UNIVERSIDAD PONTIFICIA COMILLAS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO EN INFORMATICA

PROYECTO FIN DE CARRERA

GUIADO DE UN BRAZO ROBÓTICO MEDIANTE UN SISTEMA DE VISIÓN ARTIFICIAL

AUTOR: **MANUEL LUCANIA FESSER**

MADRID, Septiembre 2007

Autorizada la entrega del proyecto del alumno/a:
MANUEL LUCANIA FESSER
.....

EL DIRECTOR DEL PROYECTO
ÁLVARO SÁNCHEZ MIRALLES

Fdo.: ..... Fecha: ...../ ...../ .....

Vo Bo del Coordinador de Proyectos
MIGUEL ÁNGEL SANZ BOBI

Fdo.: ..... Fecha: ...../ ...../ .....

Índice

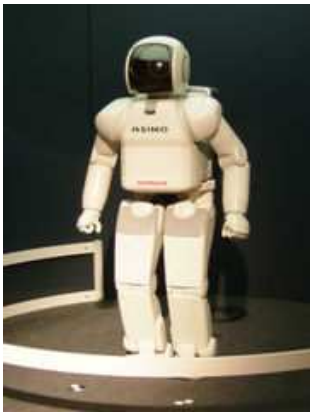
1. INTRODUCCIÓN

La Robótica surge como confluencia de varias materias: Mecánica, Electricidad, Electrónica, Automática e Informática. Hasta hace poco la mayor parte de las definiciones y clasificaciones iban en línea con el robot industrial [RENT00], por ser éste el más utilizado. Sin embargo, en la actualidad un nuevo grupo de robots, los robots de servicio, están en fase de desarrollo. Este segundo término englobaría los robots dedicados a cuidados médicos, educación, de uso en oficinas, de uso doméstico, para intervención en ambientes peligrosos, para aplicaciones espaciales, para aplicaciones submarinas, para agricultura, o para uso militar. Los robots de servicio son dispositivos electromecánicos móviles o estacionarios, dotados normalmente de uno o varios brazos mecánicos independientes, controlados por un programa de ordenador y que realizan tareas no industriales sino de servicio.

Este proyecto surge por mi interés personal hacia los robots humanoides. Un robot humanoide es un robot de servicio que imita la apariencia humana y algunos aspectos de su conducta. Tiene manos y brazos libres que utiliza para su interacción con el entorno, así como sistemas de visión artificial incorporados. Aunque no hay una aplicación determinante para este tipo de robots, actualmente la industria los está focalizando hacia el mercado del entretenimiento. También se estudian otras áreas de aplicación, como el mantenimiento en plantas industriales, construcción, servicio en el hogar, bienestar y cuidado de enfermos, o rescate en catástrofes.

La construcción de un robot que imite convincentemente (aunque sea mínimamente) la libertad de gestos y movimiento humanos, es una labor de una enorme complejidad técnica. De hecho, es un problema que en varias instancias está todavía abierto a la investigación y a la mejora. Pero existen ya varios ejemplos bastante meritorios en ese sentido, de robots humanoides que emulan ciertas conductas y capacidades humanas. Un gran conocido en este sentido, es el robot Asimo de Honda, que es capaz de marchar en dos pies, de subir y bajar escaleras y de otra serie de proezas de locomoción bípeda. También destaca COG, un robot humanoide del MIT (Massachusetts Institute of Technology) que intenta emular la sensorización y motricidad de la parte superior de un cuerpo humano.

Construir un humanoide completo (con piernas, brazos y cabeza) representaría un objetivo inalcanzable para una única persona que dispone de pocos meses de trabajo para llevarlo a cabo. Por ello pensé en reducir el problema a sus elementos mínimos: un brazo y uno ojos. Y a partir de estas limitaciones asigné al robot una funcionalidad (jugar al ping-pong) que pudiera demostrar sus capacidades.



Asimo

COG

2. OBJETIVOS DEL PROYECTO

El objetivo principal es la construcción física del robot (brazo y cabeza) y sus circuitos de control, la implementación de unos algoritmos básicos de desplazamiento que permitan mover los motores del robot a cualquier posición, y la captura y procesamiento de imágenes en tiempo real. El brazo tendrá cinco grados de libertad para emular los movimientos del brazo derecho de una persona, desde el hombro hasta la muñeca. Cada uno de los ojos tendrá dos grados de libertad (arriba-abajo, izquierda-derecha). Las órdenes de movimiento se lanzarán desde un ordenador hacia el sistema robótico, indicando la posición en la que deben situarse los motores. Una vez desplazados a la posición deseada, los motores deberán mantener dicha posición hasta recibir una nueva orden.

Como objetivos opcionales tenemos: el control de la velocidad de los motores, la implementación de un algoritmo de detección de bordes, el desarrollo de un algoritmo para la detección del centro de masas de una pelota de ping-pong naranja, la detección de la velocidad de la pelota, el seguimiento de una pelota en movimiento con los ojos, la resolución del problema cinemático directo, la resolución del problema cinemático inverso, la creación de algoritmos que permitan al extremo del brazo señalar y seguir el recorrido de una pelota en movimiento, y la implantación de reglas de juego para que el robot pueda jugar al ping-pong.

Se han cumplido todos los objetivos básicos, ya que:

- Se han construido un brazo y unos ojos cuya posición es controlable desde un PC.
- Se muestran por pantalla las imágenes capturadas y se realizan procesamientos simples (por ejemplo el paso de las imágenes a blanco y negro).

Pero además, se han cumplido los siguientes objetivos opcionales:

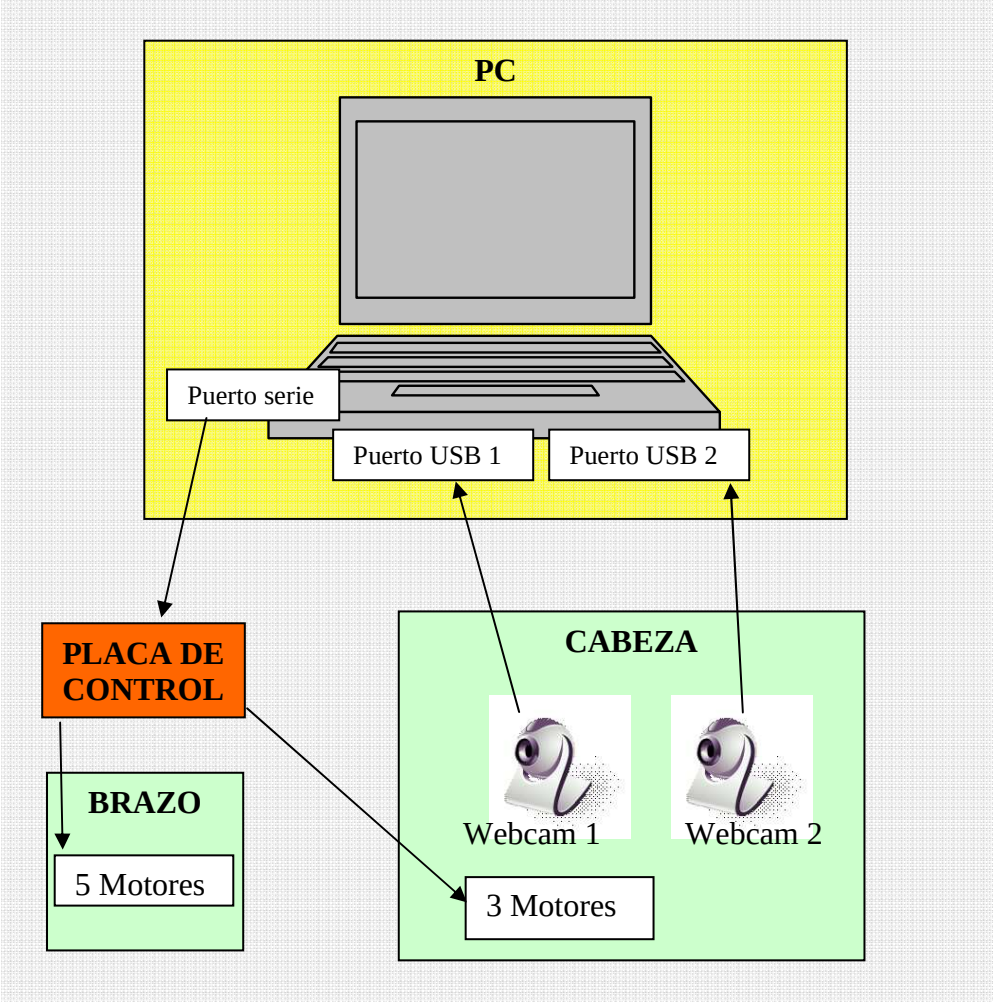
- Control de la velocidad: los servomotores que venden en el mercado sólo permiten el control de la posición. Para controlar la velocidad se ha tenido que eliminar el control de posición que llevan de serie, implementar un nuevo

control de posición partiendo de cero y además añadir un control de velocidad propio.

- Detección de bordes: se ha implementado el algoritmo de Sobel, ya que es muy utilizado para la detección de bordes. Detecta cambios bruscos en la luminosidad de las imágenes. Se muestran por pantalla los bordes resaltados en negro.
- Detección del centro de masas de una pelota de ping-pong: se ha desarrollado un algoritmo que realiza una búsqueda de puntos de tonalidad anaranjada y a continuación calcula su centro de masas. Se muestra por pantalla un punto azul que resalta el centro de la pelota.

3. METODOLOGÍA DE TRABAJO

Se ha estructurado el trabajo partiendo de un diseño básico del sistema, compuesto por cuatro elementos:



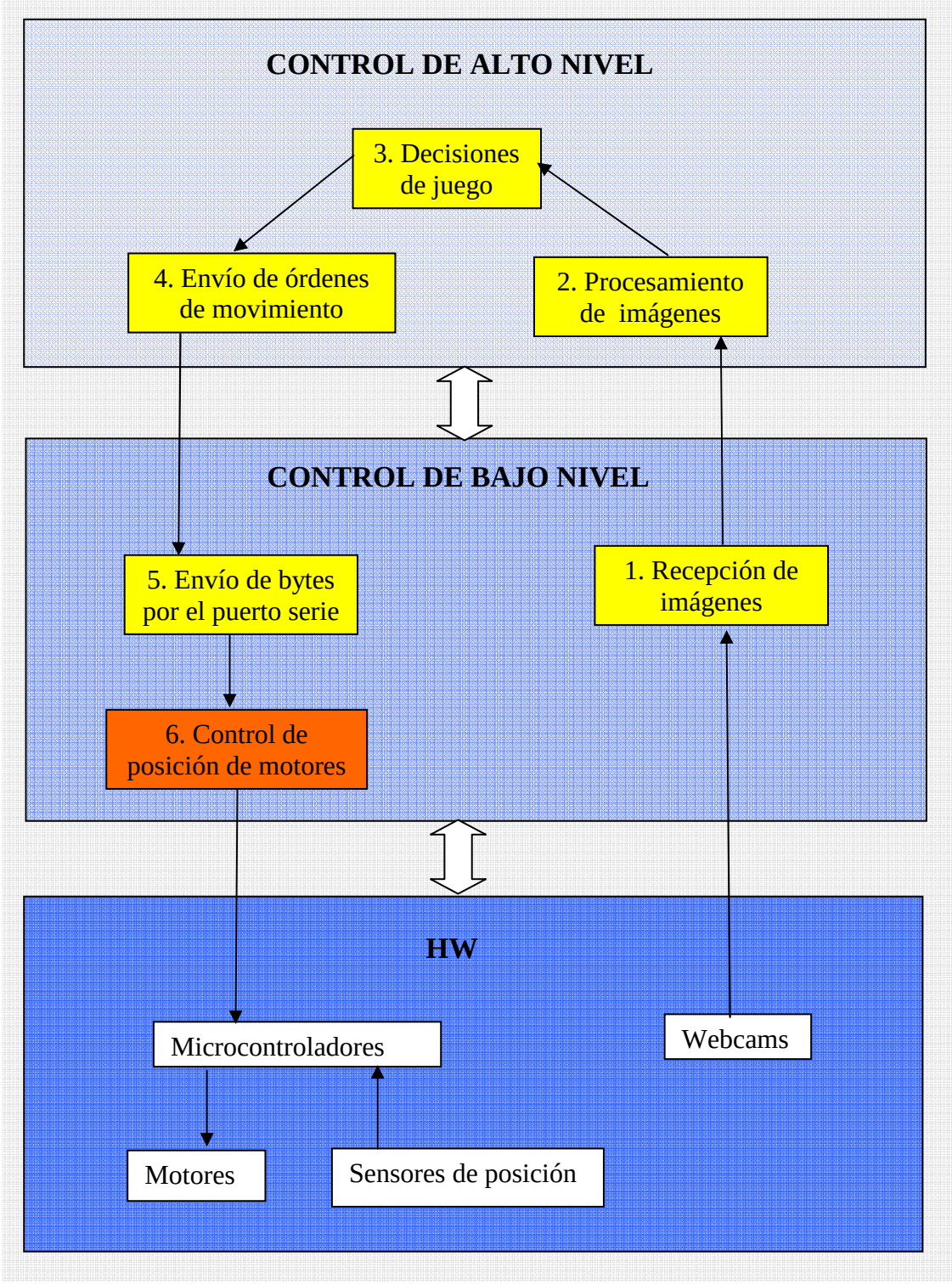
1.- PC: representa el 'cerebro' del robot. Se encarga de tomar decisiones de alto nivel acerca de los movimientos de brazos y ojos. A través del puerto serie transmite a la placa de control las órdenes que ejecutarán los motores. Además recibe y procesa las imágenes de las webcams, para detectar la posición de la pelota.

2.- Placa de control: Recibe por el puerto serie las órdenes del PC y las convierte en señales PWM de control dirigidas a los motores. Está constituida por microcontroladores y etapas de potencia. Los microcontroladores aportan una inteligencia de nivel inferior, destinada a la ejecución de órdenes y al control constante de la posición de los motores.

3.- Cabeza robótica: Consta de 2 webcams que transmiten la señal de vídeo al PC. Funcionan a modo de ojos que se mueven horizontal y verticalmente, impulsados por 3 motores que reciben señales desde la placa.

4.- Brazo robótico: Consta de 5 motores que también reciben señales PWM desde la placa de control. Imita los movimientos del brazo derecho de un humano.

Veamos el diagrama de flujo de la arquitectura básica del sistema:



El gráfico se estructura en tres capas, ordenadas de mayor a menor nivel de abstracción. En la parte inferior tenemos los elementos físicos, y a medida que ascendemos encontramos niveles más avanzados en la lógica de control. En amarillo aparecen los procesos llevados a cabo por el PC y en naranja los que realiza la tarjeta de control.

El flujo de control es el siguiente:

1. Las webcams capturan imágenes de la pelota de ping-pong y envían el vídeo al puerto USB del PC. Éste, recibe las imágenes, las muestra por pantalla y accede a los datos en su forma más básica: matrices de puntos de color (cada color viene representado por un número en dicha matriz).
2. A continuación, el ordenador procesa las imágenes para extraer información acerca de la posición de la pelota de ping-pong.
3. Se ponen en marcha reglas para tomar decisiones de juego. Son del tipo: "si la pelota está en la posición (15,20,3) con un vector de velocidad asociado (1,3,2), debo mover mis ojos a la posición (5,4) para seguir su movimiento, y mover el brazo a la posición (13,14,2) para golpear la pelota".
4. Las decisiones adoptadas se traducen en órdenes de movimiento de bajo nivel, del tipo: "mover motor 1 a la posición 14, motor 2 a la posición 5, motor 3….". Se envía una orden a cada motor.
5. Las órdenes se traducen en bytes que se envían desde el puerto serie a la tarjeta de control.
6. Los microcontroladores insertados en la placa reciben las órdenes y las procesan. Lanzan señales PWM a los motores para llevarlos a la posición deseada, y mantenerlos en dicha posición hasta nueva orden.

4. RECURSOS EMPLEADOS

En el "D*ocumento II. Presupuesto"* se detallan cada uno de los componentes empleados, y su precio de mercado. Aquí simplemente se enumeran los más importantes:

- Ordenador portátil Hp Pavilion
  - S.O. Windows XP
  - Procesador AMD Thurion 64x2
  - Java 1.6, JMF 2.1

- Tarjeta de control
  - Proto-Board (matriz 15x30)
  - Microcontroladores PIC16F876
  - Puerto serie

- Brazo robótico
  - Placas de aluminio
  - Ejes de fibra de carbono
  - Puentes de madera
  - Rodamientos
  - 3 servos Hitec HSR-5995TG, 1 servo Hitec HS-81, 1 servo Futaba S3003

- Cabeza robótica
  - 2 webcams Logitech Quickcam Messenger
  - Placas de aluminio
  - Ejes de fibra de carbono
  - Rodamientos
  - 2 servos Hitec HS-81, 1 servo Futaba S3003

5. BRAZO ROBÓTICO

En un primer momento se construyó físicamente el brazo. Posteriormente, se empezó a trabajar en el control de los movimientos. Aunque era importante tener el dispositivo físico terminado para poder empezar a probar el funcionamiento real, las primeras pruebas de control comenzaron en paralelo.

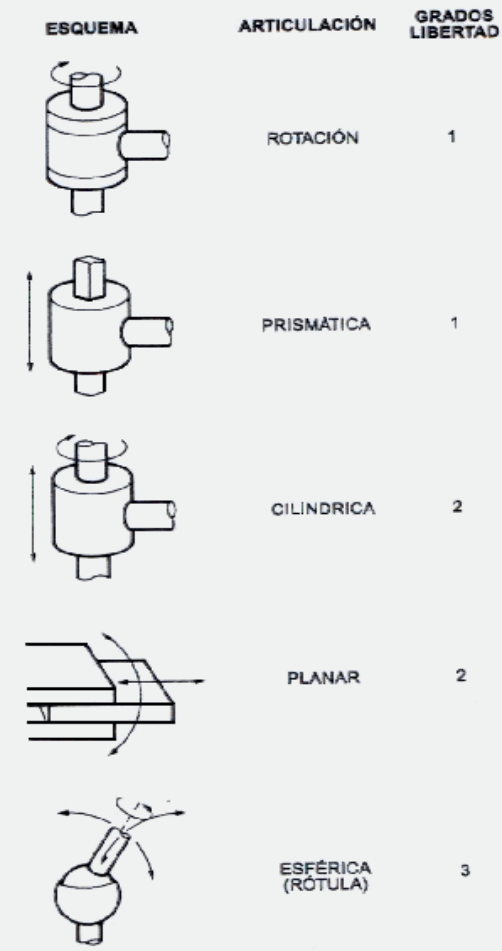
5.1. DISEÑO MECÁNICO

Este apartado trata acerca de la construcción física de un brazo robótico de dimensiones humanas.

Mecánicamente, un robot está formado por una serie de ejes o eslabones [RENT00] unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. El movimiento de cada articulación puede ser de desplazamiento, giro, o una combinación de ambos. De esta forma, existen seis tipos de articulaciones diferentes: de rotación o bisagra, prismática, cilíndrica, planar y esférica.

Cada uno de los movimientos independientes que puede realizar una articulación con respecto a la anterior, se denomina grado de libertad (GDL) [BARR99]. En la figura de la página siguiente se indica el número de grados de libertad de cada tipo de articulación.

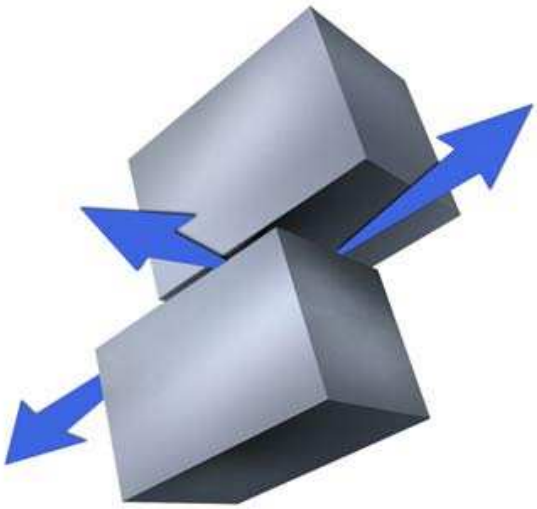
En este proyecto todas las articulaciones empleadas son del primer tipo (rotación). Aunque se trata de una simplificación del modelo humano, se consiguen emular todos los movimientos y grados de libertad de una persona, desde el hombro hasta la muñeca. Es decir, no se emulan los movimientos de las manos (los dos GDL de la muñeca, y los movimientos de los dedos).



