ESCUELA POLITECNICA NACIONAL

ESCUELA DE FORMACION DE TECNOLOGOS

CONSTRUCCIÓN DE UN BASTÓN ELECTRÓNICO PARA CONTROL DE GUARDIAS DE SEGURIDAD

PROYECTO PREVIO A LA OBTENCION DEL TITULO DE TECNOLOGO EN ELETRÓNICA Y TELECOMUNICACIONES

CRISTIAN SANTIAGO BUSTOS SÁNCHEZ

cristian_santy11@hotmail.com

DIANA MARGARITA HERRERA NÚÑEZ

dayan_90@hotmail.es

DIRECTOR: ING. PABLO WIGERTO LÓPEZ MERINO

pwlopezm@hotmail.com

Quito, Mayo del 2013

DECLARACIÓN

Nosotros, BUSTOS SÁNCHEZ ,CRISTIAN SANTIAGO y HERRERA NÚÑEZ, DIANA MARGARITA, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través del presente declaramos que la Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

…….. ……..

Cristian Santiago Bustos Sánchez Diana Margarita Herrera Núñez

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Bustos Sánchez, Cristian Santiago y Herrera Núñez, Diana Margarita, bajo mi supervisión.

……..

Ing. Pablo Wigberto López Merino

Director del Proyecto

AGRADECIMIENTO

Mi gratitud, es principalmente para Dios, por haberme dado la vida y guiado mis pasos en el transcurso de mis estudios.

Agradezco a mi madre y hermanas, que han sido las personas que me han apoyado de manera incondicional y me han dado fortaleza.

Agradezco a los docentes de la Escuela Politécnica Nacional, quienes me han acompañado durante el largo camino de estudios, impartiendo sus conocimientos y orientación académica.

A mi tutor de tesis, quien me ha orientado en todo momento en la realización de este proyecto que es un escalón importante para cumplir mis objetivos y metas.

Cristian Santiago Bustos Sánchez

AGRADECIMIENTO

Este proyecto no habría sido posible sin la colaboración de varias personas que me han brindado su ayuda, sus conocimientos y su apoyo, quiero agradecerles a todos ellos cuanto han hecho por mí.

Quedo especialmente agradecida con mis padres que siempre me supieron apoyar en todo momento, brindándome su comprensión y cariño.

Agradezco a mis hermanos por su manera incondicional de ayudarme.

También quiero expresar mi agradecimiento al Ing. Pablo López tutor de tesis, quien supo guiarnos con sus conocimientos para la realización del proyecto.

Como también agradecer a las personas que conforman la empresa de Seguridad Aneta S.A. que nos para que nuestro proyecto se haga realidad.

Diana Margarita Herrera Núñez

DEDICATORIA

A mi madre y hermanos, porque creyeron en mi capacidad y me han dado ejemplos de superación y entrega para llegar a cumplir una de mis metas que es el culminar mis estudios superiores, siempre estuvieron impulsándome en los momentos que más necesitaba de ellos. También dedico a Diana ya que ha sido la persona que ha estado incondicional acompañándome y colaborando con la elaboración este proyecto.

Cristian Santiago Bustos Sánchez

DEDICATORIA

A mis padres Fausto y María, por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo, por su gran paciencia y amor hacia mi persona, los amo padres.

A mis hermanos Edison y Katty por apoyarme y estar siempre junto a mí en los buenos y malos momentos.

También a Cristian que me apoyado incondicionalmente en toda mi trayectoria universitaria en especial al realizar el proyecto de titulación.

Todo este trabajo ha sido posible gracias a ellos.

Diana Margarita Herrera Núñez

Resumen

Para la realización de este proyecto tomamos como referencia un sistema electrónico encargado de controlar el cumplimiento de las rondas de vigilantes.

La construcción de este dispositivo solventará uno de los principales problemas que tienen relación con el manejo del recurso humano, es decir la adulteración y/o manipulación de la información de los informes con su respectivo horario de trabajo, ya que permitirá un control eficiente del recurso que se dispone, adoptando un racional manejo del mismo que conlleve a obtener un mejor rendimiento del talento humano.

El proyecto investigativo tiene como finalidad la integración de varios elementos electrónicos y eléctricos dispuestos de forma sencilla en un dispositivo que registre y almacene información que se genera en la actividad efectuada por los guardias de seguridad.

El dispositivo dispondrá de un medio eficiente y de bajo costo de almacenamiento de los datos generados en la actividad, permitiendo al usuario final una interactividad para obtener información con mejor control administrativo.

Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ANEXOS

FOTOS

CAPÍTULO 1

1 FUNDAMENTOS TEÓRICOS

1.1 LECTORES BIOMÉTRICOS

La biometría es una herramienta que permite el reconocimiento único de humanos basados en uno o más rasgos físicos. El término se deriva de las palabras griegas "bios" de vida y "metron" de medida.

Las tecnologías aplicadas más comunes en este campo explotan el reconocimiento de voz, iris, sistemas dactilares y faciales, geometría de manos, olor corporal, reconocimiento del ADN, la forma de la oreja, etc.

En la tabla 1.1 se recogen las diferentes características de los sistemas biométricos:

	Ojo (Iris)	Ojo (Retina)	Huellas dactilares	Geometría de la mano	Escritura y firma	Voz	Cara
Fiabilidad	Muy alta	Muy alta	Alta	Alta	Media	Alta	Alta
Facilidad de uso	Media	Baja	Alta	Alta	Alta	Alta	Alta
Prevención de ataques	Muy alta	Muy alta	Alta	Alta	Media	Media	Media
Aceptación	Media	Media	Media	Alta	Muy alta	Alta	Muy alta
Estabilidad	Alta	Alta	Alta	Media	Baja	Media	Media

Tabla 1.1 Comparación de Sistemas Biométricos

Actualmente la huella digital es, sin duda, el sistema biométrico más avanzado y seguro del mercado.

El análisis de huellas dactilares con fines de identificación generalmente requiere la comparación de varias características del patrón de impresión[1].

Los tres patrones básicos de las crestas dactilares son el arco, lazo, y verticilo:

Arco: Las crestas entran a un lado del dedo, en el centro formando un arco, y luego salen por el otro lado del dedo.

Figura 1.1 Cresta Dáctil Arco

[1] Es la impresión artificial de la huella de alguno de los dedos.

Bucle: Las crestas entran por un lado de un dedo, forman una curva, y luego la salen por el mismo lado.

Figura 1.2 Cresta Dáctil Bucle

Verticilo: Las crestas que se forma circularmente alrededor de un punto central en el dedo.[2]

Figura 1.3 Cresta Dáctil Verticilo

[2] Bibliografía (Wikipedia)

1.1.1 Lector de huellas digitales

Es un dispositivo de seguridad encargado de detectar los relieves del dedo por medio de luz ó por medio de sensores eléctricos, posteriormente genera una imagen digital la cuál es enviada a la computadora y almacenada en una base de datos en que se le asocia con la información de una persona. Cada vez que se coloca el dedo sobre la superficie óptica del lector, este envía la información y la computadora determina a que persona corresponde ó si se trata de alguien no identificado. El nombre que se le da en Inglés es ("Finger Print Reader"), lo que traducido al español significa lector de impresión de dedo.

1.1.2 Tipos de huellas lectores de huellas digitales

- **Capacitivo:** El lente crea una imagen virtual mediante la resistencia capacitiva que ofrece el dedo al paso de la corriente. Esto impide que la suciedad y el cambio de color afecte a su perfecto funcionamiento para la identificación de personas.[3]
- **Óptico:** Tienden a estar iluminados con filtro de cristal, el lente guarda una imagen del dedo, su ventaja principal es su rapidez de búsqueda.

La proyección de una imagen óptica de huellas dactilares implica la captura de una imagen digital impresa usando la luz visible.

La capa superior del sensor, donde se coloca el dedo, se conoce como la superficie táctil, existe una capa inferior emisora de luz fosforescente que ilumina la superficie del dedo. La luz reflejada por el dedo pasa a través de la

[3] Bibliografía (Control-accesos, 2012)

capa de fósforo a una matriz de estado sólido pixeles[4], que captura una imagen visual de la huella dactilar.

Una desventaja de este tipo de sensor es el hecho de que las capacidades de formación de imágenes se ven afectadas por la calidad de la piel del dedo. Por ejemplo, un dedo sucio o marcado es difícil obtener una imagen correcta, esta tecnología de sensor no es susceptible a daños por descargas electrostáticas.

1.2 COMUNICACIÓN

Es el proceso mediante el cual se puede transmitir información de una entidad a otra. Los procesos de comunicación son interacciones al menos de dos agentes que comparten un mismo repertorio de signos y tienen unas reglas semióticas[5] comunes.[6]

Clasificación

- Comunicación Paralela
- Comunicación Serial

[4] un dispositivo de acoplamiento de carga.
[5] es la ciencia que trata de los sistemas de comunicación dentro de las sociedades humanas.

[6] Bibliografía (Wikipedia, 2012)

1.2.1 Comunicación Paralela:

La comunicación paralela transmite todos los bits de un dato de manera simultánea, por lo tanto la velocidad de transferencia es rápida, sin embargo tiene la desventaja de utilizar una gran cantidad de líneas, por lo tanto se vuelve más costoso y tiene la desventaja de atenuarse a grandes distancias, por la capacitancia entre conductores así como sus parámetros distribuidos.

La especificación IEEE 488 para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros.[7]

1.2.2 Comunicación Serial

La comunicación serial es un protocolo muy común para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora.

El puerto serial envía y recibe bytes de información un bit a la vez, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias, puede llegar a los 1200 metros.

La comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión:

- ✓ Tierra (o referencia)
- ✓ Transmitir
- ✓ Recibir

[7] Bibliografía (Alegsa)

Diferencias entre comunicación Serial y Paralela

La diferencia básica entre un canal de comunicación serial y uno paralela es el número de hilos o cables distintos en la capa física, usados para la transmisión simultánea desde un dispositivo.

La comunicación serie es más lento que la comunicación en paralelo.

Tipos de Comunicaciones Seriales

Existen dos tipos de comunicaciones seriales: la síncrona y asíncrona

1.2.2.1 Comunicación Serial Síncrona

Se necesitan 2 líneas, una línea sobre la cual se transmitirán los datos y otra la cual contendrá los pulsos de reloj que indicaran cuando un dato es válido.

Ejemplos: de este tipo de comunicación son los protocolos:

- I2C (Inter Integrated Circuit)
- SPI (Serial Peripherical Interface)

1.2.2.1.1 Transmisión serie Síncrona

Los caracteres no se envían por separado, sino formando bloques de gran longitud. Es más eficiente, ya que no hacen falta bits de inicio (start) y parar (stop).

Al no haber separación entre caracteres, la única forma de recibir correctamente es llevar la cuenta del nº de bits que compone cada carácter.

En la transmisión asíncrona, la sincronización se hace al inicio de cada carácter (flanco descendente del bit de start). En la síncrona hay que hacerlo

al inicio del bloque: cualquier pequeña diferencia en la frecuencia de los relojes es crítica. Deben usar una misma señal de reloj en lugar de tener cada uno la suya.

Para conseguir eso hay 2 opciones:

1. Usar una línea específica para transmitir la señal de reloj.
2. Modular la señal de reloj con la señal de información y enviarlas por una única línea.

Delante del bloque de datos se incluyen unos caracteres especiales de sincronización (conocidos tanto por Tx como por Rx) que permiten que el Rx tenga tiempo para adecuarse a la frecuencia con que vienen los datos. Así se puede extraer la frecuencia de reloj antes de que empiecen a llegar los datos (útil para el caso 2 de sincronización).

Entre cada 2 tramas o bloques de datos (cuando no hay caracteres que transmitir), en lugar de dejar la línea en reposo, se continúa transmitiendo los caracteres de sincronización, para mantener al Rx sincronizado.

1.2.2.2 Comunicación Serial Asíncrona

No se establece una temporización rígida, los datos pueden ser transmitidos en cualquier momento, el TX debe indicar al RX mediante alguna señal cuando se va a enviar un dato válido por la línea en forma de serie de bits. Además de los propios datos, se necesitan señales auxiliares de protocolo.

1.2.2.2.1 Transmisión serial Asíncrona

Los caracteres se transmiten por separado: TX carácter a carácter.

El tiempo de separación entre caracteres puede ser cualquiera (con una separación mínima), Sin embargo, dentro de cada carácter la temporización

sí es rígida: a cada bit le corresponde un instante y duración (Tbit) precisos. Por tanto, es TX asíncrona a nivel de caracteres, pero síncrona a nivel de bits.

De acuerdo con la norma de TX serie asíncrona más extendida (RS-404):

- Se definen dos niveles lógico: Nivel lógico 0 – SPACE, Nivel lógico 1 – MARK
- Cuando la línea no transmite se dice que está en estado de reposo: MARK.
- Cada carácter va precedido por el nivel lógico 0 durante un Tbit. Se denomina bit de start o bit de arranque.
- Nº de bits por carácter definido: 5, 7, 8.
- Se transmite primero el bit menos significativo.
- Todo carácter termina con un bit de stop (bit de parada), de duración 1, 1.5 ó 2 Tbit (si detrás no viene otro carácter la línea vuelve al estado de reposo).
- Después del bit más significativo, puede incluirse un bit de paridad, que es un mecanismo elemental de detección de errores:
    - Paridad Par: El bit de paridad se pone a 0 ó 1 de forma que el nº de 1s totales sea par.
    - Paridad Impar: El bit de paridad se pone a 0 ó 1 de forma que el nº de 1s totales sea impar.
- La referencia de tiempos que emplea el receptor para recibir los bits del dato es el flanco de bajada del bit de arranque.
- Para que el formato de la transmisión quede definido, TX y RX deben ponerse que el formato de la transmisión quede definido, TX y RX deben ponerse de acuerdo en:

- Velocidad de transmisión define el Tbit Existen velocidades normalizadas: de transmisión, define el Tbit. Existen velocidades normalizadas: 4800, 9600, 19200 baudios
- Nº de bits de datos.
- Uso de paridad y de qué tipo.
  - Sin Paridad
  - Con Paridad: Par, Impar, Fija a 0, Fija a 1
- Nº de bits de stop: 1, 1.5 ó 2.
- Niveles de tensión asociados a los niveles lógicos.

1.2.2.2.2 Transmisión serie síncrona y asíncrona

Integrados / dispositivos para comunicación serie comunes

- UART: Universal Asynchronous Receiver Transmitter
- USART: Universal Synchronous/Asynchronous Receiver Transmitter
- DUART: Dual Universal Asynchronous Receiver Transmitter[8]

Otro Tipo de comunicación serial

1.2.2.3 Comunicación Simplex

En este caso el emisor y el receptor están perfectamente definidos y la comunicación es unidireccional. Este tipo de comunicaciones se emplean, usualmente, en redes de radiodifusión, donde los receptores no necesitan enviar ningún tipo de dato al transmisor.

[8] Bibliografía (Castaño)

1.2.2.4 Comunicación Duplex, half duplex o semi-duplex

En este caso ambos extremos del sistema de comunicación cumplen funciones de transmisor y receptor y los datos se desplazan en ambos sentidos pero no de manera simultánea. Este tipo de comunicación se utiliza habitualmente en la interacción entre terminales y una computadora central.

1.2.2.5 Comunicación Full Duplex

El sistema es similar al duplex, pero los datos se desplazan en ambos sentidos simultáneamente. Para que sea posible ambos emisores poseen diferentes frecuencias de transmisión o dos caminos de comunicación separados, mientras que la comunicación semi-duplex necesita normalmente uno solo. Para el intercambio de datos entre computadores este tipo de comunicaciones son más eficientes que las transmisiones semi-dúplex.letty
Por lo tanto el transmisor y el receptor deberán tener los mismos parámetros de velocidad, paridad, número de bits del dato transmitido y de BIT de parada.

En circuitos digitales, cuyas distancias son relativamente cortas, se puede manejar transmisiones en niveles lógicos TTL (0-5V), pero cuando las distancias aumentan, estas señales tienden a distorsionarse debido al efecto capacitivo de los conductores y su resistencia eléctrica. El efecto se incrementa a medida que se incrementa la velocidad de la transmisión.

Todo esto origina que los datos recibidos no sean igual a los datos transmitidos, por lo que no se puede permitir la transferencia de datos. Una de las soluciones más lógicas, es aumentar los márgenes de voltaje con que

se transmiten los datos, de tal manera que las perturbaciones por causa de la línea se puedan corregir.[9]

1.3 IDENTIFICACIÓN POR RADIO FRECUENCIA (RFID)

1.3.1 RADIOFRECUENCIA

El término radiofrecuencia, también denominado espectro de radiofrecuencia o RF, se aplica a la porción menos energética del espectro electromagnético, situada entre unos 3 kHz y unos 300 GHz.[10]

La radiofrecuencia se puede dividir en las siguientes bandas del espectro:

Nombre	Nombre inglés	Abreviatura inglesa	Banda ITU	Frecuencias	Longitud de onda
				< 3 Hz	> 100.000 km
Frecuencia extremadamente baja	Extremely low frequency	ELF	1	3-30 Hz	100.000–10.000 km
Super baja frecuencia	Super low frequency	SLF	2	30-300 Hz	10.000–1.000 km
Ultra baja frecuencia	Ultra low frequency	ULF	3	300–3.000 Hz	1.000–100 km
Muy baja frecuencia	Very low frequency	VLF	4	3–30 kHz	100–10 km
Baja frecuencia	Low frequency	LF	5	30–300 kHz	10–1 km
Media frecuencia	Medium frequency	MF	6	300–3.000 kHz	1 km – 100 m
Alta frecuencia	High frequency	HF	7	3–30 MHz	100–10 m
Muy alta frecuencia	Very high frequency	VHF	8	30–300 MHz	10–1 m
Ultra alta frecuencia	Ultra high frequency	UHF	9	300–3.000 MHz	1 m – 100 mm
Super alta frecuencia	Super high frequency	SHF	10	3-30 GHz	100–10 mm
Frecuencia extremadamente alta	Extremely high frequency	EHF	11	30-300 GHz	10–1 mm
				> 300 GHz	< 1 mm

Tabla 1.2 Espectro de Radiofrecuencia

[9] Bibliografía (i-micro)
[10] Bibliografía (12Di1)

1.3.2 ONDAS DE RADIO

Las ondas de radio son un tipo de radiación electromagnética, tiene una longitud de onda mayor que la luz visible, se usan extensamente en las comunicaciones, Oscilan en las frecuencias que van desde VLF hasta EHF es decir desde 3 kHZ hasta 300 GHZ y pueden llegar a ser tan extensas que alcanzan cientos de kilómetros (cientos de millas) y tienen longitudes de onda 100Km hasta menores de 1 mm. [11]

También hay cuatro clases distintas de clasificación según su radio frecuencia:

Baja Frecuencia (9-135 KHz): Los sistemas que utilizan este rango de frecuencia tienen la desventaja de una distancia de lectura de sólo unos cuantos centímetros. Sólo pueden leer un elemento a la vez.

Alta Frecuencia (13.56 MHz): Esta frecuencia es muy popular y cubre distancias de 1cm a 1.5 m. Típicamente las etiquetas que trabajan en esta frecuencia son de tipo pasivo.

Frecuencia Ultra Elevada (0.3-1.2GHz): Este rango se utiliza para tener una mayor distancia entre la etiqueta y el lector (de hasta 4 metros, dependiendo del fabricante y del ambiente). Estas frecuencias no pueden penetrar el metal ni los líquidos a diferencia de las bajas frecuencias pero pueden trasmitir a mayor velocidad y por lo tanto son buenos para leer más de una etiqueta a la vez.

Microondas (2.45-5.8GHz): La ventaja de utilizar un intervalo tan amplio de frecuencias es su resistencia a los fuertes campos electromagnéticos, producidos por motores eléctricos, por lo tanto, estos sistemas son utilizados en líneas de producción de automóviles. Sin embargo, estas etiquetas

[11] Bibliografía (Windows 2 universe)

requieren de mayor potencia y son más costosas, pero es posible lograr lecturas a distancias de hasta 6 metros.

1.3.3 IDENTIFICACIÓN POR RADIO FRECUENCIA (RFID)

Identificación por radiofrecuencia es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas, tags RFID, el propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto mediante ondas de radio.[12]

Existen 3 componentes básicos en un sistema de RFID:

- **Tag:** etiqueta o transponder de RFID consiste en un pequeño circuito, integrado con una pequeña antena, capaz de transmitir un número de serie único hacia un dispositivo de lectura, como respuesta a una petición. Algunas veces puede incluir una batería.

- **Lector:** el cual puede ser de lectura o lectura/escritura, está compuesto por una antena, un módulo electrónico de radiofrecuencia y un módulo electrónico de control.

- **Controlador:** o un equipo anfitrión, comúnmente una PC o Workstation, en la cual corre una base de datos y algún software de control.

[12] Bibliografía (12Di)

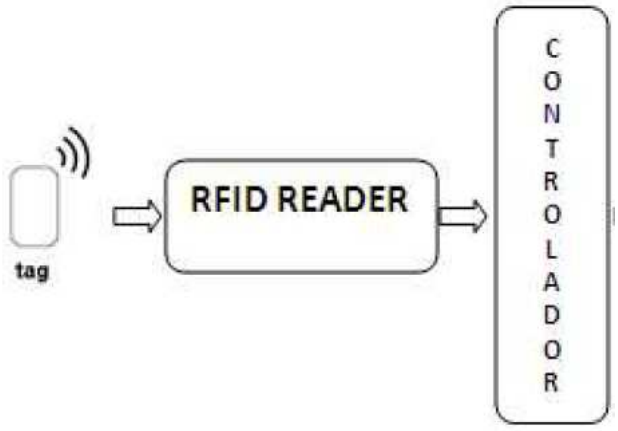


Figura 1.4 Componentes de un sistema de control RFID

1.3.3.1 Clasificación de Identificación Por Radio Frecuencia

Las tecnologías de auto identificación por radio frecuencia se clasifican en 3 tipos según el tipo del tag:

- **Sistemas pasivos:** Son etiquetas de RFID que no cuentan con una fuente de poder. Su antena recibe la señal de radiofrecuencia enviada por el lector y almacena esta energía en un capacitor. La etiqueta utiliza esta energía para habilitar su circuito lógico y para regresar una señal al lector. Estas etiquetas pueden llegar a ser muy económicas y de un tamaño no mayor a 0,3 mm. Las etiquetas pasivas de RFID pueden leerse a una distancia de aproximadamente 6 metros, son de sólo lectura, lo que significa que los datos que contienen no se pueden modificar o reescribir.

- **Sistemas activos:** Las etiquetas activas, también conocidas como transpondedores[13] porque contienen un transmisor que está siempre "encendido", se alimentan con una pila del tamaño aproximado de una moneda.

  Este tipo de etiquetas integra una electrónica más sofisticada, lo que incrementa su capacidad de almacenamiento de datos, interfaces con sensores, funciones especializadas, además de que permiten que exista una mayor distancia entre lector y etiqueta (20m a 100m).

  Las etiquetas activas de RFID pueden ser de lectura y escritura, lo que significa que los datos que contienen se pueden modificar por lo cual son más costosas y tienen un mayor tamaño
- **Sistemas Semi-Activos:** Emplean etiquetas que tienen una fuente de poder integrada, la cual energiza al tag para su operación, sin embargo, para transmitir datos, una etiqueta semi-activa utiliza la potencia emitida por el lector.
  En este tipo de sistemas, el lector siempre inicia la comunicación.

  La ventaja de estas etiquetas es que al no necesitar la señal del lector para energizarse (a diferencia de las etiquetas pasivas), pueden ser leídas a mayores distancias, y como no necesita tiempo para energizarse, estas etiquetas pueden estar en el rango de lectura del lector por un tiempo substancialmente menor para una apropiada lectura.

[13] es un aparato que recibe señales, de ser necesario las procesa, para luego ejercer su función de "respondedor", en este caso respondiendo con la emisión de la señal que recibió, pero en otra frecuencia, para evitar interferir con la señal que está recibiendo.

Esto permite obtener lecturas positivas de objetos moviéndose a altas velocidades.

Los tags activos como los pasivos se pueden subdividir de la siguiente forma:

- **Solo Lectura (RO):** En estos dispositivos, los datos son grabados en el tag durante su fabricación. Después de esto, los datos no podrán ser reescritos.
- **Una Escritura, Muchas Lecturas (WORM):** Un tag WORM, puede ser programado sólo una vez, pero esta escritura generalmente no es realizada por el fabricante sino por el usuario justo en el momento que el tag es creado.
- **Lectura y Escritura (RW):** Estas etiquetas, pueden ser reprogramadas muchas veces, típicamente este número varía entre 10,000 y 100,000 veces, incluso mayores. Esta opción de reescritura ofrece muchas ventajas, ya que el tag puede ser escrito por el lector, e inclusive por sí mismo en el caso de los tags activos. Estas etiquetas regularmente contienen una memoria Flash o FRAM para almacenar los datos.

1.3.3.2 Lectores de RFID

Un lector de RFID es también conocido como interrogador, el principal objetivo de un lector de RFID es transmitir y recibir señales, convirtiendo las ondas de radio de los tags en un formato legible para las computadoras.

El lector es necesario para transmitir energía al tag, para recibir desde el tag los datos correspondientes a las comunicaciones, y para separar estos dos tipos de señales.

La mayoría lectores son capaces de leer y escribir a un tag,

- La función lectora lee datos almacenados en el chip del tag.
- La función escritura escribe los datos pertinentes sobre el chip del tag.

La comunicación lector-tag puede utilizar cualquiera de las cuatro bandas de frecuencia: baja, alta, ultra alta, y de microondas.

Otra función de un interrogador es manejar la situación que se presenta cuando más de un tag responde simultáneamente a su interrogatorio. A esto se le llama procesamiento anti-colisión y se realiza a través de la electrónica del interrogador utilizando su software. Un lector tiene que estar conectado a través de cables de antenas para realizar la transmisión y recepción de señales.

1.3.3.3 COMPONENTES DEL LECTOR RFID

- **Transmisor:** El transmisor emite potencia y envía el ciclo de reloj a través de su antena hacia los tags que se encuentran dentro de su rango de lectura.
- **Receptor:** Este componente recibe las señales analógicas provenientes del tag a través de la antena y envía estos datos al microprocesador, donde esta información es convertida en su equivalente digital.
- **Antena:** Esta antena va conectada directamente al transmisor y al receptor. Existen lectores con múltiples puertos para antenas, lo que les permite tener múltiples antenas y extender su cobertura.
- **Microprocesador:** Este componente es responsable de implementar el protocolo de lectura empleado para comunicarse con tags compatibles. Decodifica y realiza verificación de errores a las señales recibidas. Adicionalmente, puede contener cierta lógica para realizar filtrado y procesamiento de bajo nivel de los datos leídos, esto es, eliminar lecturas duplicadas o erróneas.

- **Memoria:** La memoria es utilizada para almacenar información como los parámetros de configuración del lector, además de una lista de las últimas lecturas realizadas, de modo tal que si se pierde la comunicación con la PC, no se pierdan todos los datos.
- **Canales de Entrada/Salida:** Estos canales permiten al lector interactuar con sensores y actuadores externos. Estrictamente hablando, es un componente opcional, pero incluido en la mayoría de los lectores comerciales de la actualidad.
- **Controlador:** El controlador es el componente que permite a una entidad externa, sea un humano o un software de computadora, comunicarse y controlar las funciones del lector.

• **Interfaz de Comunicación:** Esta interfaz provee las instrucciones de comunicación, que permiten la interacción con entidades externas, mediante el controlador, para transferir datos y recibir comandos. Un lector puede tener distintos tipos de interfaz como se discute más adelante, por ejemplo: RS-232, RS-485, interfaz de red, entre otras.

• **Fuente de Alimentación:** Este componente provee de alimentación eléctrica a los componentes del lector y regularmente consiste en un cable con un adaptador de voltaje, conectado hacia la toma de corriente.

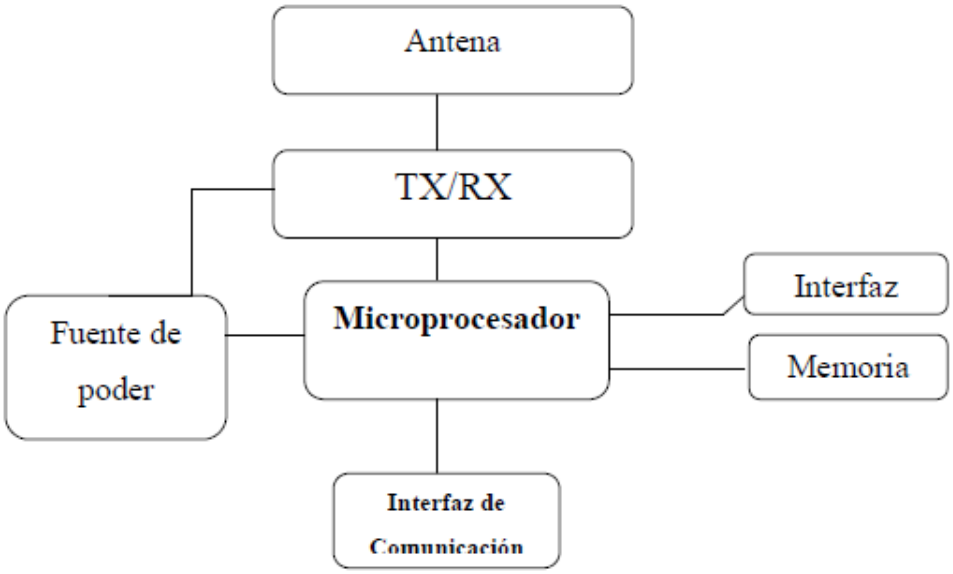


Figura 1.5 Componentes el lector RFID

1.3.3.4 Tipos de Conexión a una PC

Al desarrollar un sistema de RFID una consideración importante es la elección de la conectividad de red para los lectores de RFID. A continuación se muestra los diferentes tipos de conexiones:

- **RS-232**: Este protocolo provee sistemas de comunicación confiables de corto alcance. Tiene ciertas limitantes como una baja velocidad de comunicación, que va de 9600 bps a 115.2 kbps. El largo del cable está limitado a 30 metros, no cuenta con un control de errores y su comunicación es punto a punto.

  En las comunicaciones seriales RS-232 los valores para representar los 1's y 0's lógicos son muy diferentes de los que estamos acostumbrados a usar en el mundo TTL. Allí no existen los 5V (para el 1) y 0V (para el 0).

Para entenderlo más fácilmente veamos la figura 1.6, donde se compara la forma de onda de una señal RS-232 con la forma de onda de una señal digital convencional.

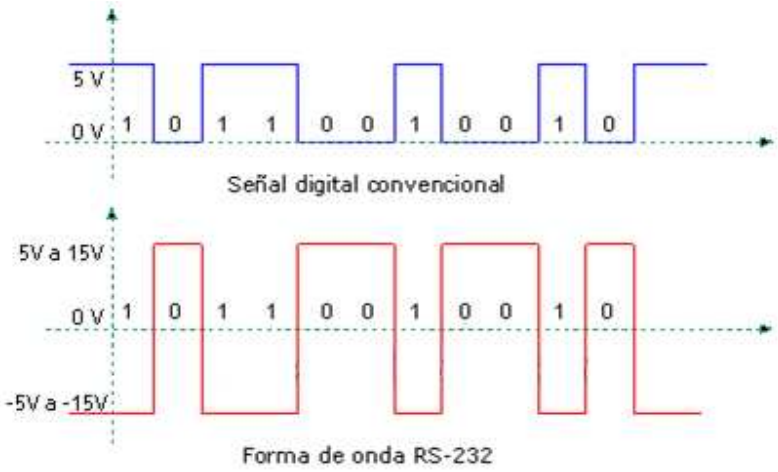


Figura 1.6 Señal Digital vrs Señal RS-232

Se puede notar la enorme diferencia: los 1 lógicos se representan con voltajes negativos y los 0 lógicos, por voltajes positivos; además del amplio rango de los voltajes.

Un 1 lógico se expresa por una tensión de –5V a –15V. Este estado se llama spacing.

Un 0 lógico se da cuando la tensión en cualquiera de las líneas es de +5V hasta +15V. Este estado se conoce como marking.

La mejora del RS-232 es el RS-485 que alcanza longitudes hasta de 1200 metros, velocidades hasta 2.5Mbps y un protocolo tipo bus lo cual permite a múltiples dispositivos estar conectados al mismo cable.

- **Ethernet:** La confiabilidad del protocolo TCP/IP sobre Ethernet asegura la integridad de los datos enviados y finalmente al ser la

infraestructura común para las redes, la mayoría de las instituciones ya cuentan con una red de este tipo, lo que permite una instalación más sencilla y menos costos de integración.

La velocidad de comunicación "ETHERNET" directamente viene asociada a la utilización de dicha tecnología, en este caso al ser en una red interna –LAN–, la distancia entre equipos activos de red es de hasta 100m, la velocidad depende del equipo de conmutación de datos que permite el funcionamiento de red. Actualmente se disponen de "Switchs" en el mercado de hasta 10Gbps.

- **Wireless 802.11:** Se utiliza en la actualidad en los lectores de RFID móviles. Además de que esta solución reduce los requerimientos de cables y por lo tanto de costos.

  El estándar IEEE[14] 802.11 define el uso de los dos niveles inferiores de la arquitectura OSI[15] (capas física y de enlace de datos), especificando sus normas de funcionamiento en una red inalámbrica. Los protocolos de la rama 802.x definen la tecnología de redes de área local y redes de área metropolitana.

