

Melanie v2.0

Robot hexápodo robusto de 3 dof¹/pata para terrenos abruptos

Alejandro Alonso Puig – mundobot.com
Abril 2004

Introducción	2
Mecánica	3
Cuerpo	3
Patas	3
Electrónica	5
Tarjeta de gestión de servos	5
Tarjeta I/O	5
Sensores	7
Programación	8
Utilización de la aplicación	8
Grabación/Reproducción de secuencias	8
Controles deslizantes	8
Programación gestual	8
Movimiento por desplazamiento de ondas sobre terreno abrupto	10
Diseño de la aplicación	10
Grabación/Reproducción de secuencias	10
Movimiento por desplazamiento de ondas	12
Cinemática Inversa	12
Cinemática Directa	13
Generación de ondas	13
Detección de obstáculos	15
Bibliografía y enlaces de interés	16

¹ DOF: Degrees of Freedom = Grados de Libertad

Introducción

Melanie es un robot hexápodo de 3 grados de libertad por pata, que por el novedoso diseño de patas de que dispone, puede transportar varios kilos sobre su cuerpo sin excesiva sobrecarga energética.



Algunas características básicas que definen a Melanie v 2.0 son las siguientes:

Mecánica

- Medidas (cm): 33 x 31 x 20
- Peso: 2.1 Kg
- Estructura de PVC y Aluminio
- 6 Patas. 3 grados de libertad por pata.
- Diseño de patas optimizado para minimizar consumo y maximizar potencia
- Accionamiento mediante servos de radiocontrol (Doce de 3kgcm y seis de 5kgcm)

Electrónica

- Circuito de control de servos con gestión de velocidad, controlado por puerto serie.
- Circuito I/O, con 28 puertos de conversión A/D y 8 entradas/salidas digitales. Todos los chips son controlados por bus I²C. Medición de la posición de los 18 servos tomando la señal de sus potenciómetros. Medición del

consumo de energía de los servos midiendo la caída de tensión en resistencias de ½ ohm.

- Batería de 6v NiMH 3300mAh para la alimentación de los servos
- Batería de 9v NiCd para la alimentación de la electrónica de control
- Sensor de medición de distancia por infrarrojos delantero orientable

Programación

- Programa de control en PC desarrollado en Visual C++ 6.0
- Grabación de secuencias de movimiento por programación gestual
- Generación de movimientos por desplazamiento de tres ondas simultáneas y aplicación de algoritmos de cinemática inversa
- Detección de obstáculos en el suelo por variación en el consumo de corriente y adaptación de las patas y trayectorias para solventar los mismos.

En este trabajo se hará mención a todos los aspectos mecánicos, electrónicos y de programación, así como conclusiones de la experiencia.



Mecánica

Cuerpo

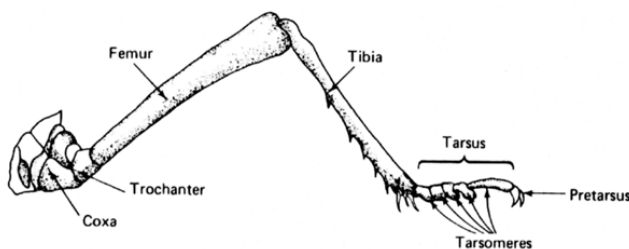
El tronco de Melanie está formado por un chasis central consistente en dos planchas de aluminio preperforado de 2mm sobre el que se acoplan las seis patas de que dispone.

Se ha procurado la máxima simetría posible en el cuerpo para que el centro de gravedad quede lo más centrado posible en el mismo y así mejorar el comportamiento físico del robot y distribuir de forma adecuada el esfuerzo de las patas.

Se ha dotado al robot de un caparazón de porspan ligero para proteger la electrónica y dotarle de una imagen más “animal”.

Patas

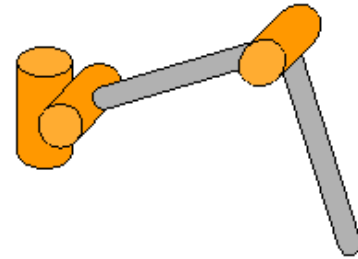
Las patas se han construido en aluminio de 2 mm y PVC de 4 mm. La estructura de las mismas difiere de la distribución de articulaciones habitual de la pata de un insecto.



Las articulaciones Coxo-trocánter y trocánter-fémur de un insecto se encuentran prácticamente fundidas entre sí y se ocupan de los movimientos del fémur en el espacio. La articulación fémur-tibia se ocupa de los movimientos de la tibia en el plano vertical al suelo.

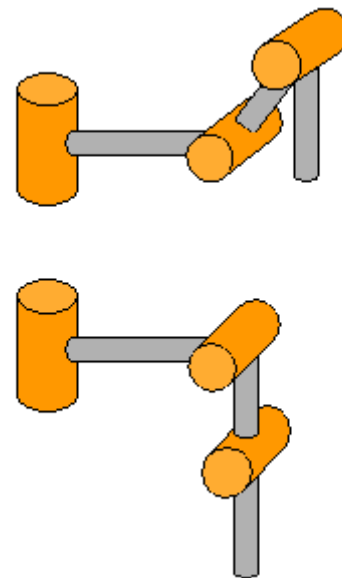
La implementación mecánica de esta estructura suele limitarse a tres grados de libertad y tradicionalmente implica la necesidad de disponer de un par motor importante en los servos debido a

la longitud del fémur. Cuanto más largo sea este, mayor par será necesario en los motores. Además, una vez efectuado el elevamiento del cuerpo, es necesaria una cuantiosa cantidad de energía para mantenerlo en dicha posición.