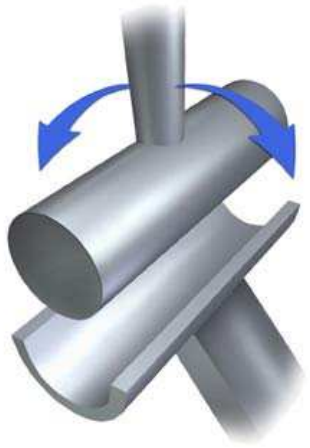
Los seis tipos de articulaciones

En el cuerpo humano, Las articulaciones se muestran en formas y tamaños variados. La unión tipo bisagra hace que nuestros codos y rodillas funcionen como una puerta. Un jugador de ping-pong utiliza el tremendo rango de movimiento de la articulación esférica de su hombro para orientar su brazo y lanzar bolas rápidamente. Y las articulaciones deslizantes de la columna vertebral hacen que la espalda sea tan flexible.

La articulación esférica [SOBO97] que se encuentra en los hombros permite un movimiento radial en casi cualquier dirección. Esta articulación es emulada en el robot mediante un conjunto de tres articulaciones de bisagra.



En una articulación deslizante, los huesos se desplazan uno con respecto al otro. Las articulaciones metacarpiales y metatarsiales de nuestras manos son deslizantes. Para este proyecto no han sido necesarias.



Una articulación de bisagra, como la del codo, permite la extensión y retracción del brazo.

Ya que aquí se trata la construcción un robot humanoide, tal vez la mejor forma de comprender este apartado sea dividir la estructura que da soporte físico al brazo en:

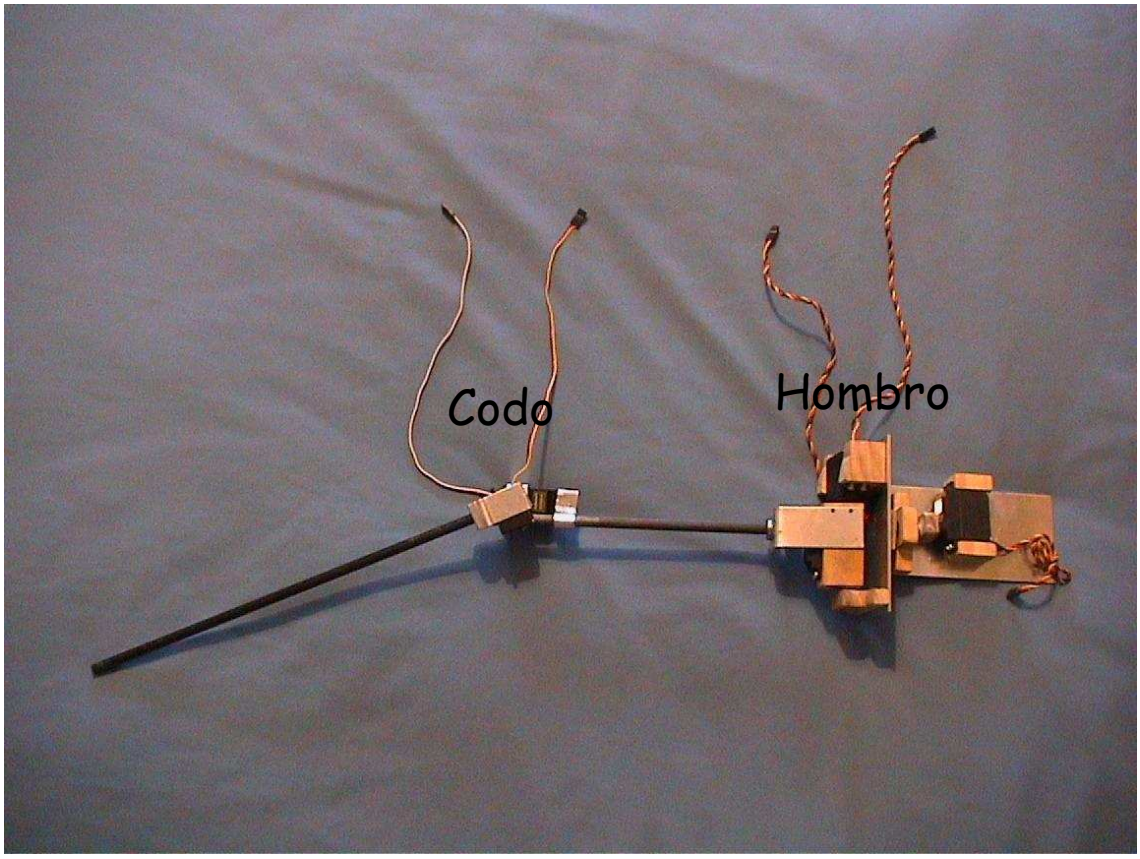
- "Huesos": los huesos son las 2 barras de color negro que aparecen en la foto de la página siguiente.
- "Articulaciones": (entendido en el sentido anatómico de la palabra, no con el significado de la Robótica). Las articulaciones son las zonas intermedias de unión en las que se da el movimiento de los huesos. Están formadas por las placas de aluminio, los puentes y soportes de madera, y los rodamientos.
- "Músculos": los músculos serían los motores.

O si preferimos verlo desde la perspectiva de la Robótica, podemos decir que cada articulación/eje (ahora sí entendiendo "articulación" desde el punto de vista de los robots) lleva asociado un sistema de accionamiento [BARR99]. Dicho sistema está constituido por:

- Actuadores: los dispositivos motores que generan el movimiento de acuerdo con las órdenes del sistema de control.
- Sistemas de transmisión y reducción: los elementos capaces de trasladar y modificar el movimiento generado por el actuador hasta la articulación.
- Sensores de posición y velocidad: dispositivos de captación de la información sobre la posición y velocidad de cada eje, cuya salida es recogida por el sistema de control para poder controlar el movimiento.

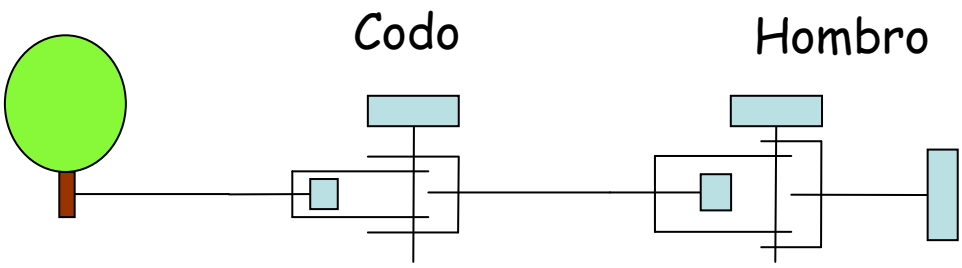
5.1.1. ESTRUCTURAS DE SOPORTE ("Huesos" y "Articulaciones")

Tomando el enfoque anatómico, podemos distinguir 2 articulaciones: la articulación del hombro y la articulación del codo. Cada articulación está formada por la confluencia de un conjunto de huesos (las barras negras).



Los materiales empleados han sido: madera, aluminio y fibra de carbono. Se han seleccionado estos y no otros, teniendo en cuenta que lo primordial era conseguir la máxima ligereza y la máxima resistencia.

El aluminio se ha utilizado sólo para las placas planas de soporte de los motores. La madera se escogió porque era necesario crear una estructura rígida con cierto volumen que pudiera soportar a una altura determinada los rodamientos que facilitan el giro de los "huesos".

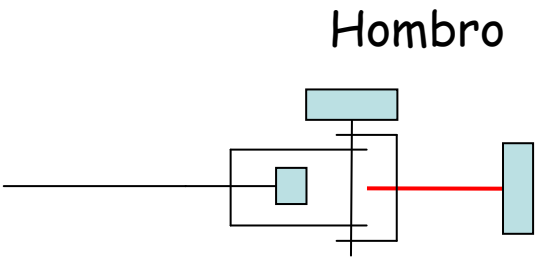


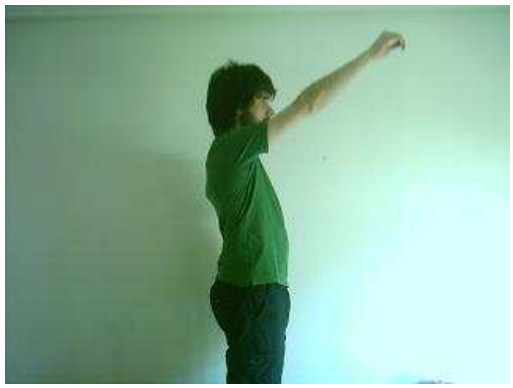
Como podemos apreciar en el esquema anterior, se muestra un brazo robótico derecho visto de frente, que consta de 2 articulaciones:

- Articulación del hombro: 3 grados de libertad (ejes 1, 2 y 3)
- Articulación del codo: 2 grados de libertad (ejes 4 y 5)

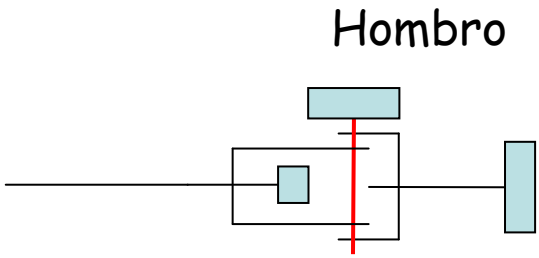
Veamos en detalle dicho esquema:

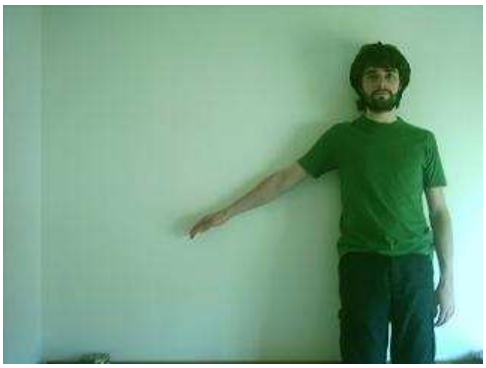
- **Eje 1**: Permite el movimiento de flexión del hombro.



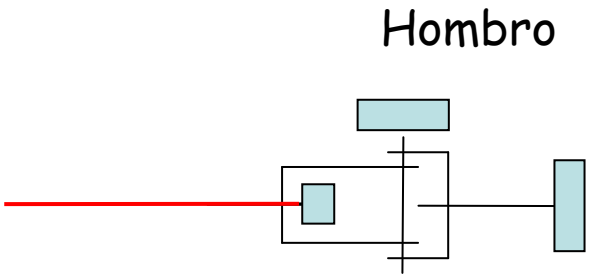


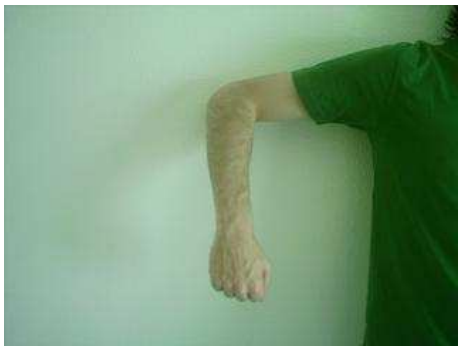
- **Eje 2**: Permite el movimiento de abducción del hombro



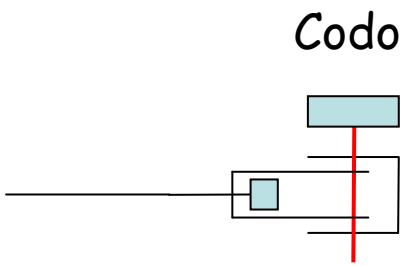


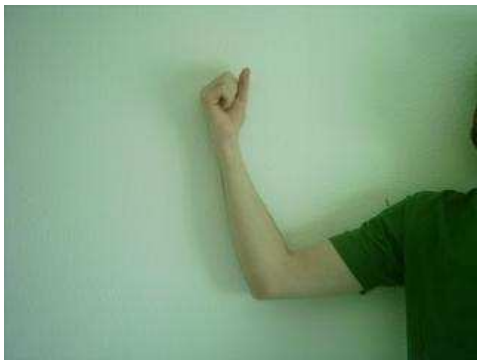
- **Eje 3**: Movimiento de rotación del hombro



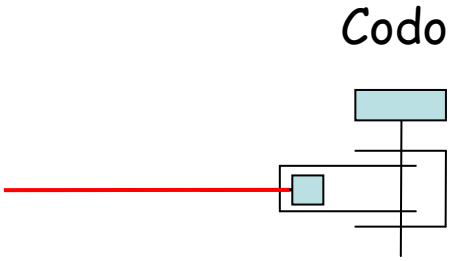


- **Eje 4**: Flexión/extensión del codo



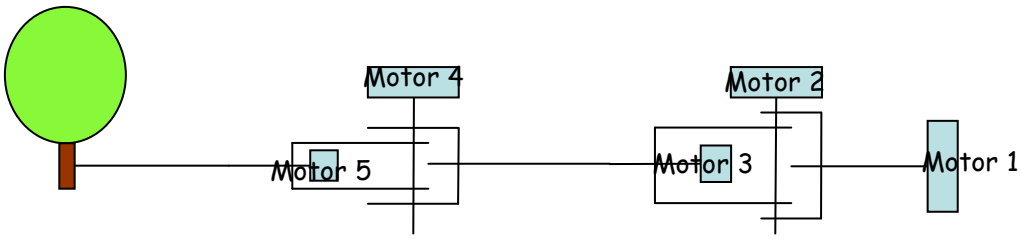


- **Eje 5**: Rotación del codo



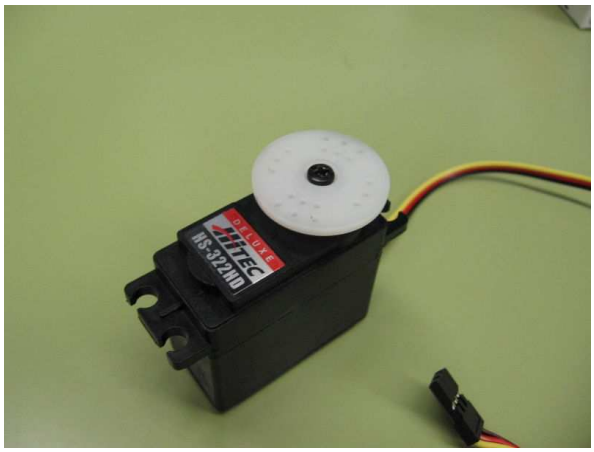
5.1.2. MOTORES ("Músculos")

Los actuadores (también denominados accionamientos) son el conjunto de elementos que hacen posible el movimiento de cada eje del robot. Representan los músculos del humanoide. Son dispositivos motores que generan el movimiento de acuerdo con las órdenes del sistema de control. Puesto que los pares estáticos que deben vencer los actuadores dependen directamente de la distancia de las masas al actuador, los motores se han dispuesto lo más cerca posible de la base del brazo (generalmente los robots tienen uno de sus extremos fijos, también llamado base).



En el esquema, podemos apreciar la disposición de los motores (en color azul)

Los accionamientos utilizados en robótica son clasificados en función del tipo de energía que utilizan, en: neumáticos, hidráulicos y eléctricos. Debido a sus características de control, sencillez, precisión y alta fiabilidad (además de su reducido coste económico) se han escogido para este proyecto motores eléctricos. Concretamente para este proyecto se han empleado servos modificados.



Servo de la marca Hitec, con sus 3 terminales (amarillo, rojo y negro)

Un servomotor (también llamado servo) es un dispositivo similar a un motor de corriente continua, que tiene la capacidad de ubicarse en cualquier posición dentro de su

rango de operación, y mantenerse estable en dicha posición. Los servos se utilizan frecuentemente en sistemas de radio control y en robótica. Están conformados por tres elementos:

- 1. Motor de corriente continua

Es el elemento que le brinda movilidad al servo. Cuando se aplica un potencial a sus dos terminales, este motor gira en un sentido a su velocidad máxima. Si el voltaje aplicado a sus dos terminales es inverso, el sentido de giro también se invierte.

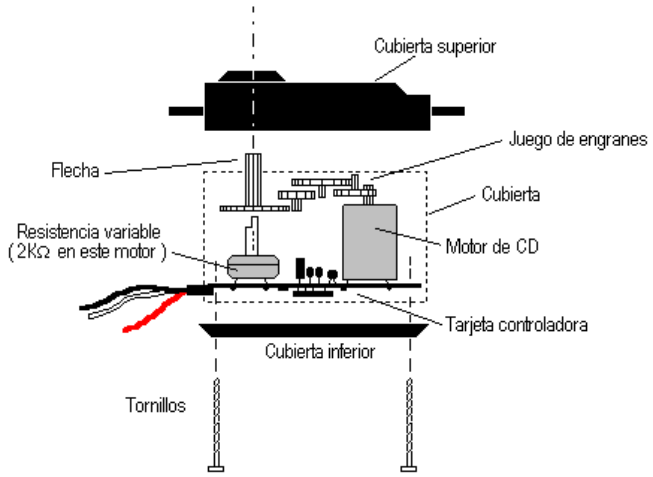
- 2. Engranajes reductores

Se encargan de convertir gran parte de la velocidad de giro del motor de corriente continua en torque.

- 3. Circuito de control

Este circuito es el encargado del control de la posición del motor. Recibe los pulsos de entrada y ubica al motor en su nueva posición dependiendo de los pulsos recibidos. Los servomotores tienen 3 terminales:

- Terminal positivo: Recibe la alimentación del motor (5 voltios)
- Terminal negativo: Referencia tierra del motor (0 voltios)
- Entrada de señal: Recibe la señal PWM de control del motor



Es posible modificar un servomotor para obtener un motor de corriente continua que, si bien ya no tiene la capacidad de control del servo, conserva la fuerza, velocidad y baja inercia que caracteriza a estos dispositivos. Y eso es precisamente lo que se ha hecho en este proyecto. La modificación llevada a cabo incluyó la completa supresión del circuito de control, y el añadido de dos terminales adicionales. De esta forma, los elementos quedaron finalmente así:

- 1. Motor de corriente continua

Permanece igual que antes. Pero ahora no sólo se permite la inversión del voltaje aplicado a sus dos terminales, sino también la variación de dicho voltaje. Gracias a esta variación se controla la velocidad, cosa que antes no podía hacerse.

- 2. Engranajes reductores

Se conservan tal cual.

- 3. Circuito de control

El circuito ha sido suprimido físicamente, y desechado. Ya no es necesario, puesto que a cada servo se le ha asignado un microcontrolador programado que realiza el control de movimientos. Pero sí que se conserva el potenciómetro (resistencia variable), para medir posiciones. Además, se añaden dos nuevos terminales, quedando la siguiente configuración:

  - Terminal 1 (conectado al motor): Recibe la alimentación del motor. Varía de 0 a 6 voltios).
  - Terminal 2 (al motor): Recibe la alimentación del motor. Varía de 0 a 6 voltios.
  - Terminal 3 (al potenciómetro): A través de él el potenciómetro recibe la señal de tierra.
  - Terminal 4 (al potenciómetro): Por él circula la señal de alimentación de 5 voltios que recibe el potenciómetro.
  - Terminal 5 (al potenciómetro): Para la señal de control que sale del potenciómetro hacia el microcontrolador. Según la intensidad de dicha señal, el microcontrolador calculará la posición en la que se encuentra el motor.

Las especificaciones de los servos incluyen la potencia y el peso q soportan, y en base a ello se han escogido los servo. Los motores del hombro tienen que soportar un par de fuerza mucho mayor, y por eso son más potentes.

