarjetas inteligentes, fax) independientes de la plataforma.

El API (*Application Programming Interface*-Interfaz de Programación de Aplicaciones) de comunicaciones no se encuentra incluido en el JDK (*Java Development Kit*) y el API (*Application Programming Interface*-Interfaz de Programación de Aplicaciones) de comunicaciones es una extensión del JDK (*Java Development Kit*), así que antes de empezar a trabajar se deberá instalar este nuevo API (*Application Programming Interface*-Interfaz de Programación de Aplicaciones) en las máquinas en las que va a realizar el desarrollo del *software* y que van a ejecutar programas que trabajen con los puertos serie y paralelo.

3.2.1.2.1 Instalación del API (Application Programming Interface-Interfaz de Programación de Aplicaciones) de comunicaciones

Lo primero que se hace es obtener el API (*Application Programming Interface*-Interfaz de Programación de Aplicaciones) de comunicaciones; éste se puede bajar fácilmente de Internet ya que no ocupa más de 300 KB.

Una vez que se desempaca el fichero se procede a realizar los siguientes pasos:

- Se copia el fichero Win32com.dll a <JDK>\jre\bin y en <JRE>\bin
- Se copia el archivo comm.jar a <JDK>\jre\lib\ext y en <JRE>\lib\ext
- Se copia javax.comm.properties a <JDK>\jre\lib y <JRE>\lib. Este fichero contendrá los *drivers* que soportará el API, esto se realiza así ya que se podrá crear nuestros propios *drivers* de puertos

Los directorios que se muestran están en el C:/JAVA.

3.2.1.2.2 Características del API (Application Programming Interface-Interfaz de Programación de Aplicaciones) de comunicaciones

En el paquete de comunicaciones javax.comm se tiene una serie de clases que permiten tratar varios niveles de programación, estos niveles son los siguientes:

- Nivel alto: En este nivel se tienen las clases CommPortIdentifier y CommPort que permiten el acceso a los puertos de comunicación.
- Nivel medio: Con las clases SerialPort y ParallelPort que cubren los interfaces físico RS-232 para el puerto serie e IEEE-1284 para el puerto paralelo.
- Nivel bajo: Este nivel ya toca el sistema operativo y en él se encuentra el desarrollo de *drivers*.

3.2.1.2.3 Servicios que proporciona este paquete

- Obtener los puertos disponibles así como sus características
- Abrir y mantener una comunicación en los puertos
- Resolver colisiones entre aplicaciones. Gracias a este servicio se podrá tener varias aplicaciones Java funcionando y,
- Al utilizar los mismos puertos no sólo se sabrá que el puerto está ocupado sino que se podrá conocer qué aplicación lo está utilizando

Se dispone de métodos para el control de los puertos de entrada/salida a bajo nivel, de esta forma no solo se limita a enviar y recibir datos sino que se podrá saber en qué estado está el puerto. Así en un puerto serie se podrá no solo cambiar los estados sino que se podrá programar un evento que notifique el cambio de cualquier estado.

3.2.1.3 Conectar Java y Access para la Base de Datos

La capacidad para acceder a bases de datos desde Java la ofrece la API JDBC (*Java DataBase Conectivity*). JDBC es un estándar para manejar bases de datos en Java.

ODBC (*Open Database Conectivity*) es un estándar de *Windows* para manejar bases de datos, de forma que cualquier programa en *Windows* que desee acceder a bases de datos genéricas debe usar este estándar.

3.2.1.3.1 Controlador JDBC-ODBC

Se establece un puente entre JDBC y ODBC. Este controlador convierte todas las llamadas JDBC a llamadas ODBC y realiza la conversión correspondiente de los resultados.

Este controlador es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real.

La necesidad del JDBC, a pesar de la existencia de ODBC, viene dada porque ODBC es un interfaz escrito en lenguaje C, que al no ser un lenguaje portable, haría que las aplicaciones Java también perdiesen la portabilidad.

En la figura 3.9 se muestra cómo funciona el controlador JDBC-ODBC.

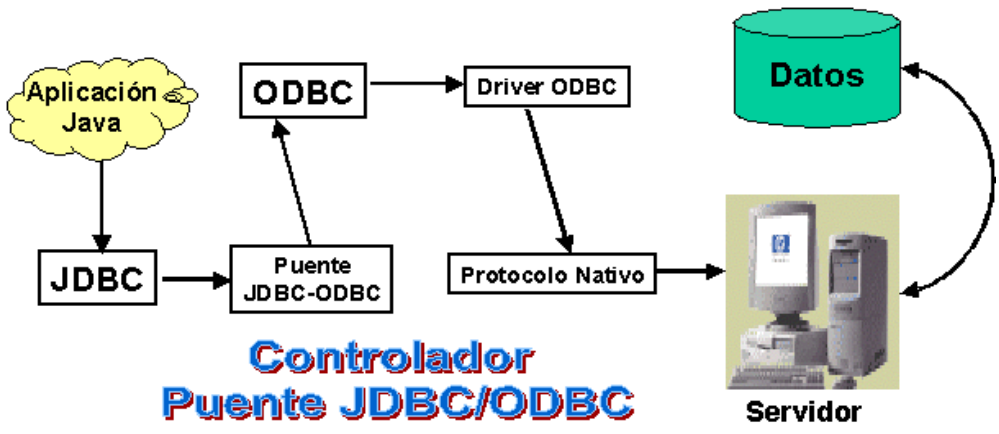


Figura 3.9: Controlador JDBC-ODBC [9]

3.2.1.3.2 Crear un nuevo DSN (Data Source Name)

Para realizar la conexión a una base de datos ODBC se necesitará crear un perfil DSN desde el panel de control y posteriormente se accederá a la base de datos a partir del nombre del perfil. En el perfil DSN lo que se hace es indicar el *driver* a utilizar, así como el archivo o archivos del origen de datos.
En el ANEXO C se describen los comandos para la programación JAVA.

3.2.2 DESARROLLO DEL CÓDIGO FUENTE PARA EL PROYECTO

El código fuente del programa para el prototipo se ilustra en el Anexo B.

A continuación se explicarán que es lo que realiza cada clase del código fuente creado.

3.2.2.1 Clase Main.java

Esta clase, mostrada en el ANEXO B.2, crea el objeto “ctrlTarjetas” que permite comunicarse con el hardware conectado, así como presentar al usuario una interfaz con la cual se puede interactuar. Es la clase que inicia el funcionamiento del programa.

3.2.2.2 Clase COMM_library.java

Esta clase, que se muestra en el ANEXO B.3, es para describir el puerto por donde se recibirán los datos de la balanza y para almacenar estos datos en una variable.

En el constructor COMM_library se crea la variable COMM para indicar el puerto de comunicaciones y sus propiedades de conexión (Velocidad, bits de datos, bits de parada, paridad), estos valores deberán ser los mismos que los utilizados por la balanza para comunicarse correctamente entre sí.

Antes de recibir los datos se borra todo lo que haya en el *buffer* para asegurar que no existan datos que alteren el valor recibido.

Se envía por el puerto serial a la balanza la letra “P”, con esto la balanza sabe que debe enviar el peso medido, el cual es almacenado en el *buffer* de entrada del puerto serial de la computadora.
Los datos recibidos de la balanza antes de guardarlos en la variable DATOS son almacenados en una variable temporal.

Luego se usa la variable DATOS para guardar el valor recibido de la balanza.

Al final, se retorna el valor de la variable DATOS a la función que hizo la petición; en este caso la petición provino de la interfaz del usuario.

3.2.2.3 Clase ControladorPuerto.java

Esta clase que se muestra en el ANEXO B.4 permite manipular eventos generados por el puerto serial, ya que está implementada de la clase SerialPortEventListener, que soporta eventos y que avisará si es que existe uno.

El constructor ControladorPuerto es para completar las propiedades de la variable COM antes creada y que permite configurar los valores que se usará en el puerto de comunicaciones de la computadora conectada al lector de etiquetas; parte de esto es indicarle quien responderá cuando hay un evento (si es que hay un dato en el pin de recepción del puerto de la computadora) y de ahí ejecutar el código correspondiente.

El sistema está diseñado de tal manera que al pasar una etiqueta por el lector debe automáticamente buscarse a qué producto pertenece, es aquí donde se utiliza el evento SerialPortEvent.DATA_AVAILABLE. Al pasar la etiqueta por el lector, éste transmite el código de la etiqueta; este código es almacenado en el *buffer* de la computadora y luego el programa es avisado del evento de disponibilidad de datos. El programa captura el evento en la función

serialEvent(SerialPortEvent event) y realiza las operaciones requeridas por el programa.

Antes de leer los datos que están en el *buffer* nos aseguramos que la variable temporal esté vacía, luego se pasa la información del *buffer* a la variable temporal y con la función *substring*(6,11) se escoge solo una parte de los datos existentes en la variable temporal; éstos son los valores desde el dato 6 hasta el 10, el dato 11 no se lo incluye. Estos 5 valores son escogidos porque los demás datos son iguales en todos los códigos de las etiquetas.

Inmediatamente se busca este código en la base de datos usando un objeto creado a partir de la clase bdd_ODBC.java que permite identificar los datos completos del producto. Una vez hecha la consulta se comprueba que este producto no exista en la lista presentada al usuario, ya que de existir no debería ser ingresado nuevamente.

3.2.2.4 Clase Mensajes_Frame.java

Esta clase, que se muestra en el ANEXO B.5, es para crear la ventana que se muestra al usuario cuando se desea eliminar un producto, pero si antes no se ha seleccionado ninguno, es decir cuando existe un error. La ventana se muestra en la figura 3.10.

Figura 3.10: Ventana creada en la librería Mensajes_Frame

Parte del código necesario para su funcionamiento es generado por el compilador según las propiedades que se vayan escogiendo para la creación de esta ventana.

3.2.2.5 Clase bdd_ODBC.java

En esta clase, que se muestra en el ANEXO B.7, se realiza la conexión de Java con el controlador JDBC-ODBC y éste a su vez con la base de datos “Productos” creada en *Microsoft Access* enlazada a ODBC de *Windows*.
Al realizar la conexión a una base de datos ODBC se necesitará crear un perfil DSN y posteriormente se accederá a la base de datos a partir del nombre del perfil.

Éstos son los pasos a llevar a cabo para crear un perfil DSN.

1. Se presiona menú inicio y se selecciona “Ejecutar”, como se muestra en la figura 3.11.

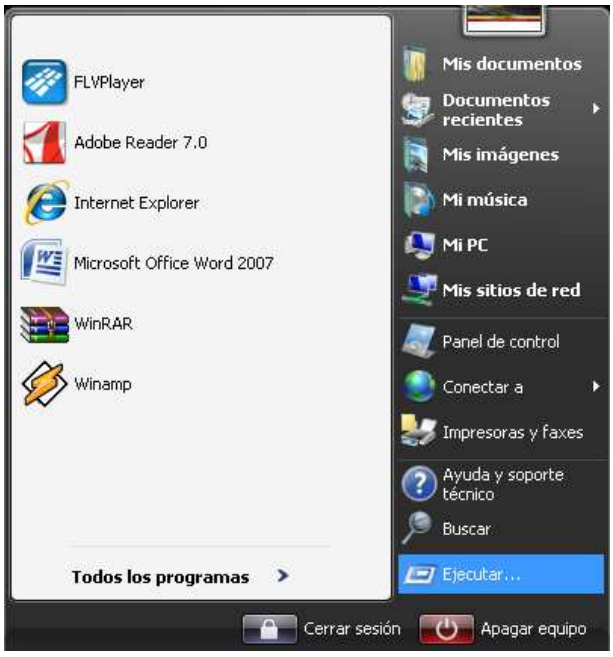


Figura 3.11: Pantalla para seleccionar “Ejecutar”

2. En la pantalla que se muestra en la figura 3.12 se escribe odbcad32 y se presiona “Enter”.

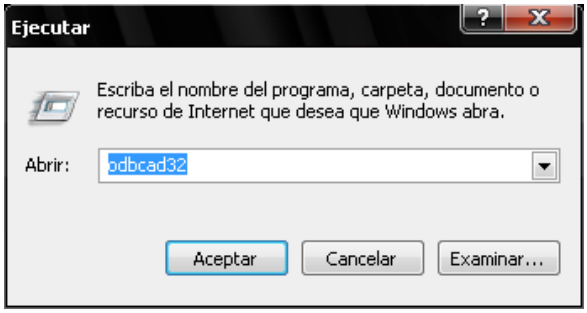


Figura 3.12: Pantalla de “Ejecutar”

3. En la pantalla que se muestra en la figura 3.13 aparecerá la pestaña “DSN usuario”. Para crear un nuevo perfil hacer *click* en “Agregar”.

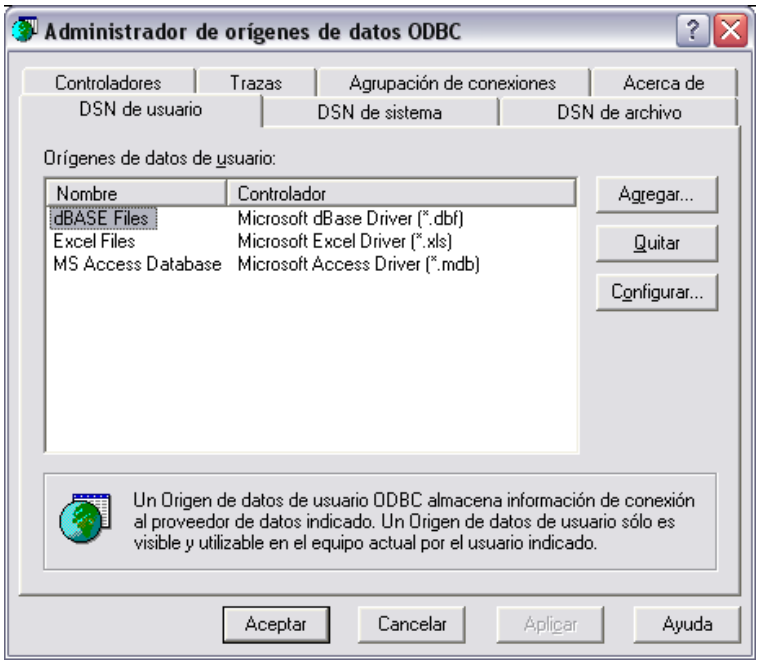


Figura 3.13: Pantalla que contiene la pestaña “DNS de usuario”

4. En la pantalla que se muestra en la figura 3.14 pedirá que se ingrese el controlador que se va a usar en el nuevo perfil. En este caso será *Microsoft Access Driver* (*.mdb).

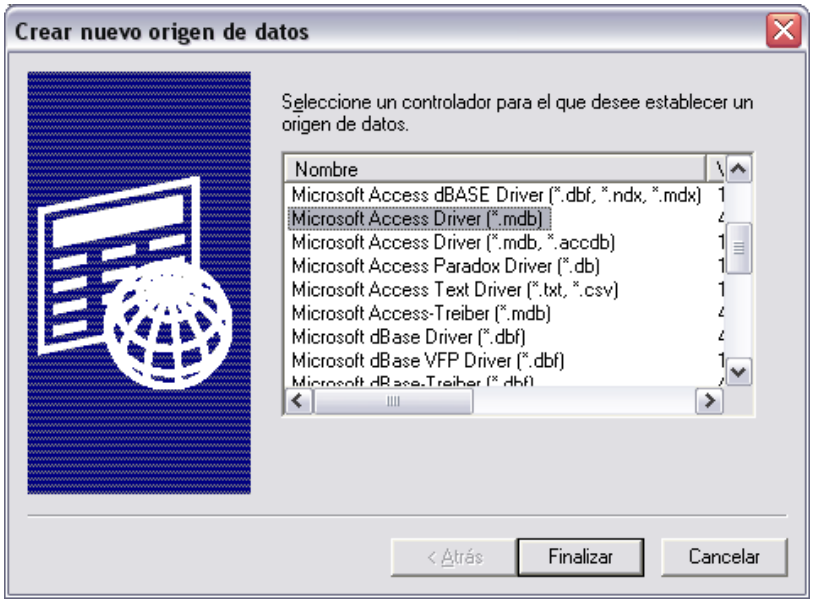


Figura 3.14: Pantalla para seleccionar un controlador

5. En la pantalla que se muestra en la figura 3.15 se da un nombre al origen de datos y se especifica el archivo .mdb de origen, en este caso para el proyecto se eligió “Productos”.

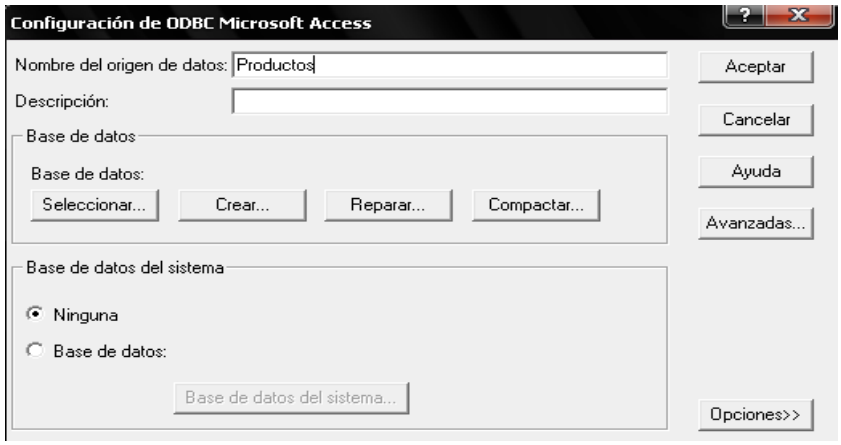


Figura 3.15: Pantalla para dar un nombre al origen de datos

6. Tras aceptar la ventana ya se tendrá creado un perfil como se observa en la figura 3.16, con lo que ya se podrá comenzar a programar.

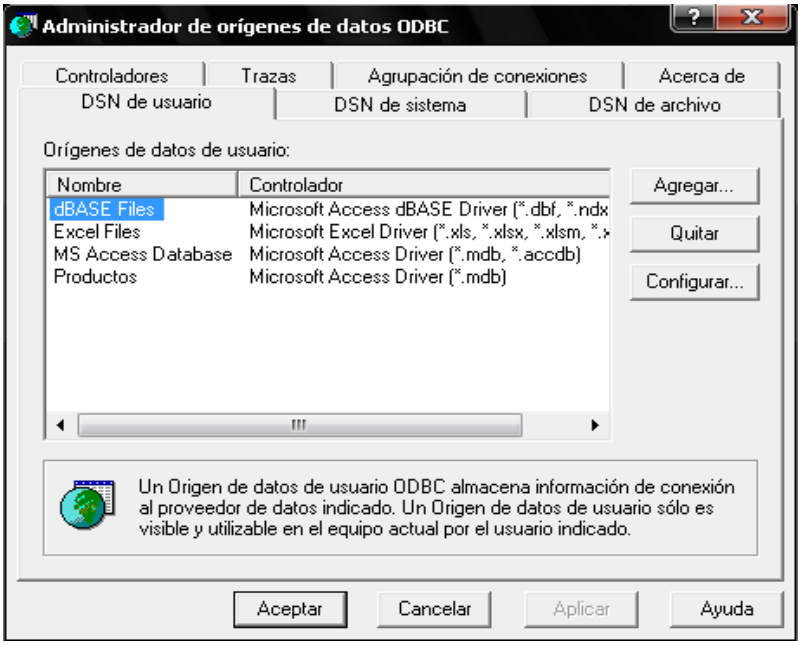


Figura 3.16: Pantalla con el perfil creado

Esta conexión es usada por el programa para acceder a los datos de los productos existentes en un archivo de *Microsoft Access* una vez que son detectados por el lector de tarjetas.

Si la conexión no se pudo realizar el programa no tendrá acceso a los datos almacenados y se dará a conocer de este error en la consola de sistema de JAVA.

3.2.2.6 Clase Productos_Frame.java

En esta clase, que se muestra en el ANEXO B.6, se abrirá y se dará propiedades al puerto de la computadora donde se conectará la balanza, esto se hace creando un objeto de la clase COMM_library explicada anteriormente.

También se crea la ventana de presentación al usuario que se muestra en la figura 3.17.

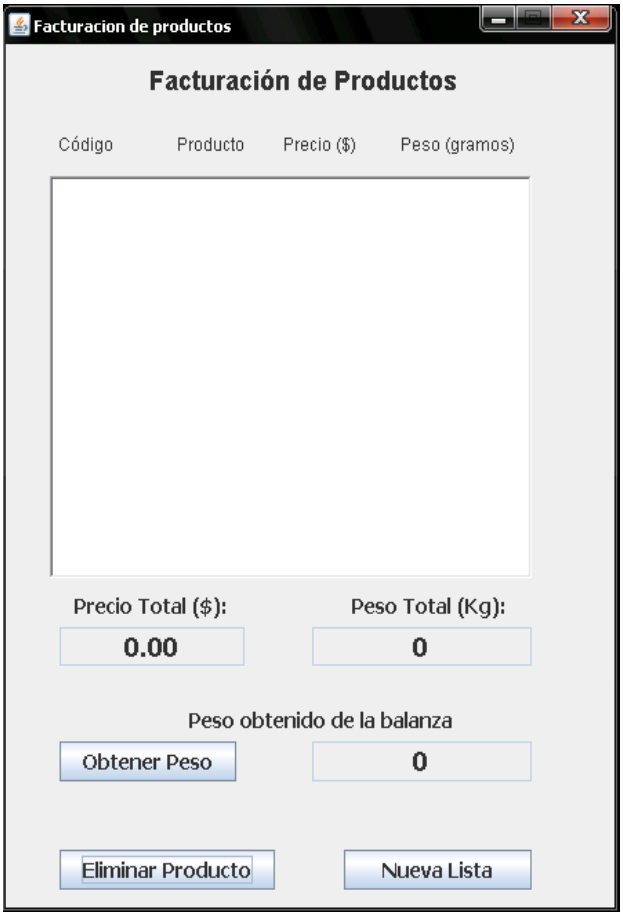


Figura 3.17: Ventana presentada al usuario

Antes de mostrar los datos del producto leído por el lector la clase ControladorPuerto.java le pide a esta clase que verifique si antes no se ha leído ya este producto, debido a que Productos_Frame.java contiene los productos registrados anteriormente.