Estructura tradicional de patas

La estructura de patas de Melanie varía de la estructura común mostrada en la figura de arriba, modificando la posición de las articulaciones tal como se muestra en la figura siguiente:



Estructura de patas de Melanie

Con esta estructura se obtienen dos ventajas importantes respecto a la tradicional mostrada previamente:

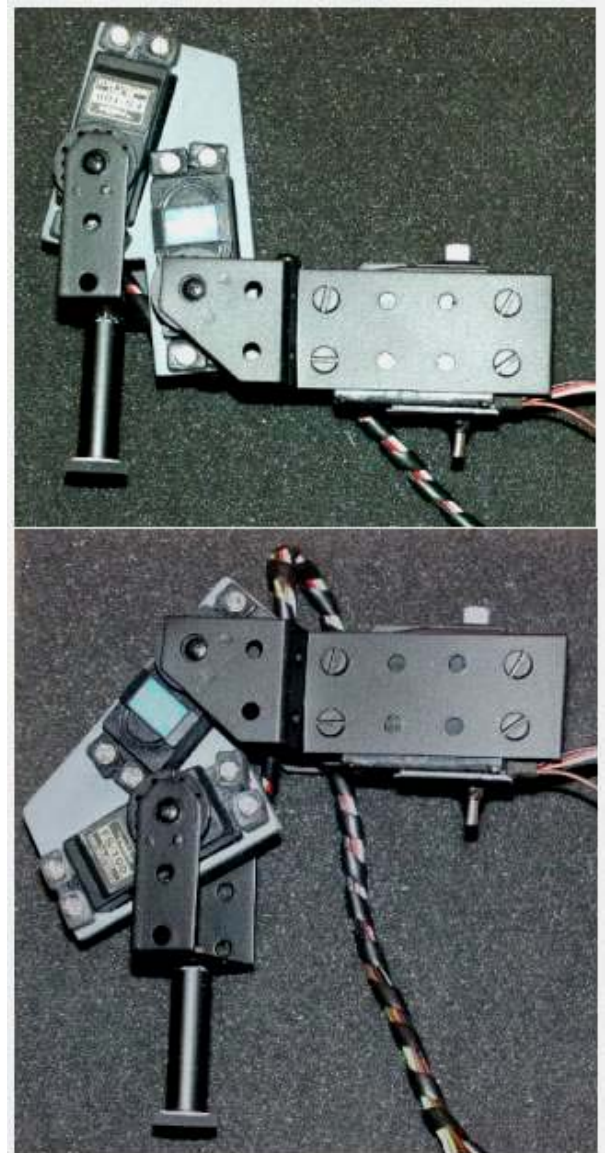
1. Una vez levantado el cuerpo se superponen las articulaciones que permiten el desplazamiento

en el plano vertical de las patas, lo que implica una drástica reducción del consumo de energía al reducir la carga en los motores de dichas articulaciones. La implementación final ha demostrado que en estado levantado puede cortarse el suministro de energía a los motores y el robot permanecerá elevado, incluso con carga adicional (test con 1kg sobre su cuerpo). Es por ello que podrá además llevar cargas adicionales importantes en esta postura.

2. Al desplazar la articulación de la zona del trocánter-fémur a la zona fémur-tibia se desplaza el peso de los motores hacia la tibia, lo que libera de peso al tronco aumentando la capacidad de soportar un peso mayor sobre el mismo. Los ensayos realizados sobre banco de pruebas han permitido comprobar que cada pata es capaz de elevar un peso de 800 gramos además del propio peso de la pata. Por lo que con 6 patas podría elevarse un tronco de 4.800 gramos. En el banco de pruebas el robot, de 2,1kg se ha levantado con una carga adicional de 4Kg.

Los servos utilizados son de 3Kgcmm de par para coxa y articulación de la tibia y 5kgcmm para la articulación intermedia.

Todos los servos utilizados han sido montados con falso eje posterior para asegurar un adecuado anclaje que evite las torsiones sobre el eje de los mismos.



Estructura de patas de Melanie

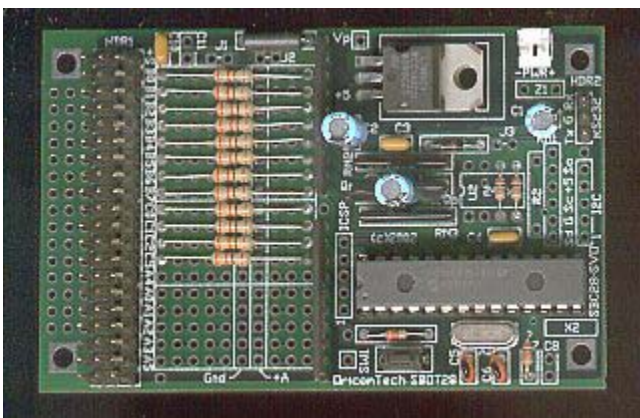
Electrónica

Melanie v2.0 se basa fundamentalmente en dos tarjetas: Un circuito comercial de gestión de servos controlado por puerto serie RS232C de bajo coste y una tarjeta I/O con 28 puertos de conversión A/D y 8 entradas/salidas digitales.



Tarjeta de gestión de servos

La tarjeta MSCC20 de Oricom Technologies (www.oricomtech.com) permite el control de hasta 20 servos y además permite controlar la velocidad de cada servo de forma independiente con un solo comando.



Para información adicional sobre dicha tarjeta puede accederse a la página sobre la misma en

Internet, en la cual se puede encontrar explicación del set de comandos utilizable:

<http://www.oricomtech.com/mscc20.htm>

La tarjeta está alimentada por una batería de 9v, estando los servos alimentados por una batería de 6v NiMH de 3.300 mAh

La tarjeta está conectada mediante un cable de tres hilos al puerto serie de un PC desde el que se envían los comandos de control.

Tarjeta I/O

La tarjeta I/O está compuesta básicamente por 7 chips PCF8591 (cuádruple A/D de 8 bits controlados por I²C) y un chip PCF8574 (expansor de puerto de 8 bits por I²C)

