DISEÑO DE ROBOTS ÁPODOS

Autor: Juan González Gómez
Tutor: Eduardo Boemo

22/Julio/2003

Resumen

En este trabajo se realiza un estudio teórico sobre los **robots ápodos** existentes, centrándose en la **robótica modular reconfigurable** aplicada al **problema de la locomoción**. Se diseña un módulo pequeño y sencillo, con un servomecanismo, que permite construir robots ápodos que se puedan mover bien en línea recta o bien en un plano. Con cuatro de estos módulos, se construye el robot ápodo **Cube Reloaded** y se adapta el software y la electrónica desarrolados en Cube 2.0, un gusano ápodo previamente diseñado. Se proponen dos maneras alternativas de controlar el gusano, usando **FPGAs**, una mediante lógica combinación y secuencial y otra con una **CPU empotrada**. Finalmente se realizan pruebas de movimiento y se consigue que Cube Reloaded pueda superar un **circuito de pruebas** constituido por un cilindro de 8 cm de diámetro y un escalón de 3.5 cm de alto, lo que pone de manifiesto la **versatilidad en la locomoción**.

Palabras clave

Robótica, microbótica, mecatrónica, gusano, ápodo, serpiente, microcontroladores, FPGA, electrónica, hardware

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Encuadre

1.1.1. Locomoción

En la robótica existen dos grandes áreas: **manipulación** y **Locomoción**. La manipulación es la capacidad de actuar sobre los objetos, trasladándolos o modificándolos. Esta área se centra en la construcción de manipuladores y brazos robóticos. La locomoción es la facultad de un robot para poder desplazarse de un lugar a otro. Los robots con capacidad locomotiva se llaman **robots móviles**.

Dentro de la locomoción hay tres puntos claves:

1. **Desplazarse un incremento en línea recta**

   La complejidad depende del tipo de robot. Es muy sencillo que un robot con ruedas avance en línea recta, pero no es tan sencillo que lo haga un robot con patas.

2. **Giros y traslaciones en múltiples direcciones**

   Nuevamente depende del tipo de robot. El hacer que un robot ápodo pueda desplazarse por un plano es más complejo que en un robot con ruedas.

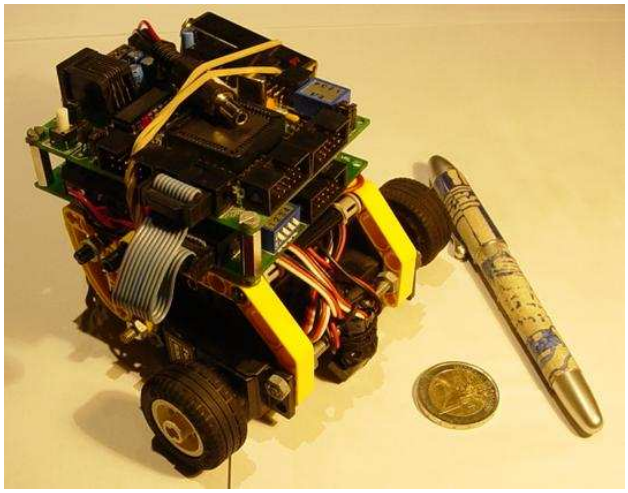
3. **Planificación de trayectorias y navegación**

   Que el robot sepa qué camino elegir para llegar a un determinado lugar.

Según los **efectores empleados** para conseguir la locomoción, tradicionalmente se ha establecido la siguiente **clasificación**:

- **Con ruedas**. Los efectores son ruedas. Por ejemplo, el microbot tritt[1] (figura 1.1), tiene tres ruedas, dos motrices y una loca.
- **Con orugas**. Por ejemplo los robots tipo “carro de combate”, como el Robot de Docencia[2] (Figura 1.1), desarrollado en la EPS de la UAM.
- **Con patas**. Cualquier robot que use patas para conseguir la locomoción: perros, gatos, hexápodos, arañas... Normalmente se trata de robots bio-inspirados. Por ejemplo el robot perro Puchobot[3] (figura 1.2), o como el Hexápodo Sheila[4] (figura 1.2).
- **Otros**. Aquí se sitúa cualquier otro tipo de locomoción no clasificable en niguna de las anteriores categorías, como por ejemplo el robot gusano **Cube 2.0** [5][6]

TRITT **ROBOT DE DOCENCIA**



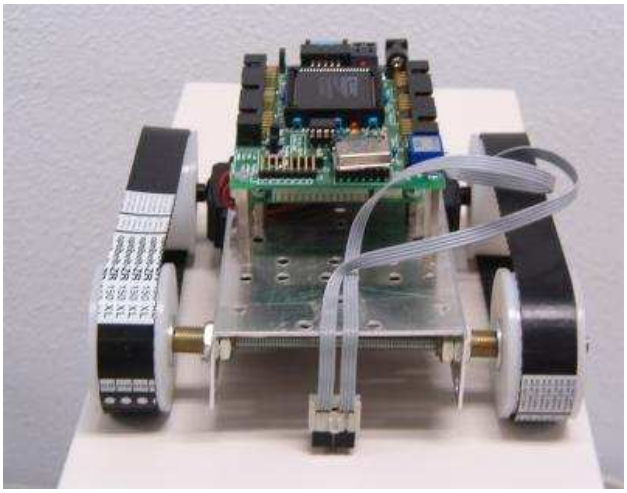
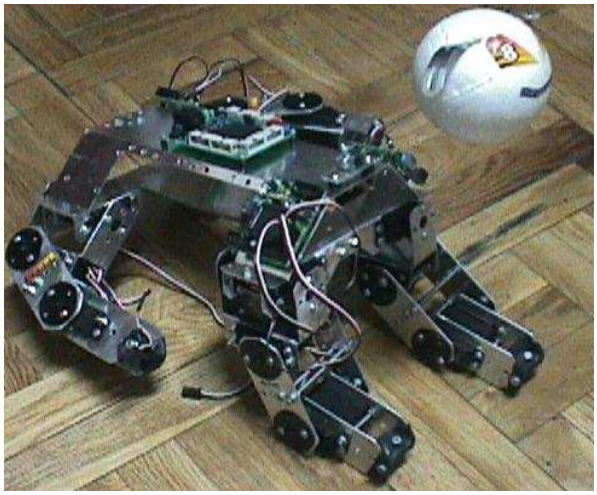


Figura 1.1: A la izquierda Tritt, un ejemplo de robot móvil con ruedas, a la derecha el robot de docencia, con orugas

PUCHOBOT **SHEILA**



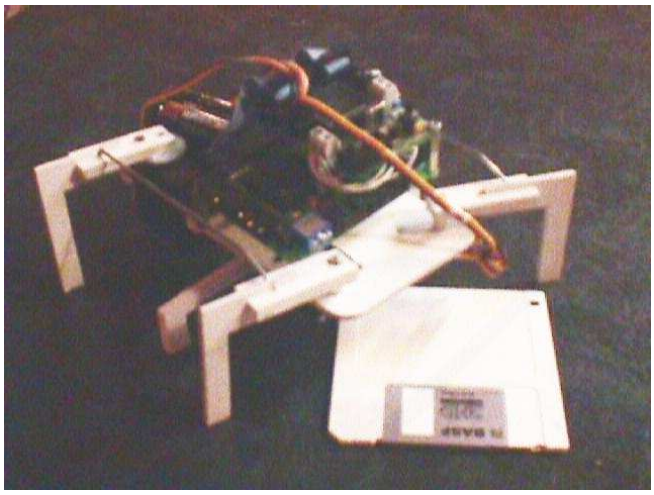


Figura 1.2: A la izquierda,Puchobot, un robot que usa cuatro patas para su locomoción. A la derecha, Sheila, un hexápodo

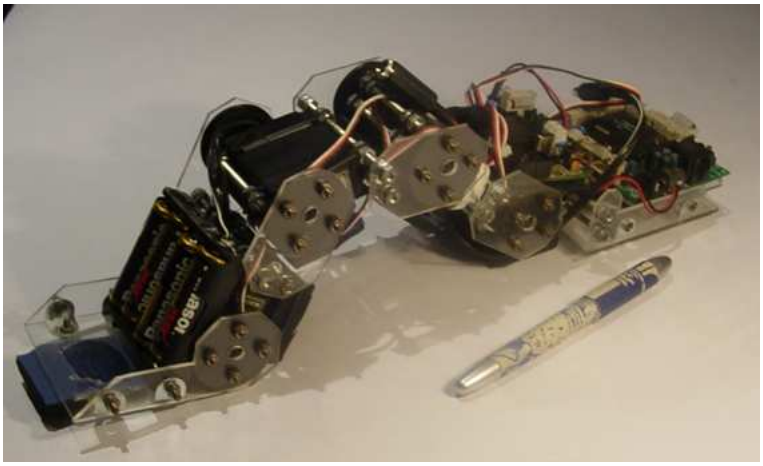


Figura 1.3: El robot ápodo CUBE 2.0

Esta clasificación no es del todo satisfactoria porque existen demasiados robots que se pueden introducir en la categoría *otros*. Mark Yim introdujo una nueva taxonomía[15], que se describe en el apartado 2.4.

Según cómo se realice la locomoción, existen dos tipos:

1. **Locomoción estáticamente estable** (statically stable locomotion)

   El robot debe tener suficientes puntos de apoyo, que conforman el **polígono de apoyo**. El centro de gravedad (CG) debe caer siempre dentro de este polígono.

2. **Locomoción dinámicamente estable** (dinamically stable locomotion)

   El robot tiene que ser estable en movimiento (no caerse) aunque puede no ser estable en reposo (ejemplo, un robot unípodo). Este tipo de locomoción requiere más control pero aporta mayor velocidad.

En este trabajo nos centraremos en la locomoción estáticamente estable aplicada a un robot ápodo, que no usa ni ruedas, ni orugas, ni patas para desplazarse. El movimiento se restinge al desplamiento en línea recta.

1.1.2. El problema de la locomoción

Uno de los grandes retos en el área de la locomoción es el de desarrollar un robot que sea capaz de moverse por cualquier tipo de entorno, por muy escarpado y complicado que sea. Esto tiene especial interés en la exploración de otros planetas, en los que no se sabe qué tipo de terreno nos podemos encontrar. Se pueden realizar estudios previos y diseñar un robot específico un determinado terreno. Pero lo interesante es conseguir un robot versátil que pueda moverse por la mayor cantidad de terrenos posibles.

La Nasa está especialmente interesada en este tipo de proyectos, financiando iniciativas destinadas a la construcción de robots para la exploración por terrenos escarpados. Dos de estos proyectos son el CMU Ambler[8][9] y el Dante II[11][12].

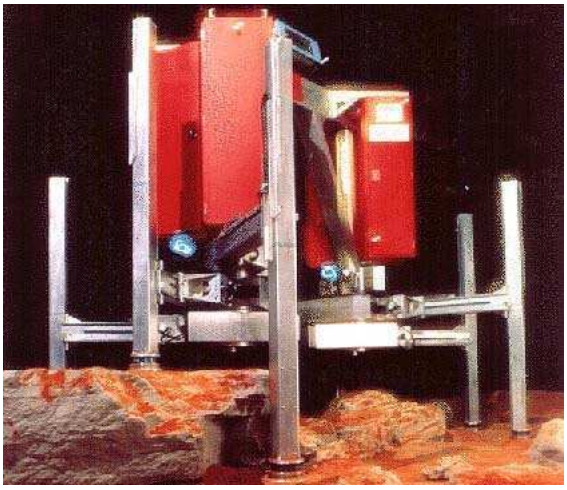


Figura 1.4: Estructura del Robot Ambler del CMU

1.1.2.1. CMU Ambler

El robot **Ambler**[8], desarrollado en el Carnegie Mellon University (CMU)[7], es un robot autónomo para la exploración de planetas[9], financiado por la **NASA**. Se trata de un robot de 6 patas, diseñado para desplazarse sobre terreno escarpado, como el que se encuentra sobre la superficie de Marte, capaz de desplazarse con la ayuda externa mínima[1].

Las especificaciones del diseño son: Bajo consumo (la locomoción tiene que ser muy eficiente), capaz de superar saltos de 1m de altura y pendientes del 60 %.

Utiliza locomoción con patas, dado que es la más adecuada para terreno escabroso y presenta mayor eficiencia teórica[10]. El sistema en total tiene 18 grados de libertad (3 por cada pata) (figura1.4). La principal característica es que todas las patas son concéntricas y pueden modificar su longitud, lo que permite que el Ambler se estabilice sobre cualquier superficie. Además el movimiento de avance es independiente del de estabilización.

Emplea una locomoción estáticamente estable. Otros datos de interés: 3.5m de altura media, 3m de anchura nominal.

1.1.2.2. Dante II

También en el CMU[7], desarrollaron el robot **Dante II**[11] que utilizaron en Julio de 1994 para la exploración del volcan del Monte Spurr en Alaska[12], proyecto subvencionado por la NASA. Se trata de un robot de 8 patas, con un sistema de locomoción denominado *framewalker*, caracterizado por el desplazamiento de dos planos paralelos a la superficie, cada uno dotado de 4 patas que pueden subir y bajar. El movimiento es estáticamente estable.

Se diseñó específicamente para descencer al volcan y obtener datos para su posterior análisis científico. Utilizó para ello una cuerda, que une al robot con la cima del volcan, y que le permite controlar el descenso (como si estuviese haciendo rapel).

Este robot no es totalmente autónomo, está telecontrolado, aunque es capaz de realizar algunas acciones por su cuenta. Los operadores se encontraba a 120Km de distancia, recibiendo la información

[1] En los proyectos de exploración de Marte, la señal tarda 45 minutos en regresar a la tierra, por lo que los robots que se envíen deben ser lo más autónomos posible.

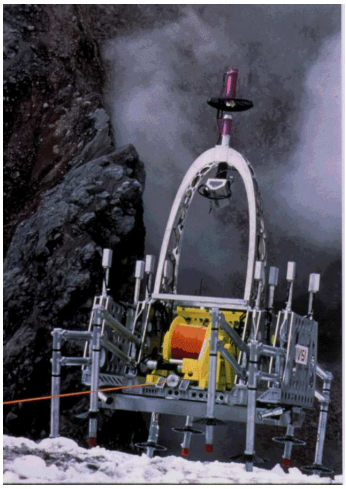


Figura 1.5: Dante II descendiento al interior del Cráter del Monte Spurr

vía satélite. El robot estuvo operativo durante 5 días, en unas condiciones adversas (altas temperaturas y presencia de gases tóxicos) y un terreno muy escarpado. Los objetivos de la misión fueron:

- Descender al interior del cráter
- Recolectar y enviar datos del interior
- Realizar toda la operación sin necesidad de presencia humana

Una descripción técnica del proyecto se puede encontrar en [12].

1.1.2.3. Necesidad de más versatilidad

Ambos robots, destinados a la exploración de terrenos no conocidos, son totalmente diferentes (sólo coinciden en que tienen patas). Intentan ser lo más versatiles posibles, pero están creados para moverse por unos entornos de unas características previamente establecidas. **Son Robots muy especializados para el tipo de función a realizar**. Los tiempos de rondan los 2 años para cada uno.

Hasta el año 93, los robots de exploración se diseñaban teniendo en cuentas unas características conocidas a priori sobre el terreno por el que iban a moverse. Un robot para cada tipo de terreno.

A partir del 94, apareció el primer robot basado en los paradigmas de la **robótica modular reconfigurable**: **Polypot**. ¿Por qué no trabajar en una nueva línea de investigación en la que el robot se adapte al terreno modificando su forma y su manera de desplazarse? (en vez de realizar un robot específico para un terreno determinado).

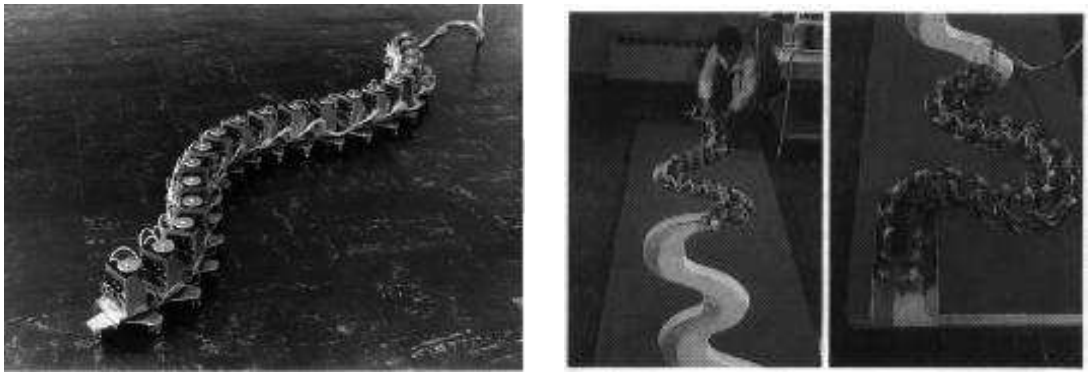


Figura 1.6: ACM III (1972-1975). Izquierda: aspecto del robot serpeiente ACM. Derecha: ACM atravesando un camino tortuoso

1.1.3. Robótica Modular Reconfigurable

Mark Yim[13] se puede considerar como el Padre de esta disciplina. Ingeniero mecánico, realizó su tesis doctoral[15] en la Universidad de Stanford[14], entre 1992-1994. En esta tesis se introducen los conceptos de **robótica modular reconfigurable**. Además se realiza lo siguiente: taxonomía de las distintas formas de locomoción, diseño y construcción del robot **POLYPOD**[16][17], capaz de generar todas las clases de locomoción estáticamente estable , evaluación y comparación de los distintos modos de locomoción de Polypod y finalmente las limitaciones en cuanto al número de módulos.

Analizaremos las dos características de estos robots: **modularidad** y **reconfigurabilidad**

1.1.3.1. Modularidad

Característica de estar construido a partir de unos componentes estandarizados que se pueden intercambiar. Según el número de módulos distintos, el robot puede ser 1-modular, 2-modular, etc.

En el **Hirose & Yoneda Lab**[19] fueron pioneros en el estudio y construcción de robots tipo serpiente (*snake-like* robots)[20], que están constituidos por módulos similares. Un prototipo es el **ACM III** (Active Cord Mechanism)[21]. Está compuesto por 20 módulos iguales, cada uno de los cuales dispone de un servomecanismo para girar a derecha o izquierda con respecto al módulo siguiente (figura 1.6) . El contacto con el suelo se realiza mediante unas ruedecillas, que permiten que los módulos se deslicen hacia delante fácilmente, ofreciendo mucha resistencia al deslizamiento lateral, lo que permite que la serpiente se impulse hacia adelante. La cabeza describe un movimiento sinusoidal, que se va propagando hacia el resto de articulaciones a una velocidad constante. Las carácterísticas del ACM III son:

- **Velocidad**: 40cm/sec
- **Longitud**: 2m
- **Número de secciones**: 20

Se puede encontrar más información en [22, 23]

Otro prototipo es **OBLIX**[25] (figura 1.7), que se puede mover en las tres dimensiones utilizando articulaciones oblicuas (**oblique swivel joints**).

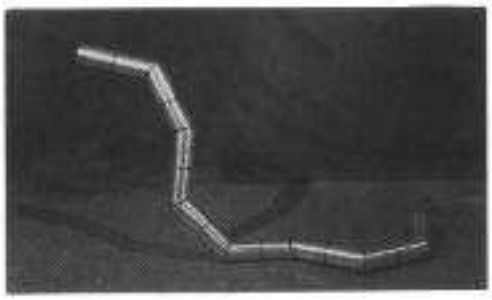


Figura 1.7: Robot serpiente OBLIX

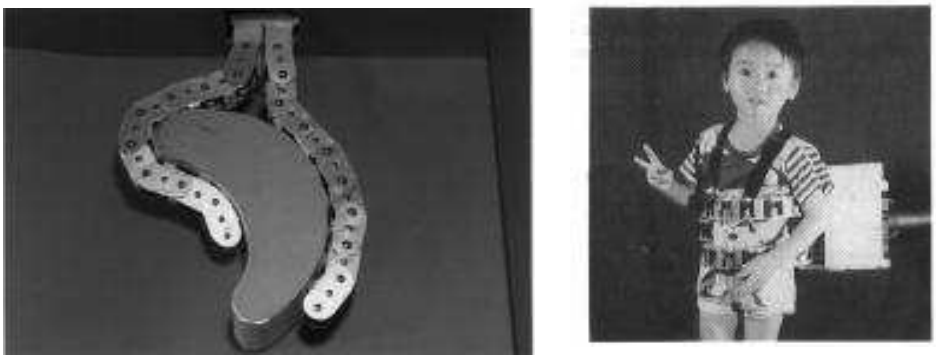


Figura 1.8: Izquierda: el manipulador SG agarrando un objeto. Derecha, SG levantando a un niño. La presión se reparte uniformemente por toda la superficie de contacto, por lo que no se provoca ningún daño

También han empleado robots serpiente para crear **manipuladores**, como el **SG**[24], constituidos también por módulos iguales. Se trata de una "pinza" o "agarrador mecánico" que adopta la forma del objeto que va a coger (fig 1.8), creando una fuerza que se reparte uniformemente por todo el objeto. El prototipo es capaz de levantar cuerpos humanos, sin que estos sufran ningún daño (fig 1.8). Si se utilizasen los manipuladores clásico, como pinzas, podrían hacer daño al humano en las zonas de contacto.

1.1.3.2. Reconfigurabilidad

Habilidad para combinar los componentes físicos del robot. Puede ser:

- **Dinámica**: El robot se autoreconfigura
- **Manual**: Otro agente reconfigura el robot (Por ejemplo un humano)

Un ejemplo de reconfiguración son los brazos Robots. El RMMS (Reconfigurable Modular Manipulator System)[26] es un brazo robot que es capaz de recalcular los parámetros dinámicos, cuando el usuario cambia los módulos de lugar (reconfiguración manual).

Un brazo robot industrial que disponga de diferentes manipuladores situados en una estantería, y que sea capaz de dejar uno y colocar otro (por ejemplo, dejar la cabeza soldadora y situar una para

pintar), sería un ejemplo de robot reconfigurable dinámicamente.

1.1.3.3. Características de los robots modulares reconfigurables

Estos robots presentan tres características importantes, que están siendo estudiadas y evaluadas[29]:

- **Versatilidad**: Al estar constituidos por módulos, pueden adoptar prácticamente cualquier forma por lo que se pueden emplear para múltiples tareas diferentes. Esta característica tiene especial importancia en la locomoción, puesto que permite que un robot pueda desplazarse por terrenos muy dispares.

- **Fiabilidad**: Debido a la alta redundancia (autoreparación). El sistema se va degradando poco a poco según se van fallando los módulos

- **Bajos coste**: n módulos diferentes: economía de escala.

1.1.3.4. Tipos de robots reconfigurables

Existen tres tipos de robots reconfigurables, según la manera en que cambian su forma[29]:

- **Robots tipo cadena**: Se unen y separan cadenas de módulos. Un ejemplo de este tipo de robots son **Polypod** y **polybot** (capítulo 2).

- **Robots tipo retículo**: Los módulos se mueven dentro de un retículo, en 3D. Similar a cómo se mueven los átomos en un cristal[27].

- **Autoreconfiguración móvil**: Los módulos se separan y se mueven independientemente hasta unirse a otro módulo en otra parte del robot. Son MUY COMPLEJOS y están poco estudiados.

1.2. Resumiendo

Las distintas soluciones dadas para resolver el **problema de la locomoción** en terrenos desconocidos y muy dispares no han sido satisfactorias. Es necesario robots mucho más versátiles. Esto dió lugar a que se empezase a trabajar en una nueva línea de investigación: los **robots modulares reconfigurables**. En vez de diseñar un robot específico para desplazarse por un determinado entorno, ¿por qué no construir un robot a base de **módulos sencillos** y además que se puedan **reconfigurar** cambiando la forma del robot para adaptarse a distintos tipos de terreno?

---

*El **diseño de robots ápodos** se puede abordar desde esta perspectiva. En vez de diseñar un robot ápodo específico, es mejor diseñar un módulo sencillo que se pueda unir formando cadenas. Este mismo módulo también servirá para construir otros robots diferentes.*

1.3. Objetivos

Este trabajo se encuadra dentro del campo de la robótica modular reconfigurable, con aplicación a la locomoción estáticamente estable, centrándose en el movimiento de los robots ápodos, que son un subconjunto de los robots tipo cadena

Los **objetivos** de este trabajo son:

1. Estudiar el origen y la evolución de los robots modulares reconfigurables
2. Diseñar y construir un módulo que sirva de base para la construcción de robots modulares
3. Rediseñar el robot CUBE 2.0 para orientarlo hacia la robótica modular: **Cube Reloaded**. Plataforma para poder investigar en Robótica Modular Reconfigurable
4. Adaptar el software de generación de movimiento a Cube Reloaded, aplicado al desplazamiento en línea recta
5. Estudiar nuevas posibilidades hardware para su control

1.4. Organización

En este capítulo hemos visto el encuadre del trabajo, haciendo un resumen de las ideas que hay detrás de la **robótica modular reconfigurable**. En el capítulo 2 estudiaremos dos ejemplos de estos robots, centrándonos en las características técnicas y las posibilidades que ofrecen, sobretodo en lo referente a la locomoción. Después presentaremos el robot **Cube Reloaded** (capítulo 3), un robot ápodo modular, inspirado en la generación I de Polybot (G1). Describiremos el trabajo previo, Cube 2.0, la mecánica y la nuevas alternativas propuesta para su control con FPGAs. En el capítulo 4 se presentan los experimentos realizados y sus resultaso. Finalmente se sacan conclusiones y se exponen los trabajos futuros.

Se ha intentado incluir en las referencias la mayor cantidad posible de URLs para facilitar la búsqueda de información.

Capítulo 2

Robótica modular reconfigurable: Polypod y Polybot

2.1. Introducción

En este capítulo estudiaremos la robótica modular reconfigurable analizndo dos robots concretos: Polypod y Polybot. Primero veremos la versatilidad que presentan en cuanto a la locomoción. Luego nos centraremos en Polypod, que se puede considerar como el primero de este tipo de robots, aunque todavía no es dinamicamente reconfigurable. Aparecen nuevas posibilidades de locomoción, lo que da lugar a una taxonomía de los distintos patrones de movimiento. Por último analizaremos todas las generaciones de Polybot y qué mejoras incorpora cada una.

2.2. Explorando la viabilidad y versatilidad

Mark Yim, en su tesis[15], construyó y diseñó el **robot Polypod**[16][17] para estudiar cuánto de versatil puede llegar a ser un sistema basado en **robótica modular reconfigurable**, restringido al ámbito de la **locomoción estáticamente estable**.

El escenario que consideró fue el siguiente, mostrado en la figura 2.1. En la foto de la izquierda Polypod está atravesando el suelo de madera de un porche, para lo que utiliza un movimiento tipo rueda (*loop*), caracterizado por su eficiencia. Al llegar al final, se transforma en un gusano, moviéndose mediante ondas longitudinales (movimiento similar al de las lombrices de tierra), lo que le permite pasar por debajo de la barandilla y superar el escalón, llegando a un terreno desigual y con maleza. Aquí se reconfigura en una araña de 4 patas, cuyo movimiento es más estable.