- **WI-FI N o 802.11N**: En la actualidad la mayoría de productos son de la especificación **B** o **G,** sin embargo ya se ha ratificado el estándar **802.11N** que sube el límite teórico hasta los 600 Mbps. Actualmente ya existen varios productos que cumplen el estándar N con un máximo de 300 Mbps (80-100 estables).

  En relación a distancias el estándar no indica y/o define coberturas, en vista que depende directamente del equipo que brinda cobertura, las distancias máximas de separación entre el equipo de radio y el equipo activo de red.

[14] Instituto de Ingenieros Eléctricos y Electrónicos

[15] Es una normativa formada por siete capas que define las diferentes fases por las que deben pasar los datos para viajar de un dispositivo a otro sobre una red de comunicaciones.

- **Bus Universal en Serie (USB[16]):** Pensando desde la tendiente desaparición del puerto serial en las computadoras, algunos proveedores de lectores RFID han habilitado sus equipos para poder comunicarse mediante el puerto USB.

  El estándar USB maneja distancias de conexión de 0,5m hasta 5m. Resultando esta característica una desventaja para el uso en RFID, ya que se debe cablear desde el lector hasta la PC para el manejo de datos.

  Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos:

  - ***Baja velocidad (1.0):*** Tasa de transferencia de hasta 1,5 Mbps (192 KB/s).

  Utilizado en su mayor parte por dispositivos de interfaz humana como los teclados, los ratones (mouse), las cámaras web, etc.

  - ***Velocidad completa (1.1):*** Tasa de transferencia de hasta 12 Mbps (1,5 MB/s) según este estándar, pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones. Ésta fue la más rápida antes de la especificación USB 2.0, y muchos dispositivos fabricados en la actualidad trabajan a esta velocidad. Estos dispositivos dividen el ancho de banda de la conexión USB entre ellos.
  - ***Alta velocidad (2.0):*** Tasa de transferencia de hasta 480 Mbps (60 MB/s) pero por lo general de hasta 125Mbps (16MB/s). Está presente casi en el 99% de los PC actuales. El cable USB 2.0 dispone de cuatro líneas, un par para datos, una de corriente y un cuarto que es el negativo o retorno.

[16] Universal Serial Bus (Bus Universal en Serie), permite conectar periféricos a una computadora.

- ***Súper alta velocidad (3.0):*** Tiene una tasa de transferencia de hasta 4.8 Gbps (600 MB/s). La velocidad del bus es diez veces más rápida que la del USB 2.0, debido a que han incluido 5 conectores extra, desechando el conector de fibra óptica propuesto inicialmente, y será compatible con los estándares anteriores. usa un cable de 9 hilos.

La velocidad común de conexión de equipos RFID, utilizando puertos USB, es la denominada "Baja Velocidad" ya que es una velocidad común en la interconexión con hardware externo.

En resumen:

- El lector transmite una señal codificada de radiofrecuencia.
- El tag o tags presentes en el radio de influencia del lector son activados por la señal.
- El tag o tags responden al lector con su número de identificación.
- El lector captura los datos de los tags y los envía al microcontrolador.
- El microcontrolador procesa la información y valida si tiene permisos de acceso.[17]

[17] Bibliografía (MORÁN, 2011)

1.4 MICROCONTROLADOR

Es un circuito integrado programable el cual contiene todos los componentes de un computador, es un computador completo de limitadas prestaciones, que está contenido en un único chip.
Se emplea para controlar el funcionamiento de una única tarea y gracias a su reducido tamaño suele incorporarse en el propio dispositivo que controla.

Normalmente dispone de una memoria pequeña, en la que se almacena un solo programa.

Las líneas de entrada y salida se conectan con sensores y actuadores al dispositivo físico que controlan.

Una vez programado el microcontrolador sólo sirve para atender la tarea para la que ha sido programado.

Las ventajas de los microcontroladores son:

- Aumento de servicios y utilidades para el usuario.
- Aumento de la fiabilidad.
- Reducción de tamaño en el producto acabado.
- Mayor flexibilidad.

Los PIC son circuitos integrados de Microchip Technology Inc., que pertenecen a la categoría de los microcontroladores

El microprocesador básicamente está formado de la CPU y la ALU[18].

[18] Unidad de Central de Procesos es posible realizar una gran cantidad de operaciones aritméticas básicas (Suma, Resta, División y Multiplicación) además de realizar algunas operaciones Lógicas.

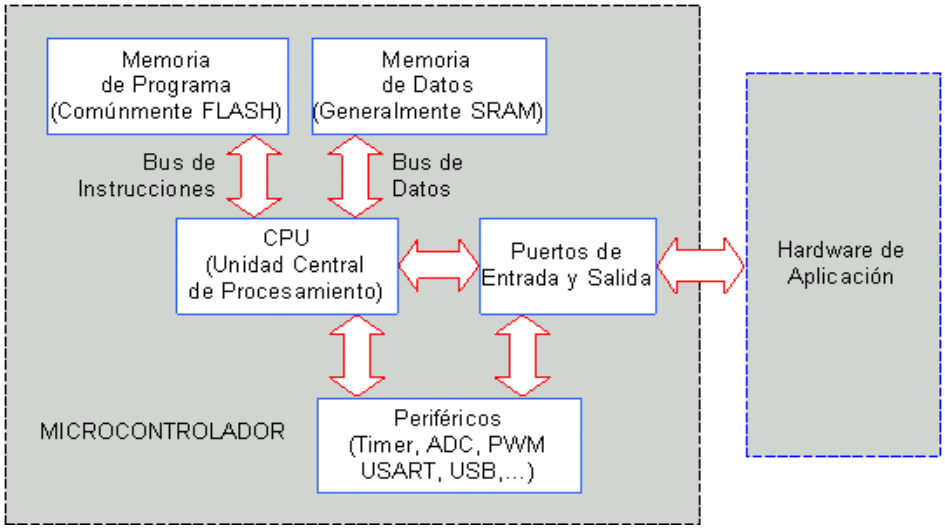


Figura 1.7 Microcontrolador

Los PIC utilizan arquitectura Harvard en lugar de la tradicional Von Neuman.

La arquitectura Harvard se caracteriza por tener dos memorias independientes, una para instrucciones y otra para datos. Cada una tiene su propio bus por lo que la CPU puede acceder simultáneamente a las dos. Esto agilita el proceso de lectura y ejecución de las instrucciones.

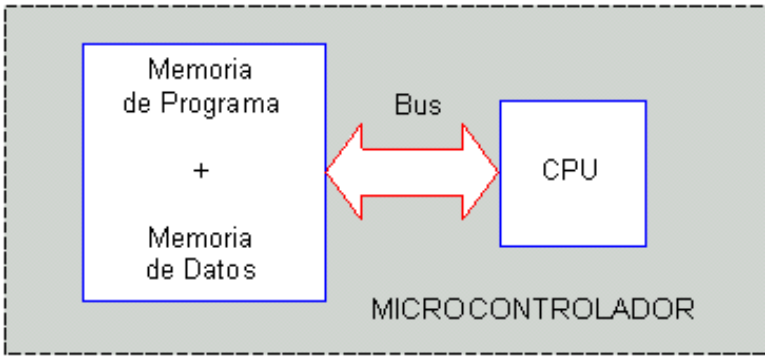


Figura 1.8 Arquitectura Von Neuman

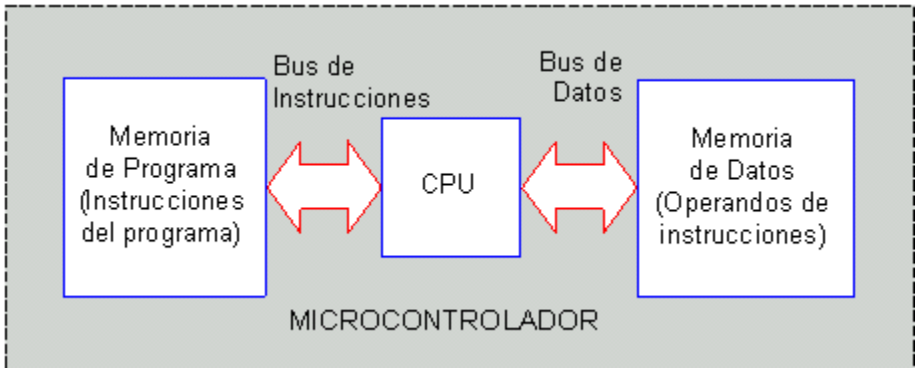


Figura 1.9 Arquitectura Harvard

1.4.1 Clasificación de los PIC

- Base Line: PIC10, PIC12, algunos PIC16
  Instrucciones de 12 bits

  Set de 33 instrucciones hasta 1MIPS, set de 35 instrucciones hasta 5MIPS, tienen muy poca memoria.

- Mid range: PIC16 y algunos PIC12
  Set de 35 instrucciones de 14 bits, 5MIPS
  Memoria hasta 8092 palabras.
- Enhanced: PIC18
  Set de 75 instrucciones de 16 bits
  Todos alcanzan velocidades de 10 MIPS, los PIC18 con bus USB llegan hasta 12 MIPS

1.4.2 Características de los PICs

- Tecnología CMOS
- Memoria de programa: Flash, OTP, ROM
- Puertos de I/O bidireccionables, configurables bit a bit

- Timers: temporizadores o contadores externos
- WatchDog: monitoriza que el PIC no se cuelgue
- ICSP: (In Circuit Serial Programming) Se programa el PIC con conexión serie.
- Bits de configuración (fuses)

1.4.3 Recursos avanzados

- Módulo PWM. Para control de velocidad de motores DC
- Conversores ADC
- Puerto serial Síncrono (MSSP). Para la comunicación con dispositivos que usan los buses I2C o SPI
- Puerto paralelo esclavo: SPP
- USART. Para comunicarse con protocolo RS232
- Módulo comparador analógico
- Módulo USB, Módulo CAN. Para hacer una pequeña red LAN. [19]

1.5 Reloj en tiempo Real (RTC)

Es un reloj de un ordenador, incluido en un circuito integrado, que mantiene la hora actual. Aunque el término normalmente se refiere a dispositivos en ordenadores personales, servidores y sistemas embebidos[20], los RTCs están presentes en la mayoría de los aparatos electrónicos que necesitan guardar el tiempo exacto.

La mayoría de los RTCs usan un oscilador de cristal, pero algunos usan la frecuencia de la fuente de alimentación. En muchos casos la frecuencia del

[19] Bibliografía (Costales, 2012)
[20] Sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas frecuentemente en un sistema de computación a tiempo real.

oscilador es 32.768 kHz. Ésta es la misma frecuencia usada en los relojes de cuarzo, y por las mismas razones, que la frecuencia es exactamente 215 ciclos por segundo, que es un ratio[21] muy práctico para usar con circuitos de contadores binarios simples.

Los RTCs a menudo tienen una fuente de alimentación alternativa, por lo que pueden seguir midiendo el tiempo mientras la fuente de alimentación principal está apagada o no está disponible. Esta fuente de alimentación alternativa es normalmente una batería de litio en los sistemas antiguos, pero algunos sistemas nuevos usan un supercapacitor[22], porque son recargables y pueden ser soldados. La fuente de alimentación alternativa también puede suministrar energía a una memoria no volátil.[23]

1.6 Módulo USB

El módulo USB es una entrada o acceso para que el usuario pueda almacenar o escribir información en una memoria flash.

Los módulos USB están diseñados para transmitir energía eléctrica al dispositivo que se encuentra conectado, de esta manera no se necesita de un cable adicional para conectarse a una toma de corriente, sirven para almacenar los datos en una memoria flash de alta capacidad.

Tiene 24 pines, proporciona acceso a los pines de la interfaz UART, FIFO (paralelo) y SPI, posee 13 pines que pueden ser configurados como entrada y salida.

[21] Es la razón o cociente de dos magnitudes relacionadas.
[22] , son como los capacitores normales pero almacenan hasta unas 10.000 veces más energía, ocupando el mismo tamaño.
[23] Bibliografía (Wikipedia, 2011)

1.7 Transistor

El transistor es un dispositivo electrónico semiconductor que cumple funciones de amplificador, oscilador, conmutador o rectificador. El término transistor es la contracción en inglés de transfer resistor (resistencia de transferencia). Actualmente se encuentran prácticamente en todos los aparatos electrónicos de uso diario: radios, televisores, reproductores de audio y video, relojes de cuarzo, computadoras, lámparas fluorescentes, tomógrafos, teléfonos celulares, etc.

Tipos de Transistores

1.7.1 Transistor de contacto puntual

Llamado también transistor de punta de contacto, fue el primer transistor capaz de obtener ganancia, inventado en 1947 por John Bardeen y Walter Brattain. Consta de una base de germanio, semiconductor para entonces mejor conocido que la combinación cobre-óxido de cobre, sobre la que se apoyan, muy juntas, dos puntas metálicas que constituyen el emisor y el colector. La corriente de base es capaz de modular la resistencia que se ve en el colector. En la actualidad ha desaparecido.

1.7.2 Transistor de unión bipolar

El transistor de unión bipolar, o BJT por sus siglas en inglés, se fabrica básicamente sobre un monocristal de Germanio, Silicio o Arseniuro de galio, que tienen cualidades de semiconductores, estado intermedio entre conductores como los metales y los aislantes como el diamante. Sobre el sustrato de cristal, se contaminan en forma muy controlada tres zonas, dos de las cuales son del mismo tipo, NPN o PNP, quedando formadas dos uniones NP.

La zona N con elementos donantes de electrones (cargas negativas) y la zona P de aceptadores o huecos (cargas positivas). Normalmente se utilizan como elementos aceptadores P al Indio (In), Aluminio (Al) o Galio (Ga) y donantes N al Arsénico (As) o Fósforo (P).

1.7.3 Transistor de efecto de campo

El transistor de efecto de campo de unión (JFET), fue el primer transistor de efecto de campo en la práctica. Lo forma una barra de material semiconductor de silicio de tipo N o P. En los terminales de la barra se establece un contacto óhmico, tenemos así un transistor de efecto de campo tipo N de la forma más básica. Si se difunden dos regiones P en una barra de material N y se conectan externamente entre sí, se producirá una puerta. A uno de estos contactos le llamaremos surtidor y al otro drenador. Aplicando tensión positiva entre el drenador y el surtidor y conectando la puerta al surtidor, estableceremos una corriente, a la que llamaremos corriente de drenador con polarización cero. Con un potencial negativo de puerta al que llamamos tensión de estrangulamiento, cesa la conducción en el canal.
El transistor de efecto de campo, o FET por sus siglas en inglés, que controla la corriente en función de una tensión; tienen alta impedancia de entrada.

Transistor de efecto de campo de unión, JFET, construido mediante una unión PN.

1.7.4 Fototransistor

Los fototransistores son sensibles a la radiación electromagnética en frecuencias cercanas a la de la luz visible; debido a esto su flujo de corriente puede ser regulado por medio de la luz incidente. Un fototransistor es, en esencia, lo mismo que un transistor normal.

Figura 1.10 Transistor

1.8 Buzzer

Este zumbador es un dispositivo de tipo audio electromagnético de señalización, que tiene una bobina en el interior de la cual oscila una placa de metal contra otro, que cuando la diferencia de voltaje dado produce un sonido de una frecuencia predeterminada. Usted debe ser consciente de este tipo de sonidos de timbre como sonido BIP en muchos aparatos

Los zumbadores electromagnéticos de alta fiabilidad son aplicables a los equipos electrónicos en general.

- Compacto, tipo pin terminal de timbre electromagnético con salida 2048 Hz.
- Tipo de pin construcción de terminal permite el montaje directo en placas de circuito impreso.[24]

Figura 1.11 Buzzer Electromagnético

[24] Bibliografía (scribd)

1.9 Regulador de voltaje

Es un dispositivo electrónico diseñado para mantener un nivel de voltaje constante.

Los reguladores electrónicos de tensión o voltaje se encuentran en dispositivos como las fuentes de alimentación de los computadores, donde estabilizan los voltajes DC usados por el procesador y otros elementos.

Un regulador simple puede hacerse de una resistencia en serie con un diodo. Debido a la curva característica del diodo, el voltaje a través del diodo cambia ligeramente debido a la corriente que pasa por él. Cuando la precisión en el voltaje no es necesaria, el diseño puede funcionar.

Los reguladores de voltaje retroalimentados operan al comparar el voltaje de salida actual con algún voltaje de referencia asignado. Cualquier diferencia es amplificada y usada para controlar el elemento de regulación para reducir el voltaje de error. Esto forma un lazo de control de realimentación negativa, haciendo que la ganancia tienda a incrementar la precisión de regulación pero reducir la estabilidad (se debe evitar la oscilación, durante los cambios de paso). También habrá una compensación entre la estabilidad y la velocidad de respuesta a los cambios.

1.9.1 Reguladores integrados

Son componentes muy parecidos a los transistores de potencia, suelen tener tres terminales, uno de entrada, un común o masa, y uno de salida, tienen una capacidad de reducción del rizado muy alta y normalmente sólo hay que conectarles un par de condensadores. Existen circuitos reguladores con un gran abanico de tensiones y corrientes de funcionamiento. La serie más conocida de reguladores integrados es la 78xx y la serie 79xx para tensiones

negativas. Los de mayor potencia necesitarán un disipador de calor, este es el principal problema de los reguladores serie lineales tanto discreto como integrado, al estar en serie con la carga las caídas de tensión en sus componentes provocan grandes disipaciones de potencia. Normalmente estos reguladores no son buenos para aplicaciones de audio por el ruido que pueden introducir en preamplificadores.

1.9.2 Reguladores conmutados

Los reguladores conmutados solucionan los problemas de los dispositivos anteriormente citados, poseen mayor rendimiento de conversión, ya que los transistores funcionan en conmutación, reduciendo así la potencia disipada en estos y el tamaño de los disipadores. Se pueden encontrar este tipo de fuentes en los ordenadores personales, en electrodomésticos, reproductores DVD, etc., una desventaja es la producción de ruido electromagnético producido por la conmutación a frecuencias elevadas, teniendo que apantallar y diseñar correctamente la PCB (Placa de Circuito Impreso) del convertidor.[25]

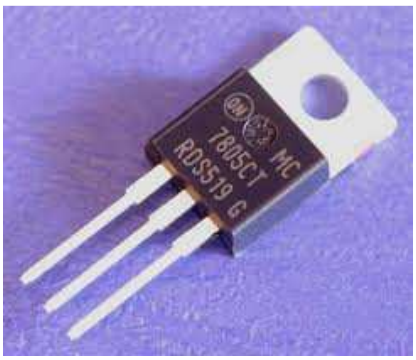


Figura 1.12 Regulador de Voltaje

[25] (Wikipedia, 2013)

1.10 Diodo

Es un componente electrónico de dos terminales que permite la circulación de la corriente eléctrica a través de él en un solo sentido. Este término generalmente se usa para referirse al diodo semiconductor, el más común en la actualidad; consta de una pieza de cristal semiconductor conectada a dos terminales eléctricos. El diodo de vacío (que actualmente ya no se usa, excepto para tecnologías de alta potencia) es un tubo de vacío con dos electrodos: una lámina como ánodo, y un cátodo.

De forma simplificada, la curva característica de un diodo (I-V) consta de dos regiones: por debajo de cierta diferencia de potencial, se comporta como un circuito abierto (no conduce), y por encima de ella como un circuito cerrado con una resistencia eléctrica muy pequeña. Debido a este comportamiento, se les suele denominar rectificadores, ya que son dispositivos capaces de suprimir la parte negativa de cualquier señal, como paso inicial para convertir una corriente alterna en corriente continua.

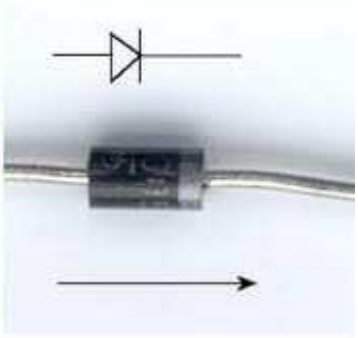


Figura 1.13 Diodo

1.11 Relé

El relé es accionado mediante la generación de un campo magnético al circular corriente en su bobina, lo que a consecuencia, atrae a un contacto cambiando el estado de reposo en el que se encontraba, el cual es retomado al dejar de circular dicha corriente por su bobina. Los contactos de un relé son siempre iguales, en reposo el contacto cierraun circuito a una de las salidas, mientras que del otro, se encuentra abierto, al circular corriente por su bobina invierte esta condición inicial.

2 terminales, son de la alimentación de la bobina, y la del medio es el terminal común de trabajo, mientras que los otros dos terminales, son sus terminales normalmente cerrado y normalmente abierto, es decir, inicialmente un contacto está como un circuito cerrado con el terminal común y el otro como circuito abierto. Al cambiar de estado por hacer circular corriente en su bobina, se invierte el estado.[26]

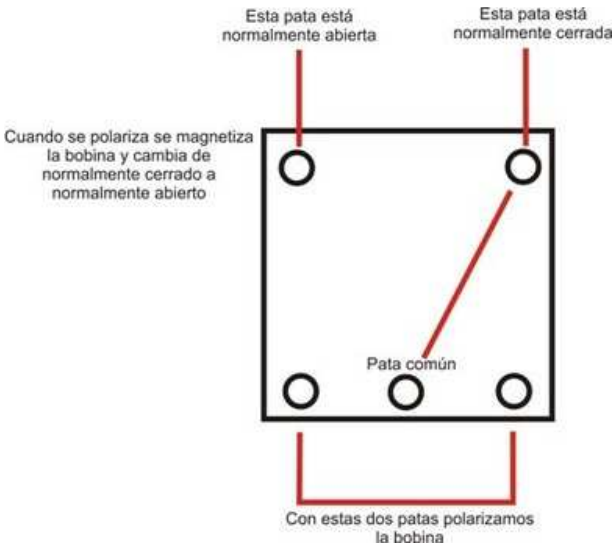


Figura 1.14 Relé de 5 Pines

[26] Bibliografía (mandobots)

CAPÍTULO 2

2 DESARROLLO DEL SISTEMA

El control de guardias por medio de un dispositivo electrónico (bastón) descartaría tanto el trabajo ineficaz como la modificación no autorizada de información de los reportes.

Este dispositivo funciona de una forma sencilla, para inicializarlo se pulsa el botón de prendido, en el cual se encenderá el lector de huellas digitales (el lente del lector encenderá con una luz azul), seguido, el led amarillo se encenderá y apagara continuamente hasta colocar sobre el lente el dedo en donde se enciende el led rojo, si la huella es correcta y se encuentra en la base de datos se encenderá el led verde, caso contrario, solo se enciende el led rojo y regresa al estado inicial. Una vez que se ha detectado correctamente la huella se enciende el led tomate el cual nos indica que debemos pasar cerca del punto (tarjeta tag), es decir, a menos de 12 centímetros del lector RFID que se encuentra en la base del bastón, se activara el buzzer al momento que el lector recopila la información del punto (tarjeta tag), si no existe ninguna tarjeta tag, el led tomate se mantendrá encendido varios segundos hasta que retorna a su estado inicial que es el led amarillo encendido.

Si se ha cumplido correctamente todos los parámetros, es decir, la huella correcta y la tarjeta tag existente se procede almacenar la información en la memoria flash, después de todo este procedimiento retorna a su estado inicial.

El guardia se encarga de realizar todo el procedimiento anterior, la ruta por la que debe cumplir su trabajo es por el punto 1 que se encuentra ubicado en el Edificio Administrativo, este se encuentra entre la calle Berlín y la Avenida Eloy Alfaro, continuando la ruta prosigue al punto 2 el cual se encuentra en la Gasolinera ubicado en la Av. Eloy Alfaro, el punto 3 se encuentra en la Garita en la calle Berlín, el punto 4 en la casa amarilla que se ubica en la calle Berlín y el punto 5 es la bodega ubicada en la calle Berlín y Av. 10 de Agosto, el guardia de seguridad tiene estrictas órdenes; que pase por cada punto a una hora especifica.

La manera para determinar los números de los puntos es en la programación, en el cual almacenamos el código de las 5 tarjetas en la memoria del PIC y las asignamos un punto específico, físicamente no podemos conocer cuál es el número del punto y a que código corresponde, por lo que necesitamos el lector RFID para obtener toda la información necesaria de la tarjeta tag.

El sistema también cuenta con un circuito que permite añadir huellas digitales en la base de datos del lector de huellas.

2.1 Diagrama de Bloques del Sistema

El bloque del microcontrolador interactúa con los bloques que corresponden: al lector de huellas, reloj a tiempo real, módulo USB y lector RFID, a continuación explicamos su interacción.

Para que el microcontrolador este actualizado en la hora y fecha se realiza una comunicación bilateral con el reloj a tiempo real el cual envía la información antes mencionada para su debida actualización.

También el microcontrolador se comunica con el lector de huellas de manera bilateral ya que envía la señal para que inicie el funcionamiento del lector de huellas y este a su vez envía al microcontrolador la información que obtuvo, posteriormente el microcontrolador también recibe la información de la tarjeta tag que permite identificar un punto, esto es enviado mediante el lector RFID, toda la información que se encuentra en el microcontrolador se envía al módulo USB para procesarla y posteriormente guardarla en una flash memory.

Para añadir nuevas huellas hemos implementado un circuito para añadir huellas el cual se comunica bilateralmente con el lector de huellas.

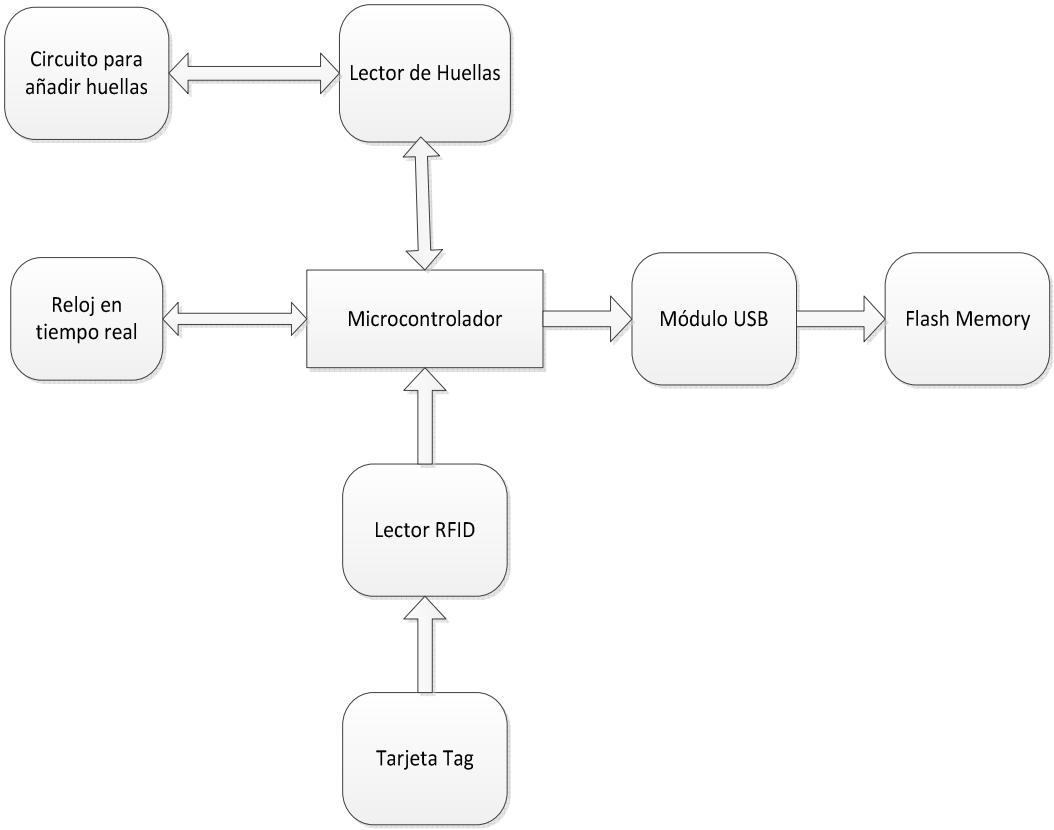


Figura 2.1 Diagrama de bloques del sistema

2.1.1 Circuito para añadir y modificar huellas

Este circuito se encuentra compuesto de un regulador de voltaje LM 7805 (el voltaje de la fuente nos brinda exactamente 5 voltios), dicho circuito tiene la finalidad de añadir huellas, se conecta físicamente el lector de huellas digitales a través de un cable plano de 10 pines (8 pines son conectados hacia el lector de huellas mientras que los dos restantes no los conectamos) por medio de conectores macho 100" P/CI 10 vías recto y conectores hembra, es una comunicación serial, también se comunica con la PC por medio de un cable adaptador RS232 a USB, dicha comunicación también es serial.

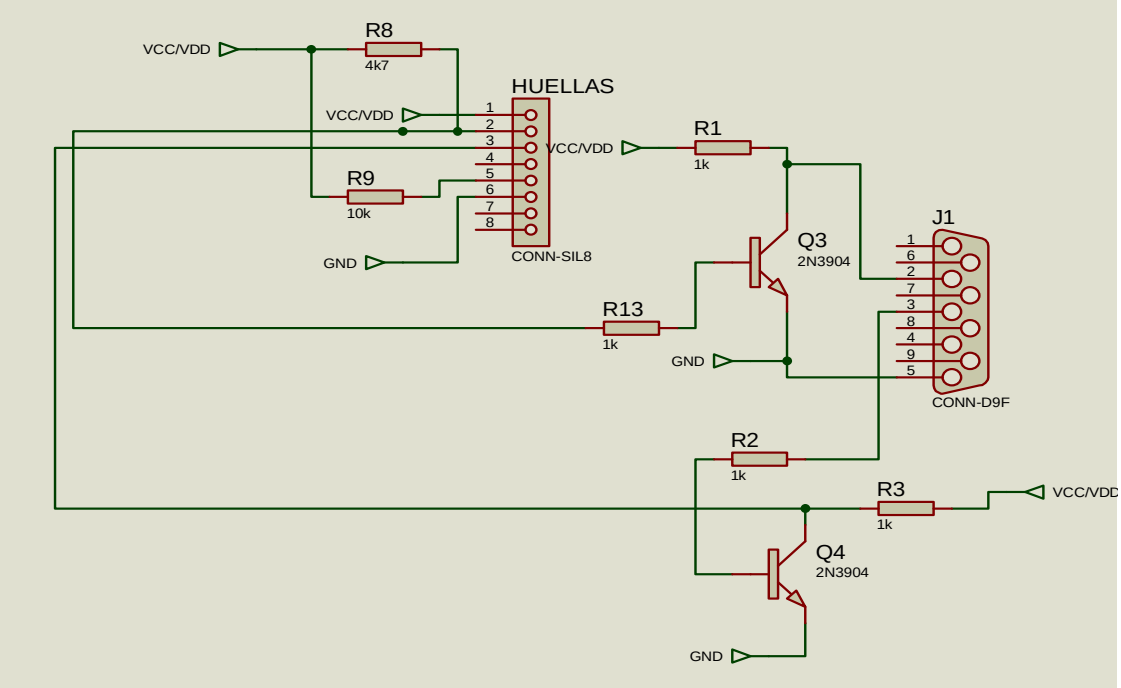


Figura 2.2 Diagrama del circuito para añadir huellas

2.1.2 Lector de Huellas Digitales

El lector de huellas digitales que hemos escogido es el Módulo de Verificación de huellas dactilares (SM-621).

Es un módulo que adopta un sensor de huellas dactilares óptica, de alto rendimiento procesador DSP[27].

Estos módulos son capaces de llevar a cabo el procesamiento de huellas digitales, generación de plantillas, coincidencia de plantilla, búsqueda de huellas digitales, almacenamiento de plantillas, etc.

Características

- Propiedad Intelectual Miaxis[28]
- Amplia gama de aplicaciones de huellas digitales
- Bajo Precio
- buena calidad de imagen,
- Algoritmo excelente
- Fácil de utilizar y ampliar
- Bajo consumo de energía
- Diferentes niveles de seguridad

Los parámetros técnicos del SM-621 se indica en la tabla 2.1 y la distribución de pines en la tabla 2.2.

2.1.2.1 Sistema de Verificación de huellas dactilares

• Detalle de huellas dactilares:

Los detalles de las huellas son extraídas mediante algoritmos y representa la información de las huellas digitales para posteriormente almacenamiento, verificación, búsqueda, etc.

[27] es único porque realiza el procesamiento de datos en tiempo real.
[28] empresa líder especializada en biometría.

- ***Verificación:***

Para identificar detalles de huellas dactilares y devolver los resultados: verificado o no verificado.

- ***Búsqueda (Searching):***

Conoce los detalles de las huellas dactilares que coinciden con la huella digital designados minucias, y devuelve resultados.

No.	Ítem	Parámetros	Condiciones de ensayo
1	Suministro de energía	3.6V - 7V	
2	Corriente de Trabajo	＜ 100mA	
3	Pico de Corriente	＜ 120mA	5V
4	Identificación de imagen Inscripción Tiempo	＜ 250ms	5V
5	1:1 Tiempo de verificación	＜ 600ms	Extracción de huellas + Verificación de huellas dactilares
6	1:240 Tiempo de búsqueda	＜ 2s	
7	almacenamiento de huellas Capacidad*	240/752/1776	
10	Interfaz externa	UART**	
11	Dimensión del módulo	56.0×38.5×8.5mm	
12	Dimensión del sensor	31×21×4.5mm	
13	Tamaño de la plantilla de huellas dactilares	256 bytes	

Tabla 2.1 Parámetros Técnicos SM-621

* Capacidad de almacenamiento se clasifica en tres niveles: 240, 752 y 1776.

