UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TEMA:

DISEÑO DE UN ROBOT MÓVIL CONTROLADO POR UNA TABLET Y PROGRAMACIÓN EN ANDROID QUE PERMITA LA COMUNICACIÓN POR BLUETOOTH

Previa la obtención del Título

INGENIERO EN TELECOMUNICACIONES

ELABORADO POR:
Marcel Geovanny Bake Fajardo
Hernán Gustavo Moreno Godoy

Guayaquil, 10 de Septiembre del 2013

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por los Sres. **Marcel Geovanny Bake Fajardo** y**Hernán Gustavo Moreno Godoy**como requerimiento parcial para la obtención del título de INGENIERO EN TELECOMUNICACIONES.

Guayaquil, 10 de Septiembre del 2013

DIRECTOR

_____

MsC. Edwin Palacios Meléndez

REVISADO POR

_____

Ing. Marcos Montenegro Tamayo.
Revisor Metodológico

_____

MsC. Néstor Zamora Cedeño.
Revisor de Contenido

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

MARCEL GEOVANNY BAKE FAJARDO
HERNÁN GUSTAVO MORENO GODOY

DECLARAMOS QUE:

El proyecto de tesis denominado “Diseño de un Robot Móvil controlado por una Tablet y programación en Android que permita la comunicación por Bluetooth” ha sido desarrollado con base a una investigación exhaustiva, 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, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Guayaquil, 10 de Septiembre del 2013

LOS AUTORES

MARCEL GEOVANNY BAKE FAJARDO
HERNÁN GUSTAVO MORENO GODOY

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Nosotros, MARCEL GEOVANNY BAKE FAJARDO
HERNÁN GUSTAVO MORENO GODOY

Autorizamos a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del proyecto titulado: “Diseño de un Robot Móvil controlado por una Tablet y programación en Android que permita la comunicación por Bluetooth”, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Guayaquil, 10 de Septiembre del 2013

LOS AUTORES

MARCEL GEOVANNY BAKE FAJARDO
HERNÁN GUSTAVO MORENO GODOY

DEDICATORIA

El presente trabajo de graduación es dedicado primeramente a nuestro padre celestial “Dios” quién nos ha guiado siempre por el sendero del buen camino, para nunca desmayar y seguir adelante con nuestra meta.

También dedicamos este logro a nuestros padres ya que sin el apoyo, consejos, comprensión, amor, y ayuda en los momentos difíciles no hubiéramos logrado alcanzar nuestro objetivo.

LOS AUTORES

MARCEL GEOVANNY BAKE FAJARDO
HERNÁN MORENO

AGRADECIMIENTO

El presente trabajo es en agradecimiento a DIOS, que con sus bendiciones nos ha permitido hacer realidad nuestro objetivo de ser profesionales en el campo de las Telecomunicaciones.

A nuestro Tutor de Tesis, MsC. Fernando Palacios Meléndez que gracias a sus conocimientos, esfuerzo, experiencia, paciencia, tiempo y dedicación, para culminar con el presente trabajo.

A todos los docentes de la Carrera de Ingeniería en Telecomunicaciones, quienes aportaron con todos sus conocimientos a nuestra formación, y en especial a mis profesores, al Decano FETD,MsC. Manuel Romero Paz y alMsC. Fernando Palacios Meléndez, por sus consejos, su enseñanza y más que todo por su amistad.

Son muchas las personas a quienes debemos agradecer pero sería imposible describirlos a cada uno de ellos, pero muchas gracias por su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de nuestras vidas.

LOS AUTORES

MARCEL GEOVANNY BAKE FAJARDO
HERNÁN MORENO

Índice General

Índice de Figuras 10
Índice de Tablas 13
Resumen 14
CAPÍTULO 1: GENERALIDADES DEL TRABAJO DE TITULACIÓN 15
1.1. Introducción. 15
1.2. Antecedentes. 15
1.3. Justificación del Problema. 16
1.4. Definición del Problema. 16
1.5. Objetivos de la Investigación. 16
1.5.1. Objetivo General. 16
1.5.2. Objetivos Específicos. 16
1.6. Idea a Defender. 17
1.7. Metodología de Investigación. 17
CAPÍTULO 2: Estado del Arte de los Microcontroladores y sus Aplicaciones. . 18
2.1. El Microcontrolador (uC). 18
2.2. Tipos de Microcontroladores. 19
2.3. Arquitectura Interna de los Microcontroladores. 22
2.3.1. Arquitectura von Neumann. 23
2.3.2. Arquitectura Harvard. 25
2.4. Arquitecturas CISC y RISC. 26
2.5. Aplicaciones de los Microcontroladores. 28
2.6. La familia de los Microcontroladores. 29
2.7. Los Microcontroladores PIC. 32
2.8. Gamas del microcontrolador PIC de 8 bits. 35
2.8.1. Gama enana: instrucciones de 8 bits. 35
2.8.2. Gama baja: instrucciones de 12 bits. 37
2.8.3. Gama media: instrucciones de 14 bits. 38
2.8.4. Gama alta y gama mejorada: instrucciones de 16 bits. 40
2.9. El Microcontrolador PIC16F887. 42
2.9.1. Características básicas del PIC16F887. 44
2.9.2. Descripción de Pines del PIC16F887. 45

2.9.3. Los puertos de Entrada y Salida (E/S) del PIC16F887. 49
CAPÍTULO 3: PROGRAMACIÓN PIC BASIC Y SISTEMA OPERATIVO ANDROID. 51
3.1. Lenguaje de programación Pic Basic. 51
3.2. Características del Lenguaje de Programación Pic Basic. 55
3.2.1. Identificadores. 58
3.2.2. Comentarios. 59
3.2.3. Constantes. 59
3.3. Declaraciones disponibles en el compilador PBP. 60
3.4. Componente y operadores en Pic Basic. 62
3.4.1. Define. 62
3.4.2. Variables. 63
3.4.3. Arrays. 64
3.4.4. Alías o símbolos. 65
3.4.5. Operadores aritméticos. 65
3.4.6. Operadores binarios. 66
3.4.7. Operadores comparadores. 67
3.4.8. Operadores lógicos. 67
3.5. Sistema Operativo Android. 68
3.5.1. Android SDK. 69
3.5.2. Entorno de Desarrollo Integrado (IDE). 71
3.5.3. Java Development Kit (JDK). 72
3.5.4. Native Development Kit (NDK). 72
3.5.5. Android Virtual Device (AVD) 73
CAPÍTULO 4: DESARROLLO EXPERIMENTAL 73
4.1. Diseño del Robot Recolector Radiocontrolado. 73
4.2. Hardware del Robot Recolector Radiocontrolado. 74
4.3. Software del Robot Recolector Radiocontrolado. 77
4.3.1. Código de programación 81
4.3.2. Algoritmos de movimiento del robot. 83
4.4. Programación del PIC16F886. 84
CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES. 91

5.1. Conclusiones. 91
5.2. Recomendaciones. 91
REFERENCIAS BIBLIOGRÁFICAS 93

Índice de Figuras

Capítulo 2

Figura 2. 1: Comportamiento de ventas comerciales de los microcontroladores. 20
Figura 2. 2: Comportamiento del crecimiento económico de fabricantes de uC´s. 21
Figura 2. 3: Comportamiento de la cuota de mercado de fabricantes de uC´s. 21
Figura 2. 4: Comportamiento de la capitalización bursátil de fabricantes de uC´s. 22
Figura 2. 5: Arquitectura Interna de uC´s. 22
Figura 2. 6: Arquitectura Von Neumann. 24
Figura 2. 7: Arquitectura Von Neumann para uC. 25
Figura 2. 8: Arquitectura Harvard. 25
Figura 2. 9: Arquitectura Harvard para uC. 26
Figura 2. 10: Arquitectura RISC y CISC. 27
Figura 2. 11: Aplicaciones de Microcontroladores 28
Figura 2. 12: Diagrama de pines del Microcontrolador Intel 8051. 30
Figura 2. 13: Recursos de los puertos de E/S 33
Figura 2. 14: El microcontrolador PIC de la gama 16F628A 34
Figura 2. 15: Microcontroladores PIC10FXX y PIC12FXX 36
Figura 2. 16: Microcontroladores PIC12F508/5009 y PIC12F675 36
Figura 2. 17: Microcontroladores PIC16F720/721. 38
Figura 2. 18: Microcontroladores PIC16F884/887. 39
Figura 2. 19: Microcontroladores PIC18F44J10 y PIC18F45J10. 41
Figura 2. 20: Microcontroladores PIC16F884/887 de 40 pines - PDIP. 43
Figura 2. 21: Microcontroladores PIC16F884/887 de 44 pines -QFN. 43
Figura 2. 22: Diagrama de bloques de Microcontroladores PIC16F884/887. ... 44
Figura 2. 23: Puertos A, B, C, D y E de E/S del PIC16F887. 50

Capítulo 3

Figura 3. 1: Niveles del Lenguaje de Programación. 51
Figura 3. 2: Diagrama de bloques de la Programación de Microcontroladores. 52
Figura 3. 3: Diagrama de bloques de la Programación en Basic para PIC´s ... 52
Figura 3. 4: Diagrama de bloques de la Programación Assembler para PIC´s 53
Figura 3. 5: Programación en Pic Basic para encender un diodo LED. 53
Figura 3. 6: Programación en Assembler para encender un diodo LED. 54
Figura 3. 7: Programación en Assembler para encender un diodo LED. 55
Figura 3. 8: Ejemplo de un programa mal escrito en Basic. 57
Figura 3. 9: Ejemplo de un programa correctamente escrito en Basic. 57
Figura 3. 10: programación Basic 58
Figura 3. 11: Representación de los identificadores en Pic Basic 59
Figura 3. 12: Representación de comentarios en Pic Basic 59
Figura 3. 13: Representación de una constante en Pic Basic 60
Figura 3. 14: Representación de constantes en formatos del sistema de numeración 60
Figura 3. 15: Android SDK Manager. 70
Figura 3. 16: Android SDK Manager. 72
Figura 3. 17: Android JDK. 72

Capítulo 4

Figura 4. 1: Diseño Mecánico del Robot Recolector. 74
Figura 4. 2: Diagrama Esquemático del Robot Recolector Radiocontrolado. ... 75
Figura 4. 3: Icono AppInventor de Android. 77
Figura 4. 4: Interfaz gráfica AppInventor. 78
Figura 4. 5: Menú Paleta. 78
Figura 4. 6: Menú Paleta. 79
Figura 4. 7: Menú Componentes. 80
Figura 4. 8: Menú Propiedades. 81
Figura 4. 9: Algoritmo Slider. 82
Figura 4. 10: Algoritmo Servo Motor. 82
Figura 4. 11: Algoritmo del movimiento hacia adelante. 83

Figura 4. 12: Algoritmo del movimiento hacia la derecha. 83
Figura 4. 13: Algoritmo del movimiento hacia la izquierda. 84
Figura 4. 14: Algoritmo del movimiento hacia atrás..... 84

Índice de Tablas

Capítulo 2

Tabla 2. 1: Fabricantes y modelos de microcontroladores 29
Tabla 2. 2: Modelos de microcontroladores INTEL de 8 bits. 30
Tabla 2. 3: Modelos de microcontroladores MOTOROLA 6805. 31
Tabla 2. 4: Modelos de microcontroladores PIC16CXX. 32
Tabla 2. 5: Características de los modelos de microcontroladores PIC12FXXX. 37
Tabla 2. 6: Características de los modelos PIC16F7XX. 38
Tabla 2. 7: Características de los modelos PIC16F7XX. 39
Tabla 2. 8: Características de los modelos PIC18F24J10/25J10/44J10/45J10. 41
Tabla 2. 9: Descripción de pines del PIC16F887. 46

Capítulo 3

Tabla 3. 1: Instrucciones de Pic Basic Pro. 61
Tabla 3. 2: Parámetros y descripción de la instrucción Define. 63
Tabla 3. 3: Tipos de variables y su descripción para programación Basic. 64
Tabla 3. 4: Tipos de operadores aritméticos. 66
Tabla 3. 5: Tipos de operadores binarios. 66
Tabla 3. 6: Tipos de operadores comparadores. 67
Tabla 3. 7: Tipos de operadores lógicos. 68

Capítulo 4

Tabla 4. 1: Desplazamiento de Control de Carro 75
Tabla 4. 2: Trama para el control de pinza. 76
Tabla 4. 3: Trama para el control de brazo 76

Resumen

El presente trabajo de titulación describe inicialmente el problema de investigación, el cual nos permite conocer la problemática encontrada, que es el desarrollo de una aplicación práctica mediante dispositivos electrónicos, es decir, un pequeño robot móvil diseñado a través de un microcontrolador PIC y controlado por una Tablet con sistema abierto como “Android” mediante el medio de transmisión inalámbrico denominado bluetooth. Lo novedoso del trabajo de titulación es el empleo de una herramienta nueva como lo es AppInventor, tema nunca visto ni estudiado en la Carrera de Ingeniería en Telecomunicaciones y Electrónica en Control y Automatismo.

En el Capítulo 1, se muestran los justificativos, antecedentes del problema de investigación, la definición del problema, posteriormente se muestran: el objetivo general, específicos, idea a defender o hipótesis y la metodología de investigación empleada en el presente trabajo de titulación.

En el Capítulo 2, se describe el Estado del Arte de los Microcontroladores y sus Aplicaciones, la misma que nos permitió para el diseño de una aplicación específica.

En el Capítulo 3, se describen las plataformas de programación de alto nivel a nivel del Microcontrolador (robot recolector), y de AppInventor que es la herramienta que permite instalar en una Tablet con Sistema Operativo Android y manipular el robot desde cualquier dispositivo electrónico que tenga dicho sistema operativo.

En el capítulo 4, se realiza el desarrollo experimental en la cual se muestra los pasos para programar en Basic y en AppInventor.

En el Capítulo 5, se presentan las conclusiones y recomendaciones.

CAPÍTULO 1: GENERALIDADES DEL TRABAJO DE TITULACIÓN

1.1. Introducción.

La electrónica ha tenido cambios tecnológicos que permiten implementar diferentes aplicaciones como en este caso el diseño de robots móviles, a través de microcontroladores, aunque los primeros diseños de robots móviles no requerían del dispositivo electrónico en mención. En la Facultad de Educación Técnica para el Desarrollo (FETD) de la Universidad Católica de Santiago de Guayaquil (UCSG) cuenta con un Laboratorio de Electrónica, el mismo que dispone de equipos y dispositivos electrónicos para realizar cualquier prototipo o pruebas experimentales que son previos al diseño final de un robot.

Adicionalmente el medio de transmisión empleado en el diseño del robot móvil para ser teleoperado, es Bluetooth, el mismo que permite comunicarse a cortas distancias. El dispositivo electrónico para poder manipular al robot móvil es una Tablet de cualquier marca (excepto el Ipad) que incluya el módulo de comunicación por Bluetooth y del sistema Android 4.0 como mínimo.

1.2. Antecedentes.

A mediados del año 2009, la FETD ha venido investigando y desarrollando proyectos a través de los microcontroladores, la mayoría de proyectos han permitido que el nivel de conocimientos se incremente. De acuerdo a cambios curriculares realizados en la Carrera de Ingeniería en Telecomunicaciones se podría considerar a la robótica como una herramienta robusta que permite un sinnúmero de aplicaciones. En mayo de 1998 (malla 1) se inicia la carrera de Ingeniería en Telecomunicaciones, ya casi al cumplirse 15 años de creación la malla curricular ha sido modificada en 2 ocasiones (malla 2 del año 2002; y la malla 4 denominada actualización curricular 2012).

Dichas reformas académicas cuya aprobación depende de la Comisión Académica y del Consejo Directivo de la FETD, y que obedece a estos cambios por el proceso acelerado de los avances tecnológicos en materias del área de

telecomunicaciones y electrónica, esta última debe considerar los laboratorios de las asignatura impartidas en la mencionada área.

1.3. Justificación del Problema.

A través del desarrollo de una plataforma (software) programada en Android, para así lograr manipular al robot móvil mediante una Tablet a través del medio de transmisión inalámbrico a cortas distancias denominado Bluetooth, es decir, que la Tablet se comporta como un control remoto (joystick) para teleoperar (controlar) al robot móvil para ciertas aplicaciones específicas.

1.4. Definición del Problema.

Actualmente en la Facultad de Educación Técnica para el Desarrollo (FETD) se han desarrollado aplicaciones a través de microcontroladores, específicamente robots móviles autónomos (no controlados) y teleoperados mediante una joystick (palanca) inalámbrico, por eso surge la necesidad de diseñar un robot móvil independientemente de su función controlado por otro dispositivo electrónico distinto al ya mencionado, es decir, a través de una Tablet (cualquier modelo, excepto las Ipad´s) que incluya el sistema Android donde se realizará la programación para controlar mediante Bluetooth al robot móvil.

1.5. Objetivos de la Investigación.

1.5.1. Objetivo General.

Diseñar un robot móvil teleoperado por una Tablet y desarrollo de la programación en Android que permita la comunicación por Bluetooth.

1.5.2. Objetivos Específicos.

1. Revisar el estado del arte de los microcontroladores y de compiladores para desarrollo de aplicaciones mediante lenguajes de programación de alto nivel.

2. Diseñar el robot móvil a través de microcontroladores y demás dispositivos electrónicos que incluya el dispositivo de comunicación Bluetooth.
3. Programar en lenguaje de alto nivel (Basic o C) al microcontrolador que se encarga de interpretar las señales enviadas desde la Tablet.
4. Programar en Android el sistema que permite el control del robot móvil.
5. Evaluar experimentalmente al robot móvil teleoperado por una Tablet.

1.6. Idea a Defender.

A través del diseño de un robot móvil que será teleoperado mediante una Tablet, permitirá mejorar los conocimientos de los estudiantes de la Carrera de Ingeniería en Telecomunicaciones, es decir, que adopten un nuevo lenguaje de programación como lo es el Android como una herramienta para desarrollar temas de investigación a nivel de trabajos de titulación de pregrado, posgrado y de proyectos a través del Sistema de Investigación y Desarrollo (SINDE).

1.7. Metodología de Investigación.

La metodología empleada es de carácter cuantitativa, ya que a través de la revisión de literatura, cuyo alcance es exploratorio, descriptivo y explicativo. Es exploratorio, porque se examina un tema o problema no investigado ni estudiado en la Carrera de Ingeniería en Telecomunicaciones de la FETD en la Universidad Católica de Santiago de Guayaquil (UCSG). Es descriptivo, porque busca especificar propiedades y características de relevancia del fenómeno analizado. Finalmente, es explicativo, porque se pretende establecer las causas que surgen del fenómeno tal y como ocurre en su contexto actual.

CAPÍTULO 2: Estado del Arte de los Microcontroladores y sus Aplicaciones.

En el presente capítulo se describirá el estado del arte delos microcontroladores (uC) que hacen referencia al proyecto de grado previo a la obtención del grado de Ingeniero en Telecomunicaciones.

2.1. El Microcontrolador (uC).

Un microcontrolador (uC) o MCU es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Incluye en su interior las tres unidades funcionales principales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada y salida.(GRIDLING, 2007)

Según (Reyes, 2008) un microcontrolador es un circuito integrado, en cuyo interior posee toda la arquitectura de uncomputador, esto es CPU, memorias RAM, EEPROM, y circuitos de entrada y salida.Un microcontrolador de fábrica, no realiza tarea alguna, este debe ser programado paraque realice desde un simple parpadeo de un led hasta un sofisticado control de un robot. Unmicrocontrolador es capaz de realizar la tarea de muchos circuitos lógicos como compuertasAND, OR, NOT, NAND, conversores A/D, D/A, temporizadores, decodificadores, etc.,simplificando todo el diseño a una placa de reducido tamaño y pocos elementos.

En cambio para (Mandado Pérez, Menéndez Fuertes, Fernández Ferreira, & López Matos, 2007) un microcontrolador es un circuito integrado digital monolítico que contiene dos elementos de un procesador digital secuencia síncrono programable de arquitectura HARVARD o PRINCETON (Von Neumann), se le suele llamar también microcomputador integrado o empotrado *(Embedded Processor)* cuya orientación se orienta para tareas de control y comunicaciones. Adicionalmente expresa que por su pequeño tamaño, permite

empotrar un procesador programable en la mayoría de productos industriales para numerosas aplicaciones; y que poseen excelentes mecanismos de seguridad para su funcionamiento.

Al ser fabricados, la EEPROM (tipo de memoria no volátil) del microcontrolador no posee datos. Para que pueda controlar algún proceso es necesario generar y luego grabar en la EEPROM algún programa, el cual puede ser escrito en lenguaje ensamblador u otro lenguaje para microcontroladores; sin embargo, para que el programa pueda ser grabado en la EEPROM, debe ser codificado en sistema numérico hexadecimal que es finalmente el sistema que hace trabajar al microcontrolador cuando éste es alimentado con el voltaje adecuado y asociado a dispositivos analógicos y discretos para su funcionamiento(GRIDLING, 2007).

En consecuencia podemos decir que los microcontroladores son dispositivos electrónicos programables muy pequeños, que permiten realizar un sinnúmero de aplicaciones y con acelerado desarrollo tecnológico. Las Instituciones de Educación Superior (IES) tanto nacionales como extranjeras han adoptado a los microcontroladores tanto en investigaciones formativas como generativas.

2.2. Tipos de Microcontroladores.

Los tipos de microcontroladores son construidos por diversos fabricantes y existen variados modelos, dificultando así la tarea de escoger el uC apropiado, aun así, se debe seguir una jerarquía que dependeránen primer lugar de los **Principales fabricantes**, que dependen del volumen de ventas, en las figuras 2.1, 2.2, 2.3 y 2.4 se muestran tanto las ventas comerciales, crecimiento económico, cuota de mercado y capitalización bursátil respectivamente.

De las figuras mencionadas podemos deducir que existen cuatro fabricantes principales que son: Microchip, ST Microelectronics, Atmel Corporation y Motorola Semiconductors. En la figura 2.1 se puede apreciar al fabricante ST Microelectronics como el mayor vendedor comercialmente, aunque no se consideraron a Texas Instruments, Infineor Technologies, Intel Corp, Royal Philips Electronics, Analog Devices, National Semiconductors, Toshiba, NEC Corp., Mitsubishi, Hitachi Corp., debido a que los mismo producen microcontroladores con aplicaciones industriales no modificables.

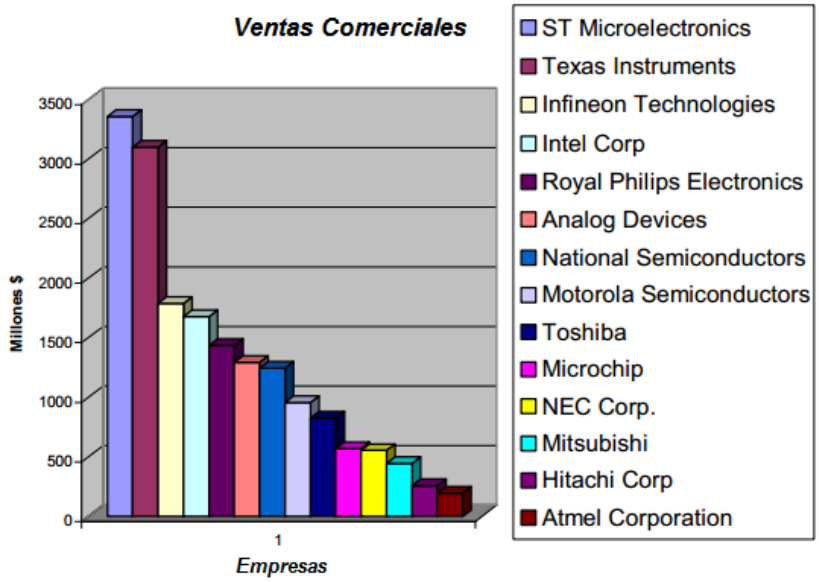


Figura 2. 1: Comportamiento de ventas comerciales de los microcontroladores.
Fuente: Los autores

Una situación similar ocurría con el crecimiento económico, cuota de mercado y de crecimiento bursátil de los fabricantes anteriormente mencionados. Es importante recalcar que los cuatro fabricantes principales producen microcontroladores para fines académicos, investigativos y son fáciles de programar en la actualidad bajo lenguaje de alto nivel, como lo es Pascal, Basic y C.