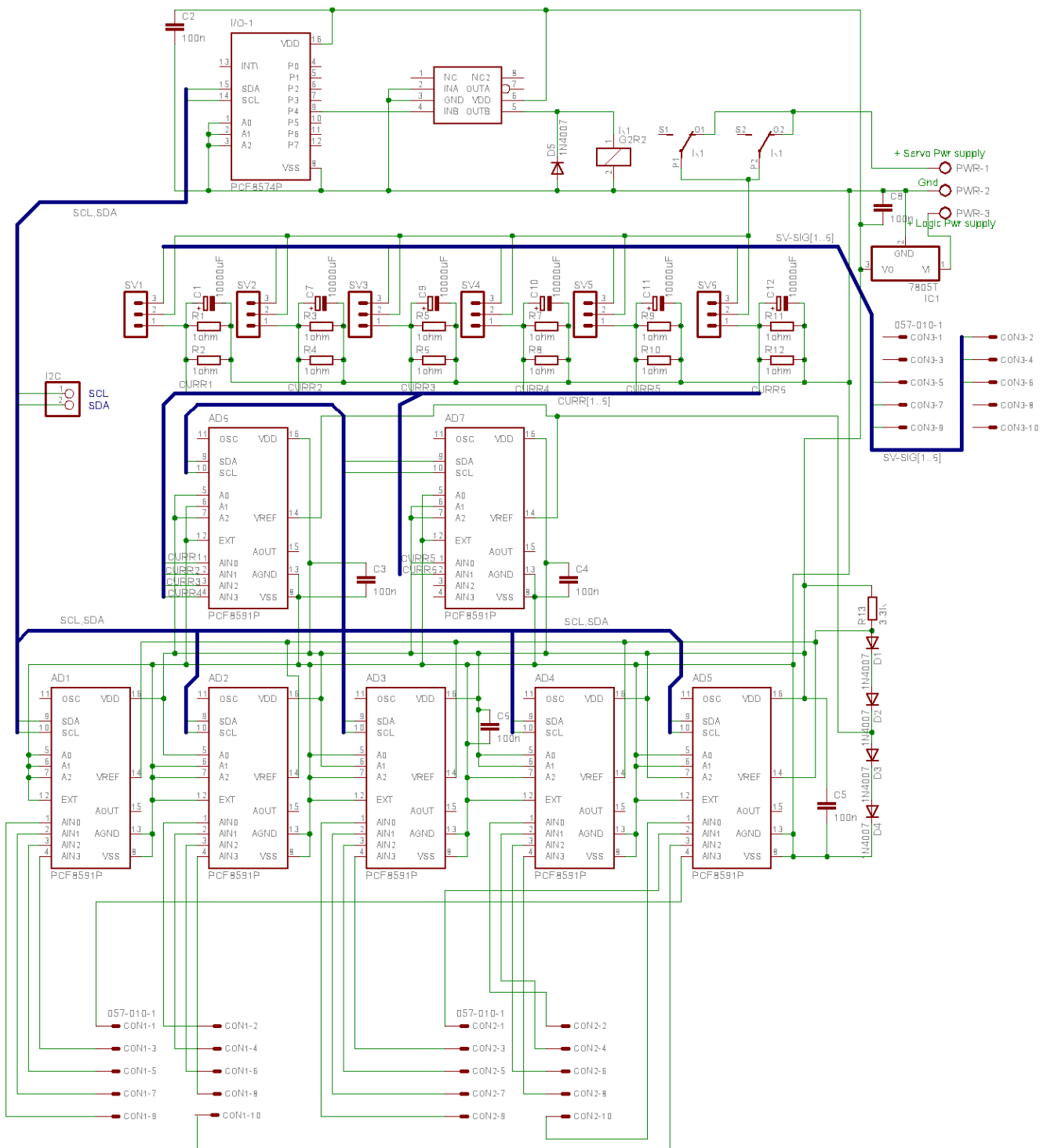
Los 28 canales A/D son utilizados para la medición de la posición de los 18 servos tomando la señal de sus potenciómetros, para la medición del consumo de energía de los servos de 5Kgcm de par, midiendo la caída de tensión en resistencias de ½ ohm y para la medición de tensión de las baterías. Como voltaje de referencia se ha utilizado 2,4v para los chips que miden la posición de los servos y 1,4v para los chips que miden el consumo de los servos y la tensión de las baterías. Para más detalles ver el plano del circuito que se muestra en la siguiente figura.

De los 8 canales digitales solo se utiliza uno de ellos como salida (canal 4). Dicho canal da señal a un driver de potencia TC4425 que alimenta un Relé. Este Relé se utiliza para cortar la tensión que va a los servos de 5Kgcm de par. Esto se utiliza para resetear dichos servos ya que son servos digitales, que mantienen la posición incluso si se corta la señal PWM de posicionamiento. Provocando una caída corta de la tensión en dichos servos tras cortar la señal PWM se quedan desactivados. Los servos normales no requieren este corte de corriente ya que al cortar la señal PWM quedan inmediatamente desactivados. Esta

desactivación se utiliza para dos fines: Desactivar por programa los servos tras un periodo de funcionamiento y desactivar selectivamente servos para poder moverlos manualmente y así realizar la programación gestual.

Puesto que la tarjeta I/O está controlada por bus I²C, se utiliza una tarjeta interface entre el robot

y el ordenador externo desde el que se controla el mismo. Esta tarjeta es la Velleman K8000. Se trata de un interfaz comercial de puerto paralelo a bus I²C. Se puede obtener información adicional en la página www.velleman.be o consultando el trabajo del autor sobre la tarjeta compatible de bajo coste KI²C en <http://www.mundobot.com/tecnicaki2c/spki2c.htm>



A continuación se muestra la relación de elementos conectados a los conectores CON1, CON2 y CON3:

CON1-1	Nada	CON1-2	Potenciómetro Servo 12
CON1-3	Potenciómetro Servo 11	CON1-4	Potenciómetro Servo 13
CON1-5	Potenciómetro Servo 6	CON1-6	Potenciómetro Servo 5
CON1-7	Potenciómetro Servo 4	CON1-8	Potenciómetro Servo 3
CON1-9	Potenciómetro Servo 2	CON1-10	Potenciómetro Servo 1
CON2-1	Potenciómetro Servo 14	CON2-2	Potenciómetro Servo 15
CON2-3	Potenciómetro Servo 16	CON2-4	Potenciómetro Servo 7
CON2-5	Potenciómetro Servo 8	CON2-6	Potenciómetro Servo 9
CON2-7	Potenciómetro Servo 18	CON2-8	Potenciómetro Servo 17
CON2-9	Potenciómetro Servo 19	CON2-10	Nada
CON3-1	Nada	CON3-2	Señal PWM Servo 2
CON3-3	Nada	CON3-4	Señal PWM Servo 12
CON3-5	Señal PWM Servo 5	CON3-6	Señal PWM Servo 15
CON3-7	Señal PWM Servo 8	CON3-8	Nada
CON3-9	Señal PWM Servo 18	CON3-10	Nada

Por otra parte, a los conectores SV1,...SV6 se conectan los siguientes servos, que son de tipo

digital (funcionaría igual con servos analógicos normales):

SV1	Servo 2	SV2	Servo 12
SV3	Servo 15	SV4	Servo 18
SV5	Servo 8	SV6	Servo 5

Las direcciones I²C de los diferentes chips se han establecido por hardware a los siguientes valores:

AD1	144	AD2	146
AD3	148	AD4	150
AD5	152	AD6	154
AD7	156	I/O-1	64

Sensores

El robot posee un sensor de distancia por infrarrojos GP2D12 en la parte frontal, pero en la versión actual no es utilizado.

