UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE TECNOLOGÍA

CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

TRABAJO DE APLICACIÓN

"IMPLEMENTACIÓN DE UN SISTEMA DE COBRO DE PRODUCTOS CON TARJETAS RFID Y LECTOR DE CÓDIGO DE BARRAS MEDIANTE PIC18F2550 PARA SUPERMERCADOS"

Examen de Grado presentado para obtener el Grado de Licenciatura en Electrónica y Telecomunicaciones

POSTULANTE: EFRAIN EDGAR QUISBERTH COLQUE

La Paz – Bolivia

Noviembre, 2021

Dedicatoria

Este trabajo de aplicación va dedicado a mis queridos padres Eusebio y Teodora quienes con su amor, paciencia y esfuerzo me han permitido llegar a cumplir hoy un sueño más, gracias por inculcar en mí el ejemplo de esfuerzo y valentía, de no temer las adversidades.

A mi esposa, que con su apoyo alcanzas de mejor manera tus metas, a través de sus consejos, de su amor, y paciencia me ayudo a concluir esta meta.

Agradecimientos

Quiero expresar un sincero agradecimiento, en primer lugar, a Dios por brindarme salud, fortaleza y capacidad; también hago extenso este reconocimiento a todos los docentes de mi educación superior, quienes me han dado las pautas para mi formación profesional.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO I

1.1. Introducción

RESUMEN

RFID es, sin duda una de las tecnologías de auto identificación que ha experimentado un crecimiento más acelerado y sostenido en los últimos tiempos. Las posibilidades que ofrece la lectura a distancia de la información contenida en una etiqueta, sin necesidad de contacto físico, junto con la capacidad para realizar múltiples lecturas (y en su caso, escrituras) simultáneamente, abre la puerta a un conjunto muy extenso de aplicaciones en una gran variedad de ámbitos, desde la trazabilidad y control de inventario, hasta la localización y seguimiento de personas y bienes, o la
seguridad en el control de accesos.
Consiste en aplicar la radio frecuencia para la identificación, por lo que nos permite identificar objetos mediante ondas de radio. Es un paso hacia delante para las tecnologías de identificación automática y una clara alternativa a sistemas tradicionales de control y rastreo de objetos o
personas.

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

El presente trabajo consiste en la implementación de un sistema de cobros, que permitirá administrar tarjetas RFID a través de información escrita en cada tarjeta donde se registrará el nombre, CI y saldo que permitirá a través de un lector de tarjetas hacer una

resta al saldo total registrado en la tarjeta para hacer el cobro del total de productos registrados.

En el desarrollo del trabajo se realiza un sistema de ventas donde se registrará los productos organizados por código, nombre y precio, en una interfaz de POS (Punto de venta) se hará la interfaz con un lector de código de barras para la detección del código de producto.

1.1. Planteamiento del problema

Una de las actividades que realizan en la compra de un objeto en un supermercado es el cobro donde se necesita dos actores el cliente y vendedor, esta actividad puede de alto tráfico cuando se tienen muchos clientes buscando comprar por lo tanto se requieren más vendedores que registren y cobren los productos lo que hace esta actividad sea morosa.

1.2. Objetivos

1.2.1. Objetivo general

Implementar un sistema de cobro de productos con tarjetas RFID y lector de código de barras mediante el pic18f2550 para supermercados

1.2.2. Objetivos específicos

- Realizar el diseño de sistema de venta de productos con módulos de productos, clientes, venta, reportes.
- Implementar una interfaz de registro de productos mediante un lector de código de barras.
- Implementar una interfaz de lectura y escritura de tarjetas RFID para el cobro del total de productos registrados.
- Realizar la codificación de los valores escritos en las tarjetas.

1.3. Justificación

La implementación del sistema permitirá acelerar proceso de cobro en un supermercado con el manejo de las tarjetas RDIF para el cobro de los productos, al tener un sistema de escaneo de productos por código de barras y un receptor de tarjetas RFID se podrá darle al cliente un control total del pago de sus productos.

El sistema planteado requiere un menor costo en la inversión de equipos que requiere el proceso de cobro, utilizando microcontroladores PIC, módulos RFID y módulos de lectura de barras que se pueden adquirir a menor costo.

1.4. Delimitación

1.4.1. Temática

El sistema de cobro está limitado por las tarjetas RFID que permiten pagar en un solo supermercado, debido a infraestructura que posee el sistema que se limita por el servidor de base de datos que será de manera local.

1.4.2. Temporal

El sistema de cobro tiene estimado un tiempo de 2 meses para su implementación, desde el análisis hasta la fase de pruebas y puesta en marcha.

1.4.3. Espacial

Se tiene como limitación geografía a las ares donde se encuentra un supermercado, que son mercados más organizados que se encuentran en zonas de alta densidad demográfica.

CAPÍTULO 2

Fundamentación teórica

2.1. Antecedentes

La tecnología RFID no tiene una historia ni un descubridor claro. A lo largo de los años desde principios del siglo XX han surgido diferentes aportaciones por parte de numerosos investigadores y gracias a la aplicación de avances en otros campos tecnológicos se ha llegado a la tecnología RFID que conocemos hoy en día. La generación y la transmisión de ondas de radio se constituyen el concepto de sistemas de identificación por radiofrecuencia.

La historia más novelesca se remonta al finalizar la Segunda Guerra Mundial, cuando Léon Theremin creó una herramienta de espionaje para la Unión Soviética que se insertó en una figura regalada al embajador estadounidense en la Unión Soviética que situó en su despacho. Este invento consistía en una antena unido a un cilindro que funcionaba como micrófono. El mismo captaba las señales de un transmisor, ubicado a las afueras de la embajada y sostenido por soldados soviéticos que lo apuntaban en dirección a la figura, la cual recibía la señal y transmitía el sonido de las voces en la sala.

En el año 1960, el sello fue presentado en las Naciones Unidas como prueba irrefutable de que los soviéticos habían estado espiando a los estadounidenses y posteriormente fue exhibido en el Museo Criptológico Nacional de la NSA, en donde aún se encuentra hoy en día.

A pesar de la historia, la patente de esta tecnología es la 3.713.148 de Estados Unidos y corresponde a Mario Cardullo en 1973. Este fue el primer antepasado verdadero de la tecnología RFID moderna; un transpondedor de radio pasivo con memoria.

Se puede decir que la base del RFID fue la combinación entre la tecnología de radiodifusión y el radar. Pues, el proceso se define en un transmisor que envía una señal la cual es reflejada de vuelta por un transpondedor en el sistema RFID pasivo, o bien este transpondedor emite una respuesta mediante una señal en el sistema activo RFID. Los desarrollos científicos durante años y los cambios de modelos comerciales han ido transformando los sistemas de RFID hasta llegar a la tecnología actual.

2.2. Definición y conceptos del RFID

2.2.1. ¿Qué es RFID?

RFID son las siglas en ingles de Radio Frecuency IDentification, una tecnología similar, a la de identificación por código de barras, pero que utiliza ondas electromagnéticas o electrostáticas para la transmisión de la señal que contiene la información.

La identificación por radiofrecuencia es una tecnología de captura e identificación automática de información contenido en etiquetas electrónicas. Cuando estas etiquetas entran en el área de cobertura de un lector RFID, este envía una señal para que la etiqueta le transmita la información almacenada en su memoria, habitualmente un código de identificación. Una de las claves de esta tecnología es que la recuperación de la información contenida en la etiqueta se realiza sin necesidad de que exista contacto físico o visual entre el dispositivo lector y las etiquetas, aunque en muchos casos se exige una cierta proximidad de esos elementos.

2.2.2. ¿Qué es un sistema RFID?

Un sistema RFID se basa en un lector/escritor y un tag o transponder. El objetivo principal es transmitir datos mediante el tag (dispositivo portátil) que es leído por el lector RFID. Los datos transmitidos dependen de cada aplicación, pero pueden pasar información sobre la localización del producto u otra información más específica.

¿Qué es un tag o transponder?

Un tag o transponder es un componente pasivo, sin batería, con un circuito integrado (chip) y una antena.
También hay tags activos que poseen una batería integrada de por vida. Este tipo de tag permite recepcionar y transmitir la información a grandes distancias.
Los tags tienen una memoria interna que cambia según el modelo, de unas decenas a unos miles de bytes. Se clasifican en dos tipos:

Solo lectura: el contenido del código es único y durante la producción se personaliza.

Lectura y escritura: toda la información almacenada en el tag puede ser modificada por el lector.

¿Qué es un lector/escritor?

Un lector tiene un circuito integrado que emite energía electromagnética a través de su antena y una electrónica encargada de recibir y descodificar la información que recibe del tag. El objetivo es enviar la información al sistema de captura de datos.

2.2.3. Frecuencias de funcionamiento

La tecnología RFID se puede clasificar según las diferentes frecuencias de radio que usan. Cada una de ellas tiene sus propias características y sirven para un determinado sector de aplicación:

- Baja
- Alta
- Ultra-alta
- Activa

Frecuencia	LF 120 ~ 134 KHz	HF 13.56 MHz	UHF 850 ~ 960 MHz
Distancia de lectura	0,5 ~ 1 m	< 1 m	>3 m
Coste	Alto	Medio	Bajo
Penetración en materiales	Excelente ⟷ Pobre		
Le afecta el agua?	No	No	Sí
Tipo de antena	Bobina inductiva	Bobina inductiva	Dipolo, "plancha metálica" (slot)
Transmisión de datos (data rate)	Más lento ⟷ Más rápido		
Anticolisión (lectura de múltiples tags)	Pobre	Buena	Muy buena
Aplicaciones	Control de acceso, identificación industrial, llaves de acceso a vehículos, automatización, auto guiado de vehículos	Farmacia, librerías, control de acceso, fidelización, aplicaciones de pago RFID, NFC, pasaporte	Trazabilidad de paquetes/pallets, trazabilidad de producto, automatización industrial, control de acceso de vehículos

Tabla 1. Tabla de Frecuencias RFID

Fuente: https://www.kimaldi.com/wp-content/uploads/2019/05/Tabla-tipos-de-frecuencia-1024x233.png

2.2.4. Ventajas de la tecnología RFID

- Esta etiqueta se puede leer a través de muchos materiales, como la pintura (prácticamente todos, salvo el metal o el agua, aunque ya existen etiquetas lavables), algo que no se puede hacer con los códigos de barras convencionales.
- Uso de la tecnología RFID para cualquier sector.
- Permite identificar y controlar objetos en una cadena de suministros.
- Integración de la tecnología RFID con otros sistemas como vídeo y sistemas de localización.
- Lectura rápida y precisa de inventario.
- Reducción de roturas de stock.

- Control de la reposición en estanterías y por lo tanto reducción por mala colocación de los productos.
- Seguimiento óptimo de la mercancía en palets y carretillas.
- Control de los productos retirados del mercado.

2.2.5. Tipos de tecnología RFID

2.2.5.1. Tecnología Mifare

La tecnología Mifare es la más conocida a nivel mundial y las más extendida. Tiene capacidad de lectura y escritura y su frecuencia de trabajo es de 13,56 MHz.
Esta tecnología respeta las normas ISO referentes a las tarjetas de proximidad y dispone de varios modelos dependiendo de la cantidad de información a almacenar.

Los datos almacenados en la tarjeta están protegidos. La tarjeta está dividida en sectores, bloques y mecanismos para dar seguridad en el momento de la identificación.
El proceso de lectura de la tarjeta con el lector consiste en acercar la tarjeta al lector. Ésta se activa e inicia el intercambio de información con el lector a través de una comunicación cifrada. El proceso de lectura entre tarjeta y lector es el mismo para todo tipo de tarjetas y aporta protección a la información almacenada.

Una de las aplicaciones más habituales con tecnología Mifare es el control de acceso en empresas o edificios. Aquí la tarjeta Mifare es utilizada como tarjeta identificativa del trabajador o usuario.

Otro ejemplo del uso de tarjetas Mifare es para el transporte público. En los últimos años el sistema de fichaje para el transporte público está cambiando por tarjetas plásticas y dejando el método de los tickets de papel. Este cambio aporta ventajas como la reducción de tiempo al no imprimir los tickets, reducción de las colas y el poder grabar cualquier tipo de información del usuario en la tarjeta.

Esta tecnología es una de los más seguras, fiables y rápidas del mercado.

2.2.5.2. Tecnología Mifare Ultralight

Esta tecnología está especialmente indicada para las aplicaciones que requieran un bajo coste en inversión y en mantenimiento.
Mifare Ultralight tiene una capacidad de memoria de 512 bits y sigue la normativa ISO 14443.
El rango de lectura es hasta 10cm y no necesita batería. Las principales diferencias respecto la tecnología Mifare son que solo tiene 512 bits de memoria, no dispone de seguridad y el coste es menor.
Las aplicaciones más comunes son las tarjetas de fidelización, tarjetas de acreditación para estadios y ferias, tarjetas para el transporte público.

2.2.5.3. Tecnología NFC

NFC (Near Field Communication) es una tecnología de comunicación de corto alcance con una frecuencia de 13,56 MHz.
Este tipo de tecnología sirve para varias aplicaciones:

- Realizar transacciones: pagos como el transporte urbano, aparcamiento público.
- Intercambiar contenido digital: acceder a información y servicios.
- Conectar dispositivos electrónicos.

Tiene dos modos de funcionamiento:

- Activo: los dos dispositivos generan su propio campo electromagnético para poder pasar los datos.
- Pasivo: solo uno de los dispositivos genera el campo electromagnético y el otro aprovecha la carga para transmitir los datos.

El método para comunicarse entre dos dispositivos es a través del envío de una señal por parte del dispositivo iniciador y una respuesta por parte del dispositivo de destino donde tienen dos antenas colocadas de igual modo que el RFID. La frecuencia utilizada es de 13,56 MHz, sin restricciones ni licencias de uso.

La tecnología NFC sirve para transmitir pocos bits de información de manera rápida para que pueda identificar y validar al usuario.

Existen 4 tipos de tags que dependiendo de la configuración, memoria, seguridad, retención de datos y resistencia de escritura proporcionan diferentes velocidades de comunicación y capacidades.

Los dispositivos compatibles con los tags NFC son los chips NXP Mifare Ultralight / Mifare Ultralight C, NXP Desfire. Todos estos chips están hechos bajo el estándar NFC.

2.2.5.4. Tecnología RFID Activa

La tecnología RFID activa transmite la información a través de señales de radiofrecuencia emitidos por tags activos. Esta tecnología emite la señal de radiofrecuencia por un tag activo y es leído por un lector RFID activo. Ambos dispositivos tienen la misma frecuencia.

Los tags activos son aquellos que se alimentan independientemente de una batería.
La batería juega un papel importante ya que proporciona energía a los tags para transmitir la información a través de la antena del tag. Esta información incluye el código de identificación del tag activo y otra adicional, como el nivel de la batería para cambiarla antes de agotarse.

El principal uso de este tipo de tecnología es para identificar personas y objetos.
El sistema para identificar al usuario es con el tag activo que posee. El tag es captado por

el lector si está dentro del rango de alcance de lectura y lo identifica. Cada uno de los tags emite un código único y así puede asociar cada tag a una persona u objeto.

2.2.6. Tipos de etiquetas

Las etiquetas RFID, que están disponibles en una amplia gama de estilos y de materiales para satisfacer cualquier uso, pueden ser clasificadas en diferentes formas

Activa/Pasiva: una etiqueta active usa las propias baterias que lleva incorporadas (por lo que es de gran tamaño), mientras que una pasiva no, ya que emplea la energía recibida de la antena lectora para transmitir sus datos.

La consecuencia directa de este hecho es que las etiquetas pasivas son de un coste mínimo y son más pequeñas; podrán contar con un rango más bajo de lectura, pero también cuenta con una vida teóricamente indefinida (pueden durar hasta 30 años, frente a los no más de 10 que dura una activa)

2.3. Lector de código de barras

Un lector de códigos de barras es un dispositivo electrónico que por medio de un láser lee el código de barras y emite el número que muestra el código de barras, no la imagen. Básicamente, consiste en el escáner propiamente dicho (que mediante un láser lee el código), un decodificador y un cable o antena wifi que actúa como interfaz entre el decodificador y el terminal o la computadora.

La función del escáner es leer el símbolo del código de barras y proporcionar una salida eléctrica a la computadora, correspondiente a las barras y espacios del código de barras. Sin embargo, es el decodificador el que reconoce la simbología del código de barras, analiza el contenido del código de barras leído y transmite dichos datos a la computadora en un formato de datos tradicional. Tiene varios medios de conexión: los más modernos por orden de aparición USB, bluetooth, wifi, los más viejos puerto serie, incluso

directamente al puerto PS2 del teclado por medio de un adaptador, cuando se pasa un código de barras por el escáner es como si se hubiese escrito en el teclado el número del código de barras.