5.2. CONTROL DE MOVIMIENTOS

En este apartado se explica en detalle el control del brazo robótico. A modo de resumen, y como adelanto, podría decirse que el funcionamiento es el siguiente:

- El ordenador lanza órdenes a un sistema de microcontroladores y sensores insertados en el brazo.
- Los microcontroladores modulan sus señales de control hacia los servos, en función de dichas órdenes y de la posición medida en cada instante.

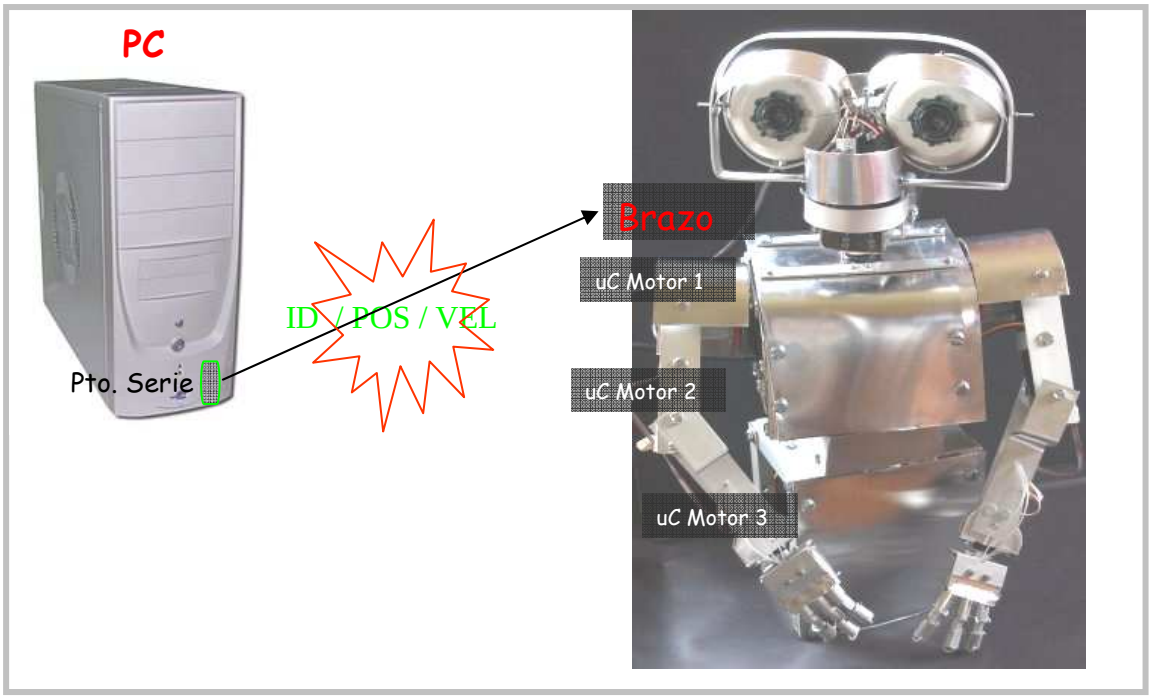
5.2.1. FUNCIONAMIENTO GENERAL

¿Cuáles son los pasos que se dan internamente en el sistema para que el brazo llegue a moverse a una posición determinada? Lo más sencillo para comprender el funcionamiento, es visualizar las fases de un esquema general que explica el comportamiento a lo largo de 3 etapas:

- ETAPA 1. ENVÍO

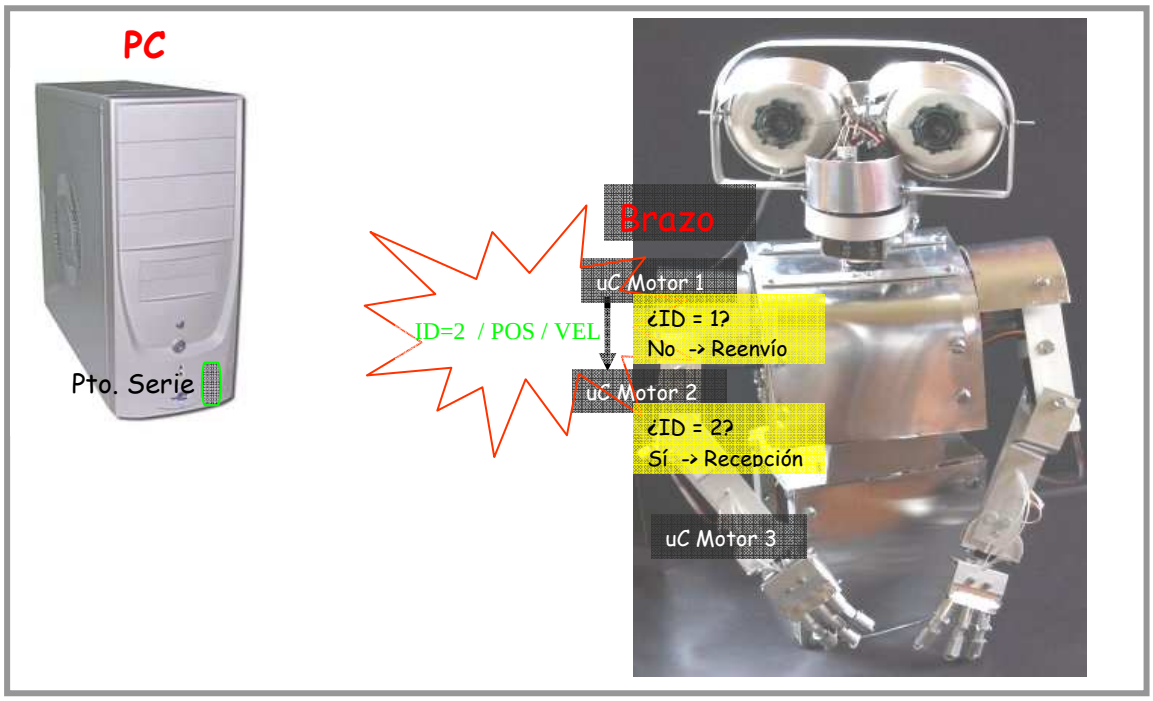
Un programa Java envía las órdenes al brazo, a través del puerto serie. Concretamente, para cada movimiento envía 5 órdenes (5 bytes, uno para cada motor). En cada byte van 3 informaciones:

- ID: El identificador del motor al cual se lanza la orden (un nº del 1 al 5, ya que hay 5 motores)
- POS: La posición en la que deberá situarse el motor (16 posiciones diferentes, 16 ángulos).
- VEL: La velocidad a la que hay que hacer el movimiento. Para simplificar el control solamente se han implementado 2 velocidades (0=lento, 1=rápido).



➢ ETAPA 2. RECEPCIÓN O REENVÍO

Los 5 bytes enviados los recibe un microcontrolador (concretamente el microcontrolador encargado del motor 1). El programa grabado en el uC, cada vez que reciba un byte, mirará el identificador, y si el mensaje no es para él lo reenviará al siguiente PIC (y así sucesivamente hasta que el PIC destinatario se queda con su mensaje para procesar la orden).



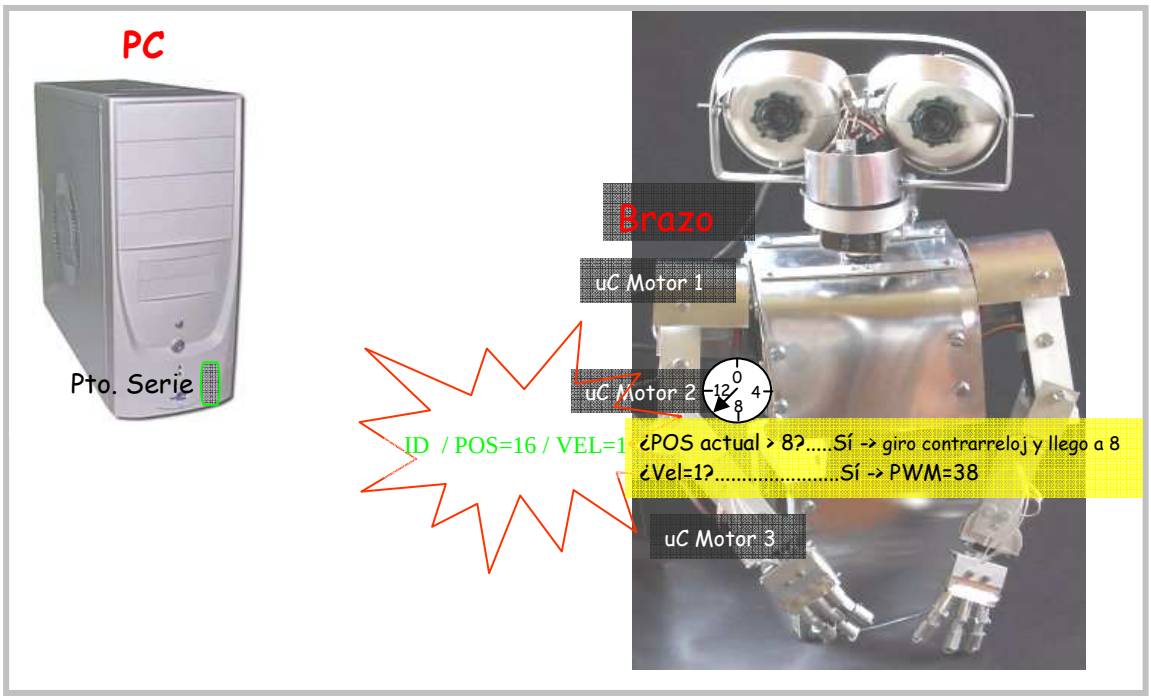
➢ ETAPA 3. PROCESAMIENTO Y CONTROL

La orden recibida indica al uC que el motor tendrá que moverse a una posición determinada, y con una determinada velocidad. Para ello, el PIC tendrá que decidir en qué dirección hacer girar su motor (hacia la derecha o hacia la izquierda) y qué potencia aplicar para conseguir la velocidad requerida.

La dirección de giro (derecha o izquierda) se consigue enviando por el PIC una señal TTL (un 1 o un 0) por una patita del uC que está conectada a la etapa de potencia del motor que controla. Hacia la derecha el motor rota en el sentido de la agujas del reloj, y hacia la izquierda en sentido contrario.

La potencia la genera lanzando una señal PWM por una de sus patitas. Según el ancho de pulso del PWM, el par al que girará el motor será mayor o menor y también su velocidad.

De esta forma, el motor empezará a girar en la dirección de la posición final, y con la velocidad indicada, hasta llegar al punto de destino. En ese momento frenará y mantendrá el motor fijo en la posición final. El PIC está constantemente controlando la posición del servo, a través de la lectura reiterada del valor del potenciómetro. Cada PIC está conectado a un potenciómetro, y realiza la conversión analógico/digital, para traducir un valor de tensión eléctrica a una posición medible determinada (un número entre 0 y 255).



Para comprender mejor todo el proceso, veamos en detalle un ejemplo práctico en el cual el PC ordena al motor 2 que se mueva a la posición 16, a velocidad rápida. Estos son los pasos:

- ETAPA 1. ENVÍO

a) El PC construye un byte con la siguiente información:
- Identificador del motor: la información va dirigida al motor 2 (en binario 010)
- Posición a situar el motor: posición 16 (1000 en binario).
- Velocidad del desplazamiento: velocidad rápida (1 en binario)

Al final, el byte completo sería: 010 1000 1

b) El PC envía el byte por el puerto serie, que está conectado al PIC 1.

- ETAPA 2. RECEPCIÓN O REENVÍO

c) El PIC 1 recibe el byte. Mira la dirección, y como no es para él reenvía el byte al siguiente PIC (el PIC 2).

d) El PIC 2 recibe el byte. Mira la dirección, y como ve que es para él, pasa a procesar la información.

- ETAPA 3. PROCESAMIENTO Y CONTROL

e) El PIC 2 lee la información de posición a situar el motor. Como tiene que ir a la posición 8 y está en la posición 10, tendrá que girar hacia la izquierda. Para ello pone a 1 la señal TTL dirigida a la etapa de potencia.

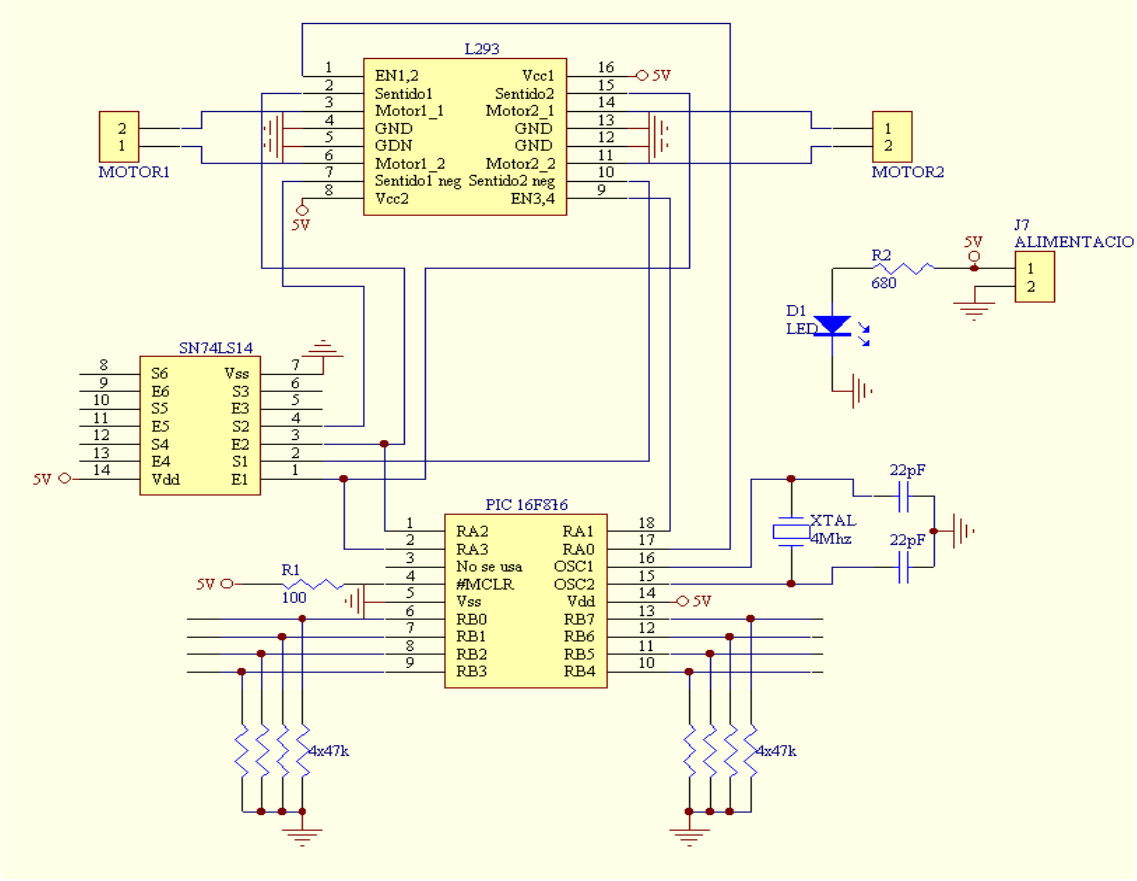
f) Constantemente y a intervalos regulares el PIC 2 va leyendo la posición en la que está, información que le proporciona el potenciómetro de su motor. De esta forma podrá saber la velocidad a la que va, y corregir la potencia de la señal PWM para variar la velocidad actual a la velocidad recibida.

g) Cuando llega a la posición destino, el PIC 2 frena, aplicando brevemente una señal PWM en el sentido opuesto al de giro actual. A continuación, deja de aplicar la señal

PWM, y controla constantemente el motor para que permanezca anclado en su posición en todo momento.

5.2.2. DISEÑO ELECTRÓNICO

En el siguiente diagrama podemos ver una simplificación del diseño electrónico del brazo. Es una simplificación ya que sólo aparece un microcontrolador (uC) PIC, en vez de los 5 PICs utilizados. Pero es suficiente para explicar el funcionamiento, ya que bastaría con multiplicar el esquema por 5, para hacerse una idea del conjunto. Faltaría en el dibujo el max233, empleado en la comunicación del uC con el puerto serie del ordenador.



Se distinguen claramente 3 elementos principales: el PIC16F876, el L293, y el SN74LS14. Los dos últimos conforman la etapa de potencia. Procedamos a analizar cada uno de estos elementos, y su funcionalidad:

1) <u>Microcontrolador PIC16F876</u>

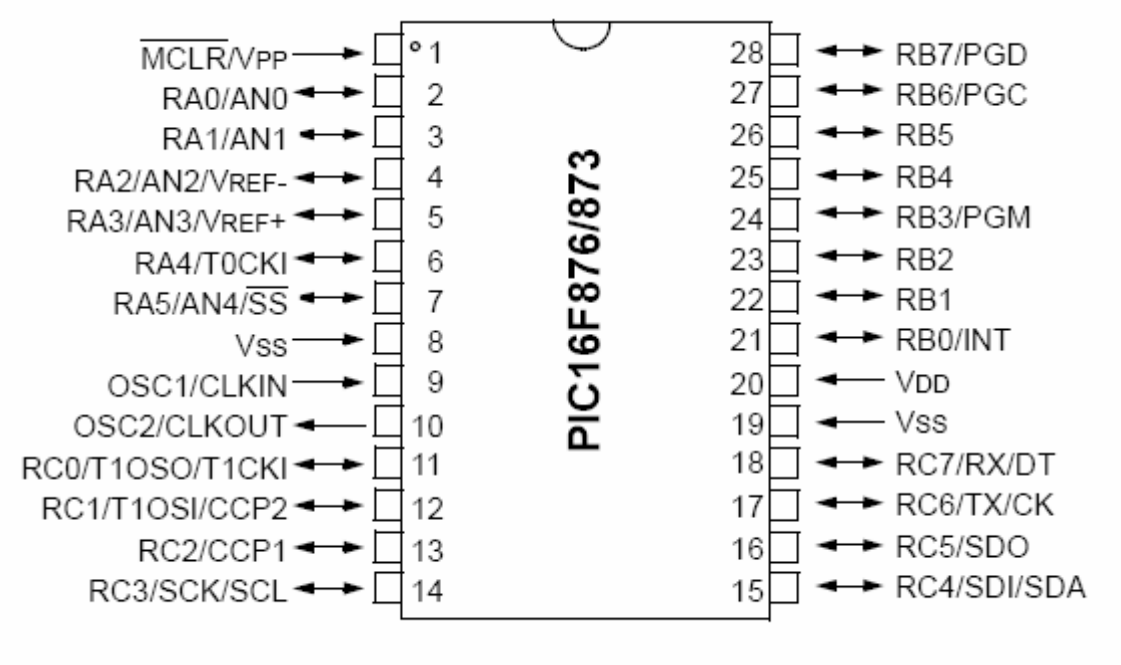
El uC empleado es el PIC16F876. Constituye el elemento principal, ya que contiene toda la inteligencia para el control continuo de la posición y velocidad del motor. Se ha seleccionado este modelo, y no otro de gama inferior, porque dispone de 3 funcionalidades indispensables para este proyecto:

- UART (comunicaciones con puerto serie)

- PWM (generación de pulsos para controlar la velocidad del motor)

- Conversor analógico/digital (para recoger los datos del sensor de posición)

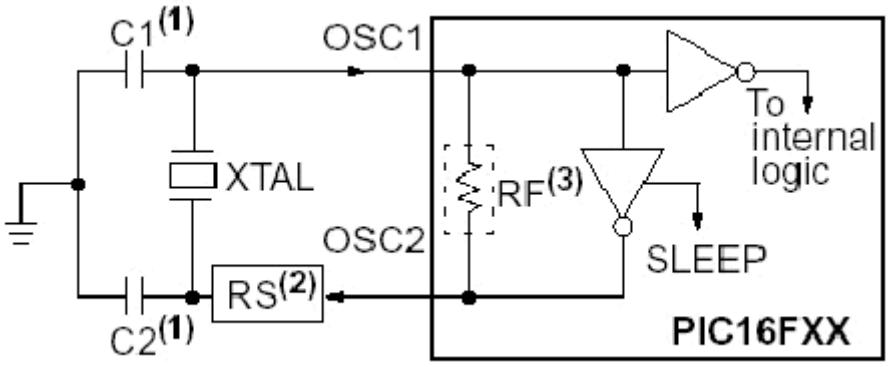
Veamos en detalle el patillaje del microcontrolador:



- MCLR: Es la patilla de RESET. Mientras esta patilla esté conectada a 5V (hay un '1' lógico en la patilla), el PIC funcionará normalmente. Cuando se ponga a 0V, el PIC se reseteará, es decir, el programa comienza desde cero.

