



Escola Universitària  
Politècnica de Mataró

**Ingeniería Técnica Industrial: Especialidad Electrónica Industrial**

**PROYECTO MIHRO**  
**(MOBILE INTELLIGENT HEXAPOD ROBOT)**  
**VOLUMEN I**

**IGNACIO PEDROSA LOJO**  
**JULIÁN HORRILLO TELLO**

OTOÑO 2008





Escola Universitària  
Politècnica de Mataró

## VOLUMEN I

## MEMORIA DESCRIPTIVA

**IGNACIO PEDROSA LOJO  
JULIÁN HORRILLO TELLO**



## **Resumen**

Este proyecto trata sobre el proceso de diseño y construcción de un robot hexápedo de tres grados de libertad por pata. Su desarrollo engloba las tareas de estudio previo y análisis de otros proyectos similares, diseño CAD de toda su estructura mecánica, su construcción a partir de chapa de aluminio, el diseño electrónico del hardware y su programación en lenguaje ensamblador. La característica más destacable de su diseño mecánico es que otorga a sus seis patas la capacidad de detectar tanto el suelo como los posibles obstáculos que puedan interrumpir la trayectoria de cada paso. Su control está distribuido entre siete placas electrónicas: un cerebro, o “master”, y seis esclavos encargados de controlar a cada una de las patas, todas ellas basadas en el microcontrolador PIC16F88. El software desarrollado permite controlar dinámicamente la posición de cada articulación con una precisión de 0'72° mediante la generación de una señal PWM para cada servomotor estándar Futaba S3003. La comunicación entre los sistemas electrónicos se realiza a través de un bus I<sup>2</sup>C, por lo que se ha programado el soporte hardware que ofrece el PIC16F88 para los dispositivos en modo “slave” y se ha implementado el modo “master” enteramente por software. Además del módulo SSP (comunicación I<sup>2</sup>C), el programa hace uso del módulo Timer1 y de las interrupciones externa y por cambio de estado de los pines RB4:RB7 del microcontrolador. El robot se opera manualmente desde un teclado o mando conectado a su placa “master”.

## **Resum**

El projecte tracta sobre el procés de disseny i construcció d'un robot hexàpode de tres graus de llibertat per pota. El seu desenvolupament engloba les tasques d'estudi previ i analisi d'altres projectes similars, disseny CAD de tota la seva estructura mecànica, la seva construcció a partir de xapa d'alumini, el disseny electrònic del hardware i la seva programació en llenguatge assemblador. La característica més destacable del seu disseny mecànic és que atorga a les seves sis potes la capacitat de detectar tant el terra com els possibles obstacles que puguin interrompre la trajectòria de cada passa. El seu control està distribuït entre sis plaques electròniques: un cervell, o “master”, i sis esclaus encarregats de controlar cadascuna de les potes, totes elles basades en el microcontrolador PIC16F88. El software desenvolupat permet controlar dinàmicament la posició de cada articulació amb una precisió de 0'72° mitjançant la generació d'un senyal PWM per a cada servomotor estàndard Futaba S3003. La comunicació entre els sistemes electrònics es realitza a través d'un bus I<sup>2</sup>C, pel que s'ha programat el suport hardware que ofereix el PIC16F88 pels dispositius en mode “slave” i s'ha implementat el mode “master” completament per software. A més del mòdul SSP (comunicació I<sup>2</sup>C), el programa fa ús del mòdul Timer1 i de les interrupcions externa i per canvi d'estat dels pins RB4:RB7 del microcontrolador. El robot s'opera manualment des d'un teclat o comandament connectat a la seva placa “master”.

# **Abstract**

This project is about the designing and construction of an hexapod robot with three degrees of freedom each leg. Its development includes tasks such as previous study and analysis of other similar projects, CAD designing of all its mechanical structure, its construction using aluminium plating, the hardware's electronic designing and its programming in assembler language. The most remarkable feature of its mechanical designing is that it gives each one of its six legs the capacity of detecting not only the floor but also the possible obstacles that may interrupt the trajectory of every step. Its control is distributed in seven electronic plates: a brain, or master, and six slaves in charge of controlling each one of the legs, all of them based on the PIC16F88 microcontroller. The developed software allows dynamic control of the position of every articulation with a  $0.72^\circ$  precision by generating a PWM signal for each standard Futaba S3003 servomotor. The communication among the electronic systems is carried out through an I<sup>2</sup>C bus, so it has been programmed its hardware support on PIC16F88 for devices in "slave" mode, and the "master" mode has been entirely implemented by software. Aside from the SSP module (I<sup>2</sup>C communication), the program uses the Timer1 module and the external interruptions as well as the interrupt by change on the microcontroller's RB4:RB7 pins. The robot operates manually from a keyboard or a pad connected to its master board.



## Índice

---

[Página en blanco]

# Índice de Contenidos

<b>1</b>	<b>Prefacio</b>	<b>1</b>
<b>2</b>	<b>Introducción al Proyecto</b>	<b>2</b>
2.1	Motivación	2
2.2	Antecedentes	3
2.2.1	Proyectos EUPMT	3
2.2.2	Otros proyectos	4
2.3	Proyecto MIHRO	9
2.4	Contenido del proyecto actual	12
2.4.1	Objetivos	12
2.4.2	Esquema general de MIHRO	14
2.4.3	Factores limitantes	15
2.4.4	Contenido de la memoria	16
2.4.5	Contenido del CD	16
<b>3</b>	<b>Diseño mecánico</b>	<b>17</b>
3.1	Premisas	17
3.2	Evolución del diseño mecánico	18
3.2.1	Metodología y herramientas	18
3.2.2	Prototipos	19
3.2.3	Ensayos (maqueta de pruebas)	23
3.2.4	Características mecánicas finales	27
<b>4</b>	<b>Fabricación y montaje del esqueleto</b>	<b>30</b>
4.1	Proceso de modificación de los servos	32
4.2	Procedimientos de fabricación del esqueleto.	37
4.3	Proceso de montaje	57
4.3.1	Modificación de los Servos	57
4.3.2	Construcción del Cuello	57
	P10 – Cuello	57
	Montaje del Cuello	58

## Índice

---

4.3.3	Construcción del Tórax	58
	P00 – Tórax Base	59
	P01 – Tapa Superior	60
	P02 – Tapa Inferior	61
	P11 – Hombro Izquierdo	61
	P12 – Hombro Derecho	62
	Montaje del Tórax:	63
	Recuento de piezas de Tórax + Cuello	70
4.3.4	Construcción 1 de las patas (Cadera y Fémur)	70
	P03 – Conector Cadera - Tórax	71
	P04 – Conector Cadera – Fémur	71
	P05 – Fémur	72
	Montaje 1 de las patas	74
	Montaje de	77
	Recuento de piezas de los 6 módulos Cadera + Fémur	80
4.3.4	Construcción 2 de las patas (Tibias)	81
	P06 – Parte Fija de la Tibia	81
	P07 – Soporte Superior del Pié	84
	P08 – Parte Móvil de la Tibia	85
	P09 – Pié	89
	Montaje de las Tibias	91
	Montaje de las Tibias + Patas + Tórax	97
	Recuento de piezas de las 6 Tibias	102
<b>5</b>	<b>Diseño electrónico</b>	<b>103</b>
5.1	PIC16f88	103
5.2	Placa “Patas” grupo A	104
5.2.1	Características	104
5.2.2	Lista de componentes	105
5.2.3	Serigrafías TOP y BOTTOM	106
5.2.4	Operación	107
5.3	Placa “Patas” grupo B	110
5.3.1	Características	110

5.4	Placa Entrenadora	112
5.4.1	Características	112
5.4.2	Lista de componentes	113
5.4.3	Serigrafías TOP y BOTTOM	113
5.4.4	Operación	115
5.5	Otros elementos de hardware	117
5.5.1	Programadora PIPO2	117
5.5.2	Otros módulos ayudantes	118
5.5.3	Diseño previo de la placa de alimentación	118
<b>6</b>	<b>Programación</b>	<b>119</b>
6.1	Herramientas de programación	119
6.2	Programas	122
6.2.1	Programa “pata_iglu_R3.asm”	122
6.2.2	Programa “master_teclado.asm”	125
<b>7</b>	<b>Resultado</b>	<b>127</b>
<b>8</b>	<b>Conclusión</b>	<b>128</b>
<b>9</b>	<b>Bibliografía</b>	<b>130</b>
9.1	Proyectos de Final de Carrera EUPMT	130
9.2	Libros	130
9.3	Data Sheets y Notas de Aplicación	131
9.4	Artículos y otras publicaciones	132
9.5	Páginas web	132

# Índice de Figuras

Figura F. 1: Robots de Carlos Vivancos y Marc Rivas.	4
Figura F. 2: IC-Hexapod	18
Figura F. 3: BF Hexapod V5 (inacabado)	5
Figura F. 4: CH3-R	5
Figura F. 5: Asterisk	6
Figura F. 6: AMOS-WD06	6
Figura F. 7: Mike Smyth's Hexapod	7
Figura F. 8: Bill-Ant	7
Figura F. 9: Ejemplar de Lauron III del IRI	21
Figura F. 10: Lauron IVc	8
Figura F. 11: Melanie III	8
Figura F. 12: Diagrama de bloques general	14
Figura F. 13: Diagrama de bloques de la comunicación I <sup>2</sup> C	15
Figura F. 15: Segundo prototipo de pata	19
Figura F. 14 Primer prototipo de pata	19
Figura F. 16: Detector de obstáculo abierto	33
Figura F. 17: Detector de obstáculo cerrado	20
Figura F. 18: Tercer prototipo de pata	20
Figura F. 19: Aspecto final del tercer prototipo	21
Figura F. 20: Primera etapa del diseño final de la pata	21
Figura F. 22: Modelo de cadera fija	22
Figura F. 21: Modelo de cadera móvil	22
Figura F. 23: Diseño final de las patas	23
Figura F. 25: Características de la señal PWM	24
Figura F. 24: Control de posición del servo	24
Figura F. 27: Imagen panorámica de la señal PWM obtenida	38
Figura F. 28: Imagen 1 de detalle de la señal PWM obtenida	38
Figura F. 29: Imagen 2 de detalle de la señal PWM obtenida	25
Figura F. 26: Primera controladora	25
Figura F. 30: Placa entrenadora	26
Figura F. 31: Collage con distintas vistas del diseño y el resultado finales	28

Figura F. 32: Patillaje del PIC16F88	103
Figura F. 33: Vista exterior de dos placas simétricas	118
Figura F. 34: Vista interior de dos placas simétricas	105
Figura F. 35: Serigrafía Top a escala real de la Placa “Patas” grupo A	119
Figura F. 36: Serigrafía Bottom a escala real de la Placa “Patas” grupo A	119
Figura F. 37: Serigrafía SST a escala real de la Placa “Patas” grupo A	106
Figura F. 38: Elementos 1 de la placa “Patas”	107
Figura F. 39: Elementos 2 de la placa “Patas”	107
Figura F. 40: Diagrama de pines del Conector Principal del grupo A	108
Figura F. 42 : Extracción del conector	122
Figura F. 43: Conexión de la programadora.	109
Figura F. 41: Acceso a los Conectores Principales	109
Figura F. 44: Serigrafía Top a escala real de la Placa “Patas” grupo B	123
Figura F. 45: Serigrafía Bottom a escala real de la Placa “Patas” grupo B	123
Figura F. 46: Serigrafía SST a escala real de la Placa “Patas” grupo B	110
Figura F. 47: Diagrama de pines del Conector Principal del grupo A	111
Figura F. 48: LEDs de estado	111
Figura F. 49: Serigrafía Top de la placa Master	114
Figura F. 50: Serigrafía Bottom de la placa Master	114
Figura F. 51: Elementos de la placa Entrenadora	115
Figura F. 52: Mando y teclado	116
Figura F. 53: Programadora PIPO2	117
Figura F. 54: Interfaz del editor EditPlus2	119
Figura F. 55: Interfaz del compilador MPASM	120
Figura F. 56: Resultado de la compilación	120
Figura F. 57: Interfaz de WinPic800	121
Figura F. 58: Configuración de WinPic800 para programar con PIPO2	121
Figura F. 59: Aspecto final de MIHRO	127

# Índice de Tablas

Tabla T. 1: Etapas de desarrollo del robot MIHRO (1)	10
Tabla T. 2: Etapas de desarrollo del robot MIHRO (2)	11
Tabla T. 3: Propiedades mecánicas del robot MIHRO	27
Tabla T. 4: Consumo eléctrico del robot	29
Tabla T. 5: Órdenes manuales mediante teclado o mando	116

[Página en blanco]

## Índice

---

[Página en blanco]

# 1 Prefacio

Mediante el presente documento tengo el enorme placer de presentarles a MIHRO, un robot hexápedo de 18 grados de libertad que supone la consecuencia de cientos de horas de trabajo, diseño y también, debo reconocerlo, de mi propia ensoñación. Quienes me conocen les dirán de mí que tengo cierta tendencia a la megalomanía, aunque quizás esto sólo sea el reflejo de mi propio afán de superación. No obstante, cuando empecé esta carrera, y aunque sentía cierta curiosidad por la robótica, en mi foro interno jamás pude verme a mí mismo dedicado en un futuro no muy lejano a diseñar y construir robots, una tarea que por aquél entonces me parecía harto compleja, hasta el punto de considerarla inaccesible. Un año como miembro del CRA (Club de Robótica Autónoma) de la EUPMT, y dos años más como presidente electo cambiaron en cierto modo esa visión temerosa de la robótica y la sustituyeron gradualmente por otra mucho más cercana, aunque respetuosa.

El CRA me brindó la posibilidad de acceder a técnicas y recursos básicos pero eficaces que me permitieron comenzar a desarrollar mis primeros prototipos, la mayoría de ellos robots relativamente sencillos pensados para competir en concursos de robótica como el que el mismo CRA organizaba anualmente, evento en el cual tuve oportunidad de participar en más de una ocasión, e incluso de colaborar en su organización. Y durante todo ese tiempo se fue fraguando en mi cabeza la posibilidad de enfrentarse a un reto tan enorme como construir y gobernar un robot hexápedo completo, un tipo de robot con un sistema de locomoción tan complejo por la gran cantidad de dispositivos que integra, que a menudo se ha descartado como opción de locomoción en cientos de proyectos similares.

Es más, durante casi tres años la seductora idea de realizar este trabajo como proyecto de final de carrera mantuvo mi inquietud ocupada indagando en todo lo referente a este tipo de robots, e incluso tuve oportunidad de realizar mis primeros diseños y experimentos en la construcción y el control de dos maquetas en forma de pata robótica articulada. Al ver que mis primeros experimentos concluían de forma satisfactoria mi concepción del robot en su estado final fue creciendo hasta convertirse en un proyecto tan extenso y ambicioso que de ningún modo podía abarcarse en el tiempo correspondiente a la asignatura de Proyecto de Final de Carrera. No obstante, esta circunstancia encajaba bastante bien con mis planes futuros de perpetuar mis estudios en un ciclo superior de Electrónica, pues la idea de realizar una parte del proyecto ahora y el resto como proyecto de final de carrera de este segundo ciclo me permitió finalmente englobar todos mis objetivos.

Es por eso que esta memoria es el testigo de la primera etapa de un trabajo que no terminará, al menos, hasta dentro de dos años; una primera etapa que además pone de manifiesto la complejidad intrínseca a estos robots, la cual, además las funciones añadidas para las que se ha diseñado el robot MIHRO en concreto, hacen que este proyecto no sólo pretenda ser el encumbramiento más célebre que puedo darle a los conocimientos que he adquirido durante la carrera, sino también la victoria sobre un profundo reto personal.

Que lo disfruten.

## 2 Introducción al Proyecto

### 2.1 Motivación

Tal y como deja entrever el prefacio, mi pasión por la robótica, infundida en gran medida por mi contacto con el Club de Robótica Autónoma durante los años que llevo en la EUPMT, ha sido la principal motivación que me ha llevado a embarcarme en este proyecto. No obstante, hay que matizar que el gusto por la robótica no es más que una motivación precursora de mi afán de aprendizaje en esta área para la posterior aplicación de los conocimientos adquiridos en mi futuro profesional. Así pues, el objeto final de este proyecto no es la diversión, la autocomplacencia o el cumplimiento de los créditos requeridos para terminar la carrera, sino el mayor aprendizaje posible para alcanzar un buen futuro profesional en este campo.

De hecho, esta primera premisa se hace patente a lo largo de todo el proyecto, pues no se ha llevado a cabo prácticamente ningún procedimiento que intente resolver de forma simplificada un proceso cuya naturaleza es más compleja. Así pues, para empezar, el lenguaje de programación escogido ha sido el ensamblador, mucho más críptico y complejo que los lenguajes de alto nivel, como C o Basic, con que se pueden programar también los microcontroladores escogidos, pero que a su vez aporta un mayor entendimiento del funcionamiento interno real de la máquina. Por otro lado, tampoco se han usado librerías pre-programadas, como las que habitualmente proporcionan los fabricantes para simplificar el desarrollo de aplicaciones, o máquinas diseñadas para resolver automáticamente procesos complejos. De este modo, no se usan librerías para gobernar la comunicación I<sup>2</sup>C, sino que se resuelve enteramente por software, y no se usan generadores automáticos de señales PWM o microcontroladores con soporte hardware para controlar un alto número de servomotores. Tampoco se pretende hallar algoritmos lineales simples para resolver procesos que deberían tomar soluciones dinámicas más complejas, como es el caso de los motores, que pueden controlarse de forma dinámica con una precisión de 0'7°, en vez de basar su control en la repetición secuencial de unas pocas posiciones pre-programadas.

De la misma manera, y quizás en este caso se hace más patente la filosofía general que rige el proyecto, no se han usado piezas prefabricadas específicamente diseñadas para construir el esqueleto de robots hexápodos, como las que distribuye la empresa *Lynxmotion*, sino que se ha diseñado y fabricado un esqueleto mecánico a partir de chapa de aluminio sin mecanizar. Es más, el diseño mecánico, aunque estéticamente inspirado en algunos de los robots más conocidos dentro del mundo de los hexápodos, es completamente inédito, y si se analiza su estructura con atención se puede comprobar que no sólo muestra diferencias fundamentales con estos famosos robots, sino que aporta nuevas estructuras mecánicas, como las tibias, que no se asemejan, por lo menos, a ningún otro de los más de 60 proyectos de este tipo que he tenido oportunidad de analizar.

## 2.2 Antecedentes

Entre los proyectos que de algún modo han inspirado el diseño tanto estético como funcional de MIHRO destacan:

### 2.2.1 Proyectos EUPMT

- ***Robot Insecte Intel·ligent*, Carlos Vivancos Gómez, año 2000.**

Este es un conocido robot de la EUPMT, pues está desde hace años expuesto en la vitrina de la entrada. Usa patas de dos grados de libertad, uno para el movimiento horizontal de la pata y otro para el movimiento vertical de una estructura de aluminio parecida a un pistón. Utiliza cuatro microcontroladores de forma jerárquica: un “master” o “cerebro” gobierna otros tres integrados “esclavos” que a su vez gobiernan cuatro servomotores cada uno. Planteó una comunicación I<sup>2</sup>C que no logró implementar, por lo que diseñó un bus propio.

- ***Disseny i creació d'un microbot insecte autònom*, Marc Rivas Capafons, año 2005.**

Este robot, de nombre Biljana, se diseñó como continuación del proyecto de Carlos Vivancos, aportando varias mejoras al proyecto inicial: incluía sensores de contacto en las patas que le permitían detectar el suelo, aunque finalmente no implementó un sistema de locomoción adaptativo, e incluía también sensores de contacto frontales a modo de antenas. Aunque el proyecto incluye varias propuestas de estructura mecánica basada en patas de tres grados de libertad, finalmente no logra implementar satisfactoriamente ninguna de ellas y recurre a la estructura mecánica de dos grados de libertad basada en pistones del robot de Carlos Vivancos. Cada una de las seis parejas de motores es controlada por un microcontrolador “esclavo”, que a su vez es controlado por un séptimo microcontrolador “master” mediante una comunicación I<sup>2</sup>C.

- ***Diseño y Construcción de un Hexápodo Autónomo*, Jordi Estrella Domene, año 2007**

Curioso proyecto de un robot hexápedo de sólo tres grados de libertad totales, que basa su movimiento en un sistema basculante de las “patas” centrales del robot.

- ***Robot hexàpode “Mordor”*, Keila Belando Baca, año 2005**

Proyecto de robot hexápedo basado también en una estructura de sólo tres grados de libertad totales. Incorpora un sistema de visión artificial basado en infrarrojos. El control se realiza de forma centralizada desde un controlador BasicStamp2, por lo que está programado en un lenguaje parecido al Basic.

- ***Construcció d'un Robot Quadrúped, Isaac Hernàndez Izquierdo, año 2002***

Robot cuadrúpedo de tres grados de libertad por pata, sensores de contacto y sistema de detección de obstáculos por infrarrojos. Utiliza potenciómetros para realimentar la posición de los servomotores, que se gobiernan desde dos microcontroladores comunicados por un bus propio.

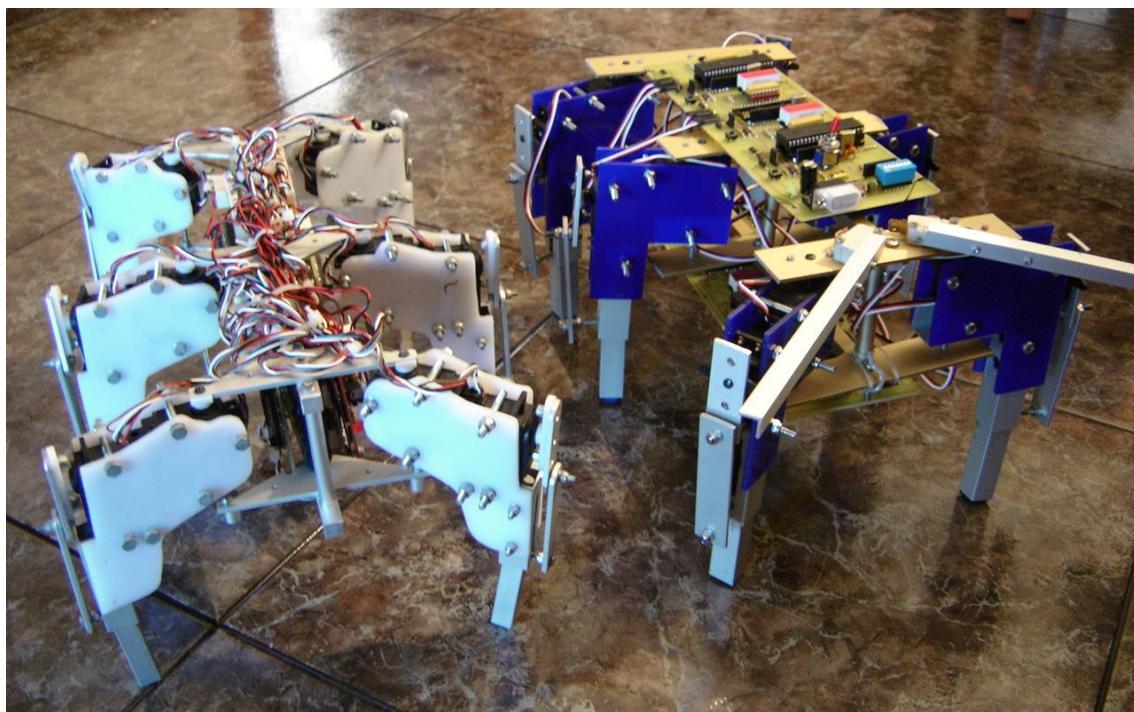


Figura F. 1: Robots de Carlos Vivancos y Marc Rivas.  
(Fuente: Galería de imágenes del CRA)

### 2.2.2 Otros proyectos

Cuando se habla de robots hexápodos es frecuente que salgan a relucir los nombres de ciertos robots que, por su trascendencia en los albores del desarrollo de esta tecnología, se han ganado una merecida fama. Es el caso de los conocidos Ghengis, Attila o Hannibal, los robots hexápodos del MIT, pero también de otros robots más modernos, aunque igualmente trascendentales, como los Tarry y Tarry II, desarrollados en la Universidad de Duisburg, o el Hamlet, de la Universidad de Canterbury, entre muchos otros. De algún modo todos estos proyectos han inspirado en parte el diseño de MIHRO, sin embargo sus referentes más directos están en robots mucho más modernos y a menudo desarrollados por responsables de instituciones docentes o por aficionados. Estos últimos son los de mayor repercusión para este proyecto, pues a menudo permiten analizar las estrategias que otros han adoptado para solucionar los mismos problemas con que se ha topado el desarrollo de MIHRO. Entre ellos destacan:

- **Micromagic Systems**

Esta empresa se dedica desde hace algunos años al desarrollo de animatrónics para la industria del cine. Matt Denton, su principal responsable, ha desarrollado una serie de robots hexápodos cuyo sistema locomotor roza la perfección. Aunque la mayoría de sus creaciones son mucho más simples que otros robots de esta categoría, su éxito reside en la fluidez con la que se mueven sus patas, producto tanto de un diseño mecánico acertado como de un control electrónico eficiente e inteligente. Sus robots se mueven de un modo tan realista que crean la ilusión de ser artrópodos vivos, y no máquinas inanimadas. Además, los rasgos mecánicos y la estética que identifican a sus robots han sido copiados en el diseño de tantos otros hexápodos que actualmente casi resulta difícil encontrar proyectos nuevos de este tipo que no recuerden a los robots de Matt Denton. En cuanto a MIHRO, en ciertos aspectos se aprecia una similitud estética con los robots de Micromagic Systems, pero es sólo en apariencia, pues su estructura mecánica sigue un patrón muy diferente, e incluso va más allá en algunos aspectos, como en la sensórica de las patas.

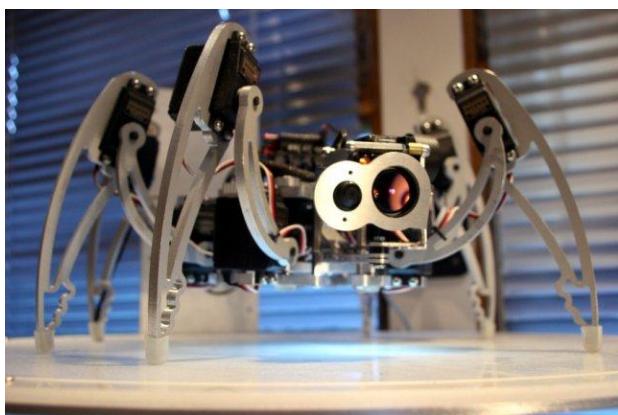


Figura F. 2: IC-Hexapod  
(Fuente: [www.micromagicsystems.com](http://www.micromagicsystems.com))



Figura F. 3: BF Hexapod V5 (inacabado)  
(Fuente: [www.micromagicsystems.com](http://www.micromagicsystems.com))



Figura F. 4: CH3-R  
(Fuente: [www.lynxmotion.com](http://www.lynxmotion.com))

- **Lynxmotion**

Esta empresa desarrolla, entre muchos otros tipos de robots, varios modelos de robots hexápodos ensamblados o por piezas. De entre estos, merecen ser mencionados los hexápodos de tipo “estrella”, un tipo de hexápodos que ofrece una distribución en círculo de sus patas, en vez de disponerlas en paralelo a lado y lado de su cuerpo, y que por ende se controlan de forma vectorial, avanzando en cualquier dirección sin necesidad de rotar el cuerpo. Su producto estrella, no obstante, es un hexápodo normal, con distribución de las patas en paralelo, llamado “Phoenix”, cuyo diseño mecánico no trata de ocultar su evidente similitud con los robots de Matt Denton.

- **Asterisk**



Figura F. 5: Asterisk  
(Fuente: [www-arailab.sys.es.osaka-u.ac.jp](http://www-arailab.sys.es.osaka-u.ac.jp))

Este es un asombroso robot de tipo “estrella” desarrollado en el Arai Lab, de la Universidad de Osaka. Fue presentado al mundo en la World Expo 2005 de Aichi, aunque tuvo su primera gran actuación en la Osaka Robot Fair de 2006, en la que se le pudo ver haciendo cosas tan insólitas como caminar boca abajo colgado de un soporte en forma de reja, recoger y transportar objetos con sus patas, escalar obstáculos altos o atravesar huecos en forma de arco agachándose y arrastrándose para pasar bajo ellos. Constructivamente, su característica

más destacable es que sus patas ofrecen cuatro grados de libertad, y que tienen un sensor de contacto que les permite adaptarse dinámicamente al terreno.

- **Projekt Marvin**

Este es un fabuloso proyecto completamente desarrollado por un solo aficionado a la robótica. Las características de este robot son tantas que cuesta enumerarlas todas, pero entre ellas destacan: cuerpo con seis patas de tres grados de libertad cada una, más una cabeza con dos grados de libertad más, en total veinte servos. Visión estereoscópica, medición de áreas de distancia 2D mediante láser y ultrasonido, control distribuido entre catorce procesadores Atmel, control del movimiento mediante cinemática inversa, reproductor WAV, sistema de equilibrio basado en giroscopios, control adaptativo al terreno, GPS, sistema de gestión de la energía, uso de una tarjeta MicroSD como memoria de masa, ... y muchas otras. El valor del contenido de este proyecto es incalculable, pero desgraciadamente su creador sólo trabaja “por amor al arte”, por lo que no publica documentación técnica, y su página web sólo está disponible en alemán.

- **AMOS-WD06**

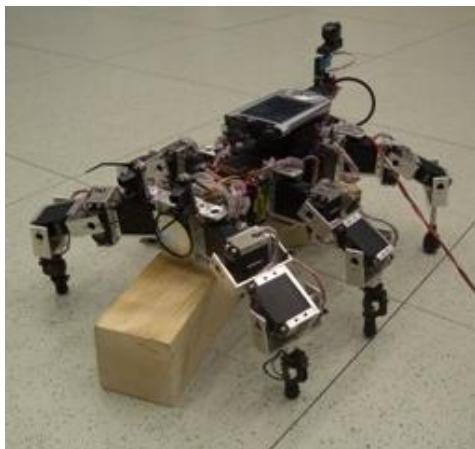


Figura F. 6: AMOS-WD06  
(Fuente: [www.chaos.gwdg.de](http://www.chaos.gwdg.de))

Proyecto realizado por el Max-Planck-Institute for Dynamics and Self-Organization de Göttingen. Este proyecto pretende dotar de capacidad reactiva a un robot hexápodo programado en forma de red neural. El robot, de tres grados de libertad por pata, posee un sensor infrarrojo en cada una de sus extremidades, que advierte al procesador central, un mini-ordenador integrado en una PDA, de la cercanía de un cuerpo desde cualquier dirección. El robot, como lo haría un animal, huye y se esconde cuando detecta una presencia extraña a su alrededor que identifica como hostil.

- **Mike Smyth's Hexapod**

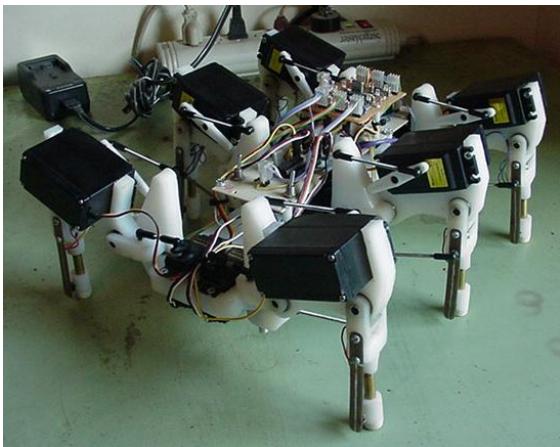


Figura F. 7: Mike Smyth's Hexapod  
(Fuente: [home.ctlnet.com/~robotguy67/hexapod](http://home.ctlnet.com/~robotguy67/hexapod))

Sencillo hexápedo de doce grados de libertad que, además de adaptarse al terreno con sensores de contacto en el piso de sus patas, incorpora la interesante característica de poder detectar los obstáculos que interrumpen la trayectoria de cada paso, para poder superarlos corrigiendo la altura de los mismos. Para lograrlo, utiliza unos sensores en forma de lengüeta que presionan un interruptor al chocar la pata contra un obstáculo; un sistema que, aunque sencillo, le confiere al robot unas posibilidades de adaptación al terreno

excelentes. El robot MIHRO se diseñó tomando esta habilidad como requisito indispensable, pero se descartó la solución mecánica que usa Mike Smyth en este robot, pues resulta frágil y permite detectar sólo obstáculos que interfieren con el extremo inferior de sus patas. MIHRO, por el contrario, utiliza un sistema mecánico mucho más complejo y robusto, que además le permite detectar obstáculos en más del 70% de la superficie de sus tibias.