Figura 1. Lector de código barras

Fuente: https://upload.wikimedia.org/wikipedia/commons/thumb/1/13/Barcode-scanner.jpg/250px-Barcode-scanner.jpg

Un escáner puede tener el decodificador incorporado en el mango o puede tratarse de un escáner sin decodificador que requiere una caja separada, llamada interfaz o emulador. Los escáneres sin decodificador también se utilizan cuando se establecen conexiones con escáneres portátiles tipo “batch” (por lotes) y el proceso de decodificación se realiza mediante el terminal propiamente dicho.

El lector de códigos de barras fue, en 1974, la primera aplicación comercial del láser. El primer registro fue el precio de un empaque de chicles.

Lectura de código de barras

Los códigos de barras se leen pasando un pequeño punto de luz sobre el símbolo del código de barras impreso. Solo se ve una fina línea roja emitida desde el escáner láser. Pero lo que pasa es que las barras oscuras absorben la fuente de luz del escáner y la misma se refleja en los espacios luminosos. Un dispositivo del escáner toma la luz reflejada y la convierte en una señal eléctrica.

El láser del escáner (fuente de luz) comienza a leer el código de barras en un espacio blanco (la zona fija) antes de la primera barra y continúa pasando hasta la última línea, para finalizar en el espacio blanco que sigue a ésta. Debido a que el código no se puede leer si se pasa el escáner fuera de la zona del símbolo, las alturas de las barras se eligen de manera tal de permitir que la zona de lectura se mantenga dentro del área del código de barras. Mientras más larga sea la información a codificar, más largo será el código de barras necesario. A medida que la longitud se incrementa, también lo hace la altura de las barras y los espacios a leer.

2.4. Lector RFID RC522

El lector RFID RC522 utiliza la alta frecuencia HF creando un campo electromagnético de 13,56 MHz. En teoría tiene un alcance máximo de 35 centímetros.

El módulo RC522 tiene un alcance máximo de 5 cm. Por encima de este valor puede que el sistema no funcione correctamente.

El lector RFID RC522 puede comunicarse con un microcontrolador a través de un bus de interfaz de periféricos serie o bus SPI (del inglés Serial Peripheral Interface) con una velocidad de datos máxima de 10 Mbps.

También es compatible con la comunicación a través del protocolo I2C y UART.

Algo muy interesante es que este módulo viene con un pin de interrupción muy útil. Sirve para que en vez de preguntar una y otra vez al lector RFID RC522 si hay una etiqueta RFID cerca, el módulo nos avisará a través del pin cuando una etiqueta RFID se acerque y así poder activar el microcontrolador.

El voltaje de operación es de 2,5V a 3,3V. Esto implica que es compatible con la mayoría de microcontroladores del mercado como los que utilizan Arduino y el ESP8266.

La buena noticia es que a pesar de ser alimentado con 3,3V, los niveles lógicos son compatibles con 5V por lo que es compatible con cualquier Arduino o microcontrolador que trabaje a 5V.

Estas sería las especificaciones completas sacadas de la hoja de características técnicas

Figura 2. Tarjeta, Modulo Lector RFID y Tag RFID

Fuente:https://programarfacil.com/wp-content/uploads/2020/03/modulo-lector-rfid-rc522.jpg

Rango de frecuencias	Banda ISM de 13,56 MHz
Interfaz	SPI/I2C/UART
Voltaje de operación	2,5V a 3,3V
Corriente máx. de funcionamiento	13-26 mA
Corriente mínima	10 uA
Niveles lógicos	5V y 3V3
Alcance	5 cm

Tabla 2. Características RC522

Fuente: https://programarfacil.com

2.4.1. Pineado lector RFID RC522

El lector RFID RC522 tiene 8 pines para conectarse a un microcontrolador.

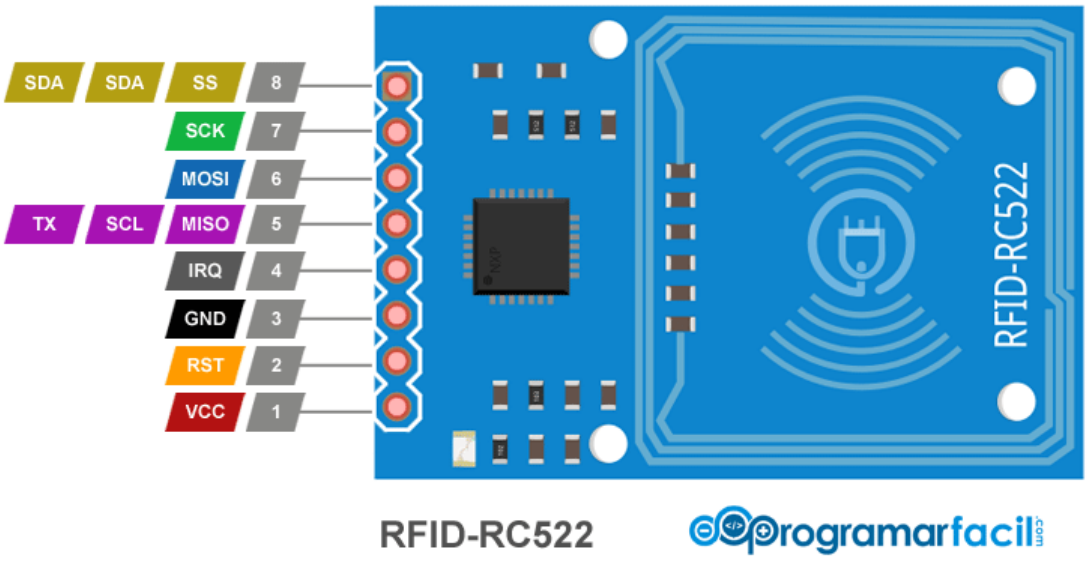


Figura 3. Pines de referencia módulo RFID RC522

Fuente: https://programarfacil.com/wp-content/uploads/2020/03/pineado-rc522-02.png

- **VCC**: pin de alimentación del lector RFID RC522. Admite un voltaje de alimentación entre 2,5V y 3,3V.
- **RST**: es un pin para encender y apagar el módulo. Mientras el pin esté en estado LOW se mantendrá apagado sin apenas consumir. Cuando el estado cambia a HIGH el RC522 se reinicia.
- **GND**: pin de tierra o GND.
- **IRQ**: pin de interrupción que alerta al microcontrolador cuando una etiqueta RFID se acerca al lector RFID RC522.
- **MISO/SCL/TX**: este pin tiene tres funciones. Cuando la interfaz SPI está habilitada funciona como salida de esclavo y entrada de máster. Cuando está activada la interfaz I2C funciona como señal de reloj y como salida serie cuando la interfaz UART está habilitada.
- **MOSI**: entrada en la interfaz SPI.

- **SCK**: señal de reloj de la interfaz SPI.
- **SS/SDA/RX**: el pin actúa como entrada de señal cuando la interfaz SPI está habilitada. Si la interfaz I2C está activa actúa como entrada de datos y como entrada de datos serie cuando la interfaz UART está habilitada.

2.4.2. Estructura de la memoria MIFARE Classic 1K

Las tarjetas o etiquetas RFID que utiliza el lector RC522 tienen una memoria de 1K organizada en 16 sectores (del 0 al 15). Cada sector se divide en 4 bloques (del 0 al 3). En cada bloque se pueden almacenar 16 bytes de datos (del 0 al 15).

Utilizan la tecnología MIFARE Classics. Esta tecnología utiliza un cifrado propietario de la empresa Philips que se llama CRYPTO1.

El problema que existe actualmente es que la seguridad a través de este sistema de cifrado presenta deficiencias.

Según la estructura de la memoria en sectores y bloques, tenemos que

$$16sectores \times 4bloques \times 16bytes = 1024bytes = 1Kb$$

Donde se puede comprobar que efectivamente esta tarjeta tiene 1 Kb de memoria.

Si miras de nuevo el monitor serie, puedes ver los sectores, bloques y datos.

```
Firmware Version: 0x91 = v1.0
Scan PICC to see UID, SAK, type, and data blocks...
Card UID: 61 08 19 34
Card SAK: 08
PICC type: MIFARE 1KB
Sector Block   0  1  2  3   4  5  6  7   8  9 10 11  12 13 14 15  AccessBits
  15     63   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         62   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         61   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         60   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
  14     59   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         58   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         57   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         56   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
  13     55   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
```

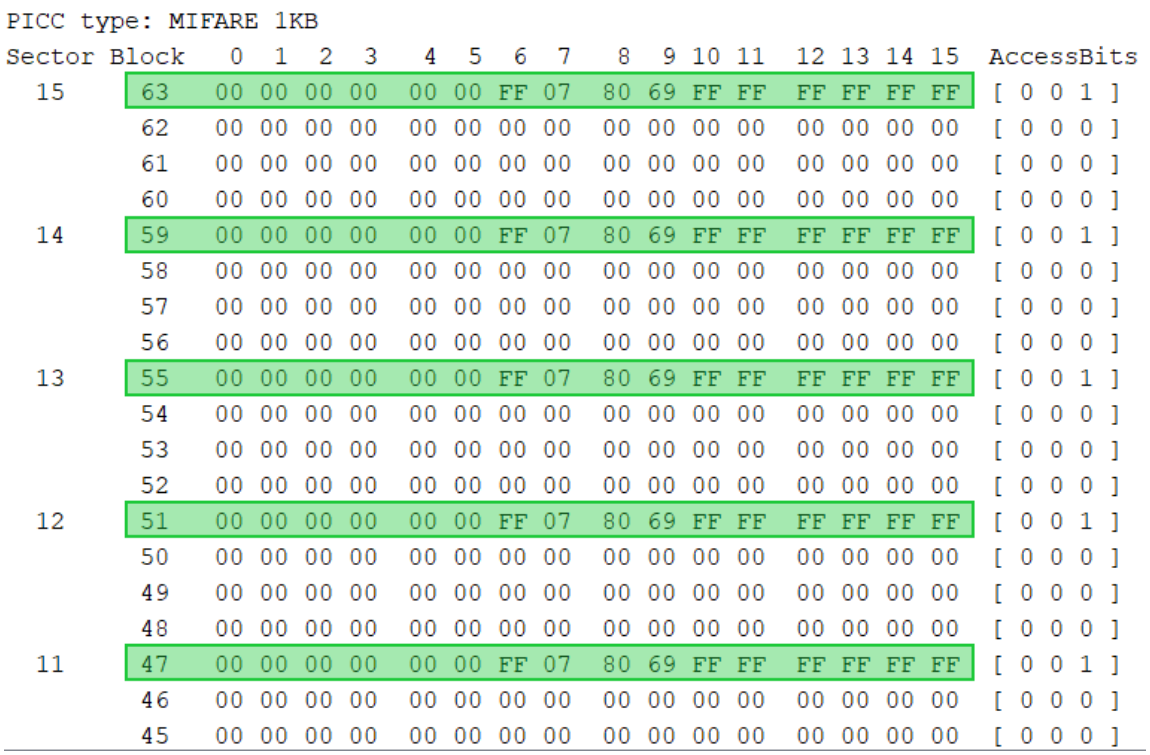
Figura 4. Sectores, Bloques y Datos

https://programarfacil.com/wp-content/uploads/2020/04/mifare-tarjeta-rfid-rc522.png

Como ya he dicho en total hay 16 sectores donde hay 4 bloques y cada bloque tiene 16 bytes.

De estos 4 bloques de cada sector, solo se pueden utilizar 3 bloques para almacenar memoria. Esto equivale a 16 bytes por 3 bloques que hacen un total de 48 bytes.

Los 16 bytes del último bloque se utilizan para guardar las claves de cifrado y los permisos para acceder a ese sector. Este bloque se conoce como trailer y coincide con el último bloque de cada sector.



```
PICC type: MIFARE 1KB
Sector Block   0  1  2  3   4  5  6  7   8  9 10 11  12 13 14 15  AccessBits
  15     63   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         62   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         61   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         60   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
  14     59   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         58   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         57   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         56   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
  13     55   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         54   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         53   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         52   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
  12     51   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         50   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         49   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         48   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
  11     47   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         46   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         45   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
```

Figura 5. Sectores 11,12,13,14,15

Fuente: https://programarfacil.com/wp-content/uploads/2020/04/mifare-tarjeta-rfid-rc522-03.png

Así el bloque 64 es el bloque trailer del sector 15, el bloque 59 es el bloque trailer del sector 14, el bloque 55 es el bloque trailer del sector 13 y así sucesivamente con el resto de sectores.

Dentro de este bloque los 16 bytes que pertenecen al trailer se dividen en dos claves para cifrar el contenido (Key A y Key B) que ocupan 6 bytes cada una. 3 bytes para establecer las condiciones y permisos de acceso al sector (Access Conditions). Y por último un byte restante (U) que no se utiliza para nada.

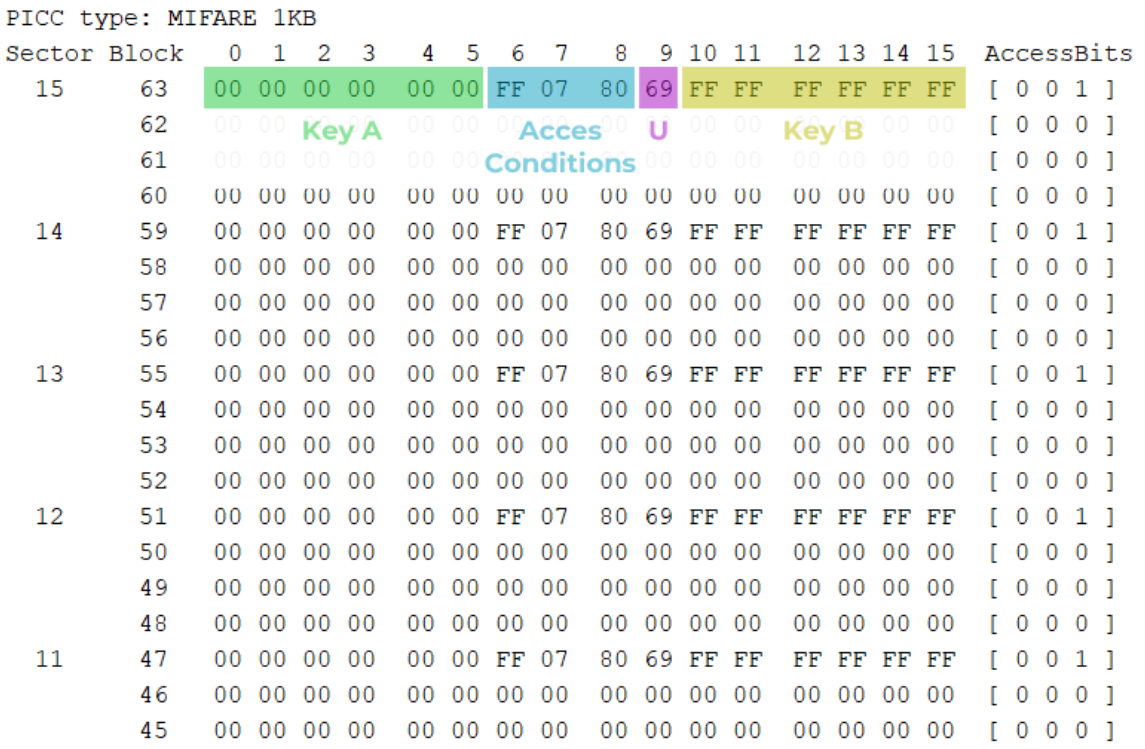


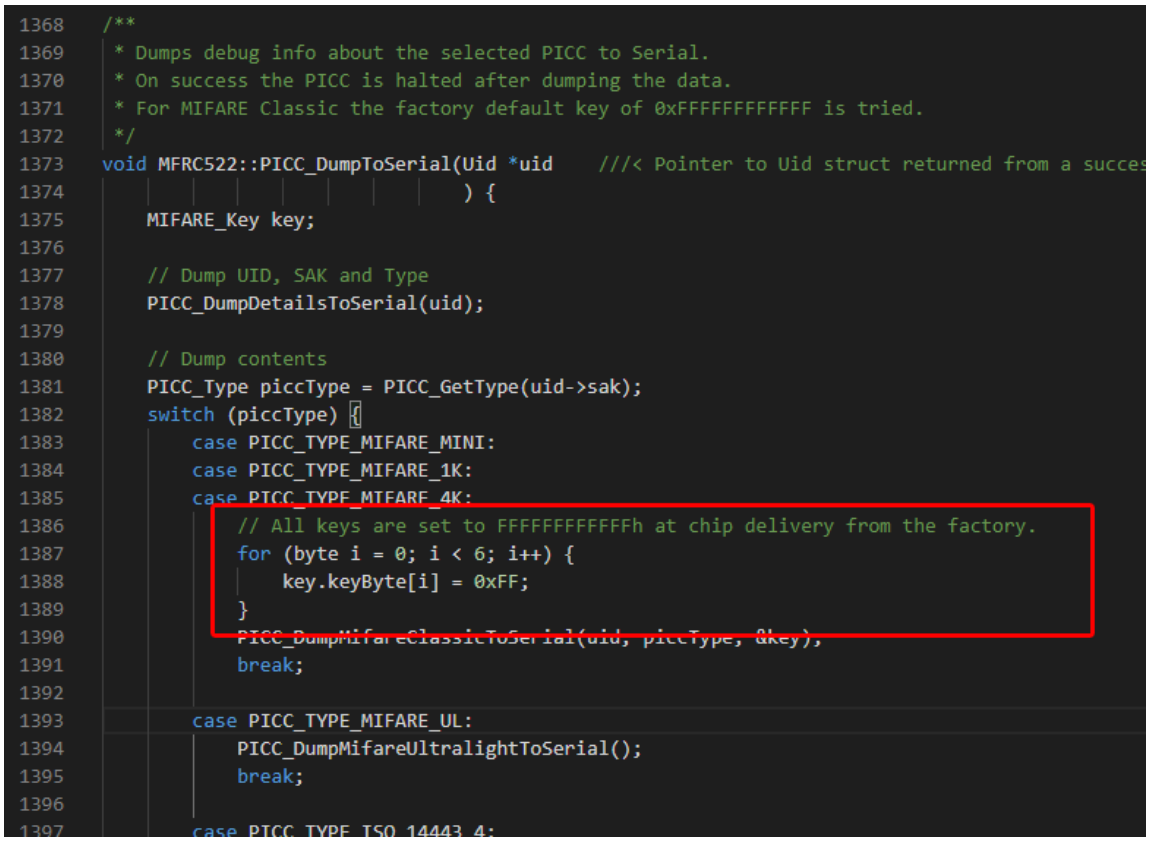
Figura 6. Descripción de un sector

Fuente: https://programarfacil.com/wp-content/uploads/2020/04/mifare-tarjeta-rfid-rc522-04.png

Para cifrar el contenido se puede utilizar o la clave Key A o la clave Key B. Cada sector tiene sus propias claves, aunque una práctica común con la tecnología MIFARE es utilizar las mismas claves en todos los sectores.

