ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

"DISEÑO Y CONSTRUCCION DE MICROROBOT BAILARIN.".

TESIS DE GRADO

PREVIA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN ELECTRÓNICA Y COMPUTACIÓN.

PRESENTADO POR:

FREDI DANILO GARCÍA NINABANDA

MAURO PATRICIO NÚÑEZ YÁNEZ

RIOBAMBA - ECUADOR

2009

A Dios por ser nuestra fuente de fortaleza en todo momento.

A nuestros padres y familia, porque gracias a su esfuerzo, apoyo incondicional, amor y paciencia logramos alcanzar esta meta, a nuestro tutor por transmitirnos su forma práctica de resolver los problemas y darnos a la vez la libertad de crear e incentivar nuestras ideas e iniciativas. Y todos aquellos que de alguna u otra forma nos ayudaron a conseguir este logro.

El presente trabajo de titulación está dedicado a nuestra familia que con su esfuerzo ha logrado entregarnos el apoyo necesario para cumplir nuestros sueños y metas. A nuestros hermanos, a quienes les deseamos los mejores éxitos en la finalización de su vida académica.

NOMBRE	FIRMA	FECHA
Dr. Ms.c. Romeo Rodríguez **DECANO DE LA FACULTAD** **DE INFORMÁTICA Y** **ELECTRÓNICA**	……	……
Ing. Paúl Romero **DIRECTOR DE LA** **ESCUELA DE INGENIERÍA** **ELECTRÓNICA.**	……	……
Ing. Hugo Moreno **DIRECTOR DE TESIS**	……	……
Ing. Paúl Romero **MIEMBRO DEL TRIBUNAL**	……	……
Tlgo. Carlos Rodríguez **DIRECTOR DPTO.** **DOCUMENTACION**	……	……
NOTA DE LA TESIS	……	

"Nosotros, **FREDI DANILO GARCÍA NINABANDA** y **MAURO PATRICIO NÚÑEZ YÁNEZ**, somos los responsables de las ideas, doctrinas y resultados expuestos en esta Tesis de Grado, y el patrimonio intelectual de la misma pertenecen a la Escuela Superior Politécnica de Chimborazo"

Fredi Danilo García Ninabanda

Mauro Patricio Núñez Yánez

ÍNDICE DE ABREVIATURAS

ALU	Unidad Aritmético Lógica.
COM	Centro de masa.
COP	Centro de presión o fuerza.
EEPROM	Memoria de solo lectura programable y borrable eléctricamente.
FLASH	Memoria no volátil de bajo consumo.
FRI	Indicador de rotación del pie.
FSR	Registro de Selector de archivo.
GCOM	Punto de proyección del centro de masa
GPR	Registros de propósito General.
HS	Cristal de alta velocidad.
HSEROUT	Salida serial de hardware asíncrono.
IA	Inteligencia Artificial.
INTOSC	Oscilador Interno.
LP	Cristal para baja potencia y bajo consumo de corriente.
MCLR	Master Clear (Reset).
OSC1/CLKIN	Entrada del oscilador (cristal). Entrada de oscilador externo.
PIC	Peripherical Interface Controller.
PWM	Modulación por ancho de pulso.
RAM	Memoria de acceso aleatorio.
RISC	Computadores de Juego de Instrucciones Reducido.
RPM	Revoluciones por minuto.
SFR	Registros de Función Específica.
SM	Margen de estabilidad.
UCP	Unidad Central de Proceso.
USART	Transmisor/Receptor Asíncrono Universal.
ZM	Punto de momento zero.
ZMP	Punto de momentos cero.

ÍNDICE GENERAL

CAPÍTULO I: GENERALIDADES

CAPÍTULO II: FUNDAMENTO TEÓRICO

CAPÍTULO III: LOS MICROMOTORES

CAPÍTULO IV: DISEÑO DEL MICROROBOT

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

La robótica se define como una ciencia aplicada que surge de la combinación de la tecnología de las máquinas-herramienta y de la informática, es decir al permitir que un programa informático controle las operaciones que antes realizaba un operario. Ligado a la robótica aparece el robot, si lo primero es la ciencia lo segundo es el objeto.

Desde que en 1917 el escritor Karel Capek usara el término robot para referirse a unas máquinas en forma de humanoide, han aparecido numerosos avances tecnológicos.

La robótica es uno de los temas más apasionantes de la tecnología actual, pero hay una rama derivada de ella que ha cobrado auge entre el aficionado y el estudiante hasta el profesional, es la mini-robótica.

Su aplicación no es exclusiva de la investigación, en el ámbito recreativo se ha extendido bastante así como su uso como herramienta educativa ha dado como consecuencia la proliferación de concursos estudiantiles de “mini-robótica” donde pequeños “engendros” haciendo gala de la electrónica de punta pueden desde seguir una línea, sortear obstáculos por medio de visión artificial, luchar entre ellos y bailar, etc.

Podemos mencionar cuatro tipos importantes de mini-robots: Terrestres, acuáticos, aéreos y espaciales, de los cuales por ahora nos limitaremos a los mini-robots terrestres.

Hoy en día la ciencia ha puesto toda la atención en desarrollar robots capaces de emular los movimientos de seres vivos.

Los robots bípedos han captado la mayor atención de la comunidad científica y tecnológica por los desafíos que conlleva de aquí es fundamental analizar la estabilidad del microrobot. Se ha investigado y desarrollado el presente trabajo que los acercará a este tema de una manera práctica y sencilla guiándolos en el entendimiento la construcción y el control de un microrobot bípedo bailarín, el microrobot tiene la capacidad de emular los movimientos de un bailarín y moverse con autonomía al son de un ritmo musical.

CAPÍTULO I

GENERALIDADES

1.1. ANTECEDENTES

La idea de la microrobótica surge aproximadamente sobre los 90 en una universidad de Europa. En esta universidad se encontraban unos investigadores y decidieron programar a unos pequeños robots con una tarea sencilla, esta consistía en recoger la mayor parte de trozos de vela que se encontraban en una habitación, cada micro robot recogía estos trozos hasta el tope de su capacidad y luego los apilaba todos en un lugar y seguía recogiendo. Luego de poner en marcha la prueba sucedió que todos los robots colaboraban entre sí y dejaban todos los trozos apilados en cierto lugar. Esto no fue coincidencia ya que se realizaban varias veces el mismo proceso y los micro robots siempre dejaban apilados todos los trozos de vela en un solo lugar de la habitación. Por lo tanto cada uno de los microrobots está programado para una tarea pequeña, y con el trabajo en grupo lograban cumplir con una tarea global.

El microrobot ha sido posible gracias a la aparición del microcontrolador en los años 90 del Siglo XX, que gobierna al microrobot y que se incrusta en el mismo.

Al ser un ordenador limitado, los microrobots están dedicados a resolver tareas que no exijan una elevada potencia y complicados algoritmos, con rapidez y precisión.

En la actualidad la construcción de los robots bailarines se lo lleva a cabo con motores normales por ende su tamaño es sumamente grande o en el mejor de los casos de tamaño mediano.

En el mercado existe algunas versiones de robots bailarines como por ejemplo podemos citar Rolly es un “objeto de diversión musical”, Qrio, “Miuro”, un Robot Bailarín USB entre otros. La mayoría de los robots son creados en Japón por la empresa Sony.

Existe un robot con un sistema que captura los movimientos de danzas populares en vídeo y programa los movimientos en un robot humanoide, principalmente con la idea de que esta parte de la cultura no se pierda.

Las principales características de un robot son:

- Movilidad
- Gobernabilidad
- Autonomía
- Polivalencia
- Repetibilidad

1.2. JUSTIFICACIÓN

A principios de los 90 y motivados por la necesidad de disminuir el peso, volumen, costo y consumo de energía de los productos electrónicos, la creación de componentes electrónicos más pequeños y los microchips se convirtieron en un gran

avance tecnológico. En este momento empezaron a ser útiles los microrobots ya que estos podían realizar tareas específicas y con una precisión del orden de la micra.

Debido a el tamaño y peso de los robot bípedos bailarines existentes en el mercado, el presente trabajo pretende disminuir de manera considerable el tamaño de los robots bailarines mediante la utilización de los micro motores y microchips permitiendo que el robot pueda coordinar mejor sus movimientos artísticos y creativos de baile al compás de la música.

La principal aplicación del microrobot bailarín será la participación en los campeonatos de robótica ya sea a nivel local como también a nivel nacional en representación de la Escuela de Ingeniería Electrónica de la Espoch.

Además se podría utilizarlo como medio de publicidad en algunos locales comerciales y como un medio de entretenimiento para todo tipo de personas.

1.3. OBJETIVOS

1.3.1 Objetivos Generales

- Diseñar y construir un microrobot bailarín bípedo con la utilización de micromotores.

1.3.2 Objetivos Específicos

- Estudiar y seleccionar los micromotores aptos para aplicar en el microrobot bailarín.
- Aprovechar las cualidades que nos ofertan los micromotores como su tamaño, peso y facilidad de operabilidad.

- Diseñar el microrobot bailarín con siete grados de libertad.
- Implementar la interfaz electrónica para que el modo de funcionamiento del robot sea autónomo.
- Programar el microrobot bailarín para que pueda bailar tres tipos de ritmos.
- Implementar el microrobot bailarín.
- Utilizar herramientas de software para la simulación de las diferentes etapas de este proyecto.

CAPÍTULO II

FUNDAMENTO TEÓRICO.

2.1. ROBÓTICA

La robótica es una ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia. Básicamente, la robótica se ocupa de todo lo concerniente a los robots, lo cual incluye el control de motores, mecanismos automáticos neumáticos, sensores, sistemas de cómputos, etc. Las ciencias y tecnologías de las que deriva podrían ser: el álgebra, los autómatas programables, las máquinas de estados, la mecánica o la informática

2.1.1. Generaciones de la robótica

La introducción de los microprocesadores desde los años 70 ha hecho posible que la tecnología de los robots haya sufrido grandes avances, los modernos ordenadores han ofrecido un "cerebro" a los músculos de los robots mecánicos. Ha sido esta fusión de electrónica y mecánica la que ha hecho posible al moderno robot, los japoneses han acuñado el término "mecatrónica" para describir esta fusión.

El año 1980 fue llamado "primer año de la era robótica" porque la producción de robots industriales aumentó ese año un 80 % respecto del año anterior.

Primera y Segunda Generación

Los cambios en Robótica se suceden tan deprisa que ya se ha pasado de unos robots relativamente primitivos a principios de los 70, a una segunda generación. La primera generación de robots era reprogramable, de tipo brazo, dispositivos manipuladores que sólo podían memorizar movimientos repetitivos, asistidos por sensores internos que les ayudan a realizar sus movimientos con precisión. La segunda generación de robots entra en escena a finales de los 70, tienen sensores externos (tacto y visión por lo general) que dan al robot información (realimentación) del mundo exterior. Estos robots pueden hacer elecciones limitadas o tomar decisiones y reaccionar ante el entorno de trabajo, se les conoce como robots adaptativos.

Tercera Generación

La tercera generación está surgiendo en estos años, emplean la inteligencia artificial y hacen uso de los ordenadores tan avanzados de los que se puede disponer en la actualidad. Estos ordenadores no sólo trabajan con números, sino que también trabajan con los propios programas, hacen razonamientos lógicos y aprenden. La IA permite a los ordenadores resolver problema inteligentemente e interpretar información compleja procedente de avanzados sensores.

Gracias al desarrollo de esta generación la vida de las personas se ha facilitado de forma inmediata ya que gracias al ordenador se ha mejorado procesos que parecen muy repetitivos y al humano se le ha ubicado en mejores sitios de trabajo.

2.1.2. Partes de un robot

La Estructura O Chasis

Es la encargada de darle forma al robot y sostener sus componentes. Puede estar constituida por numerosos materiales, como plástico, metales, etc. Y tener muchas formas diferentes.

Las fuentes de movimientos

Son las que otorgan movimientos al robot, una de las más utilizadas es el motor eléctrico. En robótica se utilizan motores de cc (corriente continua), servomotores y motores paso a paso. Cuando las fuentes de movimiento no manejan directamente los medios de locomoción del robot, se precisa una interfaz o medio de transmisión de movimiento entre estos dos sistemas, que se utiliza para aumentar la fuerza o para cambiar la naturaleza del movimiento, por ejemplo para convertir un movimiento circular en lineal, o para reducir la velocidad de giro.

Los medios de locomoción:

Son sistemas que permiten al robot desplazarse de un sitio a otro si éste debe hacerlo. El más utilizado y simple es el de las ruedas y le siguen en importancia las piernas y las orugas. Algunos robots deben sostener o manipular algunos objetos y para ello emplean dispositivos denominados de manera general medios de agarre.

La fuente de alimentación:

De los robots depende de la aplicación que se les de a los mismos, así si el robot se tiene que desplazar automáticamente, se alimentará seguramente con baterías eléctricas recargables, mientras que si no requiere desplazarse o solo lo debe hacer

mínimamente, se puede alimentar mediante corriente alterna a través de un convertidor.

Los sensores

Le permiten al robot a manejarse con cierta inteligencia al interactuar con el medio. Son componentes que detectan o perciben ciertos fenómenos o situaciones. Estos sensores pretenden en cierta forma imitar los sentidos que tienen los seres vivos.

Los circuitos de control:

Son el “cerebro” del robot en la actualidad están formados por componentes electrónicos más o menos complejos dependiendo de las funciones del robot de lo que tenga que manejar.

El sistema de control y el lenguaje de programación:

Forman el sistema de forma automática de decisiones, que incluye la planificación, el control de los movimientos y la interpretación de los datos que aportan los sensores.

Tipos de robot

Desde un punto de vista muy general los robots pueden ser de los siguientes tipos:

- Androides
- Móviles
- Industriales
- Médicos
- Los robots didácticos o experimentales
- Teleoperadores

- Poliarticulados

2.2. MICROCONTROLADOR

En la actualidad, el mercado de los microcontroladores de 8 bits se ha desarrollado hasta el punto de estar presentes en casi todo elemento de control electrónico, y han sido la base fundamental para el desarrollo de nuevas tecnologías que están presentes en los aparatos eléctricos de uso cotidiano.

Sin embargo, los microcontroladores más utilizados a nivel mundial son los de tecnología RISC de 8 bits, debido a su bajo costo y su facilidad de programación.
Un microcontrolador es un circuito integrado que contiene todos los componentes de un computador. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta última característica es la que le confiere la denominación de «controlador incrustado» (embedded controller).

El Microcontrolador es un computador dedicado. En su memoria sólo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida soportan la conexión de sensores y actuadores del dispositivo a controlar. Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada.

"Un microcontrolador es un computador completo, aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado y se designa a gobernar una sola tarea" [UNI1998].

La industria Informática acapara gran parte de los microcontroladores que se fabrican. Casi todos los periféricos del computador, desde el ratón o el teclado hasta la impresora, son regulados por el programa de un microcontrolador (véase Figura II.1).

Los electrodomésticos de línea blanca (lavadoras, hornos, lavavajillas, etc.) y de línea marrón (televisores, vídeos, aparatos musicales, etc.) incorporan numerosos microcontroladores. Igualmente, los sistemas de supervisión, vigilancia y alarma en los edificios utilizan estos chips. También se emplean para optimizar el rendimiento de ascensores, calefacción, aire acondicionado, alarmas de incendio, robo, etc.

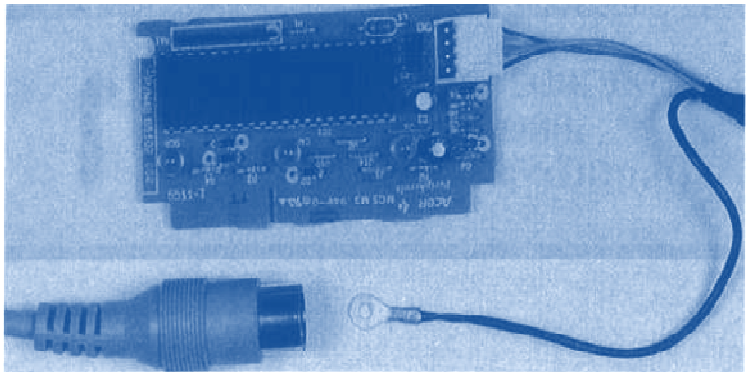


Figura II.1. Periféricos de una computadora

Las comunicaciones y sus sistemas de transferencia de información utilizan profundamente estos pequeños computadores incorporándolos en los grandes automatismos y en los modernos teléfonos.

La instrumentación y la electromedicina son dos campos idóneos para la implantación de estos circuitos integrados. Una importante industria consumidora de microcontroladores es la de automoción, que los aplica en el control de aspectos tan populares como la climatización, la seguridad y los frenos ABS.

2.2.1. Diferencia entre microprocesador y microcontrolador.

El microprocesador es un circuito integrado que contiene la Unidad Central de Proceso (UCP), también llamada procesador, de un computador. La UCP está formada por la Unidad de Control, que interpreta las instrucciones, y el Camino de Datos, que las ejecuta.

Las patitas de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de E/S y configurar un computador implementado por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que se destine (ver Figura II.2.).

Un microprocesador es un sistema abierto con el que puede construirse un computador con las características que se desee, acoplándole los módulos necesarios.

Un microcontrolador es un sistema cerrado que contiene un computador completo y de prestaciones limitadas que no se pueden modificar. La disponibilidad de los buses en el exterior permite que se configure a la medida de la aplicación.
Todas las partes del computador están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos.

Si sólo se dispusiese de un modelo de microcontrolador, éste debería tener muy potenciados todos sus recursos para poderse adaptar a las exigencias de las diferentes aplicaciones. Esta potenciación supondría en muchos casos un despilfarro.

En la práctica cada fabricante de microcontroladores oferta un elevado número de modelos diferentes, desde los más sencillos hasta los más poderosos.

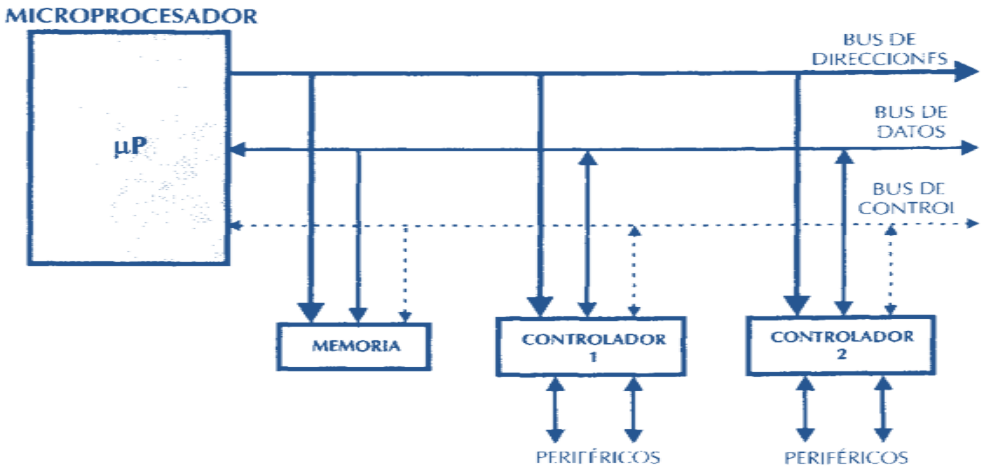


Figura II.2. Estructura de un sistema abierto basado en un microprocesador

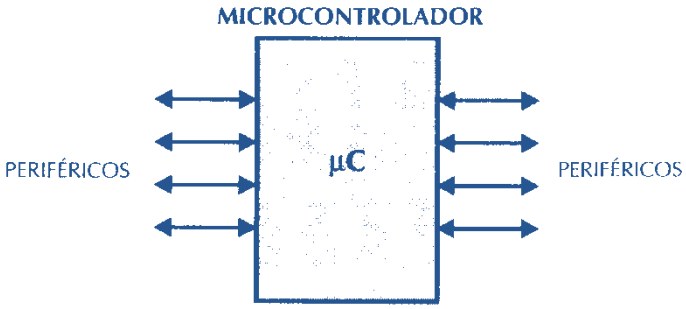


Figura II.3. El Microcontrolador en un sistema cerrado

Es posible seleccionar la capacidad de las memorias, el número de líneas de E/S, la cantidad y potencia de los elementos auxiliares, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy destacado del diseño es la selección del microcontrolador a utilizar.

2.2.2. Arquitectura de los microcontroladores

- **Arquitectura Von Neuman**

  Los microcontroladores fueron concebidos bajo una arquitectura clásica, la cual se caracteriza por poseer una sola memoria principal en la que se almacenan indistintamente los datos e instrucciones del programa.

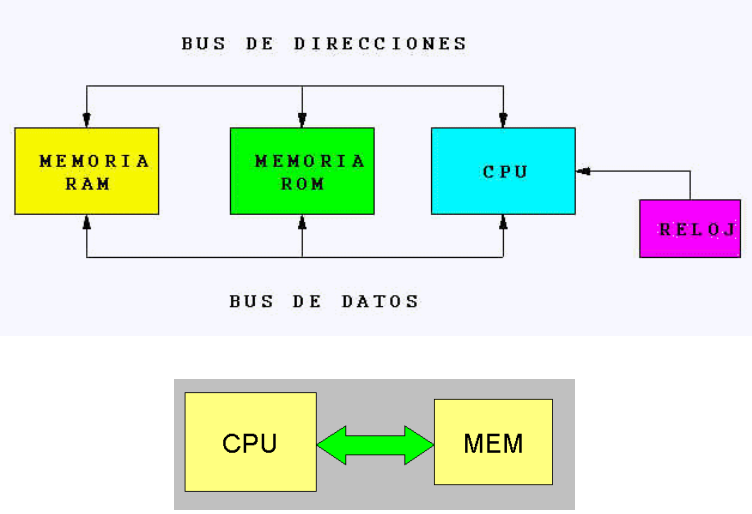
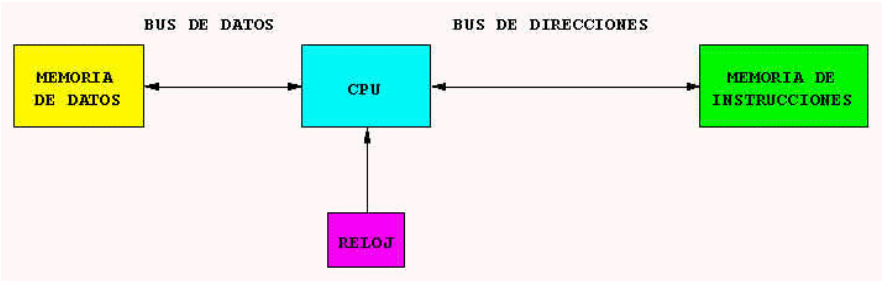


Figura II.4. Arquitectura Modelo Van Neumann

- **Arquitectura Hardvard**

Actualmente los microcontroladores son diseñados con una arquitectura HARVARD, la cual se fundamenta en el uso de dos memorias independientes, la primera que se conoce como FLASH y es en la que se almacenan las instrucciones de programa; y la otra que se denomina RAM, en la cual se almacenan los datos.

Cada una de estas memorias posee su estructura de sistema de buses de acceso, lo que posibilita realizar operaciones de manera simultánea en cada memoria.



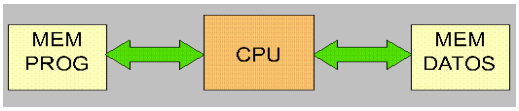


Figura II.5. Arquitectura Modelo Harvard

- **Arquitectura Risc**

RISC, abreviatura de Reduced Instruction Set Computer es una tecnología de diseño de microprocesadores / microcontroladores en la que prima un juego de instrucciones muy pequeño y de tamaño fijo, con el que se fomenta la modularidad, segmentación y portabilidad del código.

2.2.3. Arquitectura interna de un microcontrolador

Un microcontrolador posee todos los componentes de un computador, pero con unas características fijas que no pueden alterarse.

Las partes principales de un microcontrolador son:

1. Procesador
2. Memoria no volátil para contener el programa
3. Memoria de lectura y escritura para guardar los datos
4. Líneas de EIS para los controladores de periféricos:
    a) Comunicación paralelo
    b) Comunicación serie
    c) Diversas puertas de comunicación (bus l2ºC, USB, etc.)
5. Recursos auxiliares:
    a) Circuito de reloj
    b) Temporizadores
    c) Perro Guardián («watchdog»)

d) Conversores A/D y D/A

e) Comparadores analógicos

f) Protección ante fallos de la alimentación

g) Estado de reposo o de bajo consumo.

2.2.4. Métricas para la elección del microcontrolador.

Entre las características que ofrecen los microcontroladores se debe tomar en cuenta para su elección:

- **Sencillez de programación, manejo y grabación:** El encapsulado se recomienda que sea de tipo DIP para facilitar el modo de grabación y realizar las pruebas en un Bread Board.
- **Alta velocidad**
- **Capacidad de Memoria:** El microprocesador debe tener la memoria de programa (Memoria Flash) en un rango entre 16 Kb y 32 Kb.
- **Bajo precio:** Esta métrica tiene gran importancia para que permita introducir al mercado el producto con mayor facilidad.
- **Disponibilidad:** Se refiere a la facilidad de encontrarlo en nuestro país.
- **Tamaño**: El factor más importante por el tamaño del microrobot.

2.2.4.1. Estudio de la mejor alternativa de microcontrolador.

Aplicamos el Método Ponderado para elegir el microcontrolador. Para lo cual asignamos pesos o valores para la ponderación de 1 a 10 (Tabla II.1).

Tabla II.I. Asignación cuantitativa para las métricas

Métricas o factores	**Valores para la ponderación**	**Ponderación**
Programación, manejo y grabación	10	0.23
Velocidad	8	0.19
Memoria	7	0.16
tamaño	10	0.23
Bajo precio y disponibilidad	8	0.19
Total	43	1.00

Asignamos calificaciones o costos según la información presentada de los microcontroladores.

Tabla II.II. Calificaciones y costos

Factor / **Alternativa**	**Sencillez, manejo y programación**	**Velocidad**	**Memoria**	**Tamaño**	**Bajo precio y disponibilidad**
16f628A	8	7	7	10	8
16F87X	8	8	8	7	6

Con la información de la tabla II.2 obtenemos la matriz de puntos para determinar el microcontrolador es el adecuado. De la tabla (Tabla II.3) determinamos que el microcontrolador adecuado para el proyecto es el PIC16F628A.

Tabla II.III. Matriz de puntos

Factores	Ponderación	16f628A		16F87X	
		Calif	Calif. P	Calif.	Calif. P
Sencillez, manejo y programación	0.23	8	1.84	8	1.84
Velocidad	0.19	7	1.33	8	1.52
Memoria	0.16	7	1.12	8	1.28
Tamaño	0.23	10	2.3	7	1.61
Bajo precio y disponibilidad	0.19	8	1.52	6	1.14
total			8.11		**7.39**

2.2.5. Microcontrolador 16F628A

Figura II.6 Encapsulado del PIC 16f628

Principales características

- Set de instrucciones reducido (RISC). Sólo 35 instrucciones.

- Las instrucciones se ejecutan en un sólo ciclo de máquina excepto los saltos que requieren 2 ciclos.
- Opera con una frecuencia de reloj de hasta 20 MHz (ciclo de máquina de 200 ns)
- Memoria de programa: 2048 posiciones de 14 bits.
- Memoria de datos: Memoria RAM de 224 bytes (8 bits por registro).
- Memoria EEPROM: 128 bytes (8 bits por registro).
- Stack de 8 niveles.
- 16 Terminales de I/O que soportan corrientes de hasta 25 mA.
- 3 Timers.
- Módulos de comunicación serie, comparadores, PWM.

Características especiales:

- La memoria de programa se puede reescribir hasta 1000 veces.
- La memoria EEPROM se puede reescribir hasta 1000000 de veces.
- Los datos almacenados en la memoria EEPROM se retienen por 40 años y no se borran al quitar la alimentación al circuito.
- 10 fuentes de interrupción, algunas de ellas son:
    - Señal externa (RB0).
    - Desborde de TMR0 y TMR1.
- Cambio en el estado de los terminales RB4, RB5, RB6 o RB7.
- Ciclo de escritura en la memoria EEPROM completado

Descripción de los terminales

Terminales de entrada-salida (16 en total)

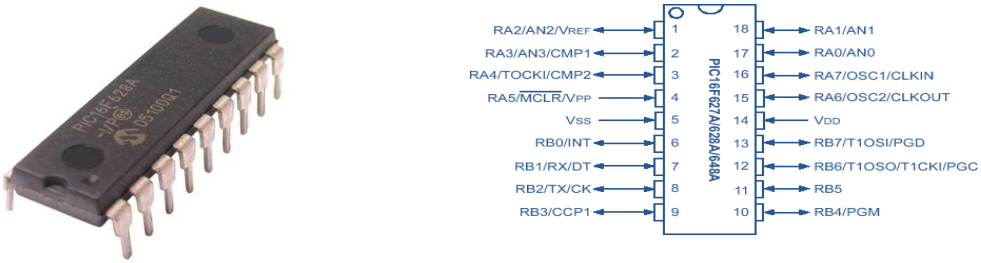


Figura II.7. Diagrama de pines 16f628

PORTA: RA0-RA7:

- Los terminales RA0-RA3 y RA6 – RA7 son bidireccionales y manejan señales TTL.
- El terminal RA5 es una entrada Schmitt Trigger que sirve también para entrar en el modo de programación cuando se aplica una tensión igual a Vpp (13,4V mínimo).
- El terminal RA4 como entrada es Schmitt Trigger y cómo salida es colector abierto. Este terminal puede configurarse como reloj de entrada para el contador TMR0.

PORTB: RB0-RB7:

- Los terminales RB0-RB7 son bidireccionales y manejan señales TTL.
- Por software se pueden activar las resistencias de pull-up internas, que evitan el uso de resistencias externas en caso de que los terminales se utilicen como entrada (permite, en algunos casos, reducir el número de componentes externos).
- RB0 se puede utilizar como entrada de pulsos para provocar una interrupción proveniente del exterior.

- RB4-RB7 están diseñados para detectar la interrupción por cambio de estado, por ejemplo, cuando se pulsa una tecla de un teclado matricial.

Otros terminales.

- VDD: Positivo de alimentación. 2 a 5,5 Vcc.
- VSS: Negativo de alimentación.
- MCLR: Master Clear (Reset). Si el nivel lógico de este terminal es bajo (0 Vcc), el microcontrolador permanece inactivo.
- OSC1/CLKIN: Entrada del oscilador (cristal). Entrada de oscilador externo.
- OSC2/CLKOUT: Salida del oscilador (cristal).

Aspecto Interno del PIC 16F628.

Nuestro PIC pertenece a la familia de la gama media con una compactación de código superior a la de sus competidores, incorpora tres características de avanzada que son:

- Procesador tipo RISC (Computadores de Juego de Instrucciones Reducido).
- Procesador segmentado.
- Arquitectura HARVARD.

Con estos recursos el PIC es capaz de ejecutar instrucciones solamente en un ciclo de instrucción, salvo las de salto, que tardan el doble.
El juego de instrucciones se reduce a 35 y sus modos de direccionado se ha simplificado al máximo.

Con la estructura segmentada se pueden realizar simultáneamente las dos fases en que se descompone cada instrucción. Al mismo tiempo que se está

desarrollando la fase de ejecución de una instrucción se realiza la fase de búsqueda de la siguiente.

La separación de los dos tipos de memoria son los pilares de la arquitectura Harvard, gracias a esto se puede acceder en forma simultánea e independiente a la memoria de datos y a la de instrucciones.

Al tener memorias separadas permite que cada una de ellas tenga el ancho y tamaño más adecuado. De esta manera en el PIC 16F628A el ancho de los datos es de un byte, mientras que la de las instrucciones es de 14 bits.

Otra característica de los PICs 16F628A es el manejo de los bancos de registros, los cuales participan de forma muy activa cuando se ejecutan las instrucciones.
En línea general, los registros se clasifican como de uso general (GPR – General Purpose Registers) y los especiales (SFR – Special Function Registers).

Los registros de uso general pueden ser usados a voluntad por el usuario, sin existir restricciones. Pueden servir para almacenar resultados que se reciben desde el registro W (works), datos que provienen de las puertas de entradas, etc.

Los registros de uso específicos no pueden ser usados como los anteriores. La verdad, que son ellos los que controlan prácticamente todo el funcionamiento de los microcontroladores, pues toda la configuración necesaria para el perfecto funcionamiento del microcontrolador es hecho a través de algún tipo de SFR.

El Procesador.

El procesador responde a la arquitectura RISC, que se identifica porque el juego

de instrucciones se reduce a 35, donde la mayoría se ejecutan en un ciclo de reloj, excepto las instrucciones de salto que necesitan dos ciclos.

La ALU (Unidad Aritmético Lógica), ubicada dentro del procesador efectúa las operaciones lógicas - aritméticas con dos operandos, uno que recibe desde el registro W (registro de trabajo) y otro que puede provenir de cualquier registro o del propio código de la instrucción. El resultado de la operación puede almacenarse en cualquier registro o en el propio W.

La memoria de Datos.

Formada por dos bloques bien diferenciados:

- El primero, es una memoria del tipo RAM (SRAM), dividida en cuatro bancos, los cuales contienen los registros de uso general (GPR), y los registros de uso específicos (SFR).

  Los registros especiales SFR ocupan los primeros 32 lugares de cada banco.

  Los registros generales GPR son 224 de 8 bits cada uno, y se pueden acceder de forma directa o indirecta a través del registro FSR.
- El segundo, es una memoria EEPROM, que puede almacenar datos que no se deseen perder una vez que se quita la alimentación, su acceso está controlado por unos registros especiales.

Esta memoria es del tipo SRAM, formado por registros de 8 bits y organizada en dos áreas:

- Registros de funciones especiales, controlan el funcionamiento del dispositivo.
- Registros de propósito general, sirven para guardar datos temporalmente mientras se ejecuta el programa.

La memoria a su vez se divide en cuatro bancos (banco 0 al banco 3), para cambiar de banco se utilizan los bits 5 y 6 (RP0 y RP1 respectivamente) del registro STATUS.

Direccionamiento de la memoria de datos.

Direccionamiento Directo:

Es el modo más empleado. El operando que utiliza la instrucción en curso se referencia mediante su dirección, que viene incluida en el código OP de la misma, concretamente en los 7 bits de menos peso.

Direccionamiento Indirecto:

En este modo se emplea los registros INDF (posición 00h de la memoria de datos) y el registro FSR (File Select Register ubicado en la posición 04h de le memoria de datos).

La forma que funciona el direccionamiento indirecto es que la dirección de memoria del registro al que se accede se introduce en el registro FSR. Los 7 bits de menos peso apuntan la posición y de más peso, junto con el bit IRP del registro STATUS, seleccionan el banco. Este modo permite acceder a 256 direcciones.
Cuando se quiere acceder a un registro cuya dirección esta almacenada en el registro FSR, se usará el registro INDF. Este registro no está implementado físicamente, por lo que no se accede directamente a él.

Por ejemplo; la instrucción: movwf INDF, esta instrucción traslada el contenido del registro W al registro apuntado por FSR.

Memoria EEPROM

El PIC16F628A tiene 128 registros de memoria de uso general del tipo EEPROM. El contenido de estos registros no se pierde cuando se quita la alimentación al circuito. La información almacenada se mantiene inalterable durante años.

Registros especiales utilizados para leer y escribir en la memoria EEPROM

La memoria EEPROM se puede leer o escribir durante la operación normal. Esta memoria no está mapeada directamente en la memoria de datos. Para acceder a alguno de los 128 registros EEPROM de 8 bits, debe utilizarse un procedimiento de acceso indirecto que involucra a 4 registros especiales:

· **EEADR:** Dirección del registro EEPROM (00H-7FH)

· **EEDATA:** Contenido del registro EEPROM.

· **EECON1:** Contienen una serie de bits de control.

· **EECON2:** Se utiliza exclusivamente en la rutina de escritura.

Mapa de la Memoria de Datos.

La memoria de Programa.

Es una memoria no volátil de tipo Flash, lo que permite que se borre y escriba eléctricamente. En el PIC 16F628A se pueden almacenar hasta 2048 instrucciones de 14 bits cada una.

El Reloj

Para que el PIC pueda procesar las instrucciones, es necesario un reloj cuya frecuencia es parámetro fundamental en el momento de establecer la velocidad de ejecución de las instrucciones y en el consumo de energía.

El tiempo en que tarda en ejecutarse una instrucción se llama ciclo de la instrucción.

En los PICs , un ciclo de instrucción emplea 4 períodos de reloj. Todas las instrucciones del PIC se realizan en un ciclo de instrucción, salvo las de salto, que tardan el doble. El PIC 16F628A posee 8 formas de configurar al oscilador. Un circuito RC como oscilador brinda una solución económica. El tipo LP empleado en aplicaciones de bajo consumo. El modo XT solo usa un cristal, el HS emplea cristales de alta velocidad. Por último el oscilador interno INTOSC configurable a alta o baja velocidad.

Los microcontroladores PIC(Peripherical Interface Controller), son fabricados por la empresa MICROCHIP Technology INC. Cuya central se encuentra en Chandler Arizona, esta empresa ocupa el primer lugar en venta de microcontroladores de 8 bits desde el año 2002. Su gran éxito se debe a la gran variedad (más de 180 modelos), gran versatilidad, gran velocidad, bajo costo, bajo consumo de potencia, y gran disponibilidad de herramientas para su programación.

2.3. CONTROL DE EQUILIBRIO.

Antes de explicar el bloque de control de equilibrio, es importante familiarizar al lector con algunos términos utilizados comúnmente:

- **Planos anatómicos.-** Son los planos que dividen a un cuerpo, en este caso la anatomía humana, tal como muestra la figura II.8.

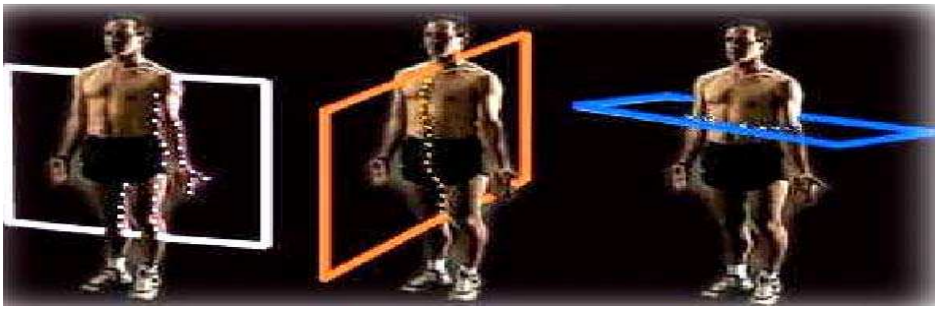


Figura II.8. De izquierda a derecha. Planos: frontal, sagital y transversal

- **COM (Center of mass – Centro de masa).-** Es un punto dentro del microrobot, en el cual éste puede ser considerado como un punto de masa concentrada.
- **GCOM (Ground projection of center of mass – Punto de proyección del centro de masa)** .- Es el punto de proyección del centro de masa sobre la superficie.
- **Polígono de soporte.**- Es la figura geométrica plana formada en el suelo por uno o dos pies, dependiendo de la posición del microrobot. Cuando el microrobot se encuentra en soporte único, el polígono de soporte es la figura determinada por el pie de apoyo, en cambio, cuando el microrobot se encuentra en doble soporte el polígono de soporte es la figura formada por los dos pies, tal como muestra la figura II.9.

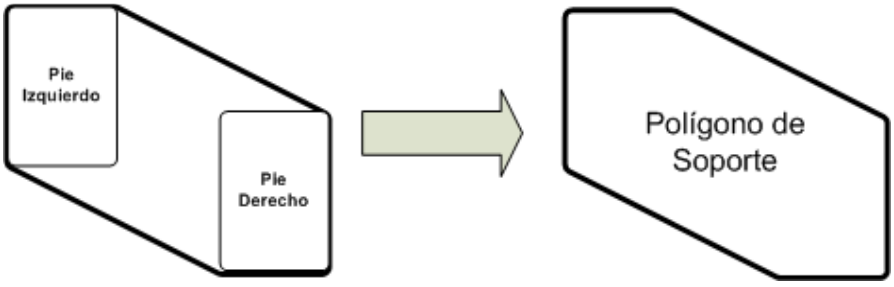


Figura II.9. Vista superior del polígono en doble soporte

- **COP (Center of pressure – Centro de presión o fuerza).-** Es un punto en la superficie de contacto (limitada por el polígono de soporte) en donde se puede considerar que actúa la fuerza neta de reacción del piso.
- **ZMP (Zero Moment Point – Punto de momentos cero).-** Es el punto dentro de la superficie de contacto, en el cual la reacción neta del piso actúa cuando el microrobot está dinámicamente estable.
- **FRI (Foot Rotation Indicator – Indicador de rotación del pie).-** Es el punto en la superficie (dentro o fuera del polígono de soporte), en el cual "la fuerza de reacción del piso tendría que actuar para que el pie de apoyo permanezca estacionario". Este punto ha sido determinado con el único objetivo de simplificar el análisis de la estabilidad del microrobot, ya que si este punto está fuera del polígono de soporte el microrobot tiende a caer, es decir, que si este punto está siempre dentro del polígono de soporte, se garantiza la estabilidad dinámica del microrobot. La posición del FRI es una consecuencia directa del estado dinámico del microrobot. Algunos autores conocen a este punto como ZM (Zero moment point – Punto de momentos cero).
- **SM (Stability Margin - Margen de estabilidad).-** Es la mínima distancia entre el FRI y los bordes del polígono de soporte (ver Figura II.10), en donde, SMx es el margen de estabilidad en el eje x, y SMz es el margen de estabilidad en el eje z.