- **Bill-ant**

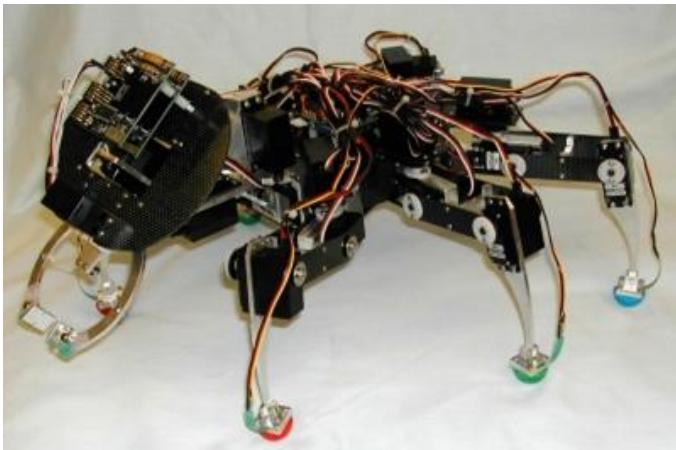


Figura F. 8: Bill-Ant  
(Fuente: [biorobots.cwru.edu/projects/billant](http://biorobots.cwru.edu/projects/billant))

Robot desarrollado por William Lewinger, del Center for Biologically Inspired Robotics Research de la Universidad Case Western Reserve. Este hexápedo incorpora un total de 22 grados de libertad reales (William Lewinger considera otros seis grados pasivos en los detectores del piso de sus patas), distribuidos en tres grados de libertad por pata, tres grados de libertad más en el cuello de su cabeza y otro más para

articular una mandíbula mecánica. El robot, claramente inspirado en la anatomía de las hormigas, lleva incorporados sensores de fuerza en el extremo de sus patas y mandíbulas que le permiten cuantificar el peso que carga cada pata o la fuerza que deben realizar las mandíbulas para sujetar objetos. Además, el robot, de casi 3 Kg de peso, puede andar mientras lleva una carga de casi otros 3 Kg más.

- **Lauron**

Este es uno de los robots hexápodos más impresionantes que pueden verse hoy día. Su fabricación viene de la mano de la alemana FZI (Forschungszentrum Informatik), y los pocos ejemplares que se producen se venden mayoritariamente a instituciones relacionadas con la investigación y el desarrollo de aplicaciones en el campo de la robótica. Se trata de un enorme robot de aluminio mecanizado de gran calidad, con tres grados de libertad por pata y dos grados más en una cabeza articulada. Dispone de características muy avanzadas en su sistema locomotor, diseñadas para potenciar su capacidad de adaptación al terreno, de navegación y de orientación. Entre estas características destacan la detección en los tres ejes de la presión que cada pata ejerce contra el suelo, la capacidad de visión artificial omnidireccional, y un avanzado sistema de visión estereoscópica que le permite orientarse en mapas tridimensionales que crea mientras recorre su entorno. El IRI, Institut de Robótica i Informática Industrial de la Universitat Politècnica de Catalunya, dispone de un ejemplar de la tercera versión de este robot (la última versión fabricada es la 4c), con el que desarrollan investigaciones sobre locomoción, orientación y visión artificial.

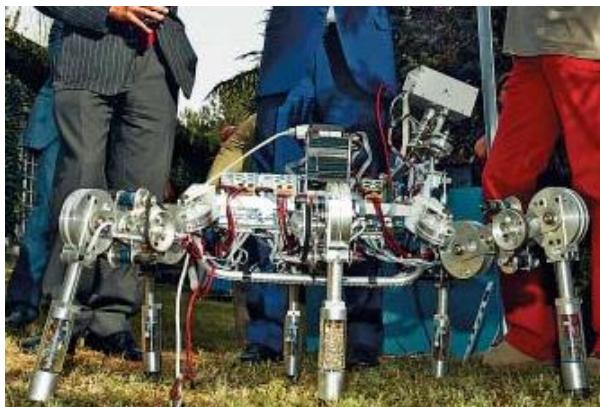


Figura F. 9: Ejemplar de Lauron III del IRI  
(Fuente: [www-iri.upc.es](http://www-iri.upc.es))



Figura F. 10: Lauron IVc  
(Fuente: [www.mca2.org](http://www.mca2.org))

- **Melanie III**



Figura F. 11 Melanie III  
(Fuente: [mundobot.com/projects/melanie/spmelanie](http://mundobot.com/projects/melanie/spmelanie))

Esta es la última versión del robot Melanie, desarrollado por el ingeniero español Alejandro Alonso Puig. Dispone de tres grados de libertad por pata y de un sistema de más de treinta sensores que le permiten adaptarse al terreno con suficiente fluidez como para que el robot pueda andar por suelo abrupto a una velocidad lineal de desplazamiento constante. A parte de esto, su característica más remarcable es el diseño de sus patas, que le permiten andar cargando más de 4 Kg de peso con un consumo energético mínimo.

## 2.3 Proyecto MIHRO

El proyecto MIHRO pretende construir un robot hexápedo de tres grados de libertad por pata, más tres grados de libertad en cuello y cabeza, 21 motores en total. Además, el robot debe disponer de la capacidad de adaptarse al terreno dinámicamente, por lo que debe incorporar sensores de contacto en las patas que permitan al robot detectar tanto el suelo como los posibles obstáculos que puedan interrumpir la trayectoria de las patas cuando éstas discurren por terreno irregular. En cuanto a sus características motrices, el robot ha de tener la capacidad de superar obstáculos altos, como escalones o rocas, de andar a buena velocidad, de mantener el equilibrio y la horizontalidad usando un inclinómetro o giroscopio, de recoger, cargar y transportar objetos ligeros con sus patas delanteras, y de moverse con suavidad, precisión y verosimilitud biológica.

En cuanto a sus características funcionales, el robot debe incorporar un sistema de medición de distancias básico, basado en un transceptor de ultrasonidos, y otro basado en luz láser que le permita rastrear espacios tridimensionales, enviarlos a un ordenador y recrearlos en un entorno virtual. Este sistema de medición de distancias se basa en la interacción entre un puntero de luz láser difractado en una lente y una cámara digital (sensor CCD, CMOS u otros) controlada desde un software básico de reconocimiento de imagen, de forma que, mientras el robot no esté llevando a cabo la tarea de rastreo de entornos, la cámara deberá proporcionar vídeo en tiempo real que se transmitirá a un ordenador o a un dispositivo portátil de fabricación propia, y que podrá reproducir la imagen en una pantalla de tipo GLCD o en un sistema de visualización de realidad virtual. Además, el robot deberá incorporar un sistema de comunicación inalámbrico con un PC, localizador GPS, conectividad USB de tipo “master”(que puede gobernar otros dispositivos USB), almacenamiento de contenido multimedia en una tarjeta de memoria SD, y un módulo de audio que, además de poder triangular la procedencia de un estímulo acústico, deberá ser un grabador y reproductor de sonido que permita recoger datos sonoros del entorno o reproducir los sonidos que envíe su controlador.

Además el robot deberá ser capaz de gestionar y rentabilizar su propia energía, consiguiendo minimizar su consumo de forma dinámica. El módulo de alimentación del robot debe permitirle, en un entorno doméstico, cargar su batería desde una toma de corriente adaptada y situada en una posición conocida por el robot, y en un entorno al aire libre, cargarla usando una célula fotovoltaica incorporada, y encontrando mediante sensores de intensidad luminosa la mejor posición de carga posible.

Por último, en la etapa final del desarrollo del robot, éste deberá servir como plataforma de desarrollo en el estudio y aplicación de sistemas emuladores de conducta basados en redes neuronales. No obstante, hasta ese momento el robot ha de pasar por múltiples etapas de desarrollo y su estructura tanto electrónica como mecánica probablemente será modificada en más de una ocasión. El proyecto actual recoge las tres primeras etapas de dicho desarrollo, las que se refieren al nivel más bajo de construcción y control del sistema locomotor del robot.

La siguiente tabla revela una previsión de las principales etapas por las que habrá de transcurrir el desarrollo de MIHRO.

<b>Etapa</b>	<b>Contenido</b>
1- Primeros experimentos	<ul style="list-style-type: none"> <li>• Diseño CAD de los cuatro primeros prototipos de pata</li> <li>• Fabricación y montaje de dos de los cuatro prototipos</li> <li>• Montaje de una maqueta de pruebas para el último de los prototipos</li> <li>• Diseño de una placa básica de control para la maqueta</li> <li>• Diseño de una placa avanzada de control para la maqueta</li> <li>• Experimentos varios sobre el control dinámico de la maqueta</li> </ul>
2- Diseño y construcción	<ul style="list-style-type: none"> <li>• Diseño CAD de la estructura mecánica definitiva del robot</li> <li>• Fabricación de las 48 piezas del esqueleto</li> <li>• Montaje del esqueleto, la motorización y los elementos sensoriales del cuerpo mecánico</li> <li>• Diseño electrónico de las placas controladoras de cada pata.</li> <li>• Pruebas de locomoción mediante la programación de cada placa</li> </ul>
3- Nivel físico del sistema locomotor	<ul style="list-style-type: none"> <li>• Programación de la coordinación del movimiento de las patas desde una placa “cerebro”</li> <li>• Implementación de la comunicación interna mediante un bus I<sup>2</sup>C</li> <li>• Control manual del robot desde un mando/teclado</li> </ul>
4- Gestión de la energía	<ul style="list-style-type: none"> <li>• Instalación de los servomotores definitivos</li> <li>• Modificación del software de control para los servos definitivos</li> <li>• Diseño y construcción de la placa de alimentación, carga y gestión de la energía</li> <li>• Incorporación de dos baterías que proporcionen al robot autonomía energética</li> <li>• Aplicación de un sistema de cinemática inversa como estrategia para el control del movimiento</li> </ul>
5- Nivel superior del sistema locomotor	<ul style="list-style-type: none"> <li>• Diseño de un sistema de equilibrio basado en un giroscopio o inclinómetro</li> <li>• Programación del control adaptativo al terreno</li> <li>• Programación de las funciones locomotrices avanzadas</li> </ul>
6- Sistema de navegación	<ul style="list-style-type: none"> <li>• Construcción de la cabeza</li> <li>• Inclusión de la sensórica de navegación</li> <li>• Desarrollo de los sistemas medidores de distancia</li> <li>• Desarrollo de la estrategia de navegación autónoma</li> <li>• Desarrollo del sistema de tratamiento de imagen y visión artificial</li> </ul>
7- Conectividad	<ul style="list-style-type: none"> <li>• Desarrollo de la conectividad master USB</li> </ul>
8- Comunicación	<ul style="list-style-type: none"> <li>• Desarrollo de un canal de comunicación inalámbrico para el control del robot</li> <li>• Desarrollo de un canal de comunicación inalámbrico para el envío / recepción de contenido multimedia</li> </ul>

**Tabla T. 1: Etapas de desarrollo del robot MIHRO (1)**  
(Fuente: Proyecto MIHRO)

<b>Etapa</b>	<b>Contenido</b>
9- Sistema de visión artificial	<ul style="list-style-type: none"> <li>• Desarrollo del sistema de captación, tratamiento y envío de imagen</li> <li>• Desarrollo del sistema de visualización portátil basado en un pequeño display LCD</li> <li>• Desarrollo de un sistema de captación del movimiento de la cabeza humana que interactúe con el cuello del robot</li> <li>• Integración del sistema de visión artificial y del sistema de captación del movimiento da la cabeza en un sistema de visión remota basado en unas gafas de realidad virtual</li> </ul>
10- Módulo de audio	<ul style="list-style-type: none"> <li>• Desarrollo de un sistema de captación, digitalización y compresión de audio procedente de un micrófono</li> <li>• Desarrollo de un sistema de triangulación de la procedencia espacial de un estímulo acústico mediante varios micrófonos</li> <li>• Implementación de un reproductor de audio normal y comprimido.</li> </ul>
11- Memoria de masa	<ul style="list-style-type: none"> <li>• Implementación de la comunicación con un dispositivo de almacenamiento masivo de memoria, como una memoria flaix USB o una tarjeta SD</li> <li>• Integración de dicha memoria como sistema de almacenamiento de los contenidos multimedia del robot</li> </ul>
12- Célula fotovoltaica	<ul style="list-style-type: none"> <li>• Modificación de la placa de alimentación para poder usar una placa fotovoltaica como fuente de energía</li> <li>• Desarrollo de un sistema de búsqueda de las condiciones óptimas de carga solar</li> </ul>
13- Red neuronal	<ul style="list-style-type: none"> <li>• Desarrollo sistema cognitivo basado en el arquetipo de las redes neuronales</li> <li>• Adaptación del sistema para nutrirse de todos los recursos desarrollados anteriormente</li> </ul>

**Tabla T. 2 Etapas de desarrollo del robot MIHRO (2)**  
(Fuente: Proyecto MIHRO)

La tabla deja entrever, por un lado, la complejidad final para la que se diseña el robot desde un principio, y por otro lado, las posibilidades futuras de desarrollo tan enormes que ofrece este proyecto, lo que lo convierte en una plataforma excelente para el aprendizaje de las técnicas y los recursos relacionados con todas sus características.

Cabe decir también que la inclusión de todos estos sistemas en el robot no sólo es fruto de la inquietud o la curiosidad por sus respectivos campos de aplicación, sino que responden conjuntamente al objetivo final del robot, que no es otro que el de ser un robot explorador. Así pues, las características para las que se diseña no están en absoluto carentes de un objetivo común, sino que se enfocan a perfeccionar el modo en que este robot discurre de forma autónoma o teleoperada por un territorio desconocido, inaccesible u hostil, potenciando su capacidad de adaptación a un terreno irregular e imprevisible y la accesibilidad de su interfaz con un usuario controlador humano (de ahí que una de las etapas de desarrollo del robot incluya la implementación de sistemas de realidad virtual que, aparentemente, pudieran estar fuera de lugar en un proyecto completamente centrado en el campo de la robótica).

Además, esta capacidad exploradora de MIHRO no sólo tiene fines académicos, sino que también tiene su aplicabilidad en el mercado y la sociedad actuales, pues cada día se invierte mayor capital en el desarrollo de aplicaciones robóticas para tareas de salvamento, exploración espacial, y manejo de sustancias peligrosas. Así pues, finalmente MIHRO puede ser adaptado para su uso como robot de rescate, entrando en zonas de difícil acceso como edificios derrumbados, incendios, cuevas,... etc. También puede recibir uso como herramienta de maniobra en ambientes químicos o radiológicos hostiles, como áreas selladas por fugas de gas o de material radiactivo, o como herramienta de detección de sustancias y aparatos peligrosos, como minas anti-persona. Del mismo modo, las capacidades de captación de entornos y desenvoltura por terrenos irregulares permitirían a MIHRO realizar tareas de exploración espacial o trabajos de campo en exploración militar. Incluso podría recibir uso como avanzadilla en misiones antiterroristas. Por último, el robot también podría recibir uso en espeleología, minería o arqueología, mediante la exploración de cámaras y recovecos concurridos en cuevas, minas y yacimientos arqueológicos, como antiguas tumbas faraónicas y catacumbas. Como curiosidad, merece la pena mencionar que ciertas cámaras de gran relevancia arqueológica del famoso monumento funerario egipcio del Valle de los Reyes no fueron descubiertas hasta que se fabricaron robots exploradores específicamente diseñados para recorrer sus estrechas cavidades.

Por último, hay que decir que la elección de “MIHRO” como nombre del robot no sólo se fundamenta en su significado como siglas de “Mobile Intelligent Hexapod Robot”, sino en la propiedad de este nombre de aludir a la capacidad de ver y a la acción de mirar, que inexorablemente relaciona al robot con su función de explorador.

## 2.4 Contenido del proyecto actual

Como se ha mencionado antes, el trabajo realizado para este Proyecto de Final de Carrera corresponde a las etapas 1, 2 y 3 de entre las explicadas en el apartado anterior. Los siguientes puntos introducen los contenidos que recoge el presente trabajo.

### 2.4.1 Objetivos

Por ahora, el desarrollo de este proyecto comprende:

- El estudio de las posibilidades de control, adquisición y tratamiento de datos de microcontroladores básicos, así como su programación e implementación.
- El estudio de las posibilidades de sensorización y adecuación de señales de sistemas robóticos típicos.
- Estudio y planificación de un sistema de locomoción que permita a un robot de seis patas y tres grados de libertad por pata interactuar con obstáculos u objetos de su

entorno, otorgándole capacidad para moverse, trazar una trayectoria, sobreponer obstáculos y mantener el equilibrio.

- Previsión y planificación de la integración de módulos electrónicos y funcionalidades futuras en el robot.
- Dominar técnicas básicas de diseño de sistemas mecánicos mediante programas de tipo CAD.
- Diseñar una estructura mecánica que cumpla todos los requisitos de funcionalidad, robustez y estética del robot.
- Fabricar y montar la estructura mecánica del robot a partir de chapa de aluminio sin mecanizar, completamente a mano y de forma artesanal.
- Cuantificar las características de la motorización del robot.
- Estudio del control de servomotores pequeños mediante señales PWM, y de su implementación mediante software.
- Diseño de un sistema electrónico distribuido en módulos funcionales que satisfaga las necesidades de control, coordinación, comunicación e interfaz del nivel más bajo del sistema locomotor del robot.
- El estudio de las posibilidades de intercomunicación entre los módulos que conforman la estructura electrónica del robot en base a un bus I<sup>2</sup>C.
- Programación en bajo nivel de una comunicación I<sup>2</sup>C *Slave* con soporte hardware e I<sup>2</sup>C *Master* enteramente implementado por software.
- Diseño de las placas electrónicas definitivas usando el programa OrCAD 16.0.
- Soldadura en placas de topos para prototipos de una cara y en placas de cobre con pistas fresadas de doble cara.
- El estudio de algunas posibilidades de interfaz y control sencillas basadas en un pequeño mando o teclado.
- Implementación y sincronización global de todos los sistemas y variables en el robot, con el fin de que pueda andar por control manual desde un mando o teclado.

Cabe decir también que, aunque el robot esté inspirado en la morfología de los insectos y busque cierto grado de verosimilitud biológica, este proyecto no pretende estudiar este parámetro desde el punto de vista biomecánico más allá de las funcionalidades necesarias para el robot. Tampoco se busca emular la locomoción de ninguna variedad de insecto por encima de las capacidades que sean óptimas para que el robot desarrolle las tareas para las que fue diseñado. La locomoción hexápoda de los insectos, igual que la de la mayoría de los artrópodos, les confiere una increíble capacidad de adaptación al medio y de desenvoltura por terrenos abruptos o de difícil acceso. Su efectividad inherente ha hecho posible que esta familia de seres vivos, una de las más antiguas de la historia de la evolución animal, haya sobrevivido hasta la actualidad sin sufrir grandes cambios evolutivos en su sistema motor. De este modo se justifica el intento de emular, con mayor o menor fidelidad, este sistema de locomoción para lograr una buena desenvoltura por terrenos irregulares, sin embargo este es un proyecto que estudia la mecánica como medio y la electrónica como fin, por lo que en ningún caso se usarán estas materias para dar soporte a un estudio biológico o de cualquier otro campo.

### 2.4.2 Esquema general de MIHRO

El control del sistema locomotor de MIHRO está distribuido entre siete placas electrónicas: una central que realiza la función de “cerebro” o Master del bus I<sup>2</sup>C, y seis esclavas dedicadas exclusivamente al control de cada una de las patas. Todas ellas están basadas en el microcontrolador de Microchip PIC16f88, del que se hablará más adelante. De estas placas sólo las esclavas permanecerán finalmente en el robot, pues la placa que realiza la función de Master es, por ahora, una placa entrenadora que deberá ser sustituida en el futuro por su versión completa y definitiva (basada en un PIC16f877).

Cada uno de los seis esclavos recibe los estímulos de dos sensores de contacto, uno que detecta la presión de la pata contra el suelo, y otro que detecta la presión de la pata contra un obstáculo eventual que se interponga en su trayectoria de avance horizontal. Además, cada esclavo controla los tres servomotores que constituyen las tres articulaciones de cada pata. Por último el Master, que recibe órdenes desde un pequeño teclado, gobierna a los esclavos mediante un bus I<sup>2</sup>C (en naranja), que queda abierto por el lado del Master para futuras ampliaciones.

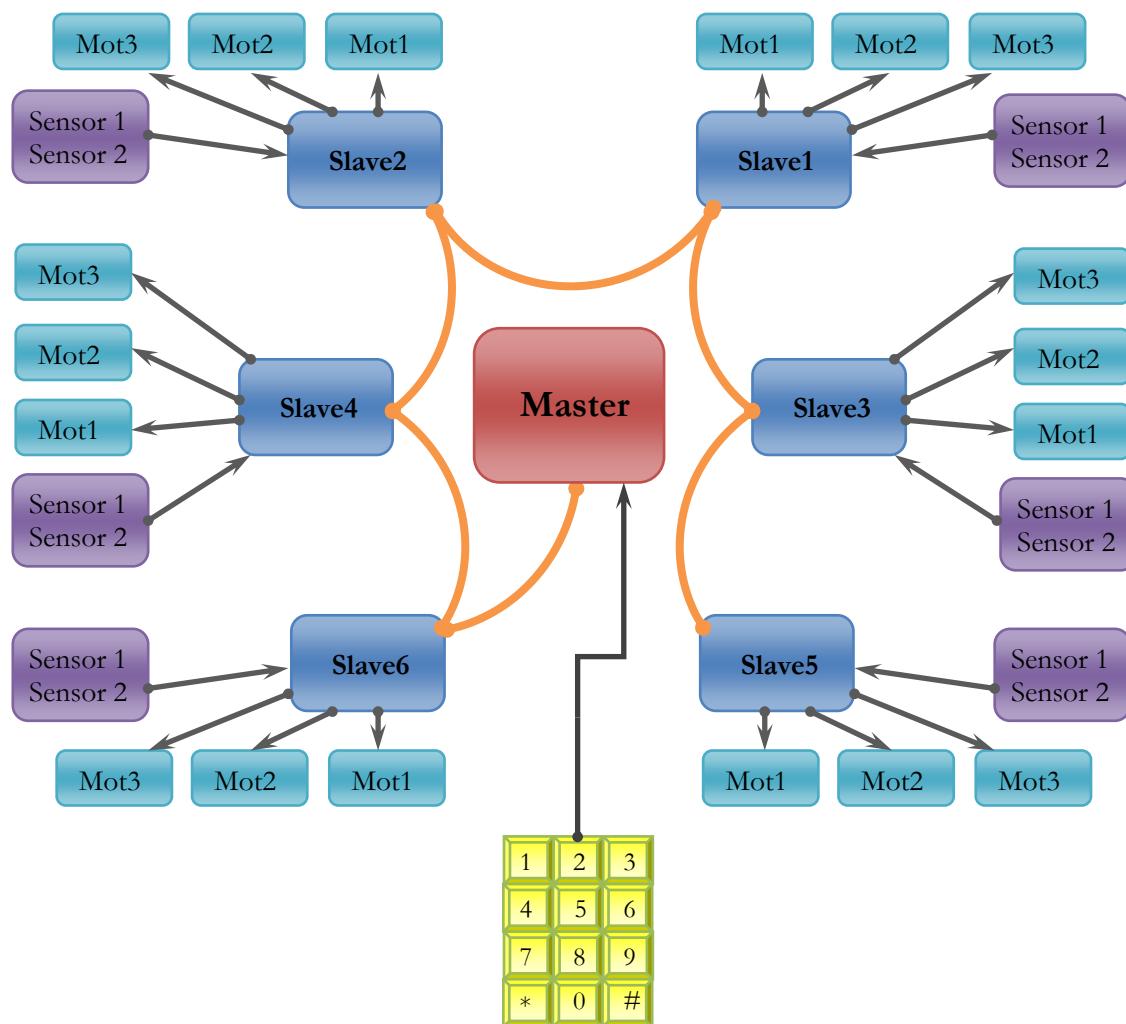


Figura F. 12: Diagrama de bloques general  
(Fuente: Proyecto MIHRO)

En el bus I<sup>2</sup>C todos los elementos se “enganchan” a unas líneas comunes referenciadas a una masa compartida. Para facilitar el conexionado físico de todos los integrantes del bus, cada controladora de las patas recibe las tres líneas de conexión (SCL, SDA y masa) del dispositivo inmediatamente anterior conectado al bus y las envía al siguiente dispositivo. De este modo, el Master sólo necesita un conector de tres pines para enviar datos a cualquiera de los eslavos conectados al bus y otro conector de tres pines para dejar el bus abierto, tal y como muestra el diagrama siguiente:

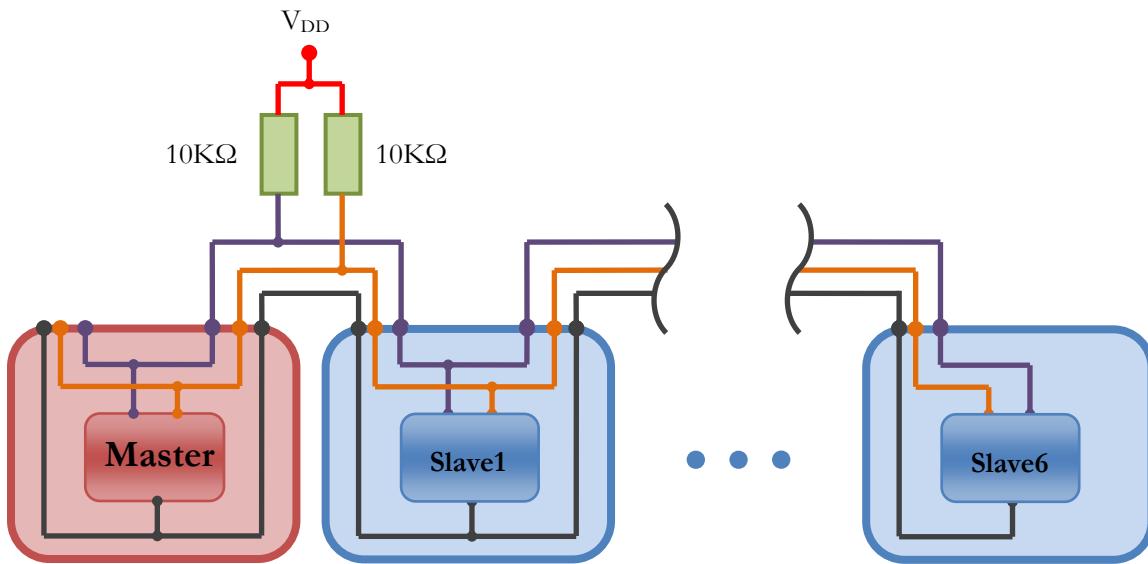


Figura F. 13: Diagrama de bloques de la comunicación I<sup>2</sup>C  
(Fuente: Proyecto MIHRO)

Las dos resistencias *pull-up* de 10KΩ están físicamente integradas en el Master, y realizan una función de AND lógico, pues mantienen el bus a nivel alto siempre que ninguno de los elementos conectados al mismo fuerce el estado del bus a nivel bajo. Esto significa que para introducir un nivel alto en el bus los elementos conectados al mismo no necesitan aplicar directamente un nivel de tensión alto, sino que simplemente se configuran en colector abierto para que la resistencia haga el resto. Si alguno de los elementos pretende poner un nivel bajo en el bus y otro de ellos un nivel alto, siempre prevalecerá el nivel bajo.

#### 2.4.3 Factores limitantes

El desarrollo de este proyecto ha sido acotado desde un principio por dos factores limitantes fundamentales: el tiempo y el presupuesto disponibles. El primero de ellos repercute directamente en la cantidad de trabajo realizable, y por tanto en las habilidades de que dispondrá el MIHRO en el momento de la defensa del proyecto; el segundo repercutre específicamente sobre el modelo de servomotor escogido y la ausencia de una batería que mantenga, aunque de forma poco optimizada, una cierta autonomía energética en el robot. En cuanto al servomotor escogido, ofrece las características mínimas para que el robot funcione, y aún así supone un gasto importante, por lo que la inclusión de servos más potentes y eficientes en el robot deberá producirse en futuras etapas de su desarrollo.

### 2.4.4 Contenido de la memoria

La función indiscutible de cualquier memoria de este tipo es dejar completa constancia del trabajo realizado a lo largo de toda la asignatura de PFC. Sin embargo, esta memoria además pretende ser, por un lado, una guía útil y suficiente para repetir todo el trabajo desarrollado aquí, y por el otro, un manual de instrucciones que permita a una tercera persona gobernar el robot sin dificultad o peligro de dañarlo.

Así pues, las páginas que siguen recogen y distribuyen toda la información referente a MIHRO en cinco bloques respectivamente referentes a su diseño mecánico, construcción, diseño electrónico, programación y resultado final. Los cinco bloques se complementan con la información contenida en el Volumen II de este proyecto. En este volumen puede encontrarse el Presupuesto, los Planos y el Código fuente del master y de uno de los esclavos. Este último documento es de vital importancia para el entendimiento del funcionamiento interno de MIHRO, pero su extensión ha hecho imposible su inclusión en esta Memoria descriptiva.

Por último, hay que puntualizar que las imágenes contenidas en el capítulo 4 de esta memoria carecen de pie de foto o referencia. Esto se debe a que el texto de este apartado está redactado de forma que se relaciona implícitamente con las imágenes que lo acompañan, por lo que ni las imágenes necesitan rotulación ni ésta acompañaría a la correcta edición y estética de este capítulo. Hay que mencionar, no obstante, que todas las imágenes contenidas en este apartado pertenecen al proyecto MIHRO.

### 2.4.5 Contenido del CD

El CD adjunto a esta memoria constituye su principal anexo, pues contiene información vital sobre el proyecto, como códigos fuente, archivos de diseño, vídeos y fotos, etc. Para la correcta visualización de sus contenidos se ha adjuntado una carpeta (*Raíz:\Programas adicionales\*) que contiene varios programas en sus versiones *demo* o *freeware* que permitirán ejecutar los ficheros de modelos tridimensionales del robot, ver los vídeos y documentos o leer con facilidad los programas contenidos en el proyecto.

De entre estos programas, EditPlus 2, una interesante herramienta de edición de código fuente para varios lenguajes de programación, necesita concretar ciertos parámetros de configuración para la correcta visualización del código fuente en formato ensamblador. El programa hace uso de un archivo de sintaxis, “PIC 16F88.stx”, para colorear los distintos elementos del programa (instrucciones, registros, comentarios, etc.), facilitando así su lectura. De forma preconfigurada, el programa busca este archivo en el directorio “D:\Programas adicionales>EditPlus 2\PIC16F88.stx”, entendiendo que el CD se reproducirá en la unidad “D”. Si no fuera así, bastaría con cambiar la letra de unidad en la ruta del archivo del campo *Syntax file*, accesible en el panel *Tools/Preferences/Settings & syntax* y seleccionando “PIC 16F88” al final de la lista *File types*.