Para obtener el precio total se toman los datos del precio que están almacenados en la lista interna del programa y se suma al obtenido en la base de datos; este valor se presenta al usuario debajo de “Precio Total”.

Para obtener el peso total se suman los datos del peso que están almacenados en las lista interna del programa de los productos detectados anteriormente con el producto obtenido de la base de datos; este valor total al estar en gramos, se multiplica por 0.001 para ser mostrado en Kg y se escogen los primeros 5 dígitos.

Al presionar el botón “Eliminar Producto” se restará el valor del precio del producto seleccionado del valor del precio total y se restará el valor del peso del producto seleccionado del valor del peso total. También se eliminan los datos del producto que son mostrados en la lista.

Cada vez que se necesite obtener el valor de la balanza se presionará el botón “Obtener Peso”, que éste al ser presionado envía el caracter de la letra “P” para que la balanza envíe por el puerto serial el dato obtenido.

El botón “Nueva Lista” es para borrar todos los datos obtenidos y realizar una nueva facturación de productos.

El código para la presentación de la ventana es generado por el compilador según qué propiedades se vayan escogiendo.

3.3 CONSTRUCCIÓN DEL PROTOTIPO

Para la construcción del prototipo se realizaron los siguientes pasos:

1. Conectar al lector una fuente *switching* para la alimentación eléctrica con un voltaje de 12 voltios; como se muestra en la figura 3.18.

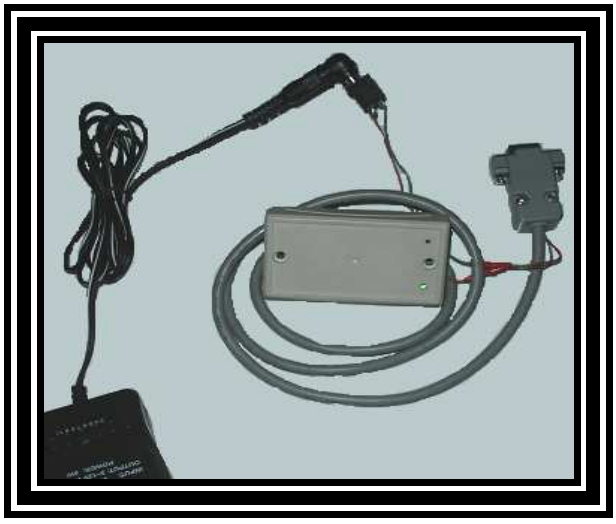


Figura 3.18: Conexión de una fuente *switching* con el lector

2. Conectar el lector con el puerto serial RS232 de la computadora; como se muestra en la figura 3.19.

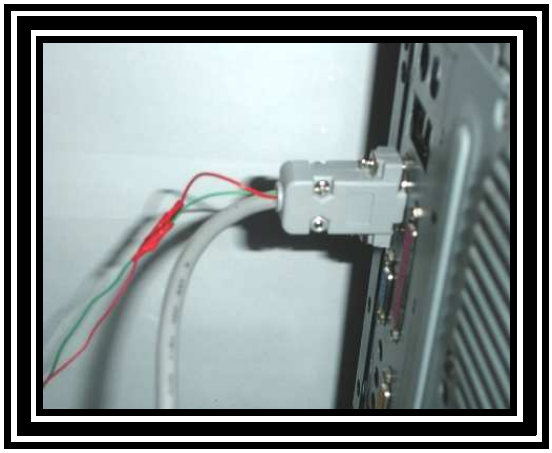


Figura 3.19: Conexión del lector con la computadora por el puerto serial

3. Comprobar con el programa de comunicaciones *HyperTerminal* si las etiquetas envían señales de respuesta al lector y éste a su vez envía al computador información recibida de las etiquetas identificadas. Si la identificación se realiza, se observarán los códigos de las etiquetas en la pantalla del *HyperTerminal,* tal como se muestra en la figura 3.20.

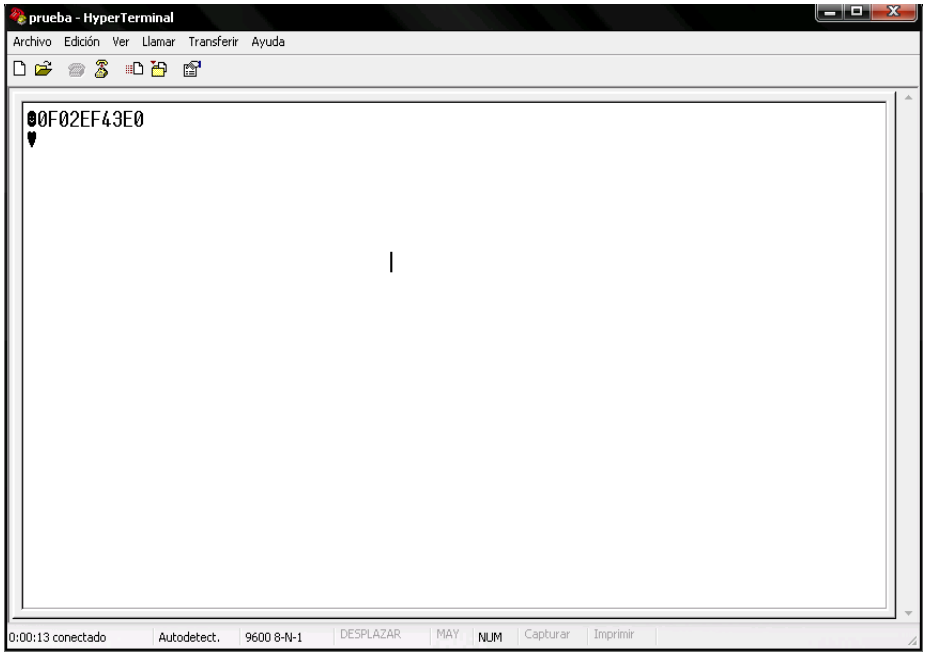


Figura 3.20: Prueba con el programa de comunicaciones *HyperTerminal*

4. Para la conexión de la balanza de mesa con la computadora se lo realiza al puerto serial USB por medio del cable convertidor de USB a RS232, porque el puerto RS232 de la computadora ya está siendo utilizado para la conexión del lector; en la figura 3.23 se muestra la conexión. Pero antes de realizar esta conexión se construyó un cable con conector DB9 hembra a DB9 hembra con conexión directa para conectar la balanza con el cable convertidor, puesto que el conector de la balanza y el conector RS232 del cable convertidor son macho, y no se pueden conectar directamente.

   La conexión del cable al conector DB9 hembra según los colores se muestra en la tabla 3.7.

Rojo	tierra
Verde	sin conexión
Blanco	sin conexión
Rojo	tierra
Verde	sin conexión
Blanco	sin conexión

Tabla 3.7: Colores para la conexión del cable al conector DB9 hembra

El cable convertidor de USB a RS232 utilizado para la construcción del prototipo se muestra en la figura 3.21.

Figura 3.21: Cable convertidor de USB a RS232

El cable con conector hembra-hembra con conexión directa se muestra en la figura 3.22.

Figura 3.22: Cable con conector hembra-hembra

La conexión de la balanza a la computadora con el cable convertidor se muestra en la figura 3.23.

Figura 3.23: Conexión de la balanza con el cable

Para que el cable convertidor de USB a RS232 funcione, se instaló el *driver* de éste, el manual de instalación se muestra en el ANEXO D.

5. Se realizaron los siguientes pasos previos al desarrollo del *software*:

- Los dispositivos tienen un formato de trama que debe seguirse, para poder establecer una comunicación adecuada. Los dispositivos decodificarán los datos siguiendo el formato de cada uno, así que se pondrá atención en el envío y recepción de datos.

  Primero se empezó por hacer la parte de la comunicación serial entre el lector y la computadora; para esto se vio en el manual del lector, que se

muestra en el ANEXO A.2, para obtener información acerca del interfaz RS-232 y su protocolo, como se muestra en la figura 3.24.

STX (02HEX)	DATA (10 HEX CHARACTERS)	CR	LF	ETX (03 HEX)

Figura 3.24: Estructura de datos para RS-232

Para el puerto RS232 serial ASCII del lector se tienen las siguientes propiedades: 9600 Baudios, sin paridad, 8 bits de datos, 1 bit de parada.

Para que la balanza envíe al puerto serial de la computadora la información del peso obtenido, se requiere que ésta reciba la letra “P” por el puerto serial.

- Una vez comprobado el funcionamiento de la comunicación entre el lector y la computadora, se conecta la balanza con la computadora como se muestra en la figura 3.25 con lo que se verifica la comunicación entre la balanza y la computadora con un programa que vino con la compra de la balanza para obtener el peso, como se indica en la figura 3.26.

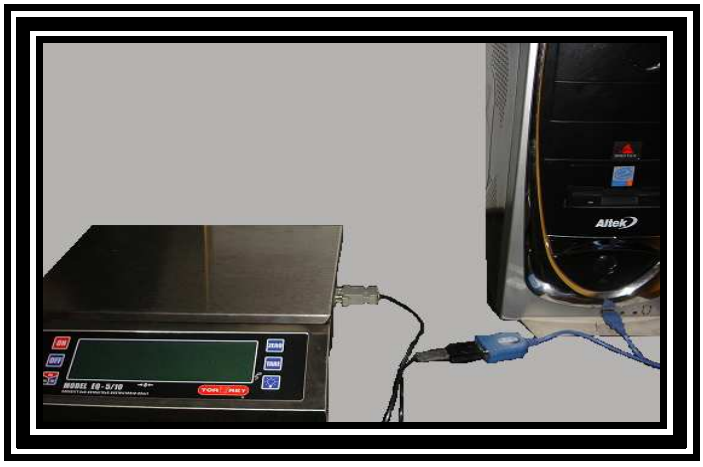


Figura 3.25: Comunicación entre la balanza y la computadora

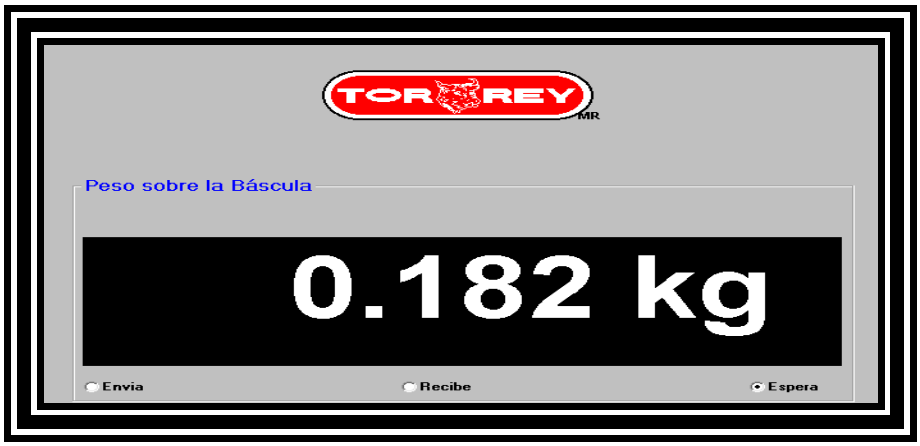


Figura 3.26: Comprobación del funcionamiento de la balanza

- Posteriormente se procedió a realizar la base de datos en *Microsoft Office Access*, en donde se puso las características del producto, nombre, precio y peso según el código de cada etiqueta, como se muestra en la figura 3.27.

  Es posible crear aplicaciones con bases de datos que interactúen con el lenguaje de programación Java, con MySQL, *Access* de *Microsoft* y SQL-*Server* 2000 de *Microsoft*, con el manejo de las AWT[5] y aplicaciones de consola.

6. Se realiza el *software* del programa siguiendo los siguientes pasos:

   - Se establece la comunicación del lector con la computadora por el puerto serial, dando propiedades al puerto por donde va a recibir los datos la computadora y se verifica si estos datos llegan bien según las propiedades dadas.
   - Se realiza la comunicación de la balanza con la computadora por el puerto serial, dando propiedades al puerto.
   - Los datos que envían el lector serán comparados en la base de datos que fue creada en *Microsoft Office Access*, para que según el código se obtengan los datos del nombre del producto, precio y peso.

5 AWT (*Abstract Window Toolkit*) Es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java. [17]

- Una vez que se tiene el producto se realiza la suma de precios y pesos para mostrar en la pantalla de presentación que se muestra en la figura 3.28.
- Los datos que envía la balanza serán mostrados en la pantalla de presentación haciendo un *click* en el botón “Obtener Peso” que se encuentra en esta pantalla, mostrada en la figura 3.28.
- Si se quiere eliminar un producto después de haber sido identificado se crea el botón “Eliminar Producto” y si se desea tener una nueva lista de productos se crea el botón “Nueva Lista”. Todos estos botones son creados en la pantalla de presentación.
- Se crea una pantalla de mensaje para cuando no se ha seleccionado ningún producto y se presiona el botón “Eliminar Producto” aparezca esta pantalla con el mensaje “No se ha seleccionado ningún producto”. La pantalla de mensaje se muestra en la figura 3.29.

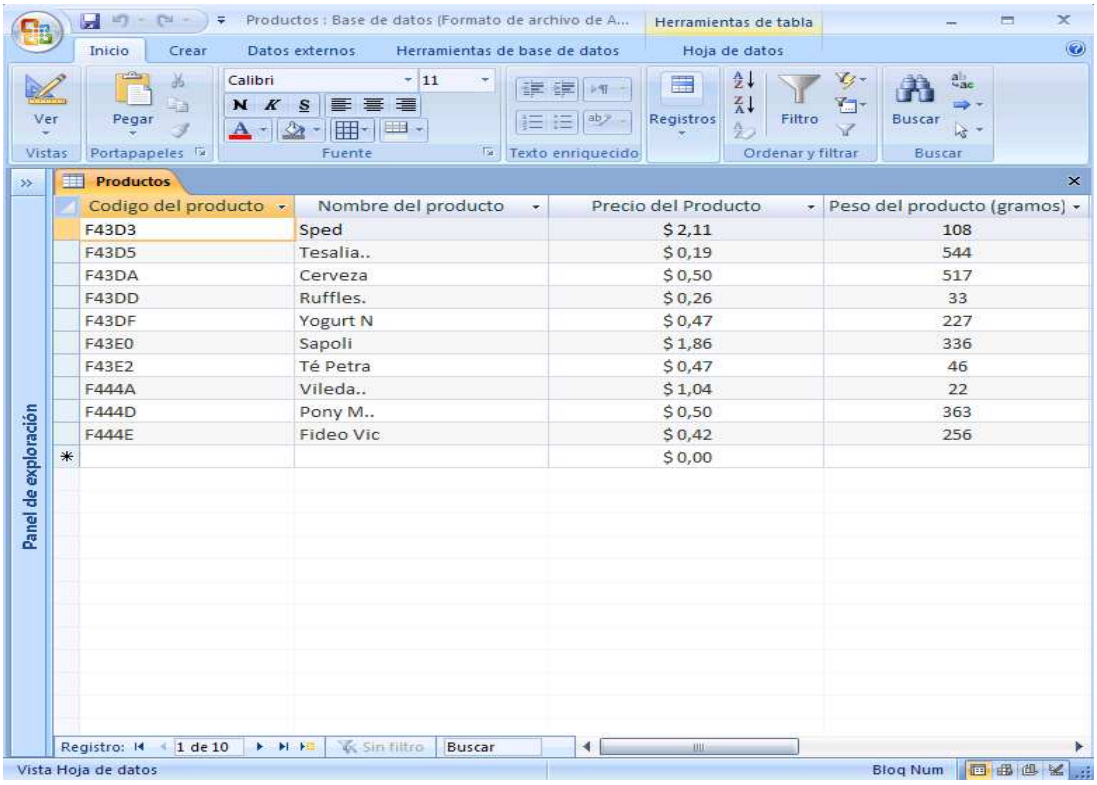


Figura 3.27: Base de datos en *Microsoft Office Access*

Figura 3.28: Pantalla de presentación al usuario

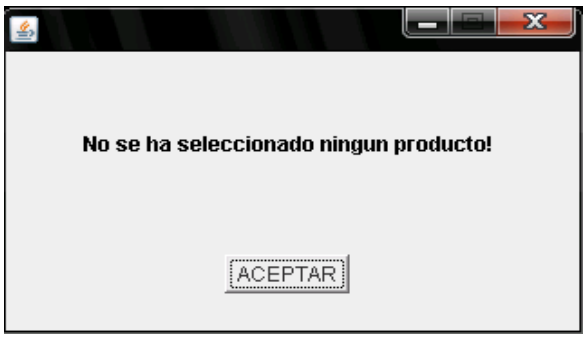


Figura 3.29: Pantalla de mensaje

El *software* realizado se muestra en el ANEXO B.

7. Para colocar los productos simulando que están en un coche de compras se utilizó una canasta que se muestra en la figura 3.30.

Figura 3.30: Canasta

8. La canasta será colocada encima de la balanza como se muestra en la figura 3.31, simulando que el coche de compras está encima de la balanza de suelo. El peso de la canasta no será influenciado en los datos para obtener el peso total, porque tiene la función de tara.

Figura 3.31: Canasta con la balanza

9. Finalmente se conectarán todos los equipos para el funcionamiento del prototipo como se muestra en la figura 3.32.

Figura 3.32: Prototipo implementado

10. Finalmente comprobado el correcto funcionamiento del prototipo se realizaron pruebas, que serán mostradas en el siguiente capítulo.

La balanza de mesa y la canasta son utilizadas por ser un prototipo simulando a una balanza de suelo y a un coche de compras respectivamente que se utilizarán para la implementación comercial.

El lector y etiquetas RFID utilizadas son de corto alcance por ser un prototipo; para la implementación comercial se deberán utilizar equipos de mayor alcance.

La razón de no utilizar los equipos para la implementación comercial es por el alto costo económico que tiene cada uno. El proyecto se lo realizó sin el auspicio, por lo que no se tienen los recursos necesarios para la implementación comercial del prototipo.

CAPÍTULO 4. PRUEBAS Y PRESUPUESTO REFERENCIAL

Se realizarán pruebas con diferentes productos colocados de distintas maneras dentro de una canasta colocada encima de una balanza.

Una de las pruebas a realizarse y que pueden presentar mayores inconvenientes en su identificación, es la colocación de productos con materiales metálicos dentro de la canasta.

Posteriormente se establecerá el presupuesto referencial de los elementos y del *software* elaborado para la implementación comercial y de un prototipo, elaborado en este proyecto.

4.1 PRUEBAS DEL PROTOTIPO

Las pruebas del prototipo a realizarse consistirán en:

- Realizar la identificación de los productos por medio de las etiquetas al estar estos en una canasta con otros productos y colocados de distintas maneras.
- Tomar el tiempo en que se tarda identificar todos los productos que están en la canasta.
- En los productos que tomen mayor tiempo en ser identificados se realizarán cambios, como colocar de distinta manera los productos o colocar aislante entre la etiqueta y el producto de difícil lectura para obtener un menor tiempo de identificación.
- Al final de cada prueba realizada se dará una conclusión y solución al problema.

En cada prueba realizada se incluirá una figura mostrando cómo se colocaron los productos y una tabla que contendrá el nombre de los productos utilizados para la prueba y el tiempo promedio en realizar la lectura de todos los productos.

4.1.1 PRUEBA CON PRODUCTOS DE DIFÍCIL LECTURA

En esta prueba con productos de difícil lectura como los productos metálicos, líquidos y en algunas ocasiones los productos químicos, como se muestra en la figura 4.1, se realiza la lectura para observar el comportamiento que tiene el lector con las etiquetas adheridas a estos productos.

Figura 4.1: Prueba con productos de difícil lectura (productos metálicos sin aislante)

En la tabla 4.1 se detallan los productos utilizados para esta prueba y el tiempo en ser identificados todos los productos por el lector.

Productos	Tiempo en detectar todos los productos (seg)
Pony M	15
Sapoli	
Tesalia	
Yogurt N	
Sped	

Tabla 4.1: Prueba con productos de difícil lectura (productos metálicos sin aislante)

En la figura 4.2 se muestran los productos identificados por el lector y mostrados en pantalla por medio del programa Java; estos productos muestran el código de la etiqueta, el nombre, el precio y el peso del producto.

Figura 4.2: Lista de productos detectados en prueba con productos de difícil lectura (productos metálicos sin aislante)

4.1.1.1 Conclusión

Con estos productos se observa que mayores inconvenientes se presentan en la identificación de los productos metálicos, ya que para que exista un reconocimiento de la etiqueta por parte del lector se debe tener una mayor cercanía entre el lector y la etiqueta. Esto ocurre ya que las etiquetas poseen un contacto total con el metal y éste refleja las señales de radiofrecuencia y es una fuente de interferencia RFID por lo que pueden limitar considerablemente el rango de lectura.