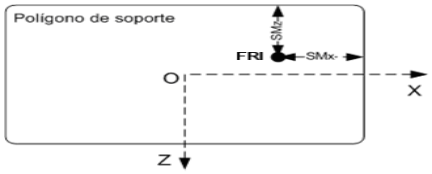


Figura II.10. Margen de estabilidad

Método de control de Equilibrio

La idea fundamental del control de equilibrio (aunque parezca obvia) es evitar que el microrobot caiga, para esto, se dispone de algunos indicadores tales como el ZMPI y el FRI. A continuación se realiza un análisis de estos puntos.

En la figura II.11 se esquematizan todas las fuerzas y momentos que actúan sobre el pie de apoyo:

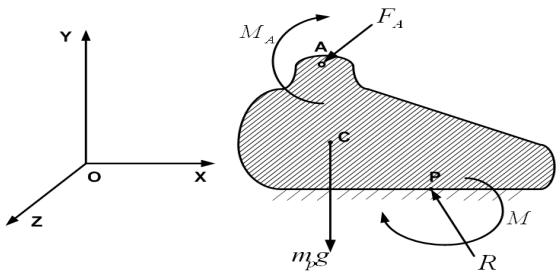


Figura II.11. Fuerzas y momentos sobre el pie de apoyo**.**

en donde, $\mathbf{M_A}$ **y** $\mathbf{F_A}$ son el momento y la fuerza resultante generado por el cuerpo en movimiento, **M** y **R** son el momento y fuerza de reacción del piso. $\mathbf{m_p}$ es la masa del pie.

Para que el pie se encuentre en equilibrio estático:

$\sum \bar{F} = 0$

$\bar{\boldsymbol{R}} + \bar{\boldsymbol{F}}_{\boldsymbol{A}} = 0$

y

$\sum \bar{\boldsymbol{M}}_{\boldsymbol{o}} = 0$

$\vec{O}Px\bar{\boldsymbol{R}} + \vec{O}Ax\bar{\boldsymbol{F}}_{\boldsymbol{A}} + \bar{\boldsymbol{M}} + \bar{\boldsymbol{M}}_{\boldsymbol{A}} + \vec{O}Cx\left(m_p\vec{g}\right) = 0$

A partir de ahora se asumirá que la fricción de la superficie es lo suficientemente grande, para no permitir que exista deslizamiento entre el pie y el piso, entonces, R_x, R_z y M_y del piso compensan las fuerzas y momentos del cuerpo en esas direcciones, por lo que:

$$\sum \bar{F} y = 0$$

$$\bar{\boldsymbol{R}} y + \bar{\boldsymbol{F}}_{\boldsymbol{Ay}} = 0$$

$$(\sum \bar{\boldsymbol{M}}_{\boldsymbol{o}})^{H} = 0$$

$$\left(\vec{O}P x \bar{\boldsymbol{R}}\right)^{\boldsymbol{H}} + \left(\vec{O}A x \bar{\boldsymbol{F}}_{\boldsymbol{A}}\right)^{\boldsymbol{H}} + (\bar{\boldsymbol{M}})^{\boldsymbol{H}} + (\bar{\boldsymbol{M}}_{\boldsymbol{A}})^{\boldsymbol{H}} + \left(\vec{O}C x (m_p \vec{g})\right)^{H} = 0$$

es decir, se ha podido reemplazar a la fuerza de reacción neta del piso por una que actúa solo en el eje y. También los momentos se han reducido al plano horizontal (ejes x, z). Una condición fundamental para que el sistema esté en equilibrio es que en el punto P,

$M_x = M_z = 0$

es decir, los momentos generados por la fuerza de reacción son cero, siendo posible reemplazar a la reacción del piso por una única fuerza en el eje y.

Cambiando el punto de referencia de O hacia A y despreciando la masa del pie:

$$(\sum \bar{\boldsymbol{M}}_{\boldsymbol{A}})^{H} = 0$$

$$\left(\vec{A}P x \bar{\boldsymbol{R}}\right)^{\boldsymbol{H}} + (\bar{\boldsymbol{M}}_{\boldsymbol{A}})^{\boldsymbol{H}} = 0$$

$$\left(\vec{A}P x \bar{\boldsymbol{R}}\right)^{\boldsymbol{H}} = -(\bar{\boldsymbol{M}}_{\boldsymbol{A}})^{\boldsymbol{H}}$$

Claramente se puede notar que la igualdad anterior se cumple eligiendo el punto P adecuado, de tal forma que el momento generado por la fuerza de reacción del piso compense al momento total generado por el cuerpo en su caminata; este punto es conocido como ZMP - Es discutible el nombre de ZMP ya que únicamente dos componentes del momento de reacción del piso son iguales a cero (en los ejes x , z) - . El ZMP existe únicamente dentro del polígono de soporte. Cuando existe el ZMP el microrobot se encuentra dinámicamente estable, siendo este punto coincidente con el COP.

Dado que el ZMP existe únicamente dentro del polígono de soporte, su cálculo nos da una idea de cuan estable se encuentra el microrobot en un determinado instante, sin embargo, existe un concepto más general y es el FRI.

“El FRI es el punto en la superficie, donde la fuerza de reacción neta del piso tendría que actuar para mantener el pie estacionario”

El FRI puede existir ya sea dentro o fuera del polígono de soporte, o sea, que puede ser también un indicador de cuan inestable es el microrobot. Cuando el FRI está dentro del polígono de soporte, el microrobot está dinámicamente estable y este punto es coincidente con el ZMP y el COP.

Entonces, de aquí nace la idea fundamental del control de equilibrio durante el movimiento del microrobot.

Para que un microrobot bípedo se encuentre dinámicamente estable durante su caminata, el FRI debe estar siempre dentro del polígono de soporte.

Después de revisar estos conceptos, la pregunta podría ser: Cuál es la diferencia entre el ZMP y COP?. Según Goswami no existe diferencia entre el ZMP y el COP, sin embargo, Vukobratovic realiza las siguientes observaciones:

1.- El COP siempre existe (ya sea en el borde del polígono de soporte), sin embargo el ZMP no existe cuando el microrobot está dinámicamente inestable.

2.- El ZMP puede existir en otras partes del cuerpo del microrobot (por Ej. en el hombro) sin que el COP esté involucrado.

En la figura II.12, para que el punto P sea considerado como FRI, la fuerza R que incide en este punto debe compensar los momentos y fuerzas que actúan sobre el pie

de apoyo, impidiendo que éste rote y el microrobot pierda su estabilidad durante su movimiento.

Cuando el momento generado por el cuerpo (MA) es demasiado grande, puede darse que el punto P se encuentre fuera del polígono de soporte (Figura II.12), en este caso, la fuerza de reacción del piso se encuentra en el borde del pie (no puede seguir al punto P, entonces, no puede compensar el momento generado por el cuerpo en movimiento), y evidentemente el microrobot tiende a perder su estabilidad. Un indicador físico de la pérdida de estabilidad del microrobot es la rotación del mismo a partir de los bordes del polígono de soporte.

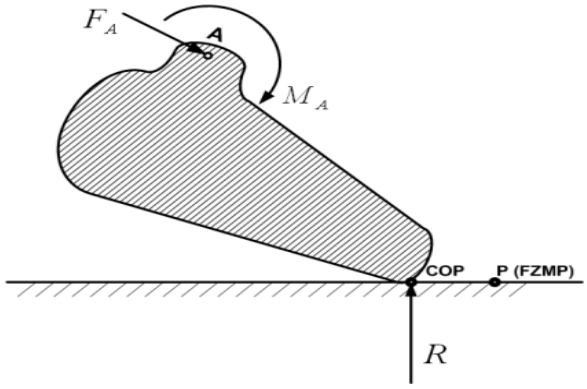


Figura II.12. Rotación del pie a partir de uno de sus bordes.

Una de las formas de calcular el FRI es a partir de la dinámica de cada uno de los eslabones que conforman el microrobot, de modo que, la ecuación para el equilibrio dinámico rotatorio, se obtiene mediante la suma de los momentos sobre el microrobot, calculado sobre cualquier punto de referencia estacionario, siendo esto igual a la suma de las razones de cambio del momento angular de los segmentos individuales entorno al mismo punto, llegándose a calcular con precisión las coordenadas del FRI, sin embargo, se requiere las coordenadas,

aceleraciones, momentos angulares, etc., del centro de masa de cada eslabón del microrobot.

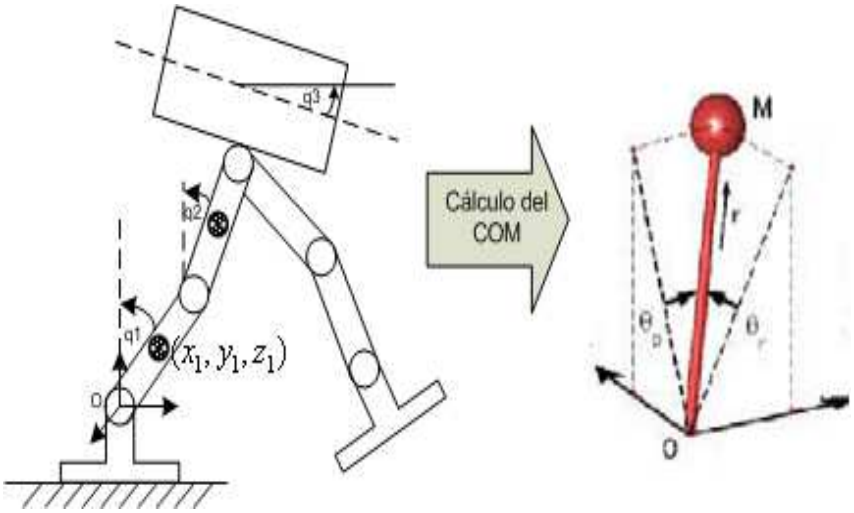


Figura II.13. Modelo simplificado del robot.

Entonces, la principal función del sistema de control de equilibrio es mantener al FRI dentro del polígono de soporte, para esto, debe realizar las correcciones necesarias en los movimientos del robot (principalmente en la trayectoria de la cadera).

2.4. MICROCODE STUDIO

El software que se utilizó para el desarrollo del proyecto fue MICROCODE STUDIO, una vez instalado el programa, se puede acceder a través del botón Inicio de Windows.(ver fig. II.14.)

En la figura II.17 podemos observar la pantalla principal del Microcode Studio en la parte izquierda podemos encontrar las diferentes partes de nuestro proyecto también podemos elegir el tipo de microcontrolador a utilizar.

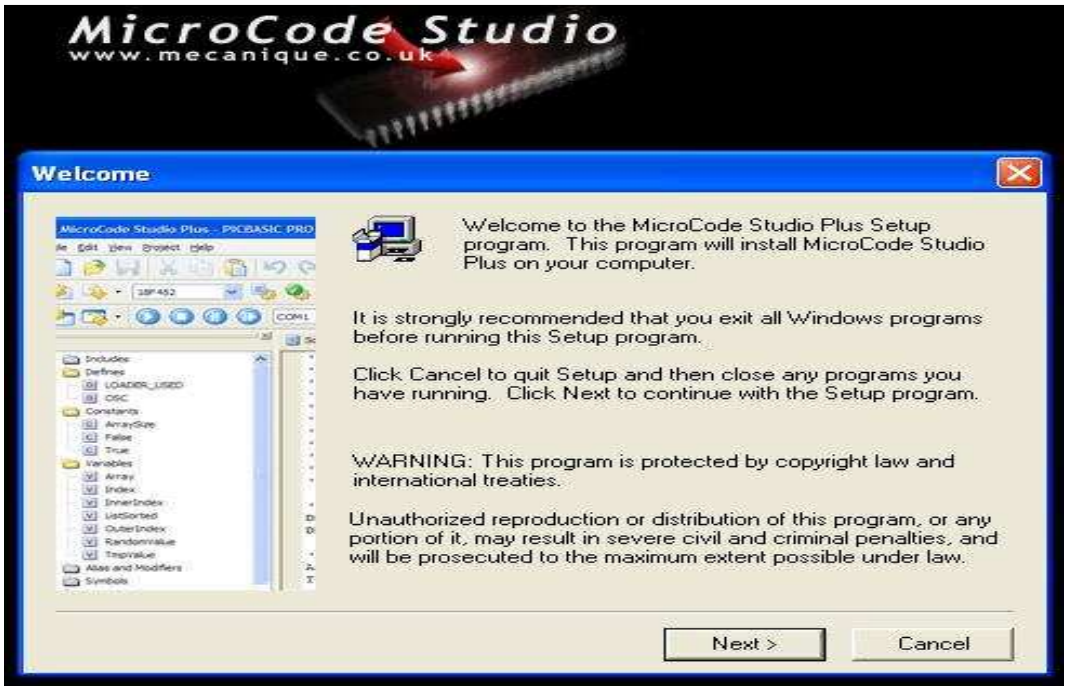


Figura II.14. Instalación de MicroCode Studio

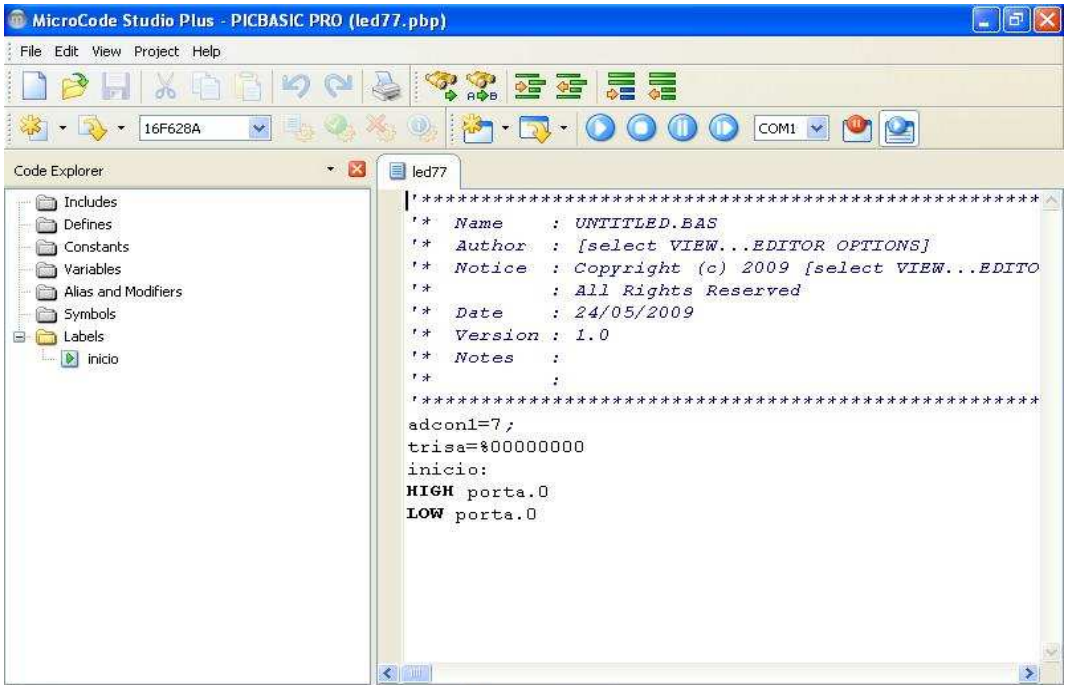


Figura II.15. Pantalla Principal de Microcode

2.5. WINPIC800

El GTP-USB [plus] es la solución para la grabación de firmware para microcontroladores y varios MCU.

Su diseño simple y compacto permite usarse con los ordenadores más modernos, portátiles etc. sin necesidad de ninguna fuente externa, sirviéndose del propio puerto USB.

Es el hardware recomendado por WinPic800, mejorándose, actualizándose y expandiéndose periódicamente le convierten en una herramienta muy potente sin los inconvenientes y problemas que contrae usar otros puertos del ordenador (ver figura II.18.)

Características:

- Soporta el control del programa **WinPic800**
- El firmware se auto actualiza con cada nueva versión.
- Muy rápido, gracias al uso del propio puerto y los algoritmos del software.
- Su salida ICSP permite programar los dispositivos en su hardware definitivo.
- Salida para toda la gama de Zócalos ZIF.
- Permite acoplar adaptadores para nuevas tecnologías y versiones de MCU's.

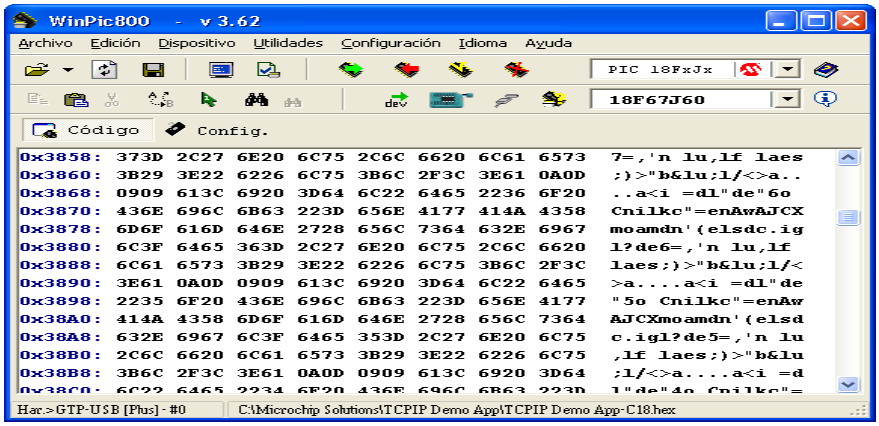


Figura II.16. Pantalla principal de WinPic800

CAPÍTULO III

LOS MICROMOTORES

3.1. SERVOS

Un servomotor (o servo) es un motor de corriente continua que tiene la capacidad de ser controlado en posición. Es capaz de ubicarse en cualquier posición dentro de un rango de operación (generalmente de 180º) y mantenerse estable en dicha posición.

En la práctica, se usan servos para posicionar superficies de control como el movimiento de palancas, pequeños ascensores y timones, para control de posición de alerones, timón, dirección (en autos), alimentación de combustible, etc, para modelos a escala, que se han vuelto populares en robótica. Estos también se usan en radio control, títeres, y por supuesto, en robots.

La fuerza de un servo se mide en oz-in, por ejemplo el servo HS-311 tiene una especificación de 42 oz-in y velocidad de *0 .19* lo que significa que puede sostener una carga de 42 onzas a una distancia medida desde el centro de rotación de su palanca de 1 pulgada lo que significa que se trata de un "torque" o "fuerza de torque".

Pasando al otro dato de la especificación, los 0.19 segundos se refiere al tiempo que tarda el servo en mover su palanca una distancia de 60 grados.

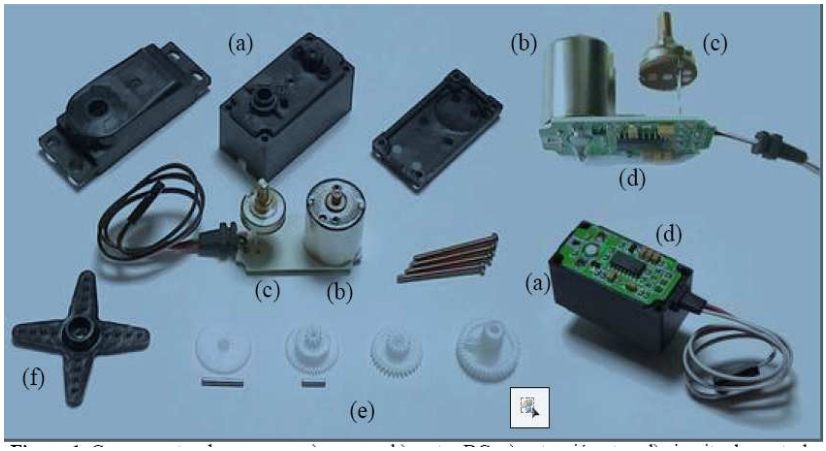


Figura III.1. Componentes de un servo

El voltaje de alimentación nominal, suele ser el que pueden proporcionar cuatro baterías de NiCd 4 x 1.2 V = 4.8 V. En la práctica este valor puede variar. Algunas compañías de radiocontrol fabrican paquetes de cinco celdas de NiCd que proporcionan un valor nominal de 6 V.

Si el voltaje es demasiado bajo, la respuesta del «servo» se hace más lenta.

En general, los servos suelen estar compuestos por 4 elementos fundamentales (ver Figura III.1):

- **Motor de corriente continua (DC):** Es el elemento que le brinda movilidad al servo.

  Cuando se aplica un potencial a sus dos terminales, este motor gira en un sentido a su velocidad máxima. Si el voltaje aplicado en sus dos terminales es inverso, el sentido de giro también se invierte.

- **Engranajes reductores:** Tren de engranajes que se encarga de reducir la alta velocidad de giro del motor para acrecentar su capacidad de torque (o par-motor).

- **Sensor de desplazamiento:** Suele ser un potenciómetro colocado en el eje de salida del servo que se utiliza para conocer la posición angular del motor.

- **Circuito de control:** Es una placa electrónica que implementa una estrategia de control de la posición por realimentación. Para ello, este circuito compara la señal de entrada de referencia (posición deseada) con la posición actual medida por el potenciómetro. La diferencia entre la posición actual y la deseada es amplificada y utilizada para mover el motor en la dirección necesaria para reducir el error.

3.1.1. Principios de funcionamiento

Los servos disponen de tres cables (Figura III.2): dos cables de alimentación (positivo y negativo/masa) que suministran un voltaje 4.8 - 6V y un cable de control que indica la posición deseada al circuito de control mediante señales PWM ("Pulse Width Modulation").

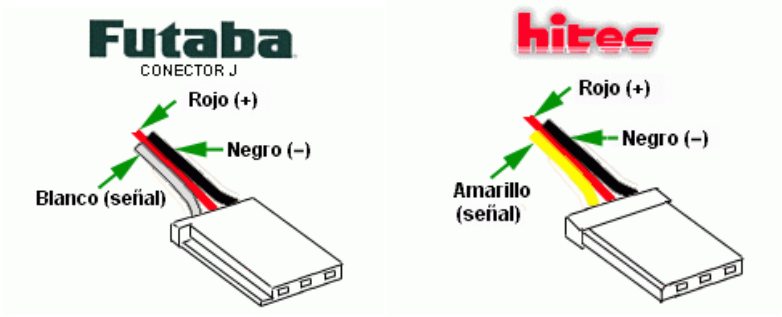


Figura III.2. Colores de los cables de los principales fabricantes de servos

Las señales PWM utilizadas para controlar los servos están formadas por pulsos positivos cuya duración es proporcional a la posición deseada del servo y que se repiten cada 20 ms (50Hz). Todos los servos pueden funcionar correctamente en un rango de movimiento de 90º, que se corresponde con pulsos PWM comprendidos entre 0.9 y 2.1 ms. Sin embargo, también existen servos que se pueden mover en un rango extendido de 180º y sus pulsos de control varían entre 0.5 y 2.5 ms (ver Figura III.3.).

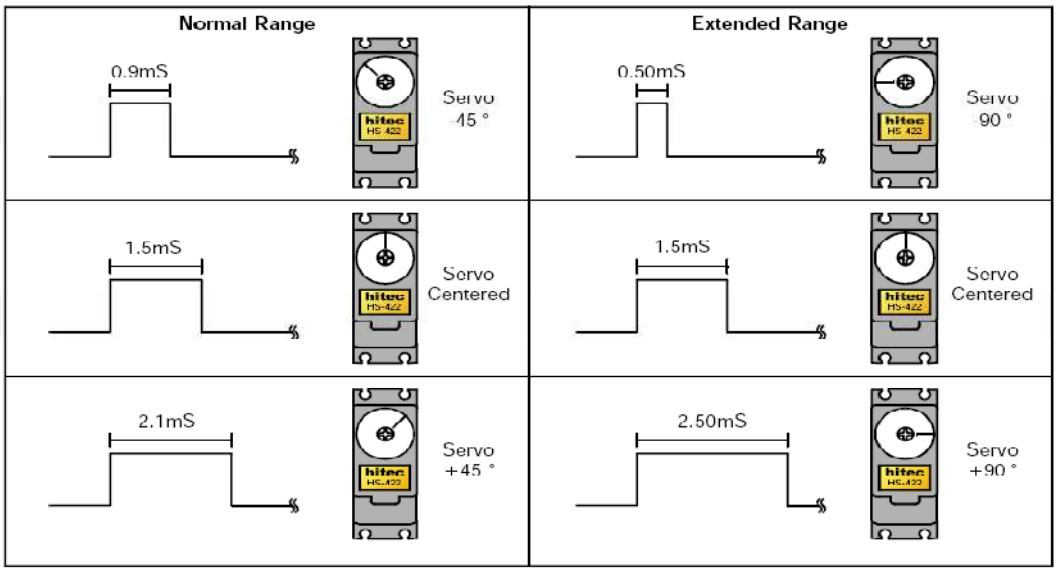


Figura III.3. Pulsos PWM para controlar servos

3.1.2. Tipologías

A lo largo de estos últimos años los servos han evolucionado enormemente en cuanto a prestaciones suministradas. El último paso adelante en la evolución de los servos es el nacimiento de los conocidos como servos digitales

Los servos digitales tienen sensibles ventajas sobre los servos convencionales

incluidos aquellos que incorporan motores "coreless" en cuanto a prestaciones, pero a cambio también muestran algunas pequeñas desventajas.

Para empezar, un servo digital es lo mismo que un servo estándar excepto que el primero incorpora un cristal de cuarzo y un microprocesador el cual analiza la señal enviada por el receptor a la vez que se encarga de controlar el funcionamiento del servomotor.

3.1.3. Principio de funcionamiento de un servo estándar o analógico

El ancho de pulso pwm oscilará dependiendo del fabricante entre 500 y 2500 msg de forma que aproximadamente un pulso de 1500 msg se le denomina pulso neutro ya que lleva el brazo a posición 0º en caso de no estar ya en dicha posición, un pulso de 500 msg representa un posicionamiento a –90º y uno de 2500 msg provoca un posicionamiento a +90º, vemos por tanto que el ancho de pulso determina el ángulo de giro del servo, es por eso que a veces cuando usamos un servo de una marca con un receptor de otra tenemos que trimmar la dirección ya que los pulsos neutros no coinciden, en la siguiente tabla (ver tabla III.1) vemos las correspondencias de algunos fabricantes:

Estos pulsos son recibidos e interpretados en el receptor y transferidos a la electrónica del servo, esta última es la encargada de realizar dos tareas fundamentales, el control de posicionamiento (detectar si hay diferencia entre la posición actual y la requerida, para ello usa el potenciómetro de posicionamiento) y la gestión de la potencia enviada al motor, es por ello que diferenciaremos entre electrónica de control y de potencia.

Tabla III.I. Tipo de servomotores

	Ancho de pulso (msg)			
Fabricante	min.	neutral.	máx.	Hz
Futaba	0.9	1.5	2.1	50
Hitech	0.9	1.5	2.1	50
Graupner/Jr	0.8	1.5	2.2	50
Multiplex	1.05	1.6	2.15	40
Robbe	0.65	1.3	1.95	50
Simprop	1.2	1.7	2.2	50

En un servo convencional cuando este se encuentra en espera no se envía tensión al motor de posicionamiento, cuando varía el ancho de pulso enviado por la emisora o se ejerce una fuerza sobre el brazo del servo que provoca la variación de posición del mismo, la electrónica de control responde ordenando a la electrónica de potencia que alimente al servomotor para mantener o alcanzar una nueva posición.

El control de la potencia enviada consiste en “chopear” la tensión nominal de alimentación del servo a una frecuencia de 50 ciclos por segundo (Hertzios), esto es lo que se conoce como control de potencia por ancho de pulso, ósea un control “PWM”, los pulsos le estarán llegando al motor cada 1/50 segundos, es decir, cada 20 milisegundos.

Una vez sentadas las bases del movimiento de un servomotor analizaremos un parámetro importante, la velocidad de posicionamiento, si variamos el ancho de los pulsos generados en la electrónica de potencia lo que estamos haciendo es aumentar el ciclo de trabajo (definimos ciclo de trabajo como el resultado de dividir el tiempo

durante el cual aplicamos tensión al motor entre el tiempo total del ciclo, es decir si el pulso tiene un ancho de 10 ms y el ancho total del ciclo es de 20msg, tendremos un ciclo de trabajo del 50%) ya que estamos aplicando tensión durante más tiempo, luego cuanto más alto sea el ciclo de trabajo, más velocidad desarrollará el servomotor y por tanto el posicionamiento será más rápido.

El segundo parámetro característico de un servo es el posicionamiento ¿cómo sabe que ha llegado a la posición que le está solicitando la emisora?, es fácil, por el potenciómetro de realimentación, la electrónica de control recibe pulsos del receptor de un ancho correspondiente a la posición deseada, a través del potenciómetro y de otros elementos obtenemos los pulsos con ancho correspondiente a la posición actual, comparándolos se obtiene el error de posicionamiento, si existe error se activa la electrónica de potencia para corregirlo, a mayor error en la posición mayor ciclo de trabajo y por tanto mayor velocidad, a medida que disminuye el error la electrónica de potencia disminuye también el ciclo de trabajo hasta alcanzar ciclo de trabajo 0.

Analizaremos ahora el tercer parámetro de funcionamiento de un servo, la banda muerta, en ingles “Deadband”, tras la explicación anterior será fácil comprender que un pulso muy estrecho, es decir, un ciclo de trabajo muy bajo, no proporcionará prácticamente ningún desplazamiento, ya que la tensión aplicada durante un lapso tan breve de tiempo no será capaz de vencer la fuerza contraelectromotriz, pues bien, definimos banda muerta como el recorrido mínimo de palote, volante o gatillo de emisora necesario para que observemos desplazamiento en el brazo del servo.

El último parámetro es la resolución, la cual se define como la mínima variación de posición alcanzable por el servo, aquí ya intervienen varios factores como son la precisión del potenciómetro de realimentación de posición y sobre todo la frecuencia de trabajo, ya que la posición no se variará con periodos inferiores a 20 milisegundos,

es decir, cada 20 ms se generará un pulso de ancho x para llevar el brazo hasta la posición deseada.

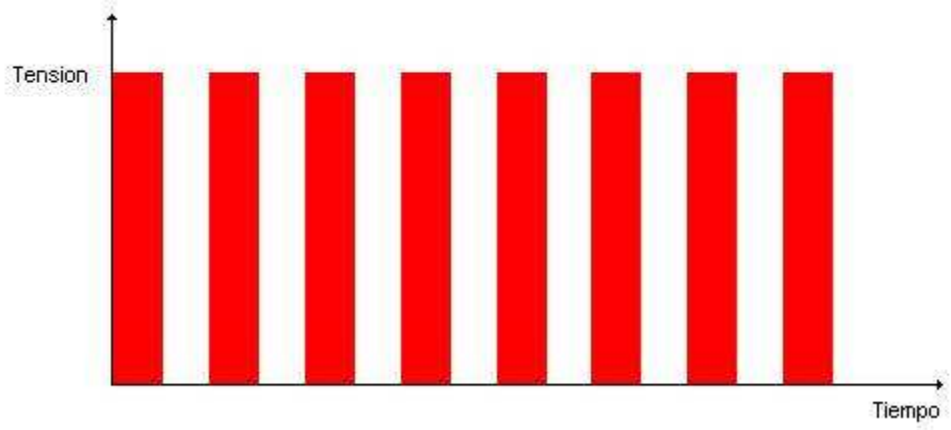


Figura III.4. Ciclo de trabajo del 50% en un servo estándar a 50 Hz

3.1.4. Principio de funcionamiento de un servo digital

En primer lugar al llevar integrado un microprocesador, es capaz mediante la aplicación de parámetros de funcionamiento de variar la forma en la que se envía potencia al servomotor, esto significa que modifica el ancho de los pulsos y por tanto el ciclo de trabajo en función de unos parámetros de funcionamiento internos (ya no solo en función de la señal enviada por el receptor) de forma que optimice el rendimiento del servomotor, también es posible modificar el funcionamiento de nuestro servo en función de nuestras necesidades, por ejemplo, invertir el sentido de giro, la velocidad de desplazamiento, ancho de pulso neutro, etc.

En segundo lugar es capaz de aumentar la frecuencia de trabajo, si con un servo estándar teníamos 50 ciclos por segundo ahora podremos tener hasta 300 ciclos por segundo con lo cual la duración del periodo baja hasta los 1/300 = 3,33 ms, lógicamente al disminuir el periodo proporcionalmente también disminuirá el ancho de pulso manejable, pero el ciclo de trabajo permanecerá constante, con lo cual

conseguimos enviar pulsos mucho más estrechos pero con más frecuencia, debido a las características constructivas y de funcionamiento de cualquier motor eléctrico se da la circunstancia de que es precisamente esta situación en la que se obtiene un mayor rendimiento del mismo, ya que con frecuencias muy altas no se descarga la bobina equivalente creada por el inducido del motor y los picos de corriente son menores, es por tanto más efectivo, en general en un motor el rendimiento es proporcional a la frecuencia de trabajo. Con este aumento de potencia no solo se consigue aumentar la velocidad de respuesta ante una variación del comando de posicionamiento si no que la variación del aumento o disminución de la potencia suministrada al aumentar la frecuencia proporciona una disminución de la banda muerta, una aceleración / deceleración mucho más rápida y suave, mayor resolución en el posicionamiento y un mayor torque, dicho aumento de torque se ve reflejado tanto en funcionamiento estático como dinámico, es decir, cuando el servo está detenido en una posición, la fuerza que hay que ejercer sobre el brazo del mismo para conseguir que gire es muy superior a la de un servo estándar, asimismo el torque de giro suministrado cuando está realizando un desplazamiento es tres veces superior al de un servo estándar.

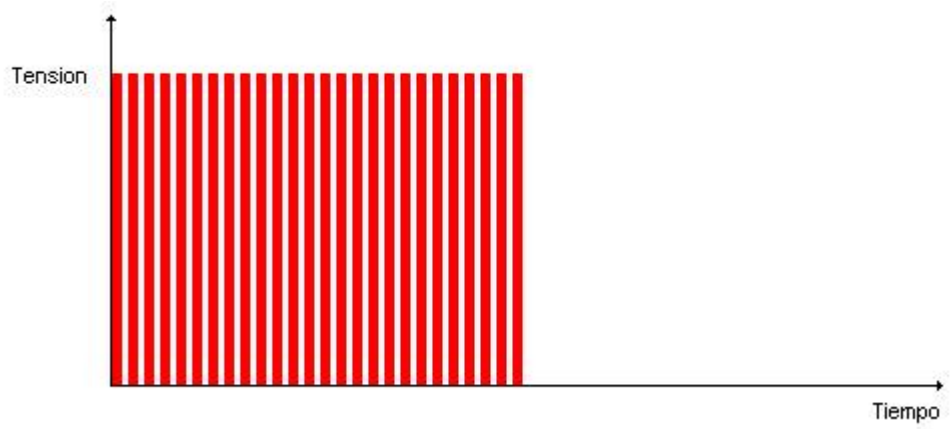


Figura III.5. Ciclo de trabajo del 50% en un servo digital a 300 Hz

Los servos digitales son capaces de memorizar parámetros de programación, que varían de acuerdo a cada fabricante pero en general son:

1 - Se puede programar el sentido de giro como "normal" o "inverso".

2 - Se puede variar la velocidad de respuesta del servo.

3 - Se puede programar una posición central (o posición neutra) diferente, sin afectar los radios de giro.

4 - Se pueden determinar diferentes topes de recorrido para cada lado.

5 - Es posible programar qué debe hacer el servo en caso de sufrir una pérdida de señal.

6 - Es posible programar la resolución, es decir cuánto se mueve el control en el radio sin obtener un movimiento en el servo.

3.1.5. Inconvenientes de los servos digitales

El principal inconveniente que presentan es el consumo de potencia, al mantener un ciclo de trabajo idéntico al de un servo estándar pero aumentar la frecuencia lógicamente también aumenta el consumo, dicho aumento de consumo sería aproximadamente un 60% superior aun servo estándar de “similares” prestaciones. Esto se hace notar ya que los servos digitales zumban constantemente indicando que están haciendo su "trabajo" aún en estado de reposo.

En resumen, es totalmente recomendable disponer de uno o más de estos fantásticos servos si lo que quieres es:

- Mayor resolución en el posicionamiento.

- Menor ancho de banda muerta.
- Mayor eficacia en el posicionamiento (Repetitibilidad).
- Respuesta más rápida ante órdenes del control.
- Mayor torque en cualquier situación.

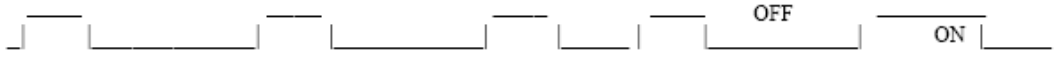
3.1.6. Control de los servos

Para controlar los servos se les deben enviar pulsos PWM a través del cable de control. En los sistemas de modelismo, se utilizan dos componentes para controlar los servos: un receptor y una emisora. El receptor es el componente que se encarga de recibir los comandos inalámbricamente de la emisora y transformarlos en los pulsos PWM correspondientes que son enviados a los servos. La emisora es un mando que transmite las órdenes al receptor a través de señales inalámbricas con modulación AM, FM o PCM.

Figura III.6. Pulsos PWM para controlar servos

Para controlar un servo, usted le ordena un cierto ángulo, medido desde 0 grados. Usted le envía una serie de pulsos. En un tiempo ON de pulso indica el ángulo al que debe posicionarse; 1ms = 0 grados, 2.0ms = máx. grado (cerca de 120) y algún valor entre ellos da un ángulo de salida proporcional. Generalmente se considera que en 1.5ms está el "centro." Entre límites de 1 ~ 2ms son las recomendaciones de los fabricantes; usted normalmente puede usar un rango mayor de 1.5ms para obtener un ángulo mayor e incluso de 2ms para un ángulo de rendimiento de 180 grados o más. El factor limitante es el tope del potenciómetro y los límites mecánicos construidos en el servo. Un sonido de zumbido normalmente indica que usted está forzando por encima al servo, entonces debe disminuir un poco.

El tiempo de OFF en el servo no es crítico; puede estar alrededor de los 20ms. Hemos usado entre 10ms y 30 ms. Esto No tiene que ser de ésta manera, puede variar de un pulso a otro. Los pulsos que ocurren frecuentemente en el tiempo de OFF pueden interferir con el sincronismo interno del servo y podría escucharse un sonido de zumbido o alguna vibración en el eje. Si el espacio del pulso es mayor de 50ms (depende del fabricante), entonces el servo podría estar en modo SLEEP entre los pulsos. Entraría a funcionar en pasos pequeños y el rendimiento no sería el óptimo.
Este es un ejemplo de la señal que debería tener el servo:



El tiempo de OFF está variando, como se puede observar. Esto no tiene efectos adversos con tal de que esté entre 10 ~ 30ms. El tiempo de ON determina la posición del brazo de salida.

¿Supongamos que queremos mover el servo a 30 grados?

Para controlarlo a 30 grados; se debe calcular la longitud (ancho) del pulso:

En 0 grados =1ms, 120 grados = 2ms => 30 grados =1.16ms. Relación lineal.

Así, si seguimos enviándole pulsos de 1.16ms, incrementaremos su posición en 30 grados. Si hay una fuerza externa que intenta bloquearlo, el servo intentará resistir activamente.

También es posible dejar de enviar pulsos después que el servo se ha movido a su posición. Si dejamos de enviar pulsos por más de 50ms (dependiendo del servo), este podría caerse. Esto significa, que este no estaría aplicando ninguna entrada al motor, o activamente resistiendo fuerzas externas; solamente la fricción sostendrá el brazo (del servo) en su lugar.

Aunque los «servos» son los posicionadores casi ideales, son también fáciles de modificar para aplicaciones especiales. Por ejemplo, se puede alterar el circuito de retroalimentación para modificar el rango de giro. La mayoría de los servomotores se han diseñado para un viaje de unos 90º, pero en muchos casos esta limitación puede superarse.

Cuando se necesite mayor cantidad de giro de la que el fabricante ha dotado al «servo», la mejor solución es actuar modificando el potenciómetro del circuito de retroalimentación.