Se trata de un nuevo enfoque al problema de la locomoción. En vez de construir un robot diseñado específicamente para moverse por un terreno, se diseña un robot más versatil capaz de cambiar su forma y la manera de andar.

Aunque Polypod no es autoreconfigurable, sí demuestra la viabilidad de este tipo de robots, implementando diferentes patrones de movimiento.

Configuracion en RUEDA | Configuracion como gusano | Configuracion como araña

Figura 2.1: Aplicación de un robot reconfigurable. Según el terreno se reconfigura y utiliza una locomoción diferente

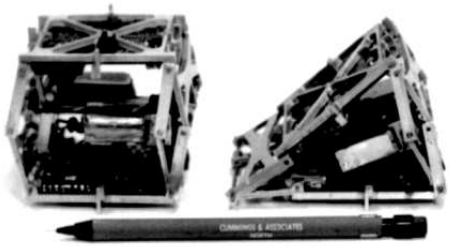


Figura 2.2: Los dos módulos de Polypod: un nodo y un segmento

2.3. Polypod

2.3.1. Los módulos

Se trata de un robot **2-modular**, con dos tipos de módulos diferentes (figura 2.2), denominados **nodos** y **segmentos**. Los **nodos** son cubos rígidos, a los que se pueden conectar otros módulos en cualquiera de sus seis caras. En ellos se sitúan las baterías y son los que permiten que se puedan construir robots con "bifurcaciones" (y no sólo cadenas de módulos, que es lo que se conseguiría si simplemente se unen segmentos). Un esquema de los segmentos se puede ver en la figura 2.3.

Los **segmentos** son módulos con 2 grados de libertad, y que sólo se pueden conectar a otros módulos a través de dos caras, formando "cadenas" de módulos. La combinación de nodos y segmentos permite construir robots más complejos, como arañas o bípedos. Los dos grados de libertado son los esquematizados en la figura 2.4. El módulo puede contraerse y expandirse y además puede cambiar el ángulo formado por dos de sus caras opuestas, comprendido entre -45 y 45 grados. Mediante dos motores de corriente continua se actúan sobre los dos grados de libertad.

2.3.2. Electrónica y sensores

El **sistema de interconexión** está constituido por las **placas de conexión** que tienen dos funciones:

- Enganchar unos módulos con otros (mecánicamente)

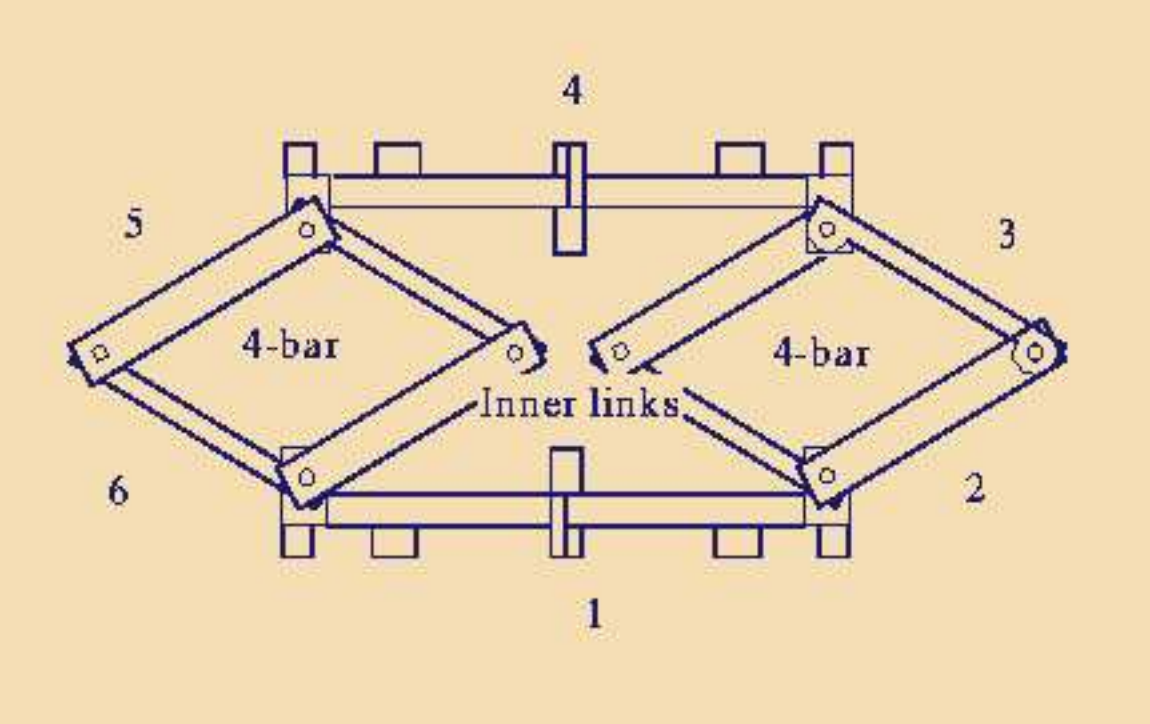


Figura 2.3: Esquema de un segmento de Polypod

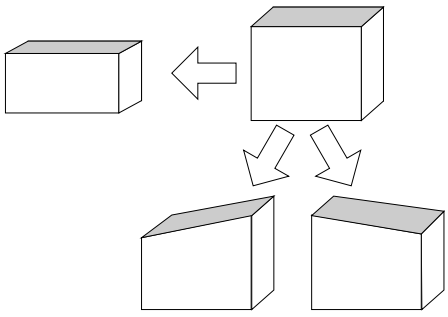


Figura 2.4: Los dos grados de libertad de un segmento de Polypod

- Conexiones eléctricas (10 líneas)
  - 4 líneas para la alimentación (2 para motores y 2 para electrónica)
  - 6 líneas para el Bus síncrono (SPI): 2 para reloj, 2 master in y 2 para master out

Los segmentos tiene dos placas de conexión, en caras opuestas, por lo que sólo se pueden conectar formando cadenas. Los nodos disponen de 6, uno en cada cara, por lo que se puede realizar "bifurcaciones" en la construcción del robot. Además incluyen las baterías.

Los sensores empleados son: 2 de posición (para cerrar el bucle de cada motor de los segmentos) y 2 pares emisores-detectores de infrarrojos usados para medir la proximidad, enganche y como bus de comunicaciones locales.

En cada módulo hay un **microcontrolador 68HC11E2** de Motorola.

2.3.3. Distintas formas de locomoción

En Polypod se han implementado y probado diferentes patrones de locomoción (*Gaits*) en línea recta, tanto simples como compuestos[18][15] (apartado 2.4). Todos ellos se caracterizan porque se pueden implementar mediante **tablas de control** (apartado 2.3.4).

- **Patrones simples** (ver figura 2.5)

  1. **Tipo rueda** (*Rolling-track*, *loop*). **RCB**. 16 segmentos. Cadena de 16 módulos en la que el primero se une con el último, formando una rueda. Este movimiento es el que presenta una **mayor eficiencia** en el aprovechamiento de la energía.
  2. **Slinky**. **RDB**. Cadena de módulos formando una "U" invertida. La parte trasera se eleva sobre la delantera, que permanece fija, y se sitúa delante, fomando una nueva "U" invertida. El movimiento es parecido a las palomas que realizan los gimnastas, pero mucho más lento y con el centro de gravedad cayendo siempre dentro de la base de apoyo.
  3. **Voltereta lateral** (*Cartwheel*). **RDB**. Se dispone de 4 miembros, de los cuales 3 están apoyados (en línea) y uno en el aire. El robot gira de manera que el que estaba en el aire pasar a ser punto de apoyo y el de apoyo más retrasdo pasa a estar al aire. Es parecido a las volteretas laterales de los gimnastas, pero más lento.
  4. **Lombriz**. **SCB**. Movimiento típico de las lombrices. Ondas longitudinales (de compresión-expansión) se propagan desde la cola hasta la cabeza. **Es el mejor patrón para atravesar obstáculos**.
  5. **Oruga** (*Carterpillar*). **SDB**. Tira de nodos con patas formadas por 2 segmentos. Mínimo tres nodos con tres patas para que ande.
  6. **Araña** (*Spider*). **SDL**. Araña con 5 patas. Es el más estable de todos.

- **Patrones compuestos** (ver figura 2.6)

  1. **cater-cater** (articulado). Es la combinación del patrón oruga, en un array de 2D. Es un ejemplo de una combinación articulada. Especialmente útil para transporte de carga.
  2. **Moonwalk** (jerárquico). Es la combinación de la rueda y la oruga (el propio autor lo califica como movimiento exótico)

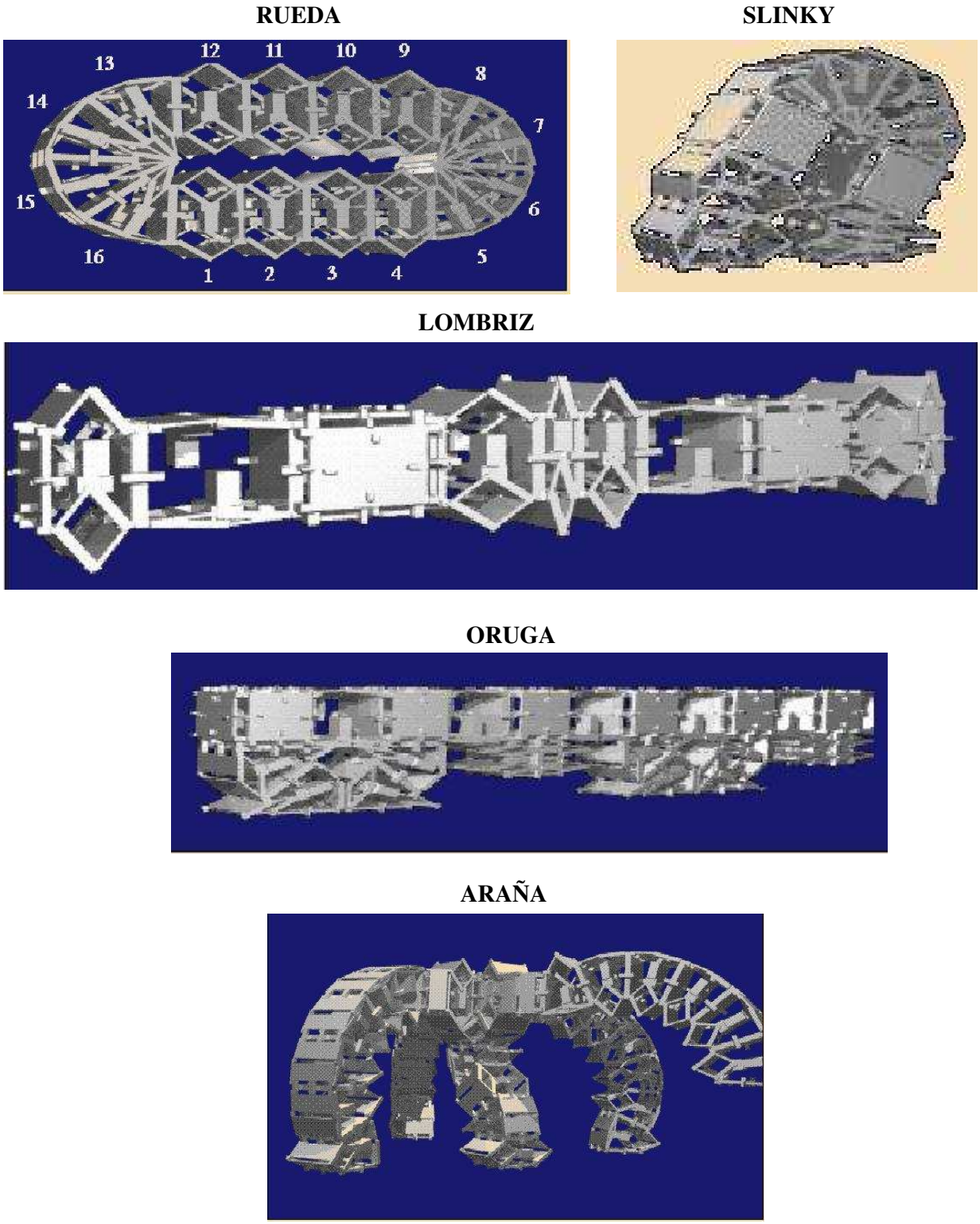


Figura 2.5: Algunos patrones simples de movimiento implementados en Polypod

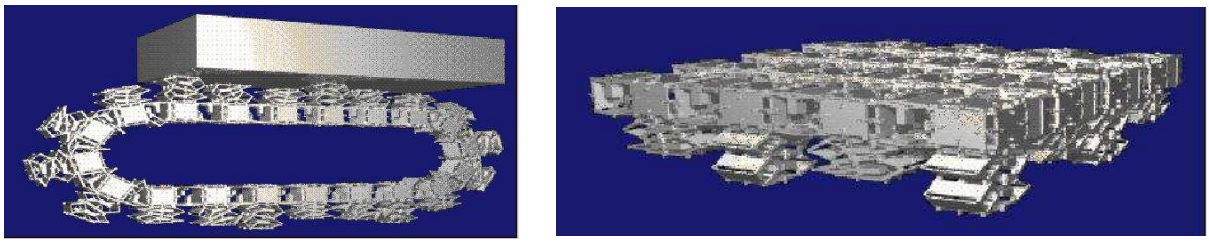


Figura 2.6: Algunos patrones compuestos implementados en Polypod. Izquierda: Monwalker, derecha: oruga-oruga (cater-cater)

2.3.4. Control

Según el número de grados de libertad (DOF) controlables, en relación con los DOF del robot, tenemos los siguientes tipos de robots:

- **no-holonómicos**: DOF controlables <DOF. Por ejemplo un coche.
- **Holonómicos**: DOF Controlables = DOF (ejemplo: brazo robot con 6 grados libertad).
- **Redundante**: DOF Controlables >DOF. (ejemplo: brazo robot con más de 6 articulaciones, un gusano robot). No sólo importa el punto final, sino la "forma" que tiene el robot.
- **Hiper-redundantes**: DOF Controlables > >DOF.

En general, **los robots modulares reconfigurables pertenecen a la categoría de Hiper-redundantes**. Por ello el elemento clave que se va buscando en el control es la **simplicidad** y también es muy importante la **escalabilidad** (y ser lo más independiente posible del número de módulos).

El control de Polypod se realiza a tres niveles:

1. **Nivel bajo**: Control PID de la posición de ambos motores de los segmentos. A este nivel se le pasa la posición deseada y éste actúa sobre sobre los motores.
2. **Nivel de comportamiento**. Se definen tres comportamientos para cada grado de libertad del módulo. Son:

   a) **Modo fin** (*end mode*): Moverse en un sentido u otro (+/-) hasta llegar al final. Si un segmento está en modo +*end*, se expanderá lo máximo posible e informará al nivel superior cuando termine. Si está en modo -*end* hará lo mismo pero contrayéndose. (Lo mismo para el otro DOF). Obsérvese que el nivel superior no especifica la posición, sino el comportamiento.

   b) **Modo muelle** (*spring mode*): Se mueve a una posición u otra en función de la fuerza/par medidos. Por ejemplo, si está en modo +*spring* y no se le aplica fuerza permanecerá expandido. Si se sitúa carga se contraerá un determinado valor (determinado por la constante k del muelle que se está simulando). El modo -*spring* se comporta de forma contrario a como lo hace un muelle. Lo mismo es válido para el otro grado de libertad.

   c) **Modo NO** (*no mode*). El módulo permanece en la misma posición.

3. **Nivel Alto**. **Tablas de control**, que definen la secuencia de comportamientos que se aplicarán a los módulos, así como el instante en el que pasar de una fila a otra. Por ejemplo, imaginemos un comportamiento muy sencillo. 4 segmentos unidos formando una cadena. Se quiere que se expandan y que luego se contraigan, todos a la vez. La tabla de control tendrá dos filas por 4 columnas. Se añade una quinta columna para indicar la condición de "disparo" que es la que determina cuando se pasa a ejecutar la siguiente fila. Una condición de disparo podría ser que todos los modulos se hayan expandido. Sólo se está utilizando el grado de libertad correspondiente a la contracción-expansión, por lo que el DOF restante de todos los módulos se pone en el *no mode*.

   En la primera fila, todos los módulos tendrían el comportamiento *+end* y en la segunda fila el *-end*. La tabla de control sería de esta manera:

	Segmentos				
Paso	1	2	3	4	**Condición de Disparo**
0	+end	+end	+end	+end	1,2,3,4
1	-end	-end	-end	-end	1,2,3,4

   El controlador de este nivel sólo tiene que ir a la primera fila, enviar el comportamiento al nivel inferior y esperar a que se cumpla la condición de disparo. A continuación se leen los valores de la siguiente fila y se repite la operación. Al llegar al final de la tabla se vuelve a comenzar.

Este sistema de control tan sencillo ha servido para implementar en Polypod todos los patrones de movimiento descritos en el apartado 2.3.3, salvo el de la araña.

2.3.5. Resumen

Se trata del primer robot modular, aunque **no es dinámicamente reconfigurable**. Capaz de implementar **muchos patrones de movimiento** (gaits) diferentes, por lo que demuestra la **viabilidad** de utilizar la robótica modular para resolver el problema de la locomoción. Asimismo, la **versatilidad** de estos robots queda también demostrada, dada la cantidad de patrones de movimiento que puede utilizar.

La estructura mecánica es compleja y los módulos tienen 2 grados de libertad. El control es muy sencillo utilizando **tablas de control**.

Se puede considerar como el robot que ha abierto el camino de la **robótica modular configurable**.

2.4. Taxonomía de los tipos de movimiento

No tiene sentido estudiar la locomoción estáticamente estable de un robot modular, puesto que tiene múltiples configuraciones. En estos robots la dificultad está en establecer los límites de lo que se puede hacer. El estudio de la locomoción hay que hacerlo en general, sin particularizar para ningún determinado robot.

La taxonomía de los distintos tipos de movimiento es necesaria para poder establecer las propiedades de una determinada familia y poder compararlos entre sí.

Mark Yim desarrolló en [15] una **taxonomía para clasificar los patrones de movimiento estáticamente estables y en línea recta, sobre un terreno genérico**. Posteriormente clasificó los movimientos existentes de diferentes robots, como el **Ambler** y el **Dante II**, así como todos los desarrollados para **Polypod**.

2.4.1. Definiciones

- ***Gait:*** Patrón de movimiento que se repite para conseguir la locomoción y que después de aplicarlo el robot se encuentra en el mismo estado, pero trasladado o rotado.

  Existen dos tipos de patrones, los **simples** que no se pueden descomponer en otros más básicos, y los **compuestos** que están formados por distintas combinaciones de otros patrones más básicos. Un ejemplo de patrón compuesto podría ser un patinador, que tiene dos piernas que se mueven (un patrón) pero en los pies hay ruedas que giran (otro patrón).

- ***G-foot:*** (G-pie). Conjunto contiguo de puntos que entran en contacto con el suelo. Por ejemplo, un vehículo con 3 ruedas tiene 3 G-pies, cada rueda es uno de ellos. Un gusano tiene un único G-pie.

- **Cuerpo**: Las partes del robot que no son G-pies se consideran que forman parte del cuerpo.

Estas definiciones tan extrañas se tienen que introducir puesto que se está haciendo una clasificación general. No se sabe qué configuraciones puede adoptar un robot modular, pero se tienen que poder clasificar sus movimientos dentro de esta taxonomía.

2.4.2. Patrones simples

Para cada patrón de moviento simple se caracteriza mediante tres componentes binarias ortogonales. A cada valor de las componentes se le asigna una letra.

1. **R-S** (***R****oll*-***S****wing*). Esta componente describe si un G-pie rota (*Roll*) o bien realiza un movimiento de vaivén (*Swing*). En general, siempre que ningún G-pie rote se consirará que es **S**.

   Cualquier vehículo con ruedas tiene la componente R, puesto que las ruedas (sus G-pies) rotan respecto a un eje. Lo mismo ocurre con los vehículos tipo tanque, que tiene orugas. También es de tipo R el movimiento de rueda (*loop*) que puede realizar Polypod, ya que en ese caso el robot está constituido por un único G-pie que rota.

   Un robot tipo perro se clasifica como S, al igual que el movimiento de un gusano (el único G-pie que tiene no rota).

2. **C**-**D** (**C**ontinuo-**D**iscreto). Determina cómo es el contacto que realizan los G-pies con el suelo. Si nunca se pierde el contacto con el suele, es de tipo Continuo (**C**). Si en algunos instantes sí se pierde, es de tipo Discreto (**D**).

   Un vehículo con ruedas es de tipo C, puesto que las rudas nunca pierden contacto con el suelo (o al menos voluntariamente). Un robot gusano también es de tipo C, puesto que su único G-pie simpre está en contacto con el suelo. Un robot araña es de tipo D, ya que las patas se levantan para llegar a la nueva posición (pierden contacto con el suelo).

3. **B**-**L** (**B**ig, **L**ittle). Esta componente tiene que ver con el "tamaño" del pie. Siempre que al menos sean necesarios tres G-pies para conseguir la estabilidad (y que el GC caiga dentro del polígoono de apoyo) se dicen que es de tipo **L** (Little, pies pequeños). Si sólo son necesarios 2 o más puntos de apoyo de un G-pie, se dice que es de tipo **B** (Big, pies grandes)

   Un coche con cuatro ruedas es de tipo L, porque necesita que al menos tres estén apoyadas. Un gusano es de tipo B porque sólo necesita que haya al menos dos puntos de su cuerpo apoyados en el suelo (tiene un único G-pie).

Teniendo en cuenta estas componentes, se pueden clasificar todos los movimientos que Polypod es capaz de realizar. Así por ejemplo, el movimiento de tipo rueda (*loop*) pertenece a la familia **RCB**.

Los robots gusanos que se mueven utilizando ondas que recorren su cuerpo pertenecen a la familia SCB**.**

2.4.3. Patrones compuestos

Los patrones simples se pueden mezclar de tres maneras diferentes para conseguir otros movimientos más complejos:

- **Articulados**: Unión por medio de articulaciones. Cualquiera dos patrones se pueden componer usando articulaciones. Un ejemplo típico con las cadenas de vehículos (varios vehículos unidos, como si fuesen vagones de un tren). Otro ejemplo es el patron oruga-oruga (u oruga 2D) de Polypod.
- **Jerarquías**. Un patron se sitúa en la parte superior y otro en la inferior, en contacto con el suelo. El ejemplo típico es el del patinador. En general, los patrones así generados suman las velocidades.
- **Morfológica**. Combinación de patrones en diferentes direcciones. Tiene la propiedad de que se pueden mover en diferentes direcciones. Un ejemplo sería un robot esfera, constituido por varios patrones de tipo rueda.

2.4.4. Análisis

En general, los patrones con componente **R** con **menos complejos** de construir. Los que tienen componente **C** suelen realizar **movimientos más suaves** en terrenos planos y pueden ser **más rápidos**. Los que tienen componente **L**, la **planificación es más sencilla** puesto que sólo hay que encontrar puntos de apoyo para poder formar el polígono.

En cuanto a los patrones compuestos, los que se combinan usando **articulaciones** son **mejores para cruzar obstáculos**, los j**erárquicos suman sus velocidades** y los **morfológicos** permiten **movimientos en cualquier dirección del plano**.

2.5. Polybot

2.5.1. Introducción

Polybot[30][31] es uno de los robots reconfigurables más versátiles de todos. Puede desde mover una caja hasta montar en triciclo. Existen tres generaciones de módulos, **G1**, **G2** y **G3** (Dentro de la G1 hay algunas versiones a tener en cuenta, como la **G1v4**).

Polybot utiliza **tablas de control** para el movimiento. Se trata de un control en bucle abierto, pero es suficiente para conseguir movimiento de tipo serpiente, rueda y araña. Normalmente un módulo contiene el conjunto de las tablas de control que se descargan a los otros, según se necesiten (control centralizado). Esto tiene sus limitaciones, ya que no se pueden almacenar todos los posibles movimientos, que crecen exponencialmente con el número de módulos. No se explota toda la versatilidad que se podría tener.

A continuación veremos los detalles de las tres generaciones y al final presentaremos un resumen con las ideas clave introducidas en cada generación.

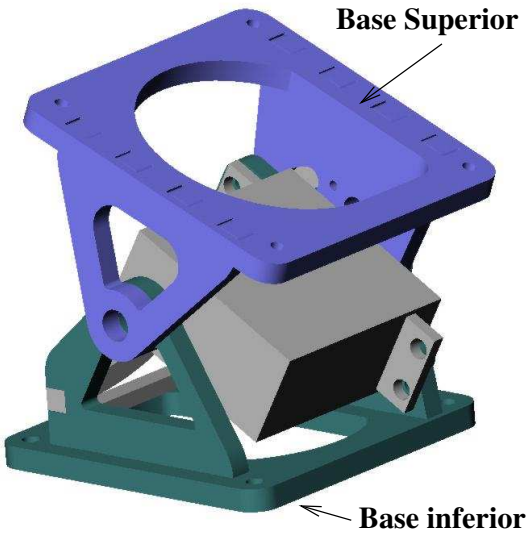


Figura 2.7: Aspecto de los módulos G1

2.5.2. Generación 1 (G1)

Todavía no se puede realizar reconfiguración compleja con ellos, es necesario unirlos mediante tornillos. No tienen la capacidad para poderse unir dinámicamente. Sin embargo sí que ha conseguido implementar una reconfiguración sencilla (ver apartado 2.5.4)

2.5.2.1. Mecánica

Estos módulos contienen las ideas básicas que se repiten en todas las generaciones. Representan un gran avance respecto de los módulos de Polypod, ya que son más pequeños (cubos de 5cm de arista) y **sólo hay un grado de libertad por cada módulo** (ver figura 2.7 y 2.8). Está construido a partir de piezas de plástico y un **servomecanismo de radio control**, que está atornillado a la pieza inferior. La pieza superior está sujeta a la corona del servo por un lado y a un falso eje por el otro. Tanto la pieza superior como la inferior tienen bases, que llamaremos **base superior** y **base inferior**, que sirven para la unión entre módulos.

Este módulo tiene una propiedad fundamental: tanto la base inferior como la superior son cuadradas y esto permite que los módulos se puedan conectar con orientaciones diferentes.

Si todos los módulos se conectan con la misma orientación, se obtendrá una cadena en la que todos sus puntos permanecerán dentro del mismo plano pero si un módulo se conecta rotado 90 grados con respecto al anterior, los puntos ya no pertenecen al mismo plano. En el primer caso diremos que los módulos están conectados **en fase**, y en el segundo que están conectados **desfasados**. Esto permite construir robots ápodos que se pueden mover por un plano y no sólo que avancen en línea recta.

Las piezas, al ser de plástico, son muy fáciles de cortar y mecanizar (corte por láser). Los servos son baratos, fáciles de encontrar y comprar.