- AN0: Entrada por la cual se recibe la señal del sensor de posición (el potenciómetro insertado en el servo). El PIC recoge esta señal analógica y la convierte a digital para poder operar con ella. De esta forma, controlará la posición y velocidad del motor.

- CLKIN: Entrada del reloj. Para que el PIC pueda funcionar, necesita de un circuito de reloj que le marque los tiempos. El fabricante especifica cómo deber ser este circuito. Consta básicamente de un cristal de cuarzo, que es el que produce la oscilación. Se ha usado un reloj de 4Mhz. El datasheet del PIC16F876 indica que la configuración más apropiada para su montaje es:



- CLKOUT: Salida del reloj.
- RB0-RB7: Puerto B del PIC, usado como salida. Durante la fase de programación del PIC y también en las pruebas, se ha conectado el puerto a 8 leds para hacer el debugging necesario.
- Vdd: Alimentación. A esta patilla van los 5V para que funcione.
- Vss: Tierra.
- RX: Patilla de salida de datos del PIC al puerto serie.
- TX: Entrada de datos desde el puerto serie al PIC.

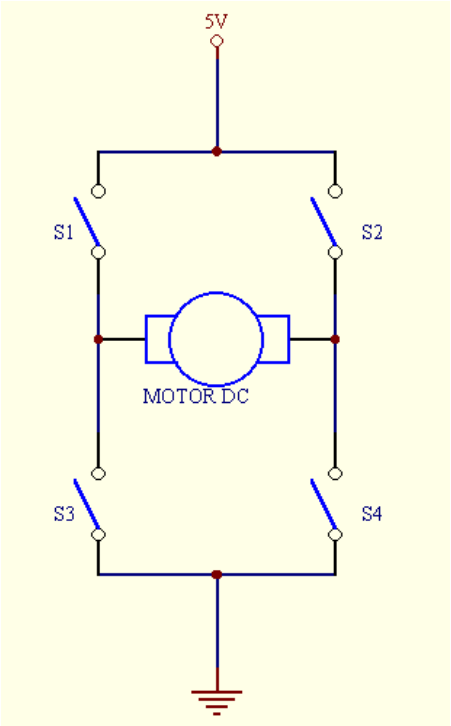
2) La etapa de potencia

Para poder gobernar motores con un PIC, debemos crear una etapa intermedia, entre la electrónica de control, y los motores. Esta etapa se denomina "etapa de potencia". Vemos su funcionamiento:

- **Puente H**

Un motor nunca se puede conectar directamente a un microcontrolador, porque absorbe mucha corriente (unos 500mA, depende del tamaño del motor). La patilla de un PIC puede dar en torno a los 25mA: mucho menos que lo que pide un motor. Si conectáramos el uC directamente al motor, se quemaría. Por eso utilizamos un puente H.

Observemos la siguiente figura:



Se trata de un motor de corriente continua, flanqueado por cuatro interruptores, que a su vez están conectados a 5V por un lado, y a tierra por el otro. Si cerramos los interruptores S1 y S4, dejando S2 y S3 abiertos, en el lado izquierdo el motor estará conectado a 5V, y en el derecho a tierra (0V). La diferencia de potencial entre los bornes del motor es de 5V (5-0=5), con lo que el motor se pone a girar en un sentido.

Si abrimos S1 y S4, cerrando S2 y S3 sucede lo contrario: el lado derecho está a 5V y el izquierdo a 0V. Ahora la diferencia de potencial es -5V (0-5=-5). Esto quiere decir que el motor girará el sentido contrario a como giraba antes.

El esquema anterior es muy parecido en el caso de los transistores. Pensemos en un transistor como un elemento de tres patillas. Dos de ellas funcionan exactamente como un interruptor, si está cerrado pasa corriente, y si no, no pasa. La tercera patilla es la que gobierna al interruptor: si en esa patilla hay un '1' lógico, se cierra el interruptor que forman las otras dos. Si se pone un '0' lógico, se abre el interruptor.

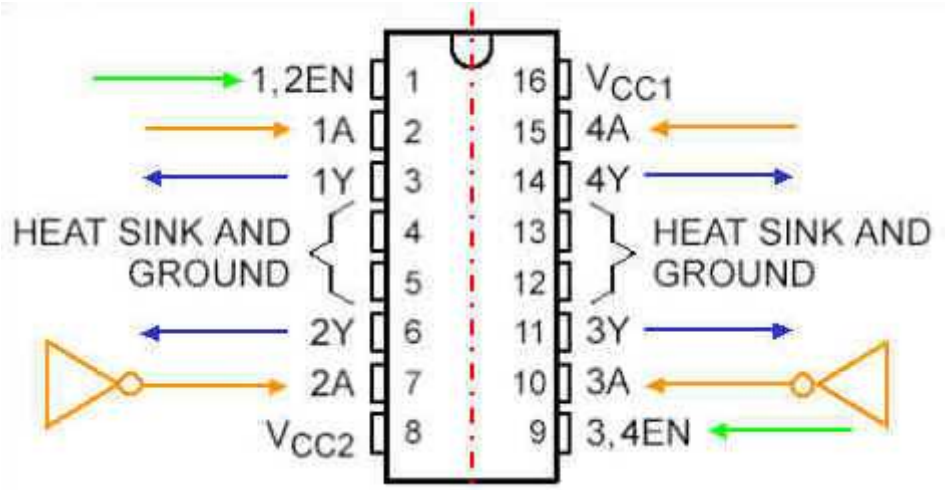
Entonces, tendremos 4 transistores flanqueando a un motor. Las patillas que controlan cada uno de los transistores, van conectadas al PIC, de esta forma se puede controlar por

software cuándo se abren y cierran, es decir cuándo si giramos el motor en un sentido o en otro. A esta estructura se le llama puente H. Como se puede ver en la figura anterior, los interruptores/transistores, flanquean al motor formando una H.

Los puentes H empleados en este proyecto vienen en un circuito integrado, que se llama L293D.

Este chip lleva integrados dos puentes H (uno para cada motor). Por tanto, para un brazo de 5 motores, han sido necesarios 3 puentes H.

Veamos cómo es el patillaje de estos chips, y para qué sirve cada cosa:



El chip tiene dos partes completamente independientes (en la figura, separadas por la línea roja punteada). Cada una de ellas, gobierna a un motor. Veamos el patillaje de la parte derecha:

- Patilla 1A: indicará el sentido de giro del motor. Si 1A=0, girará en un sentido (según como hayamos conectado el motor), y si 1A=1, girará en el sentido contrario.

- Patilla 2A: tiene que llegar justo la inversa de 1A. Esto se puede hacer por software, o por hardware. Es más seguro hacerlo por hardware, haciendo pasar a la señal 1A por un inversor.

- Patilla 1,2EN: es una entrada de habilitación del chip. Esto quiere decir que si 1,2EN es un '1', el chip funcionará como se ha indicado hasta ahora. Si 1,2EN fuese '0', el motor se para, independientemente del valor que tengan las patillas 1A y 2A (el chip está deshabilitado).

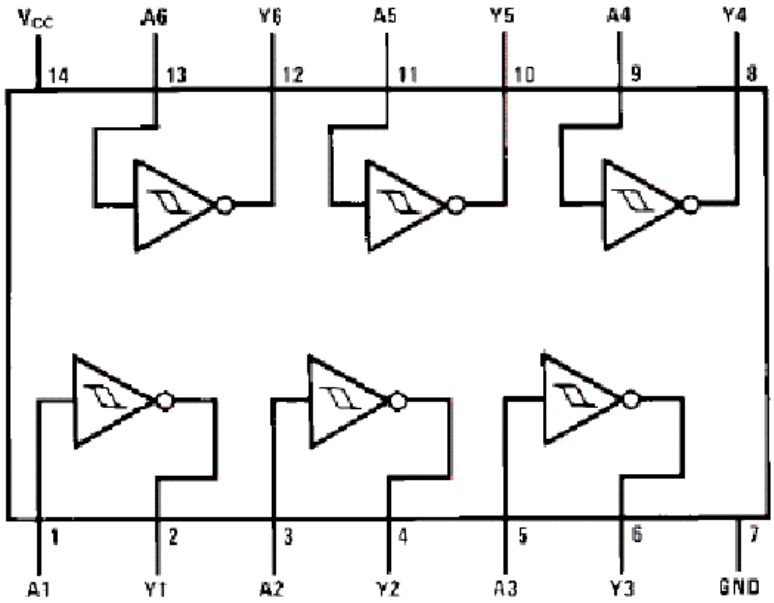
1,2EN	1A	2A	Motor
0	0	1	Parado
0	1	0	Parado
1	0	1	Gira en un sentido
1	1	0	Gira en el otro sentido

- GND: Tierra.
- Vcc1: Es la alimentación de la electrónica, como en el caso de los otros integrados. Se conecta a 5V.
- Vcc2: es la alimentación de los motores.
- Patillas azules: son las salidas hacia los motores. A esas patillas van conectados directamente los bornes del motor.

La otra parte del L293D funciona igual que la derecha.

- **Inversor**

El inversor que se ha empleado es un circuito integrado. Concretamente, se trata del 74LS14 (seis inversores, con trigger schmith). El integrado tiene el siguiente patillaje:

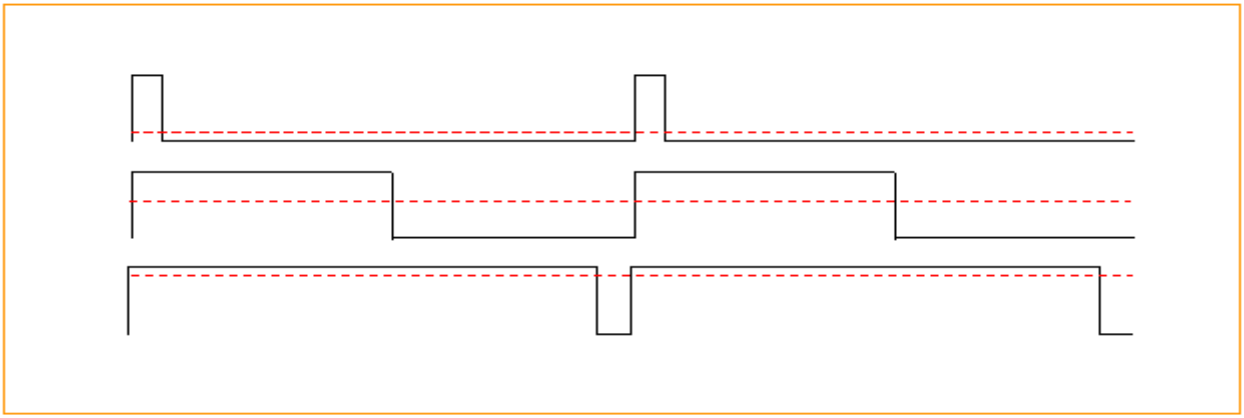


Su funcionamiento es muy sencillo: si por la patilla 1 introduzco un valor ('0' ó '1'), por la patilla 2 sale el contrario. La patilla 7 es de alimentación, y la 14 tierra.

- **PWM**

Con la técnica denominada PWM (Modulación del Ancho de Pulso) se consigue el control de la velocidad de los motores del brazo robótico.

Fijémonos en la siguiente figura:



En el primer caso, el de más arriba, se muestra el valor de la patilla 1,2EN. Cuando 1,2EN='1', el motor gira. Si 1,2EN='0', el motor está parado. En este primer caso, está encendido poco tiempo, y parado más tiempo. Si aumentamos la frecuencia con la que

cambiamos de encendido a apagado llega un momento en que conseguimos controlar la velocidad del motor.

En la primera línea de la figura, el motor está encendido un 10% del tiempo, con lo que el motor girará al 10%. En la segunda, el pulso activo del motor dura el 50%, luego iremos a la mitad de velocidad que el motor puede dar. En el tercer caso, el motor va al 90%.

5.2.3. ALGORITMOS DE POSICIONAMIENTO

La cinemática del brazo de un robot trata del estudio analítico del movimiento del brazo del robot con respecto de un sistema de referencia fijo como una función del tiempo, sin tener en cuenta las fuerzas que originan el movimiento. En Robótica, las especificaciones de movimiento se refieren al extremo del robot, o a un punto fijo de su herramienta, que se conoce como Tool Central Point (TCP) [BARR00]. En nuestro caso el TCP correspondería con el centro de la raqueta de ping-pong, ya que es el punto con el que debería golpearse la pelota.

Clásicamente se considera que hallar el modelo cinemático del robot es resolver su problema cinemático directo e inverso:

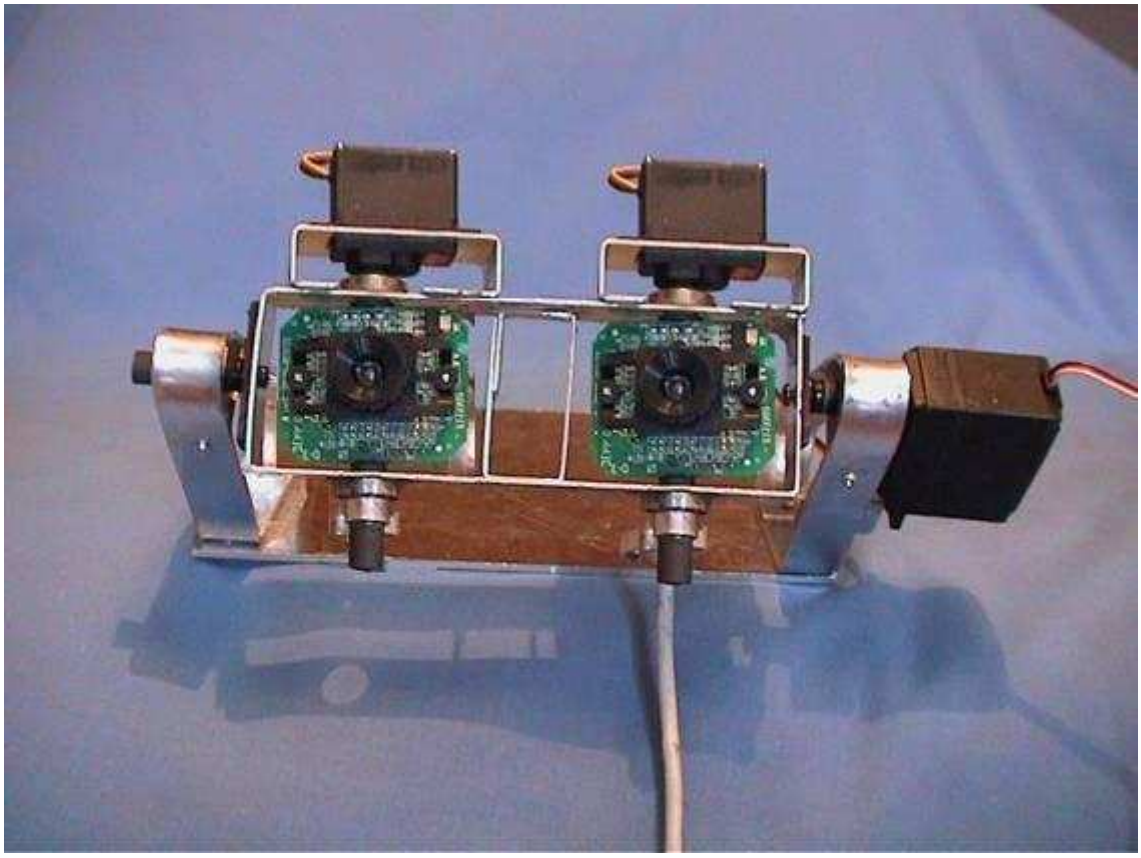
El problema cinemático directo trata de obtener la posición y orientación del extremo final del robot, a partir de las coordenadas articulares y de parámetros geométricos del brazo. Este problema siempre tiene solución y además es única. Esto significa que conociendo todas las posiciones en que están los servos del brazo sólo hay una posible posición de la raqueta en el espacio, y además es fácilmente obtenible. Para ello bastaría con aplicar el algoritmo de Denavit-Hartenberg. Es un método con el cual conseguiríamos la solución únicamente realizando operaciones de multiplicación de matrices, lo cual hubiera sido fácilmente implementable.

Mas difícil hubiera sido resolver el problema cinemática inverso. Trata de obtener las coordenadas articulares que se corresponden con una orientación y una posición dados del extremo final del robot. Este problema puede no tener solución (posiciones no alcanzables por el robot, por ejemplo), o en caso de tenerla, puede que ésta no sea única.

Ninguno de los dos problemas ha sido solucionado o implementado para este proyecto. Lo que sí se ha hecho es resolver el control de la posición de los servos desde un PC, sin tener en cuenta el TCP de la raqueta. Se ha programado una clase Java para el envío de bytes de control por el puerto serie.

Como el JDK no viene con una API de comunicaciones serie para Windows, la solución más rápida ha sido utilizar una librería nativa (rxtx-2.1-7r2). RXTX es una librería bajo licencia gnu LGPL, que provee al Java Development Toolkit (JDK) de comunicaciones con el puerto serie.

6. SISTEMA DE VISIÓN ARTIFICIAL



Igual que antes, podemos seguir comparando el robot con un humano. Y, haciendo una analogía, se podría hablar de la ‘cabeza’ del robot en la cual se encuentran los ‘ojos’.

En un primer momento comenzó a construirse físicamente la cabeza y la estructura que da movimiento a los ojos. Poco después, y en paralelo, se empezó a trabajar en los algoritmos de Visión Artificial.

6.1. DISEÑO MECÁNICO

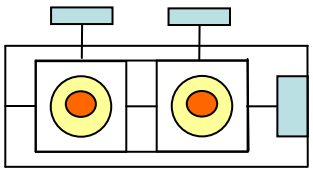
Este apartado trata acerca de la construcción física de una cabeza robótica en la que se insertan dos webcams, a modo de ojos.