Para ello se añade una resistencia de un valor comprendido entre 1K5 a 2K2 en serie con cada extremo del potenciómetro y luego se vuelve a montar. De esta forma los pulsos de la señal de control aumentarán el rango de giro.

También se puede modificar el «servo» para que se comporte como un pequeño motor controlador mediante pulsos. Si se quita el potenciómetro interno y se sustituye por dos resistencias de 2K5, el circuito interno creerá que el eje del motor se encuentra siempre en posición centrada, así pues, si se envía señal de control para que se posicione a la derecha, el «servo» tratará de corregir continuamente la posición y girará en ese sentido. Se tiene de este modo un motor con engranajes cuya dirección de rotación pueden ser controladas por un tren de pulsos mediante la técnica PWM.

3.1.7. Ventajas

Entre las ventajas que aporta el empleo de un «servo» están las siguientes: poco peso, alta potencia (torque de fuerza), fiabilidad, fortaleza (los «servos» y su electrónica normalmente sobreviven a choques y funcionan en ambientes de alta temperatura, suciedad, humedad y vibraciones), simplicidad, versatilidad y bajo coste.

3.2. MICROSERVOS

Cada vez es más difícil encajonar en una categoría a los servos. Ya que lo que antes llamábamos un microservo ahora serían miniservos y los que hace apenas unos tres o cuatro años eran los nanoservos ahora son microservos, además depende de la marca el nombre que le da a sus servos: nano, mico, mini, estándar, gigantes, etc.

Los microservos se utilizan principalmente en aviones de vuelo de interiores no muy pequeños como los foamies y van aproximadamente de los 4 gramos a los 8 gramos de peso aunque los servos de hasta 19 gramos aún se pueden considerar microservos. Después vienen los Miniservos que quedarían entre los 20 gramos y los 40 gramos donde ya se empieza a dibujar la línea de los servos estándar.

3.2.1. Micromotor servo Kanakatta Hiji

El micromotor servo KANAKATTA HIJI (ver figura III.7.) es un micromotor servo digital programable para su control a través de un protocolo serial asíncrono a 2400 bps.

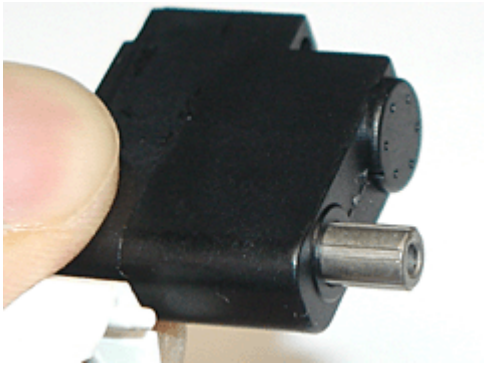
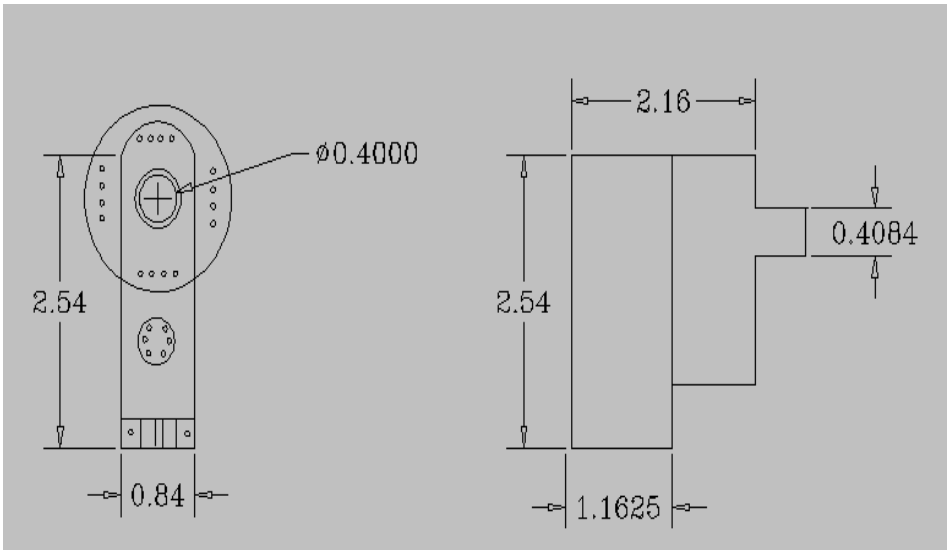


Figura III.7. Micromotor servo Kanakatta Hiji

Dimensiones:

- Altura 2.16 cm
- Ancho 2.54 cm
- Profundidad 0.84 cm



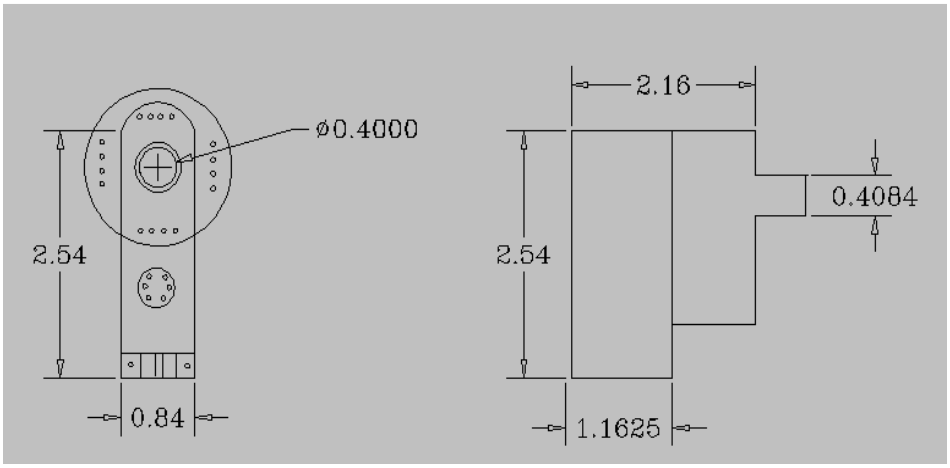


Figura III.8. Dimensiones del micromotor

Características principales:

- Peso 12 gr.
- Máximo grado de movimiento 220º,
- Engranajes de metal en embrague.
- Rated Motors Speed (RPM) : 22.4
- Rated Servo Speed (RPM) : 31
- Gear Ratio : 3.33
- Servo Speed (RPM) : 0.32
- Starting Torque (gcm) : 723.33
- Starting Torque (Kgcm) : 2.5
- Torque Constante (Kgcm) : 1.26
- Utiliza comunicación serial asíncrona.

Componentes fundamentales del micromotor servo:

- **Motor de corriente continua (DC):** Este micromotor servo posee un motor

dc tipo brushless de 6mm de diámetro y 14mm de longitud (ver figura III.9)

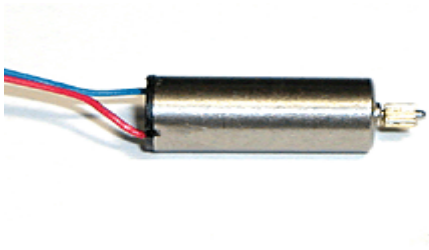


Figura III.9 Motor DC

- **Engranajes reductores:** El motor tiene montado en su eje un piñón de 7 dientes (ver figura III.12) y esta acoplado a un tren de engranajes de 4 escalonamientos y este acoplado al engranaje de salida como se muestra en las figuras III.10 y III.11

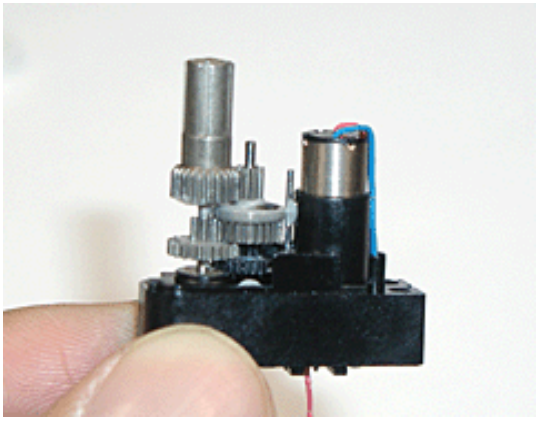


Figura III.10. Engranajes del motor

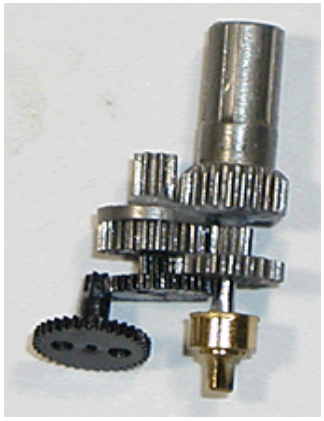


Figura III.11. Acoplamiento de engranajes

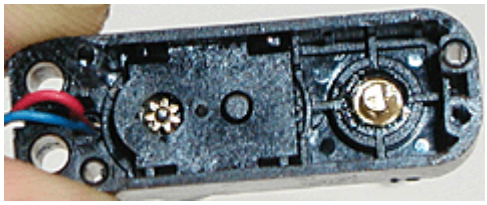


Figura III.12. El piñón del motor (a la izquierda) muestra un período de siete dientes

En el Primer escalonamiento (ver figura III.13.) La rueda motriz tiene 34 dientes y la conducida tiene 6 dientes.

Figura III.13. Primer escalonamiento

En el segundo escalonamiento (ver figura III.14) La rueda motriz tiene 31 dientes y la conducida tiene 6 dientes.

Figura III.14 Segundo Escalonamiento

En el Tercer escalonamiento(ver figura III.15) La rueda motriz tiene 20 dientes y la conducida tiene 8 dientes.

Figura III.15. Tercer Escalonamiento

En el Cuarto escalonamiento (ver figura III.16.) La rueda motriz tiene 24 dientes y la conducida tiene 8 dientes.

Figura III.16. Cuarto Escalonamiento

Y el último engranaje consta de 24 dientes (ver figura III.17.)

Figura III.17. Quinto Escalonamiento

Por lo tanto, la relación es:

La primera etapa 7:34

La segunda etapa 6:31

La tercera etapa 6:20

La cuarta etapa 8:24

La quinta etapa 8:24

calculamos la relación de reducción por medio de la formula:

$i = \frac{\pi Dm}{\pi Dc}$ donde;

I= relación de reducción.

Dm =numero de dientes de la rueda motriz.

Dc= numero de dientes de la rueda conducida.

i= 752

se tiene una reducción de 1:752.

- **Sensor de desplazamiento:** El sensor de desplazamiento está compuesto por un potenciómetro que se utiliza para conocer la posición angular del motor.

Figura III.18. Potenciómetro

- **Circuito de control:** El circuito de control del servo está compuesto por una placa electrónica que está compuesta por un microcontrolador para el control de posición del motor y un puente H(ver fig. III.19 con mosfet para el control de giro del motor dc).

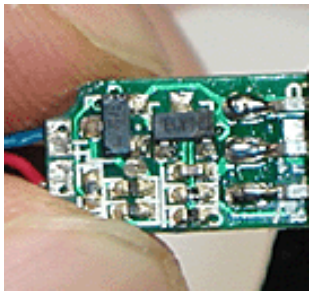


Figura III.19. Puente H

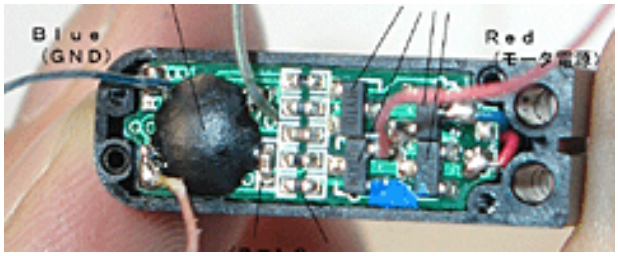


Figura III.20. Microcontrolador

3.2.2. Principios de funcionamiento del micromotor servo

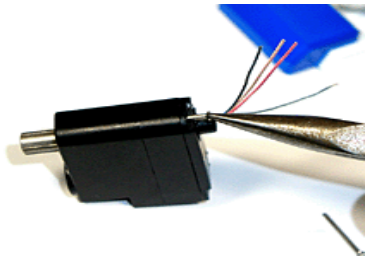


Figura III.21. Micromotor servo

Este micromotores servos para su control y alimentación disponen cuatro cables (Figura III.21.):

- cable Rojo: energía para el servomotor (Vcc)
- cable Azul: GND
- cable naranja: Más potencia para el servo (3.3V).
- cable verde: señal (serie asíncronos 2400bps, no paridad, 1 bit de parada, sin control de flujo)

El control de la posición del micromotor servo se realiza por medio del protocolo

consecutivo (2400 8-N-1), básicamente el MCU envía una trama de 8 bytes (1 bytes de orden de inicio de trama, 5 bytes de servo, una suma de control y un byte de final de trama).

[inicio de trama] [posición mservo 1] [posición mservo 2] [posición mservo 3] [posición mservo 4] [posición mservo 5] [Suma de control] [fin de trama].

Lo que nos permite manejar hasta 5 micromotores servos con una sola trama de datos, cuando se maneja menos de 5 micromotores servos se utiliza el valor 128 para los bytes donde no hay servo para manejar.

El valor del byte de inicio de trama es 0x05, y el valor del byte de fin de trama es 0xFF (es el pulso positivo más largo de la trama: 3.76ms)

El byte de suma de control se calcula sumando los 6 bytes anteriores a esta suma se le resta 255 hasta obtener un valor menor o iguala 255

Ejemplo:

Si tenemos la trama

[5] [38] [53] [128] [128] [128] [225] [255]

5 +38 +53 +128 +128 +128 = 480, restando 255 = 225

CAPÍTULO IV

DISEÑO DEL MICROROBOT.

4.1. REQUERIMIENTOS DEL SISTEMA

En este proyecto se requirió diseñar e implementar un microrobot bípedo con diecisiete grados de libertad el mismo que será distribuido de la siguiente manera: cinco grados de libertad en cada pierna, tres en cada brazo y finalmente uno en la cabeza. El robot es autónomo el mismo que bailara al son de un ritmo musical, permitiéndose bailar hasta 3 ritmos diferentes. Este tipo de robots es ideal para la utilización en concursos de robótica como también se podría utilizarlo como medio de publicidad en locales comerciales y un medio de entretenimiento para todo tipo de personas. En la figura IV.8 se muestra la estructura del microrobot.

4.2. PARTES DEL SISTEMA

En el desarrollo de cualquier proyecto siempre hay que tener un orden de actuación determinado, es decir una buena planificación. En este caso, se ha recurrido a dividirlo en dos niveles o etapas:

Estructura mecánica. Comprende la estructura física, las unidades motoras y las etapas de potencia. Es posible encontrar desde sistemas muy sencillos basados en un único motor, hasta estructuras sumamente complejas.

Electrónica de control. Incluye los circuitos más básicos que relacionan las salidas de los sensores con las restantes unidades. Partiendo de una simple lógica digital hasta potentes microcontroladores, se busca dotar al microrobot de la capacidad para procesar la información, así como actuar de una manera controlada sobre las unidades motoras.

4.2.1. Estructura Mecánica

Para diseñar el microrobot se han tenido en cuenta todas las ideas anteriores y alguna otra añadida por la propia experiencia de los autores. A continuación se enumeran todas, sin ningún orden de importancia:

1. La estructura será de plástico (extremidades), acrílico (cuerpo) y madera de balsa (cabeza, manos y pies) de manera que será bastante ligero y liviano.
2. Los motores serán servomecanismos de aeromodelismo del tipo kanakata hiji el modelo más optimo por el torque de salida, sus dimensiones y al ser muy económico.
3. Los ejes de giro serán dobles, de forma que el robot adquiera una mayor rigidez y estabilidad.
4. Las extremidades, que serán cuatro, estarán situadas al cuerpo. De esa forma la apariencia será la de un humano.
5. Para el diseño de las diferentes partes y piezas de la estructura mecánica del microrobot se ha utilizado el software Autocad 2007 el mismo que oferta facilidad, robustez y operabilidad de diseño.

La estructura está formada por diecisiete micromotores, cinco por cada pierna, tres por cada brazo y uno para la cabeza.

En la figura IV.1 se detalla el diseño de las diferentes partes que conforman el microrobot:

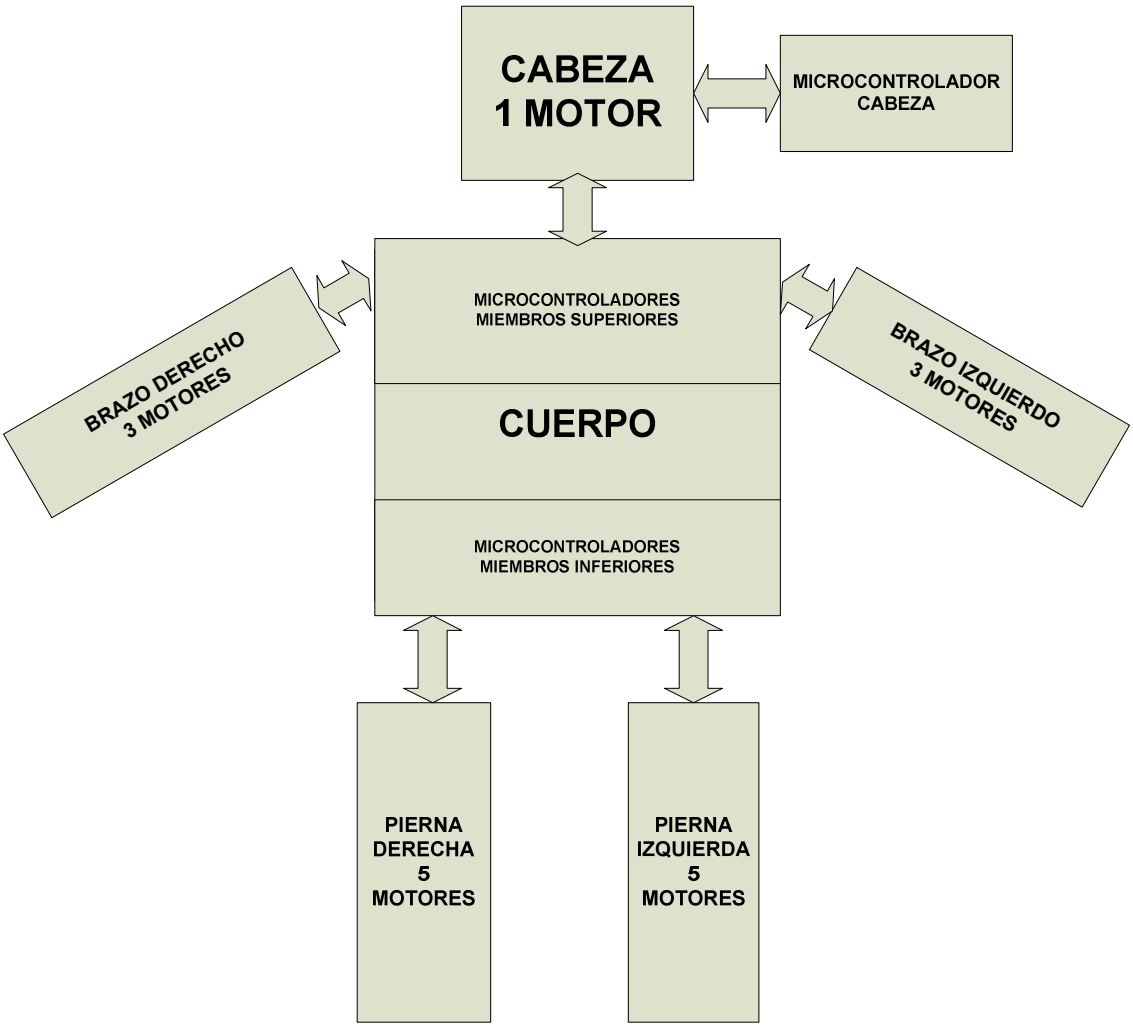


Figura IV.1. Diseño de la estructura del microrobot

Cabeza

Para la construcción de la estructura de la cabeza se cortó un pedazo de balsa y se procedió a moldearla de acuerdo al modelo diseñado.

La cabeza está unida al cuerpo por medio de un micromotor esta tiene un grado de libertad de movimiento el mismo que permitirá mover a la cabeza hacia la derecha e izquierda hasta en un ángulo de 90° respectivamente , en la figura IV.2.

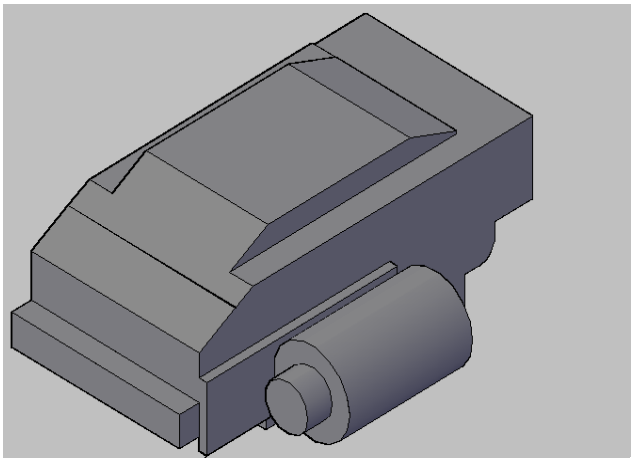


Figura IV.2. Diseño de la cabeza

Brazos

En la figura IV.3 se muestra la estructura de uno de los brazos, apreciándose como el primer micromotor (M1) está unido al cuerpo del microrobot y mediante su eje mueve la primera articulación (A1), que está unida al eje del segundo micromotor (M2).

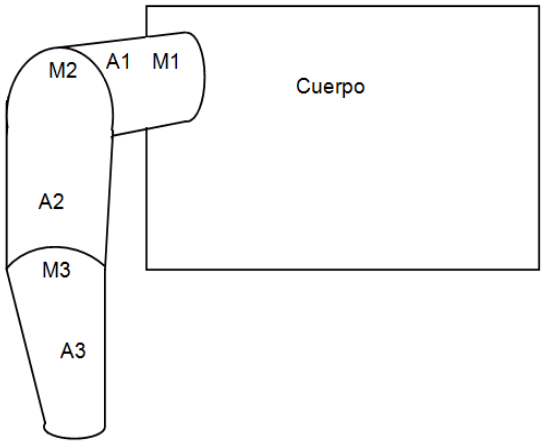


Figura IV.3. Diagrama del brazo del microrobot.

Siendo éste el que forma la articulación del brazo (A2) que está unido al eje del tercer micromotor (M3) que forma la articulación del antebrazo y la mano (A3).En donde cada micromotor genera un grado de libertad de movimiento, en la figura IV.4. se observa una foto del brazo.

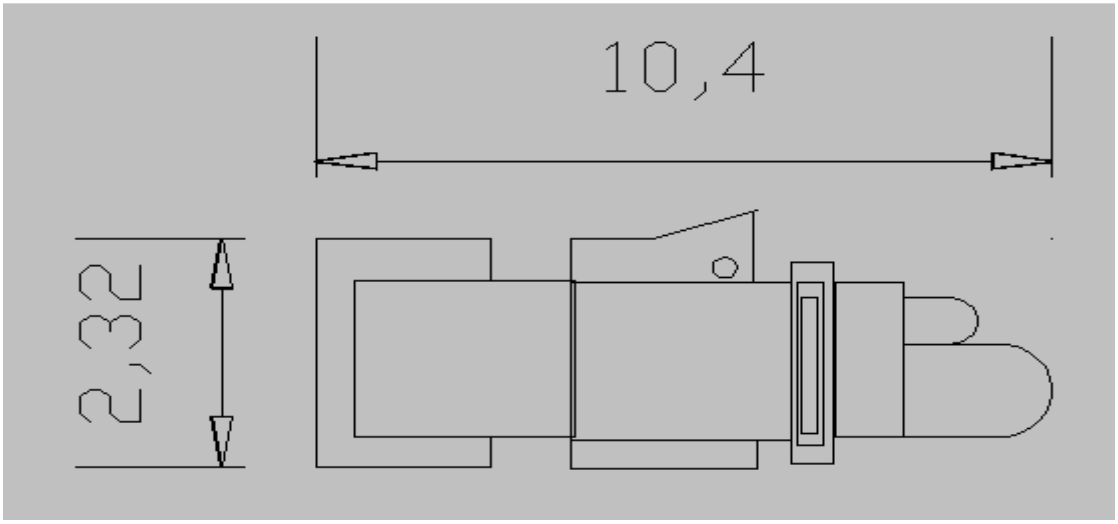


Figura IV.4. Diseño del brazo derecho.

Piernas

Sin duda que estas son la parte más importantes tanto en el diseño como en la construcción del microrobot ya que de ellas depende el soporte para los demás componentes y para el equilibrio del microrobot.

Cada pierna está constituida por 5 micromotores ubicados como se muestra en la figura IV.5, uno en el pie, 2 en la rodilla, 2 en la cadera lo que proporcionará 5 grados de libertad de movimiento a cada pierna, permitiendo simular la caminata de una persona como también diferentes ejercicios que se realizan con las piernas.

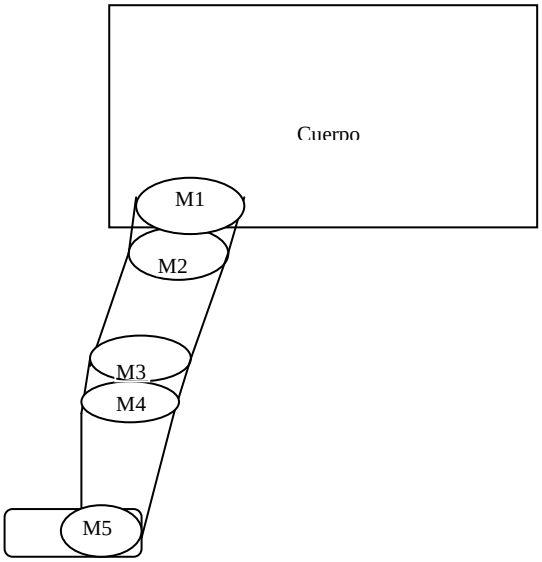


Figura IV.5 Diagrama de una pierna del microrobot

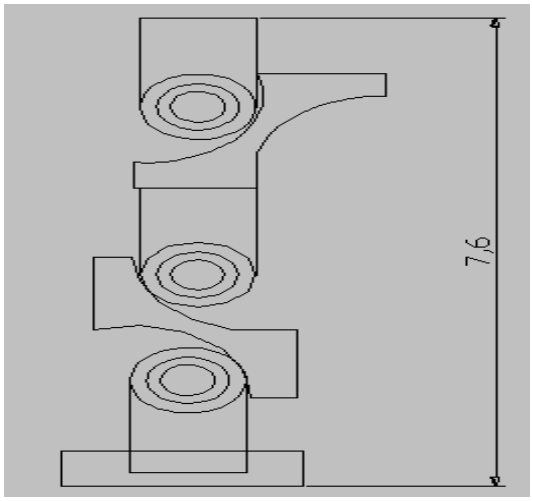


Figura IV.6 Diseño de una pierna del microrobot

Cuerpo

En esta parte es donde se alojara el circuito de control, etapa de alimentación como también es el soporte donde se unen las extremidades y la cabeza del microrobot como se ve en la figura IV.7

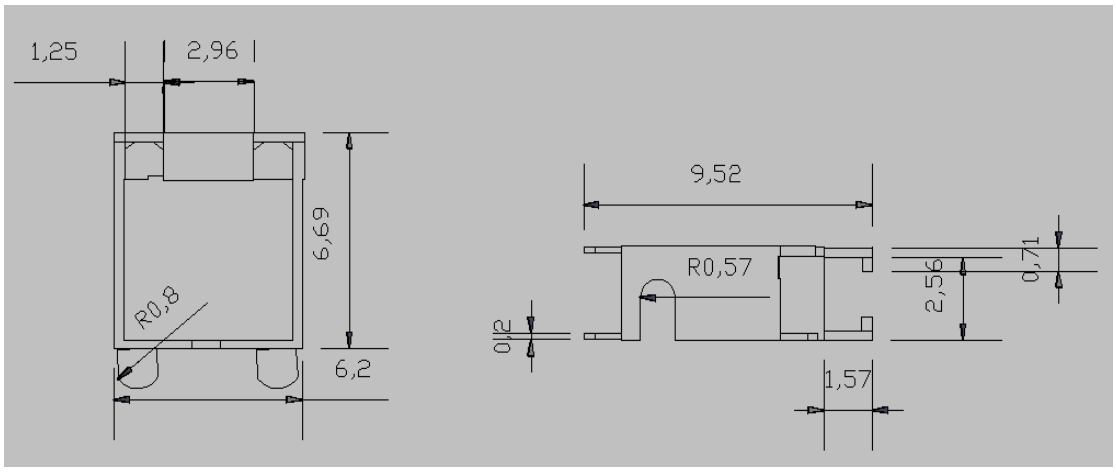


Figura IV.7 Diseño del cuerpo del microrobot

Robot Completo

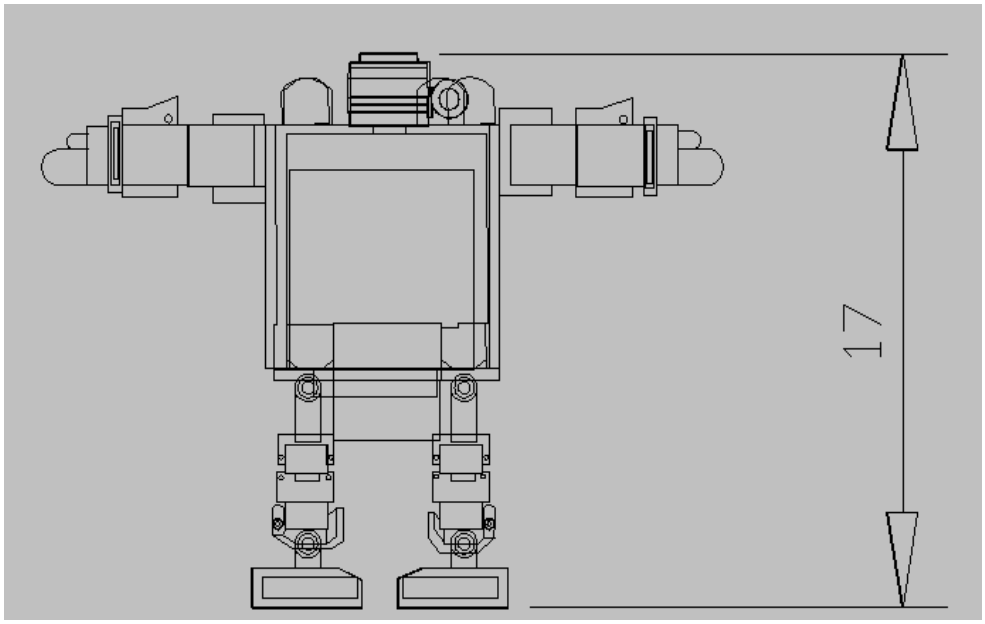


Figura IV.8. Diseño del microrobot

4.2.1.1. Métricas para la elección del micromotor

Las métricas consideradas para la selección del micromotor para el microrobot bailarín son:

- **Precio:** Un factor importante pues se necesita abaratar los costos.
- **Disponibilidad:** En nuestro medio, ¿qué tan difícil es encontrarlo?, se tomó en cuenta la disponibilidad del componente en el país y en el exterior.
- **Facilidad de control:** Muy importante pues se necesita saber que tan difícil es manejarlo al momento del diseño del microrobot.
- **Torque :** Muy importante para saber la fuerza que maneja el micromotor servo.
- **Tamaño :** el factor más importante pues se necesita reducir el tamaño del microrobot.

4.2.1.2. Estudio de la mejor alternativa

Para la selección del micromotor se aplicó el Método Ponderado que consiste en dar un valor cuantitativo a las métricas consideradas, en este caso le asignamos valores entre 1 y 10, a continuación los presentamos en la Tabla IV.I.

Tabla IV.I. Asignación cuantitativa para las métricas

Métricas (Factores)	**Valores para Ponderación**	**Ponderación (V. P/31)**
Precio	8	0,19
Disponibilidad	6	0,14
Facilidad de control	8	0,19
Torque	10	0,24
Tamaño	10	0.24
Total	**42**	1,00

De acuerdo a los precios y la información técnica de cada uno de los dispositivos se determinó una calificación o costo del 0 al 10 (Tabla IV.II.). Esta información se obtuvo de las páginas oficiales de los fabricantes

Tabla IV.II. Calificaciones y costos

Factor **Alternativa**	**Precio USD (inc. envío)**	**Disponibilidad**	**Facilidad de control**	**Tamaño**	**Torque Kg-f**
Hitec	32,00	5	7	3	3
Futaba	35,00	4	7	3	3
Kanakatta Hiji	25,00	8	9	10	2.5
Mks	25,00	4	7	8	1

Posteriormente realizando una matriz de puntos seleccionamos la alternativa adecuada. Esta matriz consta de varios cálculos matemáticos (Tabla IV.III).

Para obtener la calificación ponderada se multiplica el valor ya ponderado por la calificación asignada en la tabla anterior (Tabla IV.II). Para los valores en dólares se selecciona el valor menor, para los valores en kg-f se selecciona el valor mayor a ese valor se le asigna la mayor calificación, es decir 10. A continuación se obtienen los excedentes de los demás valores y se aplica una regla de tres. Los procesos mencionados se resumen en la siguiente tabla.

Tabla IV.III. Matriz de puntos

Factores	**Ponderación**	**Hitec**		**Futaba**		Kanakatta Hiji		Mks	
		Calif	**Calif. P**	**Calif.**	**Calif. P**	**Calif.**	**Calif. P**	**Calif.**	**Calif. P**
Precio	0,19	8	1.52	7	1.33	10	1.9	10	1.9
Disponibilidad	0,14	5	0.7	4	0.56	7	0.98	5	0.7
Facilidad de control	0,19	7	1.33	7	1.33	9	1.71	7	1.33
Torque	0,24	10	2.4	10	2.4	8	1.92	3	0.72
Tamaño	0.24	3	0.72	3	0.72	10	2.4	8	1.92
total	1,00	-	6.67	-	6.34	-	**8.91**		6.57

La alternativa que se seleccionó es la que tiene mayor costo, en este caso se observa que es: **Kanakatta Hiji.**

4.2.2. Diseño Electrónico

El microrobot depende de un control ejecutado por cinco microcontroladores PIC, un microcontrolador de la cabeza (PIC16f628A) y cuatro microcontroladores (PIC16f628A) para las extremidades los cuales con la señal de encendido se sincronizan, se empleará la arquitectura que se muestra en la figura IV.9

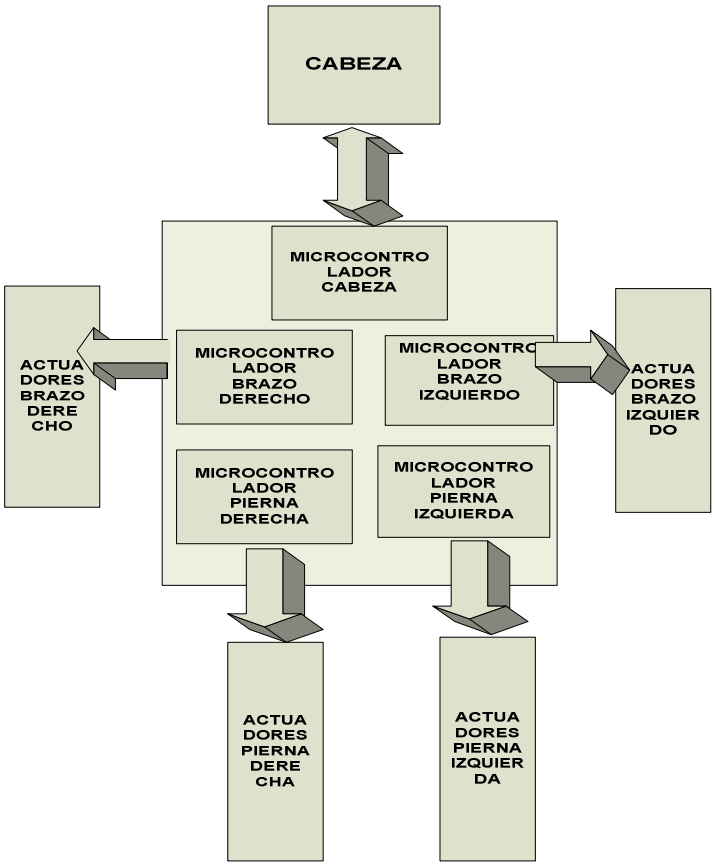


Figura IV.9. Arquitectura del software

4.2.2.1. Microcontrolador Cabeza

Este microcontrolador me permite controlar el micromotor que está ubicado en la cabeza del robot, así como también para encender los diodos led.

Utilizaremos el microcontrolador 16f628a, activado mediante una resistencia de 100KΩ conectado a 5V y ésta a su vez al pin 4 del pic que es el MCLR.

Para el modo de oscilación utilizamos el modo XT, un cristal de 4 MHz que es conectado a los pines OSC1 y OSC2 para establecer una oscilación. La Figura IV.10

muestra la conexión de estos pines a un oscilador de cristal. El diseño del oscilador requiere el uso de un cristal que sea de tipo para operación resonante en paralelo con 2 capacitores de 22pF.

El uso de un cristal para operaciones resonantes en serie puede producir una frecuencia fuera de las especificaciones del fabricante.

Para controlar el micromotor de la cabeza, utilizaremos el pin 8 RB2 (TX) del pic que realiza la función de comunicación serial (Usart), que solo se puede utilizar con la sentencia de programación HSEROUT para la transmisión de los datos.

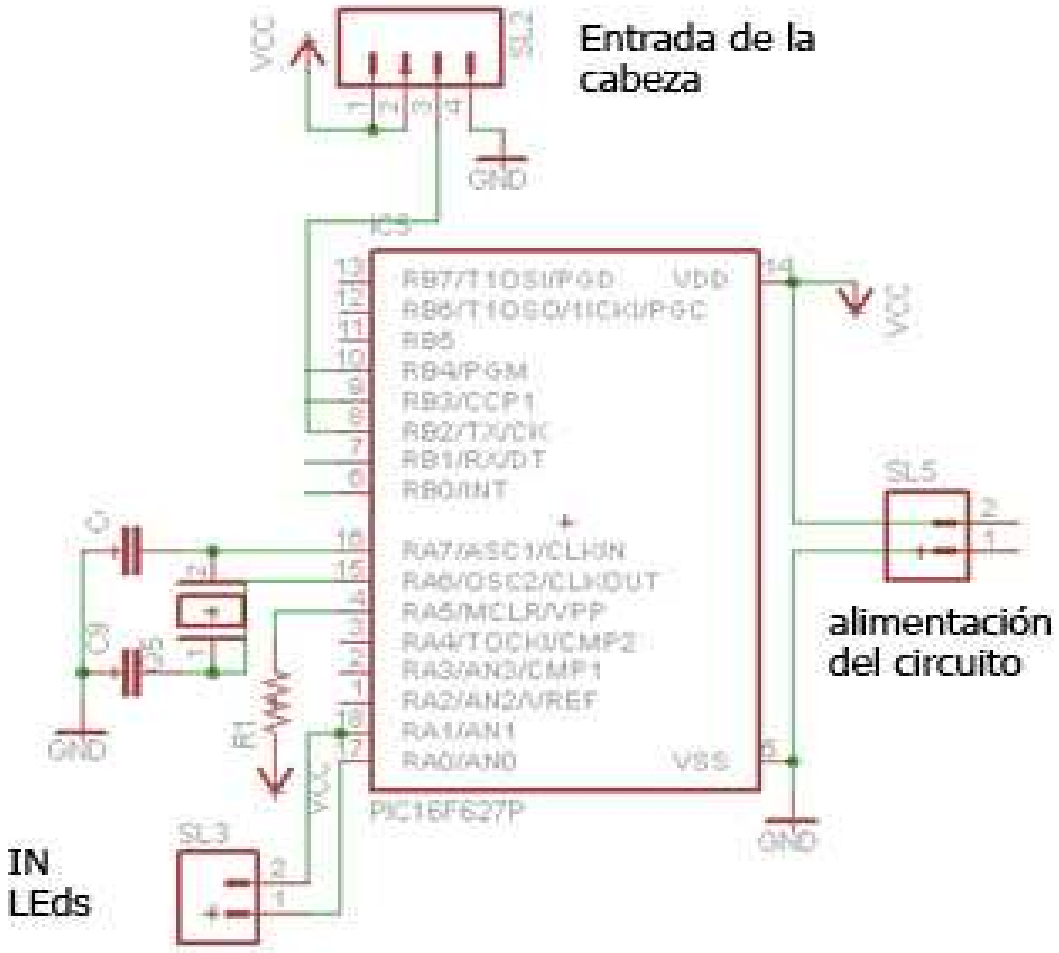


Figura IV.10. Esquema de pic para controlar la cabeza

Para enceder los diodos leds de la cabeza se utiliza los pines A0 y A1.