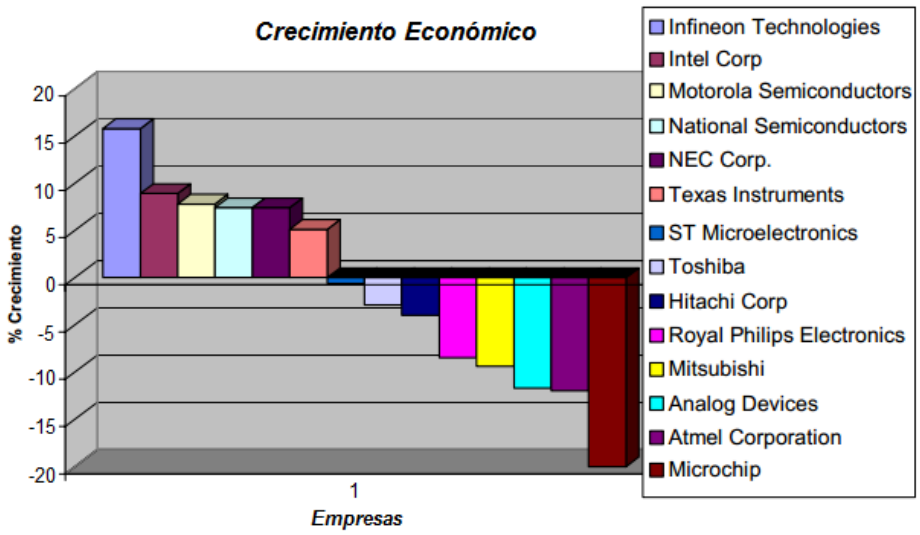


Figura 2. 2: Comportamiento del crecimiento económico de fabricantes de uC´s.
Fuente: Los Autores

Una vez descritas y elegidas los fabricantes se proceden a especificar las familias de los microcontroladores de acuerdo al ancho de las palabras, es decir, para uC's tanto de 8, 16 y 32 bits.

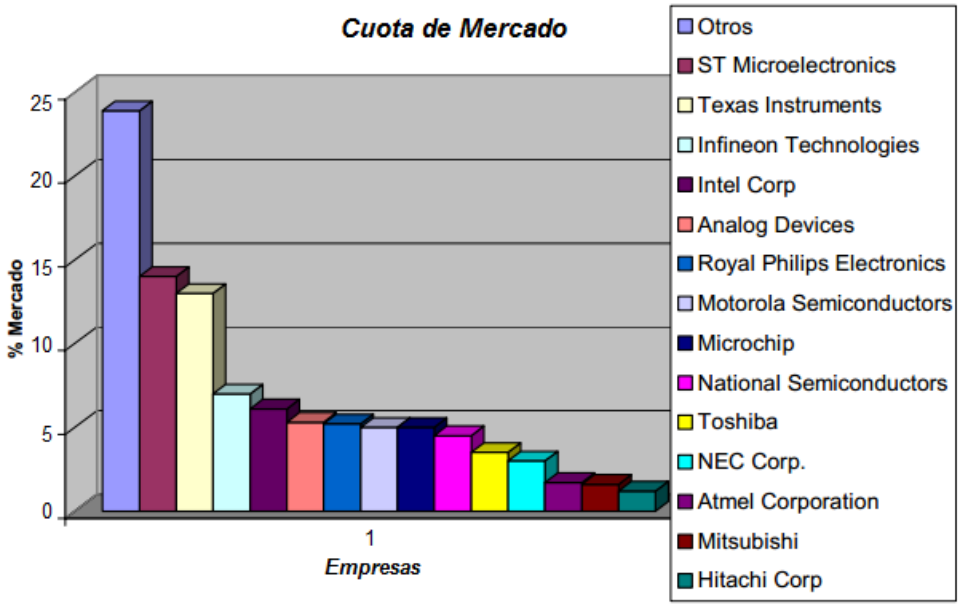


Figura 2. 3: Comportamiento de la cuota de mercado de fabricantes de uC´s.
Fuente: Los Autores

Posteriormente Primeramente mostraremosa través de varias tablas los microcontroladorescaracterísticos de 8 bits. Para lo cual la tabla 2.1 representa al fabricante de uC's Motorola Semiconductors.

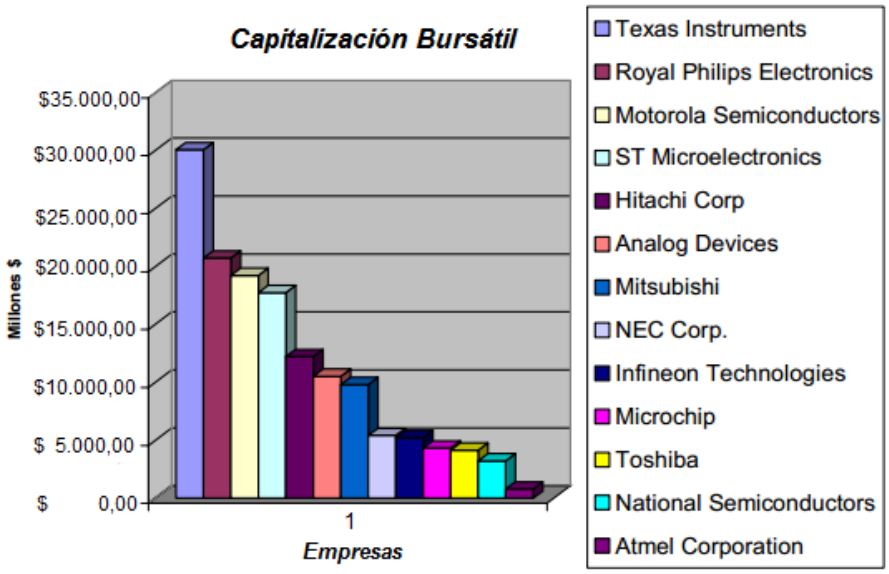


Figura 2. 4: Comportamiento de la capitalización bursátil de fabricantes de uC´s.
Fuente: Los Autores

2.3. Arquitectura Interna de los Microcontroladores.

En general la arquitectura interna de los microcontroladores es el conjunto de atributos que permiten un impacto directo en la ejecución del proceso que llevan a cabo(Mandado Pérez, Menéndez Fuertes, Fernández Ferreira, & López Matos, 2007). La arquitectura interna (ver figura 2.5) está formada por una unidad de control y una unidad operativa que se compone de un unidad de memoria y la unidad lógica aritmética (ULA).

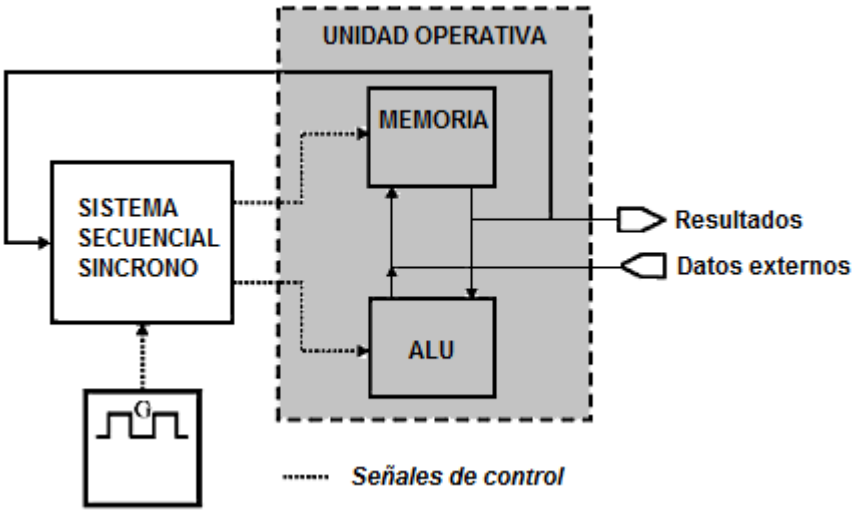


Figura 2. 5: Arquitectura Interna de uC´s.
Fuente: Los Autores

Los datos externos se memorizan en los registros adecuados de la unidad de memoria para presentarlos en secuencia en las entradas de la unidad aritmética y lógica, que los procesa y devuelve a la unidad de memoria los resultados parciales y finales obtenidos (Mandado Pérez, Menéndez Fuertes, Fernández Ferreira, & López Matos, 2007).

Los primeros Microcontroladores como el 8048 y el 8051, comercializado en 1980 por Intel, se implementaron con una unidad de control de arquitectura von Neumann (Mandado R. & Mandado P., 2007).La arquitectura dependerá de la familia de los Microcontroladores, aunque inicialmente todos adoptaron la arquitectura clásica de von Neumann, en la actualidad se impone la arquitectura Harvard.

El avance de la Microelectrónica propició el desarrollo de Microcontroladores con unidad de control de arquitectura Harvard, lo que permite a la memoria de datos y la de instrucciones disponga un número de bits diferentes en cada posición. (Mandado R. & Mandado P., 2007)

2.3.1. Arquitectura von Neumann.

La arquitectura von Neumann toma el nombre del matemático John von Neumann que propuso la idea de un ordenador con el programa almacenado *(stored-programcomputer)*. J. von Neumann trabajó en el equipo de diseñadores de la computadora ENIAC *(Electronic Numerical Integrator and Calculator)* diseñada en la Universidad de Pennsylvania durante la Segunda Guerra Mundial. (Valdés Pérez & Pallás Areny, 2007)

La arquitectura de von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control), tal y como se muestra en la figura 2.6.

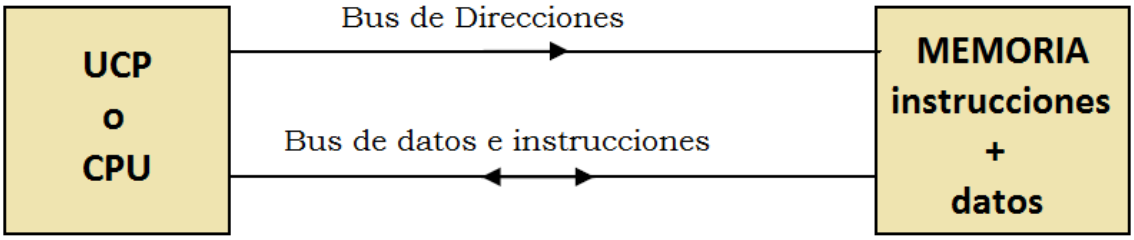


Figura 2. 6: Arquitectura Von Neumann.
Fuente: Los Autores

En la figura 2.6 se muestra un mismo bus de direcciones que localizan (direccionan) instrucciones y datos y que por un único bus de datos transitan tanto instrucciones como datos. La misma señal de control que emite la CPU para leer un dato, sirve para leer una instrucción. No hay señales de control diferentes para datos e instrucciones. Aunque debe quedar claro que la memoria ROM y RAM sirven para almacenar programas y datos respectivamente, el CPU no hace distinciones, ya que ROM y RAM forman un conjunto único (una memoria de lectura y escritura) (Valdés Pérez & Pallás Areny, 2007).

Los microcontroladores que utilizan la arquitectura von Neumann disponen de un solo bloque de memoria y de un bus de datos de 8 bits. Como todos los datos se intercambian por medio de estas 8 líneas (ver figura 2.7), este bus está sobrecargado, y la comunicación por sí misma es muy lenta e ineficaz.

La CPU puede leer una instrucción o leer/escribir datos de/en la memoria. Los dos procesos no pueden ocurrir a la vez puesto que las instrucciones y los datos utilizan el mismo bus.[1](Verle, 2010)

[1] Online recuperado de la página web: http://www.mikroe.com/chapters/view/79/capitulo-1-el-mundo-de-los-microcontroladores/

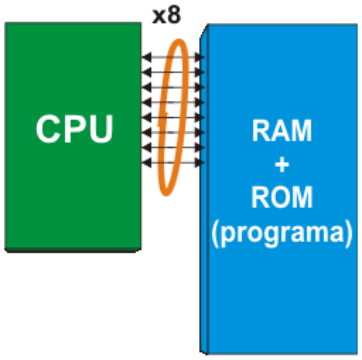


Figura 2. 7: Arquitectura Von Neumann para uC.
Fuente:http://www.mikroe.com/chapters/view/79/capitulo-1-el-mundo-de-los-microcontroladores/

2.3.2. Arquitectura Harvard.

El término arquitectura Harvard se debe al nombre del lugar donde Howard Aiken diseñó los ordenadores Mark I, II, III y IV. Estos ordenadores fueron los primeros en utilizar memorias separadas para instrucciones y datos, una concepción diferente al ordenador de programa almacenado. (Valdés Pérez & Pallás Areny, 2007) La Arquitectura Harvard fue desarrollada en 1970 para solucionar los problemas de velocidad de procesamiento que presentaba la Arquitectura Von Neumann.(Staff, 2011)

Para el caso de la arquitectura tipo Harvard, se tienen dos memorias independientes (tanto para datos como instrucciones) conectadas al CPU a través delos buses de datos y de dirección, tal y como se muestra en la figura 2.8. Estos tipos de buses (datos y dirección) son independientes y pueden ser de distintos anchos, lo que permite que la CPU pueda acceder simultáneamente a ambasmemorias, consiguiendo que las instrucciones se ejecuten en menos ciclos de reloj.

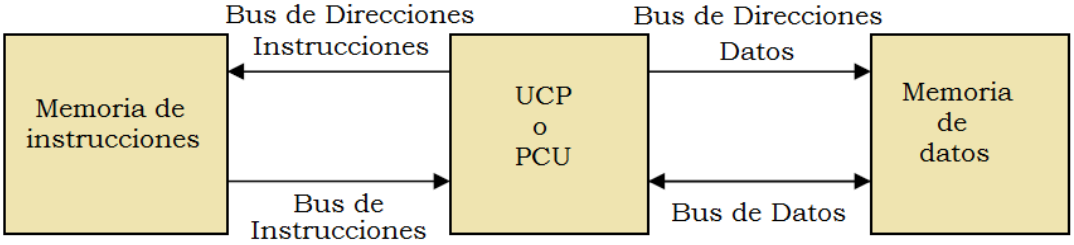


Figura 2. 8: Arquitectura Harvard.
Fuente: Los Autores

Como se aprecia en la figura 2.8 la arquitectura Harvard dispone de memorias independientes, es decir, la primera, la Memoria de Instrucciones y la segunda, la Memoria de Datos; y se comunican mediante sus respectivos sistemas de buses, permitiendo ejecutar operaciones de acceso (lectura o escritura) de manera simultánea en ambas memorias. Según (Caprile, 2013), esto duplica efectivamente el espacio accesible de memoria y de velocidad teórica de ejecución, ya que la unidad de proceso puede operar directamente sobre dos buses a la vez y por ejemplo, leer la instrucción y el dato al mismo tiempo.

Los microcontroladores que utilizan esta arquitectura disponen de dos buses de datos diferentes. Uno es de 8 bits de ancho y conecta la CPU con la memoria RAM, el otro consiste en varias líneas (12, 14 o 16) y conecta a la CPU y la memoria ROM, tal y como se muestra en la figura 2.9. [2](Verle, 2010)

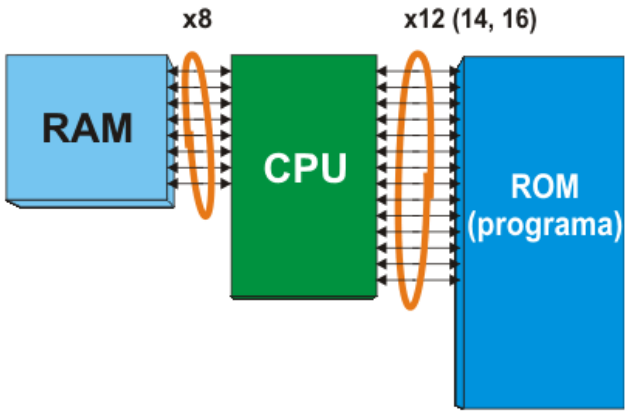


Figura 2. 9: Arquitectura Harvard para uC.
Fuente: Los Autores

2.4. Arquitecturas CISC y RISC.

Actualmente las arquitecturas CISC *(Complex Instruction Set Computer)* y RISC *(Reduced Instruction Set Computer)* son dos modelos generales de ordenadores que se muestran en la figura 2.10, desde el punto de vista de la concepción de su repertorio de instrucciones, lo cual repercute directamente

[2] Online recuperado de la página web: http://www.mikroe.com/chapters/view/79/capitulo-1-el-mundo-de-los-microcontroladores/

sobre la arquitectura de la CPU. Un ordenador CISC tiene un repertorio de instrucciones complejo y un ordenador RISC tiene un repertorio de instrucciones reducido. (Valdés Pérez & Pallás Areny, 2007)

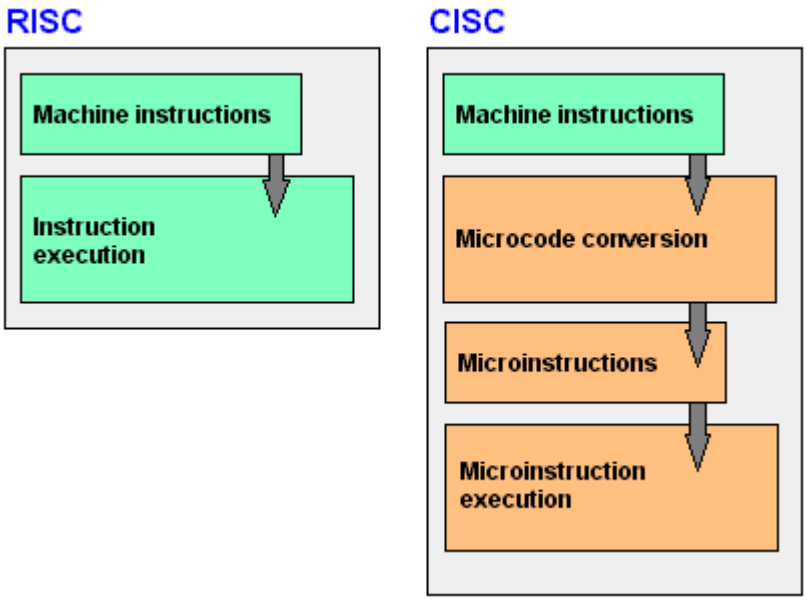


Figura 2. 10: Arquitectura RISC y CISC.
Fuente: http://rcmcomputointegrado.blogspot.com/2012/03/arquitectura-risc-y-cisc.html

Al parecer los microprocesadores y los microcontroladores, la tendencia inicial fue proveerlos de un repertorio de instrucciones lo más potente posible, de modo que el modelo predominante fue el CISC (Valdés Pérez & Pallás Areny, 2007). Siempre ha existido una gran polémica en torno a cuál de estas dos plataformas esmejor. Tal vez podamos considerar inútil estar hablando sobre esto, pero es interesante comprender la diferencia entre estas dos plataformas para entender varios aspectosde los procesadores modernos.[3](Durán Rodríguez, 2006)

Como ya se mencionó CISC se fundamentan de un conjunto de instrucciones complejas, donde la ejecución es realizada por una unidad de control,la cual es diseñada para generar señales de control. Un conjunto de instrucciones CISC habitualmente contiene varias instrucciones de diferentes tamaños (número de bits para especificar la instrucción que varíade una

[3] Online recuperado de la página web: http://galeon.com/ligrosdetec/reparar3.pdf

instrucción a otra), diferentes formatos, y que usualmente requieren más de un ciclo de reloj para su ejecución.

También se mencionó que un conjunto de instrucciones RISC incluye menos instrucciones, y a la vez son instrucciones más simples, y requieren de una unidad de control, un sistema de pipa de procesamiento más simple, lo que nos permite un procesamiento paralelo de instrucciones, un mayor número de registros, diseños más simples y reducidos, y una mayor frecuencia de reloj.

2.5. Aplicaciones de los Microcontroladores.

Los Microcontroladores actualmente son muy utilizados, tienen una considerable variedad de aplicaciones, tales como: procesos industriales, sector automotriz, aviación, electrodomésticos, equipos médicos, telecomunicaciones, robótica, etc. Todas las aplicaciones mencionadas (ver figura 2.11) se deben a dos aspectos fundamentales, el primero, es que el proceso de diseño se lo realiza exclusivamente en un software (Basic o C), el segundo, se refiere al hardware que consiste en seleccionar el Microcontrolador apropiado. Esto permite que el mantenimiento y la modificación de los diseños se hagan de una forma muy eficiente.

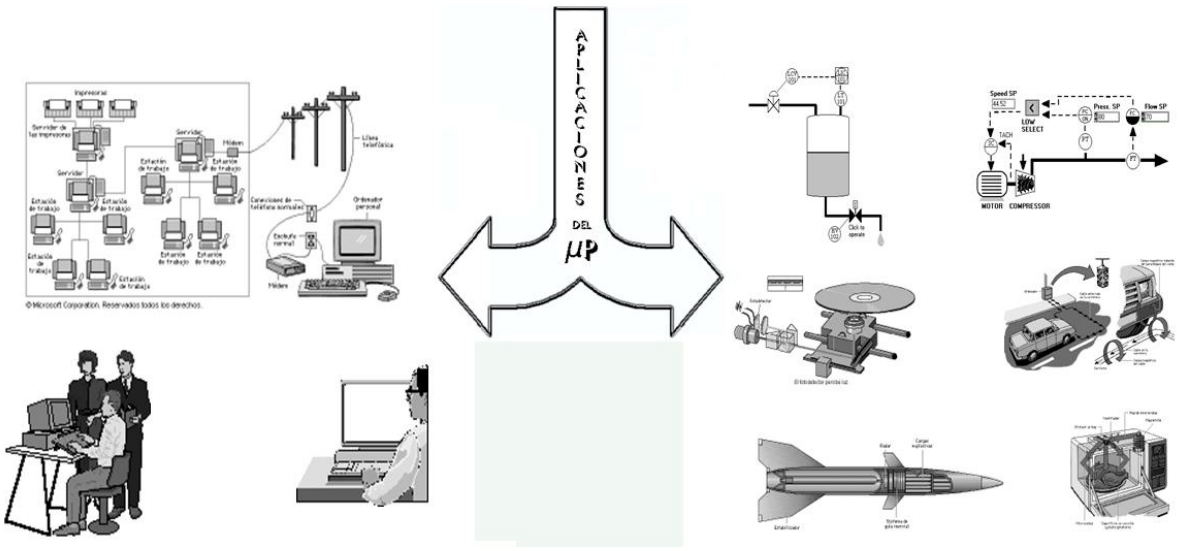


Figura 2. 11: Aplicaciones de Microcontroladores
Fuente: Los Autores

2.6. La familia de los Microcontroladores.

Actualmente existen en el mercado varias marcas reconocidas como las principales, dadas sus características, difusión y usos en productos de consumo masivo. Entre ellas están: Motorola, Intel, Philips, National, Toshiba, Hitachi, Siemens, Zilog y Microchip (ver tabla 2.1), está última se describirá en profundidad en el acápite 2.7.

Tabla 2. 1: Fabricantes y modelos de microcontroladores

Algunos Fabricantes	Algunos modelos de microcontroladores
MICROCHIP	12XX, 14XX, 16XX, 17XX, 18XX, 32XX.
INTEL	8048, 8051, 80c196, 80186, 80188, 80386EX.
MOTOROLA	68HC05, 68HC08, 68HC11, 68HC12, 68HC16 y más.
HITACHI	H8/300, H8/300L, H8/500, H8/300H.
TOSHIBA	TLCS-47, TLCS-870, TLCS-900.
PHILIPS	80C51
SIEMENS	C500, C166
ZILOG	Z8, Z86XX
TEXAS	TMS370

Fuente: http://es.scribd.com/doc/97201377/1-Introduccion-a-Los-Micros

A continuación se describirán brevemente cada una de las familias que se mostraron en la tabla 2.1, y como se indicó anteriormente en el siguiente acápite se describirá más detallado los microcontroladores PIC.

Familia Intel 8051

La familia de microcontroladores de 8 bits (ver figura 2.12 y tabla 2.2) como INTEL 8051, contiene varias referencias, cada una de ellas acondicionada para aplicaciones específicas. Todas las versiones existentes tienen la misma CPU, memoria RAM, temporizadores, puertos paralelos y

entradas/salidas de tipo serial. Éste contiene 4 Kbytes de ROM[4] programadas durante el proceso de fabricación del circuito integrado.

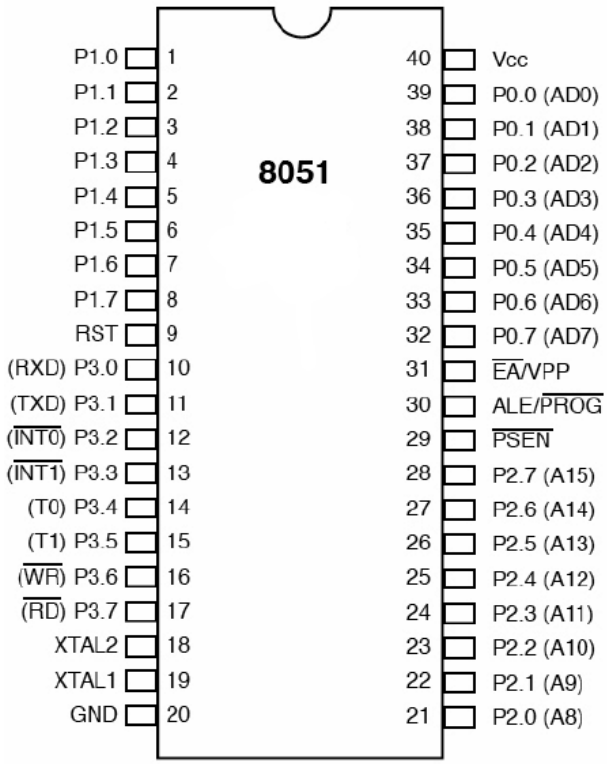


Figura 2. 12: Diagrama de pines del Microcontrolador Intel 8051.
Fuente:http://www.ustudy.in/ece/mpmc/u2

El microcontrolador INTEL 8751, la ROM se reemplaza por una memoria EPROM, que tiene la ventaja de ser programada y borrada (con luz ultravioleta) las veces que sean necesarias.