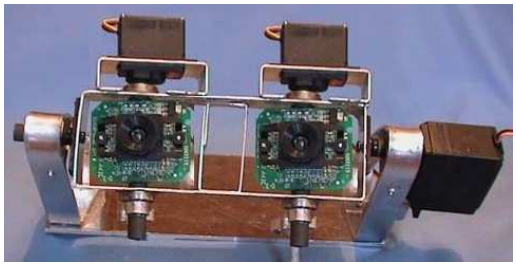
En los seres humanos, cada ojo tiene 2 grados de libertad:
- Movimiento vertical (arriba-abajo)
- Movimiento horizontal (derecha-izquierda)

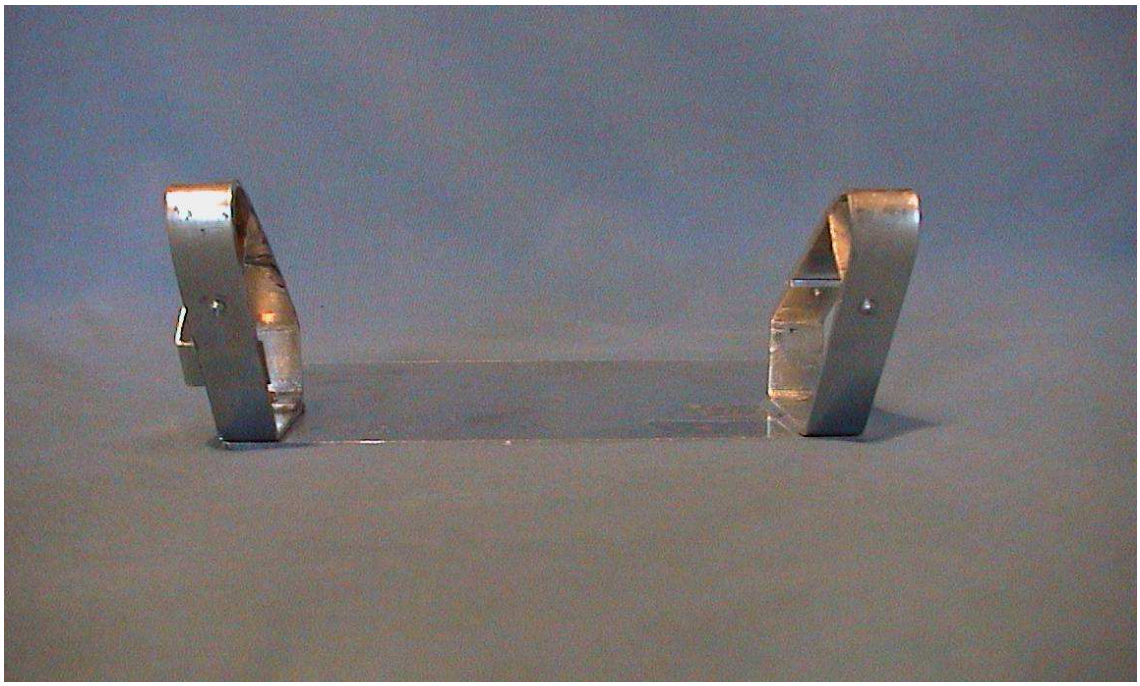
Pero además se produce una peculiaridad: el movimiento vertical de un ojo no es independiente del movimiento vertical del otro ojo. De hecho, al seguir un objeto con la mirada siempre mantenemos ambos ojos a la misma altura, de forma que ambos movimientos verticales son idénticos.

Traduciendo esto a las especificaciones del robot, para conseguir los movimientos horizontales se usan ejes verticales, y para los movimientos verticales se usan ejes horizontales. Para simular estos movimientos, podría haberse hecho con dos ejes para cada ojo (uno vertical y otro horizontal) y cuatro motores, pero si se aprovecha la peculiaridad antes citada, bastaría con dos ejes verticales (uno para cada ojo), y uno horizontal que controle el movimiento 'arriba-abajo' de ambos ojos al mismo tiempo. Esto, además de simplificar el diseño físico (3 servos y 3 ejes en vez de 4 servos y 4 ejes) tiene la ventaja de ahorrar el difícil control necesario para sincronizar adecuadamente ambos movimientos verticales.

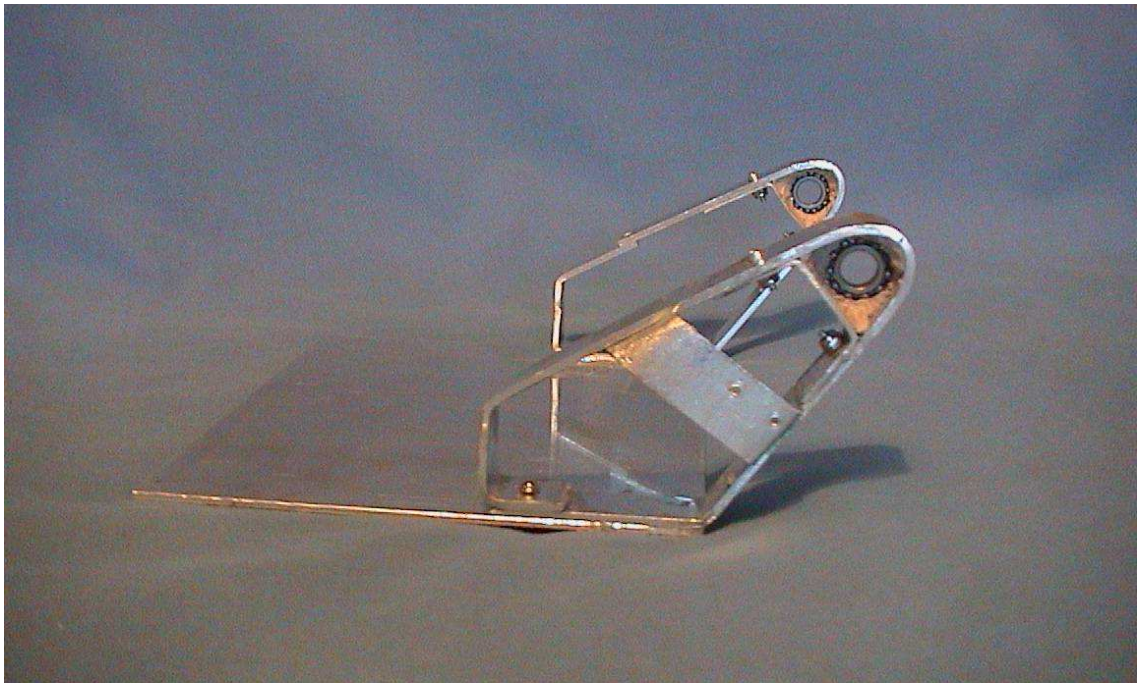
En el siguiente dibujo podemos ver un gráfico simplificado en el que aparecen los motores (en azul), los ejes y las estructuras de sujeción que conforman la cabeza (en negro), y los 2 ojos (amarillo y naranja)







Vista frontal de la estructura de soporte, sin ejes y sin cámaras

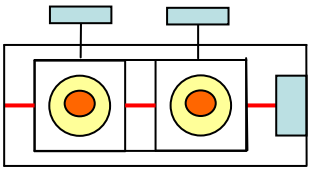


Vista lateral de la estructura de soporte, sin ejes y sin cámaras

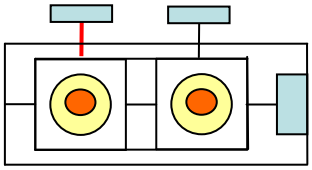
El material empleado es principalmente el aluminio, a excepción de los ejes (tubos de color negro) que son del mismo material que los huesos del brazo (fibra de carbono).

Cada ojo tiene dos grados de libertad, puesto que cada eje permite un GDL y los dos ojos comparten el eje horizontal.

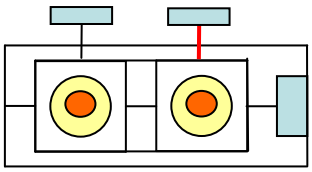
- **Eje 1**: movimientos verticales (arriba-abajo)



- **Eje 2**: movimientos horizontales del ojo derecho (derecha-izquierda)



- **Eje 3**: movimientos horizontales del ojo izquierdo (derecha-izquierda)



6.2. CONTROL DE MOVIMIENTOS

El control del movimiento de los ojos se realiza con un sistema de microcontroladores y sensores de posición conectados a los ejes de la cabeza.

El esquema de control de bajo nivel es el mismo que se ha utilizado para el brazo, ya que el problema a solucionar es el mismo:

- Girar uno ejes a una velocidad determinada
- Tras el giro, posicionar el eje en una posición final

Por tanto, de nuevo se repiten las 3 etapas de control de bajo nivel. Sin embargo, lo que sí varía es el control de alto nivel. Consiste en la lógica que, tomando como input la posición de la pelota de ping-pong, lanza las órdenes a los microcontroladores que mueven los ojos. Esta parte no ha llegado a implementarse para este proyecto y por eso en la descripción del funcionamiento general sólo se hará referencia al bajo nivel.

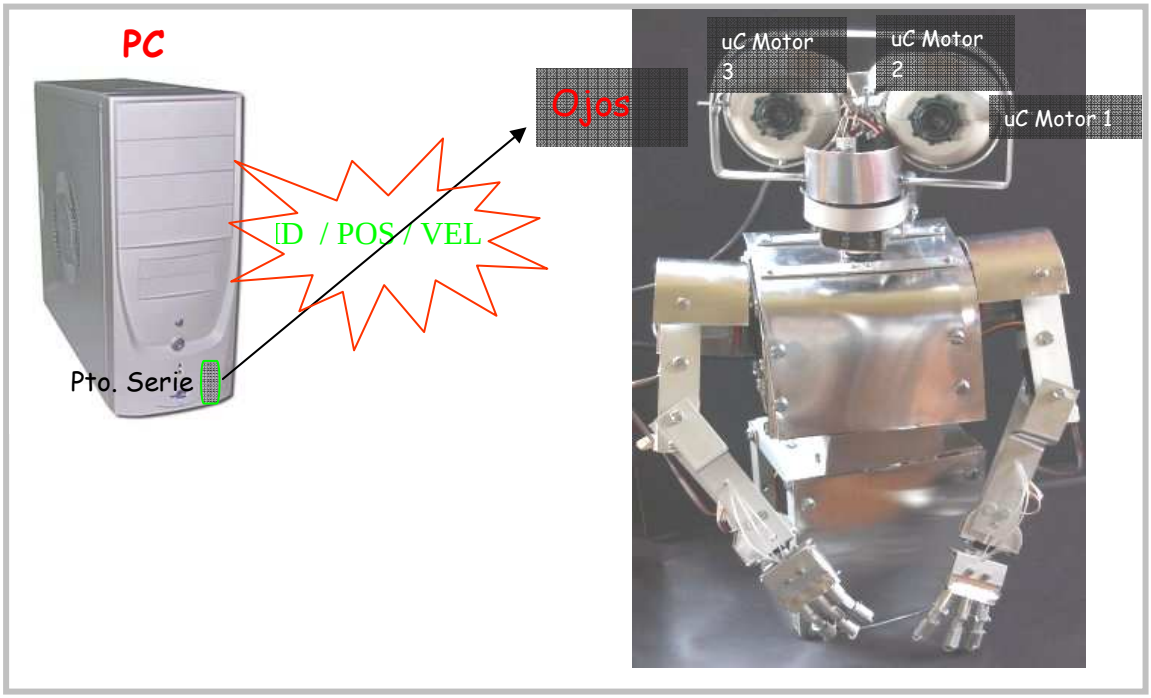
6.2.1. FUNCIONAMIENTO GENERAL

¿Cuáles son los pasos de bajo nivel que se dan internamente en el sistema para que los ojos lleguen a moverse?

➢ ETAPA 1. ENVÍO

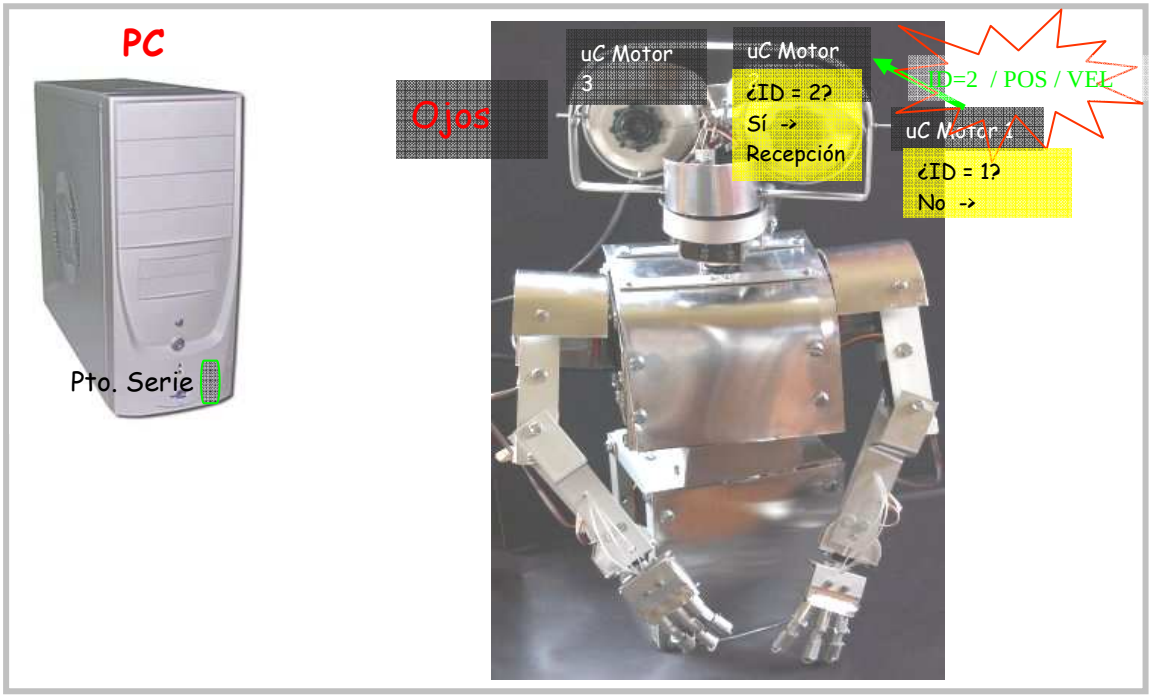
Un programa Java envía las órdenes a los ojos, a través del puerto serie. Para cada movimiento envía 3 órdenes (3 bytes). En cada byte van 3 informaciones:

- ID: El identificador del motor al cual se lanza la orden (un nº del 1 al 3, ya que hay 3 motores). El motor 1 controla el eje vertical y los motores 2 y 3 controlan los ejes verticales del ojo izquierdo y derecho, respectivamente.
- POS: La posición en la que deberá situarse el motor (16 posiciones diferentes, 16 ángulos).
- VEL: La velocidad a la que el ojo tendrá que moverse para seguir la pelota. Solamente se han implementado 2 velocidades (0=lento, 1=rápido).



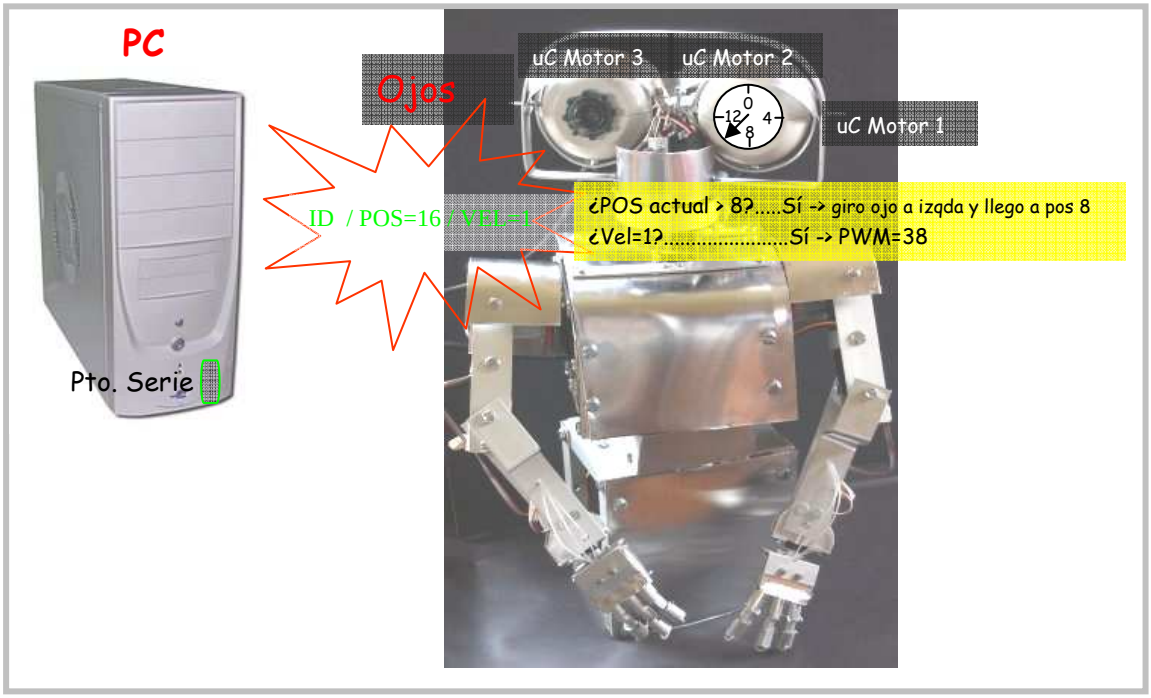
- ETAPA 2. RECEPCIÓN O REENVÍO

Los 3 bytes enviados los recibe un microcontrolador (concretamente el microcontrolador encargado del motor 1 del ojo y que controla el eje horizontal). El programa grabado en el uC, cada vez que reciba un byte, mirará el identificador, y si el mensaje no es para él lo reenviará al siguiente PIC (y así sucesivamente hasta que el PIC destinatario se queda con su mensaje para procesar la orden).



- ETAPA 3. PROCESAMIENTO Y CONTROL

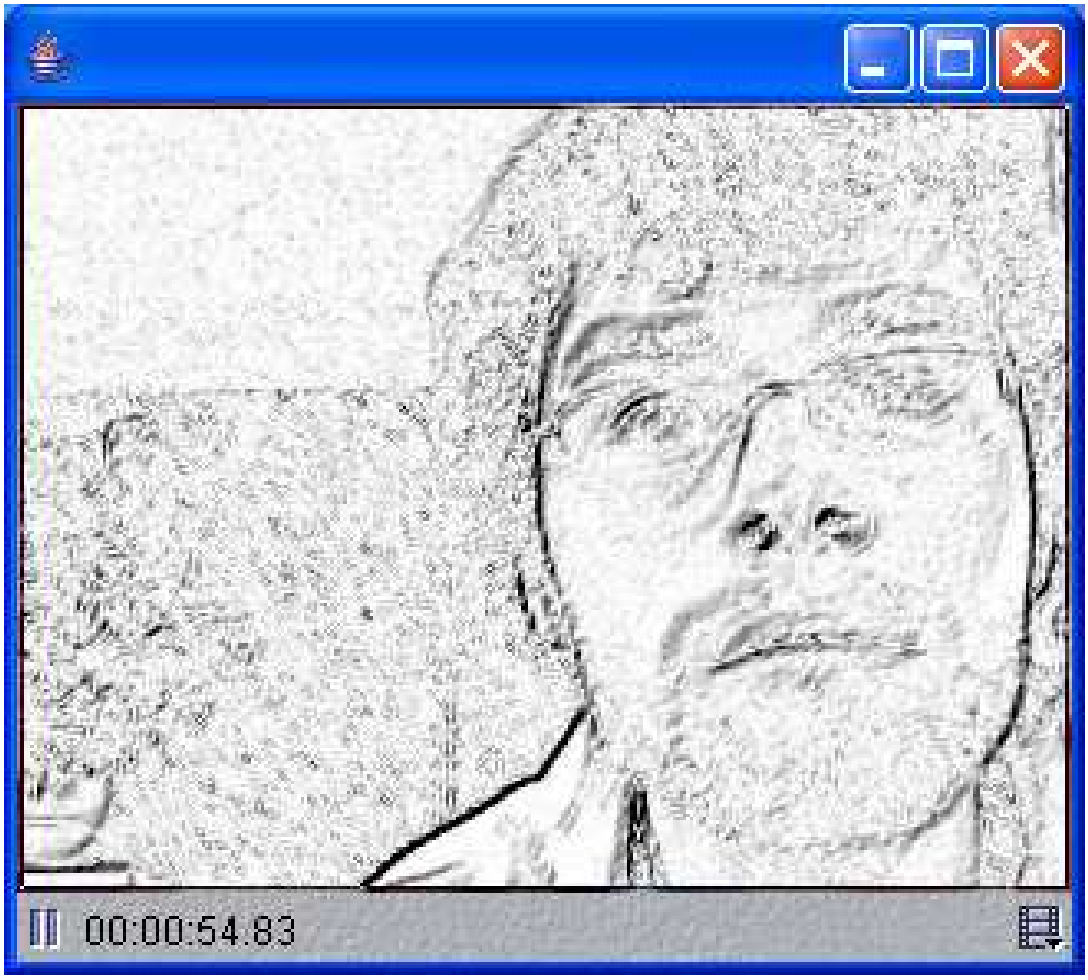
La orden recibida indica al uC que el ojo tendrá que moverse a una posición determinada, y con una determinada velocidad. Para ello, el PIC tendrá que decidir en qué dirección hacer girar su motor (hacia la derecha o hacia la izquierda) y qué potencia aplicar para conseguir la velocidad requerida.


