UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO Y CONSTRUCCIÓN DE UN ROBOT HUMANOIDE

T E S I S

que para obtener el grado de

MAESTRO EN INGENIERÍA ELÉCTRICA

P R E S E N T A

FELIPE DE JESÚS PERALTA ZARATE

DIRECTOR DE TESIS
DR. JOSÉ ABEL HERRERA CAMACHO
CODIRECTOR DE TESIS
M.I. JUAN CARLOS ROA BEIZA

Ciudad Universitaria 2009

JURADO ASIGNADO:

Presidente: Dr. Arteaga Pérez Marco

Secretario: Dr. Savage Carmona Jesús

Vocal: Dr. Herrera Camacho José Abel

1er. Suplente: MI. Santiago Cruz Lauro

2do. Suplente: MI. Roa Beiza Juan Carlos

Lugar o lugares donde se realizó la tesis:

México DF.

TUTOR DE TESIS:
Dr. Herrera Camacho José Abel

_____

COTUTOR DE TESIS:
MI. Roa Beiza Juan Carlos

_____

INDICE DE FIGURAS PAG

CAPITULO 1

CAPITULO 2

INTRODUCCIÓN

Antecedentes

En México ya existen equipos de investigación en el área de robots humanoides, entre los cuales destacan “Pioneros México” de la Universidad Panamericana Campus Guadalajara, quienes como su nombre lo indica, fueron los primeros en iniciar investigaciones con robots humanoides. En México, actualmente su proyecto consiste en dos robots humanoides de 56cm de altura que juegan fútbol de manera autónoma. Se guían por medio de una cámara la cual detecta una pelota en la cancha de fútbol y caminan como un ser humano para tratar de jugar fútbol. Uno de los robots es el delantero y el otro el portero. El equipo “Pioneros México” consta de 8 integrantes. El Instituto Tecnológico de Monterrey con su equipo “Bogobots” empezaron por utilizar un robot comercial llamado Robonova 1, en la actualidad ya han construido sus prototipos con 20 grados de libertad. El equipo se compone de 10 integrantes y un líder de equipo.

Por otra parte la UNAM empezó investigaciones con los robots Tafary, Miztli y Riu (robots de la marca Hitec, modelo Robonova-1 con adaptaciones de cámara, sensores de inclinación y giroscopio), en la actualidad ya han construido un prototipo propio basándose en la tarjeta de control del robot Robonova-1 con 20 grados de libertad, El equipo cuenta con 5 integrantes.

Para la parte de visión la mayoría de los equipos utilizan una plataforma de desarrollo denominada CMUCam3[2], la cual es de uso comercial dedicada a aplicaciones de visión.

Existe un proyecto internacional llamado RoboCup, este tiene como objetivo promover a través de competencias integradas por robots autónomos la investigación y educación sobre inteligencia artificial. El reto de este proyecto es, que para el año 2050 se haya diseñado un equipo de robots humanoides autónomos que le puedan ganar al equipo humano y que sea campeón mundial de fútbol.

En 2008 “Pioneros de México” no apareció en la lista de participantes seleccionados, para el torneo que se llevo acabo en Suzho China, ahora México presenta dos nuevos equipos “Bogobots” del Instituto Tecnológico de Monterrey Campus Estado de México y “PUMAS-UNAM” de la Universidad Nacional Autónoma de México. Las especificaciones del equipo Bogobots son las siguientes: para la categoría de Kid el equipo intentó utilizar un Robonova-I [4], debido a sus limitaciones tanto en hardware como en software decidieron ocupar servo Lynxmotion brackets [5]. Con ellos dotaron a su humanoide de 20 grados de libertad, cada servo es controlado por una tarjeta desarrollada también por Lynxmotion, la cual regula y proporciona los (PWM) requeridos para cada servo motor, la energía es suministrada por una pila de litio, como microcontrolador ocupa un Dspic, al cual llegan tanto las señales de video como las de los demás sensores. La parte de visión la realizan con una CMUcam3, con la cual aplicaron algoritmos de segmentación de color para poder hacer el reconocimiento. Con estos algoritmos son capaces de detectar líneas, marcas y caracterizar objetos, a la CMUcam3 se le adaptaron un arreglo de 2 servos de diseño propio y así la cámara podrá tener movilidad. Este mismo equipo comienza a hacer pruebas con la siguiente categoría que es la Teen size.

El equipo PUMAS-UNAM, quien empezó investigaciones con los robots humanoides comerciales Robonova-1, ha optado por la implementación de de su propio diseño mecánico con 20 grados de libertad, como control de motores de su humanoide ocupa la tarjeta de control del robot Robonova-1 y esta, para programarse, se basa en un lenguaje llamado RoboBasic. Al igual que los otros equipos mexicanos usan una CMUcam3 como parte del sistema de visión, como algoritmo hacen una segmentación de color básico en YCrCb [6]. Los sensores extras son un sensor de nivel y un giroscopio.

El uso de la CMUcam es muy socorrida por los equipos mexicanos, tiene como procesador principal un NXP LPC2106 conectado a un CMOS de corrimiento con salida digital, el procesador LPC2106 contiene un ARM7TDMI y

el código en C se puede desarrollar utilizando las herramientas GNU junto con un conjunto de bibliotecas de código abierto y programas de ejemplo.

Tiene una resolución de (352x288) sensor RGB, soportado por Windows y Linux, con 26 frames por segundo, librerías para manipulación de imágenes y color [2].

Otros equipos participantes del torneo RoboCup proporcionan su propia solución al problema de visión, así, el equipo de Osaka, utiliza un sensor de visión, una salida en RGB para el despliegue, con una entrada de video análoga no dan más especificaciones acerca del hardware. El procesamiento lo realizan tomando la imagen y la transmiten a un controlador huésped. Previamente, un operador humano hace un índice de color para reconocer el objeto meta. Este método de asignación de colores RGB es un método de asignación directa. Combinar los datos de la imagen con el color índice, realizan el histograma con el color especificado índice (coordenadas polares) y calculan la mediana del punto de la región de orientación de la selección máximo en el histograma [9].

El equipo “NimBro” de la universidad Albert-Ludwigs de Freiburg, resuelve el problema de visión de la siguiente forma, usan tres IDS uEye UI-1226LE USB2.0 cámaras omnidireccionales aumentando el campo de visión a 180º, Las cámaras incorporan un 1/3 WVGA sensor CMOS, y están equipadas con lentes de ultra ancho angular, las cámaras recogen e interpretan las imágenes con una resolución de 752 × 480 píxeles a una velocidad de 32 frames por segundo. El amplio campo de visión central permite a los robots ver sus propios pies y objetos sobre el horizonte al mismo tiempo, las otras dos cámaras tienen un campo más estrecho de visión de aproximadamente el 95%. El software de visión detecta el balón, los postes de esquina, y otros actores, lo hace en el espacio YUV. La apuesta de otros equipos va en función de innovar y dar su propia solución a cada aspecto.

El Robonova-1 esta construido con 16 servos digitales del tipo HSR-8498HB, la tarjeta de control puede encargarse de hasta 26 servo motores, a

ello se agrega la posibilidad de integrar sensores como giroscopios, acelerómetros, sensores de nivel, flexible y sofisticada interfaz que permite al usuario personalizar ROBONOVA-I. [4]. Con estas características para equipos que no deseen involucrarse en el desarrollo de una plataforma propia resulta altamente interesante, en este caso el aporte a la competencia será en visión y planeación de movimientos.

Al igual que con los sistemas de visión la tendencia es que cada equipo desarrolle sus propios humanoides, así el equipo “NimBro” deseaba ocupar su poket pc para la operación central de su robot, para esto dotó a su arquitectura de la capacidad de poder cargar tal equipo [10]. El equipo Osaka propone una arquitectura propia con 25 grados de libertad en el humanoide. Robonova-1 y Kondo KHR-1, son plataformas que permiten dar un primer paso en la construcción de un humanoide pues se puede estudiar su estructura y de ahí partir para mejoras como lo hizo en Instituto Tecnológico de Monterrey y la UNAM.

Intentando no solo implementar dispositivos ya existentes en el mercado, con esa idea se llevó acabo la construcción de un humanoide de diseño propio, el cual tiene como ventajas el contar con dos grados de libertad más en las ingles que le permiten rotar sobre su propio eje, cada paso lateral tiene la posibilidad de virar al humanoide en ese sentido. Con 18 grados de libertad cada servo motor es controlado por una tarjeta de construcción propia teniendo como controlador un PIC18F452 que también esta conectado al módulo de visión.

Objetivo

Diseñar y construir un robot humanoide de tamaño pequeño capaz de caminar hacia adelante pararse virar hacia la derecha e izquierda, caminar hacia la derecha e izquierda y dotarlo de un sistema de visión simple.

Primeramente se diseñará la estructura mecánica con los grados de libertad apropiados y con una geometría que nos permita facilitar el control, así mismo se desarrollará una tarjeta para controlar los motores que en este caso serán 18 servomotores de corriente directa, también adjuntar el sistema de visión en el cual nos basaremos el sistema de visión Cmucam3, y por último conectar bloques y depurar errores.

Aunque en el mercado internacional existen robots similares, como lo es el Robonova-1, trataremos de construir el prototipo y sistemas de control con mejoras en el diseño como lo son más grados de libertad y mejoras al sistema de equilibrio, aparte de generar una bibliografía de consulta de origen nacional.

Descripción general

La tesis esta dividida en tres capítulos, el primer capítulo se llama Aspectos Generales Del Robot, se abordan temas como conceptos de la robótica, así como las partes fundamentales del robot como son: microcontroladores sensores (como el acelerómetro H48C y giroscopio electrónico PK3), servomotores, los tipos de baterías y el sistema de visión Cmucam3.

El segundo capítulo se llama Aspectos Básicos Del Robot, y se muestran temas sobre la locomoción humana, basado en este tema se desarrolló el sistema mecánico así como los criterios para evaluar las diferentes estructuras mecánicas, también se verifican las características de los motores de Dinamixel, una revisión de las características del microcontrolador elegido y las herramientas de desarrollo para trabajar con los microcontroladores PIC.

El tercer capítulo se llama Algoritmos e Implementación Física. Considerando que ya se tienen los conocimientos básicos de los dispositivos periféricos vistos en el capítulo dos, se explica la forma en la que se utilizan los mismos, la forma en que se conectaron, los algoritmos utilizados en cada uno de ellos ya sea el sistema de control de equilibrio y el sistema de visión. Así mismo se muestra el algoritmo para buscar una pelota de color naranja.

ASPECTOS GENERALES DEL ROBOT

1.1 Elementos De Los Robots

Un robot es un dispositivo ordinariamente electromecánico, que desempeña tareas automáticamente, ya sea de acuerdo a la supervisión humana directa, a través de un programa predefinido o siguiendo un conjunto de reglas generales, utilizando técnicas de inteligencia artificial. Generalmente estas tareas sustituyen, asemejan o extienden el trabajo humano, como ensamble en líneas de manufactura, manipulación de objetos pesados o peligrosos, trabajo en el espacio, etc. Comúnmente, el trabajo "inteligente" que realiza un robot es controlado por una computadora o un microcontrolador ejecutando un programa [11].

Los robots hoy en día han pasado a formar parte de nuestra vida cotidiana tanto en la industria como en nuestra vida personal, existen máquinas automatizadas y robots en muchos lugares que nos facilitan el trabajo, que van desde un cajero automático hasta brazos robotizados en una armadora de autos.

Actualmente se ha incrementado la investigación en el campo de los robots móviles, debido al alto grado de autonomía, comparado con los manipuladores. Los robots móviles tienen la habilidad de desplazarse libremente sobre un entorno. Este tipo de robots tienen diferentes aplicaciones como son: investigación espacial, rastreadores de minas, para desactivar bombas, asistencia para discapacitados, y de servicio en la industria [27].

Existen varios tipos de control para los robots que son: de los tipos autónomos y teledirigidos, un robot del tipo teledirigido precisa la intervención de un operador humano ya sea en forma parcial o total. Los del tipo autónomo, son capaces de tomar decisiones basándose en la comprensión del entorno que los rodea.

Uno de los principales problemas en el diseño de robots autónomos es el diseño de algoritmos que sean capaces de tomar decisiones en tiempo real además de simultáneamente percibir y analizar el entorno.

El entorno está conformado por el medio en el cual se desplaza el robot como los obstáculos y los factores que intervienen como la luz, temperatura, presión, campos magnéticos y otros. Es posible clasificarlos en cuatro tipos básicos, acuáticos, terrestres, aéreos y espaciales. Estos tipos definen los mecanismos de movilización, la cantidad de sensores y actuadores, la potencia necesaria, la complejidad de los algoritmos de toma de decisiones, etc [27].

Existen también entornos naturales y controlados. Es el entorno natural en donde un ser humano realiza su actividad, no es perjudicial para su salud, y las condiciones del entorno pueden variar sin previo aviso. Un ejemplo de esta variación es el clima y la luz ambiente que son consideradas como variables del entorno.

En cuanto a los entornos controlados estos son creados por el ser humano, y están compuestos por algunas variables como son el terreno, el ambiente, los contrincantes, etc. Estos se encuentran acotadas dentro de valores conocidos, dejando solo un cierto margen fuera de control (sin intervención), esto es muy común cuando se necesita un análisis próximo a la realidad.

En el entorno terrestre los robots se basan en conceptos de mecánica muchas veces ya probados. En muchos casos solo es cuestión de dotar de inteligencia a un vehículo convencional [27].

Para percibir el ambiente que lo rodea, es necesaria la utilización de sensores. Un sensor es un dispositivo que detecta una determinada acción externa como cualidades o fenómenos físicos, ya sea la energía, velocidad, aceleración, tamaño, cantidad, etc. [12].

La inteligencia de un robot móvil, generalmente esta controlada por un microcontrolador. Un microcontrolador es un sistema optimizado para ser utilizado en el control de equipos electrónicos. Físicamente es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de un ordenador: CPU, memoria y unidades de E/S. Es decir, se trata de un computador completo en un

solo circuito integrado. Aunque sus prestaciones son limitadas, además de dicha integración, su característica principal es su alto nivel de especialización [13].

Los robots móviles tienen un papel muy importante en el campo de la robótica, ya que estos tienen la capacidad de desplazarse por aire, por agua o por tierra, sin necesidad de ser tripulados. Sin embargo, nos enfocaremos a hablar de robots del tipo terrestres que se desplazan por medio de ruedas mejor conocidos como tipo vehículo, aunque existen otras formas de desplazarse, como orugas, robots con patas, o del tipo ápodo, que no usan ni ruedas ni patas para desplazarse (estos últimos se mueven mediante ondas sinusoidales que recorren el cuerpo del robot, parecido al movimiento de un gusano). Y exclusivamente de los robots rastreadores.

En este tipo de robots es usual encontrar sensores del tipo visión simple, como son ultrasónicos, láser, infrarrojos, de efecto Hall, optoreflectivos, los cuales se encargan de detectar la presencia de algún objeto y en algunos casos se puede calcular la distancia del sensor al objeto con ayuda de un microcontrolador.

En la parte de sensado el robot utiliza sensores normalmente del tipo optoreflectivo para poder ver la línea, y se basa en la cantidad de luz reflejada para poder discriminar cuándo está viendo un color u otro.

En la etapa de acondicionamiento de señal, es como su nombre lo indica se ajusta la señal ya sea para amplificarla en corriente o en voltaje y se transforma en una señal digital, para finalmente ser entregada al microcontrolador.

En la etapa de control, podemos encontrar normalmente microcontroladores en los cuales se programan los algoritmos de control, en este caso pueden ser algoritmos que controlan la velocidad, dirección, y en algunos casos algoritmos para buscar la línea en caso de que el robot la llegue a perder, ya sea por exceso de

velocidad o por inercia, el microcontrolador entrega las señales de control para la corrección de posición a la etapa de potencia.

La etapa de potencia normalmente se compone de transistores de potencia para amplificar en corriente las señales que llegan del microcontrolador, aunque es usual encontrar circuitos integrados especializados para este tipo de aplicación.

En cuanto a los actuadores podemos encontrar: solenoides, motores de corriente directa, motores de pasos, y aunque es inusual motores de corriente alterna.

Estos motores son alimentados por la etapa de potencia ya descrita anteriormente, y se encargan del movimiento del robot como son la tracción y la dirección de éste.

El sistema mecánico se refiere a la morfología del mismo, como lo es el tipo de chasis, la forma de acomodar los motores, la forma de acomodar los sensores y el diseño de toda la estructura, este es uno de los sistemas más importantes en el diseño de los robots, lo explicaremos en forma más detallada en los siguientes capítulos.

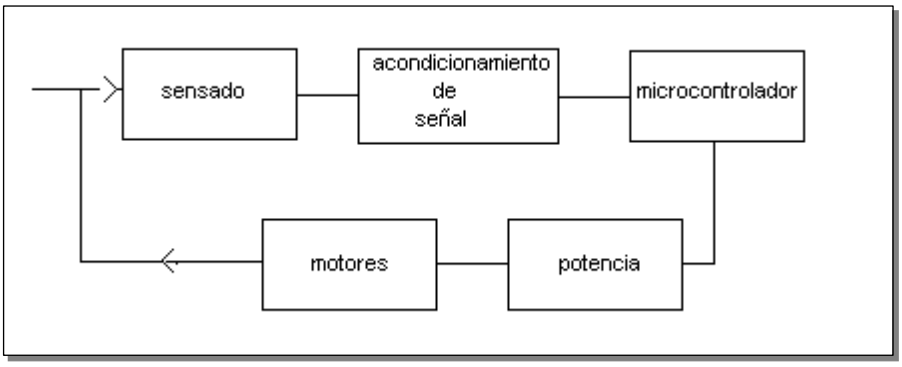

Figura 1.1.1 Diagrama de bloques de las etapas de un robot móvil

1.2 Servomotores De Corriente Directa

Un servomotor es un dispositivo pequeño que tiene un eje de rendimiento controlado. Este puede ser llevado a posiciones angulares específicas al enviarle una señal codificada. En la práctica, se usan servos para posicionar superficies de control como el movimiento de palancas, pequeños ascensores y timones. Ellos también se usan en radio control y por supuesto, en robots.

Los servos son sumamente útiles en robótica. Los motores son pequeños, tienen internamente un circuito de control y es sumamente poderoso para su tamaño. Un servo normal o estándar como el HS-300 de Hitec tiene 42 onzas*pulgada o 3kg*cm de torque que es bastante fuerte para su tamaño. Su potencia es proporcional para cargas mecánicas. Un servo, por consiguiente, no consume mucha energía.

Se muestra la composición interna de un servo motor en la Figura 1.2.1, se observa el circuito de control, el motor, un juego de piñones, y la caja. También se pueden observar los tres alambres de conexión externa. Uno es para alimentación Vcc (+5volts),otro para conexión a tierra GND y el alambre blanco es de control.

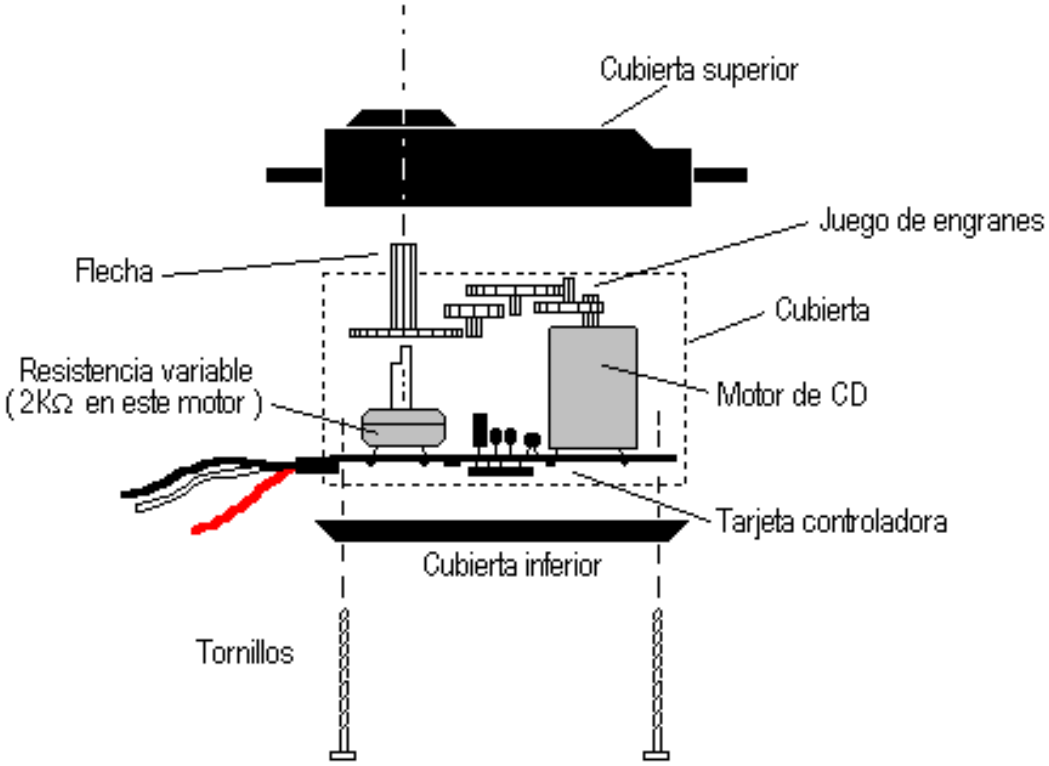

Figura 1.2.1 servomotor de corriente directa

El motor del servo tiene algunos circuitos de control y un potenciómetro, éste es conectado al eje central del servo motor. En la Figura 1.2.1. [19] se puede observar al lado derecho del circuito. Este potenciómetro permite a los componentes de control, supervisar el ángulo actual del servo motor. Si el eje está en el ángulo correcto, entonces el motor está apagado. Si el circuito verifica que el ángulo no es correcto, el motor girará en la dirección adecuada hasta llegar al ángulo correcto. El eje del servomotor es capaz de girar alrededor de los 180 grados. En algunos servomotores llega a los 210 grados, pero varía según el fabricante. Un servo normal se usa para controlar un movimiento angular de entre 0 y 180 grados [19].

1.3 Características Del Acelerómetro H48C

Se denomina acelerómetro a cualquier instrumento destinado a medir aceleraciones.

El acelerómetro H48C es un módulo integrado que puede sensar aceleraciones de ±3g sobre los 3 ejes (x,y,z), el módulo está contenido sobre una tarjeta rectangular alimentado con 3.3v, un convertidor analógico digital MCP3204 de 4 canales de 12 bits para leer las señales del H48C. El tamaño de la tarjeta es de 17.8x20.3 mm. La adquisición de datos del módulo se hace a través de una comunicación de sincronía serial con el lenguaje de programación PicBasic, que resulta bastante sencilla con los comandos SHIFTOUT y SHIFTIN [28]. En la Figura 1.3.1 [28] se muestra físicamente el módulo H48C.

Figura 1.3.1 Sensor H48C de Parallax

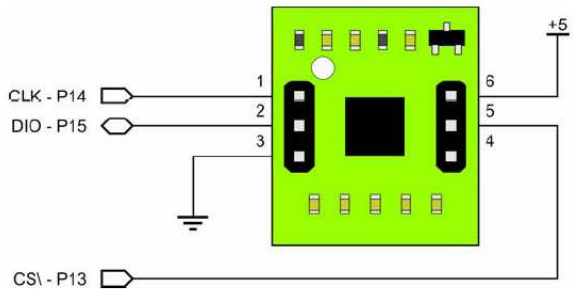

Figura 1.3.2 Configuración de terminales del módulo H48C de Parallax

La configuración de terminales se muestra en la Figura 1.3.2. [28], vista desde la parte superior de la tarjeta.

1.4 Microcontroladores De Tecnología RISC

Todas las marcas de microcontroladores se pueden diferenciar según el tamaño y componentes especiales, pero, en general todos tienen los siguientes bloques básicos:

- Procesador o CPU
- Memoria RAM para contener los datos
- Memoria para el programa tipo ROM, PROM, EPROM, EEPROM o FLASH.
- Líneas E/S para comunicarse con el exterior.
- Recursos auxiliares.
- Temporizadores
- Puertos serie y/o paralelo.
- Convertidores D/A y A/D.
- Generador de impulsos de reloj para sincronizar el funcionamiento de todo el sistema [20].

Microcontrolador PIC18F452

Características generales:

1.-Arquitectura HARVARD. El término Arquitectura Harvard originalmente se refería a las arquitecturas de computadoras que utilizan dispositivos de

almacenamiento físicamente separados para las instrucciones y para los datos (en oposición a la Arquitectura von Neumann).

2.- CPU RISC. En la arquitectura computacional, RISC del inglés *Reduced Instruction Set Computer* (Computadora con Conjunto de Instrucciones Reducido).

Son microprocesadores con las siguientes características fundamentales:

- Instrucciones de tamaño fijo, presentadas en un reducido número de formatos.

- Sólo las instrucciones de carga y almacenamiento acceden a memoria por datos.

- Memoria RAM de 1536x8 bites.
- Memoria EEPROM de 256x8 bites.
- Memoria FLASH de 32Kx14 bites.
- Multiplicador de 8 x 8 bits en un ciclo de instrucción.
- Cinco puertos paralelos con función alterna en la mayoría de ellos:
    - Puerto A de 6 bits bidireccional
    - Puerto B de 8 bits bidireccional
    - Puerto C de 8 bits bidireccional
    - Puerto D de 8 bits bidireccional
    - Puerto E de 3 bits bidireccional
- Convertidor analógico/digital, 8 canales de 8 o 10 bits de resolución.
- Puerto serie asíncrono (SCI/USART).
- Puerto serie síncrono (SSP).
- Puerto serie paralelo (PSP).
- Protocolo I2C.
- Sistema temporizador (CCP):
    - Módulo de captura
    - Módulo de comparación
    - Módulo PWM (principalmente usado en control de motores)

- 40 pines.
- 3 registros de propósito general:
    - Registro W
    - Program counter
    - Stack pointer
- Modos de direccionamiento directo, indirecto y relativo [30].

En la Figura 1.4.1. [30] se muestra el diagrama de bloques del microcontrolador PIC18F452.

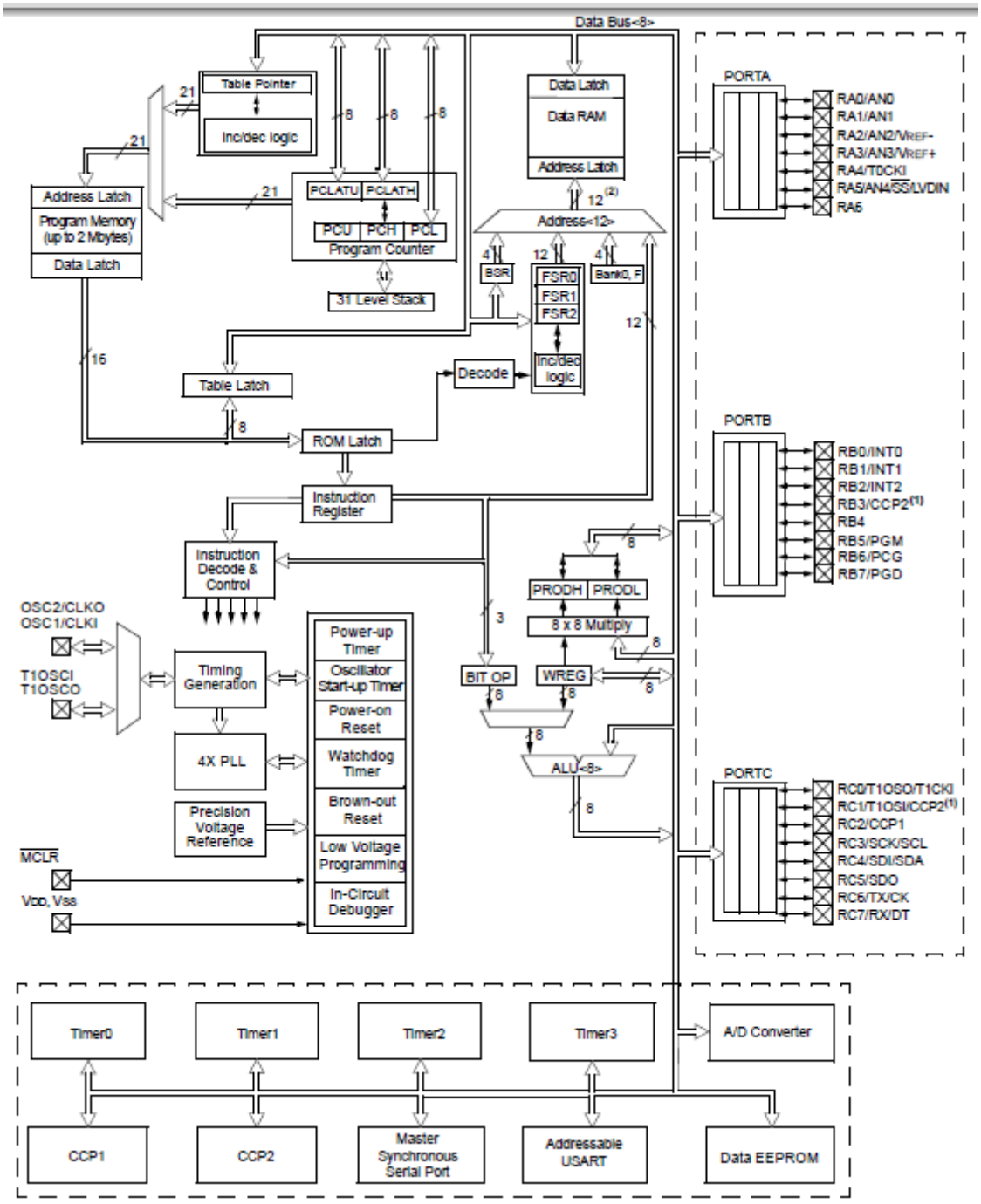

Figura 1.4.1. Diagrama de bloques de microcontrolador PIC18F452

En esta figura podemos ver cómo está constituido el microcontrolador PIC18F452 y de manera detallada se muestran en la parte de abajo los periféricos, en la parte derecha los puertos entrada-salida, en la parte de arriba los buses de datos y por último en la parte de la entrada del oscilador el PLL que multiplica la frecuencia de entrada por cuatro, o sea que con un cristal de 10Mhz se puede hacer trabajar a 40Mhz.