## 3 Diseño mecánico

### 3.1 Premisas

- El robot será un hexápodo de tres grados de libertad por pata y tres grados de libertad más entre cuello y cabeza.
- Su esqueleto, sobretodo en esta etapa en que se usan motores poco potentes, deberá ser robusto pero ligero, por lo que se construirá enteramente de aluminio.
- Sus patas se distribuirán a lado y lado de su cuerpo, pero no formando dos líneas rectas paralelas, sino dos curvas convexas más separadas en el centro, cosa que permitirá al robot cambiar más fácilmente de un esquema de locomoción paralelo a uno circular o vectorial.
- Sus articulaciones siempre tendrán dos puntos de apoyo, formadas por el eje motriz de cada servo y por un eje trasero que deberá añadirse manualmente.
- Sus patas serán lo más estrechas posible para minimizar la posible obstaculización del movimiento horizontal, siempre priorizando su requisito de robustez.
- Sus patas dispondrán de una articulación “cadera” lo más chata posible, para minimizar el esfuerzo que sufrirá el eje trasero del motor que otorgue el movimiento horizontal.
- El segmento más alejado de las patas, o “tibias”, deberá estar provisto de un sistema electromecánico capaz de captar la presión contra el suelo y contra un eventual obstáculo que interrumpa la trayectoria de la pata.
  - Este mecanismo deberá ceder sin ofrecer casi resistencia ante la presión en el sentido de avance de la pata, pero también deberá mostrarse firme en el sentido de retroceso para aportar un buen punto de apoyo en el avance del robot.
  - Este mecanismo deberá ofrecer la mayor superficie de detección posible, incluyendo sobretodo el extremo más alejado de la pata, o “pié”.
  - Este mecanismo deberá integrarse en el conjunto de la pata sin romper los criterios estéticos del robot.
- Sus patas deberán poder elevarse lo máximo posible para lograr superar obstáculos altos o, en el caso de las patas delanteras, para poder ejecutar una secuencia de movimientos capaz de recoger un objeto ligero del terreno y cargarlo sobre sí.
- Sus patas deberán poder descender lo máximo posible para que el robot pueda andar con el cuerpo lo más elevado posible en caso de que éste discorra por terreno muy abrupto.
- El cuerpo del robot dispondrá de espacio suficiente para alojar a todos sus sistemas electrónicos, además de la batería y la cabeza.
- Tomando como prioritario todo lo anterior, el robot se diseñará de acuerdo con un criterio estético (que no funcional) inspirado en la anatomía de las arañas.

## 3.2 Evolución del diseño mecánico

### 3.2.1 Metodología y herramientas

El rasgo más determinante en el diseño mecánico de MIHRO es la forma de sus patas; tanto es así que el trabajo de diseño de éstas ha supuesto aproximadamente el 95% del total. La complejidad de este proceso reside en la necesidad de satisfacer las exigentes premisas reflejadas en el apartado anterior mediante el diseño de piezas que puedan fabricarse a partir de simple chapa de aluminio. Es por eso que antes de alcanzar el diseño definitivo de las patas y del robot, el proceso ha tenido que pasar por cuatro etapas previas que han permitido corregir fallos, poner a prueba las posibles soluciones y desarrollar una metodología de diseño eficiente basada en el software de tipo CAD.

Concretamente, y a excepción del primer prototipo desarrollado, todo el diseño de la mecánica de los prototipos de pata y del robot definitivo se ha llevado a cabo mediante el software Google SketchUp, en su versión 6.0.515. Este programa de bocetaje en 3D fue originalmente desarrollado por @Last Software con el objeto de ofrecer una solución sencilla y accesible para cualquier público ante el problema del diseño en 3D. Con esta filosofía, SketchUp basa su funcionamiento en un número reducido de herramientas básicas, muy magnetizadas dentro del área de diseño, e increíblemente potentes a la par que sencillas de usar. De este modo se consigue que el usuario, con una curva de aprendizaje de apenas unos días, sea capaz de expresar en 3D y de forma intuitiva sus conceptos de diseño. Con un poco más de experiencia en su manejo, el programa se convierte en una herramienta excelente de diseño mecánico, pues permite modelar piezas y volúmenes con gran rapidez y precisión, pudiendo luego evaluar la movilidad e interferencia de conjuntos articulados y elementos móviles.

En el caso del proyecto MIHRO, este programa ha jugado un papel crucial en el diseño mecánico del robot, pues ha permitido modelar y corregir las piezas, simular su movilidad e interferencia y, una vez verificadas, desplegarlas para construir plantillas que puedan traspasarse directamente a la chapa de aluminio para su posterior recorte. Sólo de esta manera ha sido posible construir piezas complejas que encajaran perfectamente entre sí, manteniendo una correspondencia con el diseño informático original de aproximadamente  $\pm 1\text{mm}$  de error medio.

Se recomienda encarecidamente instalar el programa (incluido en el CD) para poder visualizar el modelo mecánico definitivo de MIHRO. Además, para la correcta visualización de los casi 900 elementos que comprende el modelo, entre piezas y ayudantes, se recomienda navegar a través de la jerarquía de grupos usando la ventana *Outliner*, accesible desde el menú *Window*. Del mismo modo, se recomienda el uso de las órdenes *Edit/Hide* y *Edit/Unhide* para mostrar u ocultar las partes grandes del modelo que obstaculicen la visión de las pequeñas.

### 3.2.2 Prototipos

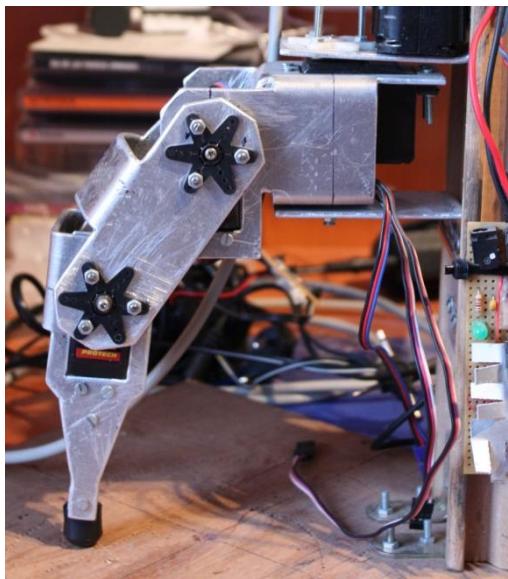


Figura F. 14 Primer prototipo de pata  
(Fuente: Proyecto MIHRO)

El primer prototipo desarrollado fue el único que no se diseñó usando SketchUp, por lo que su estructura es mucho más tosca y está menos optimizada. Esta pata jamás llegó a moverse, pues se consideró que su diseño no cumplía con los requerimientos de este proyecto, y por tanto el trabajo prosiguió con el diseño de otro prototipo más optimizado, y no con el desarrollo de la electrónica de control. Entre sus principales problemas destacan la anchura de las patas, pues la propia estructura de soporte de los servos se extiende hacia la parte posterior de éstos para hacer las veces de eje trasero, la escasa movilidad vertical del conjunto y la ausencia de los sistemas de detección de suelo y obstáculos.

El segundo prototipo que se diseñó jamás llegó a ver la luz, pues fue rápidamente sustituido por el tercer prototipo. Aunque conserva algunos patrones de su predecesor, mejora sustancialmente su movilidad vertical e incorpora elementos que han permanecido hasta el diseño final, como el sensor de contacto con el suelo basado en una varilla metálica, acabada en un taco de goma y suspendida sobre un muelle. Además, en este prototipo se estudió la posible inclusión de un *encoder* realimentador de la posición vertical de la pata, que se descartó cuando se desarrolló una estrategia de software capaz de emular esta función incluso con mayor precisión.

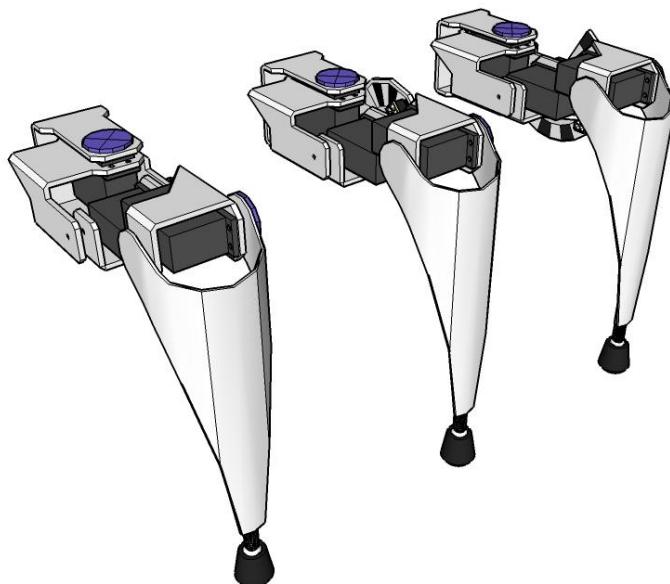
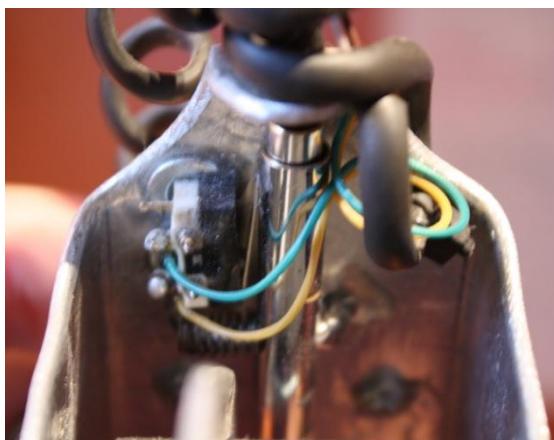
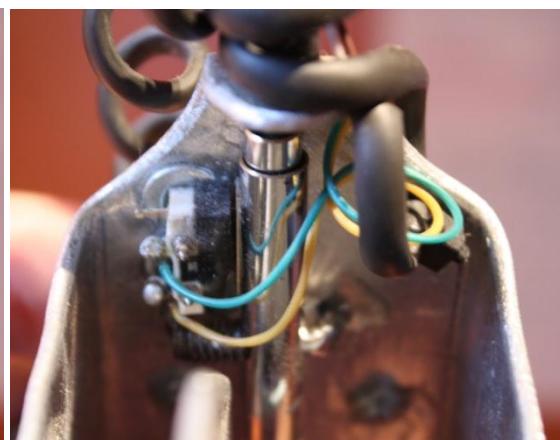


Figura F. 15: Segundo prototipo de pata  
(Fuente: Proyecto MIHRO)

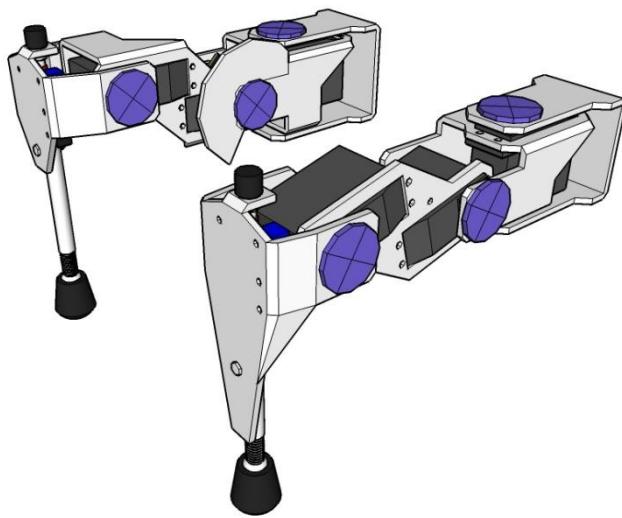
El tercer prototipo se diseñó con la idea específica de incluir un mecanismo de detección del suelo y de los obstáculos. Este mecanismo, distinto al que finalmente incorporan las patas de MIHRO, consiste en un tubo estrecho envainado dentro de otro tubo algo más grueso, de modo que el primero, en cuyo extremo se asienta el tope de goma que hace de “pié”, puede ascender por el interior del segundo para presionar un pulsador cuando la pata hace contacto con el suelo. Al mismo tiempo, el tubo grueso se sujet a la estructura de la tibia atravesado sólo por un tornillo, cosa que le permite pivotar en el plano paralelo a la cara frontal de la tibia cuando la trayectoria de la pata interfiere con un obstáculo. Al pivotar, el movimiento de la parte superior del tubo presiona un pequeño interruptor que hace efectiva la detección del obstáculo.



**Figura F. 16: Detector de obstáculo abierto**  
(Fuente: Proyecto MIHRO)



**Figura F. 17: Detector de obstáculo cerrado**  
(Fuente: Proyecto MIHRO)



**Figura F. 18: Tercer prototipo de pata**  
(Fuente: Proyecto MIHRO)

El diseño original de este prototipo aún comprendía la inclusión del *encoder*. Más tarde este diseño se revisó para eliminar dicho dispositivo, incluir los ejes traseros en los servos (cosa que estrechó la pata) y adecuar su estética al aspecto del segundo prototipo. Finalmente la pata se fabricó y se acopló al soporte de madera del primer prototipo. Esta pata y los sistemas electrónicos que la hacen funcionar constituyen la maqueta de pruebas del proyecto MIHRO. Entre sus defectos mecánicos más importantes destaca la insuficiente capacidad de elevación

de las tibias, la excesiva distancia entre el eje del motor elevador del “fémur” y el eje del motor de movimiento horizontal (cosa que hace que el eje trasero de éste sufra) y la insuficiente superficie de detección de obstáculos de las tibias.

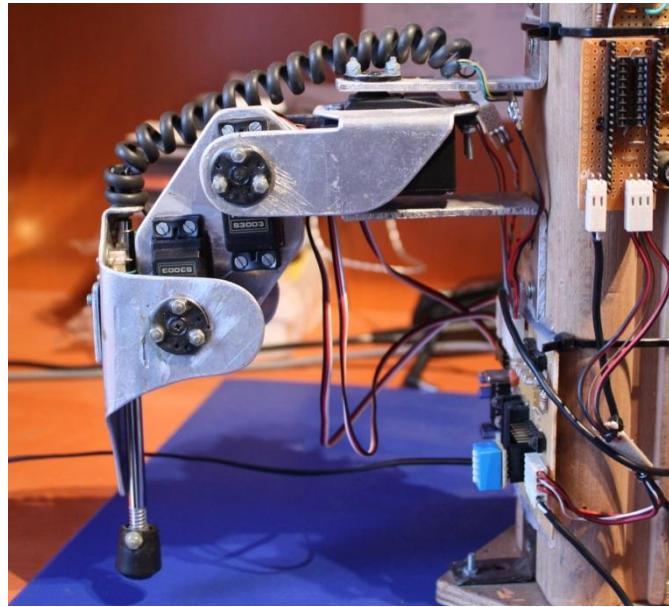


Figura F. 19: Aspecto final del tercer prototipo  
(Fuente: Proyecto MIHRO)

En cuanto al cuarto prototipo, quizás no sea demasiado exacto llamarlo así, pues en realidad se trata de un largo proceso de diseño que parte de un modelo completamente distinto a los tres prototipos anteriores, y que culmina con el diseño final de la pata de MIHRO. En la primera etapa de dicho proceso, se diseña un modelo de pata con mayor capacidad para recogerse sobre sí mismo y mayor capacidad de elevación vertical de la tibia. Así mismo, este primer concepto ofrece la ventaja de que, en su posición recogida, el fémur permanece completamente vertical, lo que reduce drásticamente el consumo de los motores de la pata cuando ésta se halla en su posición inicial. Durante esta misma etapa se realizan varias pruebas de rediseño del mecanismo de detección de suelo y obstáculos para adaptarlo a la nueva estructura y se intenta, sin éxito, diseñar un fémur que permita situar su motor de elevación bajo el motor de movimiento horizontal de la cadera.

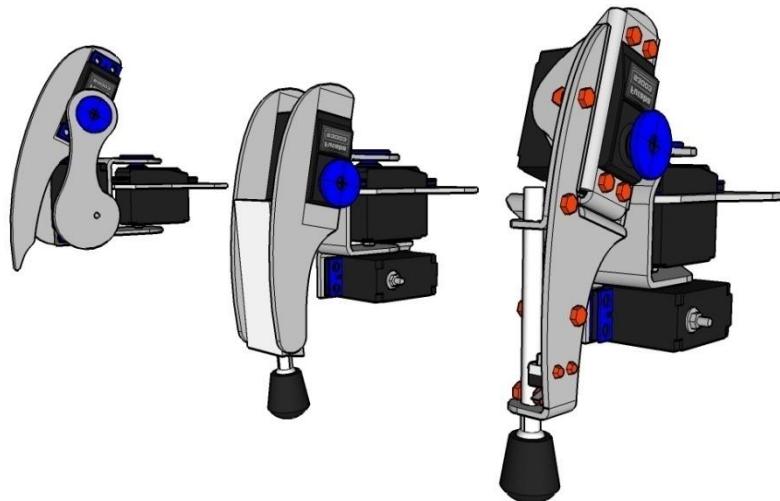
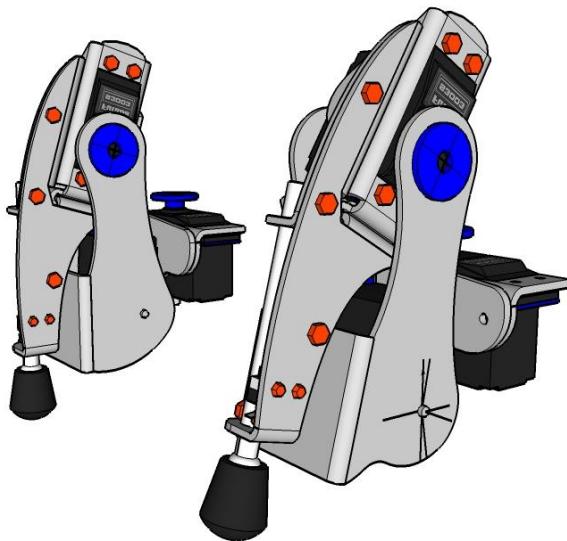


Figura F. 20: Primera etapa del diseño final de la pata  
(Fuente: Proyecto MIHRO)

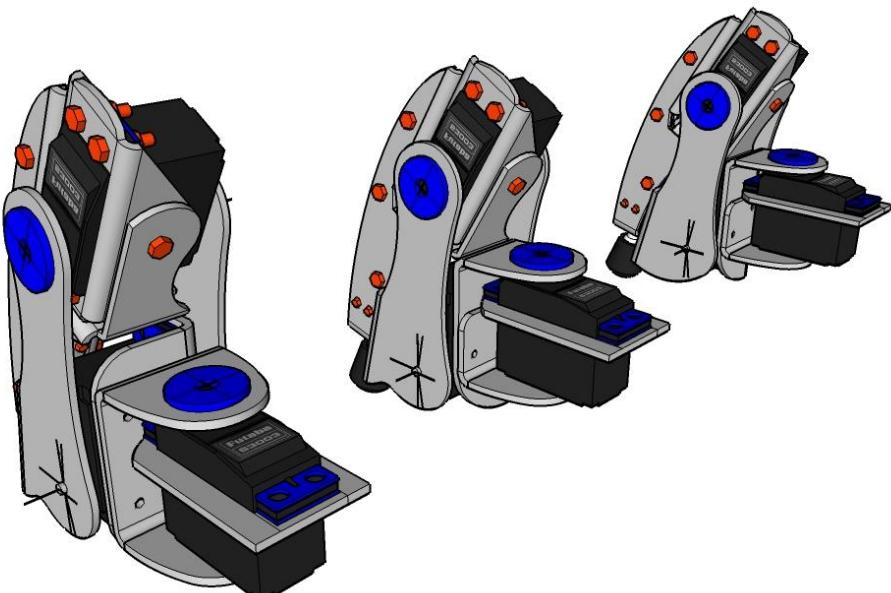


**Figura F. 21: Modelo de cadera móvil**  
(Fuente: Proyecto MIHRO)

Durante una segunda etapa se ponen a prueba dos estructuras de cadera posibles que condicionarían drásticamente el aspecto final del robot. En la primera estructura, el motor de movimiento horizontal de la cadera forma parte de la pata, por lo que el cuerpo del robot deberá sujetarlo por sus dos ejes, dejando suficiente espacio libre en su interior como para que el motor, junto con la pata, se muevan libremente. Conceptualmente, esta es la misma estructura que usan los robots de Matt Denton (ver capítulo de Antecedentes), por lo que cualquier parecido con un sistema tan trillado se ha considerado

poco deseable. Además, aunque este sistema otorga a la pata una gran capacidad de movimiento horizontal, reduce su capacidad de movimiento vertical. No obstante, para atenuar esta circunstancia, se diseña un fémur con el “puente” (el segmento de metal que une las dos aletas del fémur) avanzado, cosa que aumenta la capacidad de descenso del mismo.

En la segunda estructura, por el contrario, el motor de movimiento horizontal de la cadera forma parte de la estructura del “tórax” y no de la pata, por lo que, para empezar, el motor no se mueve dentro del cuerpo del robot, de manera que todo el área no ocupada por los motores del mismo se convierte en zona útil para acoplar los sistemas que conforman su electrónica.



**Figura F. 22: Modelo de cadera fija**  
(Fuente: Proyecto MIHRO)

Este sistema necesita de una estructura que conecte los ejes del servo de movimiento horizontal (sujeto al cuerpo del robot por sus aletas de soporte) a la estructura de la pata. Dicha estructura se diseña en primera instancia como una sola pieza que conecta el eje motor del servo de movimiento horizontal a las aletas de soporte del motor de elevación del fémur, y una segunda pieza que sujeta el eje trasero del servo de movimiento horizontal a la primera pieza (Figura F.22, centro y derecha). De esta forma, esta segunda solución sólo reduce el rango de movimiento horizontal de las patas y no mejora su capacidad de ascenso, aunque sí de descenso, y más aún si se usa el fémur de puente avanzado.

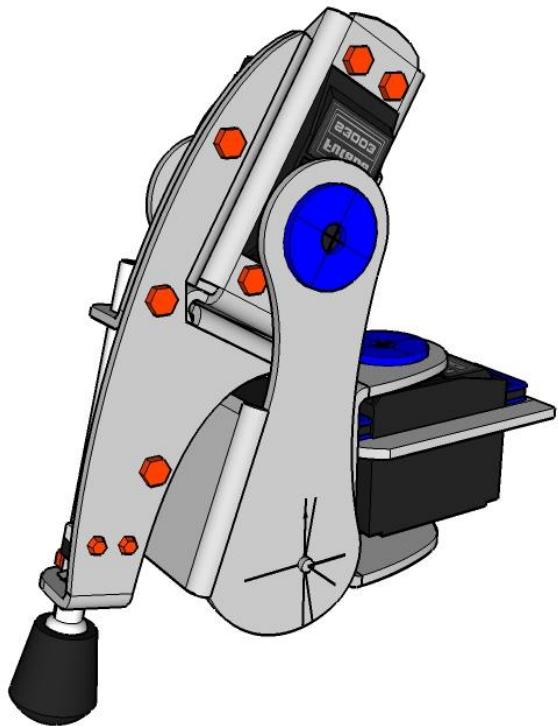


Figura F. 23: Diseño final de las patas  
(Fuente: Proyecto MIHRO)

No obstante, si se modifican estas piezas de la cadera, de modo que exista una única pieza que abrace los dos ejes del servo de movimiento horizontal y otra pieza que sujete las aletas de soporte del servo del fémur, se puede mejorar la capacidad de movimiento horizontal de la pata y sobretodo su capacidad de elevación, eso sí, a costa de empeorar su capacidad de descenso (Figura F.22, izquierda). Finalmente, se halló que la opción más equilibrada en cuanto a capacidad de movimiento horizontal, elevación y descenso de la pata surgía al diseñar un nuevo fémur de puente avanzado pero recortado. Y visto que las propiedades mecánicas de este diseño eran las mejores de cuantas se habían probado, y que satisfacían perfectamente las premisas de este proyecto, se optó por escoger esta solución como estructura definitiva de las patas de MIHRO.

### 3.2.3 Ensayos (maqueta de pruebas)

Los experimentos realizados con la maqueta y el tercer prototipo de pata han resultado ser de vital importancia para el desarrollo final del proyecto, sobre todo en lo que se refiere a la implementación de las principales estrategias de software para generar el movimiento de los servos. Con la primera controladora, una pequeña placa electrónica basada en el PIC16F84A, se lograron establecer las principales rutinas de generación de señales de control PWM (Pulse Width Modulation) para los servos escogidos, los Futaba S3003. Este es un servo analógico estándar cuyo control se basa en la modulación del tiempo de nivel alto, o  $T_{ON}$ , de una señal “consigna” que mantiene un período constante  $T$  de 20ms. En el interior del servo esta consigna se compara constantemente con una tensión de referencia.

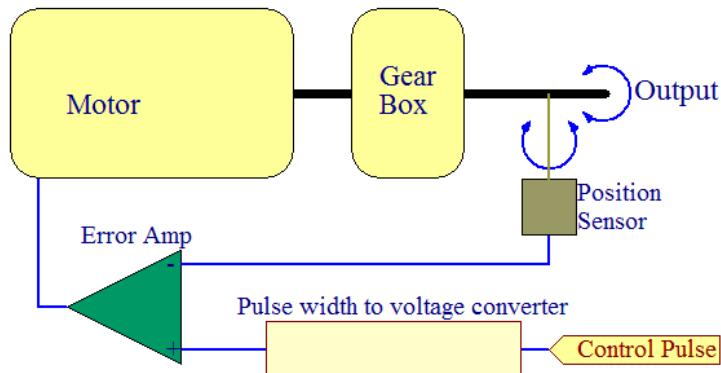


Figura F. 24: Control de posición del servo  
(Fuente: [www.digitalnemesis.com](http://www.digitalnemesis.com))

Dicha tensión de referencia está condicionada por la posición de un potenciómetro, que a su vez está mecánicamente conectado al eje de salida del servomotor. De esta manera, la señal PWM es convertida a un nivel de tensión que se compara con el nivel de realimentación de la posición del servo; si hay diferencia, el amplificador del error hará que el motor gire en un sentido u otro para minimizarla, y si las tensiones de referencia y consigna se igualan, el motor se detendrá en esa posición. En el caso de los Futaba S3003, y de la mayoría de servos analógicos que hay hoy día en el mercado, este  $T_{ON}$  representa una pequeña fracción de aproximadamente el 10% del período  $T$ , cosa que pone de manifiesto la ineficiencia inherente a este tipo de servos en comparación con los llamados "digitales", en los que este pulso de control llega a alcanzar entre el 20% y el 60% del período total.

Si se indaga un poco sobre el parámetro  $T_{ON}$  se descubre que la mayoría de servos analógicos de esta escala responden ante una anchura de pulso que varía aproximadamente entre 0'3 y 2'3ms, de forma que una señal con el pulso mínimo conducirá al servo a posicionarse en el extremo inferior de su rango de movimiento, mientras que una con el pulso máximo le llevará a su extremo superior, pudiendo configurarse en cualquiera de sus posiciones intermedias.

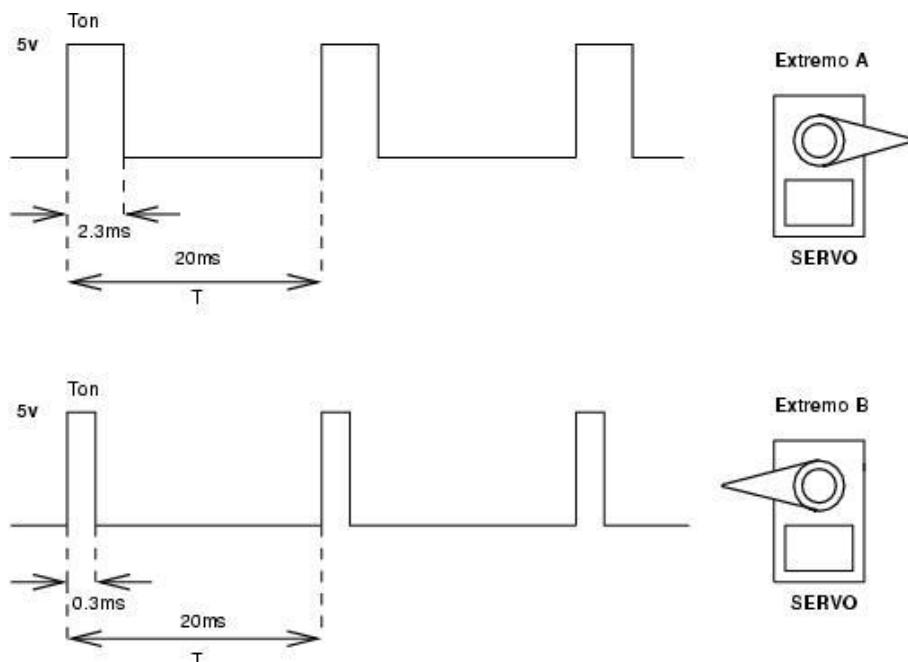
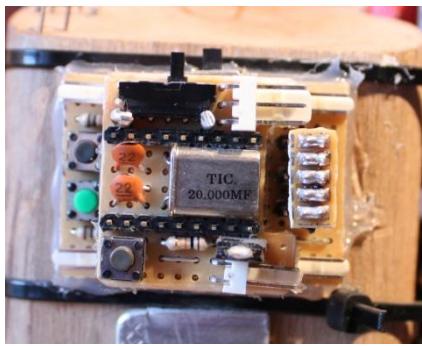


Figura F. 25: Características de la señal PWM  
(Fuente: [www.iarobotics.com](http://www.iarobotics.com))



**Figura F. 26: Primera controladora**  
(Fuente: Proyecto MIHRO)

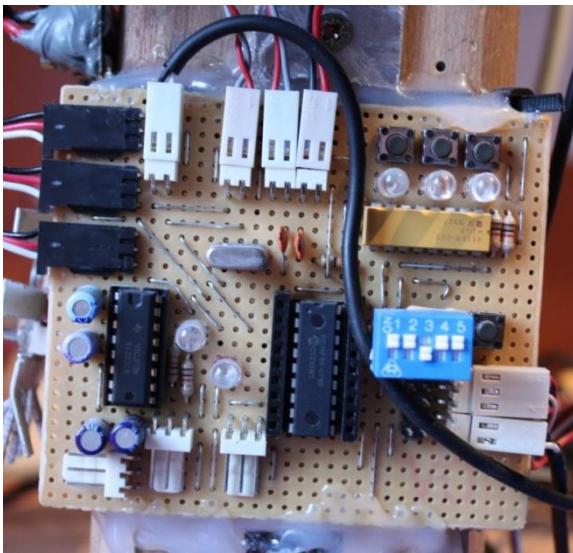
De este modo, el primer experimento que se realizó sobre la maqueta fue la de determinar la correcta temporización de los  $T_{ON}$  de tres señales de control para hallar la correlación exacta con las posiciones reales de los tres servos. Para ello la placa se servía de dos botones que aumentaban o disminuían una variable de control que actuaba sobre una rutina de retardo que permitía cuantificar el tiempo transcurrido desde que el pulso comienza en nivel alto hasta que termina volviendo al nivel bajo. Además, hubo que programar el sistema interruptivo y el módulo Timer0 del microcontrolador para poder repetir el proceso cada 20ms. Este sistema, de una forma casi idéntica, ha trascendido hasta los programas finales de control de MIHRO, por lo que la explicación más detallada del código fuente se realizará en el apartado de programación de esta memoria. Finalmente, se determinó que el servo se posiciona en  $0^\circ$  (límite inferior) mediante un pulso de 0'48ms y a los  $180^\circ$  (límite superior) a los 2'28ms. El intervalo de tiempo entre estos niveles mínimo y máximo se cuantificaba mediante una variable que ofrecía un rango de 150 valores posibles, por lo que la posición del servo podía controlarse con  $1'2^\circ$  de precisión.



**Figura F. 27**  
**Figura F. 28**  
Imágenes panorámica y de detalle de la señal PWM obtenida fluctuando entre dos valores  
**Figura F. 29**  
(Fuente: Proyecto MIHRO)

Después de éste y otros experimentos realizados hubo que abandonar esta primera placa de control por problemas inherentes al PIC16f84A, un dispositivo explotado hasta la saciedad por ser uno de los más vendidos de la marca Microchip, pero hoy día bastante desfasado. El sustituto escogido fue el PIC16f88, un recambio generacional relativamente reciente del PIC16f84, que integra en un espacio también de 18 patillas la mayoría de módulos de otros microcontroladores mucho más grandes, convirtiéndose en uno de los PICs más versátiles en relación a su tamaño que ofrece Microchip en esta gama. Para explorar las principales opciones de comunicación y procesamiento de este dispositivo, se diseñó una nueva placa entrenadora que permite igualmente controlar los tres motores del prototipo, a la vez que explotar las posibilidades de comunicación I<sup>2</sup>C y USART (para RS-232) de este microcontrolador. La placa además incorpora tres botones de control, cinco LEDs y conectividad para la programadora y para los dos sensores de contacto de la tibia del tercer prototipo.