** Tasa de baudios 57600bps.

PIN	NOMBRE	DEFINICIÓN	TIPO	FUNCIÓN
1	VIN	Suministro de energía	P	DC: 3.6V-7V
2	TD	Transmisión de datos	O	OC salida; el host estará conectado a impedancia pull-up
3	RD	Recepción de datos	I	Nivel TTL (3.3V or 5V)
4	NC	No está definido	..	..
5	EN	ABLE control	I	Cuando esté en suspenso o impedancia pull-up, el módulo funciona con normalidad. Cuando se conecta a Tierra, la fuente de alimentación interna se corta y el módulo no funciona.
6	GND	Tierra	P	Fuente de alimentación o tierra.

Tabla 2.2 Distribución de Pines SM-621

Para nuestro proyecto el pin 1 se conecta a VCC, el pin 2 está conectado a un circuito de conmutación, específicamente al terminal 4 del relé 1, el pin 3 está conectado al terminal 4 del relé 2, el pin 5 a VCC mediante una resistencia y el pin 6 a tierra.

2.1.3 Reloj en tiempo real

El reloj que se ha seleccionado para nuestro proyecto es el DS1307 el cual presenta ventajas por el bajo consumo de energía.

Es un dispositivo de bajo consumo de energía, con código binario decimal (BCD), reloj/calendario más 56 bytes de NV SRAM. Dirección y datos son transferidos a través de 2 hilos serie, bus bi-direccional. El reloj/calendario provee información de, segundos, minutos, horas, día, fecha, mes y año. El final de fecha de mes se ajusta automáticamente durante meses menores de 31 días, incluyendo correcciones para el año bisiesto. Funciona en formato

de 24 horas o en 12 horas con indicador AM/PM. El DS1307 tiene incorporado un circuito sensor de tensión que detecta fallas de energía y cambia automáticamente al suministro de batería de respaldo.

2.1.3.1 CARACTERÍSTICAS

- Reloj en tiempo real (RTC) Cuenta segundos, Minutos, horas, fecha del mes, mes, día de la semana, y año con año bisiesto Compensación Válido hasta 2100.
- 56-Byte, con respaldo de batería, no volátil (NV) de RAM para almacenamiento de datos.
- Interface Serie I2C.
- Onda-Cuadrada programable de la señal de salida.
- Detector Automático Fallo-Energía y Circuito Conmutación.
- Consume menos de 500nA en la batería -- Modo de copia de seguridad con el oscilador funcionando.
- Rango de temperatura Industrial Opcional: -40 °C a +85 °C
- Disponible en 8-Pin Plástico DIP o SO.

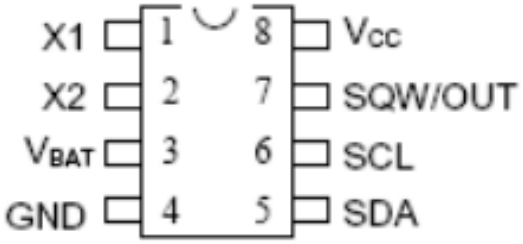


Figura 2.3 Asignación de pines

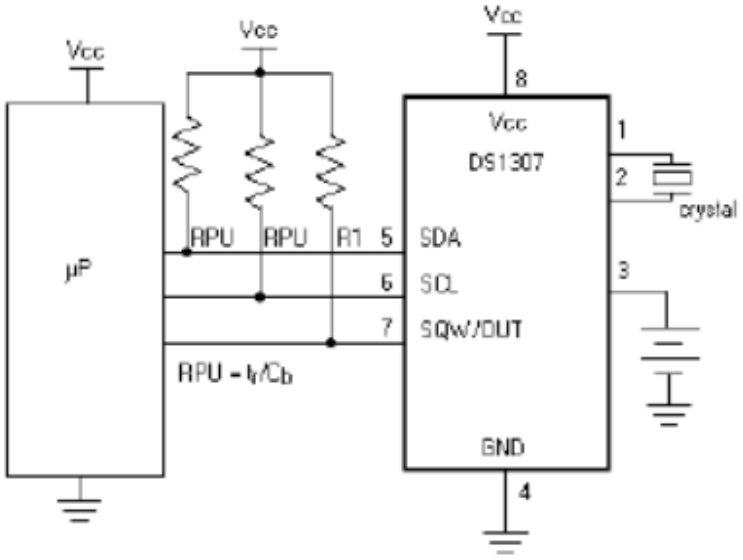


Figura 2.4 Circuito Típico

2.1.3.2 Funcionamiento

VCC, GND.- La alimentación DC del dispositivo se ofrece en estos pines. VCC es entrada de +5 V. Cuando se aplican 5V dentro de límites normales, el dispositivo es totalmente accesible y los datos pueden ser escritos y leídos. Cuando una batería de 3V se conecta al dispositivo y VCC es inferior a 1,25 x VBAT, se inhiben lectura y escritura. Sin embargo, la función de la hora normal no se ve afectada por la baja tensión de entrada.

Como VCC caiga por debajo de VBAT, la RAM y el cronometro se cambian a la fuente de energía externa (nominal 3.0V DC) en VBAT.

VBAT.- Entrada de Batería para cualquier célula de litio estándar 3V u otra fuente de energía. El voltaje de la batería debe ser mantenido entre 2,0 V y 3,5 V para su correcto funcionamiento. La tensión nominal de protección de escritura punto de disparo en el cual el acceso al RTC y la memoria RAM de usuario es denegado, es fijado por el circuito interno como nominal 1,25 x

VBAT. Un batería de litio con 48mAh o mayor mantendrá copia de seguridad del DS1307 durante más de 10 años en ausencia de energía a 25 º C.

SCL (Serial Clock Input).- SCL se utiliza para sincronizar el movimiento de datos en la interfaz serie, requiere una RPA (Resistencia de Polarización a Alto externa), corresponde al Pin 6 del DS1307 el cual va conectado hacia el Pin 18 del Microcontrolador.

SDA (Serial Data Input/Output).- SDA es el pin entrada/salida para el interfaz 2-hilos serie. El SDA es el pin de drenaje abierto, que requiere una RPA (Resistencia de Polarización a Alto externa), corresponde al Pin 5 del DS1307 el cual se conecta con el Pin 23 del Microcontrolador.

SQW/OUT (Onda Cuadrada/controlador de Salida).- Cuando se activa, el bit SQWE se establece en 1, el pin SQW/OUT es la salida de una de las cuatro frecuencias de onda cuadrada (1 Hz, 4 kHz, 8 kHz, 32 kHz). El pin SQW/OUT es de drenaje abierto y requiere una RPA[29]. SQW/OUT funcionará con cualquiera Vcc o Vbat aplicada.

X1, X2.- Conexiones para un cristal de cuarzo estándar 32.768kHz. El circuito oscilador interno está diseñado para funcionar con un cristal con una capacitancia de carga específica (CL) de 12.5pF.

El DS1307 también puede ser impulsado por un oscilador externo de 32.768kHz. En esta configuración, el pin X1 está conectado con el oscilador externo de la señal y el pin X2 está flotando.

La precisión del reloj depende de la exactitud del cristal y la precisión de igualdad entre la carga capacitiva del circuito oscilador y la carga capacitiva para los que el cristal se ha recortado. Se añadirá el error adicional de

[29] Resistencia de Polarización a Alto externa

frecuencia del cristal por la deriva causada por cambios de temperatura. El ruido exterior del circuito, junto al circuito oscilador puede resultar en el reloj corriendo rápido.

2.1.4 Microcontrolador

El microcontrolador que hemos escogido es el PIC 18F452 el cual se encuentra energizado por 5 voltios, en los pines 11 y 32 se encuentra la alimentación positiva del integrado (Vdd), en niveles de valor lógico alto (5Vdc), mientras que en los pines 12 y 31 se debe conectar a la señal de alimentación de nivel lógico bajo (0Vdc). Los pines 13 y 14 corresponden a las señales de reloj externas, que pueden provenir generalmente de una configuración de cristales de cuarzo o una resonancia de un circuito RC, dependiendo de la configuración que se utilice, también se podría aprovechar el pin 14 como una pin de E/S adicional como sexto bit del puerto A.

2.1.4.1 Características del PIC 18F452

Arquitectura RISC (Reduced Instruction Set Computer).

- ✓ Juego de instrucciones reducido para ejecución rápida.
- ✓ Oscilador hasta 40 MHz → 10 MIPs (Million Instructions Per second).
- ✓ Optimizado para compilación desde lenguaje C.
- ✓ Micro de 8 bits.

Arquitectura de memoria Hardvard:

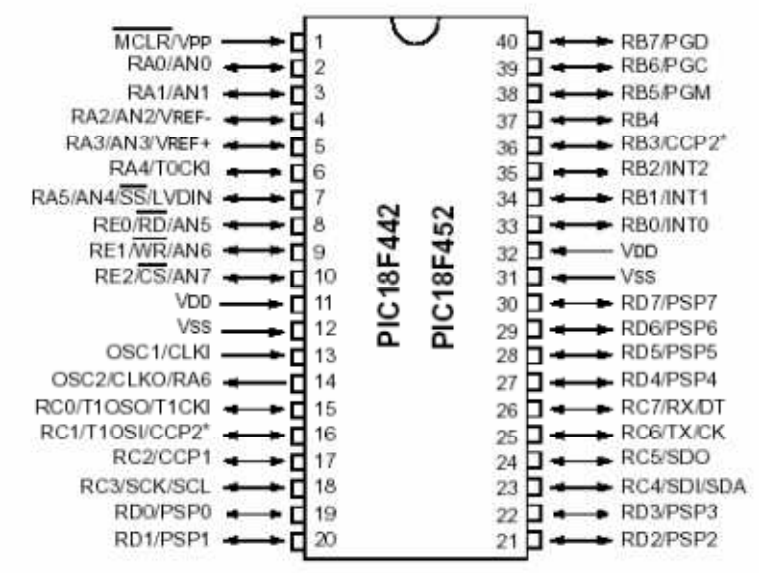
- ✓ Memoria interna de programa FLASH de 32 Kb
- ✓ Memoria interna RAM de 1536 bytes
- ✓ Memoria interna EEPROM de 256 bytes
- ✓ Contador de programa de 21 bits → hasta 2 Mb de memoria de programa
- ✓ Direccionamiento de 12 bits en memoria de datos → 4Kb

Periféricos integrados:

- ✓ Temporizadores, contadores, comparadores, unidades de captura
- ✓ Modulación en ancho de pulso PWM (Pulse Width Modulation)
- ✓ Interrupciones internas y externas
- ✓ Canal serie USART (Universal Synchronous/Asnynchronous Serial Receiver/Transmiter)
- ✓ Canal serie SPI (Serial Peripheral Interface)
- ✓ Canal serie I2C (Inter-Integrated Circuit)
- ✓ Puerto paralelo esclavo PSP (Parallel Slave Port)
- ✓ Conversión A/D de 10 bits
- ✓ Perro guardián WDT (Watchdog Timer). Universidade de Vigo – EUITI Informática Industrial 2004-2005

2.1.4.2 Descripción de Pines

El PIC18F452 es un diseño de microcontrolador con 40 pines, del cual se utilizan del tipo PDIP[30], el cual se detalla a continuación:



[30] Paquete en línea dual

Figura 2.5 PIC 18F452

En el pin 1 corresponde a MCLR/Vpp, siendo un pin de ingreso, donde MCLR es el master clear, siendo activado al nivel lógico bajo, y cuando recibe este nivel, lo que hace es resetear el PIC, también es el pin de voltaje de programa cuando actúa en este modo. Los pines 2, 3, 4, 5, 6, y 7 corresponden al puerto bidireccional A de 6 bits, el cual tiene sus funciones alternas como prosigue; aquí se encuentra un convertidor analógico a digital de 5 bits.

Pin No.	Nombre de Pin	Descripción
2	RA0	E/S digital 0
	AN0	Ingreso analógico 0
3	RA1	E/S digital 1
	AN1	Ingreso analógico 1
4	RA2	E/S digital 2
	AN2	Ingreso analógico 2
	VREF-	Ingreso del voltaje de referencia A/D
5	RA3	E/S digital 3
	AN3	Ingreso analógico 3
	VREF+	Ingreso del voltaje de referencia A/D
6	RA4	E/S digital 4
	T0CKI	Señal de ingreso externa para el timer 0
7	RA5	E/S digital 5
	AN4	Ingreso analógico 4
	SS	Selección de integrado SPI esclavo
	LVDIN	Detección de nivel bajo de voltaje

Tabla 2.3 Pines del Puerto A

En los pines 8, 9 y 10 corresponde al puerto bidireccional E, de 3 bits, el mismo que tiene como función alterna el control del puerto de

comunicaciones paralelo, encontrándose en estos pines las señales de lectura, escritura y habilitador.

Pin No.	Nombre de Pin	Descripción
8	RE0 RD	E/S digital 0 Señal de lectura para el puerto paralelo
9	RE1 WR	E/S digital 1 Señal de escritura para el puerto paralelo
10	RE2 CS	E/S digital 2 Señal de selección del integrado

Tabla 2.4 Pines del Puerto E

Los pines 15, 16, 17, 18, 23, 24, 25 y 26 pertenecen al puerto C con sus respectivas funciones alternas que cada uno posee, en este puerto como función alterna, se encuentra un timer, los dos generadores de señales PWM, un ingreso de captura y también los pines asociados a las comunicaciones series de este Pic, siendo la comunicación I2C, SPI, y USART; en la tabla 2.5 se indican las correspondencias de cada Pin con lo mencionado.

Pin No.	Nombre de Pin	Descripción
15	RC0	E/S digital 0
	T1OSO	Salida de oscilador del timer1
	T1CKI	Ingreso de la señal de reloj exterta
16	RC1	E/S digital 1
	T1OSI	Ingreso señal oscilador timer 1
	CCP2	Compare input/ouput 2, salida PWM 2
17	RC2	E/S digital 2
	CCP1	Compare input/ouput 1, salida PWM 1
18	RC3	E/S digital 3
	SCK	Señal de reloj en modo SPI
	SCL	Señal de reloj en modo I2C
23	RC4	E/S digital 4
	SDI	Dato de ingreso en modo SPI
	SDA	Dato de ingreso/salida en modo I2C
24	RC5	E/S digital 5
	SDO	Dato de salida en modo SPI
25	RC6	E/S digital 6
	TX	Transmisor en modo USART Asincrono
	CK	Señal de reloj en modo USART
26	RC7	E/S digital 7
	RX	Receptor en modo USART Asincrono
	DT	Datos en modo USART Asincrono

Tabla 2.5 Pines de Puerto C

En el puerto C del microcontrolador para nuestro proyecto se ha conectado de la siguiente manera:

El Pin 18 se conecta con el Pin 6 SCL del DS1307.

El Pin 23 se conecta con el Pin 5 SDA del DS1307.

El Pin 25 correspondiente a Tx del microcontrolador se conecta al terminal 5 del relé 2.

El Pin 26 correspondiente a Rx del microcontrolador se conecta al terminal 5 del relé 1.

El puerto D con un ancho de 8 bits se lo puede configurar como entrada o salida, también la de ser un puerto de comunicaciones esclavo paralelo.

Los pines 33, 34, 35, 36, 27, 38, 39 y 40 pertenecen al puerto B, también de 8 bits, siendo este puerto también como función principal el de ser de propósito general tanto de entrada como de salida, adicionalmente este puerto posee funciones alternas tales como las interrupciones externas, 1 módulo generador de PWM y un control de voltajes para las comunicaciones seriales, cada uno de estos pines lo podemos analizar en detalle en la tabla 2.6. Adicionalmente cuando este puerto se configure como salida, se debe tener en cuenta que internamente posee resistencia pull-up que deberán ser activadas de acuerdo al uso de puerto.

Pin No.	Nombre de Pin	Descripción
33	RB0 INT0	E/S digital 0 Interrupción externa 0
34	RB1 INT1	E/S digital 1 Interrupción externa 1
35	RB2 INT2	E/S digital 2 Interrupción externa 2
36	RB3 CCP2	E/S digital 3 Compara-Ingreso/salida 2 PWM2
37	RB4	E/S digital 4
38	RB5 PGM	E/S digital 5 Habilitador programacion ICSP
39	RB6 PGC	E/S digital 6 Clock de programacion ICSP
40	RB7 PGD	E/S digital 7 Dato de la programación ICSP

Tabla 2.6 Pines del Puerto B

En el puerto B del microcontrolador para nuestro proyecto se ha conectado de la siguiente manera:

El Pin 33 va conectado un diodo Led de color Rojo.

El Pin 34 va conectado un diodo Led de color Verde.

El Pin 35 va conectado un diodo Led de color Amarillo.

El Pin 36 va conectado un diodo Led de color Naranja.

El Pin 39 va conectado al terminal base del transistor 2N3904 Q1.

El Pin 40 va conectado al terminal base del transistor 2N3904 Q2.

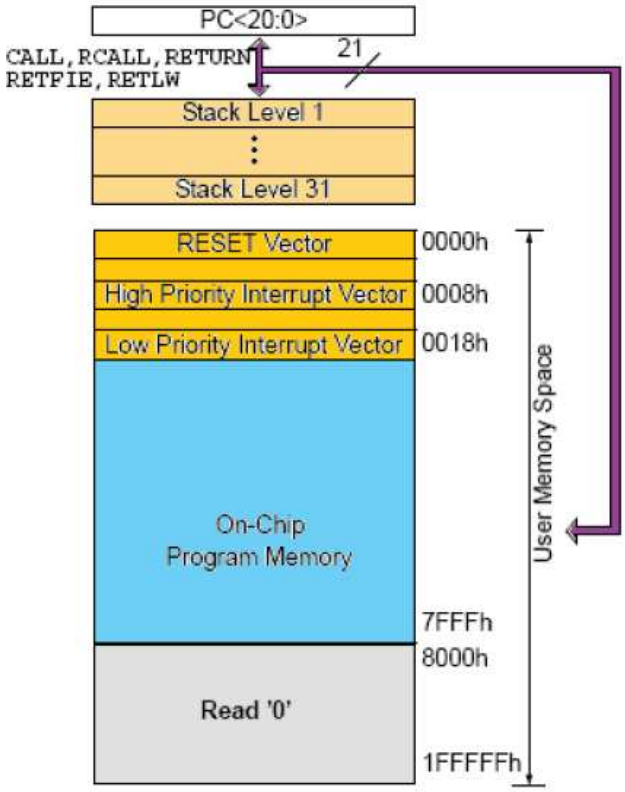


Figura 2.6 Organización de la memoria

El Stack[31] de los PIC18FXX2 posee 31 palabras de 21 bits, direccionadas por un apuntador de Stack de 5 bits.

Cada vez que se produce una interrupción o una llamada a subrutina, el apuntador del Pila se incrementa en 1 y el valor del contador de programa es almacenado en una de las palabras de 21 bits.

2.1.4.3 Temporizadores

Los PIC18 disponen de hasta 5 temporizadores:
TMR0: Temporizador de 8 o 16 bits de resolución que puede trabajar como contador.

TMR1 y TMR3: Temporizadores de 16 bits de resolución que pueden trabajar como contadores.

TMR2 y TMR4: Temporizadores especiales de 8 bits.

El valor de todos los temporizados es accesible por medio de registros (TMR0H:TMR0L, TMR1H:TMR1L, TMR2…). Todos los temporizadores llevan asociada una interrupción:

Por desbordamiento del temporizador/contador (TMR0, TMR1 y TMR3).

Por alcanzar el valor del registro de periodo (TMR2 y TMR4). El TMR3 y el TMR4 no se verán en detalle porque son prácticamente iguales que el TMR1 y el TMR2, respectivamente.

En el modo de 16 bits, el registro TMRH no contiene el valor real, sino que éste está almacenado en un buffer interno no accesible. El registro TMR0H se actualiza con el valor real durante la lectura del registro TMR0L. La

[31] Pila es un método de estructuración datos usando la forma LIFO (último en entrar, primero en salir), que permite almacenar y recuperar datos.

escritura del valor real en el buffer interno desde el registro TMR0H se realiza durante la escritura del registro TMR0L. Se debe leer primero el TMR0L y después el TMR0H Se debe escribir primero el TMR0H y después el TMR0L.[32]

2.1.4.4 Registros

2.1.4.5 INCONT

Función principal: controlar las interrupciones

Una interrupción, como el nombre lo sugiere, es un evento que hace que el microcontrolador deje de realizar lo que está haciendo y pase a ejecutar otra tarea. Al finalizar retorna a su actividad inicial.

Está ubicado en la localidad 0BH de la RAM

Su formato es:

Figura 2.7 Formato Registro INCONT

GIE: permiso global de las interrupciones, con 1 habilita

EEIE: habilitación de interrupción para grabación de la EEPROM

TOIE: 1 habilita interrupción de TMR0, 0 lo deshabilita

INTE: 1 habilita la interrupción RB0/INT, 0 la deshabilita

RBIE: habilita interrupciones en RB4 a RB7

TOIF: bandera de TMR0

INTF: bandera de interrupciones en RB0

RBIF: bandera de interrupciones en RB4 a RB7

[32] Bibliografía (slideshare)

2.1.4.6 Registro T0CON

Figura 2.8 Formato Registro T0CON

TMR0ON: Bit de puesta en marcha del Temporizador 0

T08BIT: Bit de configuración del modo 8-bit/16-bit:

T08BIT='0': Modo 16-bit

T08BIT='1': Modo 8-bit

T0CS: Bit de configuración del modo contador/temporizador

T0CS='0': Modo temporizador (TMR0 se incrementa en cada ciclo de instrucción FOSC/4)

T0CS='1': Modo contador (TMR0 se incrementa en cada transición de la línea RA4/T0CKI)

T0SE: Bit de selección de flanco en modo contador

T0SE='0': TMR0 se incrementa en los flancos de subida de la línea RA4/T0CKI

T0SE='1': TMR0 se incrementa en los flancos de bajada de la línea RA4/T0CKI

PSA: Bit de activación del pre-escalar

PSA='0': Pre-escalar activado

PSA='1': Pre-escalar no activado

T0PS2..T0PS0: Bits de selección del pre-escalar del Temporizador 0[33]:

[33] Bibliografía (MICROCONTROLADOR)

T0PS2	T0PS1	T0PS0	Valor del pre-escalar
0	0	0	1:2
0	0	1	1:4
0	1	0	1:8
0	1	1	1:16
1	0	0	1:32
1	0	1	1:64
1	1	0	1:128
1	1	1	1:256

Tabla 2.7 Valor del Temporizado

2.1.5 Lector RFID

El lector escogido es el RFID-ID12, con antena incorporada. Características: 5V de Alimentación y 30mA, 125KHz Frecuencia de Lectura, Compatible con Etiquetas EM4001 64-bit RFID, Salidas 9600bps TTL y RS232. [34]

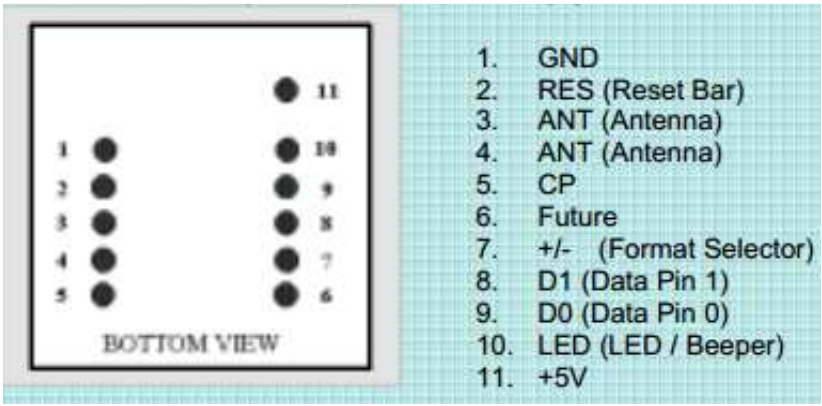


Figura 2.9 Lector RFID-ID12

[34] (Datasheet lector RFID)

PIN #	Descripción	ASCII	Imán de emulación	Wiegand26
1	Cero voltios y puesta a tierra de condensadores	GND 0V	GND 0V	GND 0V
2	Trabaja a +5V	Reset Bar	Reset Bar	Reset Bar
3	A la antena externa y ajuste Condensador	Antena	Antena	Antena
4	Antena externa	Antena	Antena	Antena
5	Tarjeta Presente	Sin función	Tarjeta Presente*	Sin función
6	Futuro	Futuro	Futuro	Futuro
7	Formato selector (+/-)	Junto a GND	Junto al pin 10	Junto a +5V
8	Dato 1	CMOS	Reloj *	Una salida
9	Dato 0	TTL de datos (invertida)	Dato *	Cero Salidas *
10	3.1KHz Lógicos	Buzzer / LED	Buzzer / LED	Buzzer / LED
11	Voltaje de alimentación DC	+5V	+5V	+5V

* Requiere resistencias Pull-up 4K7 a +5V

Tabla 2.8 Descripción de Pines y formatos de datos de salida

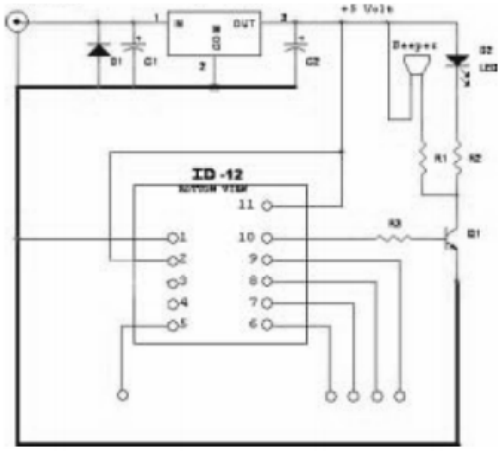


Figura 2.10 Circuito de polarización del ID-12

El Pin 1 va a tierra (GNR) por polarización, Pin 2 a VCC con una resistencia para setear el ID-12, Pin 7 va tierra ya que en el microcontrolador se programa en hexadecimal y el código de la tarjeta tag está en ASCII por lo que con este Pin transformamos de ASCII a Hexadecimal, Pin 8 se conecta al terminal 3 del relé 1 para transmitir los datos del ID, Pin 10 se conecta al

terminal positivo del Buzzer y el Pin 11 se conecta a VCC para su polarización.

2.1.6 Tarjeta Tag

Tarjeta para identificación por radiofrecuencia (RFID) básica escogida es de la empresa Sparkfun electronics que trabaja en los 125 KHz, viene con una identificación única de 32 bits, es blanca y medianamente flexible, es similar a una tarjeta de crédito.

Sus principales características: Esta basada en la norma ISO EM4001, la frecuencia de la portadora 125 KHz, ASK 2 kbps, Código Manchester, ID de 32-bits, Stream total de 64-bits, cada tarjeta posee un código único de fabricación que se representan en 12 caracteres, es una tarjeta pasiva.

Figura 2.11 Tarjeta tag

2.1.7 Módulo USB

El modulo escogido es el modulo VDIP1 por sus características favorables y de bajo costo, este modulo sirve para que el usuario pueda almacenar o escribir información en una memoria flash.

El módulo VDIP1 es un sistema embebido que permite escribir archivos .FAT con previa configuración de la memoria para que almacene los datos, se

recomienda que para agilitar su configuración la memoria flash no sea mayor a 2 GB. Este módulo tiene las condiciones necesarias para controlar un pendrive (memory flash) y lo necesario para ser controlado mediante un microcontrolador.

No sólo es ideal para el desarrollo y la creación rápida de prototipos de diseños VNC1L[35], sino también un atractivo descuento por volumen estructura.

Necesita de 5 Voltios para su alimentación auxiliar, buena potencia de salida de alimentación externa 3.3V/150mA y LEDs indicadores de tráfico

Figura 2.12 Modulo VDIP1

Es capaz de manejar la interfaz de host USB y las funciones de transferencia de datos, debido al microcontrolador incorporado y una memoria flash integrada, Vinculum[36]. Al interactuar con los dispositivos de almacenamiento masivo como unidades flash USB, Vinculum también maneja de manera transparente la estructura de archivos FAT[37] comunicación a través de

[35] Vinculum (VNC1L) es un dispositivo controlador USB que permite implementar gran número de aplicaciones relacionadas con dispositivos USB.
[36] Empresa que fabrica los módulos USB.
[37] son los sistemas de fichero utilizados en DOS y primeras versiones Windows.

UART, SPI o interfaces paralelas FIFO a través de aplicar sencillos conjuntos de comandos.

El protocolo UART sería una comunicación serial normal Transmisión (Tx) y Recepción (Rx), el protocolo SPI es más o menos parecido a la comunicación I2C.

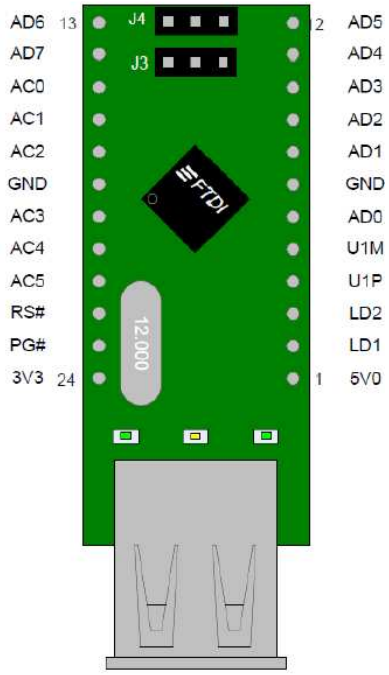


Figura 2.13 Pines del modulo VDIP1

PIN #	Nombre	Tipo	Descripción
6	TXD	salida	La salida de transmisión de datos asíncrona
8	RXD	entrada	Recibir la entrada de datos asincrónica
9	RTS #	salida	Solicitud de envío de salida de control / señal Handshake
10	CTS #	entrada	Cancelar el envío de entrada de control / señal Handshake
11	DTR #	salida	Terminal de datos Preparado para control de salida / Handshake señal
12	DSR #	entrada	Conjunto de datos de entrada y Control / Handshake señal
13	DCD #	entrada	Datos de Entrada de control
14	RI #	entrada	Indicador de llamada Contro l de entrada. Cuando el RemoteakeW a opción está habilitada en la EEPROM, teniendo RI # bajo se puede utilizar para reanudar el controlador USB de la PC Host desde la suspensión
15	TXDEN #	entrada	Habilitar Transmisión de datos RS485 para los diseños

Tabla 2.9 Configuración de Pines- Interfaz UART

Para polarización del modulo VDIP1 se utiliza el Pin 1 que va a Vcc de 5 voltios y el pin 7 a tierra.

Para nuestro proyecto utilizamos el pin 8 para receptar la señal que envía el PIC mediante una comunicación UART, este pin se encuentra conectado hacia el terminal 3 del relé 2.

También cancelamos el envió de señales de control que ingresa al modulo ya que no es necesario activarlos por lo que enviamos al pin 10 a tierra.

2.1.8 Memoria Flash

Para el proyecto hemos utilizado una memoria flash de 1 GB, para que funcione de manera correcta con el modulo VDIP1 se debe realizar los siguientes pasos:

1- Formatear dispositivo pendrive conectándolo a una PC, en FAT32

2- Entrar en ejecutar- CMD - y poner format G: /A:512 (donde G es la letra del pendrive)
3- Bajar último flash de vinculum desde http://www.vinculum.com/downloads.html, archivo para VDAP, reflash FTD.
4- Renombrar el archivo a FTRFB.FTD
5- Copiar archivo en pendrive
6- Para poder testear el VDIP ,conectar el VDIP a la computadora con cable rs232 y max232:

   pin 10 del vdip a masa

   pin 7 vdip a masa

   pin 1 vdip a +5v

   pin 6 vdip al pin 11 max232 ->tx

   pin 8 vdip al pin 12 max232 -> rx

   pin 13 max 232 al pin 2 hembra db9

   pin 14 max232 al pin 3 hembra db9

   pin 5 hembra db9 a masa (cable del rs-232)

   ambos jumpers del vdip entre pin 1 y 2

7- abrir el hyperterminal de windows y escoger el puerto COM que corresponda, ponerlo en modo 9600 8-1-n ninguno
8- properties- pestaña settings - ascii setup - todas opciones marcadas excepto force incoming data 7 bits y send line ends with line feed
9- alimentar vdip

10- conectar el pendrive al vdip y esperar actualización , ver leds vdip e hyperterminal indicando el final
11- desconectar el pendrive y borrarle el archivo .ftd y volver a conectarlo al VDIP
12- entrarle al hyperterminal fwv y nos da la versión actual del firmware
13- ipa (enter) (configura el VDIP de comandos para trabajar con valores en ascii.)
14- opw test.txt (enter) (abre/crea un archivo con nombre test.txt (no acepta nombres de más de 8 caracteres)
15- wrf 12 (enter y espera le ingresemos la próxima línea) (le indicamos que vamos a escribirle 12 caracteres al archivo)
16- hello world! (enter automático al llegar al llegar al límite de caracteres indicados en la línea anterior)
17- clf test.txt (enter) (cerramos archivo)

Otros elementos importantes del circuito

2.1.9 Batería de Litio

Es un dispositivo diseñado para almacenamiento de energía eléctrica que emplea como electrolito, una sal de litio que procura los iones necesarios para la reacción electroquímica reversible que tiene lugar entre el cátodo y el ánodo.

De alta calidad 7.4 v li-ion batería recargable pack, capacidad de 2,2 Amperios, dos células, incluye cargador.