1.5 Giroscopio De Radio Control PK3

Es un giroscopio piezoeléctrico de un canal utilizado habitualmente en aviones y helicópteros de radio control para compensar de forma automática los giros bruscos. El giróscopo es completamente electrónico, sin partes móviles, lo que hace que sea muy pequeño, tenga un bajo consumo y una respuesta muy rápida. La entrada del giróscopo se conecta en lugar del servo que se quiere compensar y el servo se conecta a su vez al giróscopo. Todas las órdenes de control pasan directamente al servo de forma transparente. Tan pronto como se produce un movimiento angular (por ejemplo inclinación) el giróscopo manda al servo una señal proporcional para compensar el movimiento. Por ejemplo, aplicado al rotor de cola de un helicóptero, cuando el helicóptero acelera, la cola tiende a girar lateralmente como consecuencia del par rotor. Si se utiliza el giróscopo en este caso, lo que hace es mandar una señal de control al servo que controla la cola, para aumentar el paso y compensar la desviación. En el caso de los robots, la salida en lugar de conectarse a un servo, se conecta a un microcontrolador que puede leer la anchura del pulso del giróscopo y así saber cuando se produce un giro. Los giróscopos son esenciales en los robots de tipo balancín y el los sistemas de guiado de precisión en el que hay que medir y compensar el momento de giro. Características técnicas: Dimensiones 26 x 27.5 x 11.3 mm. Peso: 7 g. Tensión de funcionamiento: 4.8 - 6V. Consumo de corriente: 10 mA. Ajuste de sensibilidad y de valor neutro mediante dos potenciómetros [29]. En la Figura 1.5.1 [29] se muestra el aspecto físico del giroscopio PK3, y el eje en el que mide la rotación. En la Figura 1.5.2 [29] se muestra la forma en la que se conecta normalmente con un receptor de radio control y el servomotor que compensa.

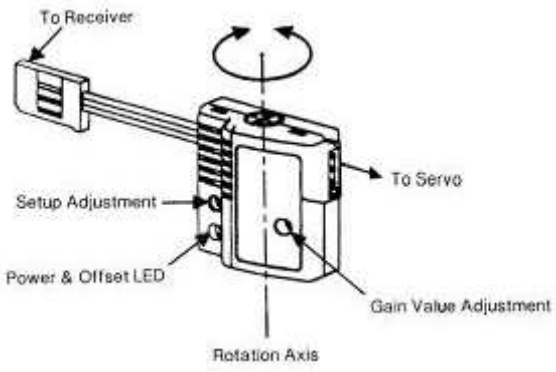

Figura 1.5.1 Descripción del giroscopio

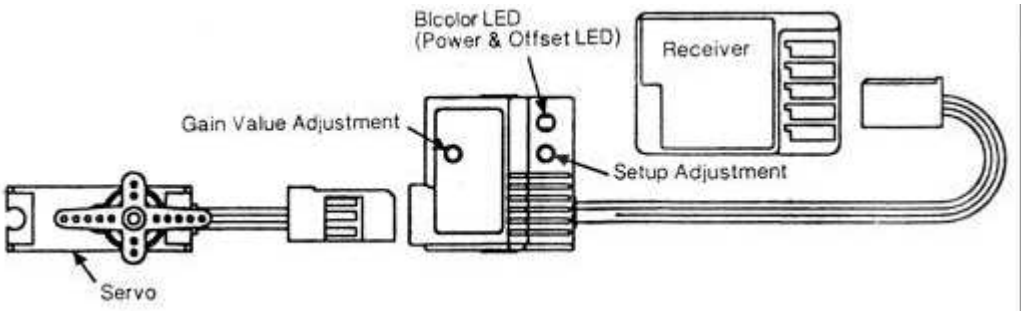

Figura 1.5.2 Modo de conexión del servomotor con el giroscopio

1.6 Sistema Embebido De Visión Cmucam3

Esta cámara, fue diseñada por la Universidad de Carnegie Mellon. Consiste básicamente en un fotorreceptor Cmos de la firma Omnivisión, y un microcontrolador que la maneja, con 2 entradas y salidas seriales, además de un puerto para 4 servomotores, y una ranura para introducir memorias Smart-Card.

Esta cámara pretende dar a los usuarios que requieren de la manipulación de imágenes, una especie de producto “todo en uno”, donde una persona que sepa algo de programación en C, pueda, con un poco de ingenio, hacer sus robots con visión “inteligente”.

La CMUCam3, muestra serias deficiencias en cuanto a su estructura, su funcionalidad y su desempeño. Por un lado, aún cuando la idea de integrar una cámara a un microcontrolador pueda ser algo bastante útil, la CMUCam 3 requiere de muchas mejoras. Así, uno de los grandes defectos de esta cámara, es que no tiene suficiente memoria RAM para cargar tan siquiera una cuarta parte de una imagen en baja resolución. Esto es algo muy lamentable, considerando que el dispositivo pretende precisamente esto: Manipular imágenes.

Esto trae como consecuencia que las librerías de reconocimiento de color que se entregan con la CMUCam 3 sean de baja calidad, ya que examinan la imagen píxel por píxel y no por conjunto de píxeles [30]. En la Figura 1.6.1 [30] se muestra el aspecto físico del sistema de visión Cmucam3.

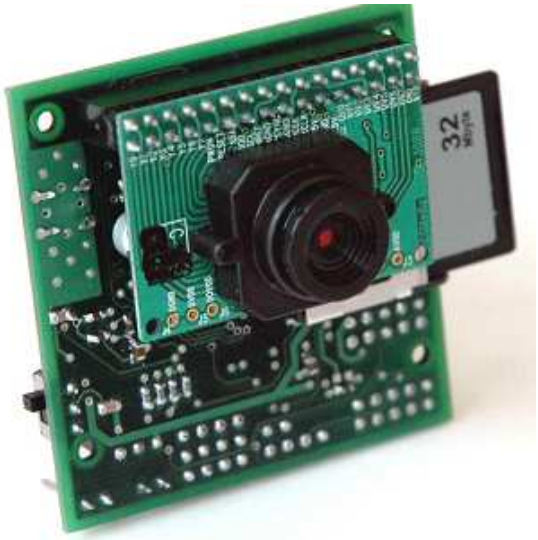

Figura 1.6.1 Cmucam3 parte frontal

1.7 Tipos De Baterías

Una batería es un dispositivo electroquímico el cual almacena energía en forma química. Cuando se conecta a un circuito eléctrico, la energía química se transforma en energía eléctrica. Todas las baterías son similares en su construcción y están compuestas por un número de celdas electroquímicas. Cada una de estas celdas está compuesta de un electrodo positivo y otro negativo además de un separador. Cuando la batería se está descargando un cambio electroquímico se está produciendo entre los diferentes materiales en los dos electrodos. Los electrones son transportados entre el electrodo positivo y negativo vía un circuito externo (bombillas, motores de arranque etc.) [15].

Es necesario distinguir entre baterías recargables (acumuladores) y pilas o baterías desechables. La diferencia fundamental entre ambos tipos está en que las baterías recargables permiten revertir la reacción química en la que está basado su funcionamiento, mientras que las desechables no. En este capítulo estudiaremos las características de las baterías recargables, ya que económicamente representan un ahorro debido a sus propiedades de recarga.

Baterías de plomo: Las primeras baterías de plomo-ácido (acumuladores de plomo), fueron fabricadas a mediados del siglo XIX por Gaston Planté. Hoy en día aún son uno de los tipos de baterías más comunes. Se descubrió que cuando el material de plomo se sumergía en una solución de ácido sulfúrico se producía un voltaje eléctrico el cual podía ser recargado [15].

Este tipo de baterías es único en cuanto a que utiliza el plomo, material relativamente barato, tanto para la placa positiva como para la negativa. El material activo de la placa positiva es óxido de plomo (Pbo_2). El de la placa negativa es plomo puro esponjoso y el electrolito está disuelto en acido sulfúrico (H_2SO_4).

Cuando hablamos de material activo en las baterías de ácido-plomo, nos referimos al óxido de plomo y al plomo esponjoso. Las baterías de plomo reinan en nuestros vehículos pero están sólo destinadas a cubrir las necesidades de arranque, iluminación e ignición (no tienen suficiente energía para mover el coche). En estas baterías la vida útil será mayor cuanto menor sea la descarga en cada ciclo de carga-descarga [15].

Baterías Alcalinas: No son muy comunes las baterías alcalinas recargables. La mejor característica que tienen es que aportan una tensión de 1.5 voltios. Sin embargo, son difíciles de encontrar en el mercado y requieren un cargador adecuado también poco común.

Níquel-Cadmio: Son las más habituales. Proporcionan tensiones de 1.2 voltios. Contienen cadmio, un metal pesado que representa un peligro ecológico. Interiormente tienen dos electrodos, el de cadmio (negativo) y el de hidróxido de níquel (positivo), separados entre sí por un electrolito de hidróxido de potasa. Llevan también un separador situado entre el electrodo positivo y la envoltura exterior y un aislante que las cierra herméticamente como se muestra en la Figura 1.7.1. [15]

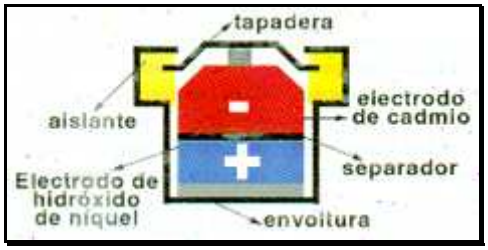

Figura 1.7.1 Partes de una batería de Níquel

Su aspecto más positivo es el precio. Aunque parezcan caras, por el número de veces que se pueden recargar en comparación con una batería no-recargable, resulta económico el precio. Sin embargo, tienen una característica llamada “efecto memoria”. Significa que antes de recargarlas es necesario haberlas agotado completamente ya que en caso contrario su vida se va acortando. Una solución es, cuando se vea que empiezan a perder energía, dejar el equipo encendido hasta que se agoten completamente. Además son contaminantes.

Níquel-Metal hidruro: No tienen metales pesados como el cadmio y por eso son menos perjudiciales para el medio ambiente. Además de ser menos contaminantes proporcionan tensiones de 1.3 voltios y tienen una capacidad mucho mayor por lo tanto duran más que las de Níquel-Cadmio y dan más energía. No tienen efecto memoria, de modo que se pueden recargar aunque no se hayan agotado al completo. Las mejores llegan a soportar hasta 1,000 procesos de carga.

Litio-Ion: las ***baterías de litio***, junto quizá a las de hidruro metálico son las que van encontrando un mayor consenso en cuanto a su potencial y un mayor esfuerzo en su investigación y desarrollo a nivel mundial.

Son muchas las razones que han originado este consenso. En primer lugar ***el litio es el metal más ligero*** y esto da lugar a una alta capacidad específica (Figura 1.7.2. [16]), lo que permite obtener la misma energía con un peso muy inferior (Figura 1.7.3) [16].

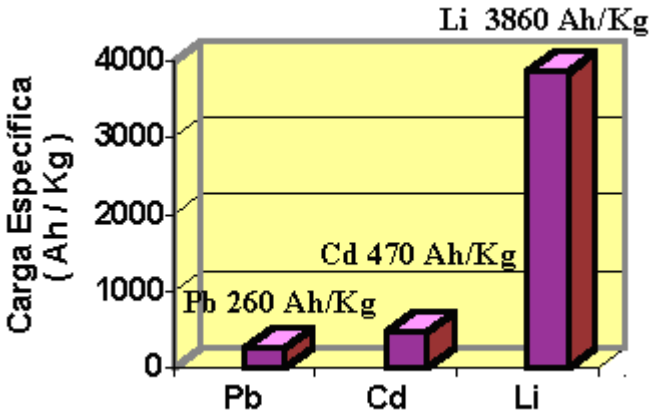

Figura 1.7.2 Carga especifica para distintos ánodos

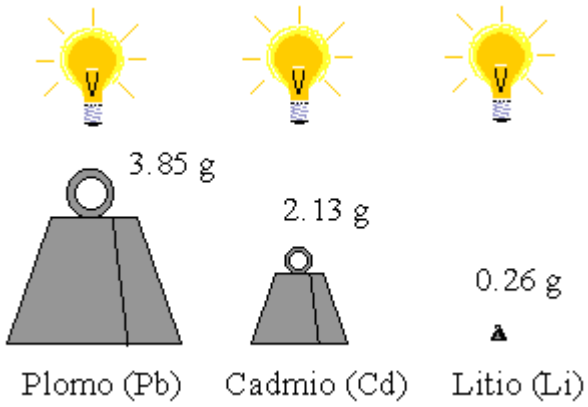

Figura 1.7.3 Masa necesaria para producir un Amper durante 1 hora

Sin embargo, estas baterías tienen un gasto de producción elevado y muy costoso que se refleja en su precio final. Su ciclo de vida se sitúa entre los 500-600 ciclos de carga/descarga. Sin embargo, ofrece una capacidad equivalente y más fiable dando una densidad de energía más elevada y constante que las baterías de Ni/Cd o Ni/MH. Tiene muy poca auto descarga 6% al mes y la tensión de trabajo típica es de 3.6 volts.

Método de carga

El método de carga es relativamente sencillo, pero debe ser muy preciso. En baterías con ánodo de grafito la tensión de final de carga es de 4.1V y en baterías con ánodo de carbón la tensión de final de carga es de 4.2V, por tanto, al llegar a esa tensión la carga debe detenerse inmediatamente. La precisión requerida es del **1%**. Si la sobrepasamos podemos acortar el ciclo de vida de la batería. Y si no se llega a esa tensión la carga no será completa. El método de carga más usado es el conocido como "corriente constante - tensión constante". Consiste en empezar la carga con una corriente constante. Cuando se aproxima al final, el último empujón se le da con una tensión constante. Hay otros métodos más simples pero quizás más lentos. Lo importante es detener la carga cuando se alcanza el límite. Estas baterías tienen un rendimiento energético muy bueno durante la carga: casi toda la energía que reciben se usa para cargar la batería, con muy poco o nulo calentamiento [17].

Problemas de estas baterías

Existe un problema (y además, poco documentado): la **pasivación**. No confundir este fenómeno con el llamado "efecto memoria" de las baterías de Ni-Cd [17].

La pasivación consiste en la formación de una película de cloruro de litio (LiCl) en la superficie del ánodo. De algún modo sirve para evitar la auto descarga, cuando la batería no está siendo usada. Esta delgada película es, funcionalmente, una resistencia. Pero está claro que puede producir una caída de tensión o "retraso" en la entrega de energía tal como se ve en esta Figura 1.7.4. [17].

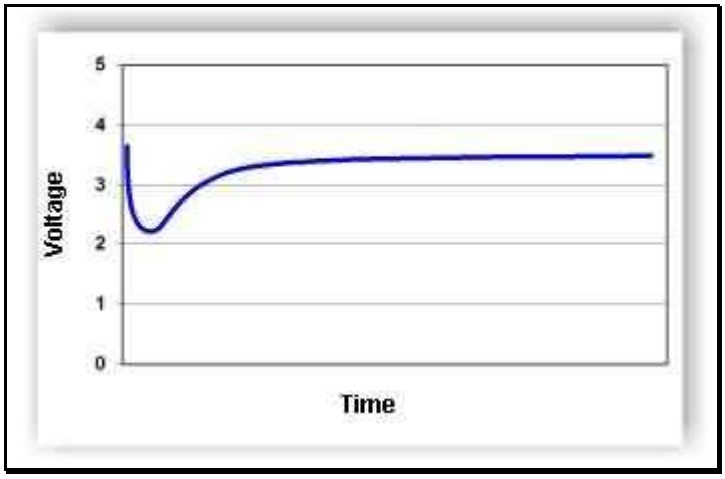

Figura 1.7.4 forma en la que entrega la energía una batería de litio

Conforme la batería va siendo usada, esta fina película va desapareciendo. El problema está en que la caída de tensión puede ser lo suficientemente abrupta como para que nuestro equipo se apague. Cuanto mayor sea la energía requerida al principio, más acusado puede ser el problema. Cuando dejamos de usar la batería, la película vuelve a formarse.

Este fenómeno depende del diseño y constitución de la batería, tiempo sin usar (cuanto mayor sea este tiempo, más gruesa será la capa de LiCl). En cuanto a la temperatura de almacenamiento, a mayor temperatura, mayor pasivación sin embargo cuando la temperatura de uso es baja, este efecto será más visible. No obstante esta es una de las mejores baterías para cualquier prototipo en el que el reducido peso sea una prioridad. Por lo tanto este tipo de baterías se acopla a nuestras necesidades de peso, tamaño, costo y durabilidad.

ASPECTOS BASICOS DEL ROBOT

2.1 Locomoción Humana Normal

La locomoción humana normal se ha descrito como una serie de movimientos alternantes, rítmicos, de las extremidades y del tronco que determinan un desplazamiento hacia adelante del centro de gravedad. Más concretamente, la locomoción humana normal puede describirse enumerando algunas de sus características. Aunque existen pequeñas discrepancias en la forma de la marcha de un individuo a otro, estas diferencias caen dentro de pequeños límites.

El ciclo de la marcha empieza cuando el pie contacta con el suelo y termina con el siguiente contacto con el suelo del mismo pie. Los componentes básicos del ciclo de la marcha son dos: la etapa de apoyo y la etapa de balanceo. Una pierna está en fase de apoyo cuando está en contacto con el suelo y está en fase de balanceo cuando no contacta con el suelo [34].

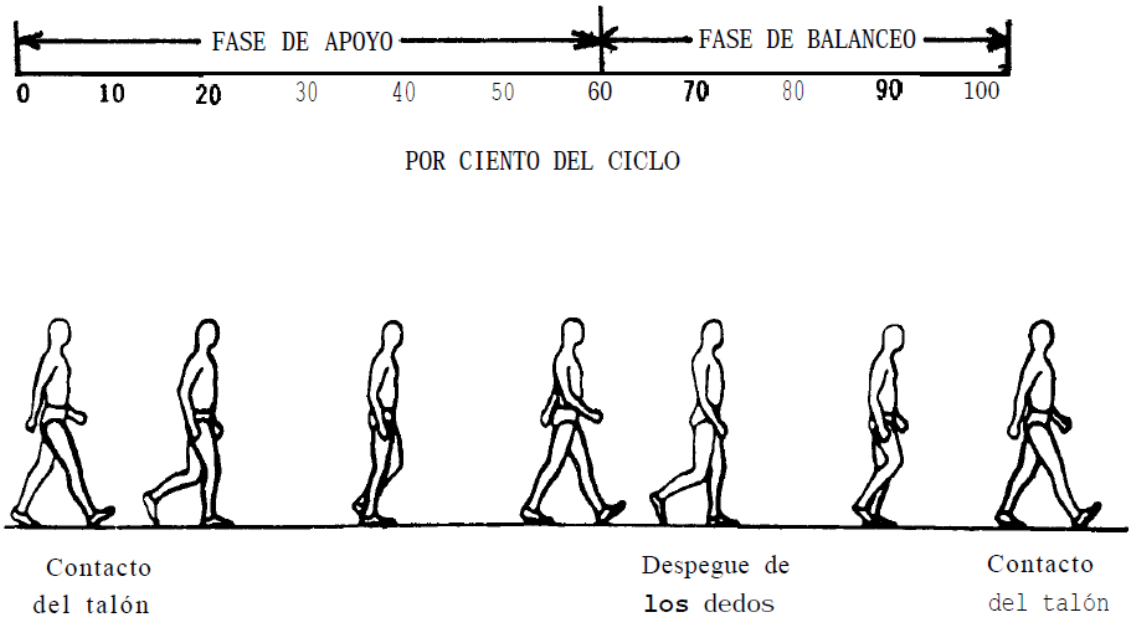

Figura 2.1.1 Fases de apoyo y balanceo de un paso completo

La longitud del paso completo es la distancia lineal entre los sucesivos puntos de contacto del talón del mismo pie. Longitud del paso es la distancia lineal en el plano de progresión entre los puntos de contacto de un pie y el otro pie.

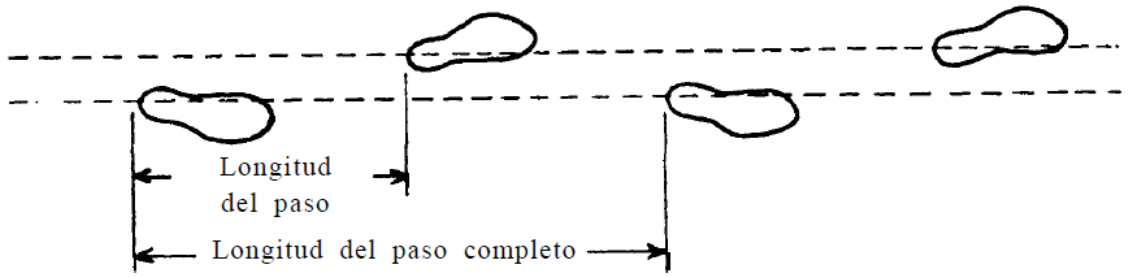

Figura 2.1.2 Longitud del paso

Apoyo sencillo: Se refiere al período cuando sólo una pierna está en contacto con el suelo. El ciclo de doble apoyo ocurre cuando ambos pies están en contacto con el suelo simultáneamente. Para referencia del pie significa que por un corto período de tiempo, la primera parte de la etapa de apoyo y la última parte de la fase de apoyo, el pie contra lateral esta también en contacto con el suelo. La ausencia de un período de doble apoyo diferencia el correr del caminar.

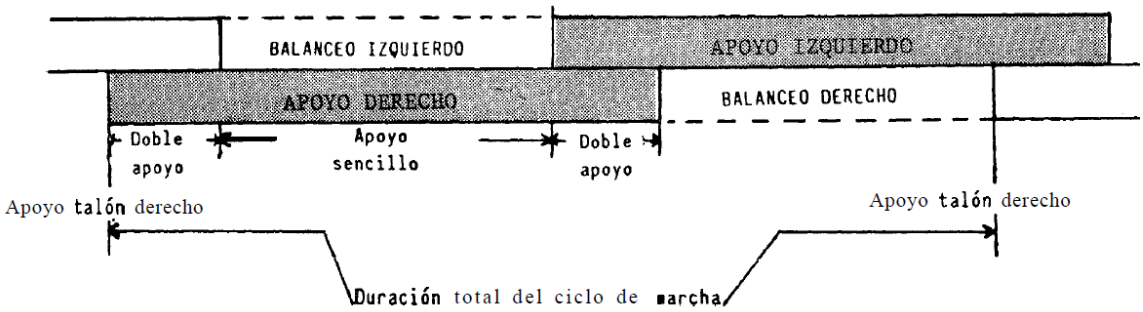

Figura 2.1.3 Duración de la marcha y apoyos

Hay cinco momentos que son útiles al subdividir la fase de apoyo: Contacto del talón, apoyo plantar, apoyo medio, elevación del talón y despegue del pie.

El contacto del talón se refiere al instante en que el talón de la pierna de referencia toca el suelo. El apoyo plantar se refiere al contacto de la parte anterior del pie con el suelo. El apoyo medio ocurre cuando el trocánter mayor está alineado verticalmente con el centro del pie, visto desde un plano sagital.

La elevación del talón sucede cuando este se eleva del suelo, y el despegue del pie ocurre cuando los dedos se elevan del suelo [34].

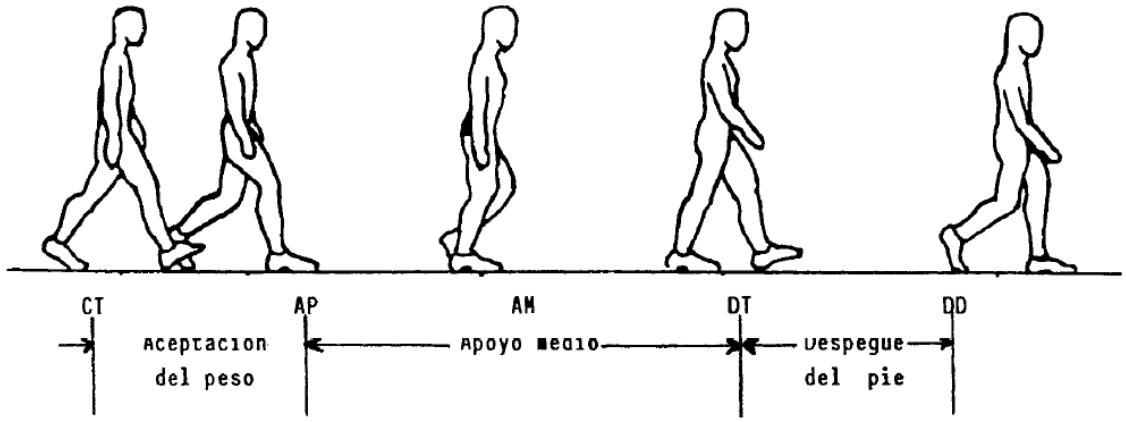

Figura 2.1.4 Subdivisiones de la fase de apoyo

La etapa de balanceo puede dividirse en tres intervalos designados con los términos de aceleración, balanceo medio y deceleración**.** Cada una de estas subdivisiones forma aproximadamente un tercio de la fase de balanceo. El primer tercio, referido como período de aceleración, se caracteriza por la rápida aceleración del extremo de la pierna inmediatamente después de que los dedos abandonan el suelo. Durante el tercio medio de la fase de balanceo, el intervalo del balanceo medio, la pierna balanceada pasa a la otra pierna, moviéndose hacia delante de la misma, ya que está en fase de apoyo. El tercio final de la fase de balanceo está señalado por la deceleración de la pierna que se mueve rápidamente cuando se acerca al final del intervalo [34].

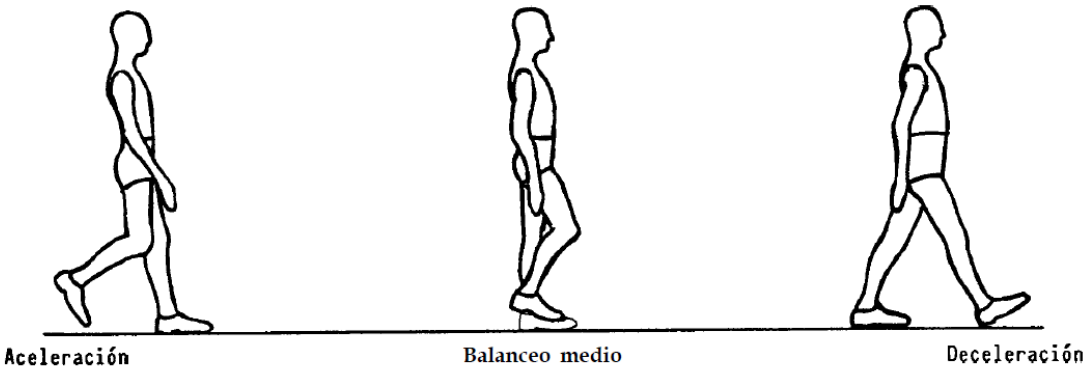

Figura 2.1.5 Fases de la etapa de balanceo

El centro de gravedad generalmente se desplaza hacia el pie de apoyo, mientras que cuando el movimiento es lento el centro de masa permanece centrado en la planta del pie de apoyo, cuanto más rápido es el movimiento el centro de masa se mueve menos hacia los lados, puede incluso estar fuera de la planta de el pie de apoyo como se muestra en la Figura 3.3.7 [35].

