UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TÍTULO:

IMPLEMENTACIÓN DE UN CIRCUITO PORTATIL INTERCOMUNICADOR DE MENSAJES DE TEXTO PARA DISCAPACITADOS CON AFONÍA FUNCIONAL

AUTORES:

Juan Carlos Holguín Aguilar

Jhon Orlando Chichande Luna

Previa la obtención del Título

INGENIERO EN TELECOMUNICACIONES

TUTOR:

MsC. Daniel Bohórquez Heras

Guayaquil, Ecuador

2015

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

CERTIFICACIÓN

Certificamos que el presente trabajo fue realizado en su totalidad por los Sres. **Juan Carlos Holguín Aguilar y Jhon Orlando Chichande Luna** como requerimiento parcial para la obtención del título de INGENIERO EN TELECOMUNICACIONES.

TUTOR

_____

MsC. Daniel Bohórquez Heras

DIRECTOR DE CARRERA

_____

MsC. Miguel A. Heras Sánchez

Guayaquil, a los 20 del mes de Febrero del año 2015

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Nosotros, **Juan Carlos Holguín Aguilar y Jhon Orlando Chichande Luna**

DECLARAMOS QUE:

El trabajo de titulación "IMPLEMENTACIÓN DE UN CIRCUITO PORTATIL INTERCOMUNICADOR DE MENSAJES DE TEXTO PARA DISCAPACITADOS CON AFONÍA FUNCIONAL" previa a la obtención del Título de Ingeniero en Telecomunicaciones, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía. Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del Trabajo de Titulación referido.

Guayaquil, a los 20 del mes de Febrero del año 2015

LOS AUTORES

_____ _____

JUAN CARLOS HOLGUÍN AGUILAR JHON ORLANDO CHICHANDE LUNA

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Nosotros, **Juan Carlos Holguín Aguilar y Jhon Orlando Chichande Luna**

Autorizamos a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del Trabajo de Titulación: "IMPLEMENTACIÓN DE UN CIRCUITO PORTATIL INTERCOMUNICADOR DE MENSAJES DE TEXTO PARA DISCAPACITADOS CON AFONÍA FUNCIONAL", cuyo contenido, ideas y criterios es de nuestra exclusiva responsabilidad y autoría.

Guayaquil, a los 20 del mes de Febrero del año 2015

LOS AUTORES

_____ _____

JUAN CARLOS HOLGUÍN AGUILAR JHON ORLANDO CHICHANDE LUNA

DEDICATORIA

A Dios por mostrarnos día a día que con humildad, paciencia y sabiduría todo es posible.

Gracias a esas personas importantes en nuestras vidas, que siempre estuvieron listas para brindarme toda su ayuda, ahora regresamos un poquito de todo ese inmenso cariño y amor que nos han otorgado. Con todo nuestro cariño está tesis la dedicamos a ustedes:

Rosita Edith Chichande L.,

Carmen Luna Pata

Cecilia del Pilar Aguilar P.

Juan Artemio Holguín S.

Carlos Luis Aguilar P.

LOS AUTORES

JUAN CARLOS HOLGUÍN AGUILAR
JHON ORLANDO CHICHANDE LUNA

AGRADECIMIENTO

El presente trabajo de titulación agradecemos a ti Dios por bendecirnos para llegar hasta donde hemos llegado, y hacer realidad esta etapa de nuestras vidas. Queremos expresar nuestro más sincero agradecimiento, reconocimiento y cariño a nuestros padres: Cecilia del Pilar Aguilar P., Juan Artemio Holguín S., Rosita Edith Chichande L., Carmen Luna Pata por todo el esfuerzo que hicieron para darnos una profesión y hacer de nosotros personas de bien, gracias por los sacrificios y la paciencia que demostraron todos estos años, gracias a ustedes hemos llegado a donde estamos, estos han sido apoyo incondicional en toda nuestra vida y futuro. A nuestros tíos, tías, primos y primas como: Carlos Luis Aguilar P., María Concepción Tobar F., Manuel Gabriel Guerrero T., Yessenia Carola Guerrero T.,

A la Universidad Católica de Santiago de Guayaquil y a la Facultad de Educación Técnica para El Desarrollo por darnos la oportunidad de obtener los conocimientos que son impartidos por el personal docente en sus aulas de clases. A nuestro tutor del trabajo de titulación, MsC. Daniel Bohórquez Heras por su esfuerzo y dedicación, quien con sus conocimientos, su experiencia, su paciencia y su motivación ha logrado que podamos concluir con éxito el presente trabajo.

También nos gustaría agradecer a nuestros profesores que durante toda nuestra carrera universitaria, porque todos han aportado con un granito de arena a nuestra formación profesional, y en especial a nuestros profesores y amigos Ing. Carlos Zambrano Montes, Ing. Edwin Palacios Meléndez, Ing. Efraín Suarez Murillo, Ing. Armando Heras Sánchez y el Ing. Bayardo Bohórquez Escobar, por sus consejos, sus enseñanzas y más que todo por la amistad brindada.

De igual manera agradecer a nuestro oponente, Ing. Luis Córdova Rivadeneira por su visión crítica de muchos aspectos que supo transmitir para mejorar nuestro trabajo de titulación. También a nuestros amigos y amigas que siempre nos apoyan de diferentes maneras según sea necesario, pero independientemente del problema o la situación sabemos que podemos

contar con ellos, entre ellos están: Jaime Gastón Benavides M. Shirley Maritza Freire, Arq. Sandra Esparza, Mayra Crespín T., Luís Alberto Suares A. Danny Ramírez, Andrés Posligua Coox, Cleiver Alfonso Fiallos, Jorge Freire Castellanos.

Y por último a esas personas que no están físicamente pero están presentes: José Miguel Alfonso Aguilar, Ana Elizabeth Suarez A. y Jorge Solís.

Son muchas las personas que han formado parte de este logro, a las que nos encantaría agradecerles su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de nuestras vidas. Algunas están aquí con nosotros y otras en nuestros recuerdos, sin importar en donde estén queremos darles las gracias por formar parte de nosotros, por todo lo que nos han brindado y por todas sus bendiciones.

Para ellos:

Muchas gracias y que Dios los bendiga.

LOS AUTORES

JUAN CARLOS HOLGUÍN AGUILAR
JHON ORLANDO CHICHANDE LUNA

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

CALIFICACIÓN

_____

MsC. Daniel Bohórquez Heras

TUTOR

Índice General

Índice de Figuras

Capítulo 2:

Capítulo 3:

Índice de Tablas

Capítulo 3:

Resumen

El aporte que se pretendió brindar al trabajo de titulación fue de diseñar un sistema intercomunicador de mensajes de texto para personas que padecen de afonía funcional, es decir, personas que sufren de cáncer o han nacido con este problema de salud. Se tuvieron que realizar procesos experimentales al momento de diseñar el intercomunicador de mensajes de textos a través de simulación y programación de alto nivel C

Se diseñaron dos aplicaciones como complemento del circuito intercomunicador a través de los programa AppInventor y Java. Al inicio no fue tarea fácil debido a que en la Carrera de Ingeniería en Telecomunicaciones no enseñan el diseño y programación de aplicaciones para dispositivos fijos y móviles.

CAPÍTULO 1: GENERALIDADES DEL TRABAJO DE TITULACIÓN

1.1. Introducción.

Afonía proviene del griego: a-, sin y phonos, sonido, que en la práctica es una terminología muy utilizada por los médicos para describir a la incapacidad de hablar. La afonía medicamente es considerado como una lesión más grave que la disfonía. Las causas que originan la afonía, es el rompimiento del <<nervio laríngeo recurrente>>, este nervio controla cada uno de los músculos existentes en la laringe.

El daño muscular del nervio, es ocasionado por la extirpación mediante cirugía de la tiroides o tumor. La mayoría de pacientes que adolecen de este mal, deben utilizar equipos para poder comunicarse pero su coste es elevado. Aunque, a menudo la mudez es asociado a la sordera, pero ciertos sordomudos, nacen con este inconveniente por lo que nunca han oído y jamás podrán articular palabras correctamente. En otras palabras, las personas pueden ser mudas de nacimiento o pierden su capacidad vocal durante alguna etapa de su vida, ya sea por una lesión o enfermedad.

1.2. Justificación.

En Ecuador un pequeño grupo de personas nacen siendo mudos, sordomudos y otro grupo tienen cáncer de tiroides, los mismos producen afonía. Actualmente, existen dispositivos diseñados para este tipo de incapacidad pero los costes son elevados y no son cubiertos por el Ministerio

de Salud para tratamientos de cáncer y mucho menos de niños que adolecen de este mal congénito.

1.3. Definición del Problema.

Necesidad de implementar un circuito portátil intercomunicador de mensajes de texto para discapacitados con afonía funcional.

1.4. Objetivos del Problema de Investigación.

1.4.1 Objetivo General.

Implementar un circuito portátil intercomunicador de mensajes de texto para discapacitados con afonía funcional.

1.4.2 Objetivos Específicos.

- Describir la Fundamentación Teórica de los Microcontroladores PIC.
- Diseñar el circuito portátil intercomunicador de mensajes de texto para discapacitados con afonía funcional.
- Desarrollar aplicaciones en AppInventor y Java que permita la comunicación con el circuito portátil intercomunicador.

1.5. Idea a Defender.

A través de la implementación del circuito portátil intercomunicador de mensajes de textos permitirá que los discapacitados con afonía funcional puedan comunicarse con familiares, amigos y cualquier ser humano.

1.6. Metodología de Investigación.

Para el presente trabajo de titulación, el tipo de investigación fue exploratorio, porque se examinó un problema existente pero que en nuestra universidad nunca ha sido abordado y menos estudiado.

También es de carácter explicativo, porque describe los fundamentos teóricos de los microcontroladores y sus aplicaciones, que responden a las causas para la ejecución del trabajo de titulación.

En base a lo explicado, el trabajo de titulación es un estudio empírico-analítico.

CAPÍTULO 2: Fundamentación Teórica de Microcontroladores PIC.

2.1. Introducción a los Microcontroladores Embebidos.

La función principal de la Microchip PIC® y otros microcontroladores embebidos es proveer a bajo costo, controles lógicos programables y las interfaces con dispositivos externos. Esto significa que por lo general no están obligados a proporcionar funciones altamente complejas que no pueden reemplazar el procesador ***"Opteron"*** en el servidor del ISP (proveedor de Internet).

Son muy adecuadas para el seguimiento de una variedad de entradas, incluidas las señales digitales, las pulsaciones de botones y entradas analógicas, y responder a ellas mediante las instrucciones pre-programadas que son ejecutadas por el procesador de la computadora incorporado. Un microcontrolador embebido, puede responder a estas entradas con una amplia variedad de salidas que sean apropiados para los diferentes dispositivos. Estas capacidades están disponibles a costos muy razonables y sin mucho esfuerzo.

2.2. Tipos de microcontroladores

Si se fuera a estudiar los microcontroladores de diferentes fabricantes, probablemente estaría desconcertado del número de dispositivos diferentes, que están ahí fuera y todas sus características y capacidades. Resulta útil

pensar que para el mercado de los microcontroladores se tiene tres principales subpartidas:

a. Sistemas Microcontroladores Embebidos
b. Sistemas Microcontroladores con apoyo externo
c. Sistemas Microcontroladores DSP

Hay absolutamente una amplia gama de dispositivos (autónomos) incorporados disponibles. Un microcontrolador tiene integrado todos los recursos necesarios, tales como señal de reloj (clock), Reset, pines de entrada y salida (denominado E/S): disponibles en un chip de muy bajo costo. En el circuito de aplicación, no se tiene que ofrecer mucha potencia (debido a que utilizan un par de pilas AA). Mientras que el software para el procesador del ordenador incorporado en el microcontrolador se almacena en una memoria no volátil (siempre disponible), que también está integrado en el chip.

Los sistemas de microcontroladores embebidos se han convertido en el nuevo estándar para realizar diferentes aplicaciones. Cuando nos fijamos en algunos de los microcontroladores más potentes, es posible confundirlo con los microprocesadores pese a la diferencia entre ellos. Hay un número de chips que se llama "microcontroladores" (con datos típicamente de 32 bits de direcciones y rutas) que requieren memoria externa y circuitos de interfaz añadido a ellos para que puedan ser utilizados en diferentes aplicaciones.

Estos chips son típicamente llamados microcontroladores porque tienen algunas de las características incorporadas de los microcontroladores embebidos, tales como un generador de reloj o de la interfaz de serie, o porque se han incorporado en los circuitos de interfaz para tipos específicos de memoria. Los microcontroladores tienden a requerir circuitos de soporte para registros y pueden tener una gama muy amplia de dispositivos de interfaz y de memoria externos conectados a ellos.