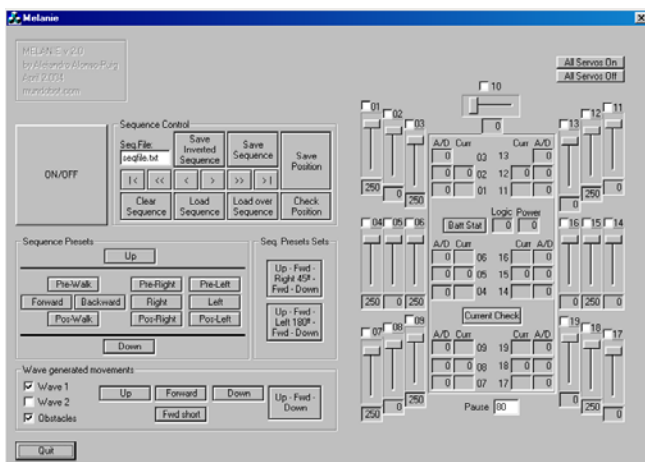


Programación

Para el control del robot se ha generado la aplicación “Melanie v2.0” en Visual C++ 6.0. Dicha aplicación se encuentra localizada en un ordenador externo al robot (Pentium II – 233Mhz, 64Mb RAM, Windows 98). Se ocupa de enviar los comandos adecuados a la tarjeta de control de servos del robot vía puerto serie y controlar la tarjeta I/O por bus I²C. Dicho control por bus I²C se hace a través de la tarjeta interface K8000 o KI²C conectada al puerto paralelo que se ha mencionado en el apartado de Electrónica. El control de esta tarjeta se lleva a cabo mediante el uso de unas librerías específicas. Para más detalles ver el trabajo del autor: <http://mundobot.com/tecnica/ecoi2c/specoi2c.htm> en el que se especifica ampliamente su uso.

Utilización de la aplicación

El interface con el usuario consiste básicamente en un cuadro de diálogo como el que se muestra a continuación.



El cuadro de diálogo muestra una serie de controles que permiten manejar al robot de forma manual y generar secuencias de movimientos que pueden ser grabadas y posteriormente reproducidas.

Para activar el robot habrá de pulsarse en primer lugar el botón ON/OFF, que activará las patas dejándolas fijas en la posición actual.

Grabación/Reproducción de secuencias

La grabación de secuencias puede llevarse a cabo de dos formas:

- Moviendo los servos mediante controles deslizantes y grabando las posiciones
- Moviendo manualmente las patas y grabando las posiciones mediante programación gestual

Controles deslizantes

La parte derecha muestra los valores de posicionamiento de los servos. Moviendo cada control deslizante se moverá un servo concreto.

Pueden grabarse secuencias posicionando los servos como se desee para que el robot tenga una postura determinada, estableciendo un valor de pausa (campo debajo del esquema del robot), que determinará el tiempo que habrá de pasar entre la postura que anteriormente tuviese y la nueva fijada. Dependiendo de dicho valor, que vendrá medido en centésimas de segundo, se aplicará una mayor o menor velocidad a los servos.

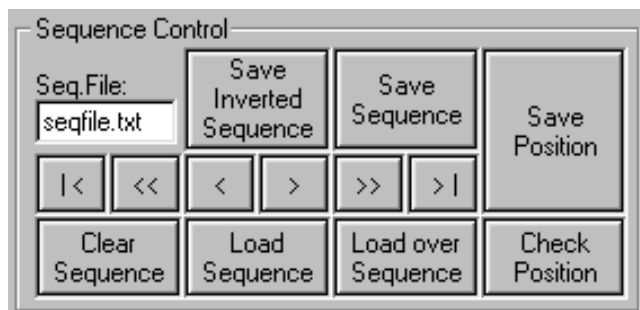
Una vez seleccionada postura y pausa, se pulsaría el botón “Save Position” del área “Sequence Control”, que almacenará la nueva posición en memoria dinámica. Se puede de esta manera ir guardando nuevas posiciones generando una secuencia completa de movimientos. Poniendo un nombre de fichero en el campo “Seq.File” y pulsando “Save Sequence” se grabará en disco duro un fichero con la secuencia generada.

Programación gestual

Para llevar a cabo esta programación habría que desactivar los servos que se desee para poder moverlos manualmente. Esto se lleva a cabo desmarcando los check-box de los servos

deseados en la parte derecha de la pantalla. Tras esto pueden moverse manualmente hasta la posición deseada.

Una vez seleccionada postura y pausa, se pulsaría el botón “Check Position”, que verificará la posición de los servos de las patas y luego “Save Position” del área “Sequence Control”, que almacenará la nueva posición en memoria dinámica. Se puede de esta manera ir guardando nuevas posiciones generando una secuencia completa de movimientos. Poniendo un nombre de fichero en el campo “Seq.File” y pulsando “Save Sequence” se grabará en disco duro un fichero con la secuencia generada.



Los demás controles del área “Sequence Control” llevan a cabo las siguientes funciones:

- **Save Inverted Sequence:** Graba la secuencia pero en sentido inverso. Muy útil cuando un movimiento es idéntico a otro existente pero a la inversa, como por ejemplo andar hacia delante y hacia atrás.
- **Clear Sequence:** Borra la secuencia que hubiese en memoria.
- **Load Sequence:** Carga en memoria una secuencia pregrabada. Ha de especificarse el nombre de la misma en el campo “Seq.File”
- **Load over Sequence:** Carga en memoria una secuencia desde la posición actual de memoria, es decir, si tenemos una secuencia en memoria y nos colocamos al final de la misma con los botones que se explicarán a continuación, podemos pulsar “Load Over Sequence” para cargar una secuencia sobre la existente desde la posición actual. De esta manera se pueden enlazar secuencias.
- “|<” Permite colocarse al principio de la secuencia existente en memoria
- “>|” Permite colocarse al final de la secuencia.
- “>” Permite avanzar una posición en la secuencia cargada en memoria. Esto permite ir paso a paso en la secuencia para depurarla.
- “<” Caso inverso del anterior.
- “>>” Permite reproducir la secuencia desde el punto actual hasta el final sin interrupción.
- “<<” Igual que el anterior pero de forma inversa.