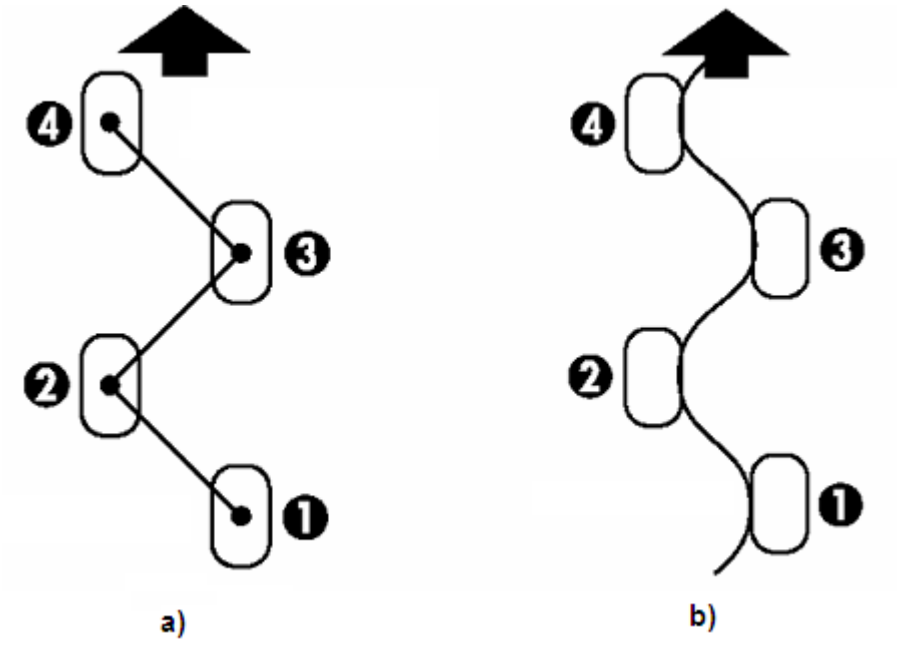

Figura 2.1.6 a) Centro de gravedad cuando la caminata es lenta, b) Cuando la caminata es más rápida

2.2 Diseño Mecánico

Para el diseño de la estructura mecánica se tomo en cuenta que debería parecer humano, es decir contener los bloques principales con los que cuenta la estructura humana. El robot debe tener dos brazos, dos piernas, una cabeza y todos deben estar unidos a un tronco, también debe de ser capaz de caminar sobre sus dos piernas.

Otro criterio que se tomó en cuenta en el momento del diseño mecánico, es el peso, ya que dependiendo del peso y del centro de gravedad nos permitirá escoger los motores con el torque adecuado.

La primera estructura mecánica que se desarrolló contaba con 16 grados de libertad, muy parecida al robot comercial Robonova-1.

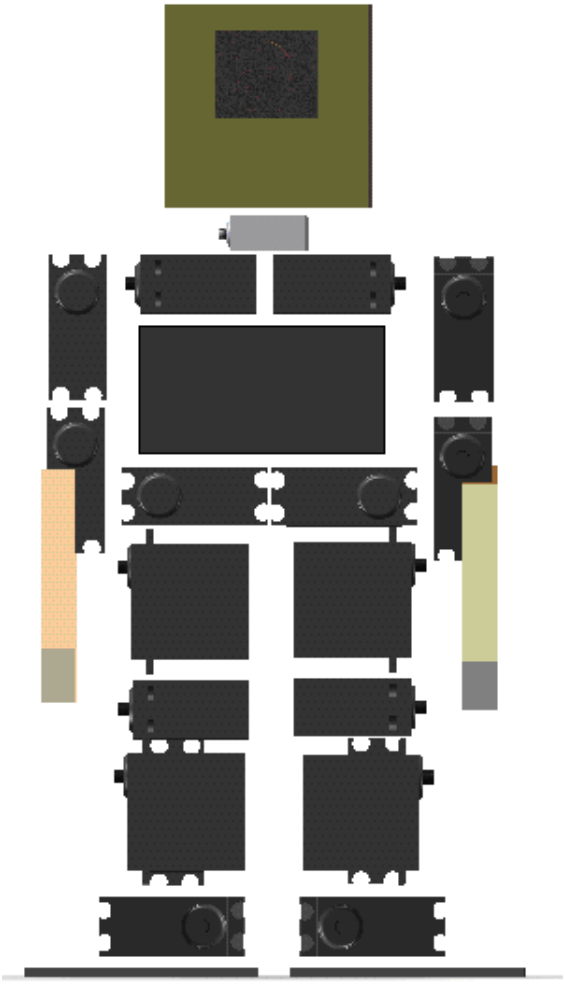

Figura 2.2.1 Primer estructura mecánica

En la Figura 2.2.1 se muestra la primera estructura mecánica, se construyo con servomotores marca Futaba de 3.2 kg*cm, cada servomotor de este tipo tiene un peso de 43 gramos, con esta estructura se pudo observar el potencial de los servomotores. Sin embargo, el objetivo era una mejora a esa estructura por eso se propuso una estructura con 18 grados de libertad como la que se muestra a continuación.

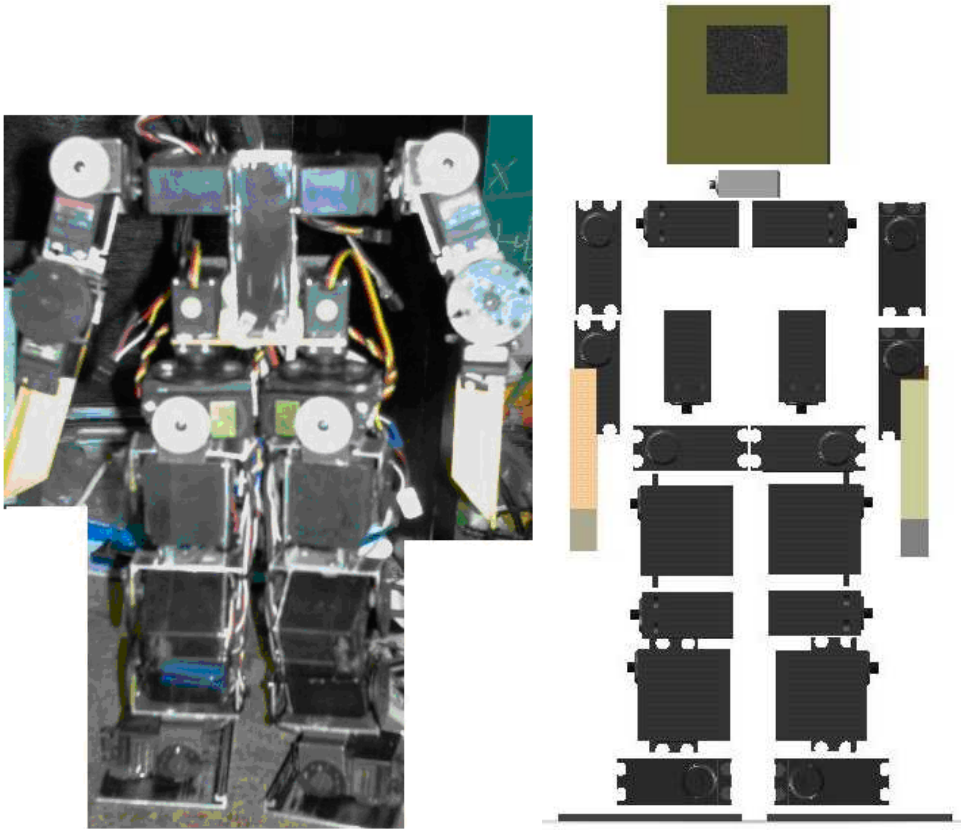

Figura 2.2.2 Segunda estructura mecánica

Esta estructura fue construida con servomotores Hitec de 3.5kg*cm los cuales trabajan por ancho de pulso para su manejo se diseño una tarjeta con 2 microcontroladores que controlaban a los 18 motores, sin embargo se hizo una modificación en las piernas para tratar de bajar el centro de masa y reducir un poco el peso de la estructura. Se redujo el tamaño en las piernas, tobillos y tórax. En las Figuras 2.2.3 y 2.2.4 se muestran las modificaciones a las piernas y al tórax con acotaciones en centímetros.

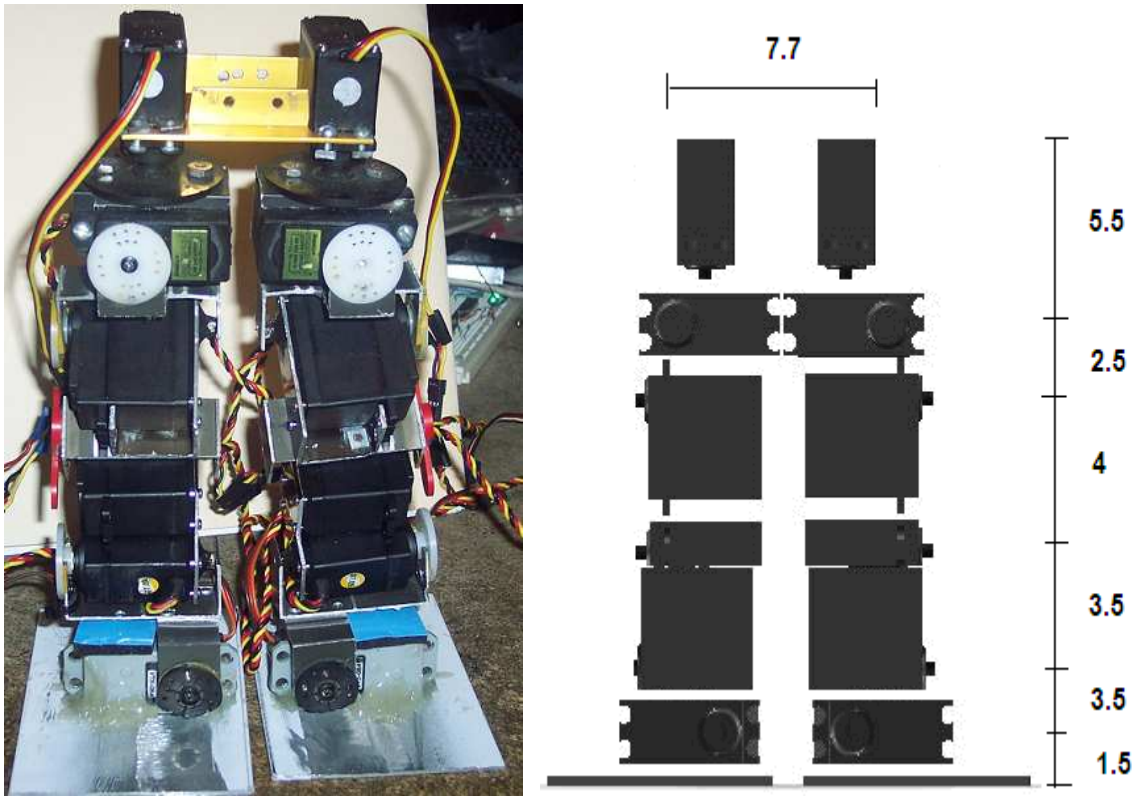

Figura 2.2.3 Juego de piernas terminado y funcional

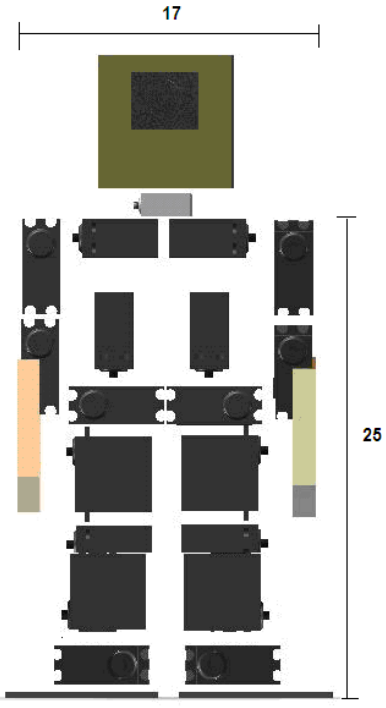

Figura 2.2.4 Acotaciones de la tercera estructura mecánica

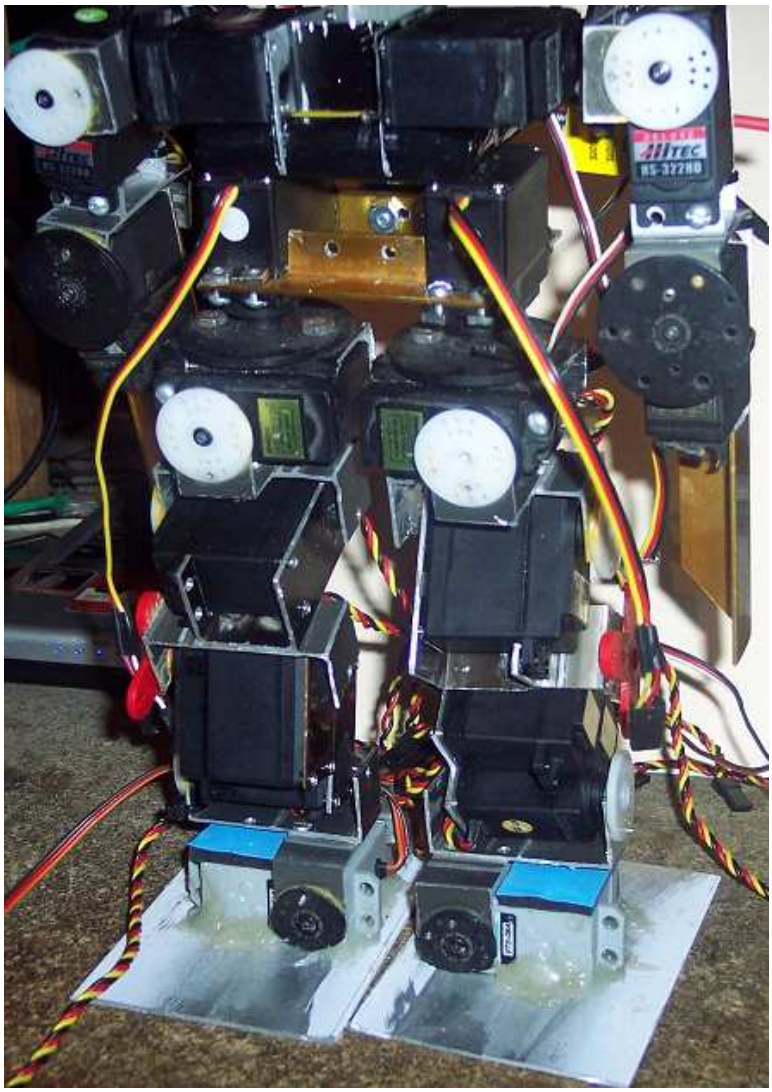

Figura 2.5.5 Tercera estructura mecánica lista para hacer las pruebas

Una vez terminada nuestra estructura mecánica se encontró la altura de su centro de masa mediante un sencillo dispositivo parecido a una balanza. En la Figura 2.5.6 se muestra la forma en la que se coloca el robot. Una vez que esté balanceado se mide la distancia del punto de apoyo a la base del pie, esta es la medida de altura del centro de masa.

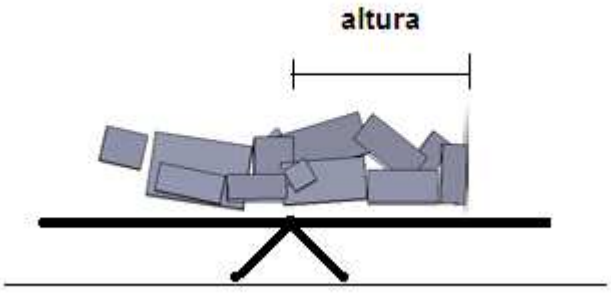

Figura 2.2.6 Balanza para determinar el centro de masa

Para nuestra tercer estructura mecánica el centro de masa es de 14.2 cm. Otro dato de interés es determinar el momento en el punto de volteo respecto a la articulación del tobillo, y compararlo con la torca máxima del motor. Esto para poder determinar si los motores son adecuados.

La planta del pie mide 8 cm y la articulación se encuentra en el punto medio, el punto de volteo se obtiene cuando el centro de masa está exactamente en la orilla del pie, si lo llevamos a un diagrama queda como el de la Figura 2.2.7.

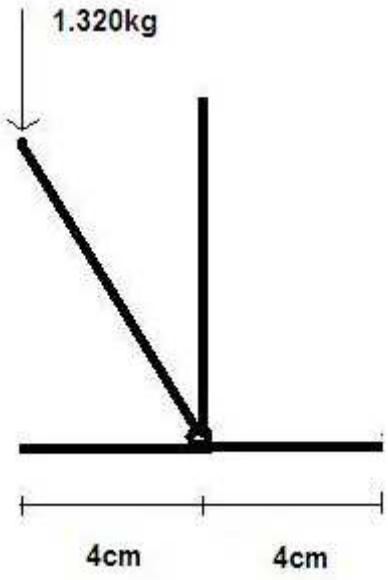

Figura 2.2.7 Diagrama para calcular la torca máxima

Como se sabe que T=FxD la torca resultante es: T=4cm x 1.32kg=5.28 kg *cm considerando que la torca máxima de los servos de los tobillos es de 3.5 kg *cm podemos inferir que los motores de los tobillos necesitan más torque.

El conjunto de todos los motores en la posición de firmes consumía aproximadamente 1500mA para dar un paso la corriente llegaba a 3000mA debido a que estaban trabajando por encima de la torca nominal, por consiguiente en pocos minutos los motores de los pies sufrían gran calentamiento provocando la caída inmediata. La tarjeta que controla los 18 servomotores contiene 2 microcontroladores PIC16F877 por lo que la tarjeta de control era demasiado grande y pesada. Debido a estas características se optó por sustituir los motores por unos de mayor torque y menor consumo de corriente, creando así la cuarta estructura mecánica. Después de una investigación sobre servomotores y la valoración de la información se decidió sustituir los motores por unos de la marca Dynamixel AX-12+, con esto se obtuvo una estructura mecánica óptima que se muestra en la Figura 2.2.8.

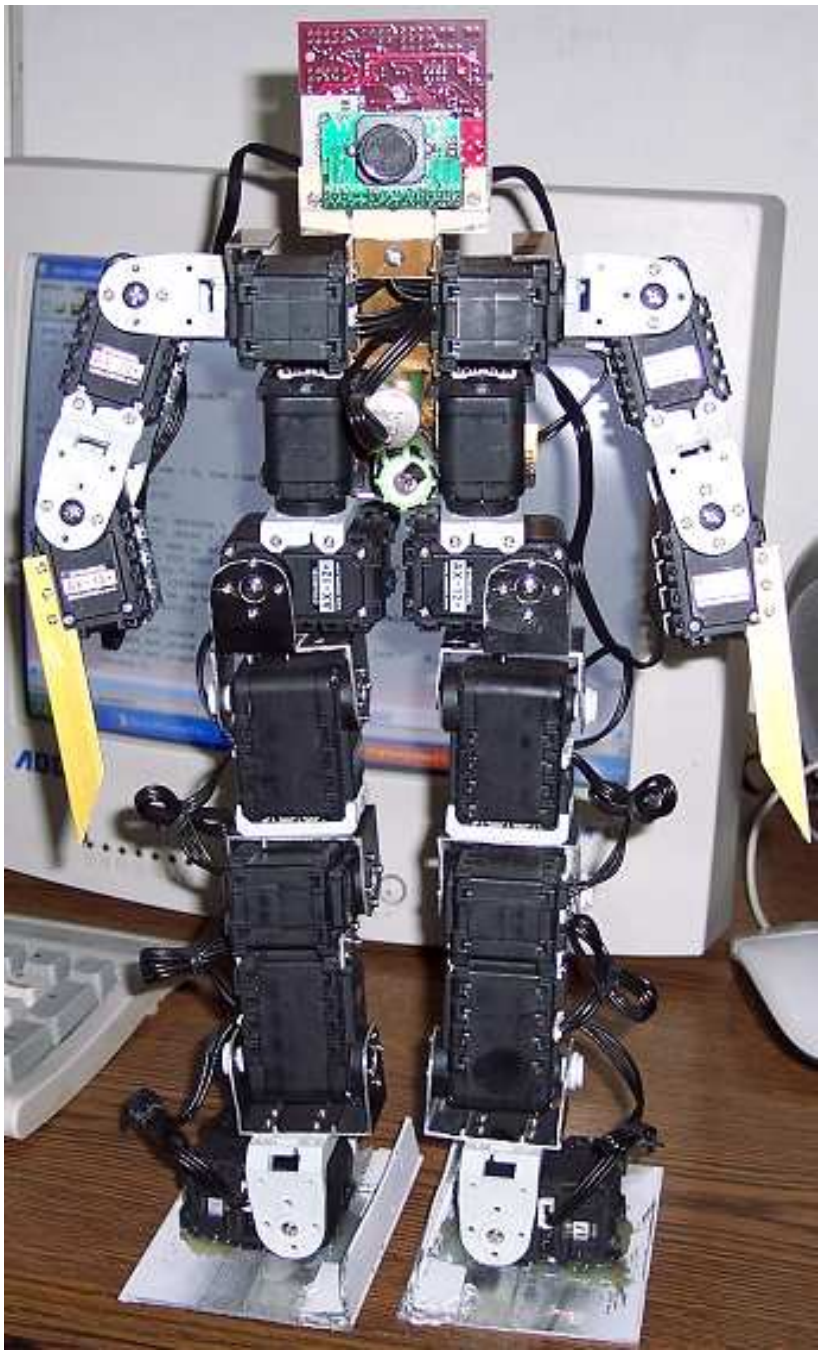

Figura 2.2.8 Estructura mecánica óptima

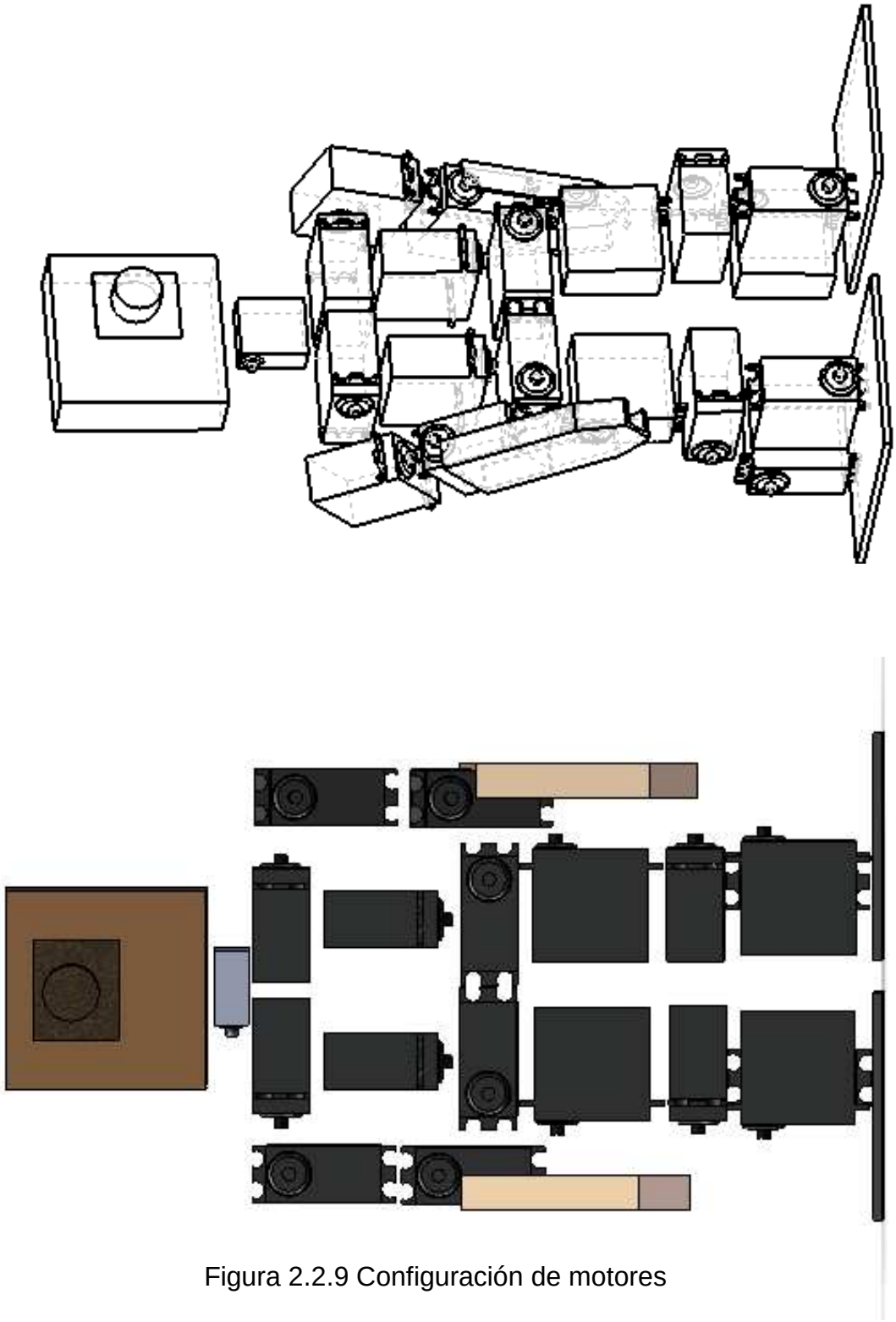

Figura 2.2.9 Configuración de motores

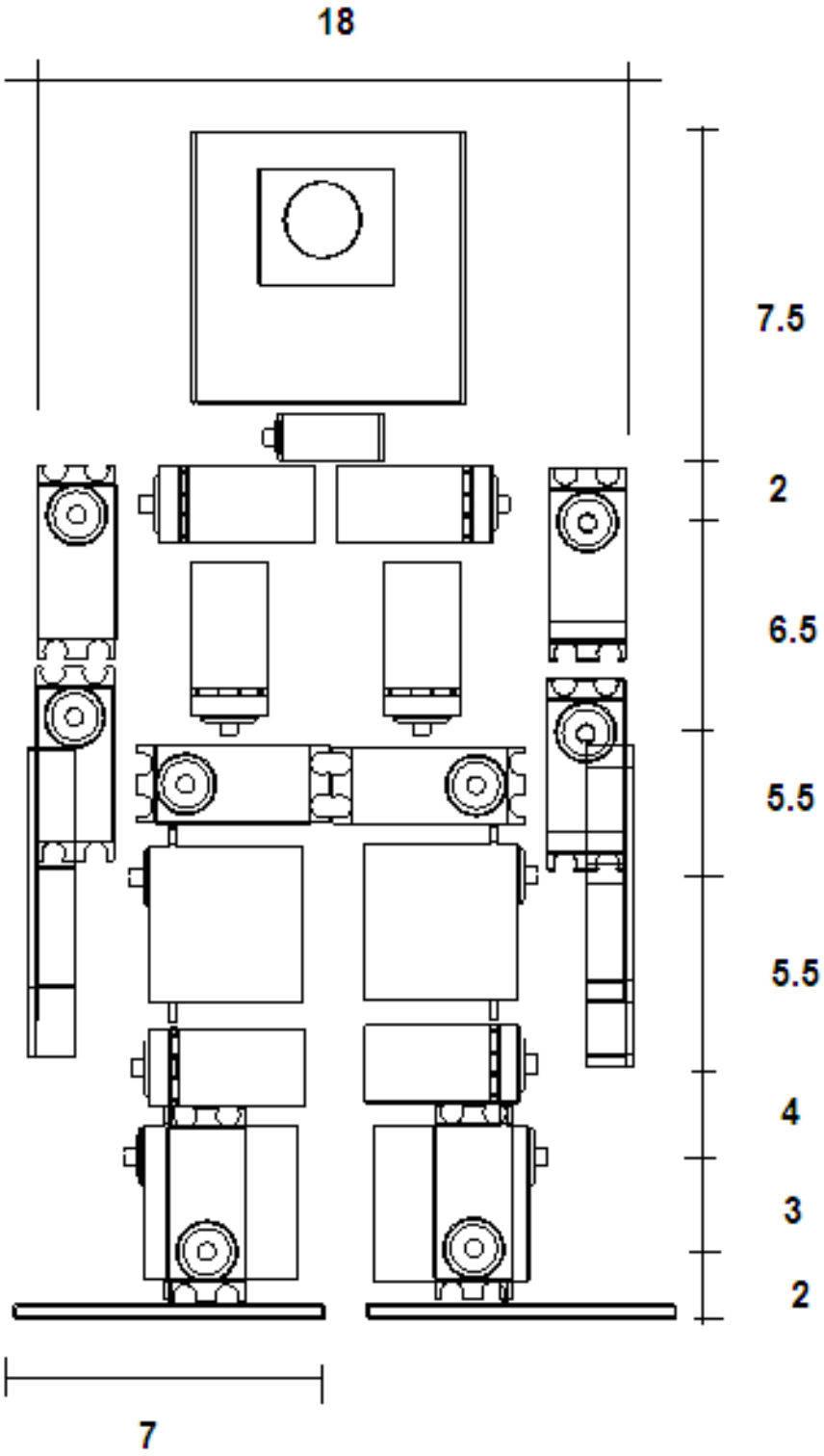

Figura 2.2.10 Medidas del humanoide en centímetros

El peso de esta estructura es de 1.675 kg con baterías y sin baterías de 1.565 kg, el centro de masa se encuentra a una altura de 14.2 cm y el consumo de corriente en posición de firmes es de 1300mA. En la Figura 2.2.11 se observa el método para calcular la torca máxima del servomotor dolocaso en

el tobillo. El peso es de 1.675 , la distancia a el punto de volteo 6.5 cm y la torca resultante es de 10.88 kg*cm.