Los productos líquidos también requieren una cercanía entre el lector y la etiqueta para poder ser identificados, esto debido a que las señales de radiofrecuencia son

absorbidas por el producto y limitan así el rango o impiden totalmente las operaciones de lectura. Pero en este caso no es necesario acercar mucho el lector a estos productos, porque la superficie de la etiqueta asignada a este producto no está totalmente adherida como las etiquetas de los productos metálicos.

En este caso el producto químico no es de difícil lectura ya que la composición de la que está elaborado no es absorbente.

4.1.1.2 Solución al problema

Para poder solucionar este problema de tener que acercar el lector mas a los productos metálicos, para que puedan ser detectados por el lector, se colocará un aislante entre la superficie metálica del producto y la etiqueta. De esta forma las ondas que envía el lector no serán reflejadas por la superficie metálica y podrán llegar a la etiqueta para su detección. En este caso se utilizó como aislante una espuma flex como se indica en la figura 4.3.

Figura 4.3: Solución para detectar productos metálicos con un aislante

En la figura 4.4 se muestra a los productos de difícil lectura puestos un aislante entre la etiqueta y el producto.

Figura 4.4: Prueba con productos de difícil lectura (productos metálicos con aislante)

En la tabla 4.2 se observa la misma prueba que anteriormente se realizó, pero con un aislante en los productos metálicos entre la etiqueta y el producto. En esta prueba se observa la variación en el tiempo para detectar todos los productos.

Productos	**Tiempo en detectar todos los productos (seg)**
Pony M	13.36
Sapoli	
Tesalia	
Yogurt N	
Sped	

Tabla 4.2: Prueba con productos de difícil lectura (productos metálicos con aislante)

En la figura 4.5 se muestran los productos detectados por el lector que son mostrados en pantalla, en la prueba con productos de difícil lectura (productos metálicos con aislante).

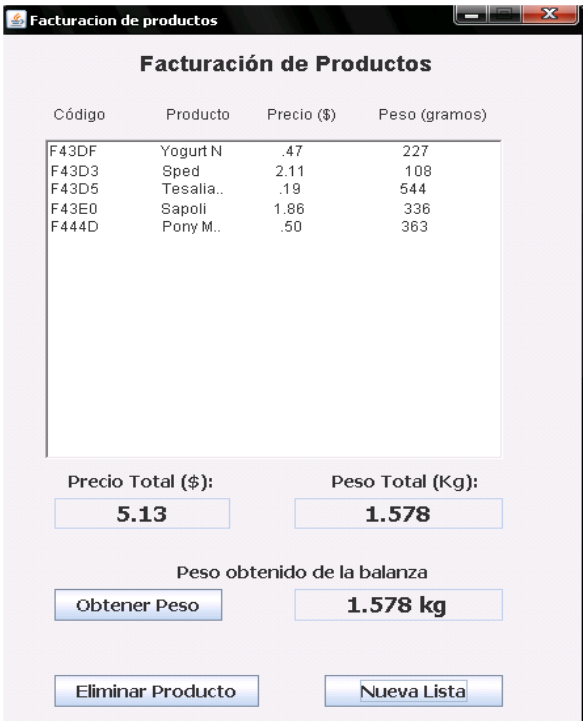


Figura 4.5: Lista de productos detectados en la prueba (productos metálicos con aislante)

4.1.2 PRUEBA CON PRODUCTOS DE FÁCIL Y DIFÍCIL LECTURA

Figura 4.6: Prueba con productos de fácil y difícil lectura

En esta prueba con productos de fácil y difícil lectura, como se muestra en la figura 4.6, se realiza la lectura para observar el comportamiento que tiene el lector con las etiquetas adheridas a estos productos.

En la tabla 4.3 se detallan los productos utilizados para esta prueba y el tiempo que tardan en ser leídos todos los productos por el lector.

Productos	Tiempo en detectar todos los productos (seg)
Fideo Vic	11.5
Cerveza	
Tesalia	
Yogurt N	
Vileda	
Sped	

Tabla 4.3: Prueba con productos de fácil y difícil lectura

En la figura 4.7 se muestran los productos detectados por el lector que son mostrados en pantalla.

Figura 4.7: Lista de productos detectados en la prueba con productos de fácil y difícil lectura

4.1.2.1 Conclusión

La lectura se realizó en menor tiempo que en la prueba con productos de difícil lectura (productos metálicos sin aislante), ya que también hay productos de fácil lectura para realizar esta prueba; sin embargo para poder detectar los productos metálicos y líquidos se debe acercar más el lector a estos productos, al igual que en la pruebas con productos de difícil lectura, debido a que toda la superficie de las etiquetas están adheridas a estos productos.

4.1.2.2 Solución al problema

Para poder solucionar este problema de tener que acercar el lector mas a los productos metálicos y líquidos para que puedan ser detectados por el lector, ya no se colocará un aislante entre la etiqueta y el producto de difícil lectura; en este caso se colocará la etiqueta de tal forma que no esté totalmente adherida a la superficie del producto, tal como se muestra en la figura 4.8.

Figura 4.8: Solución para detectar productos metálicos y líquidos no colocando toda la superficie de la etiqueta al producto

En la figura 4.9 se muestran los productos de fácil y difícil lectura, no colocando toda la superficie de la etiqueta a los productos de difícil lectura.

Figura 4.9: Prueba con productos de fácil y difícil lectura, no colocando toda la superficie de la etiqueta al producto de difícil lectura

Productos	**Tiempo en detectar todos los productos (seg)**
Fideo Vic	9.84
Cerveza	
Tesalia	
Yogurt N	
Vileda	
Sped	

Tabla 4.4: Prueba con productos de fácil y difícil lectura, no colocando toda la superficie de la etiqueta al producto de difícil lectura

En la tabla 4.4 se observa la misma prueba que anteriormente se realizó, pero en este caso toda la superficie de la etiqueta no estará adherida a los productos metálicos y líquidos para observar el tiempo en detectar todos los productos.

En la figura 4.10 se muestran los productos detectados por el lector que son mostrados en pantalla.

Figura 4.10: Lista de productos detectados en la prueba con productos de fácil y difícil lectura, sin colocar toda la superficie de la etiqueta al de difícil lectura

4.1.3 PRUEBA CON LAS ETIQUETAS JUNTAS ENTRE SÍ Y ADHERIDAS A LOS PRODUCTOS

En esta prueba con las etiquetas juntas entre sí y adheridas a los productos como se muestra en la figura 4.11, se realiza la lectura para observar el comportamiento que tiene el lector con las etiquetas juntas entre sí y adheridas a los productos.

Figura 4.11: Prueba con las etiquetas de los productos juntas entre sí

En la tabla 4.5 se detallan los productos utilizados para esta prueba y el tiempo en ser leídos todos los productos por el lector.

Productos	**Tiempo en detectar todos los productos (seg)**
Té Petrona	No lee todos los productos
Vileda	

Tabla 4.5: Prueba con las etiquetas juntas entre sí y adheridas a los productos

En la figura 4.12 se muestran los productos detectados por el lector que son mostrados en pantalla.

4.1.3.1 Conclusión

En esta prueba no se detectó todos los productos porque las etiquetas de los productos están juntas entre sí y el lector sólo detecta una de ellas, porque existe una colisión de ondas que se da cuando más de una etiqueta refleja la señal hacia el lector al mismo tiempo.

4.1.3.2 Solución al problema

Para poder solucionar este problema no se colocará las etiquetas de los productos juntas para que la lectura se realice con normalidad.

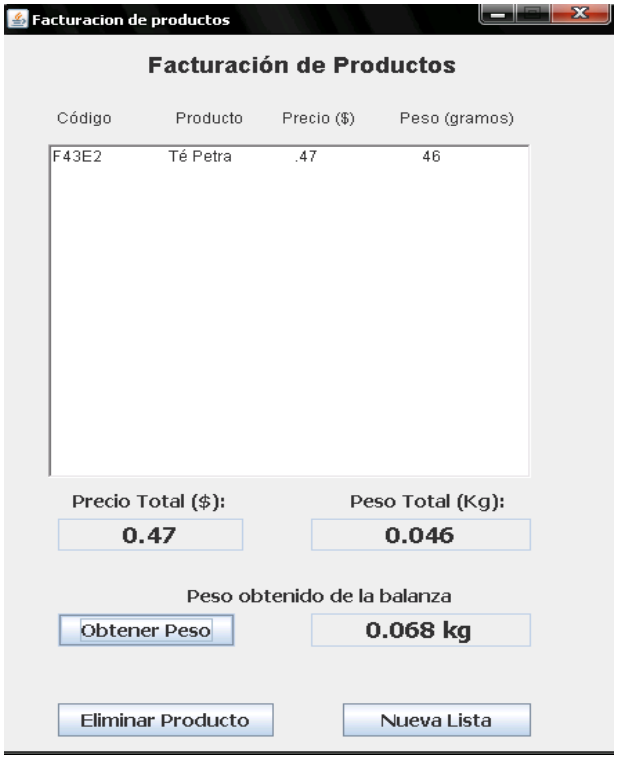


Figura 4.12: Lista de productos detectados en la prueba con las etiquetas juntas entre sí y adheridas a los productos

Figura 4.13: Prueba con las etiquetas separadas entre sí y adheridas a los productos

En la tabla 4.6 se observa la misma prueba que se realizó con las etiquetas juntas entre sí y adheridas a los productos, pero colocando las etiquetas separadas entre sí.

Productos	Tiempo en detectar todos los productos (seg)
Té Petrona	3.09
Vileda	

Tabla 4.6: Prueba con las etiquetas separadas entre sí y adheridas a los productos

En la figura 4.14 se muestran los productos detectados por el lector en esta prueba que son mostrados en pantalla.

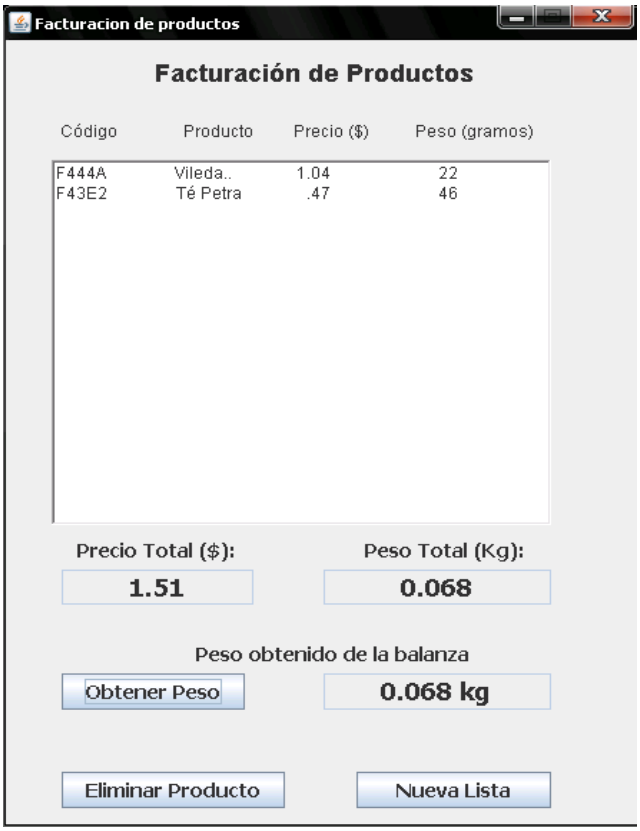


Figura 4.14: Lista de productos detectados en la prueba con las etiquetas separadas entre sí y adheridas a los productos

4.1.4 PRUEBA CON PRODUCTOS FÁCIL LECTURA

En esta prueba con productos de fácil lectura, como se muestra en la figura 4.15, se realiza la lectura para observar el comportamiento que tiene el lector con las etiquetas adheridas a estos productos.

Figura 4.15: Prueba con productos de fácil lectura

En la tabla 4.7 se detallan los productos utilizados para esta prueba y el tiempo en ser leídos todos los productos por el lector.

Productos	**Tiempo en detectar todos los productos (seg)**
Fideo Vic	9.65
Ruffles	
Té Petrona	
Vileda	
Sped	

Tabla 4.7: Prueba con productos de fácil lectura

En la figura 4.16 se muestran los productos detectados por el lector que son mostrados en pantalla.

4.1.4.1 Conclusión

La lectura se realizó sin problemas porque ningún producto de estos presenta inconvenientes para que el lector pueda detectarlos y porque las etiquetas no están juntas entre si.

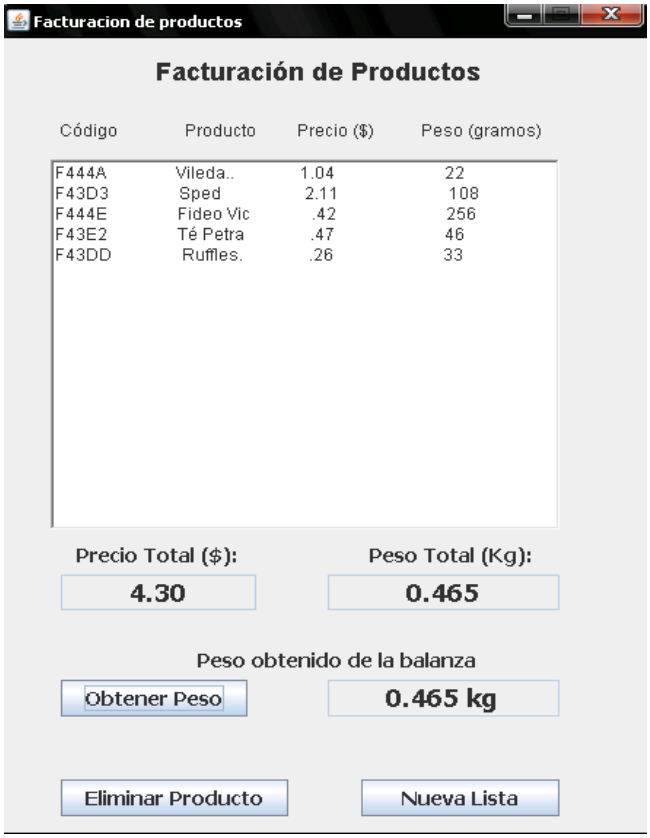


Figura 4.16: Lista de productos detectados en la prueba con productos de fácil lectura

4.1.5 PRUEBA CON PRODUCTOS COLOCADOS UNO ENCIMA DE OTRO

En esta prueba con los productos colocados uno encima de otro, se realiza la lectura para observar el comportamiento que tiene el lector con las etiquetas adheridas a estos productos.

En la figura 4.17 se muestran los productos colocados uno encima de otro.

Figura 4.17: Prueba con productos colocados uno encima de otro

En la tabla 4.8 se detallan los productos utilizados para esta prueba y el tiempo que tardan en ser leídos todos los productos por el lector.

Productos	Tiempo en detectar todos los productos (seg)
Vileda	7.67
Fideo Vic	
Sped	
Té Petrona	

Tabla 4.8: Prueba con productos colocados uno encima de otro

En la figura 4.18 se muestran los productos detectados por el lector que son mostrados en pantalla.

Figura 4.18: Lista de productos detectados en la prueba con productos colocados uno encima de otro

4.1.5.1 Conclusión

La lectura se realizó sin problemas porque ningún producto de estos presenta inconvenientes para que el lector pueda detectarlos, el único inconveniente que podría presentarse es que las etiquetas estén juntas entre sí, pero para está prueba no se colocó juntas las etiquetas.

4.1.6 PRUEBA CON UN MATERIAL METÁLICO EN LA PARTE SUPERIOR DE LOS PRODUCTOS

En esta prueba con un material metálico en la parte superior de los productos, como se muestra en la figura 4.19, se realiza la lectura para observar el comportamiento que tiene el lector con las etiquetas adheridas a estos productos.

Figura 4.19: Prueba con un material metálico en la parte superior de los productos

En la figura 4.20 se muestran los productos detectados por el lector que son mostrados en pantalla.

En la tabla 4.9 se detallan los productos utilizados para esta prueba y el tiempo en ser leídos todos los productos por el lector.

Figura 4.20: Lista de productos detectados en la prueba con un material metálico en la parte superior de los productos

Productos	Tiempo en detectar todos los productos (seg)
Vileda	9.43
Fideo Vic	
Sped	
Té Petrona	

Tabla 4.9: Prueba con un material metálico en la parte superior de los productos

4.1.6.1 Conclusión

En esta prueba no se presentaron problemas en la lectura, ya que se evitó que toda la superficie de las etiquetas de los productos estén juntas al material metálico, para que no provoquen que las ondas enviadas por el lector hacia la

etiqueta, sean reflejadas por este material metálico y así no se activen, por lo que no serían detectadas por el lector.

4.1.7 PRUEBA CON UN MATERIAL METÁLICO EN LA PARTE INFERIOR DE LOS PRODUCTOS

En esta prueba con un material metálico en la parte inferior de los productos, como se muestra en la figura 4.21, se realiza la lectura para observar el comportamiento que tiene el lector con las etiquetas adheridos a estos productos.

Figura 4.21: Prueba con un material metálico en la parte inferior de los productos

En la tabla 4.10 se detallan los productos utilizados para esta prueba y el tiempo que tardan en ser leídos todos los productos por el lector.

Productos	Tiempo en detectar todos los productos (seg)
Vileda	8.78
Fideo Vic	
Sped	
Té Petrona	

Tabla 4.10: Prueba con un material metálico en la parte inferior de los productos

En la figura 4.22 se muestran los productos detectados por el lector que son mostrados en pantalla.

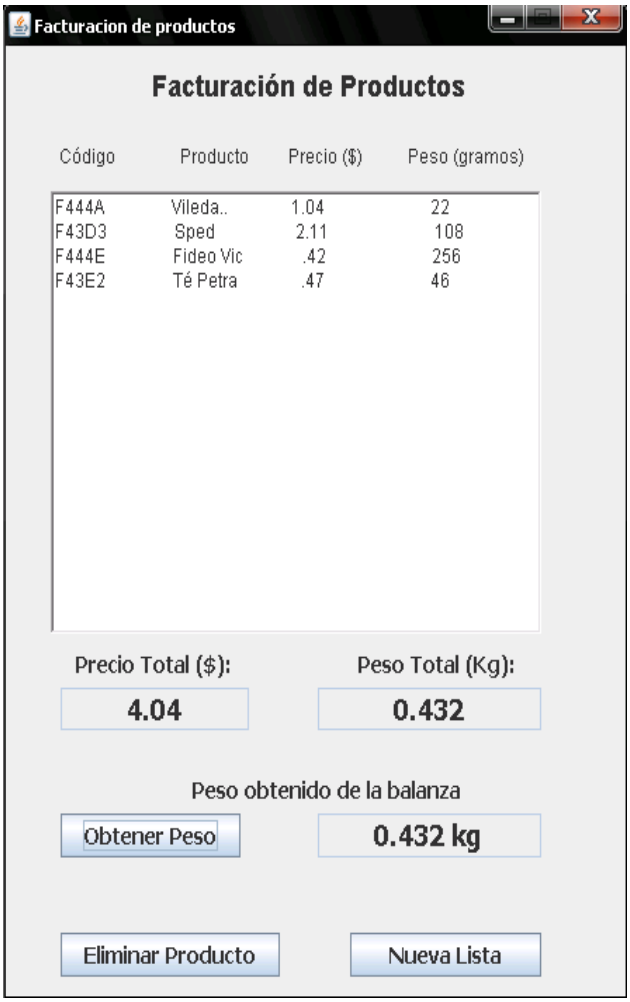


Figura 4.22: Lista de productos detectados en la prueba con un material metálico en la parte inferior de los productos

4.1.7.1 Conclusión

En esta prueba no se presentaron problemas en la lectura por el lector, porque tal como se hizo en la prueba anterior, se evitó que toda la superficie de las etiquetas de los productos esté junta al material metálico. De esta manera las ondas enviadas por el lector hacia la etiqueta serán activadas y no existirán ondas reflejadas por este material metálico, y por lo que serán detectadas por el lector.

4.2 PRESUPUESTO PARA LA IMPLEMENTACIÓN DEL PROTOTIPO

En la tabla 4.11 se exponen los precios de cada uno de los elementos antes mencionados para la implementación del prototipo.

Elementos	Precio ($)
Lector RFID	76,58
10 Etiquetas RFID (1,43 c/u)	14,30
Gastos de exportación	38,21
Gastos de envío	120
Balanza de mesa	275
Desarrollo del *software*	400
PC	700
Fuente *switching*	10
Canasta	4
Horas de trabajo para el desarrollo de la implementación del prototipo	600
Cables: DB9-DB9 De RS232 a USB	 6 24
Total	**2268.09**

Tabla 4.11: Tabla de precios para el prototipo

4.3 CARACTERÍSTICAS DE LOS EQUIPOS PARA LA IMPLEMENTACIÓN DE UN SISTEMA COMERCIAL

En este punto del capítulo se determinará el presupuesto económico para la implementación de un sistema comercial, para lo cual se establecerán las características de cada elemento en consideración.

Los equipos que se elegirán serán los que trabajen en la frecuencia UHF (300-3000 MHz), porque a esta frecuencia el rango típico de lectura es de 1 a 5 metros, distancia que se necesita para que el lector pueda detectar las etiquetas colocadas en los productos en el coche de compras de los supermercados.

En la tabla 4.12 se muestran características de la banda UHF de RFID.

	Características
Banda de frecuencia	300-3000 MHz
Frecuencia RFID	915 MHz (EEUU) 869 MHz (UE)
Sensibilidad a humedad	Alta
Sensibilidad a metales	Media
Interferencia de entorno	Baja
Rango de lectura máximo	100 m (utilizando antenas de grandes dimensiones)
Rango de lectura típico	1 a 5 m
Velocidad de transmisión de datos	1 a 160 Kbps
Tamaño antenas	Pequeño
Lectura direccional	Si
Precio etiqueta	Bajo
Disponibilidad de equipos	Media
Complejidad de los equipos	Media

Tabla 4.12: Características de la banda UHF de RFID

Los elementos para la implementación del prototipo serán los siguientes:

4.3.1 LECTOR RFID

El lector utilizado para la implementación del prototipo sería el Lector RFID de largo alcance UHF-EPC Gen 2 modelo MP9320.