Por supuesto que para que todo funcione correctamente el lector necesita conocer la clave que se está utilizando. De no ser así no podrá ni leer ni escribir información.

Las etiquetas RFID que vienen con el módulo RFID RC522 vienen de fábrica con la clave por defecto FF FF FF FF FF FF.



```
/**
 * Dumps debug info about the selected PICC to Serial.
 * On success the PICC is halted after dumping the data.
 * For MIFARE Classic the factory default key of 0xFFFFFFFFFFFF is tried.
 */
void MFRC522::PICC_DumpToSerial(Uid *uid    ///< Pointer to Uid struct returned from a succes
                                ) {
    MIFARE_Key key;

    // Dump UID, SAK and Type
    PICC_DumpDetailsToSerial(uid);

    // Dump contents
    PICC_Type piccType = PICC_GetType(uid->sak);
    switch (piccType) {
        case PICC_TYPE_MIFARE_MINI:
        case PICC_TYPE_MIFARE_1K:
        case PICC_TYPE_MIFARE_4K:
            // All keys are set to FFFFFFFFFFFFh at chip delivery from the factory.
            for (byte i = 0; i < 6; i++) {
                key.keyByte[i] = 0xFF;
            }
            PICC_DumpMifareClassicToSerial(uid, piccType, &key);
            break;

        case PICC_TYPE_MIFARE_UL:
            PICC_DumpMifareUltralightToSerial();
            break;

        case PICC_TYPE_ISO_14443_4:
```

Figura 7. Clave por defecto de una Tarjeta RFID

https://programarfacil.com/wp-content/uploads/2020/04/rfid-rc522-1.png

Si la clave puede ser leída no es segura y por lo tanto no se utiliza. Por eso en la imagen anterior aparece.

Key A: 00 00 00 00 00 00 (no puede ser leída)

Key B: FF FF FF FF FF FF (puede ser leída y no es segura)

Esta configuración se establece en los bytes de Access Conditions.

Por último, quiero hablarte del bloque 0 del sector 0. Este bloque no se utiliza para almacenar datos ya que de fábrica contiene información muy valiosa.

Este bloque se llama el bloque del fabricante (en inglés manufacturer block) y contiene el UID o identificador de la tarjeta y el identificador del fabricante.

El UID es un identificador único que MIFARE asigna a cada etiqueta RFID. Cada tarjeta tiene el suyo propio y consta de 4 bytes de sólo lectura.

Los otros 12 bytes del bloque se utilizan para almacenar información del fabricante.

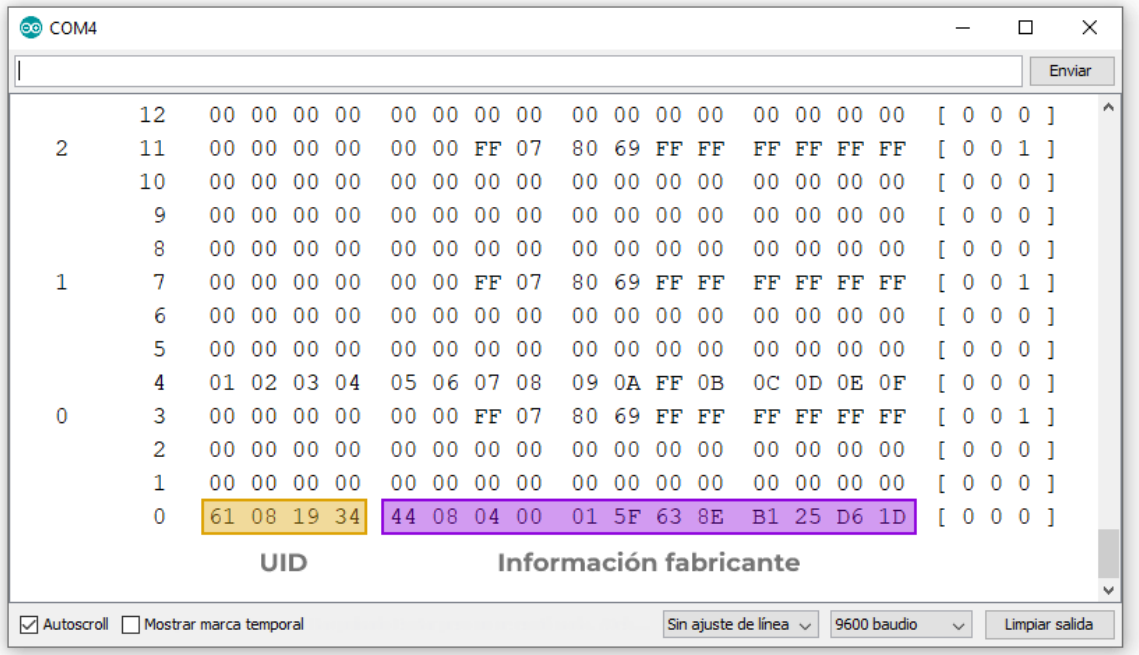


Figura 8. Bloque del fabricante no debe modificarse

Fuente: https://programarfacil.com/wp-content/uploads/2020/04/mifare-tarjeta-rfid-rc522-02.png

El bloque del fabricante no se debe modificar ni sobrescribir ya que esto puede bloquear permanentemente la tarjeta o etiqueta RFID.

2.5. Código de barras CODE 39

También denominado “Code 3 of 9” o “Alpha39,” el código de barras Code 39 fue el primer código en utilizar tanto números como letras. Es un código de barras de longitud variable que puede codificar hasta 43 caracteres alfanuméricos al mismo tiempo. Se utiliza con mayor frecuencia en la industria militar y la industria automotriz.

Especificaciones: El código de barras Code 39 posee un símbolo de inicio y fin para definir el comienzo y el final del código para el escáner, comúnmente representado como un * en fuentes normales. Aparte del carácter de inicio/fin, este código de barras técnicamente solo puede codificar los números 1-10. Pero al utilizar designaciones especiales, es capaz de designar letras, separándolas por categorías. Por ejemplo, a las primeras 10 letras (A-J) se les asigna valores numéricos, precedido por una designación de “letras”. A las 10 siguientes (K-T) se las designa como “+10 letras”. De esta forma, K sería +10 letras, seguido de un 1, indicando que es la 11ª letra del abecedario. U-Z pertenecen a “+20 letras”. Y, por supuesto, los números también poseen su propia designación.

Figura 9. Imagen código de barras en formato CODE39

Fuente:https://www.cognex.com/library/media/resources/symbologies/code39.jpg?h=250&w=447&la=es-CL&hash=A958A0171869B1A2F36B562710A2E35C

Ventajas: La utilización tanto de letras como de números hace que el Code 39 sea más versátil. Además, no requiere un número de comprobación debido a que es de "autocomprobación" (aunque aun así se recomienda tener uno).

Desventajas: Se limita a un máximo de 43 caracteres. Asimismo, su método de asignación de valores numéricos a letras para posibilitar la lectura limita su versatilidad y excluye a otros caracteres.

2.6. Microcontrolador PIC18F2550

El microcontrolador PIC18F2550 es un dispositivo programable que se compone de una computadora digital, una unidad de memoria de datos, una unidad de memoria de programa y puertos de entrada/salida en un circuito integrado, funciona como un controlador de periféricos en un sistema mínimo. El microcontrolador depende de una alimentación de al menos 5V y 0V en sus entradas de Vdd y Vss respectivamente para su operación, requiere de una señal de reloj que le indique la frecuencia de trabajo, está señal la introducimos a través de un oscilador de cristal de cuarzo en los pines OSC1 y OSC2, y una alimentación al pin MCLR, que es un pin de reset que activa al microcontrolador. El funcionamiento del microcontrolador está determinado por un programa almacenado en su memoria Flash ROM y puede programarse más de una vez para cambiar su estado y su comportamiento, lo que lo convierte al microcontrolador en una pieza esencial en el rápido desarrollo de aplicaciones electrónicas.

2.6.1. Características principales del PIC18F2550

Procesador: microcontrolador de alto rendimiento, multifunciones PIC18F2550-I/SP de 48 Mhz, 28 pines encapsulado DIP, de Microchip.

Arquitectura: Harvard, memoria de código de 16 bits, separada de la memoria de datos de 8 bits. Procesamiento "pipeline".

Tecnología: RISC (reduced instruction set computer), con 75 instrucciones.

Puerto USB v2.0: transceptor integrado al microcontrolador. 12 Mb/s

Memoria: 16K localidades de 16 bits de FLASH (ó 32 Kbytes), 2 Kb localidades (8 bits) de RAM, 256 localidades (8 bits) de EEPROM.

Autoprogramación de la memoria FLASH: a través del puerto USB, por medio de un firmware bootloader residente.

Puertos digitales: puerto A de 5 bits, puerto B de 8 bits, puerto C de 8 bits un total de 21 bits programables como entradas o como salidas.

Capacidad de salidas: cada bit de salida puede tomar ("sink"), ó generar ("source"), hasta 25 miliamperes.

Puertos seriales: USART compatible RS232. SSP Puerto serial síncrono con 2 modos de operación: **SPI** (Serial Peripheral Interface, modos Master/Slave) e **I2C** (Integrated, Integrated Circuit. Modo Slave)

Temporizadores: 4 temporizadores de 16 bits. Un generador de PWM

Convertidores A/D: 10 canales, con 10 bits de resolución.

Funciones adicionales: power-on reset, brown out reset, power up timer, watch dog, code protection, sleep (bajo consumo).