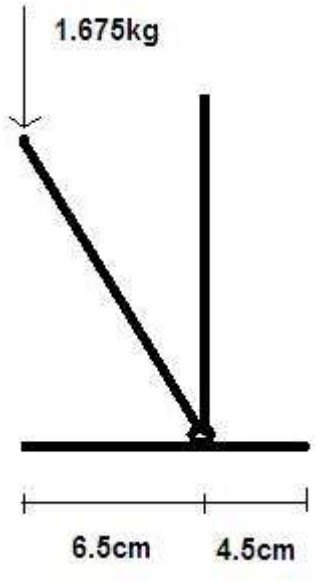

Figura 2.2.11 Diagrama para calcular la torca máxima

La estructura de la Figura 2.2.8 es la más estable debido a que los motores tienen un torque de 16.5kg*cm y la torca máxima del tobillo de 10.88 kg*cm, un consumo de corriente cuando da un paso no mayor a 1300mA. El juego en cada articulación es muy reducido dando como resultado una estructura muy firme, por lo anterior se concluyó que esta estructura es la idónea para cumplir las metas propuestas.

2.3 Servomotores Dynamixel AX-12+.

El servo Dynamixel AX-12+ es mucho más que un servo digital, es un componente robótico sofisticado. Cada servo tiene la capacidad de controlar su velocidad, temperatura, posición, tensión y carga soportada. El algoritmo de control utilizado para mantener la posición de cada servo puede ajustarse individualmente, permitiendo el control retroalimentado de la velocidad y la carga que soporta cada actuador.

Cada actuador Dynamixel AX-12+ tiene un microcontrolador que entiende 50 comandos, la mayoría de los cuales fijan o leen parámetros que definen su comportamiento. El típico servo de radiocontrol de Hobby sólo entiende la orden "ángulo objetivo" (dada por una señal PWM), pero estos motores permiten utilizarlos como un actuador profesional con sensores. Inclusive leer el estado del motor como puede ser la posición actual, la corriente consumida, o la variación de la temperatura del servo con la carga aplicada en el mismo, lo que permite un control retroalimentado sofisticado controlando el par que soporta cada articulación del robot. Esto tiene aplicaciones, por ejemplo en los robots bípedos, ya que sin necesidad de inclinómetros o acelerómetros, se pueden conseguir efectos de equilibrio [26].

Figura 2.3.1 servomotor Dynamixel AX-12+ utilizado

Estos motores cuentan con 2 arneses en la parte posterior, en los cuales los pines están conectados punto a punto como se muestra en la Figura 2.3.2 [26.]

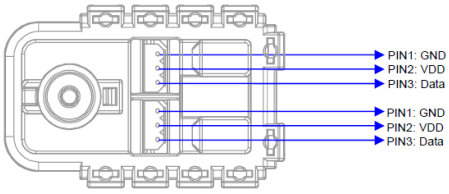

Figura 2.3.2 Disposición de terminales del servomotor AX-12+

La comunicación de los motores es serial asíncrona a 1 megabit por segundo Half Duplex para la cual se necesita construir un circuito como el de la Figura 2.3.3 [26].

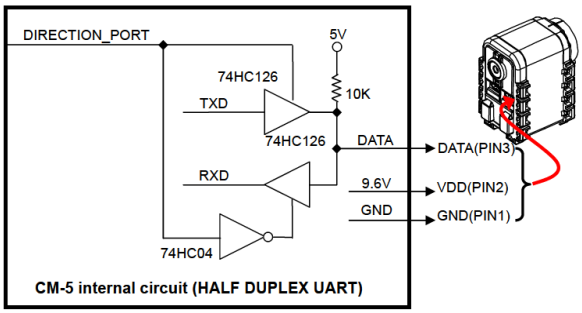

Figura 2.3.3 Circuito interfaz entre el microcontrolador y el servomotor

Para la posición se cuenta con una resolución de 1024 pasos para 300 grados, en la Figura 2.3.4 [26] se muestra la posición de cada paso.

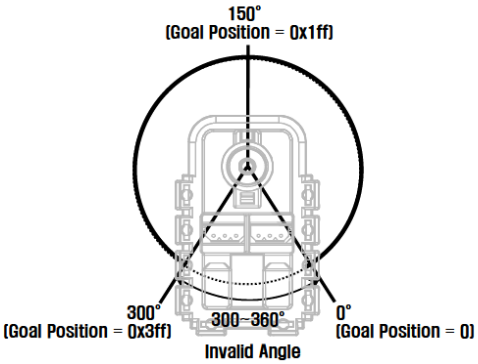

Figura 2.3.4 Disposición de las posiciones válidas

2.4 Características Del Microcontrolador PIC18F452

Bajo la denominación de PIC18F452 se hace referencia a una subfamilia de microcontroladores PIC de la gama alta, que se identifica por tener como memoria de programa una del tipo FLASH y una serie de recursos semejante a los modelos mas potentes.

Dos de los cuatro modelos que componen esta subfamilia están encapsulados con 28 patitas (18F252), mientras que los otros dos tienen 40 patitas (PIC18F452). Con la intención de seguir potenciando la línea con memoria FLASH.

A continuación se muestran las características relevantes del PIC18F452:

1.- Disponen de 5 puertos PA de 6 bits, PB de 8 bits, PC de 8 bits, PD de 8 bits, PD de 8 bits y PE de 3 bits E/S con un total de 33 líneas para conectar los periféricos exteriores.

2.- Canal A/D con 8 canales de entrada.

3.-Puerto paralelo esclavo.

Los recursos fundamentales se enuncian a continuación:

1.- Procesador de arquitectura RISC avanzada con arquitectura HARVARD

La arquitectura Harvard dispone de dos memorias independientes, una que contiene sólo instrucciones y otra sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias [20].

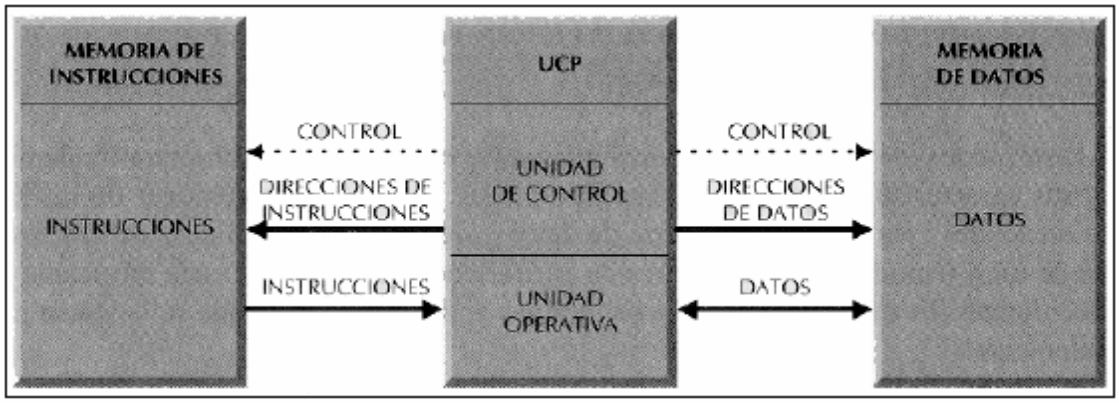

Figura 2.4.1 Buses para instrucciones y datos independientes

2.- Juego de 77 instrucciones con 16 bits de longitud. Todas ellas se ejecutan en un ciclo de instrucción, menos las de salto que tardan dos.

3.- Frecuencia de operación hasta de 40 MHz.

4.- Hasta 32K palabras de 16 bits para memoria de código, tipo FLASH.

Es una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos y es más pequeña. A diferencia de la ROM, la memoria FLASH es programable en el circuito.

5.-Hasta 1536 bytes de memoria de datos RAM (Memoria de acceso aleatorio). Se utiliza para guardar los datos temporales que se necesitan en la ejecución del programa. En esta sección se alojan los registros operativos fundamentales en el funcionamiento del procesador y en el manejo de todos sus periféricos, además de registros que el programador puede usar para información de trabajo propia de la aplicación [30].

6.- Hasta 256 bytes de memoria de datos EEPROM (Electrical Erasable Programmable Read Only Memory). Es una memoria que una vez instalada en un circuito pueden grabarse y borrarse cuantas veces sea necesario sin ser retirados de dicho circuito. Para ello se usan “grabadores en circuito” que

confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo. Sin embargo se garantizan hasta 1000,000 de ciclos de escritura/borrado.

7.- Encapsulados compatibles con los PIC16F877.

8.- Modos de direccionamiento, indirecto y relativo.

9. - Perro Guardián (WDT) o "Watch Dog Timer ". Cuando el PC se bloquea por un fallo del software u otra causa se pulsa el botón del reset y se reinicia el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continua las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema. Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y al completar su temporización provocara el reset [30].

10.- Código de protección programable.

11.- Modo SLEEP de bajo consumo. Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, hasta que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los PIC disponen de una instrucción especial (SLEEP), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se "congelan" sus circuitos asociados, quedando sumido en un profundo "sueño" el microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo [30].

12.- Programación serie en circuito con 2 patitas.

13.- Voltaje de alimentación comprendido entre 2 y 5.5 V.

14.- Bajo consumo de energía (menos de 2mA a 5V y 5MHz).

Dispositivos periféricos.

1.- Timer0: Temporizador- contador de 8 bits con predivisor de 8 bits.

2.- Timer1: Temporizador- contador de 16 bits con predivisor.

3.- Timer2: Temporizador-contador de 8 bits con predivisor y postdivisor.

4.- Timer3: Temporizador-contador de 16 bits

5.- Dos módulos de Captura-Comparación-PWM.

6.- Conversor A/D de 10 bits.

7.- Puerto serie asíncrono (SSP) con SPI e I2C.

8.- USART

9.- Puerta Paralela Esclava (PSP).

Descripción de las terminales del microcontrolador PIC18f452

En la Figura 2.4.2. [30] se muestra el diagrama de asignación de pines del PIC18F452.

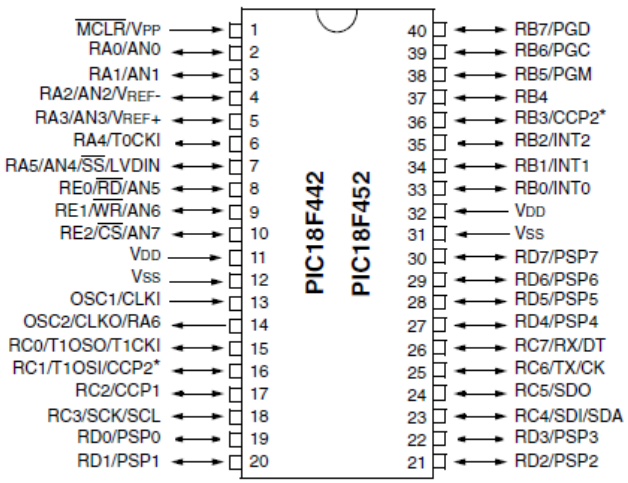

Figura 2.4.2 Diagrama de asignación de pines del PIC18F452

La empresa Microchip y otras que utilizan los PIC ponen a disposición de los usuarios numerosas herramientas para desarrollar hardware y software. Son muy abundantes los programadores, los simuladores de software, los emuladores en tiempo real, ensambladores, compiladores C, intérpretes y compiladores BASIC. Las razones de la excelente aceptación de los PIC son las siguientes:

1.- Sencillez de manejo: Tienen un juego de instrucciones de 77 en la gama alta.

2.-Buena información, fácil de conseguir y económica.

3.-Precio: Su costo es comparativamente inferior al de sus competidores.

4.-Poseen una elevada velocidad de funcionamiento. Buen promedio de

Parámetros: velocidad, consumo, tamaño, alimentación y código compacto.

5.- Herramientas de desarrollo fáciles y baratas. Muchas herramientas de software se pueden recoger libremente a través de Internet desde Microchip.

6.- Existe una gran variedad de herramientas de hardware que permiten grabar, depurar, borrar y comprobar el comportamiento de los PIC.

7.- La gran variedad de modelos de PIC permite elegir el que mejor responde a los requerimientos de la aplicación.

2.5 Herramientas Para Trabajar Con Microcontroladores PIC.

Existe una gran gama de herramientas en Internet para trabajar con los microcontroladores de microchip, aparte de la gran cantidad de información y libros de texto.

MPLAB IDE Y MPLAB ICD 2

En la parte de software podemos utilizar el programa MPLAB IDE de microchip, este lo podemos bajar de la página www.microchip.com, y lo más importante, es gratuito.

El MPLAB-IDE es una Plataforma de Desarrollo Integrada bajo Windows, con múltiples prestaciones, que permite escribir el programa en lenguaje assembler o C (el compilador C se compra aparte), crear proyectos, ensamblar o compilar, simular el programa y finalmente programar el componente, si es que se cuenta con el PICSTART-PLUS o alguna otra herramienta para programar como MPLAB ICD 2 [21].

Partes del MPLAB-IDE:

EDITOR: Editor incorporado que permite escribir y editar programas u otros archivos de texto.

PROJECT MANAGER: Organiza los distintos archivos relacionados con un programa en un proyecto. Permite crear un proyecto, editar y simular un programa. Además crea archivos objetos y permite bajar archivos hacia emuladores (MPLAB-ICE) o simuladores de hardware (SIMICE).

SIMULADOR: Simulador de eventos discretos que permite simular programas con ilimitados breakpoint, examinar/modificar registros, observar variables, tiempos y simular estímulos externos.

ENSAMBLADOR: Genera varios tipos de archivos objetos y relacionados, para programadores Microchip y universales.

LINKER: Permite unir varios archivos objetos en uno solo, generados por el ensamblador o compiladores C como MPAB-C18 o compiladores de terceros.

INTERFACE A HARDWARE COMO: PICSTART-PLUS, PROMATE II, MPLAB-ICD, MPLAB-ICE, ICEPIC, SIMICE, OTROS

MPLAB-ICD2 es un debugger y programador a la vez, puede cumplir ambas funciones. Pero cuando trabaja en conjunto con el MPLAB-IDE, cumple una u otra función, no ambas a la vez.

La idea del ICD2 es realizar el debugging y la programación del uC IN-CIRCUIT, es decir, directamente en la placa. El procesador puede estar incluso soldado.

ICD2 se comunica con MPLAB-IDE vía puerto serial o USB (recomendado). Por otro lado, debe ser conectado al procesador. Para ello hace uso de algunos recursos del uC: en los PIC16/18 RB6/PGC, RB7/PGD, VppMCLR, Vdd y Vss como se muestra en la Figura 2.5.1. [31]

También utiliza algunos recursos de RAM y SP. Los recursos de RAM y SP dependerán de uC en particular. Para más detalles, consultar la ayuda del MPLAB-IDE.

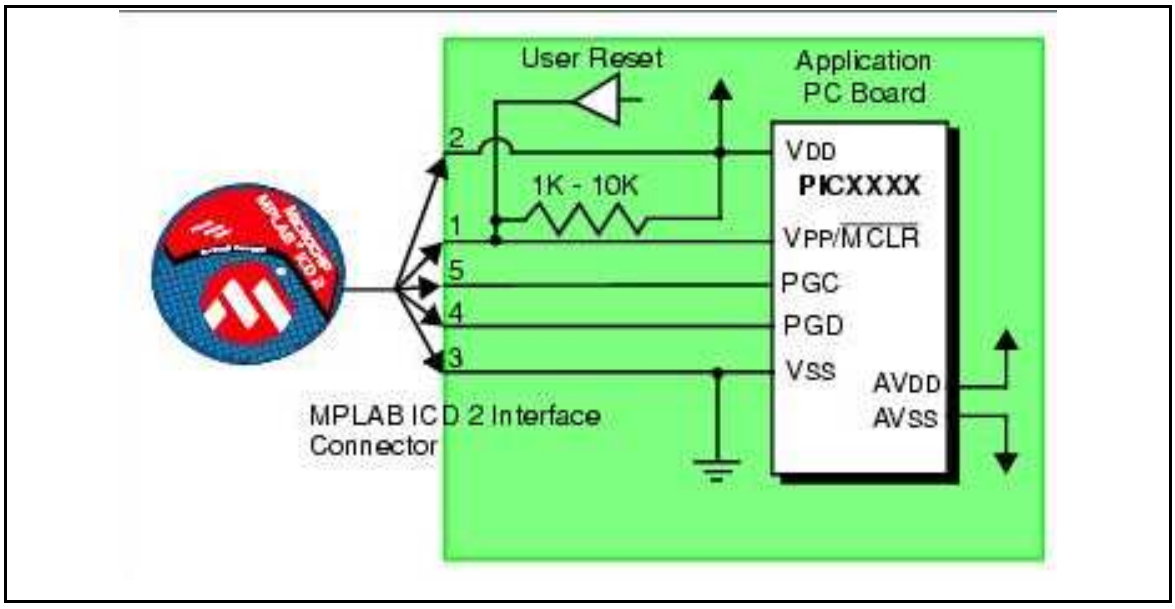

Figura 2.5.1 Conexión del MPLAB ICD 2 con un PIC

El voltaje Vpp (13.5V) es suministrado por el ICD2. El ICD2 puede ser alimentado desde la placa del usuario (recomendado) o desde el puerto USB. En el primer caso, las señales Vdd y Vss, son suministradas por la placa del usuario.

El ICD2 se puede utilizar de 2 formas:

- Como programador serial para poner el código en la memoria de programa de uC.
- Como debugger, que permite realizar una simulación a nivel de hardware del código, poner break points. El debugger requiere utilizar el In Circuit Deduger hardware que está presente dentro de los micros.

Para el Debugging y programación, el ICD2 hace uso de los pines PCG y PGC.

Para los efectos de programación, el ICD2 no necesita que el uC tenga asociado un oscilador (xtal o similar). Además suministra el voltaje Vpp, ingresado al uC por medio del pin MCLR/Vpp, un clock por medio del pin PGC y los datos seriales por el pin PGD.

El modo hace uso de algunos recursos del procesador. En el caso del PIC16F877A, el ICD2 hace uso de:

- Memoria de Programa: 1F00h – 1FFFh
- Ram: 70h, F0h,170h,1E5h-1F0h
- SP: 2 niveles

MPLAB IDE puede trabajar con varios tipos de programadores. El usuario debe seleccionar con cuál trabajará, haciendo click en opción Programmer/ Select programmer, se pueden seleccionar 3 programadores distintos:

- Picstart-plus
- MPLAB-PM3
- Promate II
- MPLAB-ICD2

Antes de seleccionar el ICD2 como debugger o programador, seleccione el micro con el que trabajará, por medio de la opción Configure/Select Device.

Para seleccionar el ICD2 como programador, escoja la opción Programmer/Select programmer/MPLAB ICD2 como se muestra en la Figura 2.5.2.

MPLAB-IDE detecta si el Sistema Operativo del ICD2 corresponde al uC seleccionado, si no corresponde, lo bajará automáticamente y luego establecerá comunicación con el uC.

Cualquier error de comunicación, conexiones o falta de alimentación, provocará que aparezca en pantalla la respectiva ventana de error con el detalle.

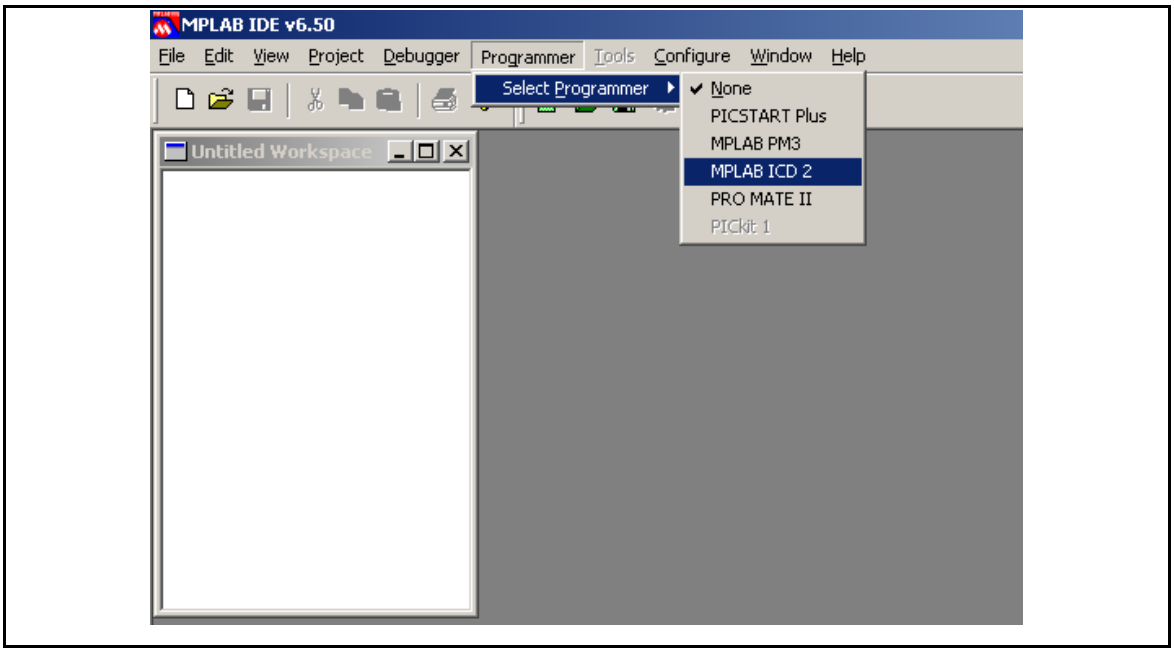

Figura 2.5.2 Selección del MPLAB ICD 2 como programador

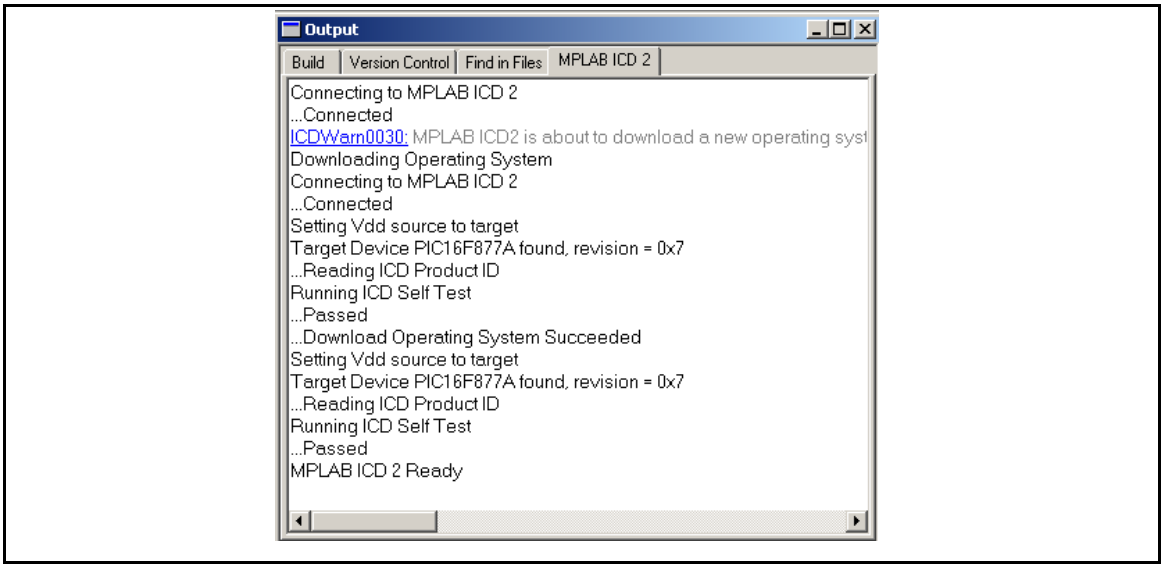

Figura 2.5.3 Verificación de la conexión y el tipo de uC por el MPLAB IDE

La creación de un proyecto comienza con la escritura del programa. Para ello escoja la opción *File/New* y el editor del MPLAB-IDE presentará una página en blanco. Es sugerible que ponga nombre al archivo fuente desde el principio.

Para ello, escoja la opción *File/Save As*, póngale nombre al programa y agregue la extensión .bas, le aparecerá una pantalla como la que se muestra en la Figura 2.5.4.

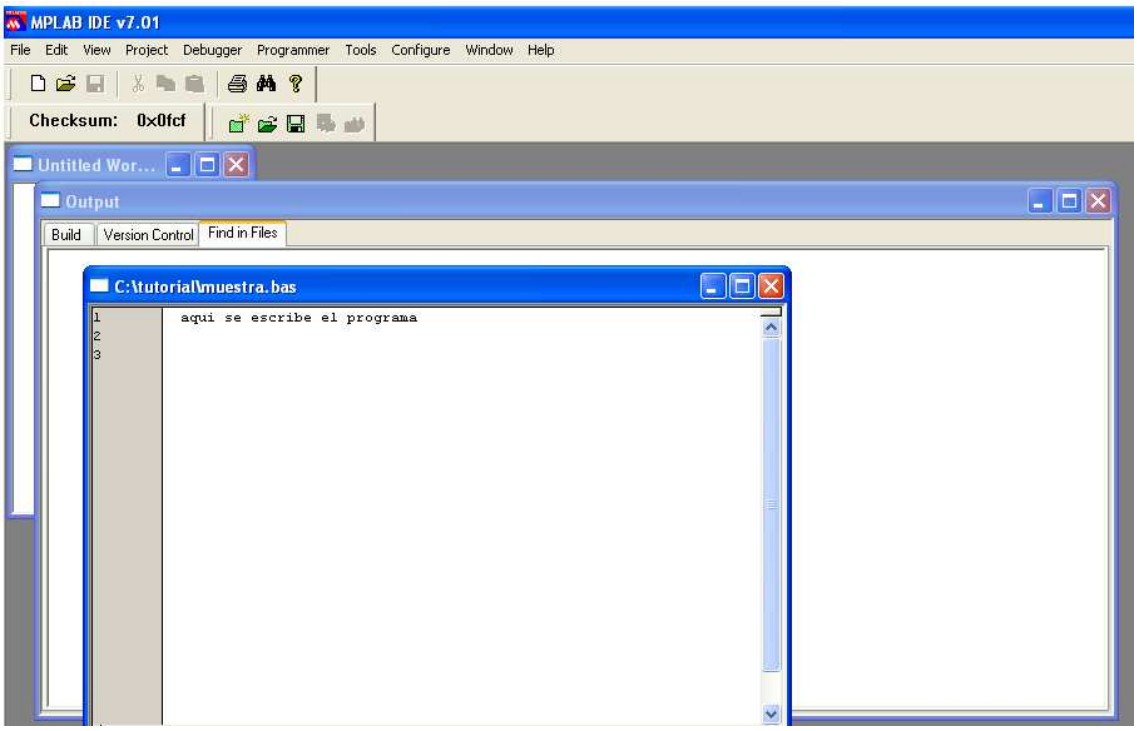

Figura 2.5.4 Primer paso para crear un nuevo proyecto en MPLAB IDE

Una vez escrito el programa, es necesario crear un proyecto para poder compilar, simular, debugear y programar.

Para crear un proyecto, escoja la opción *Proyect/Proyect Wizard* y siga las instrucciones. Seleccione el procesador con el que desea trabajar Seleccione el lenguaje que usará (Assembler, C ó Basic). Póngale nombre al proyecto, de preferencia, el mismo del archivo fuente, luego, indique en qué directorio quedará el proyecto, se sugiere, el mismo donde se encuentra el archivo fuente. Busque y agregue el archivo fuente al proyecto. Una vez que lo encuentre, selecciónelo y haga click en *Add* y aparecerá en la venta derecha. Si el archivo fuente se encuentra en un directorio distinto al directorio que definió en el paso anterior para el proyecto, haga click en el checbox de la derecha. Esto provocará que el MPLAB haga una copia del archivo fuente en el

directorio del proyecto. En las Figuras 2.5.5 a 2.5.8 se muestran las pantallas detalladas de los pasos antes mencionados.