En la figura IV.11 podemos observar la disposición de los componentes utilizados al realizar el circuito

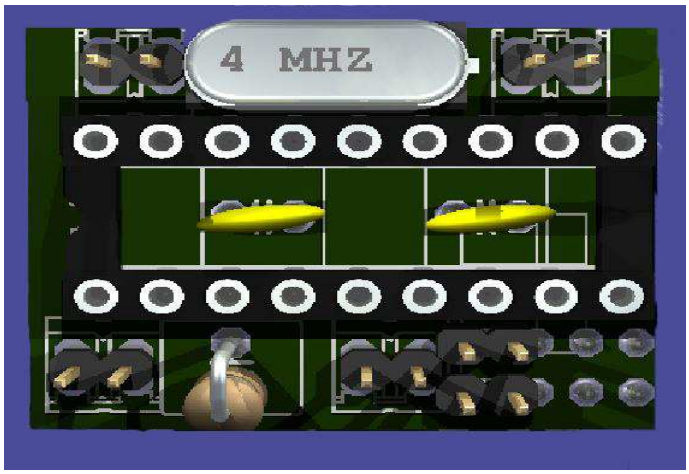


Figura IV.11. Distribución del pic para el control de la cabeza

HSEROUT

Envíe uno o varios datos en serie que es una comunicación consecutiva asincrónica. El HSEROUT es una de varias funciones consecutivas asincrónicas incorporadas, sólo puede ser usado con dispositivos que tienen un hardware USART.

Los parámetros consecutivos y la velocidad de transmisión son especificados de la siguiente manera:

DEFINA HSER_RCSTA 90 Determinar la habilitación del registro de transmisión

DEFINA HSER_TXSTA 20h Determinación de la velocidad de transmisión

DEFINA HSER_BAUD 2400 Determinar la rata de transmisión

DEFINA HSER_SPBRG 25 Determina directamente el SPBRG (normalmente puesto por HSER_BAUD)

El HSEROUT asume un oscilador de 4MHz calculando la velocidad de transmisión. Mantener el cronometraje de velocidad de transmisión apropiado con otros valores de oscilador, estar seguro para DEFINIR <define.html> el OSC que pone al nuevo valor de oscilador.

4.2.2.2. Microcontroladores para las extremidades

Para manejar cada una de las extremidades superiores e inferiores del robot, se realiza mediante 2 etapas.

En cada etapa, utilizamos dos microcontroladores pic 16f628a de la serie de microchip que será el transmisor de datos a cada una de las extremidades del robot, para esto utilizamos el pin 8 RB2(TX) del pic que realiza la función enviar tramas de datos de manera serial mediante la sentencia HSEROUT.

Del mismo modo que el pic para el control de la cabeza, utiliza los mismos elementos de configuración de activación así como también la misma configuración de oscilación TX, con un cristal de 4Mhz.

Extremidades Inferiores (Etapa 1)

Para la pierna derecha como para la pierna izquierda se utiliza un PIC respectivamente. La alimentación de la etapa 1 se realiza mediante 3 baterías en serie de 1.5V de 800mA.

En la figura IV.13 podemos observar la disposición de los diferentes componentes utilizados en la realización de la placa, SL1 y SL3 son pines para la alimentación, J1 y

J2 son las entradas de las piernas tanto derecha como izquierda y SL2 es para utilización de propósitos generales.

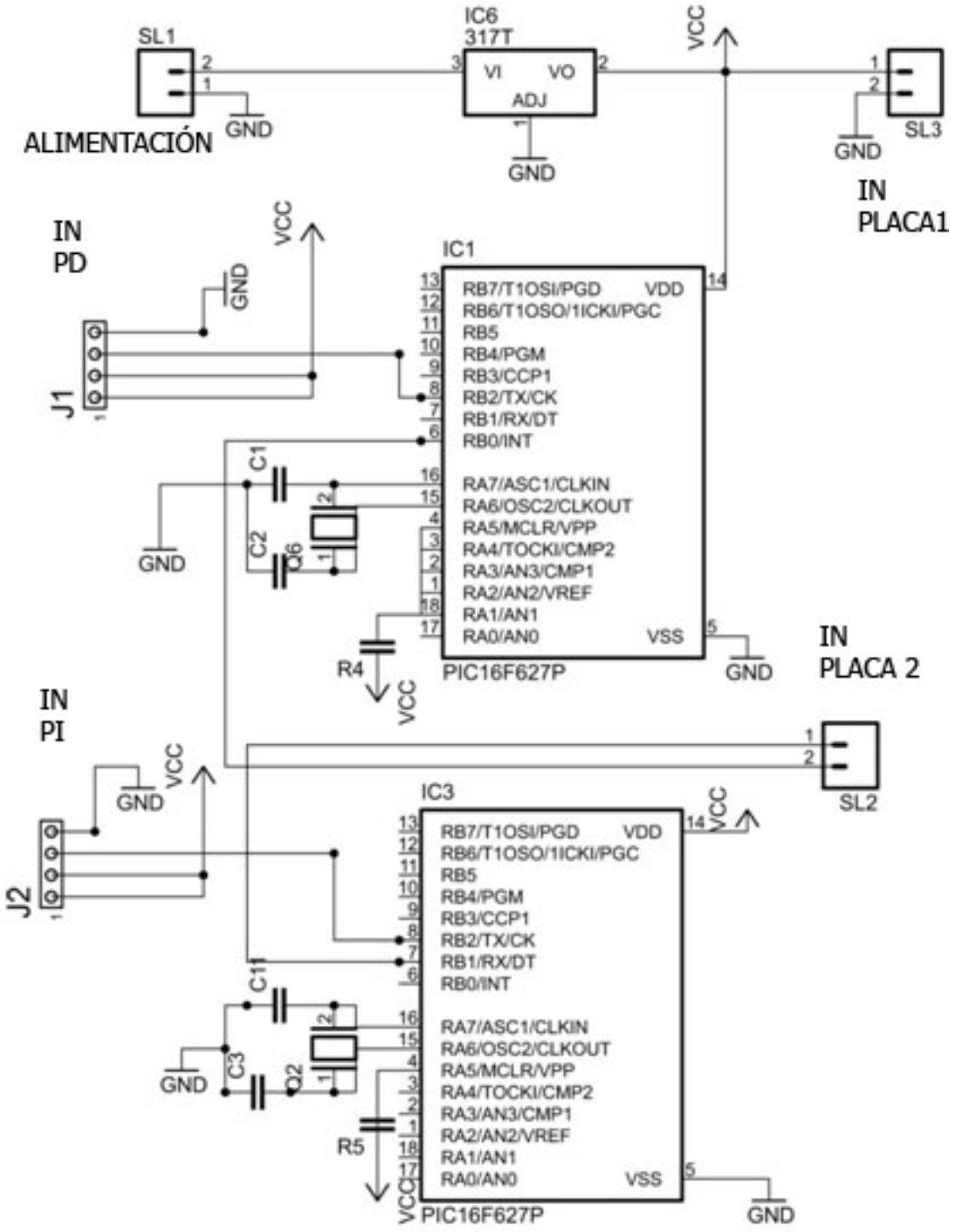


Figura IV.12. Esquema de pics extremidades inferiores etapa 1

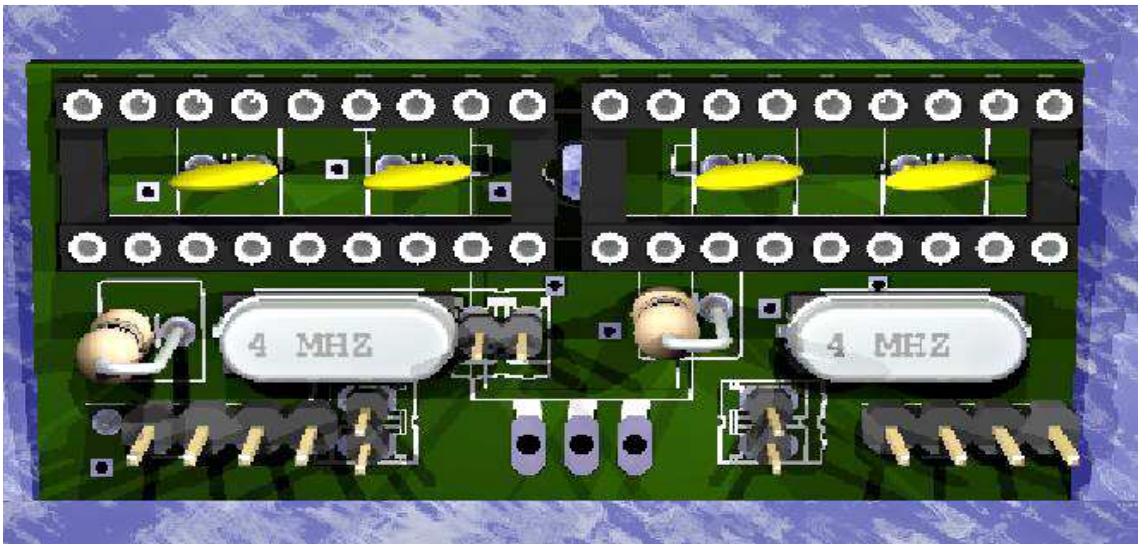


Figura IV.13. Distribución de los pics etapa 1

Extremidades Superiores (Etapa 2)

Por cada brazo se utiliza un pic como se muestra en la figura IV.14.La alimentación de la etapa 2 se alimenta de la etapa 1. En la figura SL1 y SL3 son pines de entrada de alimentación, el J1 y J2 son entradas de los brazos derechos e izquierdo respectivamente y SL2 son puertos para utilización general.

En la figura IV.15 podemos observar la disposición de los diferentes componentes utilizados en la realización de la placa.

En la figura IV.16 podemos observar el diagrama de flujo , el mismo que al encenderse envía una señal de sincronización y setea la cabeza , realiza la función de encender y apagar los diodos led .

En la figura IV.17 podemos observar el diagrama de flujo de los PIC de las extremidades, el mismo que al encenderse con la señal de sincronización setea todo los micromotores del microrobot, el flujograma envía las tramas para empezar a ejecutar las sentencias programadas.

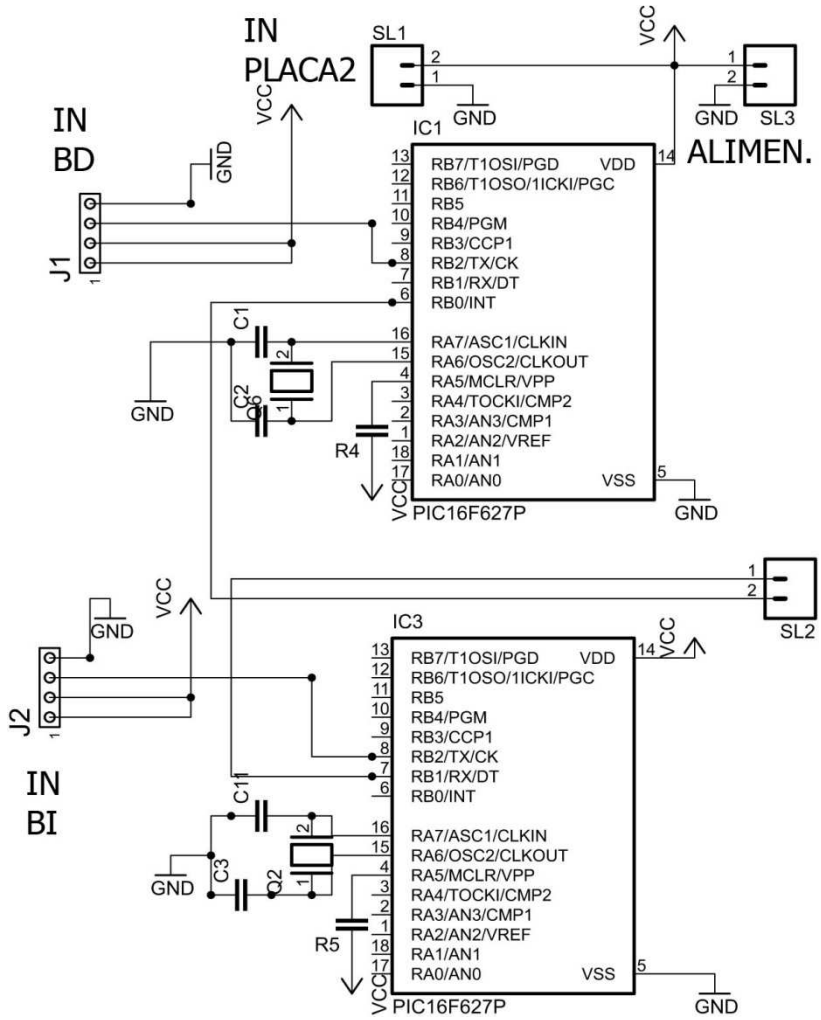


Figura IV.14. Esquema de pics extremidades superiores etapa 2

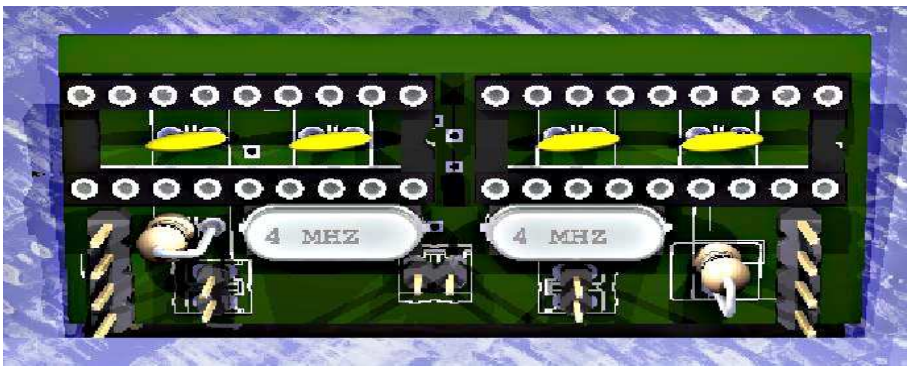


Figura IV.15. Distribución de los pics etapa 2

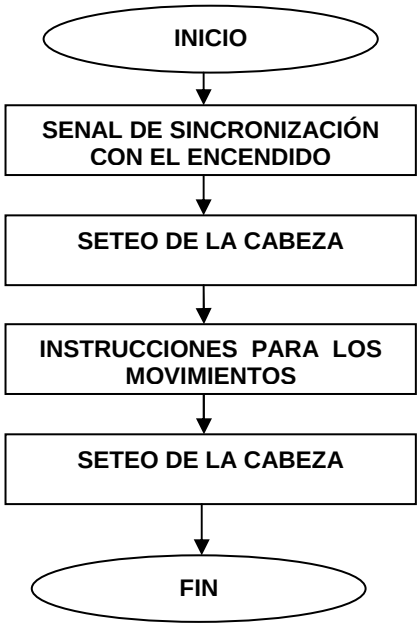


Figura IV.16. Diagrama de flujo del PIC cabeza

Figura IV.17. Diagrama de flujo de los PICs extremidades

CAPÍTULO V

IMPLEMENTACIÓN DEL MICROROBOT.

5.1. CONSTRUCCIÓN DEL ROBOT

Para la construcción del microrobot se ha planificado de la siguiente manera:

- Construcción de las extremidades Inferiores.
- Construcción de las extremidades Superiores.
- Construcción del Cuerpo

Extremidades inferiores

Construcción del pie

1. Se cortó un pedazo de madera de balsa de 2 cm grosor siguiendo las medidas del diseño como se muestra en las figuras V.1 y V.2.
2. Se moldeó la balsa con la ayuda de un estilete ver figuras V.3. y V.4.
3. Por último se procedió a lijarla ver figura V.5. para luego pintarla

Figura V.1. Midiendo y señalando para cortar

Figura V.2. Cortando la balsa

Figura V.3. Moldeo de la balsa

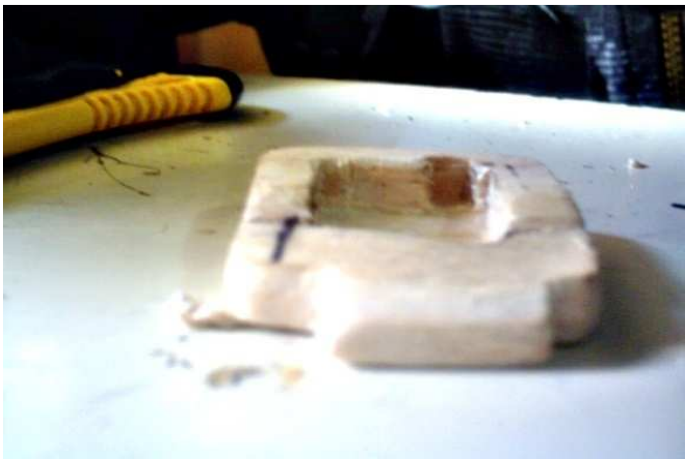


Figura V.4. Pie moldeado en balsa

Figura V.5. Lijando el pie par luego pintarlo

4. En las figura V.6 se muestra el pie terminado.

Figura V.6. Pie terminado

Armado de las piernas

En la figura V.7 se muestra los micromotores, el pie y las diferentes piezas para armar la pierna.

Figura V.7. Partes que conforman la pierna

1. Se sujetó el eje del quinto micromotor (micromotor para el pie) al soporte (articulación) para el siguiente micromotor como se muestra en la figura V.8.

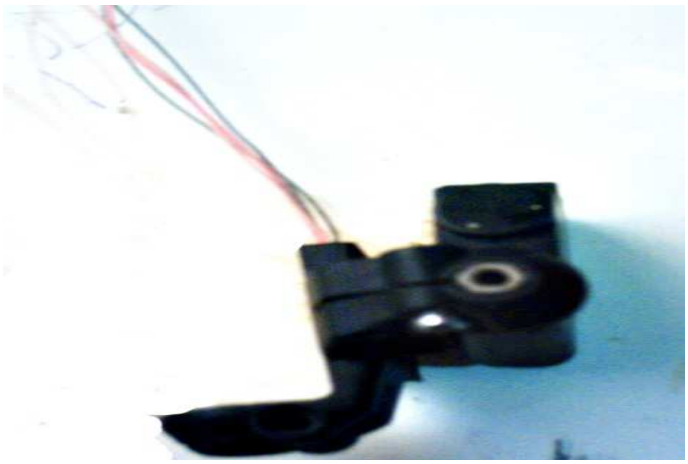


Figura V.8. Colocación del micromotor para el pie

2. Se montó la rodilla la cual esta compuesta por 2 micromotores (cuarto y tercer micromotor) como se muestra en las figuras V.9 a V.11

Figura V.9. Micromotores que conforman la rodilla

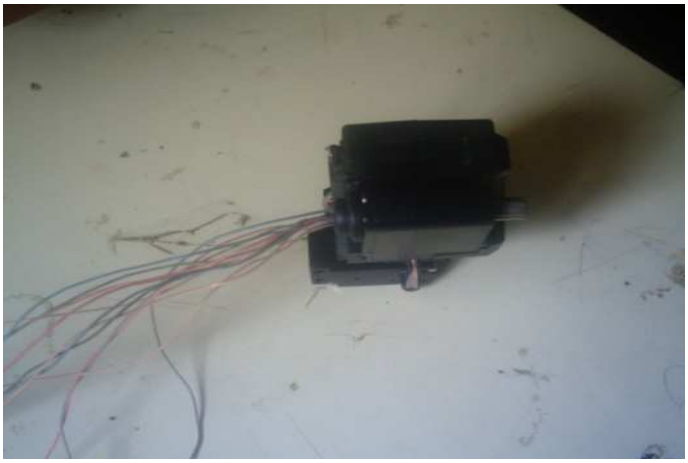


Figura V.10. Rodilla armada

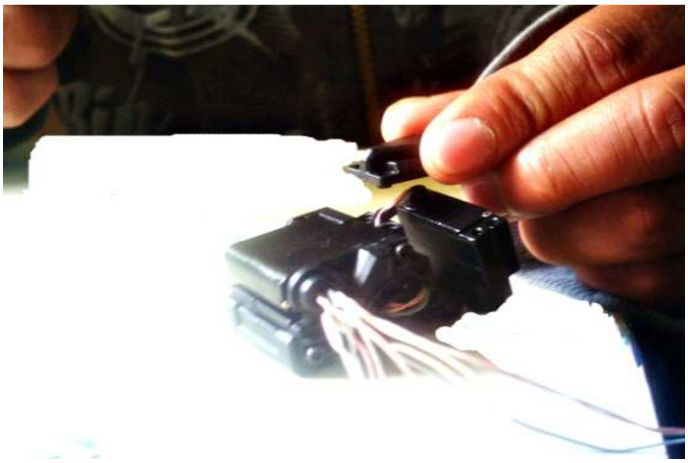


Figura V.11. Unión de la rodilla con el micromotor del pie

3. A continuación se sujetó el siguiente micromotor (segundo micromotor) el cual forma la articulación que une la rodilla con la siguiente pieza, como se muestra en las figuras V.12.

Figura V.12. Unión de la rodilla con el siguiente micromotor (cadera)

4. Finalmente se ubicó la última articulación, la cual une el cuerpo del microrobot con el resto de la pierna, para ello se unió el primer micromotor, todo esto se muestra en las figuras V.13, en la figura V.14. y V.15. se muestra la pierna terminada

Figura V.13 Colocación del micromotor de la cadera el cual se inserta en el cuerpo

Figura V.14. Pierna sin el pie

Figura V.15 Pierna terminada

Extremidades Superiores

Construcción de la mano

Para la construcción de la mano se procedió de igual forma como se hizo con el pie. Se cortó una pieza de madera de balsa de un grosor de 2cm siguiendo las medidas del diseño, luego se procedió a tallarla para finalmente pintarla ver figura V.16.

.

Figura V.16. Mano terminada

Armado del brazo

En la figura V.17. se muestra los micromotores y las diferentes piezas para armar el brazo.

Figura V.17. Partes que conforman el brazo

1. Primero se unió la tercera articulación de cada brazo, es decir, la que forma la mano y el antebrazo e incorpora el tercer micromotor. Ver figuras V.18.a V.19.

Figura V.18 Colocación del micromotor para la mano

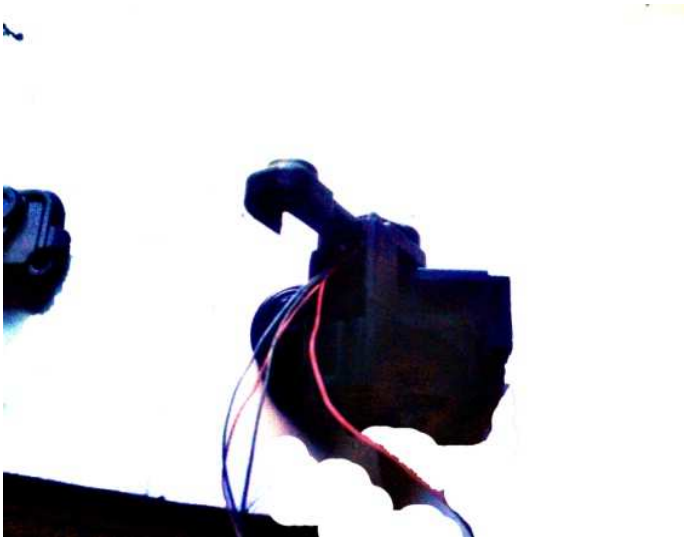


Figura V.19. Micromotor colocado correctamente

2. Se ubicó la segunda articulación la que forma el brazo y el hombro la misma que incorpora al segundo micromotor como se ve en las figuras V.20. y V.21.

Figura V.20. Ubicación del segundo micromotor

Figura V.21. Micromotores formando el brazo

3. Se colocó el primer micromotor el mismo que irá en el cuerpo del microrobot y forma la primera articulación la cual permite el movimiento del brazo ver figura V.22.

4. Por último se procedió a unir la mano al brazo ver figura V.23 con lo cual queda terminado el brazo.

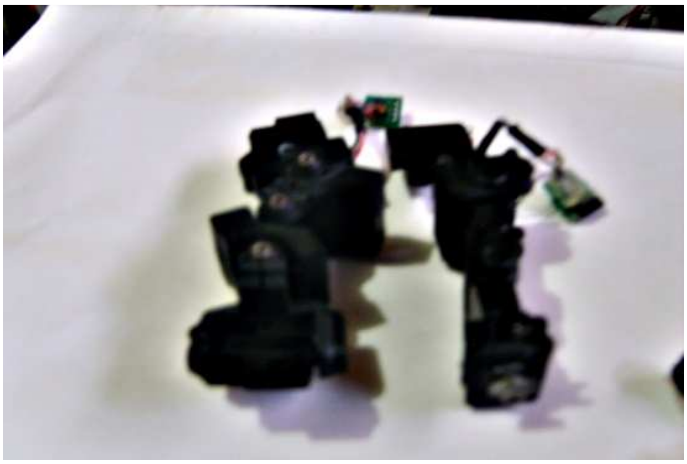


Figura V.22 Colocación del micromotor que une el cuerpo con el brazo

Figura V.23. Brazo terminado

Construcción del cuerpo del microrobot

1. Se cortó un pedazo de lámina de acrílico de 2mm de grosor de acuerdo a las medidas de diseño ver figura V.24.

Figura V.24 Corte de lamina de acrílico

2. A continuación por medio de calor se procedió a dar la forma del armazón como también de las cubiertas como se ve en las figuras V.25. y V.26.

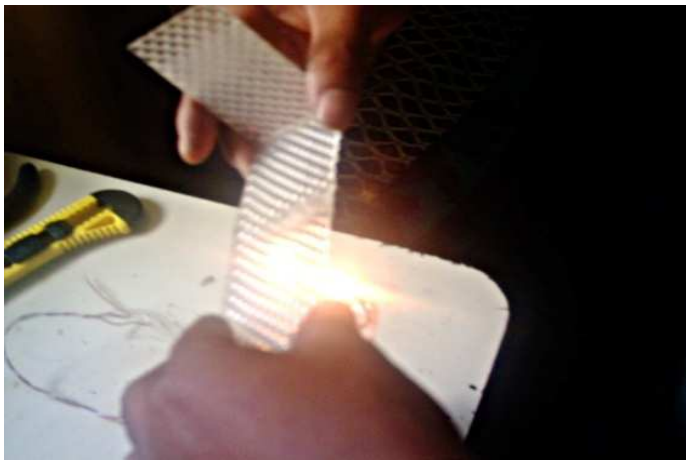


Figura V.25. Calentando el acrílico para darle forma

3. Por último se procedió a la unión de las diferentes partes con la ayuda de silicona como se muestra en las figuras V.27.

   En la figura V.28 y V.29. se muestra todas las partes del cuerpo ya terminadas.

Figura V.26 Formación de una parte del cuerpo.

Figura V.27. Unión del cuerpo con silicona

Figura V.28. Parte trasera del cuerpo

Figura V.29. Piezas que forman el cuerpo

Implementación Electrónica

Ver anexo F

CAPÍTULO VI

PRUEBAS Y ANÁLISIS.

6.1. FUNCIONAMIENTO DEL MICROSERVO

Durante las pruebas realizadas al micromotor, se armó en el protoboarb un circuito con un microcontrolador PIC 16f628A por medio del cual se envió las tramas correspondientes al primer motor a través del puerto B.2 que es el transmisor de la comunicación usart (asíncrona) del Pic con una velocidad de 2400 bps, con lo que se comprobó que su eje se desplaza de acuerdo a la trama enviada.

En la figura VI.1 podemos observar el circuito para verificar el funcionamiento del micromotor servo.

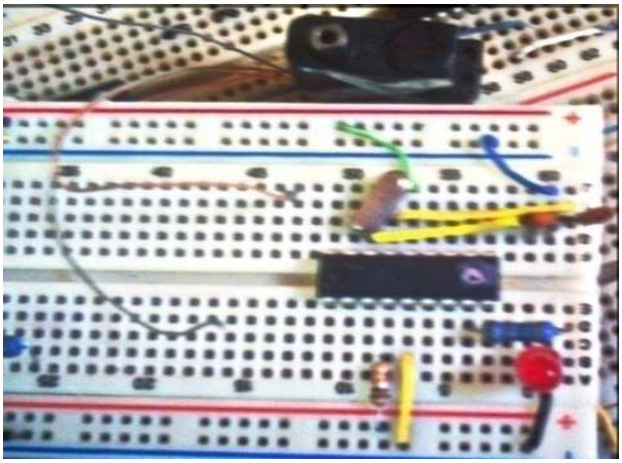


Figura VI.1. Pruebas del micromotor

Una vez que todos los circuitos y sus respectivas placas han sido conectados, encendemos la fuente de alimentación. Se visualiza el diodo LED color verde que empieza a parpadear. Este parpadeo indica que todos los circuitos se encuentran correctamente polarizados. A continuación se detalla las pruebas realizadas en el microrobot. Empezaremos con el seteo de los motores de cada extremidad, mediante el envío de tramas, cada trama está formada por 8 datos, el primer dato es la cabecera (5), los 5 datos siguientes controlan el movimiento de cada uno de los motores respectivamente con un valor entre 0 y 255, el sexto dato es el checksum que está formada mediante la suma de la trama con un valor no mayor a 255, y el último dato es el fin de trama (255)

6.2. SETEO DE MOTORES

Extremidades superiores.

Para el seteo de cada extremidad superior se lo realiza mediante dos seteos distintos de acuerdo al posicionamiento de cada motor.

La trama de seteo del brazo derecho es la siguiente:

RS[0]=5

RS[1]=220

RS [2]=210

RS [3]=130

RS [4]=0

RS [5]=0

RS [6]=55

RS[7]=255

La trama del brazo izquierdo es la siguiente:

RS[0]=5

RS[1]=30

RS[2]=50

RS[3]=130

RS[4]=0

RS[5]=0

RS[6]=215

RS[7]=255

Los datos 4 y 5 están en cero debido a que cada extremidad superior solo posee 3 micromotores.

En la siguiente figura podemos observar el seteo de la extremidad superior derecha.

Figura VI.2 Seteo de las extremidades superiores

De cualquier posición del micromotor se ubican en las posiciones que se les envía en la trama y permanecen por un tiempo determinado de acuerdo al programa.

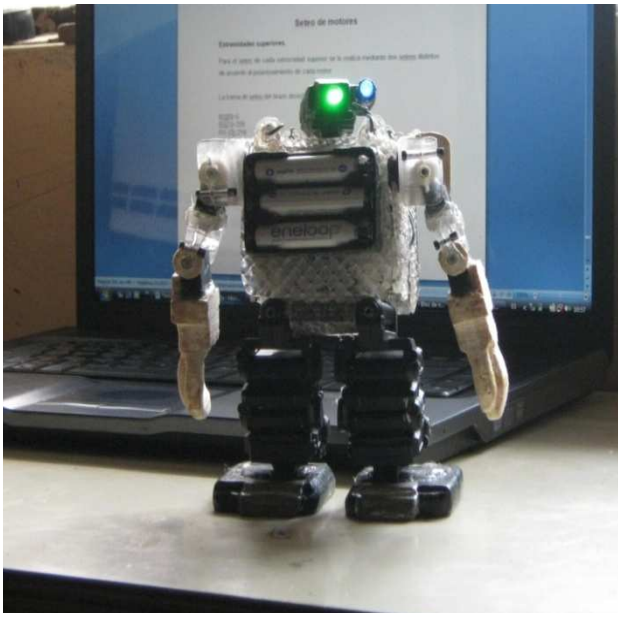


Figura VI.3 Microrobot Seteado

Extremidades inferiores.

Del mismo modo que el seteo de las extremidades superiores es similar a las inferiores solo que cambian los datos

Trama de seteo de la extremidad inferior derecha:

Rs[0]=5

Rs[1]=93

Rs[2]=110

Rs[3]=180

Rs[4]=80

Rs[5]=130

Rs[6]=88

Rs[7]=255

Trama de seteo de la extremidad inferior izquierda

RS[0]=5

RS[1]=125

RS[2]=140

RS[3]=190

RS[4]=190

RS[5]=125

RS[6]=10

RS[7]=255

En las siguientes imágenes se muestra el seteo de las extremidades inferiores, cada micromotor se ubica en la posición de seteo para realizar las instrucciones del programa para que baile.

Figura VI.4 Seteo de las extremidades inferiores

Figura VI.5 Microrobot seteado

6.3. FUENTES DE ALIMENTACIÓN

La alimentación de cada etapa se la realiza mediante tres pilas AAA de 1.5V a 800 mA. En la figura VI.6 se muestra la fuente de alimentación que proveerá de energía a cinco PICs y a diecisiete micromotores, debido a que alimenta a toda a esa cantidad de dispositivos electrónicos el consumo de las pilas es muy rápido.

La duración total de cada ritmo es aproximadamente de dos minutos.

Figura VI.6 Baterías parte frontal

6.4. PRUEBAS DE LAS EXTREMIDADES

Para probar las extremidades tanto superiores e inferiores, se procedió a modificar la posición de cada uno de los micromotores de acuerdo a los datos entre 0 y 255, las tramas se envían en tiempos determinados de acuerdo al ritmo que está bailando; las extremidades inferiores soportan aproximadamente el peso del microrobot (0,35kg).

A continuación se muestra las fotos de algunos movimientos de las extremidades superiores e inferiores combinadas en un solo movimiento.

Figura VI.7 Movimiento brazos y pierna izquierda

Figura VI.8 Brazo derecho extendido al costado

Figura VI.9 Brazo izquierdo extendido al costado

Figura VI.10 Brazos a la cintura

Figura VI.11 Brazos extendidos hacia arriba

Figura VI.12 Movimiento de los brazos hacia arriba y abajo.

Figura VI.13 Posturas de las extremidades

6.5. PRUEBA FINAL DE LOS RITMOS (ESTABILIDAD).

Para la estabilidad del microrobot se la hizo mediante compensación de pesos, mientras levanta las extremidades inferiores se lo equilibra con las extremidades superiores.

A continuación se muestra algunos ejemplos de estabilidad:

Figura VI.14 Balanceo lado derecho

Para levantar la pierna izquierda debe extender el brazo izquierdo para compensar el peso, y viceversa (ver fig.VI.15.)

Cuando el microrobot posiciona sus dos primeros micromotores formando un ángulo de 180º con sus piernas, los brazos deben permanecer abiertos y extendidos para evitar que se balancee hacia adelante o atrás (ver fig.VI.16).

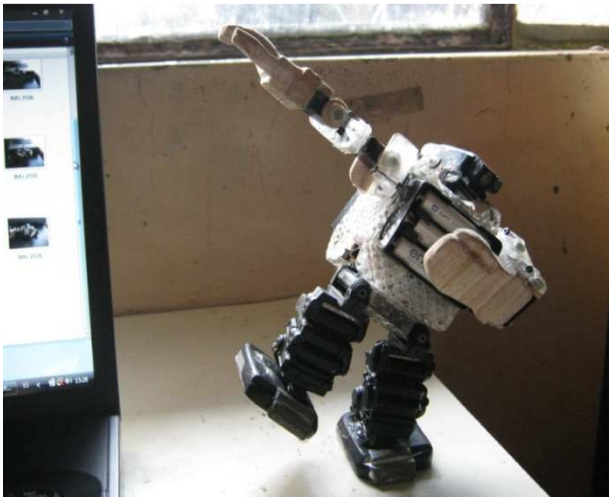


Figura VI.15 Balanceo lado izquierdo

Figura VI.16 Posicionamiento de los micromotores

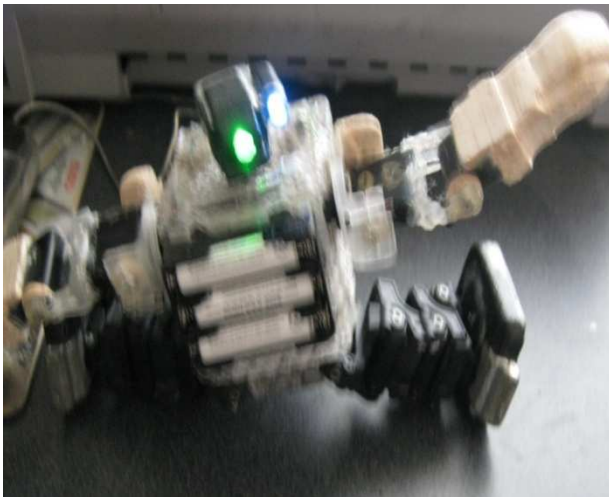


Figura VI.17 Microrobot Bailando

6.6 TIPOS DE MOVIMIENTOS

Los movimientos ejecutados por el microrobot de cada extremidad se realizan de manera independiente debido a que el PIC usado solo posee un solo puerto USART

para el envió de datos, los movimientos se realizan debido al envío de tramas, a continuación se detalla los movimientos más importantes:

Tramas de los brazos:

Seteo:

RS[0]=5
RS[1]=210
RS[2]=210
RS[3]=130
RS[4]=0
RS[5]=0
RS[6]=45
RS[7]=255

Indica que el motor número 1 ,2 y 3 se ubica en la posición 210, esto quiere decir que de cualquier posición que estuviere el motor , se ubica en la posición inicial, lo mismo ocurre con cada uno de los seteos de las extremidades, según el dato que corresponda.

Brazo flexionado hacia adelante:

BA[0]=5'$5 BLADO
BA[1]=140'$60
BA[2]=210'$0
BA[3]=180'$0
BA[4]=0'$0
BA[5]=0'$0
BA[6]=25'$0
BA[7]=255'$FF

Los micromotores 1,2, y 3, se ubican en la posición indicada, para flexionar el brazo hacia adelante

Brazos en movimiento alternado

Para realizar este tipo de movimiento, se lo realiza con dos tramas que simulan el movimiento enviándolas alternadamente por un tiempo determinado.

```
Bc[0]=5'$5   PASITO1 RITMO1      Bc1[0]=5'$5
Bc[1]=160'$60                    Bc1[1]=180'$60
Bc[2]=210'$0                     Bc1[2]=210'$0
Bc[3]=180'$0                     Bc1[3]=180'$0
Bc[4]=0'$0                       Bc1[4]=0'$0
Bc[5]=0'$0                       Bc1[5]=0'$0
Bc[6]=45'$0                      Bc1[6]=65'$0
Bc[7]=255'$FF                    Bc1[7]=255'$FF
```

Enviamos una orden de posición al micromotor 1 y luego en la siguiente trama vuelve a la posición anterior.

Tramas de las piernas:

Seteo de las Piernas:

El Seteo de las piernas se lo realiza en los 5 motores, de tal manera que el robot quede completamente equilibrado.

```
Rs[0]=5'$5       'SETEO
Rs[1]=93'$130
Rs[2]=110'$120
Rs[3]=180'$65
Rs[4]=80'$65
Rs[5]=130'$130
Rs[6]=88'5
Rs[7]=255'$255
```

Levantar la pierna:

Se lo realiza con una trama distinta en cada pierna, por ejemplo la pierna derecha levanta la pierna mientras que la izquierda equilibra el peso.

```
PA[0]=5'$5       P ALZADA        PA[3]=180'$65
PA[1]=168'$130                   PA[4]=80'$65
PA[2]=110'$120                   PA[5]=130'$130
```

```
PA[6]=163'5
PA[7]=255'$255

CI2[0]=5'5      'IZQUIERDA
CI2[1]=0'130
CI2[2]=110'150
CI2[3]=180'65
CI2[4]=80'110
CI2[5]=150'134
CI2[6]=15'84
CI2[7]=255'255
```

Camina hacia adelante

Lo realizamos con 2 tramas distintas, mientras camina con la pierna izquierda, se equilibra con la derecha.

```
CA[0]=5'5      'RODILLAS
CA[1]=93'
CA[2]=140'
CA[3]=10'
CA[4]=95'
CA[5]=0'
CA[6]=88'
CA[7]=255'
```

```
CI2[0]=5'5      'IZQUIERDA
CI2[1]=0'130
CI2[2]=110'150
CI2[3]=180'65
CI2[4]=80'110
CI2[5]=150'134
CI2[6]=15'84
CI2[7]=255'255
```

CONCLUSIONES.

1. El proyecto se implementó con micromotores servo Kanakatta Hiji los cuales son muy pequeños pero nos oferta grandes cualidades como por ejemplo su gran torque ya que al poseer un sistema de reducción excelente nos facilita la operabilidad y manejo de los mismos, lo que fue determinante para el desarrollo de este proyecto.
2. Al manejar cinco microcontroladores para el control del microrobot, para solucionar la sincronización se ha utilizado el encendido general del microrobot, al encender el microrobot este envía una señal a los microcontroladores para setear los micromotores y empezar a bailar los diferentes ritmos programados.
3. Se escogió materiales livianos pero a la vez muy resistentes, aunque los micromotores poseen un gran torque no son capaces de soportar demasiado peso y al utilizar un material más pesado esto influiría directamente en el equilibrio del microrobot y por lo tanto en el movimiento del mismo.
4. Para que un microrobot bípedo se encuentre dinámicamente estable, el FRI (Indicador de rotación del pie) debe estar siempre dentro del polígono de soporte. La posición del FRI es una consecuencia directa del estado dinámico del microrobot.
5. Durante las investigaciones y pruebas iniciales se concluyó: cuanto más grados de libertad existe en el microrobot este puede realizar pasos más elegantes y tiene mayor movilidad en sus extremidades por lo tanto se implementó con diecisiete grados de libertad.