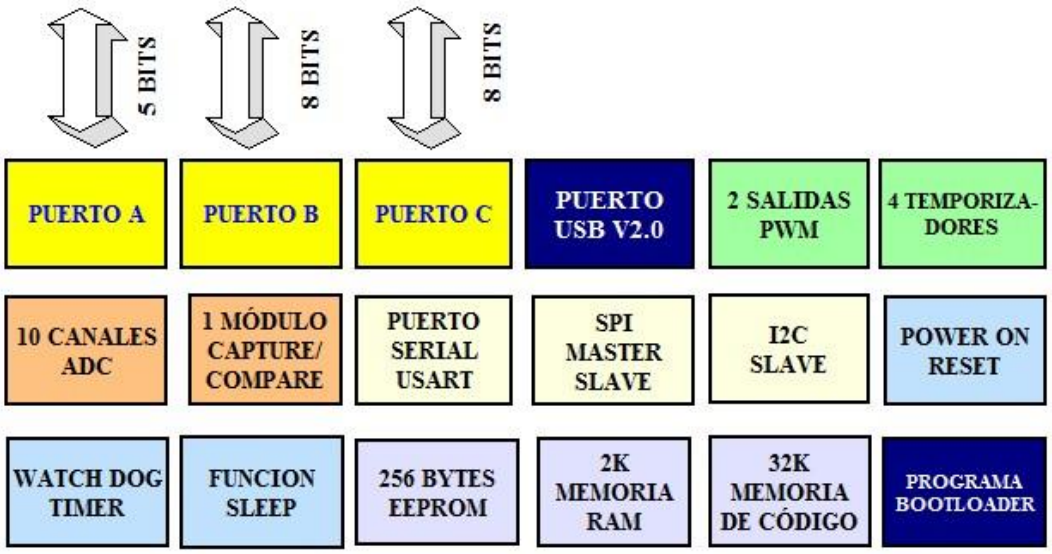


Figura 10. Diagrama de bloques 18F2550

Fuente:https://www.puntoflotante.net/DIAGRAMA%20DE%20BLOQUES%2018F2550.jpg

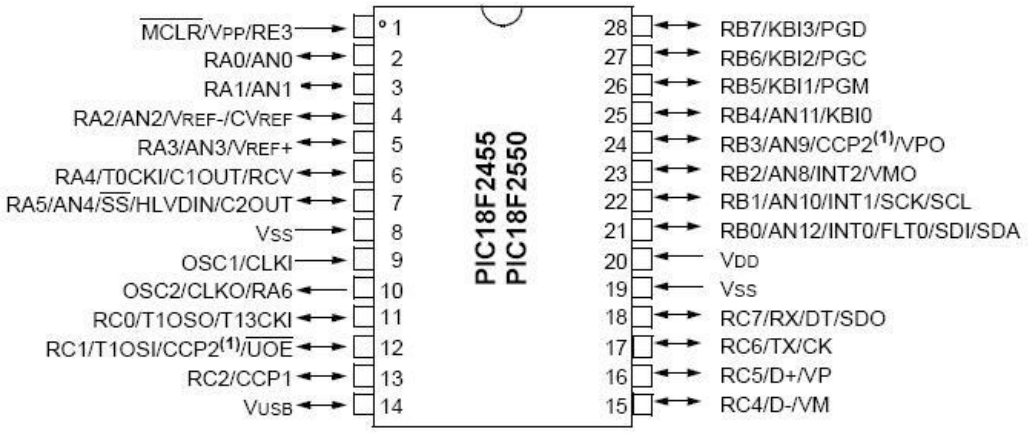


Figura 11. Pines del microcontrolador 18F2550

Fuente:https://www.puntoflotante.net/18F2550%20PINES.jpg

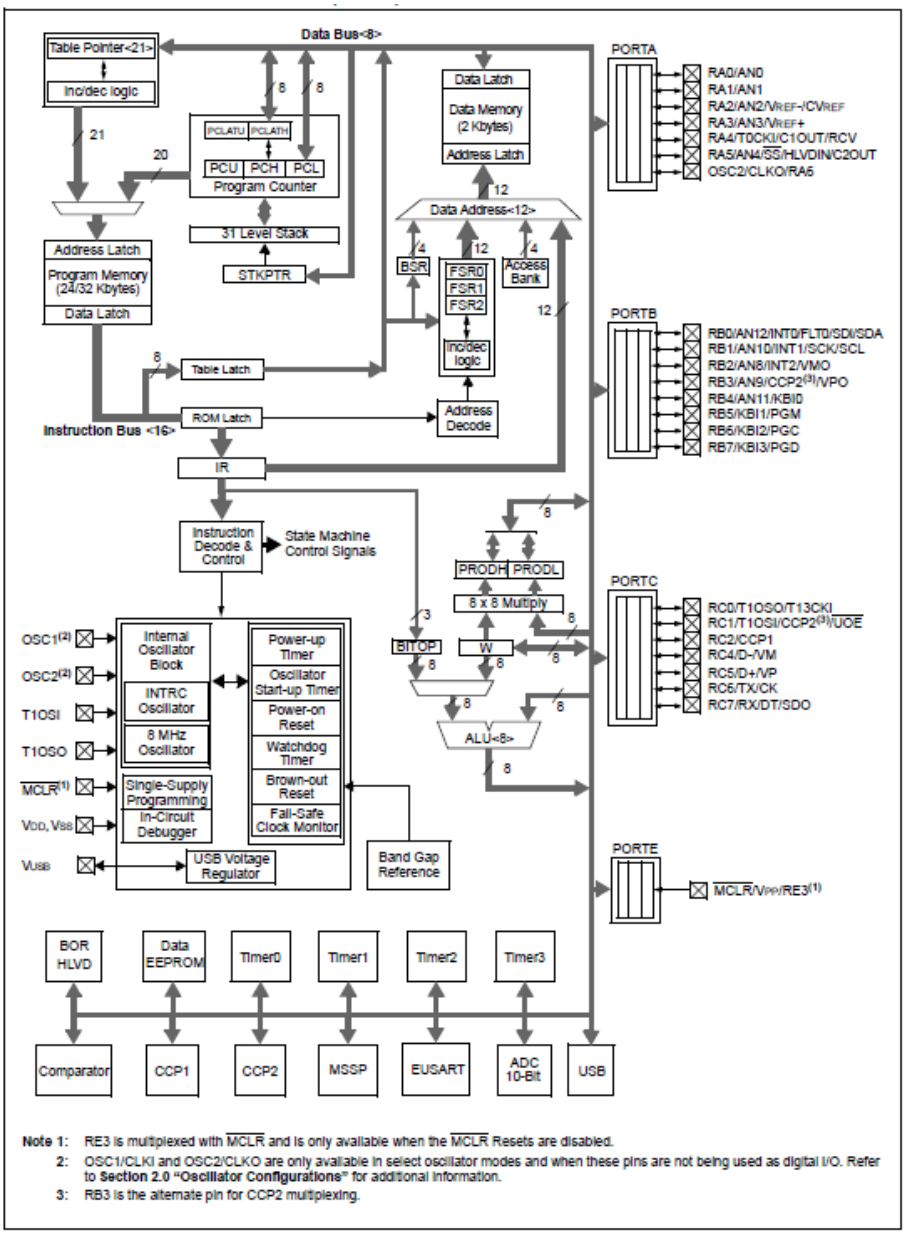
VSS	Tierra
OSC1/CLKIN	Entrada al cristal cuarzo o reloj externo.
OSC2/CLKOUT/RA6	Salida del cristal externo. En modo R-C por este pin sale ¼ FOSC1. E/S propósito general.
RC0/T1OSO/T13CL1	E/S digital del Puerto C. Conexión del oscilador externo para el temporizador TMR1 o entrada de reloj para el TMR1/TMR3.
RC1/T1OSI/CCP2: E/S	Pin de Entrada/Salida. Entrada oscilador TMR1. Entrada módulo Captura2/Salida comparador 2/Salida PWM2. Externa USB
RC2/CCP1	E/S digital del Puerto C. Conexión del oscilador externo para TMR1 o salida del módulo 2 de captura/comparación.
VUSB	Regulador de tensión interna USB
RC4/V-/VM	Entrada digital. Línea diferencial de datos USB. Entrada VM USB.
RC5/D+/VP	Entrada digital. Línea diferencial de datos USB. Salida VP USB.
RC6/TX/CK	E/S digital. Transmisión serie asíncrona. Entrada de reloj para comunicación serie síncrona.
RC7/RX/DT	E/S digital. Recepción serie asíncrona. Línea de datos en la comunicación serie síncrona.
VDD	Entrada del positivo de la alimentación.
RB0/AN12/INT0/FLT0/SDI/SDA	E/S digital o entrada analógica. Interrupción externa 0. PWM entrada. Datos entrada SPI. I2C datos E/S.
RB1/AN10/INT1/SCK/SCL	E/S digital o entrada analógica. Interrupción externa 1. Entrada reloj serie síncrono/salida modo SPI. Entrada reloj serie síncrono/salida modo I2C.
RB2/AN8/INT2/VMO	E/S digital o entrada analógica. Interrupción externa 2. Salida VMO USB.
RB3/AN9/CCP2/VPO	E/S digital o entrada analógica. Entrada módulo Captura2/Salida comparador2/Salida PWM2.Salida VPO USB.

RB4/AN11/KBI0	E/S digital o entrada analógica. Interrupción de cambio de pin.
RB5/KBI1/PGM	E/S digital. Interrupción de cambio de pin. ICSP programador baja tensión.
RB6/KBI2/PGC	E/S digital. Interrupción de cambio de pin. ICSP reloj.
RB7/KBI3/PGM	E/S digital. Interrupción de cambio de pin. ICSP datos.

Tabla 3. Asignación de pines para el microcontrolador PIC18F2550 y funciones que realizan

Fuente: https://core.ac.uk/download/pdf/30045385.pdf

Una vez explicado el funcionamiento de cada pin del PIC18F2550, en la siguiente figura se muestra su arquitectura interna, con el diagrama de bloques de los periféricos y las líneas de entrada y salida.



Note 1: RE3 is multiplexed with MCLR and is only available when the MCLR Resets are disabled.
2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.
3: RB3 is the alternate pin for CCP2 multiplexing.

Figura 12. Arquitectura interna del microcontrolador PIC18F2550

Fuente: https://core.ac.uk/download/pdf/30045385.pdf

2.6.2. Organización de la memoria

El PIC18F2550 tiene dos tipos de memoria: memoria de programa y memoria de datos. En la memoria de programa se almacenan todas las instrucciones del programa de control, que debe estar almacenado de forma permanente. En la memoria de datos se almacenan los datos que manejan los programas. Éstos van variando continuamente, lo que exige que sea de lectura y escritura.

2.6.3. Memoria de programa.

Los microcontroladores de la familia PIC18 implementan un contador de programa (CP) de 21-bit, el cual es capaz de direccionar hasta 2Mbytes de memoria de programa. Si se intenta acceder a alguna ubicación entre el límite superior de la memoria implementada (8000h) y los 2Mbytes de memoria totales (20000h) devolverá ‘0’s (instrucción NOP). El PIC18F2550 tiene 32kbytes de memoria Flash, que pueden almacenar hasta 16.384 instrucciones de una palabra. Los PIC18 tienen dos vectores de interrupción: el vector RESET que está en la dirección 0000h y los vectores de interrupción, que están en las direcciones 0008h y 0018h.

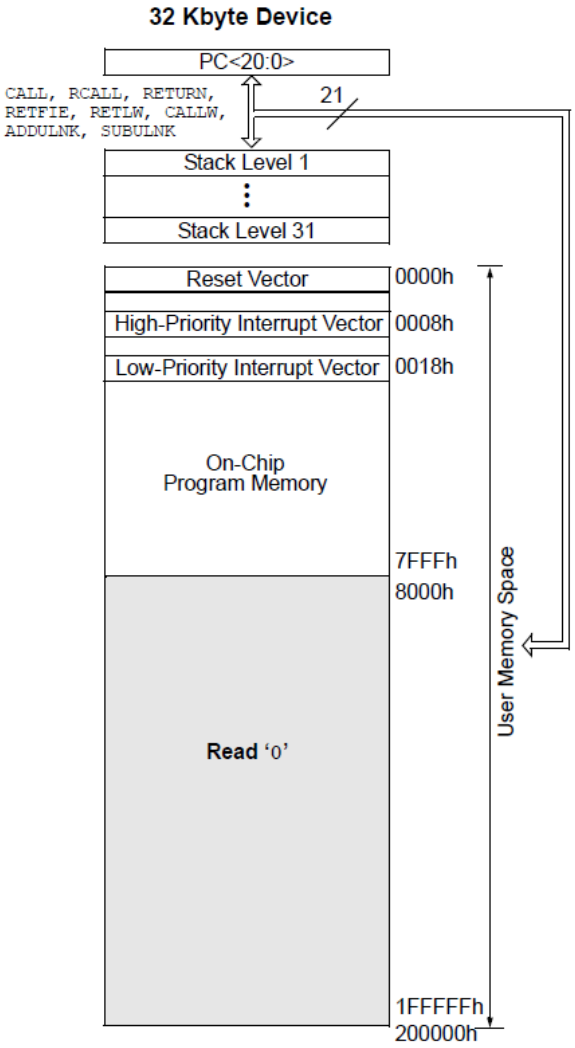


Figura 13. Mapa de memoria del microcontrolador PIC18F2550

Fuente: https://core.ac.uk/download/pdf/30045385.pdf

2.6.4. Direccionamiento de la memoria de programa.

El contador del programa (CP) de 21 bits, está compuesto por tres registros de 8 bits cada uno (ver Figura 4.8). El byte bajo del registro, conocido como PCL, puede ser leído y escrito. El byte intermedio (PC<15:8>) está alojado en el registro PCLH, y sobre él no se puede leer ni escribir, pero se puede acceder a él indirectamente a través del registro

PCLATH. El byte superior (PC<20:16>) está alojado en el registro PCLU, y sobre él no se puede leer ni escribir, pero se puede acceder a él indirectamente a través del registro PCLATU.

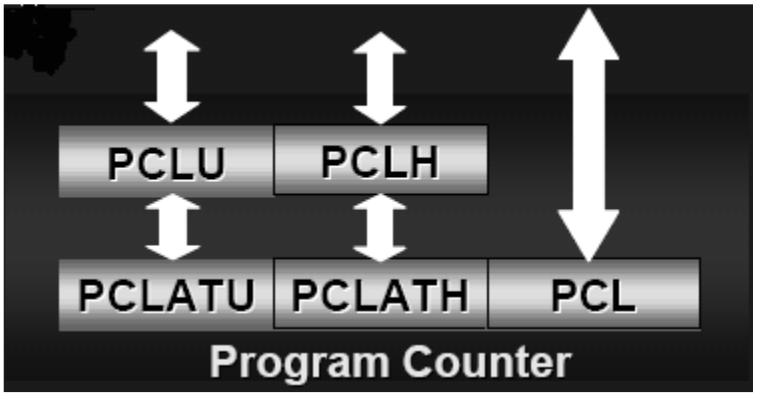


Figura 14. Contador de programa

Fuente: https://core.ac.uk/download/pdf/30045385.pdf

2.6.5. Memoria de datos: banco de registros y EEPROM.

La memoria de datos contiene Registros de Función Especial (SFRs) y Registros de Propósito General (GPRs). Los SFRs son usados para control y estado del controlador y para funciones de los periféricos, mientras que los GPRs son usados para almacenamiento de datos de las aplicaciones del usuario. Cualquier lectura de una localización no implementada debe ser leída como 0's.

Con el conjunto de instrucciones se pueden realizar operaciones en todos los bancos. Existen los siguientes modos de direccionamiento: Directo, Indirecto o Indexado.

La memoria de datos en los PIC18 se implementa como una RAM estática. Cada registro en la memoria de datos tiene 12 bits de direccionamiento, permitiendo hasta 4096 bytes de memoria de datos. El espacio de memoria es dividido en 16 bancos de registros que

contienen cada uno 256 bytes. El PIC18F2550 tiene implementados 8 bancos completos para un total de 2048 bytes.

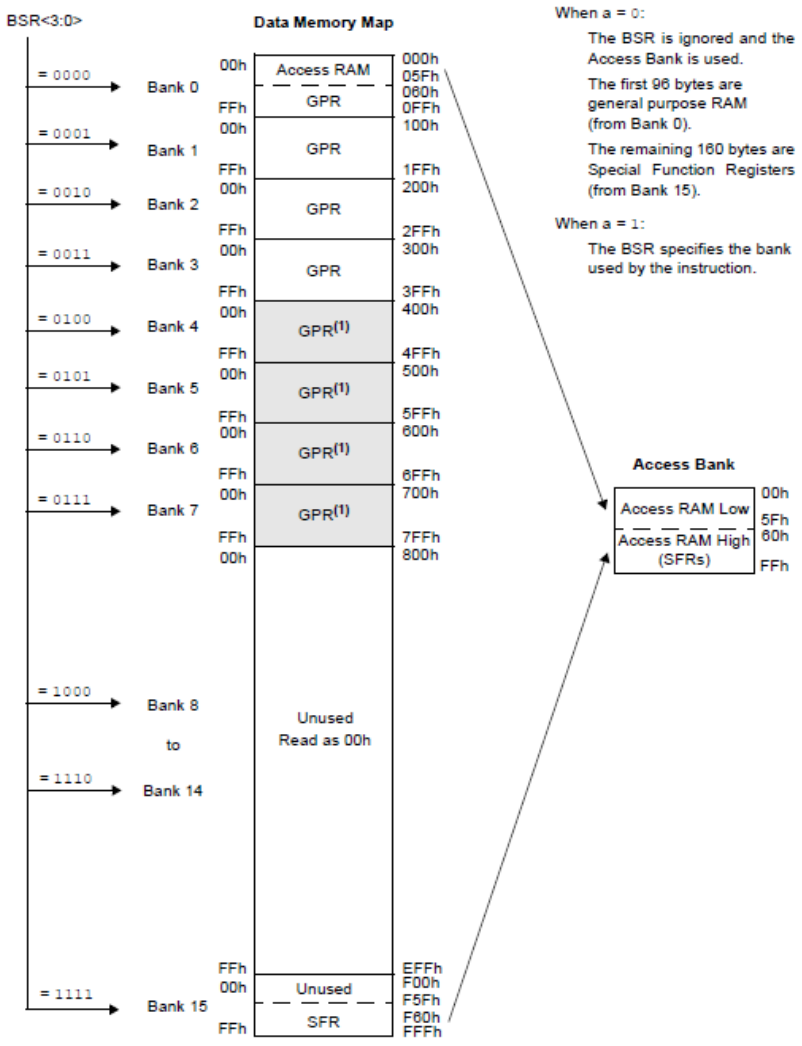


Figura 15. Banco de registros del PIC18F2550

Fuente: https://core.ac.uk/download/pdf/30045385.pdf

La memoria de datos EEPROM de 256 bytes en el PIC18F2550 es una memoria no volátil que almacena a largo plazo cualquier dato que se desee retener cuando se apague la alimentación. Esta memoria es de 8 bits, no forma parte del espacio normal direccionable y sólo es accesible en lectura y escritura a través de dos registros.

2.6.6. Líneas de Entrada-Salida

Hay tres puertos disponibles, llamados A, B y C, siendo cada uno de ellos de 8 bits

bidireccionales. Cada puerto tiene 3 registros para su funcionamiento:

- Registro TRIS (datos del registro de dirección).
- Registro PORT (lee los niveles en los terminales del dispositivo).
- Registro LAT (retención de salida).

Mediante los registros TRIS se configuran los puertos como entrada o salida. Si se está conectado con otro dispositivo como por ejemplo un sensor, al configurarlo como entrada, cada vez que se reciba una señal de éste, avisará al microcontrolador para que ejecute la instrucción correspondiente. Si está configurado como salida, será el microcontrolador el encargado de comunicarle al dispositivo las tareas a realizar.

Los pines de cualquier puerto, A, B ó C, pueden funcionar como pines de Entrada/Salida, pero además suelen tener otras funciones asociadas, como por ejemplo de reloj, de conversión Analógico/Digital recepción de datos del puerto serie, etc. Con el registro PORT se selecciona cómo se utilizarán en cada caso los puertos.

Con los registros LAT se almacenan y leen los datos de los puertos.

2.6.7. Reloj principal

Es un elemento imprescindible, ya que es el encargado de generar la base de tiempos para la ejecución de cada instrucción. Los PIC18 tienen un oscilador flexible, es decir, el microcontrolador puede estar trabajando a 12MHz y al mismo tiempo a los 48MHz necesarios para el USB. Hay osciladores primarios, secundarios e internos, existiendo en total 12 osciladores.

1.	XT	Crystal/Resonator
2.	XTPLL	Crystal/Resonator with PLL enabled
3.	HS	High-Speed Crystal/Resonator
4.	HSPLL	High-Speed Crystal/Resonator with PLL enabled
5.	EC	External Clock with Fosc/4 output
6.	ECIO	External Clock with I/O on RA6
7.	ECPLL	External Clock with PLL enabled and Fosc/4 output on RA6
8.	ECPIO	External Clock with PLL enabled, I/O on RA6
9.	INTHS	Internal Oscillator used as microcontroller clock source, HS Oscillator used as USB clock source
10.	INTXT	Internal Oscillator used as microcontroller clock source, XT Oscillator used as USB clock source
11.	INTIO	Internal Oscillator used as microcontroller clock source, EC Oscillator used as USB clock source, digital I/O on RA6
12.	INTCKO	Internal Oscillator used as microcontroller clock source, EC Oscillator used as USB clock source, Fosc/4 output on RA6

Figura 16. Modos del oscilador para el microcontrolador PIC18F2550

Fuente: https://core.ac.uk/download/pdf/30045385.pdf

2.6.8. Puertos de comunicación.

Los pines del microcontrolador se agrupan formando puertos. A través de estos puertos se puede establecer comunicación entre el microcontrolador y otros dispositivos para la recepción y/o envío de datos. Las características más destacadas de los periféricos de los PIC18 son:

- Alta corriente de alimentación (25 mA).
- 3 interrupciones externas.
- 4 módulos de Timer (Timer0 a Timer3).
- 2 Módulos de Captura/Comparación/PWM (CCP):
    - Capture es de 16-bit, máx. resolución 5.2 ns (TCY/16).
    - Compare es de 16-bit, máx. resolución 83.3 ns (TCY).
    - Salida PWM con resolución de 1 a 10 bits.
- Módulo de Capture/Compare/PWM mejorado (ECCP).
    - Múltiples modos de salida.
    - Polaridad seleccionable.
    - Programación de tiempo muerto.
    - Auto-apagado y Auto-encendido.
- Módulo USART mejorado:
    - Soporta bus LIN.
- Módulo Master Synchronous Serial Port (MSSP) que soporta 3-cables SPI™ (4 modos) y modo maestro – esclavo de I2C™.
- 10-bit, hasta 13-canales del módulo convertidor Análogo-a-Digital (A/D).
- Doble comparador analógico con entrada multiplexada.