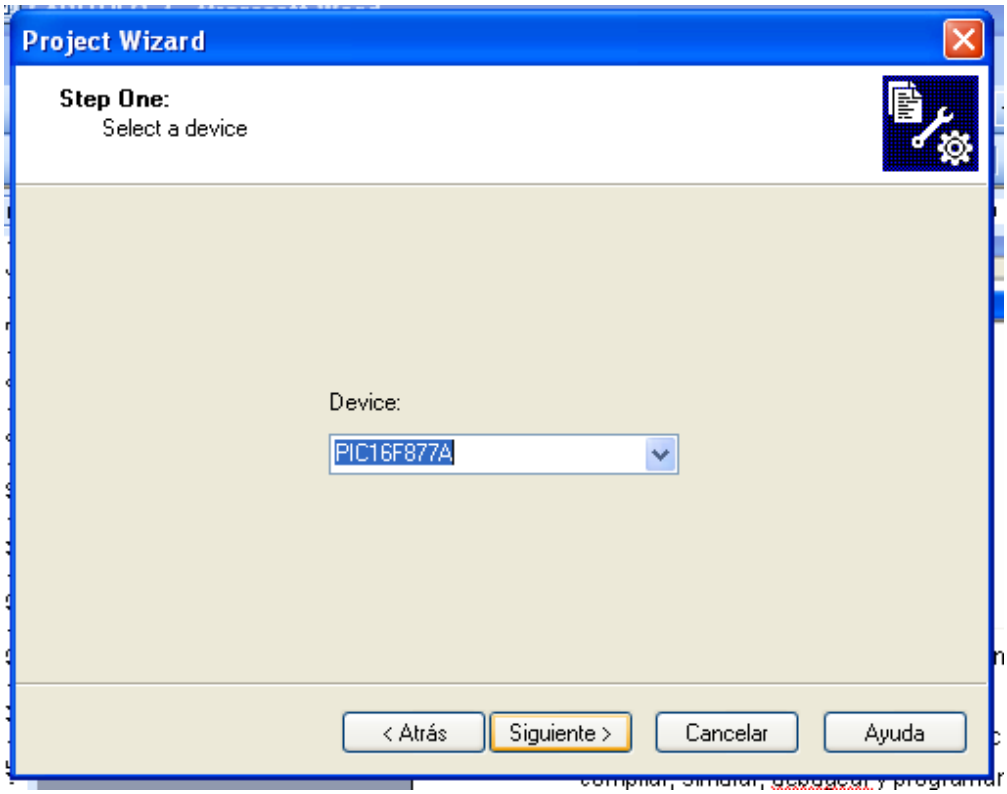

Figura 2.8.5 Pantalla donde podemos elegir el procesador a utilizar

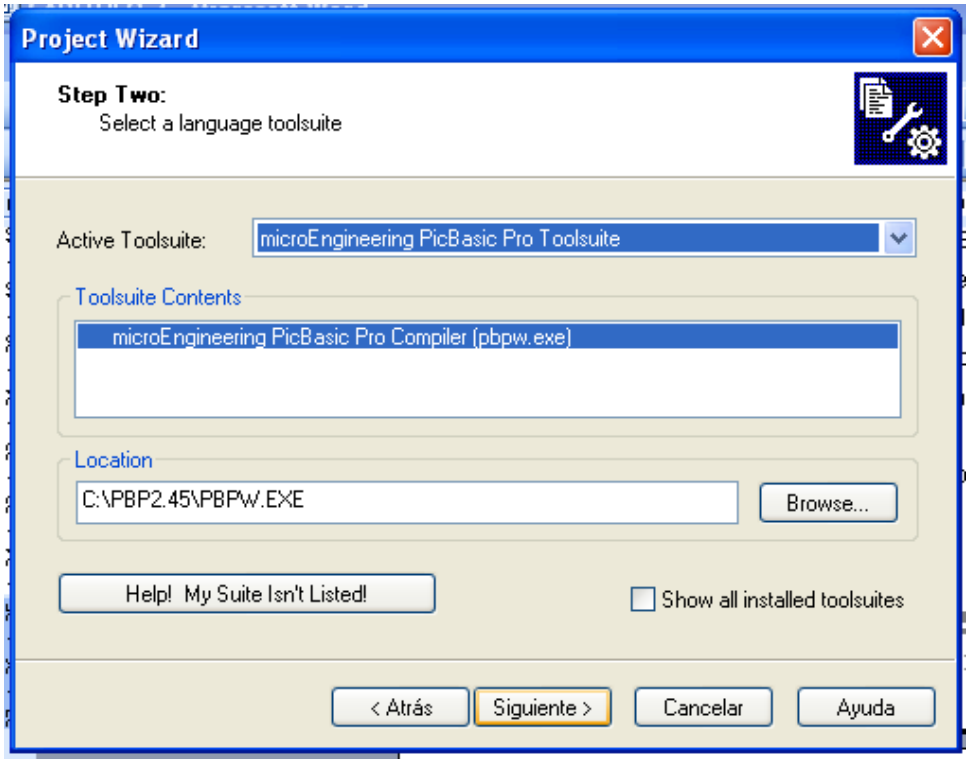

Figura 2.8.6 Pantalla en la cual elegimos el lenguaje de programación

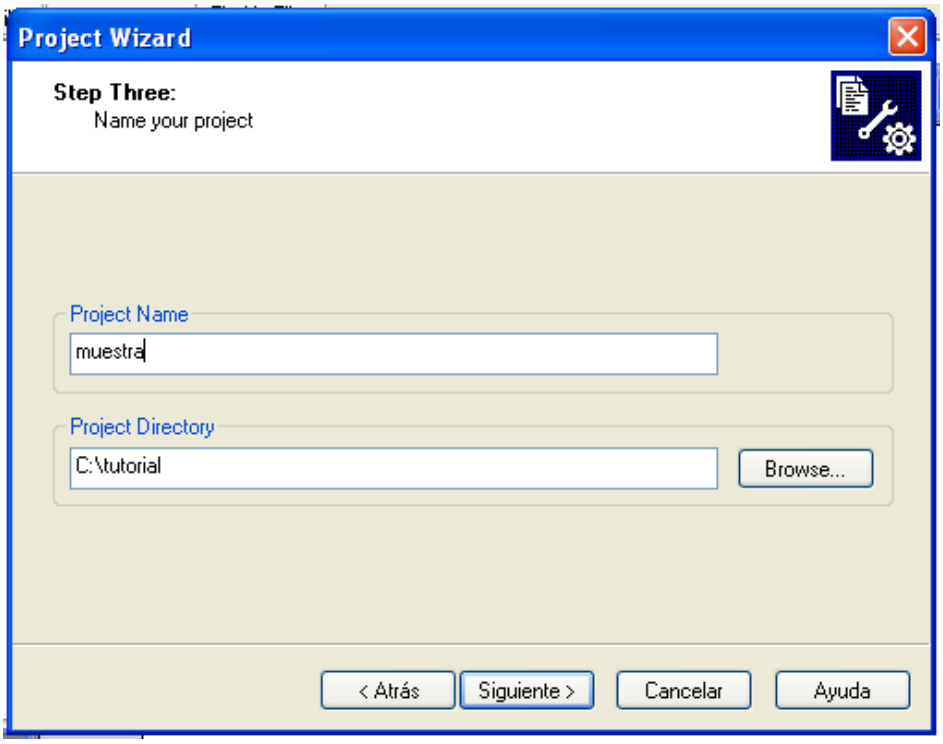

Figura 2.5.7 Elegimos el directorio donde se guardara el proyecto

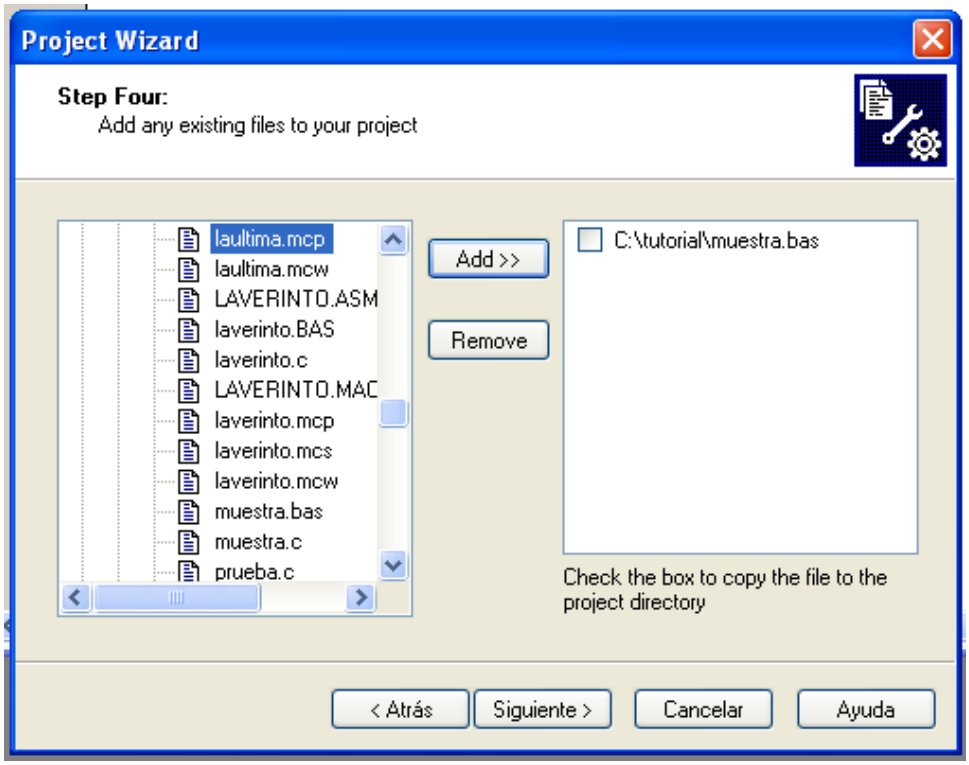

Figura 2.5.8 Buscamos y agregamos el archivo fuente del proyecto

Si todo es correcto, se crea un archivo con el nombre del proyecto y extensión mcp, por ejemplo, *timer.mcp*. Se debe leer el resumen para ver si todo está configurado de forma adecuada. En la Figura 2.5.9 se muestra la pantalla final del proceso.

De ser así, se hace click en el botón finalizar, de lo contrario, cancele y repita el proceso.

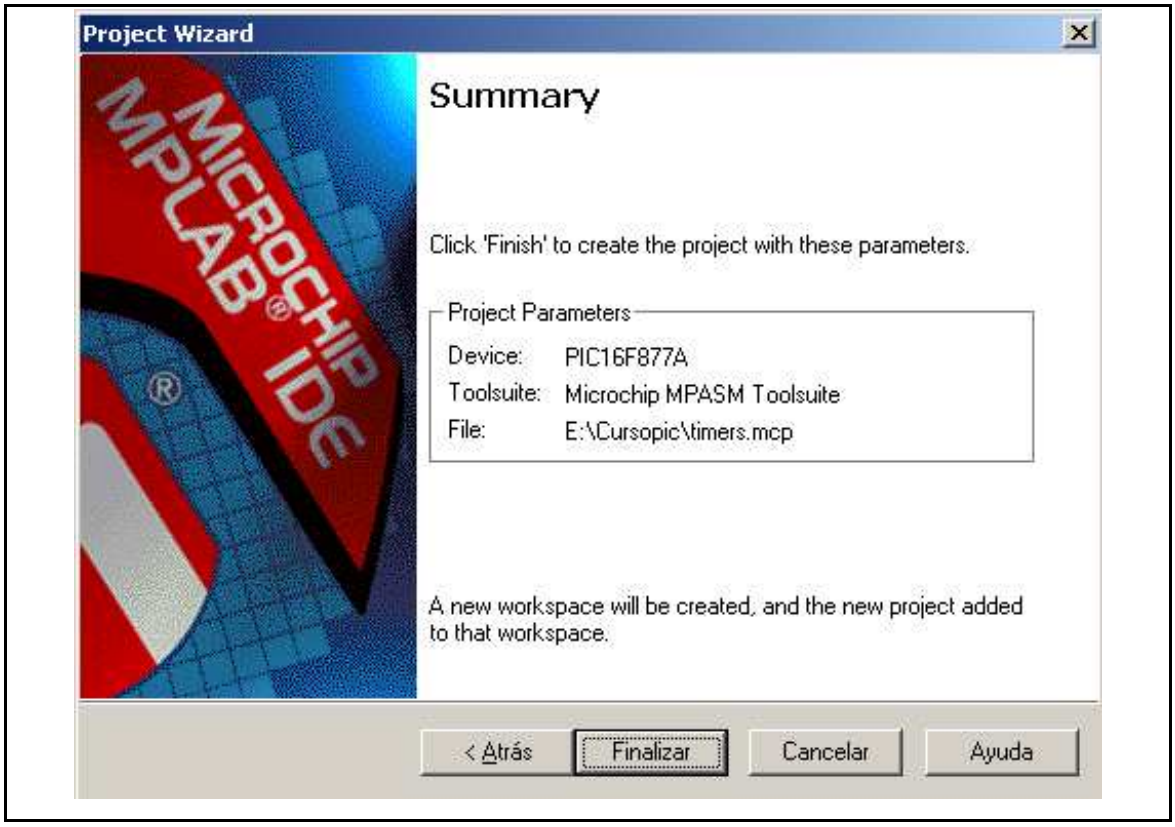

Figura 2.5.9 Datos del proyecto hecho en MPLAB IDE

Una vez creado el proyecto, se escribe el programa, en el lenguaje seleccionado anteriormente y se hace click en el ícono Built All. Con esto compilará el programa y se cerrarán archivos de error, mapa del programa, archivos objetos y archivos hex.

Si aparece algún error de compilación, se debe verificar el código.

Se escoge el programador deseado (en este caso el ICD2) por medio del menú *Programmer/Select programmer.*

ICD2 verificará el micro escogido y las conexiones respectivas. Cuando el programador es seleccionado, los íconos relacionados con él aparecerán.

Luego de establecer comunicación con el programador, es necesario configurar los BITs de CONFIGURACIÓN de micro. Las opciones disponibles varían de micro en micro.

Es sumamente importante tener claro los valores de los bits de configuración, de lo contrario, el micro no trabajará adecuadamente, aunque el programa esté bien estructurado. Para configurar los Bits de configuración, el usuario debe hacer click en *Configure/Configuration Bits* y aparecerá una pantalla como la mostrada en la Figura 2.5.10

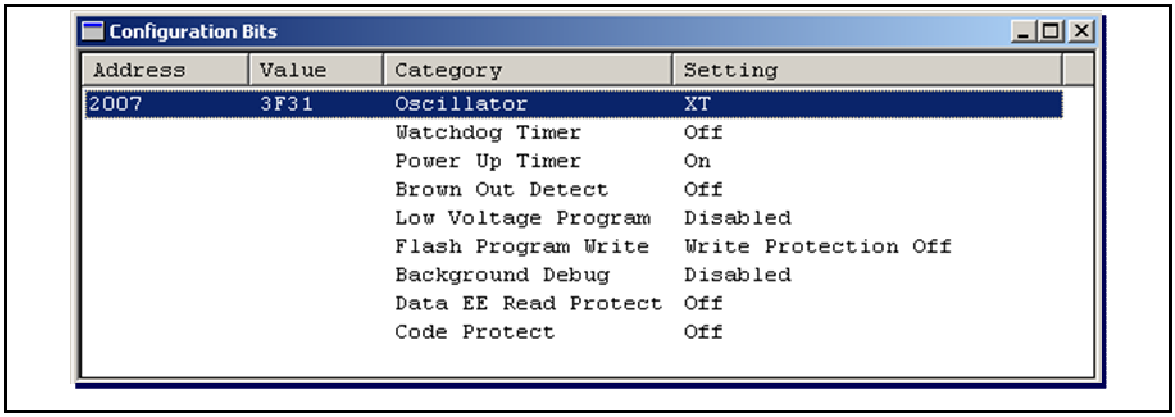

Figura 2.5.10 Pantalla para la configuración de los bits de configuración

Sólo una vez teniendo los bits de configuración en los valores deseados, se puede proceder a programar el micro escogiendo la opción *Programmer/Program*

Si el proceso no presenta problemas, la ventana de salida mostrará un mensaje que indica que la programación fue exitosa.

Antes de programar, en ocasiones es necesario, configurar el programador. Este, por defecto, programa sólo la cantidad de memoria suficiente, dependiendo del tamaño del programa.

Una vez configurado el programador, se da click sobre el ícono Programar uC para iniciar el proceso de programación.

El programador realiza los siguientes pasos:

- Borra el uC.
- Programa la memoria de acuerdo al rango definido en la configuración del programador.
- Verifica si la programación es correcta.
- Programa bits de configuración.
- Verifica programación de bits de configuración.

En la Figura 2.5.11 se muestra el seguimiento de la programación del PIC cuando fue exitosa.

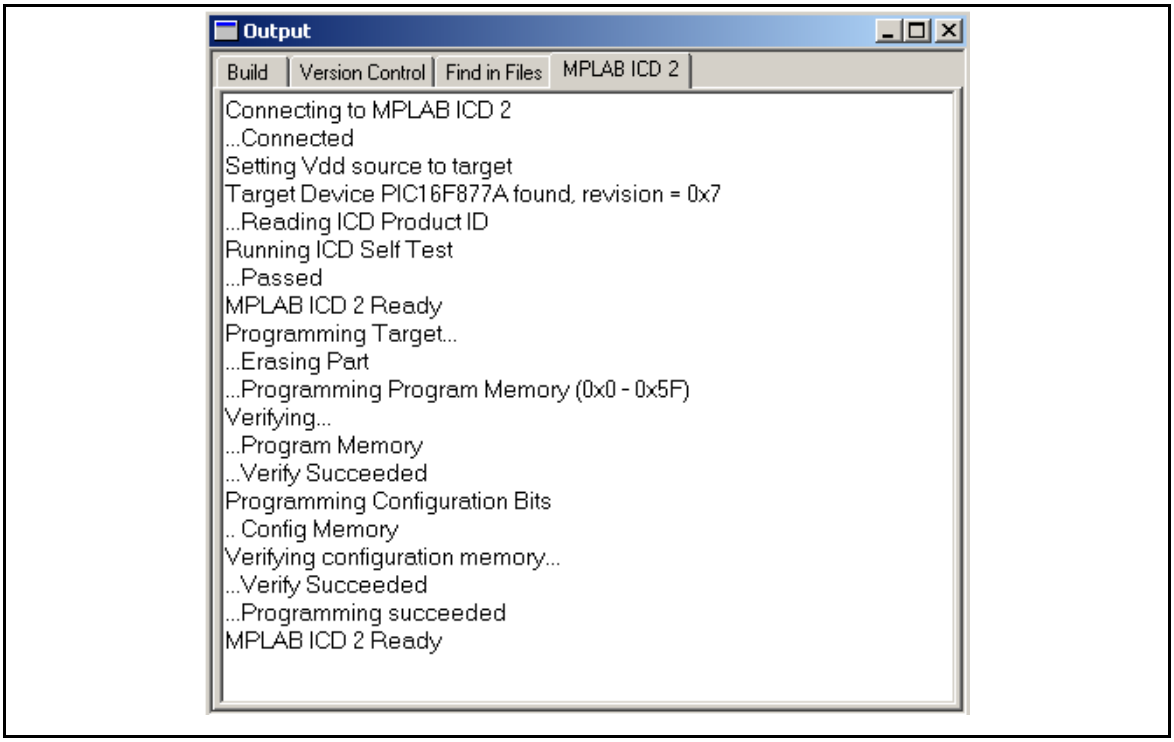

Figura 2.5.11 Ventana de programación del MPLAB IDE con MPLAB ICD

Si desea realizar el debugging del programa, primero se deshabilita el ICD2 si es que fue seleccionado como programador, luego se escoge la opción *Debugger/Select Tool/MPLAB ICD 2*. ICD2 establecerá nuevamente comunicación con el micro, verificará conexiones y alimentación y se activarán los íconos del ICD2. La parte de debugging es muy importante para el proyecto, dado que en el momento de hacer pruebas ya que no se podian conocer los valores que está recibiendo el uC en tiempo real, y así corregir detalles del hardware con el software. En muchas ocasiones, el programa está correcto, y los valores que esta recibiendo el uC son incorrectos, y esto puede provoca pérdida de tiempo en la búsqueda de errores en el programa.

Lenguaje de programación PIC BASIC PRO

El compilador Pic Basic Pro (PBP) es un lenguaje de programación de nueva generación que hace más fácil y rápido programar el micro controlador Pic micro de **Microchip Technology** .

El lenguaje Basic es mucho más fácil de leer y escribir que el lenguaje ens amblador.

PBP por defecto crea archivos que corren en un PIC 16F84-04/P con un reloj de 4 Mhz. Solamente muy pocas partes son necesarias dos capacitores de 22 pf para el cristal de 4Mhz un resistor de 4.7K en el pin/MCLR y una fuente de 5 volts. Otros uC PIC además del 16F84, así como otros osciladores de frecuencias distintas pueden ser usados por este compilador.[12]

El PBP produce código que puede ser programado para una variedad de microcontroladores PIC que tengan de 8 a 68 pines y varias opciones en el chip incluyendo convertidores A/D, temporizadores y puertos seriales.

Hay algunos micros PIC que no trabajaran con el PBP, por ejemplo las series PIC 16C5X incluyendo el PIC 16C54 Y PIC 15C58. Estos micros PIC están basados en el viejo núcleo de 12 bits en lugar del núcleo más reciente de

14 bits. El PBP necesita alguna de las opciones que solamente están disponibles con el núcleo de 14 bits como el stack (pila) de 8 niveles [22].

Hay muchos micros PIC, algunos compatibles pin a pin con la serie 5X, que pueden ser usados con el PBP. La lista incluye PIC16C554, 556, 558, 61, 62(A), 620, 621, 622, 63, 64(A), 65(A), 71, 710, 711, 715, 72, 73(A), 74(A), 84, 923, 924, el PIC16F83 y 84, el PIC12C671 y 672 y el PIC14C000, y Microchip sigue agregando otros. Para reemplazo directo de un PIC16C54 o 58, el PIC16C554, 558, 620 y 622 funcionan bien con el compilador y tienen aproximadamente el mismo precio [22].

El PIC16F877 contiene 256 bytes de memoria de datos no volátil que puede ser usada para archivar los datos de programa y otros parámetros, aún cuando no haya energía. A ésta área de datos, se puede acceder simplemente usando las órdenes “Read” y “Write” del PBP. (El código del programa es permanentemente guardado en el espacio de código del micro PIC, tanto si hay o no energía.)

Entre las instrucciones que nos interesan más están las de modulación por ancho de pulso (para controlar la velocidad de los motores):

El primer parámetro es el canal por el cual va a mandar el tren de pulsos, en este caso es CCP1, el segundo parámetro es un valor que va de 0 a 255 y es el valor del porcentaje en estado alto, en este caso 127 corresponde a un 50%, y por último el tercer parámetro es la frecuencia de los pulsos, en este caso es de 1kHz [22].

```
Hpwm 1,127,1000       ' Send a 50% duty cycle PWM signal at 1kHz
```

Convertidor analógico digital (para leer los voltajes de los sensores):

En esta instrucción, primero es necesario definir 3 parámetros del convertidor A/D: número de bits del resultado, en este caso son 10

(recordemos que también podemos utilizarlo con 8 bits de resolución), tipo de reloj a utilizar, en este caso es el tipo 3 = RC y el tiempo de muestreo que en este caso son 50 uS estos valores deben de estar definidos en el encabezado de nuestro programa (recomendado), aunque pueden definirse en cualquier parte de nuestro programa.

```
Define ADC_BITS    10      ' Número de bits a convertir

Define ADC_CLOCK   3       ' Fuente del reloj (3=rc)

Define ADC_SAMPLEUS 50     ' Tiempo de muestreo en uS
```

La instrucción ADCIN tiene 2 parámetros, el primero es el canal de conversión, y el segundo es la variable “valor” que tuvo que haber sido definida en nuestro encabezado del programa, esta debe tener la longitud suficiente para guardar el valor del resultado de la conversión, para 8 bits debe de definirse de tamaño BYTE, para 10 bits debe definirse de tamaño WORD. [22]

```
ADCIN 4, valor
```

Las instrucciones de acceso a la memoria EEPROM. (ya que en este lugar guardaremos la ruta de nuestro robot)

```
READ AP, REAL      ‘lee la dirección AP y lo guarda en la variable REAL

WRITE AP, CRUCE ‘escribe el valor de la variable CRUCE en la dirección AP
```

Estas son las instrucciones que nos facilitaran en gran medida nuestra parte de programación, las demás instrucciones que utilizaremos las iremos explicando en nuestro programa como comentarios.

Instalación del software

Dependiendo de la versión del software, se puede instalar copiando los archivos a una carpeta en c llamada PBP, o sea "C:\PBP". Cuando el software trae un ejecutable, este se encarga de instalarlo correctamente, sin embargo, para ejecutarlo desde MPLAB IDE, necesitamos bajar unos plugins que podemos bajar de la página http://www.melabs.com/ downloads/PBPlugins.zip, que es la página de la empresa creadora de PBP MICRO ENGINEERING LABS, INC. El archivo es "PBregister.bat" el cual lo tenemos que ejecutar en el directorio C:\PBP\PBregister.bat, cerramos MPLAB y lo volvemos a ejecutar, y listo, aparecerá como lenguaje compatible. Vea la Figura 2.5.12 [23].

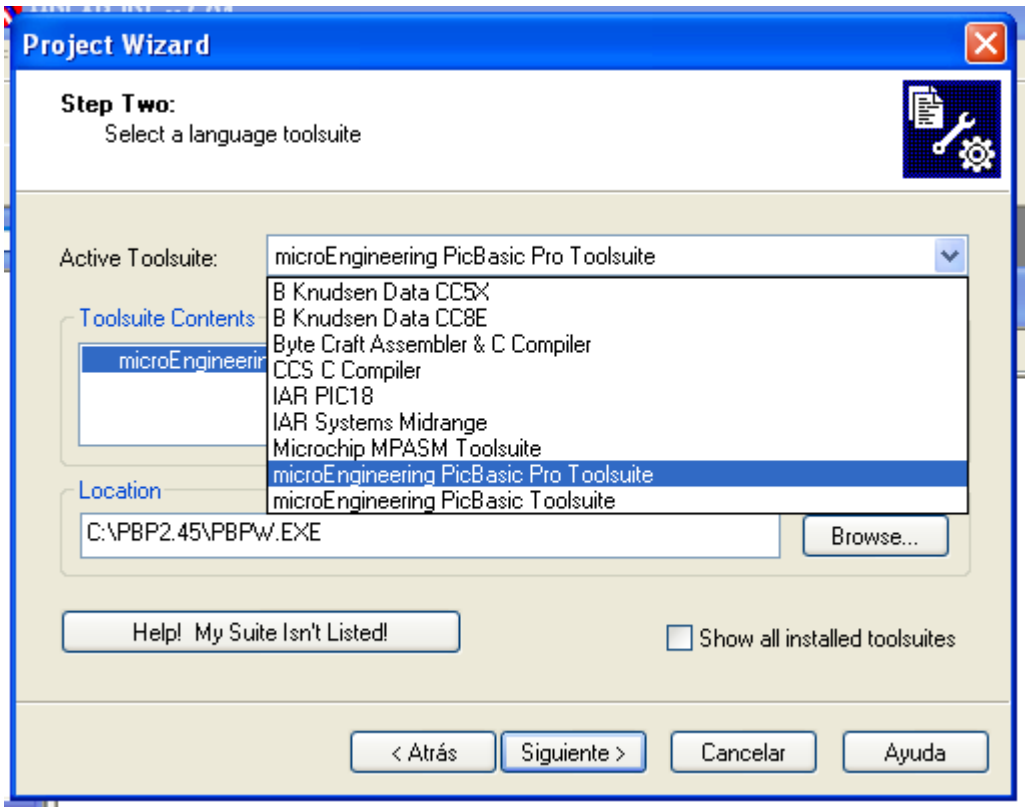

Figura 2.5.12 MPLAB reconoce PBP como lenguaje compatible

Otras herramientas para programar PICS Y DSPICS

Existe en Internet una gran gama de herramientas y software para programar microcontroladores PIC, y ahora DSPIC, entre los más famosos para programar PIC, se encuentra el llamado JDM Programer, que funciona con un programa llamado ICPROG, estas herramientas las podemos encontrar en Internet en la página http://www.ic-prog.com/, existe otro programador de PIC y de DSPIC que funciona por puerto USB llamado GTP-USB que trabaja con un software llamado WINPIC800 el cual podemos descargar libremente desde Internet desde http://perso.wanadoo.es/siscobf/ , u otras páginas.

Es importante contar con herramientas de bajo costo para trabajar con microcontroladores, ya que para gran parte de los estudiantes, resulta difícil adquirir herramientas de grabación caras. La diferencia del precio entre construir un programador como los vistos aquí y comprar uno de fábrica es considerable.

ALGORTIMOS E IMPLEMENTACIÓN FÍSICA

3.1 Estructura General Del Robot

El robot cuenta básicamente con la parte mecánica la cual ya se explicó en el capítulo dos, como pudo observarse, se construyó con servomotores de dynamixel AX-12+ los cuales manejan una comunicación serial asíncrona a 1Mbps preestablecida de fábrica, se manejó el microcontrolador PIC16F877 con el cual se construyó el sistema de control para las estructuras mecánicas anteriores. Sin embargo, la velocidad de comunicación asíncrona en este modelo de microcontrolador trabajándolo a 20MHz es de 57.6Kbps, por eso se recurrió a un microcontrolador de la misma familia pero de la gama alta. El más adecuado a los requerimientos del robot es el PIC18F452, el cual puede trabajar a una frecuencia de 40MHz entregando 10MIPS, en este microcontrolador la velocidad máxima de comunicación serial asíncrona trabajando a 40MHz es de 500kHz, la mitad de la velocidad que demanda el servomotor, aun así no se alcanza la velocidad de 1Mbps, no obstante el desarrollo de un programa en lenguaje ensamblador permite modificar la configuración de velocidad de comunicación.

Este tipo de servomotores maneja comunicación Full Duplex, se controla enviando y recibiendo paquetes de datos, hay dos tipos de paquetes, “paquete de instrucción” (que envía el controlador hacia el servomotor) y los “paquetes de estado” (que envía el servomotor hacia el controlador)[26].