2.3. Hardware interno.

Si procedemos a quitar el embalaje de plástico (llamado encapsulado) alrededor del microcontrolador, podemos visualizar internamente el chip, es decir que veríamos un rectángulo de silicio, semejante al mostrado por la figura 2.1.

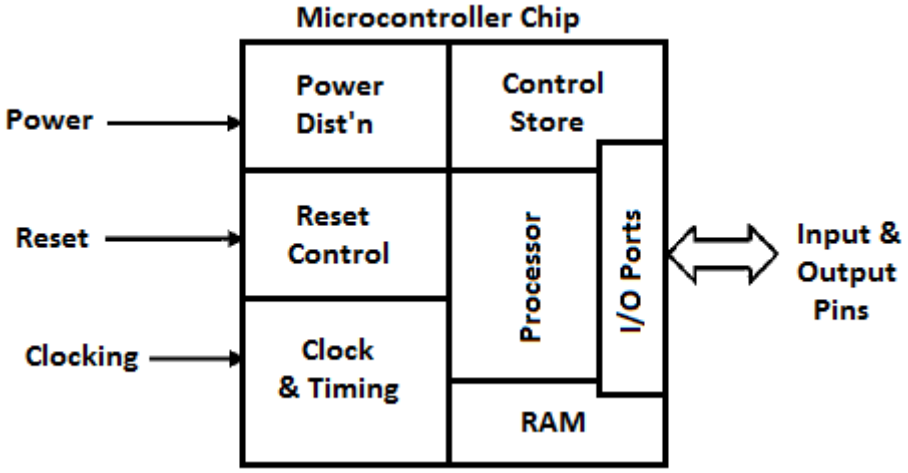

Figura 2. 1: Diagrama de bloques con características básicas que tienen internamente un microcontrolador.
Fuente: Rossano, V. (2013)

Con cada una de las funciones previstas dentro del chip siendo visible diferentes circuitos circundantes, que junto con la circuitería básica presentada en el diagrama de bloques (véase la figura 2.1) la mayoría de los microcontroladores modernos tienen diferentes características incorporadas en los chips:

a. Circuitos de programación (disponible en el encendido) de memoria del programa no volátiles.
b. Capacidad de interrupción (a partir de una diversidad de fuentes).
c. Entradas y salidas analógicas (E/S), tanto PWM y corriente continua variable (CC) de E/S.
d. Interfaz serial de E/S (transferencias de datos síncrono y asíncrono).
e. Interfaces de bus/memoria externa (para memorias RAM y ROM).

Todas estas características aumentan la flexibilidad del dispositivo considerablemente y no sólo hacer que el desarrollo de todas las aplicaciones sea más fácil, pero permiten la creación de aplicaciones que podrían no ser posibles de otra manera. La mayoría de estas opciones permiten mejorar la función de los diferentes pines de E/S y no afectan su funcionamiento básico, y por lo general puede deshabilitarse (disabled), restauración de los pines de E/S.

La mayoría de los dispositivos modernos están fabricados utilizando tecnología CMOS, lo que disminuye el tamaño del chip actual y los requisitos de energía considerablemente en relación a los primeros dispositivos con

tecnologías NMOS o HexMOS. Para la mayoría de los microcontroladores modernos, la corriente requerida es desde unos pocos microamperios (UA) en el modo de suspensión a hasta aproximadamente 1 miliamperios (mA), para que el microcontrolador funcione a 20 MHz.

Un chip de tamaño más pequeño significa que junto con menos potencia que se requiera para el chip, más chips pueden ser construidos en una sola oblea. Es decir, que mientras más chips se construyan sobre una oblea, menor será su precio unitario.

Hay que tener en cuenta que la circuitería CMOS, la alimentación positiva viene etiquetada como "V_{dd}" y la negativa o tierra como "V_{ss}". Estas corresponden a los TTL cuyas conexiones son "V_{cc}" y "Gnd". Las velocidades máximas para los dispositivos típicamente son de pocas decenas de Megahertz (MHz), con el principal factor que limita el tiempo de acceso de la memoria incorporada en los chips.

De los ciclos de ejecución y el retardo de las rutinas limitan la capacidad de los microcontroladores (MCU) en procesar complejas formas de onda de entrada y salida. Posteriormente, en el presente trabajo de titulación se describirá las características del hardware de los microcontroladores PIC avanzados que proporcionan funciones de interfaz, así como algoritmos "bit-banging" para simular interfaces al mismo tiempo dejando suficientes ciclos de procesamiento.

A pesar de las ventajas que tiene un microcontrolador diseñado para almacenar programas y memoria RAM interna variable, hay ocasiones (depende de aplicaciones) donde se requiere agregar memoria externa al microcontrolador. Existen tres formas básicas de hacer esto:

a. El primer método, es añadir dispositivos de memoria al microcontrolador como si se tratara de un microprocesador. Muchos microcontroladores están diseñados con hardware incorporado para acceder a los dispositivos externos como un microprocesador (con los circuitos de interfaz de memoria añadido al chip, véase la figura 2.2) con el ejemplo clásico de esto es el Intel 8051.

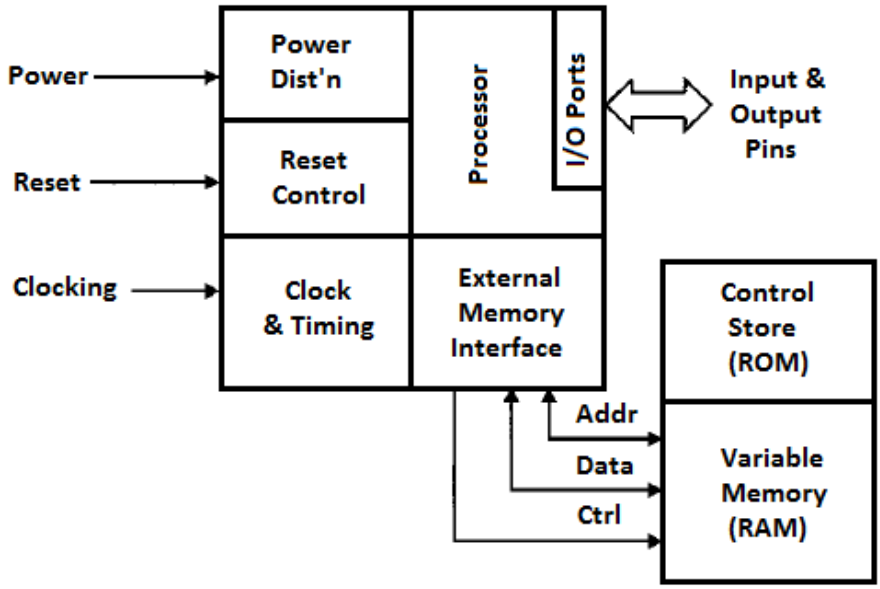

Figura 2. 2: Diagrama de bloques del microcontrolador PIC con circuito integrado para acceder a los dispositivos de memorias externas.
Fuente: Verle, M. (2010)

Una aplicación típica de un microcontrolador con memoria externa, es como una caché de disco duro o memoria intermedia que distribuyen

grandes cantidades de datos. Los buses de datos del microprocesador (uP) 8051 permite la adición de hasta 64 K, además de 64K de memoria RAM variable. Una característica interesante del 8051 es que la memoria no volátil interna puede deshabilitarse, permitiendo al 8051 ser utilizado inclusive si la programación ha sido desarrollada con programas de nivel inferior.

b. El segundo método de la adición de memoria externa, es simular operaciones de buses de datos del microprocesador con pines de E/S en el chip. Este método tiende a ser mucho más lento que el que tiene un microcontrolador que se puede acceder a los dispositivos externos directamente, como el 8051. Si bien no se recomienda simular un bus de microprocesador para dispositivos de memoria, no es raro ver a un microcontrolador simulando un bus de microprocesador para permitir el acceso a un chip con periféricos de E/S especiales.

c. El tercer y último método es utilizar un protocolo de bus que ha sido diseñado para proporcionar memoria adicional y capacidades de E/S a los microcontroladores. El protocolo de dos hilos I^2C, es un estándar de bus muy utilizada que proporciona esta capacidad. Este estándar permite a los dispositivos de E/S y a múltiples microcontroladores comunicarse entre sí, sin protocolos de bus complejos.

2.4. Aplicaciones de los microcontroladores.

En el presente trabajo de titulación, se utilizará el término "aplicación" para describir colectivamente los circuitos de hardware y software necesarios para desarrollar circuitos basados en microcontroladores PIC. Se debe considerar que para desarrollar proyectos mediante microcontroladores nos basaremos en múltiples actividades de desarrollo (por los circuitos y software) y no el resultado de una sola disciplina. Se presentará los cinco elementos de un proyecto con microcontroladores y explicar algunos de los términos y conceptos relacionados con ellos. Estos cinco elementos son:

- Microcontroladores y circuitos de soporte.
- Proyección de alimentación.
- Software de aplicación.
- Interfaz de Usuario (UI).
- Dispositivos de entrada/salida (E/S).

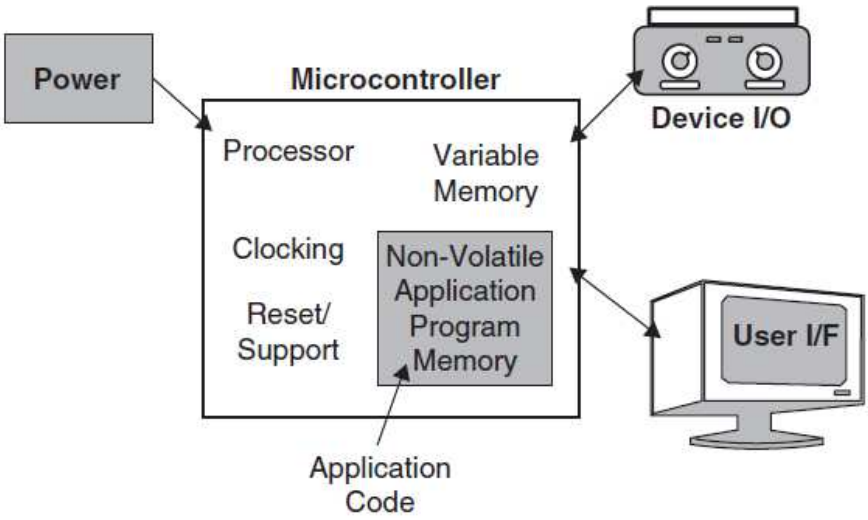

Figura 2. 3: Diagrama de bloques de aplicaciones de microcontroladores embebidos que muestra los cinco aspectos para el desarrollo de proyectos.
Fuente: Verle, M. (2010)

Estos elementos se muestran trabajando juntos en la figura 2.3, el microcontrolador con sus características internas (procesador, reloj, memoria, etc.) es simplemente el chip más completo. Aparte del propio chip, existen circuitos microcontroladores que sólo requieren de alimentación junto con un condensador de desacoplamiento y a menudo un circuito de reposición y un oscilador para funcionar.

El diseño de los microcontroladores (MCU) PIC (como ocurre con la mayoría de los microcontroladores) hace que la especificación de potencia y partes externas casi triviales; las posibilidades son, aparte de potencia y un condensador de desacoplamiento, no se requerirá ninguna otra parte para apoyar el microcontrolador integrado en cualquier aplicación.

Lo más recientes MCU PIC (así como los chips de otros fabricantes) ahora son capaces de correr dentro de una sorprendente gama amplia de tensiones (entre 2 y 6 voltios), que le permitirá utilizar pilas alcalinas simples y prescindir de reguladores de voltaje para la mayoría de aplicaciones.

Un condensador de desacoplamiento (normalmente 0.01uF a 0,1uF conectado a través de alimentación Vdd y tierra Vss) siempre debe estar conectada a la conexión de alimentación de cada chip en la circuitería diseñada para diferentes aplicaciones, con un pin lo más cercano como sea posible al pin de entrada de alimentación positiva.

Los condensadores de desacoplamiento se utilizan para minimizar los efectos sobre los chips con relación a los rápidos cambios en los niveles de potencia y disponibilidad actual causados por otros chips en la conmutación de circuito. Un condensador de desacoplamiento puede ser pensado como un filtro que suaviza las partes ásperas de la fuente de alimentación y proporciona corriente adicional para situaciones de alta carga en la pieza. Es importante el uso de un condensador de desacoplamiento sobre el MCU PIC y nunca debe ser dejado fuera en cualquier aplicación desarrollada.

2.5. Arquitecturas del procesador.

Existen una variedad de puntos fuertes para soportar las opciones que están disponibles en arquitecturas de computadora. Mientras que RISC está de moda en estos momentos, muchas personas sienten que CISC ha sido calumniado injustamente. Esto también es válido para los defensores de las arquitecturas de computadoras Harvard sobre "Princeton" y si las instrucciones del procesador debería ser difícil de ser codificado o microcodificado.