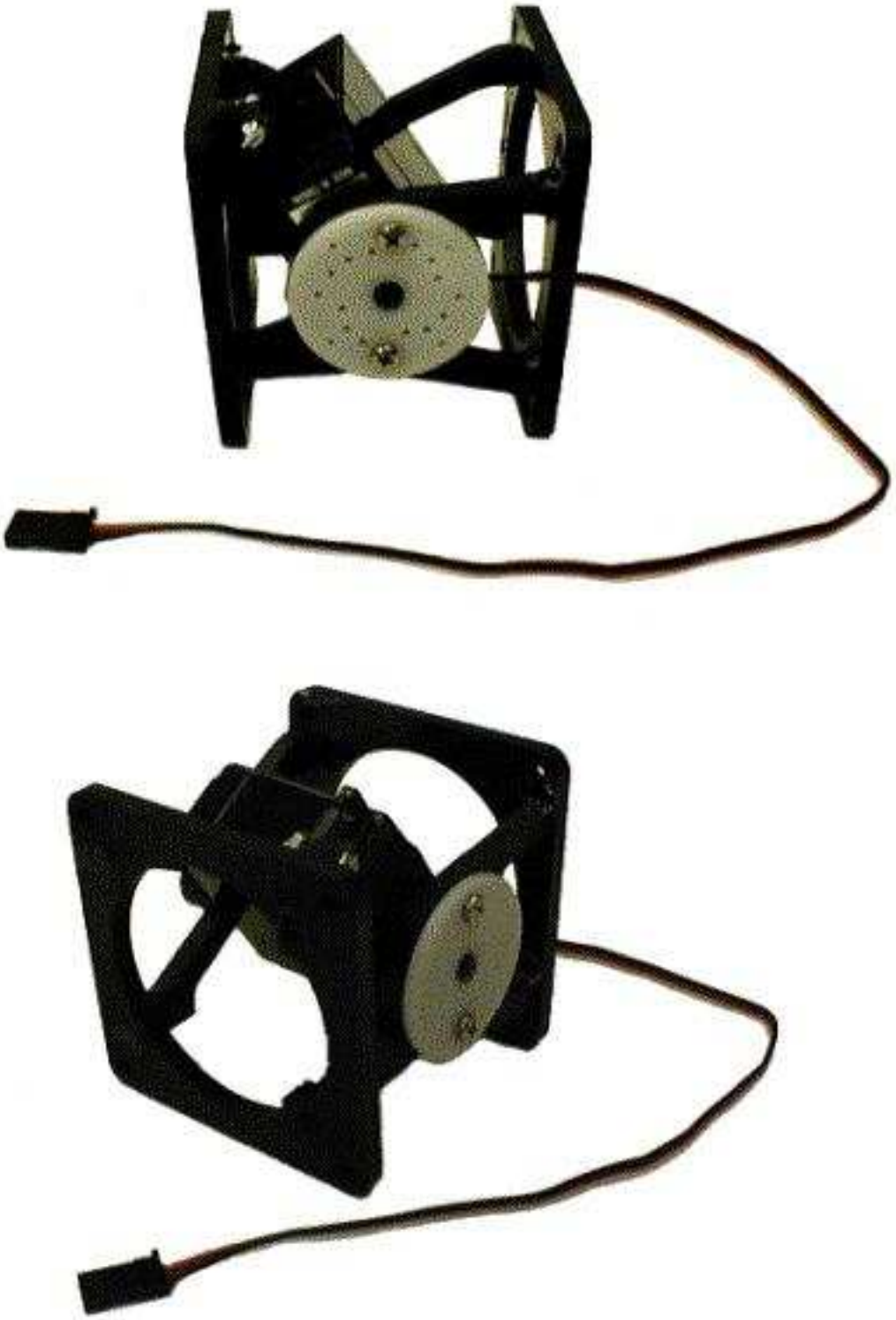


Figura 2.8: Fotos del módulo G1 visto desde distintos ángulos

Figura 2.9: Fotos de la primera reconfiguración dinámica de **Polybot G1**. En la foto de la izquierda, se está desplazando como una rueda, en la del centro se ha convertido en un gusano que es capaz de descender por unas escales y en la de la derecha se está introduciendo en un agujero.

2.5.3. Electrónica, sensores y actuadores

Estos módulos llevan **servos de radio control**, controlados con **señales PWM**, en **bucle abierto**. No llevan ningún tipo de sensor. Tanto la electrónica como la alimentación está situada fuera del robot. Como CPU utilizan un **68HC11E2**, de motorola.

2.5.4. Experimentos

En 1997 se realizó el primero ejemplo de reconfiguración simple para locomoción[32]. Polybot, formado por 12 módulos G1 en configuración de rueda, inicialmente avanza en línea recta. A continuación la rueda se abre y se convierte en un gusano que se desplaza mediante ondas sinusoidales, capaz de bajar por unas escaleras (ver fotos en la figura 2.9).

Otros experimentos interesantes realizados[32]:

- **Locomoción a través de tubos**. Esta es una característica muy interesante de los gusanos robots: el poderse desplazar por zonas donde otros no pueden llegar (figura 2.10).
- **Giros**. Conectando módulos G1 desfasados y alternados, se puede conseguir que el gusano gire (ver figura 2.10).
- **Araña de 4 patas**(ver figura 2.10)

Un experimento muy interesante que se puede realizar con polybot es la **manipulación distribuida**[33], lo que nos permite hacernos una idea de la versatilidad de este tipo de robots. Consiste en colocar varios módulos con distintos DOF formando un **array bidimensional**. Esta estructura es capaz de rotar y desplazar objetos situados en su superficie. Se han realizado varias pruebas, cuyas fotos se pueden ver en la figura 2.11:

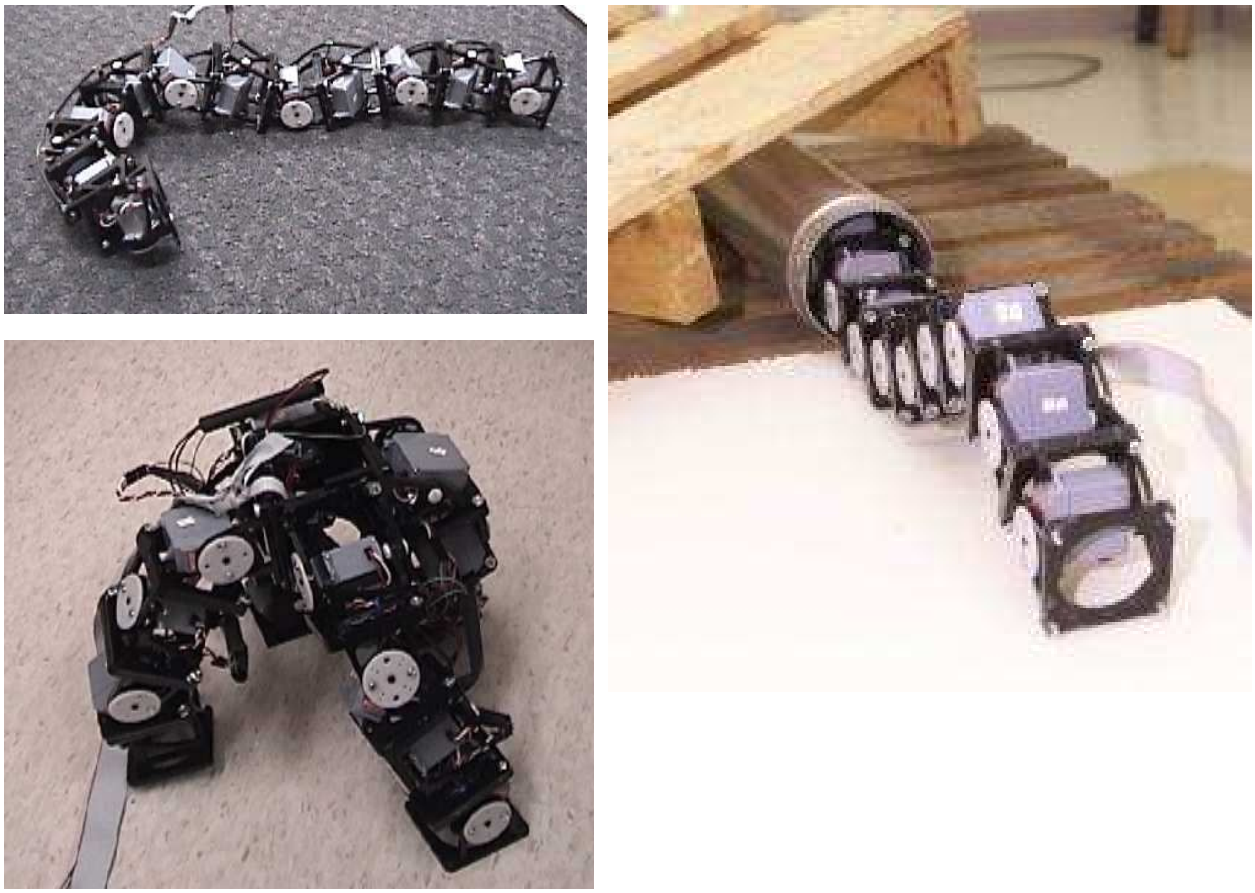


Figura 2.10: Algunos experimentos con Polybot G1. Arriba a la izquierda, se ha construido un gusano con módulos desfasados que puede girar. A la derecha un gusano está atravesando un tubo. Abajo a la izquierda los módulos forman una araña de 4 patas.

Figura 2.11: Ejemplos de polybot G1 usado para manipular objetos. En la foto de la izquierda está desplazando una hoja. En la central está rotando una hoja y en la de al derecha están pasando una bola de un lado a otro

- Array de 4x4 módulos de 1 DOF, capaz de desplazar una hoja de papel y de rotar una caja.
- Array de 3x3 módulos de 2 DOF
- Array de 2x2 módulos de 3 DOF, capaz de pasarse una pelota de unos a otros.

2.5.5. Generación G1v4

Todavía no son autoreconfigurables, pero son muy fácilmente ensamblables a mano (no hay que atornillar). Por las conexiones se pasa la alimentación y las líneas de comunicaciones. Tanto la alimentación como la electrónica se encuentran dentro del propio módulo, por lo que son un poco más grandes que los G1 (ver figura 2.12). Se siguen utilizando **servos de RC**, pero ahora se manejan en **bucle cerrado**. Además de leer el potenciómetro del servo, tiene sensores de fuerza. El objetivo es el estudio de la locomoción.

Los módulos tienen **4 placas de conexión** en cada módulo, lo que permite conectarlo con otros cuatro módulos. O bien se pueden enganchar para formar una cadena y utilizar las placas restantes para conectar elementos pasivos, como pies o ganchos.

Como alimentación utilizan se utilizan 6v proporcionados por pilas AAA recargables (para cada módulo). 10 módulos conectados formando una rueda (patrón de movimiento más eficiente), fueron capaces de recorrer 0.5Km en 45 minutos.

Cada módulo dispone de un microcontrolador **PIC16F877** y se comunican entre sí mediante un bus serie RS232 (o 485).

Todas las aplicaciones de estos módulos se han orientado hacia el estudio de la locomoción. Los experimentos realizados son los siguientes:

- **Montar en triciclo**. Este uno de los experimentos más interesantes y fue portada la revista IEEE Spectrum[29]. Es un ejemplo de cómo los robots modulares se pueden utilizar para manipular objetos humanos. Un total de 20 módulos G1v4 se unieron formando dos piernas y una cintura y se apoyaron en los pedales de un triciclo. El movimiento coordinado de estas piernas hace que el triciclo avance, con el robot encima (figura 2.13).
- **Adaptación al terreno**: Al disponer de sensores de fuerza, los módulos pueden detectar la presión ejercida sobre la superficie por la que avanzan, pudiéndo adoptar su misma forma. Los experimentos se han realizado con una configuración tipo rueda, de 10 módulos y los resultados

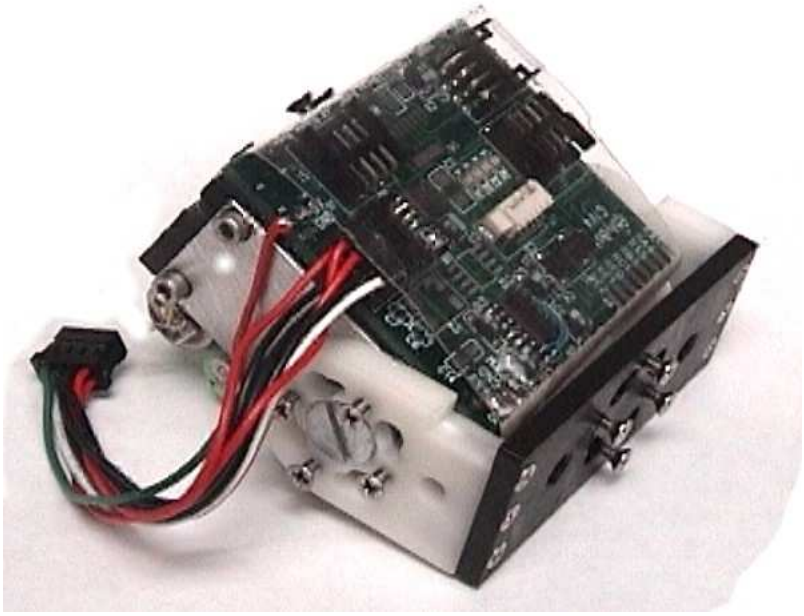


Figura 2.12: aspecto de un módulos G1v4

Figura 2.13: El experimento del triciclo, movido por módulos de tipo G1v4

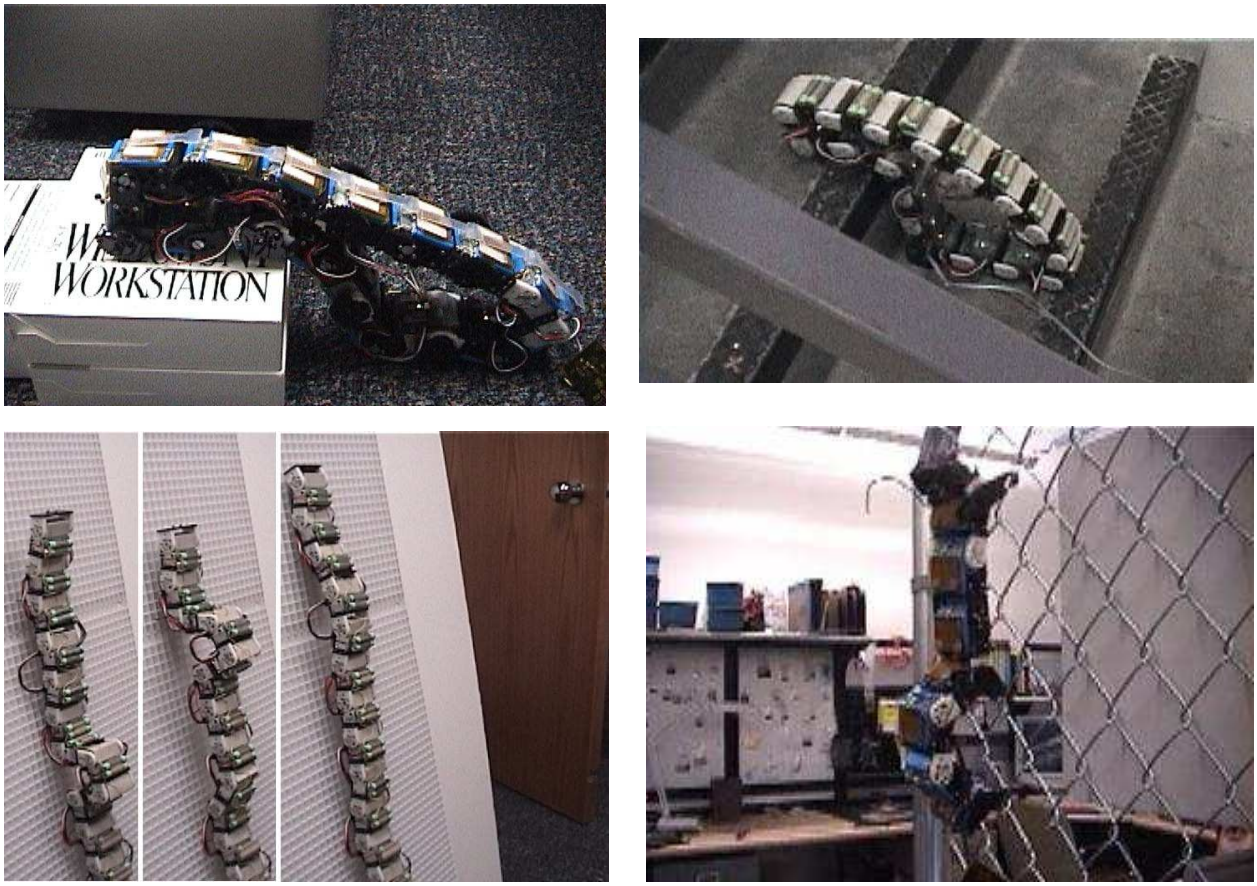


Figura 2.14: Esperimentos con los módulos G1v4. En la foto superior izquierda, el robot con forma de rueda se adapta a la forma del terreno para atravesarlo. Arriba a la derecha el mismo robot subiendo una escalera. En las fotos de abajo está trepando, en la izquierda sobre un material poroso y en la derecha sobre una verja.

han sido impresionantes. Es capaz de atravesar obstáculos de lo más diverso así como subir escaleras (ver figura 2.14)

- **Trepar** : Otra interesante aplicación es la habilidad para trepar por vayas y meteriales porosos, usando bien unos ganchos o bien unos pinchos. Mientras una parte del gusano permanece enganchada, otra parte se se sitúa en la nueva posición, generándose una onda que se propaga de la cola hasta la cabeza (ver figura 2.14). Se puede encontrar información en [34].

2.5.6. Generación G2

Estos son los primeros módulos que son autoreconfigurables, capaces de unirse y soltarse. Al igual que en polybot, hay dos tipos de módulos: nodos y segmentos. Tienen un tamaño de 11x7x6 cmm y utilizan un motor de corriente contínua sin escobillas en lugar de un servomecanismos como en los anteriores (ver figura 2.15). El rango de giro es de -90 a 90 grados. La estructura es metálica, mucho más robusta.

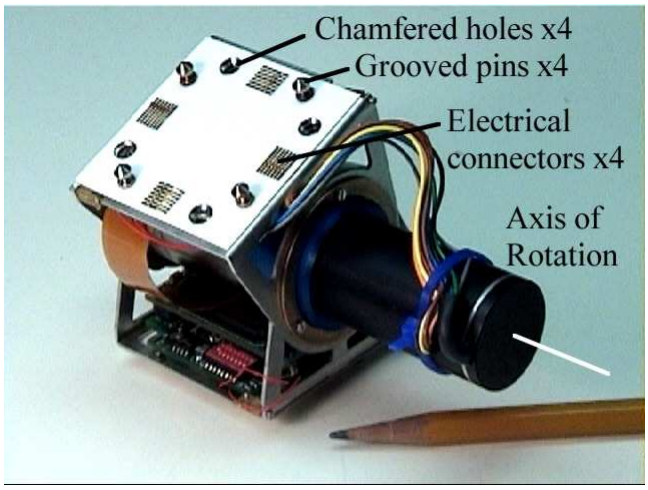


Figura 2.15: Aspecto de los módulos G2

Dispone de **dos placas de conexión** iguales, en caras opuestas para la conexión de unos módulos con otros. Por los conectores metálicos va la alimentación y las comunicaciones. La conexión se puede realizar en fase o desfasada y es del tipo "enganchar y listo", no hay que atornillar nada. El propio robot puede enganchar y desenganchar los módulos.

Cada módulo dispone de un **PowerPc 555** de motorola mas **1MB de RAM**, lo que le confiere una gran capacidad de cálculo. No se está utilizando toda esta capacidad, pero se reserva para futuros experimentos.

Todos los módulos se comunican a través del **BUS CAN**, lo que permite que el sistema sea más distribuido y más fiable.

Se han realizado experimentos en el campo de la locomoción, configurando el robot como un gusano, una araña y desplazándose tipo rueda. Pero el experimento más interesante es el de la configuración dinámica[35], en el que Polybot, desplazándose inicialmente como una rueda de 12 módulos , se convierte en un gusano, moviéndose mediante ondas sinusoidales y finalmente se transforma en una araña, en dos fases. Primero la cabeza y la cola del gusano se enganchan al módulo central (que es un nodo) adquierindo una forma de "8". Dos pares de módulos opuestos se sultan, tomando una forma de "X". Las patas se apoyan en el suelo y el bicho se levanta (ver figura 2.16).

2.5.7. Generación G3

Esta es la última generación en la que están trabajando. Su aspecto se puede ver en la figura 2.17. Estos módulos son más pequeños, con unas dimensiones de 50x50x45mm. Los motores son Maxon y mucho más pequeños. Siguen disponiendo de **dos placas de conexión**, pero incorporan muchos más sensores: de posición, acelerómetros, 4 IR para comunicaciones locales y sensores de fuerza. El procesador sigue siendo el mismo que el de los G2: un motorola **PowerPC 555** y un bus de comunicaciones tipo **CAN**.

2.5.8. Resumen de Polybot

Polybot es el robot modular más versátil. Actualmente se está trabajando en la tercera generación G3. En cada generación se han ido incluyendo una serie de características nuevas y se han realizado

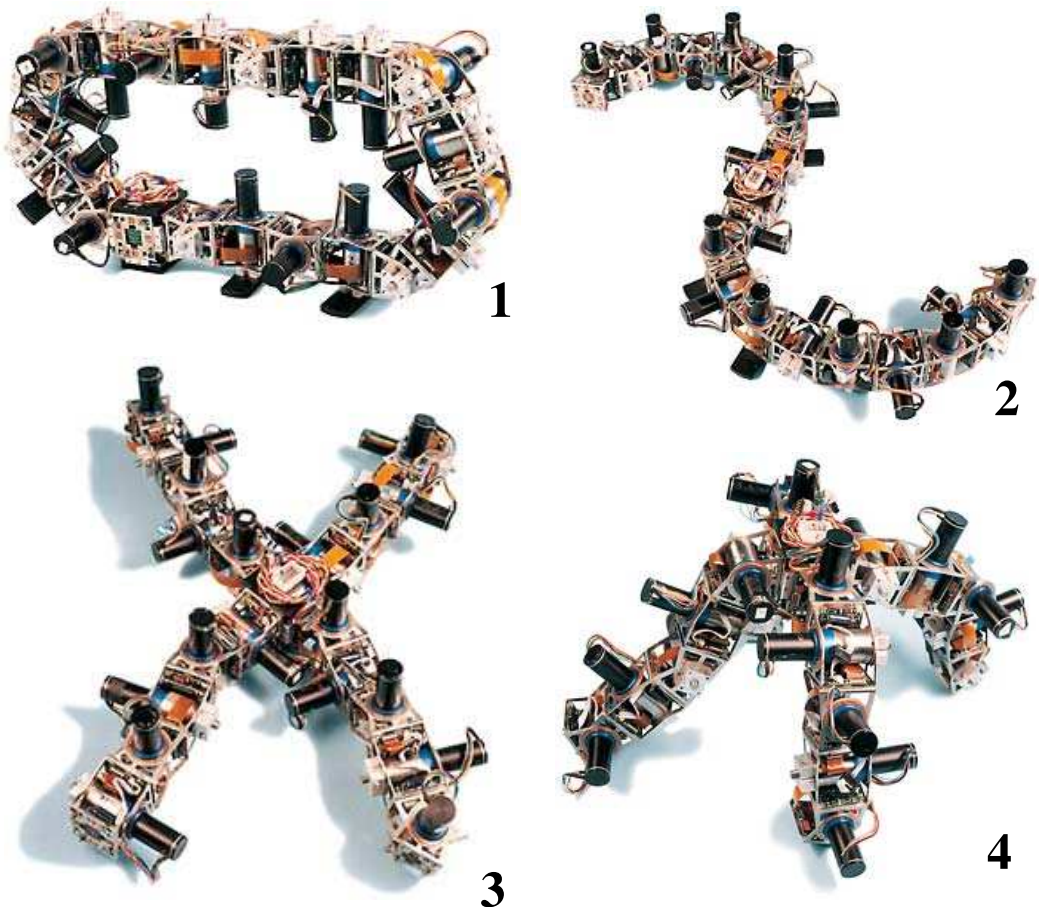


Figura 2.16: Ejemplo de autorreconfiguración probado con Polybot G2. (1) El robot avanza tipo rueda. (2) Se convierte en una serpiente y continúa avanzando. (3) La cabeza y la cola se unen al nodo central y dos pares demódulos opuestos se sueltan, formando una X. (4) El robot se levanta, ahora se ha convertido en una araña de 4 ptas.

Figura 2.17: Aspecto de los módulos G3

unos experimentos. A continuación se resumen las ideas fundamentales

- **G1**.
  - Control en bucle abierto de servos de RC
  - Se introduce la idea de conexión de módulos en **fase-desfase**
  - Piezas de plástico
  - Demostrada y probada versatilidad
  - Electrónica y alimentación fuera del robot
- **G1v4**
  - Control lazo cerrado: sensores fuerza: adaptación terreno
  - Alimentación + electrónica dentro del módulo
- **G2**
  - Motor DC
  - Estructura mecánica
  - autorreconfigurable
  - PowerPC
  - CAN BUS
- **G3**

- Más sensores
- Motor más pequeño, lo que permite una estructura más pequeña

Capítulo 3

El robot ápodo Cube Reloaded

3.1. Introducción

En este capítulo se describe el robot modular **Cube Reloaded**, construido a partir de módulos iguales y que utiliza un patrón de movimiento generado a partir de ondas sinusoidales.

Primero resumiremos el trabajo previo, **Cube 2.0**, del que se reutilizan bastantes partes. Después presentamores los **módulos Y**, diseñados específicamente como plataforma para robótica modular, que permiten construir diferentes robots. Uniendo cuatro de estos módulos en fase tenemos a Cube Reloaded, cuya mecánica y electrónica se discuten en los siguientes apartado. Se mostrarán las tres alternativas para su control, basado en **tablas de control**. Una alternativa usando un **microprocesador específico**, y otras dos empleando **FPGAs**.

3.2. Trabajo previo: CUBE 2.0

Cube Reloaded está basado en el robot ápodo Cube 2.0[5](ver figura 3.1), trabajo realizado previamente, que se encuentra disponible en línea[6]. El objetivo de este trabajo era poder responder a la siguiente pregunta:

¿Es posible construir un robot que no tenga patas y que se pueda mover?

Con **Cube 2.0** se demostró la viabilidad de ello, construyendo un robot gusano, con 4 servos, capaz de avanzar en línea recta utilizando ondas sinusoidales que recorren su cuerpo desde la cola hasta la cabeza. Asímismo se desarrollaron las herramientas software necesarias para poder **generar las secuencias de movimiento automáticamente**, a partir de los parámetros amplitud y longitud de onda de una función sinusoidal.