PC
Pto. Serie
Ojos
uC Motor 3
uC Motor 2
uC Motor 1
ID / POS=16 / VEL=1
¿POS actual > 8?.....Sí -> giro ojo a izqda y llego a pos 8
¿Vel=1?.....Sí -> PWM=38

6.2.2. DISEÑO ELECTRÓNICO

El diseño electrónico de los ojos es igual que el realizado para el brazo. Unos microcontroladores controlan los motores utilizando una etapa de potencia. Lo único que cambia es el número de PICs: solamente son necesarios 3 uC, ya que sólo hay 3 motores a controlar.

6.2.3. PROCESAMIENTO DE IMÁGENES



Lo que 've' un ojo del robot tras aplicar un filtro de Sobel

Para el procesamiento de imágenes en tiempo real se ha utilizado la JMF. Java Media Framework es una API (Application Programming Interface) que proporciona herramientas para la captura, procesamiento y almacenamiento de datos multimedia. Incluso permite su transmisión y recepción a través de Internet.

Más concretamente permite:

– Reproducir ficheros multimedia en applets y aplicaciones java.

– Reproducir flujos multimedia recibidos en tiempo real a través de la red.

– Capturar audio y vídeo desde un micrófono o una cámara de vídeo.

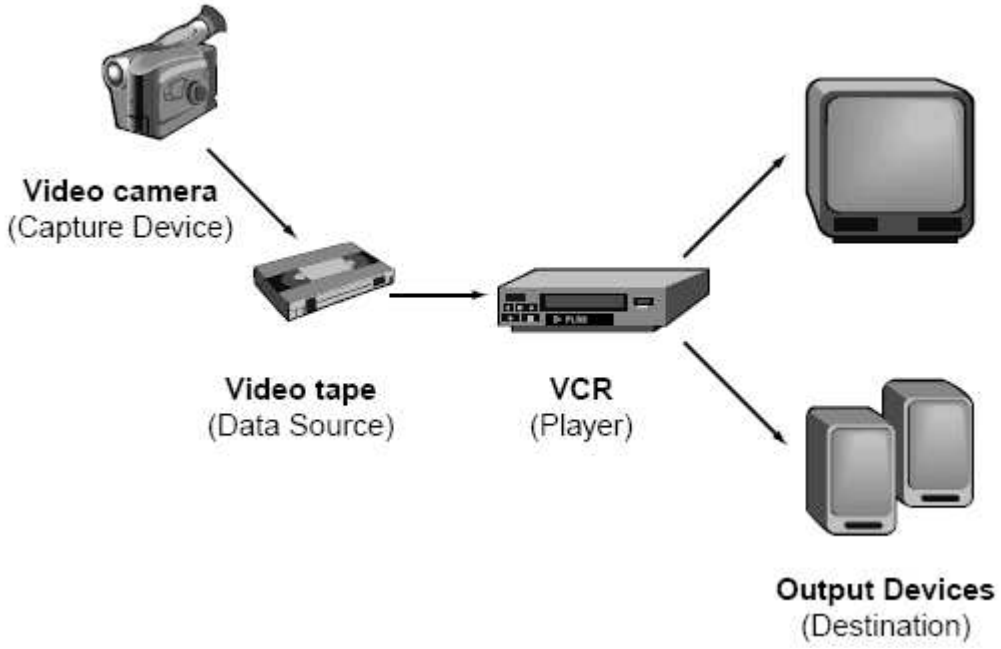
– Procesar flujos de datos multimedia.

– Implementar soluciones a medida, de forma que los desarrolladores basándose en la API, puedan integrar nuevas funcionalidades al framework existente.

– Acceder a los datos multimedia “en bruto”, en su formatos más básicos.

La documentación de JMF usa un modelo básico, que explica de forma resumida la arquitectura de alto nivel de la API. En él se pueden apreciar tres clases fundamentales:

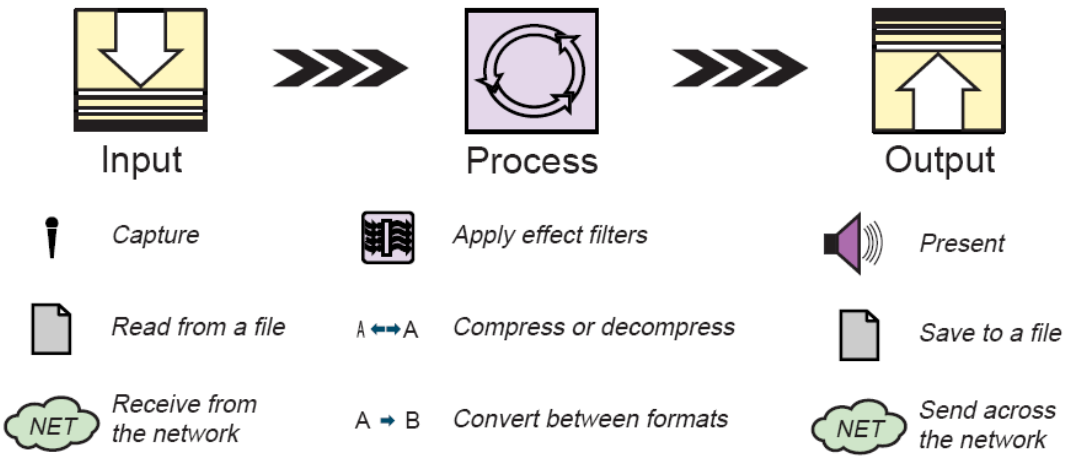
- CaptureDevice.class
- DataSource.class
- Player.class



Modelo JMF de captura, procesado y presentación multimedia

JMF es mucho más complejo que todo eso. Pero podríamos decir a grandes rasgos que las clases Java programadas para este proyecto utilizan esas 3 clases de la API, extendiendo sus funcionalidades.

En el siguiente gráfico podemos ver el modelo de procesamiento de datos multimedia de la JMF:



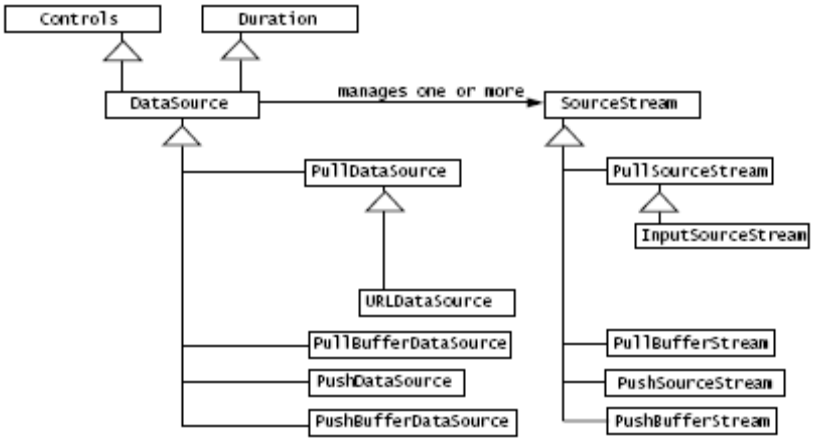
Modelo de procesamiento JMF

Con el mismo esquema se puede también explicar el funcionamiento básico del procesamiento de imágenes llevado a cabo por el sistema de Visión Artificial del robot. Veamos en detalle los 3 elementos:

- 1. Input: En este proyecto hay 2 inputs. Son las 2 web-cams, que funcionan como ojos del robot. Se capturan por tanto las imágenes en tiempo real, no se leen desde un archivo ni se reciben por la red.

  Lo primero que hace el programa es buscar en el sistema la cámara logitech (el ojo), que es un dispositivo de captura conectado al PC al que Windows asigna el identificador "vfw:Microsoft WDM Image Capture (Win32):0". El código es el siguiente:

```
CaptureDeviceInfo dev = CaptureDeviceManager.getDevice("vfw:Microsoft WDM Image Capture (Win32):0");
MediaLocator ml = dev.getLocator();
```

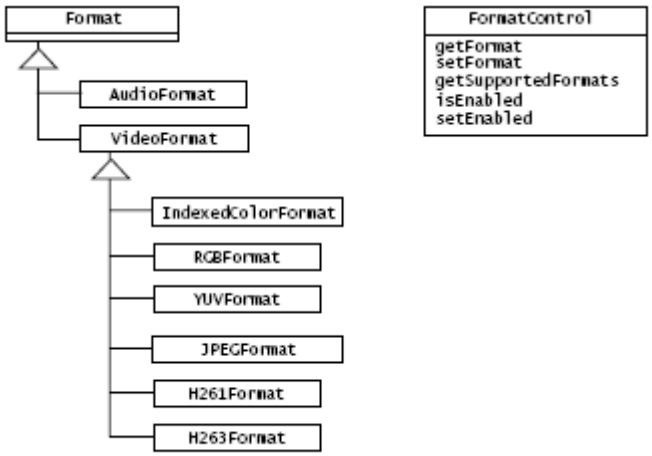


Modelo de datos de JMF

JMF admite muchos formatos de vídeo. En todos ellos tenemos por un lado una pista de vídeo, y por otro una pista de audio. Nos interesa obtener sólo la de vídeo, para analizar la imagen.

Format	Content Type	Quality	CPU Requirements	Bandwidth Requirements
Cinepak	AVI QuickTime	Medium	Low	High
MPEG-1	MPEG	High	High	High
H.261	AVI RTP	Low	Medium	Medium
H.263	QuickTime AVI RTP	Medium	Medium	Low
JPEG	QuickTime AVI RTP	High	High	High
Indeo	QuickTime AVI	Medium	Medium	Medium

Formatos de vídeo admitidos por JMF

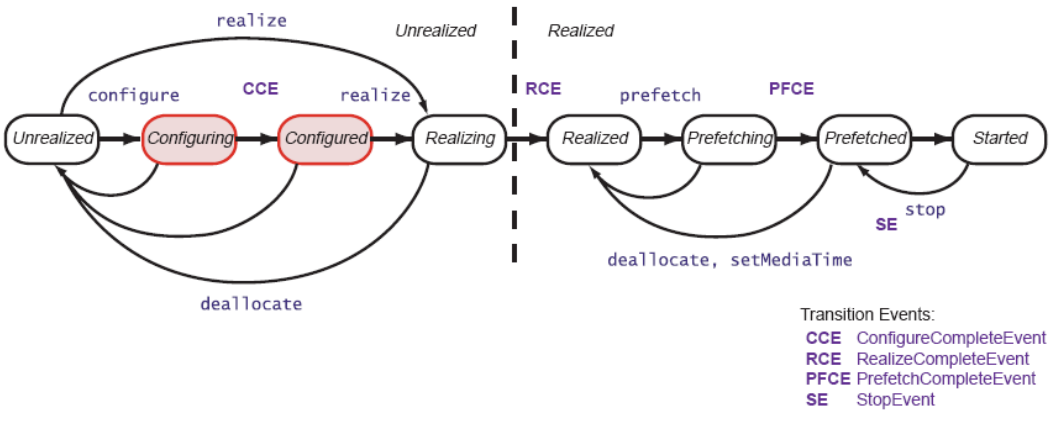


Clase Format, de JMF

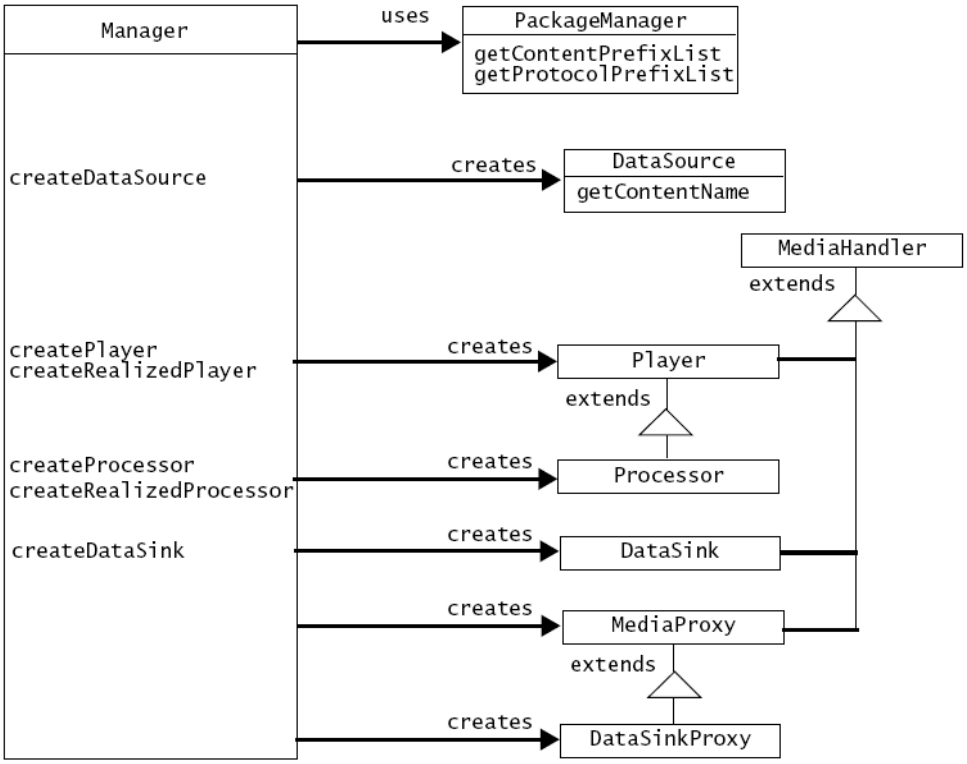
Concretamente el formato del vídeo recibido por la cámara Logitech es el YUV.

```
// Buscamos el control (track control) de la pista de video (video track).
TrackControl tc[] = p.getTrackControls();
TrackControl videoTrack = null;
for (int i = 0; i < tc.length; i++)
{
        if (tc[i].getFormat() instanceof VideoFormat)
        {
                videoTrack = tc[i];
                break;
        }
}
```

- 2. Procesamiento: Una vez caputurados los datos, están listos para procesar. Para ello se crea un procesador (Processor) que tendrá que pasar por una serie de estados hasta poder empezar a procesar.



Estados de un Processor en JMF



La clase Manager de JMF

```
//Creamos un Processor, utilizando la clase Manager
Processor p = Manager.createProcessor(ml);

//Ponemos el Processor en estado 'configured'. De esta forma se podrá usar como Player
p.configure();
videoTrack.setCodecChain(codecs);
cls = ClassLoader.getSystemClassLoader().loadClass(videorenderer);
videoTrack.setRenderer((Renderer) cls.newInstance());

// Ponemos el processor en estado 'Realize'
p.realize();

// Obtenemos uso exclusive de los recursos
p.prefetch();

//Arrancamos el processor
p.start();
```

Para extraer información de las imágenes, se ha creado una clase que hereda de la clase Effect de JMF. Un efecto (Effect) se diferencia de un Codec en que no modifica el formato de los datos multimedia, sino sólo su contenido. Y precisamente eso es lo que se hace en este proyceto: se analiza la imagen aplicando un filtro. Concretamente se han creado dos clases java: 'DetectaPelota.class' y 'DetectaBordes.class'. La primera lo que hace es detectar el centro de masas de las zonas de color naranja de la imagen, ya que la pelota de ping-pong con la que se han hecho las pruebas es de color naranja. Se supone que dicho centro de masas coincidirá con el centro de masas de la pelota de ping-pong.

```
public class DetectaPelota implements Effect
{
    public void accessFrame(Buffer frame)
    {
        int buf[] = (int[])frame.getData();
        int ofs = 0;
        int contaNar = 0;  //Contador de naranjas
        int contaNarWidthMax = 0;
        int contaNarHeightMax = 0;
        int maxNarHeight = 0;
        int maxNarWidth = 0;
        int h,w;



        for(h=1;h<height-1;h++)
        {
            contaNar=0;
            for(w=1;w<width-1;w++)
            {
                ofs = h*width+w;
                if(! esNaranja4(buf[ofs]))
                buf[ofs]=0; //a negro todo lo q no sea naranja
                else
                contaNar++;
            }
            if (contaNar>contaNarWidthMax)
            {
                contaNarWidthMax = contaNar;
                maxNarHeight = h;
            }
```

```
        }

        for(w=1;w<width-1;w++)
        {
            contaNar=0;
            for(h=1;h<height-1;h++)
            {
                    ofs = h*width+w;
                     if(! esNaranja4(buf[ofs]))
                            buf[ofs]=0; //a negro todo lo q no sea naranja
                    else
                            contaNar++;
            }
            if (contaNar>contaNarHeightMax)
            {
                contaNarHeightMax = contaNar;
                maxNarWidth = w;
            }
        }


        ofs = maxNarHeight*width+maxNarWidth;
        buf[ofs]=16711680; //Señalo con un puntito azul el centro de masas de la pelota naranja
        buf[ofs+1]=16711680;
        buf[ofs+2]=16711680;
    }
```

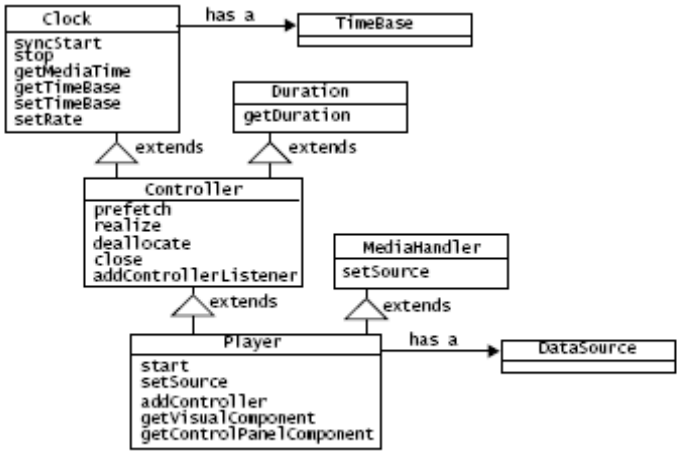
En DetectaBordes, se aplica un filtrado por gradientes de brillo de la imagen (filtro de Sobel). Es decir, que se resaltan en negro las zonas en las que haya un paso brusco de un brillo moderado a uno fuerte o viceversa. Esta clase se utiliza para detectar el perfil circular de la pelota de ping-pong.

```
public class DetectaBordes implements Effect
{
    private void accessFrame(Buffer frame)
    {
        //Le da el objeto original, es decir el puntero. Por eso si modificamos
        //buf es como si modificaramos el atributo 'data' del objeto 'frame'
        int buf[] = (int[])frame.getData();
        int ofs;
        int p;

        // Paso a B&N: Calculo la media de los 3 colores (r,g,b)
```

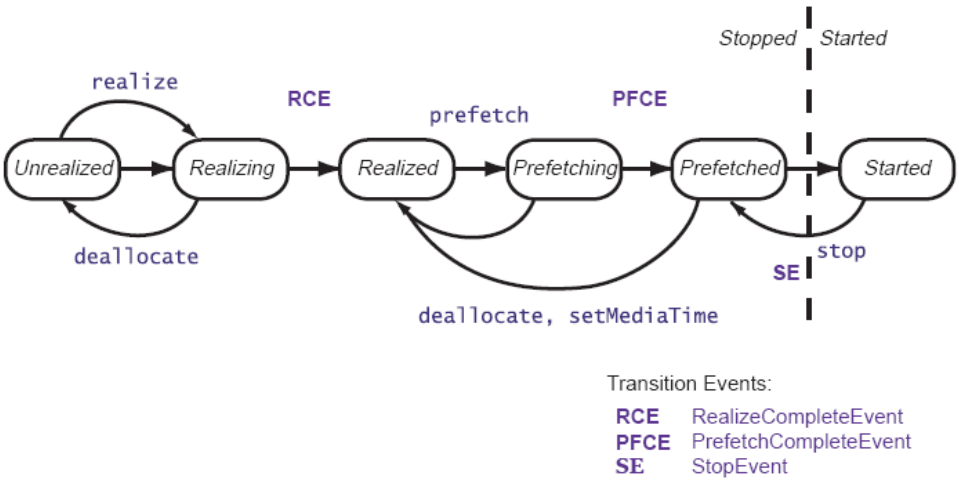
```
 for(int h=70;h<169;h++)
{
      for(int w=100;w<199;w++)
            {
                  ofs = h*width+w;
                  p = buf[ofs];
                   int br = (short)((p&0xff) + ((p>>8)&0xff) + ((p>>16)&0xff));
                   br /= 3;
                   if(br<80)
                  br = 80;

                   brillos[ofs]= (short)br;
                  buf[ofs] = buf[ofs] = (br<<16)+(br<<8)+br;
            }
}

    // Filtro de deteccion de bordes
    for(int h=70;h<170;h++) //esto para aligerar, solo analizo fovea (100 a 200)
    {
            for(int w=100;w<200;w++) //esto para aligerar, solo analizo fovea
            {
                  ofs = h*width+w; //height es 240 y width 320
                  int gradH,gradV,grad; //Gradiente horizontal, vertical y total

                  // Sobel
                 gradH = -1*brillos[ofs-1] + brillos[ofs+1]; if(gradH<0) gradH=-gradH;
                   gradV = -1*brillos[ofs-width] + brillos[ofs+width];
                  if(gradV<0) gradV=-gradV;
                  grad = gradH+gradV;


                  //Pasamos los gradientes a una escala de grises visualizable
                  //(cuanto mayor sea el gradiente, más oscuro se verá)
                   if(grad>255) grad=255;  // Max(grad,255)
                   gradientes[ofs] = (short)grad;
                  grad = 255-grad;
                  //pongo con mismo valor los 3 bytes rgb para q sea gris
                  buf[ofs] = (grad<<16)+(grad<<8)+grad ;
            }
      }
}
```

- 3. Output: tras haber procesado los datos, se muestra por pantalla la imagen procesada, en tiempo real. Esto no es necesario para el funcionamiento interno

del robot, pero sí para que el desarrollador pueda hacer posibles debugs y monitorice en todo momento el estado del sistema.