2.1.10 Relé

Los relés son perfectos para trabajar con nuestro microcontrolador, ya que funcionan a 5V, sin necesidad de disponer de fuentes externas para activar su bobina.

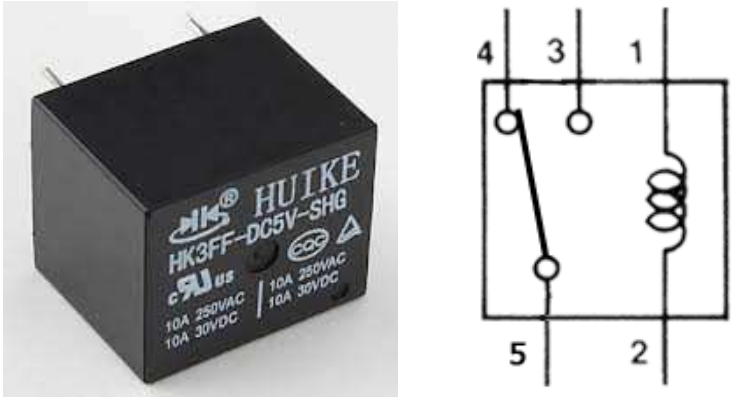


Figura 2.14 Relé de 5 terminales

2.1.11 Cristal

Es un componente electrónico que al ser conectado a un circuito apropiado genera una señal senoidal, en nuestro proyecto utilizaremos cristales como es para el microcontrolador que necesita un cristal de 20 MHz y para el DS1307 que necesita un cristal de 32768 MHz.

Figura 2.15 Cristal

Todo microprocesador o microcontrolador requiere de un circuito que le indique a qué velocidad debe trabajar. Este circuito es conocido por todos como un oscilador de frecuencia el cual consta de dos capacitores y un

cristal. Este oscilador es como el motor del microcontrolador por lo tanto, este pequeño circuito no debe faltar. En el caso del PIC18F452 el pin 13 y el pin 14 son utilizados para introducir la frecuencia de reloj.

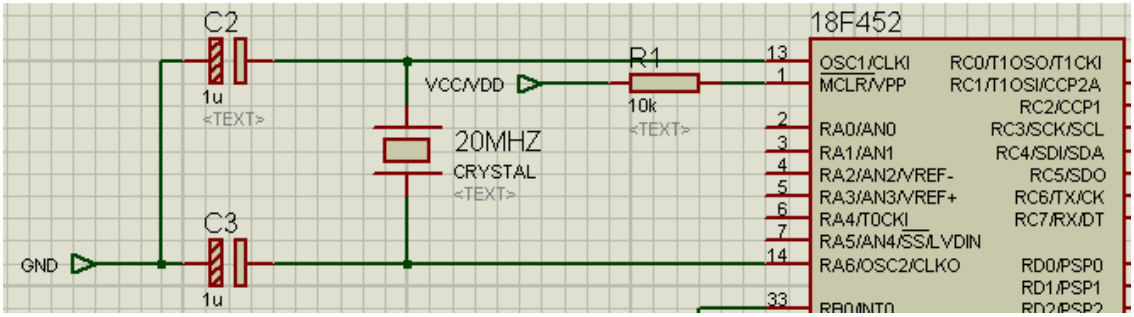


Figura 2.16 Circuito oscilador de frecuencia

2.1.12 Buzzer

Trabaja con un voltaje de 3-15 voltios (DC), corriente de 600mA, una frecuencia de 2048 Hz.

Parámetros	Valores
Voltaje de operación	3-15V DC
Corriente	60mA
Frecuencia	2048Hz
Presión sonora a 10cm	80db a 12V DC
Resistencia de Bobina dc	60 Ohms
Temperatura de Operación	-40 A +80°C

Tabla 2.10 Parámetros Técnicos del Buzzer

Figura 2.17 Buzzer

2.1.13 Transistor 2N3904

El Transistor 2N3904 Es uno de los más comunes Transistores NPN generalmente usado para amplificación. Este tipo de Transistor fue patentado por Motorola Semiconductor en los años 60, junto con el Transistor PNP 2N3906, y representó un gran incremento de eficiencia, con un encapsulado TO-92 en vez de el antiguo encapsulado metálico.

Está diseñado para funcionar a bajas intensidades, bajas potencias, tensiones medias, y puede operar a velocidades razonablemente altas. Se trata de un transistor de bajo coste, muy común, y suficientemente robusto como para ser usado en experimentos electrónicos.[1]

Es un transistor de 200 miliamperios, 40 voltios, 625 milivatios, con una Frecuencia de transición de 300 MHz, con una beta de 100. Es usado primordialmente para la amplificación analógica.[38]

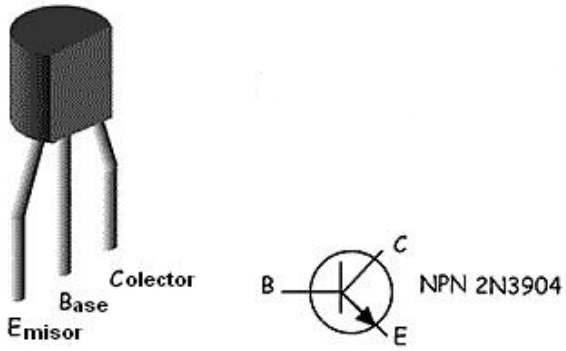


Figura 2.18 Transistor 2N3904

Para nuestro proyecto los transistores, están conectados de la siguiente manera, el terminal del colector hacia el terminal 2 de los relés, también hacia el ánodo del Diodo 1N4007 y el emisor a tierra.

[38] Bibliografía (wikipedia)

2.1.14 Diodo 1N4007

El diodo es un componente básico de los circuitos electrónicos. Su funcionamiento se basa en la unión de dos materiales semiconductores, uno de tipo P y otro de tipo N. Recuerda que al terminal que sale del semiconductor de tipo P se le denomina ánodo; y al que sale del semiconductor de tipo N, cátodo.

El diodo se puede emplear para controlar el paso de la corriente eléctrica por un circuito, a modo de interruptor.[39]

Características

Corriente nominal:	1 A
Pico de tensión inversa repetitiva:	50 ... 2000 V
Caja de plástico:	DO-41 DO-204AL
Peso aprox.:	0,4 g

Material plástico tiene clasificación UL 94V-0.[40]

Figura 2.19 Diodo 1N4007

El ánodo se conecta con el terminal colector del transistor 2N3904 y el cátodo se conecta a VCC.

[39] Bibliografía (ecnologiajavier)

[40] Bibliografía (Diotec Semiconductor)

2.1.15 Regulador de Voltaje LM7805

Son uno de los modelos más usados en circuitos electrónicos porque tienen una salida ideal para alimentar otros circuitos y microcomponentes. Este recibe un voltaje de hasta 20V de un lado y te entrega 5V del otro, siempre. Puede trabajar con 1A pero puede funcionar con un poco más.
El regulador de voltaje 7805 tiene 3 pines. El de en medio va a tierra (GND), el de la izquierda (IN) recibe el voltaje que deseas regular, puede ir desde 7V hasta 20V y el pin restante te entrega 5V.

En la parte superior tienen un agujero, porque este microcomponente se calienta con facilidad en especial cuando se le lleva al límite de su funcionamiento. Se puede emplear un tornillo y/o hacer contacto con una superficie metálica para ayudar a disipar el calor. También se puede encontrar unos disipadores especiales para el 7805 en tiendas de electrónica preferida, el que utilizaremos para nuestro proyecto es el disipador TO 220 PL.[41]

Los más comunes son la familia de los 78XX donde ** representa el voltaje que se obtendrá a la salida esto significa: si se tiene un 7805 entonces obtendrá un voltaje regulado a la salida a 5 volts.[42]

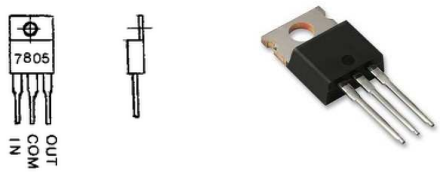


Figura 2.20 LM7805

[41] Bibliografía (jprogr)
[42] Bibliografía (yahoo)

2.2 Descripción de los diagramas esquemáticos del circuito

El Microcontrolador se sincroniza con el DS1307 para actualizar la hora y la fecha en su reloj interno, posteriormente emite una señal para la iniciación del lector de huellas mediante una comunicación serial, para la verificación de este proceso se enciende un led de color amarillo. Esto permite que el detector de huellas este atento a la espera de recibir información, es decir, espera la detección de una huella dactilar, para verificar este proceso tenemos un led de color rojo y verde, el led rojo nos indica que existe una huella sobre el detector, si esta se encuentra almacenada en la base de datos se encenderá el led verde y la información se enviara automáticamente hacia el Microcontrolador, caso contrario volverá a esperar la detección de otra huella.

Una vez que el microcontrolador recibe la señal del detector de huellas, este realiza la conmutación del relé 1 asignado para la recepción, que se encuentra normalmente cerrado con el detector de huellas, a normalmente abierto con el lector RFID 12, una vez realizada esta conmutación se encenderá el led tomate el cual es un indicador para pasar la tarjeta tag por el lector RFID 12 a una distancia máxima de 12 cm, si la tarjeta fue reconocida por el lector se activara el buzzer, el lector RFID registra el código del tag lo cual se encuentra asociada con un punto, este código y punto son enviados mediante comunicación serial al microcontrolador, caso contrario si no existe ninguna tarjeta que pase por el RFID mientras el led tomate se encuentre encendido este regresara a su estado inicial.

Al momento que se realizo la conmutación del relé 1, también se realizo la conmutación del relé 2 asignado para la trasmisión, esta normalmente cerrado con el lector de huellas cambiando su estado a normalmente abierto con el modulo VDIP1 (USB).

Una vez que el microcontrolador obtenga almacenado en su memoria EEPROM la información de la hora, fecha, código de la tarjeta tag, huella y el punto al que se asigno la tarjeta tag, toda esta información será enviada mediante una comunicación serial hacia el modulo VDIP1 que posteriormente se almacena en una Flash Memory.

Una vez realizados los procesos antes mencionados, los relés 1 y 2 vuelven a su estado inicial (normalmente cerrados) con lo cual se restablece el sistema.

Para agregar y modificar huellas se realiza mediante un circuito externo que tiene conectividad con el lector de huellas y a la PC a través de un cable adaptador USB a RS-232, con una interfaz amigable al usuario se llena el formulario con datos básicos de la persona a registrarse, a esta persona se le asigna un número de huella la cual se almacena en el lector de huellas, a este proceso pertenece el diagrama 2.2.

El diagrama de la figura 2.21 también se encuentra la parte del regulador de voltaje, en este caso la salida será de 5 V lo cual se ha utilizado el LM7805.

El diagrama de la figura 2.22 es el circuito para añadir huellas visualizado en Ares.

El diagrama de la figura 2.23 es el circuito principal del sistema visualizado en Ares.

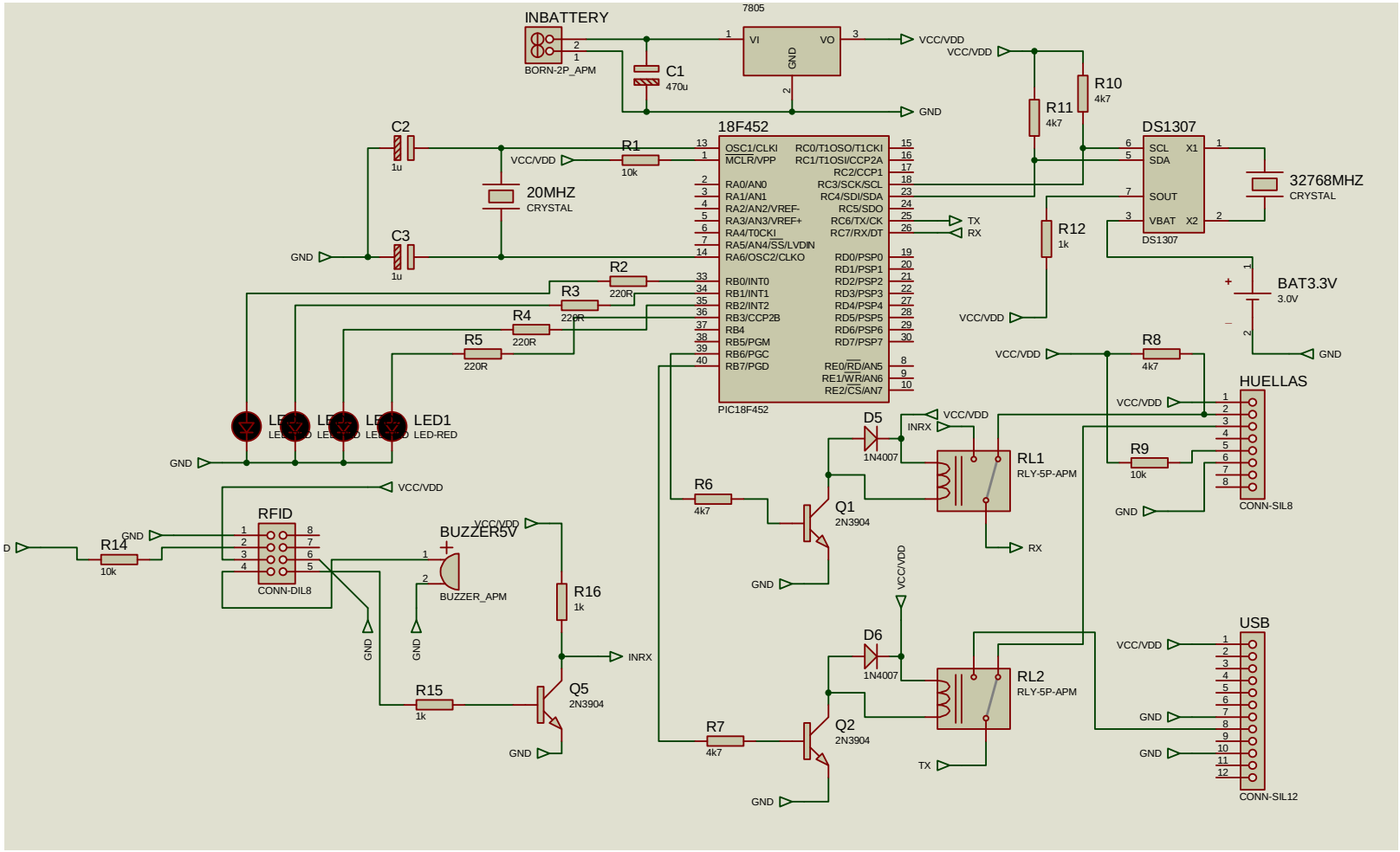


Figura 2.21 Diagrama esquemático del circuito

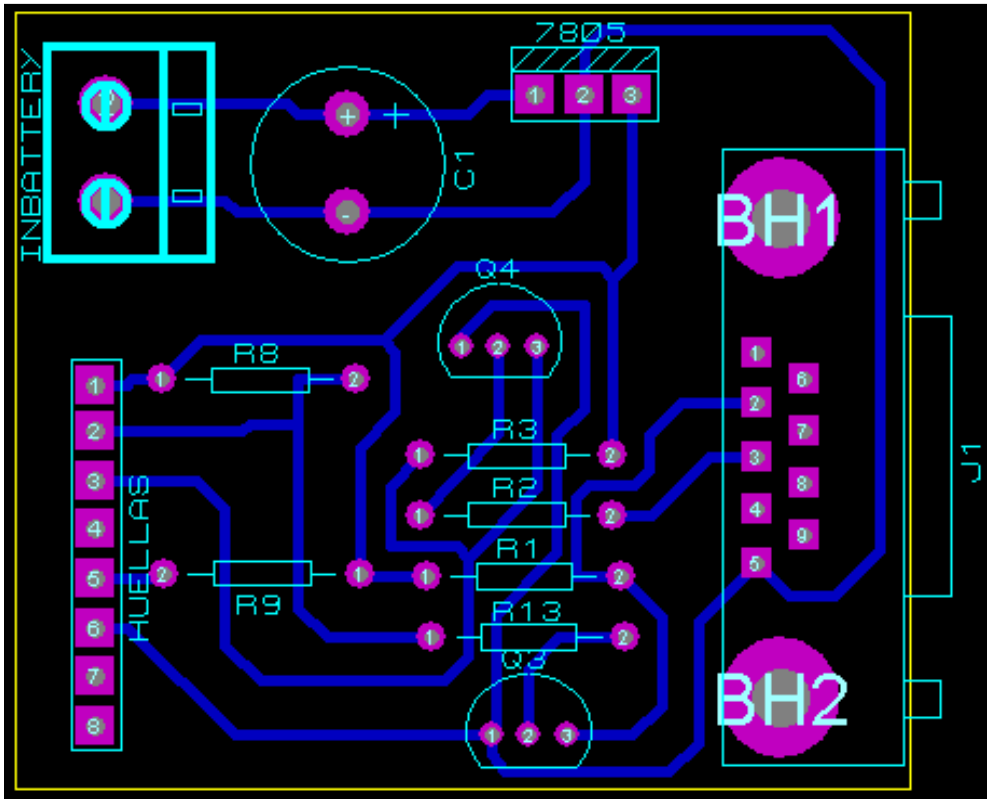


Figura 2.22 Circuito para añadir huellas en Ares

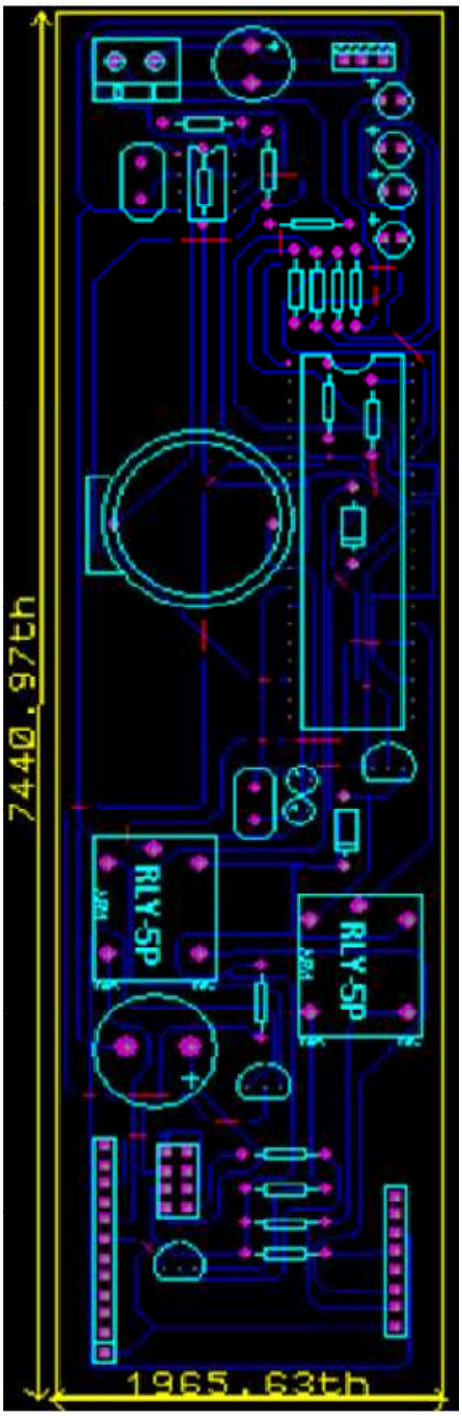


Figura 2.23 Circuito del sistema en Ares

2.3 Estructura del Estuche

El estuche ha sido realizado a base de fibra de vidrio porque es un material de bajo peso, facilitando transporte e instalación, alta resistencia mecánica, resistencia a corrosión y la intemperie, menor necesidad de mantenimiento, difícilmente se rompen o agrietan, se reparan fácilmente y a un costo económico.

El diseño del estuche se lo realizo de manera que todos los elementos encajen por lo que posee las siguientes dimensiones:

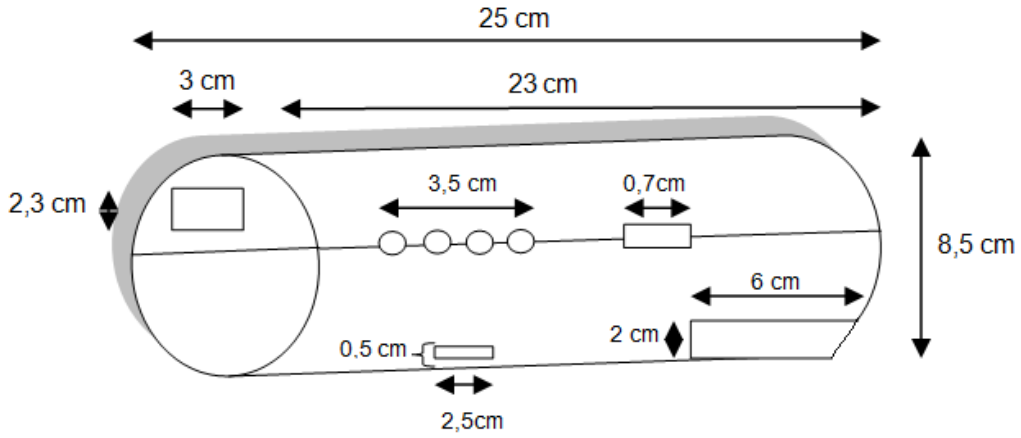


Figura 2.24 Estuche parte frontal

2.3.1 Fotografías del bastón

Vista Frontal del estuche

CAPÍTULO 3

3 Desarrollo del software del sistema e Implementación

3.1 Programación

3.1.1 Lenguaje de Programación

El microcontrolador ejecuta el programa cargado en la memoria Flash. Como es más fácil trabajar con el sistema de numeración hexadecimal, el código ejecutable se representa con frecuencia como una serie de los números hexadecimales denominada código Hex.

Como el proceso de escribir un código ejecutable era considerablemente arduo, en consecuencia fue creado el primer lenguaje de programación denominado ensamblador (ASM). Siguiendo la sintaxis básica del ensamblador, era más fácil escribir y comprender el código. Las instrucciones en ensamblador consisten en las abreviaturas con significado y a cada instrucción corresponde una localidad de memoria. Un programa denominado ensamblador compila (traduce) las instrucciones del lenguaje ensamblador a código máquina (código binario).

Para nuestro proyecto hemos escogido Mickro C.

3.1.2 Fases de Compilación

El archivo fuente contiene el código en mikroC que se escribe para programar el microcontrolador, el preprocesador se utiliza automáticamente por el compilador al iniciarse el proceso de la compilación. El compilador busca las directivas del preprocesador (que siempre empiezan por '#') dentro del código y modifica el código fuente de acuerdo con las directivas. En esta fase se llevan a cabo inclusión de archivos, definición de constantes y macros etc, lo que facilita el proceso. Más tarde vamos a describir estas directivas en detalle. El analizador sintáctico (parser) elimina toda la información inútil del código (comentarios, espacios en blanco). Luego, el compilador traduce el código a un archivo binario denominado archivo .mcl. El enlazador (linker) recupera toda la información requerida para ejecutar el programa de los archivos externos y la agrupa en un solo archivo (.dbg). Además, un proyecto puede contener más de un archivo fuente y el programador puede utilizar funciones predefinidas y agrupadas dentro de los archivos denominados librerías. Por último, el generador .hex produce un archivo .hex., es el archivo que se va a cargar en el microcontrolador.

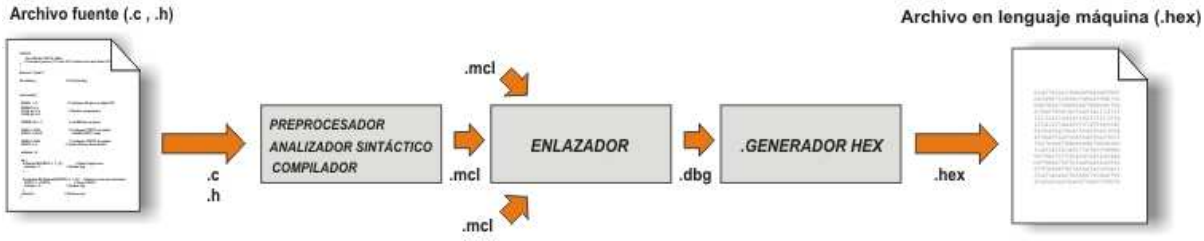


Figura 3.1 Fases de Compilación

3.1.3 Mickro C

El lenguaje mikroC desarrollado por Mikroelektronika, es muy similar al C estándar, no obstante en determinados aspectos difiere del ANSI estándar en algunas características. Algunas de estas diferencias se refieren a las mejoras, destinadas a facilitar la programación de los microcontroladores PIC, mientras que las demás son la consecuencia de la limitación de la arquitectura del hardware de los PIC. El término C se utilizará para referirse a las características comunes de los lenguajes C y mikroC.