En la figura 4.23 se muestra el lector y el lector con las antenas.

El lector MP9320 UHF-EPC es un producto de avanzada tecnología debido a:

- Su soporte multi-protocolo, su flexibilidad en la frecuencia para soportar diferentes regiones y su facilidad de programación.
- Está diseñado para leer cualquier etiqueta.
- Posee un amplio rango y velocidad de lectura y transmisión de datos.
- Especialmente indicado para la administración de activos y aplicaciones logísticas que requieran la lectura simultánea de un gran número de etiquetas a grandes distancias.
- Posee la capacidad de conectar hasta cuatro antenas, permitiendo su uso en estaciones de carga, cintas transportadoras, *tracking* de contenedores y *pallets*, así como también administración de inventarios.
- Sensibilidad y poder de lectura ajustable.
- *Software* de configuración e instalación disponible
- Gracias a su diseño expandible, soporta múltiples protocolo y etiquetas pudiendo actualizarse fácilmente para soportar nuevos modelos de etiquetas y protocolos estándares.

Figura 4.23: Lector MP9320 UHF-EPC con antenas [12]

4.3.1.1 Descripción

En la tabla 4.13 se muestran las características del lector.

Las características mas detalladas del lector se muestran en el anexo E.1.

Modelo	MP9320
Etiquetas soportados	EPC Class 0, EPC Class 0+, EPC Class 1, EPC Class 1 Gen 2, ISO 18000-6A, ISO 18000-6B, UCODE EPC 1.19, EM4022, EM4222, EM4223, IntellitagTM
Frecuencia de operación	902-928 MHz (100 KHz *steps*) 869.525 MHz 865-868 MHz (200 KHz *steps*)
RF Power	4W EIRP (FCC) 500 mW/2W ERP (ETSI)
Temperatura de operación	-20 a 70 C
Antenas	Hasta 4 antenas conectores SMA
Interfases	RS-232 (EIA/TIA-232F) RS-485 (EIA/TIA-485A) 10 Base-T Ethernet 10/100 MBPS
Gabinete Metálico	Aluminio
Dimensiones	127 mm x 178 mm x 241 mm
Peso	1.8 kg

Tabla 4.13: Características del lector MP9320 UHF-EPC [12]

Las características mas detalladas del lector se muestran en el anexo E.1.

4.3.2 ANTENA PARA EL LECTOR RFID

Para el lector MP9320 UHF-EPC se utilizará la antena que se muestra en la figura 4.24.

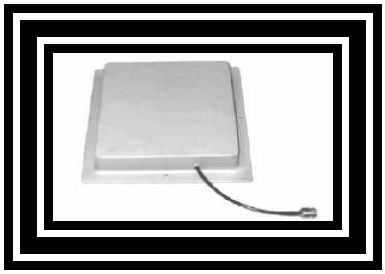


Figura 4.24: Antena para el lector MP9320 UHF-EPC [13]

4.3.2.1 Descripción

En la tabla 4.14 se muestran las características de la antena para el lector.

Modelo	UHF 27 x 18
Ganancia	6 dBi
Frecuencia entrada	860-930 MHz
VSWR	Típica menor que 1,5:1
Axial Ratio	Menor a 1dB
Potencia máxima	10 W
Polarización	Circular
Dimensiones	245 mm x 235 mm x 40 mm
Peso	470 g

Tabla 4.14: Características de la Antena para el lector MP9320 UHF-EPC [13]

4.3.3 ETIQUETAS RFID PASIVAS

La etiqueta RFID elegida para la implementación del prototipo es la etiqueta adhesiva de *AdActiv* que se muestra en la figura 4.25. Estas etiquetas están especialmente diseñadas para aplicaciones de logística y cadenas de suministro donde se requieren largas distancias de lectura. Las etiquetas adhesivas pueden personalizarse mediante impresión térmica. Gracias a su permanente capa adhesiva, se pueden fijar fácilmente sobre los productos.

Figura 4.25: Etiqueta MP9320 UHF-EPC con antenas [14]

4.3.3.1 Descripción

En la tabla 4.15 se muestran las características de la etiqueta RFID.

Modelo	UHF 27 x 18
Tamaño de Antena	23 x 14 mm
Tamaño Etiqueta Completa	27 x 18 mm
Tecnologías disponibles	NXP U-Code HSL, U-Code EPC Gen2, STM XRA00
Frecuencias	865 MHz – 915 MHz

Tabla 4.15: Características de la etiquetas RFID [14]

4.3.4 BALANZA DE SUELO

La balanza para *pallet*, como se muestra en la figura 4.26, es ideal si no desea montar o empotrar ninguna balanza en el suelo. Las rampas adjuntas le permiten pasar por encima de la balanza para *pallet* con carretillas o contenedores rodantes, o como en este caso coches de compras de supermercados. La pantalla es conectada a un cable de 3 m y tiene puerto RS-232 para la transmisión directa al PC.

Figura 4.26: Balanza de suelo [15]

4.3.4.1 Descripción

En la tabla 4.16 se muestran las características de la balanza.

Modelo	PCE-TP 1500
Rango de pesado	0 -1500 kg
Rango de taraje	En todo el rango de pesado
Tiempo de respuesta	< 4 s
Unidades de pesado	Kg
Sobrecarga máxima	200 %
Pantalla	LED de alto contraste
Plataforma útil	1250 x 1250 mm
Dimensiones de la plataforma	1500 x 1250 mm
Dimensiones de la rampa	1500 x 470 x 55 mm (cada rampa)
Temperatura ambiental	-10 ... +40 ºC
Alimentación	230 V / 50 Hz
Carcasa	pantalla de acero noble IP 65 / balanza según variante: acero lacado o acero noble
Peso	350 kg

Tabla 4.16: Características de la balanza de suelo PCE-TP 1500 [15]

Las características mas detalladas de la balanza se muestran en el anexo E.2.

4.3.5 CABLES

Los cables que se utilizarán será un DB9 hembra a DB9 hembra con conexión directa como se muestra en la figura 4.27 para conectar la balanza con el cable convertidor, y un cable convertidor de USB a RS232 como se muestra en la figura 4.28 para conectar el cable DB9 con el puerto USB de la computadora.

Figura 4.27: Cable DB9 (hembra a hembra)

Figura 4.28: Cable de RS232 a USB

4.3.6 SISTEMA DE ALIMENTACIÓN

La fuente de alimentación que será utilizado es Puls SLV20 como se indica en la figura 4.29.

Figura 4.29: Fuente de alimentación [16]

Esta fuente suministra corriente de sustitución sin necesidad de baterías ni de conexión de CA; puede suministrar 20 amperios durante 200 milisegundos o 1 amperio durante 4 segundos a 24 Vcc.

La fuente de alimentación tendrá una tensión de 22,5 V y una toma de corriente de 20 A. Con tomas de corriente de 100 mA, el dispositivo ofrece un tiempo de seguridad de 43 segundos. Es posible conectar varias unidades en paralelo para extender el tiempo de autonomía.

Las características mas detalladas de la fuente se muestran en el anexo E.3.

4.4 PRESUPUESTO REFERENCIAL

En la tabla 4.17 se exponen los precios de cada uno de los elementos antes mencionados para la implementación del prototipo comercial.

Elementos	**Precio ($)**
Lector RFID	2999
Antena para el lector RFID	263
1000 Etiquetas RFID	350
Balanza de suelo	2452
Fuente de alimentación eléctrica	95
PC	700
Desarrollo del *software*	450
Horas de trabajo para la implementación por cada caja	150
Cables: DB9-DB9 De RS232 a USB	 6 24
Total	**7489**

Tabla 4.17: Tabla de precios

Este total es el valor por cada una de las cajas de los supermercados.

Por lo que se puede observar la implementación comercial del prototipo no es económica, pero en un futuro se espera abaratar costos cuando la tecnología RFID sea más difundida.

Para abaratar los costos se puede utilizar computadores en red para que sólo uno de ellos contenga el *software* de la identificación.

Se puede observar la diferencia en el precio entre el prototipo comercial y el prototipo implementado en el proyecto, esto es debido a que para un prototipo comercial se necesita un *kit* RFID con características más exigentes a las del prototipo.

El prototipo implementado posee limitaciones con respecto al prototipo comercial como son la distancia de identificación, la balanza, entre otras más, esto debido al costo que representa elaborar un prototipo comercial.

CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES

La realización de este proyecto ha dejado diferentes conclusiones y recomendaciones:

5.1 CONCLUSIONES

- La tecnología RFID proporcionará información en tiempo real, esto ayuda de gran manera en aplicaciones de cadenas de suministros donde la producción estará regulada por la información proporcionada por esta tecnología.

- La implementación del prototipo ayudará a prevenir robos de productos por parte de los clientes. El cliente que ha hurtado un producto se acerca a la caja y éste es identificado descubriendo que dicho producto no está en el coche de compras por la diferencia de pesos que uno observa entre el peso total obtenido la identificación de todos los productos, incluido el robado, con el peso obtenido de la balanza.

- En un supermercado está tecnología no está limitada a proporcionar solo el costo total de la compra, se puede ampliar su aplicación para conocer si los productos están por terminarse para hacer un pedido, para conocer qué productos del supermercado están por caducar y realizar promociones de estos productos; también para hacer un control de inventario en tiempo real. Por esto varias empresas grandes utilizan este sistema como: *Wal-Mart, Metro, Carrefour, Gillete,Tesco, Procter&Gamble* y algunas de estas obligan a sus proveedores a poner en sus *pallets* y cajas, etiquetas RFID.

- La tecnología RFID tiene una variedad de aplicaciones debido a que puede trabajar con 3 tipos de frecuencia: LF (*Low frequency*), HF (*High frequency*) y UHF (*Ultra High Frequency*). Esto permite conocer, dependiendo de la aplicación, a qué frecuencia se necesitará tener tanto las etiquetas como los lectores.

- Para obtener resultados positivos en la implementación del prototipo, se requerirá pruebas con el mayor número posible de productos, para saber cómo colocar las etiquetas en sitios estratégicos en los productos.

- Para la realización de las pruebas se debió clasificar los productos en dos grupos (de fácil lectura y de difícil lectura) para saber cómo colocar las etiquetas en los productos y en qué posición colocar los productos en la canasta, para tener una rápida identificación de las etiquetas adheridas a los productos.

- Este proyecto tiene numerosas vías futuras de ampliación por las mejoras de la tecnología que se están dando, para que las aplicaciones sean más seguras, fáciles y económicas para la implementación.

- Se debe tener en cuenta que esta tecnología no es aún rentable, debido al costo de las etiquetas, puesto que las etiquetas pueden resultar más costosas que el propio producto donde será adherida la etiqueta, por lo que se espera que las etiquetas bajen de precio y de tamaño cuando exista mayor demanda.

- En la realización de este proyecto se pudo apreciar la utilización de varias asignaturas dictadas en la carrera, como por ejemplo: Comunicación Digital, Comunicación Inalámbrica y Telemática, para poder entender la comunicación entre los dispositivos utilizados como el lector con el PC y el lector con las etiquetas. Se utilizaron los conocimientos adquiridos en Comunicaciones Inalámbricas en donde se estudió esta tecnología. Se hizo uso de Circuitos Eléctricos para conocer qué fuente será necesaria para la implementación y conexión de los equipos. Con la elaboración del proyecto se concluye que todas las materias aprendidas en la carrera son complementarias y utilizadas en la vida profesional.

- La tecnología RFID es más eficiente que otros métodos de registro de productos ya implementados en los supermercados, puesto que no

necesita una línea de vista para identificar a las etiquetas adheridas en los productos, puede leer varios productos simultáneamente y las etiquetas RFID pueden almacenar más información acerca del producto que solo un código.

- Con la tecnología RFID se puede tener una seguridad física adicional de los productos, utilizándola a la salida del local para identificar cuáles no han sido tomados en cuenta para la elaboración de la factura.

- Para la realización del *software* de este proyecto se puede emplear otros lenguajes de programación, prescindiendo de las ventajas que nos brinda JAVA como por ejemplo el lenguaje de programación*VisualBasic.net*.

5.2 RECOMENDACIONES

- Para mejorar la aplicación de la tecnología RFID se puede interactuar entre algunas tecnologías como: *Wi Fi* o *Wi Max* en el interior de establecimientos o con la tecnología GPRS en el exterior de abastecimientos. Esta interacción de tecnologías se conoce como RTLS (*Real Time Location System*).

- Para evitar la interferencia mutua que existe entre los lectores cuando están colocados cerca, es recomendable utilizar TDMA (acceso múltiple por división de tiempo); es decir, que cada uno de los lectores utiliza un tiempo para el intercambio de información o de FHSS (espectro ensanchado por salto de frecuencia) que hará que los dispositivos deban saltar de canal en forma aleatoria y así evitar la interferencia.

- Para evitar la interferencia de la señal del lector con los datos de las etiquetas, se recomienda utilizar un aislante entre el producto y la etiqueta, que ayudará que la potencia de la señal no se disminuya o se pierda.

- Para la fuente de alimentación del lector se recomienda que sea lo más exacta en la salida del voltaje, que no varíe mucho con la carga utilizada, porque de este voltaje dependerá el rango de identificación.

- Para comprobar el funcionamiento del lector sin tener un lenguaje de programación desarrollado se puede utilizar el programa *HyperTerminal* para saber si la etiqueta está comunicando al lector sus datos y si a su vez el lector envía la información de forma serial a la computadora.

- Si no es necesario que el programa sea colocado en varias plataformas del Sistema Operativo, que tenga portabilidad, se puede utilizar otro *software* pero se prescindirá de todas las ventajas que tiene Java.

- Se recomienda implementar este prototipo para proporcionar una disminución en los gastos y en el tiempo utilizado por las empresas para la facturación de productos. En un supermercado que implemente este prototipo no será necesario hacer grandes colas para poder pagar lo adquirido, dando satisfacción al cliente.

BIBLIOGRAFÍA

[1] Estudio, Diseño y Simulación de un Sistema de RFID basado en EPC http://upcommons.upc.edu/pfc/bitstream/2099.1/3552/2/40883-2.pdf

[2] INFORME ELABORADO POR EL CENTRO DE DIFUSIÓN DE TECNOLOGÍAS (CEDITEC), Tecnología RFID: Aplicaciones en el ámbito de la Salud www.ceditec.etsit.upm.es/dmdocuments/CITIC%20RFID%20Salud.pdf

[3] N- ECONOMÍA, Tecnologías RFID: Aplicaciones www.n-economia.com/informes_documentos/pdf/sintesis_documentos/SINTESIS_NE_09-2008.PDF

[4] IT&C-MIN2006, RFID: Introducción y Aplicaciones http://itcmin2006.li2.uchile.cl/presentaciones/ITC-MIN2006_Aplicaciones_RFID.pdf

[5] ESTUDIOS DE I+D+I, Sistema de agentes portables incrustados para entornos naturales seguros (SAPIENS) www.imsersomayores.csic.es/documentos/documentos/imserso-estudiosidi-43.pdf

[6] UNIVERSIDAD DE VALENCIA, Estudio de la identificación por radiofrecuencia (RFID) http://pc23te.dte.uma.es/Recursos/RFID/RFID_Memoria.pdf

[7] FKI LOGISTEX, Identificación por frecuencia de radio (RFID) para el mundo real www.dl.com.co//local/FKI_Logistex_RFID_White_Paper_Spanish.pdf

[8] KIOSKEA, Cables y conectores DB9
http://es.kioskea.net/elec/connecteur-prise-db9.php3

[9] JAVA DESDE CERO
htpp://mmc.igeofcu.unam.mx/cursos/mcst-2007-II/Java/Java%20desde%20Cero.pdf

[10] DEITEL H. M.; P. J. DEITEL, Como Programar en JAVA, Primera edición, Prentice Hall Hispanoamerica S. A., México, 1998

[11] JOYANES AGUILAR Luis; FERNÁNDEZ AZUELA Matilde, Java 2 Manual de Programación, Primera edición, McGraw-Hill, España, 2001

[12] NODOS, Lectores RFID largo alcance UHF-EPC Gen2
www.nodos.com.ar/Productos_Profile.asp?Producto=MP9320&cMarca

[13] MISAKO, Mejora la gestión de stocks e incrementa las ventas gracias a la tecnología RFID "RFID magazine"
www.rfid-magazine.com/_images/451/012_Misako.pdf

[14] ADACTIV, Tarjetas ISO, Tags y Transponders
www.adactiv.es/PDFs/Catalogo%20Transponders%20y%20Tags%20RFID.pdf

[15] PC GROUP, Balanza para Palet PCE-TP1500
www.balanzas-basculas-pce.com/datos-tecnicos/balanza-palets-pce-tp1500.htm

[16] OLFER, PULS SLV20.200
www.pulspower.com/index.php?reqNav=prodInfo

[17] WIKIPEDIA, La enciclopedia libre
http://es.wikipedia.org

[18] KIMALDI, Lectores y tags 125 khz
http://www.kimaldi.com/productos/sistemas_rfid/lectores_rfid_y_tags_125_khz

[19] ZHEJIANG LIANLONG ELECTRON & ELECTRIC APPLIANCES CO. LTD, Llas 500 plug-in Switching Power Adapter with Six Detachable Plugs
http://www.lianlong.com.cn/cw/gsol/pex/en/cwdesign1//jsp/main.jsp?supplier_id=2008800967609&section=ProductDetail&product_id=1000379193&prod_idx=0&rand=1221593568170

[20] RFID MAGAZINE, Noticias y Soluciones
http://rfid-magazine.com
LAS FUNCIONES DE LOS LECTORES RFID
www.asersa.com/asersa/Articulos/Articulo38.pdf

[21] EPC&RFID, Estudio de etiquetado con productos líquidos
www.epcglobalsp.org/areas_generales/publicaciones/estudiorfidliquidos.pdf

[22] Tecnología y aplicaciones de RFID
http://mami.uclm.es/rhervas/articulos/Ramon%20Hervas%20-%20RFID%20-%20Curso%20CCMM05.pdf

[23] REPLY LOGISTICS, RFID: Radio Frequency Identification
www.salle.url.edu/ctraining/icil/RFID%20Reply.pdf

[24] RFID MAGAZINE, Tecnología RFID: Introducción
www.mas-rfid-solutions.com/docs/RFID_introduccion.pdf

[25] L I B E R A W H I T E P A P E R S E R I E S, RFID: Tecnología, aplicaciones y perspectivas
www.libera.net/website/tecnologia/whitepapers/Whitepaper%20WP-RFID-001%20Sept%202006.pdf

[26] JOURNAL MEGAZINE, RFID
http://www.rfidjournal.com

[27] MIGUEL SÁNCHEZ LÓPEZ y VICTOR ALONSO BARBERÁN, El lenguaje de programación JAVATM
www.redes.upv.es/redes/images/capitulo7.pdf

[28] EPCGLOBAL, Interferencia en lectores RFID: *Soluciones prácticas a problemas complejos*
http://www.epcglobalsp.org/tech/Readers/FINAL_Reader_Interference_SPA.pdf

[29] PRIMERA EMPRESA ARGENTINA DEDICADA A SOLUCIONES CON RFID
http://www.pratea.com.ar/rfid.htm

[30] NETBEANS IDE 6.1, Download NetBean IDE
www.netbeans.org

[31] RFID HANDBOOK, Radio-Frequency-Identification
http: //www.rfid-handbook.com

GLOSARIO

Absorción: Cuando las ondas electromagnéticas atraviesan algún material, generalmente se debilitan o atenúan. La cantidad de potencia perdida va a depender de su frecuencia y, por supuesto, del material.

Algoritmo: Son las acciones y el orden en que esas acciones deben ejecutarse.

Antena: Elemento conductivo con capacidad para radiar.

Auto-ID Labs: Laboratorio de investigación sin ánimo de lucro, con sede en el Instituto de Tecnología de Massachussets (MIT). Investiga el desarrollo del EPC y las tecnologías relacionadas.

***Broadcast*:** En castellano difusión, es un modo de transmisión de información donde un nodo emisor envía información a una multitud de nodos receptores de manera simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo.

***Bytecodes*:** Son un conjunto de instrucciones muy parecidas al código de máquina, pero que no son específicas para algún procesador.

***Buffer*:** El área donde se almacenará de forma temporal en la memoria de la computadora

Clases: Las clases son lo más simple de Java. Todo en Java forma parte de una clase, es una clase o describe cómo funciona una clase. El conocimiento de las clases es fundamental para poder entender los programas Java.

Encriptación (*encryption*): Método para enmascarar el contenido de la información, para evitar que se pueda interceptar y visualizar la información que viaja de la etiqueta al lector. Sólo es posible leerlo si se conoce el método.

Entorno (Java): El entorno básico del JDK de Java que proporciona Sun está formado por herramientas en modo texto.

Espectro electromagnético: Las ondas electromagnéticas abarcan un amplio rango de frecuencias (y correspondientemente, de longitudes de onda). Este rango de frecuencias y longitudes de onda es denominado espectro electromagnético. La parte del espectro más familiar a los seres humanos es probablemente la luz.

Interacción: Se refiere a una acción recíproca entre dos o más objetos con una o más propiedades homólogas.