6. Durante la programación de los ritmos se determinó que los movimientos de todas sus extremidades deben estar sincronizadas para el microrobot no pierda a su estabilidad y los movimientos estén acordes al ritmo.

RECOMENDACIONES

1. Los micromotores no deben ser forzados o mal manipulados porque esto causaría el aislamiento de los dientes de los engranajes , como también se puede romper los cables de alimentación o de control o puede quemarse el microcontrolador interno del micromotor.
2. Se recomienda a los usuarios del microrobot estudiar el protocolo de comunicación de los micromotores Kanakatta Hiji para entender el funcionamiento.
3. En el momento de la búsqueda de nuevas tramas para la programación de nuevos pasos para el baile se debe tener sumo cuidado ya que una manipulación incorrecta podría dañar los micromotores.
4. Si se desea que el microrobot baile por más tiempo se debe añadir otras tres pilas ya que el consumo es excesivo por alimentar a varios dispositivos electrónicos y además deberán equilibrar el peso para que el microrobot no pierda la estabilidad.
5. En la programación de los micromotores tener cuidado en las instrucciones utilizadas, se recomienda utilizar instrucciones propias del microcontrolador para la comunicación USART ya que caso contrario los micromotores se setearan por la introducción de ruido.
6. Aplicar los niveles de voltaje apropiados a cada una de las etapas del proyecto, como también a los micromotores con su respectiva y correcta polaridad, de lo contrario estos pueden sufrir daños en su funcionamiento o quemarse.

RESUMEN.

Utilizando micromotores servos y microchips se diseñó y construyó un microrobot bípedo bailarín buscando disminuir su tamaño, peso y aumentando su portabilidad y manipulación con miras a participar en concursos de robótica en representación de la Escuela de Ingeniería Electrónica de la Escuela Superior Politécnica de Chimborazo.

Fue construido en plástico, balsa y acrílico, para el control se utilizó microcontroladores PIC16f628A uno por cada extremidad y uno para la cabeza, trabaja con 6 baterías de 1.5 voltios cada una. En el diseño se aplicó el método inductivo hasta encontrar la solución óptima mediante la aplicación de técnicas de investigación, análisis y aplicación.

El microrobot mide 17 cm de altura, 14 cm de ancho con 17 grados de libertad; el baile lo realiza mediante un programa ejecutado en el PIC con sentencias repetitivas; baila tres tipos de ritmos: música nacional, desplazándose hacia la derecha e izquierda (3 cm) durante dos minutos, electrónica, desplazándose 8 cm durante dos minutos y pop donde levanta brazos y piernas durante dos minutos, para mantener la estabilidad se compensó su peso, así: mientras levanta la extremidad inferior derecha, se equilibra levantando la extremidad superior derecha, el mismo proceso realiza para las extremidades izquierdas.

El micromotor servo Kanakatta Hiji al ser muy pequeño, permitió disminuir considerablemente el peso y tamaño con respecto a robots existentes en el mercado.

Se recomienda a los usuarios del microrobot, estudiar el protocolo de comunicación para entender su funcionamiento.

SUMMARY

A biped-dancing micro robot has been designed and built by using micro servomotors and microchips in order to reduce its height, weight and increasing its carrying and manipulation so that it can participate in contest of robotic representing to Electronic engineering School of Escuela Superior Politécnica de Chimborazo.

It was made of plastic, species of ceiba and acrilyc. Micro controllers PIC16f628A one per each extremity was used to control it. One microcontroller in the head works with 6 batteries of 1.5 watts each one. The inductive method was used to find the best solution by means of the application of techniques of application, analysis and application.

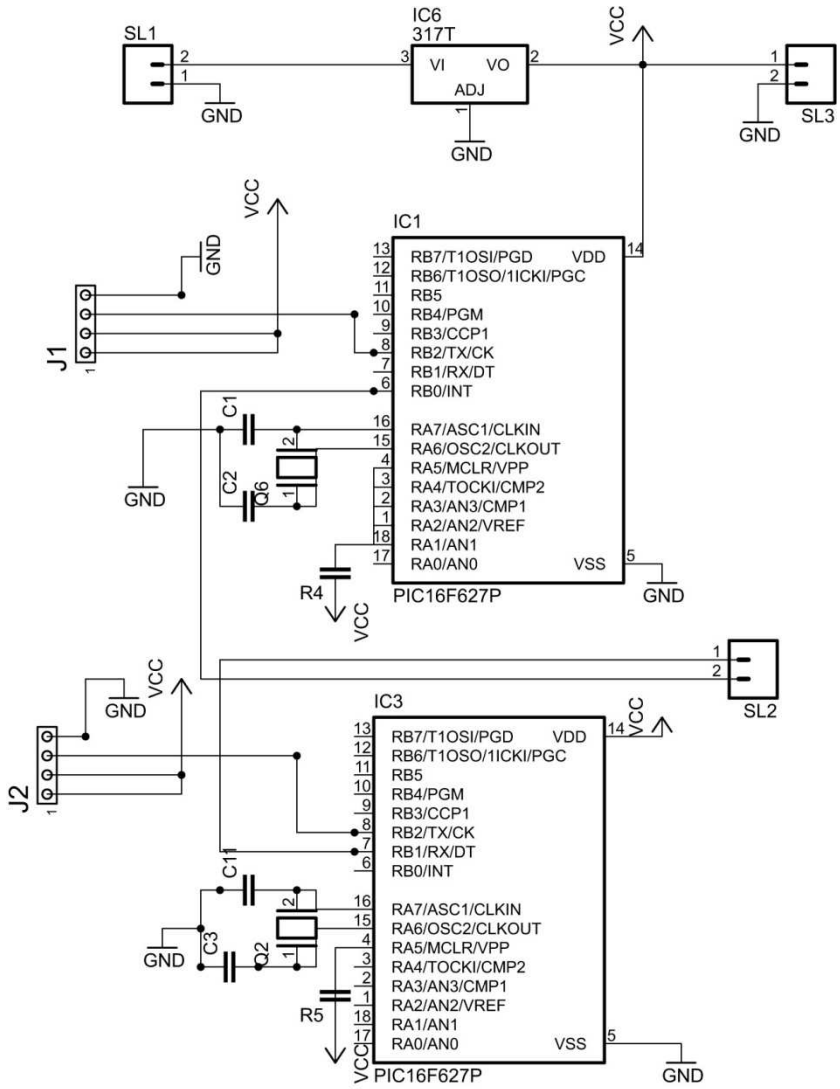
The mentioned micro robot measures 17 cm of height, 14 cm of width 17 grades of freedom. Dancing is carried out by means of a program executed in the PIC with repetitive statements; it dances three kinds of rhythm: slow salsa moving to the right and left (3cm) during 1 minute, regaeton, moving 8cm during 2 minutes and pop rising its hands and legs during 2 minutes. Its weight was equilibrated to maintain the stability so that, while it lifts the right lower extremity, it equilibrates by lifting the upper extremity ; the same process is made for left extremities.

The microservomotor KanaKatta Hiji because its tiny size allows reducing the weight and size regarding the robots existing in the market so it is recommended that the users of the micro robot study the document of communication to understand its running.

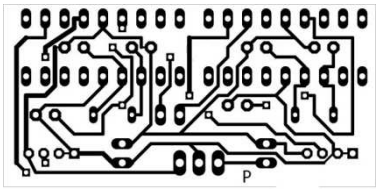
ANEXOS

Anexo A. Diagrama y circuito impreso para el control de la pierna

- Diagrama del circuito

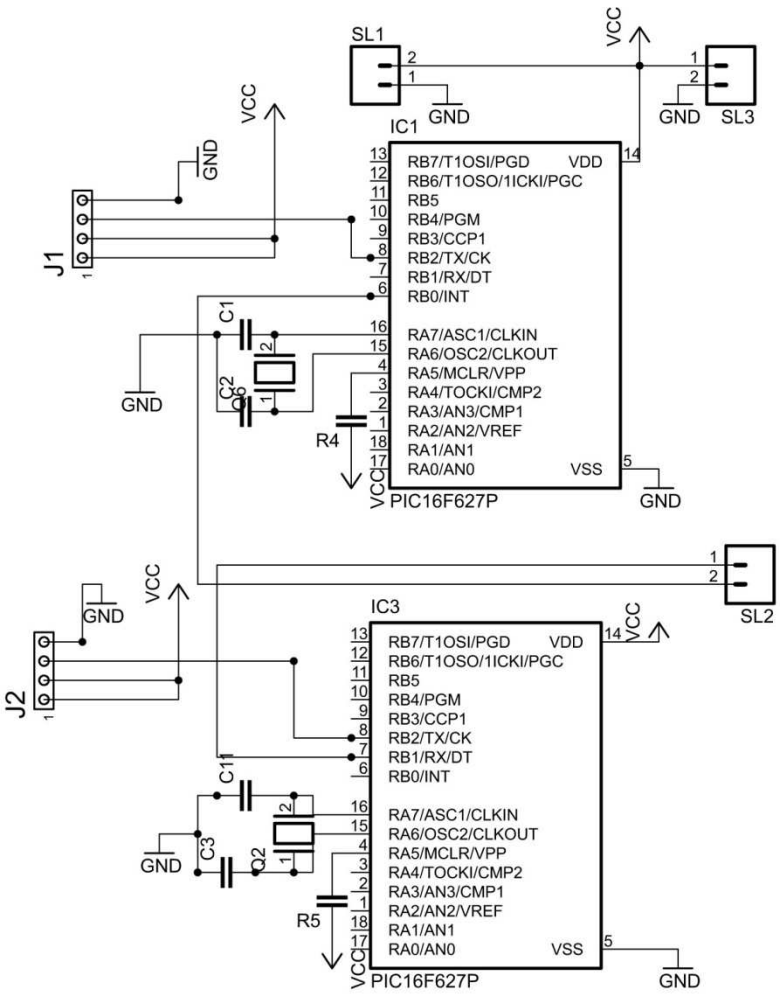


- Circuito impreso

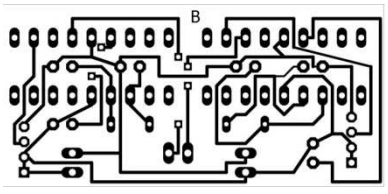


Anexo B. Diagrama y circuito impreso para el control de los brazos

- Diagrama del circuito

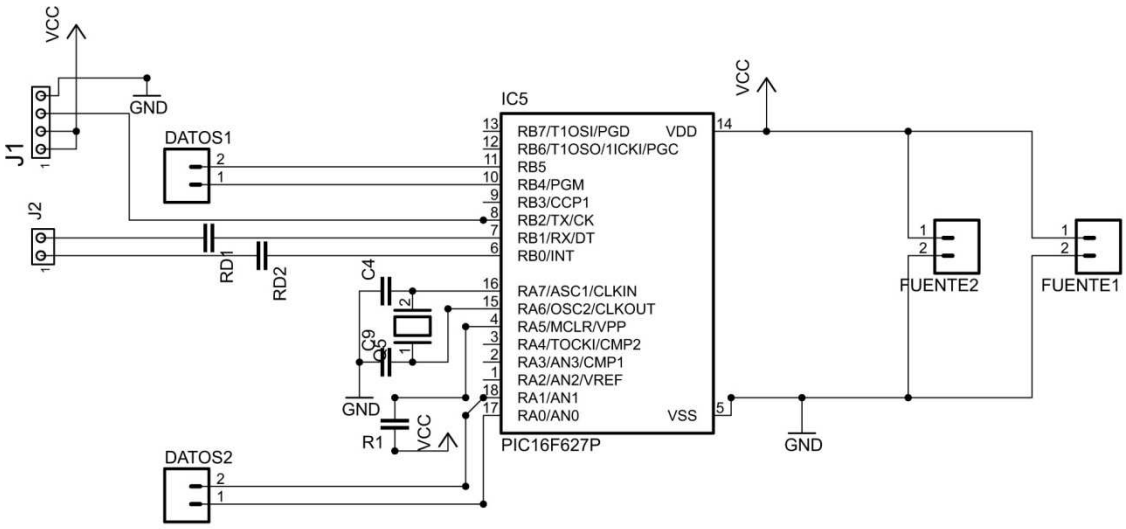


- Circuito Impreso

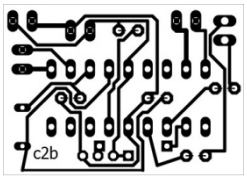


Anexo C. Diagrama y circuito impreso para el control de la cabeza

- Diagrama del circuito



- Circuito Impreso



Anexo D. Datasheet del PIC 16f628A

PIC16F627A/628A/648A

Data Sheet

FLASH-Based

8-Bit CMOS Microcontrollers

PIC16F627A/628A/648A

18-pin FLASH-Based 8-Bit CMOS Microcontrollers

High Performance RISC CPU:

- Operating speeds from DC - 20 MHz
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes
- 35 single word instructions
  - All instructions single cycle except branches

Special Microcontroller Features:

- Internal and external oscillator options
  - Precision Internal 4 MHz oscillator factory calibrated to ±1%
  - Low Power Internal 37 kHz oscillator
  - External Oscillator support for crystals and resonators.
- Power saving SLEEP mode
- Programmable weak pull-ups on PORTB
- Multiplexed Master Clear/Input-pin
- Watchdog Timer with independent oscillator for reliable operation
- Low voltage programming
- In-Circuit Serial Programming™ (via two pins)
- Programmable code protection
- Brown-out Reset
- Power-on Reset
- Power-up Timer and Oscillator Start-up Timer
- Wide operating voltage range. (2.0 - 5.5V)
- Industrial and extended temperature range
- High Endurance FLASH/EEPROM Cell
  - 100,000 write FLASH endurance
  - 1,000,000 write EEPROM endurance
  - 100 year data retention

Low Power Features:

- Standby Current:
  - 100 nA @ 2.0V, typical
- Operating Current:
  - 12 μA @ 32 kHz, 2.0V, typical
  - 120 μA @ 1 MHz, 2.0V, typical
- Watchdog Timer Current
  - 1 μA @ 2.0V, typical
- Timer1 oscillator current:
  - 1.2 μA @ 32 kHz, 2.0V, typical
- Dual Speed Internal Oscillator:
  - Run-time selectable between 4 MHz and 37 kHz
  - 4 μs wake-up from SLEEP, 3.0V, typical

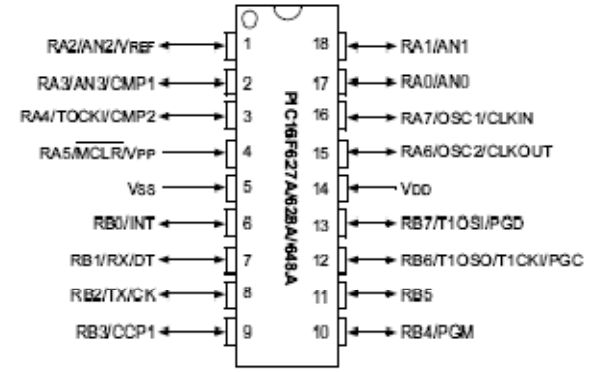
Peripheral Features:

- 16 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Selectable internal or external reference
  - Comparator outputs are externally accessible
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Timer1: 16-bit timer/counter with external crystal/clock capability
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Capture, Compare, PWM module
  - 16-bit Capture/Compare
  - 10-bit PWM
- Addressable Universal Synchronous/Asynchronous Receiver/Transmitter USART/SCI

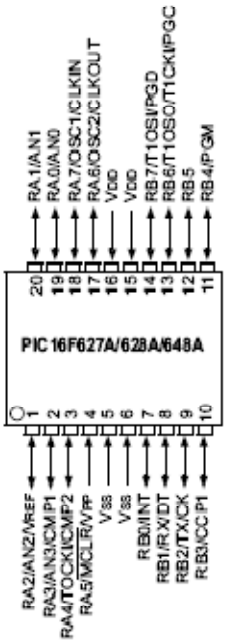
Device	Program Memory	Data Memory		I/O	CCP (PWM)	USART	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)					
PIC16F627A	1024	224	128	16	1	Y	2	2/1
PIC16F628A	2048	224	128	16	1	Y	2	2/1
PIC16F648A	4096	256	256	16	1	Y	2	2/1

Pin Diagrams

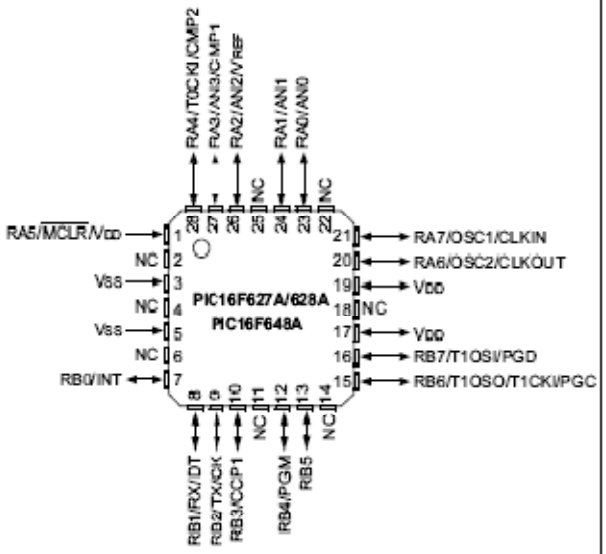
PDIP, SOIC



SSOP



28-Pin QFN



1.0 GENERAL DESCRIPTION

The PIC16F627A/628A/648A are 18-Pin FLASH-based members of the versatile PIC16CXX family of low cost, high performance, CMOS, fully-static, 8-bit microcontrollers.

All PICmicro® microcontrollers employ an advanced RISC architecture. The PIC16F627A/628A/648A have enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data. The two-stage instruction pipeline allows all instructions to execute in a single-cycle, except for program branches (which require two cycles). A total of 35 instructions (reduced instruction set) are available, complemented by a large register set.

PIC16F627A/628A/648A microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in their class.

PIC16F627A/628A/648A devices have integrated features to reduce external components, thus reducing system cost, enhancing system reliability and reducing power consumption.

The PIC16F627A/628A/648A has 8 oscillator configurations. The single-pin RC oscillator provides a low cost solution. The LP oscillator minimizes power consumption, XT is a standard crystal, and INTOSC is a self-contained precision two-speed internal oscillator. The HS is for High-Speed crystals. The EC mode is for an external clock source.

The SLEEP (Power-down) mode offers power savings. Users can wake-up the chip from SLEEP through several external interrupts, internal interrupts and RESETS.

A highly reliable Watchdog Timer with its own on-chip RC oscillator provides protection against software lock-up.

Table 1-1 shows the features of the PIC16F627A/628A/648A mid-range microcontroller families.

A simplified block diagram of the PIC16F627A/628A/648A is shown in Figure 3-1.

The PIC16F627A/628A/648A series fits in applications ranging from battery chargers to low power remote sensors. The FLASH technology makes customizing application programs (detection levels, pulse generation, timers, etc.) extremely fast and convenient. The small footprint packages makes this microcontroller series ideal for all applications with space limitations. Low cost, low power, high performance, ease of use and I/O flexibility make the PIC16F627A/628A/648A very versatile.

1.1 Development Support

The PIC16F627A/628A/648A family is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a low cost in-circuit debugger, a low cost development programmer and a full-featured programmer. A Third Party "C" compiler support tool is also available.

TABLE 1-1: PIC16F627A/628A/648A FAMILY OF DEVICES

		PIC16F627A	PIC16F628A	PIC16F648A	PIC16LF627A	PIC16LF628A	PIC16LF648A
Clock	Maximum Frequency of Operation (MHz)	20	20	20	4	4	4
Memory	FLASH Program Memory (words)	1024	2048	4096	1024	2048	4096
	RAM Data Memory (bytes)	224	224	256	224	224	256
	EEPROM Data Memory (bytes)	128	128	256	128	128	256
Peripherals	Timer module(s)	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2
	Comparator(s)	2	2	2	2	2	2
	Capture/Compare/PWM modules	1	1	1	1	1	1
	Serial Communications	USART	USART	USART	USART	USART	USART
	Internal Voltage Reference	Yes	Yes	Yes	Yes	Yes	Yes
Features	Interrupt Sources	10	10	10	10	10	10
	I/O Pins	16	16	16	16	16	16
	Voltage Range (Volts)	3.0-5.5	3.0-5.5	3.0-5.5	2.0-5.5	2.0-5.5	2.0-5.5
	Brown-out Reset	Yes	Yes	Yes	Yes	Yes	Yes
	Packages	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN

All PICmicro® Family devices have Power-on Reset, selectable Watchdog Timer, selectable Code Protect and high I/O current capability.
All PIC16F627A/628A/648A Family devices use serial programming with clock pin RB6 and data pin RB7.

2.0 PIC16F627A/628A/648A DEVICE VARIETIES

A variety of frequency ranges and packaging options are available. Depending on application and production requirements, the proper device option can be selected using the information in the PIC16F627A/628A/648A Product Identification System, at the end of this data sheet. When placing orders, please use this page of the data sheet to specify the correct part number.

2.1 FLASH Devices

FLASH devices can be erased and re-programmed electrically. This allows the same device to be used for prototype development, pilot programs and production.

A further advantage of the electrically erasable FLASH is that it can be erased and reprogrammed in-circuit, or by device programmers, such as Microchip's PICSTART® Plus, or PRO MATE® II programmers.

2.2 Quick-Turnaround-Production (QTP) Devices

Microchip offers a QTP Programming Service for factory production orders. This service is made available for users who chose not to program a medium to high quantity of units and whose code patterns have stabilized. The devices are standard FLASH devices but with all program locations and configuration options already programmed by the factory. Certain code and prototype verification procedures apply before production shipments are available. Please contact your Microchip Technology sales office for more details.

2.3 Serialized Quick-Turnaround-Production (SQTP℠) Devices

Microchip offers a unique programming service where a few user-defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number, which can serve as an entry-code, password or ID number.

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16F627A/628A/648A family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16F627A/628A/648A uses a Harvard architecture, in which program and data are accessed from separate memories using separate busses. This improves bandwidth over traditional von Neumann architecture where program and data are fetched from the same memory. Separating program and data memory further allows instructions to be sized differently than 8-bit wide data word. Instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions (35) execute in a single-cycle (200 ns @ 20 MHz) except for program branches.

Table 3-1 lists device memory sizes (FLASH, Data and EEPROM).

TABLE 3-1: DEVICE MEMORY LIST

Device	Memory		
	FLASH Program	RAM Data	EEPROM Data
PIC16F627A	1024 x 14	224 x 8	128 x 8
PIC16F628A	2048 x 14	224 x 8	128 x 8
PIC16F648A	4096 x 14	256 x 8	256 x 8
PIC16LF627A	1024 x 14	224 x 8	128 x 8
PIC16LF628A	2048 x 14	224 x 8	128 x 8
PIC16LF648A	4096 x 14	256 x 8	256 x 8

The PIC16F627A/628A/648A can directly or indirectly address its register files or data memory. All Special Function Registers, including the program counter, are mapped in the data memory. The PIC16F627A/628A/648A have an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation, on any register, using any Addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC16F627A/628A/648A simple yet efficient. In addition, the learning curve is reduced significantly.

The PIC16F627A/628A/648A devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is 8-bit wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the working register (W register). The other operand is a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

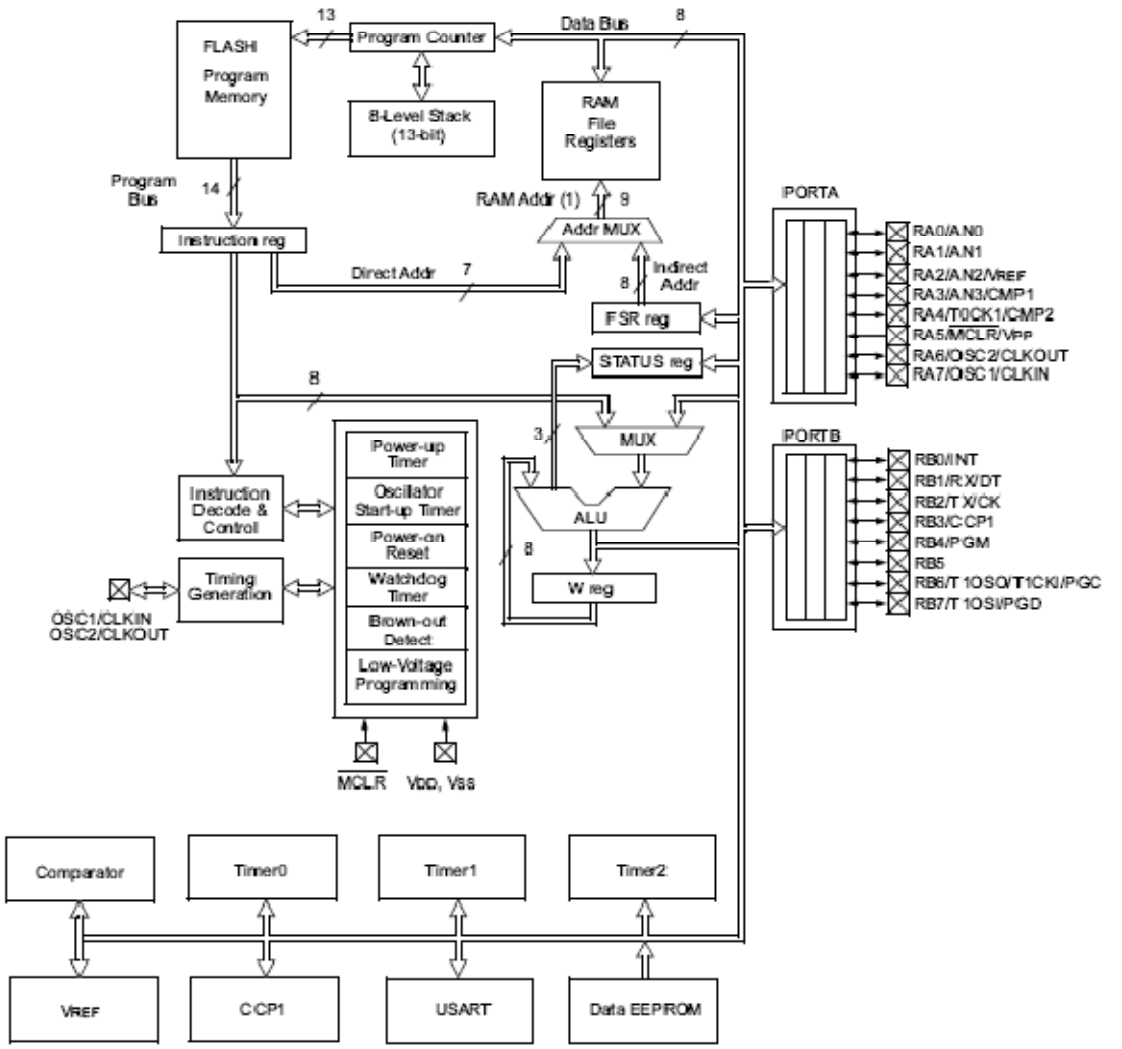
The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), and Zero (Z) bits in the STATUS register. The C and DC bits operate as a Borrow and Digit Borrow out bit, respectively, bit in subtraction. See the SUBLW and SUBWF instructions for examples.

A simplified block diagram is shown in Figure 3-1, and a description of the device pins in Table 3-2.

Two types of data memory are provided on the PIC16F627A/628A/648A devices. Non-volatile EEPROM data memory is provided for long term storage of data such as calibration values, look up table data, and any other data which may require periodic updating in the field. These data are not lost when power is removed. The other data memory provided is regular RAM data memory. Regular RAM data memory is provided for temporary storage of data during normal operation. Data are lost when power is removed.

FIGURE 3-1: BLOCK DIAGRAM



Note: Higher order bits are from the STATUS register.

TABLE 3-2: PIC16F627A/628A/648A PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bi-directional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bi-directional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bi-directional I/O port
	T0CKI	ST	—	Timer0 clock input
	CMP2	—	OD	Comparator 2 output
RA5/MCLR/VPP	RA5	ST	—	Input port
	$\overline{MCLR}$	ST	—	Master clear. When configured as $\overline{MCLR}$, this pin is an active low RESET to the device. Voltage on MCLR/VPP must not exceed VDD during normal device operation.
	VPP	—	—	Programming voltage input.
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bi-directional I/O port
	OSC2	—	XTAL	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In RC/INTOSC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bi-directional I/O port
	OSC1	XTAL	—	Oscillator crystal input
	CLKIN	ST	—	External clock source input. RC biasing pin.
RB0/INT	RB0	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	INT	ST	—	External interrupt.
RB1/RX/DT	RB1	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	RX	ST	—	USART receive pin
	DT	ST	CMOS	Synchronous data I/O.
RB2/TX/CK	RB2	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	TX	—	CMOS	USART transmit pin
	CK	ST	CMOS	Synchronous clock I/O.
RB3/CCP1	RB3	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	CCP1	ST	CMOS	Capture/Compare/PWM I/O

Legend: O = Output — = Not used TTL = TTL Input
CMOS = CMOS Output I = Input OD = Open Drain Output
P = Power ST = Schmitt Trigger Input AN = Analog

TABLE 3-2: PIC16F627A/628A/648A PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RB4/PGM	RB4	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	PGM	ST	—	Low voltage programming input pin. When low voltage programming is enabled, the interrupt-on-pin change and weak pull-up resistor are disabled.
RB5	RB5	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
RB6/T1OSO/T1CKI/PGC	RB6	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSO	—	XTAL	Timer1 oscillator output.
	T1CKI	ST	—	Timer1 clock input.
	PGC	ST	—	ICSP Programming Clock.
RB7/T1OSI/PGD	RB7	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSI	XTAL	—	Timer1 oscillator input.
	PGD	ST	CMOS	ICSP Data I/O
VSS	VSS	Power	—	Ground reference for logic and I/O pins
VDD	VDD	Power	—	Positive supply for logic and I/O pins

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

3.1 Clocking Scheme/Instruction Cycle

The clock input (OSC1/CLKIN/RA7 pin) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2.

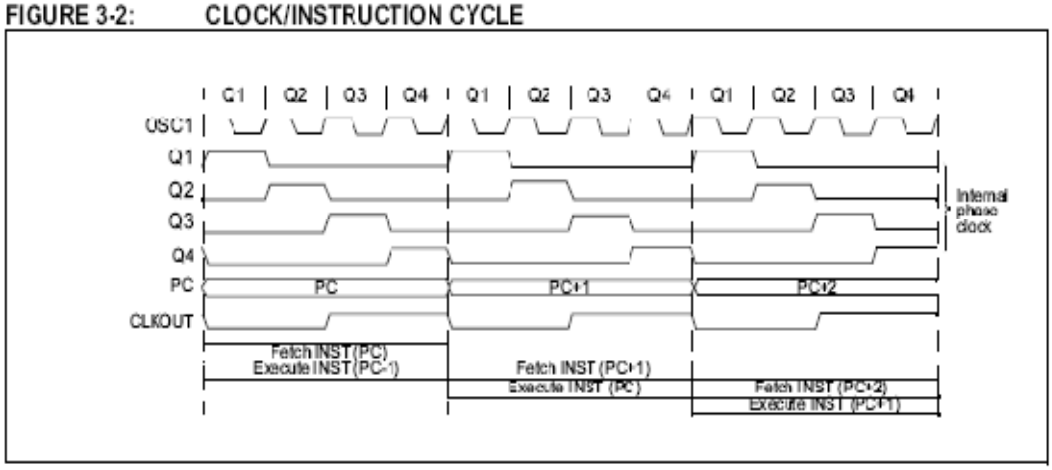
3.2 Instruction Flow/Pipelining

An instruction cycle consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO) then two cycles are required to complete the instruction (Example 3-1).

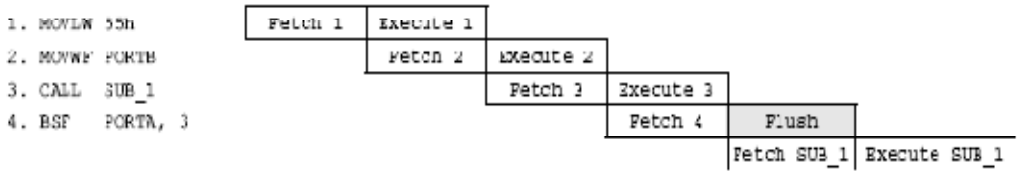
A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 3-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW



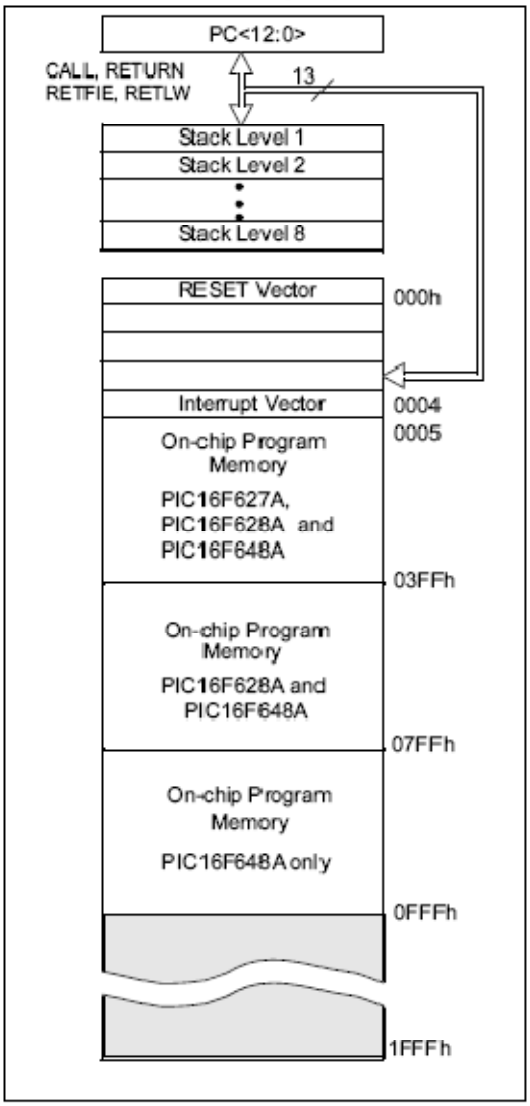
All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

4.0 MEMORY ORGANIZATION

4.1 Program Memory Organization

The PIC16F627A/628A/648A has a 13-bit program counter capable of addressing an 8K x 14 program memory space. Only the first 1K x 14 (0000h - 03FFh) for the PIC16F627A, 2K x 14 (0000h - 07FFh) for the PIC16F628A and 4K x 14 (0000h - 0FFFh) for the PIC16F648A are physically implemented. Accessing a location above these boundaries will cause a wrap-around within the first 1K x 14 space (PIC16F627A), 2K x 14 space (PIC16F628A) or 4K x 14 space (PIC16F648A). The RESET vector is at 0000h and the interrupt vector is at 0004h (Figure 4-1).

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK



4.2 Data Memory Organization

The data memory (Figure 4-2 and Figure 4-3) is partitioned into four banks, which contain the general purpose registers and the Special Function Registers (SFR). The SFR's are located in the first 32 locations of each Bank. There are general purpose registers implemented as static RAM in each Bank. Table 4-1 lists the general purpose register available in each of the four banks.

TABLE 4-1: GENERAL PURPOSE STATIC RAM REGISTERS

	PIC16F627A/628A	PIC16F648A
Bank0	20-7Fh	20-7Fh
Bank1	A0h-FF	A0h-FF
Bank2	120h-14Fh, 170h-17Fh	120h-17Fh
Bank3	1F0h-1FFh	1F0h-1FFh

Addresses F0h-FFh, 170h-17Fh and 1F0h-1FFh are implemented as common RAM and mapped back to addresses 70h-7Fh.

Table 4-2 lists how to access the four banks of registers via the STATUS Register bits RP1 and RP0.

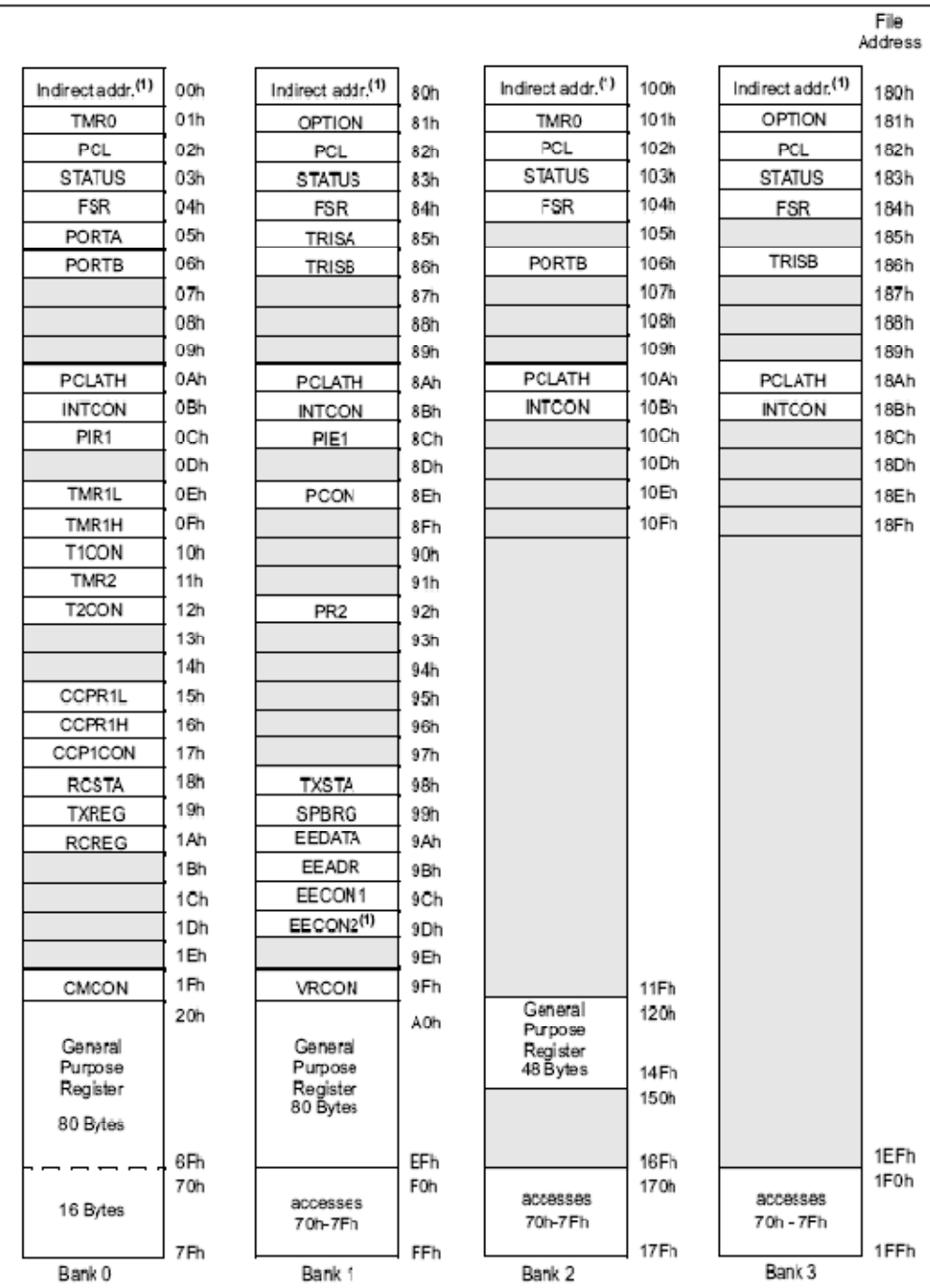
TABLE 4-2: ACCESS TO BANKS OF REGISTERS

	RP1	RP0
Bank0	0	0
Bank1	0	1
Bank2	1	0
Bank3	1	1

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 224 x 8 in the PIC16F627A/628A and 256 x 8 in the PIC16F648A. Each is accessed either directly or indirectly through the File Select Register (FSR), See Section 4.4.