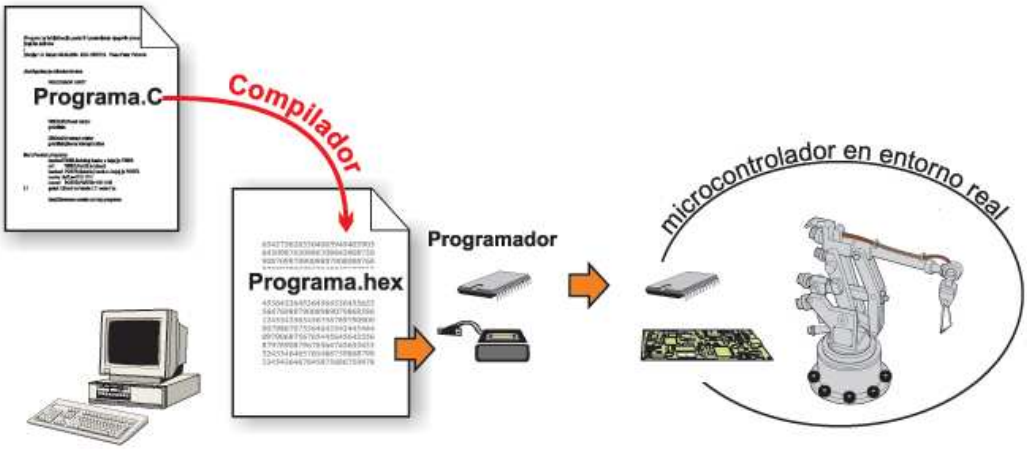


Figura 3.2 Proceso de almacenar el programa en el PIC

3.1.4 Diagrama de Flujos del sistema

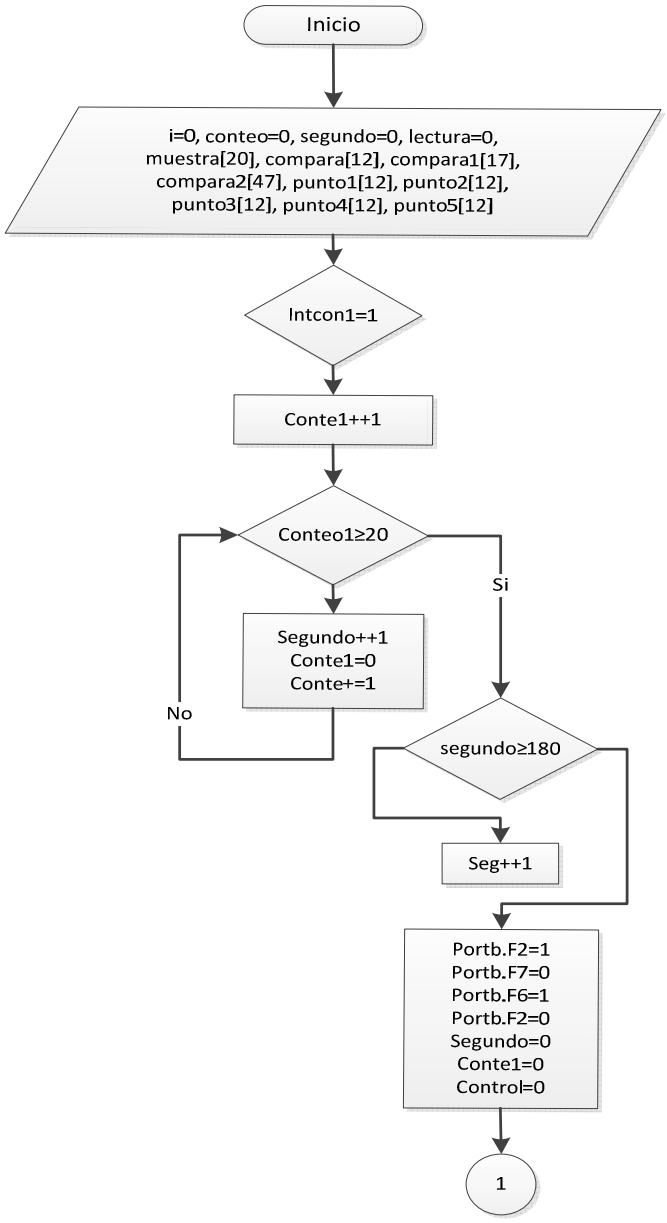


Figura 3.3 Diagramas de flujo Interrupciones y tiempo de espera

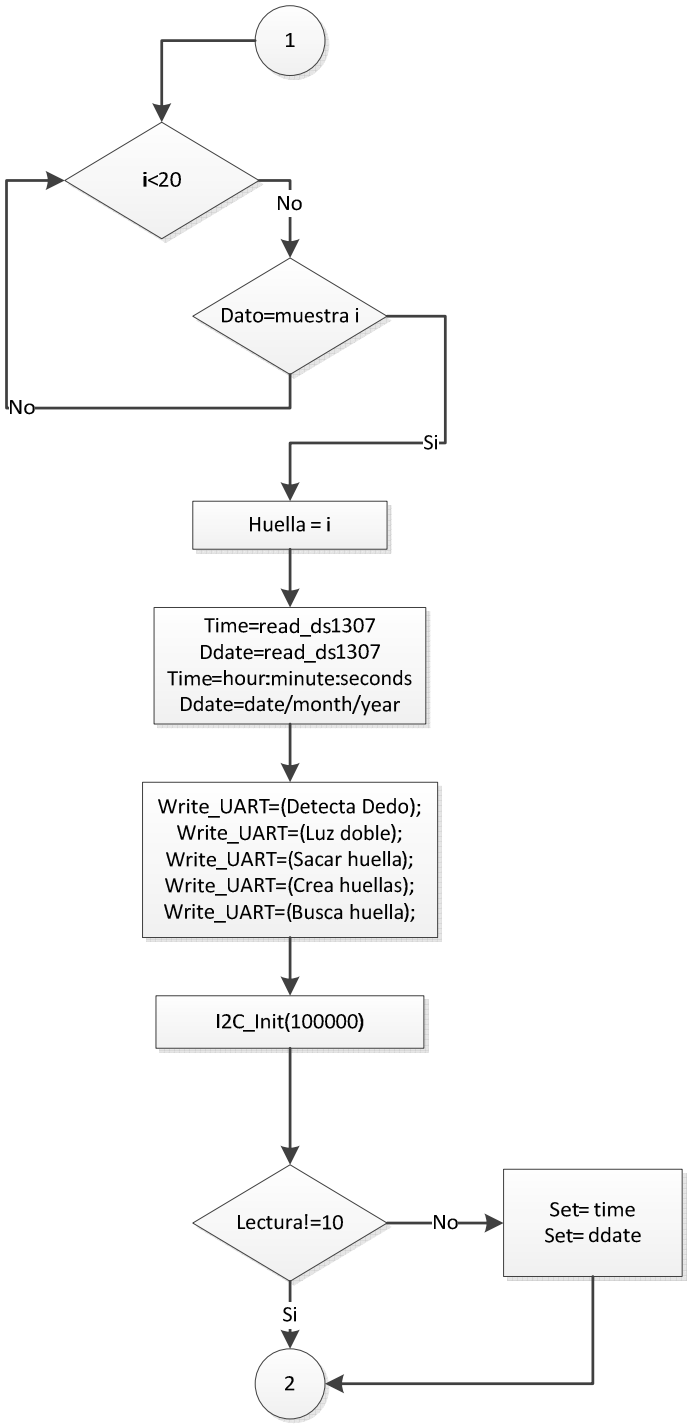


Figura 3.4 Diagrama de flujo Recuperación de Huella y seteo de reloj

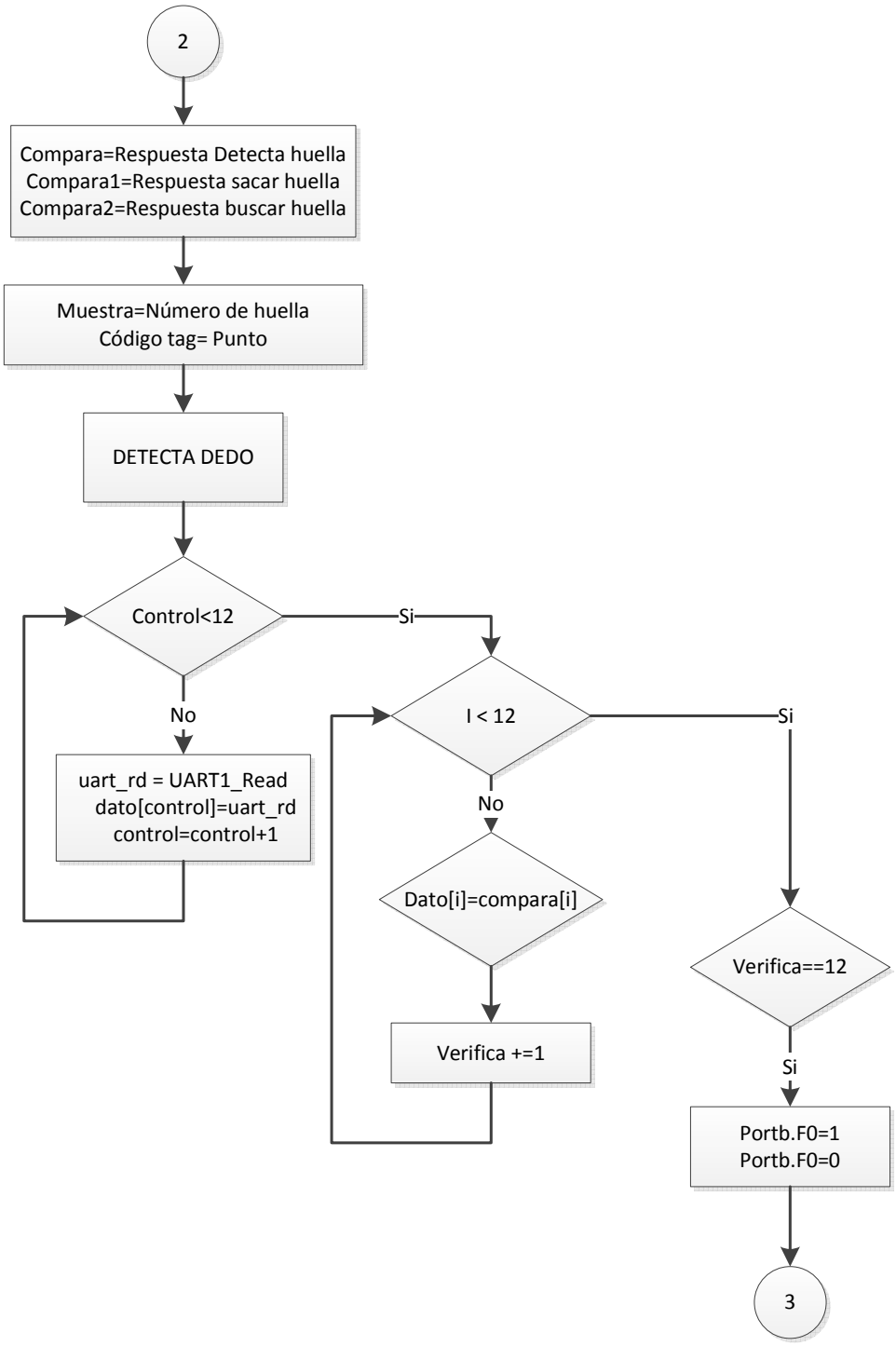


Figura 3.5 Diagrama de flujo Respuesta detecta huellas

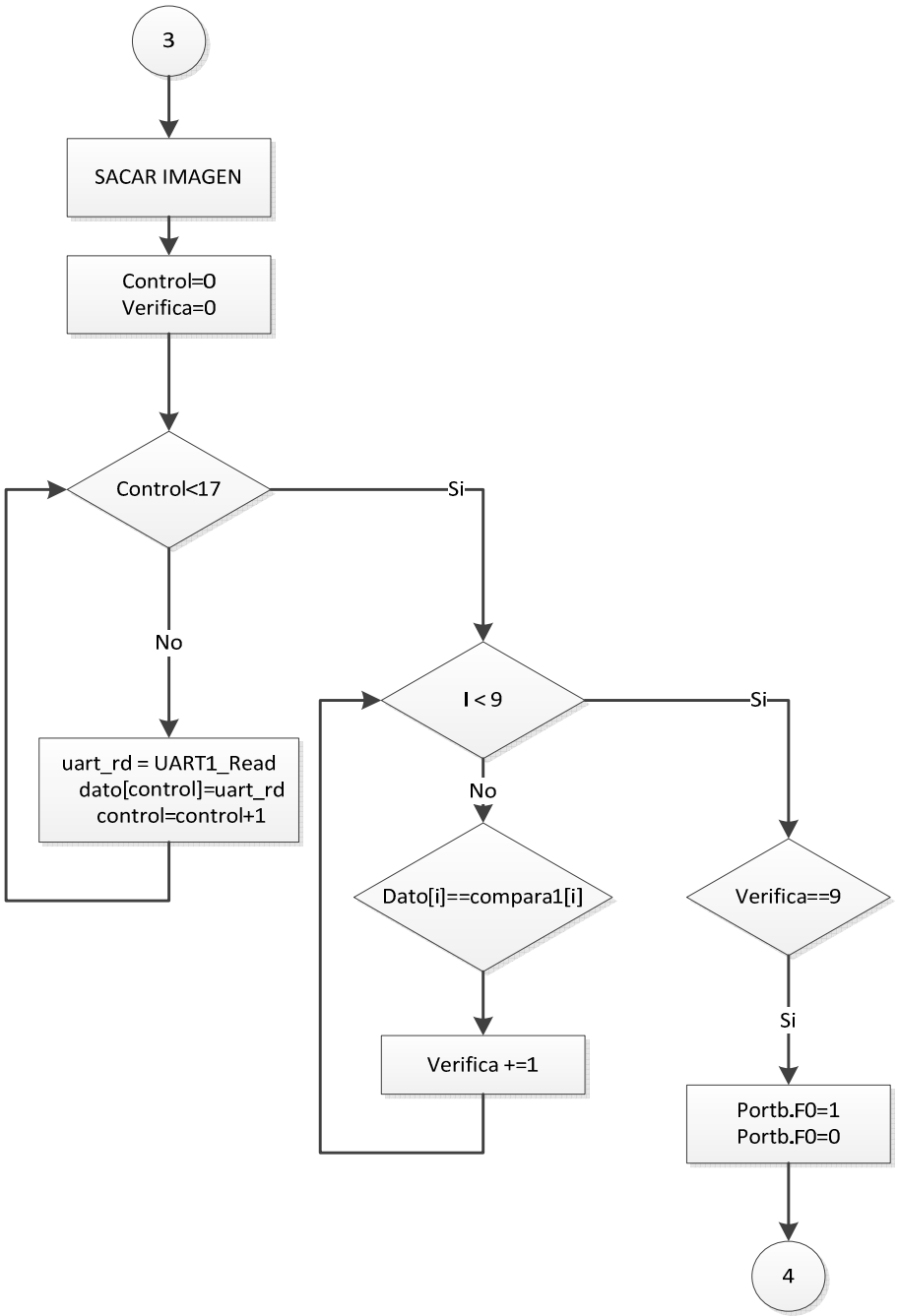


Figura 3.6 Diagrama de flujo Obtener la imagen de la huella

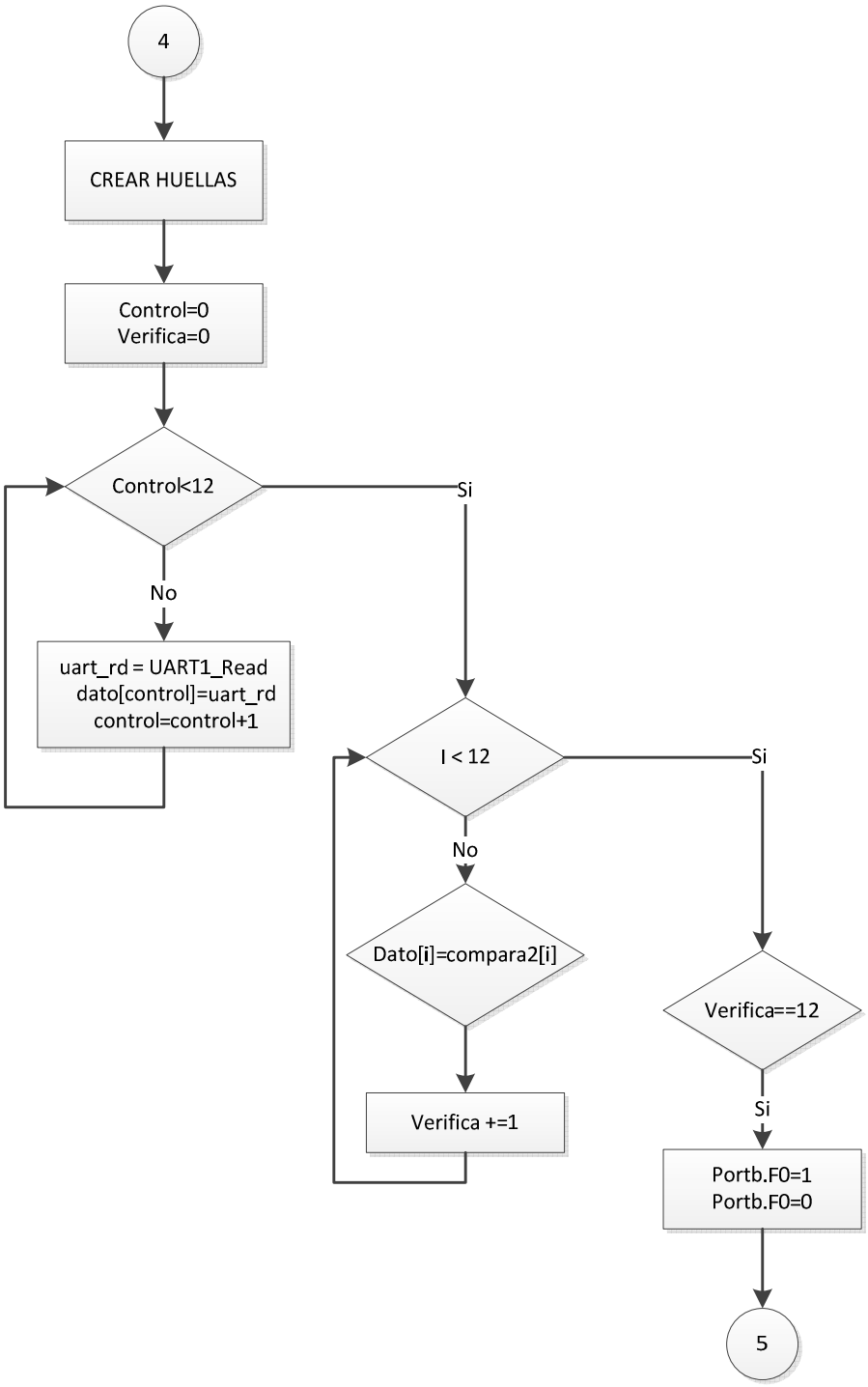


Figura 3.7 Diagrama de flujo Crear Huellas

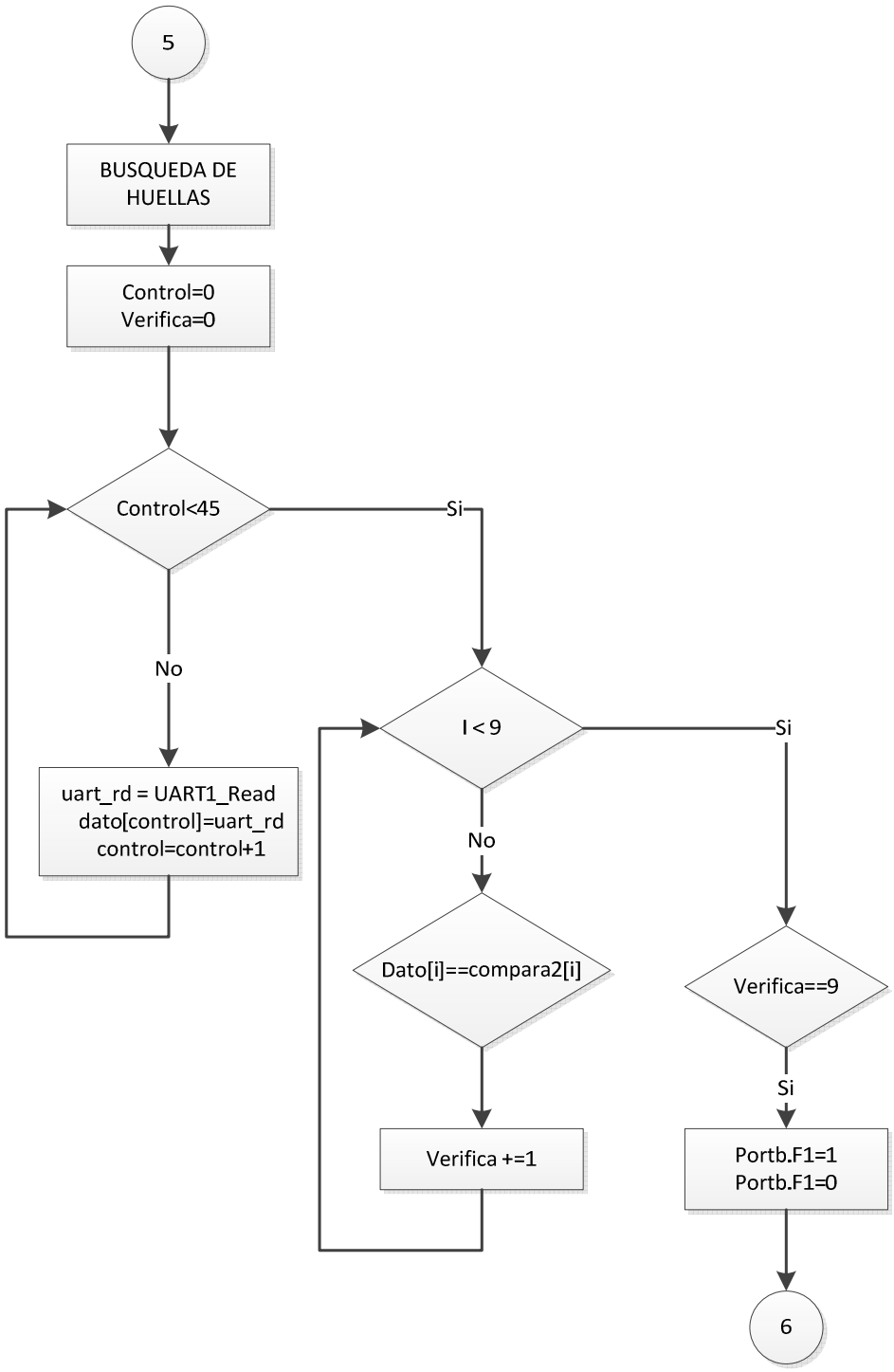


Figura 3.8 Diagrama de flujo Búsqueda de huellas

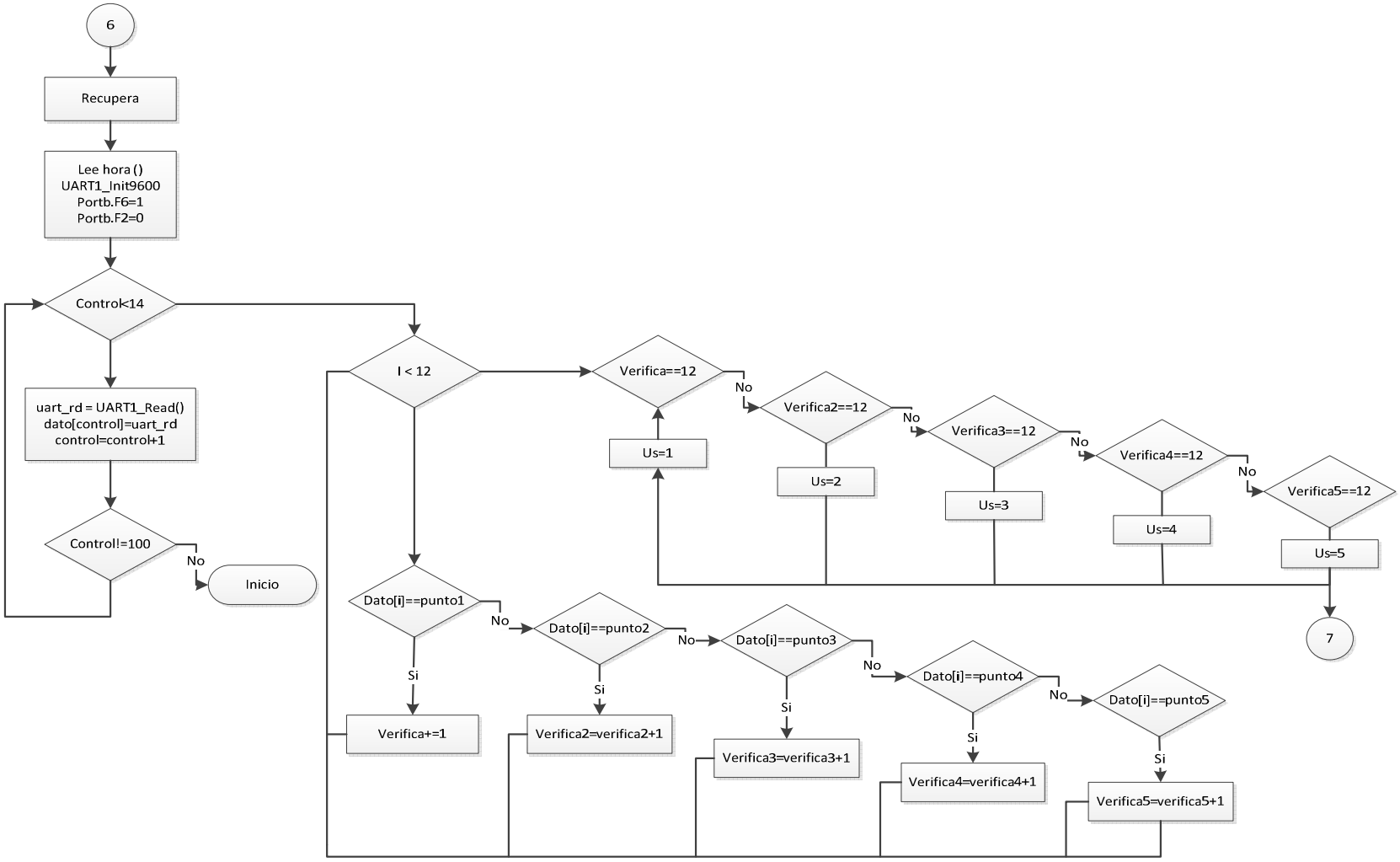


Figura 3.9 Diagrama de flujo Habilitar el lector RFID

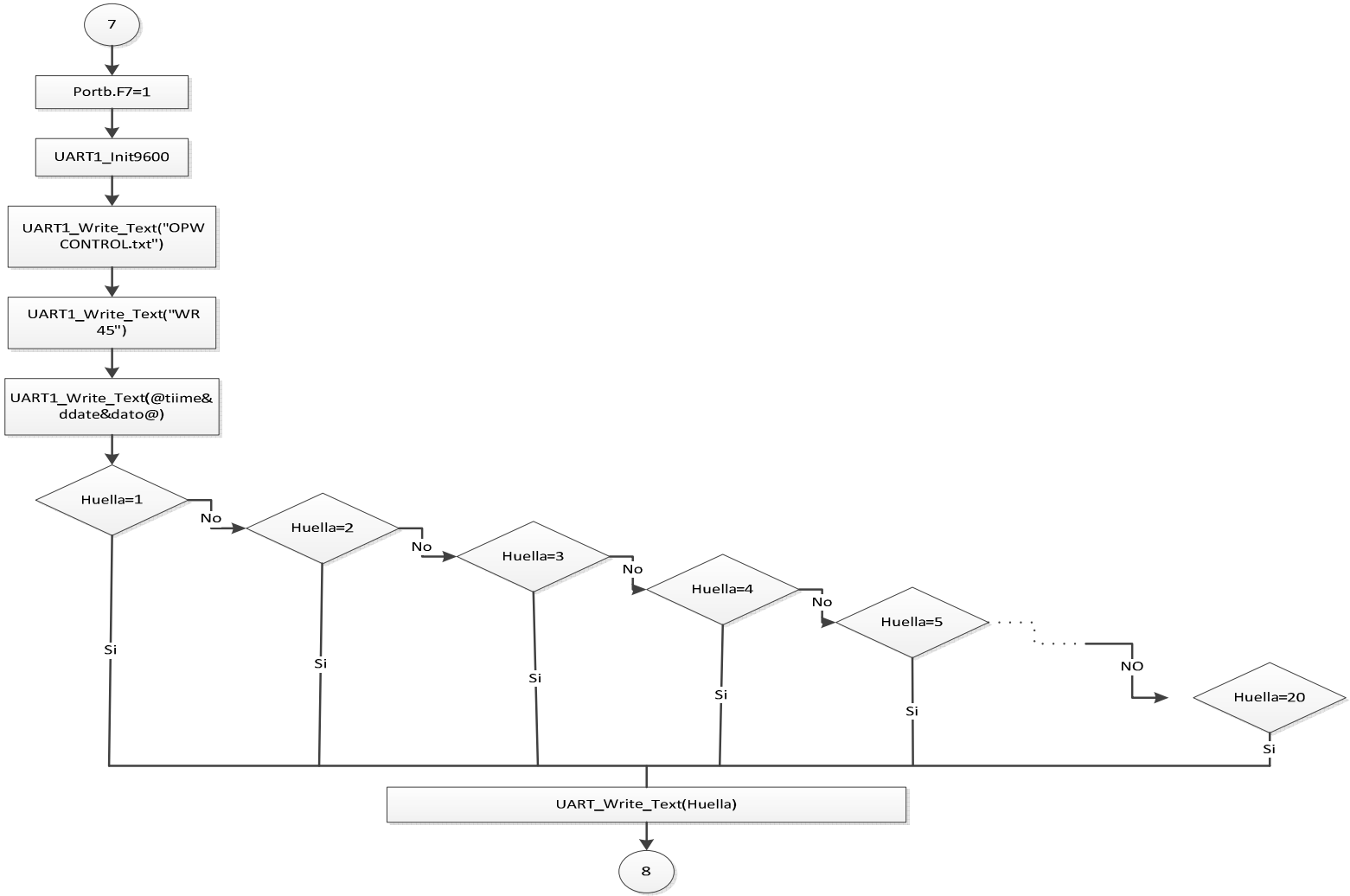


Figura 3.10 Diagrama de flujo Inicialización de comunicación del VDIP1

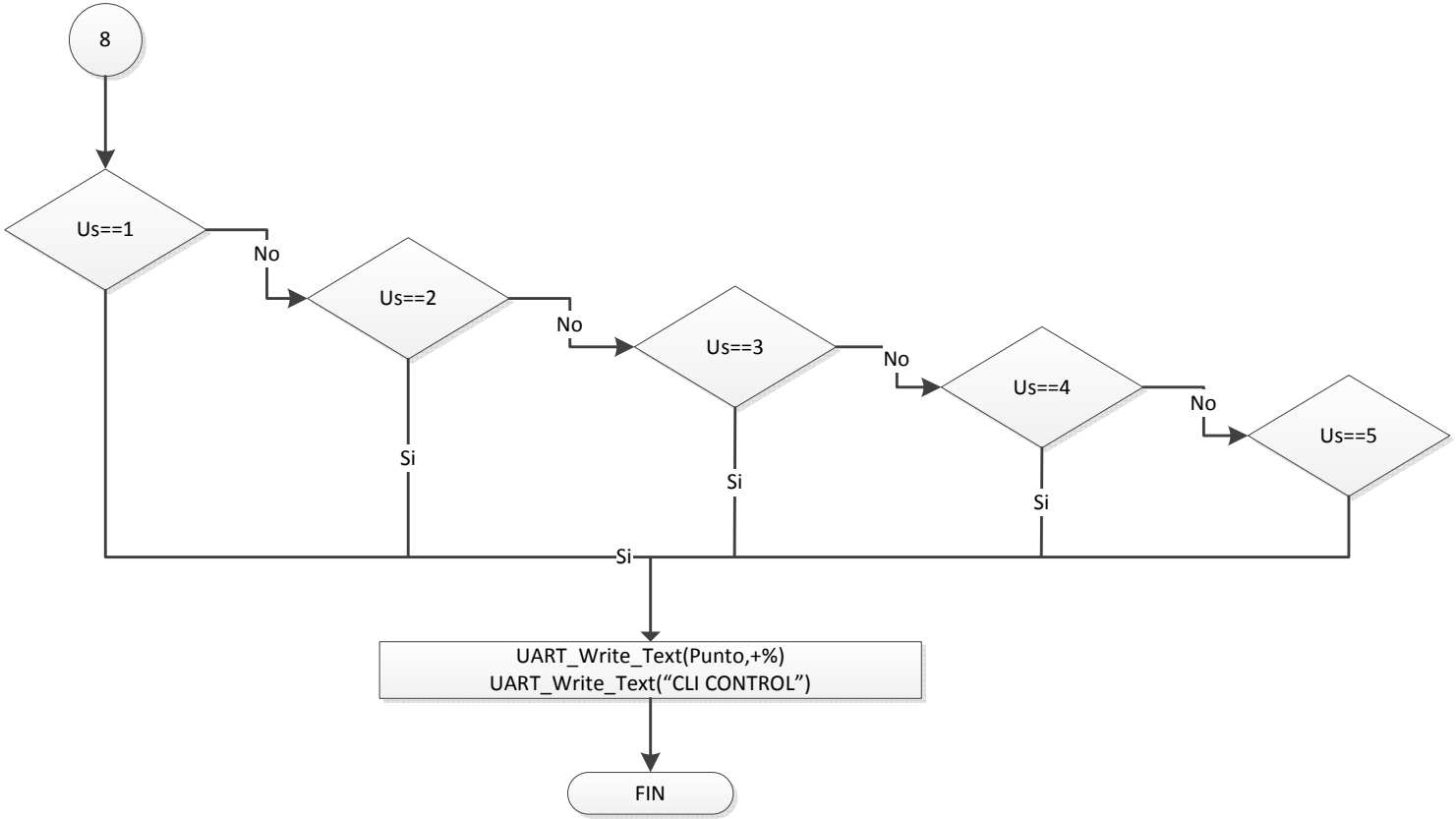


Figura 3.11 Diagrama de flujos

3.1.4.1 Tiempo de Espera

En el diagrama de la figura 3.3 es una interrupción de TIMER 0, es decir es un contador que al terminar su función ingresa a una interrupción, el contador es de 0 a 255, se le da un valor inicial al contador TIMER0L=177 por lo que cuenta solo desde 177 a 255

La fórmula para sacar el tiempo en el cual el programa viene a esta interrupción es:

Tiempo: (1/20000000)*256*(255-177) Siendo Prescaler: 256

Tiempo: 0,0009984 segundos Oscilador externo: 20 MHz

Entonces esta interrupción entra cada 1ms aproximadamente, cada que entra aumentamos 1 en conteo1, cuando llega a 20 tenemos un tiempo de 18ms y la variable segundos se incrementa en 1 hasta 180 teniendo un tiempo de 3240 ms, que es 3.24 segundos.

Las dos variables conteo1 y segundos hacen un total de 3.24 segundos es más o menos el tiempo que espera el programa después de haber pasado el dedo para que lea la tarjeta si pasa ese tiempo y no puso la tarjeta se reiniciara el programa.

3.1.4.2 Recuperación de huella

En diagrama de la figura 3.4 es para recuperar las huellas, existe un contador hasta 20 que es el número de muestras de las huellas que tenemos guardadas en el Lector, comparamos entre el dato de dedo detectado con las muestras, mientras esta comparación no sea verdadera seguirá en el bucle hasta que encuentre coincidencia, en este caso se guardara la huella.

Inicializamos la comunicación I2C con el microprocesador y el DS1307 para leer la hora y fecha en tiempo real, la condición lectura igual a 10 nos

permite igualar el reloj, cuando lectura sea diferente a 10 el programa continuara.

Posteriormente escribimos en el microprocesador los comandos a detalle del lector de huellas los cuales son: Detecta, luz doble, sacar huella, crear huella y búsqueda de huella.

3.1.4.3 Respuesta detecta huellas

En la figura 3.5 asignamos la respuesta del detecta huellas en compara, sacar huellas con compara1 y búsqueda de huellas con compara2, luego asignamos un números de huellas a cada muestra y un código de la tarjeta tag con un número de punto.

Posteriormente iniciamos con la detección del dedo para lo cual leemos el formato del paquete que se encuentra en el microcontrolador mediante una variable denominada control que obtiene los 12 caracteres, cuando se ha leído completamente el paquete anterior realizamos la comparación del paquete respuesta carácter por carácter con el dato, cada vez que realizamos dicha comparación la variable verifica aumenta en 1, cuando se han comparado los 12 caracteres los verificamos, si son correctos los 12 caracteres se encenderá el led amarillo.

3.1.4.4 Obtener la imagen de la huella

En la figura 3.6 sacamos la imagen de las huellas, para el cual seteamos los valores de las variables verifica y control en cero, luego leemos el formato del paquete que se encuentra en el microcontrolador con la variable control que obtiene los 17 caracteres, cuando se ha leído completamente el paquete, realizamos la comparación del paquete respuesta carácter por carácter con el dato, cada vez que realizamos dicha comparación la variable verifica

aumenta en 1, cuando se han comparado los 9 caracteres los verificamos, si son correctos los caracteres se encenderá el led amarillo.

3.1.4.5 Crear Huella

En la figura 3.7 crea huellas, para el cual seteamos los valores de las variables verifica y control en cero, luego leemos el formato del paquete que se encuentra en el microcontrolador con la variable control que obtiene los 12 caracteres, cuando se ha leído completamente el paquete, realizamos la comparación del paquete respuesta carácter por carácter con el dato, cada vez que realizamos dicha comparación la variable verifica aumenta en 1, cuando se han comparado los 12 caracteres los verificamos, si son correctos los caracteres se encenderá el led amarillo.

3.1.4.6 Búsqueda de Huellas

En diagrama de la figura 3.8 buscamos las huellas, seteamos los valores de las variables verifica y control en cero, luego leemos el formato del paquete que se encuentra en el microcontrolador con la variable control que obtiene los 45 caracteres, cuando se ha leído completamente el paquete, realizamos la comparación del paquete respuesta carácter por carácter con el dato, cada vez que realizamos dicha comparación la variable verifica aumenta en 1, cuando se han comparado los 9 caracteres los verificamos, si son correctos se encenderá el led verde.

3.1.4.7 Habilitar el lector RFID

En diagrama de la figura 3.9 habilitamos la velocidad con la que trabaja el RFID y el módulo VDIP, luego leemos el código de cada tarjeta tag que se encuentra en el microcontrolador, con la variable control que obtiene 14

caracteres, el primer carácter es para el inicio de la trama, los 12 siguientes pertenecen al código y el ultimo es para la finalización de la trama, cuando se ha leído completamente los caracteres de la trama, realizamos la comparación entre el dato y los 12 caracteres que pertenecen a un punto determinado, averiguando a qué punto pertenece este código, una vez detectado el punto realizamos la verificación de dicho punto y asignamos su número correspondiente.

3.1.4.8 Inicialización de comunicación del VDIP1

En diagrama de la figura 3.10 inicializamos la comunicación entre el modulo VDIP y el microcontrolador, para almacenar la información abrimos un block de notas denominado control con un limitante de 45 caracteres el cual está conformado con un indicador: @ para inicio de la trama, el tiempo, &, la fecha, &, código de la tarjeta y % para la finalización de la trama, averiguamos el numero de huella asignada a un usuario y el punto designado, se procede a guardar la información y a cerrar el bloc de notas.
La información como podemos observar en el programa se presenta en un bloc de notas y nos presenta todos los parámetros establecidos como son: la hora, la fecha, el código de la tarjeta tag, el número de la huella almacenada y el punto al que corresponde la tarjeta tag.

Si el bloc de notas es eliminado de la memoria flash no es necesario crearlo manualmente ya que en el programa está estipulado que se cree un nuevo bloc de notas denominado CONTROL.txt.

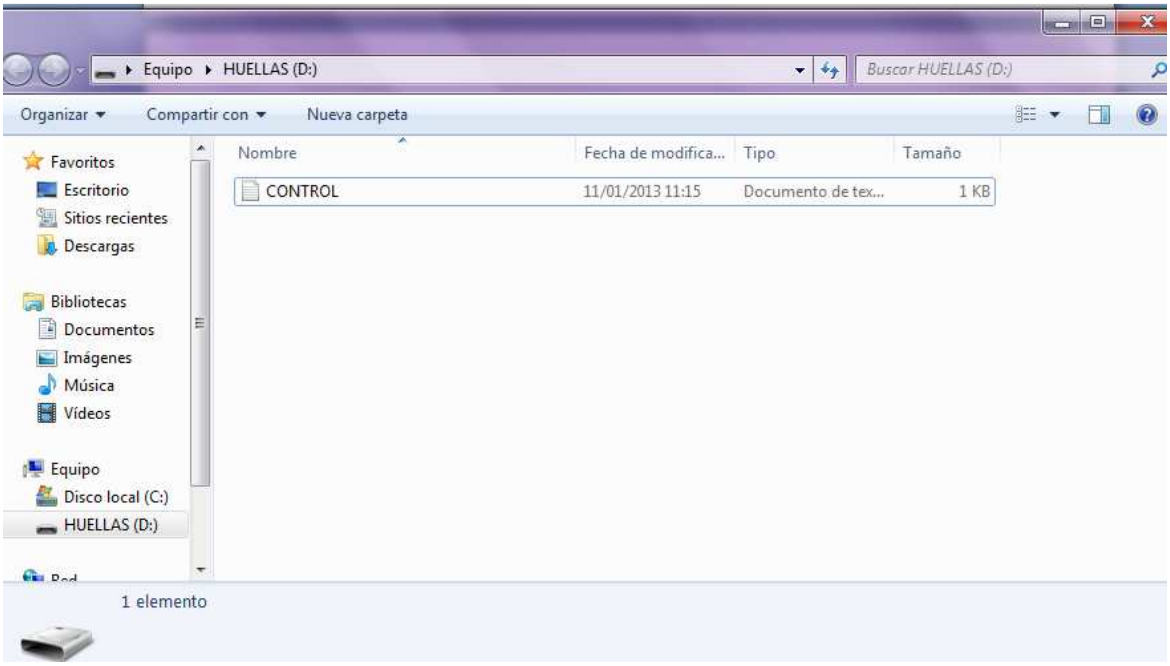


Figura 3.12 Pantalla de la información almacenada en la memoria flash

El documento CONTROL contiene la información representada de la siguiente manera:

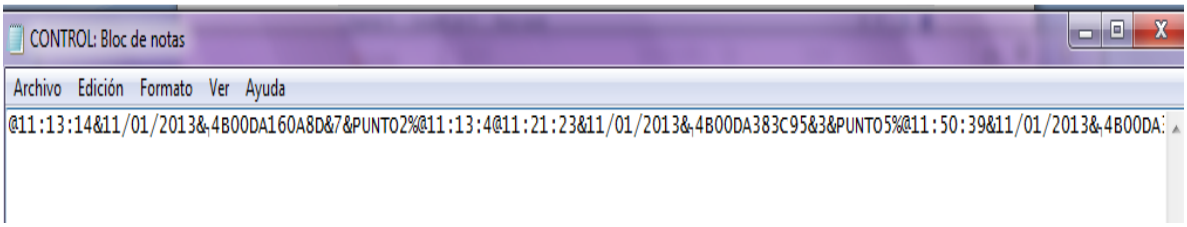


Figura 3.13 Pantalla de presentación de la Información

Cada vez que se realice los procesos correctamente el bastón electrónico almacena las tramas una a continuación de la otra, se identifican las tramas debido al símbolo @ con el que inicia y % con el que finaliza, también se

separa la información que contiene cada trama con símbolos intermedios, es decir, hay un símbolo que separa la hora, la fecha, el código de la tarjeta tag, la huella y el punto correspondiente al tag.