2.7. Comunicación SPI

La SPI fue desarrollada por Motorola (ahora parte de NXP Semiconductors) aproximadamente en 1985. Se trata de una interfaz serial síncrona prevista para la comunicación entre dispositivos a corta distancia. Desde entonces, se ha convertido en un estándar de-facto empleado por muchos fabricantes de semiconductores, especialmente en microprocesadores y microcontroladores.

El motivo de la popularidad de SPI radica en sus muchas ventajas. La primera es que es una interfaz direccionada de hardware simple que ofrece completa flexibilidad para la cantidad de bis transferidos. Usa un modelo de maestro-secundario con un maestro simple y puede manejar varios dispositivos secundarios usando comunicaciones dúplex que operan a velocidades de reloj de hasta 50 MHz. No usa un protocolo estándar y transfiere solo paquetes de datos, lo que la hace ideal para transferir flujos de datos largos.

SPI usa un máximo de cuatro líneas de señal (Figura 1). El dispositivo maestro, por lo general un procesador o controlador, suministra y controla el reloj (SCK) y líneas de selección de chip (CS). La operación multiplexor completa se maneja a través de las líneas de datos Master Out Slave In (MOSI) y Master In Slave Out (MISO). En un maestro individual simple, con configuración del dispositivo secundario individual, la línea de selección de chip puede eliminarse y se puede forzar la entrada de CS al dispositivo secundario al estado lógico habilitado. Si el dispositivo secundario solo puede enviar datos (comunicación semidúplex), luego la línea MOSI también puede eliminarse, y así reducir el conteo de señales adicionalmente. Los datos salen a través de la señal del reloj de tal forma que la transferencia de datos se asemeja a un registro de turnos con un bit cambiado para cada reloj.

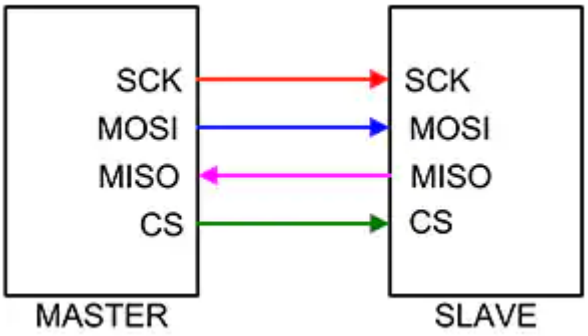


Figura 17. *La conexión dúplex SPI básica*

Fuente: https://www.digikey.com/-/media/Images/Article%20Library/TechZone%20Articles/2019/February/Why%20How%20to%20Use%20Serial%20Peripheral%20Interface%20Simplify%20Connections%20Between%20Multiple%20Devices/article-2019february-why-and-how-to-use-fig1.jpg

La conexión directa usa una línea de selección de chips para cada dispositivo secundario. La mayoría de los microprocesadores posee tres o cuatro líneas de selección de chip. Esto limita la cantidad máxima de dispositivos secundarios al número de líneas de selección de chip. En la mayoría de los casos, esto no resulta ser un problema, pero si un diseño requiere más dispositivos en el bus, se pueden configurar algunos usando el enfoque encadenado. Con un encadenamiento, se usa una selección de chips comunes para varios dispositivos secundarios y los datos se transfieren hacia afuera en una línea de datos común. Una vez más, si se usa el modelo de los dispositivos secundarios SPI como un registro de turnos, los datos de los dispositivos secundarios se propagan en un flujo multiplexado serial.

El maestro controla y genera el reloj. Los dos atributos de reloj son la polaridad del reloj (CPOL) y la fase de reloj (CPHA). Estos controlan el borde del reloj activo, donde se cronometra el dispositivo secundario en relación con los datos. CPOL = 0 establece el reloj en inactivo a una lógica 0. CPOL = 1 presenta el reloj inactivo en la lógica 1. CPHA

= 0 establece los datos en el reloj en el borde principal, y CPHA = 1 establece los datos en el reloj en el borde de rastreo.

Existen cuatro modos en el cual se puede enviar información dependiendo de dos párametros basados en la señal de reloj. El primer de ellos es la polaridad y el segundo es la fase. Al tener dos parámetros donde cada uno puede tomar dos estados se tendrá entonces cuatro modos distintos de poder llevar a cabo el proceso de transmisión y envío de información.

- Modo 0: CPOL = 0 y CPHA = 0. Modo en el cual el estado del reloj permanece en estado lógico bajo y la información se envía en cada transición de bajo a alto, es decir alto activo.
- Modo 1: CPOL = 0 y CPHA = 1. Modo en el cual el estado del reloj permanece en estado lógico bajo y la información se envía en cada transición de alto a bajo, es decir bajo activo.
- Modo 2: CPOL = 1 y CPHA = 0. Modo en el cual el estado del reloj permanece en estado lógico alto y la información se envía en cada transición de bajo a alto, es decir alto activo.
- Modo 3: CPOL = 1 y CPHA = 1. Modo en el cual el estado del reloj permanece en estado lógico alto y la información se envía en cada transición de alto a bajo, es decir bajo activo.

La configuración de modos es independiente para cada esclavo con esto quiere decir que cada esclavo puede tener una configuración de CPOL y CPHA distinta a la de otros esclavos, inclusive una frecuencia de trabajo distinta y entonces para esto, el maestro deberá adaptarse a la configuración de cada esclavo. Por esta razón es recomendable que el sistema trate de trabajar con los mismos parámetros de ser posible porque si no, será un dolor de cabeza.

2.8. Comunicación UART

UART significa transmisor-receptor asíncrono universal y define un protocolo o un conjunto de reglas, para intercambiar datos en serie entre dos dispositivos. El UART es muy simple y solo utiliza dos cables entre el transmisor y receptor para transmitir y recibir en ambas direcciones. Ambos terminales también tienen una conexión a tierra. La comunicación en el UART puede ser simplex (los datos se envían en una sola dirección), semidúplex (cada lado transmite, pero solo uno a la vez), o dúplex completo (ambos lados pueden transmitir en simultaneo). Los datos en el UART se transmiten en la forma de tramas. El formato y el contenido de estas tramas se describe y explica brevemente.

Temporización y sincronización de los protocolos UART

Una de las grandes ventajas del UART es que es asíncrono: el transmisor y el receptor no comparten una señal de reloj común. Aunque esto simplifica enormemente el protocolo, pone ciertos requisitos al transmisor y al receptor. Dado que no comparten un reloj, ambos terminales deben transmitir a la misma velocidad preestablecida para que tengan la misma sincronización de bits. Las velocidades de baudios del UART más comunes que se utilizan en estos días son 4800, 9600, 19.2K, 57.6K y 115.2K. Además de tener la misma velocidad en baudios, ambos lados de una conexión UART también deben utilizar los mismos parámetros y estructura de trama. La mejor manera de entender esto es mirar una trama UART.

Figura 18. Las tramas UART contienen bits de inicio y parada, bits de datos, y un bit de paridad opcional

Fuente:https://cdn.rohde-schwarz.com/pws/solution/research___education_1/educational_resources_/oscilloscope_and_probe_fundamentals/05_Understanding-UART_02_w640_hX.png

Como la mayoría de los sistemas digitales, un nivel de voltaje «alto» se utiliza para indicar un «1» lógico y un nivel de voltaje «bajo» se usa para indicar un «0» lógico. Dado que el protocolo UART no define voltajes o rangos de voltaje específicos para estos niveles, a veces a alto también se le llama «marca» mientras que a bajo se le llama «espacio». Note que en el estado de reposo (donde no se transmiten datos), la línea se mantiene alta. Esto permite una fácil detección de una línea o transmisor dañado.

2.8.1. Bits de inicio y de parada

Ya que el UART es asíncrono, el transmisor necesita indicar que los bits de datos están llegando. Esto se logra utilizando el bit de inicio. El bit de inicio es una transición del estado alto de reposo a un estado bajo, y seguido inmediatamente por bits de datos de usuario.

Después de que se terminan los bits de datos, el bit de parada indica el fin de datos de usuario. El bit de parada es una transición de regreso al estado alto o de reposo o permanece en el estado alto por un tiempo de bit adicional. Se puede configurar un segundo bit de parada (opcional), generalmente para darle tiempo al receptor de prepararse para la siguiente trama, pero esto no es común en la práctica.

2.8.2. Bits de datos

Los bits de datos son los datos de usuario o bits «útiles» y vienen inmediatamente después del bit de inicio. Puede haber de 5 a 9 bits de datos de usuario, aunque de 7 o 8 bits es lo más común. Estos bits de datos normalmente se transmiten primero con el bit menos importante.

Por ejemplo:

si queremos mandar la «S» mayúscula en código ASCII de 7 bits, la secuencia de bits es 1 0 1 0 0 1 1. Primero y antes de enviarlos, invertimos el orden de los bits para ponerlos en el orden de bit menos importante, que es 1100101. Después de que se envía el último

bit de datos, se utiliza el bit de parada para finalizar la trama y la línea regresa al estado de reposo.

Código ASCII de 7 bits para «S» (0x52) = 1 0 1 0 0 1 1

Orden de bit menos significativo = 1 1 0 0 1 0 1

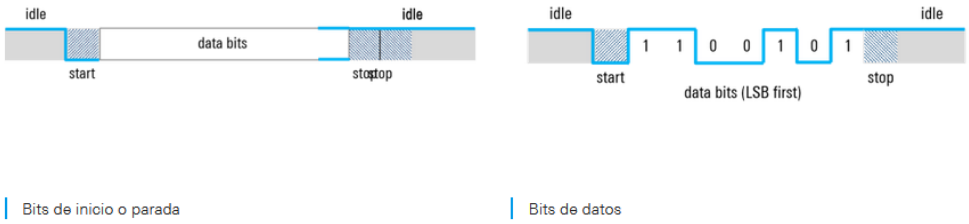


Figura 19. Bits de inicio o parada, Bits de datos

Fuente: https://cdn.rohde-schwarz.com/pws/solution/research___education_1/educational_resources_/oscilloscope_and_probe_fundamentals/05_Understanding-UART_03_w480_hX.png

2.8.3. Bit de paridad

Una trama UART puede también contener un bit de paridad opcional que puede usarse para la detección de errores. Este bit se inserta entre el final de los bits de datos y el bit de parada. El valor del bit de paridad depende del tipo de paridad que está siendo usado (par o impar):

En la paridad par, este bit se configura para que el total de números 1 en la trama será par.

En la paridad impar, este bit se configura para que el total de números 1 en la trama será impar.

Por ejemplo:
“S” mayúscula (1 0 1 0 0 1 1) contiene un total de tres ceros y cuatro unos. Si se usa la

paridad par, el bit de paridad es cero porque ya hay un número par de unos. Si se usa la paridad impar, entonces el bit de paridad tiene que ser uno para hacer que la trama tenga un número impar de unos.
El bit de paridad solamente puede detectar un solo bit invertido. Si hay más de un bit invertido, no hay manera de detectarlos de manera confiable usando un solo bit de paridad.

2.9. Lenguaje de programación Visual Basic

Visual Basic. Es un lenguaje de programación desarrollado por el alemán Alan Cooper para Microsoft. El lenguaje de programación es un dialecto de BASIC, con importantes agregados. Su primera versión fue presentada en 1991, con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y, en cierta medida, también la Programación misma. Desde el 2001 Microsoft ha propuesto abandonar el desarrollo basado en la API Win32 y pasar a trabajar sobre un framework o marco común de librerías independiente de la versión del sistema operativo, NET Framework, a través de Visual Basic, NET (y otros lenguajes como C Sharp (C#) de fácil transición de código entre ellos).

Visual basic. Constituye un IDE (entorno de desarrollo integrado, o, en inglés, Integrated Development Enviroment) que ha sido empaquetado como un Programa de aplicación; es decir, consiste en un editor de código (programa donde se escribe el código fuente), un depurador (programa que corrige errores en el código fuente para que pueda ser bien compilado), un compilador (programa que traduce el código fuente a lenguaje de máquina), y un constructor de interfaz gráfica o GUI (es una forma de programar en la que no es necesario escribir el código para la parte gráfica del Programa, sino que se puede hacer de forma visual).

Con Visual Basic se pueden desarrollar aplicaciones para Windows más rápidamente. Los errores de Programación no se generan tan frecuentemente y, si lo hacen, son más sencillos de depurar. Además, incluye dos conceptos importantes:

Un método visual de creación de aplicaciones, incluyendo formularios (Ventanas), controles y, componentes del formulario.

La habilidad de asociar código directamente a cada evento de cada elemento del diseño visual.

Es posible escribir aplicaciones sin usar componentes visuales, es decir escribir aplicaciones de consola.

2.10. Lenguaje de consulta SQL con MySql para bases de datos

La base de datos que contiene información relativa al servicio de directorio se ha realizado en lenguaje SQL utilizando concretamente el servidor de bases de datos que ofrece MySQL.

La base de datos al completo es diseñada y gestionada mediante el gestor de bases de datos que ofrece MySQL, pero las distintas consultas que deba realizar el cliente serán llevadas a cabo por el servicio web.
Para llevar a cabo la consulta a la base de datos se ha utilizado el estándar JDBC que ofrece conexión a bases de datos desde servidores realizados en lenguaje Java.

Las capacidades de MySQL son extremadamente amplias, ya que este servidor de bases de datos cuenta con un gran potencial de funcionamiento. El objetivo de este punto es el de mostrar el uso de MySQL para crear y usar una sencilla base de datos.
MySQL ofrece un programa interactivo que permite conectarnos a un servidor MySQL, ejecutar consultas y ver los resultados. Todas estas operaciones se pueden llevar a cabo tanto desde línea de comando en un shell, como desde un programa front-end gráfico que presente una interfaz gráfica de control.

Puesto que es imposible que se describan aquí en detalle muchas de las características cubiertas por MySQL, nos centraremos sólo en aquellas de relevancia e invitamos

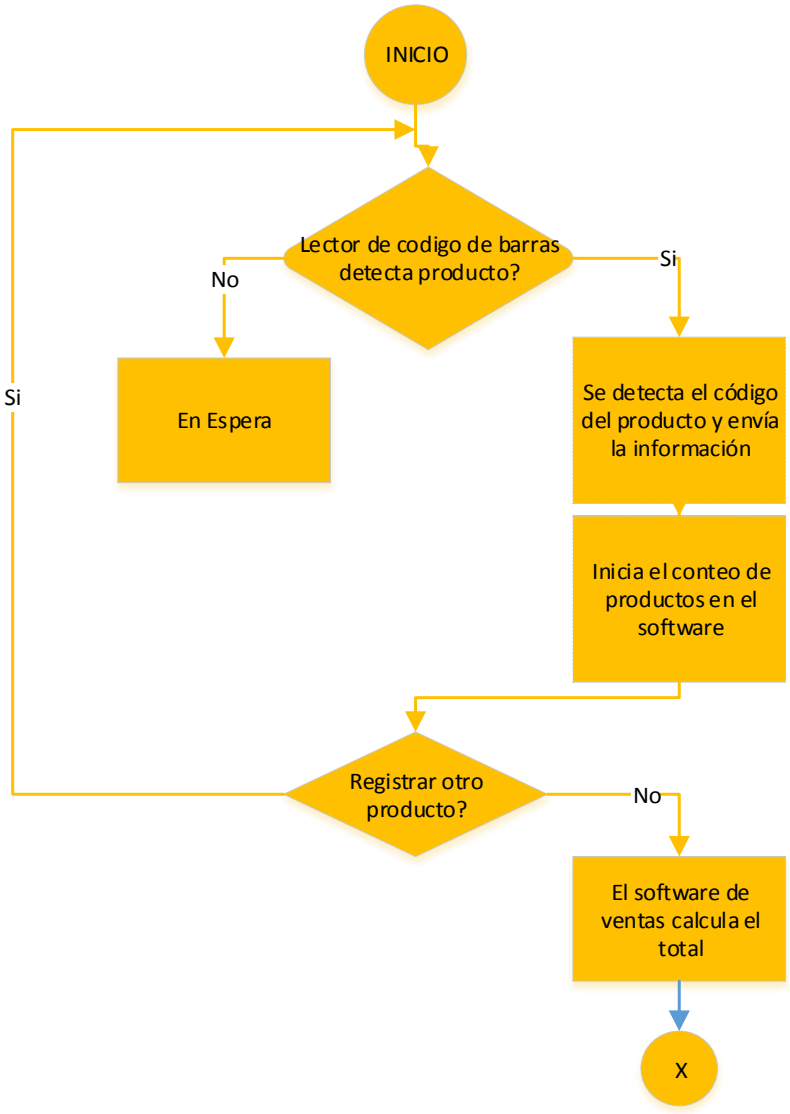
consultar el manual de MySQL para obtener más información al respecto de funcionalidades específicas.