Clase Player de JMF



Estados de transición por los que pasa un Player

7. PLANIFICACIÓN

Se ha seguido la metodología de planificación estándar para cualquier proyecto de ingeniería ordenando las tareas, tiempos y recursos a emplear. La ejecución del proyectó se distribuyó en 4 fases:

- **Fase 1: Diseño y construcción**

El **diseño** incluye:

· Diseño mecánico de brazo y ojos: elección de engranajes, motores, ejes, materiales y sensores.

· Diseño electrónico de brazo y ojos, protocolos de comunicación, puertos, pruebas de programación de microcontroladores y de control de motores con pwm.

· Diseño informático: Elección de SO, lenguaje de programación, módulos, clases.

Estas fueron las decisiones de diseño adoptadas en esta fase:

Características del ordenador portátil:

Se hicieron algunas pruebas en diversos ordenadores disponibles, para ver en cuál de ellos el rendimiento de procesamiento de imágenes era mejor. Finalmente, el que mayor número de imágenes por segundo procesó fue el elegido:

Portátil Hp Pavilion, con

- S.O. Windows XP
- Procesador “AMD Turion 64x2 Mobile, 803 MHz”
- 992 MB Ram

Motores:

Se decidió emplear servomotores modificados, para que actuaran como motores normales de continua, con la ventaja de mantener su caja reductora y por consiguiente un gran par de salida. Su velocidad y dirección de giro se controla mediante PWM y puentes en H, con microcontroladores.

Microcontroladores:

- Microcontroladores PIC de la gama media (16F876). 1 Pic encargado de la comunicación con el puerto serie del PC y con los demás PICS. Cada uC controla un motor.
- Compilador de lenguaje ensamblador: MPASM
- Programa ProgPic2 y tarjeta de programación y experimentación Velleman K8048 (para introducir en los PIC el código máquina).

Comunicaciones:

Para la comunicación entre el PC y el robot se utiliza el puerto serie del ordenador, y los módulos UART de los PIC. Aunque en principio se hicieron pruebas con el puerto paralelo, finalemente se decidió simplificar el diseño adoptando el puerto serie. Se ha creado un protocolo simple de comunicaciones, por el cual cada PIC tiene un identificador, y solo atiende las órdenes que lleven dicho ID.

Procesamiento de imágenes:

Se emplea el Java Media FrameWork (JMF), API de java para datos multimedia, que permite la captura y procesamiento de imágenes. Se escogió Java y JMF porque era la forma más rápida de desarrollar un aplicación que pudiera tomar datos provenientes de una cámara de vídeo.

Cámaras de video:

2 web-cams "Logitech QuickCam". Recogen 15 imágenes por segundo, lo cual es suficiente para el seguimiento de una pelota en movimiento.

Tras el diseño, se pasó a la **construcción** del brazo y la cabeza del robot. El hecho de tener que fabricar las piezas de forma artesanal y hacer múltiples pruebas con diversos materiales buscando dichos materiales en infinidad de tiendas, contribuyó a alargar esta fase.

- **Fase 2: Comunicaciones y control**

Los lenguajes utilizados han sido: ensamblador (para los microcontroladores) y Java (para el Sistema de Visión Artificial y el control del puerto serie). En los

microcontroladores se hace el control de bajo nivel. Es decir que no se toman decisiones en cuanto a la posición a situar los motores, simplemente se ejecutan las órdenes de alto nivel que reciben del PC. Se han llevado a cabo las siguientes tareas:

- Programación de los microcontroladores para que generen PWMs de control de velocidad en los motores, en función de la posición muestreada de los sensores.
- Programación de los microcontroladores para que sean capaces de comunicarse con el puerto serie del ordenador.

- **Fase 3: Algoritmos de visión y movimiento**

Durante esta fase se implementaron en el PC los algoritmos de Visión Artificial, y los que tomaban las decisiones de alto nivel de movimiento:

- Programación del PC para enviar órdenes al robot desde el puerto serie
- Programación del PC para detectar una pelota de ping-pong desde una web-cam

- **Fase 4: Estrategias de juego**

Estrategias para que el robot pudiera jugar al ping-pong. Son reglas del tipo: "si la pelota me viene desde la posición (2,14,65), mueve el brazo a la posición (2,15,62) a velocidad rápida". Esta última fase del proyecto no ha llegado a comenzarse.

El proyecto se inició unos meses antes del curso 2005-2006. En principio se calculó una distribución de horas que permitiera finalizarlo durante ese año, aunque con mucha carga de horas:

Sin embargo, la realidad se impuso y finalmente las complicaciones surgidas hicieron que el proyecto se prolongara durante otro curso más (2006-2007). La distribución real de horas dedicadas a lo largo de los más de 2 años ha sido:

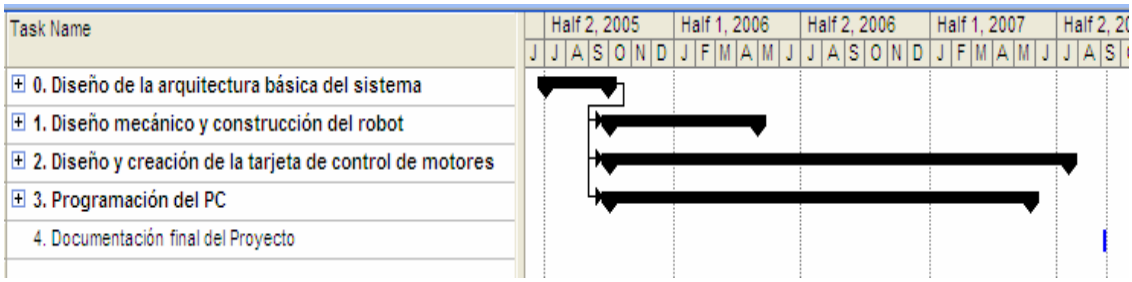
Es decir, en vez de las 650 horas planificadas inicialmente, se han tenido que dedicar 1.250 horas. No ha podido iniciarse la programación de las estrategas de juego ya que esta tarea tenía dependencias con la tarea de coordinación de los movimientos del brazo con el ojo. Ninguna de esas dos tareas ha llegado a iniciarse, pero el resto de tareas sí han sido finalizadas con éxito.

El resultado final han sido módulos independientes entre sí, que funcionan correctamente por separado sin llegar a coordinarse entre sí:

- Un brazo robótico que se mueve a cualquier posición si se le indican las coordenadas articulares. No es capaz de resolver el problema cinemática inverso.

- Un sistema de visión artificial que procesa correctamente imágenes, y que es capaz de mover los ojos a cualquier posición. No es capaz de seguir la pelota en movimiento.

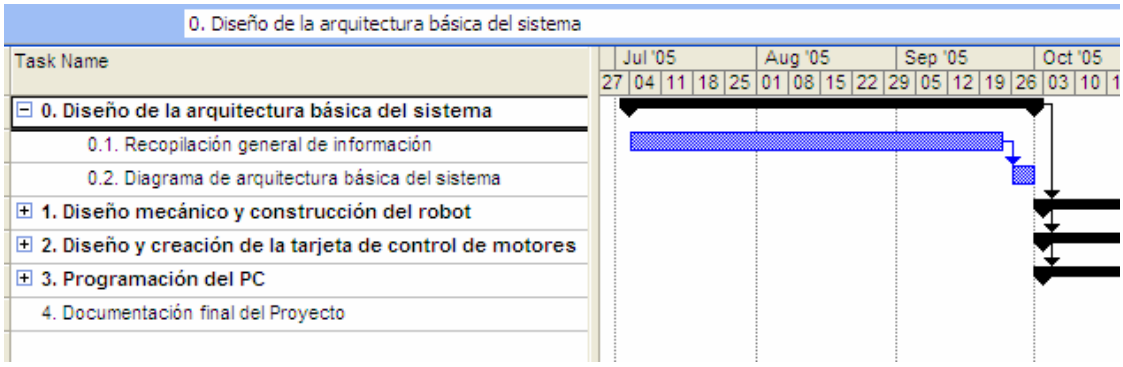
Veamos más en detalle un desglose de las tareas ejecutadas:



0. Diseño de la arquitectura básica del sistema

0.1. Recopilación general de información

0.2. Diagrama de arquitectura básica del sistema



1. Diseño mecánico y construcción del robot

1.1. Lectura de documentación

1.1.1. Lectura de libros de Anatomía

1.1.2. Lectura de documentación en Internet sobre otros proyectos de humanoides

1.2. Diseño del brazo robótico

1.2.1. Diseño de ejes y mecanismos

1.2.2. Cálculo de fuerzas

1.2.3. Elección de materiales de construcción (aluminio, madera, fibra de carbono)

1.2.4. Elección de motores

1.2.5. Elección de rodamientos

1.3. Diseño de la cabeza robótica

1.3.1. Diseño de ejes y mecanismos

1.3.2. Elección de materiales de construcción (aluminio, madera, fibra de carbono)

1.3.3. Elección de motores

1.3.4. Elección de rodamientos

1.4. Construcción del brazo y de la cabeza

1.4.1. Adquisición de materiales

1.4.1.1. Búsqueda y adquisición de materiales de prueba

1.4.1.2. Pruebas de resistencia de los materiales de prueba

1.4.1.3. Búsqueda y adquisición de materiales definitivos

1.4.2. Fabricación de componentes

1.4.2.1. Tallado manual de las piezas de madera

1.4.2.2. Recorte de placas y tiras de aluminio

1.4.2.3. Modelado de placas y tiras de aluminio

1.4.2.4. Adquisición de servomotores

1.4.2.5. Adquisición de rodamientos

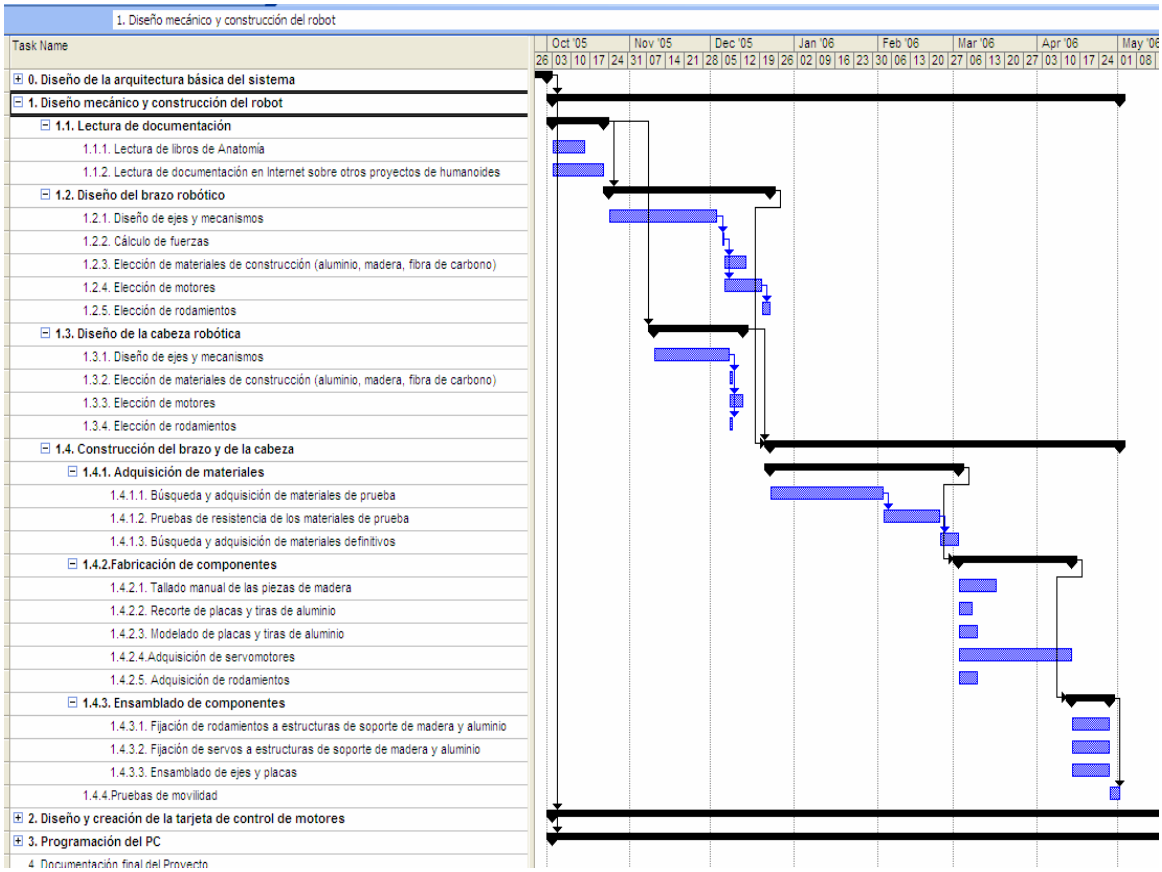
1.4.3. Ensamblado de componentes

1.4.3.1. Fijación de rodamientos a estructuras de soporte de madera y aluminio

1.4.3.2. Fijación de servos a estructuras de soporte de madera y aluminio

1.4.3.3. Ensamblado de ejes y placas

1.4.4.Pruebas de movilidad



2. Diseño y creación de la tarjeta de control de motores

2.1. Lectura de documentación

2.1.1. Lectura de manuales de programación de microcontroladores

2.1.2. Lectura de documentación básica sobre electrónica

2.1.3. Lectura del datasheet del PIC16F84

2.1.4. Lectura del datasheet del PIC16F876

2.2. Pruebas iniciales de programación

2.2.1. Adquisición de la tarjeta programadora de PICs 'K8048'

2.2.2. Creación de un circuito de pruebas con un PIC16F84

2.2.3. Programación del PIC16F84A para que ilumine un led conectado al PORTB

2.2.4. Programación del PIC16F84A para que se comunique con el puerto paralelo del PC

2.2.5. Programación del PIC16F84A para que envíe una señal PWM a un servomotor

2.2.6. Pruebas de comunicación entre 2 PICs, a través de la UART

2.3 Diseño de la tarjeta de control de motores

2.3.1. Elección del modelo final de microcontrolador (PIC16F876)

2.3.2. Elección del protocolo de comunicación con el PC (Puerto serie)

2.3.3. Diseño de la alimentación de la tarjeta

2.3.4. Diseño del circuito de PICs

2.3.5. Diseño del circuito de la etapa de control

2.3.6. Diseño del circuito de recepción de datos serie enviados por el PC

2.4. Implantación del diseño en una protoboard

2.5. Programación de los PIC

2.5.1. Control de movimientos

2.5.1.2. Control de posición

2.5.1.2.1. Programación de la lectura y conversión A/D de los datos del sensor de posición

2.5.1.2.2. Programación del envío de señales PWM para mover el motor

2.5.1.2.3. Programación del algoritmo de frenado del motor

2.5.1.3. Control de velocidad

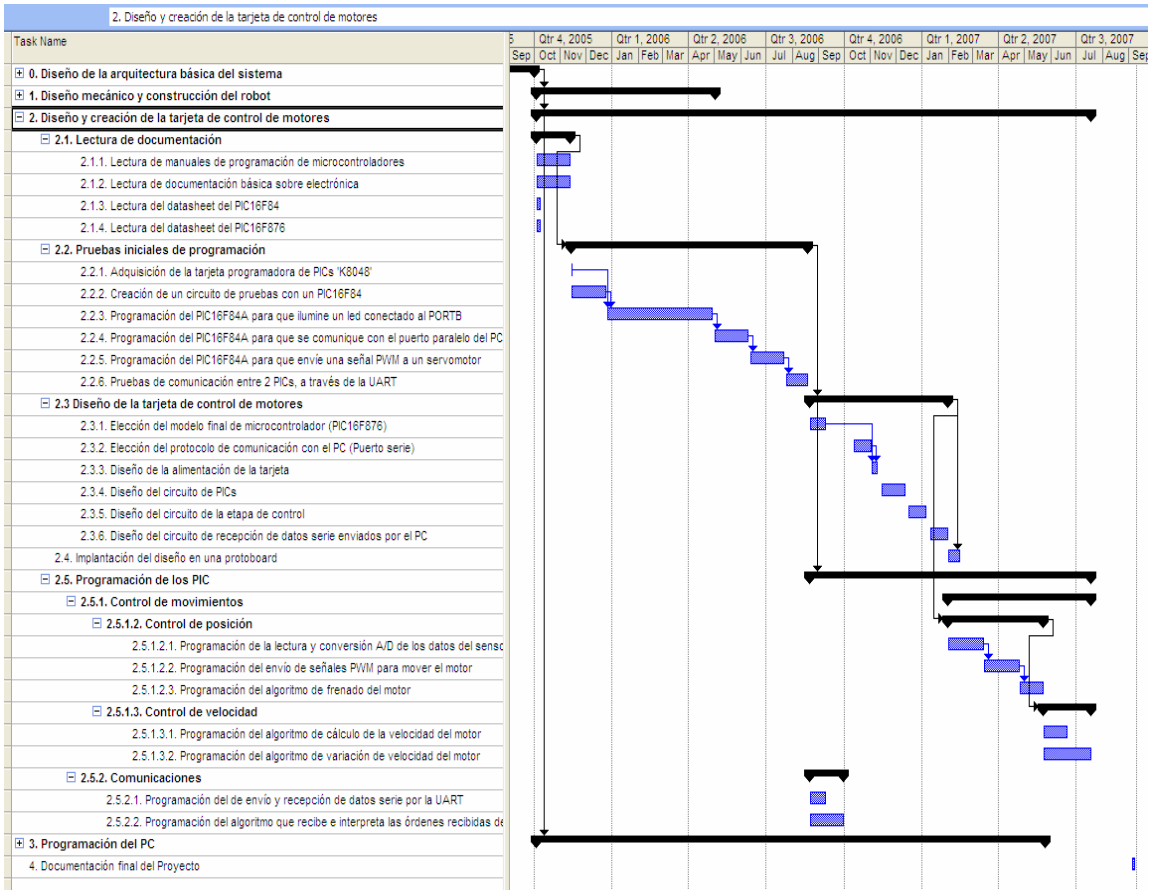
2.5.1.3.1. Programación del algoritmo de cálculo de la velocidad del motor