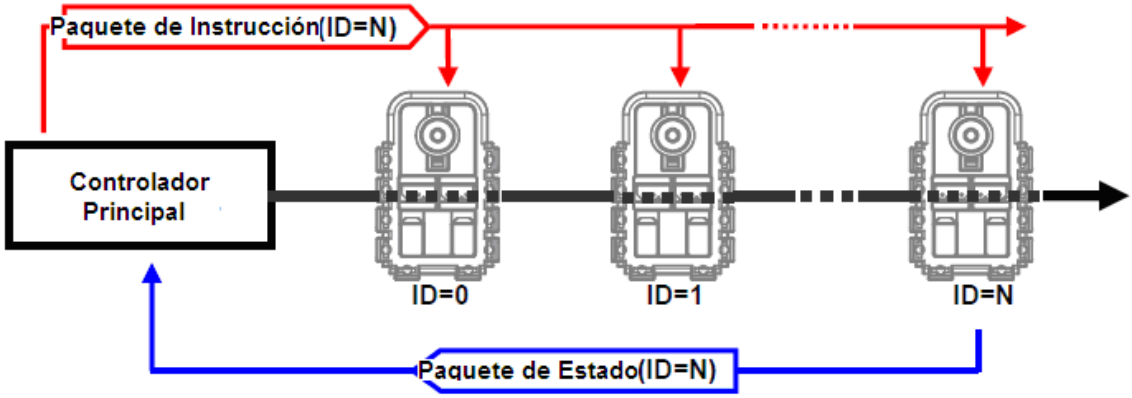

Figura 3.1.1 Envío y recepción de comandos por el mismo [26].

En la tabla 1 se da un resumen de los comandos que aceptan y transmiten estos motores, así como su dirección asignada:

Address	Item	Access	Initial Value
0(0X00)	Model Number(L)	RD	12(0x0C)
1(0X01)	Model Number(H)	RD	0(0x00)
2(0X02)	Version of Firmware	RD	?
3(0X03)	ID	RD,WR	1(0x01)
4(0X04)	Baud Rate	RD,WR	1(0x01)
5(0X05)	Return Delay Time	RD,WR	250(0xFA)
6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
10(0x0A)	(Reserved)	-	0(0x00)
11(0X0B)	the Highest Limit Temperature	RD,WR	85(0x55)
12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
14(0X0E)	Max Torque(L)	RD,WR	255(0XFF)
15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
16(0X10)	Status Return Level	RD,WR	2(0x02)
17(0X11)	Alarm LED	RD,WR	4(0x04)
18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
19(0X13)	(Reserved)	RD,WR	0(0x00)
20(0X14)	Down Calibration(L)	RD	?
21(0X15)	Down Calibration(H)	RD	?
22(0X16)	Up Calibration(L)	RD	?
23(0X17)	Up Calibration(H)	RD	?
24(0X18)	Torque Enable	RD,WR	0(0x00)
25(0X19)	LED	RD,WR	0(0x00)
26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
32(0X20)	Moving Speed(L)	RD,WR	0
33(0X21)	Moving Speed(H)	RD,WR	0
34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
36(0X24)	Present Position(L)	RD	?
37(0X25)	Present Position(H)	RD	?
38(0X26)	Present Speed(L)	RD	?
39(0X27)	Present Speed(H)	RD	?
40(0X28)	Present Load(L)	RD	?
41(0X29)	Present Load(H)	RD	?
42(0X2A)	Present Voltage	RD	?
43(0X2B)	Present Temperature	RD	?
44(0X2C)	Registered Instruction	RD,WR	0(0x00)
45(0X2D)	(Reserved)	-	0(0x00)
46[0x2E)	Moving	RD	0(0x00)
47[0x2F)	Lock	RD,WR	0(0x00)
48[0x30)	Punch(L)	RD,WR	32(0x20)
49[0x31)	Punch(H)	RD,WR	0(0x00)

Tabla 1. Distribución de las localidades de memoria [26].

La comunicación es a 8 bits con un bit de parada y sin paridad. Veamos ahora la estructura de un paquete de instrucción.

PAQUETE DE INSTRUCCIÓN	***FF***	***FF***	***ID***	***LONGITUD***	***INSTRUCCIÓN***	***PARÁMETRO 1***	***…***	***PARÁMETRO N***	***CHECKSUM***

A continuación explicaremos cada parámetro de un paquete de instrucción:

- FF-FF

Los dos primeros parámetros que enviamos son FF – FF, estos dos parámetros son obligatorios siempre que se envía algo a cualquiera de los motores o al módulo, sirve para indicar al motor que tienen que empezar a leer desde ese punto.

- ID

Es el identificador del motor que debe recibir la instrucción, que puede variar desde 0 a 253, cuando se manda el 254 todos los motores conectados reciben la instrucción.

- LONGITUD

Esta parte del paquete de información es importante, es el número, es la cantidad de parámetros que deseamos enviar+2, por tanto si se envía un 7, en realidad lo que se manda son 5 parámetros.

- INSTRUCCIÓN

Es la acción a ejecutar por el servomotor, hay básicamente 7 tipos de instrucciones, como se muestran en la tabla 2.

Instruction	Function	Value	Number of Parameter
PING	No action. Used for obtaining a Status Packet	0x01	0
READ DATA	Reading values in the Control Table	0x02	2
WRITE DATA	Writing values to the Control Table	0x03	2 ~
REG WRITE	Similar to WRITE_DATA, but stays in standby mode until the ACION instruction is given	0x04	2 ~
ACTION	Triggers the action registered by the REG_WRITE instruction	0x05	0
RESET	Changes the control table values of the Dynamixel actuator to the Factory Default Value settings	0x06	0
SYNC WRITE	Used for controlling many Dynamixel actuators at the same time	0x83	4~

Tabla 2 Instrucciones que soporta el servomotor AX-12+ de Dinamixel [26].

Según la tabla 2 vemos que hay 7 instrucciones, por ejemplo si enviamos un número 3 estamos usando la instrucción WRITE DATA, con esta lo que indicamos es que queremos escribir en la memoria del servo.

- PARÁMETROS

En la tabla 2 las instrucciones tienen determinado número de parámetros, la instrucción 3 WRITE DATA tiene por lo menos 2 parámetros por ejemplo, si queremos mover el motor a determinada posición tendríamos que mandar FF-FF-5-7-3-1E-00-02-00-02-D3, en la cual de los parámetros 1E-00-02-00-02, el primer parámetro 1E es la dirección donde va a empezar a escribir, en la tabla 3 esta dirección es la 30 en decimal Goal Position(L) y los otros siguientes parámetros 00-02-00-02 son los valores que serán escritos en las direcciones Goal Position(L),Goal Position(H), Moving Speed(L) y Moving Speed(H) respectivamente.

30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
32(0X20)	Moving Speed(L)	RD,WR	0
33(0X21)	Moving Speed(H)	RD,WR	0

Tabla 3 Posición de memoria de Goal position y Moving Speed [26].

- CHECKSUM

El número del checksum sirve para hacer la comprobación de que lo que enviamos es correcto, para esto se hace un cálculo matemático muy sencillo.

Checksum = ~ (ID + Lenght + Parámetros + Instrucción + parámetro1 + … + Parámetro N)

SI el valor calculado de Checksum es mayor a 255 el valor del byte más bajo es el que se considera como Checksum.

El programa para configurar la velocidad de transmisión se explicará posteriormente. Una vez que se configuró la velocidad de transmisión a 500kHz, a cada motor se le asignó una dirección, en la Figura 3.1.2, se muestra la asignación que corresponde a cada motor. Nótese que no existe el identificador 10, esto es para facilitar el manejo de los motores, dado que el 1 y el 11 son la misma extremidad pero del lado opuesto.

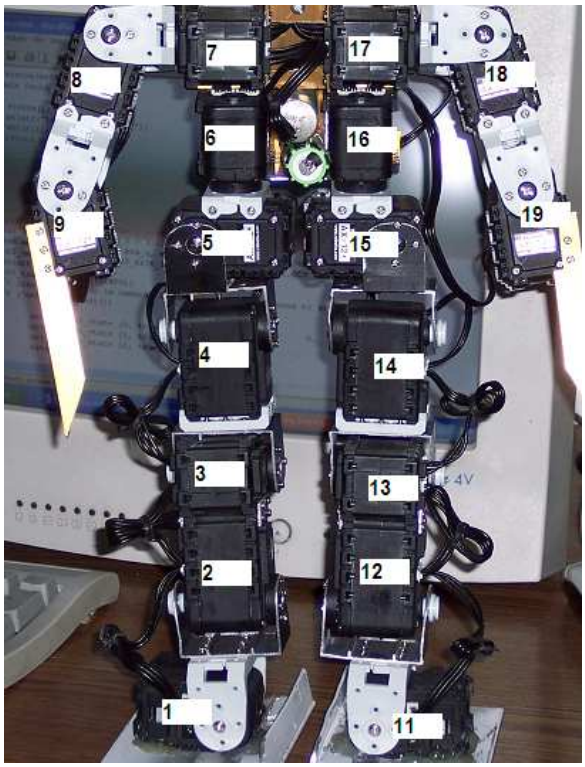

Figura 3.1.2 Identificador de cada motor

Dado que los motores trabajan con comunicación serial asíncrona Half Duplex, se procedió a construir un circuito que permitiera controlar los motores, el fabricante propone un circuito como el que se muestra en la Figura 3.1.3 [26].

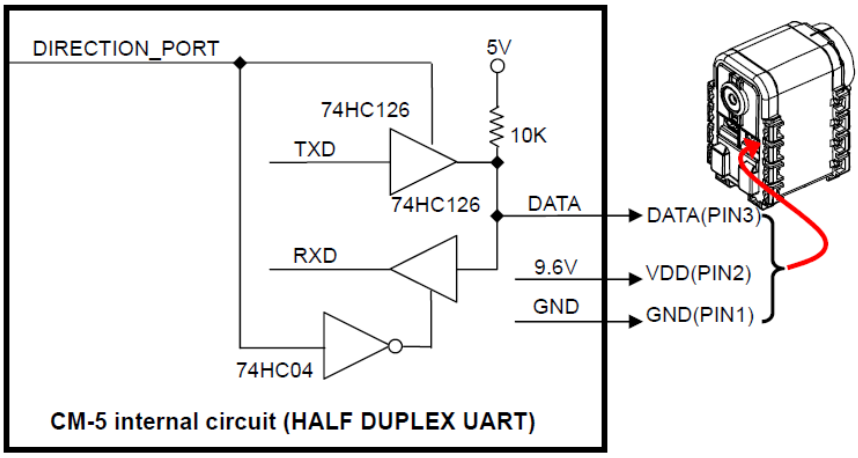

Figura 3.1.3 Circuito para controlar los motores AX-12+ de Dynamixel

En este caso se utiliza un circuito muy simple para controlar los motores, y se muestra en la Figura 3.1.4

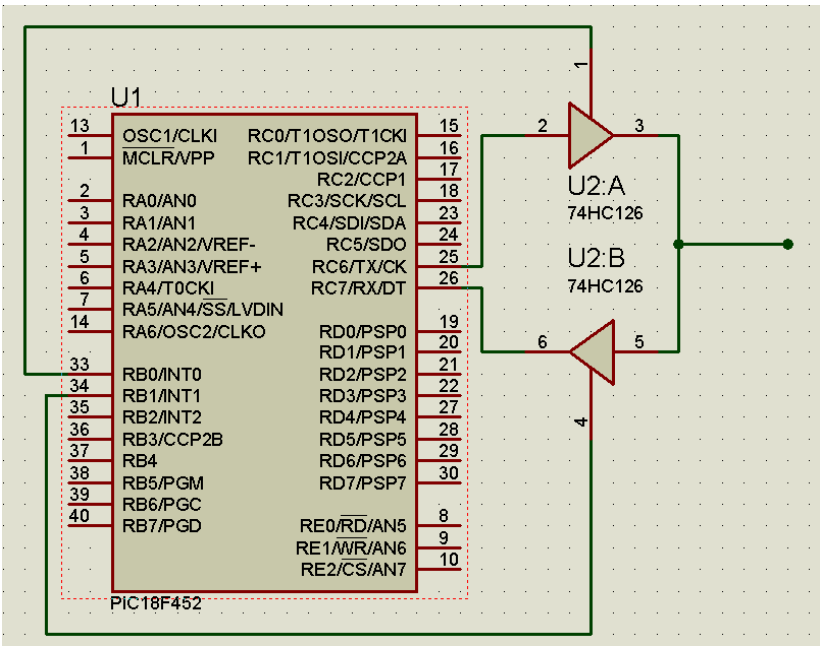

Figura 3.1.4 Circuito para controlar los motores del robot

En el circuito se ahorra el integrado 74HC04 ya que el flujo se controla por dos puertos del PIC que son RB0 y RB1. El microcontrolador es básicamente la parte que controla los movimientos del robot, el planeador de acciones, por lo tanto lo tenemos que conectar con los sensores que en este caso son el acelerómetro, el giroscopio y la CMUcam3, para el caso de el acelerómetro la conexión es mediante comunicación serial sincronía SPI, para el cual necesitamos 3 hilos de conexión hacia el microcontrolador. El microcontrolador fue conectado como se muestra en la Figura 3.1.5.

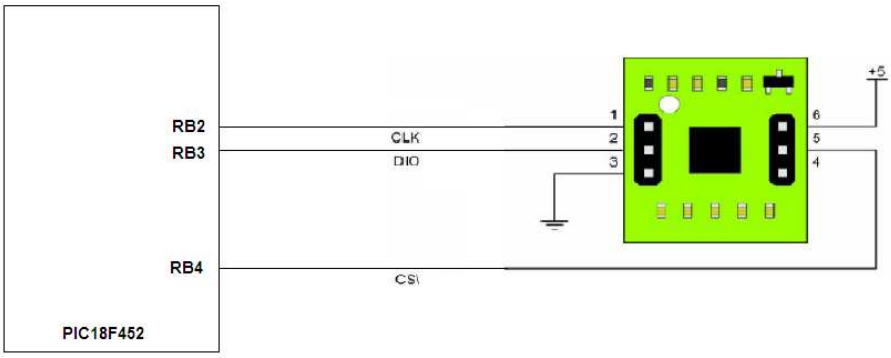

Figura 3.1.5 Conexión del acelerómetro H48C con el PIC18F452

La conexión es bastante sencilla, en el pin 1 del acelerómetro llega la señal de reloj mientras que en el pin 2 entra la petición de dato y por el mismo pin sale el dato hacia el PIC, el pin 4 del acelerómetro es el que habilita el módulo (chip select), el cual se activa con un nivel bajo.

Este acelerómetro contiene en el circuito un convertidor analógico-digital MCP3204 el cual toma simultáneamente las lecturas en los 3 ejes y en el voltaje de referencia, para calcular la aceleración se ocupa la siguiente fórmula: G((lectura en el eje n – voltaje de referencia)/4095)x(3.3/0.3663), sin embargo se puede reducir la fórmula haciendo las operaciones y quedar como: G=(lectura en el eje n – voltaje de referencia)x0.0022.

Cuando el voltaje de referencia es mayor a la lectura en el eje, entonces la aceleración es negativa. La forma en la que se leen los datos del sensor usando el leguaje PicBasic se muestra en el listado 1.

```
Get_H48C:
LOW CS
SHIFTOUT Dio, Clk, MSBFIRST, [%11\2, VRef\3] ' selecciona voltaje de referencia.
SHIFTIN Dio, Clk, MSBPOST, [rvCount\13] 'lee voltage de referencia.
HIGH CS
PAUSE 1
LOW CS
SHIFTOUT Dio, Clk, MSBFIRST, [%11\2, axis\3] ' selecciona el eje a leer.
SHIFTIN Dio, Clk, MSBPOST, [axCount\13] ' lee el voltaje del eje seleccionado.
HIGH CS
RETURN
```

Listado 1 Código para leer el acelerómetro.

En el listado 1 primero se tiene que habilitar el módulo con un nivel bajo en chip select, después se envía el canal el cual se desea leer, en este caso el canal 0 es el eje x, canal 1 eje y, canal 3 eje z y por ultimo el voltaje de referencia en el canal 4. Inmediatamente después de enviar el canal a leer en un dato de 5 bits, se lee el valor por el mismo pin, el cual siempre será de 12 bits por tanto lo guardamos en una variable Word de 16 bits.

El giroscopio ya se describió físicamente en el capítulo anterior, ahora veamos como funciona. Dado que el uso de este giroscopio es para radiocontrol la hoja de especificaciones del fabricante lo único que indica son las dimensiones, la forma de conectarse y la forma de colocarse, para poder usarlo primero se debe entender como funcionan las señales de entrada y salida, en la Figura 2.2.2 tiene un arnés con 3 cables, el café es tierra el rojo 5v y el naranja es la señal de entrada, y en la parte de salida es indicada con una “S”.

Ahora no debemos olvidar que el arnés de entrada va conectado directo a un receptor de radiocontrol, por tanto la señal que sale de este modulo es para controlar un servomotor estándar futaba, y la señal para controlar este tipo de motores se muestra en la Figura 3.1.6 a 3.1.8. [19]

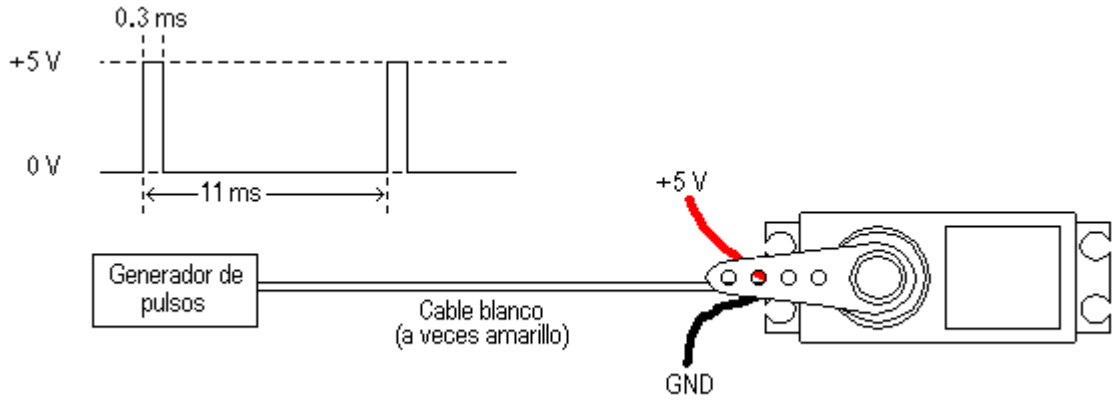

Figura 3.1.6 Pulso para posicionar el eje a la izquierda

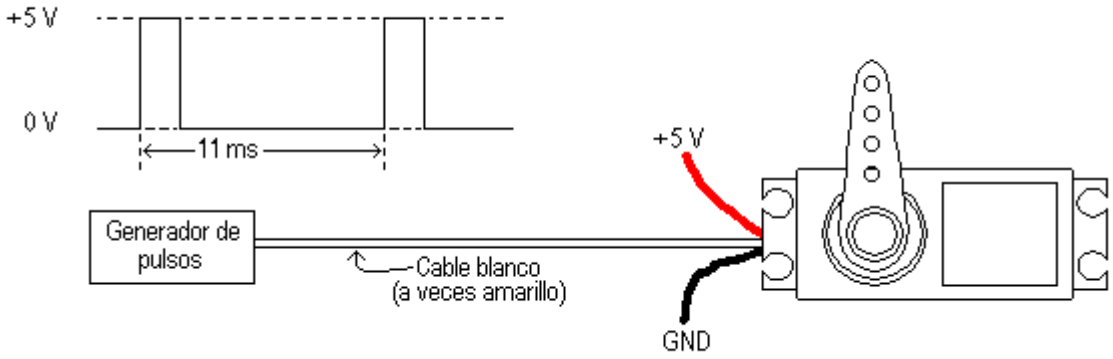

Figura 3.1.7 Pulso para centrar el eje del motor

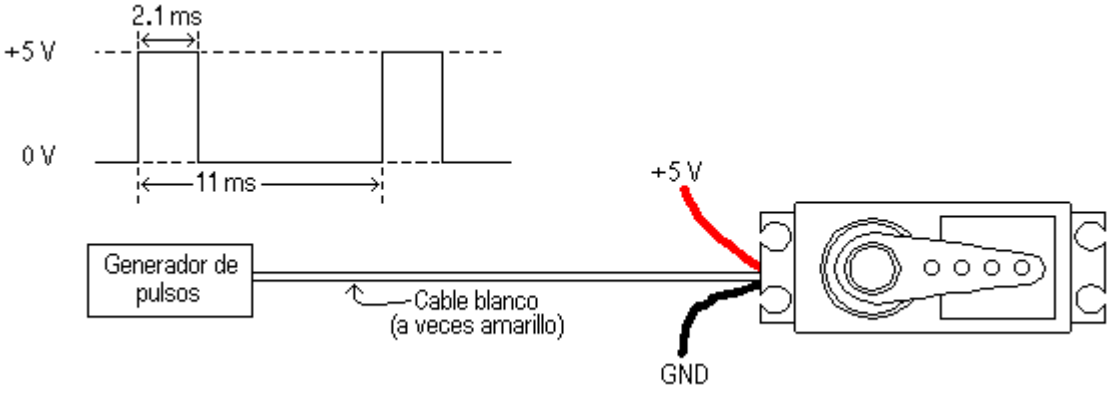

Figura 3.1.8 Pulso para posicionar el eje del motor a la derecha

Es importante recalcar que en la señal de entrada el período puede variar desde 10ms hasta 20ms dependiendo del receptor de radiocontrol que se esté utilizando. El giroscopio aumenta o disminuye el ancho de pulso tan pronto se produce un movimiento angular. En la Figura 3.1.9 se ilustra la señal de entrada contra la señal

de salida sin que se sufra movimiento angular. En la parte de arriba se muestra la señal de entrada a el giroscopio y en la parte de abajo la señal de salida, se observa que cuando no hay movimiento angular la señal de entrada solo se retraza aproximadamente 0.5ms.

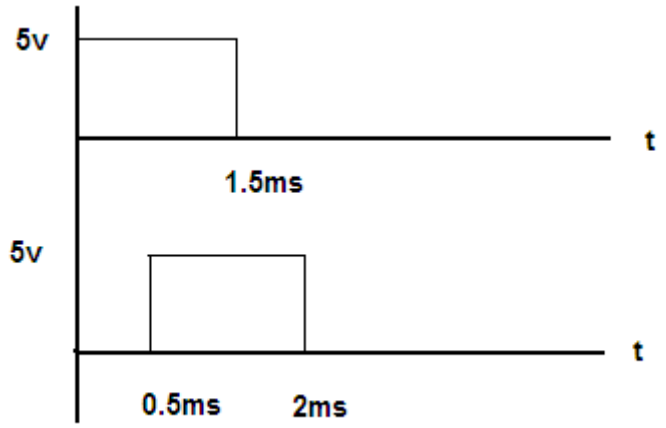

Figura 3.1.9 Señal de entrada y respuesta sin movimiento angular

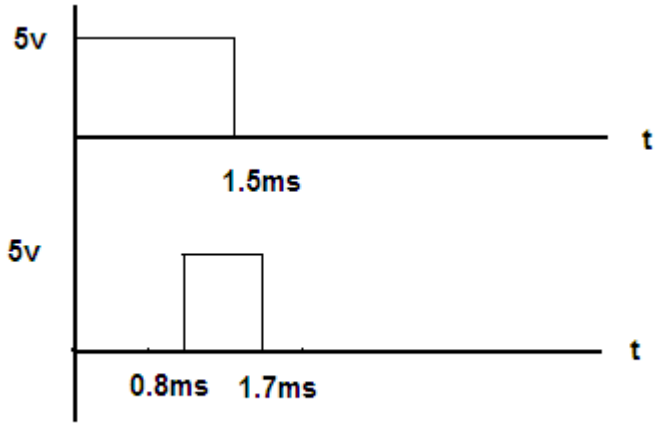

Figura 3.1.10 Señal de entrada y respuesta con movimiento angular

En la Figura 3.1.10 Arriba se muestra la señal de entrada y en la parte de abajo la señal que sale del giroscopio, esta se encuentra retrasada 0.5ms y recortada en 0.3ms de cada lado.

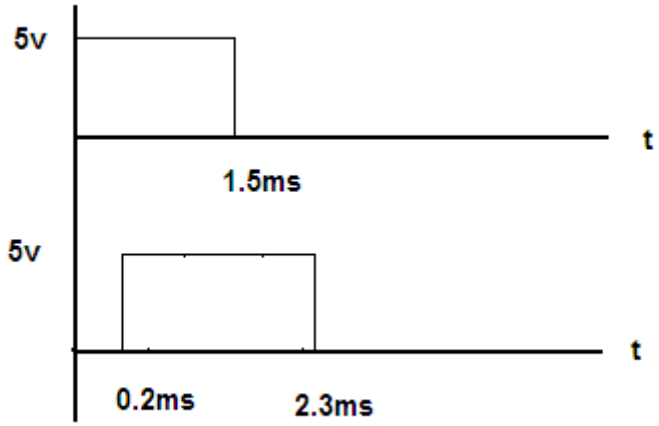

Figura 3.1.11 Señal de entrada y respuesta con movimiento angular inverso

En la Figura 3.1.11 se muestra que la señal de salida esta desfasada 0.5ms pero como el giro fue en sentido contrario el pulso aumento 0.3ms de cada lado.

Por lo tanto como se ve en las Figuras 3.1.9 a 3.1.11 el ancho de pulso de la salida varía dependiendo de hacia dónde es el giro, estos valores son los máximos que se observaron en el osciloscopio.

Por consiguiente para medir la velocidad de giro lo podemos hacer únicamente con la primera parte del pulso, dado que es simétrico, lo que se hace es alimentar con un pulso de 1.5ms, en el momento en que le viene el flanco de subida lo que tenemos que hacer es medir el tiempo de respuesta, si es aproximadamente 0.5ms no hay giro, si es menor a 0.5ms existe un giro y si es mayor a 0.5ms existe un giro en sentido contrario. La ganancia puede ser ajustada con el otro potenciómetro del módulo PK3.

En el lenguaje Pic BasicPro existe una instrucción muy sencilla para medir el tiempo de respuesta, por ejemplo RCTIME PORTB.3,0,TIEMPO, en esta instrucción lo que hace es medir el tiempo que el pin mantiene el estado en bajo y lo guarda en la variable tiempo, TIEMPO está dado en microsegundos y el valor máximo de esta puede ser de 65535, que por lo que si nuestro mínimo es de 0.2ms son 200 microsegundos y el máximo es 0.8ms son 800 microsegundos se puede truncar la espera.

Otro hardware importante en el robot es la CMUcam3, las principales características son:

· Sensor RGB en color con una resolución de 352 x 288, y baja resolución de 176 x 144.

· Ambientes de desarrollo para Windows y Linux.

· Conector para tarjeta SD o MMS con soporte FAT 16.

· Cuatro Puertos para controlar servos.

· Carga imágenes en memoria a 26 tramas por segundo.

· Compresión por software en modo JPEG.

· Emulación de la CMUCAM2.

· Puerto de comunicación serial.

El hardware de la CMUcam3 está basado en una arquitectura ARM7TFMI con un procesador NXP LPC2106 y puertos de propósito general. La Figura 3.1.12 [33] muestra la interfaz de hardware que proporciona la CMUcam3 [33].

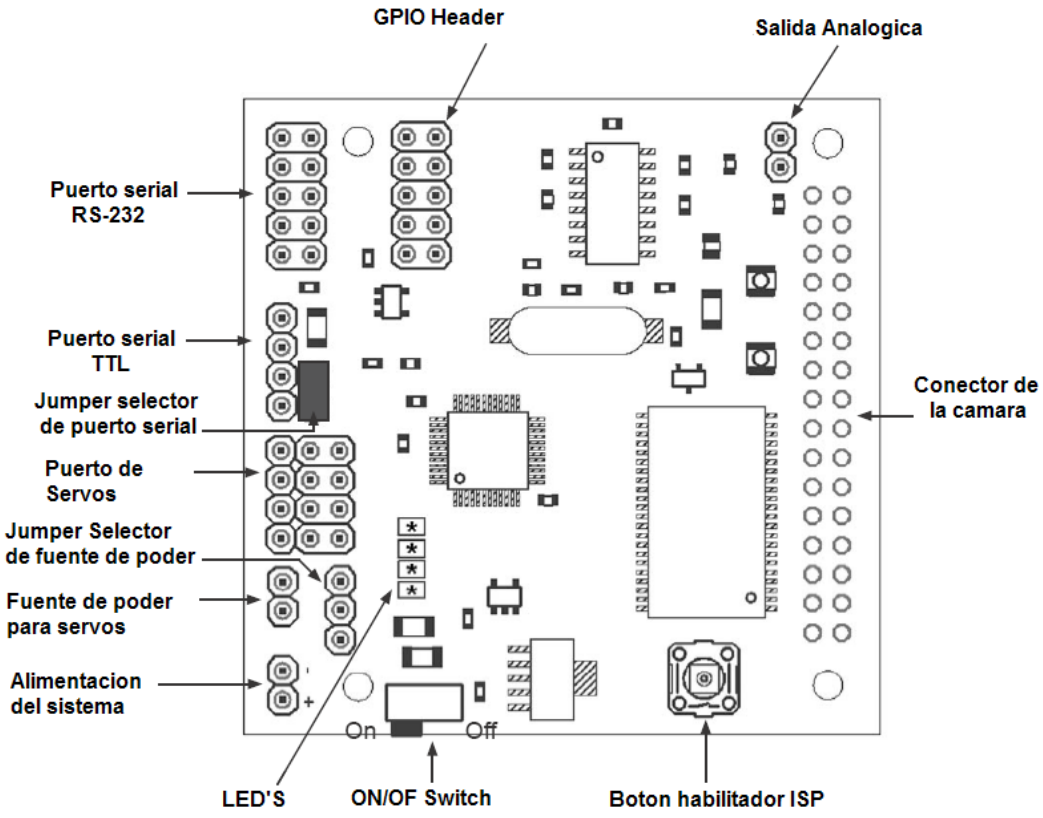

Figura 3.1.12 Puertos interfaces y alimentación de la CMUcam3

La tarjeta puede ser alimentada entre 6 y l5 VDC a una corriente de 150mA, para alimentar los servomotores trae un conector, sin embargo también se pueden alimentar con la alimentación interna únicamente poniendo un jumper. En la Figura 3.1.13 [33] se muestra la posición de los conectores con su respectiva polaridad.