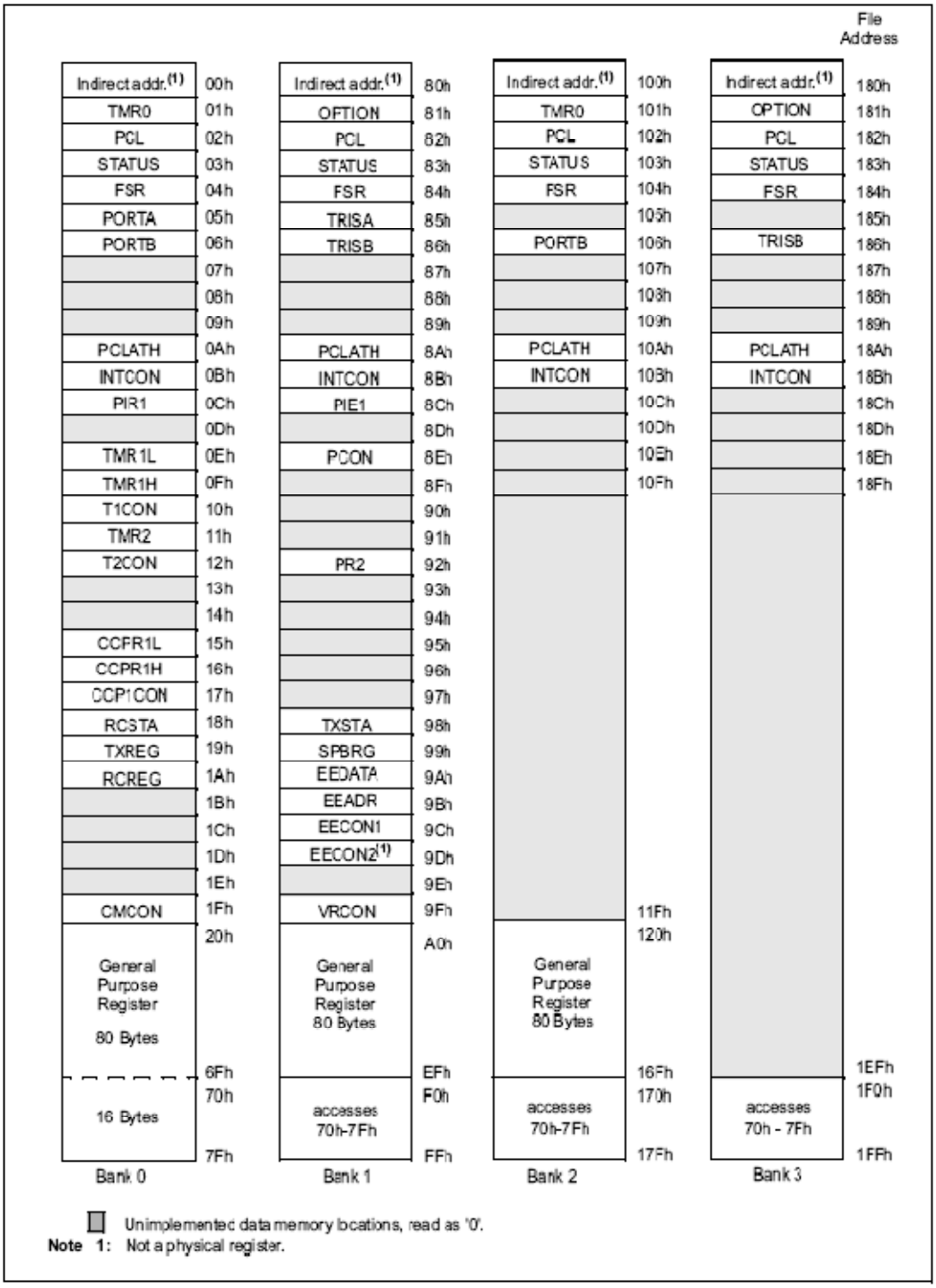
FIGURE 4-2: DATA MEMORY MAP OF THE PIC16F627A AND PIC16F628A



Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

FIGURE 4-3: DATA MEMORY MAP OF THE PIC16F648A



Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

4.2.2 SPECIAL FUNCTION REGISTERS

The SFRs are registers used by the CPU and Peripheral functions for controlling the desired operation of the device (Table 4-3). These registers are static RAM.

The special registers can be classified into two sets (core and peripheral). The SFRs associated with the "core" functions are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

TABLE 4-3: SPECIAL REGISTERS SUMMARY BANK0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset(1)	Details on Page
Bank 0											
00h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	28
01h	TMR0	Timer0 module's Register								xxxx xxxx	45
02h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28
03h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22
04h	FSR	Indirect data memory address pointer								xxxx xxxx	28
05h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx 0000	31
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	36
07h	—	Unimplemented								—	—
08h	—	Unimplemented								—	—
09h	—	Unimplemented								—	—
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter					---0 0000	28
0Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	24
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	0000 -000	26
0Dh	—	Unimplemented								—	—
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1								xxxx xxxx	48
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1								xxxx xxxx	48
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	48
11h	TMR2	TMR2 module's register								0000 0000	52
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	52
13h	—	Unimplemented								—	—
14h	—	Unimplemented								—	—
15h	CCPR1L	Capture/Compare/PWM register (LSB)								xxxx xxxx	55
16h	CCPR1H	Capture/Compare/PWM register (MSB)								xxxx xxxx	55
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	55
18h	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	69
19h	TXREG	USART Transmit data register								0000 0000	76
1Ah	RCREG	USART Receive data register								0000 0000	79
1Bh	—	Unimplemented								—	—
1Ch	—	Unimplemented								—	—
1Dh	—	Unimplemented								—	—
1Eh	—	Unimplemented								—	—
1Fh	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	61

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: For the Initialization Condition for Registers Tables, refer to Table 14-6 and Table 14-7.

TABLE 4-4: SPECIAL FUNCTION REGISTERS SUMMARY BANK1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset(1)	Details on Page
Bank 1											
80h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	28
81h	OPTION	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23
82h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28
83h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22
84h	FSR	Indirect data memory address pointer								xxxx xxxx	28
85h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	31
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	36
87h	—	Unimplemented								—	—
88h	—	Unimplemented								—	—
89h	—	Unimplemented								—	—
8Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter					---0 0000	28
8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	24
8Ch	PIE1	EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	0000 -000	25
8Dh	—	Unimplemented								—	—
8Eh	PCON	—	—	—	—	OSCF	—	$\overline{POR}$	$\overline{BOR}$	---- 1-0x	27
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	—	Unimplemented								—	—
92h	PR2	Timer2 Period Register								1111 1111	52
93h	—	Unimplemented								—	—
94h	—	Unimplemented								—	—
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	71
99h	SPBRG	Baud Rate Generator Register								0000 0000	71
9Ah	EEDATA	EEPROM data register								xxxx xxxx	89
9Bh	EEADR	EEPROM address register								xxxx xxxx	90
9Ch	EECON1	—	—	—	—	WRERR	WREN	WR	RD	---- x000	90
9Dh	EECON2	EEPROM control register 2 (not a physical register)								---- ----	90
9Eh	—	Unimplemented								—	—
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	67

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: For the Initialization Condition for Registers Tables, refer to Table 14-6 and Table 14-7.

TABLE 4-5: SPECIAL FUNCTION REGISTERS SUMMARY BANK2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset(1)	Details on Page
Bank 2											
100h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	28
101h	TMR0	Timer0 module's Register								xxxx xxxx	45
102h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28
103h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22
104h	FSR	Indirect data memory address pointer								xxxx xxxx	28
105h	—	Unimplemented								—	—
106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	36
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter					---0 0000	28
10Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	24
10Ch	—	Unimplemented								—	—
10Dh	—	Unimplemented								—	—
10Eh	—	Unimplemented								—	—
10Fh	—	Unimplemented								—	—
110h	—	Unimplemented								—	—
111h	—	Unimplemented								—	—
112h	—	Unimplemented								—	—
113h	—	Unimplemented								—	—
114h	—	Unimplemented								—	—
115h	—	Unimplemented								—	—
116h	—	Unimplemented								—	—
117h	—	Unimplemented								—	—
118h	—	Unimplemented								—	—
119h	—	Unimplemented								—	—
11Ah	—	Unimplemented								—	—
11Bh	—	Unimplemented								—	—
11Ch	—	Unimplemented								—	—
11Dh	—	Unimplemented								—	—
11Eh	—	Unimplemented								—	—
11Fh	—	Unimplemented								—	—

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented.

Note 1: For the Initialization Condition for Registers Tables, refer to Table 14-6 and Table 14-7.

TABLE 4-6: SPECIAL FUNCTION REGISTERS SUMMARY BANK3

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset(1)	Details on Page
Bank 3											
180h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	28
181h	OPTION	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23
182h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28
183h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22
184h	FSR	Indirect data memory address pointer								xxxx xxxx	28
185h	—	Unimplemented								—	—
186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	36
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter					---0 0000	28
18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	24
18Ch	—	Unimplemented								—	—
18Dh	—	Unimplemented								—	—
18Eh	—	Unimplemented								—	—
18Fh	—	Unimplemented								—	—
190h	—	Unimplemented								—	—
191h	—	Unimplemented								—	—
192h	—	Unimplemented								—	—
193h	—	Unimplemented								—	—
194h	—	Unimplemented								—	—
195h	—	Unimplemented								—	—
196h	—	Unimplemented								—	—
197h	—	Unimplemented								—	—
198h	—	Unimplemented								—	—
199h	—	Unimplemented								—	—
19Ah	—	Unimplemented								—	—
19Bh	—	Unimplemented								—	—
19Ch	—	Unimplemented								—	—
19Dh	—	Unimplemented								—	—
19Eh	—	Unimplemented								—	—
19Fh	—	Unimplemented								—	—

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: For the Initialization Condition for Registers Tables, refer to Table 14-6 and Table 14-7.

4.2.2.1 STATUS Register

The STATUS register, shown in Register 4-1, contains the arithmetic status of the ALU; the RESET status and the bank select bits for data memory (SRAM).

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are non-writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the status register as "000uu1uu" (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register because these instructions do not affect any STATUS bit. For other instructions not affecting any STATUS bits, see the "Instruction Set Summary".

Note 1: The C and DC bits operate as a $\overline{Borrow}$ and $\overline{Digit\ Borrow}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 4-1: STATUS REGISTER (ADDRESS: 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP**: Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
10 = Bank 2 (100h - 17Fh)
11 = Bank 3 (180h - 1FFh)

bit 4 **$\overline{TO}$**: Time out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for $\overline{borrow}$ the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.2.2.2 OPTION Register

The OPTION register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external RB0/INT interrupt, TMR0 and the weak pull-ups on PORTB.

Note:	To achieve a 1:1 prescaler assignment for TMR0, assign the prescaler to the WDT (PSA = 1). See Section 6.3.1.

REGISTER 4-2: OPTION REGISTER (ADDRESS: 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains the various enable and flag bits for all interrupt sources except the comparator module. See Section 4.2.2.4 and Section 4.2.2.5 for a description of the comparator enable and flag bits.

Note:	Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

REGISTER 4-3: INTCON REGISTER (ADDRESS: 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all un-masked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = When at least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.2.2.4 PIE1 Register

This register contains interrupt enable bits.

REGISTER 4-4: PIE1 REGISTER (ADDRESS: 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **EEIE:** EE Write Complete Interrupt Enable Bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 6 **CMIE**: Comparator Interrupt Enable bit
1 = Enables the comparator interrupt
0 = Disables the comparator interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **Unimplemented**: Read as '0'

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.2.2.5 PIR1 Register

This register contains interrupt flag bits.

Note: Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 4-5: PIR1 REGISTER (ADDRESS: 0Ch)

R/W-0	R/W-0	R-0	R-0	U-0	R/W-0	R/W-0	R/W-0
EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **EEIF:** EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation has not completed or has not been started

bit 6 **CMIF**: Comparator Interrupt Flag bit
1 = Comparator output has changed
0 = Comparator output has not changed

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **Unimplemented**: Read as '0'

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture Mode
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare Mode
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM Mode
Unused in this mode

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.2.2.6 PCON Register

The PCON register contains flag bits to differentiate between a Power-on Reset, an external MCLR Reset, WDT Reset or a Brown-out Reset.

Note: BOR is unknown on Power-on Reset. It must then be set by the user and checked on subsequent RESETS to see if BOR is cleared, indicating a brown-out has occurred. The BOR STATUS bit is a "don't care" and is not necessarily predictable if the brown-out circuit is disabled (by clearing the BOREN bit in the Configuration word).

REGISTER 4-6: PCON REGISTER (ADDRESS: 8Eh)

U-0	U-0	U-0	U-0	R/W-1	U-0	R/W-0	R/W-x
—	—	—	—	OSCF	—	POR	BOR
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **OSCF**: INTOSC oscillator frequency
1 = 4 MHz typical
0 = 37 kHz typical

bit 2 **Unimplemented:** Read as '0'

bit 1 **POR**: Power-on Reset STATUS bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

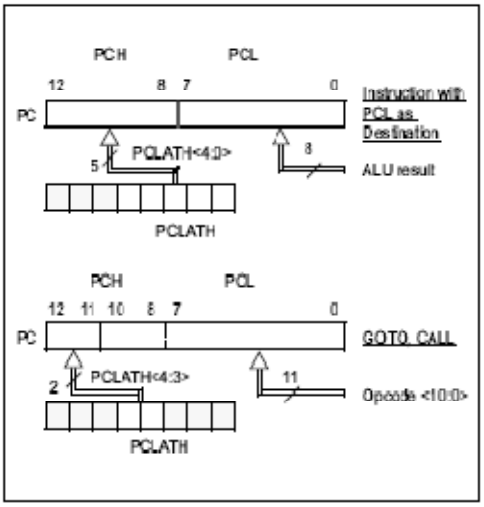
bit 0 **BOR**: Brown-out Reset STATUS bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.3 PCL and PCLATH

The program counter (PC) is 13-bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<12:8>) is not directly readable or writable and comes from PCLATH. On any RESET, the PC is cleared. Figure 4-4 shows the two situations for loading the PC. The upper example in Figure 4-4 shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in Figure 4-4 shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

FIGURE 4-4: LOADING OF PC IN DIFFERENT SITUATIONS



4.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to the application note *"Implementing a Table Read"* (AN556).

4.3.2 STACK

The PIC16F627A/628A/648A family has an 8-level deep x 13-bit wide hardware stack (Figure 4-1). The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

Note 1: There are no STATUS bits to indicate stack overflow or stack underflow conditions.

2: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW and RETFIE instructions, or the vectoring to an interrupt address.

4.4 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

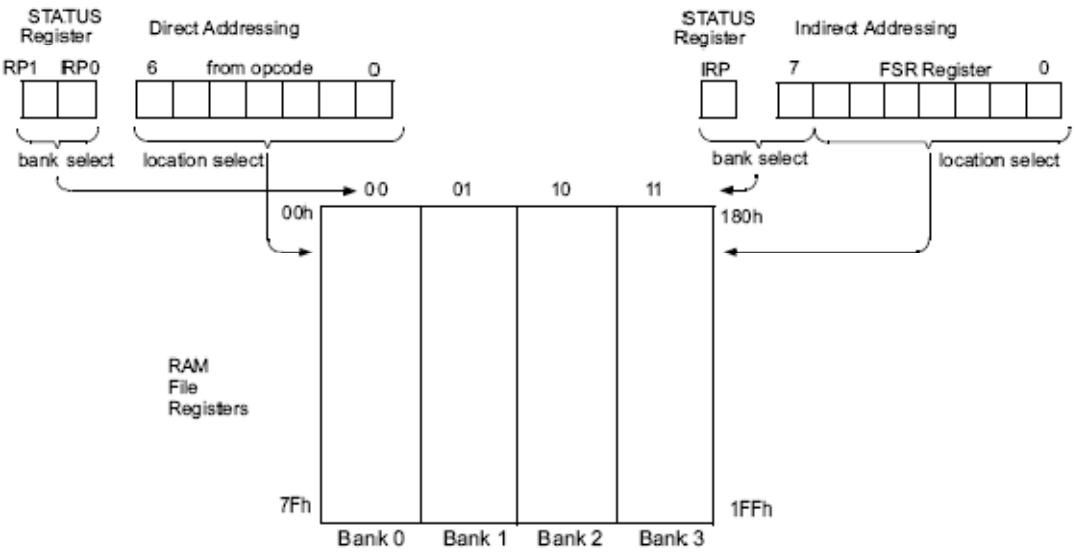
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses data pointed to by the file select register (FSR). Reading INDF itself indirectly will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-5.

A simple program to clear RAM location 20h-2Fh using indirect addressing is shown in Example 4-1.

EXAMPLE 4-1: Indirect Addressing

```
        MOVLW   0x20    ;initialize pointer
        MOVWF   FSR     ;to RAM
NEXT    CLRF    INDF    ;clear INDF register
        INCF    FSR     ;inc pointer
        BTFSS   FSR,4   ;all done?
        GOTO    NEXT    ;no clear next
                        ;yes continue
```

FIGURE 4-5: DIRECT/INDIRECT ADDRESSING PIC16F627A/628A/648A



Note: For memory map detail see Figure 4-3, Figure 4-2 and Figure 4-1.

5.0 I/O PORTS

The PIC16F627A/628A/648A have two ports, PORTA and PORTB. Some pins for these I/O ports are multiplexed with alternate functions for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

5.1 IPORTA and TRISA Registers

PORTA is an 8-bit wide latch. RA4 is a Schmitt Trigger input and an open drain output. Port RA4 is multiplexed with the T0CKI clock input. RA5(1) is a Schmitt Trigger input only and has no output drivers. All other RA port pins have Schmitt Trigger input levels and full CMOS output drivers. All pins have data direction bits (TRIS registers) which can configure these pins as input or output.

A '1' in the TRISA register puts the corresponding output driver in a High-impedance mode. A '0' in the TRISA register puts the contents of the output latch on the selected pin(s).

Reading the PORTA register reads the status of the pins whereas writing to it will write to the port latch. All write operations are read-modify-write operations. So a write to a port implies that the port pins are first read, then this value is modified and written to the port data latch.

The PORTA pins are multiplexed with comparator and voltage reference functions. The operation of these pins are selected by control bits in the CMCON (comparator control register) register and the VRCON (voltage reference control register) register. When selected as a comparator input, these pins will read as '0's.

Note 1: RA5 shares function with VPP. When VPP voltage levels are applied to RA5, the device will enter Programming mode.

2: On RESET, the TRISA register is set to all inputs. The digital inputs (RA<3:0>) are disabled and the comparator inputs are forced to ground to reduce current consumption.

3: TRISA<6:7> is overridden by oscillator configuration. When PORTA<6:7> is overridden, the data reads '0' and the TRISA<6:7> bits are ignored.

TRISA controls the direction of the RA pins, even when they are being used as comparator inputs. The user must make sure to keep the pins configured as inputs when using them as comparator inputs.

The RA2 pin will also function as the output for the voltage reference. When in this mode, the VREF pin is a very high-impedance output. The user must configure TRISA<2> bit as an input and use high-impedance loads.

In one of the Comparator modes defined by the CMCON register, pins RA3 and RA4 become outputs of the comparators. The TRISA<4:3> bits must be cleared to enable outputs to use this function.

EXAMPLE 5-1: Initializing PORTA

```
CLRF    PORTA       ;Initialize PORTA by
                    ;setting
                    ;output data latches
MOVLW   0x07        ;Turn comparators off and
MOVWF   CMCON       ;enable pins for I/O
                    ;functions
BCF     STATUS, RP1
BSF     STATUS, RP0 ;Select Bank1
MOVLW   0x1F        ;Value used to initialize
                    ;data direction
MOVWF   TRISA       ;Set RA<4:0> as inputs
                    ;TRISA<5> always
                    ;read as '1'.
                    ;TRISA<7:6>
                    ;depend on oscillator
                    ;mode
```

FIGURE 5-1: BLOCK DIAGRAM OF RA0/AN0:RA1/AN1 PINS

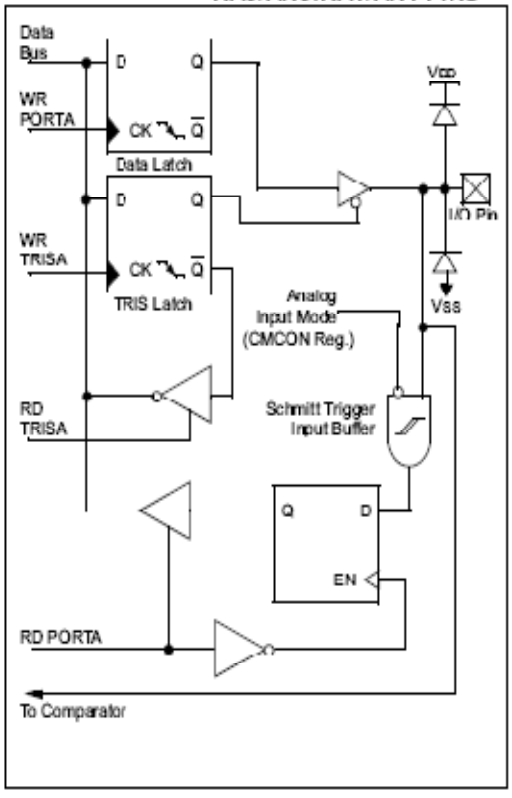


TABLE 5-1: PORTA FUNCTIONS

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bi-directional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bi-directional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bi-directional I/O port. Output is open drain type.
	T0CKI	ST	—	External clock input for TMR0 or comparator output
	CMP2	—	OD	Comparator 2 output
RA5/$\overline{\text{MCLR}}$/VPP	RA5	ST	—	Input port
	$\overline{\text{MCLR}}$	ST	—	Master clear. When configured as $\overline{\text{MCLR}}$, this pin is an active low RESET to the device. Voltage on $\overline{\text{MCLR}}$/VPP must not exceed VDD during normal device operation.
	VPP	HV	—	Programming voltage input.
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bi-directional I/O port
	OSC2	—	XTAL	Oscillator crystal output. Connects to crystal resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In RC or INTOSC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bi-directional I/O port
	OSC1	XTAL	—	Oscillator crystal input. Connects to crystal resonator in Crystal Oscillator mode.
	CLKIN	ST	—	External clock source input. RC biasing pin.

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

TABLE 5-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA(1)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on All Other RESETS
05h	PORTA	RA7	RA6	RA5(2)	RA4	RA3	RA2	RA1	RA0	xxxx 0000	qqqu 0000
85h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	1111 1111
1Fh	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	0000 0000
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: Shaded bits are not used by PORTA.

2: MCLRE Configuration Bit sets RA5 functionality.

5.2 PORTB and TRISB Registers

PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. A '1' in the TRISB register puts the corresponding output driver in a High-impedance mode. A '0' in the TRISB register puts the contents of the output latch on the selected pin(s).

PORTB is multiplexed with the external interrupt, USART, CCP module and the TMR1 clock input/output. The standard port functions and the alternate port functions are shown in Table 5-3. Alternate port functions may override TRIS setting when enabled.

Reading PORTB register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. So a write to a port implies that the port pins are first read, then this value is modified and written to the port data latch.

Each of the PORTB pins has a weak internal pull-up (≈200 µA typical). A single control bit can turn on all the pull-ups. This is done by clearing the RBPU (OPTION<7>) bit. The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on Power-on Reset.

Four of PORTB's pins, RB<7:4>, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB<7:4> pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RBIF interrupt (flag latched in INTCON<0>).

This interrupt can wake the device from SLEEP. The user, in the interrupt service routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.

b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

This interrupt on mismatch feature, together with software configurable pull-ups on these four pins allow easy interface to a key pad and make it possible for wake-up on key-depression. (See AN552)

Note:	If a change on the I/O pin should occur when a read operation is being executed (start of the Q2 cycle), then the RBIF interrupt flag may not get set.

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

TABLE 5-3: PORTB FUNCTIONS

Name	Function	Input Type	Output Type	Description
RB0/INT	RB0	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	INT	ST	—	External interrupt.
RB1/RX/DT	RB1	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	RX	ST	—	USART Receive Pin
	DT	ST	CMOS	Synchronous data I/O
RB2/TX/CK	RB2	TTL	CMOS	Bi-directional I/O port
	TX	—	CMOS	USART Transmit Pin
	CK	ST	CMOS	Synchronous Clock I/O. Can be software programmed for internal weak pull-up.
RB3/CCP1	RB3	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	CCP1	ST	CMOS	Capture/Compare/PWM/I/O
RB4/PGM	RB4	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	PGM	ST	—	Low voltage programming input pin. When low voltage programming is enabled, the interrupt-on-pin change and weak pull-up resistor are disabled.
RB5	RB5	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
RB6/T1OSO/T1CKI/PGC	RB6	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSO	—	XTAL	Timer1 Oscillator Output
	T1CKI	ST	—	Timer1 Clock Input
	PGC	ST	—	ICSP Programming Clock
RB7/T1OSI/PGD	RB7	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSI	XTAL	—	Timer1 Oscillator Input
	PGD	ST	CMOS	ICSP Data I/O

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

TABLE 5-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB(1)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on All Other RESETS
06h, 106h	PORTB	RB7	RB6	RB5	RB4(2)	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h, 181h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: u = unchanged, x = unknown

Note 1: Shaded bits are not used by PORTB.

2: LVP Configuration Bit sets RB4 functionality.

5.3 I/O Programming Considerations

5.3.1 BI-DIRECTIONAL I/O PORTS

Any instruction that writes, operates internally as a read followed by a write operation. The BCF and BSF instructions, for example, read the register into the CPU, execute the bit operation and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit5 of PORTB will cause all eight bits of PORTB to be read into the CPU. Then the BSF operation takes place on bit5 and PORTB is written to the output latches. If another bit of PORTB is used as a bi-directional I/O pin (e.g., bit0) and is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and rewritten to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the Input mode, no problem occurs. However, if bit0 is switched into Output mode later on, the content of the data latch may now be unknown.

Reading a port register reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read-modify-write instructions (ex. BCF, BSF, etc.) on a port, the value of the port pins is read, the desired operation is done to this value, and this value is then written to the port latch.

Example 5-2 shows the effect of two sequential read-modify-write instructions (ex., BCF, BSF, etc.) on an I/O port.

A pin actively outputting a Low or High should not be driven from external devices at the same time in order to change the level on this pin ("wired-or", "wired-and"). The resulting high output currents may damage the chip.

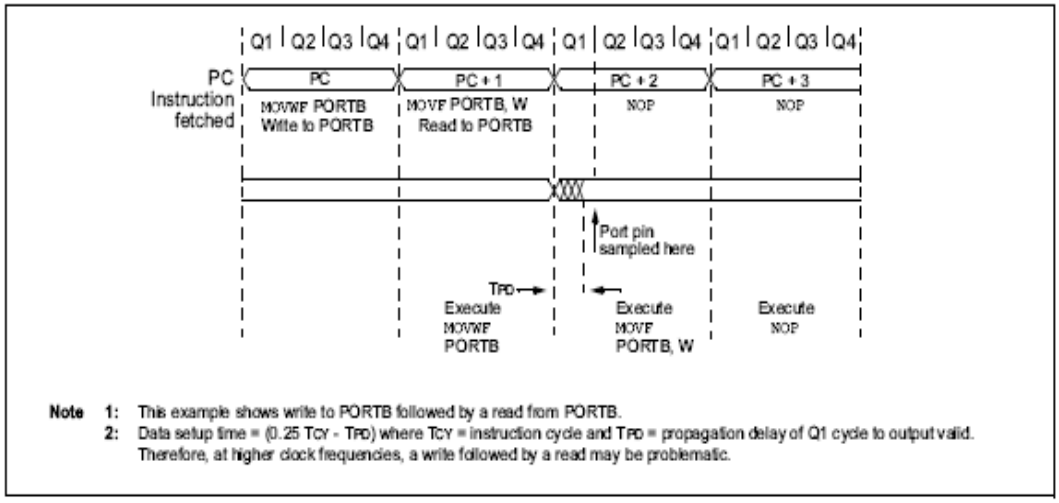
EXAMPLE 5-2: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT

```
;Initial PORT settings:PORTB<7:4> Inputs
;                      PORTB<3:0> Outputs
;PORTB<7:6> have external pull-up and are
;not connected to other circuitry
;
;                      PORT latchPORT Pins
                       ---------- ----------
    BCF STATUS, RP0    ;
    BCF PORTB, 7       ;01pp pppp 11pp pppp
    BSF STATUS, RP0    ;
    BCF TRISB, 7       ;10pp pppp 11pp pppp
    BCF TRISB, 6       ;10pp pppp 10pp pppp
;
;Note that the user may have expected the
;pin values to be 00pp pppp. The 2nd BCF
;caused RB7 to be latched as the pin value
;(High).
```

5.3.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 5-16). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before the next instruction, which causes that file to be read into the CPU, is executed. Otherwise, the previous state of that pin may be read into the CPU rather than the new state. When in doubt, it is better to separate these instructions with a NOP or another instruction not accessing this I/O port.

FIGURE 5-16: SUCCESSIVE I/O OPERATION



Note 1: This example shows write to PORTB followed by a read from PORTB.
2: Data setup time = (0.25 TCY - TPD) where TCY = instruction cycle and TPD = propagation delay of Q1 cycle to output valid. Therefore, at higher clock frequencies, a write followed by a read may be problematic.

6.0 TIMER0 MODULE

The Timer0 module timer/counter has the following features:

- 8-bit timer/counter
- Read/Write capabilities
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Figure 6-1 is a simplified block diagram of the Timer0 module. Additional information is available in the PICmicro™ Mid-Range MCU Family Reference Manual, DS33023.

Timer mode is selected by clearing the T0CS bit (OPTION<5>). In Timer mode the TMR0 register value will increment every instruction cycle (without prescaler). If the TMR0 register is written to, the increment is inhibited for the following two cycles. The user can work around this by writing an adjusted value to the TMR0 register.

Counter mode is selected by setting the T0CS bit. In this mode the TMR0 register value will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the source edge (T0SE) control bit (OPTION<4>). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 6.2.

The prescaler is shared between the Timer0 module and the Watchdog Timer. The prescaler assignment is controlled in software by the control bit PSA (OPTION<3>). Clearing the PSA bit will assign the prescaler to Timer0. The prescaler is not readable or writable. When the prescaler is assigned to the Timer0 module, prescale value of 1:2, 1:4,..., 1:256 are selectable. Section 6.3 details the operation of the prescaler.

6.1 Timer0 Interrupt

Timer0 interrupt is generated when the TMR0 register timer/counter overflows from FFh to 00h. This overflow sets the T0IF bit. The interrupt can be masked by clearing the T0IE bit (INTCON<5>). The T0IF bit (INTCON<2>) must be cleared in software by the Timer0 module interrupt service routine before re-enabling this interrupt. The Timer0 interrupt cannot wake the processor from SLEEP since the timer is shut off during SLEEP.

6.2 Using Timer0 with External Clock

When an external clock input is used for Timer0, it must meet certain requirements. The external clock requirement is due to internal phase clock (Tosc) synchronization. Also, there is a delay in the actual incrementing of Timer0 after synchronization.

6.2.1 EXTERNAL CLOCK SYNCHRONIZATION

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 6-1). Therefore, it is necessary for T0CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

When a prescaler is used, the external clock input is divided by the asynchronous ripple-counter type prescaler so that the prescaler output is symmetrical. For the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for T0CKI to have a period of at least 4Tosc (and a small RC delay of 40 ns) divided by the prescaler value. The only requirement on T0CKI high and low time is that they do not violate the minimum pulse width requirement of 10 ns. Refer to parameters 40, 41 and 42 in the electrical specification of the desired device. See Table 17-9.

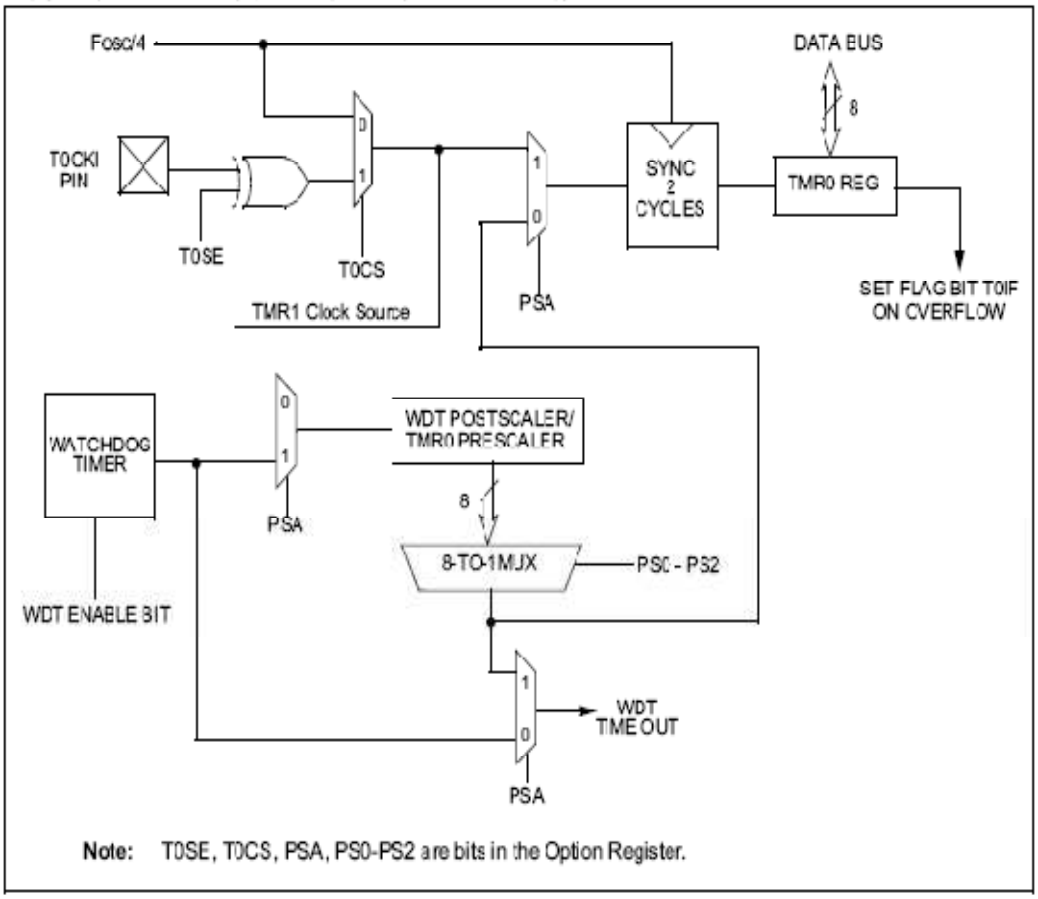
6.3 Timer0 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer. A prescaler assignment for the Timer0 module means that there is no postscaler for the Watchdog Timer, and vice-versa.

The PSA and PS2:PS0 bits (OPTION<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF 1`, `MOVWF 1`, `BSF 1, x....etc.`) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

FIGURE 6-1: BLOCK DIAGRAM OF THE TIMER0/WDT



Note: T0SE, T0CS, PSA, PS0-PS2 are bits in the Option Register.

7.0 TIMER1 MODULE

The Timer1 module is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L) which are readable and writable. The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 Interrupt, if enabled, is generated on overflow of the TMR1 register pair which latches the interrupt flag bit TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing the Timer1 interrupt enable bit TMR1IE (PIE1<0>).

Timer1 can operate in one of two modes:

- As a timer
- As a counter

The Operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

In Timer mode, the TMR1 register pair value increments every instruction cycle. In Counter mode, it increments on every rising edge of the external clock input.

Timer1 can be enabled/disabled by setting/clearing control bit TMR1ON (T1CON<0>).

Timer1 also has an internal "RESET input". This RESET can be generated by the CCP module (Section 9.0). Register 7-1 shows the Timer1 control register.

For the PIC16F627A/628A/648A, when the Timer1 oscillator is enabled (T1OSCEN is set), the RB7/T1OSI and RB6/T1OSO/T1CKI pins become inputs. That is, the TRISB<7:6> value is ignored.

REGISTER 7-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS: 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut off(1)

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Control bit
TMR1CS = 1
1 = Do not synchronize external clock input
0 = Synchronize external clock input
TMR1CS = 0
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RB6/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Disables Timer1
0 = Stops Timer1

Note 1: The oscillator inverter and feedback resistor are turned off to eliminate power drain.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.1 Timer1 Operation in Timer Mode

Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is Fosc/4. The synchronize control bit T1SYNC (T1CON<2>) has no effect since the internal clock is always in sync.

7.2 Timer1 Operation in Synchronized Counter Mode

Counter mode is selected by setting bit TMR1CS. In this mode the TMR1 register pair value increments on every rising edge of clock input on pin RB7/T1OSI when bit T1OSCEN is set or pin RB6/T1OSO/T1CKI when bit T1OSCEN is cleared.

If $\overline{\text{T1SYNC}}$ is cleared, then the external clock input is synchronized with internal phase clocks. The synchronization is done after the prescaler stage. The prescaler stage is an asynchronous ripple-counter.

In this configuration, during SLEEP mode, the TMR1 register pair value will not increment even if the external clock is present, since the synchronization circuit is shut off. The prescaler however will continue to increment.

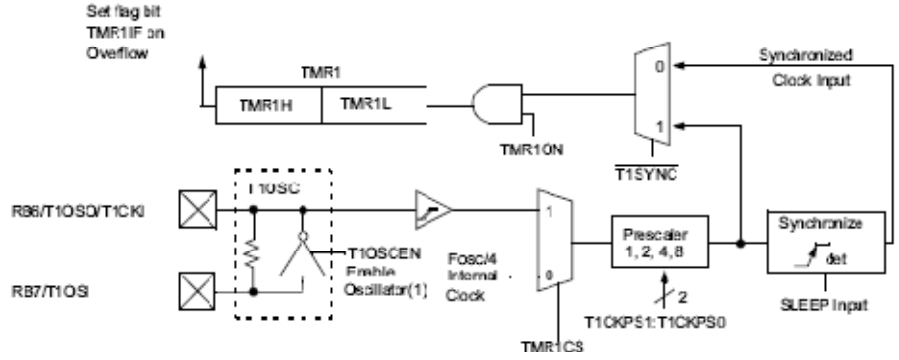
7.2.1 EXTERNAL CLOCK INPUT TIMING FOR SYNCHRONIZED COUNTER MODE

When an external clock input is used for Timer1 in synchronized Counter mode, it must meet certain requirements. The external clock requirement is due to internal phase clock (Tosc) synchronization. Also, there is a delay in the actual incrementing of the TMR1 register pair value after synchronization.

When the prescaler is 1:1, the external clock input is the same as the prescaler output. The synchronization of T1CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks. Therefore, it is necessary for T1CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to the appropriate electrical specifications, parameters 45, 46, and 47.

When a prescaler other than 1:1 is used, the external clock input is divided by the asynchronous ripple-counter type prescaler so that the prescaler output is symmetrical. In order for the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for T1CKI to have a period of at least 4Tosc (and a small RC delay of 40 ns) divided by the prescaler value. The only requirement on T1CKI high and low time is that they do not violate the minimum pulse width requirements of 10 ns). Refer to the appropriate electrical specifications, parameters 45, 46, and 47.

FIGURE 7-1: TIMER1 BLOCK DIAGRAM



Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

7.3 Timer1 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{T1SYNC}}$ (T1CON<2>) is set, the external clock input is not synchronized. The timer continues to increment asynchronous to the internal phase clocks. The timer will continue to run during SLEEP and can generate an interrupt on overflow, which will wake-up the processor. However, special precautions in software are needed to read/write the timer (Section 7.3.2).

Note:	In Asynchronous Counter mode, Timer1 cannot be used as a time-base for capture or compare operations.

7.3.1 EXTERNAL CLOCK INPUT TIMING WITH UNSYNCHRONIZED CLOCK

If control bit $\overline{\text{T1SYNC}}$ is set, the timer will increment completely asynchronously. The input clock must meet certain minimum high and low time requirements. Refer to Table 17-9 in the Electrical Specifications Section, timing parameters 45, 46, and 47.

7.3.2 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading the TMR1H or TMR1L register while the timer is running, from an external asynchronous clock, will produce a valid read (taken care of in hardware). However the user should keep in mind that reading the 16-bit timer in two 8-bit values itself poses certain problems since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers while the register is incrementing. This may produce an unpredictable value in the timer register.

Reading the 16-bit value requires some care. Example 7-1 is an example routine to read the 16-bit timer value. This is useful if the timer cannot be stopped.

EXAMPLE 7-1: READING A 16-BIT FREE-RUNNING TIMER

```
; All interrupts are disabled
    MOVF    TMR1H, W    ;Read high byte
    MOVWF   TMPH        ;
    MOVF    TMR1L, W    ;Read low byte
    MOVWF   TMPL        ;
    MOVF    TMR1H, W    ;Read high byte
    SUBWF   TMPH, W     ;Sub 1st read with
                        ;2nd read
    BTFSC   STATUS,Z    ;Is result = 0
    GOTO    CONTINUE    ;Good 16-bit read
;
; TMR1L may have rolled over between the
; read of the high and low bytes. Reading
; the high and low bytes now will read a good
; value.
;
    MOVF    TMR1H, W    ;Read high byte
    MOVWF   TMPH        ;
    MOVF    TMR1L, W    ;Read low byte
    MOVWF   TMPL        ;
; Re-enable the Interrupts (if required)
CONTINUE                ;Continue with your
                        ;code
```

7.4 Timer1 Oscillator

A crystal oscillator circuit is built in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON<3>). It will continue to run during SLEEP. It is primarily intended for a 32.768 kHz watch crystal. Table 7-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper oscillator start-up.