En las siguientes secciones se describirá algunos antecedentes sobre los distintos tipos de procesadores, y la explicación de las ventajas y desventajas de características, y poder entender por qué los ingenieros hicieron algunas decisiones sobre los demás cuando la especificación y diseño de procesador de un microcontrolador. Ellos no tienen el propósito de proporcionar una comprensión completa del diseño de la arquitectura de

procesadores de computadoras, sino que debe ayudar a explicar los conceptos detrás de las palabras de moda usados en materiales de marketing de los microcontroladores.

2.5.1. CISC versus RISC

Muchos procesadores se llaman RISC (computadoras con conjunto de instrucciones reducidas, que se pronuncia "risk"), ya que hay una percepción de que RISC (véase la figura 2.4) es más rápido que CISC (computadoras con conjunto de instrucciones complejas) porque las instrucciones que ejecutan son pequeñas y adaptada a las tareas específicas requeridas por la aplicación.

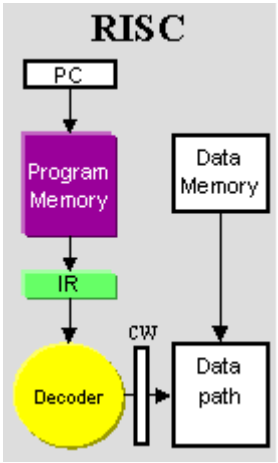

Figura 2. 4: Arquitectura RISC.
Fuente: Bren, D. (2014)

Las instrucciones CISC tienden a ser grandes y realizan funciones que el diseñador del procesador cree que será el más adecuado para ser utilizado en determinadas aplicaciones. Al elegir un microcontrolador para una aplicación específica, se le dará la opción de los procesadores RISC, RISC-

like y CISC. No hay una respuesta correcta definitiva a la pregunta de cuál es mejor. Existen aplicaciones en las que una de las metodologías de diseño es más eficiente. Un procesador RISC bien diseñado tiene un pequeño conjunto de instrucciones, que puede ser muy fácil de memorizar.

Un conjunto de instrucciones CISC ofrece funciones de alto nivel que son fáciles de implementar y no requieren que el programador este íntimamente familiarizado con la arquitectura del procesador. En cuanto a los compiladores de lenguajes de alto nivel, hay igualmente sofisticadas herramientas disponibles en el mercado, ambos permiten aplicaciones complejas.

Para los nuevos programadores, un procesador CISC (véase la figura 2.5) será más fácil por su código, pero para un programador experimentado, un procesador RISC en realidad es más fácil en crear códigos complejos. Los partidarios de las metodologías empujarán diferentes ventajas, pero cuando llegas a fin de cuentas, no es sustancialmente mejor que el otro.

En lo personal, elegiría un procesador RISC con la posibilidad de acceder a todos los registros en una sola instrucción. Esta capacidad para acceder a todos los registros en el procesador como si fueran la misma, conocida como la ortogonalidad y proporciona algunas capacidades inesperadamente potentes y flexibles a las aplicaciones.

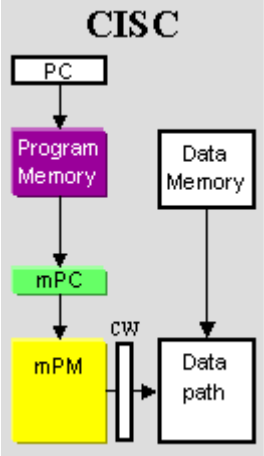

Figura 2. 5: Arquitectura CISC.
Fuente: Bren, D. (2014)

Los procesadores de los microcontroladores PIC son ortogonales, y que posteriormente se describirán en las siguientes secciones la arquitectura PIC, las instrucciones y solicitudes, se verá que las operaciones de procesamiento rápido de datos dentro del procesador puede ser muy fácil de implementar en un sorprendentemente pequeño conjunto de instrucciones.

2.5.2. HARVARD versus PRINCETON.

En la década de 1940, el gobierno de Estados Unidos pidió a las Universidades de Harvard y Princeton llegar a una arquitectura de computadores, para ser utilizado en el cálculo de las tablas de distancias de proyectiles de artillería naval para diferentes elevaciones y condiciones ambientales.

La respuesta de Princeton, era que una computadora tenía memoria común para almacenar el programa de control, así como variables, y otras

estructuras de datos. Fue mejor conocido por el nombre del jefe científico, John Von Neumann. En la Figura 2.6 se muestra el diagrama de bloques de la arquitectura Princeton.

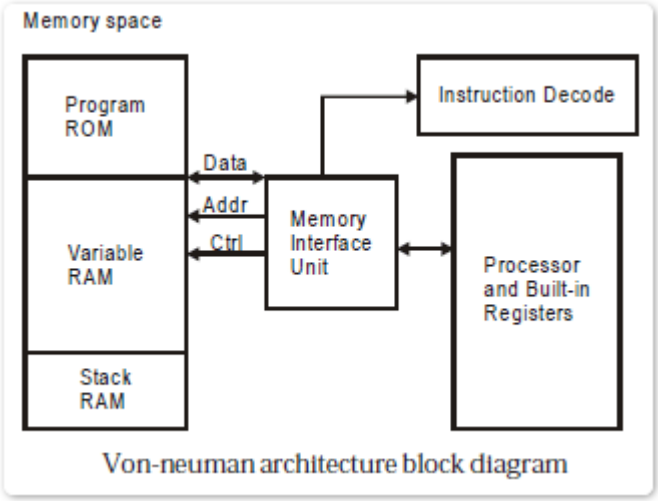

Figura 2. 6: Diagrama de bloques de Arquitectura Princeton o Von Neumann.
Fuente: http://electronicsnewsline.com/388/memory-architecture-microcontrollers.html

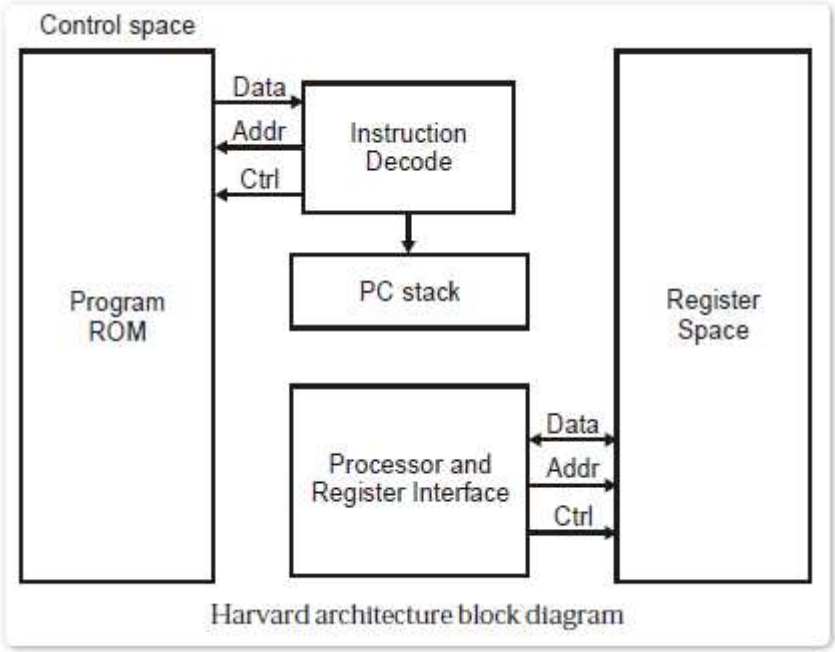

Figura 2. 7: Diagrama de bloques de Arquitectura Harvard.
Fuente: http://electronicsnewsline.com/388/memory-architecture-microcontrollers.html

En contraste, la respuesta de Harvard era un diseño (véase la figura 2.7) que utiliza bancos de memoria separados para almacenamiento de programas, la pila de procesador y memoria RAM variable. La arquitectura Princeton ganó la competencia, ya que se adaptaba de mejor manera a la tecnología de la época; un único espacio de memoria era preferible a causa de la falta de fiabilidad de la electrónica actual (antes de que los transistores fueran inventados) y la interfaz más simple sería tener menos piezas que podían fallar.

La unidad de interfaz de memoria de la arquitectura de Princeton es responsable de arbitrar el acceso al espacio de memoria entre Las instrucciones de lectura y pasar datos de ida y vuelta al procesador. Este hardware es algo así como un cuello de botella entre el hardware de procesamiento de instrucciones del procesador y el hardware de memoria para acceder.

En muchos procesadores con arquitectura de Princeton, la demora se reduce porque la mayor parte del tiempo necesario ejecuta una instrucción normalmente para buscar la siguiente instrucción (esto se conoce como búsqueda previa). Otros procesadores (más notablemente el procesador Pentium en su PC) tienen programas y datos cachés separadas, en donde pasan los datos directamente a la zona apropiada del procesador mientras se está llevando a cabo el acceso de la memoria externa.

La arquitectura Harvard fue ignorada hasta finales del año 1970, cuando los fabricantes de microcontroladores se habían dado cuenta de que la arquitectura no tenía el cuello de botella tanto de las instrucciones como datos de los ordenadores basados en la arquitectura de Princeton.

Las rutas de datos duales ofrecen computadoras con arquitectura “Harvard” con la capacidad de ejecutar instrucciones en un menor número de ciclos de instrucción que la arquitectura Princeton, debido al posible paralelismo de la instrucción en la arquitectura Harvard. El paralelismo quiere decir que obtiene instrucciones que pueden ocurrir durante la ejecución instrucciones anteriores y no espera a que sea un ciclo muerto en la ejecución de la instrucción o detener el funcionamiento del procesador, mientras que la siguiente instrucción está siendo rebuscada.

Después de leer esta descripción de cómo se transfieren los datos en las dos arquitecturas, es probable que sienta que un microcontrolador con arquitectura Harvard, es el único camino a seguir. Pero la arquitectura Harvard carece de la flexibilidad de la Princeton en el software requerido para algunas aplicaciones que normalmente se encuentran en los sistemas de gama alta, tales como servidores y estaciones de trabajo.

La arquitectura Harvard, es realmente mejor para los procesadores que no procesan grandes cantidades de memoria a partir de diferentes fuentes (que es mejor utilizar la arquitectura Von Neumann) y tienen que acceder a

esta pequeña cantidad de memoria muy rápidamente. Esta característica de la arquitectura Harvard (utilizado en el procesador del microcontrolador PIC) hace que sea muy adecuado para aplicaciones de microcontroladores.

2.5.3. Procesadores Microcoded versus Hardwired.

Una vez que la arquitectura del procesador se ha decidido, el diseño de la arquitectura va a los ingenieros responsables de diseñar aplicaciones sobre silicio. La mayoría de estos detalles se quedan debajo de las sábanas y no afectan cómo realizar las interfaces de diseño de aplicaciones. Hay un detalle que puede tener un gran efecto, es cómo se ejecutan las aplicaciones y si el procesador es un dispositivo Hardwired (cableado) o Microcoded (microcodificado).

Para la decisión de ambos tipos de implementaciones del procesador, se tiene implicaciones significativas en cuanto a la facilidad de diseño del procesador, cuando está disponible, y su capacidad para atrapar y corregir errores. Cada instrucción del procesador es de hecho una serie de instrucciones que se ejecutan para llevar a cabo la instrucción básica más grande. Por ejemplo, para cargar el acumulador en un procesador deben tomarse los siguientes pasos:

a. Dirección de salida en la instrucción a los conductores de autobuses de direcciones de memoria de datos.
b. Configurar valores del bus interno de la memoria de datos que se almacena en el acumulador.

c. Habilitar lectura de bus.

d. Comparar valores de datos leídos de la memoria a cero o cualquier otra condición importante y bits puestos en el registro STATUS.

e. Desactivar lectura de bus.

Cada uno de estos pasos debe ser ejecutado con el fin de llevar a cabo la función de instrucción básica. Para ejecutar estos pasos, el procesador está diseñado para buscar una serie de instrucciones de una memoria o ejecutar un conjunto de funciones lógicas únicas para las instrucciones.