3.2 Software para añadir huellas digitales

Este software fue proporcionado al comprar el lector de huellas dactilares, el cual está realizado en C#, este es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

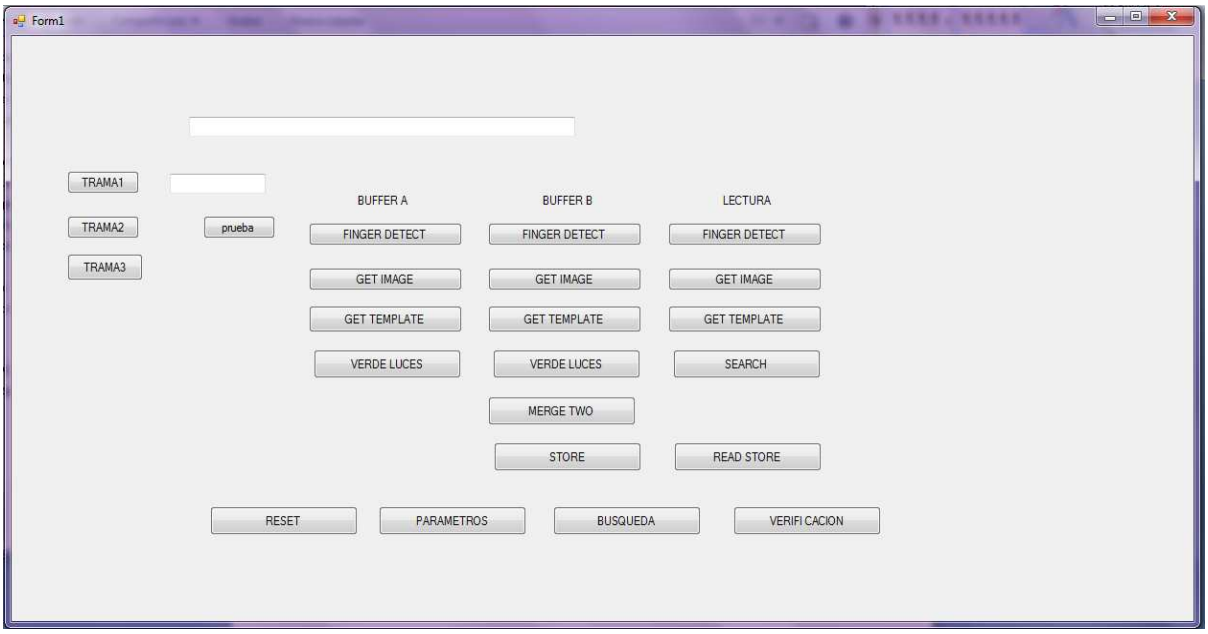


Figura 3.14 Interfaz Gráfica del software para añadir huellas

3.2.1 Diagrama de Flujo

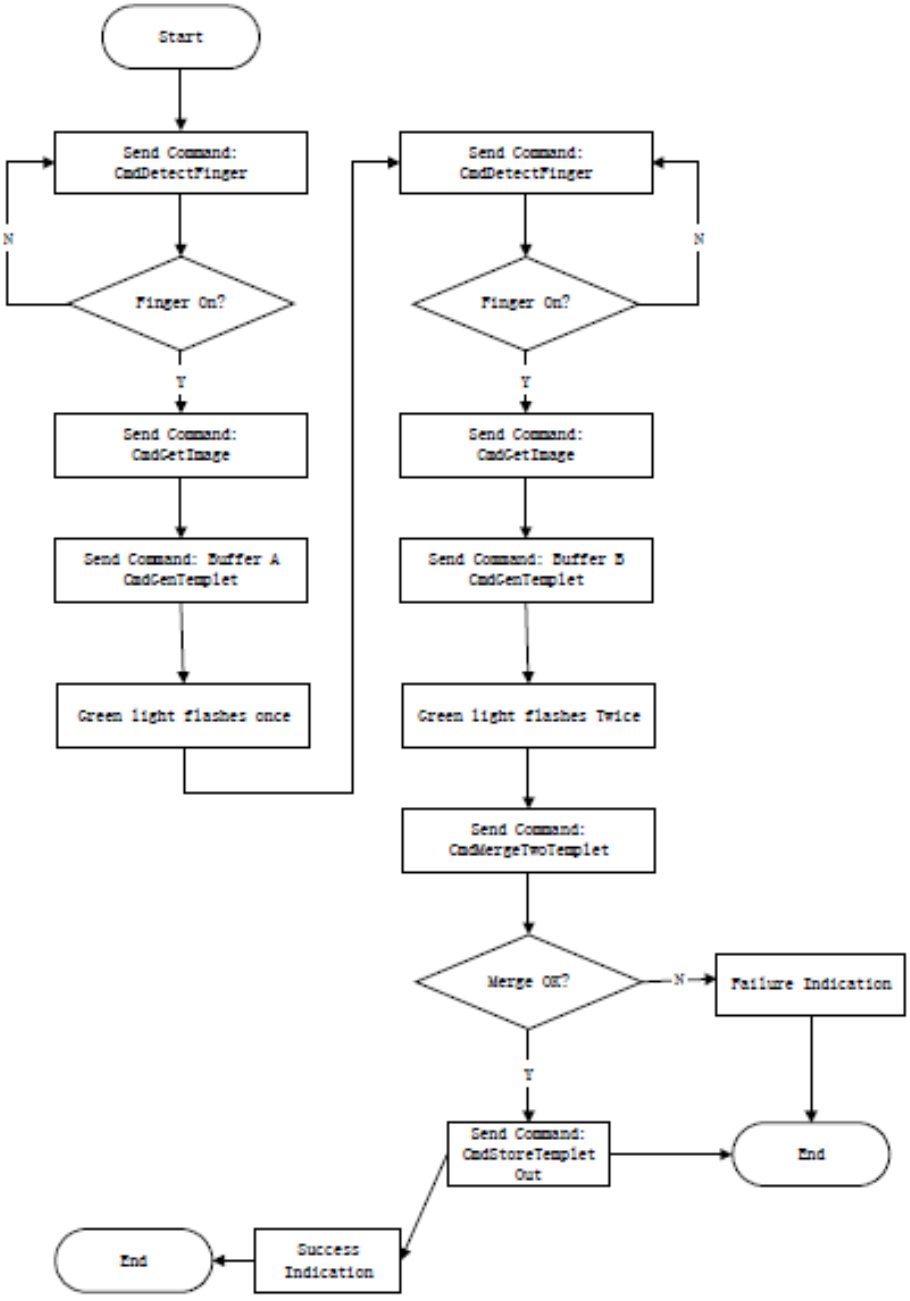


Figura 3.15 Diagrama de Flujo Software para añadir huellas

El funcionamiento del software se basa en los comandos a detalle del lector de huellas, para almacenar nuevas huellas dactilares debemos conectar el modulo SM-621 con el circuito para añadir huellas mediante cable plano, es decir es una comunicación serial.

Para añadir las huellas se debe colocar el dedo sobre el lector y presionar los botones de la columna BUFFER A denominados finger detect, get image, get template, verde luces (Opcional), en orden uno a continuación del otro, una vez que hayamos terminado la columna anteriormente, realizamos el mismo proceso con la columna BUFFER B en esta columna se encuentra el comando a detalle que almacena la huella dactilar, sin retirar el dedo del lector hay que presionar todos los botones de las dos columnas BUFFER A y BUFFER B en orden.

Al momento de almacenar las huellas se asigna un número, este número coloca el administrador en el programa, es decir en el botón store se coloca el número que se le va asignar a la huella dactilar.

De esta manera se almacena nuevas huellas, una vez terminado todo este proceso se debe desconectar el módulo del circuito que añade huellas para que funcione normalmente el bastón.

3.3 Implementación

Este proyecto fue implementado en las instalaciones de Aneta Seguridad S.A. para el respectivo control de las rondas de los guardias de seguridad, la ubicación exacta es en la Calle Berlín entre Av. 10 de Agosto y Av. Eloy Alfaro, en la siguiente imagen se puede observar toda el área a cubrir con nuestro proyecto.

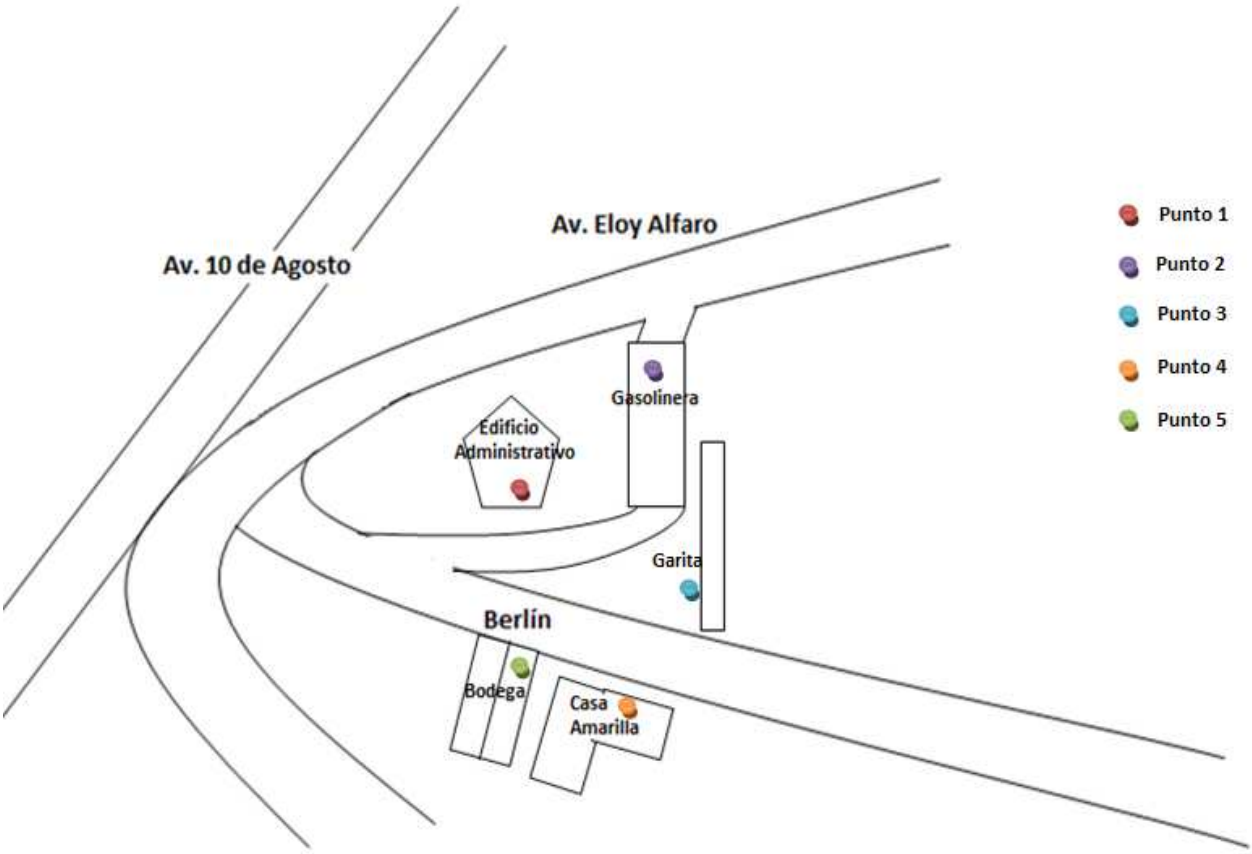


Figura 3.16 Mapa de Aneta Seguridad S.A.

Hemos instalado en 5 partes específicas los puntos es decir las tarjetas tag, a continuación observaremos imágenes en donde han sido instalados cada uno de estos puntos.

Punto 1=> Ubicado en el Edificio Administrativo

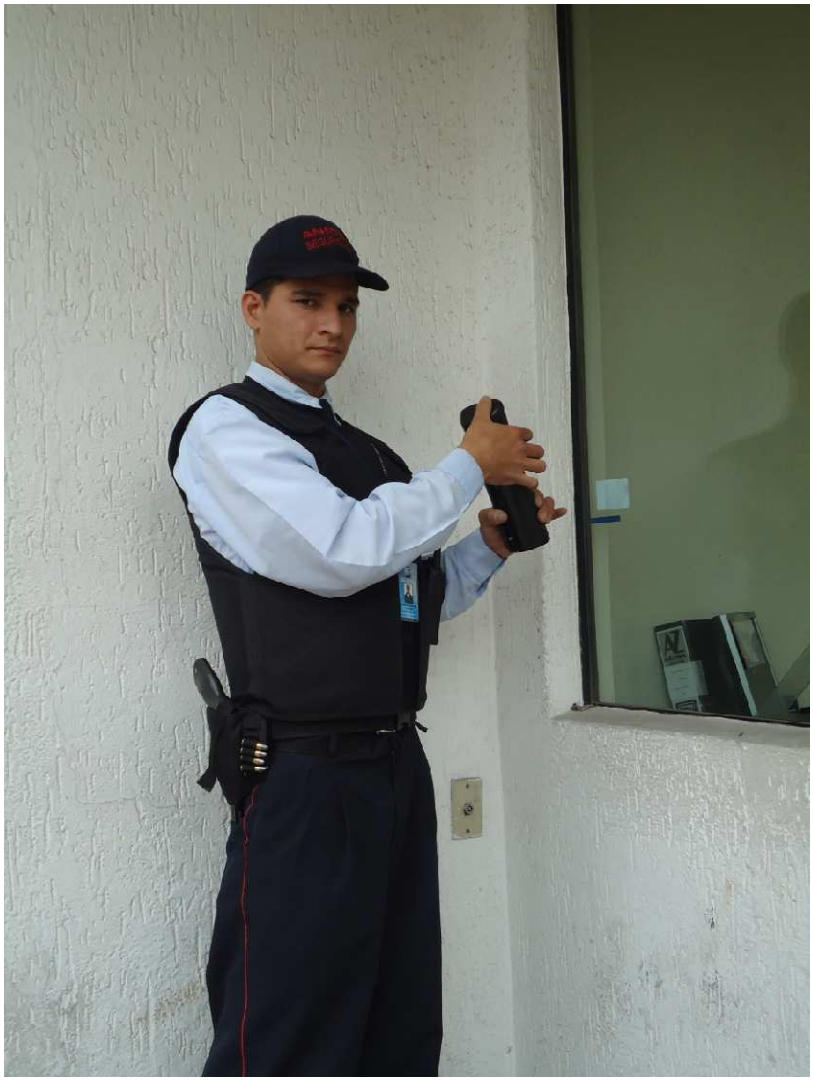


Foto 3-1

Foto 3-2

Punto 2=> Ubicado en la Gasolinera

Foto 3-3 Realizando la implementación en la gasolinera

Foto 3-4 Vista ampliada del punto de la gasolinera

Punto 3=> Ubicado en la Garita

Foto 3-5 Realizando la Implementación

Foto 3-6 Resultado final de la implementación

Foto 3-7 Registrando la huella

Foto 3-8 Registrando por el punto 3.

Punto 4=> Ubicado en la casa amarilla

Foto 3-9 Realizando la implementación

Foto 3-10 Resultado de la implementación

Punto 5=> Ubicado en la bodega

Foto 3-11 Realizando la implementacion

Foto 3-12 Resultado final de la implementación

Foto 3-13 Vista exterior de la bodega

Foto 3-14 Gerente de Aneta Seguridad con el bastón electrónico

Foto 3-15 Supervisor de los guardias con el bastón electrónico

CAPÍTULO 4

4 Conclusiones y Recomendaciones

4.1 Conclusiones

Se implementó un prototipo de bastón electrónico para el control de los guardias de seguridad para que cumplan con su correcto trabajo es decir que efectúen las rondas al tiempo establecido.

Se han realizado las pruebas necesarias de funcionamiento del sistema llegando a cumplir exitosamente nuestras expectativas para el proyecto.

Se ha realizado estudios a base de la tecnología RFID la cual hemos utilizado RFID pasivos que no necesitan de una fuente y tienen un corto alcance, también hemos analizado el lector de huellas en este caso el SM-621 (Módulo de Verificación de huellas dactilares), el cual se adaptado correctamente al bastón para la debida verificación de la persona quien lo utilice, no hay manera de alterar las huellas dactilares.

A pesar de la diferentes velocidades con las que funcionan el RFID (9600 baudios) y el modulo SM-621 (56700 baudios) se ha logrado un correcto funcionamiento es decir una comunicación exitosa entre todos los dispositivos del bastón.

El programa del PIC 18F452 está diseñado para transmitir y enviar información al modulo SM-61, también se lo diseño para recibir datos de lectores RFID que envíen 12 dígitos siendo esto una gran ventaja ya que es compatible con la mayoría de lectores RFID en el mercado que cumplen con esta característica y también fue programado para enviar la información a la memoria flash por medio del módulo DVIP1.

La interfaz amigable para el administrador es por medio de un bloc de notas en el cual se observa de manera clara todos los parámetros establecidos en el programa del PIC (hora, fecha, código de la tarjeta tag, número de huella y el punto al que corresponde el código de la tarjeta tag).

El software para añadir huellas fue proporcionado con el lector de huellas, este es de gran ayuda al momento de almacenar una nueva huella en el módulo SM-621.

4.2 Recomendaciones

Se puede recomendar que para usos avanzados del presente proyecto, se realice una base de datos para mantener de manera ordenada toda la información descargada del bastón electrónico.

Se recomienda que en un futuro para la explotación del presente sistema, se desarrolle el mismo proyecto en microcircuito electrónico para que su tamaño se reduzca de manera que sea fácil manipularlo.

Se recomienda que en un futuro para la explotación y desarrollo del presente proyecto, la ergonomía es decir el estuche sea de mejor calidad y material para que pueda soportar lugares húmedos, cálidos, etc.

4.3 Bibliografía

(s.f.). Recuperado el 11 de Diciembre de 2012, de http://es.wikipedia.org/wiki/RFID#Antecedentes

(s.f.). Recuperado el 11 de Diciembre de 2012, de http://es.wikipedia.org/wiki/Radiofrecuencia

(Enero de 2007).

Accesor Solutions. (s.f.). Recuperado el 10 de 12 de 2012, de http://www.accesor.com/esp/art2_query.php?fam=3&sfam=4

Alegsa. (s.f.). Obtenido de http://www.alegsa.com.ar/Dic/comunicacion%20paralela.php

Castaño, Á. R. (s.f.). *Universidad Sevilla.* Recuperado el Diciembre de 2012, de http://alojamientos.us.es/afcomput/docs/material/Tema7.pdf

Control-accesos. (25 de 11 de 2012). Recuperado el 10 de 12 de 2012, de http://control-accesos.es/category/lectores/lectores-biometricos

Costales, I. A. (2012). *Microcontroladores.* Quito.

datasheet ds1307. (s.f.). Recuperado el 5 de Enero de 2013, de http://www.hispavila.com/3ds/atmega/descargas/rtc_ds1307.pdf

Datasheet lector RFID. (s.f.). Recuperado el 19 de Febrero de 2013, de http://www.sparkfun.com/datasheets/Sensors/ID-12-Datasheet.pdf

Diotec Semiconductor. (s.f.). Recuperado el 5 de Enero de 2013, de http://www.electronicaembajadores.com/datos/pdf1/sm/smdi/1n4001.pdf

ecnologiajavier. (s.f.). Recuperado el 5 de Enero de 2013, de http://www.tecnologiajavier.es/4eso/t01analogica/activ01/01ficha26.pdf

i-micro. (s.f.). Recuperado el 13 de Diciembre de 2012, de http://www.i-micro.com/pdf/articulos/rs-232.pdf

jprogr. (s.f.). Recuperado el 5 de Enero de 2013, de http://www.jprogr.com/2012/05/como-usar-un-regulador-de-voltaje-7805.html

mandobots. (s.f.). Recuperado el 5 de Enero de 2013, de http://mandobots.jimdo.com/tips-y-links/como-funciona-un-rele-de-5-pines/

Miaxis Biometrics Co., L. (24 de Enero de 2007). SM-621 Fingerprint.

MICROCONTROLADOR. (s.f.). Recuperado el 2 de Enero de 2013, de http://www.pinguino.org.ve/~pinguino/descargas/Manual%20PIC%2018F4550.pdf

MORÁN, D. A. (2011). *Tesis ESPE.* Recuperado el Noviembre de 2012

scribd. (s.f.). Recuperado el 5 de Enero de 2013, de http://es.scribd.com/doc/73251181/Buzzer-Datasheet

slideshare. (s.f.). Recuperado el 5 de Enero de 2013, de http://www.slideshare.net/JonathanRuizdeGaribay/06temporizadores-9769468

wikipedia. (s.f.). Recuperado el 5 de Enero de 2013, de http://es.wikipedia.org/wiki/2N3904

Wikipedia. (s.f.). Recuperado el 10 de 12 de 2012, de http://translate.google.com.ec/translate?hl=es&langpair=en|es&u=http://en.wikipedia.org/wiki/Fingerprint_recognition&ei=vFTGUPWjFomg9QS9lICYDg

Wikipedia. (20 de Octubre de 2011). Obtenido de http://es.wikipedia.org/wiki/Reloj_en_tiempo_real

Wikipedia. (17 de Octubre de 2012). Recuperado el Diciembre de 2012, de http://es.wikipedia.org/wiki/Comunicaci%C3%B3n

Wikipedia. (11 de Febrero de 2013). Recuperado el 19 de Febrero de 2013, de http://es.wikipedia.org/wiki/Regulador_de_tensi%C3%B3n

Windows 2 universe. (s.f.). Recuperado el 11 de Diciembre de 2012, de http://www.windows2universe.org/physical_science/magnetism/em_radio_waves.html&lang=sp

yahoo. (s.f.). Recuperado el 5 de Enero de 2013, de http://es.answers.yahoo.com/question/index?qid=20080910155729AAfrFh7

4.4 Anexos

Anexos 1 Data Sheet del Pic 18F452

PIC18FXX2
Data Sheet

High-Performance, Enhanced Flash
Microcontrollers with 10-Bit A/D

MICROCHIP

PIC18FXX2

28/40-pin High Performance, Enhanced FLASH Microcontrollers with 10-Bit A/D

High Performance RISC CPU:

- C compiler optimized architecture/instruction set
  - Source code compatible with the PIC16 and PIC17 instruction sets
- Linear program memory addressing to 32 Kbytes
- Linear data memory addressing to 1.5 Kbytes

Device	On-Chip Program Memory		On-Chip RAM (bytes)	Data EEPROM (bytes)
	FLASH (bytes)	# Single Word Instructions		
PIC18F242	16K	8192	768	256
PIC18F252	32K	16384	1536	256
PIC18F442	16K	8192	768	256
PIC18F452	32K	16384	1536	256

- Up to 10 MIPs operation:
  - DC - 40 MHz osc./clock input
  - 4 MHz - 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two Capture/Compare/PWM (CCP) modules. CCP pins that can be configured as:
  - Capture input: capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1- to 10-bit, max. PWM freq. @: 8-bit resolution = 156 kHz, 10-bit resolution = 39 kHz
- Master Synchronous Serial Port (MSSP) module, Two modes of operation:
  - 3-wire SPI™ (supports all 4 SPI modes)
  - I²C™ Master and Slave mode

Peripheral Features (Continued):

- Addressable USART module:
  - Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module

Analog Features:

- Compatible 10-bit Analog-to-Digital Converter module (A/D) with:
  - Fast sampling rate
  - Conversion available during SLEEP
  - Linearity ≤ 1 LSb
- Programmable Low Voltage Detection (PLVD)
  - Supports interrupt on-Low Voltage Detection
- Programmable Brown-out Reset (BOR)

Special Microcontroller Features:

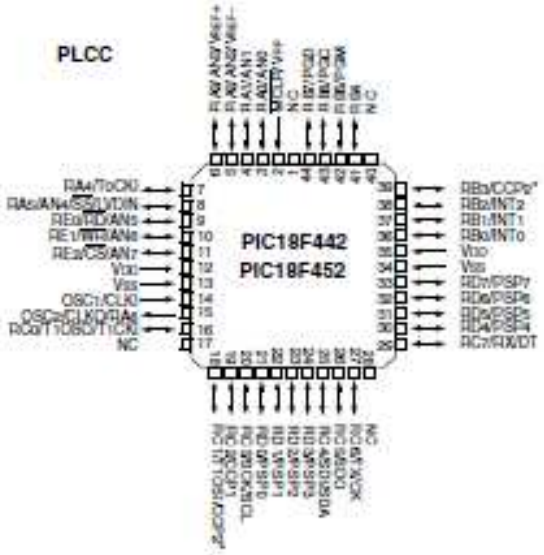
- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory
- FLASH/Data EEPROM Retention: > 40 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options including:
  - 4X Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input
- Single supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

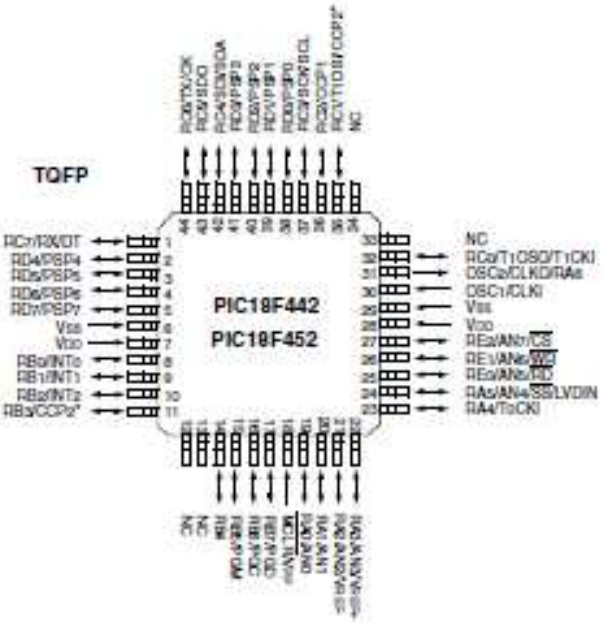
- Low power, high speed FLASH/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges
- Low power consumption:
  - < 1.6 mA typical @ 5V, 4 MHz
  - 25 µA typical @ 3V, 32 kHz
  - < 0.2 µA typical standby current

Pin Diagrams

PLCC



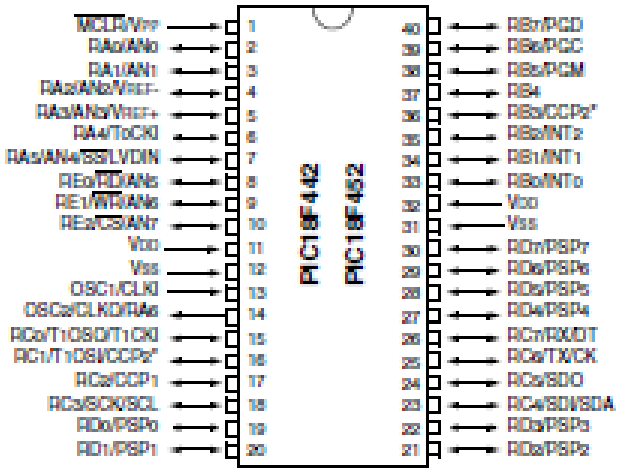
TQFP



* RB3 is the alternate pin for the CCP2 pin multiplexing.

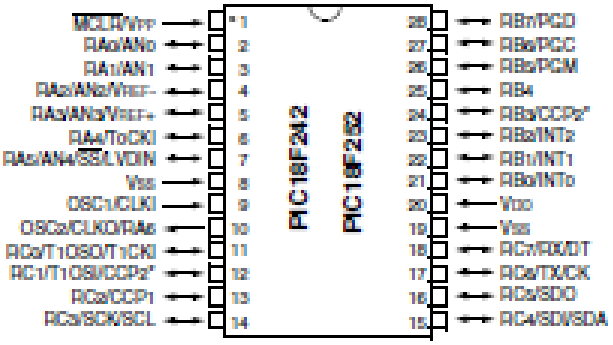
Pin Diagrams (Cont.'d)

DIP



Note: Pin compatible with 40-pin PIC16C7X devices.

DIP, SOIC



* RB3 is the alternate pin for the CCP2 pin multiplexing.

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F242
- PIC18F442
- PIC18F252
- PIC18F452

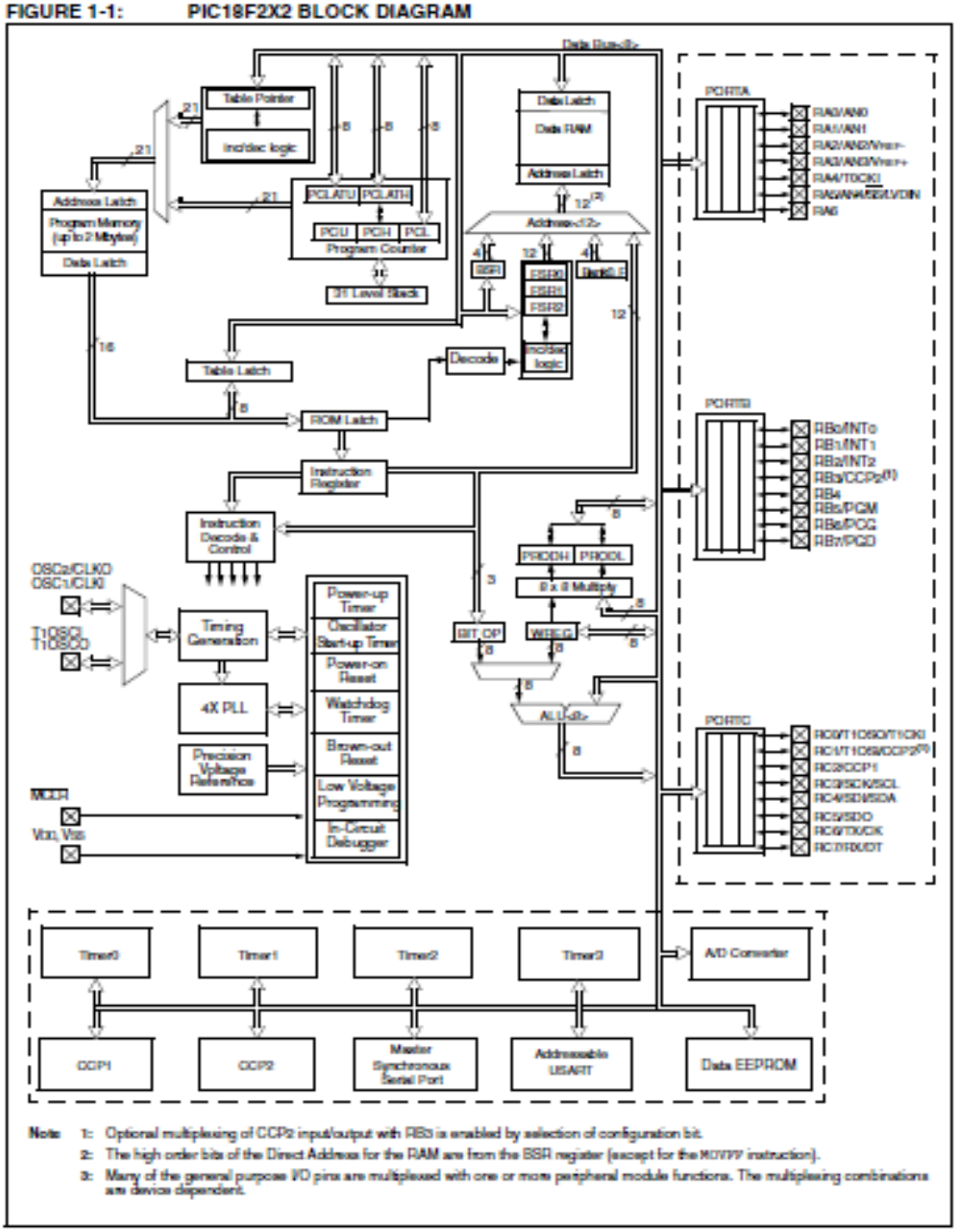
These devices come in 28-pin and 40/44-pin packages. The 28-pin devices do not have a Parallel Slave Port (PSP) implemented and the number of Analog-to-Digital (A/D) converter input channels is reduced to 5. An overview of features is shown in Table 1-1.