3.2.1. Mecánica

Está constituido por **4 módulos** (ver figura 3.2). En cada uno de ellos hay un servo Futaba 3003 con un doble eje, ambos unidos mediante unas varillas roscadas de 3.8mm. Los módulos se unen mediante unas piezas de metacrilato.

El gusano tiene tres partes: una cabeza que aloja la electrónica, un cuerpo constituido por la unicón en cadena de 4 módulos y una cola La cabeza está formada por una pieza para alojar la electrónica y en la cola.

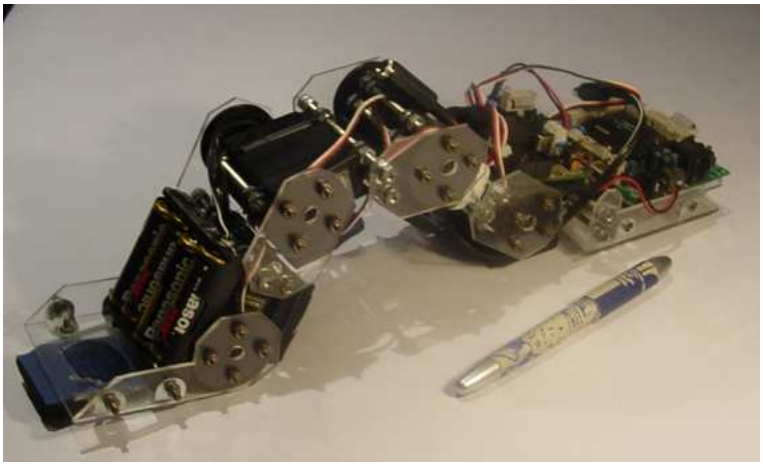


Figura 3.1: El robot ápodo CUBE 2.0

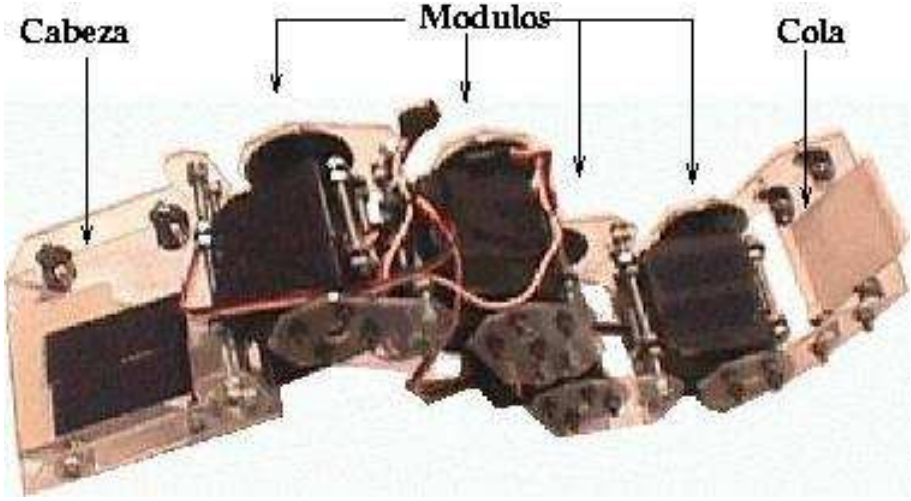


Figura 3.2: Estructura mecánica de Cube 2.0

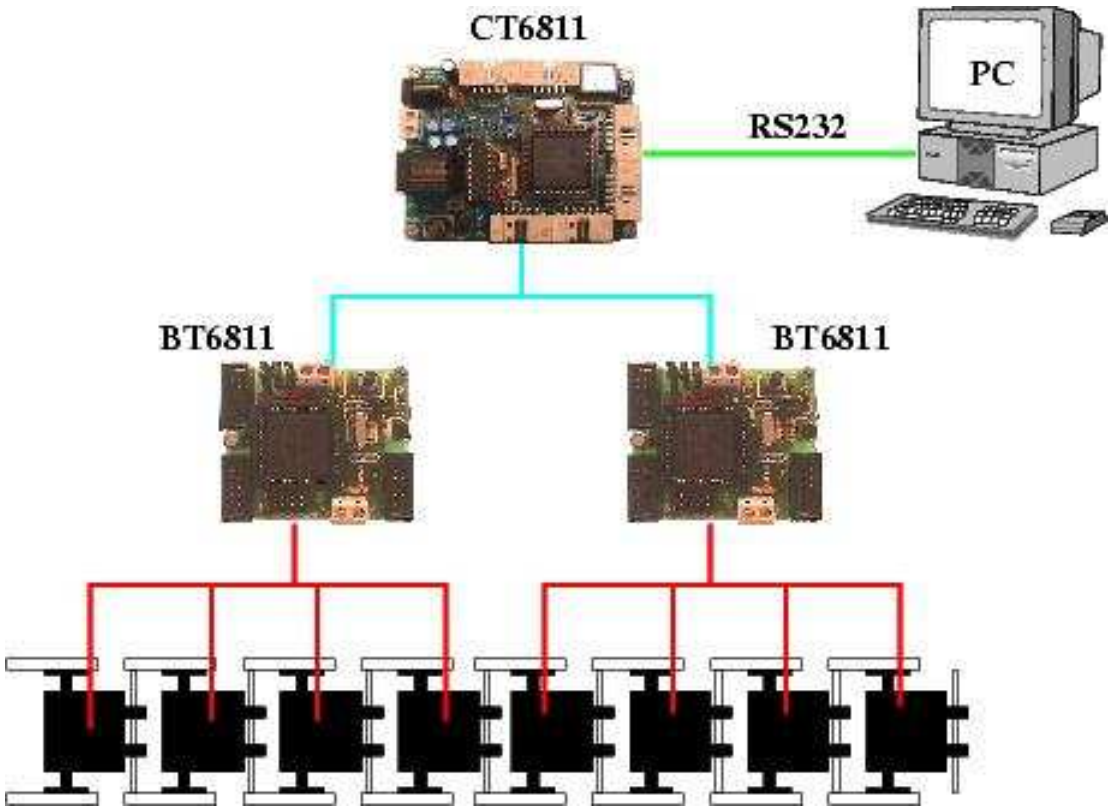


Figura 3.3: La electrónica de control de Cube 2.0

El sistema es mecánicamente expandible, en ese sentido es un robot modular. Sin embargo los módulos sólo se pueden unir en fase. Para hacer que pueda girar hay que re-hacer por completo la mecánica.

Además, las varillas roscadas hacen que los módulos sean más pesados y la estructura de doble eje, aunque es muy robusta, es complicada de construir. La pieza empleada para el doble eje es la parte superior de otro servo, que se tiene que encargar por separado en la tienda de aeromodelismo, y que no siempre se consigue.

3.2.2. Eletrónica

Para el **control** de Cube 2.0 se utiliza la red de microcontroladores diseñada para Puchobot[3], basada en el microcontrolador 68HC11E2, usando el BUS **SPI**. Las **tarjetas BT6811**[38] son las encargadas de posicionar los servos, 4 cada una, según los comandos recibidos por el **SPI**. Son los **nodos esclavos** de la red. Cube 2.0 sólo tiene 4 servos, por lo que sólo necesita una BT6811. Sin embargo, se diseñó con esta red de microcontroladores para poderse ampliar (ver figura 3.3). La **tarjeta CT6811**[39] es la maestra de la red y contiene la **tabla de control** con las posiciones en las que se deben posicionar los servos.

Aunque el sistema se puede dejar autónomo, normalmente se utiliza en conexión con el PC. En este caso la tarjeta CT6811 hace de interfaz entre el puerto serie del PC y el SPI. Desde el PC se envían las posiciones de los servos.

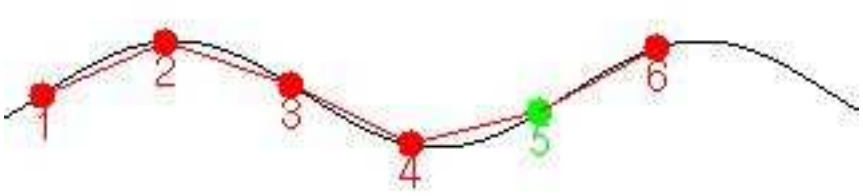


Figura 3.4: Las tablas de control se calculan usando un gusano virtual que se ajusta a una función de onda

Se utilizan alimentaciones separadas para la electrónica y los servos. Con 4 pilas AA se alimenta la electrónica (Tarjetas BT6811 y CT6811) y con una fuente de alimentación externa los servos

3.2.3. Control

Existen varios niveles de control:

- **Nivel de servo**: El posicionamiento de los servos lo realiza la tarjeta BT6811, generando la **señal PWM** en función de la posición deseada. En ellas se está ejecutando un servidor que realiza esta tarea.
- **Tablas de control**: En el siguiente nivel se recorren las tablas de control, similares a las de **Polypod** y **Polybot**. Las entradas de la tablase envían consecutivamente al servo. Es un controlador muy sencillo, en el que el tiempo que transcurre desde que se envía una entrada hasta que hay que enviar la siguiente se estima.

  Las tablas de control se almacenan en un fichero y mediante la aplicación **cube-físico**[44] se cargan y envían al gusano. También es posible posicionar los servos desde este programa, usando unas barras de desplazamiento, lo que permie realizar pruebas mecánicas y generar tablas de control (secuencias de movimiento) manualmente, guardándolas en un fichero.
- **Generación automática de secuencias**: Las tablas de control para mover el gusano se pueden generar usando la aplicación **cube-virtual**[45], en la que se especifica la amplitud y la longid de onda que se quiere para realizar el movimiento (ver figura 3.4). Esta tabla se almacena en un fichero y puede ser reproducida por el programa *cube-fisico*. Sólo se generan secuencias para un gusano de 4 articulaciones.

3.2.4. Limitaciones

Cube 2.0 ha cumplido su cometido y ha servido para desarrollar software y demostrar la viabilidad de que un gusano se mueva utilizando ondas sinusoidales. Sin embargo **no es válido como plataforma para nuevos desarrollos**, puesto que sólo se podrían construir gusanos más largos, o como mucho hacer cadenas de módulos que se mueven en el mismo plano.

Para trabajar con **robótica modular reconfigurable** y profundizar en el estudio de los patrones de movimiento es necesario tener unos módulos más versátiles.

3.3. Módulos Y1

Estos módulos están inspirados en los de primera generación de **Polybot** (**G1**)2.5.2. En particular se ha adoptado la idea de la **conexión en fase** o **en desfase**, una idea brillante. Los módulos se han diseñado desde cero, adaptados a los servos Futaba 3003. La información proporcionada por el **PARC** es descriptiva (fotos, vídeos) y en los *papers* se comentan los experimentos realizados y los resultados obtenidos, sin embargo **no están disponibles los planos ni información necesaria para poder reproducirlos**.

Puesto que los **módulos G1** fueron desarrollados por **Yim Mark**, se utiliza la primera letra de su nombre para bautizar estos módulos. El '1' indica que se trata de la primera generación.

3.3.1. Criterios de diseño

Los criterios para el diseño de los módulos Y1 han sido los siguientes:

- **Sencillez**. Tienen que ser lo más sencillos y simples posibles, con el menor número posible de piezas, para que se puedan construir prototipos a mano.
- **Fabricables**. Se tienen que poder fabricar industrialmente, por lo que se deben realizar unos planos detallados.
- **Conexión en fase y en desfase**. Esto permite que se puedan conectar los módulos de manera que todos pertenezcan al mismo plano o no. Idea tomada de los **módulos G1** de **Polybot** (ver apartado 2.5.2.1).
- **Facilitar cableado**. Que los cables puedan pasar de un módulo a otro.
- **Abiertos**. Ninguna restricción impuesta en su fabricación/utilización, para que cualquiera los pueda usar. Para ello es necesario que sus planos estén disponibles y que se puedan ver/modificar desde cualquier plataforma (Linux,Windows...). Por ello se ha optado por diseñarlos con el programa **Qcad**[36], que es multiplataforma y además es software libre (por lo que el formato fuente en el que se guardan los planos está perfectamente documentado). Asímismo, los modelos en 3D se han realizado con el programa **Blender**[37], que también es libre y multiplataforma.

3.3.2. Descripción

En la figura 3.5 se muestra una foto del módulo en la que el servo está en diferentes posiciones. El módulo se divide en dos partes, que pueden rotar una con respecto a la otra, denominadas **cuerpo** y **cabeza**. El servo se encuentra atornillado al cuerpo mediante dos tornillos de Φ3mm y 10 mm de largo. Es un servo del tipo **futaba 3003**, con una corona de Φ21mm.

La cabeza por una lado está atornillada a la **corona del servo**, con dos tornillos iguales a los anteriores y por otro lado directamente al cuerpo, formando un "**falso eje**" que permite que el peso se reparta. No obstante, la fuerza que genera el servo sólo se transmite por la corono, el falso eje sólo es de apoyo.

En la figura 3.6 se muestra el módulo en el que la cabeza está situada en diferentes ángulos con respecto al cuerpo. Y en la figura 3.7 se muestra el módulo por el lado del falso eje.

El recorrido de la cabeza es de 180 grados, que puede estar situada en una posición de -90 a 90 grados con respecto al cuerpo. El tamaño del módulo es el mínimo posible usando el servo futaba 3003. **Las bases son cuadradas**, de 52x52mm. Si el módulo está situado como en la figura 3.5, la

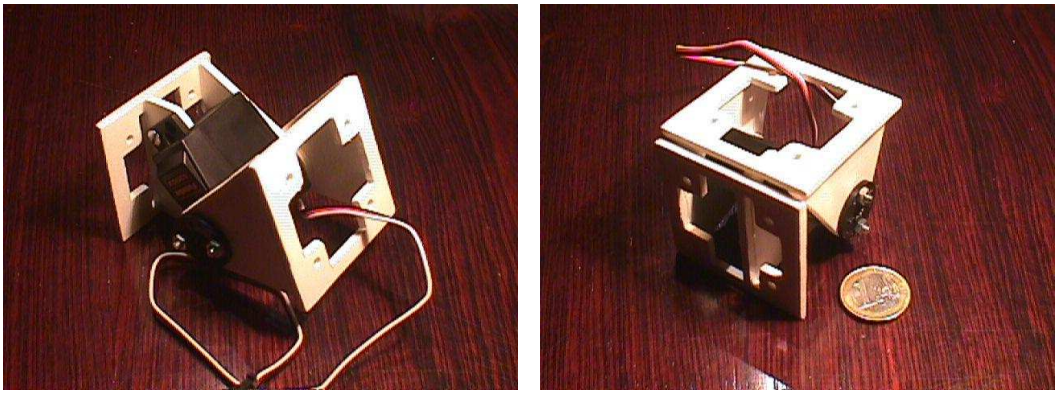


Figura 3.5: Fotos del módulo Y1. Izquierda: el servo está en su posición central. Derecha: El módulo convertido en un cubo

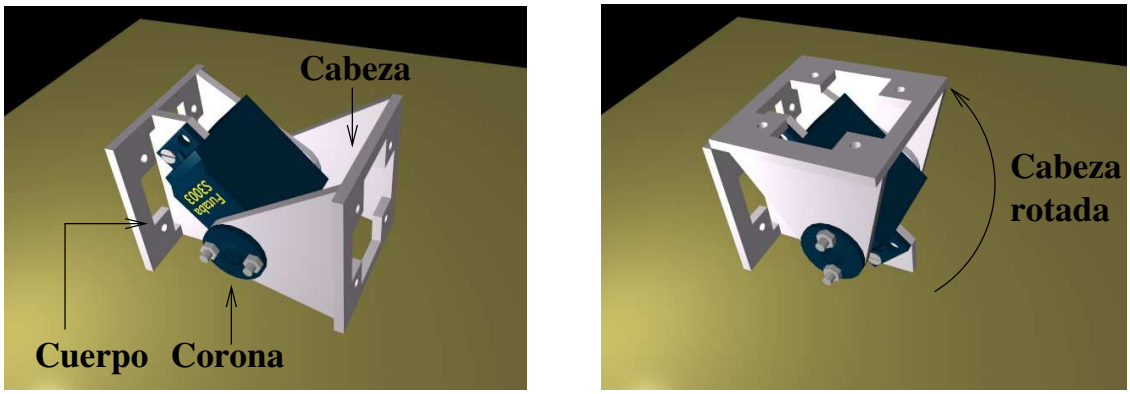


Figura 3.6: Cabeza y cuerpo del módulo. Una puede rotar con respecto a otra.

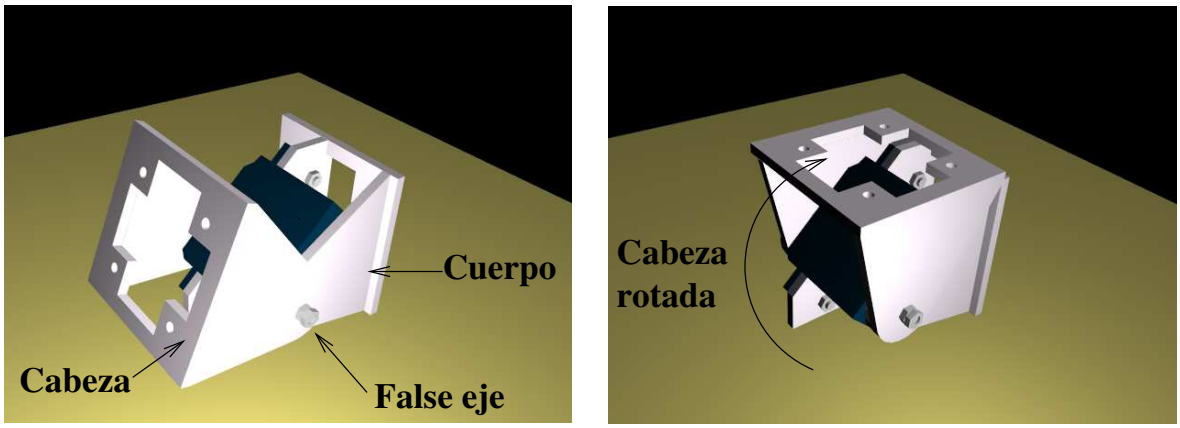


Figura 3.7: Módulo Y1 visto desde el lado del falso eje

altura y la anchura son fijas, determinadas por la base. La longitud depende de la posoción entre el cuerpo y la cabeza. Cuando el módulo está en posición natural (la cabeza forma 0 grados con respecto al cuerpo), es de 74mm (máxima) y cuando está en un extremo es de 55mm.

El módulo está constituido por 5 piezas diferentes, construidas con **PVC expandido** de 3mm de grosor. También se ha realizado un prototipo en metacrilato, que es más rígigido, pero tiene la ventaja de que es transparente y permite ver el interior. **Los planos se encuentra en el apéndice A**. La nomenclatura para nombrar las piezas es la siguiente:

- **Cuerpo**:
  - **Pieza F**: donde está atornillado el servo.
  - **Pieza B1**: Base del cuerpo. Punto de unión con otro módulo.
  - **Pieza FE**: Pieza del Falso eje. Se une a la otra pieza FE de la cabeza
- **Cabeza**:
  - **Pieza B2**: Base de la cabeza. Punto de unión con otro módulo.
  - **Pieza E**: Pieza atornillada al eje del servo (a su corona).
  - **Pieza FE**.

Una vez construidas las piezas, bien manualmente o bien llevándolas a una tienda especializada, la construcción de los módulos es sencilla. Se pueden encontrar más detalles en el apéndice B.

En el diseño de los módulos no se ha previsto que la electrónica se encuentre en su interior ni tampoco la alimentación. Con esta primera generación se ha querido evaluar la **viabilidad mecánica** y **las posibilidades de conexión en fase o desfasados**.

3.3.3. Caracteristicas

Una características fundamental es la de poder colocar dos módulos **en fase o desfasado** (apartado 2.5.2.1), como se muestra en la figura 3.8. Cuando están **conectados en fase**, todas las articulaciones se mueven permaneciendo en el mismo plano (en la figura 3.8 los dos módulos permanecen siempre en un plano perpendicular al suelo). Cuando están desfasados, pueden estar en planos distintos.

Esta característica permite que se puedan construir gusanos que se puedan mover por un plano (conexiones desfasadas), o bien gusano que sólo avancen en línea recta (todos los módulos en fase).

Para la unión de los módulos se utilizan 4 tornillos situados en las cuatro esquinas de las bases de los módulos. En este sentido, el robot no es autoreconfigurable. Los módulos hay que cambiarlos "a mano".

Las bases del módulo (B1 y B2) tienen un hueco en interior, lo que resulta tremendamente útil para pasar los cables de un módulo a otro.

Además de disponer de los planos para su fabricación, en el proceso de diseño se han realizado **modelos virtuales en 3D** de todas las piezas, tornillos y el propio servo, por lo que se ha podido construir un modelo virtual del módulo antes de disponer de la pierza fabricada.

La ventaja de disponer de un modelo virtual, es que nos podemos hacer una idea del aspecto que va a tener un determinado robot, antes de construirlo. Además es muy útil para realizar manuales y animaciones.

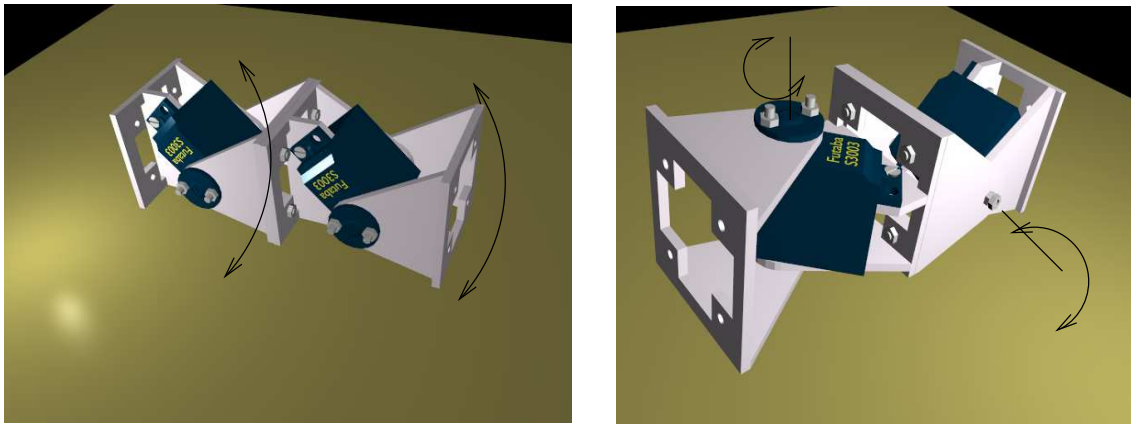


Figura 3.8: Dos módulos Y1 conectados. A la izquierda están en fase. A la derecha está desfasados

3.3.4. Pruebas

Antes de tenerse los módulos tal cual están ahora, se han desarrollado versiones previas, con las que se han ido detectando fallos y sobre las que se han añadido nuevas características. Se han realizado pruebas de funcionamiento, conectando dos módulos en fase y desfasados, 3 módulos, 4, etc.

Una vez obtenida una versión "estable", se ha procedido a juntar 4 módulos y construir Cube Reloaded. En los vídeos incluidos en el CD se encuentra algunas de las pruebas

3.4. Mecánica

La estructura mecánica de **Cube Reloaded** está compuesta por cuatro **módulos Y1**, **conectados en fase**. En la figura 3.9 se muestra una foto, junto a una pluma para hacerse una idea del tamaño. Cuando todos los módulos están en reposo, la longitud es de 29cm y la altura y anchura de 51mm. El módulo situado en la cola se le asigna el número 1 (el situado más a la izquierda según se mira la foto de la figura 3.9) y el de la cabeza el número 4.

Para las conexiones de los servos se ha construido una pequeña placa pasiva que tiene 4 conectores macho de 3 pines para los servos y un conector para cable plano de 10 vías, por el que viene la alimentación y las señales de control de los servos de la electrónica externa. Su aspecto se puede ver en la figura 3.10. El reducido tamaño es debido a que se tiene que situar dentro de uno de los módulos, pegada con velcro sobre un servo. Los cables de todos los servos se llevan a esa placa y ahí sale un cable de bus de 10 líneas que se conecta a la electrónica externa. Debido a que los módulos Y1 tienen huecos en sus bases, los cables puede pasar de un módulo a otro ningún tipo de problemas.

En total se han construido 8 módulo Y1. Cuatro se han empleado para construir a **Cube Reloaded** y hacer pruebas de locomoción. Los otros cuatro se han empleado para conectarlos desfasados y hacer pruebas tanto mecánicas como de control. Nos referiremos a este robot como **Cube larva**, puesto que las pruebas de movimiento realizadas daban la impresión de tratarse de una larva que estaba viva. Se puede ver en la figura 3.11.