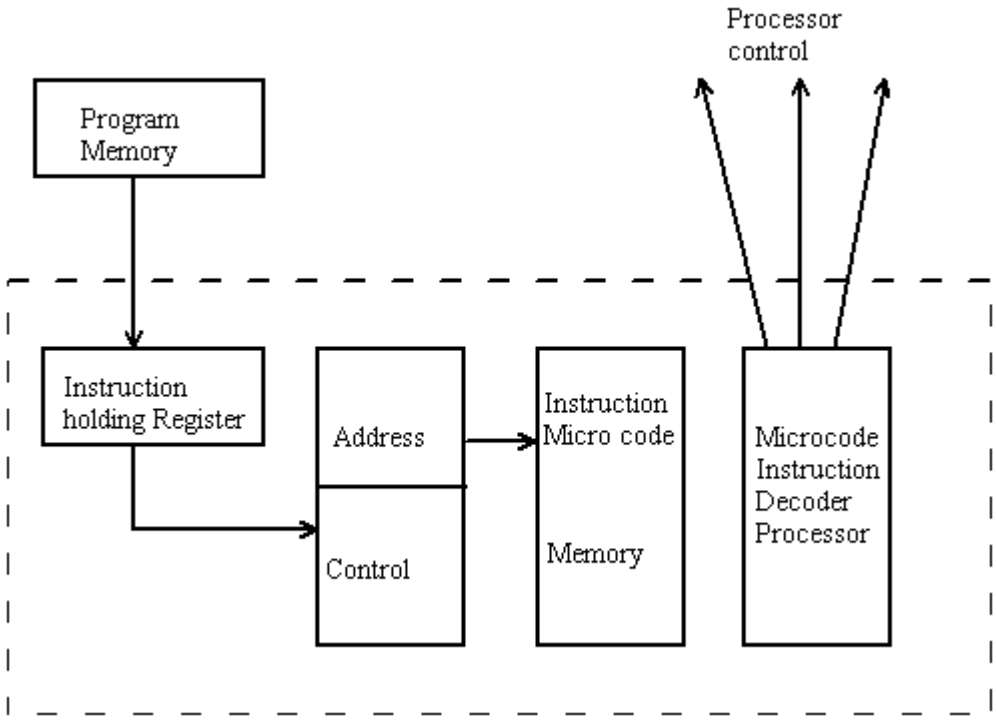

Figura 2. 8: Procesador Microcoded con memoria.
Fuente: http://nptel.ac.in/courses/Webcourse-contents/IIT-KANPUR/microcontrollers/introduction.htm

Un procesador Microcoded es realmente un procesador dentro de un procesador. En un procesador Microcoded, una máquina de estado ejecuta

cada instrucción como la dirección a una subrutina de instrucciones. Cuando una instrucción se carga en el registro de la explotación de instrucciones, ciertos bits de la instrucción se utilizan para señalar el comienzo de la rutina de instrucciones (o microcódigo), la decodificación de instrucciones y la lógica del procesador uCode ejecuta las instrucciones de microcódigo hasta que encuentra una instrucción, tal como se muestra en la figura 2.8.

En algunos procesadores, la memoria de programa es de sólo 8 bits de ancho, aunque la instrucción completa puede ser un múltiplo de este (en los 8051 la mayoría de las instrucciones son de 16 bits de ancho). En este caso, la memoria de programa tiene lugar a varias lecturas para cargar el registro de instrucción de espera antes de que la instrucción pueda ser ejecutada.

El ancho de la memoria de programa y la velocidad con la que sostiene el registro de instrucción que se puede cargar, es un factor en la velocidad de ejecución del procesador. En procesadores con arquitectura Harvard, como el PIC, la memoria de programa es la anchura de la palabra de instrucción y el registro de instrucción que sostiene podrá cargarse en un ciclo. En la mayoría de los procesadores con arquitectura Princeton, que tienen un bus de datos de 8 bits, el registro de retención de instrucciones se carga a través de múltiples lecturas de datos.

Un procesador Hardwired, utiliza el patrón de bits de la instrucción para acceder a puertas lógicas específicas (posiblemente a la única instrucción)

que se ejecutan como un circuito combinatorio para llevar a cabo la instrucción. En la Figura 2.9 se muestra cómo la instrucción cargada en el registro de retención de instrucciones para iniciar una porción específica de la lógica de ejecución que lleva a cabo todas las funciones de la instrucción.

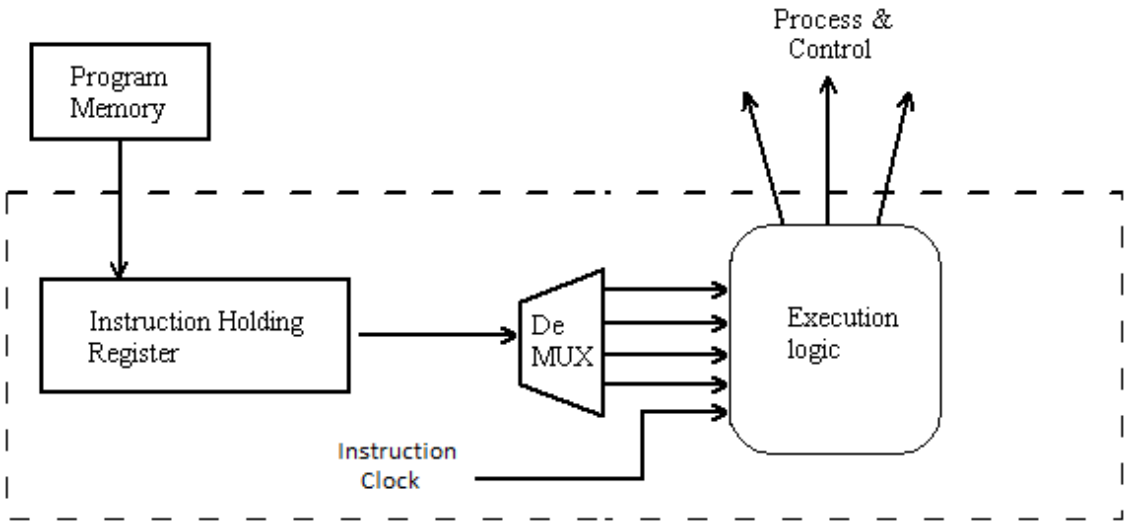

Figura 2. 9: El procesador Hardwired.
Fuente: http://nptel.ac.in/courses/Webcourse-contents/IIT-KANPUR/microcontrollers/introduction.htm

Cada uno de los dos métodos ofrece ventajas sobre el otro. Un procesador Microcoded, es generalmente más simple que un procesador Hardwired para el diseño y se puede implementar más rápido con menos posibilidades de tener problemas en condiciones específicas.

Un ejemplo de los cambios rápidos y fáciles que permite el procesador Microcoded fue realizado hace varios años cuando IBM quería tener un microprocesador que podía correr 370 instrucciones en lenguaje ensamblador. Antes de que IBM comenzará a diseñar su propio microprocesador, miraron a su alrededor en diseños existentes y se dieron

cuenta de que el Motorola 68000 tenía la misma arquitectura de hardware que el 370 (aunque las instrucciones eran completamente diferentes). Por eso IBM terminó pagando a Motorola para reescribir el microcódigo para el 68000 y se acercó con un nuevo microprocesador que era capaz de correr 370 instrucciones mucho más rápido y en una pequeña fracción del costo en desarrollar ese nuevo chip.

Un procesador Hardwired es generalmente mucho más compleja en Hardware porque las mismas funciones tienen que repetirse una y otra vez, ¿cuántas veces un registro lee o escribe una función que debe ser repetida para cada tipo de instrucción?, esto significa que el diseño del procesador probablemente será más difícil de depurar y menos flexible que un Microcoded, pero las instrucciones se ejecutarán en un menor número de ciclos de reloj.

El resultado será un punto que probablemente no son conscientes, en la mayoría de los procesadores, cada instrucción se ejecuta en un número determinado de ciclos de reloj. Este número determinado de ciclos de reloj se conoce como ciclo de instrucción del procesador. Cada ciclo de instrucción en la familia de los dispositivos microcontroladores PIC toma cuatro ciclos de reloj. Esto significa que un MCU PIC funcionando a 4 MHz, ejecutará las instrucciones a una tasa de 1 millón de instrucciones por segundo.

Utilizando un procesador Hardwired por encima del Microcoded otra vez puede dar lugar a algunas mejoras significativas de rendimiento. Por ejemplo, el 8051 fue diseñado para ejecutar una instrucción en 12 ciclos. Este gran número de ciclos requiere un reloj de 12 MHz para ejecutar el código a una velocidad de 1 MIPS (millones de instrucciones por segundo) mientras que un microcontrolador PIC con un reloj de 4 MHz consigue el mismo rendimiento.

2.6. Comunicaciones de Microcontroladores

La capacidad de los microcontroladores para comunicarse con otros dispositivos se ha convertido en los últimos años en muy importante. Hace algún tiempo atrás, la comunicación se pensaba más como una ocurrencia tardía. Con el despliegue de Internet, el número de aplicaciones que requieren los microcontroladores para poder comunicarse con otros dispositivos han crecido de manera significativa.

El propósito de esta sección, es que haremos énfasis en la habilitación de la comunicación de los microcontroladores PIC con otros dispositivos, tanto de forma directa en la comunicación punto a punto y en entornos de red. El término "comunicación punto a punto" describe la conexión de un microcontrolador con dispositivos que se conocen sus direcciones. El término puede parecer confuso, ya que abarca múltiples dispositivos, así como las vías de comunicación que enlazan dos dispositivos juntos.

Algunas memorias y los periféricos “chips” pueden ser accesibles a un microcontrolador, utilizando técnicas de comunicación punto a punto, aun cuando pudieran ser removidos durante el funcionamiento de la aplicación (como un dispositivo en una red) debido a que sus direcciones se mantienen constantes.

En las comunicaciones de red, las interfaces del hardware y software permiten cambiar las direcciones, incluso en términos prácticos el mismo hardware (y las mismas direcciones) está disponible mientras funcione la aplicación. Una característica importante de los dispositivos de red es que generalmente operan incluso cuando la red no está disponible para el dispositivo. Esta diferenciación entre la comunicación punto a punto y la comunicación de red puede parecer sutil, pero como se verá en las siguientes secciones, hay diferencias significativas en la forma en que los esquemas de comunicación son implementados.

2.6.1. Comunicación punto a punto.

La comunicación punto a punto entre dos dispositivos en una determinada aplicación se implementa normalmente mediante conexiones en serie para proporcionar una capacidad de transferencia de datos básicos. Muchos de los nuevos desarrolladores conectan los dispositivos en paralelo (bits transferidos simultáneamente en conexiones individuales) con un bus lleno (dirección, datos y líneas de control), ya que esto es algo que se sienta más cómodo.

Los microcontroladores embebidos no tienen el número de pines disponibles de un típico microprocesador. Ellos tienen sólo unos pines disponibles, lo que resultaría en la comunicación punto a punto, que está implementado mediante un formato de datos en serie. Los protocolos de serie que se utilizan son generalmente fáciles de implementar, aunque puede haber algunos trucos para su codificación.

En esta sección, se presentará los protocolos de transferencia de datos en serie muy utilizados en la comunicación punto a punto. Las aplicaciones que tienen varios dispositivos que se comunican con un microcontrolador no se limitan a sólo dos chips; hay muchas aplicaciones con múltiples chips conectados a un MCU central. Una manera obvia de conectar estos dispositivos, es el de proporcionar una conexión individual desde el microcontrolador tal y como se muestra en la figura 2.10.

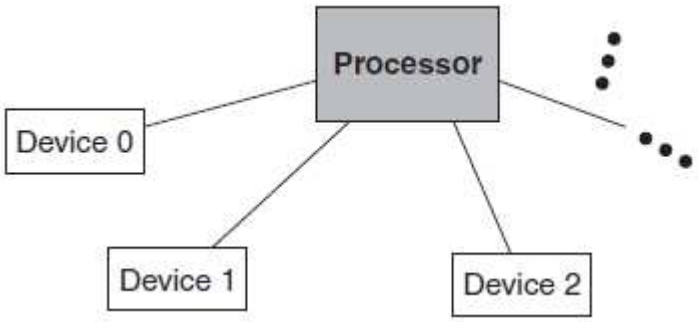

Figura 2. 10: Múltiples dispositivos conectados a un microcontrolador.
Fuente: Pérez, E. M. (2007),

Este método, obviamente, se puede utilizar solamente cuando hay suficiente pines de E/S integrados en el microcontrolador para permitir conexiones entre cada dispositivo. Si no hay suficientes pines, una conexión

similar a la figura 2.11 tendrá que ser implementado usando un método de comunicación punto a punto, que permite conectar a múltiples dispositivos entre sí y no interferir con el funcionamiento de cada uno.

También existe una serie de protocolos síncronos seriales para comunicación (tales como I^2C y Microwire) que permiten a los dispositivos de comunicación punto a punto conectarse en el formato de bus común.