CAPÍTULO 3

Desarrollo del proyecto

3.1. Diagrama de flujo

En el siguiente diagrama se definen los pasos que se realizan en el sistema de cobros.



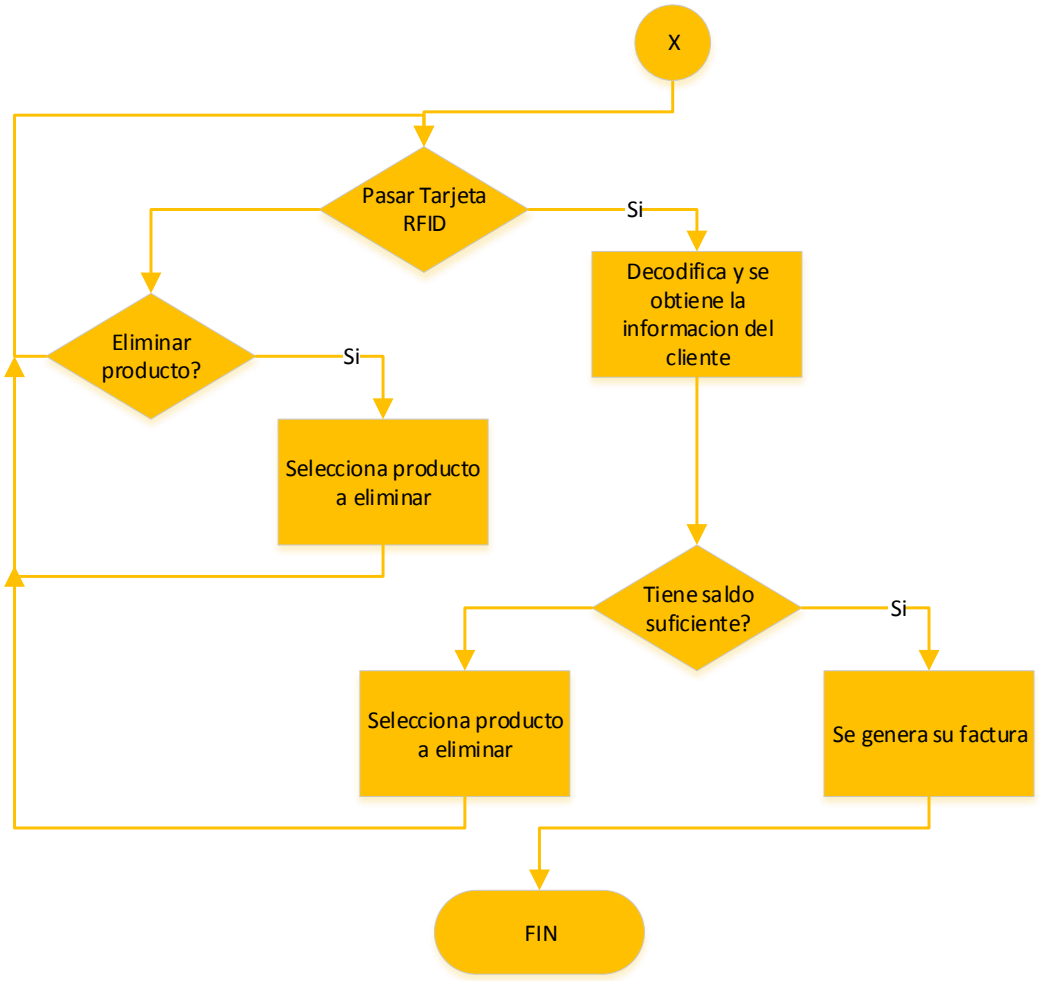


Figura 20. Diagrama de flujo del sistema de cobros

Fuente: Propia

3.2. Diagrama de circuito

El diagrama de circuito representa la conexión de los microcontroladores con los módulos RFID y comunicación serial.

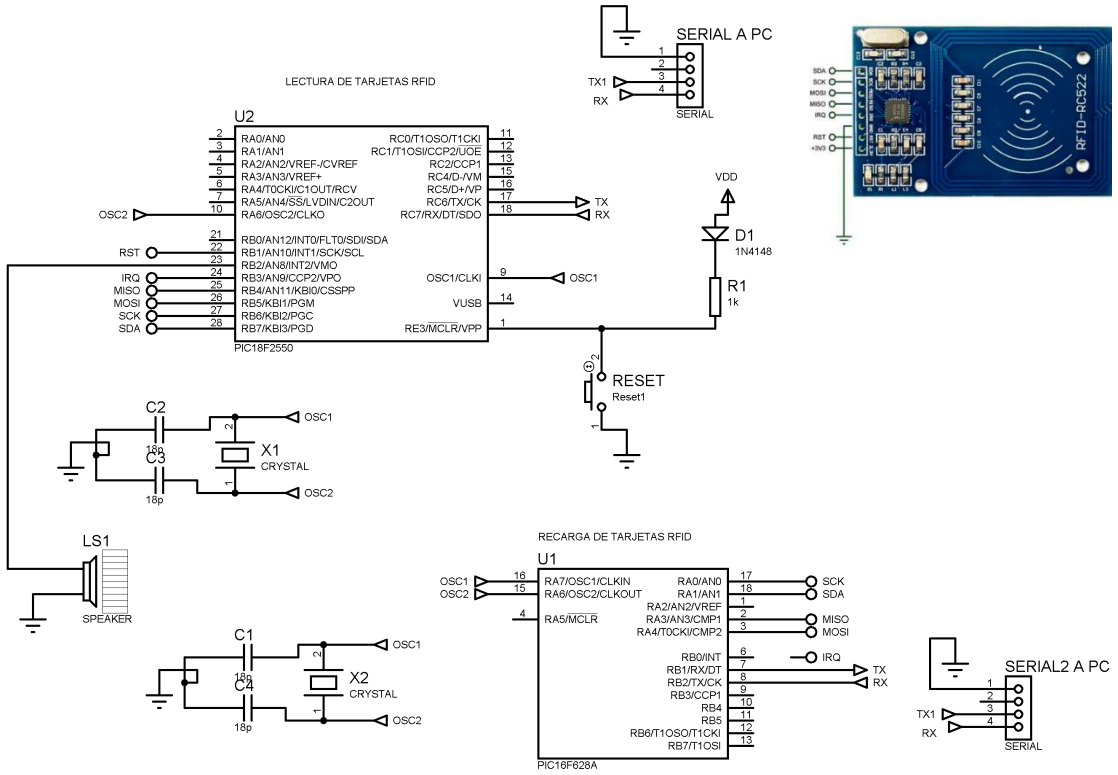


Figura 21. Diagrama de circuito

Fuente: Propia

3.3. Diagrama de bloques

El diagrama de bloques representa la arquitectura de comunicación entre los diferentes dispositivos, microcontroladores y el software

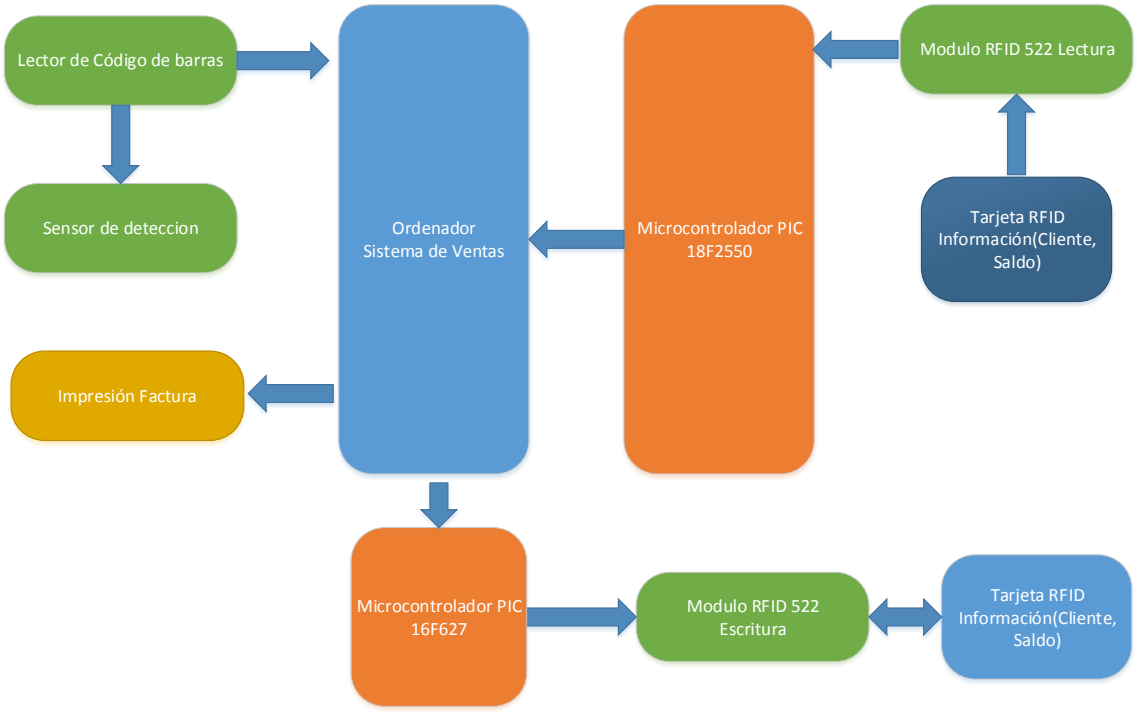


Figura 22. Diagrama de bloques

Fuente: Propia

3.4. Desarrollo del Hardware

La interfaz de hardware permitirá al cliente llevar sus productos seleccionados del supermercado al sector de cajas donde podrá escanear cada uno de sus productos, después de la lectura de los productos podrá usar su tarjeta RFID para el cobro del total de los productos.

3.4.1. Interfaz Lector de código de barras

La lectura de código de barras se realiza a través de un dispositivo que se conecta al ordenador mediante usb se configura la comunicación con el sistema de cobros.

3.4.2. Programación Modulo de lectura

El código para ir reduciendo el monto con un lector RFID se utiliza la comprobación del número del saldo si ya no tiene más saldo se restringe la tarjeta.

El desarrollo del código de programa para la lectura y escritura en la tarjeta RFID se usó el compilador C con mikroC.

Rutina para obtener el ID de la tarjeta:

```
unsigned long  Hash_algoritmo () //Obtiene el UID de la tarjeta
{
unsigned long  hash_acum = FNV_BASIS;

     for (cont=0;cont<=4;cont++)   // se guardan cinco datos del uid
        {
             // Add value into the hash
            hash_acum = (hash_acum * FNV_PRIME) ^ UID[cont];
       }

       return (hash_acum);
}
```

Programa principal:

```
void main()
{
  ADCON1=0B1111;// TODO DIGITAL
  CMCON=7; // APAGA EL COMPARADOR.
  trisb=0;
  trisc=0;

  trisc.b7=1; //entrada RX
  trisc.B1=1;
  UART1_Init(9600);
  error = 'A';
  error = Soft_UART_Init(&PORTC, 1, 2, 9600, 0);
  RCEN_bit=1; //recepcion continua.
  UART1_Write_Text( "Iniciando\r\n" );
  LATB=0;
```

```
  Soft_SPI_Init();
  MFRC522_Init();  //inicializa la tarjeta
  delay_ms(200);

  while(1){
    if (UART1_Data_Ready()) {    //Lee opción por Puerto serial
      uart_rd = UART1_Read();
//      UART1_Write(uart_rd);
      Soft_UART_Write(uart_rd);

      if(uart_rd=='r'){
        Soft_UART_Write('r');
        led2=~led2;
        leertarjeta();
        UART1_Write('f');
      }
      if(uart_rd=='w'){ // Opción para escribir datos del cliente
        led2=~led2;
        Soft_UART_Write('w');
        Soft_UART_Write(10);
        Soft_UART_Write(13);
        UART1_Write_Text("dato"); //para la escritura de la tarjeta
        UART1_Write('\r');
        UART1_Write('\n');
        delay_ms(200);
        opcion = 'a';
        capturarinf();
        UART1_Write('f');
      }
      if(uart_rd == 'u'){ // Opción para actualizar el monto del cliente
        led2=~led2;
        Soft_UART_Write('u');
        UART1_Write_Text("carga"); //para actualizar los datos de la tarjeta
        UART1_Write('\r');
        UART1_Write('\n');
        delay_ms(200);
        opcion = 'm';
        capturarinf();
        UART1_Write('f');
      }
      if(uart_rd == 'p'){ // Opción para hacer el cobro de productos registrados
        led2=~led2;
```

```
        Soft_UART_Write('p');
        UART1_Write_Text("cobro");
        UART1_Write('\r');
        UART1_Write('\n');
        delay_ms(200);
        opcion = 'p';
        comprar();
        UART1_Write('f');
      }
       led1=~led1;
    }
  }
}
```

Rutina para hacer la compra obteniendo el saldo en la tarjeta y recibiendo por puerto serial el monto a cobrar.

```
void comprar(){
  while (1)
  {
   if (UART1_Data_Ready()) { //Recibe el monto a cobrar
     memset(buffer, 0, BUFFER_LENGTH + 1);
     UART1_Read_Text(&buffer, "#", BUFFER_LENGTH);
     while(1){
      if( MFRC522_isCard( &TagType ) ) //inicia la lectura de la tarjeta detectada
      {
       if( MFRC522_ReadCardSerial( &UID ) )
       {
         size = MFRC522_SelectTag( &UID );
         intvar = atoi(buffer);
         if( MFRC522_Auth( PICC_AUTHENT1A, 7, &key, &UID ) == 0 &&
MFRC522_Auth( PICC_AUTHENT1B, 7, &key, &UID ) == 0)
         {
          //monto
          lecturasectormonto(5); //Obtiene el monto registrado
          monto = atoi(writeData);
          monto = monto - intvar;
          led3=~led3;
         }
         else
         {
          continue;
```

```
          }
          if(monto >= 0){
            inttostr(monto, writeData); //Escribe el monto restante en la tarjeta
            MFRC522_Write( 5, &writeData );
            lecturasectorinfo(5);
          }else{
            UART1_Write_Text("no hay saldo");
            UART1_Write('\r');
            UART1_Write('\n');
          }
          // procesa codigo
          codigo_rfid=0;
          Delay_ms(200);
          UART1_Write_Text("Cobrado");
          UART1_Write('\r');
          UART1_Write('\n');
          leertarjeta();
          break;
        } // lee el codigo de la tarjeta
       }//  detecta precencia RFID
      }
      break;
    }
  }
}
```

Rutina para agregar una nueva tarjeta con datos del cliente o monto a actualizar.

```
void capturarinf(){
  int sw=0;
  while(1){
  i=0;
  if (UART1_Data_Ready()) {     // Se espera una opción a=”Agregar nueva tarjeta”
                                // m=”modificar monto”
     memset(buffer, 0, BUFFER_LENGTH + 1);         // Clear the buffer.
     UART1_Read_Text(&buffer, "#", BUFFER_LENGTH);
     for( i=0; i < 20; i++){
       Soft_UART_Write(buffer[i]);
     }
     delay_ms(200);
     if(opcion == 'a'){
       Soft_UART_Write('t');
```

```
      Soft_UART_Write(10);
      Soft_UART_Write(13);
      nuevatarjeta();
     }
     if(opcion == 'm'){
      modtarjeta();
     }
     break;
        sw= 1;
   }
  }
}
```

Rutina para actualizar el monto sumando al monto registrado en la tarjeta.

```
void modtarjeta(){
while (1)
 {
    if( MFRC522_isCard( &TagType ) )
     {
      if( MFRC522_ReadCardSerial( &UID ) )
      {
        size = MFRC522_SelectTag( &UID );
        if( MFRC522_Auth( PICC_AUTHENT1A, 7, &key, &UID ) == 0 &&
MFRC522_Auth( PICC_AUTHENT1B, 7, &key, &UID ) == 0)
        {
         lecturasectormonto(5); // Lectura del monto en la tarjeta
         monto = atoi(writeData);
         intvar = atol(buffer); // Valor guardado de la lectura del Puerto serial
         if(monto != -1 && intvar != -1){
          monto = monto + intvar;
          inttostr(monto, buffer);
          MFRC522_Write( 5, &buffer ); //total resultante para el saldo
         }else{
          MFRC522_Write( 5, &buffer ); //total si no tiene saldo en la tarjeta
         }
         led3=~led3;
        }
        else
        {
         continue;
        }
        // procesa codigo
```

```
            codigo_rfid=0;
            Delay_ms(200);
            UART1_Write_Text("Recargado");
            UART1_Write('\r');
            UART1_Write('\n');
            break;
          } // lee el codigo de la tarjeta
        }//  detecta precencia RFID
  }
}
```

Rutina para agregar nueva tarjeta con datos del cliente.

```
void nuevatarjeta(){
  while (1)
  {
     if( MFRC522_isCard( &TagType ) )
      {
        if( MFRC522_ReadCardSerial( &UID ) )
        {
          codigo_rfid=Hash_algoritmo();
          LongIntToHex(codigo_rfid,texto);
          size = MFRC522_SelectTag( &UID );
          if( MFRC522_Auth( PICC_AUTHENT1A, 7, &key, &UID ) == 0 &&
MFRC522_Auth( PICC_AUTHENT1B, 7, &key, &UID ) == 0)
          {
           //codigocliente
           MFRC522_Write( 4, &buffer ); // escribe el código de cliente obtenido por el
                                                  // puerto serial
           led3=~led3;
          }
          else
          {
           continue;
          }
          // procesa codigo
          codigo_rfid=0;
          Delay_ms(200);
          UART1_Write_Text("Registrado");
          UART1_Write('\r');
          UART1_Write('\n');
          break;
        } // lee el codigo de la tarjeta
```

```
      }//  detecta precencia RFID
  }
}
```

Rutina para obtener información de un bloque de la tarjeta con respuesta del valor leido.

```
void lecturasectorinfo(int b){
  if( MFRC522_Read( b, &writeData ) == 0 )
  {
    UART1_Write_Text(&writeData);
    UART1_Write('\r');
    UART1_Write('\n');
  }
}
```

Rutina para obtener información de un bloque de la tarjeta sin respuesta.

```
void lecturasectormonto(int b){
  if( MFRC522_Read( b, &writeData ) == 0 )
  {
     for( i=0; i < 15; i++){
       Soft_UART_Write(writeData[i]);
     }
  }
}
```

3.4.3. Programación Modulo de escritura

Cambiar claves tarjeta RFID con lector RFID RC522

Si se utiliza el lector RFID RC522 lo primero que tiene que hacer es cambiar las claves Key-A y Key-B para que el sistema RFID sea más seguro.