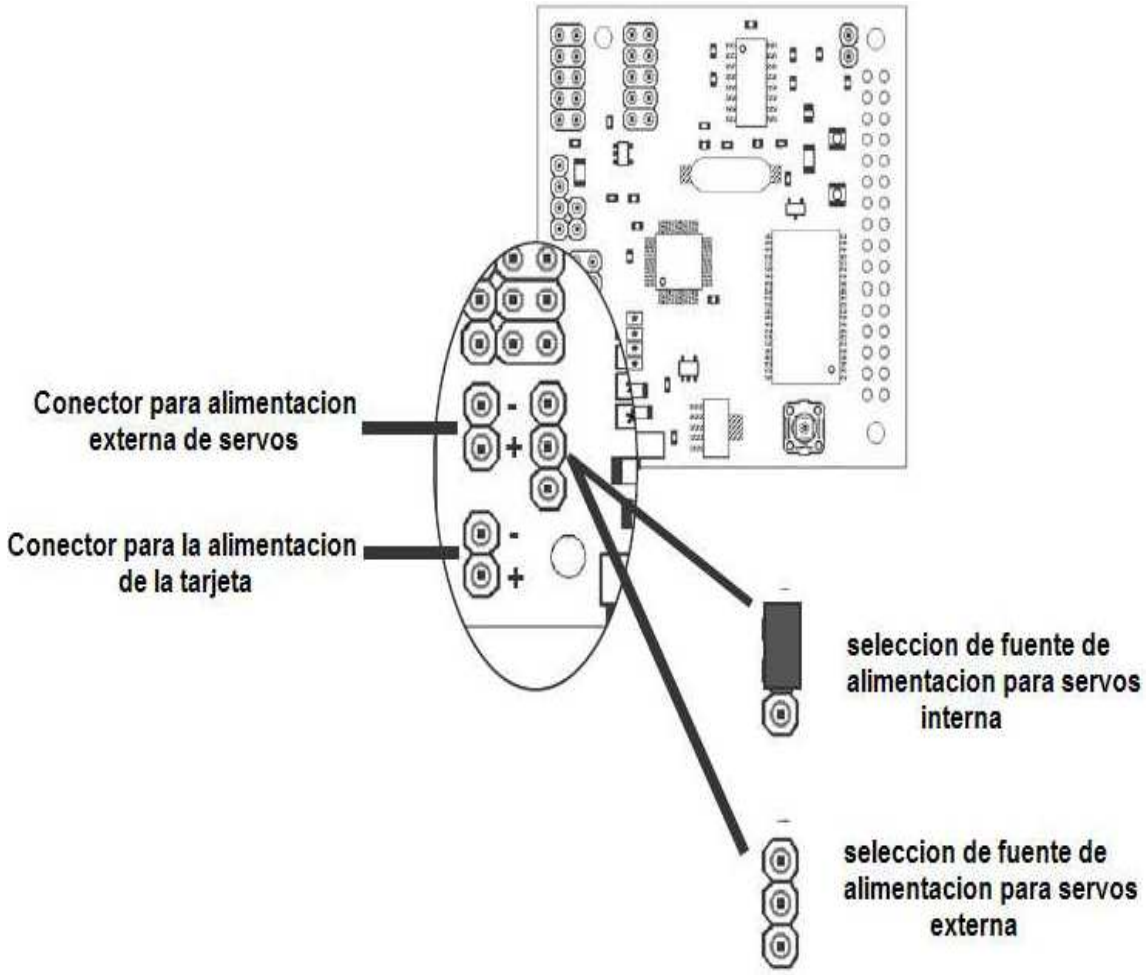

Figura 3.1.13 Conectores para alimentar la CMUcam3

Para comunicarse con nuestra PC la CMUcam3 cuenta con puerto de comunicación serial, por este puerto se comunica tanto para mandar y recibir comandos imágenes así como para ser programado el chip LPC2106, es decir que sirve como puerto para cargar nuevos Firmware. Además cuenta con un jumper para poder comunicarse en forma serial con dispositivos que manejan niveles TTL, en la Figura 3.1.14 se muestra la descripción de los puertos.

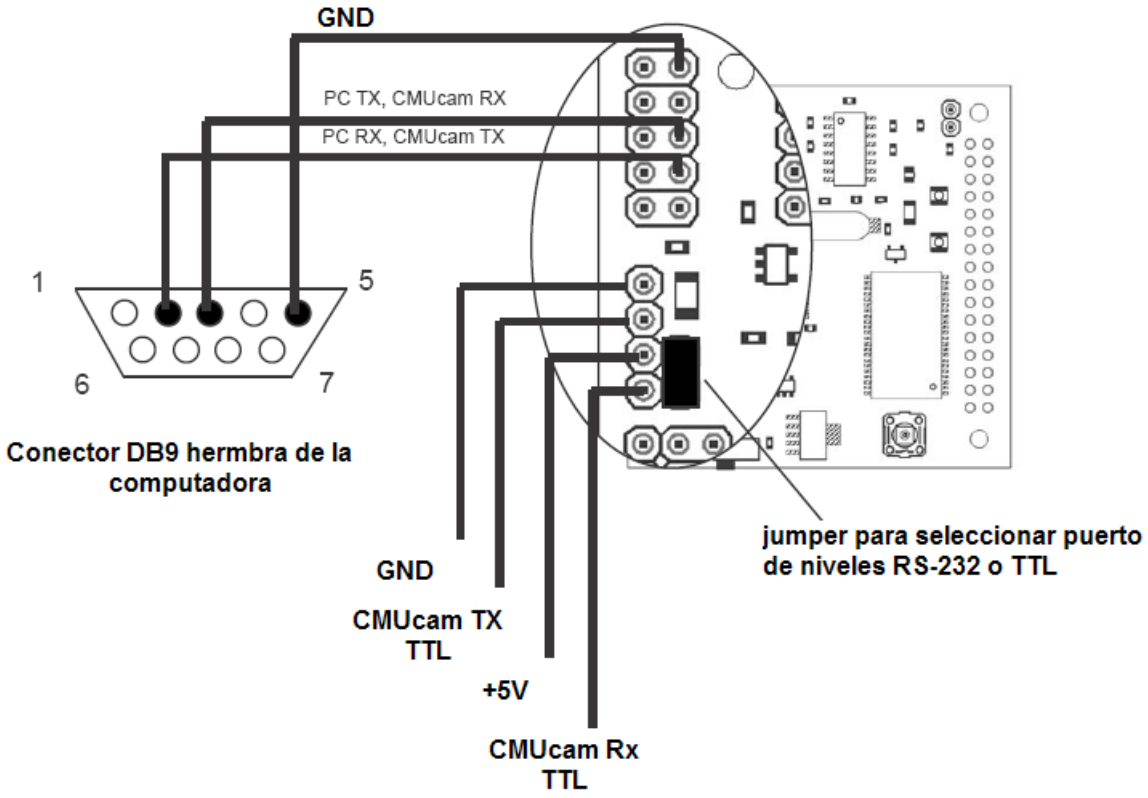

Figura 3.1.14 Conectores para la comunicación serie

El botón ISP nos permite elegir el modo en el que la tarjeta debe iniciarse. Cuando alimentamos a la tarjeta, y el botón ISP no es presionado, inicia automáticamente en el modo de ejecución, donde ejecutará el código previamente instalado. Por lo contrario, cuando se alimenta la tarjeta y a su vez se mantiene presionado el botón ISP, se entrará en el modo boot loader, esto quiere decir que la tarjeta esta lista para ser programada con un nuevo firmware.

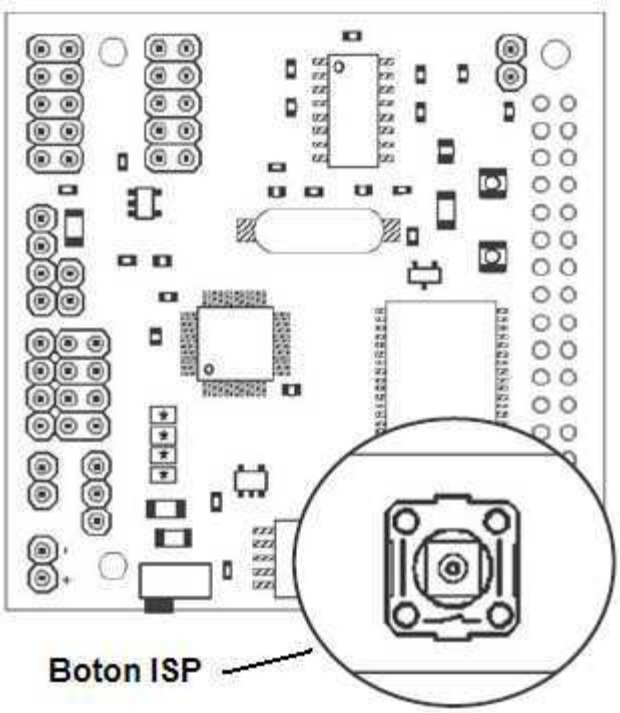

Figura 3.1.15 Ubicación del botón ISP.

En el robot la comunicación con la CMUcam3 es por el puerto de comunicación serial con niveles TTL [33].

3.2 Algoritmo Para El Sistema De Visión

El algoritmo implementado es bastante sencillo y consiste en la búsqueda de cúmulos de color, al tener la posibilidad de configurar la cámara en RGB, ponderamos los umbrales de los colores a los cuales se desea hacer reconocimiento.

En el listado 3.2.1 observamos el código en C que realiza la delimitación de los umbrales que cada color debe tener para la obtención del reconocimiento del color naranja. Lamentablemente estos valores deberían ser auto ajustables a los cambios de intensidad lumínica. Por lo que en condiciones de alta variación el sistema de visión presentará irregularidades.

La facilidad de cambiar los umbrales, abre la posibilidad de poder manejar cualquier tipo de color.

```
// parámetros a ser buscados por la función  simple_track_color

  t_pkt.lower_bound.channel[CC3_CHANNEL_RED] = 150;
  t_pkt.upper_bound.channel[CC3_CHANNEL_RED] = 255;
  t_pkt.lower_bound.channel[CC3_CHANNEL_GREEN] = 0;
  t_pkt.upper_bound.channel[CC3_CHANNEL_GREEN] = 50;
  t_pkt.lower_bound.channel[CC3_CHANNEL_BLUE] = 0;
  t_pkt.upper_bound.channel[CC3_CHANNEL_BLUE] = 50;
```

Listado 3.2.1 Delimitación de umbrales para reconocer color naranja

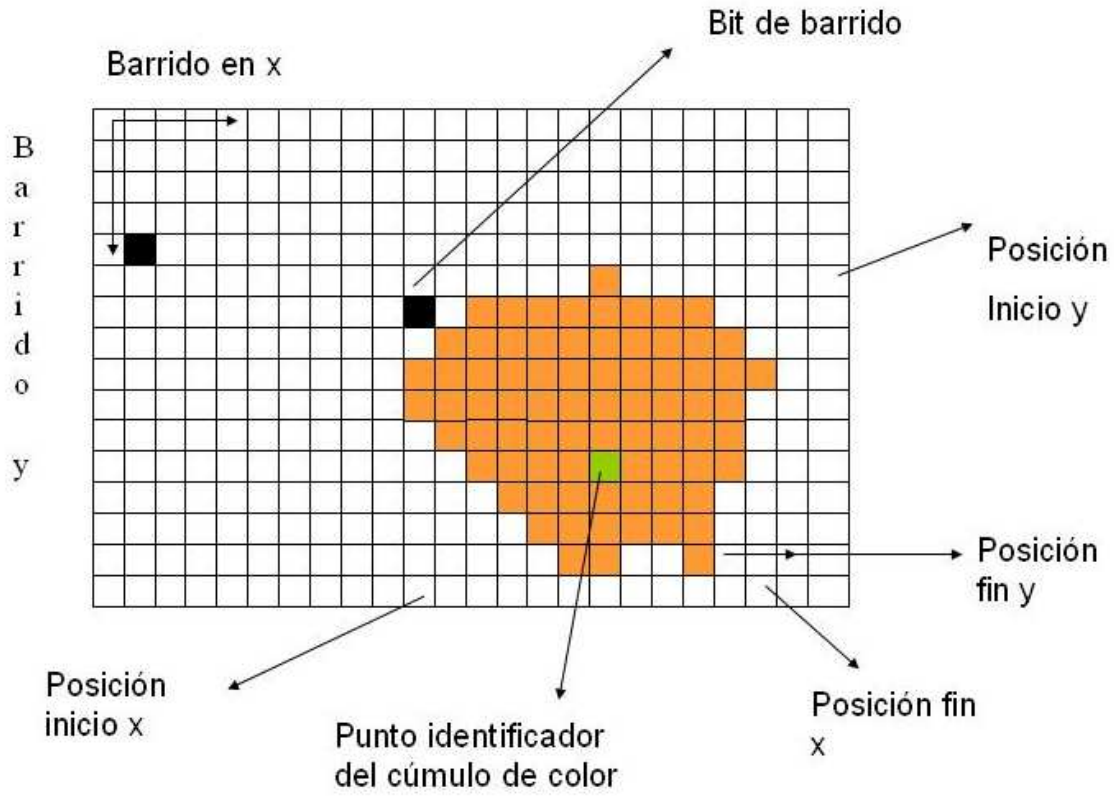

Figura 3.2.1 Centrado del cúmulo de color naranja

Como se muestra en la Figura 3.2.1 se hace un barrido en el eje x y se cuentan los puntos que cumplen con la condición antes dicha, se almacena la posición en la que se encuentren más puntos. En el eje y se hace el mismo procedimiento, con esto obtenemos las coordenadas del punto donde se encuentra la mayor cantidad de puntos que cumplen con la condición. Una vez identificado el cúmulo, enviamos las coordenadas de este por el puerto serie hacia nuestro sistema de control para que tome las decisiones correspondientes.

3.3 Programación Del Algoritmo Para Caminar

Para programar el algoritmo de caminar nos basaremos en lo que llamamos locomoción humana normal, en base a esta, iremos programando las fases de un paso lo más aproximado al humano.

Ahora para programar los movimientos en nuestro robot, primero hay que inicializar los motores, en seguida activar el torque, y después ponerlos en la posición de firmes, para poder empezar.

```
FIRMES:
        P[1]=530: P[2]=570: P[3]=760: P[4]=535: P[5]=505: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=501:P[12]=420:P[13]=275:P[14]=490:P[15]=511:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
RETURN
```

Listado 3.3.1 Posiciones de cada motor

Como se observa en el listado 3.3.1 en la etiqueta firmes se encuentran las posiciones de cada motor en el arreglo P para que el robot permanezca en la posición de firmes, también le indicamos a los motores con que velocidad tenemos que mover los motores.

Por lo tanto para ir cambiando de posición al robot tendremos que mandar prácticamente todo el arreglo completo de posiciones y velocidad hacia los motores. Para la caminata normal iremos enviando las posiciones de los motores, es importante recalcar que nuestro equilibrio únicamente lo van a compensar los motores del tobillo, basados en la lectura del giroscopio.

Figura 3.3.1 Fases para dar un paso

En la Figura 3.3.1 se muestran las 7 fases en las que se dividió un paso, para cada una de las fases en el programa se indicó a cada motor la posición que debe tomar, así como la velocidad a la que se deben mover los motores y el tiempo de respuesta, es importante observar que para cada fase la velocidad a la que se mueven los motores debe cambiar.

En el listado 3.2.2. Se muestra la secuencia que se le mando a cada motor, la llamada a subrutina para leer el giroscopio, la compensación de los tobillos y el tiempo de respuesta para cada fase de un paso.

```
FASE1:
        'Secuencia a los motores
        P[1]=570: P[2]=570: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=551:P[12]=422:P[13]=275:P[14]=490:P[15]=550:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$15
        GOSUB MANDA1
        GOSUB GIRO                  'Verifica giroscopio
        IF CUENTA >600 THEN
        P[2]= 550:P[12]=442         'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE1
        ENDIF
        IF CUENTA <400 THEN
        P[2]= 590:P[12]=402         'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE1
        ENDIF
        PAUSE 200                   'Tiempo de respuesta
FASE2:
        'Secuencia a los motores
        P[1]=570: P[2]=570: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=551:P[12]=380:P[13]=385:P[14]=450:P[15]=550:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$20
```

```
GOSUB MANDA1
GOSUB GIRO                  'Verifica giroscopio
IF CUENTA >600 THEN
P[2]= 550:P[12]=400         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE2
ENDIF
IF CUENTA <400 THEN
P[2]= 590:P[12]=360         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE2
ENDIF
PAUSE 400                   'Tiempo de respuesta

FASE3:
'Secuencia a los motores
P[1]=570: P[2]=620: P[3]=760: P[4]=465: P[5]=530: P[6]=511: P[7]=200: P[8]=200: P[9]=511
P[11]=551:P[12]=492:P[13]=275:P[14]=400:P[15]=550:P[16]=511:P[17]=820:P[18]=800:P[19]=511
VELOCIDAD=$30
GOSUB MANDA1
GOSUB GIRO                  'Verifica giroscopio
IF CUENTA >600 THEN
P[2]= 600:P[12]=512         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE3
ENDIF
IF CUENTA <400 THEN
P[2]= 640: P[12]=472        'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE3
ENDIF
PAUSE 500                   'Tiempo de respuesta

FASE4:
'Secuencia a los motores
P[1]=470: P[2]=625: P[3]=760: P[4]=465: P[5]=465: P[6]=511: P[7]=200: P[8]=200: P[9]=511
P[11]=455:P[12]=492:P[13]=275:P[14]=400:P[15]=467:P[16]=511:P[17]=820:P[18]=800:P[19]=511
VELOCIDAD=$30
GOSUB MANDA1
GOSUB GIRO                  'Verifica giroscopio
IF CUENTA >600 THEN
P[2]= 605:P[12]=512         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE4
ENDIF
IF CUENTA <400 THEN
P[2]= 645:P[12]=472         'Correccion tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE4
ENDIF
PAUSE 800                   'Tiempo de respuesta

FASE5:
'Secuencia a los motores
P[1]=470: P[2]=670: P[3]=640: P[4]=535: P[5]=435: P[6]=511: P[7]=200: P[8]=200: P[9]=511
P[11]=445:P[12]=422:P[13]=275:P[14]=490:P[15]=467:P[16]=511:P[17]=820:P[18]=800:P[19]=511
VELOCIDAD=$30
GOSUB MANDA1
```

```
        GOSUB GIRO              'Verifica giroscopio
        IF CUENTA >600 THEN
        P[2]= 650:P[12]=442     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE5
        ENDIF
        IF CUENTA <400 THEN
        P[2]= 690:P[12]=402     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE5
        ENDIF
        PAUSE 500               'Tiempo de respuesta
FASE6:
        'Secuencia a los motores
        P[1]=470: P[2]=620: P[3]=640: P[4]=585: P[5]=435: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=445:P[12]=422:P[13]=275:P[14]=490:P[15]=467:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$20
        GOSUB MANDA1
        GOSUB GIRO              'Verifica giroscopio
        IF CUENTA >600 THEN
        P[2]= 600:P[12]=442     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE6
        ENDIF
        IF CUENTA <400 THEN
        P[2]= 640:P[12]=402     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE6
        ENDIF
        PAUSE 500               'Tiempo de respuesta

FASE7:
        'Secuencia a los motores
        P[1]=470: P[2]=500: P[3]=760: P[4]=585: P[5]=435: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=445:P[12]=422:P[13]=275:P[14]=490:P[15]=467:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$10
        GOSUB MANDA1
        GOSUB GIRO              'Verifica giroscopio
        IF CUENTA >600 THEN
        P[2]= 480:P[12]=442     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE7
        ENDIF
        IF CUENTA <400 THEN
        P[2]= 520:P[12]=402     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE7
        ENDIF
        PAUSE 500               'Tiempo de respuesta
```

Listado 3.2.2. Código para dar un paso

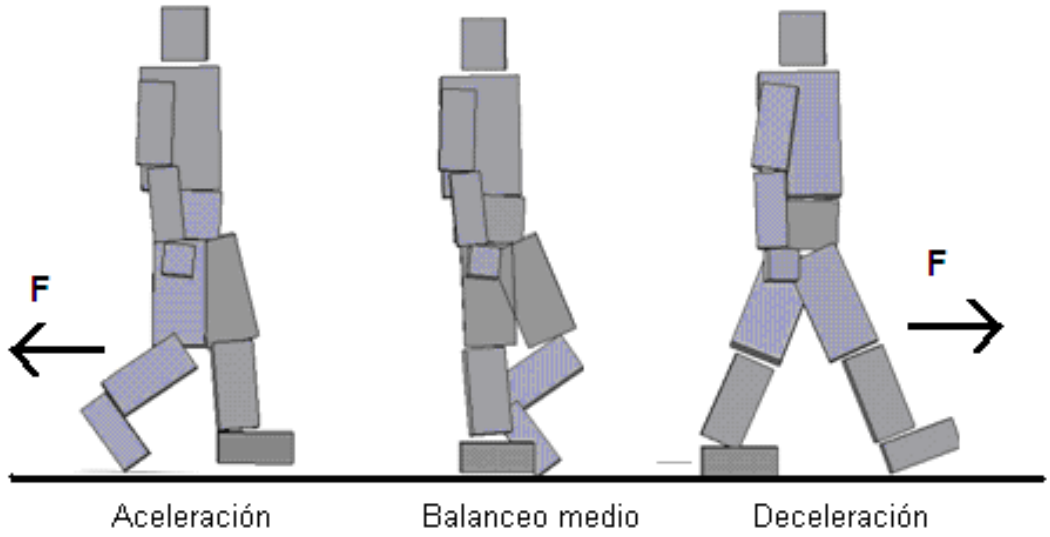

Figura 3.3.2 Fases de la etapa de balanceo

En la Figura 3.3.2 tenemos las 3 fases de la etapa de balanceo. En la primera fase el pie que está en balanceo comienza a acelerarse, esto nos genera una fuerza hacia atrás, en el balanceo medio es cuando se alcanza la máxima velocidad, y por último cuando termina la fase de balanceo, se genera una fuerza hacia adelante. Estas fuerzas deben ser compensadas en el tobillo que se encuentra en apoyo para evitar un desbalance.

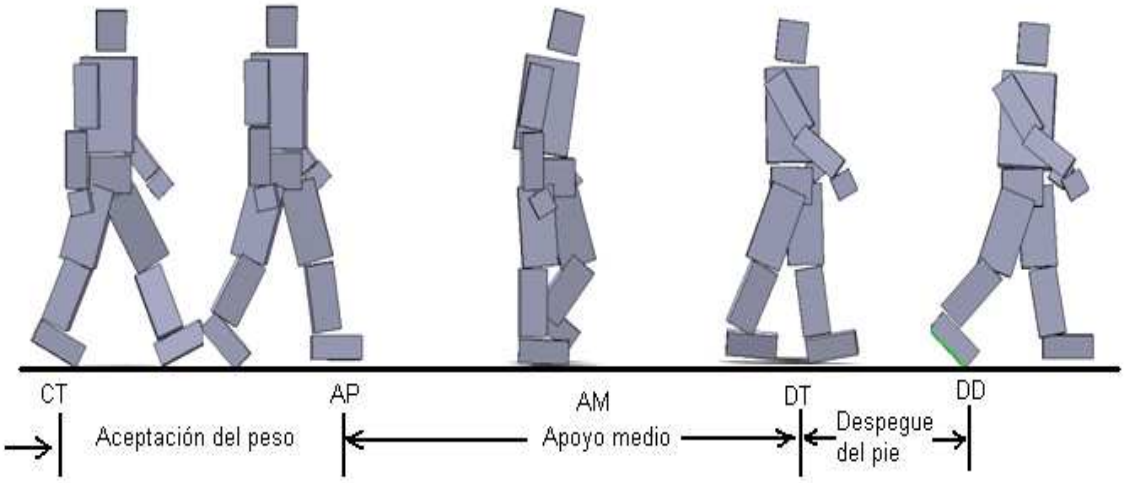

Figura 3.3.3 Fases de apoyo

En la Figura 3.3.3 tenemos primeramente la aceptación del peso, y por último el despegue del pie, estas fases son de doble apoyo, es decir, ambos pies se encuentran apoyados en el piso, la ausencia de esta etapa diferencia el correr del caminar. Cada que termine la rutina de un paso ya sea hacia adelante, giro, o tiro, usaremos el acelerómetro para verificar si el robot esta tirado, en tal caso llamaremos a la rutina para que se levante dependiendo si esta tirado de espalda o de frente.

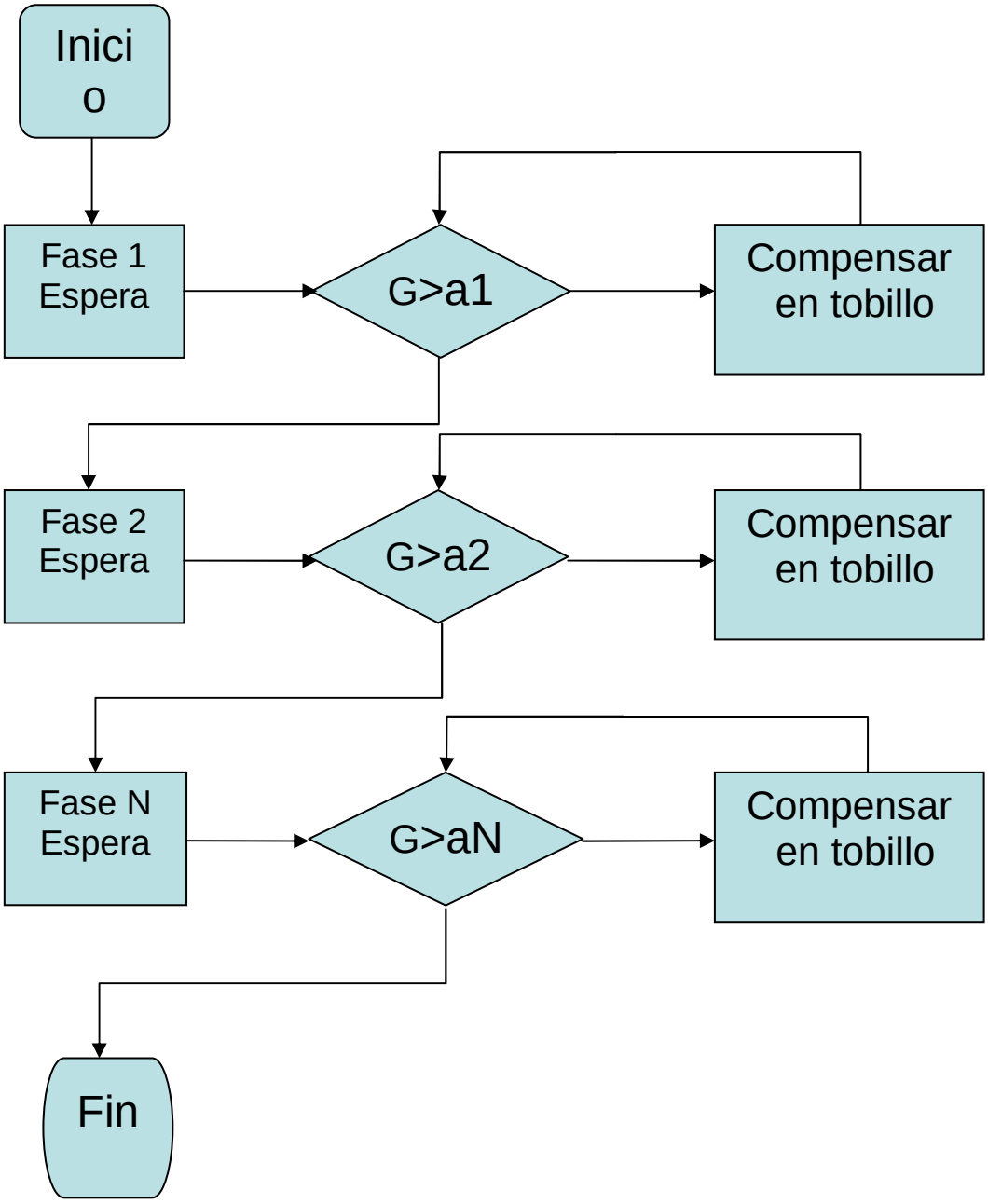

Figura 3.3.4 Diagrama de bloques de un movimiento

En la Figura 3.3.4 se observa el diagrama de bloque para un movimiento, primeramente en el bloque fase se le indica a cada motor la posición que debe tomar y el tiempo estimado para que lo haga, después se consulta el giroscopio, y si el valor de G es mayor al que establecimos, manda una compensación al tobillo, si con la compensación ya se encuentra en el intervalo correcto, pasa a la siguiente fase.