Tabla 2. 2: Modelos de microcontroladores INTEL de 8 bits.

Referencia	Memoria ROM	Memoria RAM	Timers
8051	4 K	128	2
8052	8 K	256	3
8031	Externa	128	2
8032	Externa	256	3
8751	4 K (EPROM)	128	2

Fuente: Los Autores

[4] ROM, *Read Only Memory* (Memoria de solo lectura)

Familia Motorola6805

Es una de las familias más empleadas en el mundo y optimizada para diferentes aplicaciones de control, con ausencia de procesar datos, son utilizados como dispositivos electrónicos para juguetes, equipos de video, impresoras, módems, sector automotriz y electrodomésticos. Como la tecnología de los dispositivos electrónicos evoluciona, también salen al mercado nuevos modelos que han sustituido a los ya mencionados.

Este tipo de microcontrolador es similar a cualquier dispositivo básico, es decir, que cuentan con una misma CPU de 8 bits, memorias RAM y ROM, puertos de E/S y temporizadores; aunque otros tienen adicionales, tales como: puertos seriales, convertidores A/D y memorias EEPROM o EPROM. En la tabla 2.3 se muestran varios modelos de la familia de microcontroladores MOTOROLA 6805.

Tabla 2. 3: Modelos de microcontroladores MOTOROLA 6805.

Referencia	Memoria ROM	Memoria RAM	Timers	Otros
68HC05C4	4 K	176	1	
68HC0508	8 K	176	1	
68HC05C2	2 K	176	1	
68HC705C4	4 K (Eprom)	176	1	
68HC705K1	504	32 y 64	1	
68HC05BM	2 K	176	1 mejorado	Conversores A/D

Fuente: Los Autores

Familia Microchip 16CXX

Los Microcontroladores PIC de Microchip tiene aproximadamente 11 familias y en cada una hay disponibles varios componentes. Se describe brevemente la familia PIC16CXX (posteriormente se describirá la familia del PIC apropiado para el robot controlado por bluetooth).Para el PIC16CXX cuenta con una variedad de componentes de diferentes tamaños de memoria,

diferentes velocidades, diferentes tipos de encapsulado y diferentes números de pines de E/S.

Tabla 2. 4: Modelos de microcontroladores PIC16CXX.

Referencia	Memoria ROM	Memoria RAM	Timers	Otros
16C54	512	32	12	
16C57	2 K	80	20	
16C61	1 K	48	12	Interrupción
16C71	1 K	48	12	Conversores A/D
16C84	1 K (EEPROM)	48	12	
16C509	1 K	48	6	Encapsulado de 8 pines

Fuente: Los Autores

El conjunto de instrucciones es de sólo 35, por eso se dice que es un microcontrolador de tipo RISC a diferencia de las anteriores familias que utilizan la tecnología CISC.

2.7. Los Microcontroladores PIC.

Los Microcontroladores PIC*(Peripheral Interface Controller)*es una de las tantas familias disponibles a nivel mundial de la microelectrónica, en este caso los PIC son desarrollados y fabricados porla empresa ***Microchip Technologies Inc.***, los cuales cuentan con unatecnología tipo RISC *(Reduced Instruction Set Computer)* y poseen en suarquitectura interna características especiales que varían según el modelo dePIC que deseamos utilizar.

Aunque PIC actualmente no es un acrónimo, el nombre completo es**PICmicro**, cuyo significado en español es ***controlador de interfaz periférico***.El primer PIC fue diseñado para ser implementado en la nuevaCPU de 16 bits, denominada CP16000. Fue considerado una buena CPU, aunque tenía malas prestaciones de entrada y salida, por lo que en 1975 se desarrolló el PIC de 8 bits, que permitió incrementar el rendimiento del sistema.

Según Miguel García Barba (2012), el PIC utilizaba microcódigo simple almacenado enROM para realizar estas tareas; y aunque el término no se usaba por aquel entonces, se trata de un diseño RISC que ejecuta una instrucción cada 4 ciclos del oscilador.En 1985 la división de microelectrónica de ***General Instrument*** se separa como compañía independiente que es incorporada como filial (el 14 de diciembre de 1987 cambia el nombre a ***Microchip Technology*** y en 1989 es adquirida por un grupo deinversores) y el nuevo propietario canceló casi todos los desarrollos, que para esas fechas la mayoría estaban obsoletos. El PIC, sin embargo, se mejoró con EPROM para conseguir un controlador de canal programable.(García Barba, 2012)

Lo ya mencionado nos indica que estos dispositivos se asemejan a un computador de tamaño muy reducido, debido a que cuentan con los mismos recursos, es decir, tienen disponible la memorias tanto de programa, RAM, y datos, así como puertos E/S (entrada o salida), temporizadores, convertidores A/D, comparadores,USART[5],comunicación serie I2C, entre otros tal y como se ilustra en la figura 2.13.

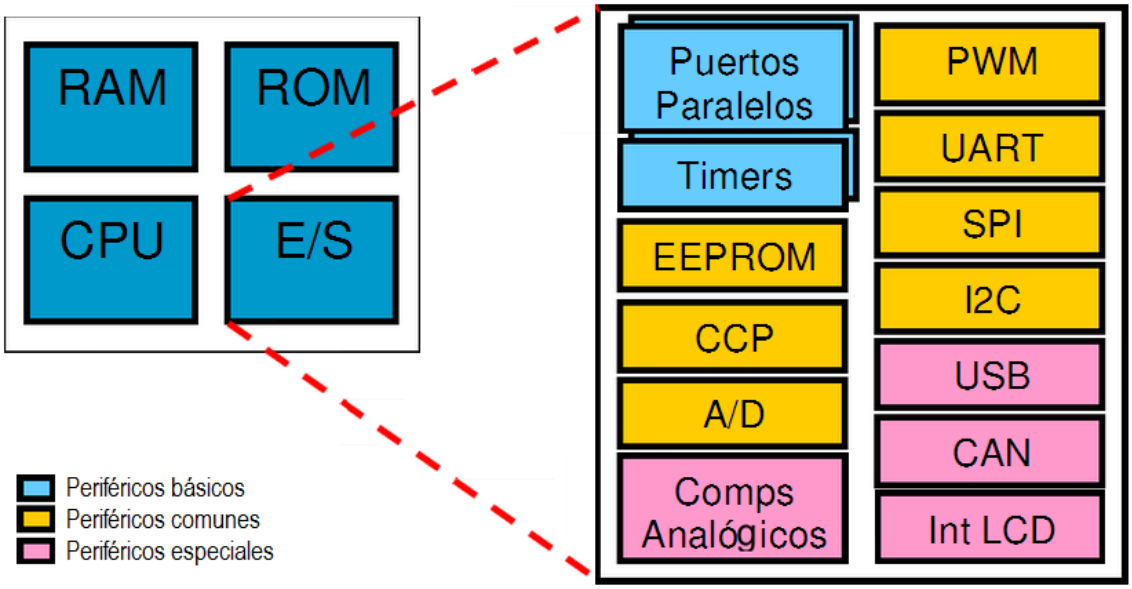


Figura 2. 13:Recursos de los puertos de E/S
Fuente: http://es.scribd.com/doc/97201377/1-Introduccion-a-Los-Micros

[5] USART (Universal Synchronous/Asynchronous Receiver/Transmitter, Transmisor/Receptor Universal Síncrono/Asíncrono)

Todas las características ya mencionadas del dispositivo PIC, sería en otras palabras el corazón del circuito. Es decir, que eluC (microcontrolador)se encargade transmitir información de un circuitoelectrónico, de acuerdo a rutina o instrucciones,que permiten funciones de control, todo esto se logra bajo un lenguaje de programación de alto nivel, tales como: Pascal, Basic y C; para el presente trabajo se utilizará ellenguaje Basic para microcontroladores PIC.

Para Carlos Reyes (2008): un microcontrolador (ver figura 2.14) es un circuito integrado, en cuyo interior posee toda la arquitectura de uncomputador, esto es CPU, memorias RAM, EEPROM, y circuitos de entrada y salida.Un microcontrolador de fábrica, no realiza tarea alguna, este debe ser programado paraque realice desde un simple parpadeo de un led hasta un sofisticado control de un robot. Unmicrocontrolador es capaz de realizar la tarea de muchos circuitos lógicos como compuertasAND, OR, NOT, NAND, conversores A/D, D/A, temporizadores, decodificadores, etc.,simplificando todo el diseño a una placa de reducido tamaño y pocos elementos.

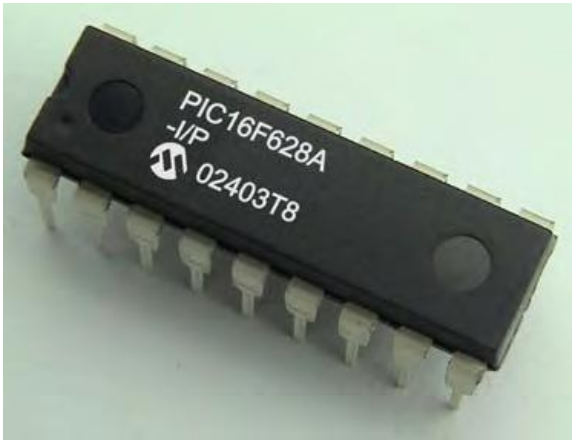


Figura 2. 14: El microcontrolador PIC de la gama 16F628A
Fuente: (Reyes, 2008)

Se debe considerar la arquitectura interna del uC a programar y aunque parezca difícil la tarea, los lenguajes de alto nivel como Basic para microcontroladores PIC la hacen de manera fácil.Obviamente al momento de programar microcontroladores PIC. Debemos considerar previamente el estudio del diseño del hardware que hará quenuestros proyectos se pongan en marcha.

También es imprescindible la identificación de los pines del uC y que función deberá ejecutar; para el caso de los puertos *IN/OUT*disponibles del uC, las funciones que deberán cumplir, las mismas son establecidas antes de ser programados en Basic, éstos son especificados por el usuario (programador)a conveniencia y de manera independiente. También podemos destinar un puerto completo

2.8. Gamas del microcontrolador PIC de 8 bits.

Como se ha indicado anteriormente la familia de los microcontroladores PIC dispone de diferentes modelos y encapsulados: gama enana, gama baja, gama media, gama alta y la gama mejorada.

Sin embargo, cada una de las versiones es construida alrededor de una arquitectura común, un repertorio mínimo de instrucciones y un conjunto de opciones muy apreciadas, como el bajo consumo y el amplio margen del voltaje de alimentación. A continuación se describirá cada una de las gamas.(Valverde Villarán, 2005)

2.8.1. Gama enana: instrucciones de 8 bits.

Los PIC que pertenecen a esta gama, son nuevos dispositivos electrónicos que han acaparado al mercado mundial, especialmente la parte académica para realizar investigaciones formativas y generativas. Los PIC12FXX tienen como característica principal su tamaño reducido, disponen de 8 pines (patitas), cuya alimentación está comprendida entre 2.5 V_{DC} y 5.5 V_{DC} y una corriente de 2 mA cuando funcionan con 5 V_{DC} a una frecuencia de oscilación de 4 MHz.

En la figura 2.15 se muestran las conexiones de los PIC10FXX (8 pines) y PIC12FXX (8 pines). Adicionalmente, los dos con instrucciones cuyo formato es de 12 o 14 bits y un repertorio de 33 o 35 instrucciones.

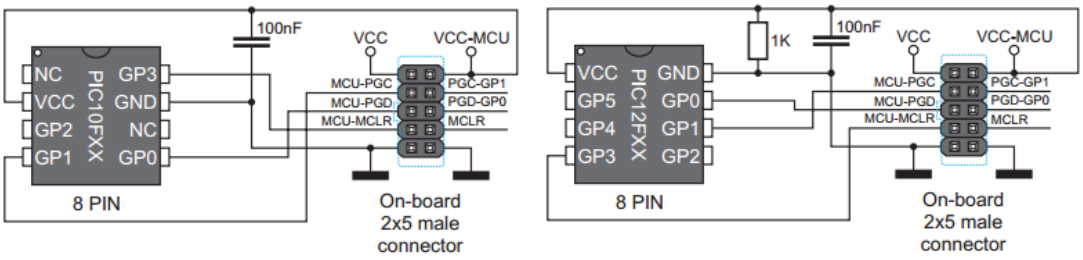


Figura 2. 15: Microcontroladores PIC10FXX y PIC12FXX
Fuente: http://www.mikroe.com/downloads/get/10/

En la figura 2.16 se puede apreciar los PIC12F508/5009 y PIC12F675 que tienen disponibles 6 pines como líneas de E/S para comunicarse con periféricos, estos modelos cuentan con un oscilador RC interno, es decir, que no requieren de un oscilador externo.

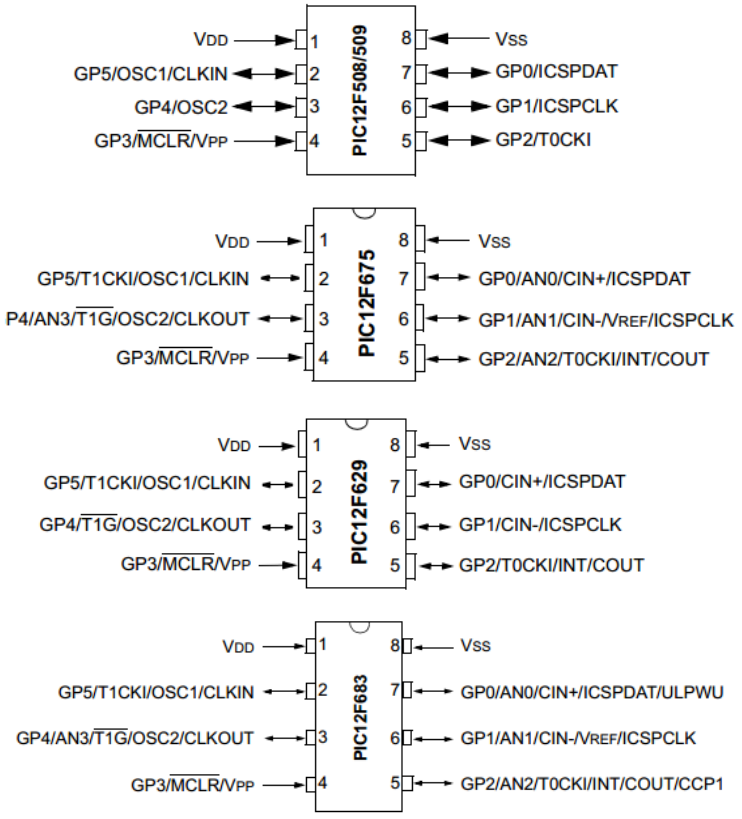


Figura 2. 16: Microcontroladores PIC12F508/5009 y PIC12F675
Fuente:http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1001&mid=10&lang=en&pageId=74

En la tabla 2.5 se muestran las características principales de la familia o gama enana (8 pines) de los PIC12FXXX, que cuentan con memoria de

programa (flash) y memoria de datos (SRAM y EEPROM), a excepción del PIC12F508 y PIC12F509 que no disponen de memoria EEPROM. Finalmente, la gama enana a futuro serán repotenciados por MICROCHIP.

Tabla 2. 5: Características de los modelos de microcontroladores PIC12FXXX.

Modelo	Memoria Programa	Memoria Datos		Líneas E/S	Pines
	Flash	SRAM (bytes)	EPROM (bytes)		
PIC12F508	512	25	-	6	8
PIC12F509	1 K	41	-	6	8
PIC12F629	1 K	64	128	6	8
PIC12F675	1 K	64	128	6	8
PIC12F683	2 K	128	256	6	8

Fuente:http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1001&mid=10&lang=en&pageId=74

2.8.2. Gama baja: instrucciones de 12 bits.

A pesar de ser un PIC de 12 bits, la serie de la gama baja posee recursos limitados, aunque dispone de mejores prestaciones en relación al coste.Existen diversas versiones o modelos de PIC16F7XX que se encuentran encapsuladas entre18 pines y 28 pines, El voltaje de alimentación admite un valor muy flexible comprendido entre 2 y 6,25 V, lo cual posibilita el funcionamiento mediante pilas corrientes teniendo en cuenta su bajo consumo (menos de 2 mA a 5 V y 4 MHz) (Valverde Villarán, 2005). Tienen un repertorio de 33 instrucciones cuyo formato consta de 12 bits. No admitenningún tipo de interrupción y la pila sólo dispone de dos niveles.

La gama baja de los PIC encuadra nueve modelos fundamentales en la actualidad. La memoria de programa puede contener 512, 1 k. y 2 k palabras de 12 bits, y ser de tipo ROM, EPROM aunque también hay modelos con memoria OTP. Sólo disponen deun temporizador (TMR0), un repertorio de 33 instrucciones y un número de pines para soportar las E/S comprendido entre 12 y 20(Valverde Villarán, 2005).

Y por otro lado, conviene nombrar dos restricciones importantes de la gama baja y es que la pila sólo dispone de dos niveles, lo que supone no poder encadenar más de dos subrutinas y además no admiten interrupciones. En la figura 2.17 se muestra el encapsulado del PIC16F720/721.

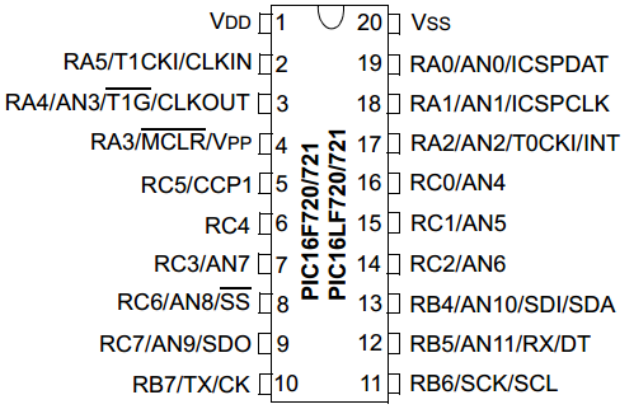


Figura 2. 17: Microcontroladores PIC16F720/721.
Fuente:http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1001&mid=10&lang=en&pageId=74

En la tabla 2.6 se muestran las características principales de los microcontroladores de la gama baja de los modelos PIC16F7XX.

Tabla 2. 6: Características de los modelos PIC16F7XX.

Modelo	Memoria Programa	Memoria Datos		Líneas E/S	Pines
	Flash	SRAM (bytes)	EPROM (bytes)		
PIC16F72	2 K	128	-	22	28
PIC16F913	4 K	256	256	24	28
PIC16F916	8 K	352	256	24	28
PIC16F818	1 K	128	128	16	18
PIC16F819	2 K	256	256	16	18

Fuente:
http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1001&mid=10&lang=en&pageId=74

2.8.3. Gama media: instrucciones de 14 bits.

La gama media de los microcontroladores PIC es la más completa, con respecto a la gama enana y baja, haciéndolos adecuados para aplicaciones complejas, los mismos tienen encapsulados entre 18 pines y 68 pines,

permitiendo cubrir o integrar la mayor cantidad de periféricos. En la figura 2.18 se muestra el PIC16F884/887 de 40 pines y en la tabla 2.7 se muestran las características de los PIC16F8XX.

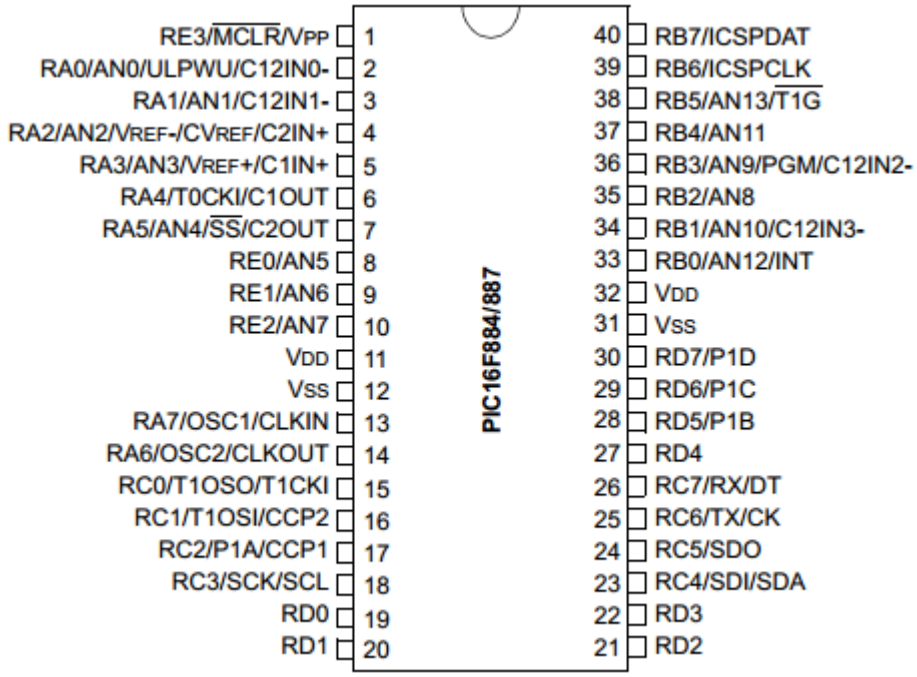


Figura 2. 18: Microcontroladores PIC16F884/887.
Fuente:http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1001&mid=10&lang=en&pageId=74

De acuerdo a Valverde (2005), los componentes de la gama media añaden nuevas prestaciones a las que poseían los de la gama baja, haciéndoles más adecuados en las aplicaciones complejas, es decir, que permiten realizarinterrupciones, poseen comparadores de magnitudes analógicas, convertidores A/D, puertos serie y diversos temporizadores. Algunos modelos disponen de una memoria de instrucciones del tipo OTP que resulta mucho más económica en la implementación de prototipos y pequeñas series. Otros en cambio disponen de una memoria de instrucciones tipo EEPROM, que al poder ser borradas eléctricamente, son mucho más fáciles de reprogramar que las EPROM.(Valverde Villarán, 2005)

Tabla 2. 7: Características de los modelos PIC16F7XX.

Modelo	Memoria Programa	Memoria Datos	Líneas E/S	Pines

	Flash	SRAM (bytes)	EPROM (bytes)		
PIC16F882	2 K	128	128	24	28
PIC16F883	4 K	256	256	24	28
PIC16F884	4 K	256	256	35	40
PIC16F886	8 K	368	246	24	28
PIC16F887	8 K	368	256	35	40

Fuente:http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1001&mid=10&lang=en&pageId=74

Este tipo de familia dispone de un repertorio de 35 instrucciones de 14 bits cada uno, estos permiten ser compatibles con los PIC´s de la gama baja, es decir, que cada uno de los modelos de la gama media tiene todos los recursos de los PIC´s de 8 bits. Adicionalmente, se pueden añadir subrutinas a través de interrupciones y pila (8 niveles).

2.8.4. Gama alta y gama mejorada: instrucciones de 16 bits.

En la actualidad, la gama alta y la mejorada estánconformadas por los modelos PIC17 y PIC18. Según lo indicado por Valverde (2005), son dispositivos que responden a microcontroladores de arquitectura abierta pudiéndose expansionar en el exterior al poder sacar los buses de datos, direcciones y control. Así se pueden configurar sistemas similares a los que utilizan los microprocesadores convencionales, siendo capaces de ampliar la configuración interna del PIC añadiendo nuevosdispositivos de memoria y de E/S externas. Esta facultad obliga a estos componentes a tener un elevado número de patas comprendido entre 40 y 44. Admiten interrupciones, poseen puerto serie, varios temporizadores y mayores capacidades de memoria, que alcanza los 128 K palabras en la memoria de instrucciones y 1500 bytes en la memoria de datos.

En la actualidad, la empresa Microchip producediferentes modelos de PIC´s de 32 bits(microcontroladores) y dspic´s de 16 bits con gran potencia y velocidad, que son utilizados en aplicaciones robustas y complejas, estos

vienen diseñados desde 28 pines hasta 84 pines, operan con un reloj de 40 MHz y tienen una memoria de código de 128 K palabras.