Si se utilizan las claves que vienen por defecto, cualquier persona podrá utilizarlas.

Como ya se define en el marco teórico, cada sector tiene su propia clave, aunque es una práctica común el utilizar la misma clave para todos los sectores. En este proyecto se va a cambiar las claves del sector 15.

El siguiente código cambia la clave de este sector por las siguientes:

- Key-A: A0 A1 A2 A3 A4
- Key-B: B0 B1 B2 B3 B4

En la rutina se define la llave de la tarjeta para el acceso a los bloques.

char key[6] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF }; // llave para acceso a los bloques

```
void leertarjeta(){
  while (1)
  {
     if( MFRC522_isCard( &TagType ) )
      {
       if( MFRC522_ReadCardSerial( &UID ) ) // En UID se almacena el ID de la tarjeta
       {
          codigo_rfid=Hash_algoritmo();
          LongIntToHex(codigo_rfid,texto);
          UART1_Write_Text("CODIGO_RFID=");
          UART1_Write_Text(texto);
          UART1_Write('\r');
          UART1_Write('\n');
          size = MFRC522_SelectTag( &UID );
          //Se compara el valor de la llave con el guardado en la tarjeta
          //PICC_AUTHENT1A o PICC_AUTHENT1B
          if( MFRC522_Auth( PICC_AUTHENT1A, 7, &key, &UID ) == 0 &&
MFRC522_Auth( PICC_AUTHENT1B, 7, &key, &UID ) == 0)
          {
            lecturasectorinfo(4); //Sector 4 codigo del usuario
            lecturasectorinfo(5);// Sector 5 Monto de la tarjeta
            led3=~led3;
          }
          else{
           continue; // no tiene acceso a la tarjeta
          }
          // procesa codigo
          codigo_rfid=0;
          Delay_ms(200);
          Soft_UART_Write('e');
          break;
```

```
        } // lee el codigo de la tarjeta
      }//  detecta precencia RFID
  }
}
```

3.5. Desarrollo del Software

Para el sistema de cobros se establece un sistema de ventas con los siguientes módulos: administración usuarios, productos, ventas, reportes, pagos, clientes, recargas.

3.5.1. Diagrama de base de datos

El diseño de base de datos requiere el manejo de productos, ventas y clientes para lo cual se definen las siguientes tablas.

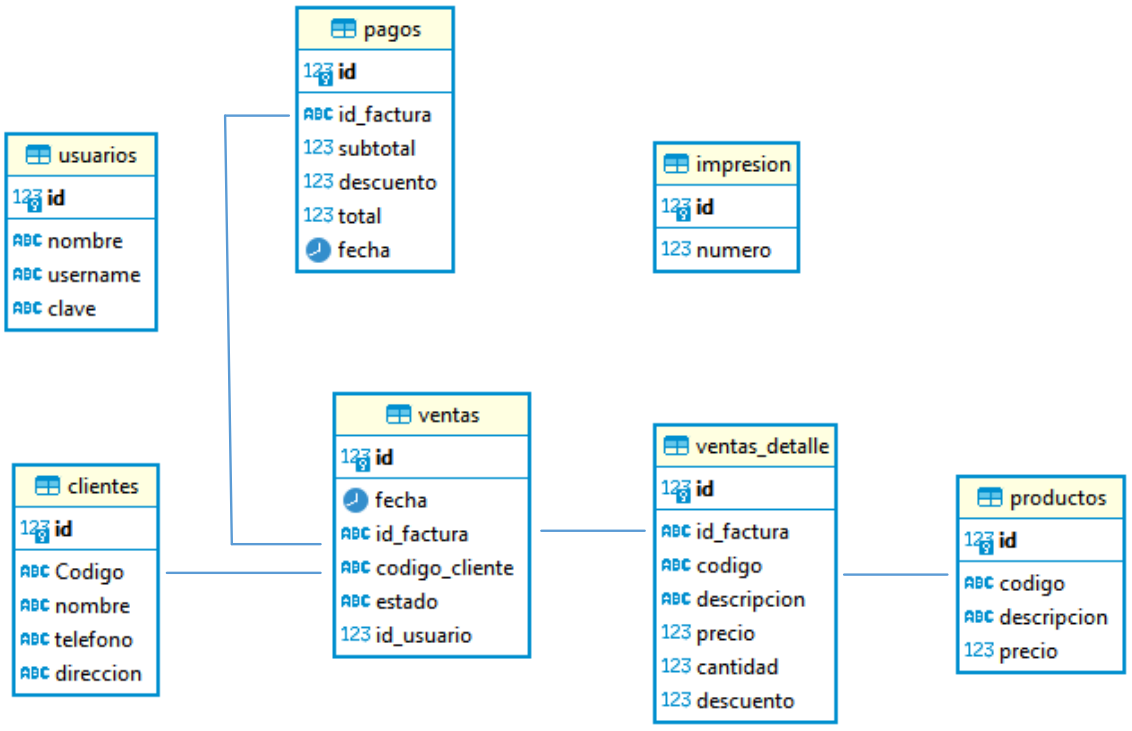


Figura 23. Diagrama de relacional

Fuente: Propia

3.5.2. Administración de usuarios

Se define el acceso al sistema a través de un login de usuario autorizados, a través de la conexión a la base de datos se controla el acceso mediante usuario y contraseña.

Función para el inicio de sesión del usuario

```
If Operaciones.Buscar("select * from usuarios where username='" & txtUser.Text & "' and clave='" & txtPassword.Text & "'") = True Then
        MainForm.Show()
        Dim fila As DataRow
        For Each fila In Acceso.ds.Tables(0).Rows
            MainForm.lbNombreUsuario.Text = fila("username")
            MainForm.lbIDusuario.Text = fila("id")
        Next
        Me.Hide()
    Else
        MessageBox.Show("Nombre de usuario o Contraseña incorrecto", "Atencion", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    End If
```

Figura 24. Interfaz de acceso al sistema

Fuente: Propia

3.5.3. Administración de productos

Los productos se registran por código descripción y precio, cada producto se registrará mediante una etiqueta que contendrá el código de barra para la lectura.

Rutinas para guardar, modificar y eliminar los productos

```
Try ‘Guardar
        If txtID.Text = "0" Then
            If MessageBox.Show("¿Seguro que desea insertar este registro?", "Atencion",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) = vbYes Then
                Operaciones.SaveData("insert into productos(codigo,descripcion,precio)
values('" & txtCodigo.Text & "','" & txtDescripcion.Text & "','" & txtPrecio.Text & "')")

                MessageBox.Show("Registro exitoso", "Atencion",
MessageBoxButtons.OK, MessageBoxIcon.Information)
                ConsultaDatos()
                LimpiaDatos()
                CrearCodigo()
            End If
        End If
    Catch ex As Exception
        MessageBox.Show(Err.Description, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
End Try


‘Actualizar

Try
        If txtID.Text <> "0" Then
            If MessageBox.Show("¿Seguro que desea actualizar este registro?",
"Atencion", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = vbYes Then
                Operaciones.SaveData("update productos set codigo='" & txtCodigo.Text &
"',descripcion='" & txtDescripcion.Text & "',precio='" & txtPrecio.Text & "' where id='"
& txtID.Text & "'")

                MessageBox.Show("Actualizacion exitosa", "Atencion",
MessageBoxButtons.OK, MessageBoxIcon.Information)
                ConsultaDatos()
```

```
            LimpiaDatos()
            CrearCodigo()
          End If
        End If
      Catch ex As Exception
        MessageBox.Show(Err.Description, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
End Try

‘Eliminar
If MessageBox.Show("¿Seguro que deseas eliminar este registro?", "Atencion",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) = vbYes Then
        Operaciones.SaveData("delete from productos where id='" & txtID.Text & "'")

        MessageBox.Show("Operacion realizada exitosamente", "Atencion",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        ConsultaDatos()
        LimpiaDatos()
        CrearCodigo()
End If
```

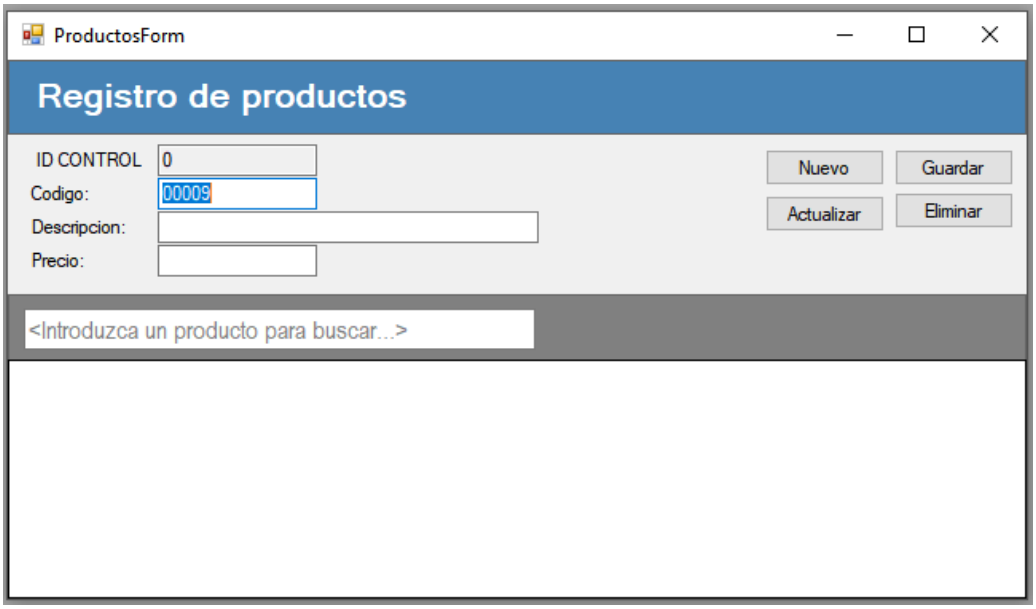


Figura 25. Interfaz de registro de productos

Fuente: Propia

3.5.4. Módulo de ventas con interfaz Lector de código de barras

Las ventas tendrán de interfaz el lector de código de barras y un monitor que refleje el producto registrado y mostrando el monto de producto con su total.

Para el proceso de cobro se desarrolla las siguientes funciones que permitirán enviar el valor total de la venta al microcontrolador.

```
Private Sub sc_DataReceived(sender As Object, e As DataReceivedEventArgs)
        Dim ID As String = DirectCast(sender, SerialController).ID
        'AddOutput(String.Format("Received from {0}: {1}", ID, e.Data))
        AddOutput(String.Format("{0}", e.Data))
    End Sub
    Private Sub AddOutput(s As String)
        My.Computer.Audio.Play(My.Resources.bell_notification,
AudioPlayMode.Background)
        If ListBox1.InvokeRequired Then
            Dim d As New TextDelegate(AddressOf AddOutput)
            ListBox1.Invoke(d, {s})
            checkcobro(s)
            TextBox4.Text = s
        Else
            If s <> "" Then
                ListBox1.Items.Add(s)
                checkcobro(s)
                TextBox4.Text = s
            End If
        End If
End Sub
Private Sub checkcobro(s As String)
        If s.Contains("cobro") Then
            sc.WritePort(total.ToString & "#")
        End If
        If s.Contains("CLIENTE") Then
            DataGridView2.Visible = True
            If  Operaciones.Buscar("select  CODIGO,NOMBRE,TELEFONO,DIRECCION
from clientes where codigo like'%" & s.Trim & "%'") = True Then
                DataGridView2.DataSource = Acceso.ds.Tables(0)
                DataGridView2.Columns(0).Visible = False
```

```
            txtCliente.Text = Acceso.ds.Tables(0).Rows(0)("NOMBRE").ToString
            txtDireccion.Text = Acceso.ds.Tables(0).Rows(0)("DIRECCION").ToString
            txtTelefono.Text = Acceso.ds.Tables(0).Rows(0)("TELEFONO").ToString
            lbCodigoCliente.Text = Acceso.ds.Tables(0).Rows(0)("CODIGO").ToString
            If DataGridView2.RowCount > 0 Then
               'DataGridView2.Rows.Item(DataGridView2.Rows.Count - 1).Selected =
True

               'txtCliente.Text                                                =
DataGridView2.CurrentRow.Cells("NOMBRE").Value.ToString
               'txtDireccion.Text                                              =
DataGridView2.CurrentRow.Cells("DIRECCION").Value.ToString
               'txtTelefono.Text                                               =
DataGridView2.CurrentRow.Cells("Telefono").Value.ToString
               'lbCodigoCliente.Text                                           =
DataGridView2.CurrentRow.Cells("codigo").Value.ToString
               DataGridView2.Visible = False
            End If
            'imprimir
            Operaciones.SaveData("insert                 into                 pagos(id_factura,
subtotal,descuento,total,fecha) " &
                        "values('" & lbCodigoFactura.Text & "'," & CDbl(txtTotal.Text) +
CDbl(txtDescuento.Text)  &  "',"  &  CDbl(txtDescuento.Text)  &  "',"  &
CDbl(txtTotal.Text) & "',?FECHA)")

            Operaciones.SaveData("update ventas set estado='COMPLETADA' where
id_factura='" & lbCodigoFactura.Text & "'")

            Dim MiForm As New Reporting.ReporteForm
            Reporting.ConsultaReporte.FacturaCliente(lbCodigoFactura.Text)
            MiForm.NombreReporte = "Reporting.FacturaCliente.rdlc"
            MiForm.MiParametro = "CONDUCE"
            MiForm.FormaReporte = "Acostado"
            MiForm.Show()
         Else
            DataGridView2.DataSource = Nothing
         End If
      End If
End Sub
```

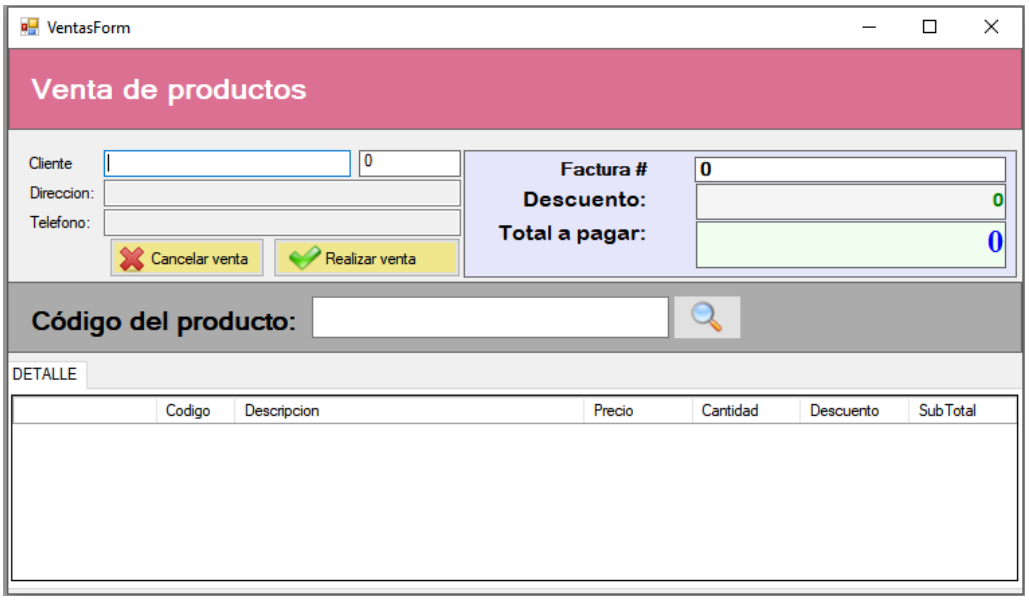


Figura 26. Interfaz de registro de productos

Fuente: Propia

3.5.5. Administración de clientes

El registro de clientes se hará en la recarga de tarjeta RFID donde se tomará los datos de una persona con el nombre, teléfono, dirección.