Interfaz de Usuario: La interfaz de usuario es la parte del programa que permite a éste interactuar con el usuario. La interfaz de usuario es el aspecto más importante de cualquier aplicación. Una aplicación sin un interfaz fácil, impide que los usuarios saquen el máximo rendimiento del programa.

ISO (*International Organization for Standardization*): Institución de Estandarización a nivel mundial.

Línea de Vista: Línea de vista se refiere a un camino (*path*) limpio, sin obstrucciones, entre las antenas transmisoras y receptoras.

Multiplataforma: Es la capacidad del programa de trasladarse con facilidad de un sistema computacional a otro.

Microchip: Es un pequeñísimo circuito que, gracias a su sofisticado diseño, ha logrado reducirse al tamaño de un grano de arroz (11,2 mm de largo por 1,2 mm de circunferencia).

Objeto: Se define como el espacio de memoria que se utilizará para guardar información que va a ser utilizada en el programa.

Reflexión: La reflexión de la luz ocurre cuando las ondas electromagnéticas se topan con una superficie que no absorbe la energía radiante. La onda, llamada rayo incidente se refleja produciendo un haz de luz, denominado rayo reflejado.

RS-232 : (también conocido como *Electronic Industries Alliance* RS-232C) es una interfaz que designa una norma para el intercambio serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (*Data Communication Equipment*, Equipo de Comunicación de datos), aunque existen otras situaciones en las que también se utiliza la interfaz RS-232.

Tara: Esta función de tara nos permite pesar cualquier objeto restándole un envase, fundas o cartones.

Operadores: Los operadores de Java son muy parecidos en estilo y funcionamiento a los de C. En la siguiente tabla aparecen los operadores que se utilizan en Java, por orden de precedencia.

Puerto Serial: Los puertos seriales (también llamados RS-232, por el nombre del estándar al que hacen referencia) fueron las primeras interfaces que permitieron que los equipos intercambien información con el "mundo exterior". El término serial se refiere a los datos enviados mediante un solo hilo: los bits se envían uno detrás del otro.

Radiofrecuencia: También denominado espectro de radiofrecuencia o RF, se aplica a la porción menos energética del espectro electromagnético, situada entre unos 3 Hz y unos 300 GHz. Las ondas electromagnéticas de esta región del espectro se pueden transmitir aplicando la corriente alterna originada en un generador a una antena.

ANEXO A

A.1 Características de los lectores de proximidad ProxID

Lectores de proximidad ProxID

GP20 – GP30

Los lectores ProxID funcionan desde 5 a 12,5 voltios con una distancia de lectura alta incluso a 5V. Haciéndolos ideales para una gran variedad de aplicaciones, particularmente en sistemas de control de accesos. La característica principal es que el lector básico puede ser configurado para las salidas más usuales de formato interface, incluyendo Wiegand, banda magnética, Clock/Data y salida RS-232 serial ASCII, haciéndolo muy fácil de aplicar en instalaciones ya existentes.

Características:

- Bajo coste
- Alto rango de lectura
- Dimensiones reducidas
- Encapsulado para usos externos
- Interface externo programable
- Resistente al agua

Modelos disponibles:

. GP20 – lector para hasta 20 cm de lectura, para instalación sobre pared.

. GP30 – lector para hasta 30 cm de lectura, para instalación sobre pared.

. Otros lectores de mayor alcance: - GP60 – hasta 60cm

Características técnicas:

Voltaje:	5-12.5V
Interface:	Wiegand, Magstripe ABA TK2, Clock/Data ó serial ASCII (RS-232)
Frecuencia:	125 KHz
Formato:	64 bits, código Manchester
Transponder:	sólo lectura
Dimensiones:	GP20 – 78x43x15mm – GP30 – 104x63x20mm

A.2 Manual del Lector ProxID GP20

Proximity Reader GP20 (5~13.5 Volts Version)

Power Requirements	5~13.5 Volts regulated DC @ 65 mA typical with a 12V supply. A linear regulator is needed.
Output Interface	Wiegand, Magstripe, 9.6K Baud Serial ASCII (RS232)
Maximum Read Range	20cm @ 13.5 VDC and 13cm @ 5V in ideal conditions
Frequency	125KHz standard
Dimensions	7.8 x 4.3 x 1.5cm
Temperature Range	-10 to 60 Deg C

Output Assignment

Red	Power + VDC
Black	Ground
White	Magstripe clock & Wiegand1, with internal 4K7 pull up
Green	RS232 data, Magstripe data & Wiegand0, with internal 4K7 pull up (pull up only for Wiegand and Magstripe)
Orange	Card Present output with internal 4K7 pull up
Yellow	Program Input
Blue	No Connection
Brown	No Connection

Output Format

he output format can be customer programmed. The available formats are Wiegand Magstripe and Serial ASCII (RS232)

Wiegand (26 bits)		**Magstripe (ABA TK2)**		**Serial ASCII (RS232)**	
Red	Power +V	Red	Power +V	Red	Power +V
Black	Ground	Black	Ground	Black	Ground
White	Data1	Green	Data	Green	TX Data
Green	Data0	White	Clock Strobe)	Yellow	No Connection
Yellow	Connect to White	Orange	Card Present	White	No Connection
Orange	No Connection	Yellow	Connect to Orange	Orange	No Connection

Data Structure

Serial ASCII (RS232): Baud 9600, No Parity, 8 data bits,1 stop bit

STX (02 HEX)	DATA (10 HEX CHARACTERS)	CR	LF	ETX (03 HEX)

Magstripe Emulation:(ABA Track 2)

Speed : Simulated to 40 IPS (Inch per Second)

10 LEADING ZEROS	SS	DATA (14 DIGITS)	ES	LRC	10 TRAILING ZEROS

Wiegand : 26 bits

MANUAL NO : GM950039

A.3 Características de las tarjetas de proximidad 125 Khz

Tarjeta prox. 125 Khz. sólo lectura modelo EM H4102

Tarjeta plástica PVC laminada por ambas caras. Tamaño ISO estándar: 85,7 x 54 mm. Grosor inferior a 0,9 mm. Chip sólo lectura EM4102.

Características chip:

- Transferencia de datos: sin contacto.
- Frecuencia: 125 Khz.
- Velocidad transferencia lectura: 20 us.
- Velocidad transferencia escritura: sólo lectura.
- Tiempo transacción: aprox. 150 ms.
- Capacidad memoria total: 64 bytes x 8 bit EEPROM.

Tarjeta prox. 125 Khz. Lectura y escritura modelo Temic E5551A

Tarjeta plástica PVC laminada por ambas caras. Tamaño ISO estándar: 85,7 x 54 mm. Grosor inferior a 0,9 mm. Chip lectura/escritura Temic E5551A.

Características chip:

- Transferencia de datos: sin contacto.
- Frecuencia: 125 Khz.
- Velocidad transferencia lectura: Configurable (8 pasos; de RF/8 a RF 128)
- Velocidad transferencia escritura: ~ 2Kbits.
- Distancia de lectura: típica 100 mm (depende del lector)

- Integridad de los datos y tratamiento de varias tarjetas en el mismo campo: OTP usando lock bits, modo password; no hay anti-colisión.
- Capacidad memoria total: 264 bits (8 bloques de 33 bits)
- Memoria de sistema: 32 bits (configuración de la modulación / codificación, bit rate...)
- Memoria del usuario: 224 bits (incluyendo 32 bits de password)
- Vida del chip: >100.000 ciclos de programación

Características de seguridad:

Protección de la memoria mediante password o bit de bloqueo.
OTP usando bits de bloqueo, modo password, no anti-colisión.
Se puede usar para ISO 1178 4 / 1178 5 función FDX-B

A.4 Características de la fuente *switching*

LLAS500 Plug-in Switching Power Adapter with Six Detachable Plugs

Brand Name:

- LIAN LONG

Country of Origin:

- China

Key Specifications/Special Features:

- Input voltage: 90 - 240V AC 50/60Hz
- Output voltage: 3/4.5/6/7.5/9/12V DC
- Maximum load: 0 - 500mA
- Input plug: AT, ET, ST, UK
- Output plug: With 6 detachable plugs
- Packing: Individual color giftbox

Primary Competitive Advantages:

- Experienced Technical Staff
- Prompt Delivery
- Service

Main Export Markets:

- Eastern Europe
- North America
- Mid East/Africa
- Central/South America
- Asia
- Western Europe
- Australasia

Payment Details:

- L/C, T/T

A.5 Características de la Báscula EQ-5/10

TORREY®

Ficha Técnica

LINEA:	BASCULAS PORCIONADORAS DE PRECISION
MODELO:	EQ-5/10, EQ-10/20

USOS PRINCIPALES

- Restaurantes.
- Bares.
- Pizzerías.
- Pastelerías.
- Cocinas Industriales
- Hoteles
- Hospitales
- etc.

CAPACIDADES Y DIMENSIONES

MODELO	EQ-5/10	EQ-10/20
Capacidad	5 kg / 10 lb / 160 oz	10 kg / 20 lb / 320 oz
División Mínima	1 g / 0.002 lb /0.05 oz	2 g / 0.005 lb / 0.1oz
Dimensiones de plato (cm)	20 x 27.9 cm (8 x 11")	20 x 27.9 cm (8 x 11")

BENEFICIOS

- Función de Tara Progresiva, es decir puede pesar los ingredientes de una receta uno por uno, todo en un mismo recipiente y al momento de agregarlos
- Funciona con batería recargable o con corriente eléctrica
- Pesa en kilogramos (kg), libras (lb) y onzas (oz).
- 5000 divisiones
- Construida en acero inoxidable.
- Fácil de usar.
- Puerto serial para conectar a computadora o impresora

LINEA:	BASCULAS PORCIONADORAS DE PRECISION
MODELO:	EQ-5/10, EQ-10/20

BENEFICIOS

EXCELENTE AUXILIAR EN TODO RESTAURANTE, COCINA O INDUSTRIA

Por sus características y precisión el uso de la báscula porcionadora EQ, es muy útil en cualquier restaurante, cocina o industria ya sea para la exacta preparación de recetas, el empaque de productos o para controlar algún proceso de producción.

HIGIENICA Y RESISTENTE:

Gabinete y plato construidos en acero inoxidable, lo cual brinda alta resistencia y mayor higiene en el contacto con los alimentos

ESPECIFICACIONES

	EQ-5/10	EQ-10/20
Capacidad	5kg/10lb/160oz	10kg/20lb/320oz
División Mínima	1g/0.002lb/0.05oz	2g/0.01lb/0.1oz
Display	Cuarzo líquido	
Back Light (Pantalla iluminada)	Incluido	
Corriente Eléctrica	110v/60hz (220v/50 hz Opcional)	
Adaptador a Corriente Eléctrica	Incluido	
Adaptador al encendedor del auto	Opcional	
Conector serial	RS-232	
Batería Recargable	Incluida con duración 200 horas	
Tara Máxima	2.5 kg/ 5 lb	5 kg/ 10 lb
Plato	20 x 27.9 cm (8 x 11")	
Temperatura de operación	-10 a 40 °C (14 a 104 °F)	
Temperatura de almacenaje	-20 a 50 °C (-4 a 122 °F)	
Peso Neto	8 kg/ 17.6 lb	8 kg / 17.6 lb
Peso con empaque	10 kg / 22 lb	10 kg / 22 lb

PESA EN KG, LB Y OZ

Las básculas EQ, cuentan con una función que les permiten pesar en diferentes unidades ya sea en Kg. Libras y onzas. Solo es necesario oprimir una tecla ahorrando tiempo del operador, que se traduce en mayor eficiencia en su trabajo.

PORTATIL

Las básculas de la línea EQ cuentan con una batería recargable que permite hasta 200 horas de trabajo continuo. De esta forma la báscula funciona aún sin estar conectada a la corriente eléctrica.

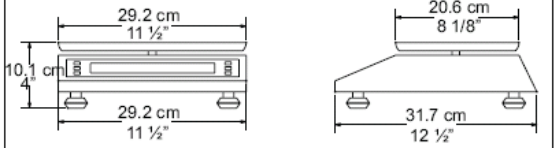
LECTURA FÁCIL Y RÁPIDA

La línea de básculas porcionadoras está equipada con una pantalla de alta resolución que despliega el peso que registra la báscula con números grandes y nítidos. Además cuenta con un accesorio que ilumina la pantalla (back light), muy útil en lugares con iluminación deficiente.

MAYOR CONTROL

Las básculas porcionadoras cuentan con tarjeta interfase ya incluida, lo que permite la comunicación de la báscula con una computadora o impresora, para un mayor control de sus procesos.

DIMENSIONES EXTERIORES



CERTIFICACIONES DE CALIDAD

NOM

DISTRIBUIDO POR

TODA LA INFORMACION ESTA SUJETA A CAMBIOS SIN PREVIO AVISO

ANEXO B

En este anexo se describe en detalle el programa realizado para el prototipo.

B.1 Cargar JDBC-ODBC

```
try{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");                                    }
catch(Exception e){ System.out.println("No se ha podido cargar el Driver JDBC-
ODBC"); }
```

B.2 Clase Main.java

```
/*
 * Main.java
*/
package id_productos;
public class Main {
public Main() {
    }
     public static void main(String[] args) {

  // TODO code application logic here

 ControladorPuerto ctrlTarjetas = new ControladorPuerto ("COM1",9600,8,1,0);
    }
}
```

B.3 Clase COMM_library.java.

```
// COMM_library.java

package id_productos;
```

```
import javax.comm.*;
import java.io.*;

public class COMM_library {
private SerialPort COM;
private String Datos;
public COMM_library (String nombre_puerto,int velocidad,int bdatos,int bstop, int paridad){

                try {
                    CommPortIdentifier comPort =
                            CommPortIdentifier.getPortIdentifier (nombre_puerto);
                    COM = (SerialPort) comPort.open("Dispositivo", 25);
                    COM.setSerialPortParams(velocidad,bdatos,bstop,paridad);
                    COM.enableReceiveThreshold(16);
                    COM.enableReceiveTimeout(500);

                } catch (Exception ex) {
                        ex.printStackTrace();
                }
        }

    public String Obtener_Dato_Dispositivo (String Peticion){
        try
        {
            InputStream clean = COM.getInputStream();
            Datos="";
            escribirDatos(Peticion);

            {
                try
                {
                    {
```

```
InputStream in = COM.getInputStream();
byte[] dt = new byte[COM.getInputBufferSize()];
int j = 0;
for (int i = 0; i < dt.length; i++)
{
    dt[i] = (byte) in.read();
    if (dt[i] == -1)
    {
        break;
    }
    j++;
}
char[] dtReceived = new char[j];
for (int i = 0; i <= j; i++)
{
    if (dt[i] == -1)
    {
        break;
    }
    else
    {
        dtReceived[i] = (char) dt[i];
    }
}
String dtStr = new String(dtReceived);
String resultado = "";
for (int i = 0; i < dtStr.length(); i++)
{
    char b = dtStr.charAt(i);
    resultado = resultado + b;
}
   Datos=resultado;
   return Datos;
```

```
                }
            }
            catch (Exception ex)
            {
                    ex.printStackTrace();
            }
        }
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
        COM.close();
    }
        COM.close();
        return "" ;
}
   private void escribirDatos(String dato) {
            try {
                    OutputStream output = COM.getOutputStream();
                    output.write(dato.getBytes());
                    output.flush();
            } catch (Exception ex) {
                    ex.printStackTrace();
            }
   }
private void serialEvent(SerialPortEvent event) {
            if (event.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
                    try {
                    Datos="";
                            InputStream in = COM.getInputStream();
                            byte[] dt = new byte[COM.getInputBufferSize()];
                            int j = 0;
                            for (int i = 0; i < dt.length; i++) {
```

```
                    dt[i] = (byte) in.read();
                    if (dt[i] == -1) {
                        break;
                    }
                    j++;
                }
                char[] dtReceived = new char[j];
                for (int i = 0; i <= j; i++) {
                    if (dt[i] == -1) {
                        break;
                    } else {
                        dtReceived[i] = (char) dt[i];
                    }
                }
                String dtStr = new String(dtReceived);
                char a = dtStr.charAt(0);
                String resultado = "";
                for (int i = 0; i < dtStr.length(); i++){
                char b = dtStr.charAt(i);
                resultado = resultado + b;
               }
                 Datos=resultado;
            } catch (Exception e) {
                    e.printStackTrace();
            }
        }
    }
}
```

B.4 Clase ControladorPuerto.java

```
/*
 * ControladorPuerto.java
*/
```

```
 package id_productos;
import javax.comm.*;
import java.io.*;
import java.awt.*;
import java.applet.Applet;

public class ControladorPuerto extends Applet implements
SerialPortEventListener {

	private SerialPort COM;
	public String Datos;
	private Productos_Frame pf = new Productos_Frame();
	private bdd_ODBC MiBDD = new bdd_ODBC("Productos","","");
	public ControladorPuerto(String nombre_puerto) {

		try {
			CommPortIdentifier comPort = CommPortIdentifier
					.getPortIdentifier(nombre_puerto);
			COM = (SerialPort) comPort.open("Control", 30000);
			COM.enableReceiveThreshold(16);
			COM.enableReceiveTimeout(500);

			// Notificar sí se recibe un dato en el pin de recepción

			COM.notifyOnDataAvailable(true);
			COM.addEventListener(this);

		} catch (Exception ex) {
			ex.printStackTrace();
		}
	}
```

```
public ControladorPuerto (String nombre_puerto,int velocidad,int
                                bdatos,int bstop, int paridad) {

        try {
                CommPortIdentifier comPort = CommPortIdentifier
                                        .getPortIdentifier(nombre_puerto);
                COM = (SerialPort) comPort.open("Control", 30000);
               COM.setSerialPortParams(velocidad,bdatos,bstop,paridad );
                COM.enableReceiveThreshold(16);
                COM.enableReceiveTimeout(500);

    // Notificar sí se recibe un dato en el pin de recepción

                COM.notifyOnDataAvailable(true);
                COM.addEventListener(this);

        } catch (Exception ex) {
                ex.printStackTrace();
        }
}

public void escribirDatos(String dato) {
        try {
                OutputStream output = COM.getOutputStream();
                output.write(dato.getBytes());
                output.flush();

        } catch (Exception ex) {
                ex.printStackTrace();
        }
}

public void imprimirPropiedadesPuerto(){
```

```
  System.out.println ("Las Propiedades del Puerto son las siguientes:\n");
            this.imprimirBaudRate();
            this.imprimirDataBits();
            this.imprimirParity();
            this.imprimirStopBits();
  }

  public void imprimirBaudRate () {
  System.out.println("Tasa de bit: "+COM.getBaudRate());
  }

public void imprimirStopBits () {
System.out.println("Tasa de bit: "+ COM.getStopBits());
  }

public void imprimirDataBits () {
System.out.println("Bits de Datos: "+ COM.getDataBits());
  }

public void imprimirParity() {
   if ( COM.getParity() == 0){

            System.out.println("Bits de paridad: Ninguno");
   }else {

            System.out.println ("Bits de paridad:" +COM.getParity());
   }
   }

public void serialEvent(SerialPortEvent event) {
            if (event.getEventType() == SerialPortEvent.DATA_AVAILABLE) {

                        try {
```

```
Datos="";
        InputStream in = COM.getInputStream();

        byte[] dt = new byte[COM.getInputBufferSize()];
        int j = 0;

        for (int i = 0; i < dt.length; i++) {
                dt[i] = (byte) in.read();
                if (dt[i] == -1) {
                        break;
                }
                j++;
        }

        char[] dtReceived = new char[j];
        for (int i = 0; i <= j; i++) {
                if (dt[i] == -1) {
                        break;
                } else {
                        dtReceived[i] = (char) dt[i];
                }
        }

        String dtStr = new String(dtReceived);
        char a = dtStr.charAt(0);
        String resultado = "";

        for (int i = 0; i < dtStr.length()-1; i++){
                char b = dtStr.charAt(i);
                resultado = resultado + b;
        }

Datos=resultado.substring(6,11);
```

```
                    MiBDD.Obtener_datos_producto(Datos);

                    if (pf.Es_Unico(Datos)==true)
                    {

pf.Agregar_producto(Datos,MiBDD.Producto_Nombre,MiBDD.Producto_Precio,Mi
BDD.Producto_Peso);
                    }

                    } catch (Exception e) {
                            e.printStackTrace();
                    }
                }

        }
}
```

B.5 Clase Mensajes_Frame.java

```
/*
*Mensajes_Frame.java
*/

package id_productos;
public class Mensajes_Frame extends javax.swing.JFrame {
    public Mensajes_Frame() {
        initComponents();
        setVisible(true);
    }

    public Mensajes_Frame(String Texto) {
        initComponents();
        lblMensaje.setText(Texto);
```

```
        setVisible(true);
    }
    public void set_mensaje(String Texto)
    {
        lblMensaje.setText(Texto);
    }
        private void initComponents() {
        lblMensaje = new java.awt.Label();
        btnOK = new java.awt.Button();
setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOS
                                    E);
        setResizable(false);
        lblMensaje.setAlignment(java.awt.Label.CENTER);
        lblMensaje.setFont(new java.awt.Font("Dialog", 1, 12));
        lblMensaje.setForeground(new java.awt.Color(0, 0, 0));
        btnOK.setLabel("ACEPTAR");
        btnOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOKActionPerformed(evt);
            }
        });

javax.swing.GroupLayout layout = new
                                        javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                        .addGroup(layout.createSequentialGroup()
                       .addGap(19, 19, 19)
```

```
                    .addComponent(lblMensaje,
javax.swing.GroupLayout.PREFERRED_SIZE, 274,
                                        javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                    .addGap(121, 121, 121)
                    .addComponent(btnOK,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(24, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(19, 19, 19)
                .addComponent(lblMensaje,
javax.swing.GroupLayout.PREFERRED_SIZE, 72,
                                        javax.swing.GroupLayout.PREFERRED_SIZE)
  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                                                        29, Short.MAX_VALUE)
  .addComponent(btnOK, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,javax.swing.GroupLayout.PREFERR
                                                                ED_SIZE)
                .addGap(22, 22, 22))
        );
        pack();
    }
    private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
      dispose();
    }
        public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
```

```
            new Mensajes_Frame().setVisible(true);
        }
    });
  }
  private java.awt.Button btnOK;
  private java.awt.Label lblMensaje;
 }
```