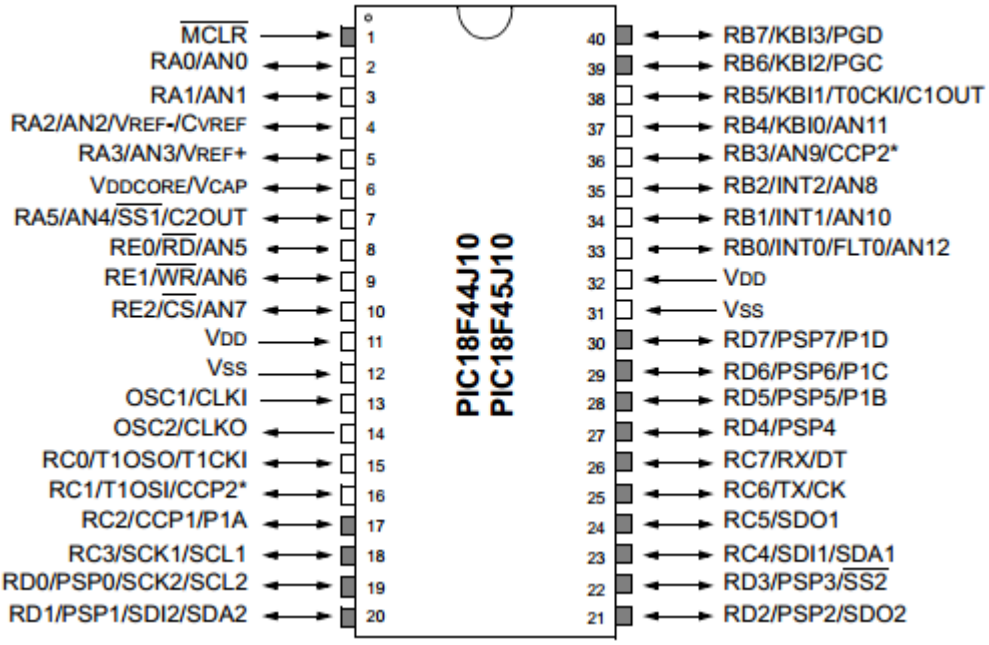


Figura 2. 19: Microcontroladores PIC18F44J10 y PIC18F45J10.
Fuente:
http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1001&mid=10&lang=en&pageId=74

En la figura 2.19 se muestra al PIC18F44J10/45J10 el mismo que alcanza 58 instrucciones de 16 bites en el repertorio y en la tabla 2.9 se muestra las características más destacadas de los modelos PIC18F24J10, PIC18F25J10, PIC18F44J10 y el PIC18F45J10.

Tabla 2. 8: Características de los modelos PIC18F24J10/25J10/44J10/45J10.

Modelo	Memoria Programa		Memoria Datos	Líneas E/S	Pines
	Flash (bytes)	Single-Word Instructions	SRAM (bytes)		
PIC18F24J10	16 K	8192	1024	21	28

PIC18F25J10	32 K	16384	1024	21	28
PIC18F44J10	16 K	8192	1024	32	40
PIC18F45J10	32 K	16384	1024	32	40

Fuente:http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1001&mid=10&lang=en&pageId=74

En el siguiente acápite se describirá el microcontrolador PIC16F887, el mismo que se utiliza en el presente trabajo de titulación.

2.9. El Microcontrolador PIC16F887.

El PIC16F887 corresponde a la gama media de los PIC´s, siendo este componente muy parecido a otros que se encuentran disponibles en la mayoría de los microcontroladores modernos, es decir, PIC y dsPIC de 32 bits.En cuanto al costo, es relativamente económico, con rango amplio de aplicaciones, alta calidad y disponibilidad, lo que lo convierte en una solución perfecta para el control automatizado de procesos industriales, de control de máquinas, para medir variables de procesos etc.

En la figura 2.20 se muestra el PIC16F887 de 40 pines del tipo PDIP[6], muy utilizados en prácticas de laboratorio por su facilidad de funcionar sobre un protoboard. Y la figura 2.21 nos muestra el PIC16F887 de 44 pines del tipo QFN[7], este se lo utiliza en aplicaciones avanzadas.

[6] PDIP: Plastic Dual In-Line Package, Encapsulado de plastic dual en línea.
[7] QFN: Quad Flat No-Lead Package, Encapsulado plano sin plomo y cuádruple.

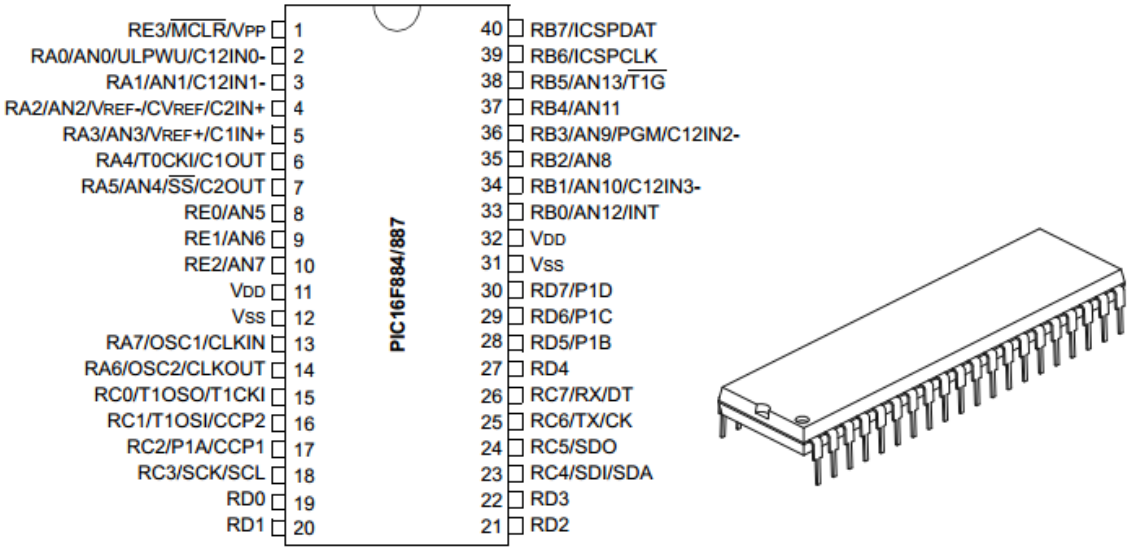


Figura 2. 20: Microcontroladores PIC16F884/887 de 40 pines - PDIP.
Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/41291G.pdf

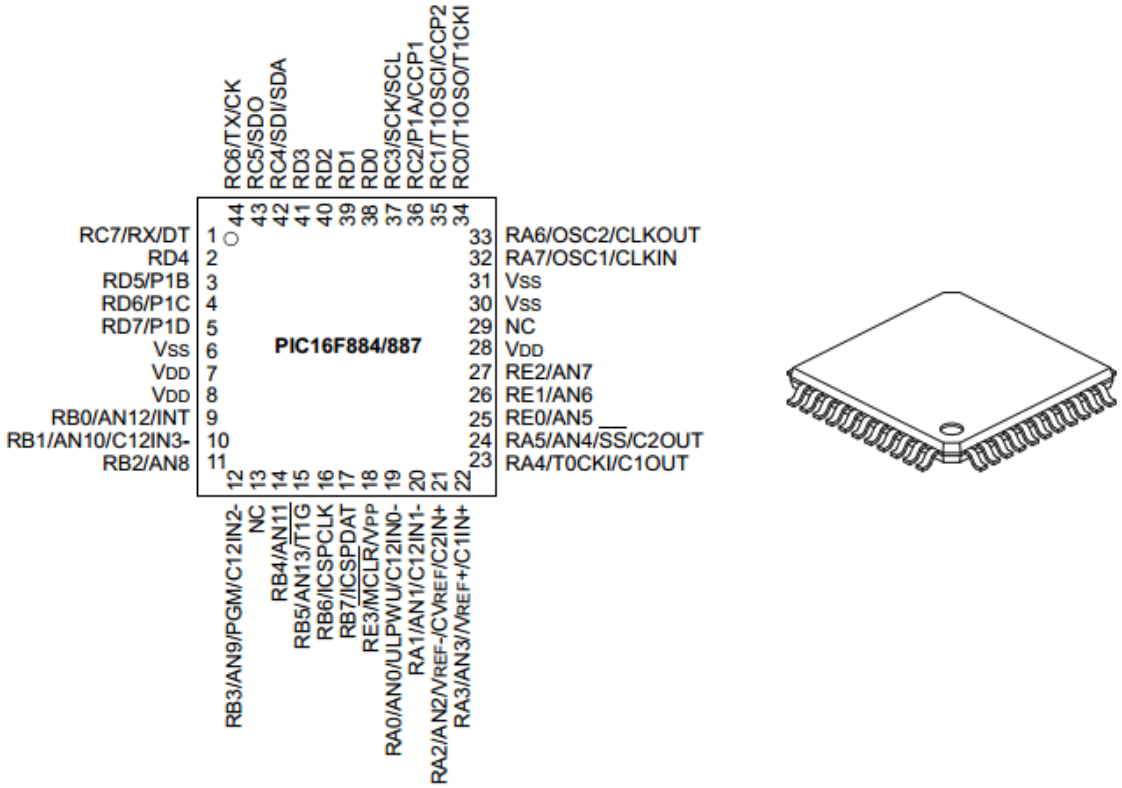


Figura 2. 21: Microcontroladores PIC16F884/887 de 44 pines -QFN.
Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/41291G.pdf

2.9.1. Características básicas del PIC16F887.

La figura 2.22 muestra un diagrama de bloques de las características del PIC16F887, que básicamente se asemeja a la arquitectura Harvard.

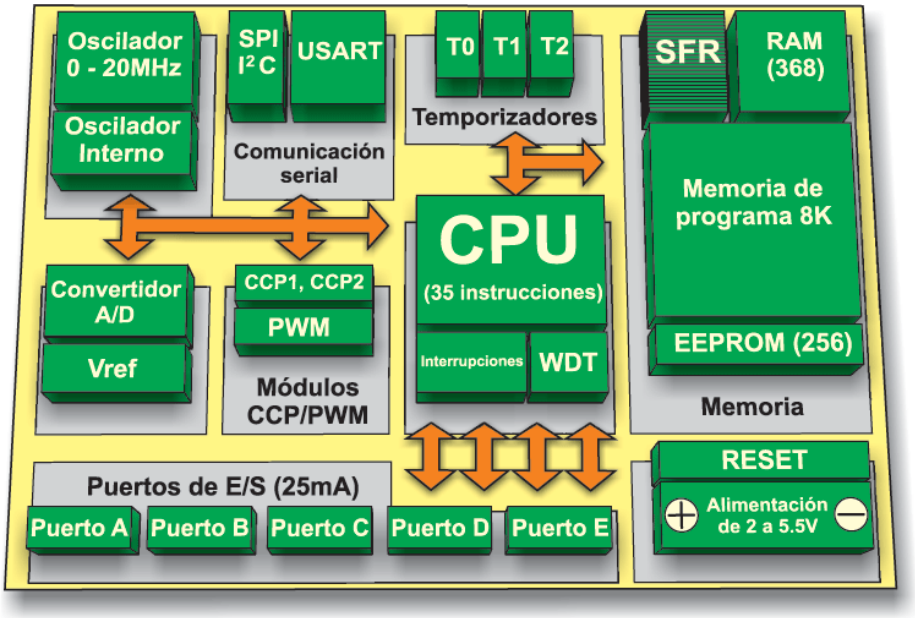


Figura 2. 22: Diagrama de bloques de Microcontroladores PIC16F884/887. Fuente:(Verle, 2010), disponible en línea:http://www.mikroe.com/chapters/view/86/libro-de-a-progamacion-de-los-microcontroladores-pic-en-basic-capitulo-3-microcontrolador-pic16f887/#c3v3

A continuación se establecen brevemente las características principales:

- Arquitectura RISC
  - ✓ El microcontrolador cuenta con solo 35 instrucciones diferentes
  - ✓ Todas las instrucciones son uni-ciclo excepto por las de ramificación
- Frecuencia de operación 0-20 MHz
- Oscilador interno de alta precisión
  - ✓ Calibrado de fábrica
  - ✓ Rango de frecuencia de 8MHz a 31KHz seleccionado por software
- Voltaje de la fuente de alimentación de 2.0V a 5.5V
  - ✓ Consumo: 220uA (2.0V, 4MHz), 11uA (2.0 V, 32 KHz), 50nA (en modo de espera)
- Ahorro de energía en el Modo de reposo
- Brown-out Reset (BOR) con opción para controlar por software
- 35 pines de entrada/salida

- ✓ Alta corriente de fuente y de drenador para manejo de LED
- ✓ Resistencias pull-up programables individualmente por software
- ✓ Interrupción al cambiar el estado del pin

- Memoria ROM de 8K con tecnología FLASH
  - ✓ El chip se puede re-programar hasta 100.000 veces
- Opción de programación serial en el circuito
  - ✓ El chip se puede programar incluso incorporado en el dispositivo destino
- 256 bytes de memoria EEPROM
  - ✓ Los datos se pueden grabar más de 1.000.000 veces
- 368 bytes de memoria RAM
- Convertidor A/D:
  - ✓ 14 canales
  - ✓ Resolución de 10 bits
- 3 temporizadores/contadores independientes
- Temporizador perro guardián
- Módulo comparador analógico con
  - ✓ Dos comparadores analógicos
  - ✓ Referencia de voltaje fija (0.6V)
  - ✓ Referencia de voltaje programable en el chip
- Módulo PWM incorporado
- Módulo USART mejorado
  - ✓ Soporta las comunicaciones seriales RS-485, RS-232 y LIN2.0
  - ✓ Auto detección de baudios
- Puerto Serie Síncrono Maestro (MSSP)
  - ✓ Soporta los modos SPI e I2C

2.9.2. Descripción de Pines del PIC16F887.

Ahora describiremos brevemente a cada uno de pines del microcontrolador PIC16F887, los mismos son multipropósito. En la figura 2.20 se puede visualizar el pin #5, cuya asignación es RA3/AN3/V_{REF}+/C1IN+, el mismo cumple las siguientes funciones:

- RA3, Puerto A3 de E/S Digital
- AN3, Tercera entrada analógica
- V_{REF}+, Referencia positiva de voltaje
- C1IN+, Entrada positiva del comparador C1

La funcionalidad de los pines presentados anteriormente es muy útil puesto que permite un mejor aprovechamiento de los recursos del microcontrolador sin afectar a su funcionamiento. Estas funciones de los pines no se pueden utilizar simultáneamente, sin embargo se pueden cambiar en cualquier instante durante el funcionamiento(Verle, 2010). En la tabla 2.9 se describen cada uno de los 40 pines.

Tabla 2. 9: Descripción de pines del PIC16F887.

Nombre	Pines	Función	Descripción
RE3/MCLR/V_{PP}	1	RE3	Entrada de propósito general en el puerto E.
		MCLR	Pin de inicio. El nivel lógico bajo en este pin reinicia al microcontrolador
		V_{PP}	Voltaje de programación.
RA0/AN0/ULPWU/C12IN0-	2	RA0	E/S de propósito general en el Puerto A.
		AN0	Entrada del canal 0 del convertidor A/D.
		ULPWU	Entrada de desactivar modo de espera.
		C12IN0-	Entrada (–) del comparador C1 o C2.
RA1/AN1/C12IN1-	3	RA1	E/S de propósito general en el puerto A.
		AN1	Canal 1 del convertidor A/D.
		C12IN1-	Entrada (–) del comparador C1 o C2.
RA2/AN2/V_{REF-}/CV_{REF}/C2IN+	4	RA2	E/S de propósito general en el puerto A.
		AN2	Canal 2 del convertidor A/D.
		V_{REF}-	Entrada de referencia (–) de voltaje del convertidor A/D.
		CV_{REF}	Salida de referencia de voltaje del comparador.
		C2IN+	Entrada (+) del comparador C2.
RA3/AN3/V_{REF+}/C1IN+	5	RA3	E/S de propósito general en el Puerto A.
		AN3	Entrada del canal 0 del convertidor A/D.
		V_{REF+}	Entrada de desactivar modo de espera.
		C1IN+	Entrada (+) del comparador C1.
RA4/T0CKI/C1OUT	6	RA4	E/S de propósito general en el puerto A.
		T0CKI	Entrada de reloj del temporizador T0.

		C1OUT	Salida del comparador C1.
RA5/AN4/SS/C2OUT	7	RA5	E/S de propósito general en el Puerto A.
		AN4	Canal 4 del convertidor A/D.
		SS	Entrada del módulo SPI (Selección del esclavo).
		C2OUT	Salida del comparador C2.
RE0/AN5	8	RE0	E/S de propósito general en el puerto E.
		AN5	Canal 5 del convertidor A/D.
RE1/AN6	9	RE1	E/S de propósito general en el puerto E.
		AN6	Canal 6 del convertidor A/D.
RE2/AN7	10	RE2	E/S de propósito general en el puerto E.
		AN7	Canal 7 del convertidor A/D.
V_{dd}	11	+	Suministro de voltaje positivo.
V_{ss}	12	-	Ground, GND (Tierra)
RA7/OSC1/CLKIN	13	RA7	E/S de propósito general en el puerto A.
		OSC1	Entrada del oscilador de cristal.
		CLKIN	Entrada del reloj externo.
RA6/OSC2/CLKOUT	14	RA6	E/S de propósito general en el puerto A.
		OSC2	Salida del oscilador del cristal.
		CLKOUT	Salida en la que se presenta la señal Fosc/4.
RC0/T1OSO/T1CKI	15	RC0	E/S de propósito general en el puerto C.
		T1OSO	Salida del oscilador del temporizador 1.
		T1CKI	Entrada de reloj del temporizador 1.
RC1/T1OSI/CCP2	16	RC1	E/S de propósito general en el puerto C.
		T1OSI	Entrada del oscilador del temporizador 1.
		CCP2	E/S de los módulos CCP1 y PWM1.
RC2/P1A/CCP1	17	RC2	E/S de propósito general en el puerto C.
		P1A	Salida del módulo PWM.
		CCP1	E/S de los módulos CCP1 y PWM1.
RC3/SCK/SCL	18	RC3	E/S de propósito general en el puerto C.
		SCK	E/S de reloj del módulo MSSP en modo SPI.
		SCL	E/S de reloj del módulo MSSP en modo I^2C.
RD0	19	RD0	E/S de propósito general en el puerto D.
RD1	20	RD1	E/S de propósito general en el puerto D.
RD2	21	RD2	E/S de propósito general en el puerto D.

RD3	22	RD3	E/S de propósito general en el puerto D.
RC4/SDI/SDA	23	RC4	E/S de propósito general en el puerto C.
		SDI	Entrada Data del módulo MSSP en modo SPI.
		SDA	E/S Data del módulo MSSP en modo I^2C.
RC5/SDO	24	RC5	E/S de propósito general en el puerto C.
		SDO	Salida Data del módulo MSSP en modo SPI.
RC6/TX/CK	25	RC6	E/S de propósito general en el puerto C.
		TX	Salida asíncrona del módulo USART.
		CK	Reloj síncrono del módulo USART.
RC7/RX/DT	26	RC7	E/S de propósito general en el puerto C.
		RX	Entrada asíncrona del módulo USART.
		DT	Datos del módulo USART en modo síncrono.
RD4	27	RD4	E/S de propósito general en el puerto D.
RD5/P1B	28	RD5	E/S de propósito general en el puerto D.
		P1B	Salida del módulo PWM.
RD6/P1C	29	RD6	E/S de propósito general en el puerto D.
		P1C	Salida del módulo PWM.
RD7/P1D	30	RD7	E/S de propósito general en el puerto D.
		P1D	Salida del módulo PWM.
V_{SS}	31	-	Ground, GND (Tierra)
V_{dd}	32	+	
RB0/AN12/INT	33	RB0	E/S de propósito general en el puerto B.
		AN12	Canal 12 del convertidor.
		INT	Interrupción externa.
RB1/AN10/C12INT3-	34	RB1	E/S de propósito general en el puerto B.
		AN10	Canal 10 del convertidor.
		C12INT3	Entrada negativa de los comparadores C1 o C2.
RB2/AN8	35	RB2	E/S de propósito general en el puerto B.
		AN8	Canal 8 del convertidor.
RB3/AN9/PGM/C12IN2-	36	RB3	E/S de propósito general en el Puerto B.
		AN9	Canal 9 del convertidor A/D.
		PGM	Habilita la programación del chip.
		C12IN2-	Entrada negativa de los comparadores C1 o C2.
RB4/AN11	37	RB4	E/S de propósito general en el puerto B.

		AN11	Canal 11 del convertidor A/D.
RB5/AN13/T1G	38	RB5	E/S de propósito general en el puerto B.
		AN13	Canal 13 del convertidor A/D.
		T1G	Entrada externa del temporizador 11.
RB6/ICSPCLK	39	RB6	E/S de propósito general en el puerto B.
		ICSPCLK	Entrada de reloj de programación serial.
RB7/ICSPDAT	40	RB7	E/S de propósito general en el puerto B.
		ICSPDAT	Pin de E/S para introducir los datos durante la programación ICSP.

Elaborado por: Los Autores

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/41291G.pdf

2.9.3. Los puertos de Entrada y Salida (E/S) del PIC16F887.

De acuerdo al acápite anterior (ver tabla 2.9) se pudo constatar que existen 35 pines de E/S de los 40 pines, que permite conectarlo con los periféricos. Según Verle (2010): el propósito de sincronizar el funcionamiento de los puertos de E/S con la organización interna del microcontrolador de 8 bits, ellos se agrupan, de manera similar a los registros, en cinco puertos denotados con A, B, C, D y E. Todos ellos tienen las siguientes características en común (véase la figura 2.23):

- Por las razones prácticas, muchos pines de E/S son multifuncionales. Si un pin realiza una de estas funciones, puede ser utilizado como pin de E/S de propósito general.
- Cada puerto tiene su propio registro de control de flujo, o sea el registro TRIS correspondiente: TRISA, TRISB, TRISC etc. lo que determina el comportamiento de bits del puerto, pero no determina su contenido. Al poner a cero un bit del registro TRIS (bit=0), el pin correspondiente del puerto se configurará como una salida. De manera similar, al poner a uno un bit del registro TRIS (bit=1), el pin correspondiente del puerto se configurará como una entrada. Esta regla es fácil de recordar: 0 = Salida 1 = Entrada.

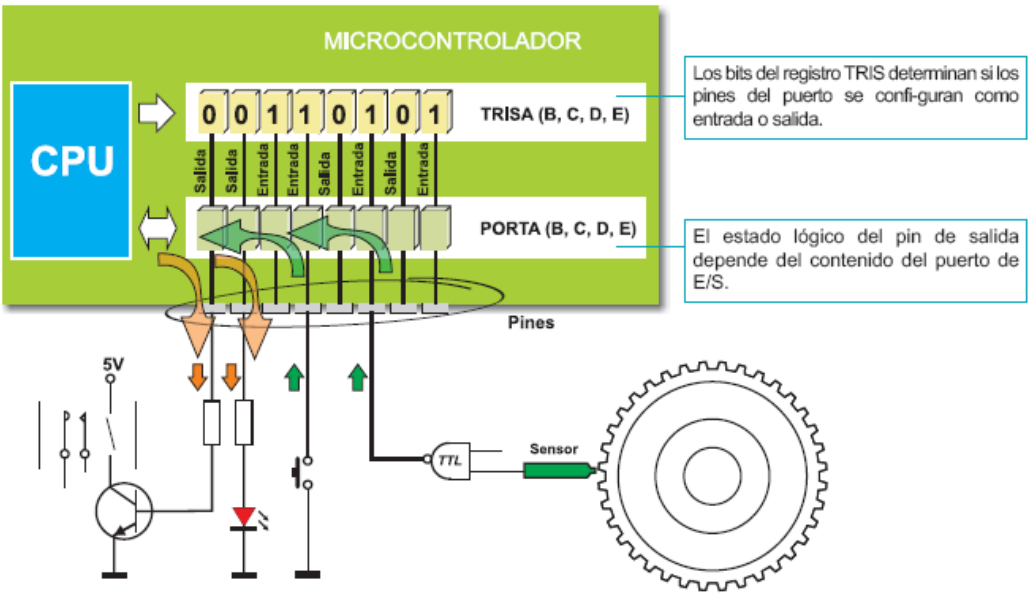


Figura 2. 23: Puertos A, B, C, D y E de E/S del PIC16F887.
Fuente: (Verle, 2010), disponible en línea:http://www.mikroe.com/chapters/view/86/libro-de-a-progamacion-de-los-microcontroladores-pic-en-basic-capitulo-3-microcontrolador-pic16f887/#c3v3

Como ya se indicó anteriormente los puertos de E/S tienen 8 bits de anchura, es decir, que el puerto A denotado como PORTA es bidireccional. En la figura 2.23 se visualiza al registro TRISA, este registro se encarga de controlar los pines del puerto A (PORTA), es decir, si se comporta como entradas o salidas digitales.

