

Melanie v1.0

Robot hexápodo robusto de 3 dof¹/pata

Alejandro Alonso Puig – mundobot.com
Marzo 2004

Introducción	2
Mecánica	3
Cuerpo	3
Patas	3
Electrónica	5
Programación	6
Utilización de la aplicación	6
Diseño de la aplicación	7
Conclusiones y líneas futuras	9
Bibliografía y enlaces de interés	10

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.

La versión 1.0 que se muestra en este informe técnico se refiere a la versión más sencilla de la implementación del robot, en la que está conectado a un PC mediante puerto serie (RS232C) para su control desde el mismo.



Algunas características básicas que definen a Melanie v 1.0 son las siguientes:

- Medidas (cm): 33 x 31 x 12
- Seis patas
- Tres grados de libertad por pata
- Accionamiento mediante servos de radiocontrol (Doce de 3kgcm y seis de 5kgcm)
- Control “suavizado” de servos mediante driver con gestión de velocidad.
- Comunicación con PC maestro mediante puerto serie RS232C.
- Sensor de medición de distancia por infrarrojos delantero orientable
- Estructura de PVC y Aluminio
- Batería de 6v NiMH 3300mAh para la alimentación de los servos
- Batería de 9v NiMH para la alimentación de la electrónica de control
- Programa de control en PC desarrollado en Visual C++ 6.0

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

Mecánica

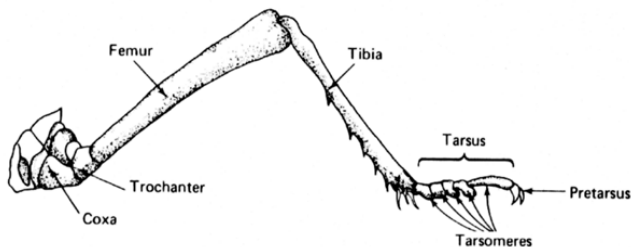
Cuerpo

El tronco de Melanie está formada 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.

Patas

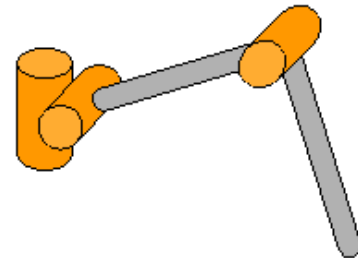
Las patas se han construido en aluminio de 2 mm y PVC de 4 mm. La estructura de las mismas difiere de la distribución de articulaciones habitual de la pata de un insecto.



Las articulaciones Coxo-trocanter y trocanter-fémur de un insecto se encuentran prácticamente fundidas entre si 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.

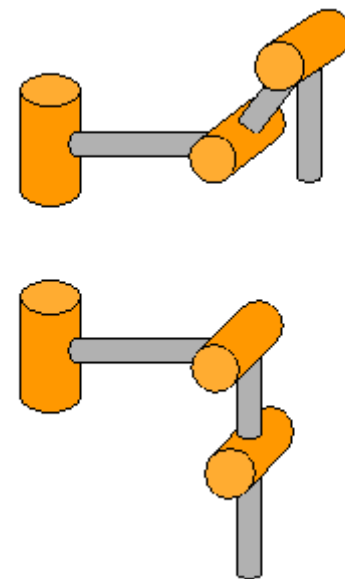
La implementación mecánica de esta estructura suele limitarse a tres grados de libertad y tradicionalmente implica la necesidad de disponer de un par motor importante en los servos debido a la longitud del fémur. Cuanto más largo sea este, mayor par será necesario en los motores. Además, una vez efectuado 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:

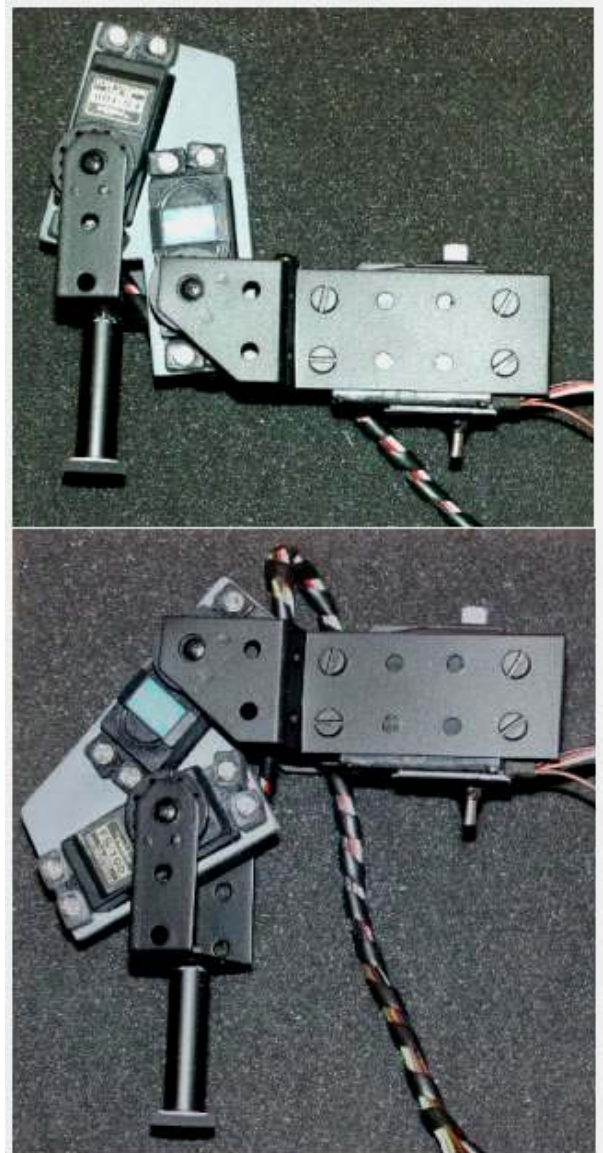
1. Una vez levantado el cuerpo se superponen las articulaciones que permiten el desplazamiento en el plano vertical de las patas, lo que implica una drástica reducción del consumo de energía al reducir la carga en los motores

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. Podrá además llevar cargas adicionales importantes en esta postura.

2. Al desplazar la articulación de la zona del trocanter-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.

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.

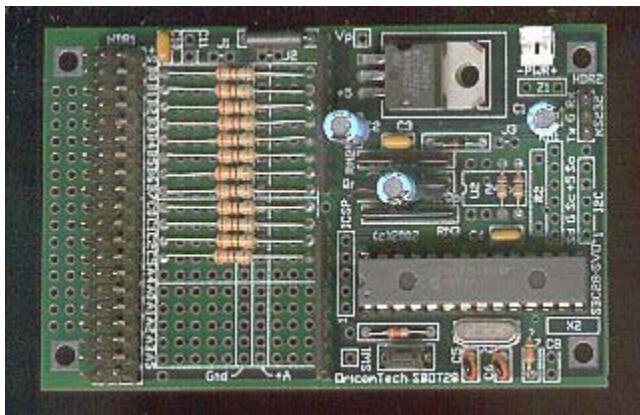


Estructura de patas de Melanie

Electrónica

La electrónica utilizada en Melanie v1.0 es muy sencilla al basarse exclusivamente en un circuito comercial de gestión de servos controlado por puerto serie RS232C de bajo coste.

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/svo-cc.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.

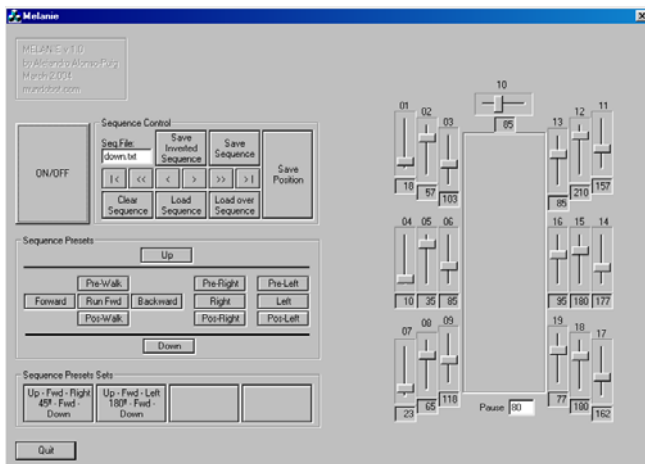
El robot posee un sensor de distancia por infrarrojos GP2D12 en la parte frontal, pero en la versión actual no es utilizado.

Programación

Para el control del robot se ha generado la aplicación “Melanie” en Visual C++ 6.0. Dicha aplicación se encuentra localizada en un ordenador externo al robot (Pentium II – 233Mhz, 64Mb RAM, Windows 98) y se ocupa de enviar los comandos adecuados a la tarjeta de control de servos del robot vía puerto serie.

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.

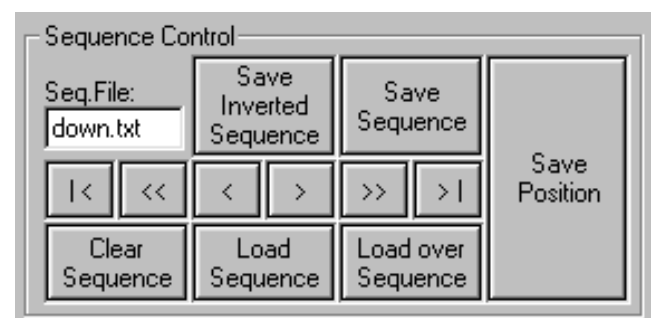
Para activar el robot habrá de pulsarse en primer lugar el botón ON/OFF, que colocará las patas en posición pegada al cuerpo.