B.6 Clase Productos_Frame.java

```
/*
 * Productos_Frame.java
*/

package id_productos;
import javax.print.attribute.standard.Media;
public class Productos_Frame extends javax.swing.JFrame {
   private String PrecioTotal;
   private int PesoTotal;
   private COMM_library Lectura_Peso = new
                                              COMM_library("COM5",9600,8,1,0);
   public Productos_Frame() {
      initComponents();
      setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
      setVisible(true);
      PrecioTotal="0.0";
   }
   public boolean Es_Unico(String ID)
   {
      int i;
      String Temp;
      Temp="";
      for(i=0;i<lstIDProductos.getItemCount();i++)
```

```
    {
        Temp=lstIDProductos.getItem(i);
         if(Temp.equalsIgnoreCase(ID))
        {
            return false;
        }
    }
    return true;
}
public String Suma_String(String Valor_1, String Valor_2){
    int Parte_Entera1;
    int Parte_Decimal1;
    int Parte_Entera2;
    int Parte_Decimal2;
    int Tempi;
    String Numero;
    String Temp;
    String DosDecimales="00";
    Numero=Valor_1.substring(0,Valor_1.indexOf("."));
    if(Numero.isEmpty())
        Numero="0";
    Parte_Entera1=java.lang.Integer.valueOf(Numero);
    Numero=Valor_1.substring(Valor_1.indexOf(".")+1);
    Temp=Numero+DosDecimales.substring(Numero.length());
    Parte_Decimal1=java.lang.Integer.valueOf(Temp);

    Numero=Valor_2.substring(0,Valor_2.indexOf("."));
    if(Numero.isEmpty())
        Numero="0";
    Parte_Entera2=java.lang.Integer.valueOf(Numero);
    Numero=Valor_2.substring(Valor_2.indexOf(".")+1);
    Temp=Numero+DosDecimales.substring(Numero.length());
    Parte_Decimal2=java.lang.Integer.valueOf(Temp);
```

```
    Parte_Decimal1=Parte_Decimal1+Parte_Decimal2;
    Tempi=Parte_Decimal1/100;
    Parte_Decimal1=Parte_Decimal1-Tempi*100;
    Parte_Entera1=Parte_Entera1+Parte_Entera2+Tempi;
    Temp=java.lang.Integer.toString(Parte_Decimal1);
    Temp=DosDecimales.substring(Temp.length())+Temp;
    return Parte_Entera1+"."+Temp;
}

public String Resta_String(String Valor_1, String Valor_2){

    int Parte_Entera1;
    int Parte_Decimal1;
    int Parte_Entera2;
    int Parte_Decimal2;

    int Tempi;
    String Numero;
    String Temp;
    String DosDecimales="00";

    Numero=Valor_1.substring(0,Valor_1.indexOf("."));
    if(Numero.isEmpty())
        Numero="0";
    Parte_Entera1=java.lang.Integer.valueOf(Numero);
    Numero=Valor_1.substring(Valor_1.indexOf(".")+1);
    Temp=Numero+DosDecimales.substring(Numero.length());
    Parte_Decimal1=java.lang.Integer.valueOf(Temp);

    Numero=Valor_2.substring(0,Valor_2.indexOf("."));
    if(Numero.isEmpty())
        Numero="0";
```

```
    Parte_Entera2=java.lang.Integer.valueOf(Numero);
    Numero=Valor_2.substring(Valor_2.indexOf(".")+1);
    Temp=Numero+DosDecimales.substring(Numero.length());
    Parte_Decimal2=java.lang.Integer.valueOf(Temp);

    if(Parte_Decimal1<Parte_Decimal2)
    {
       Parte_Decimal1=100-(Parte_Decimal2-Parte_Decimal1);
       Parte_Entera2=Parte_Entera2+1;
    }else
    {
       Parte_Decimal1=Parte_Decimal1-Parte_Decimal2;
    }

    Parte_Entera1=Parte_Entera1-Parte_Entera2;
    Temp=java.lang.Integer.toString(Parte_Decimal1);
    Temp=DosDecimales.substring(Temp.length())+Temp;
    return Parte_Entera1+"."+Temp;
}

public void Agregar_producto(String ID, String Nombre, String Precio, int Peso){

    String espacios30="                    ";
    Nombre=Nombre+espacios30.substring(Nombre.length());

    String espacios35="                     ";

    Nombre=ID+"          "+Nombre;
    jlListaProductos.add(Nombre+Precio+espacios35+Peso);
    lstIDProductos.add(ID);
    ListaPreciosProductos.add(Precio);
    ListaPesosProductos.add(java.lang.Integer.toString(Peso));
    PrecioTotal=Suma_String(PrecioTotal,Precio);
```

```
        PesoTotal=PesoTotal+Peso;
        jtPrecioTotal.setText(PrecioTotal);

        double Peso_kg;
        String Temp_peso;

        Peso_kg=PesoTotal*0.001;
        Temp_peso=java.lang.Double.toString(Peso_kg);
        if(Temp_peso.length()<5)
        {
            jtPesoTotal.setText(Temp_peso);
        }
        else
        {
            jtPesoTotal.setText(Temp_peso.substring(0,5));
        }
    }

    public void Eliminar_producto(){

        int Indice=jlListaProductos.getSelectedIndex();
        String TempS;
        float Tempf;
        int Tempi;

        if (Indice==-1){

            Mensajes_Frame Mf = new Mensajes_Frame("No se ha seleccionado
ningun producto!");

        }else{

            ListaPreciosProductos.select(Indice);
```

```
        TempS = ListaPreciosProductos.getSelectedItem();
        PrecioTotal=Resta_String(PrecioTotal,TempS);
        ListaPreciosProductos.remove(Indice);

        ListaPesosProductos.select(Indice);
        TempS = ListaPesosProductos.getSelectedItem();
        Tempi = java.lang.Integer.valueOf(TempS);
        PesoTotal=PesoTotal-Tempi;
        ListaPesosProductos.remove(Indice);
        lstIDProductos.remove(Indice);
        jlListaProductos.remove(Indice);
        jtPrecioTotal.setText(PrecioTotal);
        double Peso_kg;
        String Temp_peso;

        Peso_kg=PesoTotal*0.001;
        Temp_peso=java.lang.Double.toString(Peso_kg);
        if(Temp_peso.length()<5)
        {
            jtPesoTotal.setText(Temp_peso);
        }
        else
        {
            jtPesoTotal.setText(Temp_peso.substring(0,5));
        }
    }

}

    private void initComponents() {
    label1 = new java.awt.Label();
    jtPrecioTotal = new javax.swing.JTextField();
    jLabel1 = new javax.swing.JLabel();
```

```
        jtPesoTotal = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jtPesoMedido = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jbEliminarProducto = new javax.swing.JButton();
        jbNuevaLista = new javax.swing.JButton();
        jlListaProductos = new java.awt.List();
        label2 = new java.awt.Label();
        label3 = new java.awt.Label();
        label4 = new java.awt.Label();
        ListaPesosProductos = new java.awt.List();
        ListaPreciosProductos = new java.awt.List();
        jbPesoBalanza = new javax.swing.JButton();
        lstIDProductos = new java.awt.List();
        label5 = new java.awt.Label();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Facturacion de productos");
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowOpened(java.awt.event.WindowEvent evt) {
                formWindowOpened(evt);
            }
        });

        label1.setAlignment(java.awt.Label.CENTER);
        label1.setFont(new java.awt.Font("Arial", 1, 18));
        label1.setText("Facturaci\u00f3n de Productos");

        jtPrecioTotal.setEditable(false);
        jtPrecioTotal.setFont(new java.awt.Font("Tahoma", 1, 18));
        jtPrecioTotal.setHorizontalAlignment(javax.swing.JTextField.CENTER);
```

```
        jtPrecioTotal.setText("0.00");
        jtPrecioTotal.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jtPrecioTotalActionPerformed(evt);
            }
        });

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
        jLabel1.setText("Precio Total ($):");

        jtPesoTotal.setEditable(false);
        jtPesoTotal.setFont(new java.awt.Font("Tahoma", 1, 18));
        jtPesoTotal.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jtPesoTotal.setText("0");

        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14));
        jLabel2.setText("Peso Total (Kg):");

        jtPesoMedido.setEditable(false);
        jtPesoMedido.setFont(new java.awt.Font("Tahoma", 1, 18));
        jtPesoMedido.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        jtPesoMedido.setText("0");

        jLabel3.setFont(new java.awt.Font("Tahoma", 1, 14));
        jLabel3.setText("Peso obtenido de la balanza");

        jbEliminarProducto.setFont(new java.awt.Font("Tahoma", 1, 14));
        jbEliminarProducto.setText("Eliminar Producto");
        jbEliminarProducto.addChangeListener(new
    javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
                jbEliminarProductoStateChanged(evt);
            }
```

```
});
jbEliminarProducto.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jbEliminarProductoActionPerformed(evt);
    }
});

jbNuevaLista.setFont(new java.awt.Font("Tahoma", 1, 14));
jbNuevaLista.setText("Nueva Lista");
jbNuevaLista.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jbNuevaListaActionPerformed(evt);
    }
});

label2.setAlignment(java.awt.Label.CENTER);
label2.setText("Producto");
label3.setText("Precio ($)");
label4.setText("Peso (gramos)");

ListaPesosProductos.setVisible(false);
ListaPreciosProductos.setVisible(false);

jbPesoBalanza.setFont(new java.awt.Font("Tahoma", 1, 14));
jbPesoBalanza.setText("Obtener Peso");
jbPesoBalanza.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jbPesoBalanzaActionPerformed(evt);
    }
});

lstIDProductos.setVisible(false);
```

```
        label5.setAlignment(java.awt.Label.CENTER);
        label5.setText("C\u00f3digo");

        javax.swing.GroupLayout layout = new
                                        javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap(27, Short.MAX_VALUE)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
          .addGroup(layout.createSequentialGroup()
          .addComponent(label5,javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                          .addGap(32, 32, 32)
        .addComponent(label2, javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                          .addGap(19, 19, 19)
        .addComponent(label3, javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
                          .addGap(18, 18, 18)
          .addComponent(label4, javax.swing.GroupLayout.PREFERRED_SIZE,
                              javax.swing.GroupLayout.DEFAULT_SIZE,
                                    javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(layout.createSequentialGroup()
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(23, 23, 23)
                        .addComponent(jLabel1))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                            .addComponent(jbPesoBalanza)
                            .addComponent(jtPrecioTotal,
javax.swing.GroupLayout.PREFERRED_SIZE,                    134,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jbEliminarProducto))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(56, 56, 56)
                        .addComponent(jLabel2))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(27, 27, 27)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
                            .addComponent(jtPesoMedido,
javax.swing.GroupLayout.PREFERRED_SIZE,                    159,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                                .addComponent(jtPesoTotal,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 159,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jbNuevaLista,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 136,
javax.swing.GroupLayout.PREFERRED_SIZE))))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED))
                        .addComponent(jlListaProductos,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 349,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
                        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                                .addComponent(ListaPreciosProductos,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addComponent(ListaPesosProductos,
javax.swing.GroupLayout.DEFAULT_SIZE, 157, Short.MAX_VALUE))
                            .addGap(40, 40, 40))
                        .addComponent(lstIDProductos,
javax.swing.GroupLayout.PREFERRED_SIZE, 102,
                                javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(layout.createSequentialGroup()
                    .addGap(97, 97, 97)
                    .addComponent(label1,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(132, Short.MAX_VALUE)
                .addComponent(jLabel3)
                .addGap(117, 117, 117))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG, false)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                        .addGap(70, 70, 70)
                        .addComponent(ListaPesosProductos,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    115,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(47, 47, 47)
                        .addComponent(ListaPreciosProductos,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addGap(64, 64, 64)
                        .addComponent(lstIDProductos,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(92, 92, 92))
                    .addGroup(layout.createSequentialGroup()
```

```
                .addContainerGap()
                .addComponent(label1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,  16,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
                    .addComponent(label4,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(label2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(label3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(label5,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jlListaProductos,
javax.swing.GroupLayout.PREFERRED_SIZE,                                  271,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(10, 10, 10)))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASE
LINE)
                .addComponent(jLabel1)
                .addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASE
LINE)
                .addComponent(jtPrecioTotal,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jtPesoTotal,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(28, 28, 28)
            .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                .addComponent(jbPesoBalanza,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jtPesoMedido))
            .addGap(46, 46, 46)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASE
LINE)
                .addComponent(jbEliminarProducto)
                .addComponent(jbNuevaLista))
            .addContainerGap())
        );
        pack();
    }

    private void jbPesoBalanzaActionPerformed(java.awt.event.ActionEvent evt) {
        String Peso;
        Peso = Lectura_Peso.Obtener_Dato_Dispositivo("P");
        jtPesoMedido.setText(Peso);
    }
    private void jbNuevaListaActionPerformed(java.awt.event.ActionEvent evt) {

            PrecioTotal="0.0";
            PesoTotal=0;
            ListaPesosProductos.removeAll();
            ListaPreciosProductos.removeAll();
            jlListaProductos.removeAll();
            lstIDProductos.removeAll();
            jtPrecioTotal.setText(PrecioTotal);
            jtPesoTotal.setText(java.lang.Integer.toString(PesoTotal));

    }
private void jbEliminarProductoActionPerformed(java.awt.event.ActionEvent evt) {
            Eliminar_producto();
    }

    private  void  jbEliminarProductoStateChanged(javax.swing.event.ChangeEvent
evt) {
```

```
}

private void jtPrecioTotalActionPerformed(java.awt.event.ActionEvent evt) {
}

private void formWindowOpened(java.awt.event.WindowEvent evt) {
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Productos_Frame().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private java.awt.List ListaPesosProductos;
private java.awt.List ListaPreciosProductos;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JButton jbEliminarProducto;
private javax.swing.JButton jbNuevaLista;
private javax.swing.JButton jbPesoBalanza;
private java.awt.List jlListaProductos;
private javax.swing.JTextField jtPesoMedido;
private javax.swing.JTextField jtPesoTotal;
private javax.swing.JTextField jtPrecioTotal;
private java.awt.Label label1;
private java.awt.Label label2;
private java.awt.Label label3;
private java.awt.Label label4;
```

```
    private java.awt.Label label5;
    private java.awt.List lstIDProductos;
    // End of variables declaration
    }
```

B.7 Clase bdd_ODBC.java

```
/*
 * bdd_ODBC.java
 */

package id_productos;
import java.sql.*;
public class bdd_ODBC {
    private Connection conexion;
    private Statement sentencia;
    private ResultSet resultado;
    public String Producto_Nombre;
    public String Producto_Precio;
    public int Producto_Peso;
    public bdd_ODBC(String Nombre_ODBC, String Nombre_Usuario, String
                                Password) {
        Conectarse_ODBC(Nombre_ODBC, Nombre_Usuario, Password);
    }
      public void Obtener_datos_producto(String Producto_ID){

    try {
        Producto_Nombre = "No existe";
        Producto_Precio = "0.0";
        Producto_Peso = 0;
        resultado = sentencia.executeQuery("Select * from Productos where
Producto_ID="'+Producto_ID+'"" );
        while( resultado.next() ) {
```

```
            Producto_Nombre = resultado.getString( "Producto_Nombre" );
            Producto_Precio = resultado.getString("Producto_Precio");
            Producto_Peso = resultado.getInt("Producto_Peso");
        }

    } catch( SQLException e ) {
        System.out.println(e.getMessage());
    };

}
  public void Conectarse_ODBC(String Nombre_ODBC, String
                                        Nombre_Usuario, String Password){

        try {
          Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
        } catch( Exception e ) {
          System.out.println( "No se pudo cargar el puente JDBC-ODBC." );
          return;
        }

        try {

          conexion = DriverManager.getConnection( "jdbc:odbc:" + Nombre_ODBC
,Nombre_Usuario,Password );
          sentencia = conexion.createStatement();
        } catch( SQLException e ) {
          System.out.println( "No se pudo conectar al ODBC: " + Nombre_ODBC );
          return;
        };
     return;

    }
}
```

ANEXO C

PROGRAMACIÓN JAVA

Java posee componentes predefinidos denominados clases que se agrupan en categorías de clases, que se relacionan entre sí por medio de directorios de discos a los cuales se llaman paquetes. A los paquetes se los llama también bibliotecas de clases de Java o interfaz de programación de aplicaciones de Java (Java API).

Cuando se programa en Java, se coloca todo el código en métodos, de la misma forma que se escriben funciones en lenguajes como C.

La sintaxis de un lenguaje define los elementos de dicho lenguaje y cómo se combinan para formar un programa. Los elementos típicos de cualquier lenguaje son los siguientes:

- Comentarios
- Identificadores: los nombres que se dan a las variables
- Palabras reservadas: las palabras que utiliza el propio lenguaje
- Sentencias
- Bloques de código
- Expresiones
- Tipos de datos
- Operadores

COMENTARIOS

Los comentarios en los programas ayudan a entender que se está realizando en cada línea, le hace más comprensible al programa.
En Java hay tres tipos de comentarios:

// comentarios para una sola línea

/* comentarios de una o más líneas */

/** comentario de documentación, de una o más líneas */

IDENTIFICADORES

Los identificadores nombran variables, funciones, clases y objetos estos es; cualquier cosa que el programador necesite identificar o usar.

En Java, un identificador comienza con una letra, un subrayado (_) o un símbolo de dólar ($). Los siguientes caracteres pueden ser letras o dígitos. Se distinguen las mayúsculas de las minúsculas y no hay longitud máxima.

Palabras clave

Las palabras claves que están definidas en Java y que no se pueden utilizar como identificadores constan en la tabla C.1.

abstract	boolean	break	Byte	Case
catch	char	class	Continue	Default
do	double	else	Extends	False
final	finally	float	For	If
implements	import	instanceof	Int	Interface
long	native	new	Null	Package
private	protected	public	Return	Short
static	super	switch	Synchronized	This
throw	throws	transient	True	Try
void	volatile	while		

Tabla C.1: Palabras claves definidas en Java

Palabras Reservadas

Son palabras con un significado especial para el compilador, en la tabla C.2 se muestran estas palabras.

byvalue	cast	const	future	Generis
goto	inner	operador	outer	Rest
var				

Tabla C.2: Palabras reservadas de Java

SENTENCIAS

Una sentencia es una orden que se le da al programa para realizar una tarea específica, ésta puede mostrar un mensaje en la pantalla, declarar una variable (para reservar espacio en memoria), inicializarla, llamar a una función, etc. Las sentencias acaban con “;“, este caracter separa una sentencia de la siguiente. Normalmente, las sentencias se ponen unas debajo de otras, aunque sentencias cortas pueden colocarse en una misma línea. He aquí algunos ejemplos de sentencias.

```
int i=1;
import java.awt.*;
System.out.println("El primer programa");
rect.mover(10, 20);
```

BLOQUES DE CÓDIGO

Un bloque de código es un grupo de sentencias que se comportan como una unidad. Un bloque de código está limitado por las llaves de apertura { y cierre }.

EXPRESIONES

Una expresión es todo aquello que se puede poner a la derecha del operador asignación =. Por ejemplo:

```
x=123;
y=(x+100)/4;
area=circulo.calcularArea(2.5);
```

La primera expresión asigna un valor a la variable x.
La segunda, realiza una operación
La tercera, es una llamada a una función miembro calcularArea desde un objeto círculo de una clase determinada.

TIPOS DE DATOS PRIMITIVOS

Todo lenguaje computacional debe soportar la existencia de variables de tipos numérico, lógico y de caracter para la construcción de un programa.

Java cuenta con ocho tipos de datos primitivos para almacenar distintos rangos de valores.

Tipo	Lo que almacena	Rango
Byte	entero de 1 byte (8 bits)	de -128 a 127
Short	entero de 2 byte (16 bits)	de -32768 a 32767
Int	entero de 4 byte (32 bits)	de -2147483648 a 2147483647
Long	entero de 8 byte (64 bits)	de -2-63 a 263 – 1
Float	entero de 4 byte (32 bits)	6 dígitos significativos (10-46 , 1038)
Double	entero de 8 byte (64 bits)	15 dígitos significativos (10-324 , 10308)
Char	caracter UNICODE 2 bytes (16 bits)	Comprende el código ASCII
Boolean	variable booleana de 1 byte (8 bits)	false y true

Tabla C.3: Datos primitivos de Java

Enteros

Los tipos *byte, short, int* y *long* sirven para almacenar datos enteros. Se pueden asignar enteros normales o enteros octales y hexadecimales. Los octales se indican anteponiendo un cero al número, los hexadecimales anteponiendo 0x.

Reales de coma flotante

Los decimales se almacenan en los tipos ***float*** y ***double***. Se les llama de coma flotante por cómo son almacenados por el ordenador. Los decimales no son almacenados de forma exacta por eso siempre hay un posible error. En los decimales de coma flotante se habla, por tanto de precisión. Es mucho más preciso el tipo ***double*** que el tipo ***float*****.**

Booleanos

Los valores booleanos (o lógicos) sirven para indicar si algo es verdadero (***true***) o falso (***false***).