2.5.1.3.2. Programación del algoritmo de variación de velocidad del motor

2.5.2. Comunicaciones

2.5.2.1. Programación del de envío y recepción de datos serie por la UART

2.5.2.2. Programación del algoritmo que recibe e interpreta las órdenes recibidas desde el PC



3. Programación del PC

3.1. Control de movimientos desde el PC

3.1.2. Pruebas previas con el puerto serie

3.1.2.1. Envío de datos serie entre 2 PCs con WindMill

3.1.2.2. Envío de datos desde un PC a un PIC con WindMill

3.1.2.3. Búsqueda, instalación y prueba de diversas librerías de comunicaciones serie compatibles con Java

3.1.3. Elección de una librería (rxtx)

3.1.4. Instalación de la librería rxtx-2.1-7r2

3.1.5. Programación de una función que envía bytes por el puerto serie

3.1.6. Programación de una función que envía órdenes a la tarjeta de control

3.2. Procesamiento de imágenes desde el PC

3.2.1. Elección del lenguaje de programación a utilizar (Java)

3.2.2. Lectura de manuales de JMF (Java Media Framework)

3.2.3. Instalación del módulo JMF

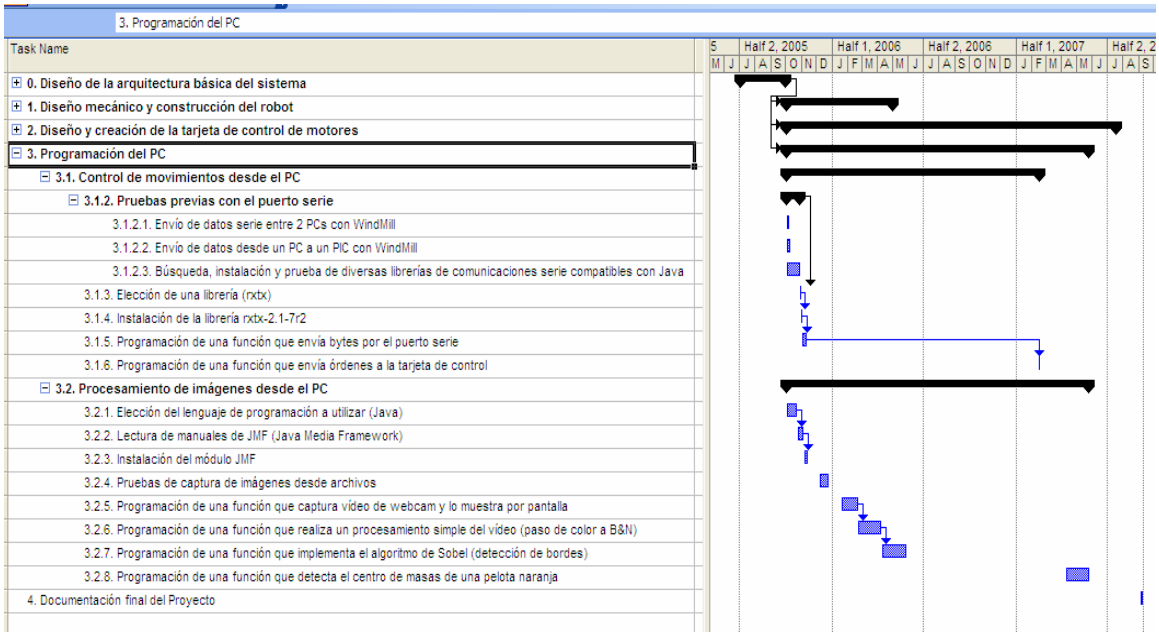
3.2.4. Pruebas de captura de imágenes desde archivos

3.2.5. Programación de una función que captura vídeo de webcam y lo muestra por pantalla

3.2.6. Programación de una función que realiza un procesamiento simple del vídeo (paso de color a B&N)

3.2.7. Programación de una función que implementa el algoritmo de Sobel (detección de bordes)

3.2.8. Programación de una función que detecta el centro de masas de una pelota naranja



4. Documentación final del Proyecto

8. PRESUPUESTO

8.1. MEDICIONES

Se tendrán en cuenta las siguientes consideraciones:

- Los precios de los diversos componentes electrónicos corresponden a los proporcionados por las empresa RS y Conectrol a Septiembre de 2007.

- No se tiene en cuenta el precio del equipo informático necesario para llevar a cabo la programación y pruebas de la placa, por ser válida cualquier configuración con los sistemas operativos adecuados. Sí se considerará, por contra, el equipo informático que contiene la 'inteligencia' del robot (procesamiento de imágenes y control de movimientos), así como las diversas herramientas y equipos de medida necesarios para la fabricación de la placa de control de motores, realización de pruebas y construcción del robot.

- El software empleado para programar los microcontroladores no aparece en el listado, ya que venía incluido con la tarjeta programadora de PICs "Velleman K8048".

- La mano de obra corresponde a la realizada por este proyectista, tanto para la construcción física del robot como para la programación y diseño de la placa de control. No han intervenido otras personas en el desarrollo del proyecto.

8.1.1. BRAZO ROBÓTICO

Estructuras de soporte:

- 2 Placas aluminio 7 x 13 x 0.2 cm
- 1 Tira de aluminio 3 x 8 x 0.2 cm
- 1 Tira de aluminio 12 x 1.5 x 0.1 cm
- 1 Tira de aluminio 12 x 1.5 x 0.2 cm
- 15 Tornillos pequeños
- 18 Tornillos grandes

Estructuras de rotación:

- 5 Rodamientos de acero
- 1 Tubo de fibra de carbono 2 x 0.8 x 0.1 cm
- 1 Tubo de fibra de carbono 3 x 0.8 x 0.1 cm
- 1 Tubo de fibra de carbono 23 x 0.8 x 0.1 cm
- 1 Tubo de fibra de carbono 32 x 0.8 x 0.1 cm

Motores:

- 3 Servos Hitec HSR-5995TG
- 1 Servo Futaba S3003
- 1 Servo Hitec HS-81

8.1.2. CABEZA ROBÓTICA

Estructuras de soporte:

- 1 Placa aluminio 17 x 13 x 0.2 cm
- 1 Tira de aluminio 34 x 1.5 x 0.1 cm
- 2 Tiras de aluminio 26 x 1.5 x 0.2 cm
- 1 tira de aluminio 11 x 0.5 x 0.1 cm
- 8 Tornillos
- 8 Tuercas

Estructuras de rotación:

- 6 Rodamientos de acero
- 2 Tubos de fibra de carbono 2 x 0.8 x 0.1 cm
- 2 Tubos de fibra de carbono 7 x 0.8 x 0.1 cm

Cámaras:

- 2 Cámares web Logitech Quickcam Messenger

Motores:

- 1 Servo Futaba S3003
- 2 Servos Hitec HS-81

8.1.3. PLACA DE CONTROL DE MOTORES

Integrados:

- 5 PIC16F876
- 3 L293D
- 1 74LS14
- MAX233CPP
- 5 Zócalos de 28 patillas
- 3 Zócalos de 16 patillas
- 1 Zócalo de 14 patillas
- 1 Zócalo de 20 patillas

Conectores:

- 1 Conector macho de cable plano de 2 vías
- 1 Clema de dos contactos (para la alimentación)
- 1 Conector DB-9 hembra

Otros elementos:

- 1 Placa protoboard “Adle-board” de 12 x 17 cm
- 5 Resistencias de 100 ohm
- 10 Condensadores cerámicos de 22 pF
- 5 Cristales de cuarzo de 4 Mhz
- 20 Leds
- 1 Alimentador AC-DC de 9V
- 1 Alimentador AC-DC de 6V

8.1.4. EQUIPOS, SENSORES Y HERRAMIENTAS

- 1 Soldador
- 1 Bobina de cable de estaño
- 1 Multímetro
- 1 Placa de programación de microcontroladores “Velleman”
- 1 Torno multipropósito “Twister”

8.1.5. MANO DE OBRA

Descripción	Cantidad
Documentación previa	75 horas
Diseño y montaje del robot	300 horas
Diseño e implementación de la placa de control de motores	300 horas
Desarrollo de software	500 horas
Depuración y pruebas	50 horas
Documentación final	25 horas

8.2. SUMAS PARCIALES

8.2.1. BRAZO ROBÓTICO

Descripción	Cantidad	Precio/ud.	Precio total
Placas aluminio 7 x 13 x 0.2 cm	2	0.09 €	0.18 €
Tira de aluminio 3 x 8 x 0.2 cm	1	0.03 €	0.03 €
Tira de aluminio 12 x 1.5 x 0.1 cm	1	0.02 €	0.02 €
Tira de aluminio 12 x 1.5 x 0.2 cm	1	0.02 €	0.02 €
Tornillos pequeños	15	0.85 €	12.75 €
Tornillos grandes	18	0.13 €	5.4 €
Rodamientos de acero	5	5.3 €	26.5 €
Tubo fibra de carbono 2 x 0.8 x 0.1 cm	1	0.2 €	0.2 €
Tubo fibra de carbono 3 x 0.8 x 0.1 cm	1	0.3 €	0.3 €
Tubo fibra de carbono 23 x 0.8 x 0.1 cm	1	2.3 €	2.3 €
Tubo fibra de carbono 32 x 0.8 x 0.1 cm	1	3.2 €	3.2 €
Servo Hitec HSR-5995TG	3	150 €	450 €
Servo Futaba S3003	1	12 €	12 €
Servo Hitec HS-81	1	11 €	11 €
Total Brazo robótico			**758,12 €**

8.2.2. CABEZA ROBÓTICA

Descripción	Cantidad	Precio/ud.	Precio total
Placa aluminio 17 x 13 x 0.2 cm	1	0.22 €	0.22 €
Tira de aluminio 34 x 1.5 x 0.1 cm	1	0.05 €	0.05 €
Tira de aluminio 26 x 1.5 x 0.2 cm	1	0.04 €	0.04 €
Tira de aluminio 11 x 0.5 x 0.1 cm	1	0.01 €	0.01 €
Tornillos	8	0.15 €	1.2 €
Tuercas	8	0.15 €	1.2 €
Rodamientos de acero	6	5.30 €	31.8 €
Tubo fibra de carbono 2 x 0.8 x 0.1 cm	2	0.2 €	0.4 €
Tubo fibra de carbono 7 x 0.8 x 0.1 cm	2	0.7 €	1.4 €
Cámara web Logitech Quickcam Messenger	2	26 €	52 €
Servo Futaba S3003	1	12 €	12 €
Servo Hitec HS-81	2	11 €	22 €
Total Cabeza robótica			**122,32 €**

8.2.3. PLACA DE CONTROL DE MOTORES

Descripción	Cantidad	Precio/ud.	Precio total
PIC16F876	5	10.3 €	173.82 €
L293D	3	5 €	15 €
74LS14	1	1.3 €	1.3 €
MAX233CPP	1	6 €	6 €
Zócalo de 28 patillas	5	0.3 €	1.5 €
Zócalo de 16 patillas	3	0.3 €	0.9 €
Zócalo de 14 patillas	1	0.3 €	0.3 €
Zócalo de 20 patillas	1	0.3 €	0.3 €
Conector macho de cable plano de 2 vías	1	1.2 €	1.2 €
Conector DB-9 hembra	1	0.65 €	0.65 €
Placa protoboard “Adle-board” de 12 x 17 cm	1	9 €	9 €
Resistencia de 100 ohm	5	0.05 €	0.25 €
Condensador cerámico de 22 pF	10	0.10 €	1 €
Cristal de cuarzo de 4 Mhz	5	1 €	5 €
Led rojo	20	0.10 €	2 €
Alimentador AC-DC de 9V	1	8 €	8 €
Alimentador AC-DC de 6V	1	8 €	8 €
Total Placa control			**234,22 €**

8.2.4. EQUIPOS, SENSORES Y HERRAMIENTAS

Descripción	Cantidad	Precio/ud.	Precio total
Ordenador portátil Hp Pavilion	1	1.000 €	1.000 €
Soldador JBC 25 W	1	30 €	30 €
Bobina de cable de estaño	1	2 €	2 €
Digital Multimeter	1	12 €	12 €
Placa de programación de microcontroladores “Velleman”	1	20 €	20 €
Torno multipropósito “Twister”	1	33 €	33 €
Total Equipos, Sensores y Herrams.			**1.097 €**

8.2.5. MANO DE OBRA

Descripción	Cantidad	Precio/ud.	Precio total
Total mano de obra	1.250 horas	50 €	**62.500 €**

8.3. *PRESUPUESTO GENERAL*

Descripción	Precio total
Total Brazo robótico	758,12 €
Total Cabeza robótica	122,32 €
Total Placa de control de motores	234,22 €
Total equipos, sensores y herramientas	1.097 €
Total mano de obra	62.500 €
PRESUPUESTO TOTAL	64.711,66 €
PRESUPUESTO TOTAL + IVA (16%)	75.065, 53 €

9. RESULTADOS

Se han efectuado pruebas de velocidad en los motores y se ha medido la capacidad de procesamiento de imágenes. Los resultados han sido más que satisfactorios:

1. Pruebas de Velocidad:

- Las articulaciones del brazo robótico alcanzan velocidades de rotación de hasta 0.45 seg/180º. Sumando el movimiento sincronizado del conjunto de articulaciones, podemos llegar a velocidades de desplazamiento de la raqueta de hasta 4.5 m/seg. El 95% de los movimientos de un jugador medio de ping-pong están por debajo de ese límite. Esto significa que, dotado de la inteligencia adecuada, el sistema físicamente sería capaz de jugar a la velocidad requerida por un humano.
- Los ojos de un ser humano pueden llegar a moverse a velocidades elevadísimas (0.1 seg/180º). Pero estos movimientos repentinos y bruscos sólo se usan cuando desviamos la mirada de nuestro foco de atención para llevarla hacia un nuevo estímulo (son los denominados movimientos saccádicos). En realidad, para las tareas de seguimiento de una pelota de ping-pong que un adversario lance contra la mesa no necesitamos más de 0.3 seg/180º, ya que está es la velocidad máxima a la que se despazará la pelota ante nuestro campo visual si es lanzada a 20 metros/seg desde el lado opuesto contra nuestro lado de la mesa. Lo habitual en un jugador medio será que el 90% de las pelotas no superen los 5 metros/seg y que por tanto basten giros oculares de 0.3 seg/180º). El sistema de Visión Artificial consigue mover los ejes oculares a una velocidad máxima de hasta 0.3 seg/180º. De nuevo, esto implica que, dotado de la inteligencia adecuada, el sistema físicamente sería capaz de jugar a la velocidad requerida por un humano.

2. Pruebas de capacidad de procesamiento de imágenes:

Se han medido las imágenes procesadas por segundo con y sin aplicar procesamiento. El sistema captura imágenes a 15 frames/seg. Si además se aplican algoritmos simples de procesamiento de imágenes (suficientes para detectar una pelota de ping-pong) se procesan 10 frames/seg.

El resultado de la finalización del proyecto ha sido la creación de:

1. *Un sistema de Visión Artificial capaz de:*
   - Capturar y mostrar por pantalla imágenes a una velocidad de 15 frames por segundo.
   - Procesar imágenes con el algoritmo de Sobel de detección de bordes, a 10 frames por segundo.
   - Detectar el centro de masas de una pelota naranja (10 frames por segundo).
   - Mover 2 cámaras web con los mismos grados de libertad que los ojos de un humano.
   - Controlar la velocidad a la que se mueven los ojos.

2. *Un brazo robótico que:*
   - Emula los movimientos de un brazo humano, tiene sus mismas dimensiones y sus mismos grados de libertad.
   - Alcanza una velocidad punta prácticamente igual a la de una persona.
   - Es capaz de mantenerse en una posición fija.
   - Puede controlar diferentes velocidades de desplazamiento.

10. CONCLUSIONES

Pese a su complejidad, el proyecto ha finalizado con éxito, ya que se han cumplido todos los objetivos principales, e incluso algunos objetivos opcionales.

Especialmente difícil ha resultado el diseño de un brazo que fuera capaz de imitar los grados de libertad de una persona. Para los ojos había modelos a seguir (por ejemplo el diseño de Cog), pero el brazo ha sido concebido sin partir de nada que no fuera la mera inspiración aportada por libros de Anatomía. La fase de construcción y ensamblado de las piezas y la búsqueda de materiales han resultado especialmente largas.

A todo ello hay que sumar la adquisición y puesta en práctica de conocimientos de disciplinas muy dispares (Informática, Robótica, Visión Artificial, Electrónica y Mecánica) algunas de ellas alejadas de los habituales proyectos de Ingeniería Informática.

Han sido necesarios más de dos años de trabajo, pero el esfuerzo ha merecido la pena si observamos los resultados obtenidos:

- Diseño y construcción de un brazo robótico ligero y resistente, con idéntica libertad de movimiento a la de un brazo humano.
- Diseño y construcción una cabeza robótica con unos ojos que imitan los movimientos de los ojos de una persona.
- Control de la posición del brazo y de los ojos desde un PC.
- Control de la velocidad de los movimientos de los motores.
- Se muestran por pantalla las imágenes capturadas y las procesadas
- Procesamiento simple de imágenes (paso de color a blanco y negro, por ejemplo)
- Implementación del algoritmo de Sobel, que detecta cambios bruscos en la luminosidad de las imágenes (se muestran por pantalla los bordes resaltados en negro).
- Detección del centro de masas de una pelota de ping-pong (se muestra por pantalla un punto azul que resalta el centro de la pelota).

11. FUTUROS DESARROLLOS

Se ha construido un sistema humanoide básico, que en futuros desarrollos podría llegar a jugar al ping-pong contra una persona. En las pruebas de rendimiento se ha conseguido procesar 10 frames por segundo, tanto con el algoritmo de Sobel como con el algoritmo de detección del centro de una pelota. Además, los algoritmos de control programados en los microcontroladores permiten mover los motores a una velocidad de hasta 0.45 seg/180º, que coincide con la velocidad a la que mueve las articulaciones un jugador no profesional de ping-pong.

Por tanto, el sistema mecánico diseñado, los procesos de control implementados, el ritmo de procesamiento de imágenes y la velocidad de los motores conseguida, son los adecuados para que el robot llegue a jugar contra un humano. Para ello, bastaría con seguir desarrollando dos de los procesos pertenecientes a la capa de control de alto nivel:

- Implementación de los algoritmos de procesamiento necesarios para la detección de la velocidad de la pelota.
- Programación de las reglas de decisión que permitieran seguirla con la vista y golpearla con el brazo.

Estas dos actividades tienen la entidad propia y la carga de horas suficientes como para desarrollarse en un nuevo proyecto.

Pero además, si se añadiera una mano robótica al final del brazo, podrían extenderse las aplicaciones de este sistema brazo-cabeza. Hay infinidad de tareas no relacionadas con el ping-pong, que no requieren una gran fuerza ni grandes dosis de procesamiento de imágenes. Por ejemplo, la asistencia en alimentación a discapacitados, o algunos juegos de mesa, como el ajedrez. También queda, por tanto, abierta esta posibilidad de desarrollo futuro.

12. BIBLIOGRAFÍA

- **Libros:**

[BARR99]
- "Fundamentos de Robótica"
Antonio Barrientos, Luis Felipe Peñín, Carlos Balaguer, Rafael Aracil
Ed. Mc Graw Hill

[RENT00]
- "Robótica Industrial. Fundamentos y aplicaciones"
Arantxa Rentería, María Rivas
Ed. Mc Graw Hill

[ANGU00]
- "Microcontroladores PIC"
José María Angulo Amusátegui, Ignacio Angulo
Ed. Mc Graw Hill

[SOBO97]
- "Atlas de anatomía humana"
J. Sobotta
Editorial Médica Panamericana

[PERE01]
- "Anatomía funcional del aparato locomotor"
A. Pérex Casas, M.E. Bengoechea
Ed. Summa

- **Otra documentación:**

- “Interfacing the Serial / RS232 Port”
  http://www.beyondlogic.org/serial/serial.htm

- “PIC16F876 Datasheet”