TABLE 7-1: CAPACITOR SELECTION FOR THE TIMER1 OSCILLATOR

Freq	C1	C2
32.768 kHz	15 pF	15 pF
These values are for design guidance only. Consult AN826 (DS00826) for further information on Crystal/Capacitor Selection.		

7.5 Resetting Timer1 Using a CCP Trigger Output

If the CCP1 module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will RESET Timer1.

Note:	The special event triggers from the CCP1 module will not set interrupt flag bit TMR1IF (PIR1<0>).

Timer1 must be configured for either timer or synchronized Counter mode to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this RESET operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPRxH:CCPRxL registers pair effectively becomes the period register for Timer1.

7.6 Resetting Timer1 Register Pair (TMR1H, TMR1L)

TMR1H and TMR1L registers are not reset to 00h on a POR or any other RESET except by the CCP1 special event triggers.

T1CON register is reset to 00h on a Power-on Reset or a Brown-out Reset, which shuts off the timer and leaves a 1:1 prescale. In all other RESETS, the register is unaffected.

7.7 Timer1 Prescaler

The prescaler counter is cleared on writes to the TMR1H or TMR1L registers.

TABLE 7-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	0000 -000	0000 -000
8Ch	PIE1	EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	0000 -000	0000 -000
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by the Timer1 module.

Anexo E. Código Fuente

Brazo Derecho

```
'*  Name    : UNTITLED.BAS
*
'*  Author  : [select VIEW...EDITOR OPTIONS]
*
'*  Notice  : Copyright (c) 2009 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved
*
'*  Date    : 17/06/2009
*
'*  Version : 1.0
*
'*  Notes   :
*
'*          :
*
INCLUDE "modedefs.bas"  ' Set receive register to receiver enabled
DEFINE HSER_RCSTA 90h   ' Set transmit register to transmitte0r enabled
DEFINE HSER_TXSTA 20h   ' Set baud rate
DEFINE HSER_BAUD 2400   ' Set SPBRG directly (normally set by HSER_BAUD)
DEFINE HSER_SPBRG 25
'REPRODUCCION DE DATOS CAPTURADOS

@ device XT_osc;  utiliza cristal externo 4mhz
@ device wdt_off;
@ device mclr_off;
@ device bod_off;
cmcon =7;
DEFINE OSC 4
TRISA=%00000000
TRISB=%00000000

I VAR BYTE
J VAR BYTE
W VAR BYTE



RS VAR BYTE[8]
BA VAR BYTE[8]
BA1 VAR BYTE[8]
BC VAR BYTE[8]
BC1 VAR BYTE[8]
BAM VAR BYTE[8]
BAM1 VAR BYTE[8]
BAD VAR BYTE[8]
BAD1 VAR BYTE[8]
BAP VAR BYTE[8]
BAP1 VAR BYTE[8]
BMC VAR BYTE[8]
BMC1 VAR BYTE[8]
MB VAR BYTE[8]
MB1 VAR BYTE[8]
BAMP VAR BYTE[8]
BAMP1 VAR BYTE[8]
BD VAR BYTE[8]
BD1 VAR BYTE[8]


RS[0]=5'$5
rS[1]=210'$60
rS[2]=210'$0
rS[3]=130'$0
rS[4]=0'$0
rS[5]=0'$0
rS[6]=45'$0
RS[7]=255'$FF

 'BDERECHOATRAS
BA[0]=5'$5
BLADO
BA[1]=235'$60
BA[2]=200'$0
BA[3]=180'$0
BA[4]=0'$0
BA[5]=0'$0
BA[6]=110'$0
BA[7]=255'$FF

'BDERECHOCADERA
BC[0]=5'$5
BLADO
BC[1]=210'$60
BC[2]=160'$0
BC[3]=220'$0
BC[4]=0'$0
BC[5]=0'$0
BC[6]=85'$0
BC[7]=255'$FF


'drechoatras movimiento

BAM[0]=5'$5
BLADO
BAM[1]=235'$60
BAM[2]=165'$0
BAM[3]=180'$0
BAM[4]=0'$0
BAM[5]=0'$0
BAM[6]=75'$0
BAM[7]=255'$FF

'derecha adelante movimiento
BAD[0]=5'$5
BLADO
BAD[1]=150'$60
BAD[2]=180'$0
BAD[3]=180'$0
BAD[4]=0'$0
BAD[5]=0'$0
BAD[6]=5'$0
BAD[7]=255'$FF

BAD1[0]=5'$5
BAD1[1]=150'$60
BAD1[2]=220'$0
BAD1[3]=180'$0
BAD1[4]=0'$0
BAD1[5]=0'$0
BAD1[6]=45'$0
BAD1[7]=255'$FF
'bderecho parabrisas
BAP[0]=5'$5
BLADO
BAP[1]=150'$60
BAP[2]=220'$0
BAP[3]=180'$0
BAP[4]=0'$0
BAP[5]=0'$0
BAP[6]=45'$0
BAP[7]=255'$FF

BAP1[0]=5'$5
BAP1[1]=150'$60
BAP1[2]=180'$0
BAP1[3]=180'$0
BAP1[4]=0'$0
BAP1[5]=0'$0
BAP1[6]=5'$0
BAP1[7]=255'$FF

'movimiento brazoDERECHO a la cabeza
BMC[0]=5'$5
BMC[1]=40'$60
BMC[2]=160'$0
BMC[3]=220'$0
BMC[4]=0'$0
BMC[5]=0'$0
BMC[6]=170'$0
BMC[7]=255'$FF

BMC1[0]=5'$5
BMC1[1]=40'$60
BMC1[2]=135'$0
BMC1[3]=220'$0
BMC1[4]=0'$0
BMC1[5]=0'$0
BMC1[6]=145'$0
BMC1[7]=255'$FF

'MOVIMIENTO BRAZO DERECHO ADELANTEATRAS
MB[0]=5'$5
MB[1]=235'$60
MB[2]=200'$0
MB[3]=130'$0
MB[4]=0'$0
MB[5]=0'$0
MB[6]=60'$0
MB[7]=255'$FF

MB1[0]=5'$5
MB1[1]=60'$60
MB1[2]=200'$0
MB1[3]=230'$0
MB1[4]=0'$0
MB1[5]=0'$0
MB1[6]=240'$0
MB1[7]=255'$FF
' brazo derechomovimiento parabrisas atras

BAMP[0]=5'$5
BLADO
BAMP[1]=235'$60
BAMP[2]=165'$0
BAMP[3]=180'$0
BAMP[4]=0'$0
BAMP[5]=0'$0
BAMP[6]=75'$0
BAMP[7]=255'$FF

BAMP1[0]=5'$5
BAMP1[1]=235'$60
BAMP1[2]=200'$0
BAMP1[3]=180'$0
BAMP1[4]=0'$0
BAMP1[5]=0'$0
BAMP1[6]=110'$0
BAMP1[7]=255'$FF

'BRAZOS MUEVE ADELANTE
BD[0]=5'$5
BD[1]=120'$60
BD[2]=210'$0
BD[3]=130'$0
BD[4]=0'$0
BD[5]=0'$0
BD[6]=210'$0
BD[7]=255'$FF

BD1[0]=5'$5
BD1[1]=20'$60
BD1[2]=210'$0
BD1[3]=130'$0
BD1[4]=0'$0
BD1[5]=0'$0
BD1[6]=110'$0
BD1[7]=255'$FF

 '---------------seteo----------


FOR J=0 TO 100
 GOsub SETEO
next j

'**************programa ******************
INICIO:

    GOSUB RITMO1
       For I=0 TO 30
      gosub seteo
     Next  i
```

```
GOTO INICIO

END

'*************************
*****************
'-------------------
subrutinas----------
SETEO:
'CORRECTO
    For I=0 TO 7
      HSerout [RS[I]]
    Next  i
RETURN
BATRAS:
  For J=0 TO 10
    For I=0 TO 7
      HSerout [BA[I]]
    Next  i
  NEXT J
RETURN

BCADERA:
  For J=0 TO 10
    For I=0 TO 7
      HSerout [BC[I]]
    Next  i
  NEXT J
RETURN

  gosub seteo
  gosub seteo
  gosub seteo
  gosub seteo

BCABEZA:
  For J=0 TO 20
    For I=0 TO 7
      HSerout [BMC[I]]
    Next  i
  NEXT J
    For J=0 TO 20
    For I=0 TO 7
      HSerout
[BMC1[I]]
    Next  i
  NEXT J
RETURN

BATRASM:
  For J=0 TO 15
    For I=0 TO 7
      HSerout [BAm[I]]
    Next  i
  NEXT J
    For J=0 TO 15
    For I=0 TO 7
      HSerout [BA[I]]
    Next  i
  NEXT J
RETURN

BADELANTEM:
  For J=0 TO 15
    For I=0 TO 7
      HSerout [BAD[I]]
    Next  i
  NEXT J
    For J=0 TO 15
    For I=0 TO 7
      HSerout
[BAD1[I]]
    Next  i
  NEXT J
RETURN

BPARABRISAS:
  For J=0 TO 10
    For I=0 TO 7
      HSerout [BAP[I]]
    Next  i
  NEXT J
    For J=0 TO 10
    For I=0 TO 7
      HSerout
[BAP1[I]]
    Next  i
  NEXT J
RETURN

BRAZO:
  For J=0 TO 10
    For I=0 TO 7
      HSerout [MB[I]]
    Next  i
  NEXT J
    For J=0 TO 10
    For I=0 TO 7
      HSerout [MB1[I]]
    Next  i
  NEXT J
RETURN
BPARABATRAS:
  For J=0 TO 10
    For I=0 TO 7
      HSerout
[BAMP[I]]
    Next  i
  NEXT J
    For J=0 TO 10
    For I=0 TO 7
      HSerout
[BAMP1[I]]
    Next  i
  NEXT J
RETURN

BADELANTE:
  For J=0 TO 10
    For I=0 TO 7
      HSerout [BD[I]]
    Next  i
  NEXT J
    For J=0 TO 10
    For I=0 TO 7
      HSerout [BD1[I]]
    Next  i
  NEXT J
RETURN

'+++++++++++++++rit
mos++++++++++++++
++++++

RITMO1:

  w=0
  while w<40
    gosub batras
  w=w+1
  wend

    w=0
  while w<10
    gosub batrasm
  w=w+1
  wend

  For J=0 TO 10
    For I=0 TO 7
      HSerout [Bam[I]]
    Next  i
  NEXT J

w=0
while w<40
  gosub bcadera
w=w+1
wend

  w=0
while w<20
  gosub bcadera
w=w+1
wend
  w=0
while w<5
  gosub bcabeza
w=w+1
wend

    w=0
while w<10
  gosub seteo
w=w+1
wend


w=0
while w<18
  gosub badelantem
w=w+1
wend

w=0
while w<37
  gosub bparabrisas
w=w+1
wend

  w=0
while w<42
  gosub brazo
w=w+1
wend

RETURN
```

Brazo Izquierdo

```
INCLUDE
"modedefs.bas"  ' Set
receive register to
receiver enabled

DEFINE
HSER_RCSTA 90h   '
Set transmit register
to transmitte0r
enabled

DEFINE
HSER_TXSTA 20h   '
Set baud rate

DEFINE
HSER_BAUD 2400   '
Set SPBRG directly
(normally set by
HSER_BAUD)

DEFINE
HSER_SPBRG 25
'REPRODUCCION
DE DATOS
CAPTURADOS



@ device XT_osc;
utiliza cristal externo
4mhz

@ device wdt_off;

@ device mclr_off;

@ device bod_off;

cmcon =7;

DEFINE OSC 4



I VAR BYTE

J VAR BYTE

W VAR BYTE

RS VAR BYTE[8]

BA VAR BYTE[8]

BA1 VAR BYTE[8]

BC VAR BYTE[8]

BC1 VAR BYTE[8]

BAM VAR BYTE[8]

BMC VAR BYTE[8]

BMC1 VAR BYTE[8]

BAM1 VAR BYTE[8]
```

```
BAD VAR BYTE[8]
BAD1 VAR BYTE[8]
BAP VAR BYTE[8]
BAP1 VAR BYTE[8]
MB VAR BYTE[8]
MB1 VAR BYTE[8]
BAMP VAR BYTE[8]
BAMP1 VAR BYTE[8]
BD VAR BYTE[8]
BD1 VAR BYTE[8]


TRISA=%00000000
TRISB=%00000000


RS[0]=5'$5
rS[1]=30'$30
rS[2]=50'$50
rS[3]=130'$130
rS[4]=0'$0
rS[5]=0'$0
rS[6]=215'$215
RS[7]=255'$FF


BA[0]=5'$5
BA[1]=5'$60
BA[2]=70'$0
BA[3]=80'$0
BA[4]=0'$0
BA[5]=0'$0
BA[6]=160'$0
BA[7]=255'$FF


'IzquierdoBrazo
cadera
BC[0]=5'$5
BC[1]=30'$60
BC[2]=100'$0
BC[3]=40'$0
BC[4]=0'$0
BC[5]=0'$0
BC[6]=175'$0
BC[7]=255'$FF


'izquierda movimiento
atras


BAM1[0]=5'$5
BAM1[1]=5'$60
BAM1[2]=105'$0
BAM1[3]=80'$0
BAM1[4]=0'$0
BAM1[5]=0'$0
BAM1[6]=195'$0
BAM1[7]=255'$FF


'movimiento
brazoIZQUIERDO a la
cabeza
BMC[0]=5'$5
BMC[1]=200'$60
BMC[2]=100'$0
BMC[3]=40'$0
BMC[4]=0'$0
BMC[5]=0'$0
BMC[6]=90'$0
BMC[7]=255'$FF


BMC1[0]=5'$5
BMC1[1]=200'$60
BMC1[2]=125'$0
BMC1[3]=40'$0
BMC1[4]=0'$0
BMC1[5]=0'$0
BMC1[6]=115'$0
BMC1[7]=255'$FF


'izquierdo adelante
movimiento
BAD[0]=5'$5
BAD[1]=80'$60
BAD[2]=100'$0
BAD[3]=60'$0
BAD[4]=0'$0
BAD[5]=0'$0
BAD[6]=245'$0
BAD[7]=255'$FF


BAD1[0]=5'$5
BAD1[1]=80'$60
BAD1[2]=60'$0
BAD1[3]=60'$0
BAD1[4]=0'$0
BAD1[5]=0'$0
BAD1[6]=205'$0
BAD1[7]=255'$F


'izquierdo parabrisas
BAP[0]=5'$5
BAP[1]=85'$60
BAP[2]=140'$0
BAP[3]=60'$0
BAP[4]=0'$0
BAP[5]=0'$0
BAP[6]=35'$0
BAP[7]=255'$FF


BAP1[0]=5'$5
BAP1[1]=85'$60
BAP1[2]=70'$0
BAP1[3]=60'$0
BAP1[4]=0'$0
BAP1[5]=0'$0
BAP1[6]=220'$0
BAP1[7]=255'$FF


 'MOVIMIENTO
BRAZO IZQUIERDO
ADELANTEATRAS
MB[0]=5'$5
MB[1]=180'$60
MB[2]=60'$0
MB[3]=30'$0
MB[4]=0'$0
MB[5]=0'$0
MB[6]=20'$0
MB[7]=255'$FF


MB1[0]=5'$5
MB1[1]=5'$60
MB1[2]=60'$0
MB1[3]=130'$0
MB1[4]=0'$0
MB1[5]=0'$0
MB1[6]=200'$0
MB1[7]=255'$FF


'brazo izquierdo
movimiento
parabrisas atras


BAMP[0]=5'$5
BAMP[1]=5'$60
BAMP[2]=70'$0
BAMP[3]=80'$0
BAMP[4]=0'$0
BAMP[5]=0'$0
```

```
BAMP[6]=160'$0
BAMP[7]=255'$FF

BAMP1[0]=5'$5
BAMP1[1]=5'$60
BAMP1[2]=105'$0
BAMP1[3]=80'$0
BAMP1[4]=0'$0
BAMP1[5]=0'$0
BAMP1[6]=195'$0
BAMP1[7]=255'$FF

'BRAZOS MUEVE ADELANTE
BD[0]=5'$5
BD[1]=220'$60
BD[2]=50'$0
BD[3]=130'$0
BD[4]=0'$0
BD[5]=0'$0
BD[6]=150'$0
BD[7]=255'$FF

BD1[0]=5'$5
BD1[1]=120'$60
BD1[2]=50'$0
BD1[3]=130'$0
BD1[4]=0'$0
BD1[5]=0'$0
BD1[6]=50'$0
BD1[7]=255'$FF
'----------seteo----------
FOR J=0 TO 100   ' 4 SEGUNDOS
 GOsub SETEO
next j

'**********programa**********
INICIO:

   GOSUB RITMO1

    For I=0 TO 30
     gosub seteo
    Next  i

  GOTO INICIO
  END

'**********subrutinas**********
SETEO:'CORRECTO
    For I=0 TO 7
     HSerout [RS[I]]
    Next  i
RETURN
BATRAS:
  For J=0 TO 10
    For I=0 TO 7
     HSerout [BA[I]]
    Next  i
  NEXT J

RETURN
BCADERA:
  For J=0 TO 10
    For I=0 TO 7
     HSerout [BC[I]]
    Next  i
  NEXT J
 RETURN

BATRASM:
  For J=0 TO 15
    For I=0 TO 7
     HSerout [BA[I]]
    Next  i
  NEXT J
    For J=0 TO 15
    For I=0 TO 7
     HSerout [BAM1[I]]
    Next  i
  NEXT J
RETURN

 BCABEZA:
  For J=0 TO 20
    For I=0 TO 7
     HSerout [BMC[I]]
    Next  i
  NEXT J
    For J=0 TO 20
    For I=0 TO 7
     HSerout [BMC1[I]]
    Next  i
  NEXT J
RETURN
 BADELANTEM:
  For J=0 TO 15
    For I=0 TO 7
     HSerout [BAD[I]]
    Next  i
  NEXT J
    For J=0 TO 15
    For I=0 TO 7
     HSerout [BAD1[I]]
    Next  i
  NEXT J
RETURN

BPARABRISAS:
  For J=0 TO 10
   For I=0 TO 7
     HSerout [BAP[I]]
   Next  i
  NEXT J
   For J=0 TO 10
   For I=0 TO 7
     HSerout [BAP1[I]]
   Next  i
  NEXT J
RETURN

BRAZO:
  For J=0 TO 10
   For I=0 TO 7
     HSerout [MB[I]]
   Next  i
  NEXT J
   For J=0 TO 10
   For I=0 TO 7
     HSerout [MB1[I]]
   Next  i
  NEXT J
RETURN
BPARABATRAS:
  For J=0 TO 10
   For I=0 TO 7
  HSerout [BAMP[I]]
   Next  i
  NEXT J
   For J=0 TO 10
```

```
    For I=0 TO 7
      HSerout [BAMP1[I]]
    Next  i
  NEXT J
RETURN

BADELANTE:
  For J=0 TO 10
    For I=0 TO 7
      HSerout [BD[I]]
    Next  i
  NEXT J
    For J=0 TO 10
    For I=0 TO 7
      HSerout [BD1[I]]
    Next  i
  NEXT J
RETURN

'+++++++++++++++++ritmos++++++++++++++

RITMO1:


  w=0
  while w<40
    gosub batras
  w=w+1
  wend
    w=0
  while w<10
    gosub batrasm
  w=w+1
  wend


  For J=0 TO 10
    For I=0 TO 7
      HSerout [Bam1[I]]
    Next  i
  NEXT J
    w=0
  while w<40
    gosub bcadera
  w=w+1
  wend
    w=0
  while w<5
    gosub bcabeza
  w=w+1
  wend
   w=0
  while w<20
    gosub bcadera
  w=w+1
  wend
    w=0
  while w<10
    gosub seteo
  w=w+1
  wend
    w=0
  while w<18
    gosub badelantem
  w=w+1
  wend
    w=0
  while w<37
    gosub bparabrisas
  w=w+1
  wend
    w=0
  while w<42
    gosub brazo
  w=w+1
  wend


RETURN
```

Pierna Derecha

```
****************

INCLUDE "modedefs.bas"  ' Set receive register to receiver enabled

DEFINE HSER_RCSTA 90h   ' Set transmit register to transmitte0r enabled

DEFINE HSER_TXSTA 20h   ' Set baud rate

DEFINE HSER_BAUD 2400   ' Set SPBRG directly (normally set by HSER_BAUD)

DEFINE HSER_SPBRG 25
'REPRODUCCION DE DATOS CAPTURADOS



@ device XT_osc;  utiliza cristal externo 4mhz

@ device wdt_off;

@ device mclr_off;

@ device bod_off;

cmcon =7;

DEFINE OSC 4

TRISA=%00000000

TRISB=%00000000

'----------variables

I VAR BYTE

J VAR BYTE

W VAR BYTE

'----------tramas

RS VAR BYTE[8]

RS1 VAR BYTE[8]

CA VAR BYTE[8]

CB1 VAR BYTE[8]

CB2 VAR BYTE[8]

CC VAR BYTE[8]

CI VAR BYTE[8]


CI1 VAR BYTE[8]

CII1 VAR BYTE[8]

CI2 VAR BYTE[8]

CI3 VAR BYTE[8]

CD VAR BYTE[8]

PA VAR BYTE[8]

LR1 VAR BYTE[8]

D VAR BYTE[8]

P VAR BYTE[8]

PR VAR BYTE[8]

PR1 VAR BYTE[8]


'-------datos

Rs[0]=5'$5
'SETEO

rs[1]=93'$130

rs[2]=110'$120

rs[3]=180'$65

rs[4]=80'$65

rs[5]=130'$130

rs[6]=88'5

Rs[7]=255'$255

RS1[0]=5'5     SET

RS1[1]=93'0

RS1[2]=110'115

RS1[3]=180'65

RS1[4]=80'102

RS1[5]=0'0
```

RS1[6]=213'42

RS1[7]=255'255

CA[0]=5'5 'RODILLAS

CA[1]=93'

CA[2]=140'

CA[3]=10'

CA[4]=95'

CA[5]=0'

CA[6]=88'

CA[7]=255'

CC[0]=5'5

CC[1]=108'130

CC[2]=140'150

CC[3]=10'65

CC[4]=85'110

CC[5]=0'134

CC[6]=93'84

CC[7]=255'255

CI[0]=5'5 'CAMINA IZQUIERDA

CI[1]=60'83 ojo

CI[2]=125'150

CI[3]=170'65

CI[4]=85'110

CI[5]=0'134

CI[6]=190'84

CI[7]=255'255

CI1[0]=5'5 'DERECHA

CI1[1]=0'130

CI1[2]=110'150

CI1[3]=180'65 5+

CI1[4]=80'110

CI1[5]=110'134

CI1[6]=230'84

CI1[7]=255'255

CI2[0]=5'5 'IZQUIERDA

CI2[1]=0'130

CI2[2]=110'150

CI2[3]=180'65

CI2[4]=80'110

CI2[5]=150'134

CI2[6]=15'84

CI2[7]=255'255

CII1[0]=5'5 'CAMINA DERECHA

CII1[1]=0'130

CII1[2]=110'150

CII1[3]=180'65

CII1[4]=80'110

CII1[5]=125'134

CII1[6]=245'84

CII1[7]=255'255

CI3[0]=5'5 'SETEO SEGUN EL PIE

CI3[1]=0'130

CI3[2]=110'150

CI3[3]=180'65

CI3[4]=80'110

CI3[5]=130'134

CI3[6]=250'84

CI3[7]=255'255

CD[0]=5'5 'CAMINA DERECHA

CD[1]=103'130

CD[2]=125'140'150

CD[3]=170'65

CD[4]=85'110

CD[5]=0'134

CD[6]=233'84

CD[7]=255'255

CB1[0]=5'5 CADERA

CB1[1]=85'130

CB1[2]=110'150

CB1[3]=180'65

CB1[4]=80'110

CB1[5]=0'134 125

CB1[6]=205'84

CB1[7]=255'255

CB2[0]=5'5

CB2[1]=103'130

CB2[2]=110'150

CB2[3]=180'65

CB2[4]=80'110

CB2[5]=0'134 140

CB2[6]=223'84

CB2[7]=255'255

PA[0]=5'$5 P ALZADA

PA[1]=168'$130

PA[2]=110'$120

PA[3]=180'$65

PA[4]=80'$65

PA[5]=130'$130

PA[6]=163'5

PA[7]=255'$255

LR1[0]=5'$5 'SETEO

Lr1[1]=113'$130

Lr1[2]=110'$120

Lr1[3]=180'$65

Lr1[4]=80'$65

Lr1[5]=0'$130

Lr1[6]=233'5

LR1[7]=255'$255

D[0]=5'$5 'SETEO

D[1]=168'$130

D[2]=110'$120

D[3]=180'$65

D[4]=80'$65

D[5]=0'$130

D[6]=33'5

D[7]=255'$255

P[0]=5'$5 'SETEO

P[1]=93'$130

P[2]=210'$120

P[3]=180'$65

P[4]=80'$65

P[5]=130'$130

P[6]=188'5

P[7]=255'$255

PR[0]=5'$5 'SETEO

PR[1]=93'$130

PR[2]=210'$120

PR[3]=180'$65

PR[4]=145'$65

PR[5]=130'$130

PR[6]=253'5

PR[7]=255'$255

PR1[0]=5'$5 'SETEO

```
PR1[1]=93'$130
PR1[2]=160'$160
PR1[3]=180'$65
PR1[4]=125'$65
PR1[5]=130'$130
PR1[6]=183'5
PR1[7]=255'$255
'----------seteo----------
FOR J=0 TO 120
 GOsub SETEO
next j
'**********programa**********
INICIO:

   w=0
   while w<25
   gosub ladosr1
   w=w+1
   wend

'   w=0
'   while w<8
'   gosub rodillas
'   w=w+1
'   wend

   w=0
   while w<10
   gosub camina1
   w=w+1
   wend
    w=0
   while w<10
   gosub lados
   w=w+1
   wend
   w=0
   while w<20
   gosub CAMINA1
   w=w+1
   wend
    GOTO INICIO
 END
'-**********subrutinas**********
SETEO:'CORRECTO
    For I=0 TO 7
      HSerout [RS[I]]
    Next  i
RETURN

CAMINA1:
   FOR J=0 TO 14
    For I=0 TO 7
      HSerout [CII1[I]]
    Next  i
    NEXT J
    W=0
  WHILE W<7
    For I=0 TO 7
      HSerout [CI2[I]]
    Next  i

  W=W+1
  WEND
     W=0
  WHILE W<7
   For I=0 TO 7
      HSerout [CA[I]]
    Next  i

  W=W+1
  WEND
   RETURN
LADOSR11:
  FOR J=0 TO 40
   For I=0 TO 7
     HSerout [RS[I]]
   Next  i
  NEXT J
    FOR J=0 TO 20
   For I=0 TO 7
     HSerout [LR1[I]]
   Next  i
  NEXT J
  FOR J=0 TO 20
   For I=0 TO 7
     HSerout [RS[I]]
   Next  i
  NEXT J
RETURN
LADOSR1:
  FOR J=0 TO 15
   For I=0 TO 7
  HSerout [RS[I]]
   Next  i
  NEXT J
    FOR J=0 TO 15
   For I=0 TO 7
     HSerout [LR1[I]]
   Next  i
  NEXT J
RETURN

LADOS:'CORRECTO
  FOR J=0 TO 30
     GOSUB SETEO
  NEXT J
    FOR J=0 TO 15
  For I=0 TO 7
     HSerout [CC[I]]
  Next  i
  NEXT J
   FOR J=0 TO 15
     GOSUB SETEO
  NEXT J
RETURN
SETEO1:
'CORRECTO
    For I=0 TO 7
      HSerout [RS1[I]]
    Next  i
RETURN
RODILLAS:
'CORRECTO
   FOR J=0 TO 10
      GOSUB
SETEO1
   NEXT J
     FOR J=0 TO 10
   For I=0 TO 7
     HSerout [CA[I]]
   Next  i
   NEXT J
   FOR J=0 TO 10
      GOSUB
SETEO1
   NEXT J
  RETURN
 End
```

Pierna Izquierda

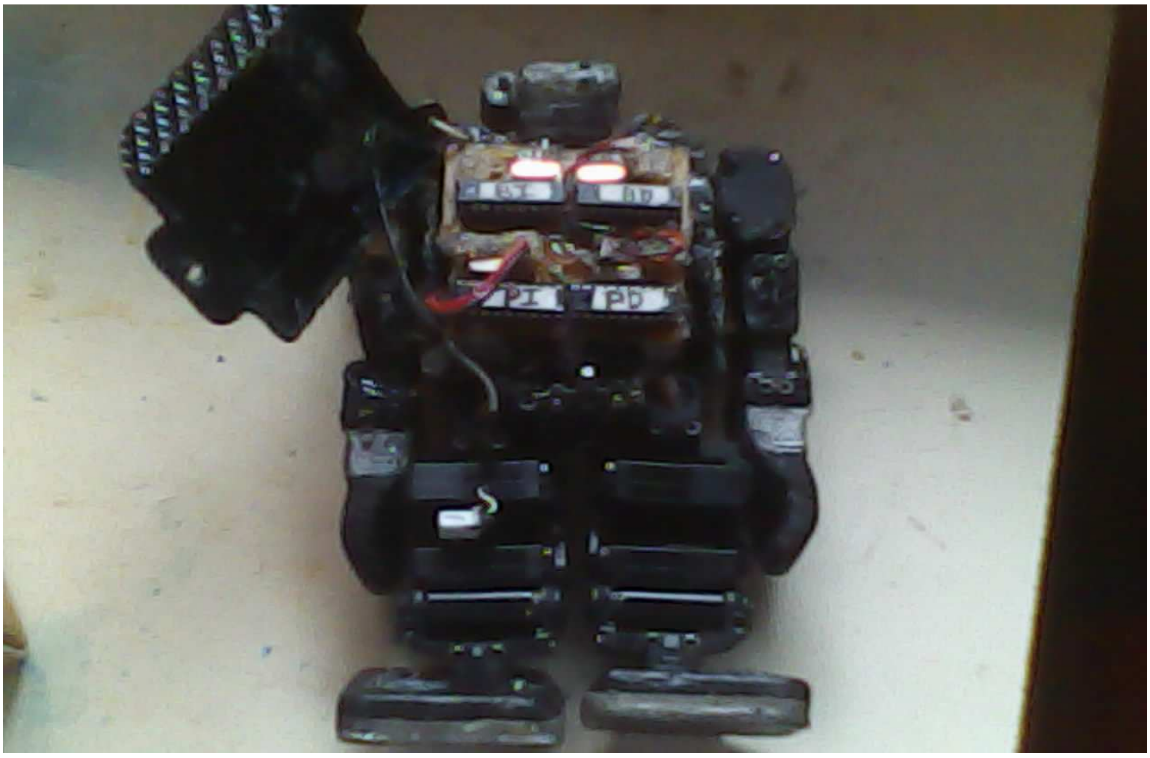
```
'*************************

INCLUDE "modedefs.bas"  ' Set receive register to receiver enabled

DEFINE HSER_RCSTA 90h   ' Set transmit register to transmitte0r enabled

DEFINE HSER_TXSTA 20h   ' Set baud rate

DEFINE HSER_BAUD 2400   ' Set SPBRG directly (normally set by HSER_BAUD)

DEFINE HSER_SPBRG 25
'REPRODUCCION DE DATOS CAPTURADOS



@ device XT_osc;  utiliza cristal externo 4mhz

@ device wdt_off;

@ device mclr_off;

@ device bod_off;

cmcon =7;

DEFINE OSC 4



I VAR BYTE

J VAR BYTE

W VAR BYTE



RS VAR BYTE[8]

RS1 VAR BYTE[8]

CA VAR BYTE[8]

CB1 VAR BYTE[8]

CB2 VAR BYTE[8]

CC VAR BYTE[8]

CI VAR BYTE[8]

CI1 VAR BYTE[8]

CI2 VAR BYTE[8]

CII2 VAR BYTE[8]

CI3 VAR BYTE[8]

CD VAR BYTE[8]

PA VAR BYTE[8]

LR1 VAR BYTE[8]

D VAR BYTE[8]

P VAR BYTE[8]

PR VAR BYTE[8]

PR1 VAR BYTE[8]



TRISA=%00000000

TRISB=%00000000



RS[0]=5'$5
SETEO

rS[1]=125'$125

rS[2]=140'$140

rS[3]=190'$190

rS[4]=190'$190

rS[5]=125'$130

rS[6]=10'15

RS[7]=255'$255



RS1[0]=5'$5      SET

RS1[1]=125'$130

RS1[2]=140'$140

RS1[3]=190'$80

RS1[4]=190'$200  180

RS1[5]=0'$130

RS1[6]=140'35

RS1[7]=255'$255



CA[0]=5'5

CA[1]=125'125

CA[2]=120'100

CA[3]=220'200

CA[4]=185'154

CA[5]=0'126

CA[6]=145'200

CA[7]=255'255



CC[0]=5'5
LADOS

CC[1]=115'125

CC[2]=120'100

CC[3]=220'200

CC[4]=185'154

CC[5]=0'126

CC[6]=135'200

CC[7]=255'255



CI[0]=5'5
CAMINA IZQ

CI[1]=115

CI[2]=120'140

CI[3]=220'190

CI[4]=185'190

CI[5]=0'125

CI[6]=135

CI[7]=255'255



CI1[0]=5'5       DER

CI1[1]=0'125

CI1[2]=140'100

CI1[3]=190'200

CI1[4]=190'154

CI1[5]=105'126

CI1[6]=120'200

CI1[7]=255'255

CI2[0]=5'5       IZQ

CI2[1]=145'125

CI2[2]=140'140'100

CI2[3]=190'190'200

CI2[4]=190'190'154

CI2[5]=145'126

CI2[6]=50'200

CI2[7]=255'255



CII2[0]=5'5
CAMINA

CII2[1]=0'125

CII2[2]=140'140'100

CII2[3]=190'190'200

CII2[4]=190'190'154

CII2[5]=130'126

CII2[6]=145'200

CII2[7]=255'255



CI3[0]=5'5
SETEO PIE

CI3[1]=0'125

CI3[2]=140'100

CI3[3]=190'200

CI3[4]=190'154

CI3[5]=125'126

CI3[6]=140'200

CI3[7]=255'255



CD[0]=5'5
CAMINA   DERECHA

CD[1]=135'125

CD[2]=120'100

CD[3]=220'200

CD[4]=185'154

CD[5]=0'126

CD[6]=155'200
```

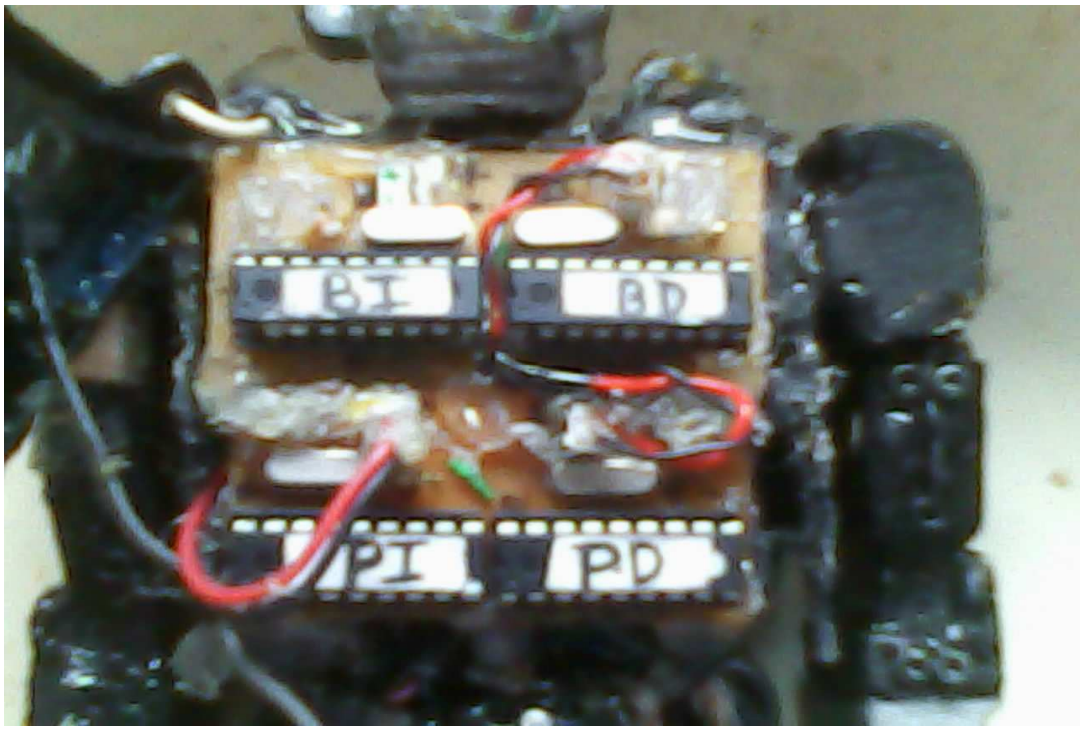
```
CD[7]=255'255


CB1[0]=5'5
CADERAS
CB1[1]=115'125
CB1[2]=140'100
CB1[3]=190'200
CB1[4]=190'154
CB1[5]=0'126    120
CB1[6]=130'200
CB1[7]=255'255


CB2[0]=5'5
CB2[1]=130'125
CB2[2]=140'140'100
CB2[3]=190'190'200
CB2[4]=190'190'154
CB2[5]=0'126    140
CB2[6]=145'200
CB2[7]=255'255


PA[0]=5'$5      P
ALZADA
PA[1]=70'$125
PA[2]=140'$140
PA[3]=190'$190
PA[4]=190'$190
PA[5]=125'$130
PA[6]=210'15
PA[7]=255'$255


LR1[0]=5'$5
SETEO
LR1[1]=105'$125
LR1[2]=140'$140
LR1[3]=190'$190
LR1[4]=190'$190
LR1[5]=0'$130
LR1[6]=120'15
LR1[7]=255'$255


D[0]=5'$5       SETEO
D[1]=50'$125
D[2]=140'$140
D[3]=190'$190
D[4]=190'$190
D[5]=0'$130
D[6]=65'15
D[7]=255'$255


P[0]=5'$5       SETEO
P[1]=125'$125
P[2]=40'$140
P[3]=190'$190
P[4]=190'$190
P[5]=125'$130
P[6]=165'15
P[7]=255'$255


PR[0]=5'$5
SETEO
PR[1]=125'$125
PR[2]=40'$140
PR[3]=190'$190
PR[4]=125'$190
PR[5]=125'$130
PR[6]=100'15
PR[7]=255'$255


PR1[0]=5'$5
SETEO
PR1[1]=125'$125
PR1[2]=90'$90
PR1[3]=190'$190
PR1[4]=145'$190
PR1[5]=125'$130
PR1[6]=170'15
PR1[7]=255'$255
'----------seteo----------
FOR J=0 TO 120
 GOsub SETEO
next j
'**********programa**********
INICIO:
    w=0
  while w<25
  gosub ladosr1
  w=w+1
  wend
  '   w=0
'   while w<8
'   gosub rodillas
'   w=w+1
'   wend


  w=0
  while w<10
  gosub camina1
  w=w+1
  wend


  w=0
  while w<10
  gosub lados
  w=w+1
  wend
    w=0
  while w<20
  gosub CAMINA1
  w=w+1
  wend
   GOTO INICIO
GOTO INICIO
END
'**********subrutinas**********
SETEO:'CORRECTO
   For I=0 TO 7
     HSerout [RS[I]]
   Next  i
RETURN
CAMINA1:
   W=0
  WHILE W<7
   For I=0 TO 7
     HSerout [CI1[I]]
   Next  i
  W=W+1
  WEND


 W=0
  WHILE W<7
   For I=0 TO 7
    HSerout [CA[I]]
   Next  i
 W=W+1
  WEND


  FOR J=0 TO  14
    For I=0 TO 7
      HSerout [CII2[I]]
    Next  i
```

```
  NEXT J
RETURN

LADOSR11:
FOR J=0 TO 20
   For I=0 TO 7
      HSerout [LR1[I]]
    Next  i
 NEXT J
FOR J=0 TO 20
   For I=0 TO 7
      HSerout [RS[I]]
    Next  i
 NEXT J
FOR J=0 TO 40
   For I=0 TO 7
      HSerout [RS[I]]
    Next  i
  NEXT J
RETURN

LADOSR1:
FOR J=0 TO 15
   For I=0 TO 7
      HSerout [LR1[I]]
    Next  i
  NEXT J
FOR J=0 TO 15
   For I=0 TO 7
      HSerout [RS[I]]
    Next  i
  NEXT J
RETURN
SETEO1:
'CORRECTO
   For I=0 TO 7
      HSerout [RS1[I]]
    Next  i
RETURN

'++++++++++RODILLAS++++++++++++++++

RODILLAS:
'CORRECTO
  FOR J=0 TO 10
    For I=0 TO 7
      HSerout [CA[I]]
    Next  i
  NEXT J

  FOR J=0 TO 20
     GOSUB SETEO1
  NEXT J
 RETURN
LADOS:'CORRECTO
 FOR J=0 TO 30
    GOSUB SETEO
 NEXT J
   FOR J=0 TO 15
 For I=0 TO 7
    HSerout [CC[I]]
 Next  i
NEXT J
  FOR J=0 TO 15
    GOSUB SETEO
NEXT J

RETURN
end
```

Anexo F. Implementación Electrónica.