Aunque para el presente trabajo de titulación sólo se ha considerado partes importantes del PIC16F887, el lector podrá dirigirse al libro de Milán Verle (2010) en la página web: http://www.mikroe.com/products/view/476/pic-microcontrollers-programming-in-basic/, ahí encontrará información de relevancia, además el texto se encuentra escrito en Inglés y traducido al Español.

CAPÍTULO 3: PROGRAMACIÓN PIC BASIC Y SISTEMA OPERATIVO ANDROID.

Para el presente capítulo describiremos las plataformas de programación tanto para el microcontrolador (PIC Basic) como el que se empleará en la Tablet (Appinventor). El lenguaje de programación escogido es PIC Basic o Mikro Basic (Microchip), por ser un lenguaje de alto nivel. Mientras que Appinventor, nos permite desarrollar programas virtuales que transmiten información por Bluetooth.

3.1. Lenguaje de programación Pic Basic.

Primero hay que diferenciar entre los dos lenguajes de programación para microcontroladores, se debe tener en claroqué es un lenguaje de alto nivel y qué es un lenguaje de bajo nivel. En la figura 3.1 se muestran los diferentes niveles de programación orientado a microcontroladores.

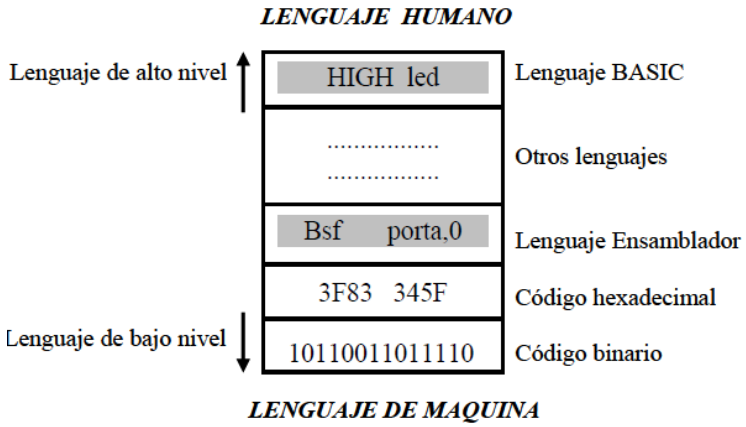


Figura 3. 1: Niveles del Lenguaje de Programación.
Fuente: (Reyes, 2008)

En la realidad Pic Basic es un programa que se encarga de traducir al código de máquina (binario), siendo este un lenguaje que entiende el microcontrolador (véase la figura 3.2). Es decir, que a partir del programa fuente (nombre.pbp), produce un archivo en formato hexadecimal (nombre.hex), que constituye el código ejecutable por el PIC, tal y como se muestra en la figura 3.3. El archivo hexadecimal es el que se tiene que abrir en el software del grabador de PIC´s (Hermosa Donate, 2010).

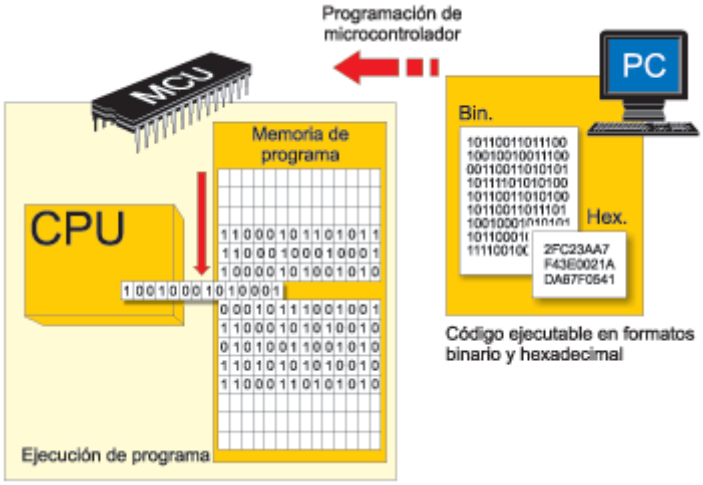


Figura 3. 2: Diagrama de bloques de la Programación de Microcontroladores.
Fuente: (Verle, 2010)

Basic se encarga de programar aplicaciones orientadas a microcontroladores PIC de cualquier gama. Según lo indicado por Hermosa (2010): "Programar en Pic Basic es realizar programas de manera rápida y sencilla en comparación con el lenguaje ensamblador. El lenguaje ensamblador tiene la ventaja de poder ser muy optimizado en recursos del microprocesador, aunque ocupan menos espacio de memoria, en la figura 3.4 se muestra el diagrama de bloques de programación Assembler. Pero se sabe que la mayoría de lenguajes de alto nivel (Pic Basic o Pic C) introducen también instrucciones en ensamblador (Assembler).

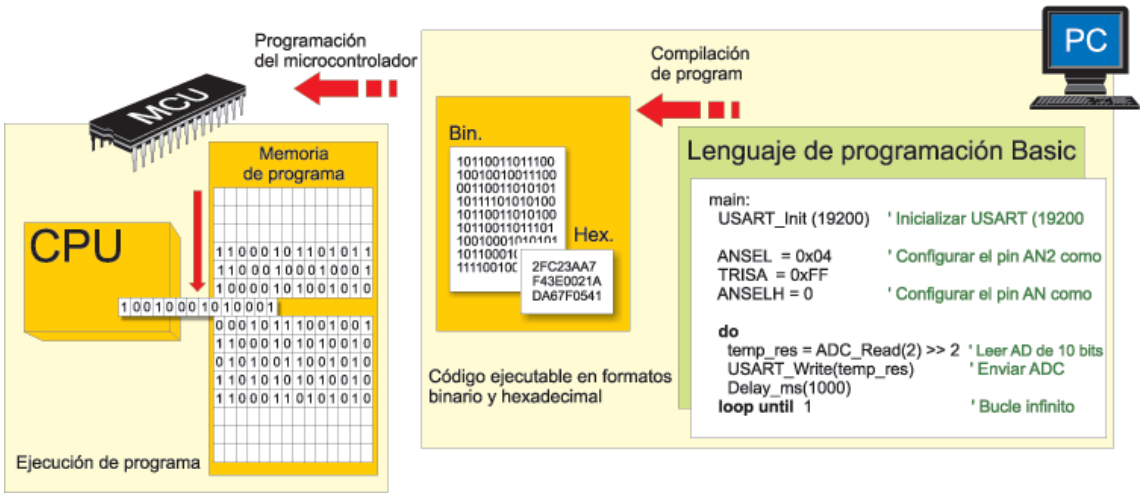


Figura 3. 3: Diagrama de bloques de la Programación en Basic para PIC´s
Fuente: (Verle, 2010)

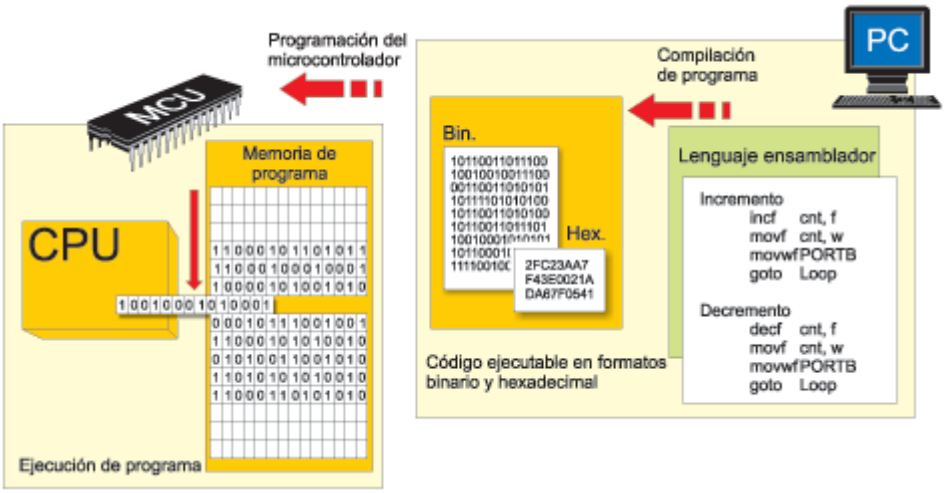


Figura 3. 4: Diagrama de bloques de la Programación Assembler para PIC´s
Fuente: (Verle, 2010)

A continuación Reyes (2008), expone a través de un ejemplo (como hacer parpadear un diodo led con intervalos de 1 segund0) como diferenciar entre un lenguaje de alto nivel (véase figura 3.5) y un lenguaje de máquina (véase figura 3.3).

```
inicio:                 ; nombre de subrutina inicio
        HIGH portb.1    ; enciende el led que esta conectado en el pin 7
        PAUSE 1000      ; espera un segundo
        LOW portb.1     ; apaga el led
        PAUSE 1000      ; espera un segundo
GOTO inicio             ; continúa el programa para siempre
```

Figura 3. 5:Programación en Pic Basic para encender un diodo LED.
Fuente: (Reyes, 2008)

En la figura 3.5 se muestra la programación en Pic Basic, en la cual con 6 líneas de programación se ejecuta el encendido de un LED. Mientras que en la figura 3.6 se muestran 40 líneas de programación en lenguaje ensamblador (bajo nivel), en la cual se utilizan instrucciones que trabajan directamente con los registros de memoria, es decir, que el usuario o programador debe conocer las posiciones de la memoria disponibles para cada gama de los PIC´s. adicionalmente deben realizar cálculos muy precisos para generar retardos de 1 segundo.

```
        list p=16F628A

status  equ     03h             ;etiquetamos cada posición de memoria
portb   equ     06h
trisb   equ     86h
cont1   equ     20h             ;etiquetamos cada variable según el lugar que el datasheet
cont2   equ     21h             ; asigne como espacio de memoria RAM
cont3   equ     22h

reset   org     0               ;se escribe en la línea 0 la instrucción
        goto    inicio          ;salta a la línea etiquetada con inicio
        org     5               ;las siguientes líneas se escribirán desde la dirección 5

retardo movlw   D'10'           ;El registro cont1 contiene el número de
        movwf   cont1           ;veces que repite 100 milisegundos
repite1 movlw   D'100'          ;El registro cont2 contiene el número de
        movwf   cont2           ;veces que repite 1 milisegundo
repite2 movlw   D'110'          ;El registro cont3 contiene el número de
        movwf   cont3           ;veces que repite los 9 microsegundos
repite3 nop                     ;de retardo generados
        nop                     ;por los 6 ciclos de las instrucciones nop (6usg)
        nop                     ;más 1 ciclo de la instrucción decfsc (1usg)
        nop                     ;más 2 ciclos del salto goto (2usg)
        nop                     ;dando en total los 9usg, siendo esta la base
        nop                     ;de tiempo, por lo tanto 1sg= 9usg*110*100*10
        decfsz  cont3           ;decrementa el reg cont3 y salta si llega a 0
        goto    repite3         ;si cont3 no es 0 entonces salta a repite3
        decfsz  cont2           ;decrementa el reg cont2 y salta si llega a 0
        goto    repite2         ;si cont2 no es 0 entonces salta a repite2
        decfsz  cont1           ;decrementa el reg cont1 y salta si llega a 0
        goto    repite1         ;si cont1 no es 0 entonces salta a repite1
        retlw   0               ;salida de la subrutina cargando w con 0

inicio  bsf     status,5        ;se ubica en el segundo banco de la RAM
        movlw   00h             ;se carga el registro w con 00h
        movwf   trisb           ;se programa el puerto B como salidas
        bcf     status,5        ;se ubica en el primer banco de la RAM

prog    bsf     portb,1         ;coloca en 1 el pin RB1 para encender el led
        call    retardo         ;Llama a la subrutina retardo de 1 segundo
        bcf     portb,1         ;Coloca en 0 el pin RB1 para apagar el led
        call    retardo         ;Llama a la subrutina retardo de 1 segundo
        goto    prog            ;salta a prog para repetir la secuencia
end
```

Figura 3. 6: Programación en Assembler para encender un diodo LED.
Fuente: (Reyes, 2008)

Mientras que para Verle (2010): escribir o realizar un programa para PIC´s en lenguaje ensamblador, probablemente sepa que la arquitectura RISC carece de algunas instrucciones. Por ejemplo, no hay instrucción apropiada para

multiplicar dos números. Por supuesto, este problema se puede resolver gracias a la aritmética que permite realizar las operaciones complejas al descomponerlas en un gran número de operaciones más simples. En este caso, la multiplicación se puede sustituir con facilidad por adición sucesiva $(a \times b = a + a + a + \cdots + a)$. En la figura 3.7, Verle (2010) nos muestra otra diferencia de programación en Basic y Ensamblador, es similar al propuesto por Reyes (2008).

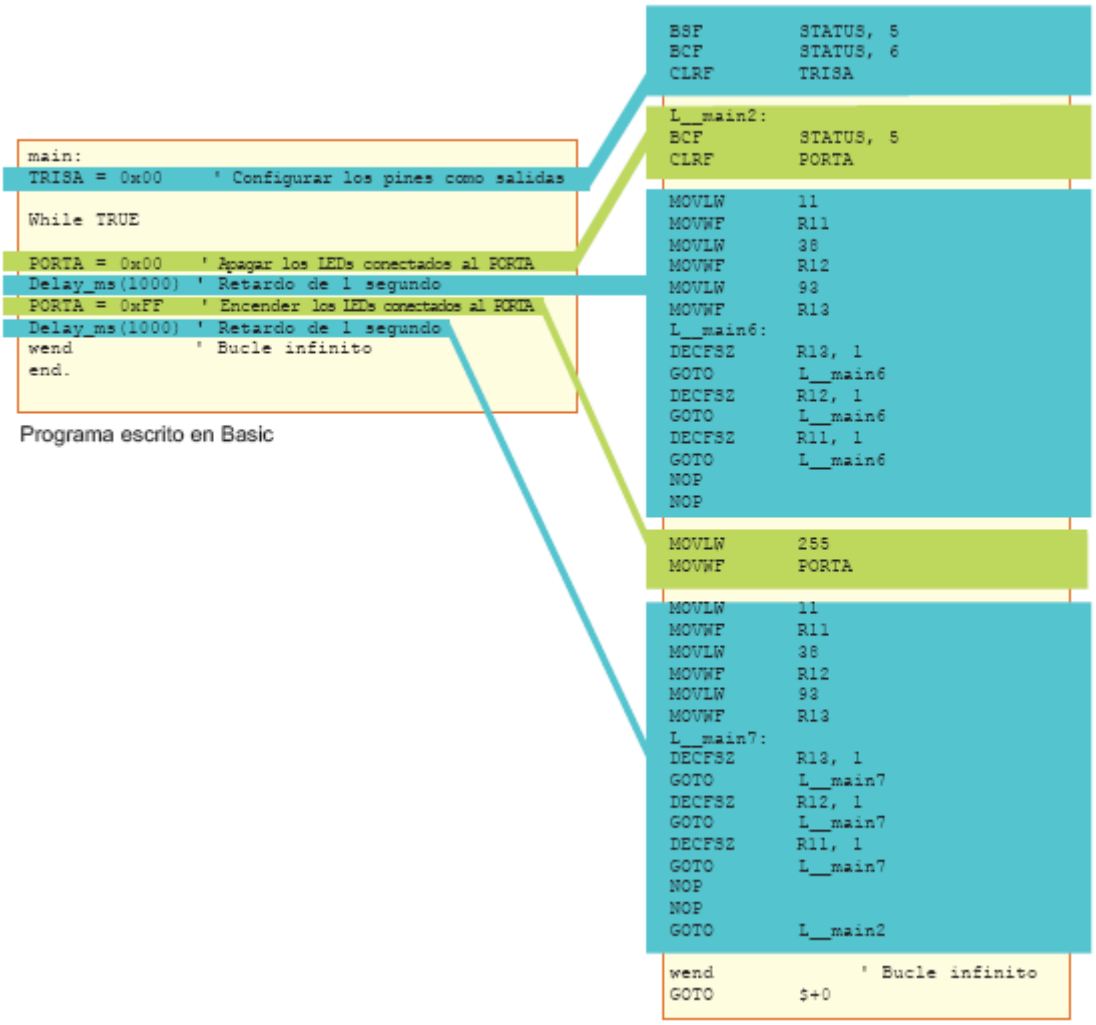


Figura 3. 7: Programación en Assembler para encender un diodo LED.
Fuente: (Verle, 2010)

3.2. Características del Lenguaje de Programación Pic Basic.

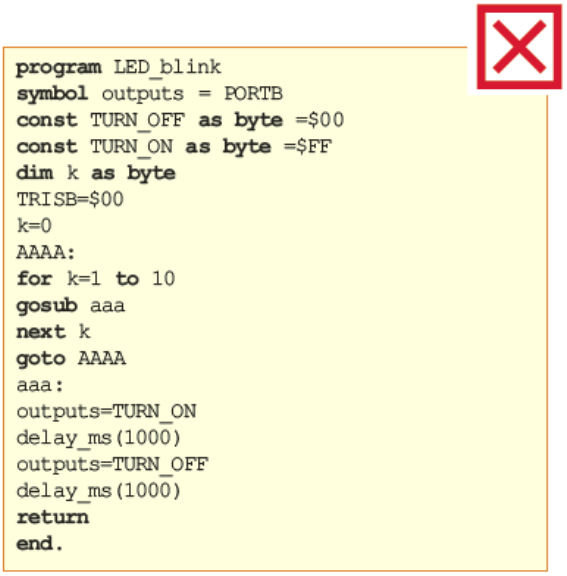
Similar al uso de cualquier lengua que no está limitada a los libros y a las revistas, el lenguaje de programación Basic no está estrechamente relacionado a un tipo particular de ordenador, procesador o sistema operativo. Esto puede

ser un problema, ya que Basic varía ligeramente dependiendo de su aplicación (como diferentes dialectos de una lengua). Es decir, que no se describirán todas las características de Basic, sino presentar una aplicación muy concreta de Basic, utilizado en compiladores tales como Pic Basic o MikroBasic PRO for PIC.

Los compiladores mencionados son compatibles, aunque en nuestro proyecto de titulación el software empleado es MikroBasic PROpara PIC´s. Para utilizarcorrectamente, basta conocer sólo unos pocos elementos básicos, estos son:

- Identificadores
- Comentarios
- Operadores
- Expresiones
- Instrucciones
- Constantes
- Variables
- Símbolos
- Directivas
- Etiquetas
- Procedimientos y funciones
- Módulos

Aunque no existen textos que sólo se dediquen a describir el lenguaje de programación en alto nivel Pic Basic, hemos escogido el texto de Verle (2010). El mismo que se han tomado varios párrafos acerca de la programación en mención. Verle (2010) propone un ejemplo en la figura 3.8, de cómo no se debe escribir un programa. Es decir, que no incluyen comentarios, nombres de etiquetas no tienen significado, secciones del código no están agrupadas.



```
program LED_blink
symbol outputs = PORTB
const TURN_OFF as byte =$00
const TURN_ON as byte =$FF
dim k as byte
TRISB=$00
k=0
AAAA:
for k=1 to 10
gosub aaa
next k
goto AAAA
aaa:
outputs=TURN_ON
delay_ms(1000)
outputs=TURN_OFF
delay_ms(1000)
return
end.
```

Figura 3. 8: Ejemplo de un programa mal escrito en Basic.
Fuente: (Verle, 2010)

En la figura 3.9, Verle (2010) nos muestra cómo se debe escribir un programa correctamente escrito, las diferencias son más que obvias...

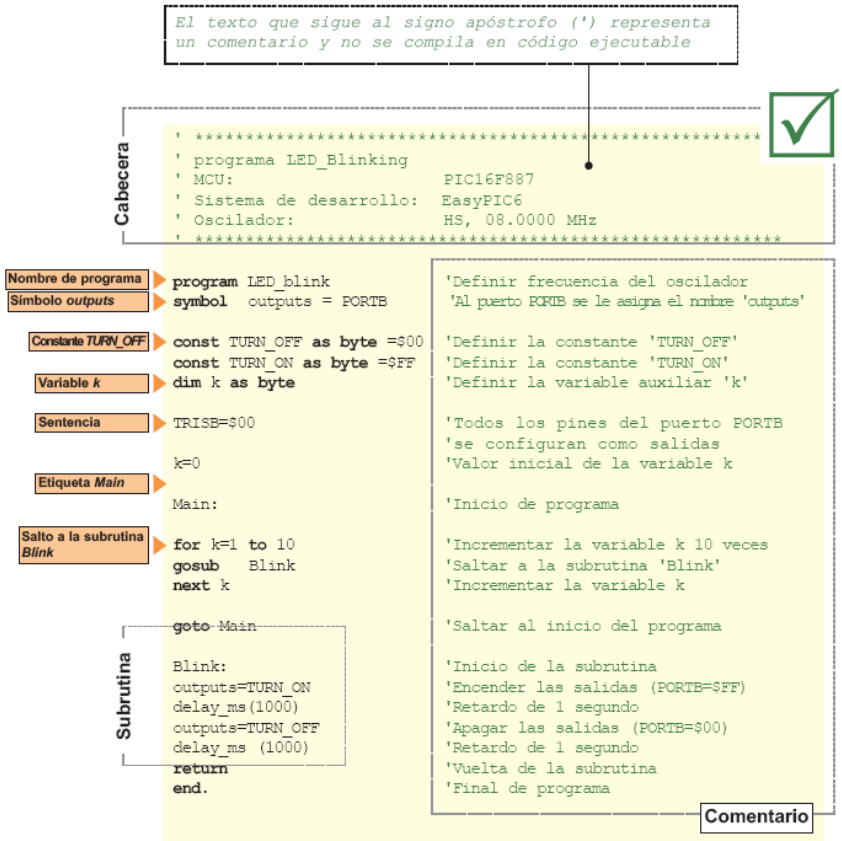


Figura 3. 9: Ejemplo de un programa correctamente escrito en Basic.
Fuente: (Verle, 2010)

Según indica Verle (2010): existen similares lenguajes de programación, pero Basic dispone de un conjunto de reglas estrictamente definidas que se deben observar al escribir un programa. Para escribir un programa en Basic, es necesario instalar un software que proporciona el entorno de trabajo apropiado y entiende estas reglas en la PC. A diferencia de la mayoría de programas a los que está acostumbrado a manejar, el proceso de escribir programas en el compilador no empieza por seleccionar la opción File>New, sino Project>New.

En la figura 3.10 se muestra la programación de un proyecto antes de ser compilado y posteriormente cuando se compila, antes de compilar se crean dos módulos: el módulo principal (programación Basic) y los módulos adicionales.Mientrasque después de la compilación, se generan módulos o código fuente

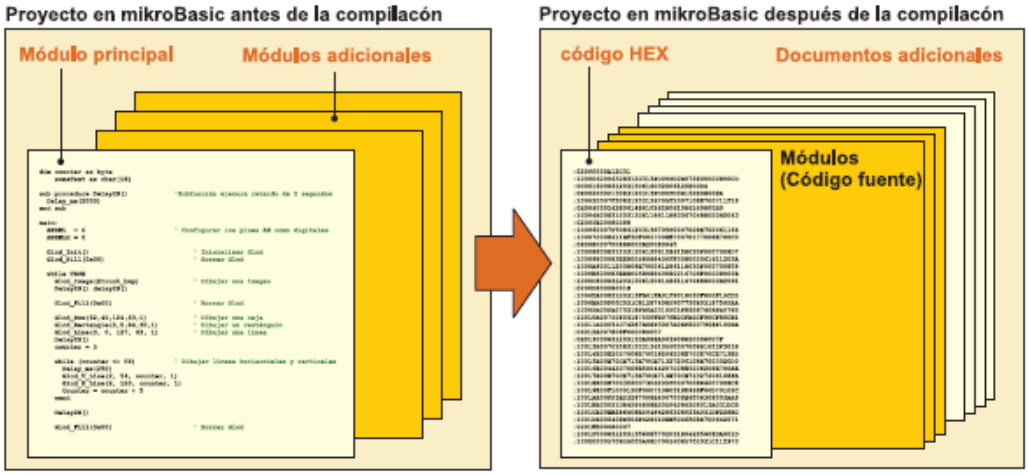


Figura 3. 10: programación Basic
Fuente: (Verle, 2010)

A continuación se describirán brevemente cada uno de los elementos descritos anteriormente.