Caracteres

Los valores de tipo caracter sirven para almacenar símbolos de escritura (en Java se puede almacenar cualquier código Unicode). Los valores Unicode son los que Java utiliza para los caracteres. Ejemplo:

```
char letra;

letra='C';      //Los caracteres van entre comillas

letra=67;       //El código Unicode de la C es el 67. Esta línea //hace lo mismo que
                la anterior
```

OPERADORES

Los operadores de Java son muy parecidos en estilo y funcionamiento a los de C. En la tabla C.4 aparecen los operadores que se utilizan en Java, por orden de precedencia. De izquierda a derecha la precedencia es la misma.

Operador			
()	[]	.	
++	--	~	
*	/	%	
+	-		
>>	>>>	<<	<<<
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	+=, -=, *=, …		

Tabla C.4: Operadores utilizados en Java

A continuación se definirán otros conceptos que intervienen para la programación en Java.

VARIABLES

Una variable es un nombre que se asocia con una porción de la memoria del ordenador, en la que se guarda el valor asignado a dicha variable. Hay varios tipos de variables que requieren distintas cantidades de memoria para guardar datos.
Se puede emplear un identificador válido, es decir una serie de caracteres que pueden ser letras, dígitos, subrayados de cualquier longitud distinguiéndose entre mayúsculas o minúsculas para dar un **nombre de variable o de referencia.**

Cuando se declara una variable dentro de un método no se lo puede ocupar fuera de este método.

Toda variable posee un nombre, un tipo, un tamaño y un valor.

Los campos de texto

Proporcionan los datos que han sido ingresados por los usuarios por medio del teclado, o para exhibir información en la pantalla.

Cuando se hace uso en un *applet* del método ***init*** es para inicializar las variables y referencias que serán utilizadas en el *applet*. Los *applets* siempre utilizarán tres métodos: *init*, *start* y *paint*. Al tener variables del tipo de datos primitivos int, se inicializan de manera automática en el valor 0; en cambio en las variables de ejemplar del tipo de datos primitivos *boolean* se inicializará de manera automática en *false*. Los enunciados ejecutables necesitan ser inicializados.

New

Para la creación de un objeto se utiliza la palabra ***new***, con esto se hace un pedido a la computadora para que exista memoria para almacenarlo.

Para las variables de tipo primitivo no es necesario lo anteriormente expuesto puesto que Java asigna automáticamente memoria para estas variables.

La interacción del usuario con la GUI produce un **evento**. Este procesamiento se conoce como programación controlada por eventos.

Action

El método ***action***, el cual procesará interacciones entre el usuario y GUI, recibe dos argumentos uno denominado *Event* (evento) que determinará qué evento ocurrió y otro *object* (objeto) que contendrá información acerca del evento.

Una ventaja de Java es que cualquier objeto de cualquier clase puede convertirse en una cadena o *String* al utilizar el método ***toString*.**

Cuando se almacena un valor en una posición de la memoria, este valor sustituirá al anterior valor que será borrado.

Todas las variables han de declararse antes de usarlas; la declaración consiste en una sentencia en la que figura el tipo de dato y el nombre que se asignan a la variable. Una vez declarada se le podrá asignar valores.

Java tiene tres tipos de variables:

- de instancia
- de clase
- locales

Las variables de instancia se usan para guardar los atributos de un objeto particular.

Las variables de clase son similares a las variables de instancia, con la excepción de que los valores que guardan son los mismos para todos los objetos de una determinada clase.

Las variables locales se declaran en el momento en el que son necesarias. Es una buena costumbre inicializar las variables en el momento en el que son declaradas.

SEPARADORES

Sólo hay un par de secuencias con otros caracteres que pueden aparecer en el código Java; son los separadores simples, que van a definir la forma y función del código. Los separadores admitidos en Java son:

() - *paréntesis.* Para contener listas de parámetros en la definición y llamada a métodos. También se utiliza para definir precedencia en expresiones, contener expresiones para control de flujo y rodear las conversiones de tipo
{} - *llaves.* Para contener los valores de matrices inicializadas automáticamente. También se utiliza para definir un bloque de código, para clases, métodos y ámbitos locales.

[] - *corchetes*. Para declarar tipos matriz. También se utiliza cuando se hace referencia a valores de matriz.

; - *punto y coma.* Separa sentencias.

, - *coma.* Separa identificadores consecutivos en una declaración de variables. También se utiliza para encadenar sentencias dentro de una sentencia for.

. - *punto*. Para separar nombres de paquete de subpaquetes y clases. También se utiliza para separar una variable o método de una variable de referencia.

CLASES

Una ventaja de Java es que contiene en los paquetes de Java API gran cantidad de clases que se pueden reutilizar en lugar de crearlas cada vez que se las necesite.

El enunciado ***import*** cargará las clases para poder compilar el programa.

import paquete.clase

En este enunciado se explica cuál es el orden del enunciado para poder cargar la clase correspondiente a determinado paquete.

Es importante conocer cómo definir una **clase** ya que en cualquier programa de Java por lo menos se debe definir una clase.

Algunos paquetes se muestran en la tabla C.5.

Paquete	Ofrece soporte para
Java.applet	Crea programas (*applets*) que son fácilmente transportable a través de la red
Java.awt	Dibuja gráficos y crea interfaces gráficas de usuario (AWT, *Abstract Windows Toolkit*)
Java.beans	Define componentes de *software* que son fácilmente combinados en aplicaciones
Java.io	Ejecuta una variedad de funciones de entrada y salida
Java.lang	Soporte general; automáticamente importado en todos los programas java
Java.math	Realiza cálculos con una alta precisión
Java.net	Comunicación a través de la red
Java.security	Restricciones de seguridad
Java.sql	Acceso a bases de datos
Java.text	Formato de texto para salida
Javax.comm	Para el acceso al API *Java Communications*
Java.util	Utilerías generales

Tabla C.5: Paquetes que pueden ser utilizados en Java

CONTROL DE FLUJO

Toma de Decisiones

Este tipo de sentencias definen el código que debe ejecutarse si se cumple una determinada condición. Java posee tres tipos de estructuras de selección: *if, if/else, switch:*

- *Estructura if:* permite tomar decisiones en base a un cumplimiento de una condición. Esta estructura también denominada estructura de selección única.

Si la condición del *if* se cumple se realiza el enunciado que se encuentra en el cuerpo del *if* caso contrario sale del cuerpo del *if* y sigue con las instrucciones.

Si la estructura *if* solo posee un enunciado no necesitará del uso de llaves caso contrario será necesario para determinar el cuerpo de la estructura *if*.

A este grupo de enunciados encerrados entre llaves se lo denomina enunciado compuesto.

- *Estructura if/else:* que se conoce también como estructura de selección doble.

  Esta estructura realiza una acción si la condición especificada se cumple y otra acción si ésta no se cumple; es decir selecciona entre dos acciones diferentes.

  El operador condicional tiene una gran relación con *if*/*else*. Éste es el denominado operador ternario de Java pues necesita tres operandos.

  En *if*/*else* se permite tener estructuras anidadas, es decir, *if*/*else* dentro de otra estructura *if*/*else*.

- *Estructura switch:* también denominada estructura de selección múltiple debido a que esta estructura selecciona entre muchas acciones dependiendo del valor de una expresión. La sentencia *switch* va seguida de una variable que será la expresión de control entre paréntesis. El enunciado *break* al final del enunciado de *switch* hace que salga de la estructura *switch*.

  Si no se encuentra coincidencia se procederá a realizar el caso *default*.

La estructura *switch* no necesita que sus acciones sean encerradas en llaves como el resto de estructuras.

Sintaxis	Ejemplos

if (condición1) {
  sentencias;
} else if (condición2) {
  sentencias;
  ...
} else if(condiciónN) {
  sentencias;
} else {
  sentencias;
}
	If
	(a == 1) {
	  b++;
	} else if (b == 1) {
	  c++;
	} else if (c == 1) {
	  d++;
	}

switch (condición) {
  case caso1: sentencias;
  case caso2: sentencias;
  case casoN: sentencias;
  default: sentencias;
}
	switch (a) {
	  case 1: b++;
	  break;
	  case 2: c++;
	  break;
	  default:b--;
	  break;
	}

Tabla C.6: Sintaxis y ejemplos de las sentencias para la toma de decisiones en Java

Bucles

Para repetir un conjunto de sentencias durante un determinado número de iteraciones se tienen las sentencias *for*, *while* y *do*...*while*:

- *La estructura for:* se encarga de la repetición controlada por un contador.

for (x1; x2; x3)

Donde x1 indicará el inicio de la repetición, x2 será la condición para la continuación de la repetición, x3 indicará el incremento de la variable de control, las mismas que podrían contener expresiones aritméticas.

El x2 se puede omitir, en tal caso Java supondrá que la condición se cumple y seguirá con el ciclo.

- *Estructura de repetición while:* Esta estructura producirá que se repita una acción mientras se cumpla una condición. El cuerpo de la estructura *while* puede que no se ejecute ni una sola vez. El cuerpo de la estructura *while* puede ser de un solo enunciado o de un enunciado compuesto.

  Cuando la condición se deja de cumplir la repetición termina y se ejecuta el enunciado que sigue de la estructura de repetición.

```
X1;
while (x2) {
x3;
}
```

Siendo x1, x2 y x3 lo mismo que para la estructura *for*.

- *Estructura do/while:* la condición de esta estructura se prueba luego de ser ejecutado el cuerpo del mismo y por esto el cuerpo del ciclo se ejecuta por lo menos una vez.

```
do {
enunciado
} while (condición);
```

Sintaxis	Ejemplo

for (inicio; condición; incremento)
{
  sentencias;
}
	for (i=1;i<10;i++)
	{
	  b = b+i;
	}

while (condición){
  sentencias;
}
	while (i < 10) {
	  b += i;
	  i++;
	}

do{
  sentencias;
} while (condición);
	do {
	  b += i;
	  i++;
	} while (i < 10);

Tabla C.7: Sintaxis y ejemplos de las sentencias para bucles en Java

Sentencias de Ruptura

Se tienen las sentencias *break* (para terminar la ejecución de un bloque o saltar a una etiqueta), *continue* (para forzar una ejecución más de un bloque o saltar a una etiqueta) y *return* (para salir de una función devolviendo o sin devolver un valor).

ESTRUCTURAS DE CONTROL

Por lo general los enunciados de un programa se dan por ejecución secuencial, que quiere decir que se ejecuta uno sentencia tras de otra, tal como están escritos.

Cuando se desea ejecutar un enunciado que no está en la secuencia por otro se denomina transferencia de control.

En Java se tiene la estructura de secuencia.

OPERADORES DE INCREMENTO Y DECREMENTO

Java posee operadores tanto de incremento como de decremento unario: ++,-- respectivamente. Si estos operadores son colocados antes de una variable se los denomina operador de preincremento o de predecremento, si se colocan los operadores después de las variables son denominados operadores de postincremento o de postdecremento. Estos valores se utilizan para sumar uno a la variable y posteriormente hacer uso de esta variable o para que se use esta variable y posteriormente se sume uno o se lo reste.

SECUENCIAS COMUNES DE ESCAPE

Secuencia de escape	Descripción
\n	Nueva línea. El cursor se colocará al principio de la siguiente línea.
\t	Colocará el cursor al siguiente tabulador.
\r	Colocará el cursor al principio de la misma línea.
\\	Este permitirá imprimir el caracter de diagonal invertida.
\’	Imprime un caracter de apóstrofo.
\”	Imprime un caracter de comillas.

Tabla C.8: Secuencias comunes de escape utilizadas en Java

\ Este caracter indica que se deberá enviar un caracter especial a la salida.

Estas secuencias de escape se pueden usar como caracteres individuales cuando están encerrados en apóstrofos o como caracteres de cadena.

MÓDULOS DE PROGRAMA EN JAVA

Para la creación de un programa es mejor hacerlo por pequeñas piezas o módulos, en Java esos módulos se los llama métodos y clases.

Para que un método realice una tarea que se necesita en un programa se requiere realizar una llamada de método en donde se especifica el nombre del método y se proporciona la información en el argumento necesario para la realización del trabajo.

Métodos

Si se empaqueta en métodos los códigos se podrá ejecutar desde varios puntos del programa solo invocándolo.

El cuerpo del método también se lo llama bloque. No se permite declarar un método dentro de otro método.

El enunciado *return* devuelve el control al punto donde se invocó el método.

Un punto importante en los métodos es la restricción de los argumentos, esto es obligarles a los argumentos sean de un tipo apropiado para que puedan llegar a un método.

ANEXO D

1. Introduction

How to find a lightest way to contact your PDA to the USB port on your PC? USB Serial Converter operates as a bridge between one USB port and standard RS-232 Serial port. You just easily hook the cable into PC or Hub's port, and it can connect any RS-232 devices, such as PDA, scanner, printer…etc.

Features

- Compliant with the USB 1.1 version specification
- Supports RS-232 serial Interface
- Supports 500kbps data transfer rate
- Supports USB suspend condition
- Plug & Play
- Draws its power from USB connection – no extra power adapter required
- Supports Windows 98SE, ME, 2000, XP, 2003 Server/ Vista and Mac OS X 10.1 ~ 10.4.

Package Contents

Before installation, please check the items of the package.

- TU-S9 x1
- Driver CD-ROM x1
- Quick Installation Guide x1

System Requirements

- IBM compatible computer or Mac
- Windows® 98SE, ME, 2000, XP, 2003 Server, Vista, Mac OS X 10.1~10.4
- Available USB port
- 64 MB RAM or more.
- Pentium 233 MHz or higher

2. Installation

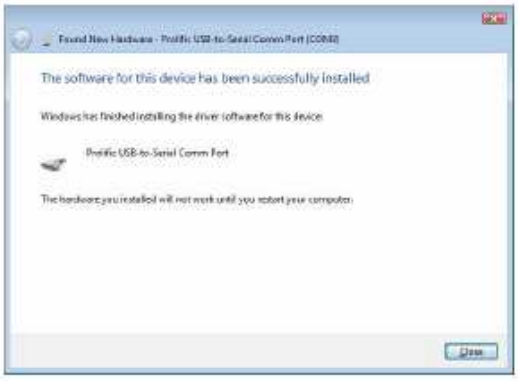
On Windows Vista

1. Connect the USB Serial Converter to the USB port on your computer.

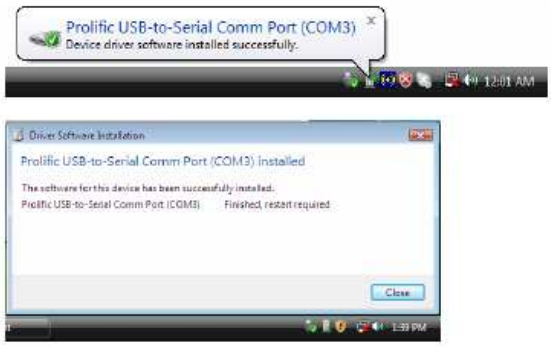
2. When the **Found New Hardware** appears, select **Locate and install driver software (recommended)**.

3. Insert the provided CD-ROM into your CD-ROM drive and click **Next**. (Sometimes the installer will automatically proceed to next step.)

4. When the installation is done, click **Close**.



5. A pop-up message will appear in the notification area indicating the installation is complete. Restart your computer if prompted to do so.

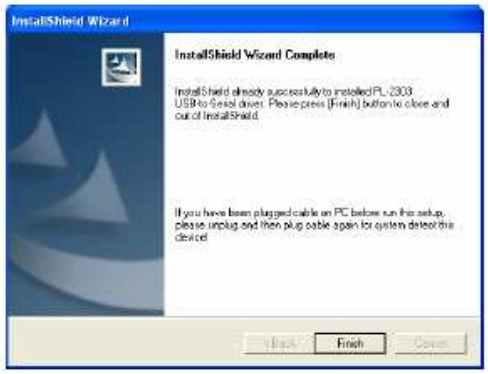


On Windows 98SE/ME/2000/XP/2003 Server

Note: DO **NOT** connect the USB Serial Converter to your computer before completing the driver installation.

1. Insert the provided CD into your CD-ROM drive. Run the **Setup** file under ***x*:\Driver\Win98_2003** where x: is your CD-ROM drive letter. When the welcome screen appears, click **Next**.

2. When the following screen appears, click **Finish**.



Once the installation is done, connect the converter to a free USB port on your computer.

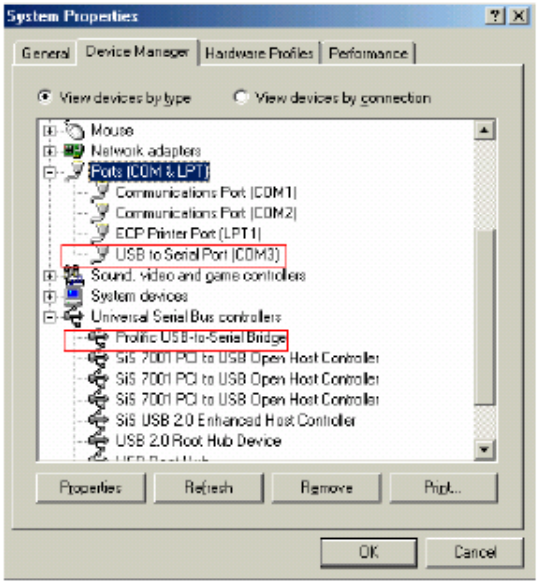
Verify the Driver Installation on Windows

To verify your converter installation, please launch **Device Manager** by the steps below:

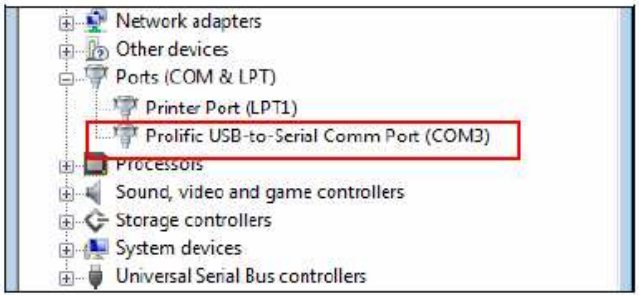
- On Windows 98SE/ME/2000/XP/2003: Right-click the My Computer icon on the desktop and select **Properties** > **System** > (**Hardware**) > **Device Manager**.
- On Windows Vista: Right-click the **Computer** icon on the desktop and select **Properties** > **Device Manager**.

In the **Ports (COM & LPT)** group, a string similar to **USB to Serial Port** or **Prolific USB to Serial Comm Port** should be displayed.
If there is a question or exclamation mark next to that item, then the driver is not properly installed. Please delete the item and repeat the installation steps.

Windows 98SE/ME



Windows 2000/XP/2003/Vista



On Mac OS X

The provided Mac X driver supports:

- Mac OS 10.1 and above for PowerPC based Mac
- Mac OS 10.4 and above for Intel based Mac

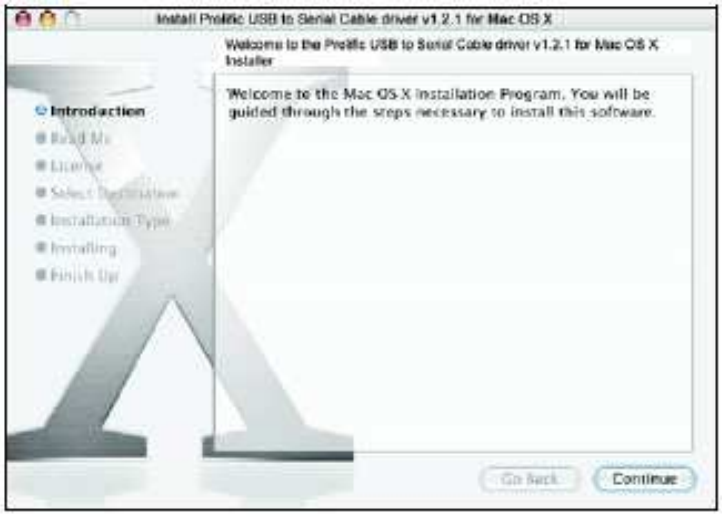
To install the drivers:

1. Insert the provided driver CD into your CD-ROM drive. Double-click the CD icon on the desktop and navigate to the directory of **\Driver\MacX**. Copy and then paste the **PL2303_1.2.1r2.dmg** file to the desktop.

2. Double-click the dmg file on the desktop to extract the file. Then double-click the **PL2303_1.2.1** file that is extracted.

PL2303_1.2.1

3. When the welcome screen appears, click **Continue**.



4. Select a destination disk to install the driver and click **Continue**.

5. Click **Install**.

6. When the **Authenticate** screen appears, enter your password in the provided field and click **OK**.

7. Click **Continue Installation**.

8. Click **Restart** to finish installing the driver and reboot your Mac.

9. Connect the converter to a free USB port on your Mac.
10. To verify the installation: Go to **Apple System Profiler** > **Extensions** and you should find **ProlificUsbSerial** among the list. This means the installation is successfully done.

Connecting Serial Device

Now you can connect your RS-232 device to the converter. If prompted to install the driver for your serial device, follow the manual that comes with your serial device to complete the installation.

3. Specifications

Data Transfer Rate	500 kbps
Function	USB1.1 Serial Converter operates as a bridge between one USB port and standard RS-232 serial port
Ports	USB 1.1, USB Type A male
Power	Bus-powered
Operating System Support	Windows 98SE, ME, 2000, XP, 2003 Server, Vista Mac OS X 10.1 ~ 10.4
Accessories	Driver CD, Quick Installation Guide

* Specification is subject to change without further notice.