Con la placa entrenadora se pudieron realizar importantes avances en el control del tercer prototipo; entre ellos destacan:



**Figura F. 30: Placa entrenadora**  
(Fuente: Proyecto MIHRO)

- Mejora de la rutina de retraso para lograr controlar la posición de los servos en base a una variable de 250 valores posibles, cifra que se corresponde mucho mejor con las características de una máquina de 8 bits. Con esta rutina se pueden controlar los servos con una precisión de  $0'72^\circ$ .
- Programación de una secuencia de movimiento que permite a la pata dar un “paso” y retroceder hasta su posición inicial. Después se mejora la secuencia para reducir la inercia en sus extremos, evitando así temblores indeseados.
- Introducción de cinco parámetros que controlan y definen todas las características del paso. Estas son:
  - **Speed:** Proporción en la que fluctúan las variables que definen la posición de los motores cada 20ms.
  - **Centro:** Posición central del paso. Permite corregir las pequeñas diferencias de montaje de cada pata, así como adaptar la posición central de las patas oblícuas.
  - **Rango:** Distancia de avance y retroceso que cubrirá el paso respecto del centro. Influye sobre la velocidad de avance del robot.
  - **Zanco:** Altura a la que se produce el movimiento horizontal de avance respecto del de retroceso de la pata.
  - **Altura:** Altura a la que se produce el movimiento horizontal de retroceso de la pata. Determina la altura del cuerpo del robot respecto del suelo mientras éste está andando.
- Control de estas cinco variables mediante los tres botones de la placa. El experimento demostró que podía controlarse el movimiento de la pata de forma dinámica.

La conclusión más importante a la que se llegó era que si se mantenía la variable *Speed* acotada en un valor inferior a 10, la pata respondía con suficiente fidelidad a las consignas de movimiento. Una velocidad superior a 10 provocaba reducciones en los rangos de movimiento, pues el motor no es lo bastante rápido como para asumir una fluctuación tan precipitada. De este modo, se considera que la pata siempre está posicionada de acuerdo con el valor instantáneo de su variable de control, cosa que hace que no sea necesario un *encoder* para realimentar la posición vertical u horizontal de la pata cuando se produce la detección del suelo o de un obstáculo; basta con capturar el valor de dicha variable.

### 3.2.4 Características mecánicas finales

Finalmente, el diseño mecánico del robot satisface las premisas de este proyecto, y además reúne las siguientes características:

Característica	MIHRO
• Peso del robot	2'250 Kg
• Tamaño robot (rectángulo circunscrito)	31,1 cm X 32,1 cm
• Tamaño Tórax Base (rectángulo circunscrito)	26 cm X 17 cm
• Área útil del Tórax Base	162 cm <sup>2</sup>
• Área útil máxima de las Tapas Superior e Inferior	281 cm <sup>2</sup> x2
• Distancia central útil	5 cm
• Distancia cervical útil	3 cm
• Distancia entre las patas delanteras a 90°	3 cm
• Distancia entre la Tapa Superior y el Tórax Base	2 cm
• Distancia entre la Tapa Inferior y el Tórax Base	3 cm
• Movilidad horizontal de las patas delanteras, Fémur plegado	-60°, + 95°
• Movilidad horizontal de las patas delanteras, Fémur desplegado	-75°, + 95°
• Movilidad horizontal de las patas centrales, Fémur plegado	-60°, +60°
• Movilidad horizontal de las patas centrales, Fémur desplegado	-70°, +60°
• Movilidad horizontal de las patas traseras, Fémur plegado	-95°, + 60°
• Movilidad horizontal de las patas traseras, Fémur desplegado	-110°, + 60°
• Capacidad de elevación del Fémur desde la posición inicial	35°
• Capacidad de descenso del Fémur desde la posición inicial	150°
• Capacidad de elevación de la tibia en posición inicial	95°
• Capacidad de elevación de la tibia en elevación máxima del fémur	25°
• Rango de movimiento máximo de las Tibias	190°
• Distancia entre los ejes horizontal y vertical de la Cadera	3,8 cm
• Distancia entre los ejes de Fémur y Tibia	7 cm
• Apertura del paso (Fémur: 25°, Tibia: 25°) con incidencia de las patas adyacentes	-35°, + 35°
• Capacidad de inclinación lateral	31°
• Capacidad de inclinación frontal	38°
• Altura máxima del cuerpo en posición horizontal	16,1 cm
• Altura máxima de las patas delanteras en inclinación frontal máxima	38,7 cm

Tabla T. 3: Propiedades mecánicas del robot MIHRO  
 (Fuente: Proyecto MIHRO)

Algunas imágenes del diseño definitivo y el resultado final.

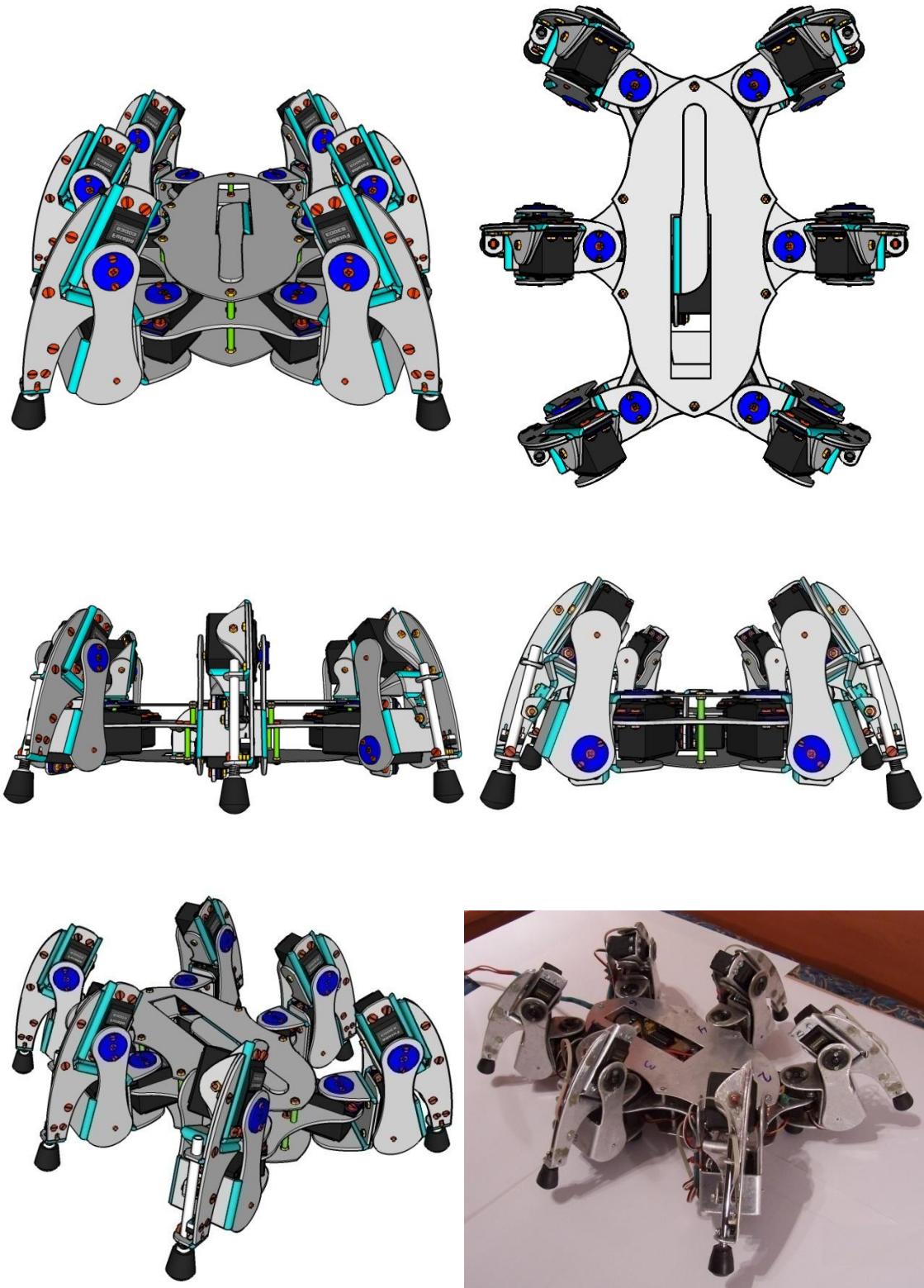


Figura F. 31: Collage con distintas vistas del diseño y el resultado finales  
(Fuente: Proyecto MIHRO)

Una variable crítica en el diseño mecánico es la distancia entre los ejes de los motores “Fémur” y “Tibia”, que el en robot MIHRO mide exactamente 7cm. Cuanto más corta sea esta distancia, menos capacidad tendrá la pata para recogerse sobre sí misma, menos altura podrá alcanzar en su posición erguida y más cortos serán sus pasos, por lo que andará a velocidad más lenta. No obstante, cuanto más larga sea esta distancia, menor será el par efectivo de los motores encargados de sostener el peso del robot, pues la fuerza que puede ejercer un motor en su eje queda dividida por el valor de esta distancia en el otro lado del fémur. De este modo, si la especificación técnica de los servos Futaba S3003 dice que pueden ejercer un par de 4'1 Kg/cm alimentados a 6V, significa que podrán ejercer un par de  $4'1\text{Kg}/7\text{cm} = 0'58\text{Kg}$  en el otro lado del fémur.

Hay que tener en cuenta que tanto el motor “Fémur” como el “Tibia” colaboran en mayor o menor medida en el levantamiento del peso del robot, dependiendo del ángulo en que se encuentren el uno del otro, y ambos respecto de la vertical. No obstante, en ningún caso salvo en el ideal, y sólo si ambos motores se hallan en una posición específica, la pata podrá ofrecer una fuerza de máxima de  $0,58\text{ Kg} \times 2 = 1'16\text{ Kg}$ . A la práctica este valor es inalcanzable, pero para corroborarlo se realiza un experimento de levantamiento de carga (incluido en el apartado de vídeos del CD) que demuestra que el robot puede levantar un total de 618g, además de su propio peso. Este es un valor pequeño, pero acorde a lo que cabía esperar de este servomotor en una aplicación así. No obstante, hay que decir que es un valor poco fiable, pues se observó durante la experimentación que cuanto más aumentaba el peso añadido, mayor se hacía el consumo eléctrico de los motores, alcanzando, a partir de cierto peso, un consumo total de 4'14 A, el máximo que puede ofrecer la fuente de alimentación. En estas circunstancias, y dado que no se dispone de una fuente de alimentación con un límite de entrega de corriente mayor, no se pudo comprobar en este experimento el peso máximo que podría levantar el robot sin estar sometido a esta restricción. Por otro lado, el experimento reveló varios datos sobre el consumo eléctrico del robot en distintas circunstancias (motores a 6V):

<b>Situación</b>	<b>Consumo</b>
• Patas recogidas, sin apoyo	290mA
• Patas extendidas, sin apoyo	740mA
• Patas extendidas, con apoyo	1'40A
• Maniobra de levantamiento sin peso	3'10A – 3'50A
• Maniobra de levantamiento con peso máximo	4'14A
• Robot caminando	1'70A – 3'80A
• Señales PWM apagadas (motor inactivo)	190mA
• Consumo medio mínimo unitario de motores activos	16mA
• Consumo medio máximo unitario de motores activos	230mA
• Electrónica, con alimentación de motores, andando	75mA – 95mA
• Electrónica, con alimentación de motores, en reposo	65mA
• Electrónica, sin alimentación de motores	90mA

**Tabla T. 4: Consumo eléctrico del robot**  
(Fuente: Proyecto MIHRO)

## 4 Fabricación y montaje del esqueleto

Entre las piezas que componen el robot se cuentan tubos metálicos, muelles, sensores de contacto, motorización, tornillería, separadores, tacos de goma y electrónica en general. El resto de piezas del robot, a las que nos referiremos como “esqueleto” han sido completamente fabricadas a mano y de forma artesanal a partir de chapa de aluminio normal de 1, 2 y 2,5 mm de espesor. Todas las piezas del esqueleto han sido diseñadas por ordenador, de acuerdo con el proceso visto en el capítulo anterior, pero ninguna de ellas es idéntica a su modelo informático, ni tampoco son idénticas entre sí. La razón es que el proceso de fabricación es impreciso, y cada pieza ha de ser ligeramente modificada durante el proceso de montaje para encajar con sus piezas colindantes. De este modo, y aunque muchas de las piezas se fabrican en grupos de seis, cada una de ellas es única y no puede ser sustituida por otra de sus análogas.

El cuerpo mecánico del robot está compuesto de un total de 853 componentes:

- 19 servomotores Standard Futaba S3003
  - [S0] – 1 servo “Cuello 1” (Elevador de Cabeza)
  - [S1] – 6 servos “Cadera” (Motor 1, eje horizontal)
  - [S2] – 6 servos “Fémur” (Motor 2, eje vertical)
  - [S3] – 6 servos “Tíbia” (Motor 3, eje vertical)
- 2 servomotores Mini (el proyecto actual no incluye el diseño de la cabeza)
  - [S4] – 1 servo “Cuello 2” en eje horizontal
  - [S5] – 1 servo “Cuello 3” en eje vertical
- 125 elementos elásticos o blandos
  - [E0] – 76 protectores de goma compatibles con servos Futaba Standard
  - [E1] – 19 soportes de plástico para el eje del servo Standard Futaba
  - [E2] – 2 soportes de plástico compatibles con servo Mini (no incluido)
  - [E3] – 12 sensores de contacto de tipo Mini-Bumper.
  - [E4] – 6 tacos huecos de goma antideslizante
  - [E5] – 6 muelles de Ø6,5mm
  - [E6] – 6 muelles de Ø4,5mm
- 18 separadores hechos a medida con tubo de aluminio de 0,5mm y Ø4mm
  - [Y0] – 6 separadores para la Tapa Superior del Tórax
  - [Y1] – 6 separadores para la Tapa Inferior del Tórax
  - [Y2] – 6 separadores para unión elástica de la Tibia Móvil
- 150 arandelas
  - [A0] – 12 arandelas normales M2,5
  - [A1] – 132 arandelas normales M3
  - [A2] – 6 arandelas de presión M2,5

- 250 tornillos y 243 tuercas
  - [T0] – 19 tornillos de fijación para ejes de servo Futaba Standard
  - [T1] – 50 tornillos de cabeza plana de Ø2mm x 10mm + 50 tuercas
  - [T2] – 6 tornillos de cabeza plana de Ø2,5mm x 20mm + 18 tuercas
  - [T3] – 8 tornillos de cabeza plana de Ø3mm x 10mm + 8 tuercas
  - [T4] – 76 tornillos de cabeza plana de Ø3mm x 20mm + 76 tuercas
  - [T5] – 6 tornillos de cabeza plana de Ø3mm x 35mm + 12 tuercas
  - [T6] – 12 tornillos de cabeza cónica de Ø2,5mm x 20mm
  - [T7] – 49 tornillos de cabeza cónica de Ø3mm x 10mm + 49 tuercas
  - [T8] – 12 tornillos de cabeza cónica de Ø3mm x 30mm + 18 tuercas
  - [T9] – 12 tornillos de cabeza hexagonal de Ø3mm x 10mm + 12 tuercas
- 48 piezas del esqueleto
  - Piezas del Cuello
    - [P10] – Cuello 1
  - Piezas del Tórax
    - [P00] – Tórax Base
    - [P01] – Tapa Superior del Tórax
    - [P02] – Tapa Inferior del Tórax
    - [P11] – Hombro Izquierdo
    - [P12] – Hombro Derecho
  - Piezas de cada pata (simétricas en dos grupos de tres)
    - [P03] – 6 Conector Cadera-Tórax
    - [P04] – 6 Conector Cadera-Fémur
    - [P05] – 6 Fémur
    - [P06] – 6 Parte Fija de la Tibia
    - [P07] – 6 Soporte Superior del Pié
    - [P08] – 6 Parte Móvil de la Tibia
    - [P09] – 6 tubos de latón cromado de Ø6mm (Pies)
  - Piezas de la Cabeza
    - (el proyecto actual no comprende el diseño de la cabeza)

Hay que mencionar que los elementos del despiece marcados como [S4], [S5] y [E2] corresponden al montaje de la cabeza (que se lleva a cabo en la etapa 6 del desarrollo global de MIHRO), por lo que no se concretan mayores detalles sobre estos elementos en este proyecto. Así mismo, las piezas [P10], [P11] y [P12], correspondientes al mecanismo de elevación del cuello, junto con los elementos que las acompañan (como el servo [S0]), han sido diseñados dentro del proyecto actual, pero sin llegar a montarse, pues su única razón de ser es el de elevar una cabeza que aún no existe, y el espacio que ocupan se hace necesario para instalar la placa entrenadora, que en esta etapa del proyecto hace las veces de “cerebro” del robot.

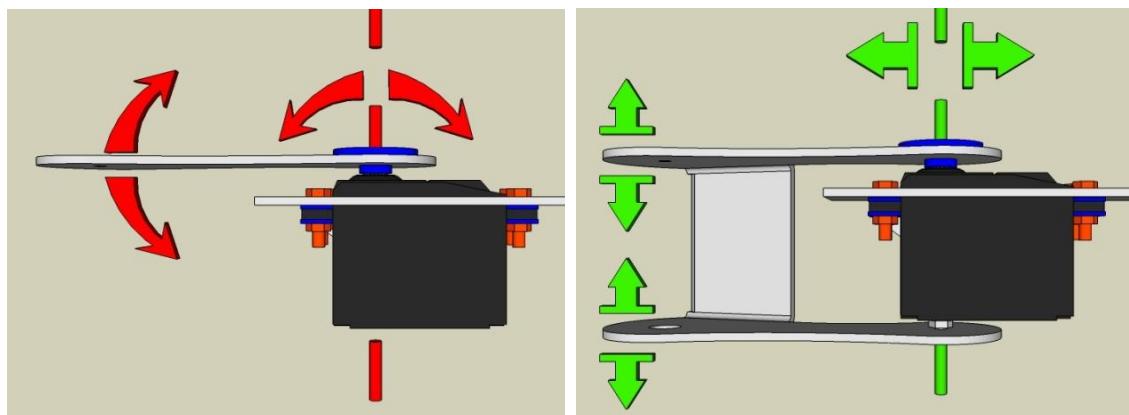
## 4.1 Proceso de modificación de los servos

El servo escogido para las patas del robot es el Futaba S3003, un servomotor de modelismo, analógico y de tamaño estándar. El principal motivo de la elección de este servo es su popularidad y su precio económico, ya que el ajustado presupuesto de que se dispone para realizar este proyecto es uno de sus principales factores limitantes, y cualquier servo de precio unitario más caro (en este caso 12€, aunque fue comprado por 10€), multiplicado por los 19 motores de este tipo que incorpora el robot habría supuesto un gasto excesivo. Este es un servo veterano y muy conocido en el ámbito del radiocontrol, puesto que se suele emplear para montar, por ejemplo, la dirección de los coches teledirigidos. Su fiabilidad está más que comprobada, ofrece buen rendimiento y es fácil de encontrar en las tiendas especializadas. Despues de hacer los cálculos de diseño y de comprobar que el par que ofrece este servo es suficiente para levantar el peso del robot, más el posible extra de carga, se decidió que era apto para aplicarse en esta etapa.



<i>Valores típicos:</i>	<b>Alim. 4,8 V</b>	<b>Alim. 6 V</b>
<b>Par:</b>	3,2 Kg/cm	4,1 Kg/cm
<b>Velocidad:</b>	0,23 sec/60°	0,16 sec/60°
<b>Reductora:</b>	Nylon	
<b>Dimensiones:</b>	41x20x36mm	
<b>Peso:</b>	37,2g	
<b>Precio:</b>	~12€	

Este servo se comercializa con piñonería de nylon y eje de plástico, y está diseñado para acoplarse sólo por un extremo de su eje de rotación. Esto significa que es vulnerable a fuerzas de torsión excesivas, pudiendo llegar a romperse si se somete a estrés fuera de su eje normal de rotación. Aunque el peso del robot no fuera suficiente para romper los ejes de estos servos, la idea de sujetar los motores sólo por uno de los extremos de su eje de rotación sólo contribuiría a aumentar la fragilidad del robot y a disminuir el tiempo de vida de los motores, por eso es necesario hacer ciertas modificaciones en cada uno de los servos antes de acoplarlos al robot. Estas modificaciones llevan, finalmente, a instalar un nuevo extremo del eje de rotación en la tapa trasera del servo, de forma que éste pueda acoplarse al robot agarrado por dos puntos y no sólo uno, aumentando así su robustez y durabilidad.



A continuación se describen los pasos a seguir para instalar los extremos traseros del eje de los servos:



1. Se desenroscan los cuatro tornillos que sujetan la tapa trasera del servo. Estos tornillos son tan largos porque también sujetan la caja de engranajes al cuerpo del servo, el cual se deberá manipular con precaución una vez sacados los tornillos, puesto que de lo contrario se podría abrir y desmontar la caja de engranajes accidentalmente.



2. Una vez abierto el servo queda por un lado la tapa y por el otro el cuerpo boca abajo del servo, mostrando su circuitería interna.



3. La tapa trasera del servo tiene forma rectangular, y sujeta los cables que salen del interior del servo mediante una pestaña sobresaliente situada en el borde de uno de los lados cortos del rectángulo. Esta pestaña se corta fácilmente con un *cutter*. Del mismo modo, es necesario abrir un hueco en el marco interior que permite el correcto encaje de tapa y cuerpo del servo, por el lado contrario al de la pestaña.



4. La pestaña muestra dos pequeños orificios en la parte interior de la tapa, que encajan con dos protuberancias situadas en esta misma posición, pero del lado del cuerpo del servo. Es fácil reconocerlas, porque están situadas justo a cada lado de la salida de los cables. De nuevo con el *cutter* se cortan estas protuberancias, pero con cuidado de no dañar los cables.



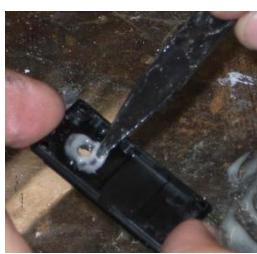
5. También muestra la tapa del servo, en el centro de su mitad más alejada de la salida de los cables, una hendidura redonda que coincide con el centro del pequeño motor de CC situado en el interior del servo. Por el lado interno de la tapa, esta hendidura toma la forma de una protuberancia que opriime este pequeño motor y lo mantiene fijo. Cabe decir que el eje del servo está situado en su mitad más cercana a la salida de los cables, mientras que la hendidura está en la mitad contraria. No obstante, si se gira la tapa trasera 180° esta hendidura queda perfectamente alineada con el eje del servo, por eso se usará para colocar justo ahí el tornillo que hará de eje trasero.



6. Es necesario terminar de perforar la hendidura con una broca de Ø3mm.



7. La perforación dejará unos bordes de plástico en la cara interna de la tapa que habrá que cortar y limar hasta que la superficie interior de la tapa quede lo más lisa posible. Como el orificio se encuentra encajonado será más práctico limar el reborde con un fresolín y una punta de lima. Entonces la tapa estará lista para colocar y fijar el tornillo que hará de eje.



8. Además de con la tuerca pertinente, el tornillo quedará firmemente fijado al servo con una buena cantidad de cola de epoxi. Esta cola tiene una doble función: por un lado evita que las vibraciones derivadas del uso continuado del motor aflojen la tuerca y debiliten el eje, y por otro lado forma un cuerpo compacto que evita que las fuerzas perpendiculares que reciba el eje dañen o deformen la tapa de plástico.



9. En primer lugar se dibuja un círculo de cola de epoxi por el lado interior de la tapa, alrededor del agujero donde irá el tornillo. Sobre esta cola se coloca una arandela del nº 3, [A1].



10. A continuación se pone algo de cola sobre el centro de la arandela y se hace pasar a través de ella un tornillo de cabeza cónica de Ø3mm x 10mm, [T7]. El sobrante de cola que queda en el centro de la arandela impregna la rosca del tornillo, lo que hará que, una vez puesta la tuerca y seca la cola, ésta quede aún más firme. Por supuesto esto significa que una vez instalado este eje ya no se podrá retirar de la tapa del servo. Para esta operación es indispensable usar un tornillo de cabeza cónica, puesto que el espacio disponible dentro del servo para la cabeza del tornillo es mínimo, y una cabeza plana se adentra ligeramente en el agujero de una arandela, ocupando así menos volumen que una cabeza plana.



11. De nuevo dibujamos un círculo con cola de epoxi, pero esta vez en el lado exterior de la tapa, y de nuevo le colocamos una arandela [A1] encima. Esta vez la arandela quedará atravesada por la parte roscada del tornillo.



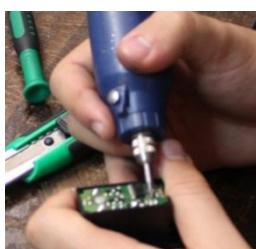
12. Finalmente se enrosca la tuerca y se aprieta bien con ayuda de una llave inglesa pequeña o alicate y un destornillador. Las arandelas hacen posible un apretado más vigoroso, ya que de no haberlas puesto la superficie de plástico de la tapa se deformaría al apretar el tornillo. Para terminar se limpia el exceso de cola procurando que ésta recubra la tuerca, pero no la parte



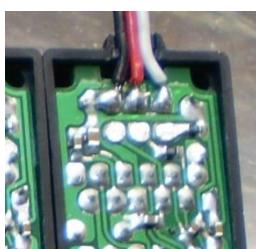
sobresaliente del tornillo, puesto que será ésta la que habrá de trabajar como eje trasero del servo.



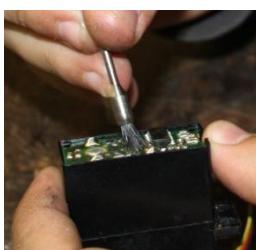
13. Una vez lista la tapa del servo, serán necesarias algunas modificaciones en su circuito interno para poder cerrarla. Dicho circuito muestra su cara trasera, de modo que el primer paso será usar unas tenazas de corte pequeñas para eliminar el sobrante de todas las patillas de los componentes electrónicos asentados al otro lado de la placa, procurando no dañar el estaño que los mantiene sujetos y en contacto.



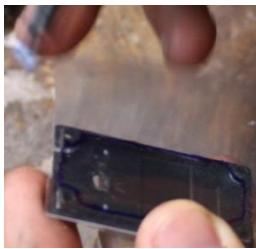
14. Acto seguido será necesario usar un pequeño taladro de maquetismo o fresolín con una fresa cónica pequeña para rebajar las tres soldaduras de estaño situadas justo debajo de las tres soldaduras que mantienen los cables sujetos a la placa. Estas tres soldaduras, que conectan el potenciómetro de realimentación a la placa, están justo debajo de la cabeza del tornillo que se ha instalado en la tapa del servo, y de no limarse nos impedirían cerrar la tapa. Esta es sin duda la operación más delicada de este proceso, ya que un movimiento en falso podría dañar las pistas de la placa, o las pequeñas resistencias SMD soldadas a la placa por este lado.



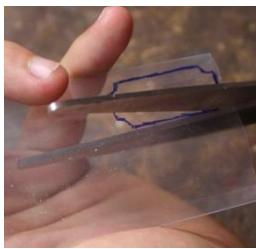
15. Una vez rebajadas lo suficiente, es importante usar un pequeño cepillo para eliminar cualquier polvo o limadura de estaño que pueda haber quedado adherido a la placa, puesto que este residuo podría llegar a producir un cortocircuito u originar un fallo de funcionamiento. Asimismo, es necesario usar un polímetro con detector de conducción sonoro para asegurarse concienzudamente de que no se ha cortado o puenteado ninguna pista ni soldadura de la placa.



16. Una vez verificada la placa, es necesario construir una lámina aislante para evitar que la cabeza del tornillo pueda puenteear las soldaduras limadas al cerrar la tapa. Esta lámina está hecha de plástico transparente y se construye dibujando el contorno interior de la tapa del servo con un marcador indeleble para poder recortarlo posteriormente. La intención de que el plástico tenga la misma forma y dimensiones que el interior de la tapa del servo es evitar que ésta se mueva en su interior.



17. Una vez recortada la lámina aislante basta con una gota de cola para fijarla a la cabeza del tornillo en su posición correcta.





18. Finalmente, se cierra la tapa del servo, se aprietan con moderación los cuatro tornillos y se deja secar durante 24h. Después del secado, convendrá comprobar su buen funcionamiento conectando el servo a la alimentación pertinente e inyectándole una señal PWM dentro de sus parámetros admitidos en su terminal de consigna.

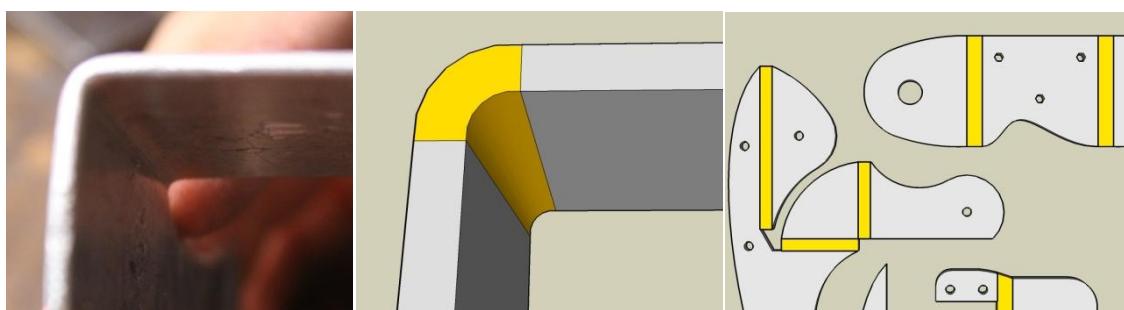


## 4.2 Procedimientos de fabricación del esqueleto.



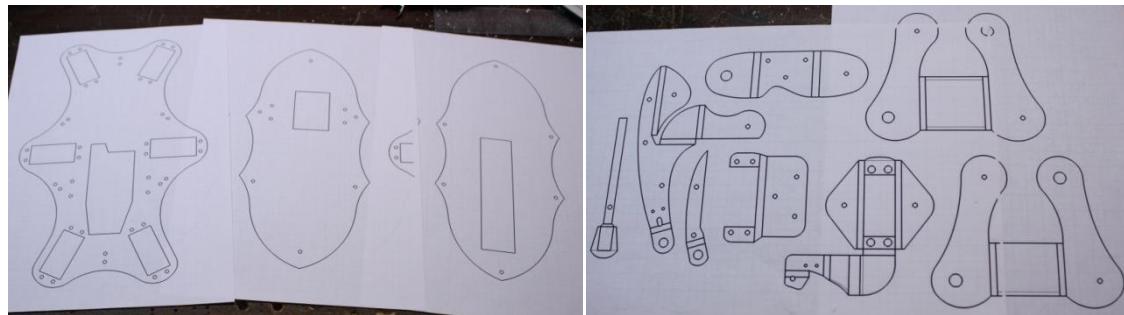
Una vez terminado el diseño detallado por ordenador, comienza la fabricación de las piezas del robot para su posterior ensamblado y montaje. Esta memoria describe por separado ambos procesos con la intención de hacer la explicación más inteligible, sin embargo lo cierto es que durante el proceso real de construcción, fabricación y ensamblado se llevaron inevitablemente a cabo de forma intercalada. También es interesante mencionar que todas las herramientas y materiales que conlleva esta etapa son típicas de un taller casero, y dado que todo el proceso de fabricación y montaje se lleva a cabo completamente a mano y de forma artesanal, el éxito de dicho proceso depende en gran medida de la habilidad de su constructor.

La fabricación del esqueleto es un proceso que comienza con el modelo informático tridimensional del robot. El software Google SketchUp Pro 6 nos permite modificar por separado cada pieza del robot y alinear sus caras hasta convertirlas en una superficie plana con un contorno definido. Para hacer esto, no obstante, es necesario conocer el material con el que se va a construir cada pieza, y haber analizado antes su maleabilidad y sus posibilidades de mecanización. En nuestro caso, el robot está fabricado en chapa de aluminio laminado normal (más rígido que el aluminio virgen, pero menos que el duraluminio) de 1, 2 y 2'5mm de espesor. Tras realizar algunas pruebas de doblado, se observa que el aluminio no se curva en perfecto ángulo recto, sino que el eje de doblado presenta un margen de área curva que, de no tenerse en cuenta deformaría las dimensiones de las piezas y evitaría que encajaran. Este margen se cuantifica en 2mm, 4mm y 5mm para los espesores mencionados. Esto significa que el diseño informático del robot ha de prever estos márgenes, y que han de quedar bien reflejados en el despliegue de las piezas.