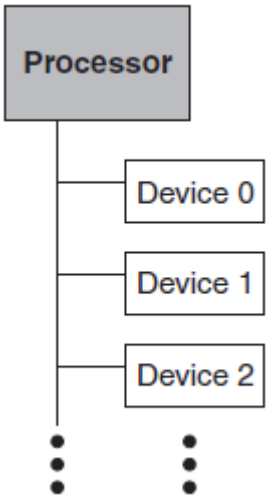

Figura 2. 11: Esquemático de un bus de comunicación para conexión de múltiples dispositivos a un solo microcontrolador.
Fuente: Pérez, E. M. (2007),

La comunicación síncrona serial es el método más básico de la transferencia de datos en serie y requiere de dos líneas, una para la transferencia de datos y otra para los datos de reloj para indicar cuando la transferencia se llevará a cabo. Una muestra de transferencia de datos de 8 bits se muestra en la figura 2.12 con la línea pulsante de reloj cuando los bits de datos son válidos.

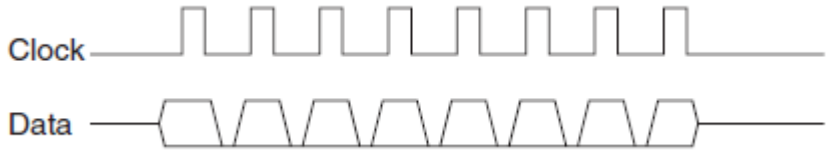

Figura 2. 12: Sincronismo serial de datos utiliza un reloj para indicar cuando los bits de datos son válidos.
Fuente: Pérez, E. M. (2007),

Siempre hay un dispositivo maestro que inicia la transferencia de datos y proporciona el reloj para la transferencia de datos (incluso si reciben datos desde otro dispositivo). El dispositivo esclavo espera la señal de reloj para recibir o enviar datos. Mientras que el término "reloj" se utiliza normalmente para indicar una señal que se produce en un intervalo regular, el reloj en las comunicaciones síncronas seriales es considerablemente más flexible.

Al configurar las comunicaciones síncronas seriales entre dispositivos, debemos tener en cuenta una serie de cosas. La primera es que si viene un “bit” en la transferencia de datos en serie, ¿será el bit menos significativo o el bit más significativo? A continuación, hay que ser conscientes de que la transferencia se llevó a cabo.

Antes de comenzar a desarrollar aplicaciones, asegúrese de haber leído las diferentes hojas de datos de dispositivos y entender cómo funcionan las transferencias de datos. Las líneas de datos individuales pueden ser utilizadas para comunicaciones de datos y hay una serie de protocolos para el envío de datos sin una línea de reloj separada. En todos estos protocolos, el tiempo de

los bits de datos debe ser conocida y el receptor debe ser capaz de determinar cuándo un bit está llegando y si es válido. El funcionamiento de un único protocolo de comunicaciones de línea de datos, sería el no retorno a cero (NRZ) que se muestra en la figura 2.13.

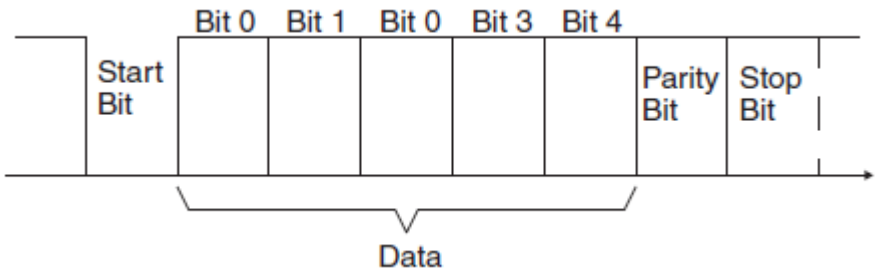

Figura 2. 13: Bits de datos NRZ con un período constante para ser enviados.
Fuente: Pérez, E. M. (2007),

Cada bit es una cantidad constante de tiempo, con el bit de inicio se utiliza para indicar al receptor que los datos que vienen con los siguientes bits de datos (primero el bit menos significativo). Después que los datos se han enviado, hay un solo bit de paridad, que es un simple código de detección de error, y un bit de parada.

El receptor continuamente lee la línea de datos y cuando la señal pasa a nivel bajo, se determina la media de los bits y luego espera un período completa antes de leer los datos. Los receptores NRZ de hardware integrado en los microcontroladores, son muy comunes y proporcionan la detección del bit de inicio y el bit de sondeo de forma automática, lo que permite al procesador leer el valor de los datos de un registro. El receptor debe estar

provisto con el período de cada bit de tal manera que pueda decodificarlas correctamente el paquete de datos entrante.

Antes de pasar a algunos de los otros protocolos de comunicación punto a punto de una sola línea, hay algunas cosas que usted debe tener en cuenta acerca de los paquetes de datos NRZ. En primer lugar, es el protocolo de datos más utilizado para RS-232 o comunicaciones asíncronas seriales utilizados en su PC; nótese que no es el protocolo eléctrico, ya que esto no será lo que usted espera y requerirá de algunos circuitos especializados para su implementación.

Cuando se utiliza el PC (en la mayoría de aplicaciones modernas) los paquetes de datos se describen como "8-N-1", lo que significa que hay 8 bits de datos, sin paridad y 1 bit de parada. El bit de paridad, como se indicó anteriormente, es un simple bit de detección de errores y cuando se utiliza indicará si la suma de los otros bits es par o impar.

Rara vez se usa los protocolos de transmisión de datos modernos porque proporcionan funciones de detección y corrección de errores más elaborados. El bit de parada es un poco de "aire muerto" en la que el receptor puede procesar el byte de entrada y el transmisor puede preparar el siguiente. Finalmente, el paquete de datos NRZ se puede utilizar como bus común de comunicación punto a punto, utilizando controladores de colector abierto.

El esquema de codificación de datos Manchester no utiliza un nivel de tensión para indicar un valor de bit, pero en su lugar utiliza la dirección de la transición de la línea de datos entrante. Como NRZ, la codificación Manchester tiene un período de bits constante (véase la figura 2.14), pero en el medio de los bits (línea discontinua) siempre hay un cambio de nivel y, dependiendo de la aplicación, un bajo a alto podría significar un “0” y un alto a bajo podría significar un “1”.

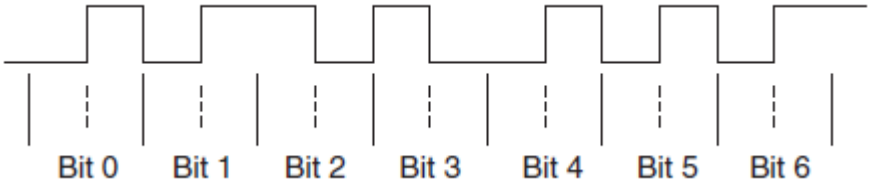

Figura 2. 14: Codificación Manchester con período de bits constante.
Fuente: Pérez, E. M. (2007),

La necesidad de tener siempre una transición puede hacer que un flujo de datos en codificación Manchester sea difícil de interpretar cuando se los observe (aunque parece que sea no intuitivo). Al pasar de uno a otro valor de bit, no hay una transición en los límites de bits, y cuando dos bits son del mismo valor, hay una transición en el punto entre los dos bits.

Por lo general, la codificación Manchester se utiliza a menudo en los protocolos de red donde reciben señales sincrónicas entrantes usando un bucle de enganche de fase y la transición de nivel, se utiliza para alternar en un valor de bit. Aunque esto suena complejo, los niveles lógicos cambiantes

son muy fáciles de implementar en hardware y no requieren ningún recurso de sincronización por parte del procesador de recepción.

El último método proporciona una comunicación serial punto a punto en una sola línea, cuyo formato de datos codificada por impulsos se muestra en la figura 2.15 en que los datos se indica por la longitud de tiempo que una señal está activa. Considerando que los valores bit NRZ se determinan como niveles lógicos en un momento específico y valores de bit de Manchester son cambios de la lógica en un momento específico, la longitud de tiempo que una señal de impulso codificado varía junto con todo el paquete de datos es el bit en.

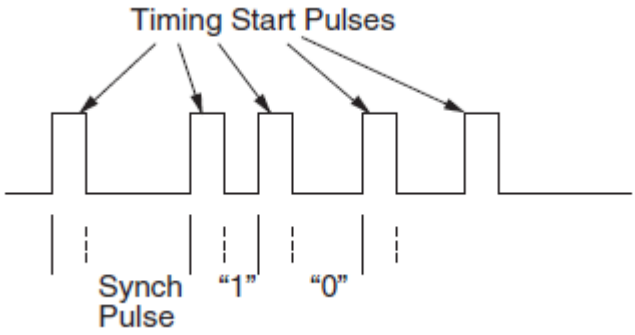

Figura 2. 15: La longitud de tiempo que indica el valor de bits de la señal activada.
Fuente: Pérez, E. M. (2007),

Este cambio en la temporización de paquetes hace que los datos de impulsos codificados sean difíciles para diseñar circuitos lógicos tradicionales que lean datos entrantes, y generalmente el código deba ser escrito para leer datos entrantes. Estos dos atributos hacen que la codificación de impulsos de datos haga una transmisión ineficiente.

Por esta razón, los datos codificados por impulsos sólo se utilizan para pequeñas cantidades de información que transmitan de vez en cuando. Una aplicación popular para datos codificada por impulsos es los controles de TV, muy utilizados para controlar robots y otras aplicaciones de microcontroladores.

2.6.2. Redes de comunicación.

Ha sido solamente en los últimos años que los microcontroladores se han considerado los dispositivos de red legítimos. La caída de los costos en los microcontroladores poderosos y chips de interfaz de red para aplicaciones tanto cableado e inalámbrico ha tenido mucho que ver con el auge de las aplicaciones de red basadas en microcontroladores.

Cuando se observan las redes inalámbricas, los protocolos Bluetooth y ZigBee también deben ser considerados junto con WiFi (802.11) como medios potenciales de la red. Del mismo modo, las redes Ethernet (por lo general en los hogares) permiten la adición de sensores en red y dispositivos de control en toda la casa.

La comprensión de cómo se coloca una red cableada hacia fuera y le informará mucho acerca de las distintas aplicaciones que se ejecutan en él y de sus características. En lugar de utilizar el término "diseño", los informáticos utilizan el término "topología" para describir cómo se organiza una red y cómo los diferentes dispositivos (por lo general a que se refiere como "nodos") están

cableados entre sí. Como una regla simple, más conexiones de nodos tienen con otros nodos de la red, el rendimiento de la red será más alto (y una mayor fiabilidad).

Múltiples redes de diferentes tipos pueden conectarse entre sí usando los nodos conocidos como puentes, que tienen interfaces de red para los diferentes tipos de red. De hecho, Internet es en realidad un conjunto de redes que han sido conectados en red en diferentes puntos. Las topologías de red se muestran en esta sección son realmente para su edificación y para que se familiaricen con algunos de los términos que se presentarán más adelante, cuando se discuta la creación de redes. Comprender plenamente las características de las topologías de red que realmente está en el campo de los científicos de la computación que están diseñando redes para aplicaciones específicas.

En la figura 2.16 muestra la red prototipo, el único bus o red de medios en el que una única conexión se utiliza para conectar todos los nodos de la red. Mientras que el "bus" es el término más comúnmente aceptado, se recomienda el uso del término "single medio" porque es más descriptivo y evita la confusión con la comunicación punto a punto con un autobús. También, se prefiere visualizar el diagrama en la parte inferior de la figura 2.16, que se parece a una burbuja (el medio de comunicación) con nodos salpicados dentro de ella.

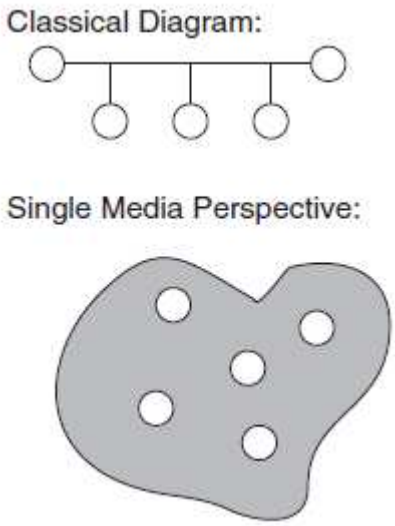

Figura 2. 16: Bus o red de medios que son extraídos como un bus simple o un número de dispositivos en un único medio de comunicación común.
Fuente: Pérez, E. M. (2007).

Si usted visualiza la burbuja (mancha) de la única red de medios, como el aire, la conexión entre los nodos sería las ondas de radio, se puede ver que sólo existe un nodo de transmisión en un momento determinado. Si tenemos una red WiFi en casa, deberíamos apreciar este modelo, ya que sólo un ordenador puede transmitir en cualquier momento.