3.2.1. Identificadores.

Los identificadores son utilizados en Pic Basic, para firmar líneas de programa y los nombres de varios símbolos. Un identificador, podría ser cualquier cadena de letras, números o incluso guiones sin permitir que inicien con un número. Es decir, que los identificadores no distinguen mayúsculas y

minúsculas, por lo que las cadenas **TASTER** y **Taster** (véase figura 3.11) son tratados de la misma manera. Para este tipo de cadena tiene como máximo 32 caracteres.

```
symbol Taster = PORTA.0
symbol LED_0 = PORTB.0
```

Figura 3. 11: Representación de los identificadores en Pic Basic
Fuente: Ejemplos del Software Microcode Pic Basic

3.2.2. Comentarios.

Los comentarios representanla explicación textual a lado de la línea de programación o de algunos de sus fragmentos en los que el programa se puede saltar a través de algunas de las instrucciones que se utilizan para cambiar el flujo del programa, en la figura 3.12 se muestra cortos comentarios “Label Main” y “Label LED_toggle”. En Pic Basic no se permiten comentarios que tengan valores numéricos.

```
symbol Taster = PORTA.0
symbol LED_0 = PORTB.0

B0 var byte

Main:                ' Label Main
        B0 = 0
        button Set,0,255,0,B0,1,LED_toggle
        goto Main

LED_toggle:          ' Label LED_toggle
        toggle LED_0
        goto Main
        end
```

Figura 3. 12: Representación de comentarios en Pic Basic
Fuente: Ejemplos del Software Microcode Pic Basic

3.2.3. Constantes.

Las constantes son elegidas con un nombre y le asigna un valor constante, suponiendo que una constante es “minutos”cuyo valor debe ser “60 segundos”, (se sabe que 1 minutos es igual a 60 segundos).En la figura 3.13 se muestra la asignación de la constante “minute” (minutos).

```
minute con 60                                   ' No. of seconds in a minute
if seconds < minute then minute = minute + 1    ' If the number of seconds is different
                                                ' from 60, raise the variable minutes
```

Figura 3. 13: Representación de una constante en Pic Basic
Fuente: Ejemplos del Software Microcode Pic Basic

Sin embargo, las constantes también se escriben en formato decimal, hexadecimal y binario, donde las constantes decimales se escriben sin ningún prefijo. El formato hexadecimal inicia siempre con el símbolo **$** y el formato binario con el símbolo **%**. Aunque, puede ser más fácil para el programador (usuario) si convierten los formatos anteriores al equivalente ASCII. En la figura 3.14 se muestran ejemplos de constantes cuyos símbolos se colocan entre comillas.

```
56              ' 56 decimal
$0F             ' 15 hexadecimal
%10001100       ' 140 binary
"A"             ' ASCII value for decimal 65
"d"             ' ASCII value for decimal 100
```

Figura 3. 14: Representación de constantes en formatos del sistema de numeración.
Fuente: Ejemplos del Software Microcode Pic Basic

3.3. Declaraciones disponibles en el compilador PBP.

Existen varias declaraciones que emplea el compilador de Pic Basic Pro (PBP) 2.47, pero tienen declaraciones reservadas con tareas específicas, las más utilizadas son: HIGH, LOW, PAUSE, GOSUB, GOTO, LCDOUT, SERIN, SEROUT, FOR, NEXT, IF, THEN, SOUND, END.(Reyes, 2008)

Por ejemplo, si la línea de programación es HIGH portb.1, microcode la reconoce y se coloca ennegrilla y mayúscula, sirve para mandar un “1” lógico (5 V) al puerto B1, es decir, permite el encendido de un LED. A continuación, la tabla 3.1 describe brevemente las 83 instrucciones disponibles:

Tabla 3. 1:Instrucciones de Pic Basic Pro.

DECLARACIÓN	APLICACIÓN
@	Inserta una línea de código ensamblador
ADCIN	Lee el conversor analógico
ASM...ENDASM	Insertar una sección de código ensamblador
BRANCH	GOTO computado (equivale a ON..GOTO)
BRANCHL	BRANCH fuera de página (BRANCH Largo)
BUTTON	Anti-rebote y auto-repetición de entrada en el pin especificado
CALL	Llamada a subrutina de ensamblador
CLEAR	Hace cero todas las variables
CLEARWDT	Hace cero el contador del Watchdog Timer
COUNT	Cuenta el número de pulsos en un pin
DATA	Define el contenido inicial en un chip EEPROM
DEBUG	Señal asincrónica de salida en un pin fijo y baud
DEBUGIN	Señal asincrónica de entrada en un pin fijo y baud
DISABLE	Deshabilita el procesamiento de ON INTERRUPT, ON DEBUG
DISABLE DEBUG	Deshabilita el procesamiento de ON DEBUG
DISABLE INTERRUPT	Deshabilita el procesamiento de ON INTERRUPT
DTMFOUT	Produce tonos telefónicos en un pin
EEPROM	Define el contenido inicial en un chip EEPROM
ENABLE	Habilita el procesamiento de ON INTERRUPT, ON DEBUG
ENABLE DEBUG	Habilita el procesamiento de ON DEBUG
ENABLE INTERRUPT	Habilita el procesamiento de ON INTERRUPT
END	Detiene la ejecución e ingresa en modo de baja potencia
FOR…NEXT	Ejecuta declaraciones en forma repetitiva
FREQOUT	Produce hasta 2 frecuencias en un pin
GOSUB	Llama a una subrutina BASIC en la línea especificada
GOTO	Continua la ejecución en la línea especificada
HIGH	Saca un 1 lógico (5 V.) por un pin
HPWM	Salida de hardware con ancho de pulsos modulados
HSERIN	Entrada serial asincrónica (hardware)
HSEROUT	Salida serial asincrónica (hardware)
I2CREAD	Lee bytes de dispositivos I2C
I2CWRITE	Graba bytes de dispositivos I2C
IF..THEN..ELSE..ENDIF	Ejecuta declaraciones en forma condicional
INPUT	Convierte un pin en entrada
LCDIN	Lee caracteres desde una RAM de un LCD
LCDOUT	Muestra caracteres en un LCD
LET	Asigna el resultado de una expresión a una variable
LOOKDOWN	Busca un valor en una tabla de constantes
LOOKDOWN2	Busca un valor en una tabla de constantes o variables
LOOKUP	Obtiene un valor constante de una tabla
LOOKUP2	Obtiene un valor constante o variable de una tabla
LOW	Hace 0 lógico (0 V.) un pin específico
NAP	Apaga el procesador por un corto período de tiempo
ON DEBUG	Ejecuta un Debug en BASIC
ON INTERRUPT	Ejecuta una subrutina BASIC en un interrupt
OUTPUT	Convierte un pin en salida
OWIN	Entrada de dispositivos one-wire
OWOUT	Salida a dispositivos one-wire
PAUSE	Demora con resolución de 1 milisegundo (mS.)
PAUSEUS	Demora con resolución de 1 microsegundo (uS.)
PEEK	Lee un byte del registro

POKE	Graba un byte en el registro
POT	Lee el potenciómetro en el pin especificado
PULSIN	Mide el ancho de pulso en un pin
PULSOUT	Genera pulso hacia un pin
PWM	Salida modulada en ancho de pulso por un pin especificado
RANDOM	Genera número pseudo-aleatorio
RCTIME	Mide el ancho de pulso en un pin
READ	Lee byte de un chip EEPROM
READCODE	Lee palabra desde un código de memoria
RESUME	Continua la ejecución después de una interrupción
RETURN	Continua en la declaración que sigue al último GOSUB
REVERSE	Convierte un pin de salida en entrada, o uno de entrada en salida
SELECT CASE	Compara una variable con diferentes valores
SERIN	Entrada serial asincrónica (tipo BASIC Stamp 1)
SERIN2	Entrada serial asincrónica (tipo BASIC Stamp 2)
SEROUT	Salida serial asincrónica (tipo BS1)
SEROUT2	Salida serial asincrónica (tipo BS2)
SHIFTIN	Entrada serial sincrónica
SHIFTOUT	Salida serial sincrónica
SLEEP	Apaga el procesador por un período de tiempo
SOUND	Genera un tono o ruido blanco en un pin
STOP	Detiene la ejecución del programa
SWAP	intercambia los valores de dos variables
TOGGLE	Hace salida a un pin y cambia su estado
USBIN	Entrada de USB
USBINIT	Inicializar USB
USBOUT	Salida de USB
WHILE…WEND	Ejecuta declaraciones mientras la condición sea cierta
WRITE	Graba bytes en un chip EEPROM
WRITECODE	Escribe palabra en código de memoria
XIN	Entrada X - 10
XOUT	Salida X -10

Elaborado: Los Autores

Fuente:(Reyes, 2008)

3.4. Componente y operadores en Pic Basic.

PIC Basic cuenta con una serie de herramientas de programación entre lascuales podemos mencionar las etiquetas, variables, identificadores,constantes, comentarios, símbolos entre otras.Algunas de estas herramientas son de uso obligatorio a la hora de realizar unprograma, y otras que no son de uso obligatorio, nos facilitarán el trabajoconsiderablemente.

3.4.1. Define.

La directiva “Define” es la más importante para programación BASIC de microcontroladores Pic, la misma nos indica una serie de parámetros, que estos al no ser considerados, la programación no funcione correctamenteen la

mayoría de los casos.Los parámetros de las directivas son relacionadas directamente con dispositivos externos del PIC. Es decir, que si el oscilador es diferente a la frecuencia que por defecto es 4 MHz, entonces sería conveniente que la velocidad del oscilador la defina el programador, utilizando la directiva:

Define Osc {frecuencia}

Los parámetros que se indicaron, también deben considerarse para la utilización de dispositivos, tales como LCD (pantallas de cristal líquido) en la cual deberían definir cada uno de los puertos deconexión, tanto para los buses de datos y control. En la tabla 3.2 se muestran los parámetros empleados para el uso de lainstrucción Define.

Tabla 3. 2:Parámetros y descripción de la instrucción Define.

Parámetros	Descripción
OSC {frecuencia}	Frecuencia del Oscilador en MHz.
LCD_DREG {puerto}	Puerto de datos LCD.
LCD_DBIT {bit}	Bit inicial del puerto de datos.
LCD_RSREG {puerto}	Puerto para RS (Register Select).
LCD_RSBIT {bit}	Pin del Puerto para RS.
LCD_EREG {puerto}	Puerto para E (Enable).
LCD_EBIT {bit}	Pin delPuerto para E.
LCD_RWREG {puerto}	Puerto para RW (Read/Write)
LCD_RWBIT {pin}	Pin del puerto para RW.
LCD_LINES {líneas}	Número de líneas de la LCD (1,2 0 4).
I2C_SCLOUT 1	Cuando en la transferencia es utilizado un oscilador más lento que 8 MHz.

Fuente:(Reyes, 2008)

3.4.2. Variables.

En las variables podemos almacenar datos temporalmente,los cuales podrán ser consultados o modificados cada vez que sea necesario.Regularmente la definición de variables se hace al inicio del

programa y paraello se utiliza la palabra VAR seguida por el tipo de variable (véase la tabla 3.3).

Tabla 3. 3:Tipos de variables y su descripción para programación Basic.

Variable	VAR	Tipo de Variable	Descripción
A1	Var	Bit	Toma los valores 0 y 1 únicamente.
Temp	Var	Byte	Toma valores entre 0 y 255 (8 bits).
dig1	Var	Word	Toma valores entre 0 y 65535 (16 bits).

Fuente:(Reyes, 2008)

El nombre de la variable es elegido por el programador y el tipo de variable sedefine según el tipo de dato que se desea almacenar temporalmente.

3.4.3. Arrays.

Las variables Arrays tienen un determinado número de“elementos”, definido según el tamaño de la variable. Las variables Arrays tipoBit, pueden almacenar 256 elementos; las variables Arrays tipo Byte puedenalmacenar hasta 96 elementos y las variables Arrays tipo Word hasta 48elementos, los cuales a su vez pueden ser acceder en cualquiera de lostres casos a través de un índice. Este índice se específica entre corchetescomo se muestra a continuación:

- ✓ Para declarar una variable Array utilizamos el siguiente formato:

  Dato Var Byte[7]

El primer elemento de esta variable es Dato[0] y el último elemento esDato[7], lo cual significa que hemos declarado una variable array de 8elementos. En este caso podemos almacenar un byte en cada elemento,siempre especificando el índice.

- ✓ Para almacenar en cada elemento de la variable “Dato” los valores 200, 15, 56, 75, 80, 20, 33, 45.

  Dato[0] = 200

  Dato[1] = 15

  Dato[2] = 56

Dato[3] = 75

Dato[4] = 80

Dato[5] = 20

Dato[6] = 33

Dato[7] = 45

En algunos casos se debe verificar la hoja de datos del microcontrolador, yaque la cantidad de elementos que se pueden almacenar en variables Arrays tipo Byte o Word puede variar según el modelo del mismo.

3.4.4. Alías o símbolos.

Proveen un nombre único y específico a elementoso variables dentro de nuestro programa. Para definir un símbolo, utilizamos lapalabra “Symbol”, seguida del alias del elemento, el símbolo de igualdad “=”, ypor último el elemento en cuestión:

Symbol {alias} = {elemento}

Por ejemplo, si deseamos controlar un motor DC a través de uno de los pinesdel puerto A de un microcontrolador, resultaría mucho más sencillo referirse aeste pin como “Motor”, en vez de referirse a él como “PortA.0”.Entonces,

Symbol Motor = PORTA.0

Veamos otros ejemplos:

Symbol Relay = PORTB.0

Symbol Sensor = PORTA.0

Symbol LED = PORTA.1

Symbol RC0 = PORTC.0

3.4.5. Operadores aritméticos.

Para la programación en Pic Basic son importantes los operadores aritméticos, estos son muy utilizados en diferentes aplicaciones que empleen

operaciones matemáticas, en la tabla 3.4 se muestran los diferentes operadores aritméticos.

Tabla 3. 4: Tipos de operadores aritméticos.

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
//	Residuo
<<	Desplaza a la izquierda
>>	Desplaza a la derecha
=	Asignación de valores
ABS	Valor Absoluto
SIN	Seno del Ángulo
COS	Coseno del Ángulo
MIN	Mínimo de un número
MAX	Máximo de un número
REV	Invertir un Bit
DIG	Valor de un dígito para un numero decimal

Fuente:(Reyes, 2008)

3.4.6. Operadores binarios.

Asimismo, como en los operadores aritméticos también se consideran las operaciones binarias, es decir, las del algebra booleana. En la tabla 3.5 se muestran los operadores binarios necesarios en programación Pic Basic.

Tabla 3. 5:Tipos de operadores binarios.

Operador	Descripción
&	AND Lógico
|	OR Lógico
^	XOR Lógico
~	NOT Lógico

&/	NAND Lógico
|/	NOR Lógico
^/	NXOR Lógico

Fuente:(Reyes, 2008)

Con estos operadores resulta muy sencillo realizar operaciones binarias,como lo demuestra el siguiente ejemplo:Si aplicamos una AND lógica, donde deseamos filtrar los siete bits mássignificativos del valor almacenado en la siguiente variable:

Var1 = %00101001

Entonces,Var1 = Var1 & %00000001

El resultado de esta operación es Var1 = %00000001

3.4.7. Operadores comparadores.

También tenemos disponibles los operadores comparadores que normalmente se utilizan siempre mediante la instrucción If…Then…, la misma que permite comparar entre variables o datos extraídos de alguna operaciónaritmética, en tabla 3.6 se muestran los operadores para comparación.

Tabla 3. 6:Tipos de operadores comparadores.

Operador	Descripción
=	Igual
<>	Diferente
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

Fuente:(Reyes, 2008)

3.4.8. Operadores lógicos.

En el caso de los operadores lógicos, no hay que confundirlos con los operadores binarios, estos se utilizan paraestablecer condiciones entre variables y muy similares a los operadores comparadores.

Tabla 3. 7: Tipos de operadores lógicos.

Operador	Descripción
AND	AND Lógico
OR	OR Lógico
XOR	XOR Lógico
NOT	NOT Lógico
NOT AND	NAND Lógico
NOT OR	NOR Lógico
NOT XOR	NXOR Lógico

Fuente:(Reyes, 2008)

3.5. Sistema Operativo Android.

Android es un sistema operativo basado en Linux, diseñado principalmente para dispositivos móviles con pantalla táctil como teléfonos inteligentes y tabletas"(Gironés, 2013). "Fue desarrollado inicialmente por Android Inc. y posteriormente en el 2005 fue comprada por Google, uno de los principales financiadores de la compañía. (…) Android se dio a conocer en el 2007 con lafundación de la *Open Handset Alliance*, un consorcio de empresas de hardware,software y telecomunicaciones dedicadas a la promoción de estándares de código libre para dispositivos móviles." (Gargenta, 2011)

Aunque el Android tiene apenas unos años de vida, la librería Java de Android consiste ya en más de 150 paquetes (APIs), que contienen miles de clases, métodos, interfaces y constantes. En la página web de *Android Developers* se encuentran documentadas todas los tipos de clases. La vasta extensión del software de desarrollo de Android (SDK) puede resultar

extremadamente abrumadora para el que se acerca por primera vez a este sistema (Amaro Soriano, 2012).Según Gironés (2013): La telefonía móvil y los dispositivos electrónicos (tablets) con sistema operativo Android, están cambiando la sociedad actual de una forma tan significativa como lo ha hecho Internet. Dicha evolución ha permitido que nuevos terminales ofrezcan igual capacidades a la de un ordenador, estos ya permiten leer nuestro correo (e-mail) e inclusive navegar por Internet.A continuación se describen los requerimientos que necesita Android.

3.5.1. Android SDK.

El SDK de Android proporcionatoda la biblioteca API, así como las herramientas de desarrollo mediante la cual pueden crear, probar y depurar aplicacionespara Android, lo interesante de SDK es que permite soportar plataformas de los sistemas operativos Linux, Max OS X 10.4.9 o posterior, y Windows XP o posterior(Developers, 2013).

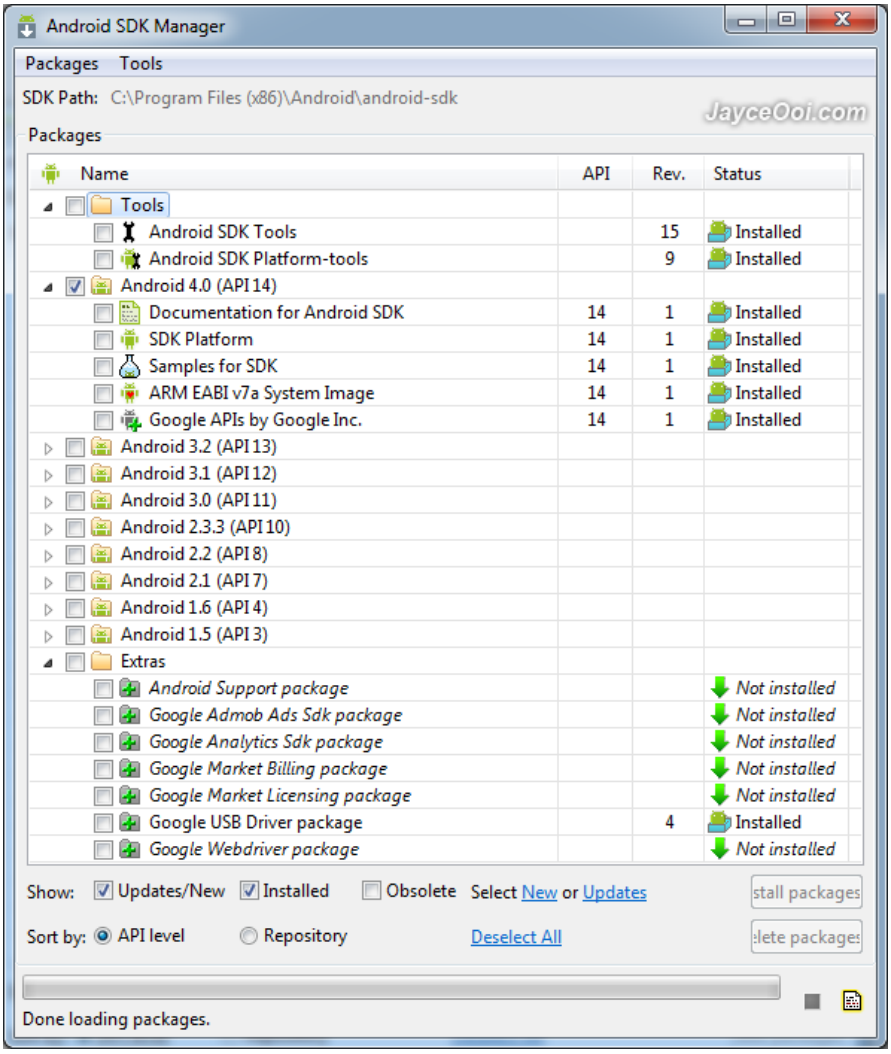


Figura 3. 15: Android SDK Manager.
Fuente: http://speakingin.net/wp-content/uploads/2012/04/Android-SDK-Manager.png

En la figura 3.15 se muestra la ventana de Android SDK Manager (esto aparece una vez instalado SDK), la misma nos indica un listado de las versiones disponibles del sistema operativo desarrolladas hasta la actualidad, y cada una con sus respectivas utilidades y funcionalidades. Es decir, que las personas interesadas en descargar otras versiones lo pueden hacer sin ningún inconveniente.

Android SDK se encargade compilar cada una de las aplicaciones (tanto recursos como los ficheros), una vez compilado muestra el archivo cuya extensión es “Apk”, es decir, que este ejecutable (Apk) se lo puede instalar en dispositivos electrónicos sean estos celulares (tipo Smartphone), tablets, etc.

3.5.2. Entorno de Desarrollo Integrado (IDE).

El IDE es una aplicación de software que brinda servicios integrales a los programadores para el desarrollo de software, es decir que dispone de tres elementos muy importantes:

1. Editor de código fuente.
2. Constructorautomatizado de herramientas, y un
3. Depurador.

La mayoría de entornos de programación Android disponen su propioIDE, así por ejemplo Developers (2013) indica que“Eclipse constituye la plataforma recomendada por la página del sistemaoperativo”. El IDE requiere de un complemento Android Development Tools Plugin (ADT), esto permitirá crear aplicaciones en Android.En la figura 3.16 se muestra el entorno de desarrollo integrado de Eclipse para la edición de ficheros.

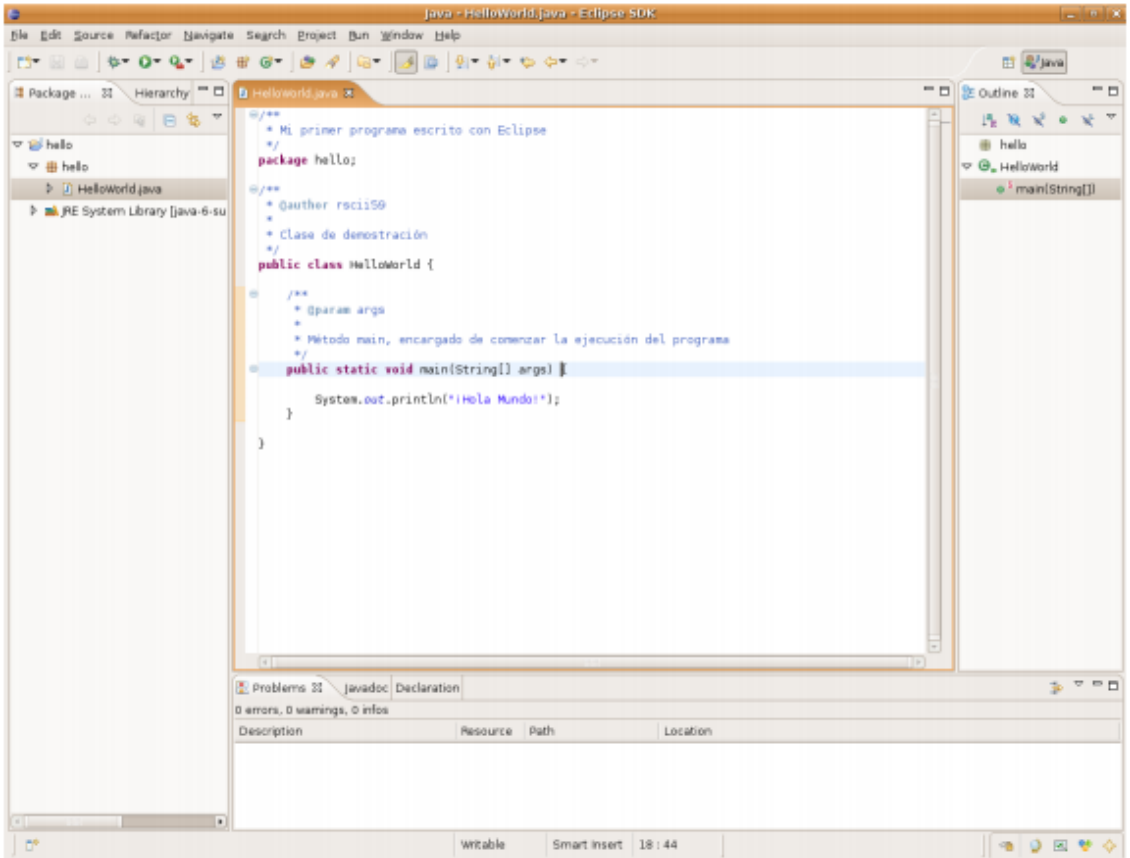


Figura 3. 16: Android SDK Manager.
Fuente:http://www-gris.det.uvigo.es/wiki/pub/Main/MiscResources/Manual_Eclipse.pdf

3.5.3. Java Development Kit (JDK).

Es un paquete de programación de software (SDK) para producir programas en Java. El JDK está desarrollado por Oracle de la división JavaSoft de Sun Microsystem. Las versiones más recientes incluyen la arquitectura de componentes JavaBeans y soporte para JDBC.[8] JDK es el entorno de desarrollo de programas cuya programación es orientada a objetos más utilizados, sobre todo en internet (web). El JDK es muy utilizado por Android en la creación y depuración de aplicaciones desde la terminal.