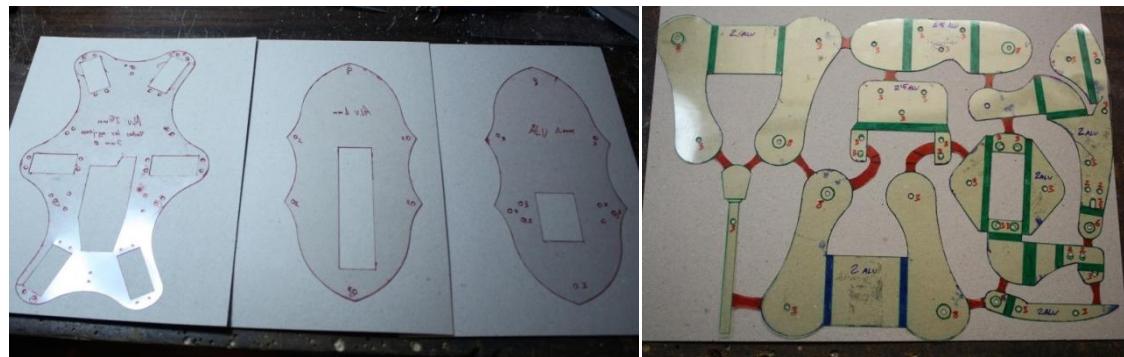


Una vez desplegadas las piezas se exportan a otro programa, LayOut, incluido en el paquete de Google SketchUp Pro 6, y que constituye una útil herramienta para gestionar la presentación e impresión de los modelos diseñados en SketchUp. Desde LayOut se imprimen varios planos a escala 1:1 donde se disponen algunas de las piezas desplegadas,

distribuyéndolas en el espacio de forma concienzuda. Ha de tenerse en cuenta que la distribución de las piezas se trasladará a la chapa de aluminio, donde habrá que cortarlas usando una simple sierra de arco, de manera que si las piezas no están bien distribuidas en el espacio disponible el proceso de corte podría entorpecerse o incluso hacerse imposible.



Una vez impreso, el plano se convierte en una plantilla calcándolo en una lámina de plástico transparente con un rotulador indeleble de punta fina. Durante el calco, la lámina de plástico y el plano deben mantenerse bien unidos para evitar que uno se mueva respecto del otro. Además, durante este proceso de calco se tendrán en cuenta ciertas áreas que harán de puente entre unas piezas y otras para mantenerlas unidas tras el recorte. De este modo se conserva en una sola plantilla la distribución de todas las piezas y no hay que volver a distribuirlas cada vez que se pasen a la chapa de aluminio. Las zonas de doblado deben quedar especialmente bien definidas y los agujeros deben quedar bien claros y centrados durante el calco. Puesto que hay agujeros de distintos tamaños, será bueno marcar el diámetro de cada uno y perforarlos usando una broca de Ø2mm antes de recortar la plantilla. También es conveniente, para evitar errores, marcar sobre cada pieza el espesor del aluminio que le corresponde.



Una vez listas las plantillas, se repasan sus bordes exteriores y los agujeros con un rotulador permanente sobre la chapa de aluminio correspondiente. Las líneas deben quedar gruesas y las áreas de doblado bien marcadas, puesto que la manipulación a la que se someterán las piezas las borrará fácilmente, de manera que tendremos que repintarlas en más de una ocasión. Una vez listo el dibujo, los agujeros se marcan usando un punzón de acero templado; de este modo ya no será necesario repintarlos constantemente. Finalmente, se recortan las piezas.



Para cortar las piezas se dispone de un banco de trabajo con dos tornillos de sujeción, y dos sierras distintas: una para cortes rectos y otra para los curvos. Los dos tornillos del banco servirán para agarrar por dos puntos las placas de aluminio cuando aún son aparatosamente grandes, ya que se venden en un formato de 60x60 cm. En el trabajo con las sierras habrá que tener en cuenta ciertas directrices: en primer lugar, el corte es impreciso y erosiona la zona adyacente al mismo, de forma que se define un margen de 1mm aproximadamente que separará la línea de la zona de corte para salvaguardarla. En segundo lugar, el ancho del corte es diferente para cada sierra: la sierra de cortes rectos elimina una franja de aproximadamente 1mm de ancho, mientras que la de cortes curvos elimina una franja de casi 3mm. Esto repercute directamente en la distancia que hay que mantener entre el corte y la línea. Por último es muy importante colocar correctamente la pieza en el tornillo de sujeción cuando se realiza un corte: la zona de incidencia de la sierra, sobretodo de la sierra de corte recto, debe estar lo más cerca posible del tornillo, porque de lo contrario el empuje que se le da a la sierra podría doblar el aluminio y volver la pieza inservible si la sierra se atasca, cosa que ocurre con frecuencia.





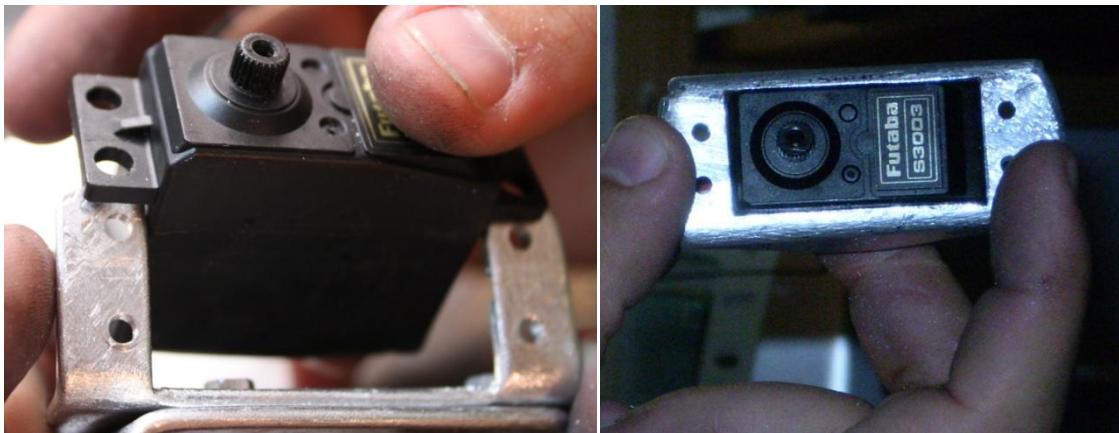
Una vez cortada, la pieza se sujeta de nuevo al tornillo para limar sus bordes hasta dejar una silueta limpia y definida. Se lima con una lima para metal, gruesa y plano-convexa, así podremos usar la misma lima para las zonas cóncavas, las rectas y las convexas. La lima debe eliminar tanto material como sea necesario hasta que la línea de contorno de la pieza prácticamente desaparezca. De nuevo hay que tener la precaución de limar sólo en la zona del borde de la pieza cercano al tornillo de sujeción, porque de lo contrario la fuerza de empuje de la lima podría doblar la pieza.

Una vez hecho el contorno, y usando una lima igual, pero más fina, se redondean los cantos. Para hacerlo primero se realiza un bisel en todo el borde de la pieza y a ambos lados. Después el bisel se redondea usando la lima con suavidad hasta que se difumina, quedando un borde mate y redondeado. Algunos recovecos serán de difícil acceso tanto en el tornillo como sobre la mesa de trabajo, o exigirán limas más pequeñas. Es por eso, y por la naturaleza de este proceso, que el trabajo de limado es lento, pesado y trabajoso.



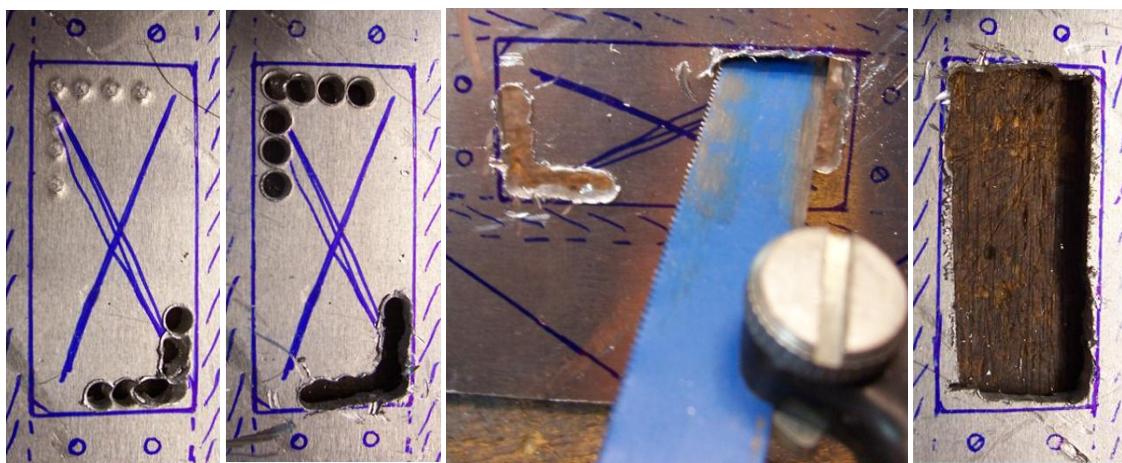
Con esto concluye el corte y el limado de los contornos exteriores, pero algunas piezas ofrecen contornos interiores que deberán ser tratados de forma algo diferente. Concretamente, aquellas piezas que abrazan y sujetan un servo muestran un contorno rectangular que debe ser vaciado, limado y redondeado sin acceso desde el exterior. Estas piezas son el Tórax Base, P00, y las 6 Partes fijas de la Tibia, P06. El Conector Cadera-Fémur, P04, también abraza un servo, pero no lo hace por los cuatro costados, sino por sólo tres. Esto significa que se puede recortar con la sierra de forma normal, accediendo por el lado abierto del rectángulo. Además, en el caso de las piezas P04 y P06, el vaciado de la cavidad del servo debe realizarse siempre después de haber doblado la pieza. El motivo tiene que ver con la robustez de la misma, puesto que en estos dos casos el doblez de la pieza pasa muy cerca de los lados más largos del hueco de los servos, de forma que una vez

vaciado el hueco, el servo queda unido a la pieza sólo por unas estrechas franjas de metal. Estas franjas son suficientemente anchas para sujetar la estructura con fuerza, pero no para doblarse de forma correcta. De este modo, si se vacía el hueco antes de doblar la pieza, al doblarse ésta se deforma, se desalinea y las franjas se agrietan, debilitando aún más la unión entre el servo y el resto de la estructura. La solución, vaciar el hueco después de doblar la pieza, ofrece un resultado satisfactorio a un alto precio, ya que el proceso de vaciado, ya de por sí engorroso, se vuelve además incómodo y complicado.



El proceso de vaciado del hueco de los servos comienza usando el punzón de acero para marcar siete puntos equidistantes formando un ángulo recto en dos de las esquinas diagonalmente opuestas del rectángulo. Sobre cada una de estas marcas se perforará con una broca de Ø3mm, de forma que deben estar suficientemente alejadas del contorno del hueco como para que la broca no sobrepase la línea de contorno, y suficientemente juntas como para que cada agujero se solape ligeramente con sus dos agujeros adyacentes. Una vez perforados, se usa una fresa de Ø3mm para eliminar el material que separa los agujeros entre sí. El resultado son dos ranuras yuxtapuestas en un vértice y formando un ángulo recto, de modo que cada una de las ranuras es ligeramente más larga que el ancho de una sierra de metal para corte recto, como la empleada anteriormente. Esto significa que, con la ayuda de un mango especial que la agarra sólo por un lado, podremos introducir la sierra por las ranuras y cerrar en ambas direcciones hasta recortar el hueco en el que deberá entrar el servo. Una vez hecho esto, de nuevo habrá que limar los contornos del hueco hasta dejarlos bien definidos para luego redondear los bordes, comprobando ésta vez que el agujero es suficiente para que el servo encaje. Cabe decir también que es conveniente perforar los agujeros de sujeción del servo después de haber terminado con el vaciado del hueco, pues de lo contrario el ajuste de la pieza en el tornillo de sujeción o el doblado, si es que se perforan estos agujeros antes de doblar, podrían combar la superficie plana donde se asentarán los tornillos que sujetarán el servo.





Una vez cortadas y limadas las piezas, y solucionado el problema de los contornos interiores, el siguiente paso será perforar las piezas según las especificaciones del diseño. No obstante, no todas las marcas hechas con el punzón deben perforarse por ahora, ya que algunos agujeros han de albergar elementos mecánicos que deben ajustar entre sí a la perfección, y su localización exacta se calculará sobre las piezas definitivas, una vez dobladas y montadas. El orden en el que deben intercalarse la perforación de cada agujero y los sucesivos procesos de la fabricación de cada pieza se explicitan en el apartado de montaje. Los agujeros se han realizado usando un taladro de columna a bajas revoluciones. Esto es preferible al taladrar metal, porque conserva la integridad de la punta de las brocas, que a altas revoluciones se calientan, y por ende se ablandan y se desafilan más. El problema es que perforar aluminio a bajas revoluciones provoca grandes extrusiones del metal en la cara opuesta a la penetración de la broca, de manera que habrá que limar y posteriormente repasar cada agujero.

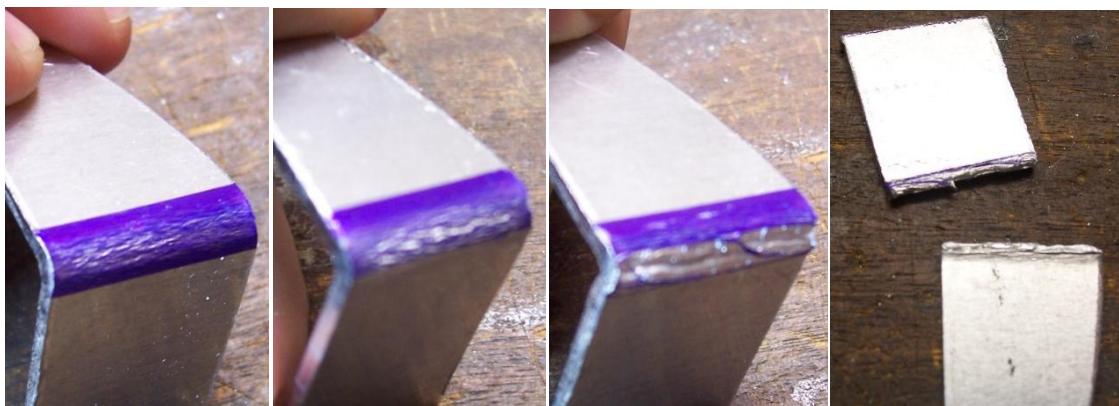




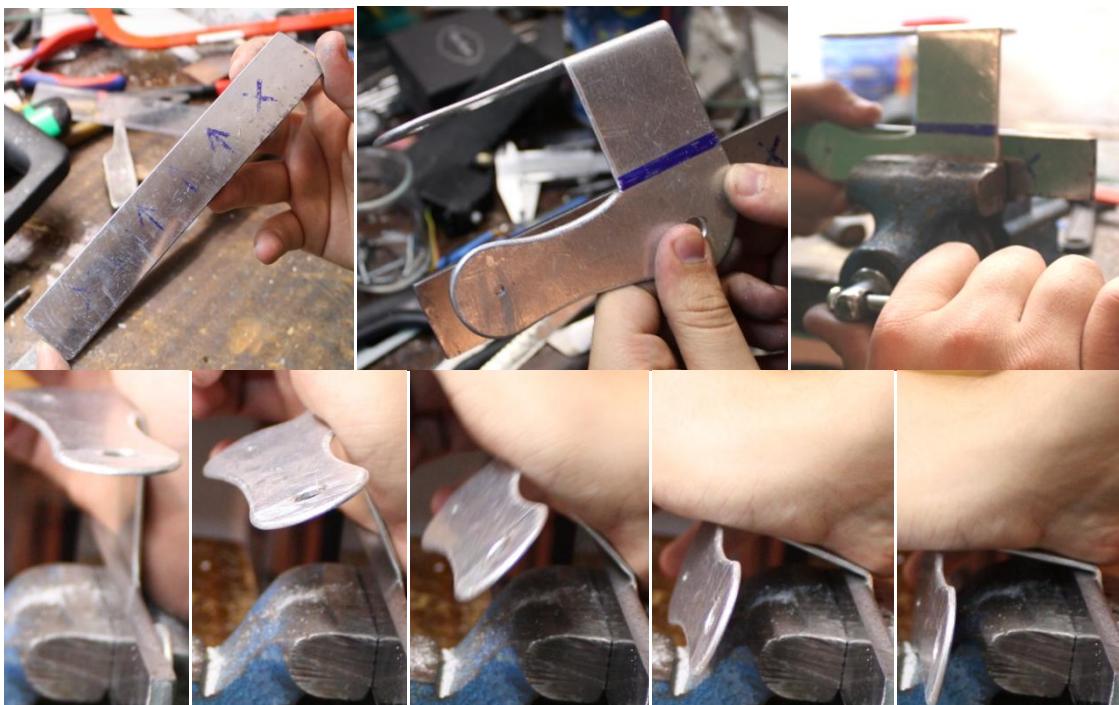
Cabe decir también que algunos agujeros grandes ([O60], [O76], [O88 – O89], [O105] y [O109]) no se pueden hacer directamente, sino que hay que usar brocas progresivamente más grandes, pues se corre un gran riesgo de que la posición del agujero varíe. También en estos casos es recomendable agujerear después de doblar la pieza, tal y como explica el proceso de montaje, pues el agujero se encuentra peligrosamente cerca de una zona de doblez. El orden de diámetro de las brocas que se han empleado sucesivamente en los agujeros mayores de Ø3mm ha sido Ø3mm, Ø6mm y Ø8mm. Por otro lado, otros agujeros albergan tornillos de cabeza cónica, y se hace necesario practicar un avellanado del borde del agujero (imagen de la izquierda) con una broca de Ø6mm para reducir el relieve de las mismas ([O34 – O39], [O75], [O79 – O81], [O86 - O87], [O106 – O107] y [O110 – O111]). Por último, algunos agujeros toman la forma de pequeñas ranuras ([O108]) u orificios alargados ([O100]). Del mismo modo que se ha visto anteriormente, estas ranuras se forman tras eliminar con una fresa el material que separa a varios agujeros adyacentes, y los orificios alargados sencillamente se moldean con la fresa a partir de un orificio inicial más pequeño.



Una vez terminada la primera tanda de agujeros se procede a doblar las piezas. El aluminio laminado es un material muy poco elástico, de manera que prácticamente no recupera su forma inicial después de someterlo a una deformación. Esto por un lado es algo positivo, ya que facilita el proceso de doblado, pero por el otro conlleva un riesgo. Cuando el aluminio se dobla, el lado interno del doblez se comprime y se densifica, mientras que el lado externo se expande y se ablanda. Si sólo se dobla una vez, la pieza es resistente y fiable, pero si este doblez se desdobra en sentido contrario, la parte comprimida del doblez no se expande de forma plástica, sino que se agrieta y se degrada. De este modo, si un doblez se realiza en sentido contrario por error, al desdoblarse quedará mucho más frágil, y si se desdobra una tercera vez puede llegar a partirse. Por eso todos los dobleces han de salir bien el en primer intento, pues no hay una segunda oportunidad: si sale mal, la pieza se desecha.



Para que los dobleces definan una línea perfecta no es recomendable fiarse de los gastados bordes del tornillo de sujeción, es mejor usar una barra de acero de borde recto y definido. Esta barra siempre se coloca entre la pieza y el tornillo en el sentido de doblado, aproximadamente 1mm o 1,5mm por encima del borde marcado del doblez para los aluminios de 2mm y 2,5mm, respectivamente. Esto se debe a que el aluminio se deformará ligeramente por el lado en que lo tenemos agarrado al tornillo en cuanto lo reafirmemos en su posición final, con ayuda de una escuadra y fuera del tornillo.



Además, en algunos dobleces especialmente anchos el aluminio resulta ser tan resistente que es imposible doblarlo por simple empuje perpendicular. En estos casos es necesario usar un martillo y un tas, que no es otra cosa que una pieza de metal no demasiado duro (cobre, en nuestro caso) que se usa sobre la pieza de aluminio para dirigir el efecto de los golpes, y para que éste no reciba directamente los impactos del martillo, que le dejarían marca.



Tal y como se ha mencionado anteriormente, algunos orificios serán mejor realizarlos después de alguna de las etapas de doblado de cada pieza, según detalla el apartado de montaje. De entre estos merecen ser mencionados algunos de los orificios grandes ([O60], [O76] y [O88 – O89]), que servirán para albergar las piezas de plástico [E1] que se



acoplarán al eje motriz de los servos. Los cuatro son orificios de Ø8mm alojados en piezas que toman la forma de dos planos enfrentados y separados una cierta distancia, cosa que permite que dichas piezas puedan sujetar los ejes trasero y motriz de un servo mediante un orificio de Ø3mm para el eje trasero en uno de los planos de la pieza, y otro de Ø8mm, el que nos ocupa, que alberga el soporte del eje motriz en el plano opuesto. De este modo, el orificio de Ø8mm debe quedar perfectamente alineado con el de Ø3mm, independientemente de lo bien o mal que hayan quedado los dobleces de la pieza en cuestión, para evitar que el eje del servo quede torcido, y por tanto su plano de rotación desviado. Así pues, aunque la pieza no haya quedado perfectamente doblada, estos orificios sí deben quedar perfectamente alineados.

Para conseguirlo, primero se perfora con una broca de Ø3mm uno de cada par de agujeros; es indiferente si se trata del agujero del eje motriz o del eje trasero. A continuación se usa un tornillo de cabeza plana de Ø3mm x 35mm, [T5], para pasarlo por el orificio y apretarlo con una tuerca. La presión de la tuerca hará que el tornillo marque la línea perfectamente perpendicular al plano en el que se halla el agujero, y su extremo roscado, que quedará muy cerca del plano opuesto, marcará la posición exacta del orificio que debe practicarse en dicho plano. Así los agujeros del eje trasero y motriz quedarán perfectamente alineados, y el servo podrá girar exactamente en el plano que le corresponde. Este proceso ha de realizarse concienzudamente, ya que la desviación de los ejes de los servos [S2] y [S3] puede provocar fallos graves de funcionamiento, como por ejemplo la activación errónea de los sensores de obstáculo cuando la pata presiona contra el suelo.



Además, para facilitar la inserción en sentido oblicuo del servo en el espacio interior de la pieza, en más de una ocasión se hace necesario practicar un avellanado interno del orificio de sujeción del eje trasero ([O75] y [O86 – O87]), que realizaremos, igual que hicimos antes, con una broca de Ø6mm, pero accediendo esta vez desde el agujero de Ø8mm que hicimos en la parte opuesta de la pieza.



Sobre estos orificios cabe decir también que, aunque servirán para albergar soportes [E1] idénticos, en el caso del agujero [O88] el soporte podrá quedar ajustado en el orificio, mientras que en el de los agujeros [O76] y [O89] el soporte deberá quedar holgado. Esto es un requisito derivado del proceso de montaje, ya que el agujero [O88] está situado en una de las dos aletas de la pieza P05, y la forma de dichas aletas les permite deformarse separándose entre sí lo suficiente como para poder encajar el servo en su interior y encerrarlo posteriormente al restablecerse la forma original de la pieza por pura elasticidad.

del metal. No obstante, ha de tenerse en cuenta que el aluminio es muy poco elástico, por lo que si el metal se somete a demasiado estrés, la pieza puede deformarse de forma irreversible. Con todo esto, el soporte de plástico [E1] puede instalarse y fijarse en este orificio antes de instalar el servo, pues aún y con esta pieza instalada, la inserción del servo es factible gracias a las propiedades mecánicas de la pieza. Por el contrario, en el caso de los agujeros [O76] y [O89] la pieza admite poca o ninguna deformación, por lo que es necesario que estos orificios estén despejados para aprovechar hasta el último milímetro de espacio maniobrable y así poder albergar el eje motriz del servo durante su proceso de inserción en la pieza. Una vez colocado el servo, se encierran sus ejes en la pieza acoplando el soporte [E1] al eje motriz del servo y atornillándolo a la pieza. El problema es que la pieza de plástico se expande ligeramente al encajar con el eje estriado del servo, por lo que si no se ha previsto una cierta holgura en su orificio se hace muy complicado colocarla en su lugar.

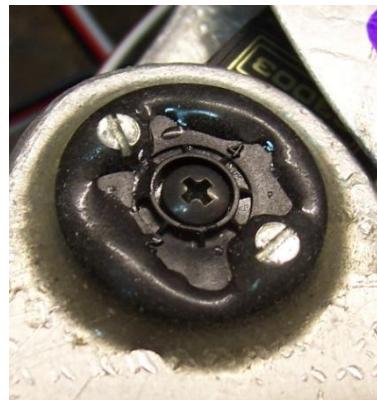
Como se ha explicado anteriormente, el proceso de inserción comienza introduciendo antes el eje trasero del servo en su orificio correspondiente, y desde una trayectoria oblicua, para luego insertar el eje motriz en el orificio opuesto. Pero esto puede suponer un problema, ya que la trayectoria de inserción del eje trasero del servo no se corresponde con la trayectoria natural del orificio. En consecuencia, y con el agravante de que el eje trasero del servo está roscado por estar construido con un tornillo, es más que probable que el eje se atasque en el agujero, o que sencillamente no entre. En ambos casos es probable que el proceso no pueda completarse, por eso puede hacerse necesario practicar un avellanado en la cara interior de dicho orificio. De este modo, el orificio acepta una trayectoria de inserción del eje trasero mucho más amplia y el proceso puede completarse. Además, y a pesar de la mayor flexibilidad de la pieza P05, puede incluso hacerse necesario practicar este avellanado en [O86] para insertar alguno de los servos [S3].





Llegados a este punto, todas las piezas del esqueleto están prácticamente fabricadas; sólo quedan por hacer algunos de los agujeros que se realizarán casi al final del proceso de montaje. Dicho proceso se explica de forma genérica más adelante, sin embargo hay algunos aspectos sobre los procedimientos de montaje que deben tratarse al detalle en este capítulo. Casi todos los puntos que se tratan a continuación se refieren al montaje de los terceros segmentos de las patas, o Tibias. La funcionalidad que se le exige a esta parte del robot eleva su complejidad estructural, y hace que su construcción y montaje supongan casi la mitad de tiempo total empleado en la construcción del robot.

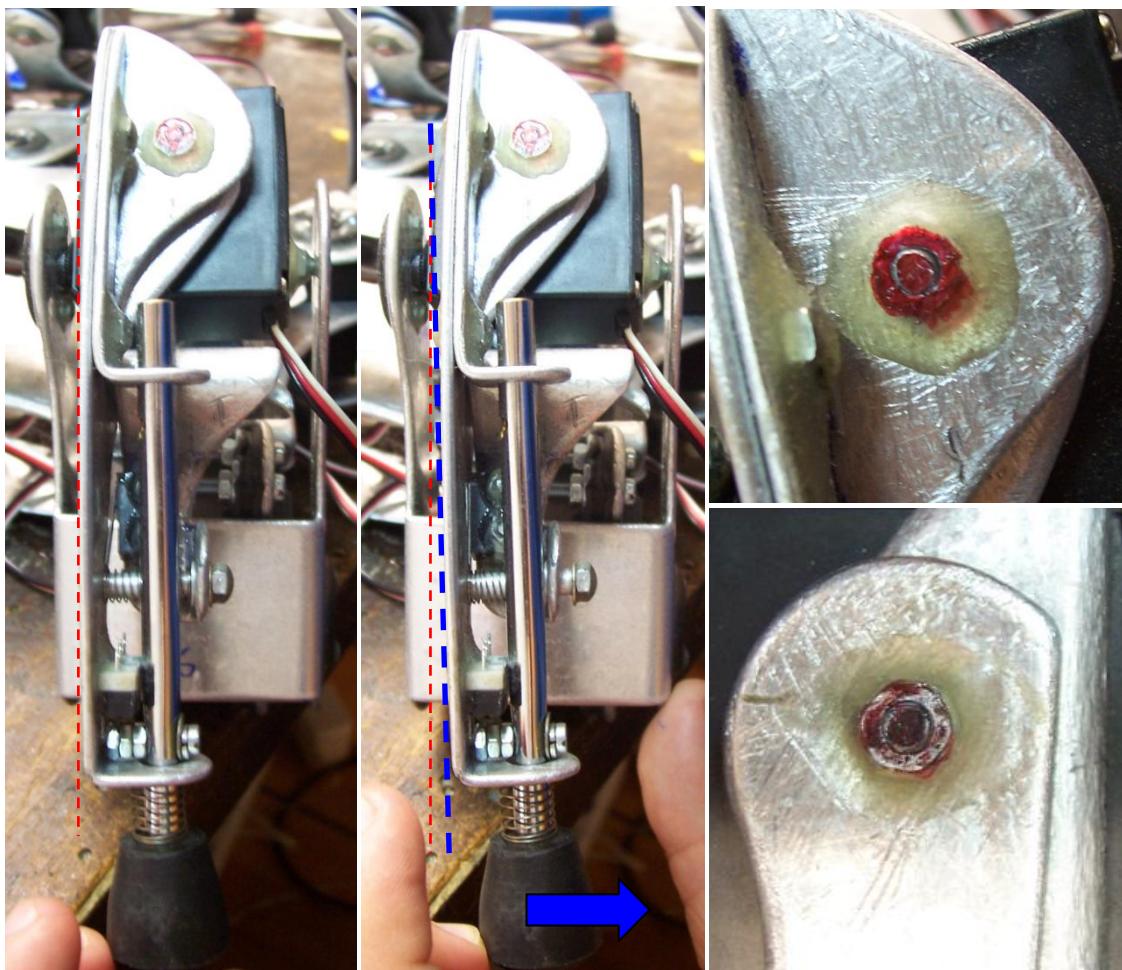
En primer lugar es importante mencionar que muchos de los tornillos y tuercas del robot están asegurados con un recubrimiento de cola de epoxi. También lo están los soportes [E1] del eje motriz de los servos y sus tornillos [T1], los bumpers [E3] y los topes de goma [E4] de los Pies. La razón del uso de este recubrimiento de cola es evitar que las vibraciones producidas por el funcionamiento continuado del robot aflojen estos tornillos o piezas, poniendo así en posible peligro la integridad de todo el conjunto.



Los que no están recubiertos de cola son aquellos tornillos que deberán aflojarse o extraerse para desmontar determinadas partes del robot, en caso de que algún sensor o motor deba ser sustituido. No obstante, en este último caso los tornillos se mantienen firmemente sujetos a pesar de no estar encolados, ya que todos ellos están atornillados sobre plástico, separadores de goma o arandelas de presión. De este modo, el sustrato que soporta la presión del tornillo es flexible, de manera que su tendencia a recuperar su forma no comprimida mantiene al tornillo siempre apretado. La cola de epoxi empleada ha sido, según las necesidades de cada etapa del montaje, Araldit Standard (izquierda) o Araldit Rápido, ambos de la marca Ceys. Los dos son pegamentos extrafuertes útiles para todo tipo de

materiales: el primero seca en 12 horas y es capaz de soportar una tracción de  $350 \text{ Kg/cm}^2$  a los 3 días, y el segundo seca en 1 hora y es capaz de soportar una tracción de  $170 \text{ Kg/cm}^2$  a las 48 horas de secado. Además, el pegamento siempre se ha aplicado sobre superficies rayadas o limadas, pero no pulidas, para aumentar su capacidad de adherencia.

Una forma especial de usar este poderoso pegamento ha sido el caso de los tornillos [T9] alojados en los orificios [O101 – O102] y [O112 – O113] de P06 y P08. Estos tornillos permiten que la parte móvil de la Tibia permanezca firmemente unida a la parte fija en el plano vertical de la pata, pudiendo a la vez pendular en el sentido de avance. De este modo, la parte móvil de la Tibia se mantiene firme cuando el robot pisa contra el suelo, pero retrocede cuando la pata choca con un obstáculo interpuesto en su trayectoria de avance. El eje de giro de esta pieza está compuesto por dos tornillos fijados con Araldit por sus dos extremos: la tuerca está fijada a la parte móvil y la cabeza a la parte fija. La tuerca, no obstante, sólo está fijada por su contorno, de manera que el Araldit une la tuerca a la parte móvil de la Tibia, pero no al tornillo, que puede girar libremente por su interior. Como esta pieza sólo debe realizar movimientos muy pequeños, el efecto de movimiento transversal de la tuerca sobre el tornillo pierde efecto, y a la práctica se comportan como un eje que gira libremente a través de un ojal.