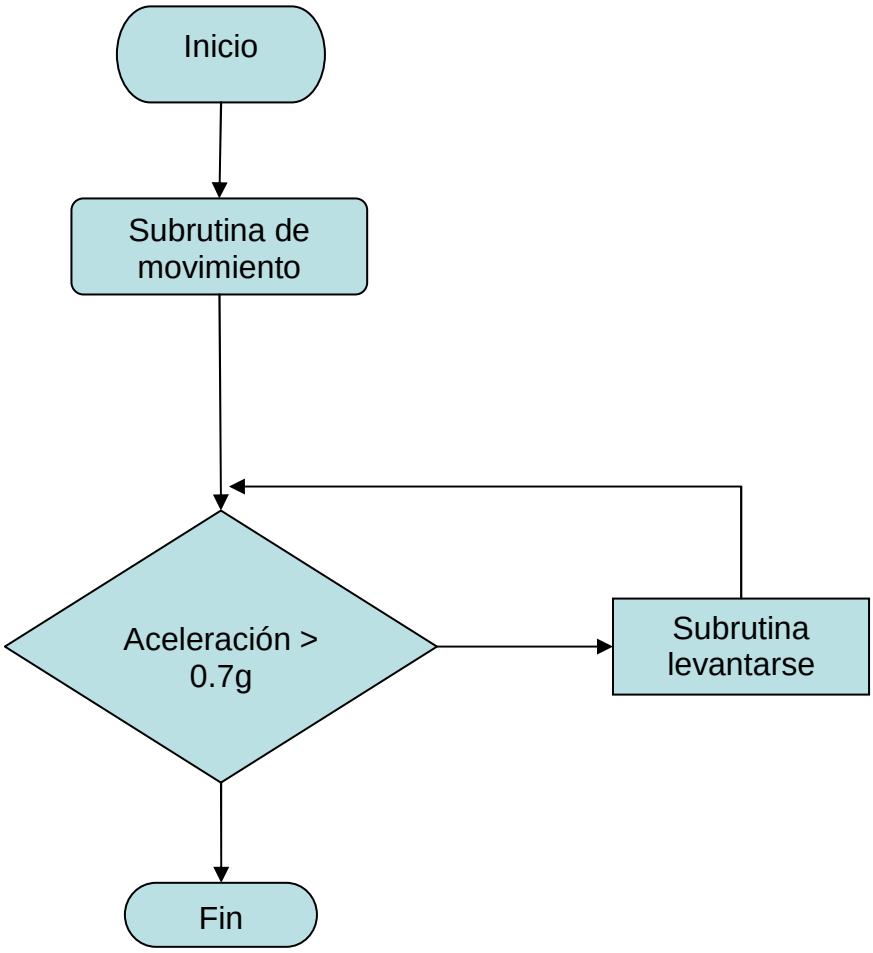

Figura 3.3.5 Llamada a subrutina de movimiento

En la Figura 3.3.5 se muestra la llamada a subrutina de un movimiento x, primero llama el movimiento, cuando termina el movimiento verifica con el acelerómetro que no se encuentre tirado, si esta de pie continúa con el siguiente proceso.

3.4 Programa Seguidor De Pelota Naranja

En la Figura 3.4.1 se muestran los bloques de nuestro robot, la entrada de comandos es por comunicación serial, la comunicación con los motores es serial Half Duplex, la comunicación con la Cmucam es serial, la comunicación con el acelerómetro es comunicación SPI y el giroscopio es alimentado por ancho de pulso dando como respuesta un retardo el cual calcula la tarjeta de control.

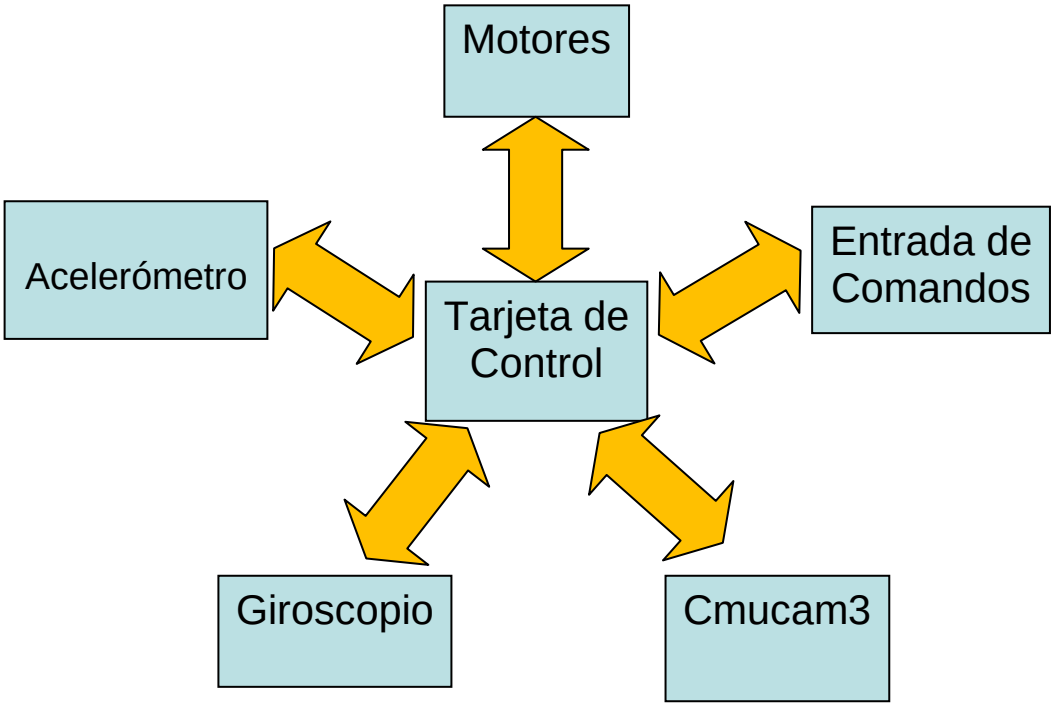

Figura 3.4.1 Diagrama de los bloques principales del robot

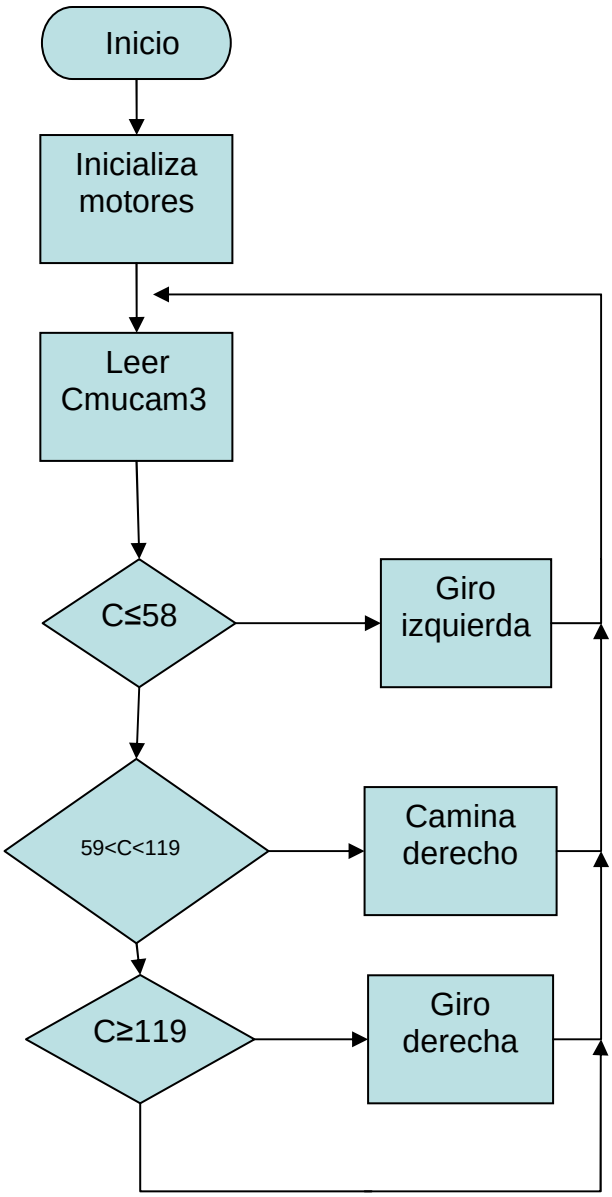

Figura 3.4.2 Diagrama de bloques seguidor de pelota naranja

Se programó una pequeña rutina para seguir una pelota de color naranja, la cual es bastante sencilla, la con muestro algoritmo para buscar el color naranja que tenemos en la Figura 3.2.1. Configurando la CMUCam3 a una resolución de 176x144 con lo cual lo único que necesitamos es la coordenada de el cúmulo de color en el eje x o sea que va de 0 a 176, ese valor lo va a recibir nuestro sistema de control, vamos a definir los umbrales de 0 a 58 en el cual el robot va a girar a la izquierda, de de 59 a 118 el robot va a caminar hacia delante y de 119 a 176 el robot va a girar a la derecha. Con esto podemos hacer un seguidor para la pelota de color naranja diagrama de flujo del sistema de control es el que se muestra en la Figura 3.4.2.

CONCLUSIONES

CONCLUSIONES

La construcción de cualquier tipo de sistema requiere de la inversión tanto de tiempo como dinero, la generación de tecnología siempre es una apuesta al futuro. Bajo esas condiciones cualquier avance en un área representa bases para futuras generaciones. La tecnología es un bien que no suele ser del carácter publico. Es por ello, que a través de los capítulos de esta tesis he desarrollado paso a paso los puntos medulares para la construcción de un robot humanoide; el cual cumple con los requerimientos de los existentes en otras partes del mundo y es provisto de 18 grados de libertad, más que algunos robots comerciales. Esta innovación era uno de los puntos centrales al ser propuesto.

La fácil programación de movimientos además de contar con una estructura manejable a modificaciones, si así se requiriera, el manejo de giroscopios y un sistema de visión adaptado, listo para su uso son puntos que se cumplieron en su totalidad. La estabilidad al dar pasos y el uso de sus dos grados de libertad en la ingle del humanoide dotan de una mejor representación de los movimientos hechos por el hombre.

Es claro que el robot aquí presentado es un pequeño avance en el diseño de robots humanoides, sin embargo, no deja de ser importante en sus aportaciones. Una de las premisas que infiere cualquier diseño es que mientras la plataforma en este caso el robot resuelva más cosas por sí solo, el diseño de software será más sencillo y por ende puede provocar que con un grado de complejidad elevado en el mismo se logren resultados sobresalientes.

Falta muchas cosas por hacer, el robot abre muchas posibilidades tanto en mejoras como en escalamiento del mismo. Se puede dotar de un mejor sistema de visión, la calidad de los motores, etc. Ahora se cuenta con una plataforma propia de la cual se conoce todo y es fácil modificar si así es requerido.

También puede fungir como modelo para hacer un humanoide de dimensiones mayores pues los retos de equilibrio, manejo de sensores y estructurales se deberán llevar bajo los mismos planteamientos con los cuales se resolvió el problema en este robot.

Por último, mencionaré que regularmente en este tipo de proyectos se ven involucrados varios especialistas. Para el diseño y construcción de todo el robot solo fue necesaria la asesoría de mis tutores, la aplicación de mis recursos y la experiencia en el tema para llevar a buen término dicho proyecto.

BIBLIOGRAFÍA

Referencias Electrónicas

[1] Reglas para el robocup categoría humanoide.
http://www.dei.unipd.it/~emg/downloads/HumanoidLeagueRules2007.pdf

[2] Página oficial de la CMUCam3
http://www.cmucam.org/

[3] Paper de robot humanoide esquipo Pioneros México
http://www.informatik.unifreiburg.de/~rc06hl/qualification/Pioneros_Mexico_Specs.pdf

[4] Página oficial del robot Robonova.
http://www.robonova.de

[5] http://www.lynxmotion.com/

[6] http://en.wikipedia.org/wiki/YCbCr

[7] http://www.ai.rug.nl/robocupathome/documents/rulebook.pdf

[8] http://mail.fibo.kmutt.ac.th/robocup2008/store/Pumas-UNAM_Mexico_TPD_02.pdf

[9] http://mail.fibo.kmutt.ac.th/robocup2008/store/teamosakakid_japan_tdp_01.pdf

[10] http://www.ais.uni-bonn.de/nimbro/papers/KI04WS9.pdf

[11] WIKIPEDIA La enciclopedia libre definición de robot
http://es.wikipedia.org/wiki/Robot

[12] WIKIPEDIA Definición de sensor
http://es.wikipedia.org/wiki/Sensor

[13] WIKIPEDIA definición de microcontrolador
http://es.wikipedia.org/wiki/Microcontrolador

[15] Artículo Sobre Baterías
http://html.rincondelvago.com/baterias.html

[16] BATERÍAS DE LITIO.
La alternativa al plomo y al cadmio.
http://www.cienciateca.com/ctslibat.html

[17]Articulo sobre las ventajas y desventajas de las baterías de litio.
http://www.imagendv.com/liion1.htm

[18]Articulo sobre robots industriales, del tipo manipulador
http://www.cpr2valladolid.com/tecno/cyr_01/robotica/industrial.htm

[19] http://www.infoab.uclm.es/labelec/Solar/electronica/elementos/servomotor.htm

[20]Curso de microcontroladores PIC16F877
Carlos Romero D.
Ingeniero en Electrónica.
http://www.scribd.com/doc/6306430/y-Pensar-Que-Habria-Sido-Hijo-Tuyo

[21]Guía de uso del MPLAB.
http://victronics.cl/noticias/mplab_proyecto.zip

[22] Artículo sobre lenguaje de programación PBP (Pic Basic Pro)
http://www.todopic.com.ar/pbp_sp.html

[23] Página de la empresa creadora de PBP micro Engineering Labs Inc.
Instalación de PBP en MPLAB IDE
http://melabs.com/support/mplab.htm

[24] Rama de estudiantes IEEE Universidad Carlos III de Madrid
http://www.ieee.uc3m.es/concurso_circ.htm

[25]Artículo sobre clasificación de los motores eléctricos
http://usuarios.lycos.es/mugresoft/clasificacion_general_de_los_motores_electricos.htm

[26] Hoja de datos del servomotor AX-12+ de Dinamixel.
http://www.robotis.com/zbxe/dynamixel_en

[27] Aspectos básicos de Robots Autónomos y Lineamientos para la Construcción de un Equipo de Fútbol Robótico
Ing. Balich, Néstor Adrián [e-mails:nestorbalich@8bytes.com.ar]
Facultad de Tecnología Informática
Universidad Abierta Interamericana
Chacabuco 90 1er. Piso
Buenos Aires – Argentina
Mayo, 2004
http://www.exa.unicen.edu.ar/cafr2004/pages_Spanish/papers/WCAFR2004-12.pdf

[28] Hoja de datos del acelerómetro H48c

http://www.parallax.com/dl/docs/prod/acc/HitachiH48C3AxisAccelerometer.pdf

[29]Piezo Gyro PK3 hoja de datos

http://www.robodacta.com.mx/UserFiles/File/Gyro_PK3.pdf

[30] Hoja de datos CMUCam3

http://www.seattlerobotics.com/CMUcam3_datasheet.pdf

Todas las referencias electrónicas fueron verificadas el 19 de junio de 2006

[31] MPLAB ICD 2, In-Circuit Debugger User's Guide

Microchip Technology Inc. 2003

[32] MANUAL DE USUARIO DEL PIC18FXX2

Microchip Technology inc. 2001

[33] Introduccion a la CMUCam3

Daniel Cortes Pichardo y
María de los A. Gamero González

Trabajo elaborado para el Laboratorio de Biorobótica de la UNAM

[34] Biomecánica De La Marcha Humana.

http://140.148.10.34/u_dl_a/tales/documentos/lep/hernandez_s_f/capitulo3.pdf

APÉNDICE

Código para leer el acelerómetro.

```
Get_H48C:
LOW CS
SHIFTOUT Dio, Clk, MSBFIRST, [%11\2, VRef\3] ' selecciona voltaje de referencia.
SHIFTIN Dio, Clk, MSBPOST, [rvCount\13] 'lee voltage de referencia.
HIGH CS
PAUSE 1
LOW CS
SHIFTOUT Dio, Clk, MSBFIRST, [%11\2, axis\3] ' selecciona el eje a leer.
SHIFTIN Dio, Clk, MSBPOST, [axCount\13] ' lee el voltaje del eje seleccionado.
HIGH CS
RETURN
```

Código para inicializar habilitar torque y colocar en posición firmes al robot.

```
INICIO:
 GOSUB INICIALIZA
        PAUSE 500
        GOSUB FIRMES
        PAUSE 3000
END
INICIALIZA:
        ID= $FE
        CWM=$0
        CCWM=$0
        CWS=$F
        CCWS=$F
        GOSUB INI
RETURN
INI:
        PORTB.0=1
        HSEROUT [255,255]
        HSEROUT [ID,9,3,24,1,1,CWM,CCWM,CWS,CCWS]
        CHK=~((ID + CWM + CCWM + CWS + CCWS + 38) & 255)  ; Calcula checksum
        HSEROUT [CHK]
        PAUSE 5
        PORTB.0=0
RETURN
MANDA1:
        FOR ID=1 TO 19
                PORTB.0=1
                POSICION=P[ID]
                HSEROUT
[255,255,ID,7,3,30,POSICION.BYTE0,POSICION.BYTE1,VELOCIDAD.BYTE0,VELOCIDAD.BYTE1]
                CHK=~((ID+POSICION.BYTE0+POSICION.BYTE1+VELOCIDAD.BYTE0+VELOCIDAD.BYTE1+40)     &
255)        ; checksum
                HSEROUT [CHK]
                PAUSE 5
                PORTB.0=0
        NEXT ID
RETURN
FIRMES:
        P[1]=530: P[2]=570: P[3]=760: P[4]=535: P[5]=505: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=501:P[12]=420:P[13]=275:P[14]=490:P[15]=511:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
RETURN
```

Código para dar un paso

```
FASE1:

        ‘Secuencia a los motores

        P[1]=570: P[2]=570: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=200: P[9]=511

        P[11]=551:P[12]=422:P[13]=275:P[14]=490:P[15]=550:P[16]=511:P[17]=820:P[18]=800:P[19]=511

        VELOCIDAD=$15
```

```
GOSUB MANDA1
GOSUB GIRO                  'Verifica giroscopio
IF CUENTA >600 THEN
P[2]= 550:P[12]=442         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE1
ENDIF
IF CUENTA <400 THEN
P[2]= 590:P[12]=402         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE1
ENDIF
PAUSE 200                   'Tiempo de respuesta
FASE2:
'Secuencia a los motores
P[1]=570: P[2]=570: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=200: P[9]=511
P[11]=551:P[12]=380:P[13]=385:P[14]=450:P[15]=550:P[16]=511:P[17]=820:P[18]=800:P[19]=511
VELOCIDAD=$20
GOSUB MANDA1
GOSUB GIRO                  'Verifica giroscopio
IF CUENTA >600 THEN
P[2]= 550:P[12]=400         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE2
ENDIF
IF CUENTA <400 THEN
P[2]= 590:P[12]=360         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE2
ENDIF
PAUSE 400                   'Tiempo de respuesta

FASE3:
'Secuencia a los motores
P[1]=570: P[2]=620: P[3]=760: P[4]=465: P[5]=530: P[6]=511: P[7]=200: P[8]=200: P[9]=511
P[11]=551:P[12]=492:P[13]=275:P[14]=400:P[15]=550:P[16]=511:P[17]=820:P[18]=800:P[19]=511
VELOCIDAD=$30
GOSUB MANDA1
GOSUB GIRO                  'Verifica giroscopio
IF CUENTA >600 THEN
P[2]= 600:P[12]=512         'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE3
ENDIF
IF CUENTA <400 THEN
P[2]= 640: P[12]=472        'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE3
ENDIF
PAUSE 500                   'Tiempo de respuesta

FASE4:
'Secuencia a los motores
P[1]=470: P[2]=625: P[3]=760: P[4]=465: P[5]=465: P[6]=511: P[7]=200: P[8]=200: P[9]=511
P[11]=455:P[12]=492:P[13]=275:P[14]=400:P[15]=467:P[16]=511:P[17]=820:P[18]=800:P[19]=511
VELOCIDAD=$30
GOSUB MANDA1
```

```
        GOSUB GIRO              'Verifica giroscopio
        IF CUENTA >600 THEN
        P[2]= 605:P[12]=512     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE4
        ENDIF
        IF CUENTA <400 THEN
        P[2]= 645:P[12]=472     'Correccion tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE4
        ENDIF
        PAUSE 800               'Tiempo de respuesta

FASE5:
        'Secuencia a los motores
        P[1]=470: P[2]=670: P[3]=640: P[4]=535: P[5]=435: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=445:P[12]=422:P[13]=275:P[14]=490:P[15]=467:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        GOSUB GIRO              'Verifica giroscopio
        IF CUENTA >600 THEN
        P[2]= 650:P[12]=442     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE5
        ENDIF
        IF CUENTA <400 THEN
        P[2]= 690:P[12]=402     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE5
        ENDIF
        PAUSE 500               'Tiempo de respuesta
FASE6:
        'Secuencia a los motores
        P[1]=470: P[2]=620: P[3]=640: P[4]=585: P[5]=435: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=445:P[12]=422:P[13]=275:P[14]=490:P[15]=467:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$20
        GOSUB MANDA1
        GOSUB GIRO              'Verifica giroscopio
        IF CUENTA >600 THEN
        P[2]= 600:P[12]=442     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE6
        ENDIF
        IF CUENTA <400 THEN
        P[2]= 640:P[12]=402     'Corrección tobillos
        GOSUB MANDA
        PAUSE 5
        GOTO FASE6
        ENDIF
        PAUSE 500               'Tiempo de respuesta

FASE7:
        'Secuencia a los motores
        P[1]=470: P[2]=500: P[3]=760: P[4]=585: P[5]=435: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=445:P[12]=422:P[13]=275:P[14]=490:P[15]=467:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$10
        GOSUB MANDA1
        GOSUB GIRO              'Verifica giroscopio
        IF CUENTA >600 THEN
```

```
P[2]= 480:P[12]=442          'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE7
ENDIF
IF CUENTA <400 THEN
P[2]= 520:P[12]=402          'Corrección tobillos
GOSUB MANDA
PAUSE 5
GOTO FASE7
ENDIF
PAUSE 500                    'Tiempo de respuesta
```

Código para mandar la posición de firmes

```
FIRMES:
        P[1]=530: P[2]=570: P[3]=760: P[4]=535: P[5]=505: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=501:P[12]=420:P[13]=275:P[14]=490:P[15]=511:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
RETURN
```

Código para girar a la derecha.

```
GIRODER:
        P[1]=480: P[2]=570: P[3]=760: P[4]=535: P[5]=465: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=460:P[12]=420:P[13]=275:P[14]=490:P[15]=470:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$35
        GOSUB MANDA1
        PAUSE 400

        P[1]=470: P[2]=720: P[3]=600: P[4]=535: P[5]=445: P[6]=511: P[7]=200: P[8]=220: P[9]=511
        P[11]=450:P[12]=400:P[13]=275:P[14]=490:P[15]=470:P[16]=441:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        PAUSE 900

        P[1]=450: P[2]=570: P[3]=760: P[4]=535: P[5]=445: P[6]=511: P[7]=200: P[8]=220: P[9]=511
        P[11]=450:P[12]=400:P[13]=275:P[14]=490:P[15]=470:P[16]=441:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$20
        GOSUB MANDA1
        PAUSE 600

        P[1]=570: P[2]=570: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=220: P[9]=511
        P[11]=551:P[12]=400:P[13]=275:P[14]=490:P[15]=570:P[16]=441:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        PAUSE 600

        P[1]=570: P[2]=590: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=220: P[9]=511
        P[11]=551:P[12]=290:P[13]=400:P[14]=490:P[15]=570:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$20
        GOSUB MANDA1
        PAUSE 1000

        P[1]=570: P[2]=570: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=551:P[12]=422:P[13]=275:P[14]=490:P[15]=550:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        PAUSE 800

        P[1]=530: P[2]=570: P[3]=760: P[4]=535: P[5]=505: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=501:P[12]=420:P[13]=275:P[14]=490:P[15]=511:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        PAUSE 400

RETURN
```

Código para caminar hacia la derecha.

```
CAMINADER:
'--------RECARGA IZAUIERDA------------
        P[1]=480: P[2]=570: P[3]=760: P[4]=535: P[5]=465: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=460:P[12]=420:P[13]=275:P[14]=490:P[15]=470:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$35
        GOSUB MANDA1
        PAUSE 400

        P[1]=470: P[2]=570: P[3]=760: P[4]=535: P[5]=405: P[6]=511: P[7]=200: P[8]=250: P[9]=511
        P[11]=440:P[12]=420:P[13]=275:P[14]=490:P[15]=470:P[16]=511:P[17]=820:P[18]=700:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        PAUSE 800

        P[1]=440: P[2]=570: P[3]=760: P[4]=535: P[5]=405: P[6]=511: P[7]=200: P[8]=250: P[9]=511
        P[11]=480:P[12]=420:P[13]=275:P[14]=490:P[15]=470:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        PAUSE 500
        '-------------------
        P[1]=570: P[2]=570: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=250: P[9]=511
        P[11]=551:P[12]=420:P[13]=275:P[14]=490:P[15]=590:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        PAUSE 1200

        P[1]=570: P[2]=570: P[3]=760: P[4]=535: P[5]=530: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=551:P[12]=440:P[13]=365:P[14]=450:P[15]=550:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$20
        GOSUB MANDA1
        PAUSE 600


        P[1]=530: P[2]=570: P[3]=760: P[4]=535: P[5]=505: P[6]=511: P[7]=200: P[8]=200: P[9]=511
        P[11]=501:P[12]=420:P[13]=275:P[14]=490:P[15]=511:P[16]=511:P[17]=820:P[18]=800:P[19]=511
        VELOCIDAD=$30
        GOSUB MANDA1
        PAUSE 400

RETURN
```

Código principal de nuestro sistema de control

```
INICIO:
SERIN2 PORTA.1,84,[DEC4 B0]  'RECIBE EL DATO DE LA CMUCAM3 Y LO GUARDA EN B0
IF B0 < 58 THEN
GOSUB GIROIZQ       'EN EL CASO DE QUE ESTE DEL LADO IZQUIERDO GIRA UN PASO A LA IZQUIERDA
ENDIF
IF B0 > 58 AND B0 < 119 THEN    ' EN EL CASO DE QUE ESTE CENTRADA CAMINA ADELANTE
GOSUB CAMINA1
ENDIF
IF B0 > 119 THEN    ' EN EL CASO DE QUE ESTE DEL LADO DERECHO GIRA UN PASO A LA DERECHA
GOSUB GIRODER
ENDIF
GOSUB VERCAIDO
IF CAIDO = 0  THEN
GOTO INICIO
ENDIF
IF CAIDO =1 THEN
GOTO LEVANTADEL
ENDIF
IF CAIDO=2 THEN
GOTO LEVANTAATRAZ
ENDIF
GOTO INICIO      '  VUELVE A EL BUCLE.
END
```

Delimitación de umbrales para reconocer color naranja Cmucam3

```
t_pkt.lower_bound.channel[CC3_CHANNEL_RED] = 150;
t_pkt.upper_bound.channel[CC3_CHANNEL_RED] = 255;
t_pkt.lower_bound.channel[CC3_CHANNEL_GREEN] = 0;
t_pkt.upper_bound.channel[CC3_CHANNEL_GREEN] = 50;
t_pkt.lower_bound.channel[CC3_CHANNEL_BLUE] = 0;
t_pkt.upper_bound.channel[CC3_CHANNEL_BLUE] = 50;
```

Codigo para Cmucam3 entrado del cúmulo de color naranja

```
                y = 0;
                x = 0;
                min_b = 0;
                centroidex = 0;
                centroidey = 0;
                puntos = 0;
                while (cc3_pixbuf_read_rows (img1.pix, 1))
                {
                        // Lee una fila de la camara.
                        for (x = 0; x < img1.width; x++)
                        {
                                // Obtiene un pixel
                                cc3_get_pixel (&img1, x, 0, &my_pix);
// Busqueda del Color Naranja (pelota)
                                if ( (my_pix.channel[CC3_CHANNEL_RED] >= min_r) &&
(my_pix.channel[CC3_CHANNEL_RED] <= max_r)
                                && (my_pix.channel[CC3_CHANNEL_GREEN] >= min_g) &&
(my_pix.channel[CC3_CHANNEL_GREEN] <= max_g)
                                && (my_pix.channel[CC3_CHANNEL_BLUE] >= min_b) &
(my_pix.channel[CC3_CHANNEL_BLUE] <= max_b))
                                {
                                        centroidex = centroidex + x;
                                        centroidey = centroidey + y;
                                        puntos = puntos + 1;
                                }
                        }
                        y++;
                }
```