The following two figures are device block diagrams sorted by pin count: 28-pin for Figure 1-1 and 40/44-pin for Figure 1-2. The 28-pin and 40/44-pin pinouts are listed in Table 1-2 and Table 1-3, respectively.

TABLE 1-1: DEVICE FEATURES

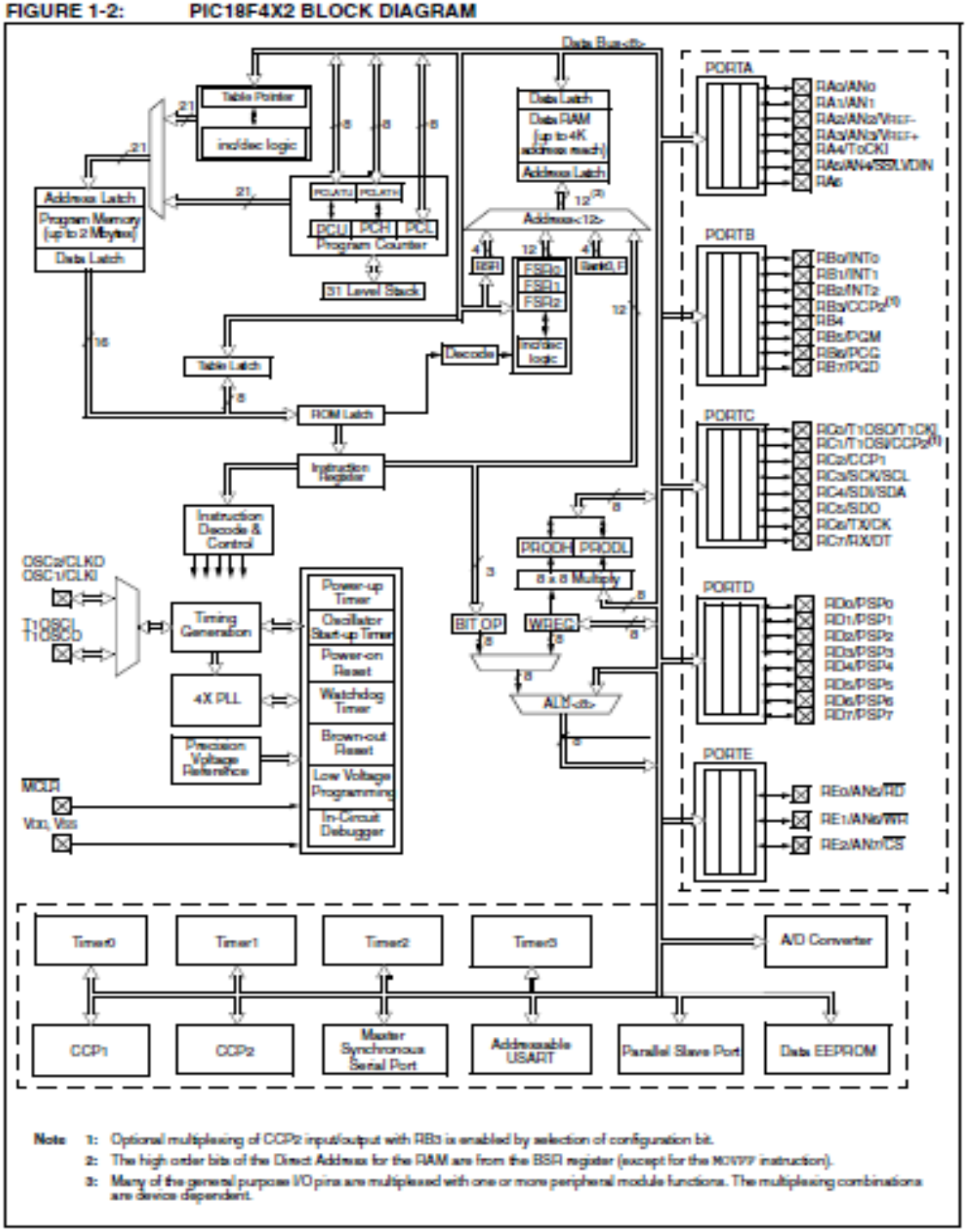
Features	PIC18F242	PIC18F252	PIC18F442	PIC18F452
Operating Frequency	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz
Program Memory (Bytes)	16K	32K	16K	32K
Program Memory (Instructions)	8192	16384	8192	16384
Data Memory (Bytes)	768	1536	768	1536
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	17	17	18	18
I/O Ports	Ports A, B, C	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, Addressable USART	MSSP, Addressable USART	MSSP, Addressable USART	MSSP, Addressable USART
Parallel Communications	—	—	PSP	PSP
10-bit Analog-to-Digital Module	5 input channels	5 input channels	8 input channels	8 input channels
RESETS (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions	75 Instructions	75 Instructions	75 Instructions
Packages	28-pin DIP 28-pin SOIC	28-pin DIP 28-pin SOIC	40-pin DIP 44-pin PLCC 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin TQFP

FIGURE 1-1: PIC18F2X2 BLOCK DIAGRAM



Note 1: Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.
2: The high order bits of the Direct Address for the RAM are from the BSR register (except for the MOVFF instruction).
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

FIGURE 1-2: PIC18F4X2 BLOCK DIAGRAM



Note 1: Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.
2: The high order bits of the Direct Address for the RAM are from the BSR register (except for the MOVFF instruction).
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

TABLE 1-2: PIC18F2X2 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	DIP	SOIC			
MCLR/VPP	1	1			Master Clear (input) or high voltage ICSP programming enable pin.
MCLR			I	ST	Master Clear (Reset) input. This pin is an active low RESET to the device.
VPP			I	ST	High voltage ICSP programming enable pin.
NC	—	—	—	—	These pins should be left unconnected.
OSC1/CLKI	9	9			Oscillator crystal or external clock input.
OSC1			I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise.
CLKI			I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
OSC2/CLKO/RA6	10	10			Oscillator crystal or clock output.
OSC2			O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
RA6			I/O	TTL	General Purpose I/O pin.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2			
RA0			I/O	TTL	Digital I/O.
AN0			I	Analog	Analog input 0.
RA1/AN1	3	3			
RA1			I/O	TTL	Digital I/O.
AN1			I	Analog	Analog input 1.
RA2/AN2/VREF-	4	4			
RA2			I/O	TTL	Digital I/O.
AN2			I	Analog	Analog input 2.
VREF-			I	Analog	A/D Reference Voltage (Low) input.
RA3/AN3/VREF+	5	5			
RA3			I/O	TTL	Digital I/O.
AN3			I	Analog	Analog input 3.
VREF+			I	Analog	A/D Reference Voltage (High) input.
RA4/T0CKI	6	6			
RA4			I/O	ST/OD	Digital I/O. Open drain when configured as output.
T0CKI			I	ST	Timer0 external clock input.
RA5/AN4/SS/LVDIN	7	7			
RA5			I/O	TTL	Digital I/O.
AN4			I	Analog	Analog input 4.
SS			I	ST	SPI Slave Select input.
LVDIN			I	Analog	Low Voltage Detect input.
RA6					See the OSC2/CLKO/RA6 pin.

Legend: TTL – TTL compatible input
CMOS – CMOS compatible input or output
ST – Schmitt Trigger input with CMOS levels
I – Input
O – Output
P – Power
OD – Open Drain (no P diode to VDD)

TABLE 1-2: PIC18F2X2 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	DIP	SOIC			
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0	21	21			
RB0			I/O	TTL	Digital I/O.
INT0			I	ST	External interrupt 0.
RB1/INT1	22	22			
RB1			I/O	TTL	
INT1			I	ST	External interrupt 1.
RB2/INT2	23	23			
RB2			I/O	TTL	Digital I/O.
INT2			I	ST	External interrupt 2.
RB3/CCP2	24	24			
RB3			I/O	TTL	Digital I/O.
CCP2			I/O	ST	Capture2 input, Compare2 output, PWM2 output.
RB4	25	25	I/O	TTL	Digital I/O. Interrupt-on-change pin.
RB5/PGM	26	26			
RB5			I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGM			I/O	ST	Low Voltage ICSP programming enable pin.
RB6/PGC	27	27			
RB6			I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGC			I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/PGD	28	28			
RB7			I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGD			I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

TABLE 1-2: PIC18F2X2 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	DIP	SOIC			
					PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	11			
RC0			I/O	ST	Digital I/O.
T1OSO			O	—	Timer1 oscillator output.
T1CKI			I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2	12	12			
RC1			I/O	ST	Digital I/O.
T1OSI			I	CMOS	Timer1 oscillator input.
CCP2			I/O	ST	Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	13	13			
RC2			I/O	ST	Digital I/O.
CCP1			I/O	ST	Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	14	14			
RC3			I/O	ST	Digital I/O.
SCK			I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL			I/O	ST	Synchronous serial clock input/output for I^2C mode
RC4/SDI/SDA	15	15			
RC4			I/O	ST	Digital I/O.
SDI			I	ST	SPI Data In.
SDA			I/O	ST	I^2C Data I/O.
RC5/SDO	16	16			
RC5			I/O	ST	Digital I/O.
SDO			O	—	SPI Data Out.
RC6/TX/CK	17	17			
RC6			I/O	ST	Digital I/O.
TX			O	—	USART Asynchronous Transmit.
CK			I/O	ST	USART Synchronous Clock (see related RX/DT).
RC7/RX/DT	18	18			
RC7			I/O	ST	Digital I/O.
RX			I	ST	USART Asynchronous Receive.
DT			I/O	ST	USART Synchronous Data (see related TX/CK).
VSS	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	20	P	—	Positive supply for logic and I/O pins.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

TABLE 1-3: PIC18F4X2 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	PLCC	TQFP			
MCLR/VPP	1	2	18			Master Clear (input) or high voltage ICSP programming enable pin.
MCLR				I	ST	Master Clear (Reset) input. This pin is an active low RESET to the device.
VPP				I	ST	High voltage ICSP programming enable pin.
NC	—			—	—	These pins should be left unconnected.
OSC1/CLKI	13	14	30			Oscillator crystal or external clock input.
OSC1				I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise.
CLKI				I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
OSC2/CLKO/RA6	14	15	31			Oscillator crystal or clock output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General Purpose I/O pin.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	3	4	20			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1.
RA2/AN2/VREF-	4	5	21			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
VREF-				I	Analog	A/D Reference Voltage (Low) input.
RA3/AN3/VREF+	5	6	22			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
VREF+				I	Analog	A/D Reference Voltage (High) input.
RA4/T0CKI	6	7	23			
RA4				I/O	ST/OD	Digital I/O. Open drain when configured as output.
T0CKI				I	ST	Timer0 external clock input.
RA5/AN4/SS/LVDIN	7	8	24			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
SS				I	ST	SPI Slave Select input.
LVDIN				I	Analog	Low Voltage Detect input.
RA6						(See the OSC2/CLKO/RA6 pin.)

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

TABLE 1-3: PIC18F4X2 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	PLCC	TQFP			
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0	33	36	8			
RB0				I/O	TTL	Digital I/O.
INT0				I	ST	External Interrupt 0.
RB1/INT1	34	37	9			
RB1				I/O	TTL	
INT1				I	ST	External Interrupt 1.
RB2/INT2	35	38	10			
RB2				I/O	TTL	Digital I/O.
INT2				I	ST	External Interrupt 2.
RB3/CCP2	36	39	11			
RB3				I/O	TTL	Digital I/O.
CCP2				I/O	ST	Capture2 input, Compare2 output, PWM2 output.
RB4	37	41	14	I/O	TTL	Digital I/O. Interrupt-on-change pin.
RB5/PGM	38	42	15			
RB5				I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGM				I/O	ST	Low Voltage ICSP programming enable pin.
RB6/PGC	39	43	16			
RB6				I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGC				I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/PGD	40	44	17			
RB7				I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGD				I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

TABLE 1-3: PIC18F4X2 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	PLCC	TQFP			
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32			
RC0				I/O	ST	Digital I/O.
T1OSO				O	—	Timer1 oscillator output.
T1CKI				I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2	16	18	35			
RC1				I/O	ST	Digital I/O.
T1OSI				I	CMOS	Timer1 oscillator input.
CCP2				I/O	ST	Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	17	19	36			
RC2				I/O	ST	Digital I/O.
CCP1				I/O	ST	Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37			
RC3				I/O	ST	Digital I/O.
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	23	25	42			
RC4				I/O	ST	Digital I/O.
SDI				I	ST	SPI Data In.
SDA				I/O	ST	I^2C Data I/O.
RC5/SDO	24	26	43			
RC5				I/O	ST	Digital I/O.
SDO				O	—	SPI Data Out.
RC6/TX/CK	25	27	44			
RC6				I/O	ST	Digital I/O.
TX				O	—	USART Asynchronous Transmit.
CK				I/O	ST	USART Synchronous Clock (see related RX/DT).
RC7/RX/DT	26	29	1			
RC7				I/O	ST	Digital I/O.
RX				I	ST	USART Asynchronous Receive.
DT				I/O	ST	USART Synchronous Data (see related TX/CK).

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

TABLE 1-3: PIC18F4X2 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	PLCC	TQFP			
						PORTD is a bi-directional I/O port, or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
RD0/PSP0	19	21	38	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD1/PSP1	20	22	39	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD2/PSP2	21	23	40	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD3/PSP3	22	24	41	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD4/PSP4	27	30	2	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD5/PSP5	28	31	3	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD6/PSP6	29	32	4	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD7/PSP7	30	33	5	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O		
RE0					ST	Digital I/O.
$\overline{RD}$					TTL	Read control for parallel slave port (see also $\overline{WR}$ and $\overline{CS}$ pins).
AN5					Analog	Analog input 5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O		
RE1					ST	Digital I/O.
$\overline{WR}$					TTL	Write control for parallel slave port (see $\overline{CS}$ and $\overline{RD}$ pins).
AN6					Analog	Analog input 6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O		
RE2					ST	Digital I/O.
$\overline{CS}$					TTL	Chip Select control for parallel slave port (see related $\overline{RD}$ and $\overline{WR}$).
AN7					Analog	Analog input 7.
VSS	12, 31	13, 34	6, 29	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	12, 35	7, 28	P	—	Positive supply for logic and I/O pins.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F442/242

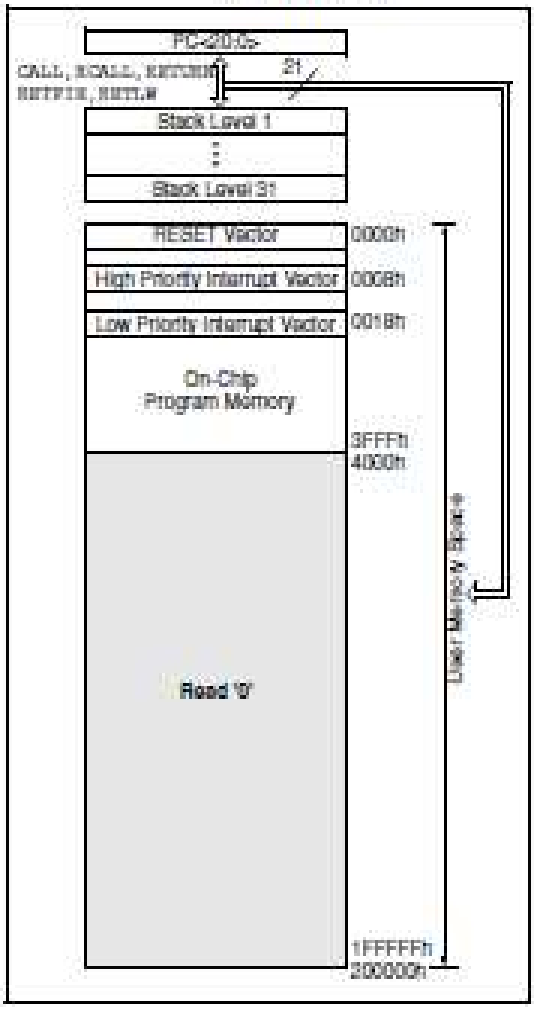


FIGURE 4-2: PROGRAM MEMORY MAP AND STACK FOR PIC18F452/252

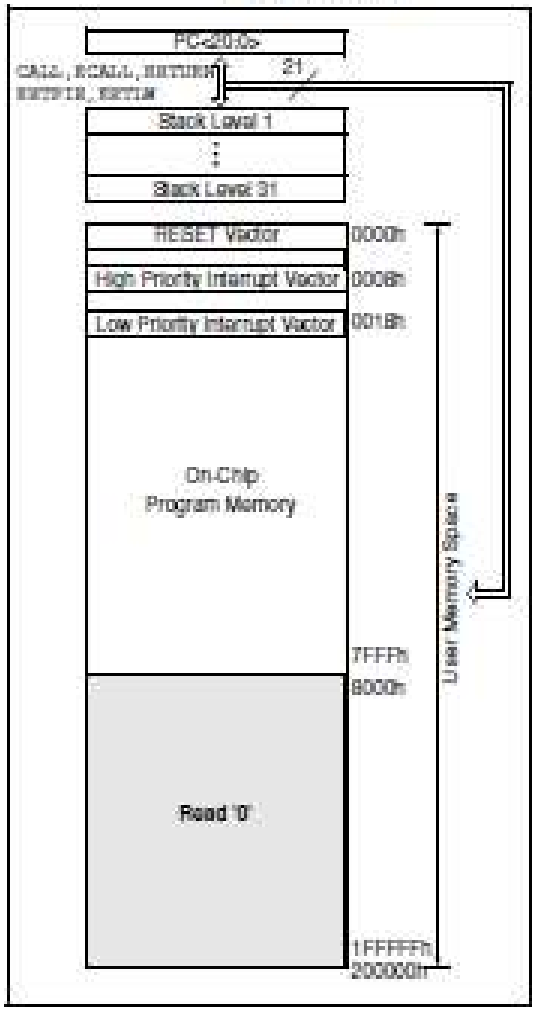
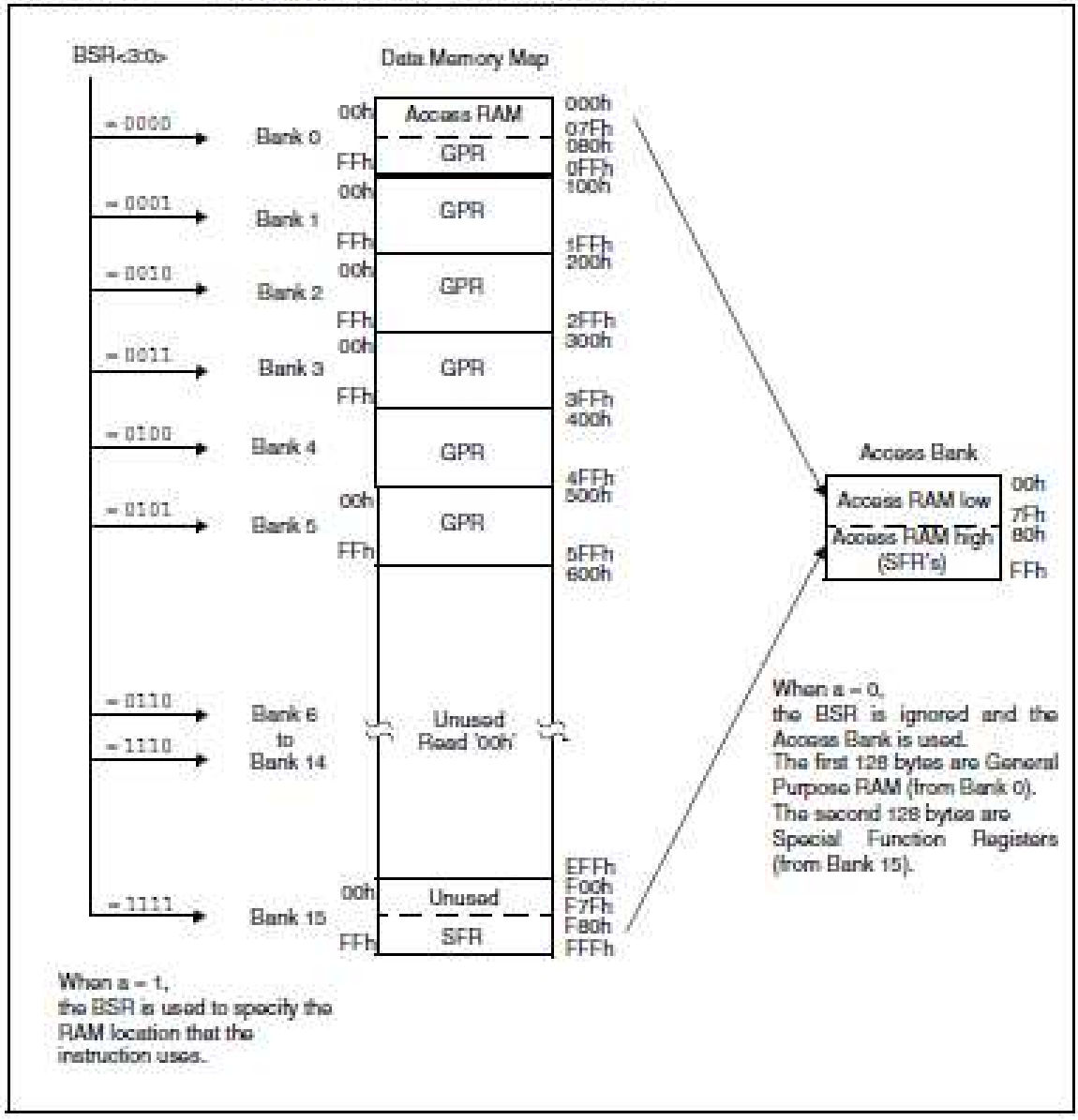


FIGURE 4-7: DATA MEMORY MAP FOR PIC18F252/452



Anexos 2 Lector de huellas digitales

Reglas de codificación

1. Paquete de datos utiliza 0xC0 como carácter de inicio y fin.
2. Si hay 0xC0 en el paquete y luego reemplazarlo con 0xDB y 0xDC.
3. Si hay 0xDB, añadir 0xDD después de ella.
4. Paquete de datos no puede entrar diagrama de flujo de ejecución de forma independiente. Debe seguir Paquete de comandos o paquete de respuesta. Longitud de paquete es el paquete de bytes de contenido (comando, parámetro o datos).

Decodificación Reglas

1. El carácter 0xC0 recibido por el destinatario es considerado como marca Fase Fronteriza.
2. Si se recibe con 0xDB siguiente 0xDC, luego reemplazarlo con 0xC0.
3. Si 0xDB es recibido con n 0xDD, a continuación, eliminar una 0xDD.
4. Check sum es la suma de todos los bytes desde Bandera de paquetes a Suma de verificación (antes de la codificación). No tenga en cuenta los valores que excedan 2 bytes.

Comando en Detalle

Detectar Dedo (Detect Finger)

Comando: DetectFinger
Función: Detección de dedos de sensor
Parámetro de entrada: Ninguno
Parámetro de retorno: Confirmar poco
Comando Código: 01H

Comando de formato de paquetes:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes
Bandera de paquete	Dirección de módulo	Tamaño del paquete	Código del comando	Check sum
01H	00Hx 4	0001H	01H	0003H

Formato Detect Finger

Respuesta Paquete Formato:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes
Bandera de paquete	Dirección de módulo	Tamaño del paquete	Código del comando	Check sum
07H	00Hx 4	0001H	xxH	sum

Nota: Código de confirmación= 00H Dedo detectado;

Código de confirmación= 01H	Error recibido en el paquete
Código de confirmación= 02H	Dedo no detectado

Sum significa Check Sum

Respuesta Detect Finger

Inscríbase Imagen (Get Imagen)

Comando: GetImage

Función: Inscríbase imagen del sensor y almacenar la imagen en Image Buffer[43]. Volver 5 parámetros, incluyendo porcentaje de área de huella digital válido, arriba / abajo / izquierda y derecha de fronteras, etc.

Entrada de Parámetros: ninguno

Parámetro de retorno: Confirmar poco, área válida (porcentaje) y el borde hacia arriba / abajo / izquierda / derecha.

Comando Código: 02H

Comando de formato de paquetes:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes
Bandera de paquete	Dirección de módulo	Tamaño del paquete	Código del comando	Check sum
01H	00Hx 4	0001H	02H	0004H

Nota: Dirección Módulo valor por defecto es 0

Formato Inscribir Imagen

[43] ubicación de la memoria en un Disco o en un instrumento digital reservada para el almacenamiento temporal de información digital

Respuesta Paquete Formato:

1 Byte	4 Bytes	2 Bytes	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	2 Bytes
Bandera de paquete	Resvd	Tamaño del paquete	Confirmación de Código	Área	Borde Superior	Borde Inferior	Borde Izquierdo	Borde Derecho	Check Sum
07H	00Hx 4	06H	xxH	ValidArea	TB	BB	LB	RB	Sum

Nota: Código de confirmación= 00H Imagen de huella exitosa;

Código de confirmación= 01H Error recibido en el paquete;

Código de confirmación= 03H Dedo de inscripción de huella;

Sum significa Check Sum

Respuesta Inscribir Imagen

Generar Detalle de Huellas (Generate Minutiae)

Comando: Get Templet

Función: Generar minucias como se definen en la lista de configuración de sistema. Huella característica del fichero, generada a partir de las imágenes en ImageBuffer, se almacena en CharBufferA o CharBufferB.

Parámetro de entrada: BufferID (No. minucias buffer)

Parámetro de retorno: Código de confirmación

Comando Código: 03H

Comando de formato de paquetes:

1 Byte	4 Bytes	2 Bytes	1 Byte	1 Byte	2 Bytes
Bandera de paquete	Dirección de módulo	Tamaño del paquete	Código del comando	Buffer ID	Check sum
01H	00Hx 4	0002H	03H	Buffer ID	sum

Respuesta Paquete Formato:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes
Bandera de paquete	Resvd	Tamaño del paquete	Código del comando	Check sum
07H	00Hx 4	0001H	xxH	sum

Nota: Código de confirmación= 00H Detalle de huella correcta;
Código de confirmación= 01H Error recibido en el paquete
Código de confirmación= 04H Error por imagen de huella (dedo no limpio).
Código de confirmación= 05H Error por imagen de huella (dedo húmedo)
Código de confirmación= 15H Imagen de huella no existe en el Buffer.
Sum significa Check Sum

Respuesta Generar Detalle de Huellas

Búsqueda de huellas dactilares (Search Fingerprint)

Comando: Search

Función: La búsqueda de la totalidad o parte de la base de datos de huellas dactilares basado en archivos de CharBufferA o CharBufferB. Si la huella digital correcta, devuelva el número de página y Información del usuario.

Parámetro de entrada: BufferID, StartPage (página de inicio), pageNum (Número de página)

Parámetro de retorno: Confirmar Bit, número de página (de la plantilla de huella digital coincide) y la información del usuario (32 bytes)

Código de comando: 05H

Comando de formato de paquetes:

1 Byte	4 Bytes	2 Bytes	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Bandera de paquete	Dirección de módulos	Tamaño del paquete	Confirmación de Código	BufferID	Parámetros	Parámetros	Check Sum
01H	00Hx 4	0006H	05H	BufferID	Inicio Pag.	Número Pag	Sum

Nota: Dirección default de módulo el valor es 0.

Nota: CharBufferA, CharBufferB y CharBufferC los códigos son 1h, 2h y 3h respectivamente.

Formato Búsqueda de Huellas

Respuesta Paquete Formato:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes	32 Bytes	2 Bytes
Bandera de paquete	Resvd	Tamaño del paquete	Confirmación de Código	Número de Página	Información de usuario	Check Sum
07H	00Hx 4	00023H	xxH	PáginaID	UserInfo	Sum

Nota: Código de confirmación= 00H Huella encontrada;
Código de confirmación= 01H Error recibido en el paquete
Código de confirmación= 09H Huella no encontrada
Sum significa Check Sum

Respuesta Búsqueda de Huellas

Combinar detalles de la huella (Generar plantilla) – Merge Minutiae（Generate Template）

Comando: MergeTwoTemplate

Función: Combinar los archivos de CharBufferA y CharBufferB para generar la plantilla; almacenar los resultados en ModelBuffer.

Parámetro de entrada: Ninguno

Parámetro de retorno: Confirmar Bit

Comando Código: 06H

Comando de formato de paquetes:

Comando de formato de paquetes:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes
Bandera de paquete	Dirección de módulo	Tamaño del paquete	Código del comando	Check sum
01H	00Hx 4	0001H	06H	0008H

Formato Generar Plantilla

Respuesta Paquete Formato:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes
Bandera de paquete	Resvd	Tamaño del paquete	Código del comando	Check sum
07H	00Hx 4	0001H	xxH	sum

Nota: Código de confirmación= 00H Combinación exitosa;
Código de confirmación= 01H Error recibido en el paquete
Código de confirmación= 09H Error dos huellas no pertenecen al mismo dedo.
Sum significa Check Sum

Respuesta Generar Plantilla

Tienda de plantillas (StoreTemplet)

Comando: StoreTemplet

Función: Guarda el archivo de plantilla en la base de datos ModelBuffer flash con designó número pageID.

Parámetro de entrada: BufferID (Buffer ID), pageid (base de datos de huellas dactilares de Plantillas ID).

Parámetro Return: Confirmar Bit

Código de comando: 07H

Comando de formato de paquetes:

1 Byte	4 Bytes	2 Bytes	1 Byte	1 Byte	2 Bytes	2 Bytes
Bandera de paquete	Dirección de módulo	Tamaño del paquete	Confirmación de Código	Buffer ID	Número de página	Check Sum
07H	00Hx 4	0004H	07H	BufferID	PáginaID	Sum

Nota: La dirección default de los módulos es 0; CharBufferA, CharBufferB y CharBufferC los códigos son 1h, 2h y 3h respectivamente.

Formato Tienda de Plantillas

Respuesta Paquete Formato:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes
Bandera de paquete	Resvd	Tamaño del paquete	Código del comando	Check sum
07H	00Hx 4	0001H	xxH	sum

Nota: Código de confirmación= 00H Almacenamiento de huellas exitoso;
Código de confirmación= 01H Error recibido en el paquete
Código de confirmación= 0bH La PáginaID ha excedido el rango de huellas en la base de datos.
Sum significa Check Sum

Respuesta Tienda de Plantillas

Luz de indicación de Flash

Comando: FlashLED

Función: Instruya módulo a parpadear las luces según la petición

Parámetro de entrada: LEDcode

Código Led	Modo flash (Led)
01h	Una vez luz roja
02h	Dos veces luz roja
03h	Tres veces luz roja
04h	Cuatro veces luz roja
11h	Una vez luz verde
12h	Dos veces luz verde
13h	Tres veces luz verde
14h	Cuatro veces luz verde
20h	login exitoso (luz roja y verde tres veces)
30h	fallo de login (luz roja y verde simultaneamente)
40h	una vez luz roja y verde
50h	base de datos llena
60h	tiempo fuera

1 Código Led

LEDtime

Tiempo de Led	Led time
1h	50ms
2h	100ms
3h	150ms
4h	200ms
5h	250ms
6h	300ms
7h	350ms

Tiempo de Led

Parámetro Return: Confirmar Bit

Comando Código: 16H

Comando de formato de paquetes:

1 Byte	4 Bytes	2 Bytes	1 Byte	1 Byte	1 Byte	2 Bytes
Bandera de paquete	Resvd	Tamaño del paquete	Confirmación de Código	Código del LED	Tiempo de LED	Check Sum
01H	00Hx 4	0003H	16H	LED code	LEDtime	Sum

Formato Flash LED

Respuesta Paquete Formato:

1 Byte	4 Bytes	2 Bytes	1 Byte	2 Bytes
Bandera de paquete	Resvd	Tamaño del paquete	Código del comando	Check sum
07H	00Hx 4	0001H	xxH	sum

Nota: Código de confirmación= 00H OK;

Código de confirmación= 01H Error recibido en el paquete

Sum significa Check Sum

Respuesta Flash LED

Anexos 3 Reloj a tiempo real DS1307 (RTC)

RTC y Mapa de Direcciones RAM

Los registros de RTC están situados en localizaciones de dirección 00h a 07h. Los registros RAM están situados en localizaciones de dirección 08h a 3Fh. Durante un acceso multi-byte, cuando el puntero llega a la dirección 3Fh, el fin del espacio de RAM, esto devuelve a la posición 00h, el principio del espacio de reloj.

00H	SEGUNDOS
	MINUTOS
	HORAS
	DIA
	FECHAS
	MES
	AÑO
07H	CONTROL
08H-3FH	RAM-56x8

El contenido de los registros de tiempo y calendario están en formato BCD. El registro del día de la semana se incrementa en la medianoche. Los valores que corresponden a los días de la semana son definidos por el usuario, pero debe ser secuencial (es decir, si 1 es igual a domingo, entonces 2 es igual a lunes, y así sucesivamente). Entradas de tiempo y fecha ilógicos causa una operación indeterminada. El Bit 7 del registro 0 es el bit interrupción de reloj alto (CH). Cuando este bit está establecido en 1, el oscilador está desactivado. Cuando se borra a 0, se habilita el oscilador.