Con los controles anteriores es posible depurar posiciones a base de avanzar paso a paso, corregir la posición deseada con los controles deslizantes y pulsar “Save Position”.

Hay que tener en cuenta que al pulsar este botón se sobrescribe la postura de todos los servos en el lugar actual de la secuencia y el puntero de secuencia se mueve a la siguiente posición quedando preparado para sobrescribirla. Si tan solo se quiere sobrescribir una posición, se hará como se ha indicado y luego se pulsará “<” y luego “>” con lo que forzamos al robot a colocarse según la posición siguiente.

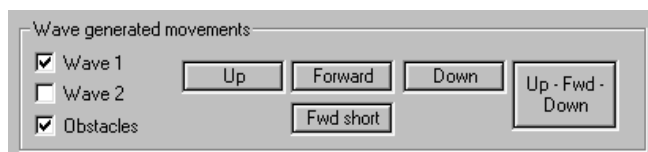
Hay una serie de secuencias básicas pregrabadas que permitirán mover el robot. Dichas secuencias se encuentran agrupadas bajo el conjunto “Sequence Presets”. Permiten levantar el robot, prepararlo para andar, girar,...



Adicionalmente hay un grupo de sets de secuencias pregrabadas bajo el conjunto “Sequence Preset Sets” que permiten ejecutar varias secuencias simples de las indicadas anteriormente para llevar a cabo una tarea concreta, demostración, etc, como levantarse, dar unos pasos, dar la vuelta y volver a tumbarse.

Movimiento por desplazamiento de ondas sobre terreno abrupto

Existe otro área en la pantalla (*wave generated movements*) que permite controlar el robot no mediante la reproducción de secuencias pregrabadas, sino mediante un sistema denominado de desplazamiento de ondas. El propio robot genera los movimientos basados en triples ondas superpuestas.



El programa está preparado para dos tipos de ondas: Wave 1 y Wave 2. La primera hace que el robot ande en modo trípode, es decir, alternando un triángulo de patas, de manera que siempre hay tres patas en el suelo. El segundo modo alterna los movimientos de las patas de manera que siempre estén 5 patas en el suelo.

Adicionalmente se puede hacer que el robot sea o no sensible a obstáculos en el suelo, marcando o no el checkbox “*Obstacles*”. En caso de marcarlo, el robot podrá detectar terreno abrupto, como por ejemplo piedras y caminar sobre ellas manteniendo el cuerpo horizontal al suelo.

Diseño de la aplicación

La aplicación se ha generado en Visual C++ 6.0. Utiliza una librería desarrolladas por P.J. Naughter para el control del puerto serie con el que se comunicará con el robot.

Para más información sobre el manejo de la mencionada librería puede consultarse el trabajo del autor existente en la dirección:

<http://www.codeproject.com/system/cserialport.asp>

Adicionalmente utiliza una librería para el control del interface paralelo-I²C

Para más información sobre el manejo de esta librería puede consultarse el trabajo del autor existente en la dirección:

<http://mundobot.com/tecnica/ki2c/spki2c.htm>

El interface con el usuario consiste básicamente en un cuadro de diálogo asociado a una clase *CMelanieDlg* dentro de la cual se encuentran todas las funciones y variables utilizadas para el control del robot.

Los ficheros fuente de la aplicación contienen numerosos comentarios explicativos y pueden ser accedidos en la dirección:

<http://mundobot.com/projects/melanie/v2/melanie2prg.zip>

No obstante a continuación explicaremos la estructura general de la aplicación.

Grabación/Reproducción de secuencias

Como se explicó anteriormente, es posible almacenar secuencias de movimientos. Para llevar a cabo esto se utilizarán punteros a estructuras almacenadas en memoria dinámica.

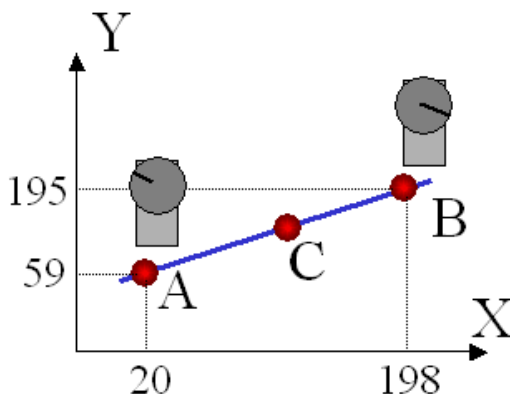
```
struct stPosition
{
    unsigned char          cServoPos[21],
                          cPause;
    struct stPosition      *nextpos,
                          *prevpos;
};
```

En cada elemento de esta estructura podemos almacenar una postura completa, que comprende la posición de 20 servos y el delay que deberá haber entre la postura anterior y esta. Adicionalmente se almacenan punteros a los elementos anteriores y posteriores para tener una lista enlazada en la que nos podamos mover libremente hacia delante y detrás.

El almacenamiento en dicha estructura se realiza al pulsar el botón “Save Position” que llama a la función “*OnSavepos()*”. Dicha función toma los valores de posición de servos existentes en el cuadro de diálogo y los almacena en la estructura.