Parte del hardware de la única red de comunicación debe ser un receptor que supervisa los mensajes salientes para asegurar que los mensajes no colisionen. Si lo hacen, el transmisor esperará una cantidad aleatoria de tiempo antes de volver a intentar él envió del mensaje. Cuando los nodos no están transmitiendo, los receptores de la red deben monitorear continuamente los mensajes que van en torno a ella y grabar cualquier cosa que se dirige para el nodo. Este método de creación de redes probablemente parece ser

innecesariamente compleja, pero como se ha señalado, fue el primer tipo de red informática desarrollada y sus deficiencias fueron dirigidas al desarrollo de otros tipos de redes.

La red tipo estrella (véase la figura 2.17) mejora en la red básica mediante la vinculación de todos los nodos a un solo switch "SW", que es responsable de dirigir los mensajes entre los nodos (directamente desde el transmisor al receptor). La adición del interruptor permite que múltiples mensajes sean transmitidos simultáneamente entre los nodos. Si se tiene acceso a una red Ethernet cableada, esta es la topología que se utiliza. Muchas redes utilizan un router en lugar de un switch para transmisión de mensajes ida y vuelta, y la diferencia entre los dos, es que un router determina la mejor ruta para un mensaje (inclusive pasarlo a otras redes).

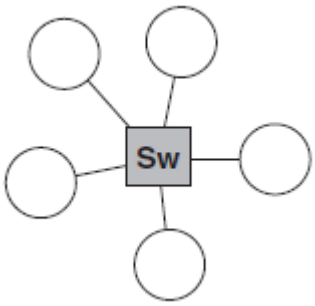

Figura 2. 17: Un switch vincula a todos los nodos mientras los mensajes se dirigen desde el transmisor al receptor.
Fuente: Pérez, E. M. (2007).

Si tenemos un módem de cable en casa, es probable que tengamos que utilizar un router para pasar los datos entre los equipos del hogar, así como Internet. Al proporcionar una conexión de dirección de cada nodo al

conmutador (SW), la red en estrella permite la transmisión de datos mucho más rápido junto con el menor tráfico innecesario a los diversos.

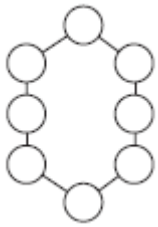

Figura 2. 18: Un anillo de nodos pasa a un símbolo que se utiliza para controlar el movimiento de mensajes entre ordenadores.
Fuente: Pérez, E. M. (2007).

La red tipo anillo (véase la figura 2.18) se esfuerza para evitar el problema de colisión al pasar un símbolo (token), que puede tener un mensaje. Los ordenadores sólo transmiten en la red en anillo si el elemento está vacío cuando se trata de ellos. Si hay un mensaje adjunto al token, el equipo debe despojar y transmitir un token de vacío en su lugar.

La transmisión de mensajes es bastante eficiente, aunque esta topología presenta un gran inconveniente: si un nodo falla, toda la red se lleva hacia abajo. A pesar de este inconveniente, las redes en anillo son utilizados en diversas aplicaciones del área de las telecomunicaciones.

Las supercomputadoras compuesta de varios procesadores y de grandes servidores de alto rendimiento, proporcionan múltiples conexiones de red de cada nodo, aunque lo ideal sería a todos los demás nodos de la red, como se muestra en la figura 2.19. Esta topología se conoce como una red de malla, evita los problemas de posibles colisiones de mensajes haciendo que

cada nodo especifique donde va cada mensaje. Las desventajas evidentes de esta topología, son el coste de tener varios adaptadores de red en todos los nodos, así como la complejidad del cableado entre cada nodo.

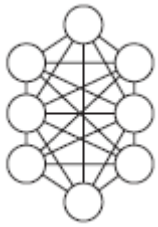

Figura 2. 19: Un red en malla que ofrece un camino directo a los computadores de la red y permite transferencias de datos extremadamente rápidas.
Fuente: Pérez, E. M. (2007).

Es imposible proporcionar una conexión entre cada nodo cuando hay más de cuatro o cinco nodos en una malla, por lo que se ha producido una gran cantidad de investigaciones en diseñar las configuraciones óptimas de malla para superordenadores o servidores construidos a partir de miles de microprocesadores. En realidad, no hay necesidad de una red de malla cuando se trabaja con sistemas de microcontroladores embebidos.

Las grandes redes, como Internet, generalmente se llaman redes híbridas porque se construyen a partir de las redes más pequeñas de diferentes topologías, como se muestra en la figura 2.20. Las interconexiones entre las redes más pequeñas y la híbrida son por lo general a través de puentes (conexiones entre dos conexiones de red diferentes) o enrutadores (cuando las conexiones son las mismas).

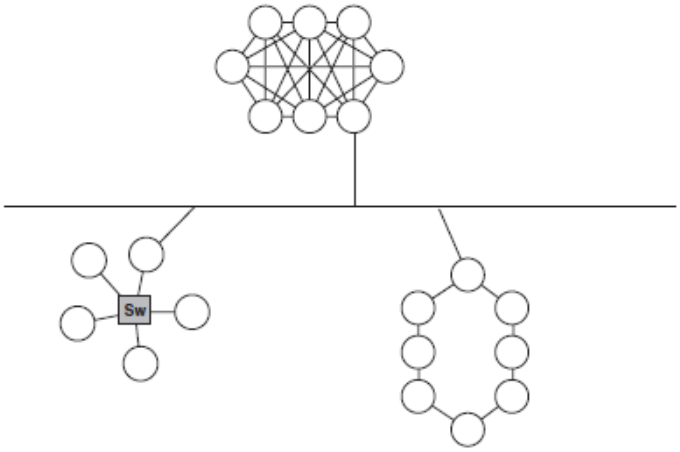

Figura 2. 20: Con una extensa red construida a partir de un número de redes pequeñas por lo general suele ser una red híbrida.
Fuente: Pérez, E. M. (2007).

Por último, hablaremos de los adaptadores de bus universal en serie (USB), también utilizados en ciertas aplicaciones comunes para los microcontroladores embebidos como el MCU PIC. Si bien podemos estar más familiarizado con ellos, son utilizados como dispositivos periféricos para PC y como dispositivos de comunicación punto a punto, que proporciona funciones de E/S para el equipo, creo que de ellos como dispositivos de red. Cuando están conectadas a un PC, van a través de un proceso de enumeración similar a la de nuevos nodos estando conectados a una red.

Ellos pueden ser diseñados para operar independientemente de la PC, usando el puerto USB del PC como un método de transferencia de datos desde el dispositivo a una aplicación de PC. Esto puede ser una manera poco convencional de ver dispositivos USB, pero parece que el desarrollo de aplicaciones para ellos es mucho más fácil si se toma esta perspectiva.

Esta discusión sobre las redes es bastante simplista y hay muchos temas que no hemos discutido con el temor de empantanarse en minucias, cuando el propósito de este capítulo es darle una introducción a los diversos conceptos que están involucrados con la creación de aplicaciones de microcontroladores embebidos.

2.7. Resumen de las características de los microcontroladores PIC.

La elección de un número de parte de un microcontroladores PIC de 8 bits sobre otro para una aplicación específica no es tan difícil como se pueda pensar. Los chips se basan en cuatro arquitecturas de procesador que proporcionan capacidades diferentes aunque requiere muchas de las mismas habilidades de pensamiento y trucos necesarios para crear una aplicación. El desarrollo de código se simplifica en gran medida por el uso algunas plataformas de programación, tales como MPLAB IDE (entorno de desarrollo integrado) de Microchip y Microcode, que se puede utilizar para todos los números de parte de los microcontroladores PIC. Los pines de entrada/salida (E/S) y de periféricos característicos están diseñados para interactuar fácilmente con los dispositivos conectados al MCU PIC utilizando interfaces estándar o pines de propósito general.

2.7.1. Opciones de empaquetado de los microcontroladores PIC.

Los microcontroladores PIC están disponibles en una variedad de encapsulados, como se puede ver en la figura 2.21 los diferentes tipos de encapsulados, así como los códigos alfabéticos especificados. En la figura

2.21 se muestra los paquetes OTP (One-Time Programmable) que se va a utilizar cuando se está aprendiendo a programar sobre microcontroladores PIC.

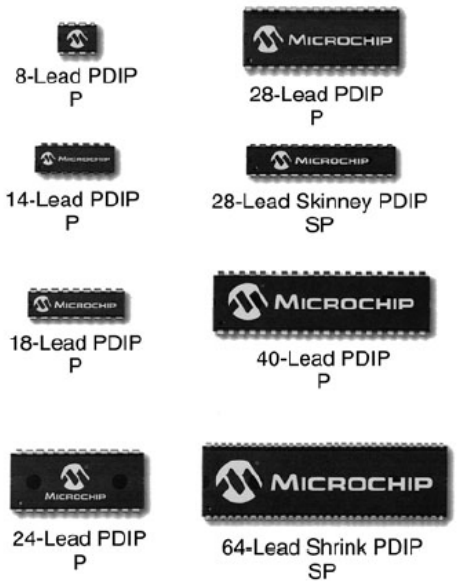

Figura 2. 21: Encapsulamiento de plástico común DIP (Dual in-line).
Fuente: Pérez, E. M. (2007).

El término "OTP" se hizo popular cuando las partes programables (que están normalmente en encapsulados cerámicos, como se muestra en la figura 2.22) se construyeron en envases de plástico sólidos. La razón para hacer esto era el coste mucho más bajo que los encapsulados de plástico y la expectativa de que la parte nunca sería reprogramada (que requiere la disponibilidad de una ventana para la luz ultravioleta hasta borrar la memoria EPROM en el chip). Los microcontroladores PIC con memoria de programa EEPROM y Flash son puestos en paquetes de plástico sólido, como los que se muestran en la figura 2.22, no necesitan de luz ultravioleta para borrar y todavía se identifican como OTP.

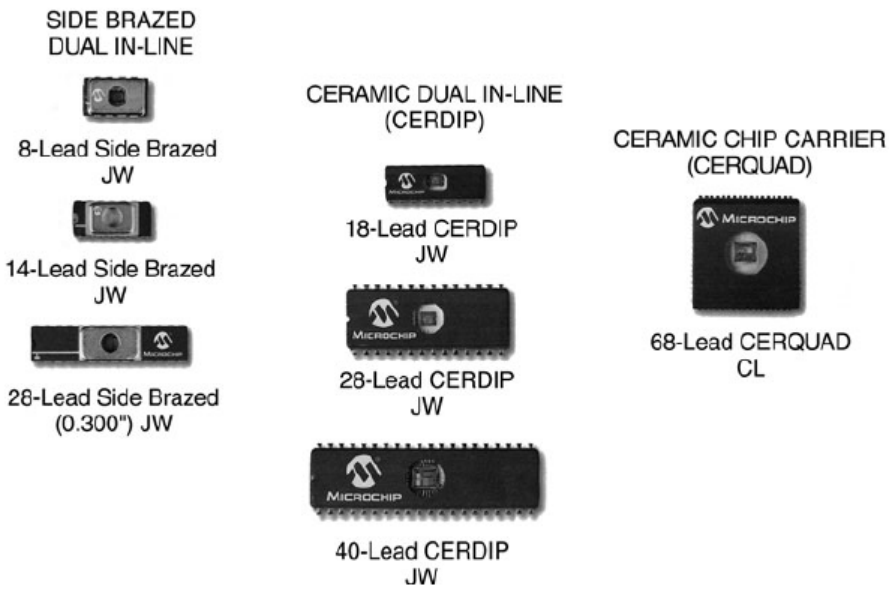

Figura 2. 22: Encapsulamiento de cerámica enventanado de los PICs.
Fuente: Pérez, E. M. (2007).

CAPÍTULO 3: DESARROLLO EXPERIMENTAL

3.1. Materiales utilizados como Hardware del trabajo de titulación.

En la presente sección se muestran los materiales necesarios para el desarrollo experimental del Trabajo de Titulación propuesto.

3.1.1. Tarjeta controladora PIC 16F886.

En la figura 3.1 se muestra el módulo controlador que permite realizar un sinnúmero de aplicaciones mediante el microcontrolador PIC16F886. Es totalmente reprogramable, para lo cual se podría decir que es un sistema microcontrolador embebido, porque a través de ciertos dispositivos electrónicos se desarrollan proyectos electrónicos.