Circuitos en el microrobot implementado

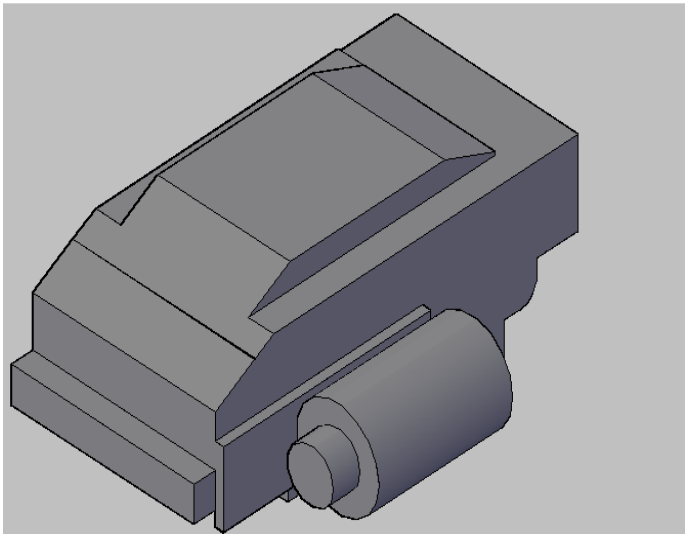




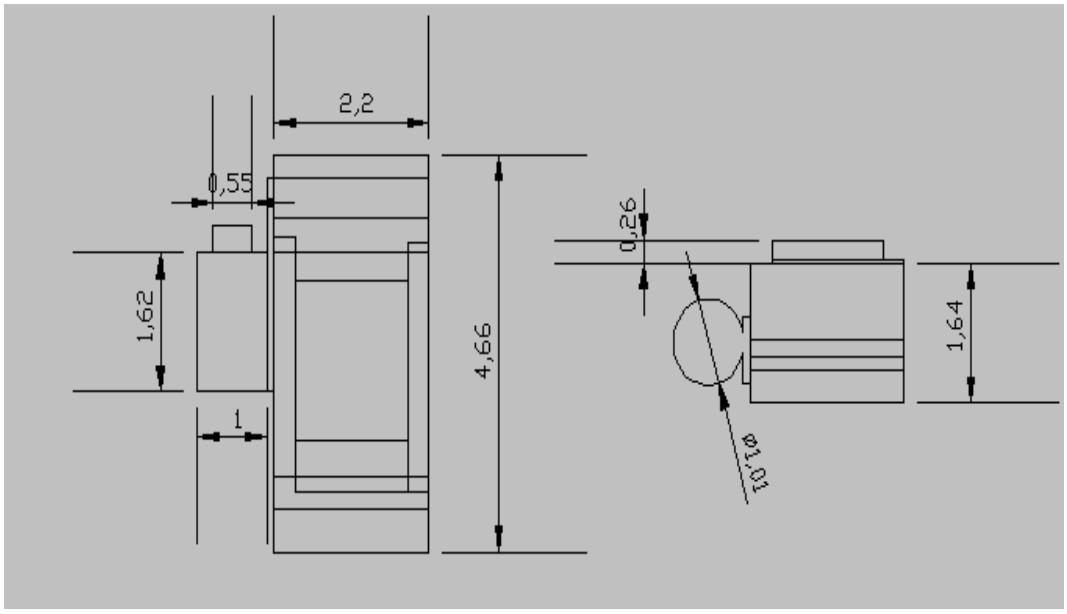
Anexo G. Diseño del microrobot bailarín en Autocad

ESTRUCTURA DEL ROBOT

Cabeza

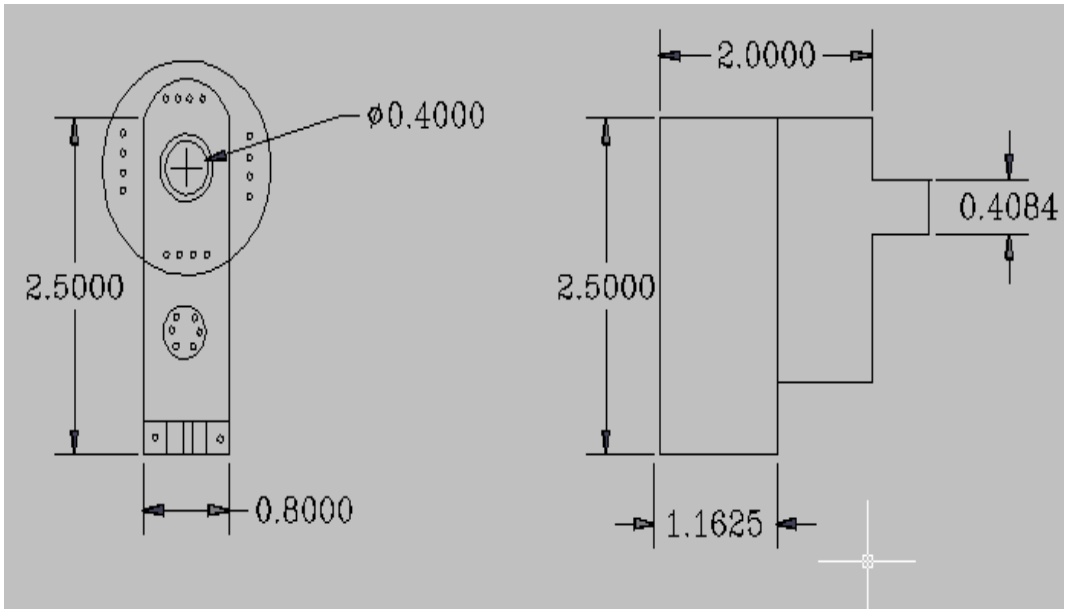


Cabeza en 3D



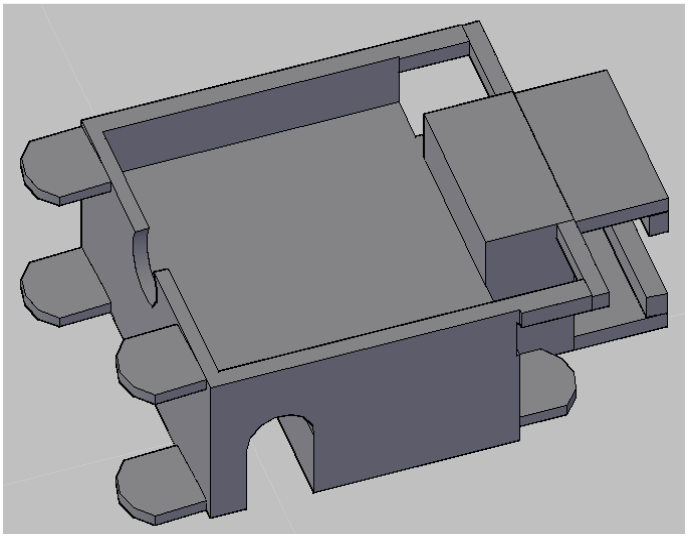
Acotaciones de la cabeza

Sujetadores del eje del Micromotor



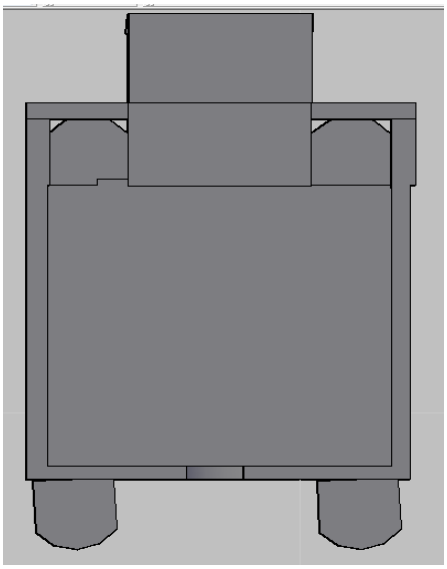
Acotación sujetadores del eje de los micromotores

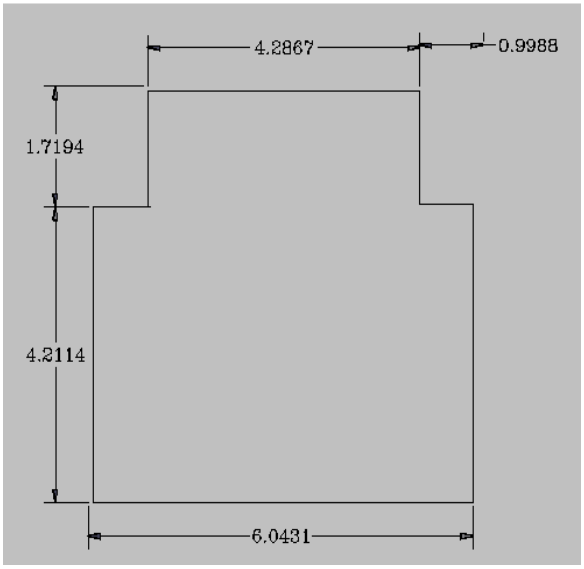
Cuerpo



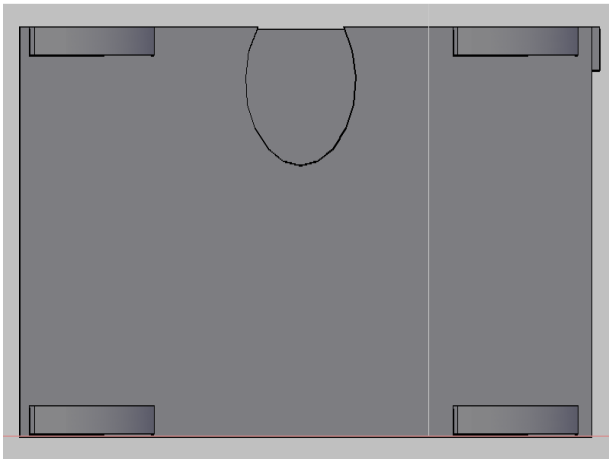
Vista del cuerpo en 3D

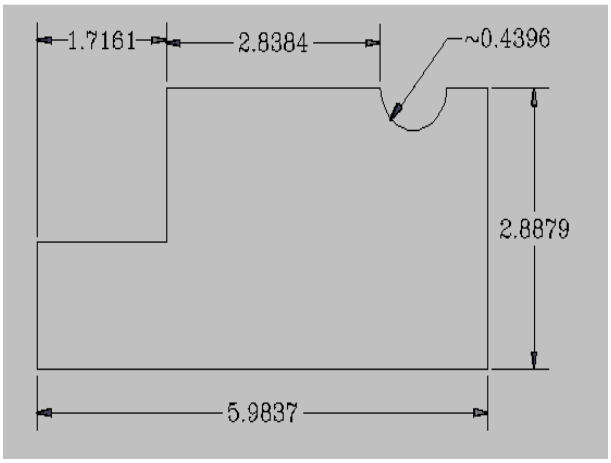
Vista superior del cuerpo





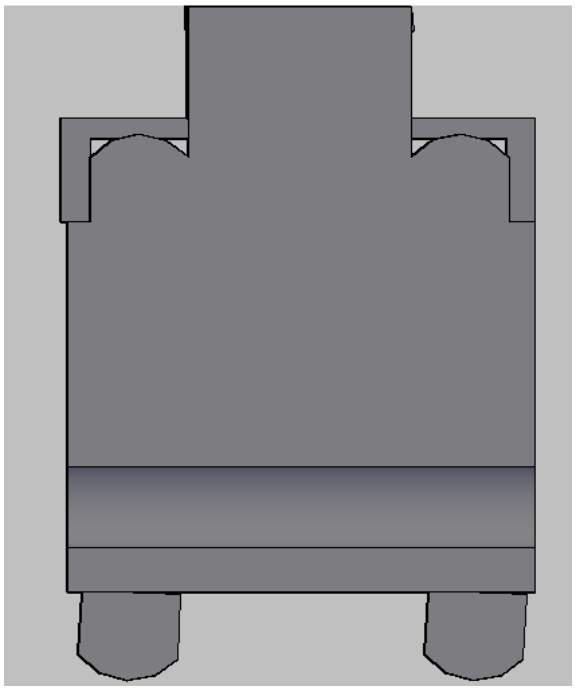
Vista Frontal del cuerpo

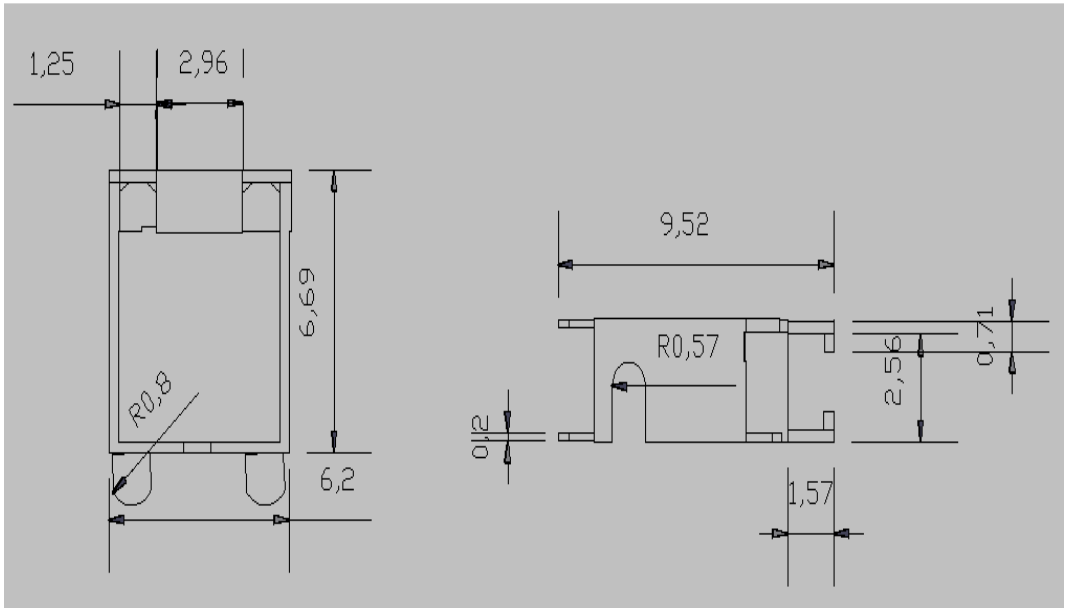




Acotacion

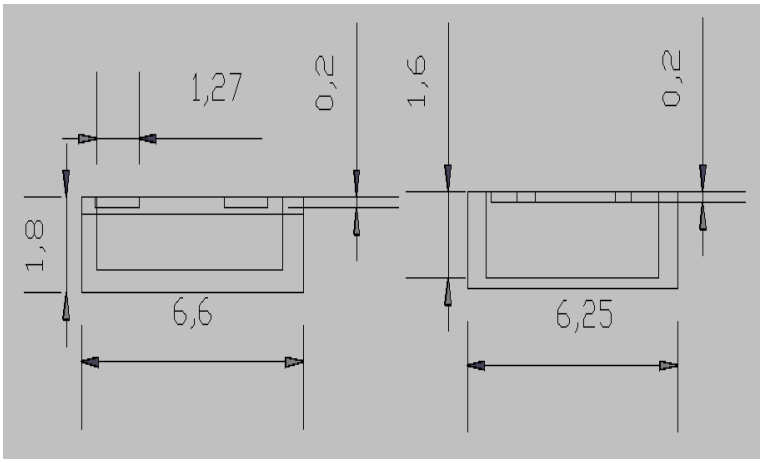
Vista lateral del cuerpo



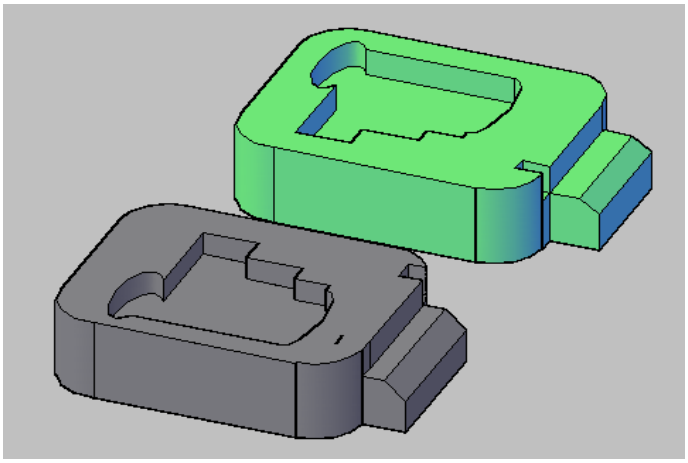


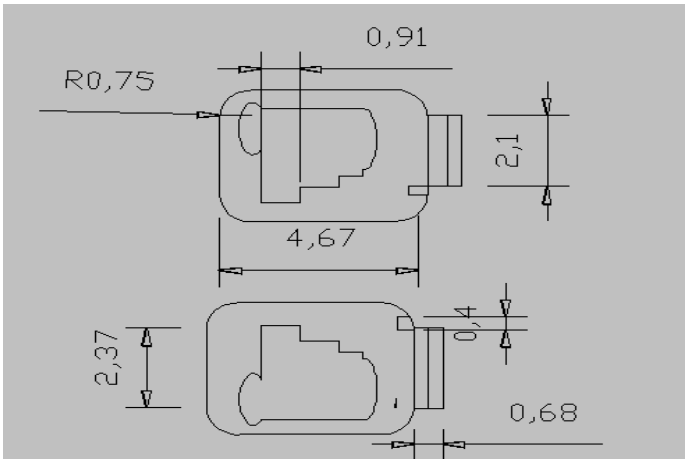
Acotación

Tapas del cuerpo



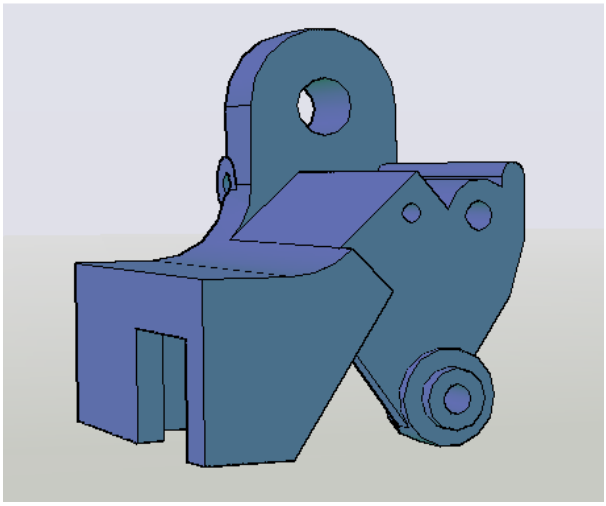
Vista en 3D de los Pies Derecho e Izquierdo





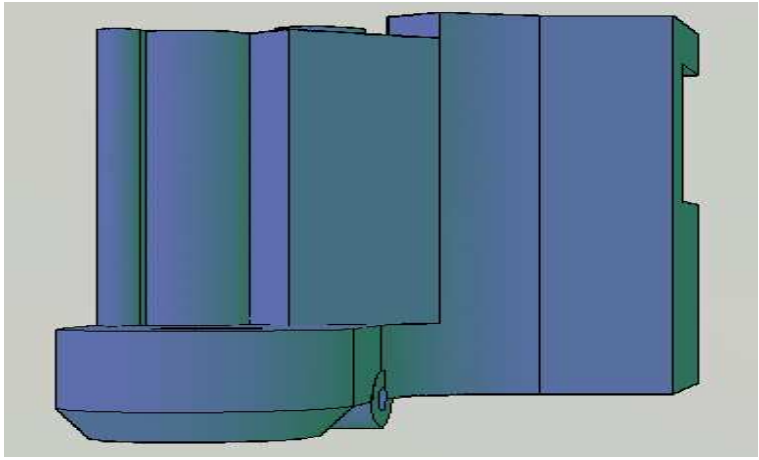
Acotacion de los pies

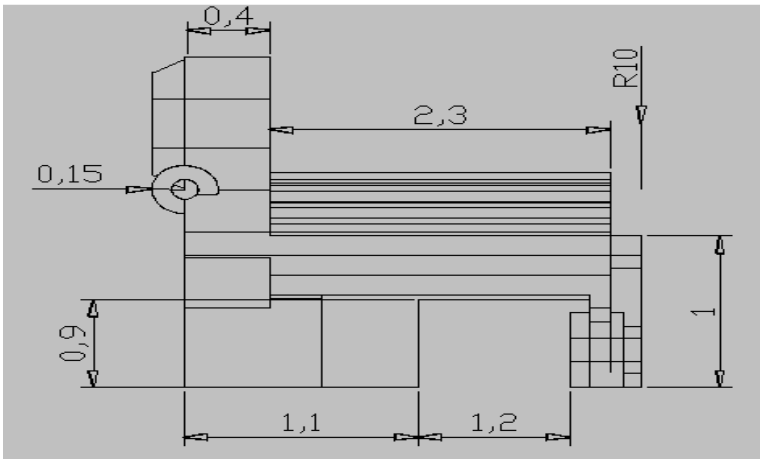
Pieza par la rodilla



Vista en 3D

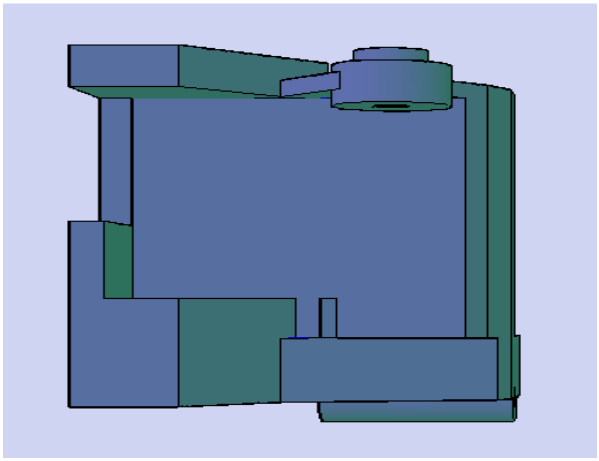
Vista superior



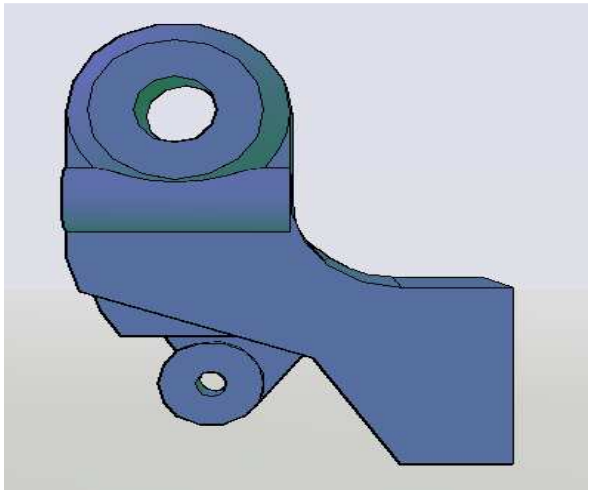


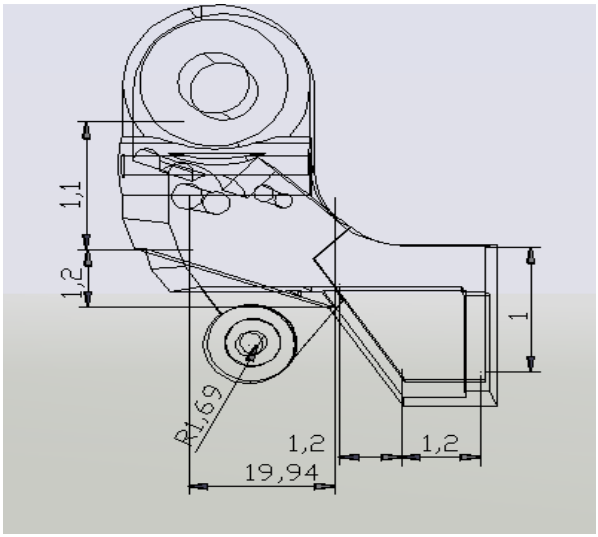
Acotación

Vista Inferior



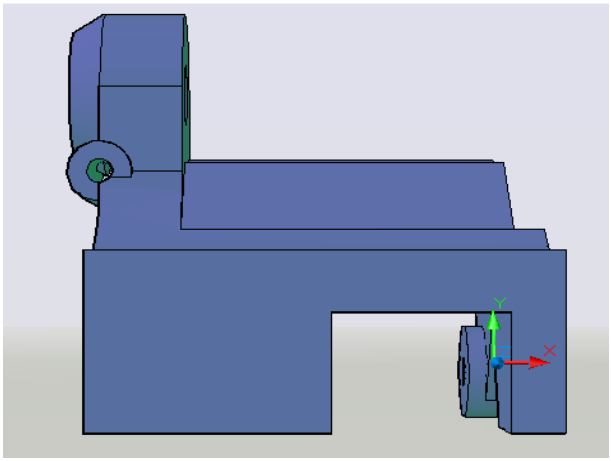
Vista frontal



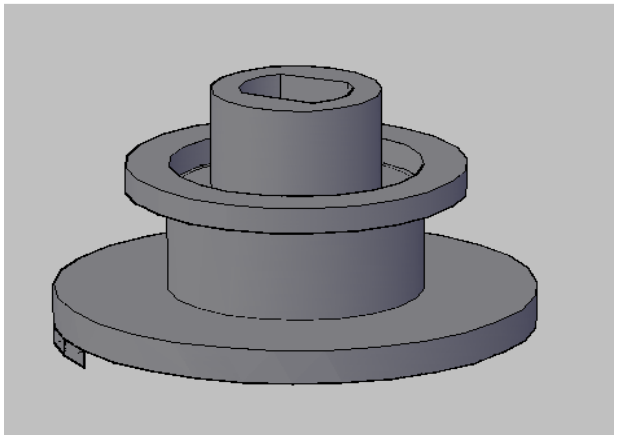


Acotación

Vista Frontal Derecha

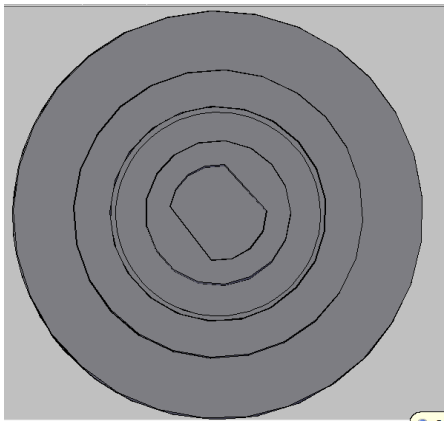


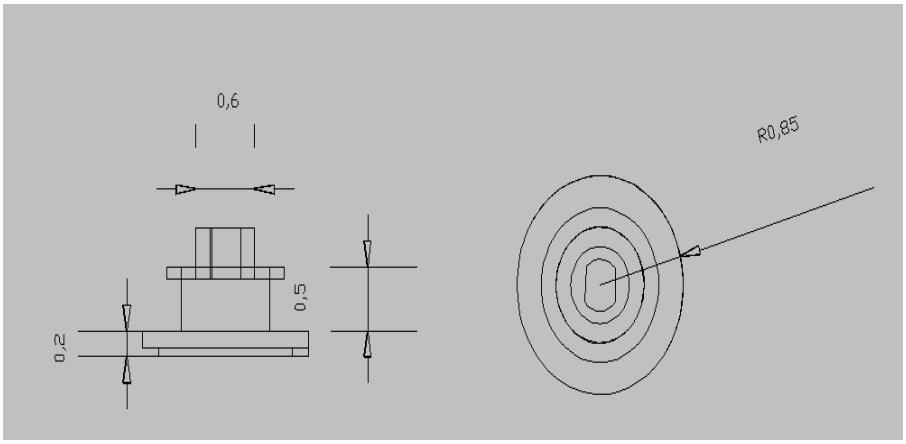
Sujetador del Brazo



Vista en 3D

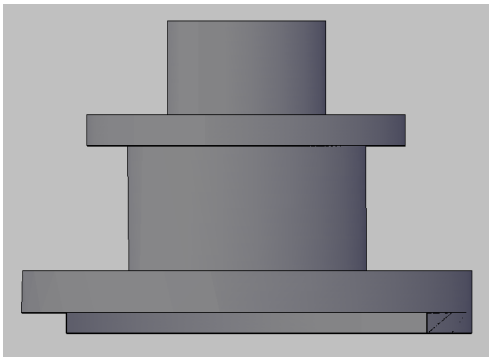
Vista Superior



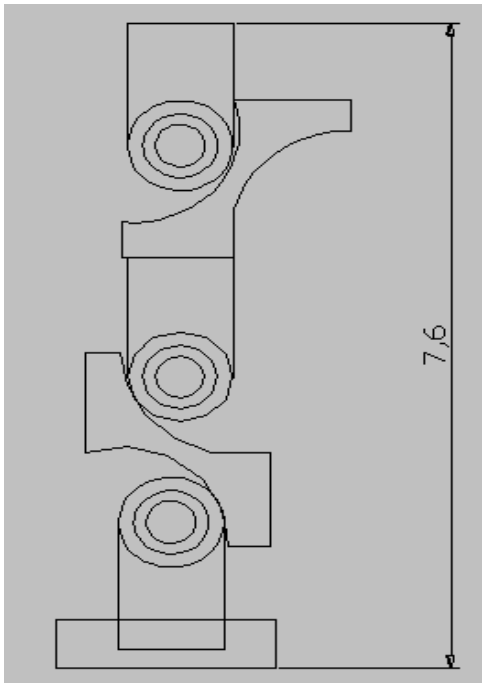


Acotacion

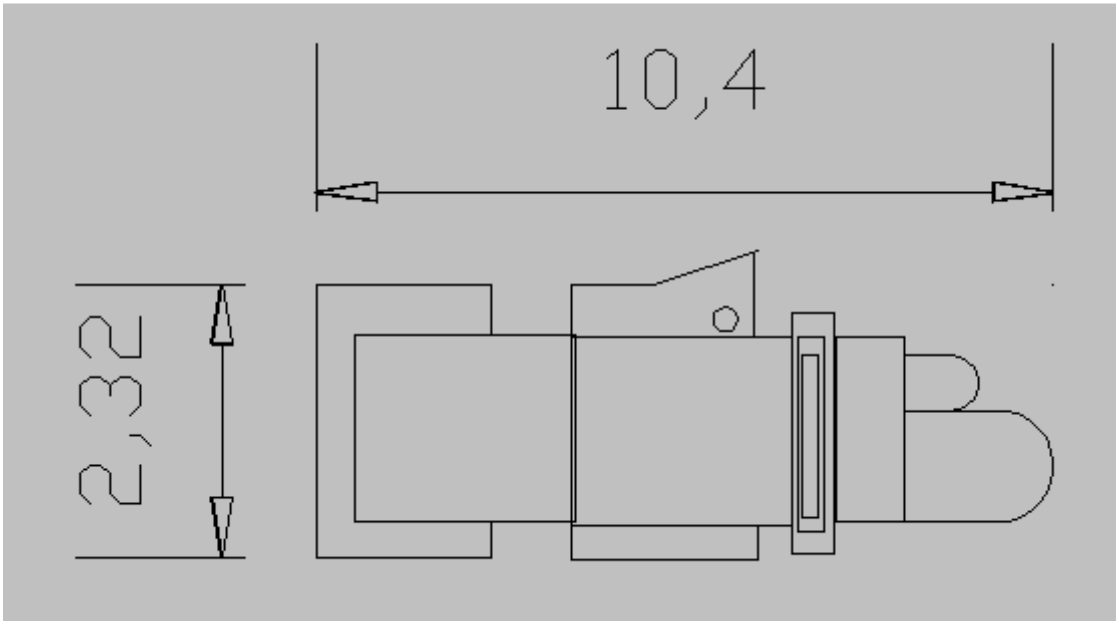
Vista Lateral



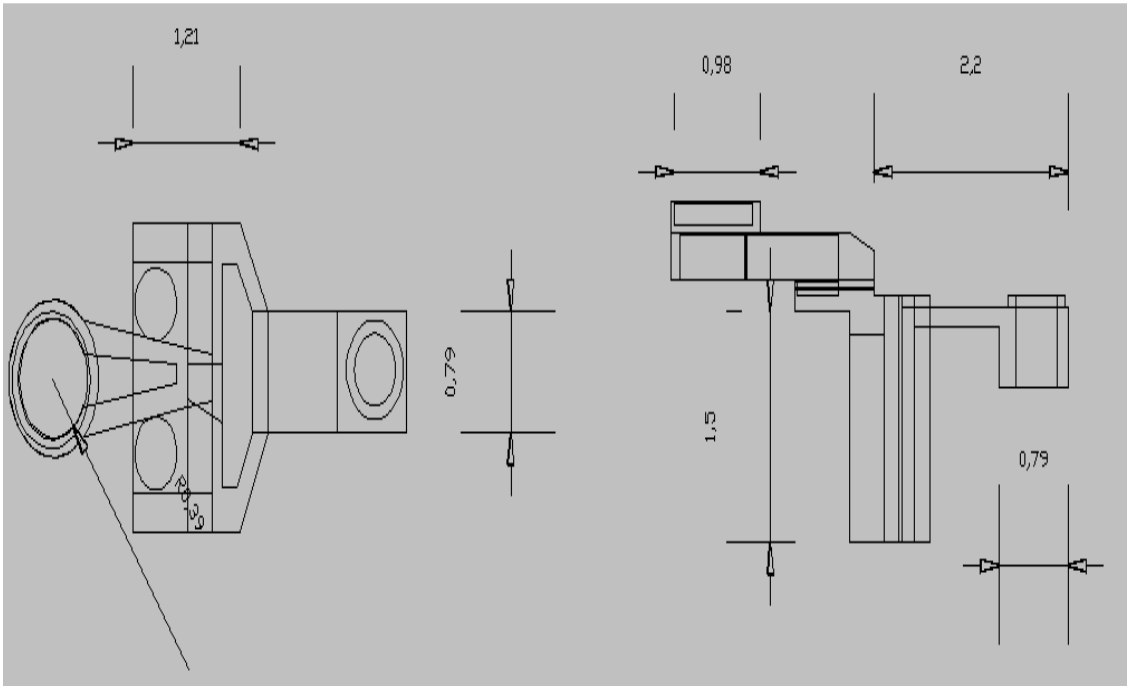
Pierna



Brazo

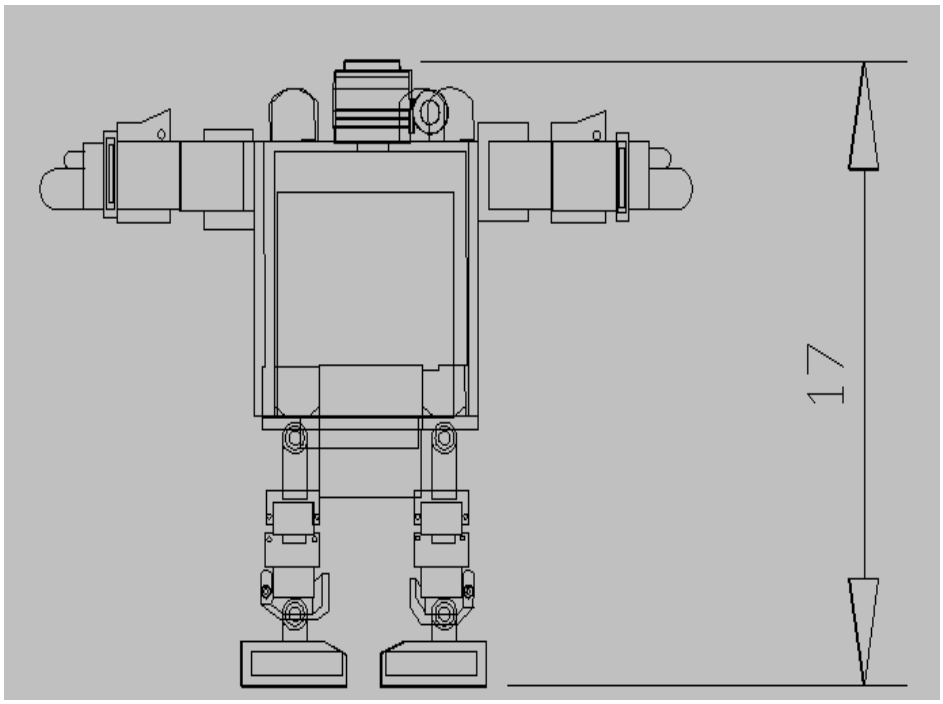


Unión de la pierna

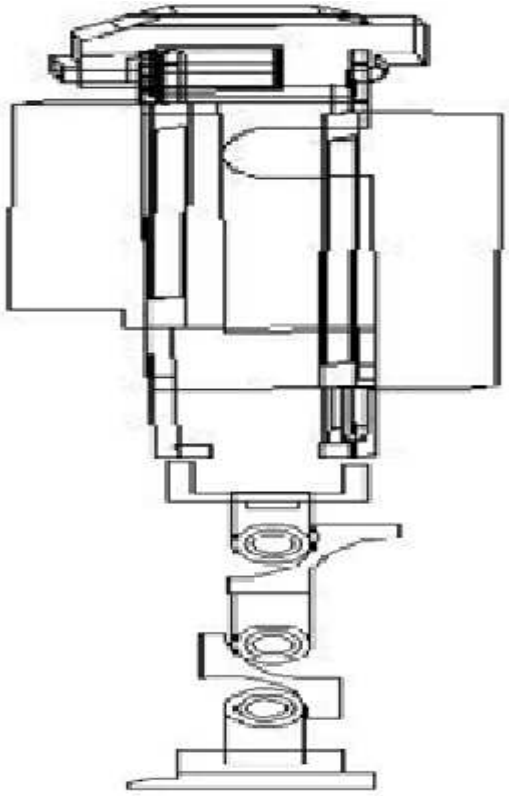


Esquema final del robot

Vista Frontal



Vista Lateral



BIBLIOGRAFÍA

LIBROS

- ANGULO, J.Mª. Robótica Práctica: tecnología y aplicaciones. 4ª ed. Madrid: Paraninfo, 1996.
- ANGULO, J.Mª; EUGENIO, Martín y ANGULO Ignacio. Microcontroladores Pic: la solución en un chip. Madrid: Paraninfo, 1997.
- ANGULO, J.Mª.; ANGULO, Ignacio. Microcontroladores Pic Diseño Y Aplicaciones: México: MC GRAW-HILL, 1997.
- ANGULO, J.Mª. Microbots: la aplicación más fascinante de los micro controladores. Madrid: Paraninfo, 1999.
- ANGULO, J.M.; ROMERO, S. ; ANGULO, I. Microbótica. Madrid: Paraninfo, 1999.
- BLAKE, R. Sistemas Electrónicos de Comunicaciones. 2da ed. Inglaterra: Thomson Learning, 2004.
- CHAPRA, CANALE. Métodos numéricos para Ingenieros. 3ª.ed. México: McGraw Hill, 2001.
- KRETZINANN Y ANGULO. Electrónica y Automática Aplicadas a la Industria. 7ª ed. Madrid: Paraninfo, 1979.
- LÓPEZ-SÁNCHEZ, J. Sistemas Electrónicos de Interfaz para Actuadores Piezoeléctricos. Tesis Doctoral en robótica.Barcelona.Universidad de Barcelona. Departamento de Electrónica, 2001.
- TAYLOR, P.M. Control Robótico. España: CEAC, 1992.
- TORRES, Fernando; POMARES, Jorge y GIL, Pablo Santiago. Robots y Sistemas Sensoriales. New Jersey: Prentice Hall, 2002.

BIBLIOGRAFÍA DE INTERNET

Robótica

http:// www.microbotica.es

2008/12/20

http://www.jeuazarru.com/html/microbots.html

2008/12/22

http://www.electronicaestudio.com

2009/01/10

http://loslocosproyectos.blogspot.com

2009/01/10

http://www.robozes.com: página de RBZ-Robótica

2009/02/18

http://www.ecojoven.com/02102000/bichos.html

2009/02/18

http://robots.iespana.es/robots/sitios_favoritos.htm

2009/03/10

http://www.mikroelektronika.co.yu

2009/03/10

Microcontroladores

http://www.microchip.com

2009/01/10

http://www.galeon.com/microchip/

2009/02/18

http://www.todopic.com.ar

2009/02/18

Micromotores

http://www.raidentech.com/microservos.html

2008/12/21
http://www.tienda2000.com/servos.htm
2008/12/21
http://www.hitecrobotics.com
2008/12/21
http://www.mks-servo.com.tw
2008/12/21