Para lograr esto, la tuerca debe quedar firmemente fijada a la pieza móvil, pero no puede entrar ni una sola brizna de pegamento dentro de la parte roscada. Como esto es prácticamente imposible, se sigue un proceso de recuperación del giro fluido de la tuerca sobre el tornillo. Primero se fijan tuerca y tornillo a la parte móvil estando ambos bien apretados, de manera que la parte de tornillo que sobresale de la tuerca también recibe parte de Araldit. Una vez seco se corta el sobrante de tornillo con una cortadora de disco de manera que no quede nada de Araldit en la cara más superior de la tuerca. A continuación se extrae el tornillo desde el interior (en este punto, por supuesto, el servo aún no ha sido instalado) y se limpia bien la rosca de tuerca y tornillo con el filo de un cutter. Se introduce de nuevo el tornillo en la tuerca y se comprueba que el giro es fluido. Se enrosca el tornillo hasta su posición final (sin llegar a apretar, pero sin holgura) y se pega la cabeza a la parte fija de la Tibia. Al secarse, la parte móvil de la Tibia oscila suavemente sobre la parte fija, pero no hay riesgo de que tuerca ni tornillo se aflojen y se caigan.

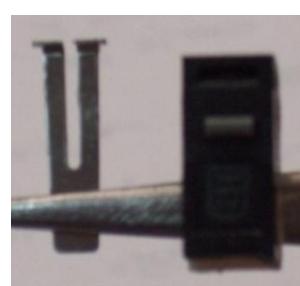
Otra de las funciones que se le ha dado al Araldit es la de fijar los topes de goma [E4] a las bolas de los extremos de los tubos de latón de los Pies. Los tacos de goma entran a presión, de manera que tienden a salirse de las bolas, por eso es necesario usar una improvisada prensa hasta que el Araldit se seca para obligar a los topes de goma a quedarse en la posición correcta. Una vez seco, es imprescindible eliminar con un cutter el sobrante de Araldit que mancha el tubo de latón en la zona cercana al tope de goma, pues ahí se instalará un ajustado muelle que no funcionará correctamente si existen manchas de Araldit que entorpezcan su movimiento de compresión.



Los sensores de contacto [E3] también han sido fijados al esqueleto con Araldit, pero en este caso el proceso ha sido mucho más delicado (y accidentado) que en el resto de casos. Para empezar, el sensor no es más que una pequeña caja de plástico fino que dispone en una de sus caras de una fina palanquilla de metal que, si se presiona, hunde un pequeñísimo



botón de plástico que abre o cierra un interruptor. La palanquilla en sí no es más que una lámina elástica de metal con una forma tal que puede sujetarse a modo de bisagra en un extremo de la caja de plástico; es el muelle situado bajo el botón de plástico lo que hace que si se deja de



aplicar fuerza sobre la palanquilla, ésta vuelva a su posición inicial. El problema es que si entra la más mínima cantidad de Araldit (o de cualquier otro pegamento de los que se han probado) en las cercanías del botón de plástico o de la bisagra de la palanquilla, el sensor deja irremisiblemente de funcionar sin posibilidad alguna de recuperarse. Además, esto sólo puede comprobarse cuando el pegamento está seco, de manera que la única forma de retirar el sensor es destruyéndolo, limpiando la zona, y comenzando de nuevo la operación con un sensor nuevo.

Pero este no es el único problema derivado de la instalación de los sensores. Según su función, estos sensores se dividen entre los que sirven de detector de obstáculos y de contacto con el suelo. Gracias a la localización más espaciada de los sensores de obstáculo, es posible fijarlos con dos pequeños tornillos [T1] de Ø2mm x 10mm y sus tuercas en su posición correcta mientras se seca el pegamento (la viscosidad del Araldit cuando aún no está seco hace imposible lograr que el sensor se mantenga estático en su posición correcta durante el secado sin los tornillos). Dichos tornillos agarran al sensor a través de dos orificios de Ø2mm que éste incorpora de fábrica. El problema aparece al instalar los sensores [E3] que detectan el contacto con el suelo, pues su cercanía al tubo del pié hace imposible colocar tuercas en los tornillos, de forma que éstos se caen. La única solución es instalar los sensores atravesados a presión por unos tornillos [T6] algo más gruesos (Ø2'5mm), de manera que cada tornillo dibuje una rosca en la superficie interior de cada orificio de Ø2mm del sensor. De este modo, el sensor hace de tuerca y puede quedar igualmente fijado mediante un tornillo sin tuerca. Al secarse el Araldit se corta el sobrante de tornillo a la altura del sensor con un disco de corte, y así ya no se obstaculiza el tubo del pié. No obstante, y a pesar de la evidencia del proceso, la instalación de estos sensores fue muy accidentada. Resulta que el orificio de sujeción del sensor está demasiado cerca del borde del mismo, de manera que lo único que cierra el círculo del orificio es una fina capa de plástico que se rompe al intentar introducir un tornillo más grueso de Ø2mm. La única solución posible fue usar una broca de Ø2mm para limar y ampliar ligeramente el diámetro de dicho orificio, de forma que el tornillo hiciera menos presión contra el orificio del sensor. Aún y así, con mucha facilidad algún sensor se deforma o se rompe y hay que volver a empezar, pero fue de esta manera como finalmente se consiguieron instalar seis sensores íntegros y funcionales.

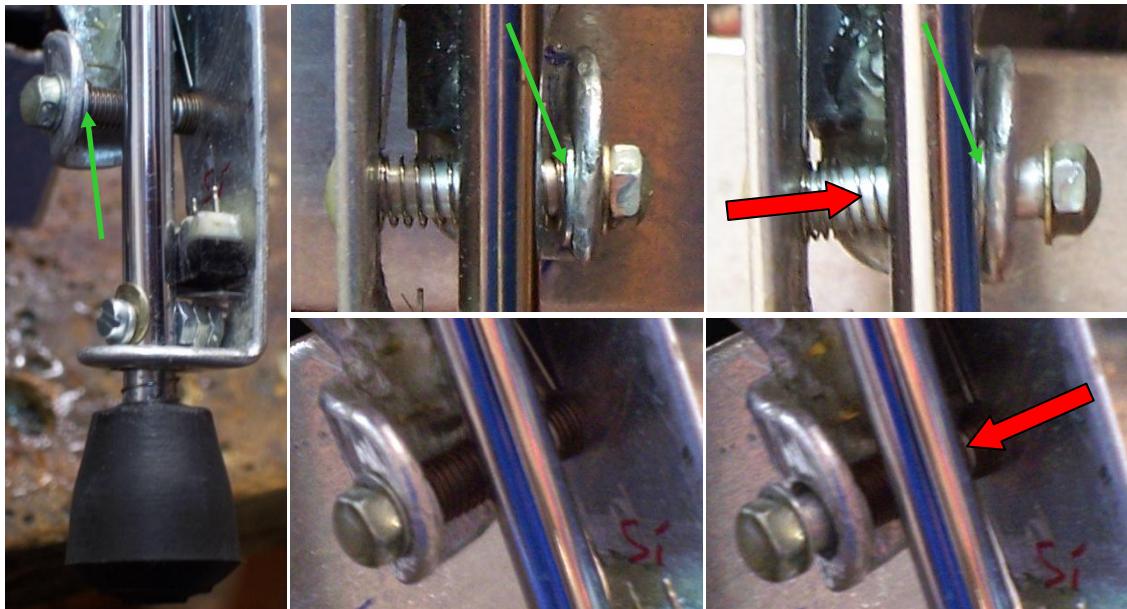


Cabe decir también que la fuerza del retorno elástico de las palanquillas de los sensores [E3] es completamente insuficiente. Usando sólo estas palanquillas, los sensores de obstáculo se activan por la simple inercia del movimiento de las patas y los tubos de los pies tiemblan y tintinean constantemente dentro de sus orificios. Por eso es necesario usar muelles que aporten cierta solidez al retorno elástico de los sensores, pero éstos deben ser suficientemente blandos como para no entorpecer la correcta detección del suelo y de los obstáculos. Para ello se usan muelles blandos de Ø6'5mm [E5] y Ø4'5mm [E6], que se mueven con libertad sobre los tubos de Ø6mm de los pies y los tubos separadores de Ø4mm [Y2] del retorno elástico añadido entre las partes fija y móvil de la Tibia. Además, estos muelles han sido probados y cortados a medida en cada ocasión, para conseguir que el muelle no aplique sobre la pieza móvil más fuerza de la necesaria. En cuanto a su instalación, en el caso de los muelles [E5] de los pies es sencilla: los tubos encajan casi perfectamente con los orificios que los sujetan, de manera que basta con deformar el extremo superior de los muelles, abriendo la espiral al final, para que sea imposible que el muelle se atasque en el orificio.



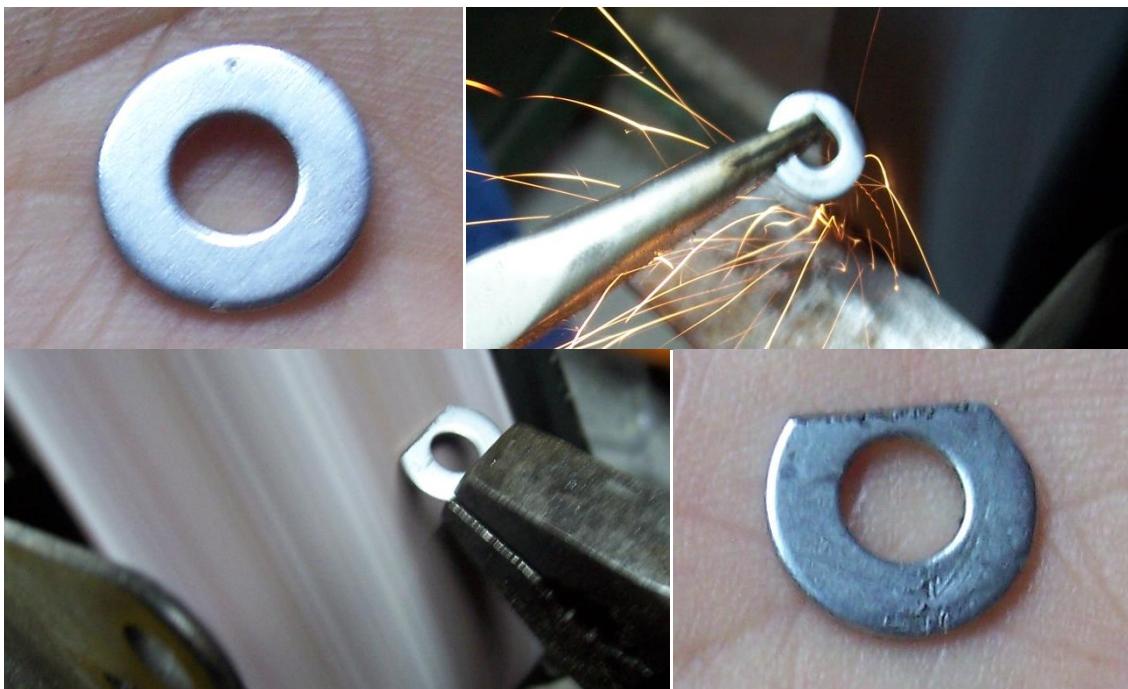
Por el contrario, la instalación de los muelles [E6] de los sensores de obstáculo es mucho más compleja. Hay un conjunto de piezas formado por un tornillo [T8] con su tuerca, un separador [Y2] y una arandela [A0] que se instalan de forma solidaria a la parte móvil de la Tibia, P08, a través del orificio [O114] (para entender la nomenclatura de la explicación que sigue es recomendable consultar el apartado correspondiente al ensamblado de las Tibias en el capítulo de montaje). Dicho conjunto atraviesa el orificio [O100] de la parte fija de la Tibia, P06, de manera que la aleta inferior de P06 circule a lo largo de [Y2], pero limitado en un extremo por el cuerpo de P08 y en el otro por la arandela [A0] que le hace de tope, cosa que permite que el movimiento pendulante de P08 quede limitado por esta estructura. Además ha de tenerse en cuenta que [O100] debe ser suficientemente ancho y alargado como para que [Y2] circule por su interior sin rozamiento alguno, pues de haberlo el mecanismo podría dejar de funcionar correctamente, pero a la vez debe ser más pequeño que el diámetro exterior de la arandela [A0], pues de lo contrario ésta dejaría de

cumplir su función como tope. El retorno elástico se hace efectivo en el momento en que se instala el muelle [E6] sobre [Y2], limitado en su extremo más cercano a P06 por una arandela [A1] cuyo orificio es ligeramente más ancho que el diámetro de [Y2], de modo que puede circular sobre este separador pero no permite que el fino muelle entre en contacto con [O100], pues de hacerlo [E6] podría meterse dentro de [O100] y el mecanismo se atascaría o dejaría de funcionar. Así pues, cuando una pata en movimiento se encuentra con un obstáculo, P08 es empujada hacia la aleta de P06, de forma que [O100] se desplaza por [Y2], arrastra a [A1] y comprime el muelle [E6], provocando la tendencia de P08 a recuperar su posición inicial.



No obstante, existe un problema en este sistema, y es que la arandela [A1] (señalada en verde en las imágenes sobre estas líneas) tiene un diámetro exterior tal que interfiere con la limitada distancia entre [Y2] y la cara de P06 paralela al mismo. Para solucionar esto se usa una esmeriladora para limar un cierto arco del perímetro de [A1] hasta dibujarle un lado recto casi a la altura de su diámetro interior. Este es un proceso muy delicado que requiere mecanizar una pieza muy pequeña, sujetando la arandela con unos alicates y con buen pulso, pues el borde giratorio de la esmeriladora es algo irregular, y el más mínimo exceso de presión entre la arandela y dicho borde giratorio puede provocar un impacto que doblará la arandela, haciéndola inservible. Una vez conseguido, se elimina el exceso de metal del borde del nuevo lado de la arandela apoyándolo ladeado sobre una lijadora de banda, que en nuestro caso comparte el motor con la esmeriladora. Hecho esto la arandela se instala en el robot, atravesada por el tubo separador y con su lado plano colocado entre [Y2] y la cara mencionada de P06, de forma que puede moverse con libertad.





Otro procedimiento que debe detallarse en este apartado es el de las numerosas piezas del robot que se cortan a medida usando una cortadora de disco. Para empezar, prácticamente todos los tornillos del robot se cortan a medida una vez apretados para eliminar el sobrante de tornillo que, en muchos casos, puede llegar obstaculizar la instalación o el movimiento de otras piezas. Dado que estos tornillos son de acero, para cortarlos se usa un taladro de mano de alta velocidad al que se le ha puesto un disco de corte en el cabezal. La misma herramienta se usa también en otro proceso que no tiene que ver con el largo de los tornillos: el vaciado de los contornos interiores de las tapas superior e inferior del cuerpo del robot. Estas piezas, P01 y P02, están hechas de aluminio de 1mm, que puede cortarse fácilmente con la cortadora de disco en un procedimiento mucho más rápido y sencillo que el que se sigue para vaciar los contornos interiores de las piezas de P00 y P06, de 2'5mm y 2mm de espesor respectivamente. En este último caso no se usa la cortadora de disco porque el aluminio es demasiado grueso, y su corte conlleva demasiado desgaste del disco y demasiado tiempo continuado de uso de esta herramienta, que está diseñada para usos cortos ya que se calienta con rapidez, pudiendo llegar a quemarse si se usa continuadamente durante demasiado rato.



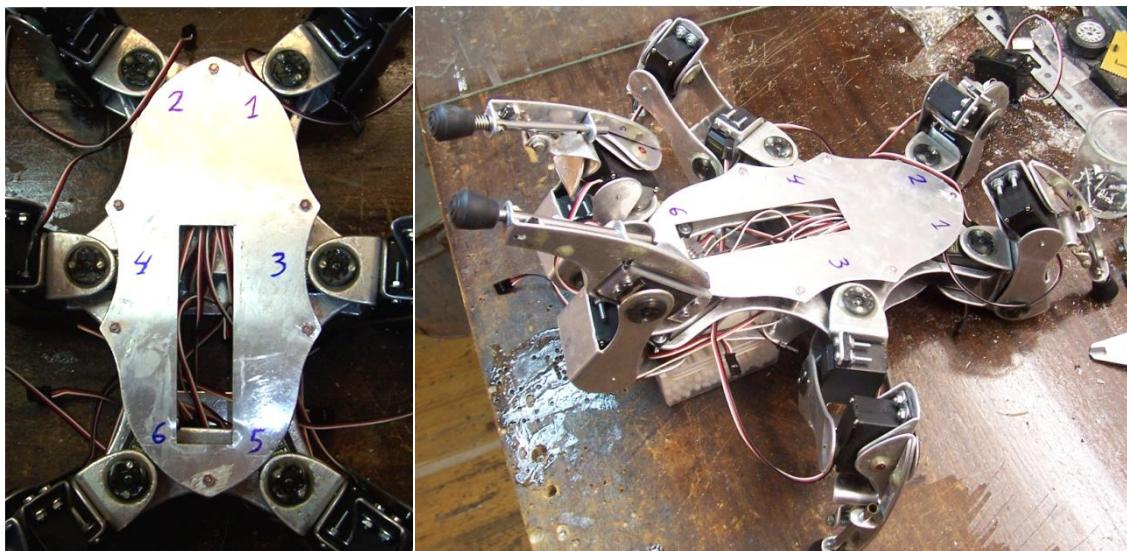
También se cortan con esta herramienta los separadores de tubo de aluminio de Ø4mm [Y0], [Y1] e [Y2]. Para cortarlos a la medida exacta, primero se calcula su longitud sobre su ubicación real con un pié de rey, y se le suma un milímetro, que se perderá entre el limado de los bordes y la compresión del tubo al apretarse el tornillo que lo sujetó. Acto seguido, se usa el mismo pié de rey para trasladar la medida al tubo con una marca de rotulador, y se corta el tubo procurando dejar la marca intacta. La alternativa al disco de corte es la sierra de arco, pero no es recomendable, ya que se hace necesario agarrar el tubo con el tornillo de sujeción del banco de trabajo, pudiendo deformarlo hasta estropearlo.



Con todo esto, ya es posible construir el robot. No obstante, como cada pieza es ligeramente diferente a las demás, aunque sean del mismo tipo, se hace necesario probar cada pieza con el soporte en el que ha de encajar hasta encontrar las parejas de piezas que encajen mejor. Incluso algunas patas encajarán en determinadas zonas del cuerpo mejor que otras, por eso se hace necesario numerar todas las piezas antes de su montaje definitivo. Además, una vez construido el robot habrá que retocar ciertos puntos críticos para conseguir que todas las piezas se muevan con suavidad. Estos retoques se realizan con una fresa cónica colocada en el cabezal del taladro de maquetismo, y limando las zonas críticas, ya sean de aluminio o excesos de Araldit. Finalmente las patas del robot consiguen la movilidad para la que se diseñaron.



Cabe decir también que se siguen ciertos criterios relacionados con la posición de las patas: aunque todas las patas deben funcionar correctamente, las delanteras 1 y 2, serán las más exigentes en cuanto a movilidad vertical y detección de suelo y obstáculos, por lo que se colocará en estas posiciones a las patas que ofrezcan mayor rango de movimiento vertical y mayor sensibilidad sensorial. Por otro lado, las patas 5 y 6 tendrán menores exigencias en cuanto a la detección de obstáculos, pero han de ser las más robustas, ya que el robot tendrá que levantar la cabeza a menudo o sostenerse a cuatro patas, incrementando el peso que éstas habrán de soportar, por lo que se escogerán aquí las patas robustas y duras en cuanto al retorno elástico de los sensores de obstáculo. Las dos patas restantes se colocarán en las posiciones 3 y 4, que tienen unos requerimientos medios.



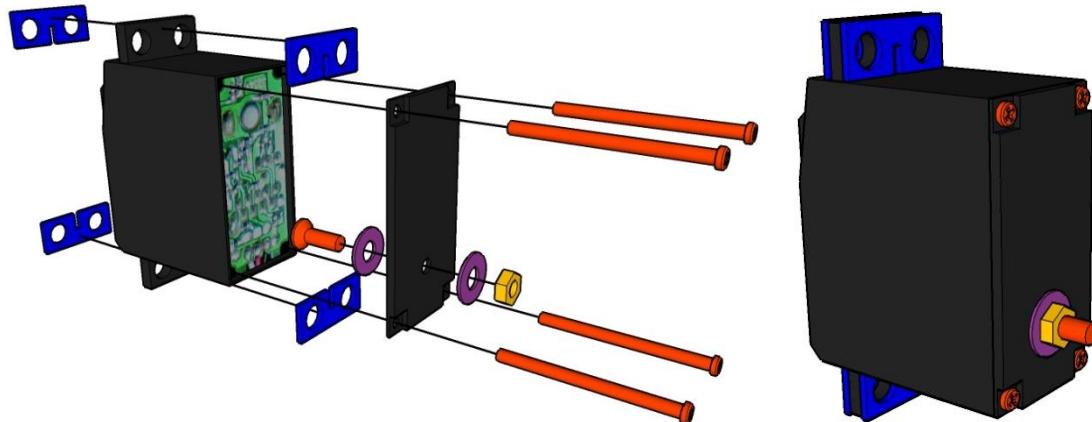
Por último, una vez montado y comprobado el robot se sueldan los terminales de los sensores usando cables blandos cortados a la medida necesaria, y agrupados, junto con los cables de los servos, en fundas de goma termo-retráctil que se fijan al esqueleto con bridas de plástico pequeñas.



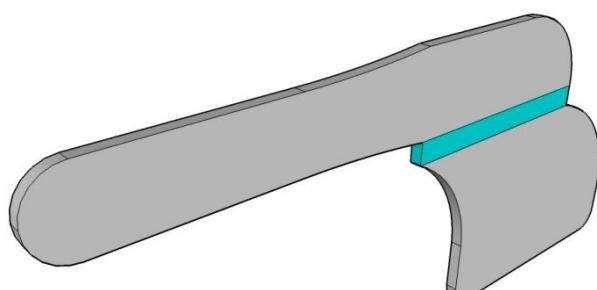
## 4.3 Proceso de montaje

Este apartado pretende ser una guía visual que ayude a entender cómo se ensamblan todos los componentes mecánicos mencionados en el despiece del robot, y qué procesos de fabricación de los explicados en el apartado anterior se siguen para dar forma a cada una de las piezas.

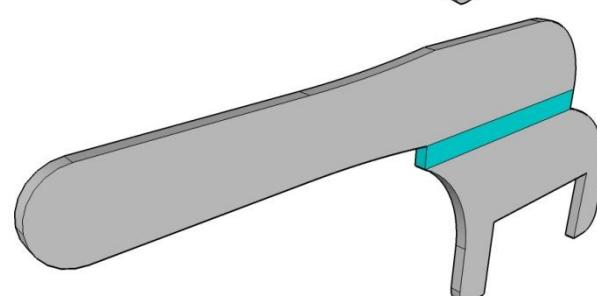
- **Modificación de los Servos:** Antes de empezar todo el proceso de montaje hay que dejar lista la instalación de los ejes traseros de los servos, pues éstos no se pueden instalar cuando el servo ya esté fijado en su lugar. Se usa un tornillo [T7], con su tuerca, dos arandelas [A1] y 4 protectores de goma [E0] para cada servo. Los cuatro tornillos internos del servo no se contabilizan en el despiece.



- **Construcción del Cuello:** El Cuello es una pértiga motorizada que se ensambla al Tórax i que permite levantar y bajar la cabeza un total de 110°. Está formado por un servo y la pieza P10.
  - **P10 – Cuello (aluminio 2'5mm):**

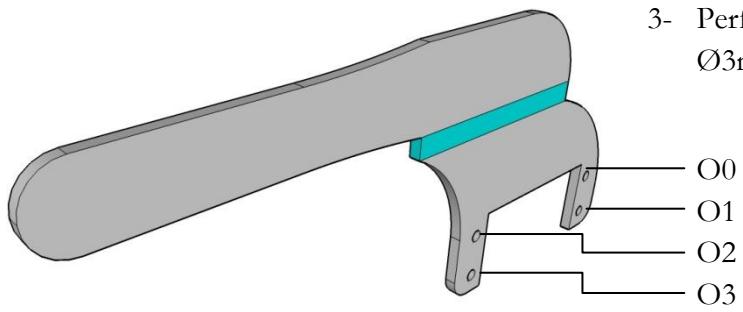


1- Corte de la preforma.

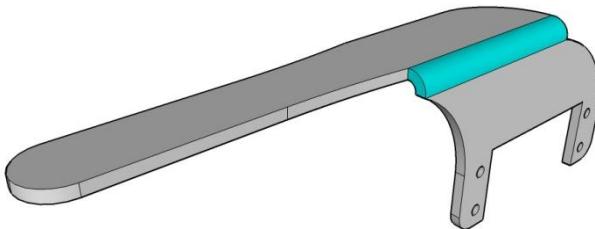


2- Corte de los contornos interiores

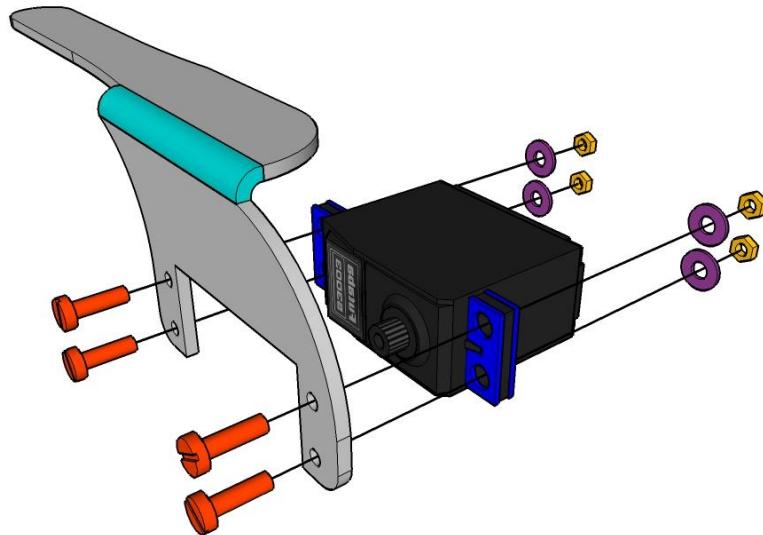
3- Perforación de los orificios de Ø3mm:



4- Doblado de la pieza.

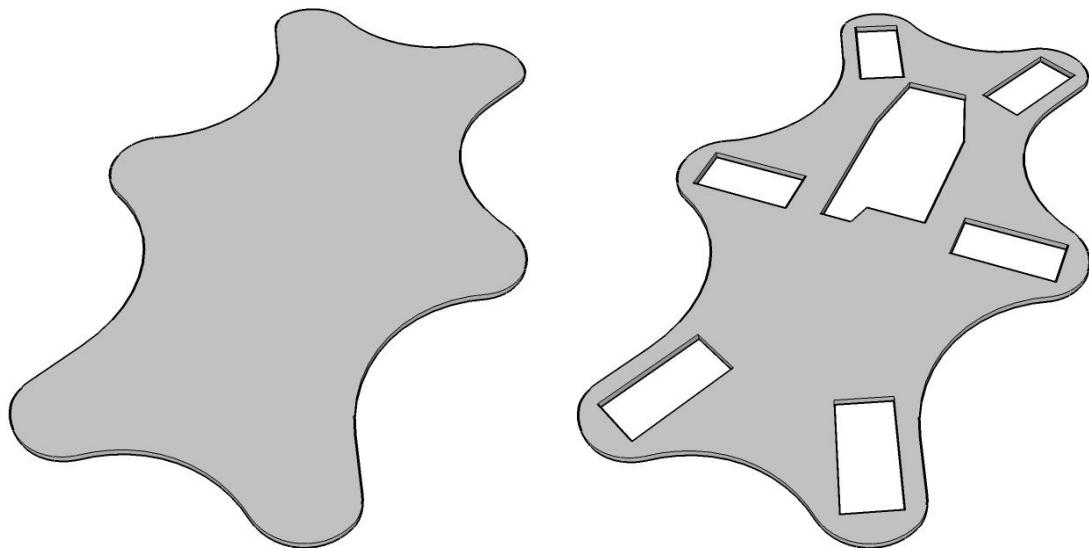


- **Montaje del Cuello:** Se hacen pasar 4 tornillos [T4] a través de los orificios [O0 – O3] de la pieza P10, del servo [S0] y de 4 arandelas [A1], para acabar apretándose en sus tuercas.



- **Construcción del Tórax:** El Tórax es el módulo central del robot; el resto de sus partes (las seis Patas y el Cuello + Cabeza), además de todos los módulos electrónicos, van unidas a él. Está formado por cinco piezas, cuyo proceso de fabricación se explica a continuación:

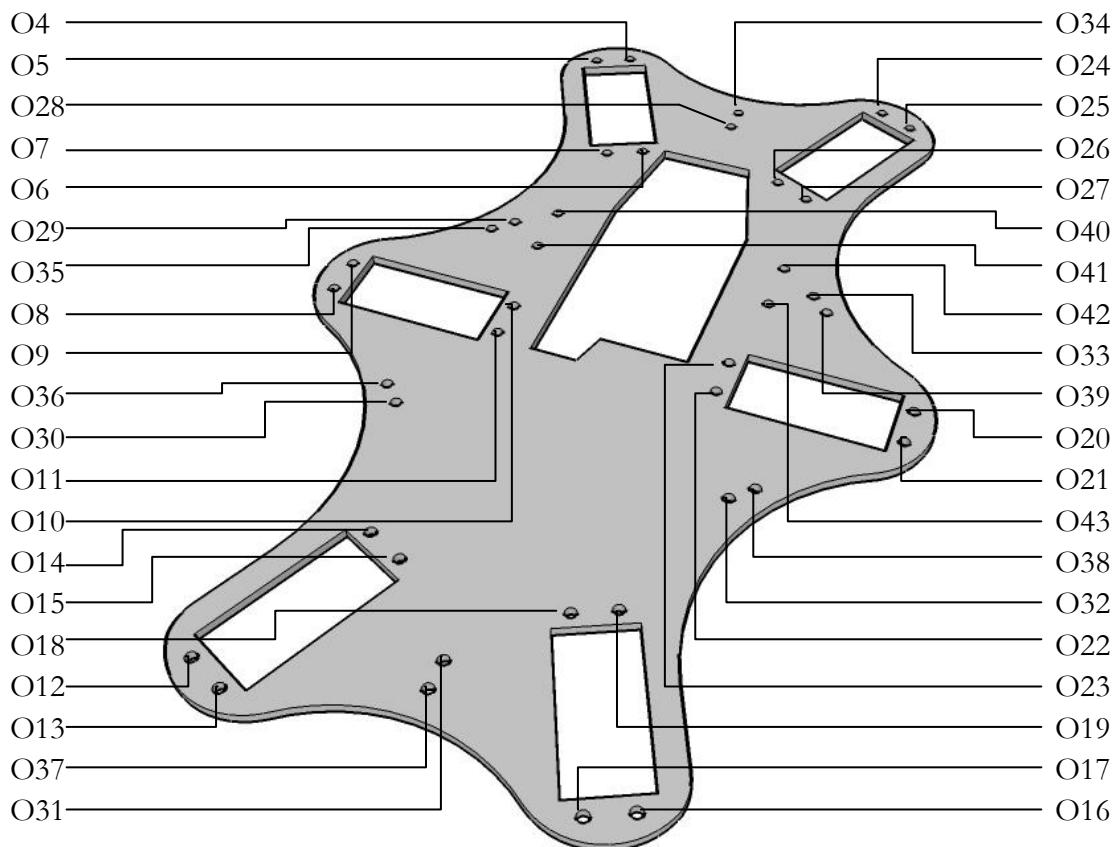
- P00 – Tórax Base (aluminio 2'5mm):

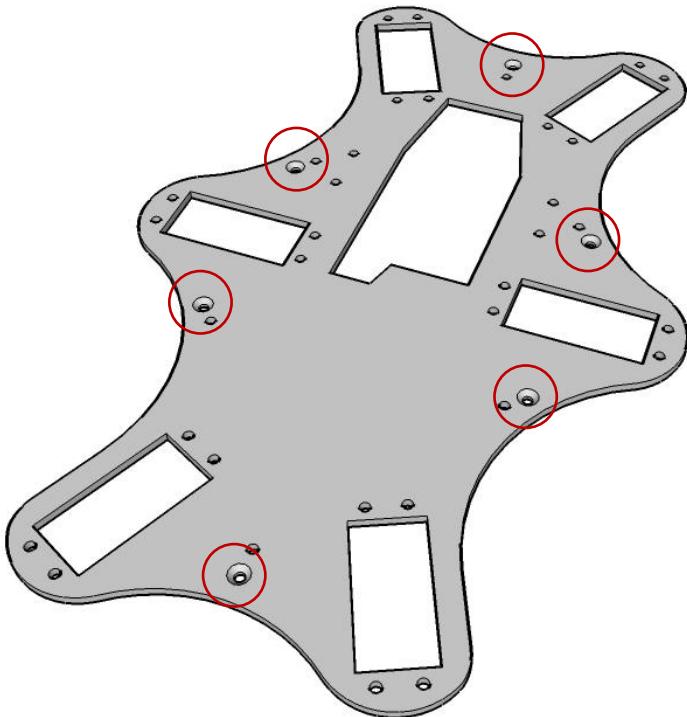


1- Corte de la preforma.