Rutinas para guardar, modificar, eliminar y registrar las tarjetas de clientes.

‘Guardar

```
Try
        If txtID.Text = "0" Then
            If MessageBox.Show("¿Seguro que desea insertar este registro?", "Atencion", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = vbYes Then
                Operaciones.SaveData("insert into clientes(codigo,nombre,telefono,direccion) values('" & txtCodigo.Text & "','" & txtNombre.Text & "','" & txtTelefono.Text & "','" & txtDireccion.Text & "')")
```

```
            MessageBox.Show("Registro exitoso", "Atencion",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            ConsultaDatos()
          End If
        End If
      Catch ex As Exception
        MessageBox.Show(Err.Description, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
End Try

‘Actualizar
Try
        If txtID.Text <> "0" Then
          If MessageBox.Show("¿Seguro que desea actualizar este registro?",
"Atencion", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = vbYes Then
            Operaciones.SaveData("update clientes set codigo='" & txtCodigo.Text & "',
nombre='" & txtNombre.Text & "',telefono='" & txtTelefono.Text & "',direccion='" &
txtDireccion.Text & "' where id='" & txtID.Text & "'")

            MessageBox.Show("Actualizacion exitosa", "Atencion",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            ConsultaDatos()
          End If
        End If
      Catch ex As Exception
        MessageBox.Show(Err.Description, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
End Try

‘Eliminar
If MessageBox.Show("¿Seguro que deseas eliminar este registro?", "Atencion",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) = vbYes Then
        Operaciones.SaveData("delete from clientes where id='" & txtID.Text & "'")

        MessageBox.Show("Operacion realizada exitosamente", "Atencion",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        ConsultaDatos()
End If
‘Registrar cliente
Private Sub sc_DataReceived(sender As Object, e As DataReceivedEventArgs)
      Dim ID As String = DirectCast(sender, SerialController).ID
      'AddOutput(String.Format("Received from {0}: {1}", ID, e.Data))
```

```
        AddOutput(String.Format("{0}", e.Data))
    End Sub
    Private Delegate Sub TextDelegate(s As String)
    Private Sub AddOutput(s As String)
        My.Computer.Audio.Play(My.Resources.bell_notification,
AudioPlayMode.Background)
        If ListBox1.InvokeRequired Then
            Dim d As New TextDelegate(AddressOf AddOutput)
            ListBox1.Invoke(d, {s})
            checkuser(s)
            checkReady(s)
            checkReadyC(s)

        Else
            If s <> "" Then
                ListBox1.Items.Add(s)
                checkuser(s)
                checkReady(s)
                checkReadyC(s)
            End If
        End If
    End Sub
    Private Sub checkReady(s As String)
        If s.Contains("dat") Then   'Verifica respuesta del microcontrolador
            sc.WritePort(txtCodigo.Text & "#")
        End If
        Try
            Dim cost As Integer = Integer.Parse(s.Trim)
            txtMonto.Text = cost
        Catch ex As Exception

        End Try
    End Sub
    Private Sub checkReadyC(s As String)
        If s.Contains("carga") Then  ' Verifica respuesta del microcontrolador
            sc.writeSingle(txtMonto.Text & "#")
            sc.WritePort(txtMonto.Text & "#")
        End If
    End Sub
    Private Sub checkuser(s As String)
        If s.Contains("CLIENTE") Then ' Verifica respuesta del microcontrolador
            txtBuscar.Text = s.Trim
```

```
            ConsultaDatos()
            'MsgBox(DataGridView1.Rows.Count)
            If DataGridView1.Rows.Count > 0 Then
                Dim rRowIndex As Integer = DataGridView1.Rows.Count - 1
                DataGridView1.Rows.Item(rRowIndex).Selected = True
                txtCodigo.Text = DataGridView1.CurrentRow.Cells("codigo").Value.ToString
                txtID.Text = DataGridView1.CurrentRow.Cells("id").Value.ToString
                txtNombre.Text =
DataGridView1.CurrentRow.Cells("nombre").Value.ToString
                txtTelefono.Text =
DataGridView1.CurrentRow.Cells("telefono").Value.ToString
                txtDireccion.Text =
DataGridView1.CurrentRow.Cells("direccion").Value.ToString
            End If
        End If
End Sub
```

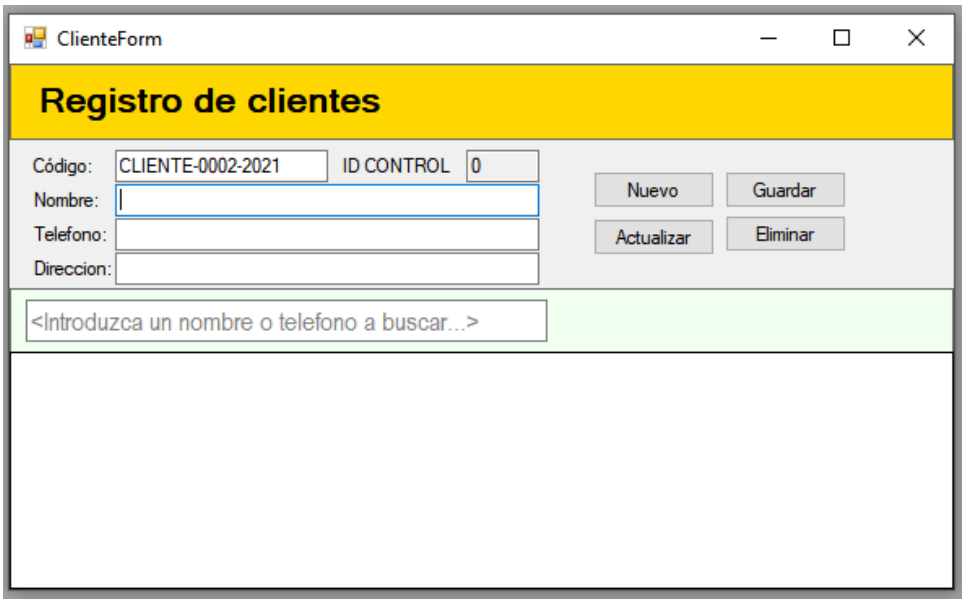


Figura 27. Interfaz de registro de clientes

Fuente: Propia

3.5.6. Módulo de recarga de tarjetas RFID

El monto a recargar se especificará después del registro del cliente para luego grabar los datos en la tarjeta RFID

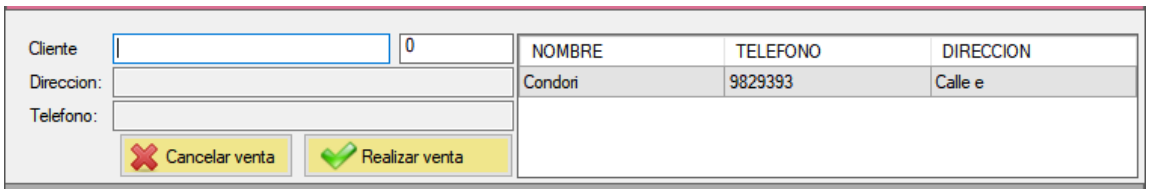


Figura 28. Interfaz de registro del monto a recargar

Fuente: Propia

3.5.7. Módulo de reportes

El módulo de reportes generara las etiquetas de los productos del supermercado para su lectura en el momento de la compra.

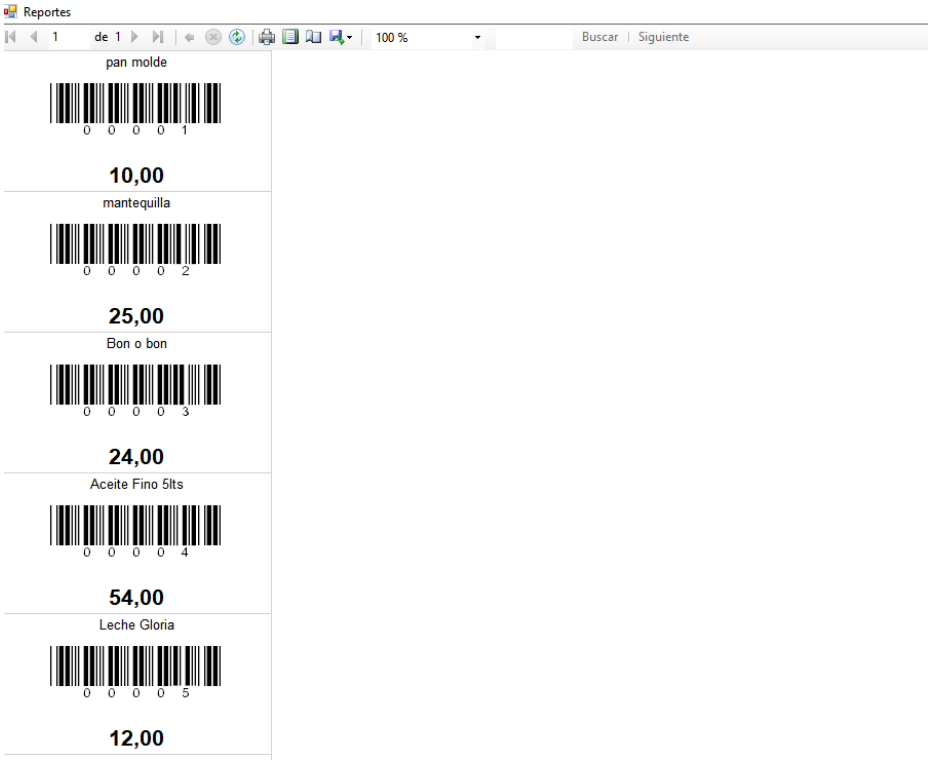


Figura 29. Reporte con las etiquetas de los productos

Fuente: Propia

Conclusiones y Recomendaciones

La tecnología RFID en Bolivia se utiliza en diferentes actividades ya sea en ventas de productos, tarjetas de presentación, control de asistencia, tarjetas de débito para el teleférico, y otras aplicaciones que requieren de profesionales en la materia para su implementación que puede aplicarse en distintos ámbitos.

El uso del RFID requiere de un control estricto en la encriptación de los datos, ya que es posible que personas malintencionadas puedan obtener información, clonando tarjetas o generando tarjetas que puedan ser usadas en algún lector de RFID.

La presente aplicación utiliza la tecnología RFID para mejorar las ventas masivas en supermercados, por lo tanto, este proyecto puede ser mejorado para ser distribuido en una cadena de supermercados.

Para lograr los objetivos planteados, teniendo como resultado la comunicación entre los diferentes dispositivos para el funcionamiento correcto del sistema, se requirió los conocimientos obtenidos teóricos y prácticos, desde la electrónica digital, telecomunicaciones, programación; por lo que podemos concluir que el desarrollo del trabajo resulto satisfactorio en la adquisición de conocimientos.

La aplicación de las tarjetas RFID son pasivas ya que se reducen los costos en relación a otros tipos de etiquetas, sin embargo, tienen un bajo alcance. Con lo cual se puede concluir que para utilizar mejor los beneficios de la tecnología RFID, mejorando y creando nuevas aplicaciones en distintas áreas, se deben usar etiquetas activas. Estos permiten almacenar mayor cantidad y de información y tienen mayores alcances de lectura.

Una de las ventajas principales al utilizar la tecnología RFID, hablando de la identificación inalámbrica, es que las tarjetas pueden ser leídas por el lector sin contacto. Por lo tanto, se

concluye que, para aprovechar los beneficios de la radiofrecuencia, esta tecnología puede utilizarse en aplicaciones que necesiten identificar objetos a grandes distancias.

Recomendaciones

Para el caso de extravió de tarjeta RFID será necesario reportar la perdida al administrador del sistema para evitar el uso indebido de esa tarjeta, se da total responsabilidad al cliente el cuidado de la tarjeta.

Para la implementación del hardware electrónico en el caso de RFID, se recomienda usar cables de corta distancia ya que se utiliza conexiones USB al ordenador. Para realizar la lectura o escritura en las tarjetas la máxima distancia es de 4cm. Para una correcta lectura o escritura de los datos de identificación en la tarjeta, en el proceso no se debe mover ni alejar la tarjeta.

Bibliografía

PAYMARKFAST (2018), Historia del RFID: Cómo, cuándo surge el RFID y su desarrollo hasta hoy en día. Recuperado el 16 de noviembre de 2021 de http://www.paymarkfast.com/historia-del-rfid/

KIMALDI (07/05/2021), RFID – Tecnología de identificación por radiofrecuencia. Recuperado el 10 de noviembre de 2021 de https://www.kimaldi.com/rfid_tecnologia_de_identificacion_por_radiofrecuencia/

CURSOSAULA21 (04/12/2019), RFID: todo lo que necesitas saber. Recuperado el 13 de noviembre de 2021 de https://www.cursosaula21.com/que-es-el-rfid/

ACTA.ES (2010), La tecnología RFID de Jose Manuel Huidoro ingeniero de telecomunicación. Recuperado el 13 de noviembre de 2021 de https://www.acta.es/medios/articulos/ciencias_y_tecnologia/058037.pdf

COGNEX (2021), Código de Barras 39. Recuperado el 11 de noviembre de 2021 de https://www.cognex.com/es-cl/resources/symbologies/1-d-linear-barcodes/code-39-barcodes

PUNTOFLOTANTE.NET (2021), PIC 18F2550, microcontrolador de alto rendimiento, 48 mhz, con puerto USB integrado, multifunciones. Recuperado el 13 de noviembre de 2021 de https://www.puntoflotante.net/18F2550.htm

PROGRAMARFACIL (02/06/2020), Lector RFID RC522 control de acceso RFID por Luis del Valle Hernández. Recuperado el 15 de noviembre de 2021 https://programarfacil.com/blog/arduino-blog/lector-rfid-rc522-con-arduino/

DIGIKEY (02/14/2019), Por qué y cómo usar la interfaz periférica serial para simplificar las conexiones entre distintos dispositivos. Recuperado el 16 de noviembre de 2021 de

https://www.digikey.com/es/articles/why-how-to-use-serial-peripheral-interface-simplify-connections-between-multiple-devices

PANAMAHITEK (15/10/2014), Cómo funciona el protocolo SPI Por Antony García González. Recuperado el 13 de noviembre de 2021 de http://panamahitek.com/como-funciona-el-protocolo-spi/

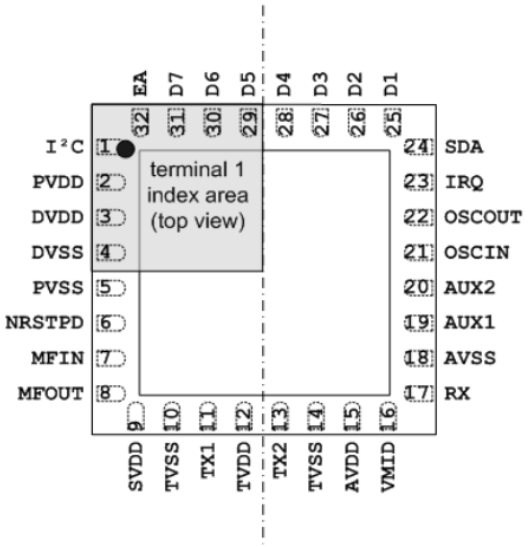
ROHDE SCHWARZ INTERNATIONAL (2021), Entendiendo el UART. Recuperado el 15 de noviembre de 2021 https://www.rohde-schwarz.com/lat/productos/prueba-y-medicion/osciloscopios/educational-content/entendiendo-el-uart_254524.html

CORE.AC (11/07/2011), Elección del microcontrolador 18f2550 por JORGE GÓMEZ MARCOS. Recuperado el 16 de noviembre de 2021 de https://core.ac.uk/download/pdf/30045385.pdf

CANALVISUALBASIC (2021), Manual Visual Basic la programación visual y guiados por eventos. Recuperado el 16 de noviembre de 2021 de http://www.canalvisualbasic.net/manual/manual-visual-basic/

Anexos

Pinning RC522



. Pinning configuration HVQFN32 (SOT617-1).

Pin description

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

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