Si se está haciendo programación gestual, antes de pulsar el botón “Save Position” habrá que pulsar el botón “Check Position” que llamará a la función “OnAD()” y esta a “ChkPos(unsigned char cPos[20], unsigned char cAD[20])” que toma los valores de posición de los servos a base de leer el valor analógico de posición de sus potenciómetros internos. Este valor es convertido al valor de tamaño byte que enviado con el comando de posicionamiento de la tarjeta de control de servos colocaría los servos en la misma posición en la que están. Para conseguir esto el programa posee una tabla (*byte bCal[21][4]*) que almacena para cada servo dos valores para dos puntos de posicionamiento de los servos. Uno de los valores es el que enviado a la tarjeta de control de servos para que posicione el servo y el otro valor es la medición analógica tomada en dicha posición. Teniendo esto para dos posiciones de un servo se pueden obtener los valores de todos los puntos intermedios. Veamos esto con un ejemplo:

Para el servo número 1 se posicionó con los controles deslizantes en el valor 20 (el mostrado en el cuadro de diálogo) y se tomó el valor del potenciómetro (conversión A/D). El valor obtenido fue 59. La misma operación se realizó colocando el servo en la posición 198 obteniendo el valor A/D 195. Puesto que hay una relación lineal entre un valor de posicionamiento y la medida A/D tomada, utilizando la ecuación de



una recta se puede calcular, en función de un valor A/D dado, cual es el valor de posicionamiento del servo. Así, si por ejemplo se lee el valor A/D del servo (voltaje en la patilla central del potenciómetro del servo) y dicho valor es, digamos 90, tendremos que el valor de posicionamiento del servo se obtendrá de la siguiente forma:

Posicion = eje X

AD = eje Y

dos puntos de la recta = (20,59) y (198,195)

$x-b = m(y-a)$ {ecuación cartesiana de una recta}

Usando los puntos: (x=20 , y=59, a=198, b=195)

$$59-195 = m(20 - 198)$$

$$m = (59-195)/(20 - 198)$$

entonces :

$$y-195 = m(x - 198) = (59-195)/(20-198)*(x-198)$$

$$AD-195 = (59-195)/(20-198)*(POS-198)$$

$$POS = 198+(AD-195)/((59-195)/(20-198))$$

Para AD=90, POS=60,6

Se han tomado por tanto los valores de A y B empíricamente para todos los servos y se cargan en la matriz *bCal* por programa. Si se cambiasen los servos habría que calibrarlos de nuevo y actualizar los valores de esta matriz.

La secuencia existente en memoria puede ser volcada a fichero mediante la función “OnSaveSeq()”. De igual manera cargarse un fichero en memoria mediante la función “LoadSequence(CString FileName)” que es llamada desde otras varias.

Solo se pueden reproducir secuencias que estén cargadas en memoria, por lo que los botones de secuencias pregrabadas simplemente llamarán a funciones que carguen en memoria una secuencia pregrabada y la ejecute.

La ejecución de cada paso de la secuencia se lleva a cabo mediante la utilización de la función “SetGlobalPos()”, que moverá los servos y controles deslizantes según los valores del puntero actual.

El delay (Pause) como se ha dicho, indicará el tiempo que debe transcurrir entre una postura y otro. Para que no existan momentos muertos, la función `cSpeed(unsigned char cServo)` calculará la velocidad que se ha de aplicar a cada servo dependiendo de la pausa mencionada y del recorrido a llevar a cabo. Para ello se basa en especificaciones del fabricante de la tarjeta de control de servos que indica que el parámetro de velocidad ha de ser un valor entero entre 1 y 255 que se puede calcular según la siguiente fórmula:

$$V = \frac{8(Pos1 - Pos0)}{T}$$

donde T es el tiempo en centésimas de segundo que se desea que tarde el servo en ir de una posición a otra, siendo Pos1 la posición de valor mayor y Pos0 la posición de menor valor. Los valores de Pos1 y Pos0 según especificaciones del fabricante de la tarjeta de control de servos han de ser valores enteros entre 0 y 250.

El envío de comandos a la tarjeta de control de servos se realiza fundamentalmente mediante la función `SendCommand(CString sCommand)`.

Aunque la tarjeta de control de servos admite gran cantidad de comandos, nosotros utilizaremos solo los siguientes:

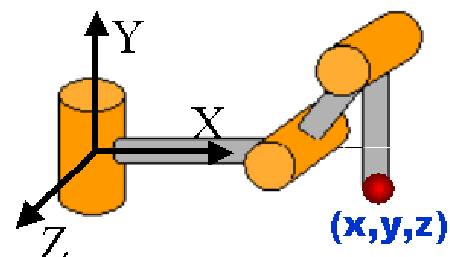
- **SO:** Activa todos los servos
- **SF:** Apaga todos los servos
- **SX1F9D:** Establece los límites de duty cycle de los servos entre 496µsg y 2512µsg
- **SEss:** activa la generación de pulsos para el servo "ss", valor que ha de estar en hexadecimal 00 - 13h (19d).
- **Mssddvv:** Mueve el servo "ss" (valor hexadecimal) a la posición de 8 bits "dd" (valor hexadecimal entre 00 y FAh [250d]) a la velocidad "vv" (valor hexadecimal entre 1 y FFh [255d])

Para más información sobre los comandos disponibles y su utilización, acceder a la documentación del fabricante:

<http://www.oricomtech.com/sub2/msc2-cmd.htm>

Movimiento por desplazamiento de ondas

Como se ha comentado anteriormente, el robot puede desplazarse generando él mismo las posiciones en lugar de utilizar posiciones pregrabadas. El modo en que realiza esto es mediante la utilización de tres ondas cuyos valores variarán en el tiempo y definirán los valores de coordenadas (x, y, z) en el espacio del extremo de cada pata.



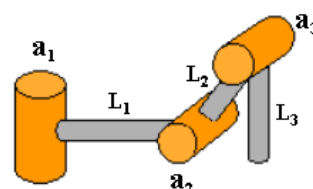
Teniendo estos valores, se utilizará un algoritmo geométrico de cinemática inversa y se obtendrá el valor de posicionamiento de cada servo para que el extremo de la pata se coloque en las coordenadas indicadas.

Cinemática Inversa

El programa hace los cálculos cinemáticos inversos que sean necesarios mediante la función

`Int InvKinematic(double value[2][3], double result[3])`

Esta función devuelve 0 si no ha ocurrido ningún error en el cálculo (por ejemplo podría darse que la coordenada dada esté fuera del espacio alcanzable por la pata).



Los datos de entrada de la función se cargarán en la matriz `value[2][3]` de manera que la primera fila de datos contendrá las longitudes en milímetros de las tres partes de la pata (L_1 , L_2 , L_3) y la segunda fila contendrá las coordenadas (x, y, z) que se desea que tenga el extremo de la pata respecto de su origen de referencia. Como resultado la función cargará en

la matriz *result[3]* los valores en grados de los ángulos que han de tener las articulaciones (a_1 , a_2 , a_3)

El algoritmo utilizado para el cálculo de la cinemática inversa es de tipo geométrico.