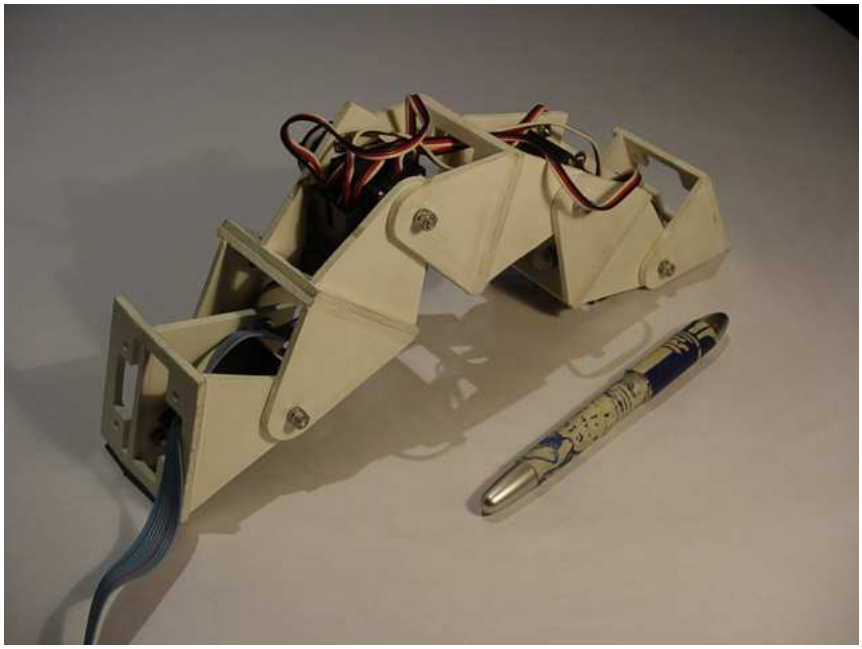


Figura 3.9: Cube Reloaded

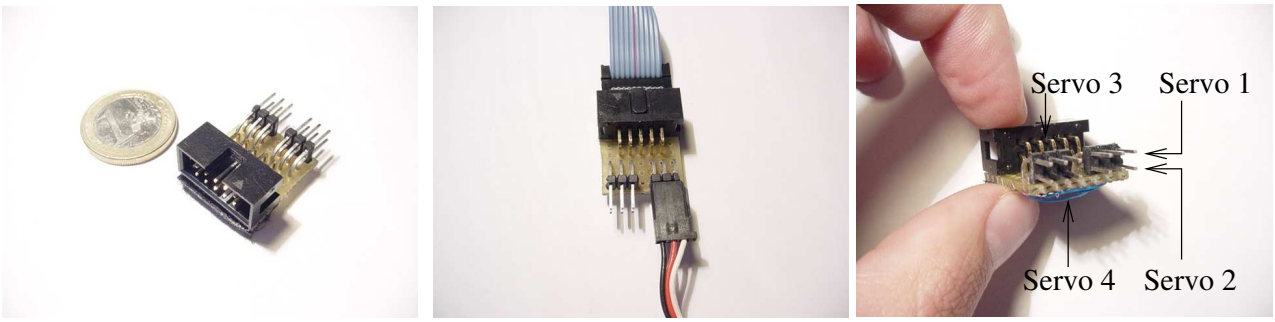


Figura 3.10: Placa pasiva pct-servos, para la conexión de los servos de los módulos a un cable de bus. En la foto de la izquierda se aprecia el reducido tamaño, necesario para entrar dentro de un módulo. En la foto central hay conectado un servo y el cable de bus. En la fogo de la derecha se indica dónde se conecta cada uno de los servos, considerando que el 1 es el de la cola

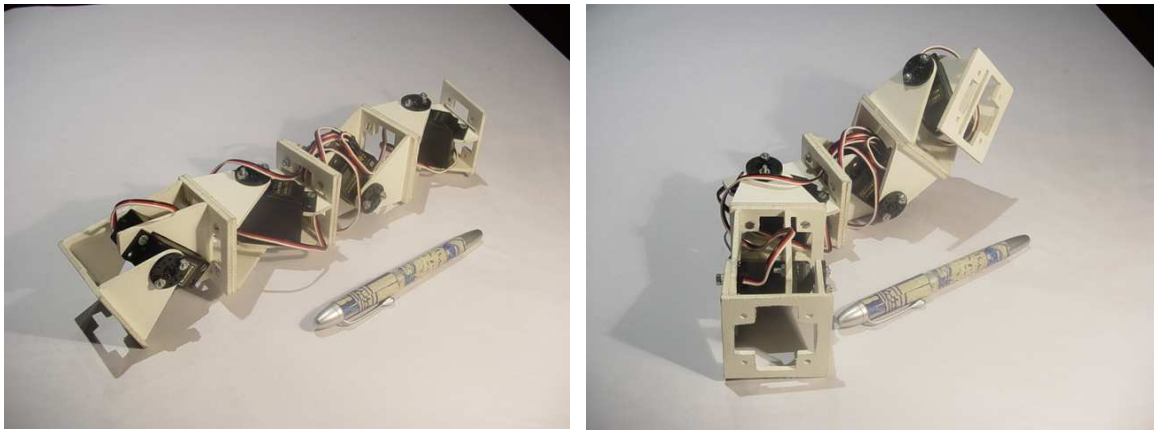


Figura 3.11: Cube larva, en dos posiciones

3.5. Electrónica

Para la locomoción de Cube Realoaded se han evaluado dos alternativas diferentes, una usando un microcontrolador específico (un 6811E2 de Motorola) y otra usando una FPGA.

3.5.1. Microcontrolador específico

Una manera de controlar Cube Reloaded es utilizando un microcontrolador determinado para generar las señales de PWM (control de servos). El problema de la escalabilidad se resuelve poniendo estos micros en red.

Para Cube Reloaded se ha seguido utilizando el microcontrolador 6811, con la tarjeta CT6811. El servidor de control se ha optimizado y ahora es posible controlar hasta 8 servos con un único comparador. **Consultar el apéndice C para más información**. El 6811 dispone de 5 comparadores, por lo que teóricamente se podrían controlar hasta 40 servos. La tarjeta BT6811 sólo tiene conectores para controlar 4 servos, por lo que se ha dejado de utilizar.

El cable de bus de 10 hilos que viene de la tarjeta pct-servos situada en uno de los módulos se lleva directamente al puerto B de la CT6811. El gusano puede tener hasta 8 módulos, sin necesidad de realizar ningún cambio. Si se quieren realizar robots con un número mayor de módulos, sólo hay que conectar las tarjetas CT6811 que sean necesarias a través del SPI (puerto D).

La **tabla de control** se puede almacenar en la memoria EEPROM, sin embargo, para hacer pruebas de locomoción es mejor generar las secuencias de movimiento desde el PC, con el programa **cube-virtual**[45], y descargarlas al gusano usando el programa **cube-fisico**[44], el mismo que se ha usado para Cube 2.0.

En la figura 3.12 se muestra a Cube Reloaded controlado desde una tarjeta CT6811. El cable de bus se conecta al puerto B. La CT6811 está conectada a la alimentación (5v) y a un PC.

El usar un microcontrolador es una solución viable y sencilla, sin embargo, si necesitamos más potencia de cálculo o queremos usar un microcontrolador más avanzado tendremos que rediseñar toda la electrónica. Nosotros nos tenemos que adaptar al microcontrolador y solucionar los problemas con él. Es una solución poco flexible.

Cube Reloaded + CT6811 **Cube Reloaded + JPS**

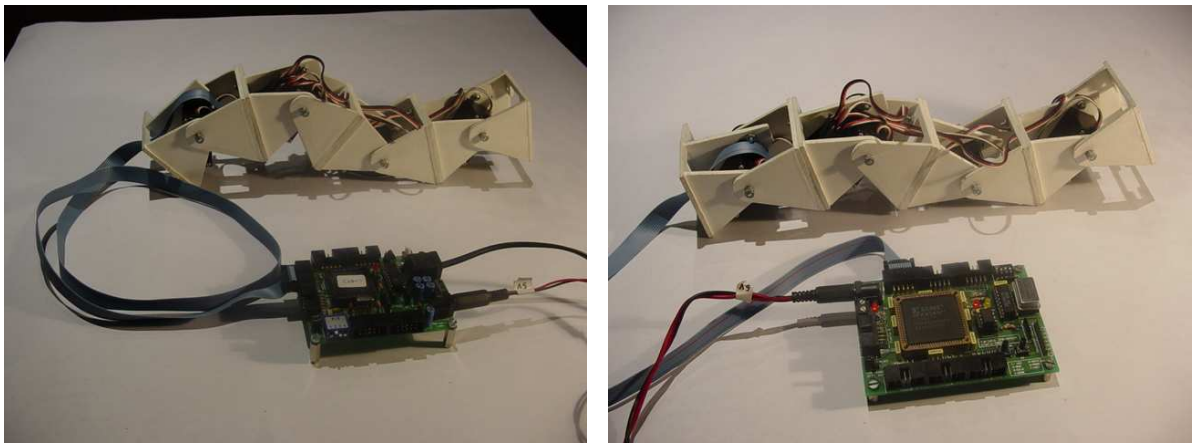


Figura 3.12: Cube Reloaded con diferente electrónica. Izquierda, controlado con una CT6811. Derecha, conectado a una tarjeta JSP, con FPGA

3.5.2. FPGA

Cube Reloaded también se puede controlar con una FPGA (figura 3.12), usando la **tarjeta JPS**[40][41]. Esta alternativa permite nuevas posibilidades como son la de diseñar una CPU específica o de la hacer un control puramente *hardware*, basado en lógica combinación y secuencia.

En ambos casos es necesario disponer de una unidad de PWM, que permita posionar el servo a partir de la información depositada en un registro de 8 bits. Se ha diseñado esta unidad en VHDL, adaptada del proyecto Labobot[42][43], usando un oscilador de 1MHZ.

3.6. Control

Al igual que Cube 2.0, el control se realiza a través de **tablas de control**. Sin embargo este sistema de control está implementando de tres formas diferentes:

1. Usando un 6811 y conexión al PC (aunque se podría dejar autónomo)
2. Usando una lógica combinacional y secuencial metida en una FPGA
3. Usando una CPU en una FPGA.

Para (2) Y (3) se utilizan unidades PWM descritas en VHDL. Se pueden encontrar más detalles en el apéndice D.

3.6.1. Microcontrolador 6811 y PC

La tabla de control se encuentra en el PC y se envía a través del puerto serie al 6811 donde se ejecuta un programa servidor que genera las señales PWM en función de las posiciones indicadas. Es interesante que este servidor pueda mover cuantos más servos mejor. En Cube 2.0, el servidor sólo controlaba 4 servos, usando en total 5 comparadores. Uno común para generar una señal de 50Hz y uno medir la anchura del pulso de cada una de las señales PWM.

Esta solución tiene un error en su concepción. Todas las señales PWM están en fase por lo que cuando las posiciones de los servos son las mismas, se producen todas las interrupciones a la vez (4) lo que ocasiona un retraso en las últimas que se ejecutan, haciendo que su anchura del pulso sea un poco mayor, por lo que se observa cómo esos servos tienen de vez en cuando unos "espasmos" de pequeña amplitud.

La idea para controlar muchos servos es generar las señales PWM desfasadas, que se puede hacer con un único comparador, evitándose el problema anterior además de poderse controlar el doble de servos con la quinta parte de los recursos del microcontrolador. Todos los detalles así como el código que se ejecuta en el 6811 se pueden encontrara en el apéndice C.

3.6.2. Lógica combinacional y secuencial en FPGA

La idea de este controlador es la de implementar por hardware la tabla de control, que se introduce en una memoria ROM y se direcciona mediante un contador.

Uno de los principales problemas es la temporización. Por un lado hay que generar una señal de reloj de 102KHZ para obtener un PWM de exactamente 50Hz. En la tarjeta JPS[40][41], empleada para la implementación de este controlador, se tiene un reloj de 1MHZ. A partir de él hay que obtener la señal de 102KHZ. Utilizando un divisor entre 10, se consigue una frecuencia de 100KHZ, lo que proporciona una frecuencia de PWM de 48.8HZ, que entra dentro de los límites de tolerancia del servo.

Por otro lado, el control de los servos es en bucle abierto y el tiempo que tarda el servo en alcanzar la posición indicada es indeterminado. Es necesario estimarlo. En la solución usando un microcontrolador, este tipo se estima en función de el ángulo que tiene que recorrer para llegar a la nueva posición (la velocidad angular nominal del servo es constante y conocida). Se estima el tiempo que le llevará al servo recorrer este ángulo y se realiza una espera, que es variable (unas veces se espera más, otras veces menos).

En el caso de la implementación por hardware se ha simplificado y se utiliza un tiempo de espera igual para todas las entradas de la tabla. Si el incremento angular de los servos es pequeño al pasar de una posición a otra, y dado que todos los servos están recorridos por la misma onda sinusoidal, el tiempo de espera se puede aproximar a una constante para todos los servos.

Es decir, en el caso de esta solución hardware, el tiempo entre el envío de una entrada a los servos y otra es constante e igual para todas las entrada. Esto se podría mejorar incluyendo en la tabla un campo que indicase el tiempo a esperar, sin embargo, por lo comentando anteriormente de tener unos incrementos angulares pequeños, no es necesario.

El diagrama de bloques se muestra en la figura 3.13. Está constituido por dos partes: las unidades de PWM y el secuenciador que devuelve consecutivamente las filas de la tabla de control. Esta tabla se encuentra en una ROM de 32x32, direccionada mediante un contador módulo 30.

El periodo de la señal de reloj del secuenciador (T_{CLK}) determina el tiempo que se tardará en enviar la siguiente posición a los servos. Cada servo tarda un tiempo t_{Ser} en alcanzar la nueva posición. Si $T_{CLK} > t_{Ser}$, el servo permanecerá un tiempo $T_{CLK} - t_{Ser}$ en reposo en la nueva posición, antes de que llegue la siguiente. Si esta diferencia es del orden de las centenas de ms, se apreciará un movimiento "a saltos" (no continuo). Por el contrario, si $T_{CLK} < t_{Ser}$, se enviará la nueva posición al servo antes de que haya alcanzado la anterior, por lo que los servos nunca llegarán a alcanzar las posiciones indicadas en la tabla de control.

Mediante un multiplexor de 8 a 1, con sus entradas de selección conectadas a tres *switches* de la tarjeta JPS, se selecciona una señal de reloj de entre 8 posibles, con los siguientes periodos: 1000,

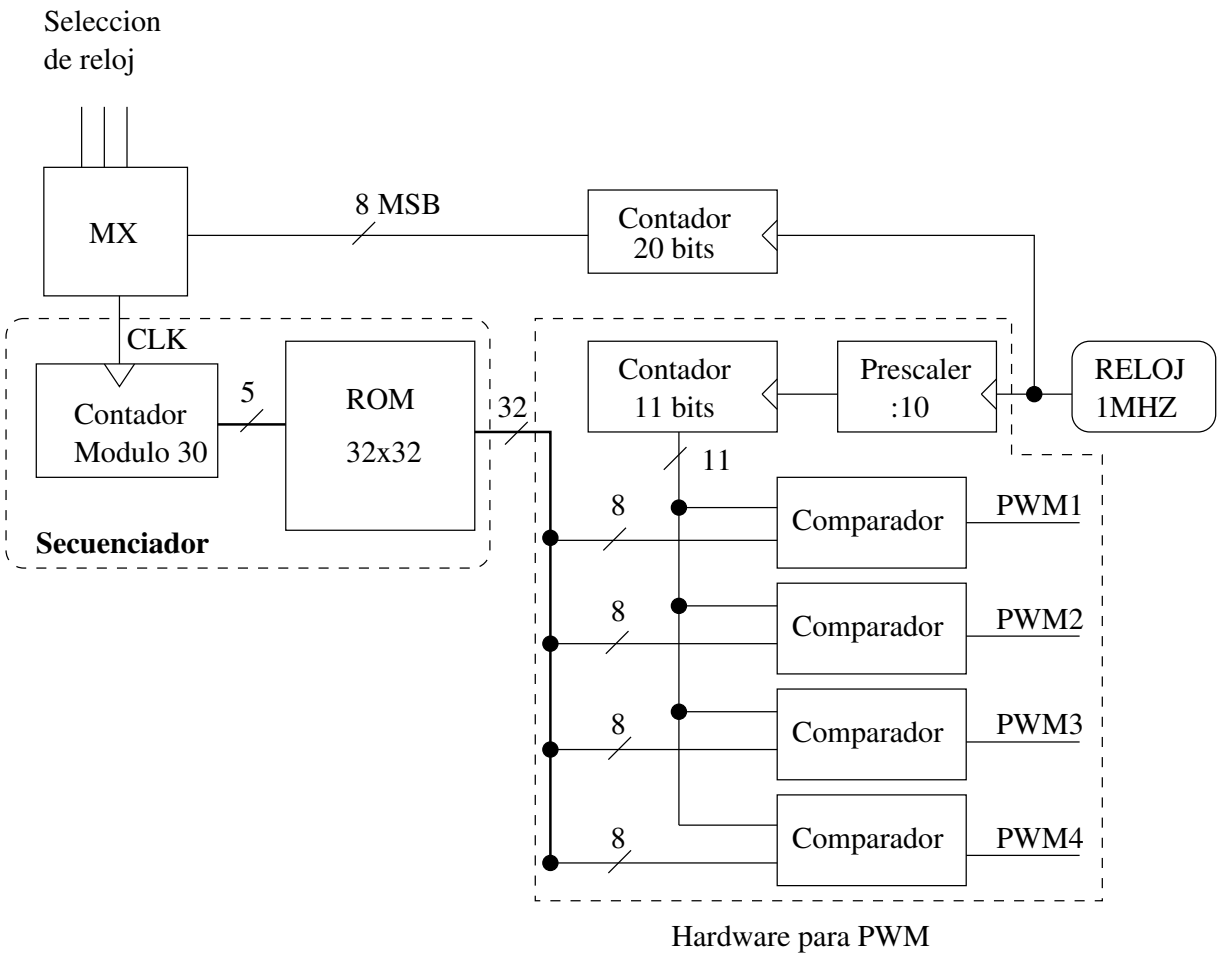


Figura 3.13: Controlador hardware basado en lógica combinacional y secuencial

524, 262, 131, 66, 33, 16 y 8 ms. Si se utiliza la de 1000 ms, los servos cambian de posición aproximadamente cada segundo. Si se utiliza la última el cambio es demasiado rápido.

3.6.3. CPU empotrada en FPGA

Este es un campo muy interesante ya que no es necesario adaptarse a una CPU o microcontrolador proporcionada por un fabricante, sino que se diseña a medida de las necesidades. En el campo de las CPUs empotradas está trabajando Iván González que está constribuyendo al proyecto Cube Reloaded implementando esta solución aplicada a la locomoción del gusano. Los siguientes apartados se han copiado literalmente de [47], y han sido escritos por **Iván González**.

El sistema tradicional de control empleado en microbots se basa en el uso de microcontroladores. Estos sistemas presentan una enorme gama de modelos, diferenciados por la cantidad de recursos/periféricos que los componen. El uso de sistemas basados en microcontrolador permite disponer de la flexibilidad del software, pero en muchas ocasiones, el número de recursos/periféricos es menor al necesario para la tarea a realizar. La combinación de microcontrolador y dispositivo reconfigurable es quizás la mejor solución para todo tipo de tareas[46], ya que permite diseñar en el dispositivo reconfigurable los periféricos necesarios: coprocesadores, unidades operacionales,etc.

En esta alternativa se ha decidido aplicar la misma idea de flexibilidad también a la CPU, y por tanto, dar un paso más incluyendo la CPU dentro del dispositivo reconfigurable, lo que permite disponer de un sistema 100 % adaptable a la tarea a realizar: tamaño de memorias, potencia de la CPU, etc. y al mismo tiempo, añadirle los periféricos necesarios. Esta flexibilidad permite además, mejorar la conexión de los periféricos a la CPU de una forma directa y más óptima, eliminando la necesidad de implementar soluciones basadas en protocolos de comunicación serie o paralelo.

3.6.3.1. La CPU Pandabear

La CPU diseñada presenta una arquitectura muy sencilla, con la intención de ir adaptándola a las necesidades que puedan surgir a lo largo del desarrollo del problema planteado, el control del robot ápodo. Las carácterísticas más relevante son:

- Arquitectura de 16 bits.
- Arquitectura Harvard, con memorias separadas para datos y programa.
- Arquitectura LOAD-STORE. Los accesos a la memoria de datos se harán sólo con estas instrucciones.
- Cuatro registros internos de propósito general.
- Cuatro registros internos de propósito general / indexación.
- Direccionamiento indexado.
- Un bit de flag para saltos condicionales.
- 8 instrucciones básicas (ver tabla 3.1).

Para cumplir con el set de instrucciones, se ha diseñado una arquitectura basada en una máquina Mealy de tres estados: captura, decodificación/ejecución y almacenamiento. Las salidas de la máquina son las señales de control que se envían a los distintos elementos que la acompañan. Un diagrama de

Instrucción	Función
LOAD	Rd := Md[Ri + Offset]
STORE	MD[Ri + Offset] := Rs1
MOVE	Rd := Rs
ADD	Rd := Rs1 + Rs2
SUB	Rd := Rs1 - Rs2
CMP	If Rs1 = Rs2 flag := 1 else flag := 0
BF, BNF	If (flag xor c) = 1 then PC := PC + offset
JMP	PC := PC + offset

Cuadro 3.1: Set de instrucciones de la CPU

está arquitectura se muestra en la figura 3.14. Además, se ha añadido un interface de comunicación con dos memorias de 16 bits de datos y 8 bits de dirección, síncronas. La memoria de programa es una ROM, mientras que la de datos es SRAM, y ambas están implementadas dentro del dispositivo.

3.6.3.2. Componentes adicionales y aplicación

Para permitir que la CPU pueda controlar el robot se ha añadido al sistema formado por la CPU y las memorias, el conjunto de unidades pwm usadas en el apartado 3.6.2. Los cuatro comparadores que forman parte de esta unidad tienen cada uno un registro asociado que es mapeado en la memoria de datos, de modo que el manejo de las unidades PWM es completamente transparente.

Para poder comparar esta alternativa con la presentada en 3.6.2, se emplea la CPU diseñada para reproducir exactamente la misma secuencia de movimientos por lo que en la memoria de datos asociada al sistema se almacena la tabla de control. El programa a ejecutar consiste en una rutina de lectura de las posiciones consecutivas de la memoria de datos y escritura de esos datos en las posiciones de memoria correspondientes a las unidades pwm.

Para el desarrollo del programa, y complementando al diseño VHDL de la CPU, se ha diseñado un ensamblador para esta arquitectura de modo que sea más sencilla su programación. Un breve estracto del programa se muestra a continuación:

```
inicio:
   ; Capturamos secuencias
   ; (indexacion mediante registro i1 e i2)
   load r1,sec1_2,i1
   load r2,sec1_2,i2
   ; Enviamos secuencias a pwm
   store pwm1_2,r1
   store pwm3_4,r2

   ; Retardo
   load r1,cero
   load r2,uno
   load r3,contador
retardo:
   add r1,r1,r2
```

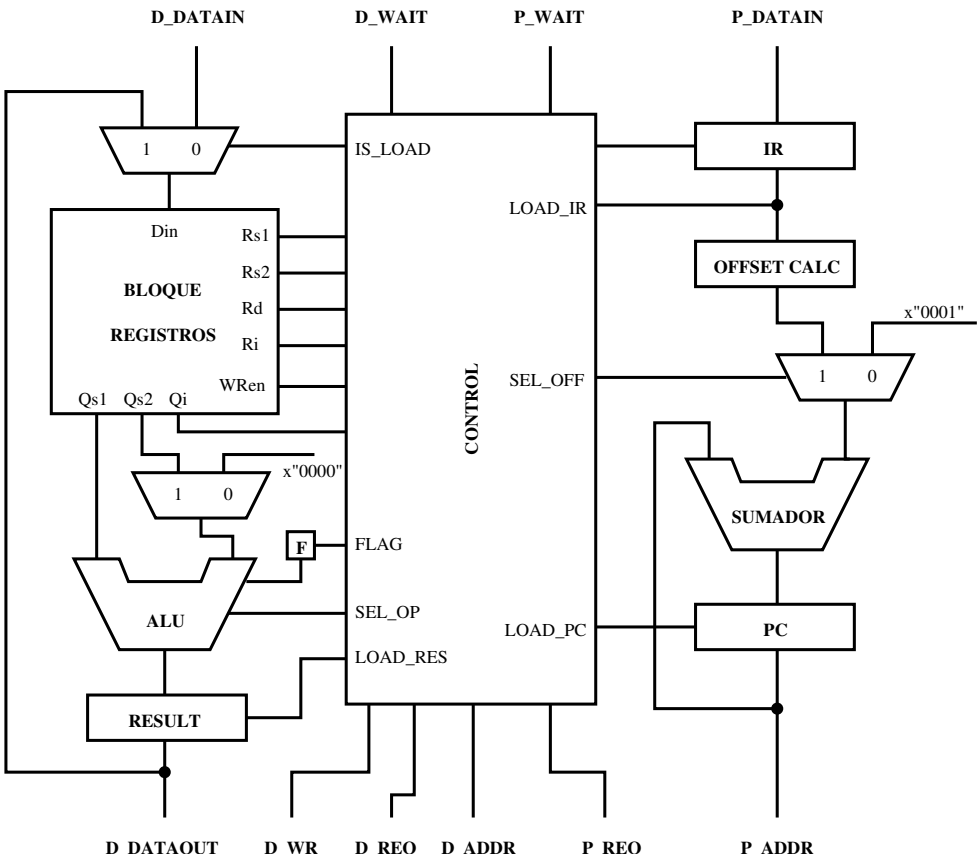


Figura 3.14: Diagrama de bloques de la CPU (alternativa II). La interface con el exterior se realiza a través de dos buses de datos y dos buses de direcciones (arquitectura Hardvard). Además se han añadido señales de control para el manejo de memorias externas.

```
cmp r1,r3
bnf retardo
```

Capítulo 4

Experimentos y resultados

4.1. Introducción

En este capítulo describiremos algunos de los experimentos realizados con Cube Reloaded. Presentaremos las pruebas de locomoción, realizadas a partir de diferentes tipos de ondas y con distintas amplitudes, mostrando lo resultados. Asimismo describiremos el circuito de pruebas que ha superado el gusano. Por último mostraremos los resultados de la implementación de los controladores en una FPGA

4.2. Pruebas de locomoción

4.2.1. Ondas sinusoidales periódicas

Definimos el parámetro k (número de ondas) como: $k = \frac{L}{\lambda}$, siendo L la longitud del gusano y λla longitud de onda. Este parámetro nos da una idea del número de ondas completas que están recorriendo el gusano. Se han generado secuencias de movimiento correspondientes a K=1 y K=2.

En la tabla 4.1 se pueden ver los tiempos empleados por el gusano en recorrer una distancia igual a su longitud, para diferentes valores de la amplitud.

En general, cuanto mayor sea la amplitud de la onda, saltos mayores podrá sobrepasar el gusano.

Para **k=2**, el número de puntos de apoyo es mayor, lo que le da al movimiento una mayor estabilidad, pero la amplitud máxima es de 30 unidades. Con ella algunas articulaciones pasan por la posición -90 ó 90, rango máximo de giro. Si se usa una amplitud mayor, las articulaciones no pueden situarse sobre la función. En la figura 4.2, el gusano virtual no puede adaptarse a la onda porque las articulaciones 2 y 4 están en su límite físico. Fijándonos en los datos obtenidos, cuanto mayor sea la amplitud, mayor es la velocidad de avance (menor es el tiempo). La amplitud máxima es de 30, por lo que habrá un límite en la altura del obstáculo que se puede superar.

Para **K=1**, sólo hay dos puntos de apoyo por lo que será menos estable y el consumo mayor, ya que hay mayor cantidad de rticulaciones haciendo fuerza. Además habrá un momento en el que el gusano sólo estará apoyado sobre la cola y la cabeza, , como se muestra en la figura 4.3, donde el consumo será máximo. Dependiendo de otros parámetros como la longitud total del gusano y la fuerza de los servos, el arco será o no estable. Las amplitudes que se pueden conseguir con **k=1** son mayores, por lo que se pueden superar obstáculos más altos.