Antes de hacer una lectura, se requiere hacer al menos una escritura, para enviar una dirección que pondrá el puntero del registro en el DS1307.

El DS1307 se puede ejecutar en modo de 12 horas o 24 horas. El bit 6 del registro de las horas se define como bit del modo de seleccionar 12 o 24 horas. Cuando el modo seleccionado es alto, es de 12 horas. En el modo 12 horas, el bit 5 es el bit AM/PM con lógica alta es PM. En modo 24 horas, el bit 5 es el bit, segundas 10 horas (20 - 23 horas).

Al leer o escribir los registros de hora y fecha actual se transfiere a un segundo conjunto de registros (buffer), para evitar errores cuando los registros internos se actualizan.

Cuando se leen los registros de hora y fecha, los buffers de usuario se sincronizan con los registros internos en cualquier START I²C. La información horaria se lee de estos segundos registros, mientras que el reloj sigue funcionando. Esto elimina la necesidad de volver a leer los registros, en caso de actualización de los registros internos durante una lectura. La cadena de divisores se reinicializa, cada vez que el registro segundos sea escrito. La transferencia de escritura en el I²C se produce con un reconocimiento desde el DS1307. Una vez que la cadena de divisores es

reinicializada, para evitar problemas de volcado, los registros de fecha y tiempo restante deben ser escritos dentro de un segundo.

REGISTRO DE CONTROL

En el DS1307 el registro de control se usa para controlar el funcionamiento del Pin SQW/OUT.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

Bit 7: OUT (Output control - control de Salida): Este bit controla el nivel de salida del Pin SQW/OUT cuando la onda cuadrada de salida se desactiva. Si SQWE = 0, el nivel lógico en el pin SQW/OUT es 1, si OUT = 1 y SQW/OUT es 0 si OUT = 0.

Bit 4: SQWE (Square Wave Enable - Onda Cuadrada Habilitada): Este bit, cuando se establece a lógica 1, habilita la salida del oscilador. La frecuencia de onda cuadrada de salida depende del valor de los bits RS0 y RS1. Con la onda cuadrada de salida establecida a 1Hz, el reloj registra la actualización sobre el borde decreciente de la onda cuadrada.

Bits 1, 0: RS (Rango Seleccionado): Estos bits controlan la frecuencia de onda cuadrada de salida cuando han habilitado la salida de onda cuadrada.

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

Anexos 4 Programa del sistema (Mikro C)

```
//------------TIMER 0-------------
short int aa=0;
short int bb=0;
short int cc=0;
short int dd=0;
short int ee=0;
short int ff=0;
short int lectura=0;
int segundos=10;
int minutos=59;
int horas=11;
int conteo1=0; //Variable conteo segundos
short int habilita=0;
/-----------------PARA EL RELOJ------------
unsigned short read_ds1307(unsigned short address );
void write_ds1307(unsigned short address,unsigned short w_data);
unsigned short sec;
unsigned short minute;
unsigned short hour;
unsigned short day;
unsigned short date;
unsigned short month;
unsigned short year;
unsigned short dat;
char time[9];
char ddate[11];
unsigned char txt[10];
unsigned char BCD2UpperCh(unsigned char bcd);
unsigned char BCD2LowerCh(unsigned char bcd);
//-------------FIN RELOJ------------
unsigned char _data = 0x1E;
unsigned char dato[100];
unsigned char compara[12];
unsigned char compara1[17];
unsigned char compara2[47];
unsigned char muestra[20];
unsigned char punto1[14];
unsigned char punto2[14];
unsigned char punto3[14];
unsigned char punto4[14];
unsigned char punto5[14];
unsigned char numhuella[2];
short int huella, us=0;
char uart_rd;
short int control=0;
shortinti=0,
verifica=0,verifica2=0,verifica3=0,verifica4=0
,verifica5=0;

void interrupt()
{
if (INTCON.T0IF=1) //Entra interrupcion timer 0
{
   conteo1=conteo1+1;
   if (conteo1>=20)
   {
      segundos=segundos+1;
      conteo1=0;
   }
   if (segundos>=180)
   {
     portb.F2=1;
```

```
      portb.F7=0;
      portb.F6=0;
      Delay_ms(500);
      portb.F2=0;
      control=100;
      segundos=0;
      conteo1=0;
   }
   TMR0H=0;
   TMR0L=177;
   INTCON.T0IF=0;
}
}
void recupera()
{
   for (i=0;i<20;i++)
   {
      if (dato[10]==muestra[i])
      {
          huella=i;
      }
   }
}

void leehora()
{
//----------------------------------------
sec=read_ds1307(0); // read second
minute=read_ds1307(1); // read minute
hour=read_ds1307(2); // read hour
day=read_ds1307(3); // read day
date=read_ds1307(4); // read date
month=read_ds1307(5); // read month
year=read_ds1307(6); // read year
time[0] = BCD2UpperCh(hour);
time[1] = BCD2LowerCh(hour);
time[2] = ':';
time[3] = BCD2UpperCh(minute);
time[4] = BCD2LowerCh(minute);
time[5] = ':';
time[6] = BCD2UpperCh(sec);
time[7] = BCD2LowerCh(sec);
time[8] = '\0';
ddate[0] = BCD2UpperCh(date);
ddate[1] = BCD2LowerCh(date);
ddate[2] ='/';
ddate[3] = BCD2UpperCh(month);
ddate[4] = BCD2LowerCh(month);
ddate[5] ='/';
ddate[6] = '2';
ddate[7] = '0';
ddate[8] = BCD2UpperCh(year);
ddate[9] = BCD2LowerCh(year);
ddate[10] = '\0';
Delay_ms(50);
}
void detectadedo()
{
UART1_Write(0xC0);
UART1_Write(0x01);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x01);
UART1_Write(0x01);
UART1_Write(0x00);
UART1_Write(0x03);
UART1_Write(0xC0);
}
void luzdoble()
```

```
{
UART1_Write(0xC0);
UART1_Write(0x01);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x03);
UART1_Write(0x16);
UART1_Write(0x12);
UART1_Write(0x07);
UART1_Write(0x03);
UART1_Write(0x03);
UART1_Write(0xC0);
}
void detectadedo2()
{
UART1_Write(0xC0);
UART1_Write(0x01);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x01);
UART1_Write(0x02);
UART1_Write(0x00);
UART1_Write(0x04);
UART1_Write(0xC0);
}
void detectadedo3()
{
UART1_Write(0xC0);
UART1_Write(0x01);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x02);
UART1_Write(0x03);
UART1_Write(0x02);
UART1_Write(0x00);
UART1_Write(0x08);
UART1_Write(0xC0);
}
void final()
{
UART1_Write(0xC0);
UART1_Write(0x01);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x06);
UART1_Write(0x05);
UART1_Write(0x02);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x00);
UART1_Write(0x0f);
UART1_Write(0x00);
UART1_Write(0x1d);
UART1_Write(0xC0);
}
void main()
{
I2C1_Init(100000); //DS1307 I2C is running
at 100KHz
PORTB = 0;
```

```
TRISB = 0; // Configure PORTB as output
TRISC = 0xFF;
lectura=EEPROM_Read(0x02);
//Set Time
if (lectura!=10)
{
write_ds1307(0,0x80); //Reset second to 0 sec. and stop Oscillator
write_ds1307(1,0x15); //write min 27
write_ds1307(2,0x11); //write hour 14
write_ds1307(3,0x03); //write day of week 2:Monday
write_ds1307(4,0x08); // write date 17
write_ds1307(5,0x01); // write month 6 June
write_ds1307(6,0x13); // write year 8 --> 2008
write_ds1307(7,0x10); //SQWE output at 1 Hz
write_ds1307(0,0x00); //Reset second to 0 sec. and start Oscillator
}
trisb=0b00000000;
portb=0;

UART1_Init(57600);
UART1_Write_Text("ENVIANDO");
Delay_ms(1000);
UART1_Write_Text("TRANSMISION");
Delay_ms(1000);
dato[0]='a';
dato[1]='b';
dato[2]='A';
dato[3]='b';
dato[4]='A';
dato[5]='b';
//------Detector de dedo----------
compara[0]=0xC0;
compara[1]=0x07;
compara[2]=0x00;
compara[3]=0x00;
compara[4]=0x00;
compara[5]=0x00;
compara[6]=0x00;
compara[7]=0x01;
compara[8]=0x00;
compara[9]=0x00;
compara[10]=0x08;
compara[11]=0xC0;
//----------Sacar imagen de huella----------
compara1[0]=0xC0;
compara1[1]=0x07;
compara1[2]=0x00;
compara1[3]=0x00;
compara1[4]=0x00;
compara1[5]=0x00;
compara1[6]=0x00;
compara1[7]=0x06;
compara1[8]=0x00;
compara1[9]=0x5d;
compara1[10]=0x08;
compara1[11]=0x08;
compara1[12]=0x08;
compara1[13]=0x08;
compara1[14]=0x00;
compara1[15]=0x8a;
compara1[16]=0xC0;
//----------Compara huella final----------
compara2[0]=0xC0;
compara2[1]=0x07;
compara2[2]=0x00;
compara2[3]=0x00;
compara2[4]=0x00;
```

```
compara2[5]=0x00;
compara2[6]=0x00;
compara2[7]=0x23;
compara2[8]=0x00;
//--------muestras de huellas----------
muestra[0]=(0x00);
muestra[1]=(0x01);
muestra[2]=(0x02);
muestra[3]=(0x03);
muestra[4]=(0x04);
muestra[5]=(0x05);
muestra[6]=(0x06);
muestra[7]=(0x07);
muestra[8]=(0x08);
muestra[9]=(0x09);
muestra[10]=(0x0A);
muestra[11]=(0x0B);
muestra[12]=(0x0C);
muestra[13]=(0x0D);
muestra[14]=(0x0E);
muestra[15]=(0x0F);
muestra[16]=(0x11);
muestra[17]=(0x12);
muestra[18]=(0x13);
muestra[19]=(0x14);
//-----Punto uno codigo tag-------
punto1[0]=(0X34);
punto1[1]=(0X42);
punto1[2]=(0X30);
punto1[3]=(0X30);
punto1[4]=(0X44);
punto1[5]=(0X41);
punto1[6]=(0X34);
punto1[7]=(0X37);
punto1[8]=(0X39);
punto1[9]=(0X46);
punto1[10]=(0X34);
punto1[11]=(0X39);
//------Punto uno codigo tag------
punto2[0]=(0X34);
punto2[1]=(0X42);
punto2[2]=(0X30);
punto2[3]=(0X30);
punto2[4]=(0X44);
punto2[5]=(0X41);
punto2[6]=(0X31);
punto2[7]=(0X36);
punto2[8]=(0X30);
punto2[9]=(0X41);
punto2[10]=(0X38);
punto2[11]=(0X44);
//------Punto uno codigo tag--------
punto3[0]=(0X34);
punto3[1]=(0X42);
punto3[2]=(0X30);
punto3[3]=(0X30);
punto3[4]=(0X44);
punto3[5]=(0X41);
punto3[6]=(0X33);
punto3[7]=(0X37);
punto3[8]=(0X32);
punto3[9]=(0X39);
punto3[10]=(0X38);
punto3[11]=(0X46);
//-------Punto uno codigo tag-------
punto4[0]=(0X34);
punto4[1]=(0X42);
punto4[2]=(0X30);
punto4[3]=(0X30);
punto4[4]=(0X44);
punto4[5]=(0X41);
punto4[6]=(0X30);
```

```
punto4[7]=(0X43);
punto4[8]=(0X31);
punto4[9]=(0X42);
punto4[10]=(0X38);
punto4[11]=(0X36);
//----Punto uno codigo tag---
punto5[0]=(0X34);
punto5[1]=(0X42);
punto5[2]=(0X30);
punto5[3]=(0X30);
punto5[4]=(0X44);
punto5[5]=(0X41);
punto5[6]=(0X33);
punto5[7]=(0X38);
punto5[8]=(0X33);
punto5[9]=(0X43);
punto5[10]=(0X39);
punto5[11]=(0X35);
while(1)
{
//--------------------------------
detectadedo();
do
{
  if (UART1_Data_Ready())
  {
     uart_rd = UART1_Read();
     dato[control]=uart_rd;
     control=control+1;
  }
}while(control<12);

for (i=0;i<12;i++)
{
    if (dato[i]==compara[i])
    {
        verifica=verifica+1;
    }
}
if (verifica==12)
{
 portb.F0=1;
 Delay_ms(100);
 portb.F0=0;
Delay_ms(100);
detectadedo2();
 control=0;
 verifica=0;
 do
 {
    if (UART1_Data_Ready())
    {
       uart_rd = UART1_Read();
       dato[control]=uart_rd;
       control=control+1;
    }
 }while(control<17);


 for (i=0;i<9;i++)
 {
   if (dato[i]==compara1[i])
   {
      verifica=verifica+1;
   }
 }

 if (verifica==9)
 {
   portb.F0=1;
```

```
Delay_ms(100);
portb.F0=0;
Delay_ms(100);
Lcd_Out(2,0,"IMAGEN LEIDA");
Delay_ms(200);
detectadedo3();
verifica=0;
control=0;
do
{
  if (UART1_Data_Ready())
  {
      uart_rd = UART1_Read();
      dato[control]=uart_rd;
      control=control+1;
  }
 }while(control<12);
 for (i=0;i<12;i++)
 {
    if (dato[i]==compara[i])
    {
      verifica=verifica+1;
    }
 }
 if (verifica==12)
 {
   portb.F0=1;
   Delay_ms(100);
   portb.F0=0;
   Delay_ms(100);
   Lcd_Out(2,0,"HUELLA GENERADA");
   Delay_ms(200);
   final();
   //----AQUI SACA QUE HUELLA ES----
   control=0;
   verifica=0;
do
{
if (UART1_Data_Ready())
{
  uart_rd = UART1_Read();
  dato[control]=uart_rd;
  control=control+1;
}
}while(control<45);

for (i=0;i<9;i++)
{
   if (dato[i]==compara2[i])
   {
     verifica=verifica+1;
   }
}
  if (verifica==9)
{
  portb.F1=1;
  Delay_ms(100);
  portb.F1=0;
  Delay_ms(100);
  control=0;
  verifica=0;
  recupera();
  ShortToStr(huella,txt);
  Lcd_Out(1,6,txt);
  Delay_ms(500);
  //-----LEE HORA--------
  leehora();
  Lcd_Out(1,6,time);
  Lcd_Out(2,6,ddate);
  //-----FIN LEE HORA----
  portb.F6=1;
  UART1_Init(9600);
```

```
 Delay_ms(4000);
 portb.F2=1;
 Lcd_Out(1,6,"TARJETA");
luzdoble();
 //---LECTURA DE TAG---
 control=0;
 //----ACTIVAR TIMER 0----
 INTCON = 0b11100000;
 TMR0H=0;
 TMR0L=177;
 INTCON.TMR0IE=1;
 INTCON.T0IF=0;
 T0CON=0b11000111;
  //----FIN ACTIVACION----
 do
 {
  if (UART1_Data_Ready())
  {
    uart_rd = UART1_Read();
    dato[control]=uart_rd;
    control=control+1;
  }
  else
  {
          }
 }while(control<14);
 //----VERIFICACION DE TAG----
 if (control!=100)
 {
 for (i=0;i<12;i++)
 {
    TMR0H=0;
    TMR0L=177;
    INTCON.TMR0IE=1;
    INTCON.T0IF=0;
    intcon=0;
    if (dato[i+1]==punto1[i])
    {
      verifica=verifica+1;
    }
    if (dato[i+1]==punto2[i])
    {
      verifica2=verifica2+1;
    }
    if (dato[i+1]==punto3[i])
    {
      verifica3=verifica3+1;
    }
    if (dato[i+1]==punto4[i])
    {
      verifica4=verifica4+1;
    }
    if (dato[i+1]==punto5[i])
    {
      verifica5=verifica5+1;
    }
 }
 if (verifica==12)
 {
     us=1;
 }
 if (verifica2==12)
 {
     us=2;
 }
 if (verifica3==12)
 {
     us=3;
 }
 if (verifica4==12)
 {
      us=4;
```

```
        }
        if (verifica5==12)
        {
            us=5;
        }
        portb.F7=1;
        Delay_ms(3000);
        UART1_Init(9600);
        UART1_Write_Text("OPW
CONTROL.txt");
        Delay_ms(1000);
        UART1_Write(0X0D);
        Delay_ms(1000);
        UART1_Write_Text("WRF   45");   //36
antes de huella
        Delay_ms(1000);
        UART1_Write(0X0D);
        Delay_ms(1000);
        //-----------------------------------
        UART1_Write_Text("@");
        UART1_Write_Text(time);
        UART1_Write_Text("&");
        UART1_Write_Text(ddate);
        UART1_Write_Text("&");
        UART1_Write_Text(dato);
        UART1_Write_Text("&");
        if (huella==0)
          UART1_Write_Text("0");
        if (huella==1)
          UART1_Write_Text("1");
        if (huella==2)
          UART1_Write_Text("2");
        if (huella==3)
          UART1_Write_Text("3");
        if (huella==4)
          UART1_Write_Text("4");
        if (huella==5)
          UART1_Write_Text("5");
        if (huella==6)
          UART1_Write_Text("6");
        if (huella==7)
          UART1_Write_Text("7");
        if (huella==8)
          UART1_Write_Text("8");
        if (huella==9)
          UART1_Write_Text("9");
        UART1_Write_Text("&");
        if (us==1)
          UART1_Write_Text("PUNTO1");
        if (us==2)
          UART1_Write_Text("PUNTO2");
        if (us==3)
          UART1_Write_Text("PUNTO3");
        if (us==4)
          UART1_Write_Text("PUNTO4");
        if (us==5)
          UART1_Write_Text("PUNTO5");
        UART1_Write_Text("%");
        //-------DATOS-------
        Delay_ms(1000);
        UART1_Write(0X0D);
        Delay_ms(1000);
        UART1_Write_Text("CLF
CONTROL.txt");
        Delay_ms(1000);
        UART1_Write(0X0D);
        Delay_ms(1000);
    }
    else
    {
        intcon=0;
       Lcd_Out(2,2,"TIEMPO EXPIRADO");
```

```
}
        verifica2=0;
        verifica3=0;
        verifica4=0;
        verifica5=0;
        Delay_ms(3000);
        control=0;
        verifica=0;
        UART1_Init(57600);
        portb.F7=0;
        portb.F6=0;
        Delay_ms(400);
        portb.F2=0;
        //-------FIN LECTURA DE TAG-------
      }
      else
      {
        portb.F3=1;
        Delay_ms(300);
        portb.F3=0;
        control=0;
        verifica=0;
        dato[8]=(0x04);
        UART1_Init(57600);
        Delay_ms(1000);
      }
      //------FIN DETECCION DE HUELLA----
    }
    else
    {
          portb.F3=1;
        Delay_ms(300);
        portb.F3=0;
       control=0;
       verifica=0;
    }
  }
  else
  {
       portb.F3=1;
       Delay_ms(300);
      portb.F3=0;
      control=0;
      verifica=0;
  }
}
else
{
      portb.F3=1;
      Delay_ms(100);
      portb.F3=0;
    control=0;
    verifica=0;
}
control=0;
verifica=0;
i=0;
}
}
//---------SOLO REJOJ----------
unsigned short read_ds1307(unsigned short address)
{
I2C1_Start();
I2C1_Wr(0xd0); //address 0x68 followed by direction bit (0 for write, 1 for read) 0x68 followed by 0 --> 0xD0
I2C1_Wr(address);
I2C1_Repeated_Start();
I2C1_Wr(0xd1); //0x68 followed by 1 --> 0xD1
dat=I2C1_Rd(0);
```

```
I2C1_Stop();
return(dat);
}
unsigned char BCD2UpperCh(unsigned char bcd)
{
return ((bcd >> 4) + '0');
}
unsigned char BCD2LowerCh(unsigned char bcd)
{
return ((bcd & 0x0F) + '0');
}
void write_ds1307(unsigned short address,unsigned short w_data)
{
EEPROM_Write(0x02,10);
I2C1_Start(); // issue I2C start signal
//address 0x68 followed by direction bit (0 for write, 1 for read) 0x68 followed by 0 --> 0xD0
I2C1_Wr(0xD0); // send byte via I2C (device address + W)
I2C1_Wr(address); // send byte (address of DS1307 location)
I2C1_Wr(w_data); // send data (data to be written)
I2C1_Stop(); // issue I2C stop signal
}
```

Anexos 5 Programa en C# del circuito para añadir huellas

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Threading;
using System.Windows.Forms;
namespace PRUEBALECTOR
{
    public partial class Form1 : Form
    {
        public string respuesta;
        public int ab;
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            serialPort1.Open();
        }
        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
```

```
            respuesta = serialPort1.ReadExisting();
            this.Invoke(new EventHandler(lecturas));
        }
        private void lecturas(object sender, EventArgs e)
        {
textBox1.AppendText(respuesta);
        }
        private void button1_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[36];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;

            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x01;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x03;
            buffer_out1[11] = 0x02;
            buffer_out1[12] = 0x00;
            buffer_out1[13] = 0x00;
            buffer_out1[14] = 0x00;
            buffer_out1[15] = 0x00;
            buffer_out1[16] = 0x00;
            buffer_out1[17] = 0x00;
            buffer_out1[18] = 0x00;
            buffer_out1[19] = 0x00;
            buffer_out1[20] = 0x02;
            buffer_out1[21] = 0x08;
            buffer_out1[22] = 0x00;
            buffer_out1[23] = 0x00;
            buffer_out1[24] = 0x00;
            buffer_out1[25] = 0x00;
            buffer_out1[26] = 0x00;
            buffer_out1[27] = 0x00;
            buffer_out1[28] = 0x00;
            buffer_out1[29] = 0x00;
            buffer_out1[30] = 0x08;
            buffer_out1[31] = 0xC0;       serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }
        private void button2_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
```

```
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x15;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x17;
            buffer_out1[11] = 0xC0;           serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }
        private void button3_Click(object sender, EventArgs e)
        {
            int a=1;   Console.WriteLine(a.ToString("x"));
            MessageBox.Show(a.ToString("x"));
        }
        private void button4_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];


            buffer_out1[0] = 0xC0;

            buffer_out1[1] = 0x01;

            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x03;
            buffer_out1[8] = 0x16;
            buffer_out1[9] = 0x12;
            buffer_out1[10] = 0x1B;
            buffer_out1[11] = 0xC0;
serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }
        private void button7_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x01;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x03;
            buffer_out1[11] = 0xC0;           serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }
        private void button6_Click(object sender, EventArgs e)
```

```
        {
            byte[] buffer_out1 = new byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x02;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x04;
            buffer_out1[11] = 0xC0;
serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }
        private void button5_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[13];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x02;
            buffer_out1[8] = 0x03;
            buffer_out1[9] = 0x01;
            buffer_out1[10] = 0x00;
            buffer_out1[11] = 0x07;

            buffer_out1[12] = 0xC0;

            serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }

        private void button8_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[14];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x03;
            buffer_out1[8] = 0x16;
            buffer_out1[9] = 0x11;
            buffer_out1[10] = 0x07;
                        //CHECK SUM
            buffer_out1[11] = 0x03;
```

```
            buffer_out1[12] = 0x02;
            buffer_out1[13] = 0xC0;            serialPort1.Write(buffer_out1,
0, buffer_out1.Length);
        }

        private void button12_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x01;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x03;
            buffer_out1[11] = 0xC0;       serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }
        private void button11_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];


            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x02;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x04;
            buffer_out1[11] = 0xC0;
serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }

        private void button10_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[13];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x02;
```

```
            buffer_out1[8] = 0x03;
            buffer_out1[9] = 0x02;
            buffer_out1[10] = 0x00;
            buffer_out1[11] = 0x08;
            buffer_out1[12] = 0xC0;
serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }
        private void button9_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[14];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x03;
            buffer_out1[8] = 0x16;
            buffer_out1[9] = 0x12;
            buffer_out1[10] = 0x07;
            //CHECK SUM
            buffer_out1[11] = 0x03;
            buffer_out1[12] = 0x03;
            buffer_out1[13] = 0xC0;        serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }
        private void button16_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x01;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x03;
            buffer_out1[11] = 0xC0;
serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }
        private void button15_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];


            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
```

```
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x02;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x04;
            buffer_out1[11] = 0xC0;       serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }
        private void button14_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[13];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x02;
            buffer_out1[8] = 0x03;
            buffer_out1[9] = 0x02;
            buffer_out1[10] = 0x00;
            buffer_out1[11] = 0x08;
            buffer_out1[12] = 0xC0;        serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }

        private void button13_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[17];


            buffer_out1[0] = 0xC0;

            buffer_out1[1] = 0x01;

            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x06;
            buffer_out1[8] = 0x05;
            buffer_out1[9] = 0x02;
            buffer_out1[10] = 0x00;
            buffer_out1[11] = 0x00;
            buffer_out1[12] = 0x00;
            buffer_out1[13] = 0x0f;

            buffer_out1[14] = 0x00;
            buffer_out1[15] = 0x1d;
```

```
            buffer_out1[16] = 0xC0;

            serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }

        private void button17_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new      byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffe_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            bufferout1[8] = 0x0f;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x11;
            buffer_out1[11] = 0xC0;
serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }

        private void button18_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x02;
            buffer_out1[8] = 0x25;
            buffer_out1[9] = 0x01;
            buffer_out1[10] = 0x00;
            buffer_out1[10] = 0x29;
            buffer_out1[11] = 0xC0;
serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }

        private void button19_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
```

```
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x04;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x06;
            buffer_out1[11] = 0xC0;           serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }

        private void button20_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];


            buffer_out1[0] = 0xC0;

            buffer_out1[1] = 0x01;

            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x15;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x17;
            buffer_out1[11] = 0xC0;
serialPort1.Write(buffer_out1, 0, buffer_out1.Length);
        }

        private void button21_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[12];
            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x01;
            buffer_out1[8] = 0x06;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x08;
            buffer_out1[11] = 0xC0;            serialPort1.Write(buffer_out1,
0, buffer_out1.Length);
        }

        private void button22_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[15];

            buffer_out1[0] = 0xC0;
```

```
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x04;
            buffer_out1[8] = 0x07;
            buffer_out1[9] = 0x03;
            buffer_out1[10] = 0x00;
            buffer_out1[11] = 0x07;
            buffer_out1[12] = 0x00;
            buffer_out1[13] = 0x14;
            buffer_out1[14] = 0xC0;         serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }
        private void button23_Click(object sender, EventArgs e)
        {
            byte[] buffer_out1 = new byte[14];


            buffer_out1[0] = 0xC0;
            buffer_out1[1] = 0x01;
            buffer_out1[2] = 0x00;
            buffer_out1[3] = 0x00;
            buffer_out1[4] = 0x00;
            buffer_out1[5] = 0x00;
            buffer_out1[6] = 0x00;
            buffer_out1[7] = 0x03;
            buffer_out1[8] = 0x08;
            buffer_out1[9] = 0x00;
            buffer_out1[10] = 0x04;
            buffer_out1[11] = 0x00;
            buffer_out1[12] = 0x10;
            buffer_out1[13] = 0xC0;           serialPort1.Write(buffer_out1, 0,
buffer_out1.Length);
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

Anexos 6 Descripción de los pines de VDIP1

Pin No.	Name	Pin Name on PCB	Type	Description
1	5V0	5V0	PWR Input	5.0 V module supply pin. This pin provides the 5.0V output on the USB 'A' type socket, and also the 3.3V supply to VNCL2, via an on-board 3.3 V L.D.O.
2	LED1	LD1	Output	USB port 1 traffic activity indicator LED. This pin is hard wired to a green LED on board the PCB. It is also brought out onto this pin which allows for the possibility of bring- ing out an additional LED traffic indicator out of the VDIP1 board. For example, if the VDIP1 USB connector is brought out onto an instrument front panel, an activity LED could be mounted along side it.
3	LED2	LD2	Output	USB port 2 traffic activity indicator LED. This pin is hard wired to a green LED on board the PCB. It is also brought out onto this pin which allows for the possibility of bring- ing out an additional LED traffic indicator out of the VDIP1 board. For example, if the VDIP1 USB connector is brought out onto an instrument front panel, an activity LED could be mounted along side it.
4	USBD1P	U1P	I/O	USB host / slave port 1 - USB Data Signal Plus with integrated pull up / pull down resistor. Module has on board 27 Ω USB series resistor. This pin can be brought out along with pin 5 to provide a second USB port, if required
5	USBD1M	U1M	I/O	USB host / slave port 1 - USB Data Signal Minus with integrated pull up / pull down resistor. Module has on board 27 Ω USB series resistor. This pin can be brought out along with pin 4 to provide a second USB port, if required
6	ADBUS0	AD0	I/O	5V safe bidirectional data / control bus, AD bit 0
7	GND	GND	PWR	Module ground supply pin
8	ADBUS1	AD1	I/O	5V safe bidirectional data / control bus, AD bit 1
9	ADBUS2	AD2	I/O	5V safe bidirectional data / control bus, AD bit 2
10	ADBUS3	AD3	I/O	5V safe bidirectional data / control bus, AD bit 3
11	ADBUS4	AD4	I/O	5V safe bidirectional data / control bus, AD bit 4
12	ADBUS5	AD5	I/O	5V safe bidirectional data / control bus, AD bit 5
13	ADBUS6	AD6	I/O	5V safe bidirectional data / control bus, AD bit 6
14	ADBUS7	AD7	I/O	5V safe bidirectional data / control bus, AD bit 7
15	ACBUS0	AC0	I/O	5V safe bidirectional data / control bus, AC bit 0
16	ACBUS1	AC1	I/O	5V safe bidirectional data / control bus, AC bit 1
17	ACBUS2	AC2	I/O	5V safe bidirectional data / control bus, AC bit 2
18	GND	GND	PWR	Module Ground Supply Pin
19	ACBUS3	AC3	I/O	5V safe bidirectional data / control bus, AC bit 3
20	ACBUS4	AC4	I/O	5V safe bidirectional data / control bus, AC bit 4
21	ACBUS5	AC5	I/O	5V safe bidirectional data / control bus, AC bit 5
22	RESET#	RS#	Input	Can be used by an external device to reset the VNC1L. This pin can be used in combination with PROG# and the UART / parallel FIFO / SPI interface to program firmware into the Vinculum
23	PROG#	PG#	Input	This pin is used in combination with the RESET# pin and the UART / parallel FIFO / SPI interface to pro gra m fir mware into t he VNC1L.
24	3V3	3V3	PWR	3.3V output from VDIP1's on board 3.3V L.D.O.

Anexos 7 Configuración de de los pines I/O

Pin No.	Name	Pin Name on PCB	Type	Description	Data and Control Bus Configuration Options			
					UART Interface	Parallel FIFO Interface	SPI Slave Interface	I/O Port
6	ADBUS0	AD0	I/O	5V safe bidirectional data / control bus, AD bit 0	TXD	D0	SCLK	PortAD0
8	ADBUS1	AD1	I/O	5V safe bidirectional data / control bus, AD bit 1	RXD	D1	SDI	PortAD1
9	ADBUS2	AD2	I/O	5V safe bidirectional data / control bus, AD bit 2	RTS#	D2	SDO	PortAD2
10	ADBUS3	AD3	I/O	5V safe bidirectional data / control bus, AD bit 3	CTS#	D3	CS	PortAD3
11	ADBUS4	AD4	I/O	5V safe bidirectional data / control bus, AD bit 4	DTR#	D4		PortAD4
12	ADBUS5	AD5	I/O	5V safe bidirectional data / control bus, AD bit 5	DSR#	D5		PortAD5
13	ADBUS6	AD6	I/O	5V safe bidirectional data / control bus, AD bit 6	DCD#	D6		PortAD6
14	ADBUS7	AD7	I/O	5V safe bidirectional data / control bus, AD bit 7	RI#	D7		PortAD7
15	ACBUS0	AC0	I/O	5V safe bidirectional data / control bus, AC bit 0	TXDEN#	RXF#		PortAC0
16	ACBUS1	AC1	I/O	5V safe bidirectional data / control bus, AC bit 1		TXE#		PortAC1
17	ACBUS2	AC2	I/O	5V safe bidirectional data / control bus, AC bit 2		RD#		PortAC2
19	ACBUS3	AC3	I/O	5V safe bidirectional data / control bus, AC bit 3		WR		PortAC3
20	ACBUS4	AC4	I/O	5V safe bidirectional data / control bus, AC bit 4				PortAC4

Anexos 8 Interfaz UART

Pin No.	Name	Type	Description
6	TXD	Output	Transmit asynchronous data output
8	RXD	Input	Receive asynchronous data input
9	RTS#	Output	Request To Send Control Output / Handshake signal
10	CTS#	Input	Clear To Send Control Input / Handshake signal
11	DTR#	Output	Data Terminal Ready Control Output / Handshake signal
12	DSR#	Input	Data Set Ready Control Input / Handshake signal
13	DCD#	Input	Data Carrier Detect Control Input
14	RI#	Input	Ring Indicator Contro l Input. When the RemotealkeW up option is enabled in the EEPROM, taking RI# low can be used to resume the PC USB Host controller from suspend
15	TXDEN#	Input	Enable Transmit Data for RS485 designs

Anexos 9 Serial Peripheral Interface (SPI)

Pins No	Name	Type	Description
6	SCLK	Input	SPI Clock Input, 12MHz maximum.
8	SDI	Input	SPI Serial Data Input
9	SDO	Output	SPI Serial Data Output
10	CS	Input	SPI Chip Select Input