Posteriormente pueden convertirse los valores de ángulos en grados de cada servo a valores de posicionamiento aceptados por la tarjeta de control de servos mediante el uso de la función *unsigned char cConvDegToPos(char cServo, double Deg)*, donde *cServo* es el número del servo al que nos referimos y *Deg* es el valor en grados que deseamos que tenga dicha articulación. El valor devuelto por la función es el valor a enviar a la tarjeta de control de servos.

Para conseguir esto se utilizan valores de calibración para cada servo que han sido tomados empíricamente y son cargados al principio del programa en la matriz *cPosConv[20][2]*.

El primer índice indica el servo y el segundo índice indica una posición de 0 grados y una de 45 grados. Por ejemplo, para la pata 1 se cargan los siguientes valores:

```
cPosConv[ 1][0]=203;      cPosConv[ 1][1]=140;
cPosConv[ 2][0]= 77;      cPosConv[ 2][1]=136;
cPosConv[ 3][0]=103;      cPosConv[ 3][1]=160;
```

La primera fila comprende los valores para el servo 1. 203 es el valor que requiere la tarjeta de control de servos para posicionar el servo de manera que el ángulo de la articulación sea 0 grados y 140 es el valor que requiere la tarjeta de control de servos para posicionar el servo de manera que el ángulo de la articulación sea 45 grados.

Con estos valores y, dada la relación lineal entre valor de posicionamiento y ángulo en grados de la articulación, podemos obtener siempre el valor de posicionamiento del servo en función del valor del ángulo de la articulación simplemente utilizando la ecuación de una recta.

Existe una función inversa a esta llamada *double cConvPosToDeg(char cServo, unsigned char cPos)*

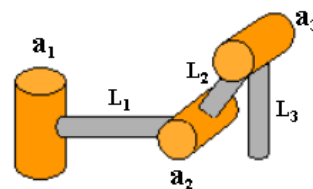
Cinemática Directa

El programa también lleva a cabo en algunos casos cálculos cinemáticos directos. Para ello utiliza la función:

int FwdKinematic (double value[2][3], double result[3])

Esta función devuelve 0 si no ha ocurrido ningún error en el cálculo.

El algoritmo utilizado es también de tipo geométrico.

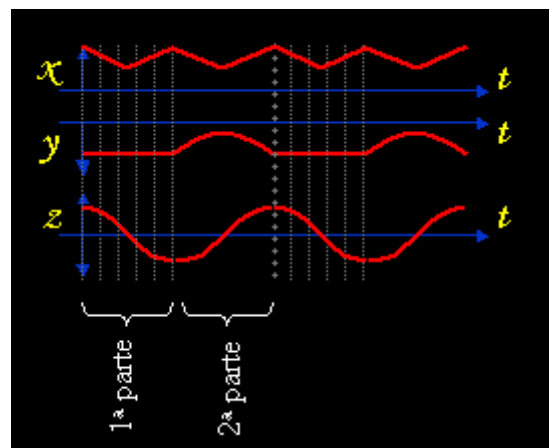


Los datos de entrada de la función se cargarán en la matriz *value[2][3]* de manera que la primera fila de datos contendrá las

longitudes en milímetros de las tres partes de la pata (L_1 , L_2 , L_3) y la segunda fila contendrá los ángulos en grados de las articulaciones (a_1 , a_2 , a_3). Como resultado la función cargará en la matriz *result[3]* los valores de las coordenadas (x , y , z) que tendrá el extremo de la pata respecto de su origen de referencia.

Generación de ondas

Como se ha comentado, son tres ondas las que definen las coordenadas que ha de adoptar cada pata en cada momento.



La coordenada X es calculada mediante una onda triangular, la coordenada Y mediante una semionda y la coordenada Z mediante una onda sinusoidal completa. La frecuencia de la onda triangular ha de ser doble de las otras dos para que la pata se mueva coordinadamente.

Variando el desfase entre las ondas de las diferentes patas y la relación entre la longitud de la primera y segunda parte del ciclo de la onda se podrán obtener diferentes “gaits”.

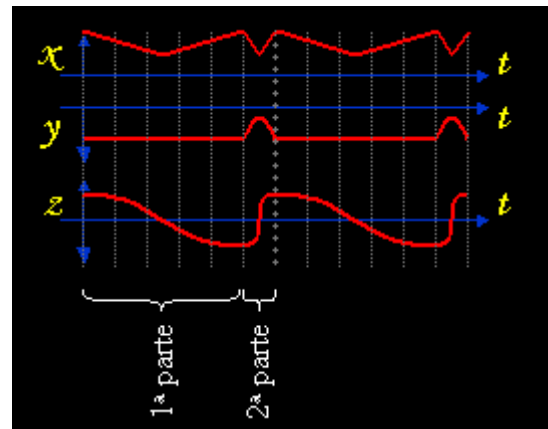
La primera parte del ciclo corresponde al momento en que el extremo de la pata está tocando el suelo y desplazándose hacia atrás (cuerpo avanzando), mientras que la segunda parte del ciclo corresponde al elevamiento y desplazamiento en arco de la pata hacia delante (posicionamiento para un nuevo paso). Así por ejemplo en la figura anterior, la primera y segunda parte del ciclo son de la misma longitud (en el tiempo), es decir que tardará lo mismo en desplazar la pata hacia atrás que en posicionarla de nuevo adelante. Si utilizamos este tipo de ondas en todas las patas, pero damos un desfase de 180 grados a tres de las patas, obtendremos un “gait” del tipo trípole, según el cual el cuerpo siempre estará soportado por tres patas.

Si por ejemplo utilizamos ondas del tipo anterior, pero con la 2ª parte del ciclo de una longitud 1/5 de la de la 1ª parte tendremos que la pata se elevará y colocará adelante en un tiempo determinado, mientras que se desplazará hacia atrás durante un tiempo 5 veces superior.

Si combinamos esta situación con un desfase de 36° ($180/5$) entre las ondas de las diferentes patas obtendremos un “gait” según el cual siempre habrá 5 patas en el suelo y solo una en el aire, pero el cuerpo se estará moviendo con una velocidad continua hacia delante.