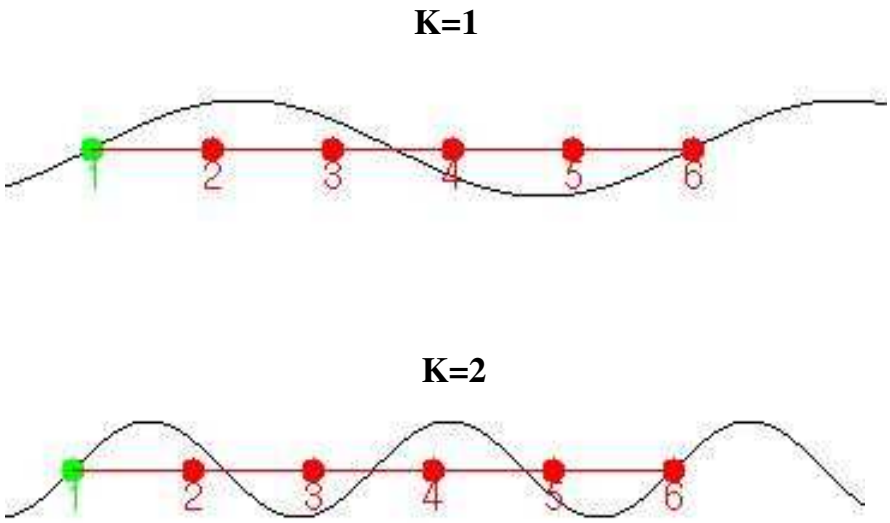


Figura 4.1: Ondas sinusoidales empleadas para la generación de las secuencias de movimiento de prueba

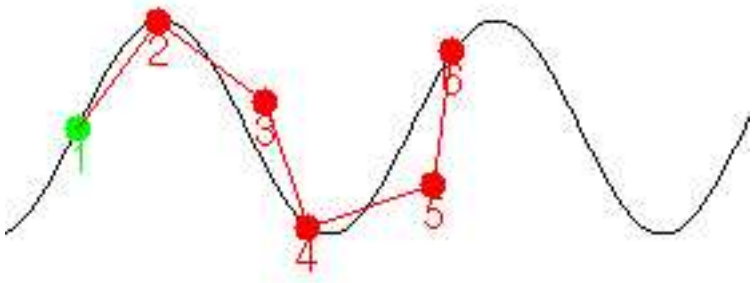


Figura 4.2: Cube no es capaz de tomar la forma de la función indicada. Tanto la articulación 2 como la 4 han llegado a sus límites físicos de giro

K=2		**K=1**	
Amplitud	**tiempo**	**Amplitud**	**Tiempo**
1	2’:20”	1	3’12”
2	29”	2	1’17
3	15”	3	41”
4	12”	4	33’9
5	10”	5	25
6	9’9	6	20’3
7	9’2	7	17’5
8	8’40	8	13’4
9	8	9	13’3
10	7’3	10	13’1
11	7’2	12	10’8
12	7’3	14	10’4
13	7’3	16	9’3
14	6’5	18	8
15	6’3	20	7’8
16	6’5	22	7’4
17	6’6	24	6’5
18	6’6	26	6’3
19	7	28	6’3
20	6’3	30	6’5
21	6	32	6’2
22	6	34	6’1
23	6’2	36	6’9
24	6’3	38	7
25	6’1	40	7’3
26	5’9	50	9’4
27	5’4	60	15’4
28	5’2	–	–
29	**5**	–	–
30	5,7	–	–

Cuadro 4.1: Tiempos que tarda Cube Reloaded en recorrer una distancia igual a su propia longitud, empleando una señal sinusoidal con valores de k=1 y k=2, en función de la amplitd

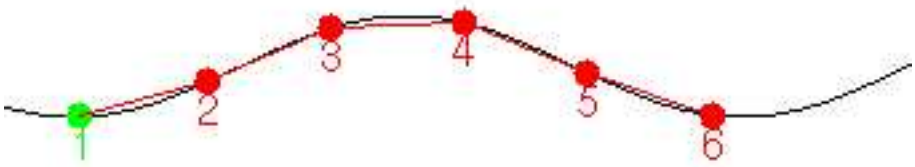


Figura 4.3: Instante en el que cube sólo está apoyado en la cola y la cabeza, cuando k=1

Figura 4.4: La semionda atravesando a cube virtual

4.2.2. Semiondas

Otra forma de mover a Cube es usando semiondas, quedarse sólo con un lóbulo positivo de una onda sinusoidal que recorre el cuerpo del gusano. El gusano parte del reposo, es atravesado por la semionda y vuelve al reposo. Es un movimiento similar al de los gusanos de seda. Hasta que la onda no ha alcanzado la cabeza, la cola no vuelve a generar otra.

En la tabla 4.2 se muestran los tiempos empleados para recorrer una distancia igual a su propia longitud. Este movimiento es más lento, pero más estable (hay un mayor número de puntos en contacto con la superficie). A mayor amplitud, mayor velocidad y obstáculos más altos se pueden sobrepasar.

4.2.3. Un circuito de pruebas

Para hacernos una idea de las posibilidades de los robots ápodos, vamos a hacer que Cube Reloaded supere un circuito de pruebas (figura 4.5) constituido por un cilindro de cartón de 8 cm de diámetro y 30 cm de largo y un escalón de 3'5 cm de altura.

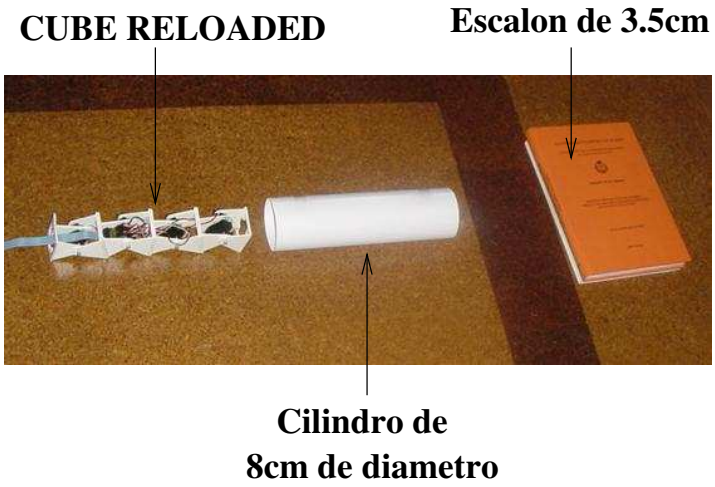
Para el control se usa la CT6811 conectada a un PC, donde el operador selecciona los patrones de movimiento más adecuados para cada obstáculo.

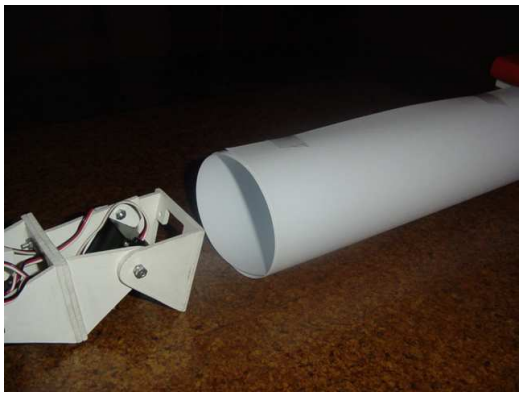
El primer obstáculo representa una restricción en la altura máxima que puede alcanzar, por lo que se tendrá que mover con una amplitud pequeña, lo que tendrá como contrapartida que irá muy despacio. Eligiendo una semionda de amplitud 10 y longitud de onda 250 (parámetros del programa cube-virtual) se consigue atravesar exitosamente el tubo.

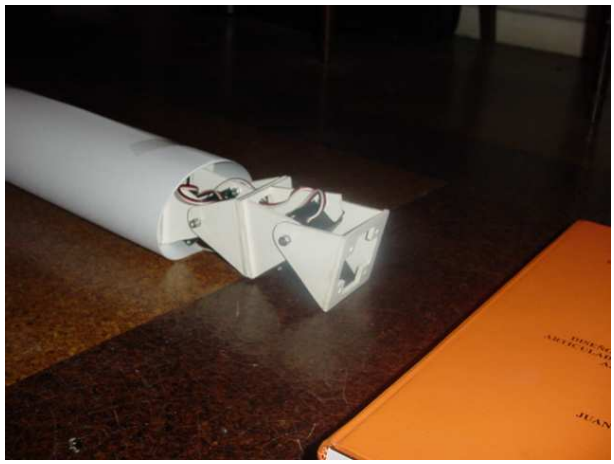
Para superar el escalón es necesario que la amplitud sea lo suficientemente grande como para que

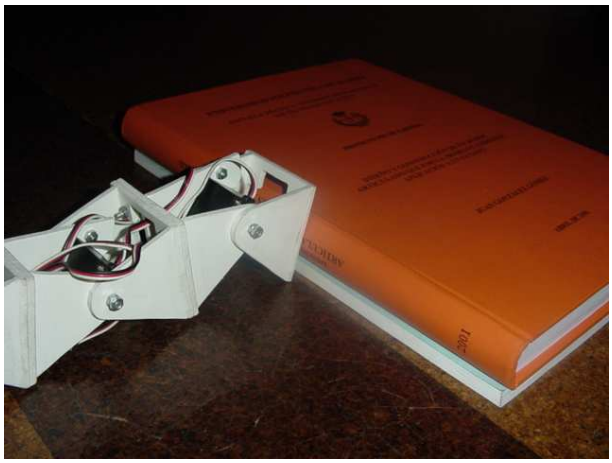
Amplitud	**Tiempo**
1	No avanza
2	3:46
3	2:10
4	1:33
5	1:06
6	52
7	52
8	48
9	46’7
10	43’7
12	39’9
14	34’5
16	32
18	27’3
20	25’5
22	25
24	22’6
26	22
28	21
30	18’7
35	16’7
40	15
45	13’7
50	12’2
60	9’1

Cuadro 4.2: Tiempos que tarda Cube Reloaded en recorrer una distancia igual a su propia longitud, empleando una semionda de longitud de onda 250 unidades, en función de la amplitd


CUBE RELOADED
Escalon de 3.5cm
Cilindro de
8cm de diametro







Controlador	CLBs	%
Lógica comb+sec	78	39
CPU empotrada	212	108

Cuadro 4.3: Resultados de la implementación de los controladores en una FPGA

la cabeza del gusano pueda apoyarse en su superficie. Seleccionando la misma onda anterior pero con una amplitud de 50 se consigue tanto subir como bajar el escalón.

Después del escalón ya no quedan obstáculos, sólo un terreno plano, por lo que se utiliza una onda periódica con k=2 y amplitud de 29, para conseguir la máxima velocidad de avance.

4.3. Pruebas con la FPGA

La implementación del circuito de control presentado en el apartado 3.6.2 se ha implementando en una FPGA Spartan I, en la placa JPS, con un oscilador de 1MHZ. La tabla de control empleada se corresponde con una onda sinusoidal periódica de amplitud 20.

La solución con CPU empotrada también se ha sintetizado, sin embargo todavía no se ha probado. Los resultados se muestran en la tabla 4.3:

La solución con la CPU ocupa más recursos y no cabe en la Spartan I. Esta solución incluye la CPU, la memoria con las tablas de control y el programa y las 4 unidades PWM. Es necesario utilizar dos tarjetas JPS, particionando el diseño de la siguiente manera: una FPGA con la CPU y la memoria de programa y la segunda con los datos y las unidades PWM, ya que estas últimas comparten con la memoria los buses de datos y dirección.

Se ha considerado probar esta alternativa en la placa Digilab2E, que posee una XC200E. En este dispositivo el diseño ocupa 285 slices (12 %) sin aprovechamiento de las BRAM.

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

Al estudiar los orígenes, características y prototipos de robots ápodos construidos en diferentes centros de investigación se ha descubierto una nueva área de investigación en robótica: los **robots modulares y reconfigurables**. Se han estudiado los dos robots más importantes dentro de esta área: **Polypod**, el precursor, creado por Mark Yim en Stanford y **Polybot**, desarrollado en el PARC y sobre el que se están realizando nuevas investigaciones. Ambos robots ponen de manifiesto una de las características más importantes: la **versatilidad**. Fundametalmente se han orientado hacia el **problema de la locomoción**.

En vez de diseñar robots específicios, lo que se pretende es diseñar unos módulos a partir de los cuales poder construir robots muy diferentes. Además estos robots se pueden **reconfigurar**, tomando una u otra forma en función del terreno por el que se desplacen.

Siguiendo este enfoque, hemos diseñado y construido el **módulo Y1**. Tienen una característica muy importante: la posibilidad de unirse dos módulos en **fase** o **desfasados**, lo que permite construir robots ápodos que bien se desplacen sólo por una línea recta (todos los módulos en fase) o bien que se puedan mover por un plano (desfasados).

Uniendo cuatro módulos Y en fase se ha construido la estructura de **Cube Reloaded**, y junto con el software y la electrónica de control constituyen una **plataforma para realizar investigaciónes en esta área de la robótica**, aplicada fundamentalmente a la locomoción.

La electrónica de control se ha mejorado, usándose un microcontrolador específico, igual que en Cube 2.0, pero usando un servidor más óptimo, capaz de controlar el doble de servos usando sólo un único comparador. Asimimos se han evaluado dos **alternativas nuevas**, basadas en **FPGA**. Una de ellas empleando **lógica combinacional y secuencial** para la implementación de las **tablas de control** y lograr la locomoción de Cube Reloaded en línea recta. Se ha implementado en la placa JPS. Otra alternativa basada en una **CPU específica, empotrada en la FPGA**, que necesita más recursos y no se puede implementar en una Spartan I. Se ha simulado pero no se ha aplicado al todavía al control del gusano.

Finalmente hemos realizado **experimentos de locomoción**, generándose diferentes **patrones de movimiento** a partir de ondas sinusoidales y semiondas. Las medidas tomadas indican que la velocidad del robot ápodo depende de la amplitud y la longitud de la onda empleadas, así como la estabilidad en el movimiento. Para demostrar la versatilidad en la locomoción, hemos hecho que Cube Reloaded pase por un circuito de pruebas y cambiando las características de la onda generadora, consigue superarlos.

5.2. Trabajos futuros

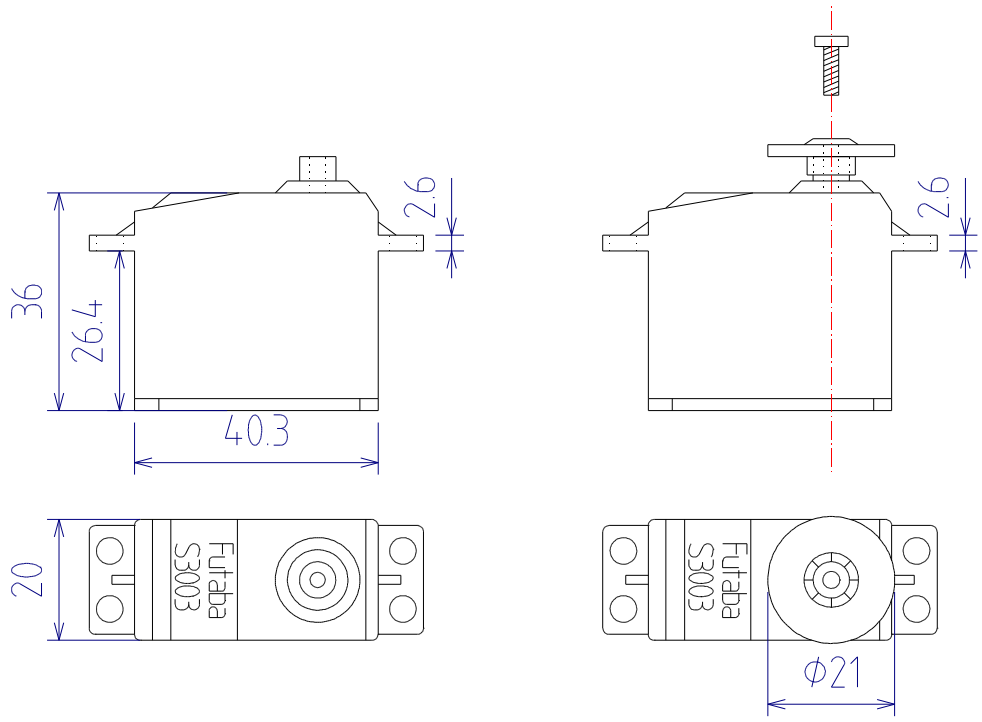
- Ampliar la longitud del gusano a 8 ó 12 segmentos. Modificación del software para que la generación sea independiente de la longitud y que se puedan implementar nuevos patrones de moviiento como por ejemplo el tipo rueda.
- Estudio teórico sobre la locomoción en línea recta usando ondas que recorren el gusano: determinación de condiciones de estabilidad, así como las ecuaciones de velocidad. Comparar los diferentes patrones de movimiento que se pueden generar.
- Diseñar un nuevo módulo tipo Nodo, como los de Polypod y polybot que permita no sólo construir robots tipo cadena, sino robots más complejos, como arañas.
- Desarrollar un modelo físico del gusano para generar patrones de movimiento usando algoritmos genéticos
- Diseñar una segunda generación de módulos con la electrónica en su interior, usando exclusivamente FPGAs, aumentado la versatilidad de estos robots, puesto que no hay que ceñirse a un microprocesador específico.

Apéndice A

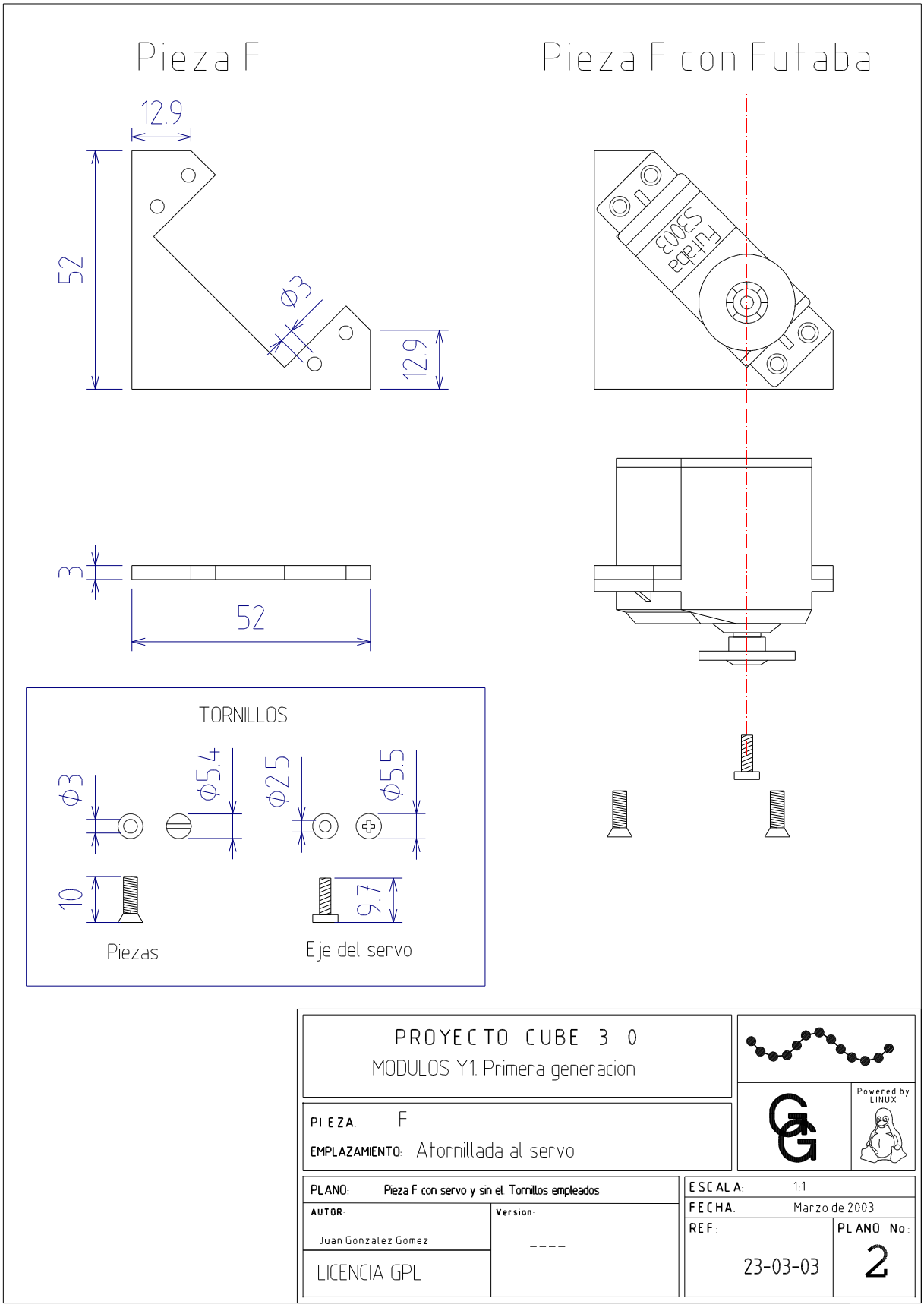
Planos del módulo Y1

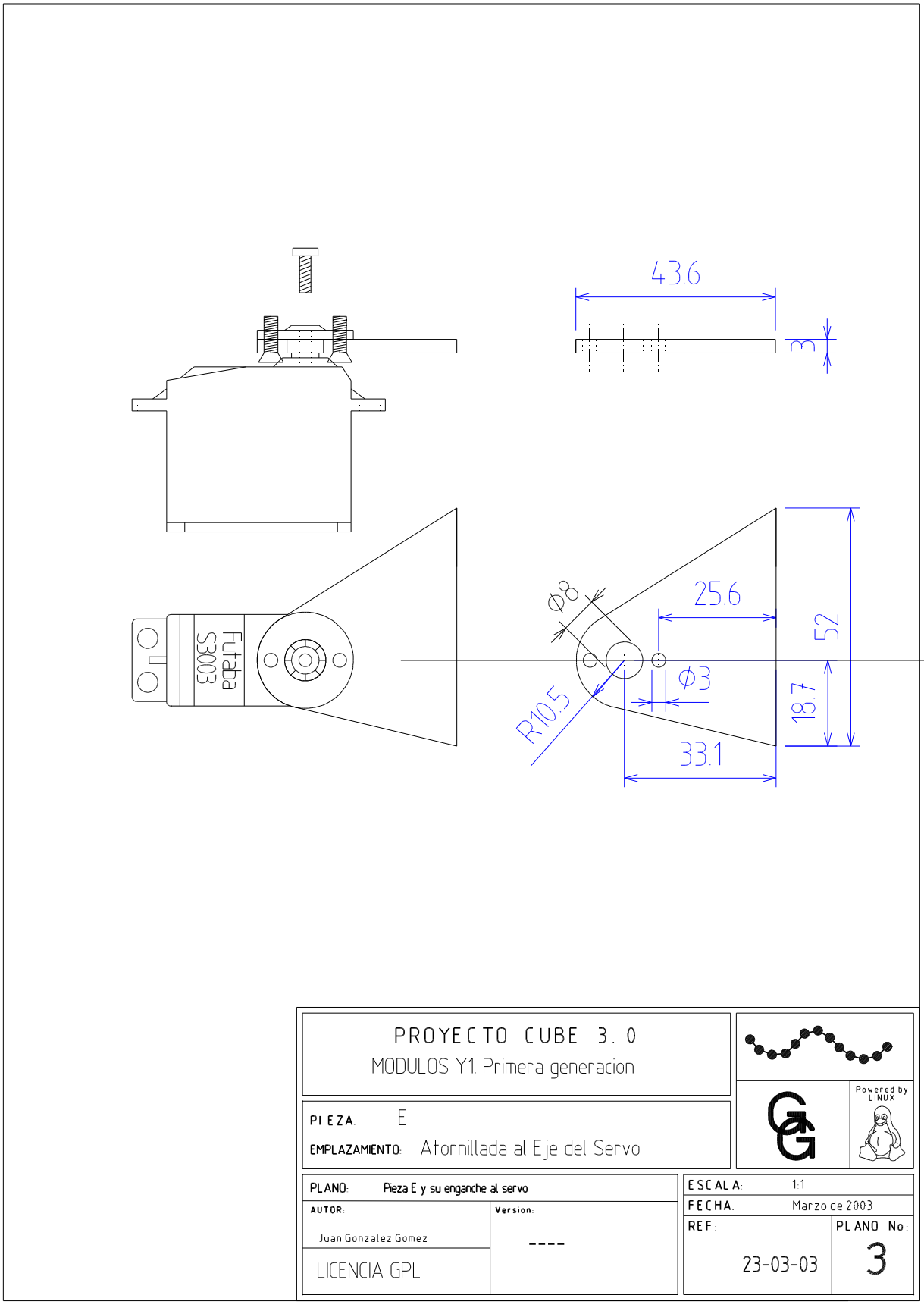
Los planos aquí mostrados no están a escala 1:1, se han reducido un 20 % para su correcta integración en este apéndice. Los planos incluidos son los siguientes:

1. Servomecanismos Futaba 3003
2. Pieza F
3. Pieza E
4. Piezas B1 y B2
5. Pieza FE
6. Módulo montado (I)
7. Módulo montado (II)
8. 4 Módulos Y1 conectados en fase
9. Tornillos empleados
10. Plantillas para la fabricación de 2 módulos.