2- Corte de los contornos interiores.

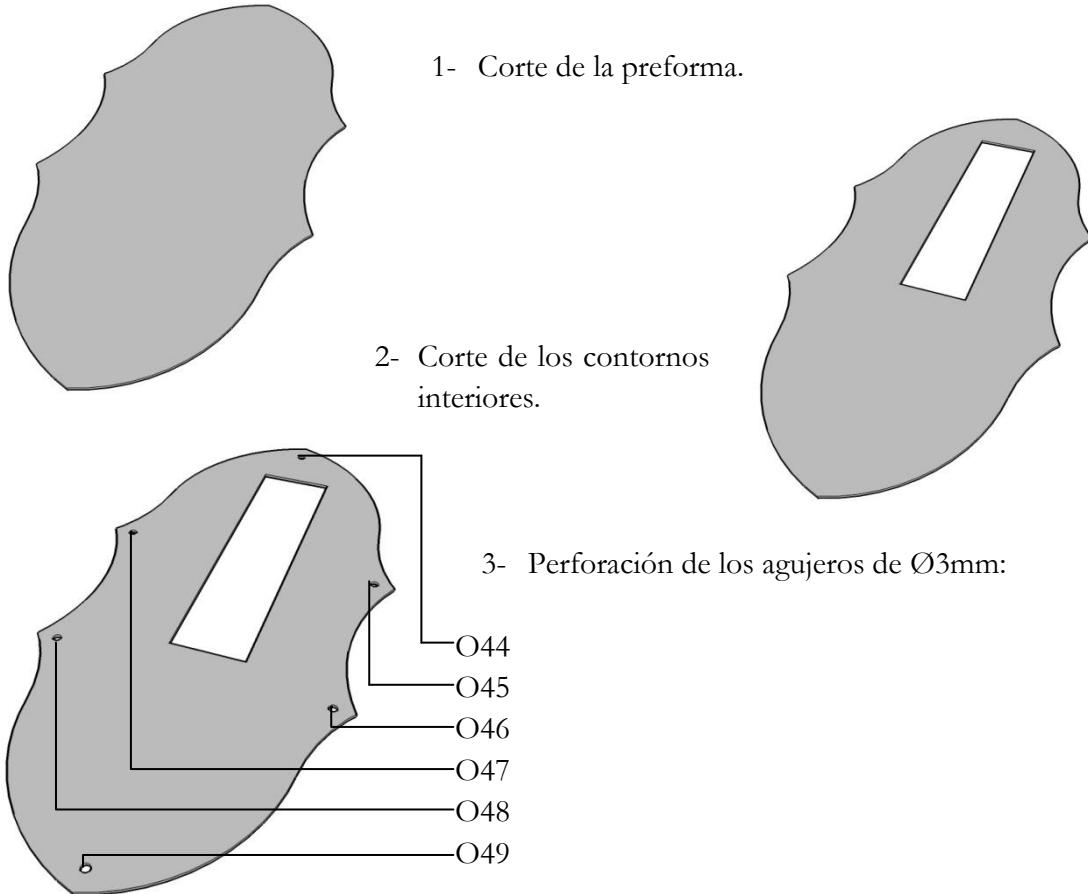
- 3- Perforación de los orificios de Ø3mm [O4 – O43]:



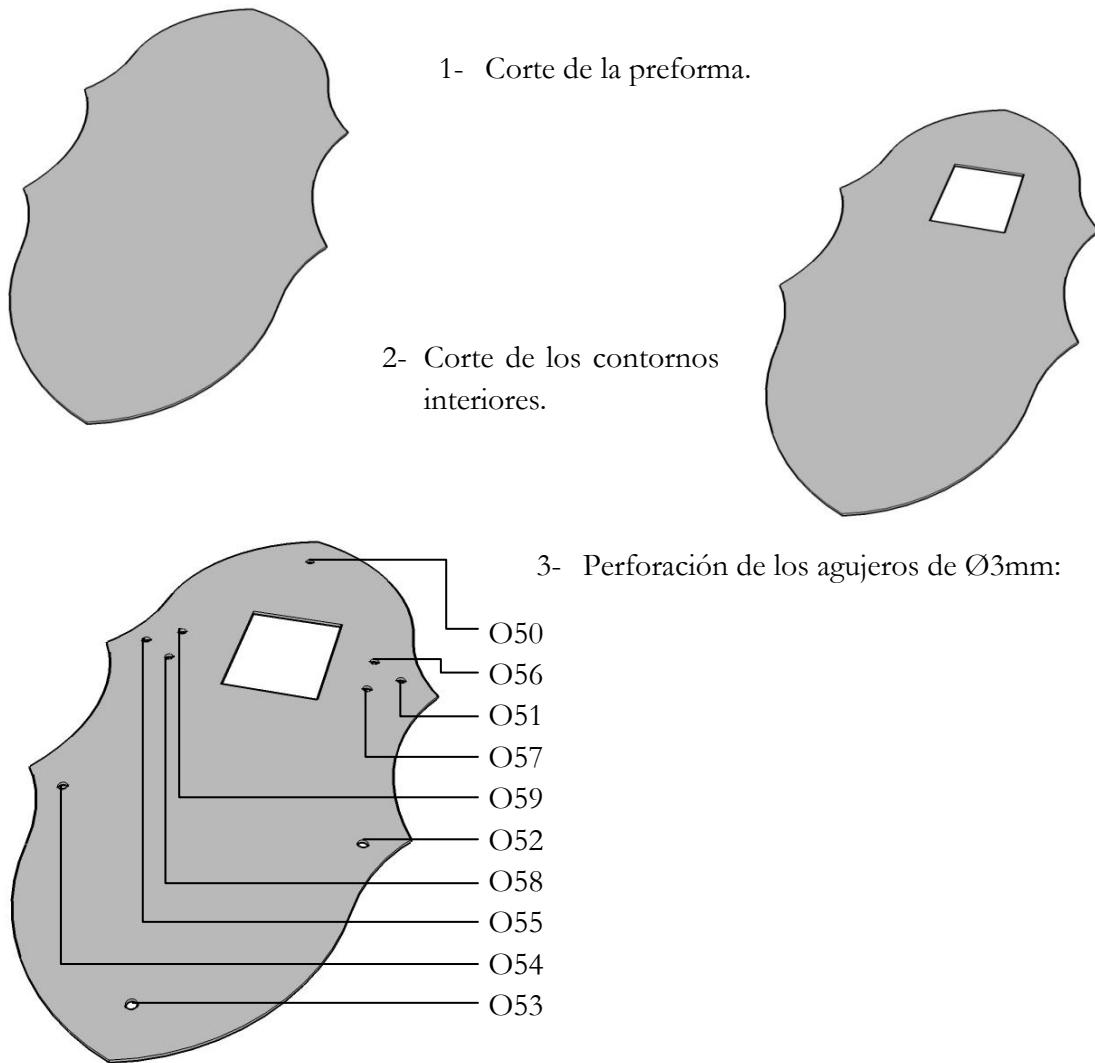


- 4- Avellanado de los orificios [O34 – O39]. El avellanado se realiza por la cara inferior de la pieza (en ésta y las otras imágenes la pieza está boca arriba), ya que los tornillos que encajarán en estos orificios tienen la cabeza cónica.

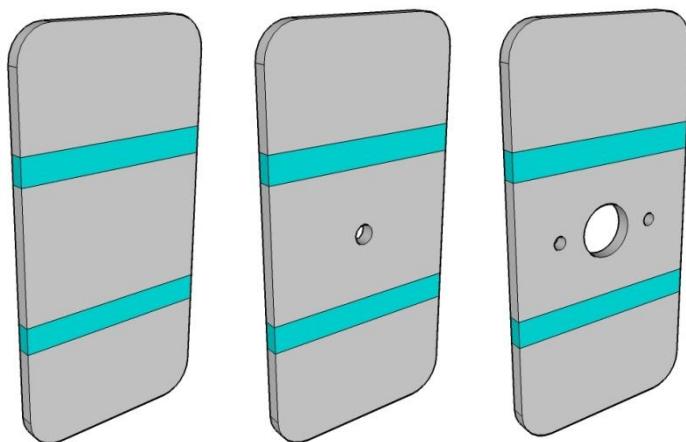
- P01 – Tapa Superior (aluminio 1mm):



- P02 – Tapa Inferior (aluminio 1mm):

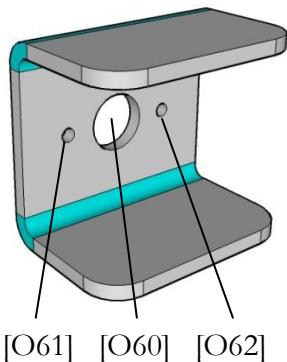


- P11 – Hombro Izquierdo (aluminio 2mm):



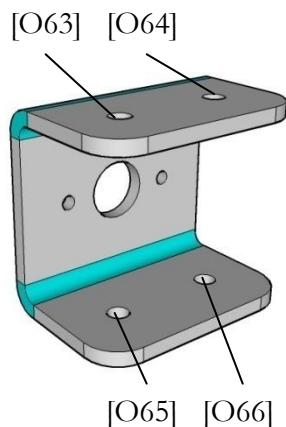
(De izquierda a derecha)

- 1- Corte de la preforma
- 2- Orificio [O60] de Ø3mm para el soporte del eje del servo.
- 3- [O60] se agranda hasta los Ø8mm y, usando el soporte del eje del servo como modelo, se perforan los dos orificios laterales de Ø2mm [O61] y [O62].

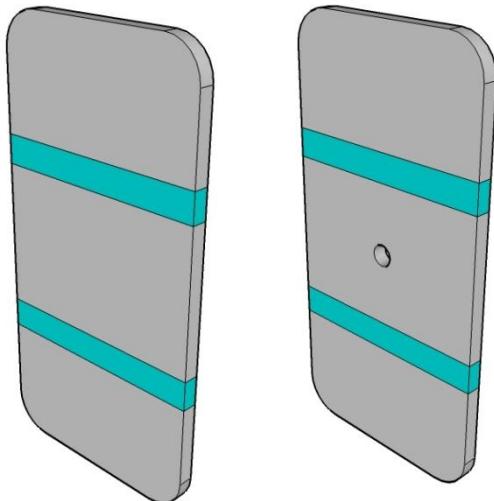


4- Doblado de la pieza.

5- Se añaden los orificios de Ø3mm [O63 – O66]. La medida exacta de estos orificios se toma de los [O42-O43] y [O56-O57], colocando la pieza en su lugar durante el montaje.



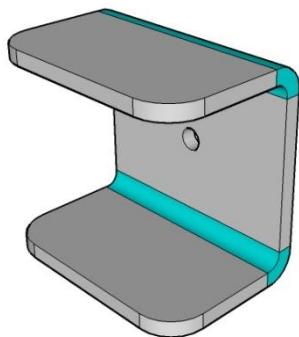
- **P12 – Hombro Derecho (aluminio 2mm):**



(De izquierda a derecha)

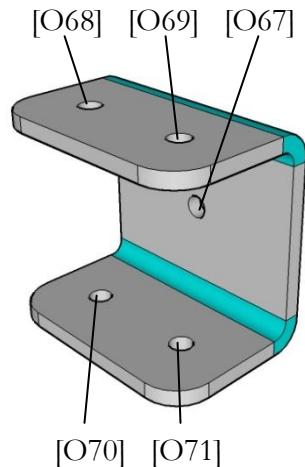
1- Corte de la preforma.

2- Perforación [O67] de Ø3mm para el eje trasero del servo.



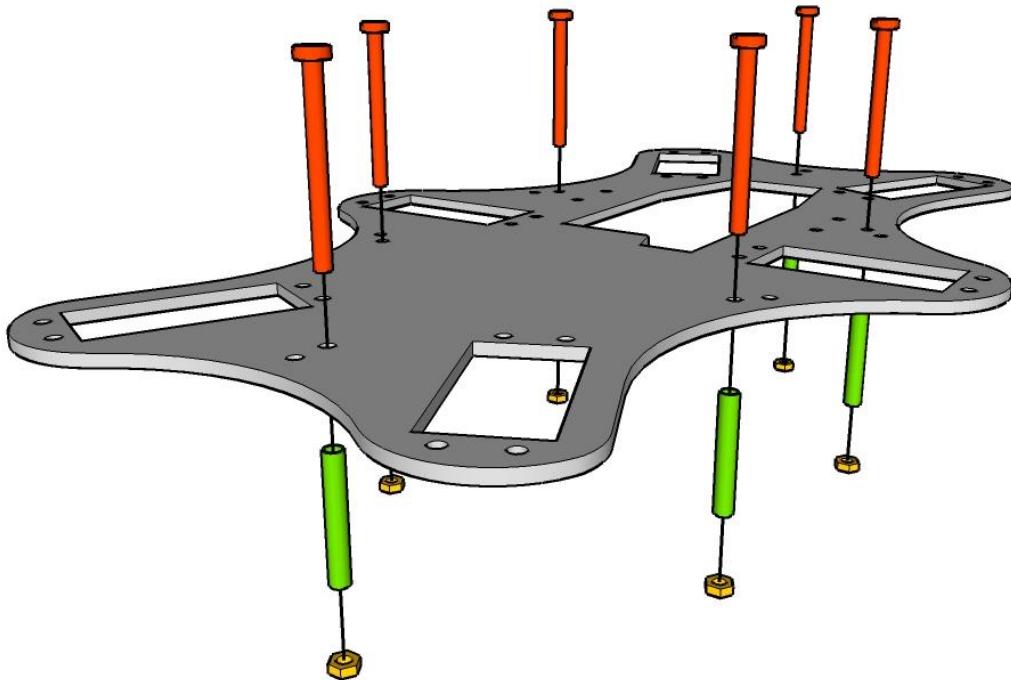
3- Doblado de la pieza

4- Se añaden los orificios de Ø3mm [O68 – O71]. La medida exacta de estos orificios se toma de los [O40-O41] y [O58-O59], colocando la pieza en su lugar durante el montaje.

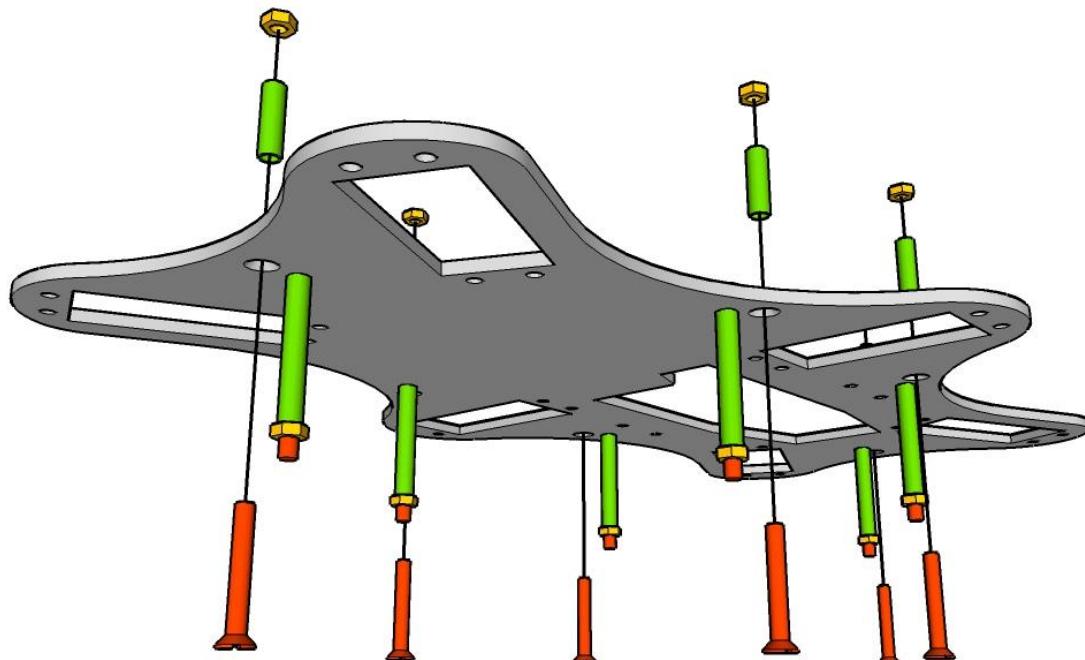


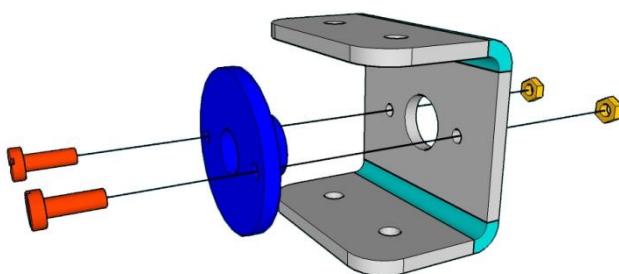
○ Montaje del Tórax:

- 1- El montaje comienza con la inserción de 6 tornillos [T5], a través de los orificios [O28 – O33] de la pieza P00, de forma que sus tuercas sujeten firmemente a 6 separadores [Y1]. El sobrante de tornillo no se recorta hasta el montaje de P02.



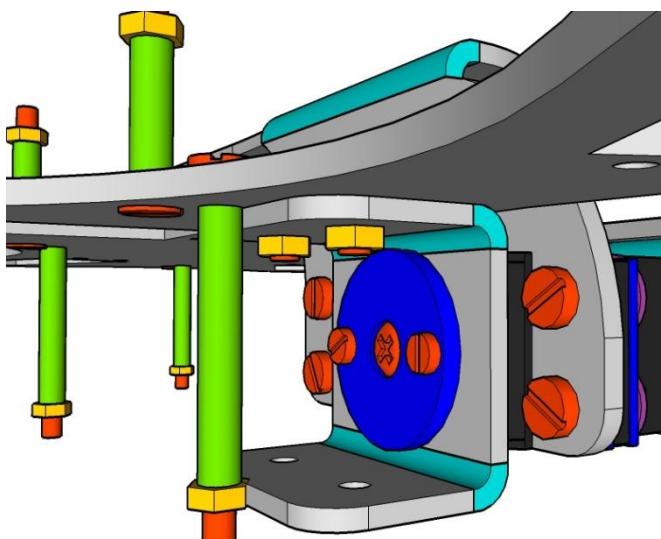
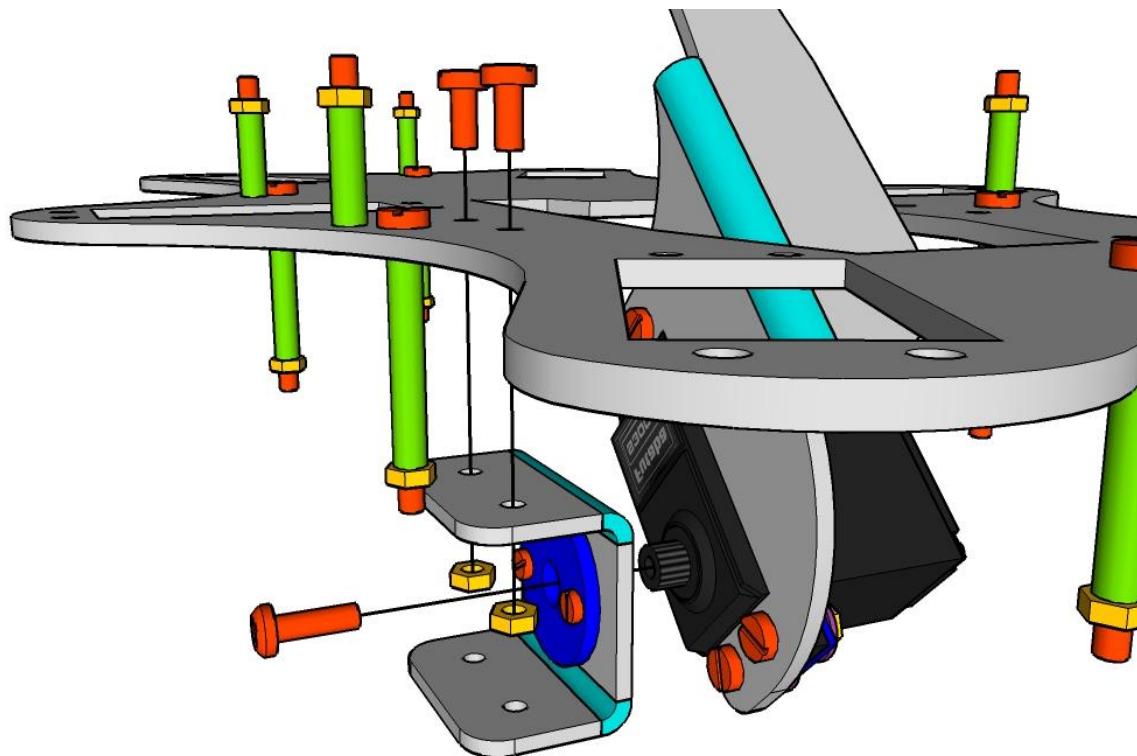
- 2- Del mismo modo, se usan 6 tornillos [T8] para sujetar 6 separadores [Y0] a través de los orificios [O34 – O39] de la pieza P00. Tampoco aquí se recorta el sobrante de tornillo, ya que debe quedar suficiente tornillo para acoplar P01 más adelante.





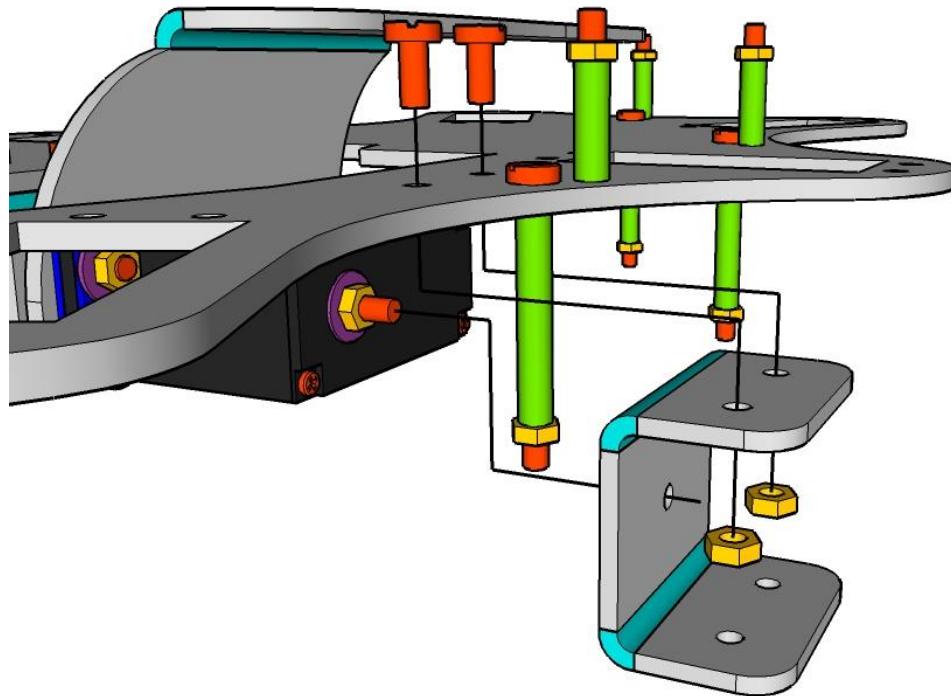
3- Se acopla el soporte del eje del servo [E1] al orificio [O60] de la pieza P11. [E1] se fija al Hombro Izquierdo usando dos tornillos [T1] y sus tuercas, a través de los orificios [O61] y [O62].

- 4- Un tornillo [T0] fija la pieza P11 al servo [S0] del Cuello, atravesando [E1] y [O60], al mismo tiempo que dos tornillos [T3] y sus tuercas unen las piezas P11 y P00 a través de los orificios [O42], [O43], [O63] y [O64].

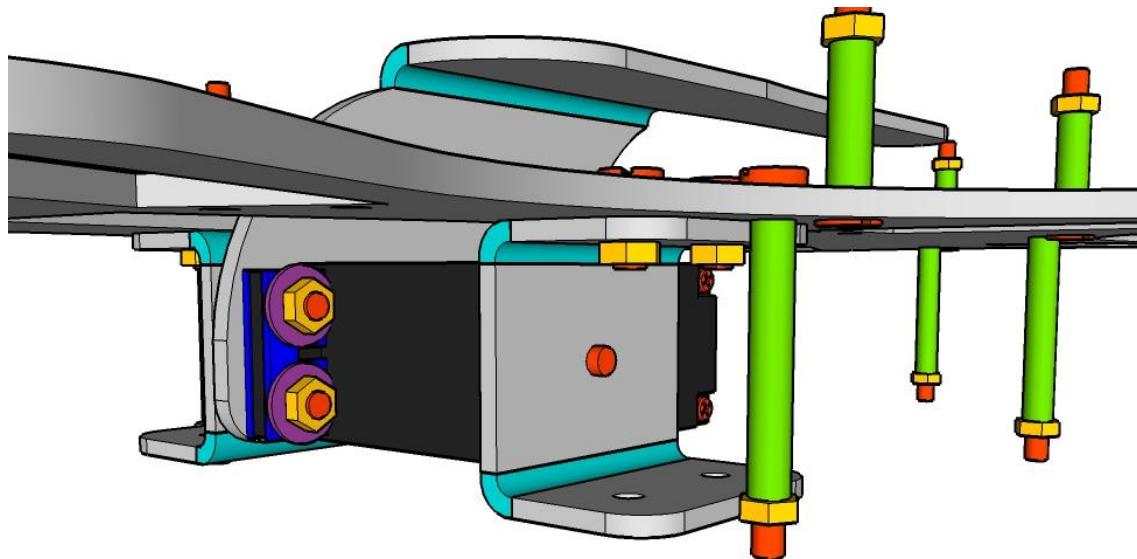


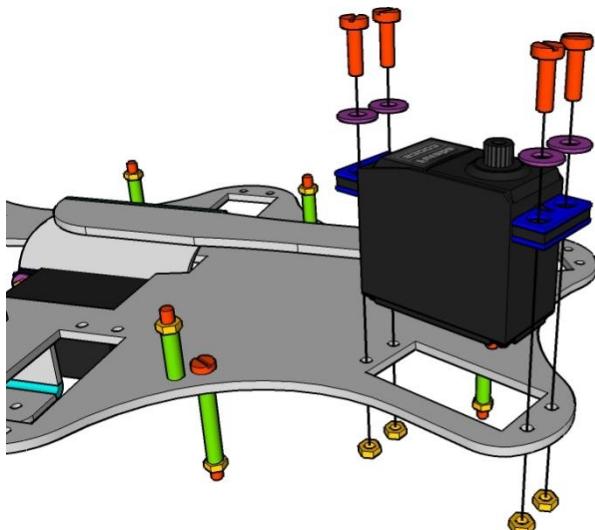
Detalle del Hombro Izquierdo, P11, una vez acoplado al Tórax Base, P00, y al conjunto del Cuello.

- 5- Para montar el Hombro Derecho se acopla la pieza P12 al Tórax Base, P00, mediante dos tornillos [T3] y sus tuercas, y a través de los agujeros [O40], [O41], [O68] y [O69], teniendo en cuenta que el eje trasero del servo ha de atravesar el orificio [O67]. Para esquivar el separador inferior acoplado en P00 justo delante del emplazamiento de P12, éste se debe insertar en el conjunto de abajo a arriba.

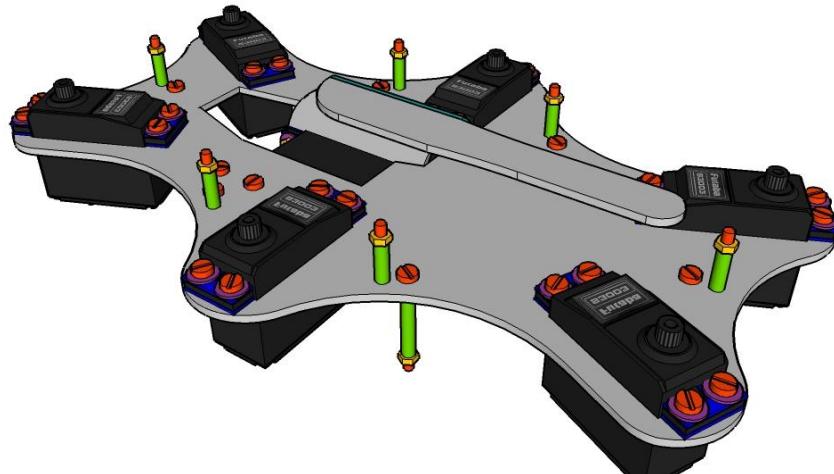
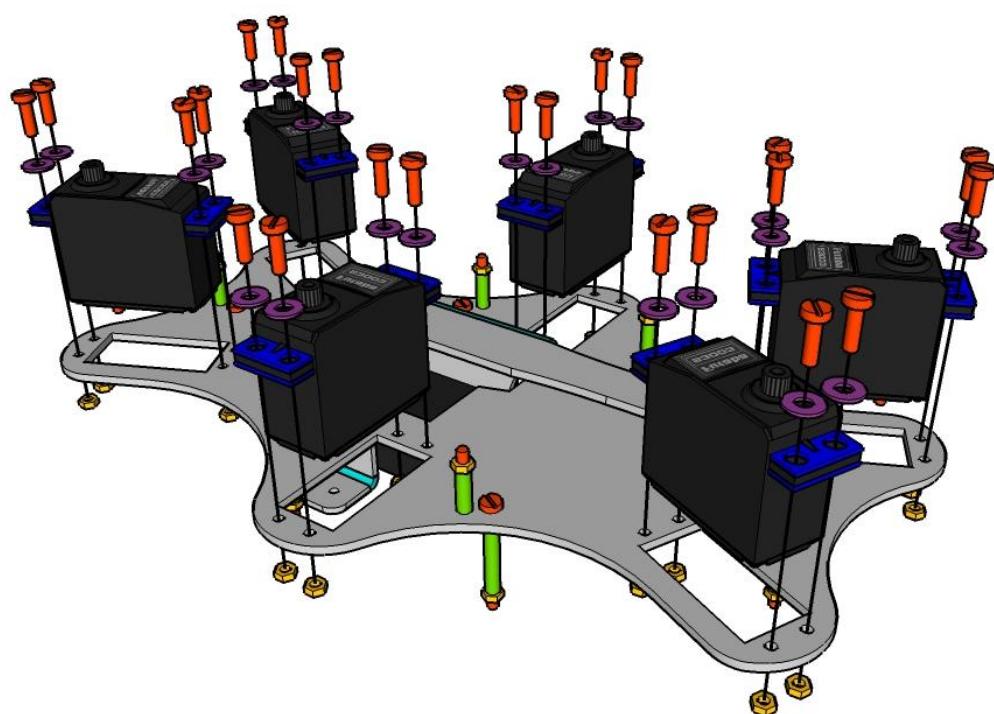


Detalle del Hombro Derecho, P12, una vez acoplado al Tórax Base, P00, y al conjunto del Cuello.