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.



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

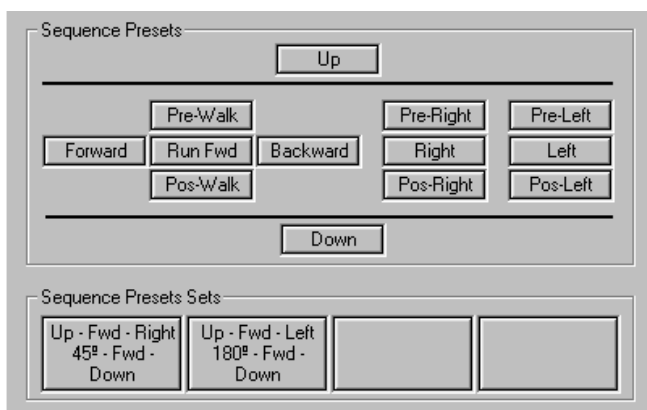
existente desde la posición actual. De esta manera se pueden enlazar secuencias.

- “|<” 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 en el plano. Dichas secuencias se encuentran agrupadas bajo el conjunto “Sequence Presets”. Permiten levantar el robot, prepararlo para andar, hacerlo correr, 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.

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>

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/melanieprg.zip>

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

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.

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 ejecuten.

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 otra. 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 han de ser valores enteros entre 0 y 250.

El envío de comandos 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:

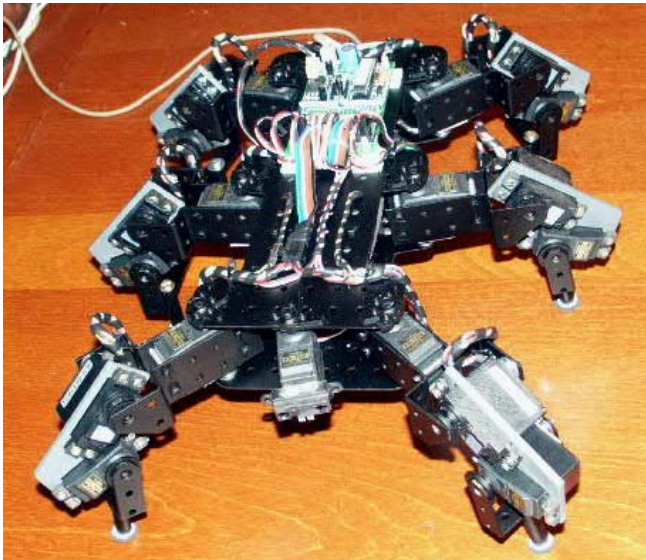
- **SO**: Activa todos los servos
- **SF**: Apaga todos los servos
- **SX1F9D**: Establece los límites de duty cycle de los servos entre 496µs y 2512µs
- **SEss**: activa la generación de pulsos para el servo “ss”, valor que ha de estar en hexadecimal 00 - 13h (19d).
- **Mssddvv**: Mueve el servo “ss” (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>

Conclusiones y líneas futuras

En el presente estudio se ha expuesto el diseño de un robot hexápodo de diseño mecánico óptimo, capaz de llevar cargas adicionales con un incremento de esfuerzo adecuado.










Se ha conseguido por tanto el objetivo deseado, consistente en crear un robot hexápodo capaz de andar con secuencias pregrabadas.

El robot carece de elementos sensoriales que en versiones posteriores podrán incluirse para una adecuada percepción del entorno y reacción ante el mismo. Entre los elementos sensoriales recomendados estarían los siguientes:

- Percepción de distancia a obstáculos por infrarrojos. El robot dispone del sensor, pero no lo utiliza en la versión actual.
- Detección de la posición real de las articulaciones de las patas, lo que permitiría una programación gestual del robot así como detección de obstáculos y sobrecargas.
- Detección del consumo de corriente de los servos. Asimismo permitirá detectar obstáculos al aumentar el consumo del servo antes de llegar a la posición esperada debido a un bloqueo por impacto o por el contrario detectar la falta de suelo firme al tener un consumo de corriente inferior al esperado al posicionar una pata en el punto deseado.

El robot se ha construido de una forma robusta, por lo que habrá de permitir múltiples pruebas y mejoras en un futuro.

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>
-  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/>
-  MSCC20. credit card-sized control computers for driving up to 20 R/C servos. Oricom Technologies:
<http://www.oricomtech.com/svo-cc.htm>
-  CSerialPort v1.03 - Serial Port Wrapper. PJ Naugter:
<http://www.codeproject.com/system/cserialport.asp>