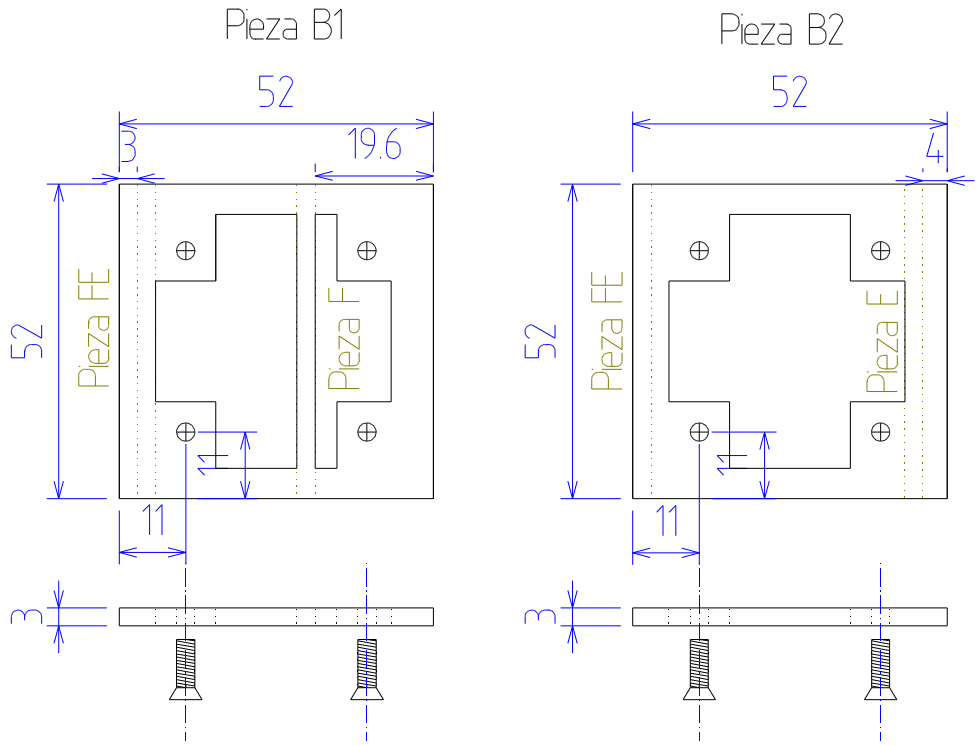


PROYECTO CUBE 3.0 MODULOS Y1. Primera generacion		
PIEZA: Servo Futaba 3003 EMPLAZAMIENTO:		
PLANO: Servo futaba 300 con pieza atornillada al eje	ESCALA: 1:1	
AUTOR: Juan Gonzalez Gomez	Version: ----	FECHA: Marzo de 2003
LICENCIA GPL		REF: 23-03-03 / PLANO No: 1

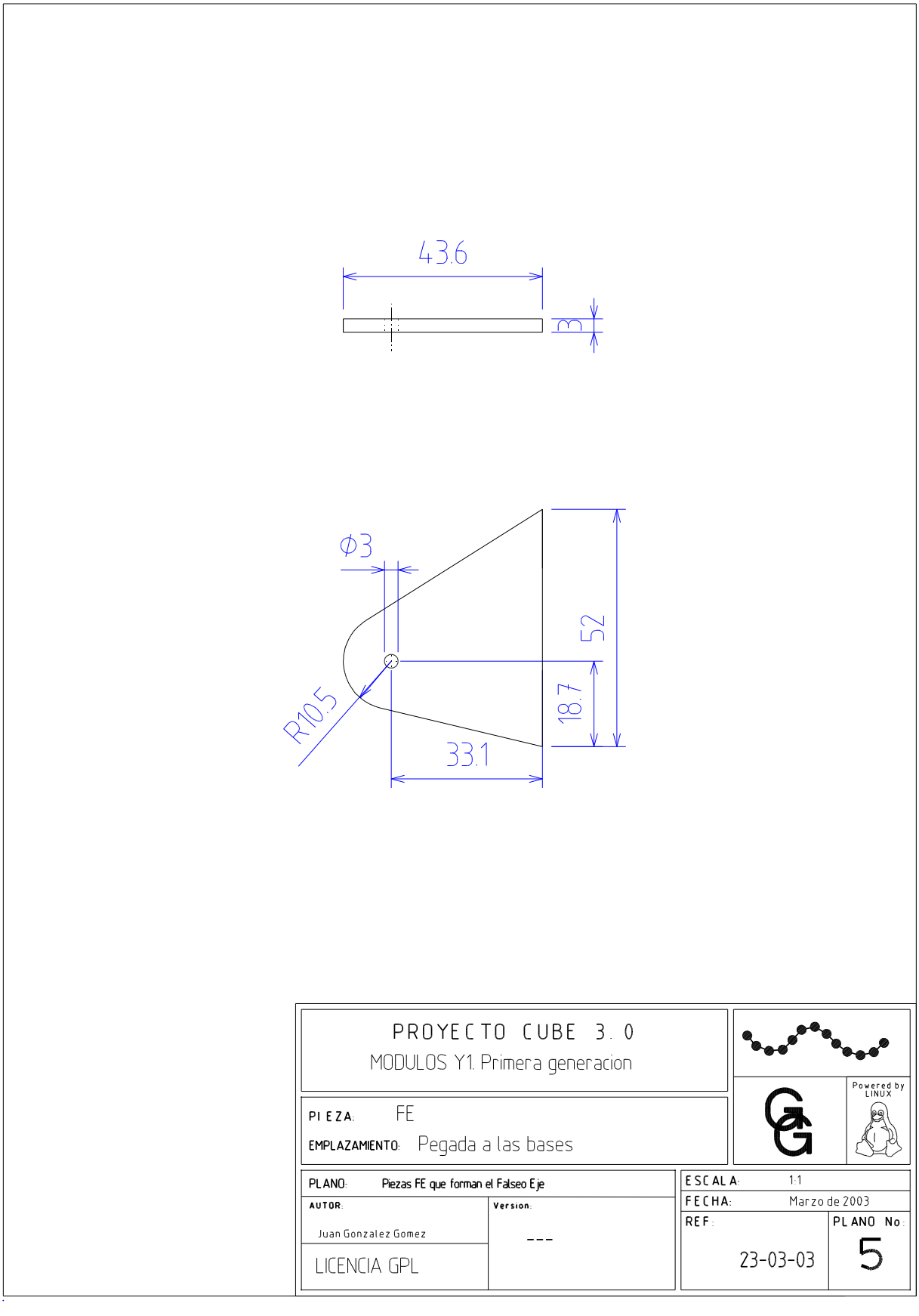

Pieza F
Pieza F con Futaba
12.9
52
Ø3
12.9
Futaba S3003
3
52
TORNILLOS
Ø3
Ø5.4
Ø2.5
Ø5.5
10
9.7
Piezas
Eje del servo
PROYECTO CUBE 3.0
MODULOS Y1. Primera generacion
PIEZA: F
EMPLAZAMIENTO: Atornillada al servo
PLANO: Pieza F con servo y sin el. Tornillos empleados
AUTOR: Juan Gonzalez Gomez
Version: ----
LICENCIA GPL
ESCALA: 1:1
FECHA: Marzo de 2003
REF: 23-03-03
PLANO No: 2
Powered by LINUX

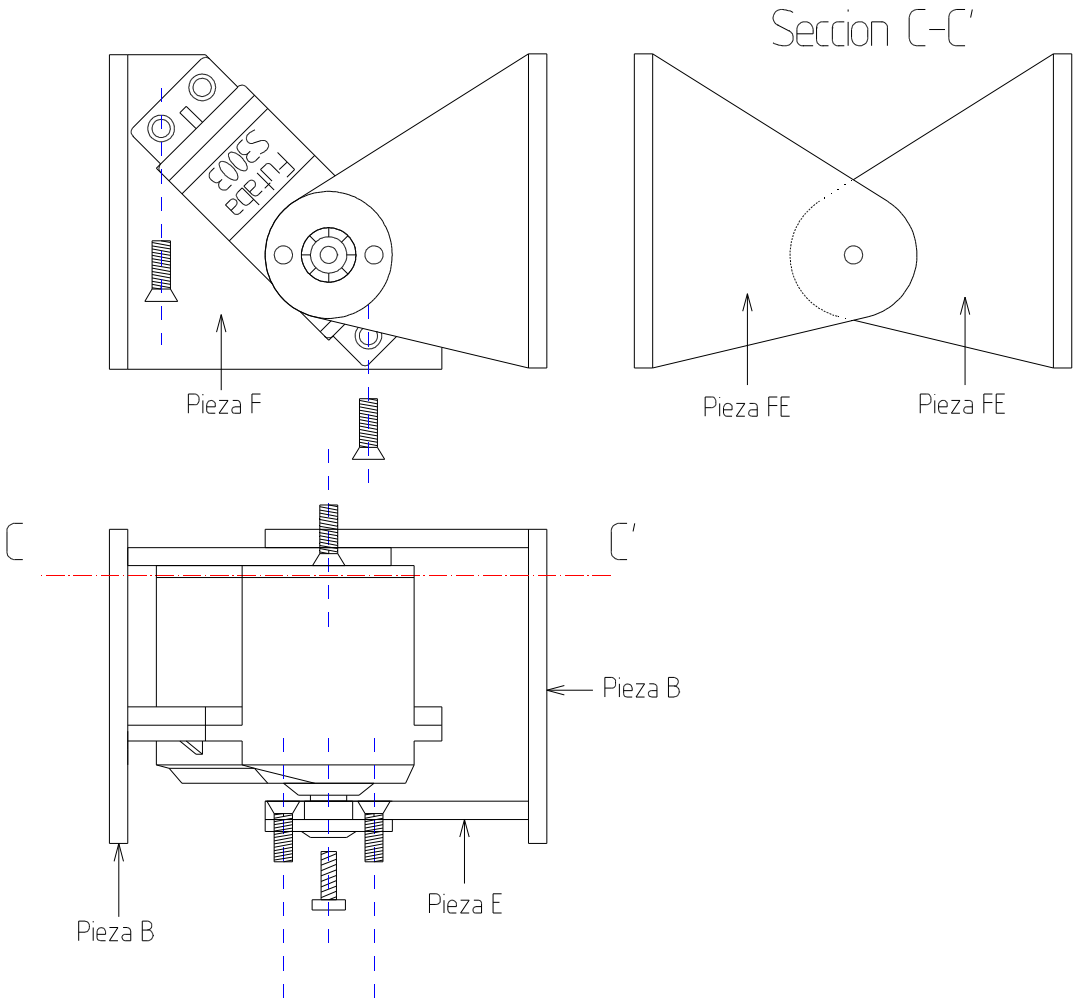


PROYECTO CUBE 3.0 MODULOS Y1. Primera generacion			
PIEZA: E EMPLAZAMIENTO: Atornillada al Eje del Servo			
PLANO: Pieza E y su enganche al servo		ESCALA: 1:1	
AUTOR: Juan Gonzalez Gomez	Version: ----	FECHA: Marzo de 2003	
LICENCIA GPL		REF: 23-03-03	PLANO No: 3

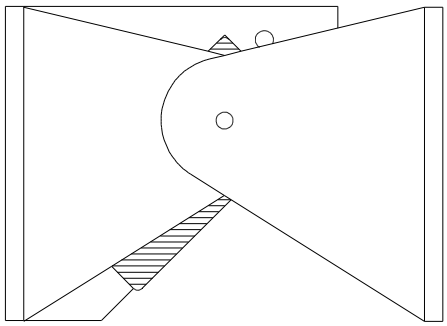


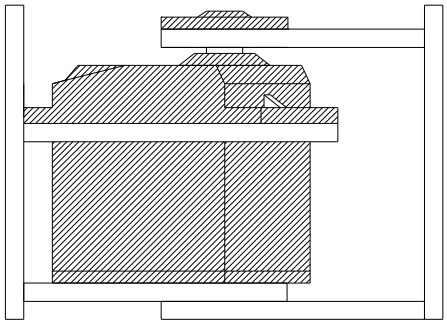
PROYECTO CUBE 3.0 MODULOS Y1. Primera generacion		
PIEZAS: B1 y B2 EMPLAZAMIENTO: Pegadas a piezas F, FE y E		
Piezas de union entre modulos		ESCALA: 1:1
AUTOR: Juan Gonzalez Gomez	Version: ----	FECHA: Marzo de 2003
LICENCIA GPL		REF: 30-03-03 / PLANO No: 4


43.6
3
Φ3
52
18.7
R10.5
33.1
PROYECTO CUBE 3.0
MODULOS Y1. Primera generacion
PIEZA: FE
EMPLAZAMIENTO: Pegada a las bases
Powered by LINUX
PLANO: Piezas FE que forman el Falseo Eje
AUTOR: Juan Gonzalez Gomez
Version: ---
LICENCIA GPL
ESCALA: 1:1
FECHA: Marzo de 2003
REF: 23-03-03
PLANO No: 5



PROYECTO CUBE 3.0 MODULOS Y1. Primera generacion	
PIEZA: EMPLAZAMIENTO:	
PLANO: Modulo 02 montado	ESCALA: 1:1
AUTOR: Juan Gonzalez Gomez	Version: ----
	FECHA: Marzo de 2003
LICENCIA GPL	REF: 26-03-03
	PLANO No: 6





PROYECTO CUBE 3.0

MODULOS Y1. Primera generacion

PIEZA:

EMPLAZAMIENTO:

Powered by LINUX

PLANO: Modulo montado. Posicion natural en el gusano

AUTOR: Juan Gonzalez Gomez

Version: ----

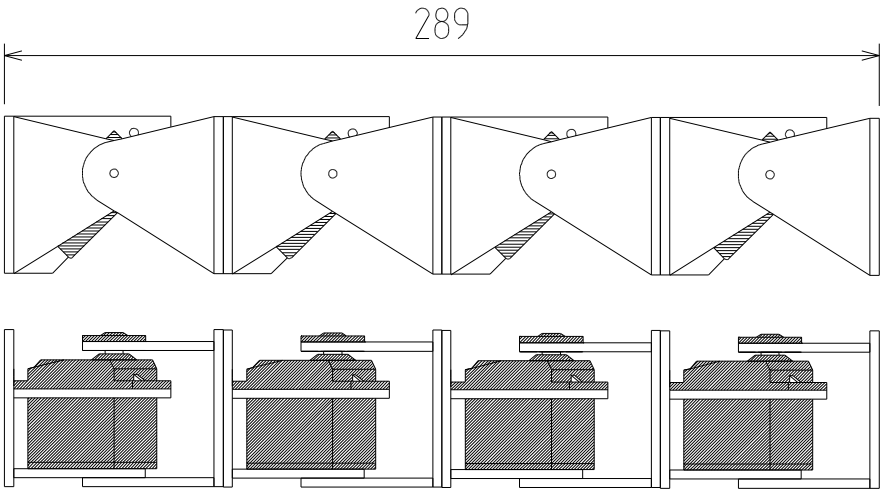
LICENCIA GPL

ESCALA: 1:1

FECHA: Abril de 2003

REF: 10-04-03

PLANO No: 7



PROYECTO CUBE 3.0

MODULOS Y1. Primera generacion

PIEZA:

EMPLAZAMIENTO:

Powered by LINUX

PLANO: 4 modulos unidos en fase

ESCALA: 1:2

FECHA: Abril de 2003

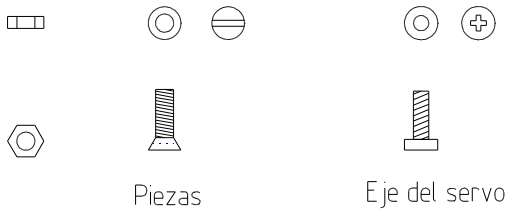
AUTOR: Juan Gonzalez Gomez

Version: ----

REF: 11-04-03

PLANO No: 8

LICENCIA GPL



PROYECTO CUBE 3.0

MODULOS Y1. Primera generacion

PIEZA:

EMPLAZAMIENTO:

Powered by LINUX

PLANO: Tornillos y tuercas empleados

AUTOR: Juan Gonzalez Gomez

Version: ----

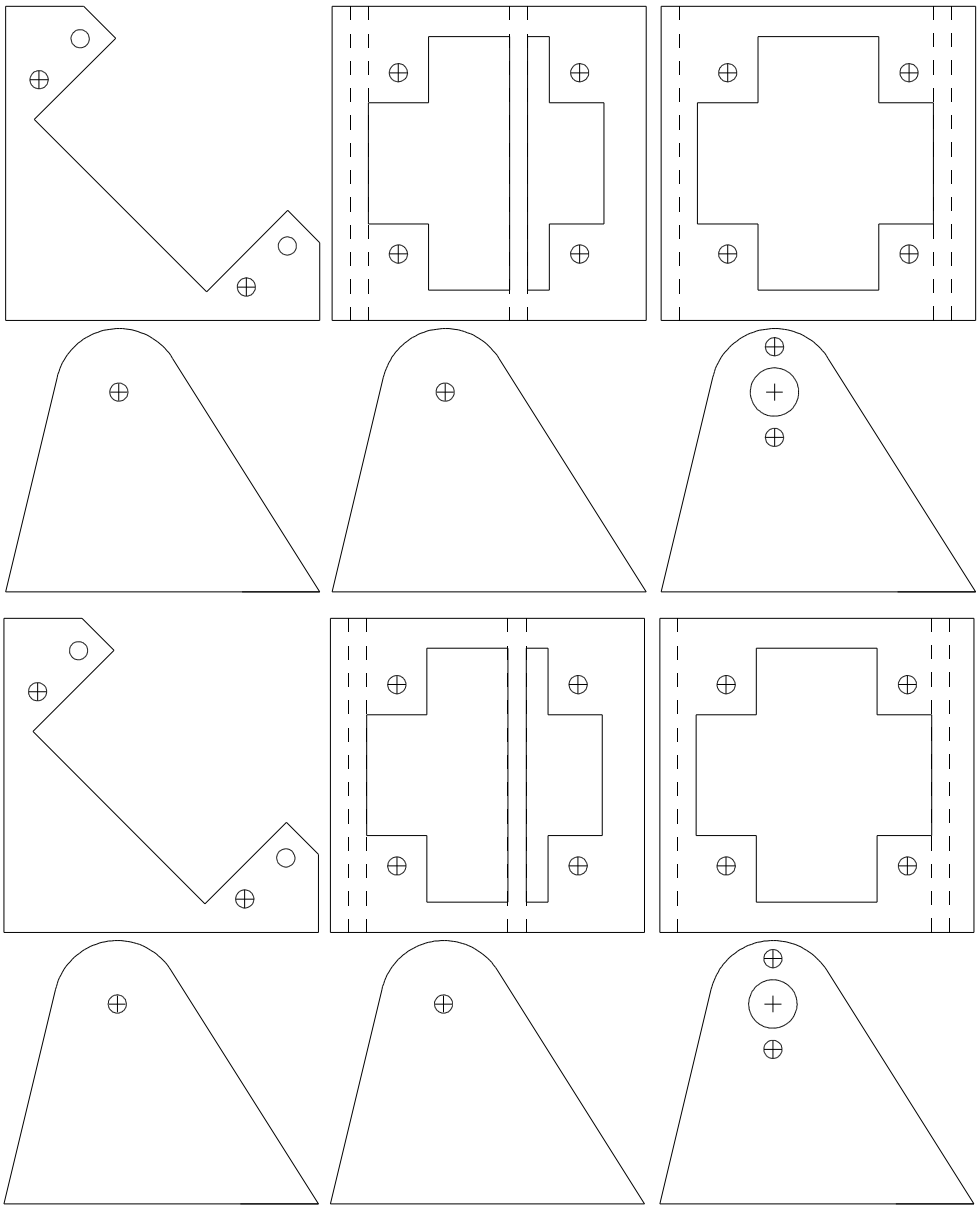
LICENCIA GPL

ESCALA: 1:1

FECHA: Abril de 2003

REF: 17-04-03

PLANO No: 9



Apéndice B

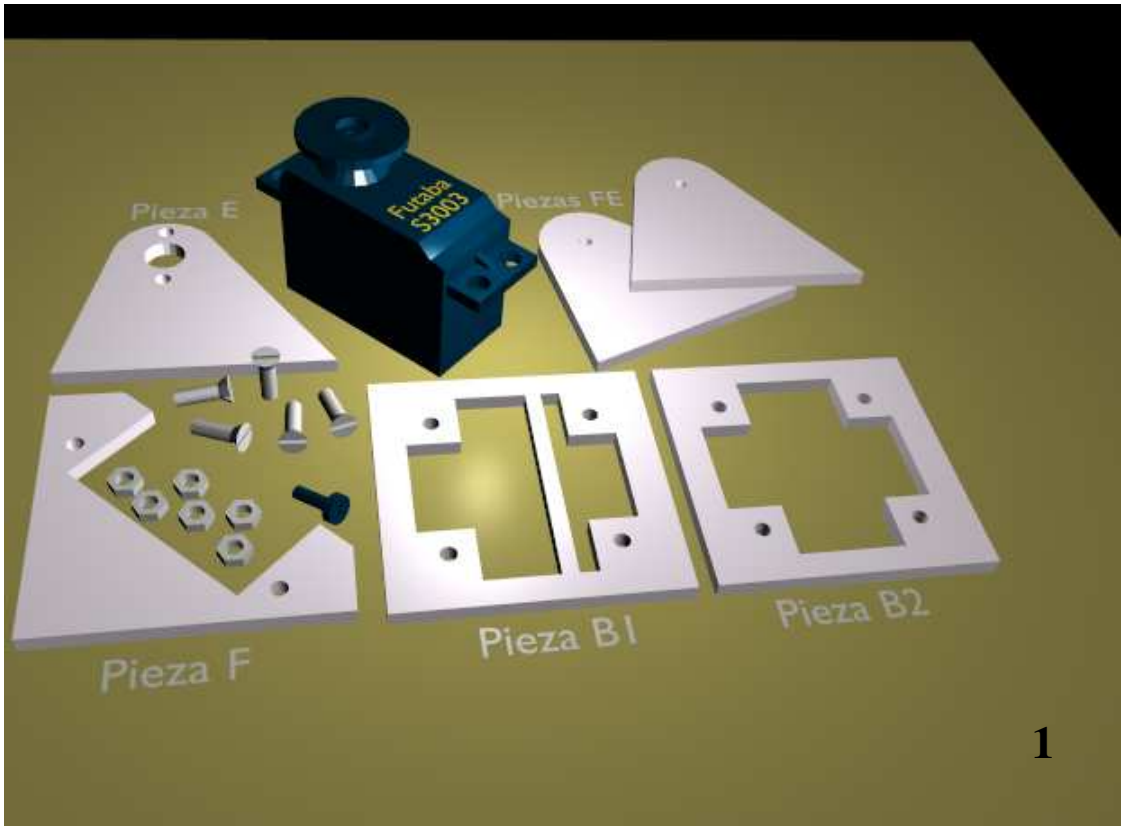
Montaje de los módulos Y1

En este apéndice se describe cómo montar los módulos Y1 a patir de las piezas básicas. Las piezas se pueden obtener de las siguientes maneras:

1. Construírselas uno mismo. Para ello lo mejor es imprimir la plantilla de la piezas sobre un papel A4 transparente tipo pegatina. Pegarlo sobre una lámina del material empleado para su construcción (PVC expandido, metacrilato, etc). Cortar las piezas y hacer los taladros. Para los primeros prototipos hemos utilizado una segueta de las usadas para cortar madera.
2. Dar la plantilla a una tienda en la que realicen corte de las piezas. Esta solución es la mejor construir muchos módulos.

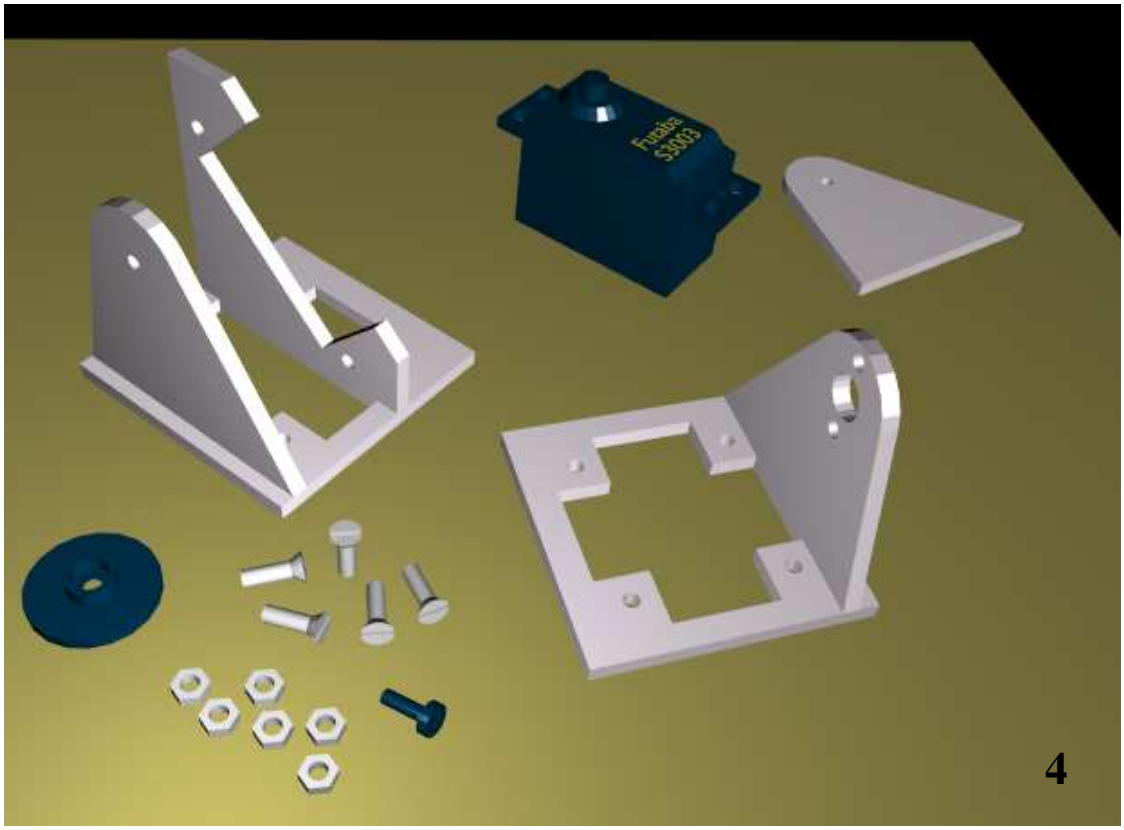
Una vez que se dispone de todas las piezas, se puede comenzar con el montaje, resumido en los siguientes pasos:

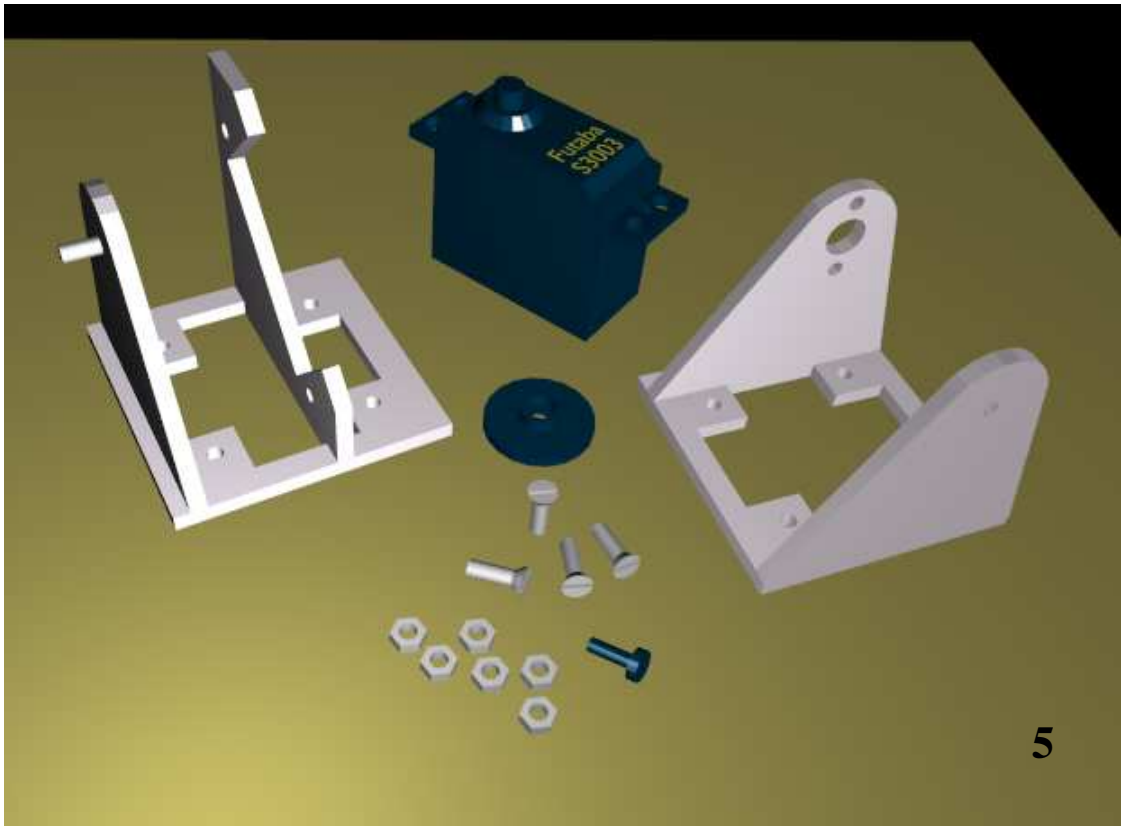
1. **Material necesario**. Dos piezas FE, una E, una F, una B1 y una B2. En total 6. Un servomecanismo del tipo Futaba 3003 con una corona de Φ21mm y su tornillo (Incluidos con el servo). Además son necesarios 5 tonillos de Φ3mm y 10mm de largo y 6 tuercas para esos tornillos.
2. **Paso 2**. Las piezas B1 y F se pegan usando un pegamento especial para plásticos.
3. **Pase 3**. Una de las piezas FE se pega también a la base B1. Obsérvese que no está justo en el extremo, sino unos milímetros hacia el interior.
4. **Paso 4**. La pieza E se pega a la B2, pero no junto al extremo, sino alineada con la cara interior.
5. **Paso 5**. La otra pieza FE se pega también a la B2, pero esta sí está alineada con la cara exterior. Un tornillo se sitúa en la otra pieza FE, que hará de falso eje.
6. **Paso 6**. La corona del servo se atornilla a la pieza E.
7. **Paso 7**. El servo se atornilla la pieza F. Con esto se obtiene una de las dos partes móviles del módulo. Esta parte se denomina **cuerpo del módulo**. La otra parte, que se atornilla al eje del servo y al false eje, se denomina **Cabeza del módulo**.
8. **Paso 8**. Finalmente se unen el cuerpo y la cabeza para obtener el módulo definitivo. Las dos tuercas se ponen en el falso eje.