4. Regulatory Compliance

FCC Conditions

This equipment has been tested and found to comply with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) This device must accept any interference received. Including interference that may cause undesired operation.

CE

This equipment is in compliance with the requirements of the following regulations:
EN 55 022: CLASS B

ANEXO E

E.1 Características de los lectores RFID largo alcance UHF-EPC Gen 2

Lector RFID largo alcance UHF-EPC Gen 2.	
Marca: SIRIT	Modelo: MP9320

Características

- Tecnología Multi-protocolo: Suporta todos los protocolos de tag (simultáneamente o individualmente)
- Hasta 4 antenas: Ideal para instalación en puertas de depósitos o aplicaciones de cintas transportadoras
- Alto rendimientos para la lectura y escritura de múltiples tag simultáneamente.
- Opciones de interfaces múltiples: RS-232, RS-485, 10BaseT 10/100 Mbps *Ethernet*
- Capacidad de integración con la infraestructura existente
- Sensibilidad y poder de lectura ajustable
- *Software* de configuración e instalación disponible (RF *Command Suite Configurator*)

El lector MP9320 UHF-EPC es un producto de avanzada tecnología debido a su soporte multi-protocolo, su flexibilidad en la frecuencia para soportar diferentes regiones y su facilidad programación.

Esta diseñado para leer cualquier *tag* EPC, incluyendo Clase 0, 0+, Clase 1, ISO y EPC Clase 1 Gen 2 (Generación 2) además de ISO 18000-6A y 6B, EM Marin 4022, 4222 y 4223, *Intermec Intellitag, Philips* UCODE EPC 1.19.

Posee un amplio rango y velocidad de lectura y transmisión de datos.

Especialmente indicado para la administración de activos y aplicaciones logísticas que requieran la lectura simultánea de un gran número de *tags* a grandes distancias, posee la capacidad de conectar hasta cuatro antenas, permitiendo su uso estaciones de carga, cintas transportadoras, *tracking* de contenedores y *pallets*, así como también administración de inventarios.

Gracias a su diseño expandible, el MP9320 soporta múltiples protocolo y *tags* pudiendo actualizarse fácilmente para soportar nuevos modelos de *tags* y protocolos estándares.

Especificaciones técnicas.

Tags soportados	EPC Class 0, EPC Class 0+, EPC Class 1, EPC Class 1 Gen 2, ISO 18000-6A, ISO 18000-6B, UCODE EPC 1.19, EM4022, EM4222, EM4223, IntellitagTM
Frecuencia de operación	902-928 MHz (100 KHz steps)
	869.525 MHz
	865-868 MHz (200 KHz steps)
RF Power	4W EIRP (FCC)
	500 mW/2W ERP (ETSI)
Temperatura de operación	-20 a 70 C
Antenas	Hasta 4 antenas conectores SMA
Interfaz	RS-232 (EIA/TIA-232F)
	RS-485 (EIA/TIA-485A)
	10 Base-T Ethernet 10/100 MBPS
Gabinete Metálico	Aluminio
Dimensiones	127 mm x 178 mm x 241 mm
Peso	1.8 kg

E.2 Características de la balanza de suelo PCE-TP 1500

+ Protección IP 65 / 68

- Verificable, se envía verificada de fábrica
- Disponible en 2 versiones (de acero lacado o de acero noble)
- Pantalla de acero noble (sin importar la versión que adquiera)
- Funciones de tara, suma y cómputo de piezas
- Pantalla conectada a un cable de 3 m
- Puerto RS-232 para la transmisión directa al PC
- Las 2 rampas se incluyen en el envío
- El trípode se incluye en el envío
- Admitido para pesados según la clase comercial III

Especificaciones Técnicas	
Rango de pesado	0 ... 1500 kg
Capacidad de lectura	500 g
Valor de verificación	500 g
Carga mínima	10 kg
Rango de taraje	en todo el rango de pesado
Tiempo de respuesta	< 4 s
Unidades de pesado	Kg.
Sobrecarga máxima	200 %
Verificación	verificable según la clase M III (se envía verificado)
Pantalla	LED de alto contraste
Plataforma útil	1250 x 1250 mm
Dimensiones de la plataforma	1500 x 1250 mm
Dimensiones de la rampa	1500 x 470 x 55 mm (cada rampa)
Temperatura ambiental	-10 ... +40 °C
Alimentación	230 V / 50 Hz
Carcasa	pantalla de acero noble IP 65 / balanza según variante: acero lacado o acero noble
Tipo de protección	IP 68
Peso	350 kg
Homologación	CE / OIML clase III

E.2 Características de la fuente de alimentación eléctrica PULS SLV20.200

Data sheet

Buffering: Electrolytic caps. instead of accumulators

SLV20.200

- Buffering for 24V loads
- Minimum hold-up time: 0.2s/20A (max. buffer time depends on load)
- Fit for industrial use: Energy storage in electrolytic caps., no accumulators
- Clear status indication by Status LED and signalling terminals

PULS

CB scheme IEC60950

cULus UL508 LISTED IND. CONT. EQ. 18 WM, 60°C

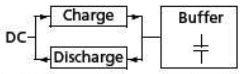
cULus UL60950 E137006 CUL/CSA-C22.2 No 60950

CE EMC and Low Volt. Directive

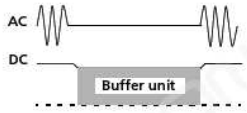
Short description

The buffer unit is a supplementary device for regulated DC 24V power supplies. It buffers load currents during typical mains faults and switching events or load peaks.

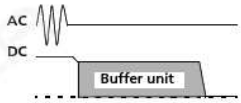
Working principle
In times when the power supply provides sufficient voltage, the buffer unit stores energy in integrated electrolytic capacitors. In case of a mains voltage fault, this energy is released again in a regulated process.



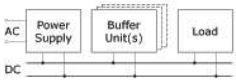
Bridges mains faults without interruption
Statistics show that 80 percent of all mains faults last less than 0.2s. These mains faults are completely bridged by the buffer unit and will have no influence on the DC power (startup-delay of power supply used might be taken under consideration. This increases the reliability of the system as a whole.



Extended hold-up time
Once the main power fails or is switched off, the buffer unit will continue to provide the load current for a defined period of time. Process data can be saved and processes can be terminated before the DC power switches off. Controlled restarts are subsequently possible.



Easy to handle, expandable and maintenance-free
The buffer unit does not require any control wiring. It can be added parallel to the load circuit at any given point. Any given number of buffer units can be switched parallel to increase the output capacity or the hold-up time. The dual terminals allow for easy wiring. In addition, there is a housing connection.



Short Overview - Technical Data

Rated voltage	DC 24V
Voltage range	DC 24...28.8V
Buffered voltage	selectable by front jumper setting Vin -1V: 23 - 27.8V (variable threshold) 22.5V fixed: 22.5V (fixed threshold)
Reversed power immunity	max. +35V
Protection against polarity reversal	max. -35V
Charging current	<600mA
Buffering current	0...20A
Current limitation (Buffer operation)	>20A
Charging time	18...27s (primary charge)
Hold-up time • minimum • typical	see diagramm (page 2) 0,2s (22,5V/20A) or 28s (22,5V/100mA) 0,31s (22,5V/20A) or 43s (22,5V/100mA)

Short Overview - Technical Data

Idling input current	typ. 80mA
Power dissipation	typ. 1.9W
Degree of protection	IP20 (EN 60529)
Dimensions (W x H x D)	64mm x 124mm x 102mm (without DIN rail)
Weight	740g

Safety

Terminal voltage	SELV, IEC/EN 60950
Classification	PELV (IEC364-4-41) PELV (EN50178) PELV (EN 60204)
Isolation resistance	5MOhm (terminal→housing)
Degree of protection	IP20 (EN 60529)
Penetration protection	> 3.5 x 3.5 mm
Internal fusing	none
Galvanic isolation to signal path	500V

Order information

Order number	Description
SLV20.200	DIN rail electrolytic capacitor buffer unit
XF-1x4s/270-60	Mating connector for signalling terminals (part of delivery)
SLZ11	Adapter for S7-300 rail
SLZ02	Wall mounting set (two pcs. per package)

Technical Data

Buffer Charging	
Charging delay time	typ. 4s
Charging current	0.4...0.6A
Charging time	18...27s (primary charge / cold start)
Buffer Operation	
Rated output current	20A
Current limitation	>20A
Hold-up time • minimum • typical	see diagramm (page 2) 0,2s (22,5V/20A) or 28s (22,5V/100mA) 0,31s (22,5V/20A) or 43s (22,5V/100mA)
To increase buffer current and/or extend hold-up time any given number of buffer unit can be switched parallel (max. load per terminal = 30A)	
Activation threshold	
"22.5V fixed"	Buffering starts if terminal voltage <22.5V, voltage is kept at 22.5V.
"Vin -1V"	Buffering starts if terminal voltage decreases by more than 1V, faster than typ. 0.54V/s. Voltage is kept at that level. Buffering ends when voltage increases once more by 1V.
Noise (spikes)	<200mV$_{pp}$ (20MHz bandw., 50Ω–measurement, buffer operation only)
Over voltage protection	limited to max. ±35V
Operation indicator	Green LED (see below table 'Operating modes')

Environmental Data

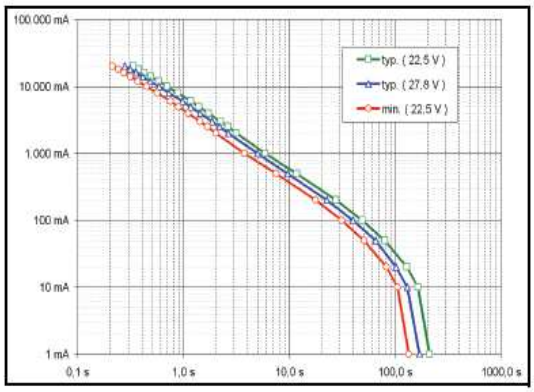
Temperature • Storage/Transport • Operation • Derating • Cooling	 -25°C...+85°C -10°C...+70°C (measured at 25mm below the unit) not necessary natural convection
Humidity	5...95% (condensation not permissible)
Vibration • Sinus • Random	 2 – 17.8Hz: ±1.6mm 17.8Hz – 500Hz 2g (IEC 60068-2-6) 2...500Hz 0.5m^2 (a^3) (IEC 60068-2-64)
Shock	15g/6ms and 10g/11ms (IEC 60068-2-27)
Degree of pollution	2 (EN 50178)
Installation level	2.000m above sea level

Reliability

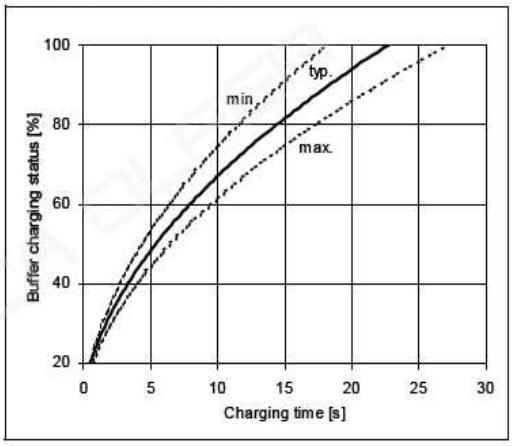
MTBF	480.000h t.b.c. (unit on stand-by, T_{amb} = +40°C)
Life time	>42.000h calculated life expectancy (T_{amb} = +40°C)

Note: t.b.c. = to be calculated (data will follow)

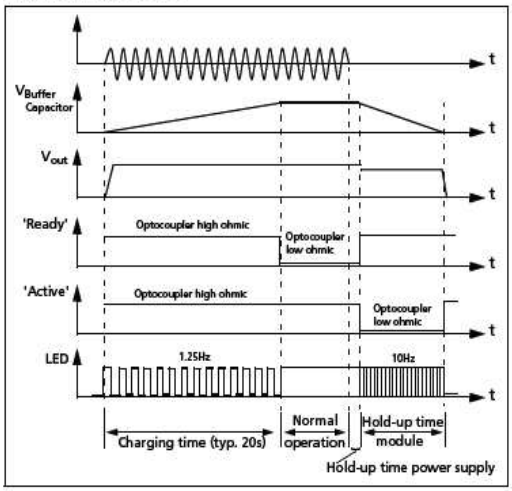
Hold-up time



Buffer charging time



Operating modes



Operating modes

	Current	Time	Status LED	Output 'Active'	Output 'Ready'	Bulk capacitor array
Buffer charging	400...600mA	18-27s	flashes 1.25Hz	blocking	blocking	charging
Stand-by	80mA	./.	steady light	blocking	low ohmic	fully charged
Buffer operation	0...20A	see diagramm hold-up time	flashes 10Hz	low ohmic	blocking	discharging
Inhibit mode	15mA	./.	off	blocking	blocking	discharged
Unit not ready	15mA	./.	off	blocking	blocking	discharged

Connections

Terminals	Fingertouch-proof terminals with captive screws for 5.5mm slotted screwdriver or Philips cross-recessed screwdriver No. 2
Positioning	Easy to reach terminals on the front panel. Signal connectors and powers terminals are clearly separate from each other.
Tightening torque	0.7Nm recommended
Connector size range	
• solid	0.5 ... 6mm^2 20AWG ... 10AWG
• flexible	0.5 ... 4mm^2 20AWG ... 12AWG
Ferrules	admissible
Stripping length	7mm

Front Elements, Operating Indicators and Elements

⊕	Positive power in/out (twice)
⊖	Negative power in/out (twice)
Chassis Ground ⏚	Possibility to connect housing to ground
'Back-up Threshold'	
• Jumper pos. 2-3 (or missing)	Backup voltage: DC 22.5V fixed
• Jumper pos. 1-2	Backup voltage, variable: V_{in} -1V; backup activation on drop faster than typ. 0.54V/s and >1V
LED 'Status'	
• Off	Buffers are discharged, no external voltage or external voltage <22.5V
• Flashes (1.25Hz)	Buffer capacitors are charging
• On	Unit ready for operation, buffer is fully charged
• Flashes (10Hz)	Unit is buffering

Electromagnetic Compatibility(EMC)

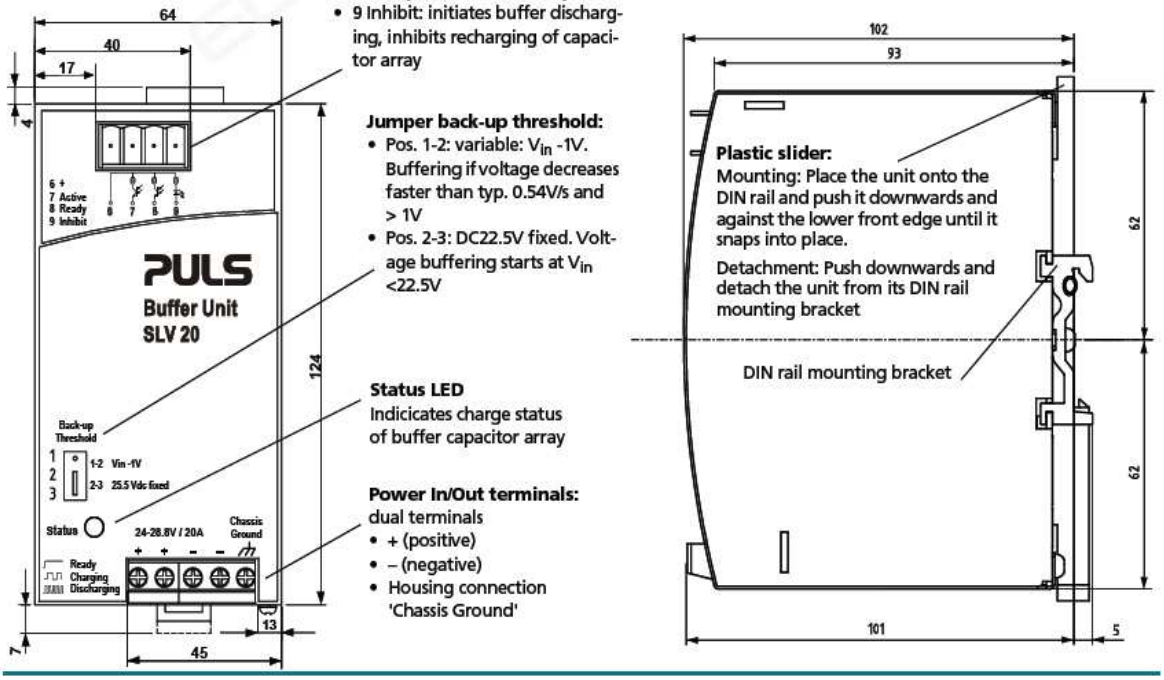
Emissions	EN 61000-6-3 (also includes EN 61000-6-4) radiated noise and interference voltage on DC lines
Immunity	EN 61000-6-2 (also includes EN 61000-6-1)
• Electrostatic Discharge (ESD)	EN 61000-4-2, Level 4 (withstands 8kV direct discharge, 15kV air discharge; DIN rail earthed)
• Electromagnetic radiated fields	EN 61000-4-3, Level 3 (10V/m) ENV 50204 (10V/m)
• Burst, coupled to: – DCout lines	EN 61000-4-4, Level 3 (2 kV)
• Surge transients – Differential mode (+→housing, –→housing)	EN 61000-4-5 500V 500V
– Common mode (+→–)	
• Conducted noise immunity	EN 61000-4-6, Level 3 (10V, 150kHz - 80MHz)

Approvals and Declarations of Conformity

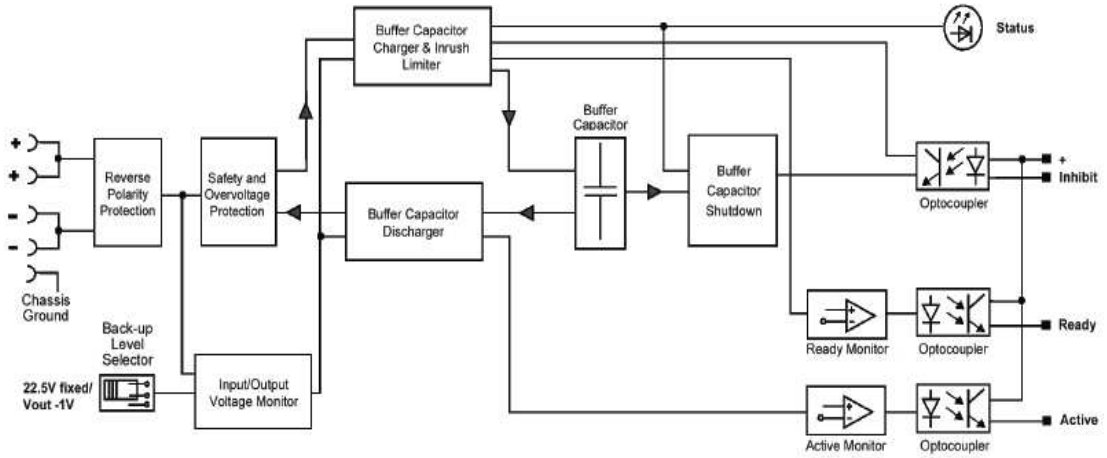
The unit complies with all major **safety approvals**:
EU (EN 60950), USA (UL 60950 recognized, UL 508 LISTED), CBscheme (IEC 60950), Canada (CAN/CSA-C22.2 No. 60950 [cUR], CAN/CSA-C22.2 No. 14 [cUL])

This unit has the following **declarations of conformity**:
Europe (CE acc. to EMC and low voltage directive)

Operating indicators and elements



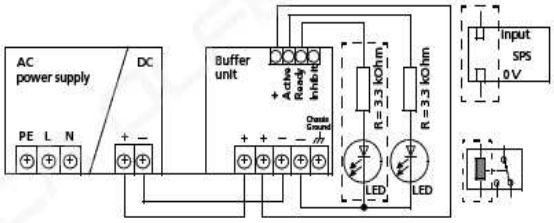
Schematic



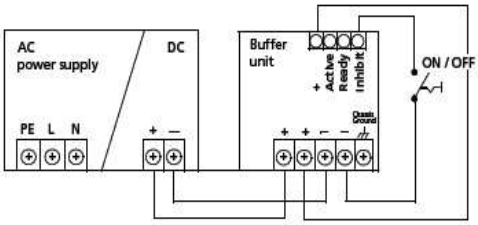
Signalling Terminals

Shared ⊕signal → signal (e.g. Inhibit)	DC 35V max.
Signal outputs • 7 – Active • 8 – Ready	Optocoupler low ohmic, while buffer capacitors are discharging low ohmic, when buffer is fully charged
Current	10mA max. permissible
Voltage drop across opto coupler	0.9V/1mA...3V/5mA (while low ohmic)
Leakage current	<100µA (while optocoupler blocks)
Signal input • 9 – Inhibit	Optocoupler 'High' input signal initiates unit shutdown and buffer discharge
Shutdown threshold	>7...10V
Input current	<4mA
Isolation voltage	AC 500V against power path

Signal outputs and control input are protected against short-circuit, open circuit and overload.



Signalling output variants:
- LED + R = 3.3kOhm (see above)
- Relay (R_L = 2kOhm)
- SPS input



Installation Notes

Mounting position	vertical; power in/out terminals below, signal terminal above

Admissible area of application: The buffer unit SLV20.200 has been designed for use in panel-board installations or other building-in applications where a suitable mechanical enclosure shall be provided to fulfill the requirements for shock-hazard protection and/or protection from hazardous energy levels as well as for fire protection.

Unless otherwise stated, specifications are valid for 'Ready' state, DC 24V input voltage and +25°C ambient temperature. They are subject to change without prior notice.

CONTENIDO