En la figura 3.17 se muestra la ventana del JDK instalado en una Tablet que tenga disponible el Sistema Operativo Android.

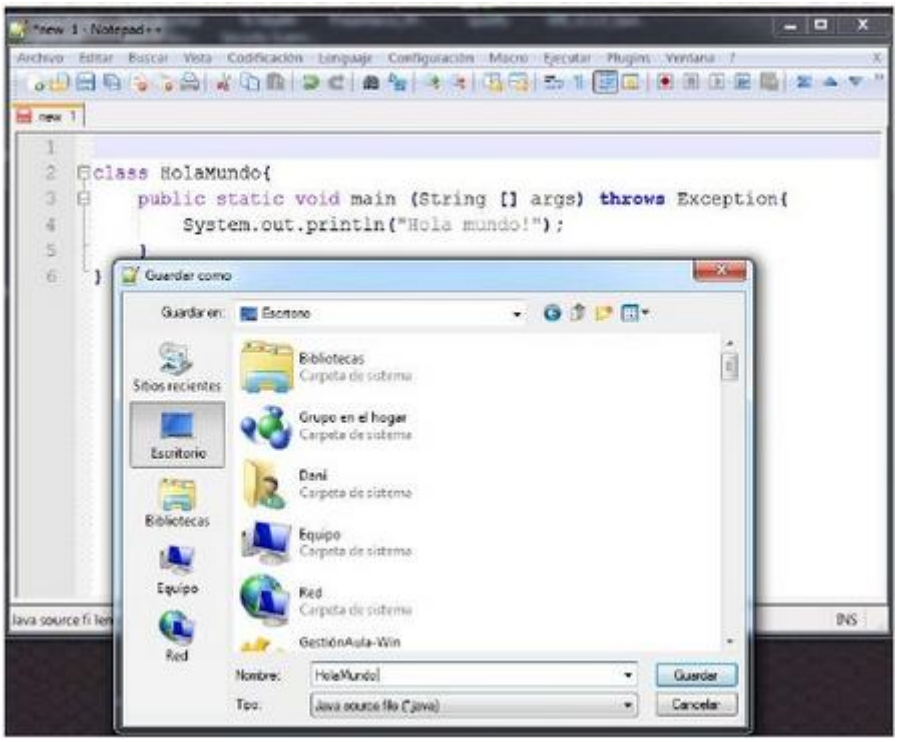


Figura 3. 17: Android JDK.
Fuente:http://www.slideshare.net/DaniSantia/instalacin-de-java-development-kit-jdk

3.5.4. Native Development Kit (NDK).

[8] Online recuperado de la página web:http://citec.tol.itesm.mx/guias/tol_jdk.pdf

El NDK dispone de herramientas para implementar partes de una aplicación mediante lenguajes de programación de alto nivel, tales como C y C++ a través de sus códigos nativos.Aunque no todas las aplicaciones pueden ser útiles Para ciertos tipos de aplicaciones esto puede ser de ayuda, sin embargo, para la mayoría de las aplicaciones no es necesario su uso y no se requiere su instalación. (Developers1, 2013)

3.5.5. Android Virtual Device (AVD)

Un AVD (dispositivo virtual de Android) es una configuración del emulador que permite modelar un dispositivo real mediante la definición de las opciones de software que emula el emulador de Android y hardware. La forma más sencilla de crear un AVD es usar la gráfica Administrador de AVD , que se inicia a partir de Eclipse. También puede iniciar el Administrador de AVD de la línea de comandos al llamar alandroide de herramientas(Developers2, 2013).

CAPÍTULO 4: DESARROLLO EXPERIMENTAL

En el presente capítulo se mostrará tanto el desarrollo del hardware y de la programación (software) en Android a través de AppInventor y del Microcontrolador (PIC16F886). Para el Hardware se diseñó un pequeño robot con una aplicación específica, en tanto para la programación, AppInventor permite la comunicación bluetooth hacia el microcontrolador (placa de entrenamiento). Esto nos permitió diseñar algo novedoso que hasta la presente no se ha realizado en la Facultad de Educación Técnica para el Desarrollo, lo que a futuro los estudiantes podrán desarrollar programación Android en dispositivos electrónicos, es decir, tanto para Smartphone (telefonía móvil) como tabletas compatibles con el sistema operativo en mención.

4.1. Diseño del Robot Recolector Radiocontrolado.

En si el diseño mecánico del Robot Recolector consiste de una placa hexagonal, en la misma se colocaran las tarjetas electrónicas tales como microcontrolador, puente H, módulo bluetooth y servo motor (para el

movimiento de un brazo); y de una pinza para recolectar objetos, en la misma se coloca otro servo motor, tal y como se ilustra en la figura 4.1.

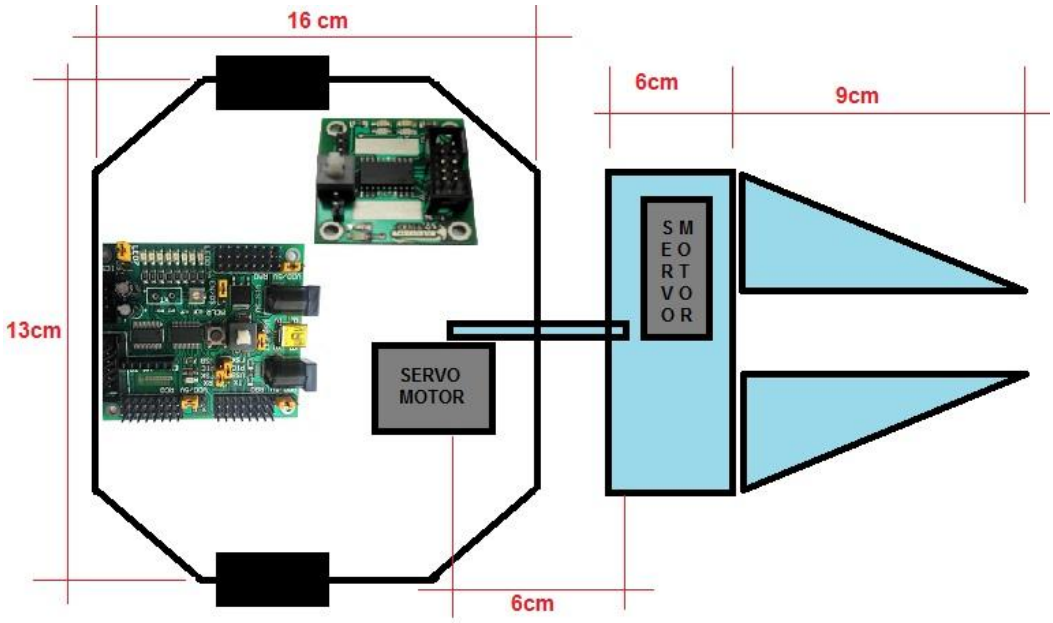


Figura 4. 1: Diseño Mecánico del Robot Recolector.
Fuente: Los autores

De acuerdo a la figura 4.1, las dimensiones del robot son 13 cm de ancho y 29 cm de largo, como podemos observar el diseño no es muy complejo, solo se necesita de un poco de creatividad, aunque inicialmente se pensó en el desarrollo de un robot solamente móvil, es decir, sin el brazo recolector. La idea del brazo recolector, fue para que los estudiantes o lectores del presente trabajo de titulación no piense que fácil es programar una Tablet o que exista alguna aplicación existente para manipular (controlar) un robot móvil.

En los siguientes acápites se describe el hardware (PIC16F886) con su respectiva programación (software) en Pic Basic o Mikro Basic y AppInventor.

4.2. Hardware del Robot Recolector Radiocontrolado.

Para el desarrollo del robot no autónomo, se pensó en una aplicación sencilla pero robusta, el robot puede agarrar objetos pequeños mediante dos tenazas que son controladas desde una Tablet (con Sistema Operativo Android). El diseño se lo realizó a través de la tarjeta electrónica M.E. I&T04, el mismo es un módulo de entrenamiento y desarrollo que permite ejecutar varias

aplicaciones, esto se debe al microcontrolador PIC16F886 que puede ser programado para cumplir distintas funciones.

En la figura 4.2 se muestra el diagrama esquemático realizado en Proteus Profesional, este Hardware consta de una batería de 6 V, módulo de entrenamiento (M.E. I&T04), módulo del puente H (P.H. I&T04), módulo de comunicación bluetooth y los servomotores.El algoritmo en el receptor se lo realiza, mediante el diseño de 3 tramas de datos para el control individual de las tres partes del Robot.

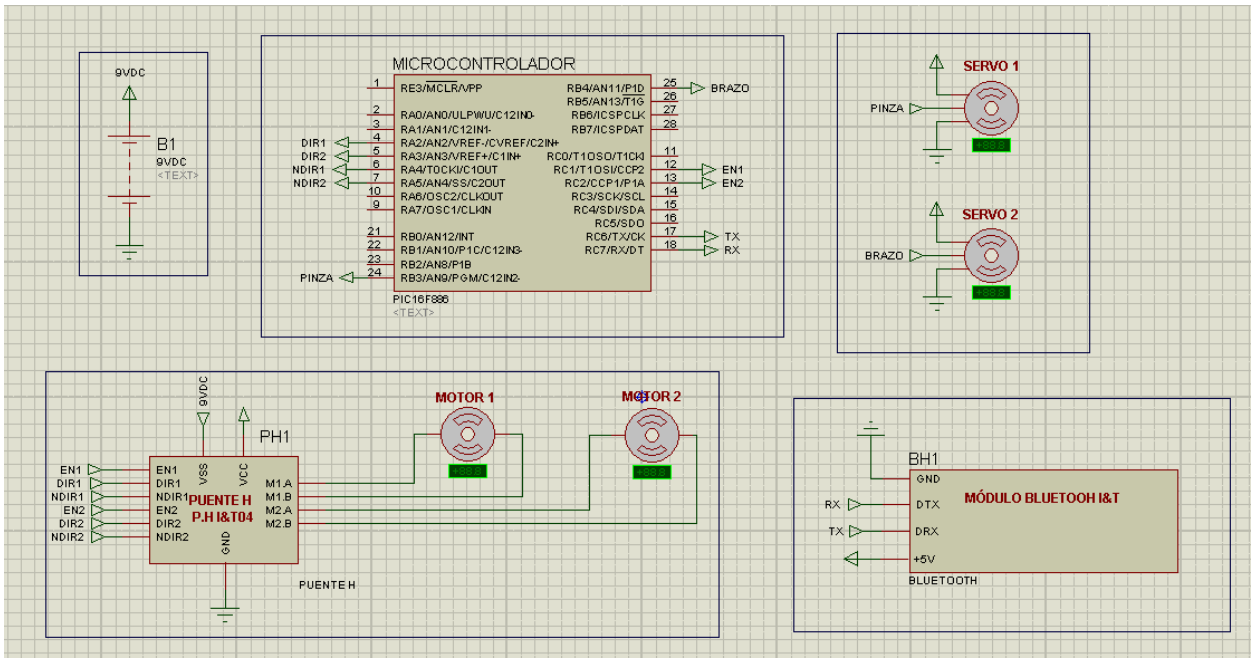


Figura 4. 2: Diagrama Esquemático del Robot Recolector Radiocontrolado.
Fuente: Los autores

Primero se valida que los datos que se están recibiendo sean para el recolector mediante el byte o código inicio, luego se consulta que se desea controlar carro (código: 0x37, véase la tabla 4.1), pinza (código: 0x32, véase la tabla 4.2) o carro (código: 0x34, véase la tabla 4.3).

INICIO	CARRO	VEL-MOTOR 1	DIR-MOTOR 1	VEL-MOTOR 2	DIR-MOTOR 2	FIN
0X24	0X37	0-255	0 o 1	0-255	0 o 1	0X46

Tabla 4. 1: Desplazamiento de Control de Carro
Fuente: Los autores

En la tabla 4.1 se muestre la trama para el desplazamiento del control de carro, y dicha trama es la siguiente:

a. Podremos manejar el desplazamiento del carro mediante la información vel-motor1, dir-motor1, vel-motor2, dir-motor2.
b. El valor de vel-motor1 puede ser un entero entre 0 y 255 siendo mínimo=0 y máximo=255 para el control de la velocidad del motor 1.
c. El valor de dir-motor1 puede ser un entero de 0 o 1 siendo hacia atrás=0 y hacia delante=1, para el control de la dirección de giro del motor 1.
d. El valor de vel-motor2 puede ser un entero entre 0 y 255 siendo mínimo=0 y máximo=255 para el control de la velocidad del motor 2.
e. El valor de dir-motor2 puede ser un entero de 0 ó 1 siendo hacia atrás=0 y hacia delante=1, para el control de la dirección de giro del motor 2.

INICIO	PINZA	INC_ANGULO PINZA	FIN
0X24	0X32	0x30 - 0x31	0X46

Tabla 4. 2: Trama para el control de pinza.
Fuente: Los autores

En la tabla 4.2 se muestra la trama del control de pinza, y la trama es la siguiente:

a. Podremos manejar el desplazamiento de la pinza (abrir o cerrar),mediante la información inc_angulo pinza.
b. El valor de inc_angulo pinza puede ser un entero entre 0 y 1, siendo decrementar ángulo pinza = 0 e incrementar_angulo pinza =1, por cada trama recibida se decrementa o incrementa el ángulo de la pinza,para lo cual el rango de movilidad de la pinza se encuentra entre 150 y 210 grados

INICIO	BRAZO	INC_ANGULO BRAZO	FIN
0X24	0X34	0x30 - 0x31	0X46

Tabla 4. 3: Trama para el control de brazo.
Fuente: Los autores

En la tabla 4.3 se muestra la trama del control de brazo, cuya trama es la siguiente:

a. Podremos manejar el desplazamiento del brazo (subir o bajar),mediante la información inc_angulo brazo.
b. El valor de inc_angulo brazo puede ser un entero entre 0 y 1, siendo decrementar ángulo brazo = 0 e incrementar_angulo brazo =1, por cada trama recibida se decrementa o incrementa el ángulo de la brazo, para lo cual el rango de movilidad del brazo se encuentra entre 150 y 220 grados.

4.3. Software del Robot Recolector Radiocontrolado.

Para poder controlar el robot recolector a través de una Tablet, se procedió a desarrollar la programación en AppInventor (para Sistema Operativo Android, véase la figura 4.3). Según (MIT, 2013)AppInventor: permite desarrollar aplicaciones para los teléfonos Android con un explorador Web y, o bien un teléfono conectado o un emulador de teléfono en pantalla. Los servidores del MIT AppInventor almacenan su trabajo y ayuda a mantener un registro de sus proyectos.

Figura 4. 3: Icono AppInventor de Android.
Fuente: (MIT, 2013)

Es decir, que AppInventor es un entorno de desarrollo visual web, que mediante el uso de bloques alestilo de piezas de rompecabezas, permite construir aplicaciones que no requieren un uso extensivo de las librerías de Android. Al estar formado por bloques visuales, el framework permite obtener aplicaciones funcionales, sin la necesidad de escribir líneas de código en algún lenguaje de programación(Iskandar Morine, 2013). “Fue anunciado en 2010 por Google, y está basado en la biblioteca Open Blocks Java, del MIT” (MIT, 2013).En la figura 4.4 se muestra la ventana de la interfaz gráfica de AppInventor en la misma se va a realizar la programación del robot recolector.

En la figura 4.5 se ilustran cada uno de los componentes utilizados en la programación del robot recolector, tanto botones, cuadros de texto, imágenes, etc. Los elementos deben ser arrastrados a la ventana de vista para su respectivo procedimiento de programación.

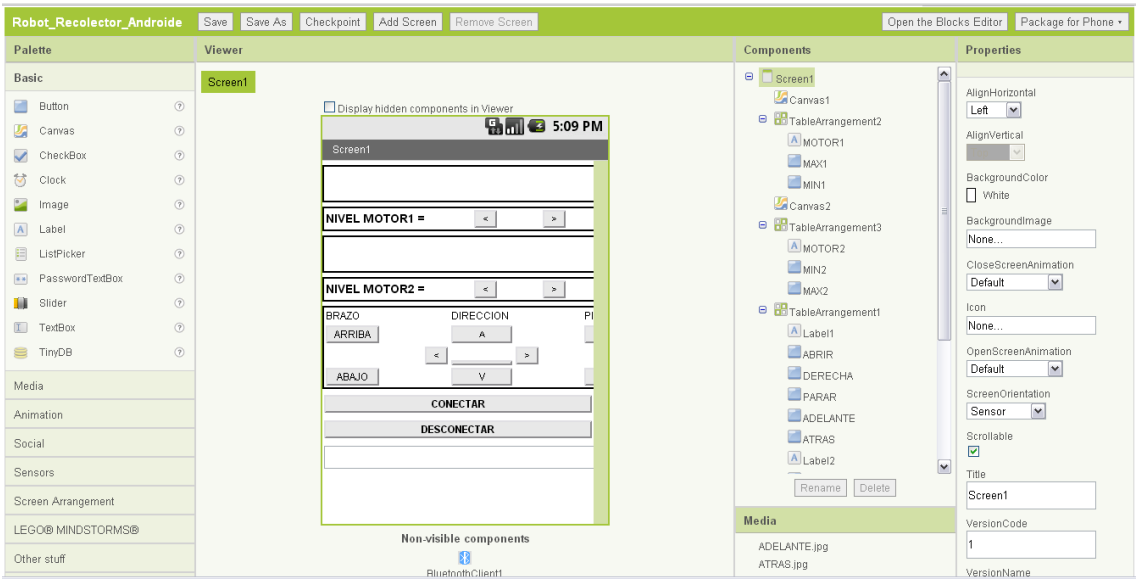


Figura 4. 4: Interfaz gráfica AppInventor.
Fuente: Los Autores

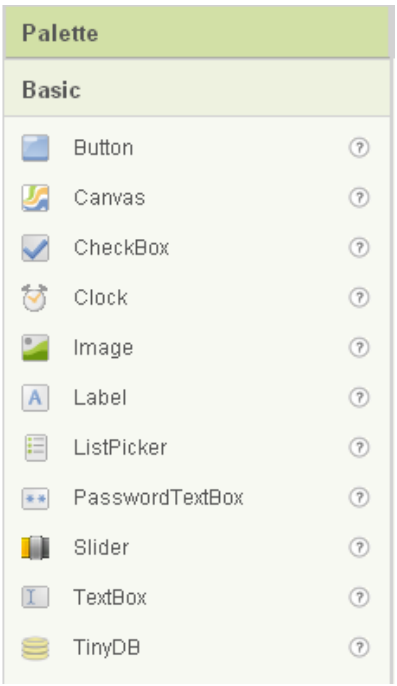


Figura 4. 5: Menú Paleta.
Fuente: Los Autores

Mientras que la figura 4.6 nos muestra la ventana que permite presentar cada uno de los elementos a utilizar dependiendo de los elementos arrastrados por el desarrollador. Es decir, que esta ventana representará la aplicación (robot recolector) y su interfaz gráfica.

Figura 4. 6: Menú Paleta.
Fuente: Los Autores

Se crea un menú de componentes, tal como se muestra en la figura 4.7, dicho menú es en sí una interfaz que permite alos usuarios tomar decisiones conforme a la programación, así como eliminar componentes y elementos no utilizados, y a la vez presenta una lista de elementos utilizados y a utilizar por el usuario. Los componentes se presentan en esta ventana una vez que el desarrollador incorpore sus herramientas o elementos a trabajar.

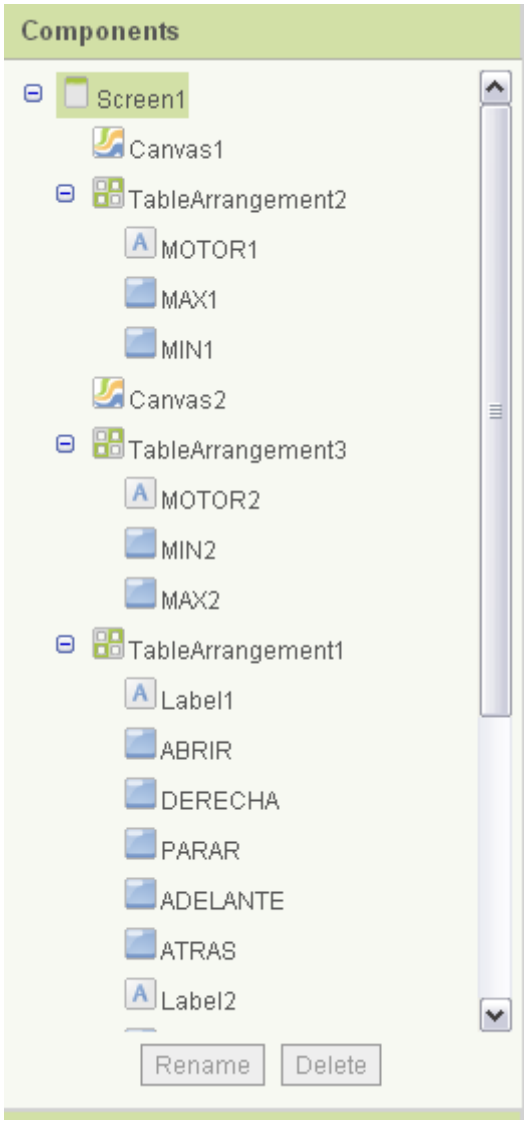


Figura 4. 7: Menú Componentes.
Fuente: Los Autores

Posteriormente, se creará el menú propiedades (véase la figura 4.8) la misma que permitirá operar la interfaz de presentación de los componentes, como por ejemplo:

- ✓ Editar tamaño de texto.
- ✓ Negritas
- ✓ Subrayado
- ✓ Color
- ✓ Posición de los elementos

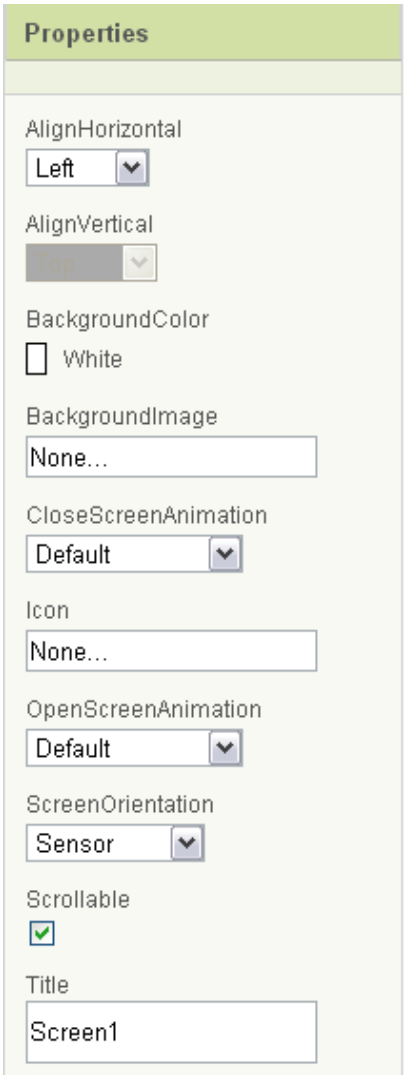


Figura 4. 8: Menú Propiedades.
Fuente: Los Autores

4.3.1. Código de programación

A continuación se procederá a crear un Slider conocido como código de programación, este segmento de código permite crear un desplazamiento de una barra que permite coordinar con el microcontrolador receptor las limitantes para imponer un determinado valor de velocidad. En la figura 4.9 se muestran estas funciones que son utilizadas para los dos servos motores del robot recolector.