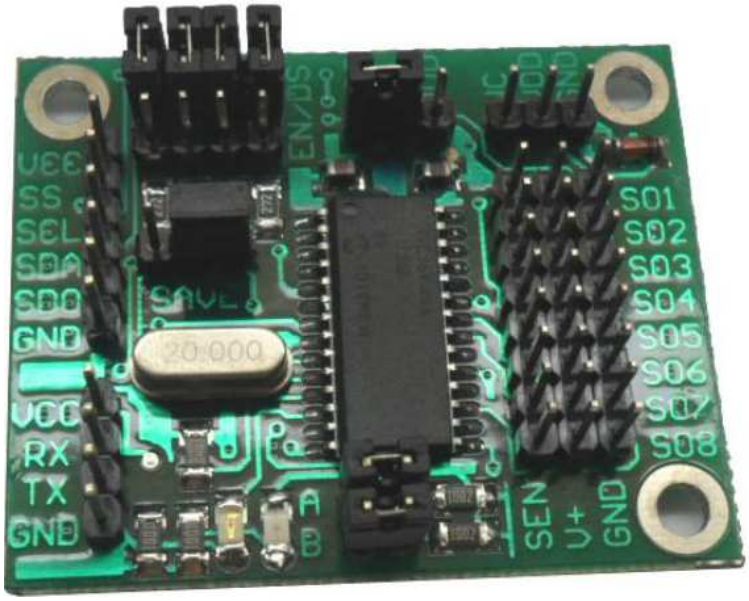

Figura 3. 1: Tarjeta controladora PIC 16F886.
Fuente: El Autor.

A continuación se muestran algunas de las especificaciones técnicas de la tarjeta controladora (véase la figura 3.1):

- Comunicación por UART.
- 10 entradas y salidas digitales.
- 1 led indicador de presencia de voltaje.
- Programación por ICSP.
- Fuente de voltaje mínima 4.5 v
- Fuente de voltaje máxima 5.0 v

En la figura 3.2 se muestra el mismo módulo controlador pero especificando los pines necesarios y el PIC 16F886 para programación: (1) pines que permiten la conexión con antenas de comunicación inalámbrica Xbee y Bluetooth, (2) Leds que permiten visualizar el encendido y alimentación de voltaje en corriente directa, (3) el PIC 16F886 para ser programado, (4) pines digitales de entrada y salida (E/S) y (5) pin para seleccionar alimentación de voltaje externo.

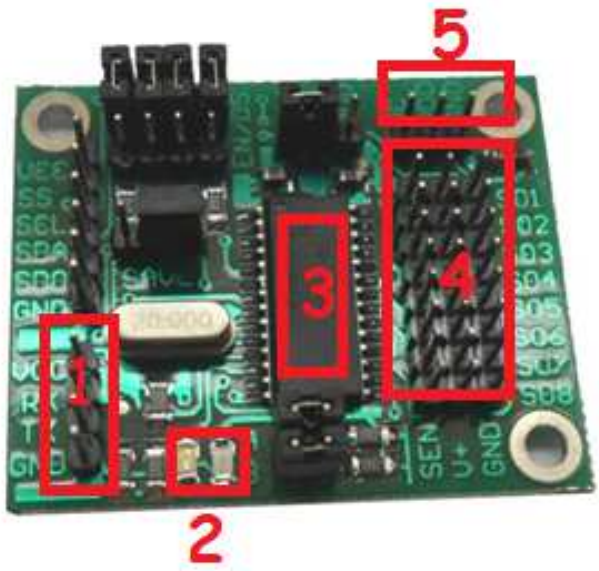

Figura 3. 2: Especificación de la tarjeta controlador PIC 16F886.
Fuente: El Autor.

3.1.2. Antena para comunicación Inalámbrica – Bluetooth.

La antena de Bluetooth HC06 (véase la figura 3.3) transmite los caracteres escritos en la interfaz en java hacia el teléfono por el terminal de conexión TX del microcontrolador PIC16F886.

Figura 3. 3: Dispositivo electrónico – Antena Bluetooth.
Fuente: El Autor.

Este tipo de dispositivo Bluetooth es de clase 1, es decir, que esta denotado por su largo alcance de hasta 100 metros y consume hasta 120mA reduciendo así el tiempo de uso del dispositivo de a lo mucho 6 horas. Mientras, que la tasa de transmisión de bits está configurada desde fábrica a 9600 baudios y es reconfigurable por comandos AT.

3.1.3. Antena para comunicación Inalámbrica – XBee.

Los dos principales estándares inalámbricos utilizados principalmente para redes de sensores inalámbricos son ZigBee e IEEE 802.15.4, debido al bajo consumo de energía, el despliegue de red simple, bajo costo de instalación y las transmisiones de datos confiables, estas dos normas se prefieren sobre todo a través de Wi-Fi y Bluetooth.

ZigBee se construye en la parte superior del estándar IEEE 802.15.4, que define el control de acceso al medio (MAC) y las capas físicas, que opera en una banda sin licencia de 2,4 GHz con una velocidad de transferencia de datos de 250 kbps.

Mediante la interfaz gráfica en java desarrollada para el presente trabajo de titulación, nos permite digitar una serie de caracteres desde el teclado del computador y conectado a la antena Xbee, para transmitir sus datos hacia el concentrador de datos, para ser mostrados en un display LCD 16x2 y sean retransmitidos o reproducidos a una aplicación (app) instalada en el teléfono inteligentes (Smartphone).

A continuación se muestran algunas características del módulo que se muestra en la figura 3.4.

- Se alimenta de 5 voltios DC, la señal de control son digitales.
- Consume de 20mW.
- La tasa de trasmisión de bits (BAUDRATE) es de 9600.

Figura 3. 4: Dispositivo electrónico – Antena XBee Pro.
Fuente: El Autor.

3.2. Desarrollo del Proyecto.

En la presente sección procedemos a describir el funcionamiento del sistema intercomunicador de mensajes de texto para discapacitados con afonía funcional. Adicionalmente se mostrará el diseño de una aplicación (app) para sistemas operativos Android que permiten la comunicación mediante tecnología Bluetooth. Es decir, que dicha aplicación debe instalarse en cualquier teléfono inteligente que soporte S. O. Android.

3.2.1. Diagrama esquemático del circuito máster.

En la figura 3.5 se muestra el diagrama esquemático del circuito máster del sistema intercomunicador, operado y controlado por la circuito portátil controlado por el PIC16F886.

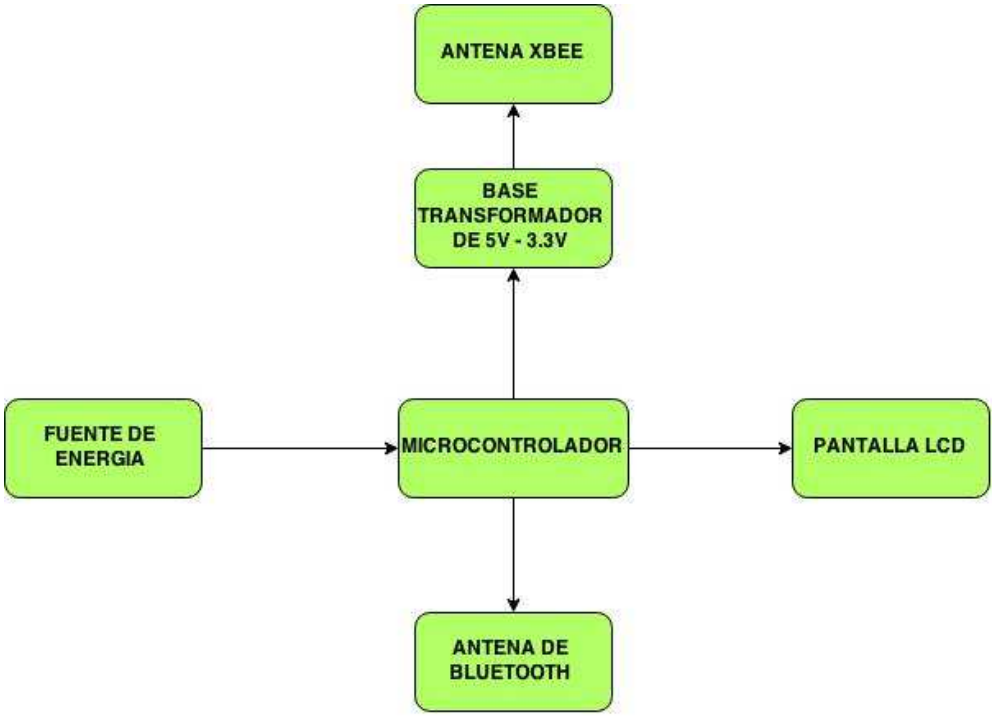

Figura 3. 5: Diagrama esquemático del sistema intercomunicador para discapacitados con afonía funcional.
Fuente: El Autor.

3.2.2. Interconexiones del circuito máster.

En las tablas 3.1 a 3.3 se muestran las interconexiones de los pines de la tarjeta controladora PIC 16F886 entre los dispositivos: LCD 16x2, antenas Xbee y Bluetooth. En otras palabras, la tarjeta controladora o master del sistema intercomunicador se encarga de la transmisión y recepción de datos.

Tabla 3. 1: Interconexión entre tarjeta controladora PIC 16F886 y LCD 16x2.

LCD 16X2	PIC 16F886
RS	PIN RB0
EN	PIN RB2
D4	PIN RA0
D5	PIN RA1
D6	PIN RA2
D7	PIN RA3
-	GND
+	5V

Fuente: Los Autores

Tabla 3. 2: Interconexión entre tarjeta controladora PIC 16F886 y Antena XBee.

ANTENA XBEE	PIC 16F886
-	GND
+	+ 5V
PIN TX	PIN RX

Fuente: Los Autores

Tabla 3. 3: Interconexión entre tarjeta controladora PIC 16F886 y Antena Bluetooth.

ANTENA BLUETOOTH	PIC 16F886
-	GND
+	+ 5V
PIN RX	PIN TX

Fuente: Los Autores

3.2.3. Diseño de la aplicación ‘app’ sobre Android.

Otra parte importante del sistema intercomunicador, es el desarrollo de una aplicación sobre AppInventor, la misma se compone de botoneras, servidor de comunicación bluetooth, reloj y del convertidor de texto-audio.

a) Interface principal.

La aplicación es totalmente amigable con el usuario, basta con presionar el botón CONECTAR (véase la figura 3.6) para que el teléfono smartphone comience a recibir los datos enviados desde el microcontrolador PIC concentrados en caracteres.

Figura 3. 6: Pantalla principal del asistente intercomunicador para discapacitados de afonía funcional.
Fuente: El Autor.

b) Interface de programación.

AppInventor presenta un diseño mediante diagramas de bloques (véase la figura 3.7) y que servirán para utilizar elementos incorporados en el panel de componentes. Se utilizan sentencias de validación IF ELSE, WHILE, etc.

Esto permitirá la reproducción de los caracteres escritos por el usuario mediante la codificación de la recepción de datos vía bluetooth.

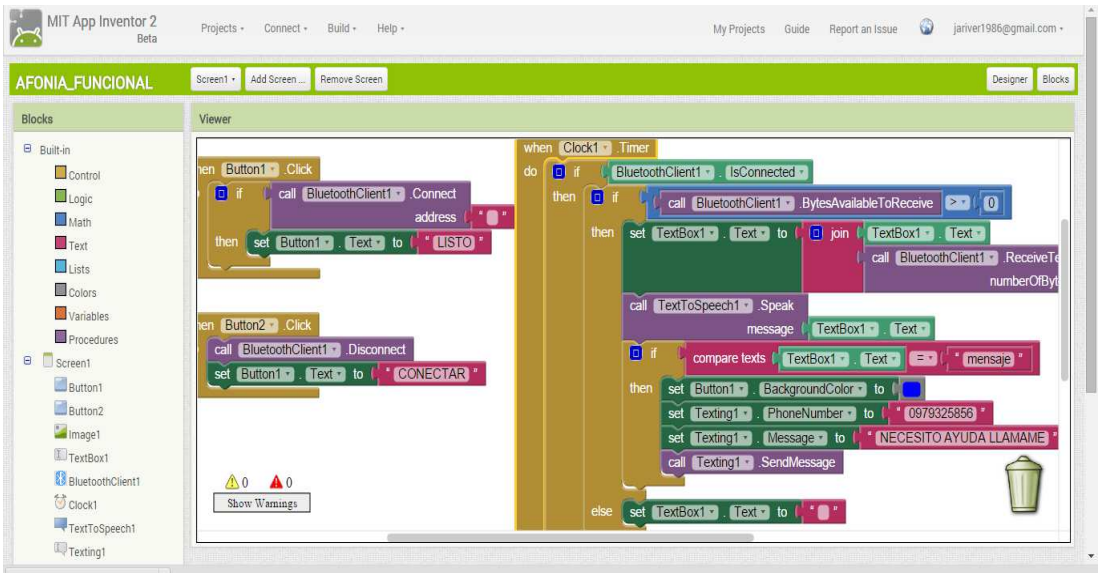

Figura 3. 7: Pantalla de la interface de programación en AppInventor.
Fuente: El Autor.

c) Código de Comunicación con Dispositivos.