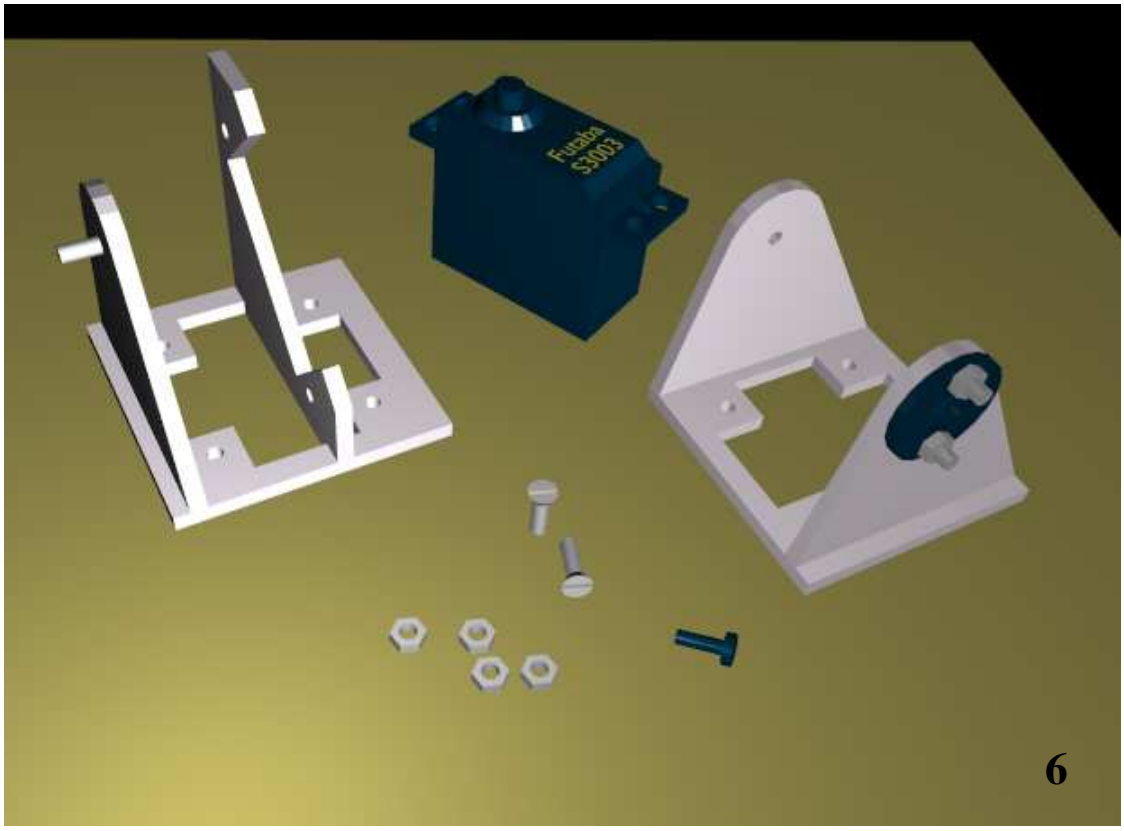

Pieza E
Futaba S3003
Piezas FE
Pieza F
Pieza B1
Pieza B2
1


Futaba S3003
2

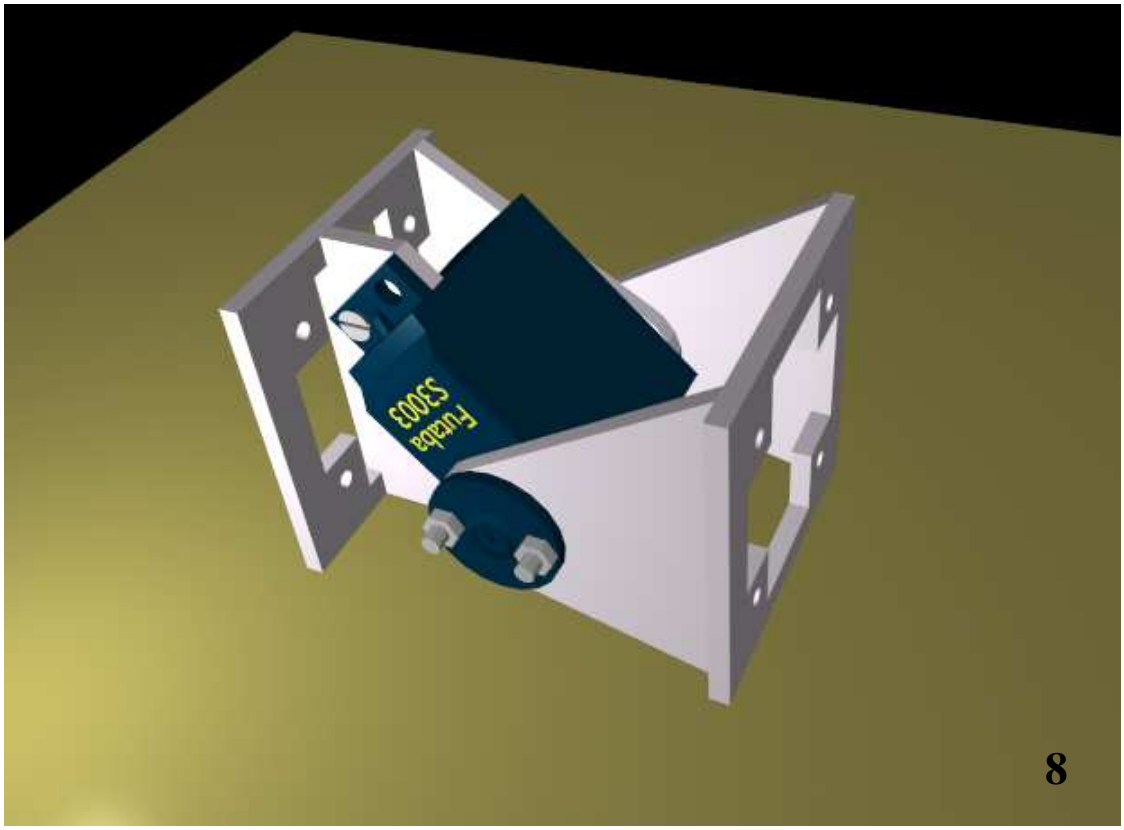

Futaba
S3003
3


Futaba
S3003
4


Futaba
S3003
5


Futaba
S3003
6

7



8

Apéndice C

Control de 8 servos con un único comparador

En este apéndice describimos el funcionamiento del programa servidor que se ejecuta en un 6811 y que permite controlar hasta 8 servos usando un único comparador. Este es el servidor que se está usando para Cube Reloaded. Las ideas son las siguientes:

1. El intervalo de 20ms (50Hz) se divide en 8 "ventanas" de 2.5ms (400Hz). En cada ventana se genera la señal para controlar un servo. El ancho máximo de la señal del servo es 2.3ms por lo que cabe perfectamente en la ventana. Cada vez que se produce una interrupción que te indica el comienzo de una nueva ventana, hay que temporizar la anchura (que se puede hacer con otro comparador diferente). Así, la primera interrupción te indica que hay que actualiar el servo 1, la segunda interrupción el servo 2... así hasta el servo 8 y vuelta a empezar.

2. Cada ventana de 2.5ms se puede dividir en dos partes, una en la que la señal PWM está a nivel alto (la anchura es la posición POSI) y otra en la que la señal está a nivel bajo (2.5ms - POSI). La anchura total es de 2.5ms, pero entre medias se ha producido otra interrupción que te marca el paso del PWM del servo actual a 0.

En la figura C.1 se muestran las 8 ventanas de tiempo en las que se divide el periodo de la señal PWM. Cada ventana está asociada a un servo, en la que se genera la anchura del pulso. Obsérvese que en cada ventana sólo hay una señal activa.

En la figura C.2 se muestran las dos partes en las que se divide una ventana. En la parte de la izquierda la señal PWM está a 1 (Estado S1) y en la parte de la izquierda está a 0 (S0). La anchura de cada parte está perfectamente determinada, ya que al ser la anchura de la ventana fija (2.5ms) si la anchura del pulso es de Ton, entonces la duración del estado S0 es de 2.5ms - Ton. Las duraciones de cada pulso se encuentra almacenadas en una tabla. Cada fila de la tabla indica el anchu del pulso dentro de la ventana de tiempo correspondiente. Así la primera fila de la tabla indica la duración del pulso para la ventana 1, la fila 2 para la ventana 2, etc.

La programación del microcontrolador para poder generar las 8 señales PWM se puede hacer con una única interrupción, que tiene que hacer lo siguiente:

- **Llega interrupción del comparador**
- **Si estamos en el estado S1**:

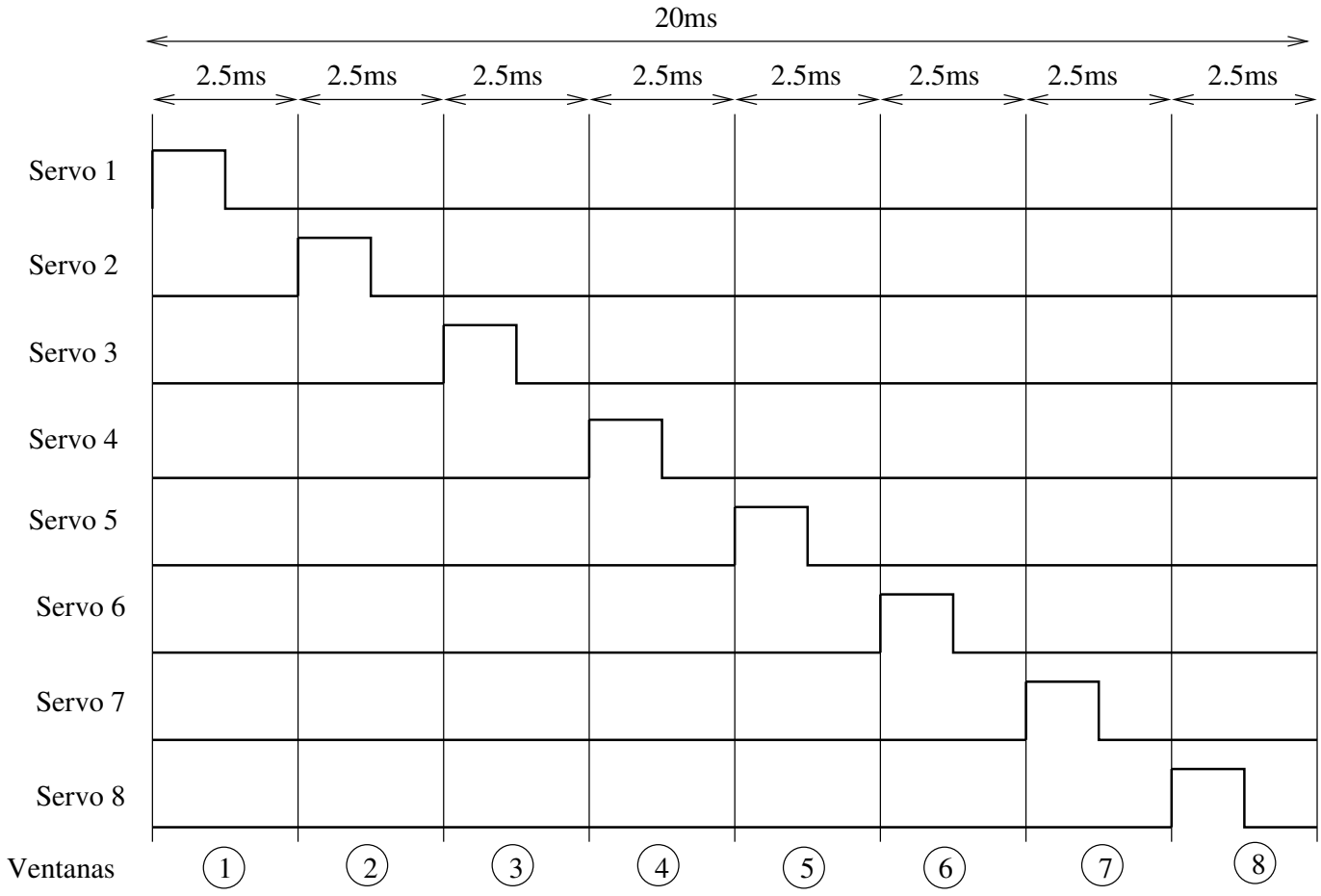


Figura C.1: Periodo de 20ms dividido en 8 ventanas de tiempo

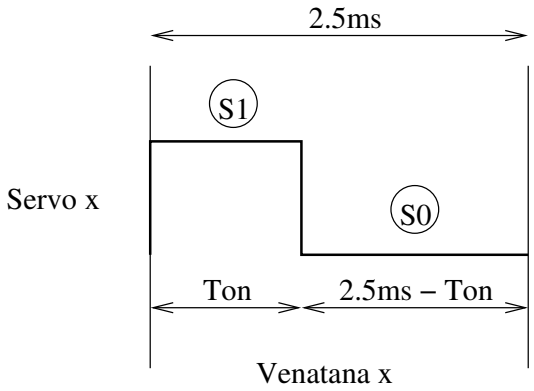


Tabla de posiciones PWM

Servo 1	Ton_1
Servo 2	Ton_2
Servo 3	Ton_3
Servo 4	Ton_4
Servo 5	Ton_5
Servo 6	Ton_6
Servo 7	Ton_7
Servo 8	Ton_8

Figura C.2: Las dos partes dentro de cada ventana y la tabla de PWM

- Poner la señal correspondiente al servo actual (ventana actual) a 1
- Activar el comparador para que la siguiente interrupción sea dentro de Ton_1 ms
- Pasar al estado S0.
- Terminar

- **Si estamos en el estado S0**

  - Poner señal correspondiente al servo actual (ventana actual) a 0
  - Activar comparador para interrumpir dentro de 2.5 - Ton_1
  - Incrementar puntero de la tabla PWM, para apuntar a la posición del siguiente servo
  - Pasar al estado S1.
  - Terminar

Apéndice D

Unidad hardware de PWM

Se ha diseñado en VHDL y está compuesta por las siguientes entidades:

- Contador.vhd. Contador de 11 bits.
- comparador.vhd.
- presmod10.vhd. Un divisor de frecuencia entre 10

Contador.vhd:

```
------------------------------------------------------------
-- contador.vhd.  Juan Gonzalez. Feb-2002                 --
-- Licencia GPL.                                          --
------------------------------------------------------------
-- PROYECTO LABOBOT                                       --
------------------------------------------------------------
-- Contador de 11 bits                                    --
--                                                        --
-- Entradas: clk   : Reloj                                --
--           reset : Puesta a cero asíncrona (Nivel bajo) --
-- Salidas:                                               --
--       -q  : Datos de salida                            --
------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity contador is
     port (
                clk   : in std_logic; -- Reloj
                reset : in std_logic;
                q     : out std_logic_vector (10 downto 0)); --Salida
end contador;
architecture beh of contador is

        signal cuenta : std_logic_vector(10 downto 0);


begin
```

```
        output: process(clk,reset)
        begin
                -- Actualizar la cuenta
                if (reset='0') then        -- Reset asíncrono
                        cuenta <= (others=>'0');   -- Inicializar contador
                elsif (clk'event and clk='1') then  -- Flanco de subida en reloj
                        cuenta <= (cuenta+1);      -- Incrementar contador
                end if;
        end process;

        q <= cuenta;


end beh;
```

Comparador.vhd:

```
------------------------------------------------------
-- comparador.vhdl.  Juan Gonzalez. Mayo-2003       --
-- Licencia GPL.                                    --
------------------------------------------------------
-- PROYECTO CUBE-FPGA                               --
------------------------------------------------------
-- Comparador para la implementacion de una unidad  --
-- de PWM.                                          --
------------------------------------------------------
-- Comparador de 8 bits                             --
------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
entity comparador is
  port (opa : in std_logic_vector(10 downto 0);  -- Operador A
        opb : in std_logic_vector(7 downto 0);   -- Operador B
        o   : out std_logic);                    -- Salida pwm
end comparador;
architecture beh of comparador is
signal sg  : std_logic;
signal pos : std_logic_vector(7 downto 0);
signal msb : std_logic_vector(2 downto 0);
begin
 -- Obtener los 8 bits menos significativos de la entrada a
 -- (la posicion en la que se quiere poner el servo)
  pos<=opa(7 downto 0);
 -- Este es el comparador
  process (pos, opb)
  begin
    if (pos<opb) then
      sg <= '0';
    else
      sg <= '1';
    end if;
  end process;
```

```
  -- Señal PWM de salida
  o<= (not opa(8)) and (not opa(9)) and (not opa(10))
           and (not sg);
end beh;
```

presmod10.vhd:

```
------------------------------------------------------------
-- presmod10.vhdl.  Juan Gonzalez. Mayo-2003          --
-- Licencia GPL.                                      --
------------------------------------------------------------
-- Prescaler modulo 10 (divide señal por 10)          --
--                                                    --
-- Entradas: clk : Senal de reloj                     --
--           clear: Puesta a cero asincrona (NB)      --
-- Salidas:                                           --
--      -q: Señal de salida                           --
------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity presmod10 is
  port (clk     : in std_logic; -- Reloj
        clear   : in std_logic;
        q       : out std_logic); --Salida
end presmod10;
architecture beh of presmod10 is
signal cuenta : std_logic_vector (3 downto 0);
begin
  output: process(clk,clear)
  begin
    if (clear='0') then
      q<='0';
      cuenta<="0000";
    elsif (clk'event and clk='1') then
      if cuenta="1001" then
        cuenta<="0000";
        q<='0';
      else
        cuenta<=(cuenta+1);
        q<=cuenta(3);
      end if;
    end if;
  end process;
end beh;
```

Bibliografía

[1] Robot Tritt. http://www.microbotica.com/kitbot.htm

[2] Robot de docencia. Desarrollado en la EPS de la UAM. http://www.ii.uam.es/~mecatron/index.php3?seccion=4&pagina=8. [última consulta: 4/Junio/2003]

[3] A. Prieto-Moreno. "Diseño, construcción y control de un robot articulado mediante una red de microcontroladores". PFC. ETSIT-UPM. 2001.

[4] Robot hexápodo Sheila. http://www.iearobotics.com/personal/juan/proyectos/sheila/sheila.html. [Última consulta: 4/Junio/2003]

[5] J. González. "Diseño y construcción de un robot articulado que emula modelos animales: aplicación a un gusano". PFC. ETSIT-UPM. 2001.

[6] Robot Cube 2.0 en línea. http://www.iearobotics.com/personal/juan/proyectos/cube-2.0/cube.html

[7] Carnegie Mellon University (CMU).http://www.cmu.edu/

[8] Robot Ambler en el CMU.http://www.ri.cmu.edu/projects/project_362.html. [Última consulta: 22/Mayo/2003]

[9] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W.L. Whittaker, "Ambler: An Autonomous Rover for Planetary Exploration," IEEE Computer, Vol. 22, No. 6, June, 1989, pp. 18-26. Disponible on-line en: http://www.ri.cmu.edu/pubs/pub_1431.html#abstract

[10] J. Bares and W. Whittaker, "Configuration of an Autonous Robot for Mars Exploration," Proc. World Robotics Conf., Society of Mechanical Engineers, Gaithersburg, Md., May 1989.

[11] Robot Dante II.http://www.ri.cmu.edu/projects/project_163.html. [Ultima consulta: 23/Mayo/2003]

[12] J. Bares and D. Wettergreen, "Dante II: Technical Description, Results and Lessons Learned," International Journal of Robotics Research, Vol. 18, No. 7, July, 1999, pp. 621-649.

[13] Página personal de Mark Yim. http://robotics.stanford.edu/users/mark/.[Última consulta: 21/mayo/2003]

[14] Universidad de Stanford http://www.stanford.edu/. [Última consulta: 21/mayo/2003]

[15] M. Yim."Locomotion with a unit-modular reconfigurable robot". Ph.D. thesis, Stanfofd University. Dic,1995. Disponible on-line en http://www-db.stanford.edu/TR/CS-TR-95-1536.html [Última consulta: 21/Mayo/2003].

[16] PolyPod. Página en Stanford. http://robotics.stanford.edu/people/mark/polypod.html. [Última consulta: 21/Mayo/2003]

[17] PolyPod. Página del Parc.http://www2.parc.com/spl/projects/modrobots/chain/polypod/index.html. [Última consulta: 04/Junio/2003]

[18] "Locomotion Gaits with Polypod", Video Proc. of the IEEE Intl. Conf. On Robotics and Automation, San Diego, CA. 1994

[19] Investigaciones de robótica en el Hirose & Yoneda Lab. http://www-robot.mes.titech.ac.jp/research/ [Ultima consulta: 23/Mayo/2003]

[20] Hirose & Yoneda Lab. Snake-like robots.http://www-robot.mes.titech.ac.jp/research/snake/snake.html. [Ultima consulta: 23/Mayo/2003]

[21] ACM III. Hirose & Yoneda Lab. http://www-robot.mes.titech.ac.jp/research/snake/acm3/acm3.html

[22] "Biologically Inspired Robots (Snake-like Locomotor and Manipulator)". Shigeo Hirose. Oxford University Press, 1993.

[23] "Biomechanical Study of Active Cord-Mechanism with Tactile Sensors". Yoji Umetani. Int. Symp. on Industrial Robots, Nottingham. pp.c1-1-c1-10 (1976)

[24] SG(Soft Gripper). Hirose & Yoneda Lab.http://www-robot.mes.titech.ac.jp/research/snake/sg/sg.html

[25] Oblix.http://www-robot.mes.titech.ac.jp/research/snake/oblique/oblique.html

[26] RMMS. *http://www.ri.cmu.edu/projects/project_105.html

[27] Robots tipo retículo en el PARC.http://www2.parc.com/spl/projects/modrobots/lattice/index.html. [Última consulta: 7/Jun/2003]

[28] Taxonomía de la locomoción estáticamente estable, en el PARC. http://www2.parc.com/spl/projects/modrobots/chain/polypod/locomotion.html. [última consulta: 6/Jun/2003]

[29] Mark Yim, Ying Zhang & David Duff, Xerox Palo Alto Research Center (PARC), "Modular Robots". IEEE Spectrum Magazine. Febrero 2002.http://www.spectrum.ieee.org/WEBONLY/publicfeature/feb02/mrobo.html [Última consulta: 4/jun/2003].

[30] Mark Yim, David G. Duff, Kimon D. Roufas,"PolyBot: a Modular Reconfigurable Robot", IEEE Intl. Conf. on Robotics and Automation (ICRA), San Francisco, CA, April 2000. Disponible on-line en http://www2.parc.com/spl/projects/modrobots/publications/index.html. [Última consulta: 5/Jun/2003]

[31] PolyBot. Página del Parc.http://www2.parc.com/spl/projects/modrobots/chain/polybot/index.html. [Última consulta: 5/Jun/2003]

[32] Demostraciones de Polybot G1. http://www2.parc.com/spl/projects/modrobots/chain/polybot/demonstrations/g1demos.html#reconf. [Última consulta: 8/Jun/2003]

[33] "Two Approaches to Distributed Manipulation", M. Yim, J. Reich, A. Berlin, Distributed Manipulation, ed. by H. Choset and K. Bohringer, Kluwer Academic Publishing, 2000

[34] "Climbing with Snake-like Robots", Proc. of IFAC Workshop on Mobile Robot Technology, Jejudo, Korea, May 2001

[35] Demostraciones de Polybot G2. http://www2.parc.com/spl/projects/modrobots/chain/polybot/demonstrations/g2demos.html#reconf[Última consulta: 8/Jun/2003]

[36] QCAD. Programa de diseño en 2D Multiplataforma. http://www.qcad.org/. [Última consulta: 9/Jun/2003]

[37] BLENDER. Programa de diseño en 3D Multiplataforma. http://www.blender3d.com/. [Última consulta: 9/Jun/2003]

[38] Tarjeta BT6811. http://www.iearobotics.com/proyectos/bt6811/bt6811.html. [Última consulta: 9/Jun/2003]

[39] Tarjeta CT6811. http://www.iearobotics.com/proyectos/ct6811/ct6811.html. [Ultima consulta: 9/Jun/2003]

[40] J. Gonzalez, P. Haya, S. Lopez-Buerdo, and E. Boemo, "Tarjeta entrenadora para FPGA, basada en un hardware abierto", Seminario Hispabot 2003, Alcalá de Henares, Mayo 2003.

[41] Tarjeta JPS. http://www.iearobotics.com/personal/juan/doctorado/jps-xpc84/jps-xpc84.html. [Última consulta: 11/Jun/2003]

[42] Proyecto Labobot. http://www.iearobotics.com/personal/juan/doctorado/labobot/labobot.html. [Última consulta: 11/Jun/2003]

[43] Proyecto Labobot en la web de Labotmat. http://www.ii.uam.es/~laboweb/LabWeb/index.php3?seccion=1&pagina=5. [Última consulta: 11/Jun/2003]

[44] Aplicación Cube-fisico. http://www.iearobotics.com/personal/juan/proyectos/cube-2.0/control_i.html. [Última consulta: 9/Jun/2003]

[45] Aplicación Cube-virtual. http://www.iearobotics.com/personal/juan/proyectos/cube-2.0/control_ii.html. [Última consulta: 9/Jun/2003]

[46] XS40, XSP Board V1.4 User Manual, XESS Corp, Apex, NC, 1999 http://www.xess.com/.

[47] I. González, J. González, E. Boemo, “Alternativas hardware para la locomoción del robot ápodo Cube Reloaded”. Borrador de artículo para el Seminario JCRA’03, UAM, Septiembre 2003.