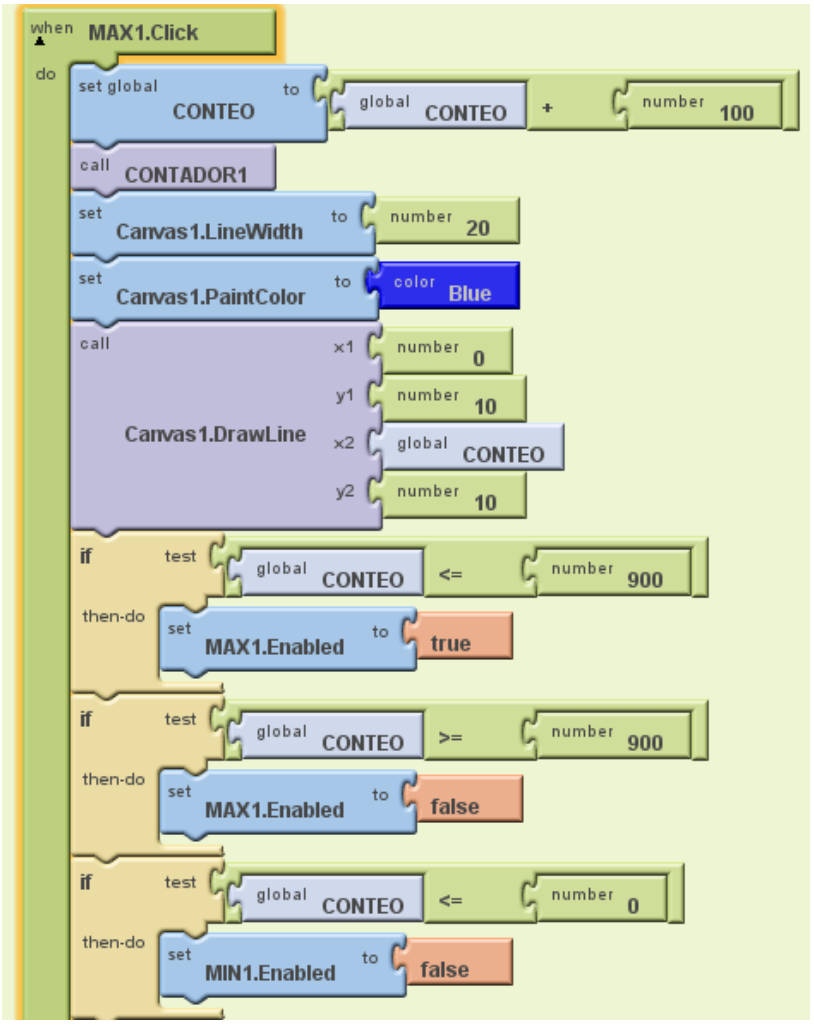


Figura 4. 9: Algoritmo Slider.
Fuente: Los Autores

En la figura 4.10 se muestra la función para el funcionamiento del servo motor, es decir, que dicho segmento de código permite al servomotor se imponga en un determinado ángulo de apertura para la recolección de objetos.

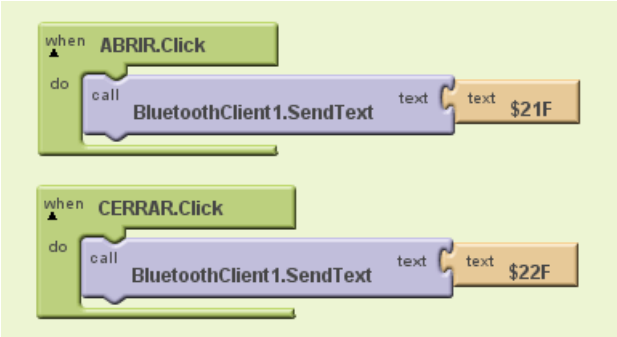


Figura 4. 10: Algoritmo Servo Motor.
Fuente: Los Autores

4.3.2. Algoritmos de movimiento del robot.

A continuación se diseñan los algoritmos de programación, que permiten que el robot pueda desplazarse a través de una superficie plana, siempre dependiendo de los botones de direccionamiento presionados. En la figura 4.11 se muestra la función que permite al robot recolector avanzar hacia adelante.

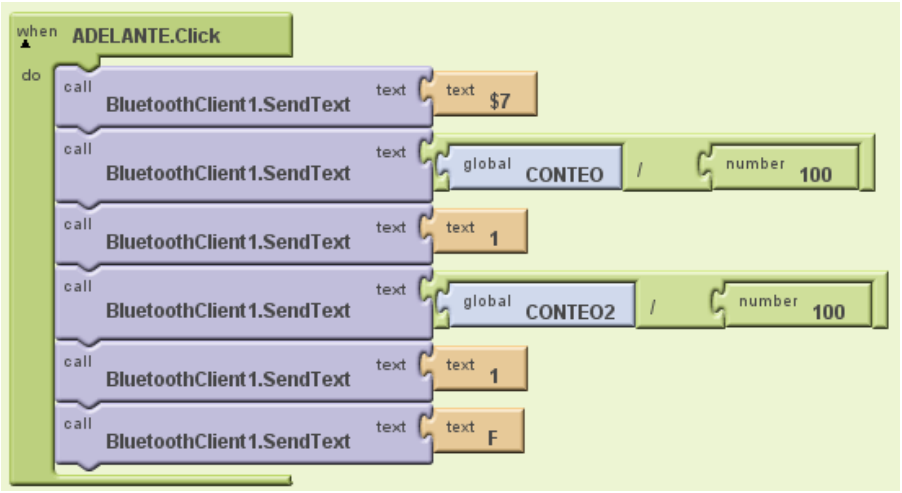


Figura 4. 11: Algoritmo del movimiento hacia adelante.
Fuente: Los Autores

En las figuras 4.12, 4.13, 4.14 y 4.15 donde cada una ilustra la función que permite al robot recolector desplazarse hacia la derecha, hacia la izquierda y hacia atrás respectivamente.

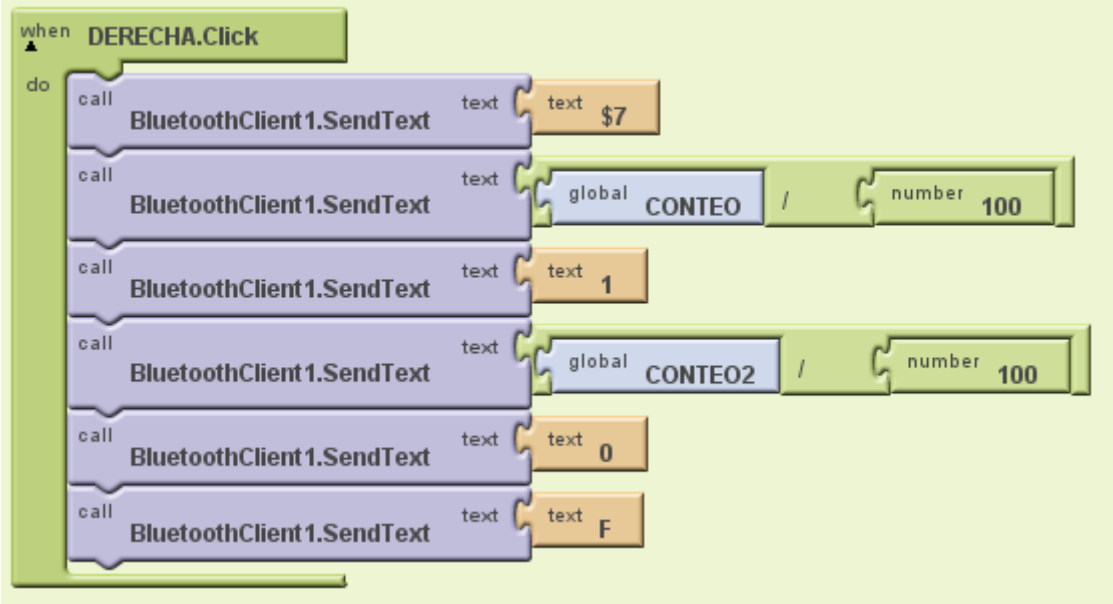


Figura 4. 12: Algoritmo del movimiento hacia la derecha.
Fuente: Los Autores

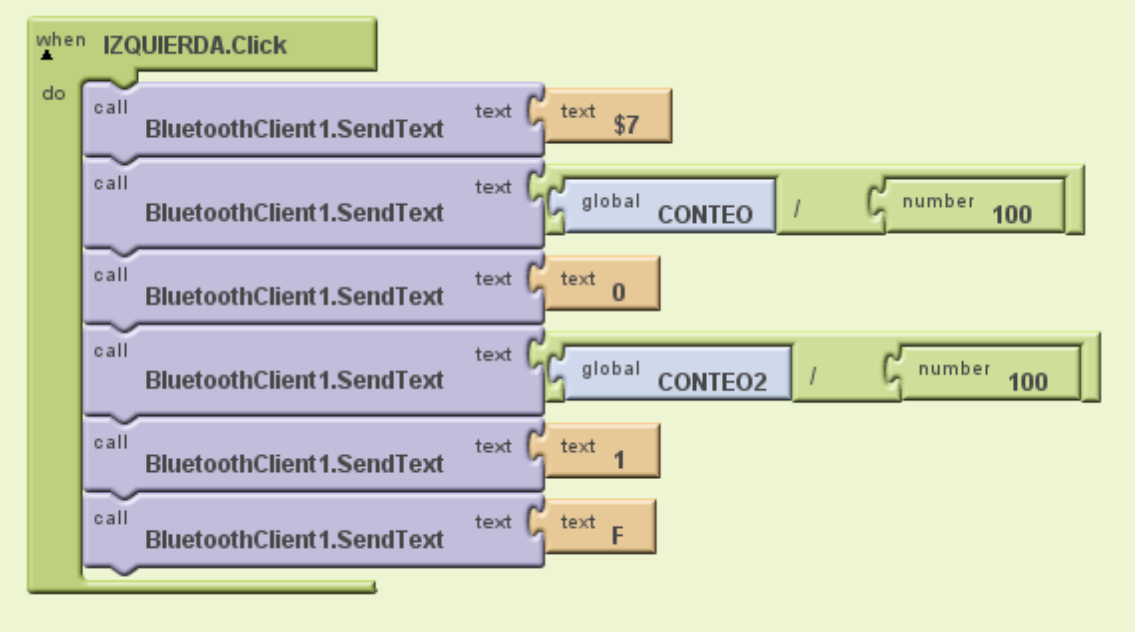


Figura 4. 13: Algoritmo del movimiento hacia la izquierda.
Fuente: Los Autores

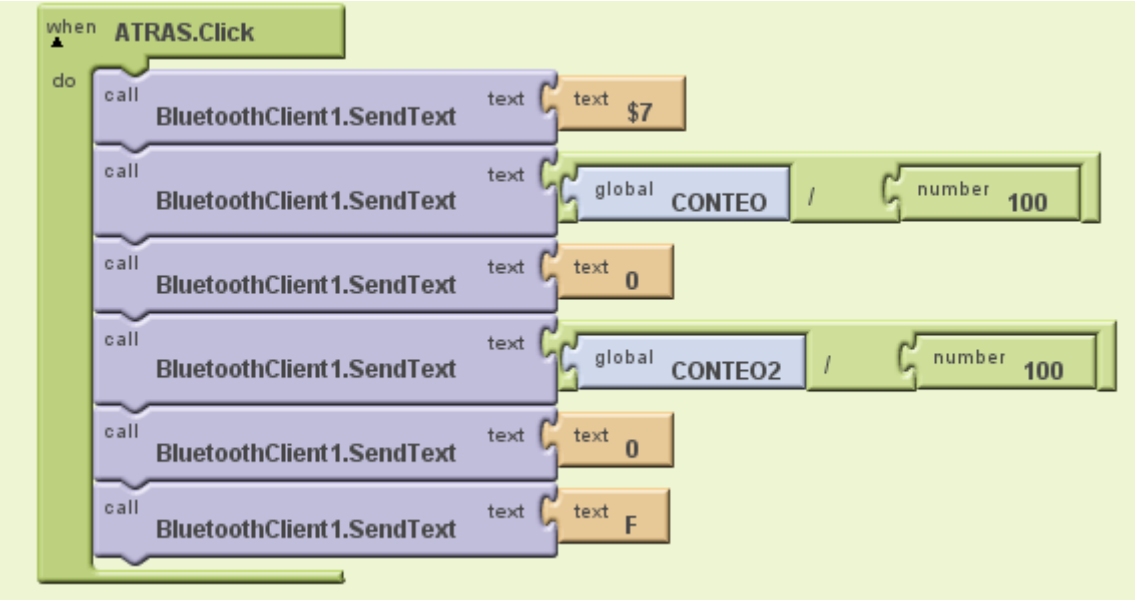


Figura 4. 14: Algoritmo del movimiento hacia atrás.
Fuente: Los Autores

4.4. Programación del PIC16F886.

A continuación se muestran las líneas de programación realizada en Pic Basic, la misma que servirá para posteriormente grabar en el PIC16F886.

```
program Recolector_RadioControlado
symbol DIR1=PORTA.2
symbol NDIR1=PORTA.4
symbol DIR2=PORTA.3
symbol NDIR2=PORTA.5

dim indice,byte_recibido, angulo_pinza
,angulo_brazo,inc_angulo_pinza,inc_angulo_brazo,orden as byte
dim vel_motor1,vel_motor2,dir_motor1,dir_motor2 as integer

dim Servo        as byte[8]
    Salidas      as byte
    Index        as byte
    Paso         as byte
    n_servo      as byte
    MaskPort     as byte

    sub procedure interrupt
  'TIMER 0
  if INTCON.T0IF=1 then
if paso=0 then
MaskPort=salidas and n_servo ' Mascara para las salidas
PORTB.3=MaskPort.0           ' Servo 1
     PORTB.4=MaskPort.1           ' Servo 2
TMR0=230                    ' Delay de 0.3ms
Inc(paso)
else
     if paso=1 then
TMR0=Servo[index]          ' Anchura de pulso del servo
Inc(paso)
     else
       if paso=2 then
PORTB=0
       TMR0=Servo[index]        ' Valor menos periodo de servo
       paso=0
       n_servo=n_servo<<1       ' Siguiente servo
inc(index)
       if n_servo=0 then
        n_servo=1
        index=0
         end if
       end if
     end if
end if

   INTCON.T0IF =0               ' Reseteamos la bandera de la interrupcion
end if
end sub
```

```
SUB PROCEDURE SET_MOTOR(DIM MOTOR1,MOTOR2 AS INTEGER)
 '******************** MOTOR 1***************
  IF(MOTOR1<0)THEN
    MOTOR1=-MOTOR1
    DIR1=0
    NDIR1=1
  ELSE
  IF(MOTOR1>0)THEN
    DIR1=1
    NDIR1=0
  END IF
  END IF
  '******************** MOTOR 2***************
  IF(MOTOR2<0)THEN
    MOTOR2=-MOTOR2
    DIR2=0
    NDIR2=1
  ELSE
  IF(MOTOR2>0)THEN
    DIR2=1
    NDIR2=0
  END IF
  END IF
  PWM1_Set_Duty(MOTOR2)
  PWM2_Set_Duty(MOTOR1)
END SUB

main:
 OSCCON=0X75
 OPTION_REG=%10000011            ' Prescaler 1:16, PORTB pull-up
deshabilitado
INTCON.T0IF=0                 ' TOIF=0  borra flag interrupcion del timer
 INTCON.T0IE=1
 TMR0 = 0                     ' Reset del Timer0
 INTCON.GIE=1                  ' Interrupciones globales
 TRISA=0
 TRISB=0
 TRISC=%10000000
 ANSEL=0
 ANSELH=0
 indice=0
 Index=0
 Paso=0
 orden=0
 N_servo=1
 salidas=0xFF
```

```
 PORTA=0
 PORTB=0
 PORTC=0
UART1_Init(9600)
 Delay_ms(100)
 PWM1_Init(1000)
 PWM2_Init(1000)
 PWM1_Start()
 PWM2_Start()
 PWM1_Set_Duty(0)
 PWM2_Set_Duty(0)
Delay_ms(200)
 angulo_pinza=150
 angulo_brazo=220
 Servo[0]= angulo_pinza
 Servo[1]= angulo_brazo
Delay_ms(1000)
 WHILE(1)

 if(UART1_Data_Ready = 1)then
      byte_recibido=UART1_Read()
    select case indice
    case 0
      if(byte_recibido=0x24)then  'BYTE INICIO
        indice=1
      else
        indice=0
      end if
    case 1
      if(byte_recibido=0x37)then  'ID  CARRO
        indice=2
        orden=1
      else
       if(byte_recibido=0x32)then  'ID PINZA
        indice=6
        orden=2
      else
       if(byte_recibido=0x34)then  'ID BRAZO
        indice=7
        orden=3
      else
        indice=0
      end if
      end if
      end if
 '%%%%%%%%%%%%%%%%%%% DATOS CARRO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
```

```
case 2
       vel_motor1=byte_recibido         'DATOS
        indice=3
      case 3
        dir_motor1=byte_recibido         'DATOS
        indice=4
      case 4
       vel_motor2=byte_recibido         'DATOS
        indice=5
      case 5
        dir_motor2=byte_recibido         'DATOS
        indice=8
 '%%%%%%%%%%%%%%%%%% DATOS PINZA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
      case 6
       inc_angulo_pinza=byte_recibido         'DATOS
       indice=8
      case 7
       inc_angulo_brazo=byte_recibido         'DATOS
indice=8
      case 8
        if(byte_recibido=0x46)then    'FIN
          indice=9
       else
          indice=0
       end if
      end select
end if
if(indice=9)then
if(orden=1)then
'%%%%%%%%%%%%%%%%%%%%%%%%% CONTROL CARRO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
        dir_motor1=dir_motor1-0X30
        dir_motor2=dir_motor2-0X30
        vel_motor1=vel_motor1-0X30
        vel_motor2=vel_motor2-0X30
        vel_motor1=(255/9)*vel_motor1
vel_motor2=(255/9)*vel_motor2

if((dir_motor1=1) AND (dir_motor2=1))then
         SET_MOTOR(vel_motor1,vel_motor2)
        else
        if((dir_motor1=1) AND (dir_motor2=0))then
         SET_MOTOR(vel_motor1,-vel_motor2)
        else
        if((dir_motor1=0) AND (dir_motor2=1))then
```

```
         SET_MOTOR(-vel_motor1,vel_motor2)
        else
          SET_MOTOR(-vel_motor1,-vel_motor2)
        end if
        end if
        end if
end if
if(orden=2)then
'%%%%%%%%%%%%%%%%%%%%%%%% CONTROL PINZA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
inc_angulo_pinza=inc_angulo_pinza-0X30

if(inc_angulo_pinza=1)then   'INCREMENTAR ANGULO PINZA
          if(angulo_pinza >150)then
dec(angulo_pinza )
          else
           angulo_pinza  =150
end if
           Servo[0]=angulo_pinza
        else
        if(inc_angulo_pinza=2)then   'DECREMENTAR ANGULO PINZA
          if(angulo_pinza <210)then
Inc(angulo_pinza)
          else
            angulo_pinza = 210
end if
            Servo[0]=angulo_pinza
         end if
         end if
else if(orden=3)then
inc_angulo_brazo=inc_angulo_brazo-0X30
 '%%%%%%%%%%%%%%%%%%%%%%%% CONTROL BRAZO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
         if(inc_angulo_brazo =1)then   'INCREMENTAR ANGULO BRAZO
          if(angulo_brazo  >150)then
           dec(angulo_brazo )
          else
           angulo_brazo =150
end if
           Servo[1]=angulo_brazo
        else
if(inc_angulo_brazo =2)then   'DECREMENTAR ANGULO BRAZO
          if(angulo_brazo  <220)then
            Inc(angulo_brazo )
          else
            angulo_brazo  = 220
```

```
end if
          Servo[1]=angulo_brazo
        end if
        end if
end if
end if
indice=0
    orden=0
end if
 WEND
end.
```

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES.

5.1. Conclusiones.

- A través de la fundamentación teórica también conocida como Estado del Arte, nos permitió comprender la importancia de los microcontroladores que permiten realizar diferentes aplicaciones relacionadas a las Ciencias Aplicadas, tanto para Carreras como Ingeniería en Telecomunicaciones como la de Electrónica en Control y Automatismo, donde se pueden integrar cualquier medio de comunicación, específicamente en el presente trabajo la tecnología Bluetooth (/medio de transmisión) a través de una Tablet para que se pueda comunicar con el Robot Recolector.

- Se describió las plataformas de programación de alto nivel, tales como Pic Basic, la misma que sirve para programar el PIC16F886 (hardware); y Android, que es una plataforma de programación abierta (open source) que opera de manera conjunta con AppInventor, lo que al final nos permitió desarrollar la aplicación del Robot Recolector.

- Se pudo comprobar o validar el presente trabajo de titulación que es el diseño de un Robot Móvil (Robot Recolector) controlado por una Tablet que mediante el desarrollo en AppInventor se comunica a través de Bluetooth, la aplicación fue expuesta en la Feria de Universidades organizada por la Unidad Educativa Mariscal Sucre el mismo que cumplió con los objetivos propuestos.

5.2. Recomendaciones.

- Es necesario que la Facultad de Educación Técnica para el Desarrollo, compren licencias profesionales de algunos programas que permiten desarrollar proyectos mediante los microcontroladores, tales como Pic Basic, Mikro Basic, Microcode Estudio y Proteus Profesional.

- Incluir en la malla curricular materias optativas que involucren programación de aplicaciones de móviles a través de AppInventor que pertenece al Sistema Operativo Android y así contribuir como temas de investigación formativa o básica con aplicaciones en Telecomunicaciones.

- Incentivar a la comunidad estudiantil de la Facultad de Educación Técnica para el Desarrollo a través de Concursos Internos de Proyectos Básicos que involucren dispositivos electrónicos y así mejorar el proceso de aprendizaje.

REFERENCIAS BIBLIOGRÁFICAS

Amaro Soriano, J. E. (2012). *Android: Programación de dispositivos móviles a través de ejemplos* (1era ed.). Barcelona: Marcombo.

Caprile, S. R. (2013). *Desarrollo con Microcontroladores ARM.* Buenos Aires: Caprile.

Developers. (27 de 05 de 2013). *Developers Android*. Obtenido de http://developer.android.com/sdk/index.html

Developers1. (20 de 05 de 2013). *Developers Android*. Obtenido de http://developer.android.com/tools/sdk/ndk/index.html

Developers2. (21 de 05 de 2013). *Developer Android*. Obtenido de http://developer.android.com/tools/devices/index.html

Durán Rodríguez, L. (2006). *Ampliar, configurar y repara su PC.* Barcelona: Marcombo.

García Barba, M. Á. (12 de 12 de 2012). *Universidad Politécnica de Cartagena.* Obtenido de Repositorio: http://repositorio.bib.upct.es/dspace/bitstream/10317/2742/1/pfc4285.pdf

Gargenta, M. (2011). *Learning Android.* Estados Unidos: O´Reilly.

Gironés, J. T. (2013). *El Gran Libro de Android* (3era ed.). Barcelona: Marcombo.

GRIDLING, G. y. (2007). *Introduction to Microcontrollers.* Viiena, Austria: Vienna University of Technology.

Hermosa Donate, A. (2010). *Electrónica digital fundamental y programable: curso profesional teoría-práctica.* Barcelona: Marcombo.

Iskandar Morine, R. J. (2013). *Estudio comparativo de alternativas y frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android.* Barcelona: Universidad Politécnica de Catalunya.

Mandado Pérez, E., Menéndez Fuertes, L., Fernández Ferreira, L., & López Matos, E. (2007). *Microcontroladores PIC: Sistema Integrado para el Autoaprendizaje.* Barcelona, España: Marcombo.

Mandado R., Y., & Mandado P., E. (2007). *Sistemas Electrónicos Digitales.* Barcelona: Marcombo.

MIT. (25 de 05 de 2013). *AppInventor MIT*. Obtenido de http://appinventor.mit.edu/explore/sites/teach.appinventor.mit.edu/files/MIT%20App%20Inventor%20Development%20Overview_0.pdf

Ramírez Benavides, K. D. (23 de 05 de 2013). *Escuela de Ciencias de la Computación e Informática.* Obtenido de http://www.kramirez.net/ci-2657/materialci2657_ii-2013/

Reyes, C. A. (2008). *Microcontroladores PIC Programación en Basic.* Quito, Ecuador: RISPERGRAF.

Staff, U. (2011). *Microcontroladores: Funcionamiento, Programación y Usos Prácticos.* Buenos Aires: USERS.

Valdés Pérez, F. E., & Pallás Areny, R. (2007). *Microcontroladores: fundamentos y aplicaciones con PIC.* Cataluña, España: Marcombo.

Valverde Villarán, A. (2005). *Sistema de Desarrollo para el Microcontrolador PIC18F452.* Sevilla: Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación.

Verle, M. (2010). *PIC Microcontrollers - Programming in Basic.*Belgrado: mikroElektronika.