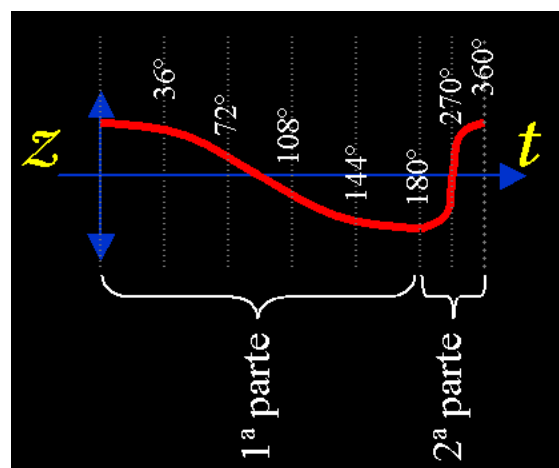
Esto no es casual. Si deseamos que el cuerpo se desplace a velocidad continua, siempre ha de haber un número suficiente de patas en contacto con el suelo desplazándose hacia atrás, no pudiendo estar ninguna parada. Si queremos que siempre haya 5 patas en contacto con el suelo, habrá que ir levantando las patas una a una, de manera que 5 patas habrán de tener tiempo de

elevarse y colocarse delante de forma consecutiva en el tiempo que la sexta pata se desplaza hacia atrás. Es decir que el tiempo de elevación y desplazamiento en el aire hacia delante de una



pata habrá de ser 1/5 del tiempo que tarda en desplazarse la pata hacia atrás. Esto es precisamente la longitud en tiempo de la 2ª parte del ciclo. Por otra parte, como se desea que solo una pata esté en el aire en cada momento (5 patas en el suelo), las ondas de las patas habrán de estar desplazadas $1/5$ de 180° , ya que 180° es el punto en que la onda pasa de la 1ª a la 2ª parte del ciclo.

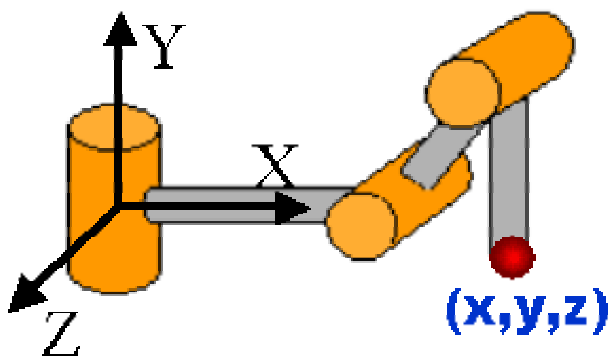
Es importante en este punto resaltar que las ondas utilizadas son las indicadas al principio de este apartado, es decir triangular, semionda y onda sinusoidal completa y que la variación de la longitud en tiempo de la 1ª o 2ª parte del ciclo no es más que una especie de “estiramiento” del tiempo y por tanto los valores en grados para su obtención también se “estiran”. Véase a lo que nos referimos para el caso anterior de la onda sinusoidal (Cosenoidal) que genera el valor de la coordenada Z:



En el programa se encuentra la función *void OnWaveFwd()* que hará caminar hacia delante según el método de desplazamiento de ondas explicado.

Siguiendo el mismo método de desplazamiento de ondas se ha generado la función *MoveLeg* que mueve el extremo de una o varias patas de un punto en el espacio a otro, bien trazando una línea recta entre ambos puntos o realizando un movimiento arqueado de levantamiento, desplazamiento y bajada.

Debe recordarse que las coordenadas manejadas para cada pata son siempre relativas al punto origen de coordenadas de cada pata, según se muestra en la siguiente figura:



Detección de obstáculos











El programa está preparado para la detección de obstáculos en el suelo y adaptar el “gait” para permitir al robot caminar sobre dichos obstáculos.

El modo en que se hace esto es midiendo el consumo de corriente de los servos en el momento de bajada de la pata para posar su extremo sobre el suelo.

La medición del consumo de corriente se hace mediante la función *void ChkCurr()* que envía los comandos necesarios para llevar a cabo la lectura de los conversores A/D por bus I²C y carga en la matriz *Current[6]* los valores de medición de corriente de los 6 servos que más fuerza han de hacer para soportar el cuerpo.

Si la medida de consumo supera un umbral dado, se detiene la pata, se obtiene su posición y se modifica temporalmente la longitud de la segunda parte del ciclo de las ondas de dicha pata y se genera un pequeño desfase para asegurar que se desplazan coordinadamente con el resto de patas. Los valores de fase y longitud de la segunda parte del ciclo de dicha pata vuelven a sus valores normales en el siguiente ciclo, no produciéndose ninguna interrupción en la continuidad de las ondas durante la transición.

Bibliografía y enlaces de interés

-  Entomología Agrícola. Daniel G. Pesante Armstrong. Capítulo 7:
<http://www.uprm.edu/wciag/anscience/dpesante/4008/cap-7.PDF>
-  Puchobot. Robot Cuadrúpedo. Andrés Prieto Moreno:
<http://www.iearobotics.com/personal/andres/proyectos/pucho/pucho.html>
-  Cube Reloaded. Gusano robot. Juan González:
<http://www.iearobotics.com/personal/juan/doctorado/cube-reloaded/>
-  Curso de cinemática directa e inversa (Univ.Guadalajara-Mexico)::
<http://proton.ucting.udg.mx/materias/robotica/r166/r78/r78.htm>
-  Legs. General design considerations: <http://www.frasco.demon.co.uk/rodney/mechan/design.htm>
-  The Walking Machines Catalogue: <http://www.walking-machines.org/>
-  Silo4. Robot cuadrúpedo desarrollado por el CSIC: <http://www.iai.csic.es/users/silo4/>
-  An introduction to robot kinematics: http://www.generalrobotics.org/ppp/Kinematics_final.ppt
-  MSCC20. credit card-sized control computers for driving up to 20 R/C servos. Oricom Technologies:
<http://www.oricomtech.com/mscc20.htm>
-  CSerialPort v1.03 - Serial Port Wrapper. PJ Naugter:
<http://www.codeproject.com/system/cserialport.asp>