La presente interface presenta la codificación para enlaces de datos vía bluetooth (véase la figura 3.8), los caracteres son transmitidos y recibidos a una frecuencia de 2,4 GHz con estándar ISM (Industrial, científica y médica).

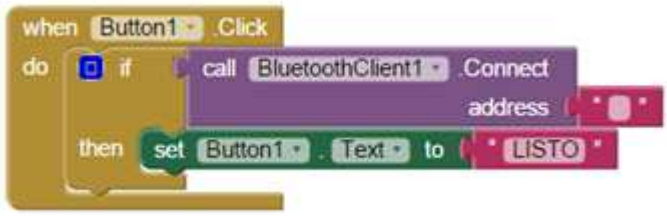

Figura 3. 8: Pantalla de la codificación para enlaces vía Bluetooth.
Fuente: El Autor.

d) Código de recepción de datos.

Los caracteres son almacenados en una variable y permiten interpretar con el único propósito de que el oyente entienda los pensamientos del usuario de este dispositivo. En la figura 3.9 se muestra el código de recepción.

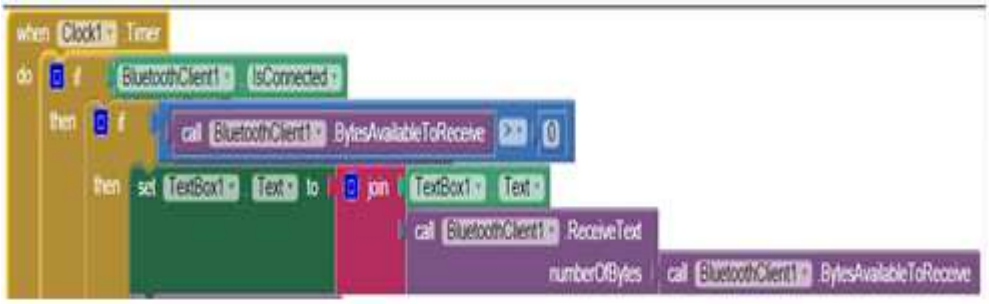

Figura 3. 9: Pantalla de la codificación para recepción de datos.
Fuente: El Autor.

e) Código de reproducción de la recepción de texto.

La función TextToSpeech mostrada por la figura 3.10, reproduce en voz los caracteres digitados por el minusválido de afonía funcional.

Figura 3. 10: Pantalla de la codificación para reproducción de datos recibidos.
Fuente: El Autor.

f) Código para romper enlaces de datos.

La función Disconnect que se muestra en la figura 3.11, permite romper el enlace entre el teléfono y el dispositivo asistente.

Figura 3. 11: Pantalla de la codificación para reproducción de datos recibidos.
Fuente: El Autor.

3.2.4. Código de programación del intercomunicador para discapacitados con afonía funcional.

```
program afonia_funcional
dim LCD_RS as sbit at RB0_bit
   LCD_EN as sbit at RB2_bit
   LCD_D4 as sbit at RA0_bit
   LCD_D5 as sbit at RA1_bit
   LCD_D6 as sbit at RA2_bit
   LCD_D7 as sbit at RA3_bit
   LCD_RS_Direction as sbit at TRISB0_bit
   LCD_EN_Direction as sbit at TRISB2_bit
   LCD_D4_Direction as sbit at TRISA0_bit
   LCD_D5_Direction as sbit at TRISA1_bit
   LCD_D6_Direction as sbit at TRISA2_bit
   LCD_D7_Direction as sbit at TRISA3_bit
dim TEXTO as char[33]
main:
PORTA = %00000000
PORTB = %00000000
PORTC = %00000000
TRISA = %00000000
TRISB = %00000000
TRISC = %10000000
ANSEL = %00000000
ANSELH= %00000000
UART1_Init(9600)
Lcd_Init()
Lcd_Cmd(_LCD_CLEAR)
Lcd_Out(1,1,"AFONIA FUNCIONAL")       ' escribe el mensaje en la primera
fila
Lcd_Out(2,1,"POR: JCH & JCL ")          ' escribe el mensaje en la segunda fila
delay_ms(2000)                    ' tiempo de espera
Lcd_Cmd(_LCD_CLEAR)                ' encera lcd
Lcd_Cmd(_LCD_CURSOR_OFF)             ' apaga cursor
while 1
   if (UART1_Data_Ready() <> 0) then  ' Si el dato es recibido
    UART1_Read_Text(TEXTO,":",32)     ' lee la cadena hasta que encuentre la
letra K mayuscula
    UART1_Write_Text(TEXTO)          ' envia un eco de la cadena
    Lcd_Cmd(_LCD_CLEAR)             ' encera lcd
    Lcd_Out(1,1,TEXTO)              ' me muestra la cadena que escribiste via
serial en el lcd
    Delay_ms(1000)
```

```
    end if
  wend
  end.
```

ASIGNACION DE LCD

```
dim   LCD_RS as sbit at RB0_bit
      LCD_EN as sbit at RB2_bit
      LCD_D4 as sbit at RA0_bit
      LCD_D5 as sbit at RA1_bit
      LCD_D6 as sbit at RA2_bit
      LCD_D7 as sbit at RA3_bit
      LCD_RS_Direction as sbit at TRISB0_bit
      LCD_EN_Direction as sbit at TRISB2_bit
      LCD_D4_Direction as sbit at TRISA0_bit
      LCD_D5_Direction as sbit at TRISA1_bit
      LCD_D6_Direction as sbit at TRISA2_bit
      LCD_D7_Direction as sbit at TRISA3_bit
```

CONFIGURACION DE TASA DE BITS

```
UART1_Init(9600)
```

CONFIGURACION MENSAJE A LCD

```
Lcd_Init()
Lcd_Cmd(_LCD_CLEAR)
Lcd_Out(1,1,"AFONIA FUNCIONAL")    ' escribe el mensaje en la primera fila
delay_ms(2000)                     ' tiempo de espera
Lcd_Cmd(_LCD_CLEAR)                ' encera lcd
Lcd_Cmd(_LCD_CURSOR_OFF)           ' apaga cursor

   if (UART1_Data_Ready() <> 0) then    ' Si el dato es recibido
    UART1_Read_Text(TEXTO,":",32)       ' lee cadenade caracteres
    UART1_Write_Text(TEXTO)             ' envia un eco de la cadena
    Lcd_Cmd(_LCD_CLEAR)                 ' encera lcd
    Lcd_Out(1,1,TEXTO)                  ' me muestra la cadena que escribiste via
                                        serial en el lcd
   end if
```

3.2.5. Diseño de aplicación para comunicación de puertos seriales.

Para la comunicación entre puertos seriales, se desarrolla una aplicación en Java, es decir, que usaremos librerías JSSC (Java-Simple-Serial-Connector), es una librería para java que permite comunicaciones con puertos seriales, y soportado por varios Sistemas Operativos de código abierto y cerrado. En la figura 3.12 se muestra la interfaz del programa.

Figura 3. 12: Pantalla de la interface de comunicación serial.
Fuente: El Autor.

Declaración de Variables

A continuación se muestran las variables que se usaremos:

```
static SerialPort puerto;
static String[] nombresPuertos;
String[] puertosNuevos ;
String nuevoPuerto;
static boolean conectado=false;
int limite=32;
```

Donde,

Puerto: es el nombre de la variable con el puerto COM que permite las comunicaciones, y mantiene la conexión.

NombresPuertos: es un arreglo donde almacenaremos temporalmente el nombre de todos los puertos COM conectados a la pc.

PuertosNuevos: es un arreglo donde almacenaremos temporalmente el nombre de los puertos nuevos que son detectados.

Conectado: es una bandera para saber el estado de la conexión.

Límite: es una variable tipo entero para señalar el límite de caracteres que se podrá escribir en la pantalla.

Método comprobarPuertos

Este Método estará corriendo en segundo plano para detectar nuevas conexiones, estará en un hilo, antes de iniciar la conexión, para así poder detectar el dispositivo para hacer la comunicación serial. Se estará actualizando el arreglo <<puertosNuevos>> llamando al método <<getPortNames>>, que es una función que provee librerías JSSC para obtener la lista de los puertos, entonces si la lista de puertos nuevos es diferente a la lista de los puertos listados anteriormente, entonces asignamos a la variable <<nuevoPuerto>> el nombre del puerto que nos retornara el método <<compararPuertos>>, una vez hecho esto ya procedemos a hacer la conexión llamando al método conectar Puerto.

```
void comprobarPuertos()
{
    try {
            puertosNuevos = SerialPortList.getPortNames();
           if(puertosNuevos.length!= nombresPuertos.length)
           {

               if(puertosNuevos.length>nombresPuertos.length)
               {
                nuevoPuerto=compararPuertos();
                conectarPuerto();
               }


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

Método compararPuertos (retorna una cadena de caracteres)

Este método retorna el nombre del nuevo puerto detectado, con unos bucles FOR esta comparado el contenido de los arreglos puertosNuevos y nombresPuertos hasta encontrar el nuevo Puerto, que en este caso sería el del dispositivo conectado, este método es llamado desde el método comprobarPuertos , para asignar a una variable el nombre del nuevo puerto.

```
String compararPuertos()
{
    String nuevo="";
    for (int i = 0; i < puertosNuevos.length; i++) {
        for (int j = 0; j < nombresPuertos.length; j++) {
            if(puertosNuevos[i].equalsIgnoreCase(nombresPuertos[j]))
              break;
          else if(j==nombresPuertos.length-1)
                nuevo=puertosNuevos[i];
        }
    }
return nuevo;
}
```

Método conectarPuerto

Este método nos permite realizar la conexión entre nuestro programa y el puerto serial para poder establecer la comunicación, para lograr esto se utiliza una instancia de la clase SerialPort (que la provee la librería JSSC) enviándole como parámetro la cadena con el nombre del puerto nuevo. Si se puede abrir el puerto entonces se inicializa la conexión pasándole los parámetros tales como, BaudRate, DataBits, etc.

```
void conectarPuerto()
{
    String p=nuevoPuerto;
     puerto = new SerialPort(p);
    try {
        if(puerto.openPort())
        {
            puerto.setParams(SerialPort.BAUDRATE_9600,SerialPort.DATABITS_8,
                             SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);
            conectado=true;
            texto.setEnabled(true);

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

}
```

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES.

4.1. Conclusiones.

- La fundamentación teórica de los Microcontroladores PIC permitió describir los tipos de sistemas embebidos disponibles, así como las diferentes aplicaciones que puede realizar los PICs mediante las arquitecturas utilizadas por microchip, y a la vez se estableció la robustez de este dispositivo, lo cual sirvió de gran ayuda para diseñar la implementación del prototipo portátil de intercomunicación.

- Mediante el diseño del circuito portátil se establecieron los parámetros necesarios para el correcto funcionamiento del sistema intercomunicador, así como los demás dispositivos (periféricos) necesarios para el correcto funcionamiento del sistema.

- Mediante el desarrollo de dos aplicaciones tales como, AppInventor y Java se permitió realizar la comunicación interactiva con los dispositivos electrónicos dentro del circuito portátil intercomunicador.

4.2. Recomendaciones.

- Utilizar con mayor frecuencia los dispositivos electrónicos embebidos PICs a través de aplicaciones prácticas relacionadas con la carrera con apoyo de los estudiantes de Ingeniería en Telecomunicaciones y Electrónica

REFERENCIAS BIBLIOGRÁFICAS

Bren, D. (2005). *NISC Technology & Toolset.* Escuela de Información y Ciencias de la Computación, Universidad de California, Irvine, Estados Unidos.

García B., E. (2008). *Compilador C CCS y simulador PROTEUS para Microcontroladores PIC*. Barcelona: Marcombo.

González L., A. J. (2013). *Aplicaciones prácticas de Microcontroladores a través de la plataforma de programación MatLab.* Trabajo de titulación publicado en el repositorio de la Carrera de Ingeniería en Telecomunicaciones, Universidad Católica de Santiago de Guayaquil, Guayaquil, Ecuador.

Melchor, N. (2009). *Tarjeta de Desarrollo para Microcontroladores PIC*. Tesis de Pregrado publicado en el repositorio del Instituto Politécnico Nacional, México.

Pérez, E. (2007). *Microcontroladores PIC: Sistema Integrado para el autoaprendizaje.* Editorial Marcombo, Barcelona, España.

Rossano, V. (2013). *Electrónica & Microcontroladores PIC*. USERSHOP.

Valdés, F. & Pallás, R. (2007). *Microcontroladores: Fundamentos y aplicaciones con PIC*. Editorial Marcombo, Barcelona, España.

Verle, M. (2010). *PIC Microcontrollers – Programming in Basic.* MikroElektronika, Belgrado.