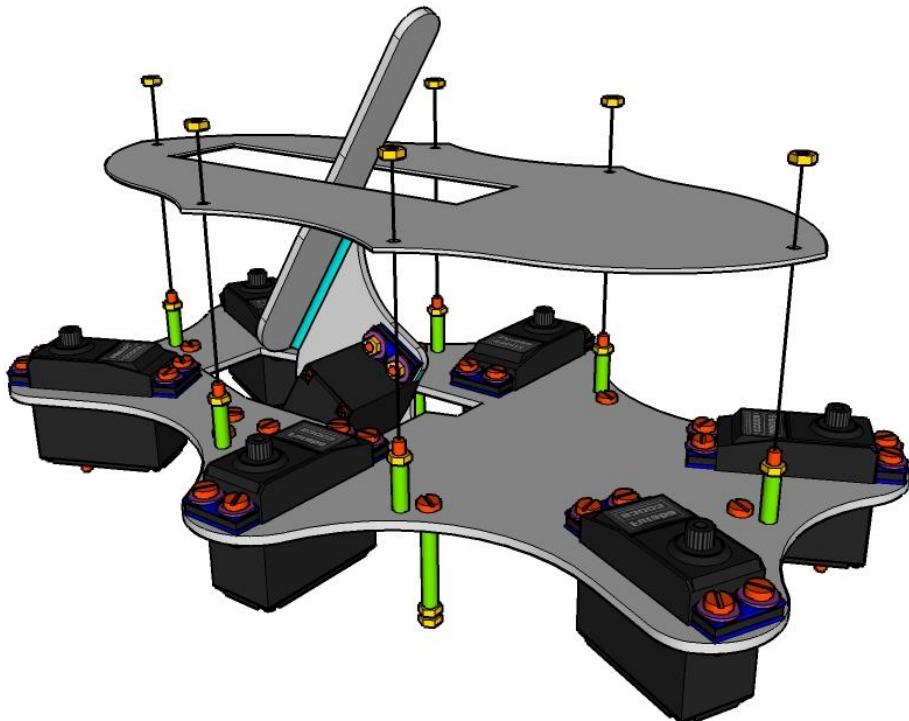


6- El siguiente paso es ensamblar los 6 servos [S1] en sus emplazamientos de la pieza P00, usando 4 tornillos [T4] con sus tuercas y 4 arandelas [A1] por cada servo (24 de cada en total), a través de los orificios [O4 – O27].



Arriba, una vista del montaje completo. A la izquierda, el aspecto del montaje terminado.

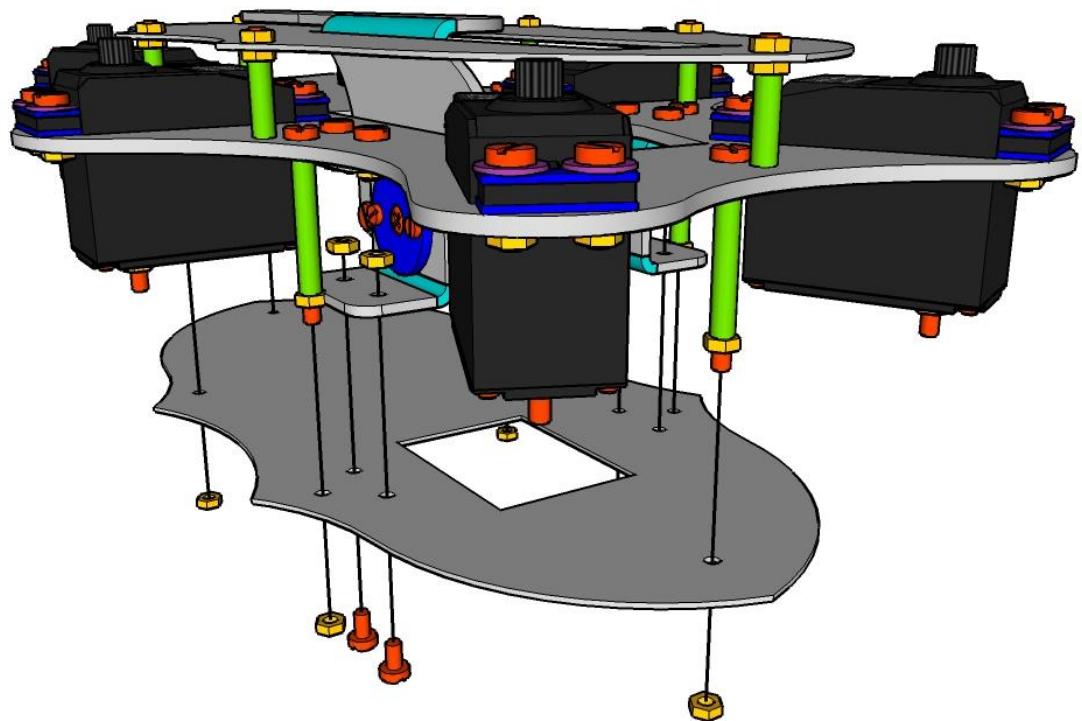
- 7- Una vez instalados los servos, se acopla la Tapa Superior del Tórax, P01, en el sobrante de los tornillos [T8] de los separadores superiores, añadiendo una tuerca a cada tornillo y a través de los orificios [O44 – O49]. Estas tuercas no se aprietan en exceso, ya que la tapa superior deberá retirarse cada vez que haya que manipular los componentes instalados en el Tórax. Para instalar P01 habrá que levantar el conjunto Cuello para hacer pasar P10 por el contorno interior rectangular de P01. Una vez instalada la pieza, ya se pueden recortar a la medida exacta los sobrantes de los tornillos [T8].



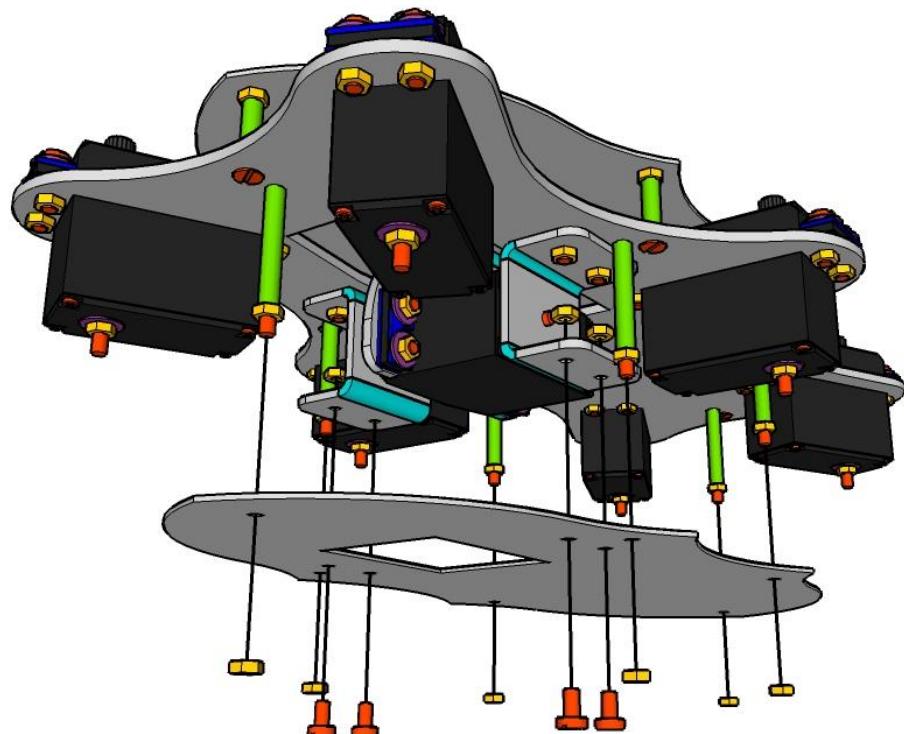
- 8- Siguiendo el mismo proceso se instala la Tapa Inferior del Tórax, P02, añadiendo una tuerca a cada tornillo [T5], y a través de los orificios [O50 – O55]. En este caso habrá que tener en cuenta que también hay que fijar la pieza P11 a P02 mediante dos tornillos [T3] que atravesarán los orificios [O56 – O57], de P02, y [O65 – O66], de P11. Del mismo modo, se fija P12 a P02 usando también dos tornillos [T3] que atravesarán en este caso los orificios [O58 – O59], de P02, y [O70 – O71], de P12. Una vez instalada la pieza, pueden recortarse los sobrantes de todos los tornillos para mejorar el acabado de la Tapa Inferior del Tórax.

Con este último paso concluye el montaje del Tórax, sin embargo las Tapas Superior e Inferior del mismo deberán retirarse en varias ocasiones después de este punto para instalar el resto de componentes electrónicos del robot. Por eso es recomendable que los tornillos [T3] de la Tapa Inferior se instalen al revés (de arriba abajo, en vez de abajo a arriba, como muestra el diagrama de montaje), y que sus tuercas no se aprieten demasiado hasta que el robot esté finalizado y la Tapa Inferior pueda montarse de forma permanente.

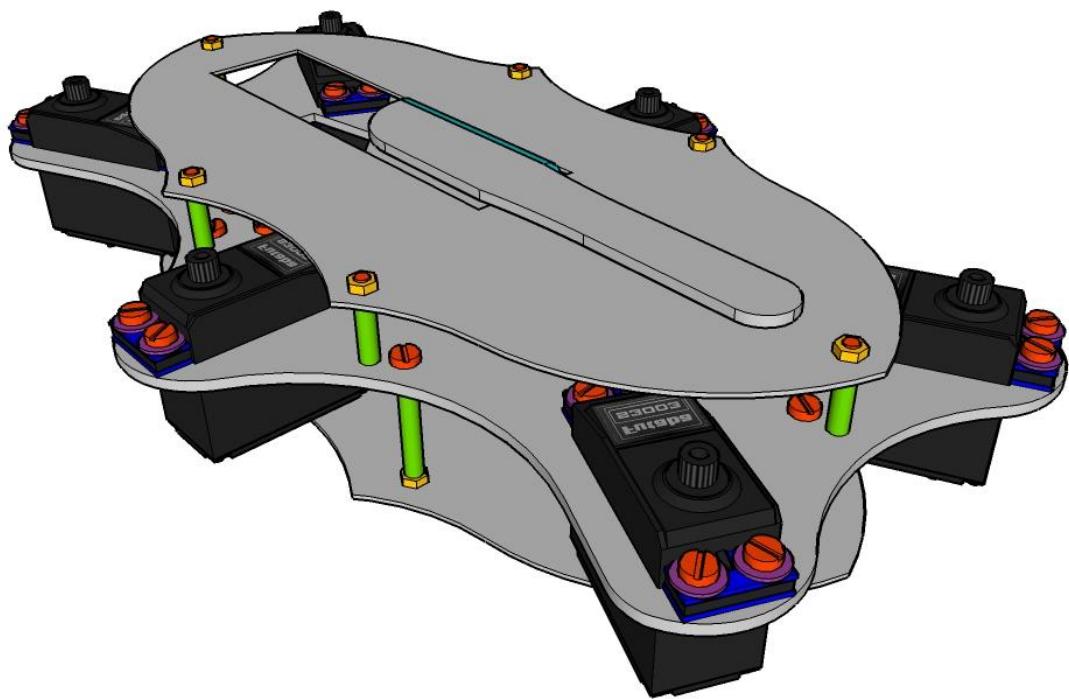
Vista superior del montaje de P02.



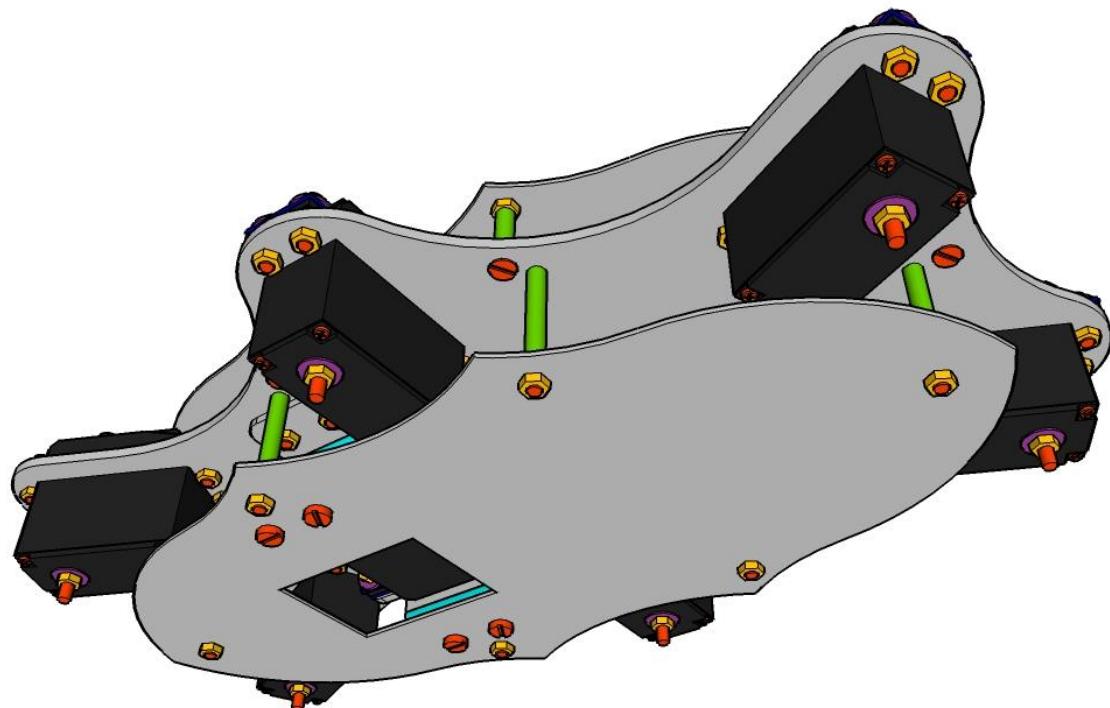
Vista Inferior del montaje de P02.



Vista superior del Tórax + Cuello terminados.



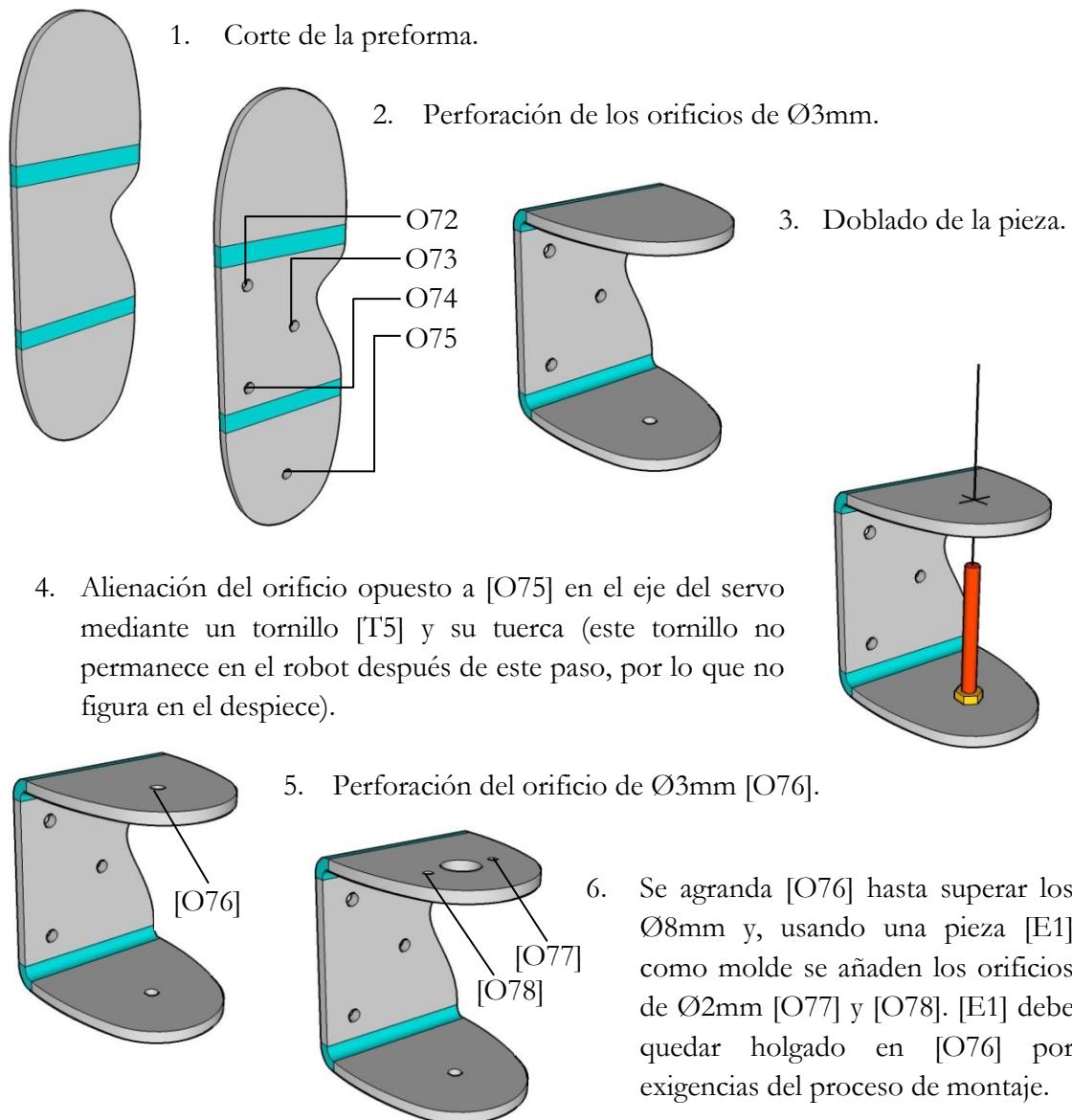
Vista inferior del Tórax + Cuello terminados.



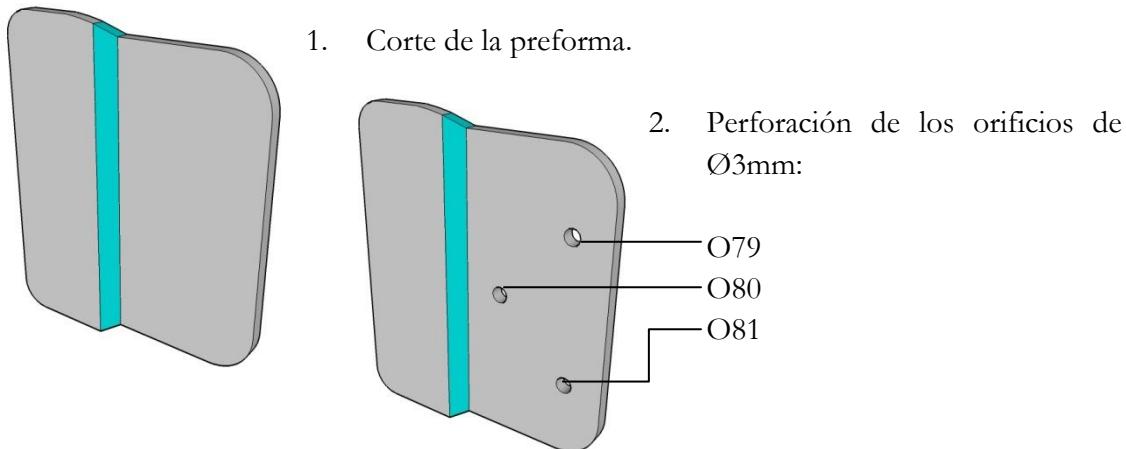
- Recuento de piezas de Tórax + Cuello (Total 223 componentes: 6 piezas, 7 servos, 29 elementos blandos, 12 separadores, 42 arandelas, 58 tornillos y 69 tuercas):
  - P00: Tórax Base
  - P01: Tapa Superior del Tórax
  - P02: Tapa Inferior del Tórax
  - P10: Cuello 1
  - P11: Hombro Izquierdo
  - P12: Hombro Derecho
  - [S0]: Servo “Cuello 1”
  - [S1]: 6 Servos “Cadera”
  - [E0]: 28 Protectores de goma para servo Futaba
  - [E1]: 1 Soporte de plástico para eje Futaba
  - [Y0]: 6 Separadores Superiores
  - [Y1]: 6 Separadores Inferiores
  - [A1]: 42 Arandelas M3
  - [T0]: 1 Tornillo
  - [T1]: 2 Tornillos y 2 Tuercas
  - [T3]: 8 Tornillos y 8 Tuercas
  - [T4]: 28 Tornillos y 28 Tuercas
  - [T5]: 6 Tornillos y 12 Tuercas
  - [T7]: 7 Tornillos y 7 Tuercas
  - [T8]: 6 Tornillos y 12 Tuercas
  
- **Construcción 1 de las patas (Cadera y Fémur):** La fabricación y montaje de las patas es un proceso que se divide en dos etapas. En esta primera etapa se fabrican y se montan las tres primeras piezas, correspondientes a la unión de la Cadera y el Fémur. Una vez completadas, y antes de instalar las Tibias, este módulo de pata incompleta se acopla al tórax. Medido en tiempo, el montaje del tórax y el primer módulo de las patas ocupa aproximadamente el 60% del tiempo total de construcción del robot. El 40% restante se dedica únicamente a construir e instalar las Tibias.

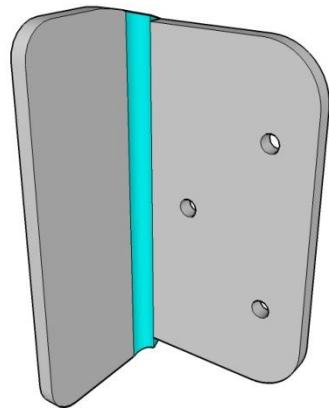
Cabe decir también que las patas son simétricas en dos grupos de tres, las patas derechas y las izquierdas. Las preformas de todas las piezas son idénticas, pero el proceso de doblado las definirá como piezas derechas o izquierdas. Debido a las propiedades del aluminio explicadas anteriormente este proceso es irreversible, por lo que requiere una especial atención.

- P03 – Conejero Cadera - Tórax (aluminio 2'5mm):

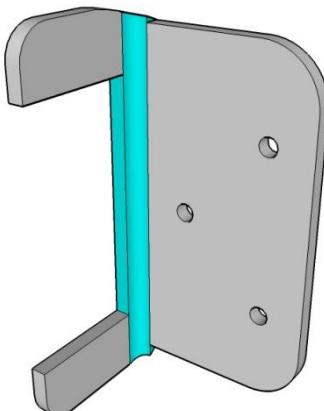


- P04 – Conejero Cadera – Fémur (aluminio 2'5mm):

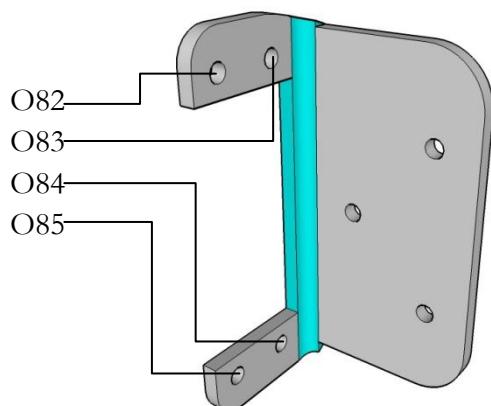




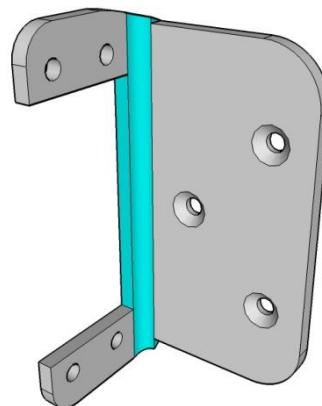
3. Doblado de la pieza.



4. Corte del hueco para el servo.



5. Perforación de los orificios de Ø3mm:

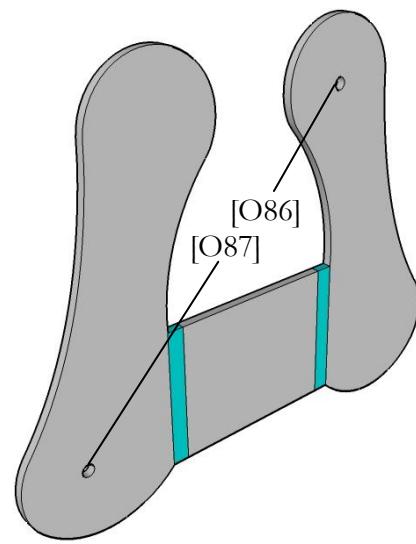
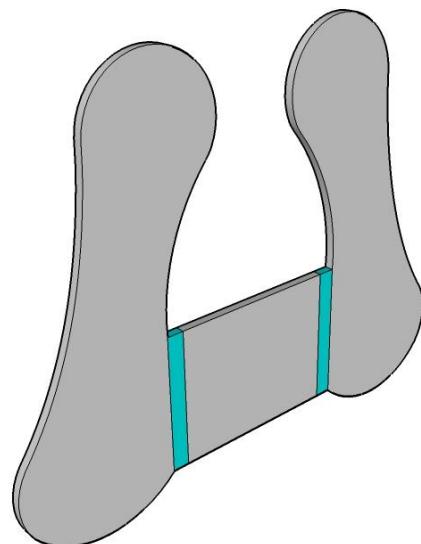


6. Avellanado de los orificios [O79], [O80] y [O81].

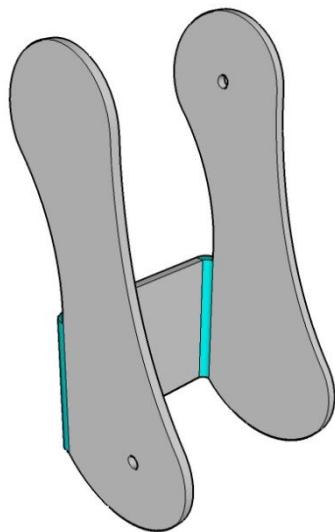
○ P05 – Fémur (aluminio 2mm):

1. Corte de la preforma

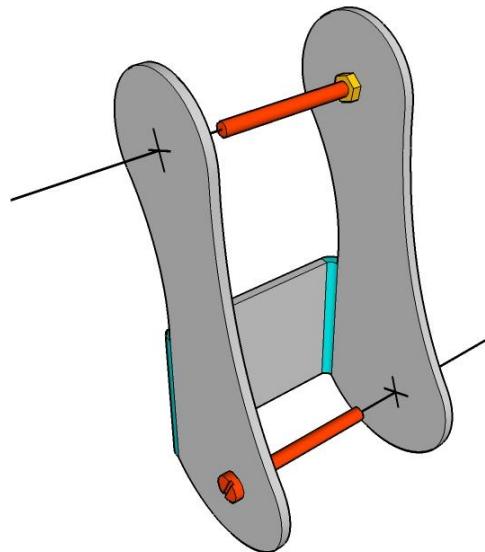
2. Perforación de los orificios de Ø3mm:



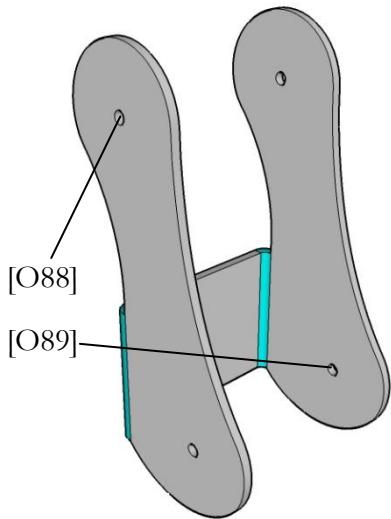
2. Doblado de la pieza



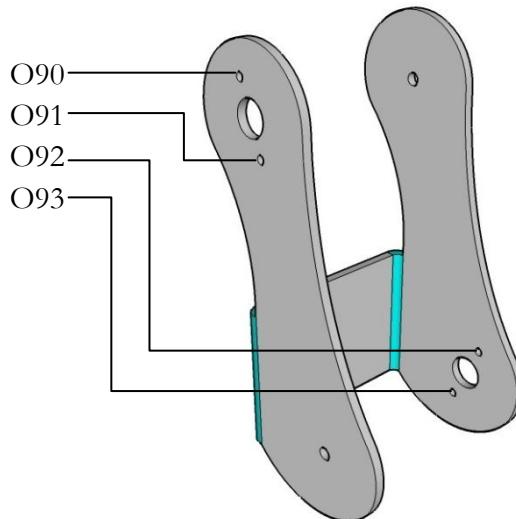
4. Alienación de los orificios opuestos a [O86] y [O87] mediante dos tornillos [T5] y sus tuercas (de nuevo estos tornillos son temporales, por lo que no figuran en el despiece).



5. Perforación de los orificios [O88 – O89] de Ø3mm:



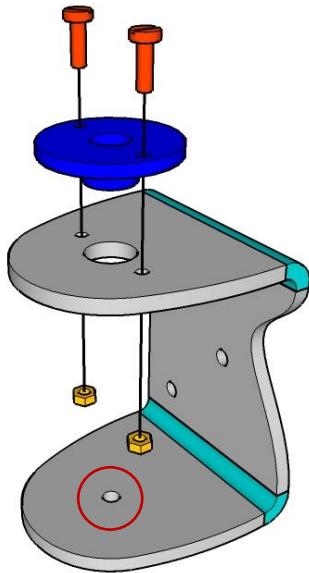
6. Se agrandan [O88] y [O89] hasta los Ø8mm y, usando una pieza [E1] como molde, se añaden los orificios de Ø2mm [O90 – O91] y [O92 – O93].



[E1] debe quedar holgado en [O89] por exigencias del proceso de montaje, mientras que [O88] puede quedar ajustado. Esto se debe a que, excepto con [O88], el eje del servo tendrá que entrar en el orificio antes que [E1] para poder quedar encerrado en la pieza.

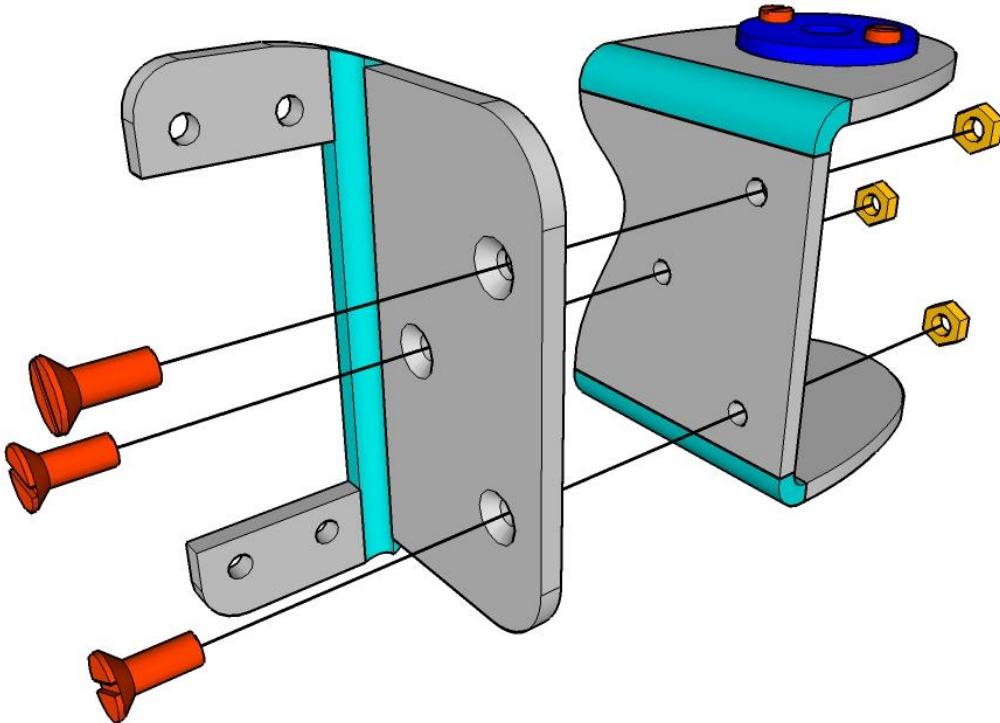
- Montaje 1 de las patas (Cadera + Fémur):

1. El montaje comienza con la instalación de una pieza [E1] en el interior del orificio [O76] del Conector Cadera – Tórax, P03, y sujetándola con dos tornillos [T1] y sus tuercas a través de los orificios [O77] y [O78]. La pieza [E1] sobresale aproximadamente 0'5mm por la cara interior de P03,

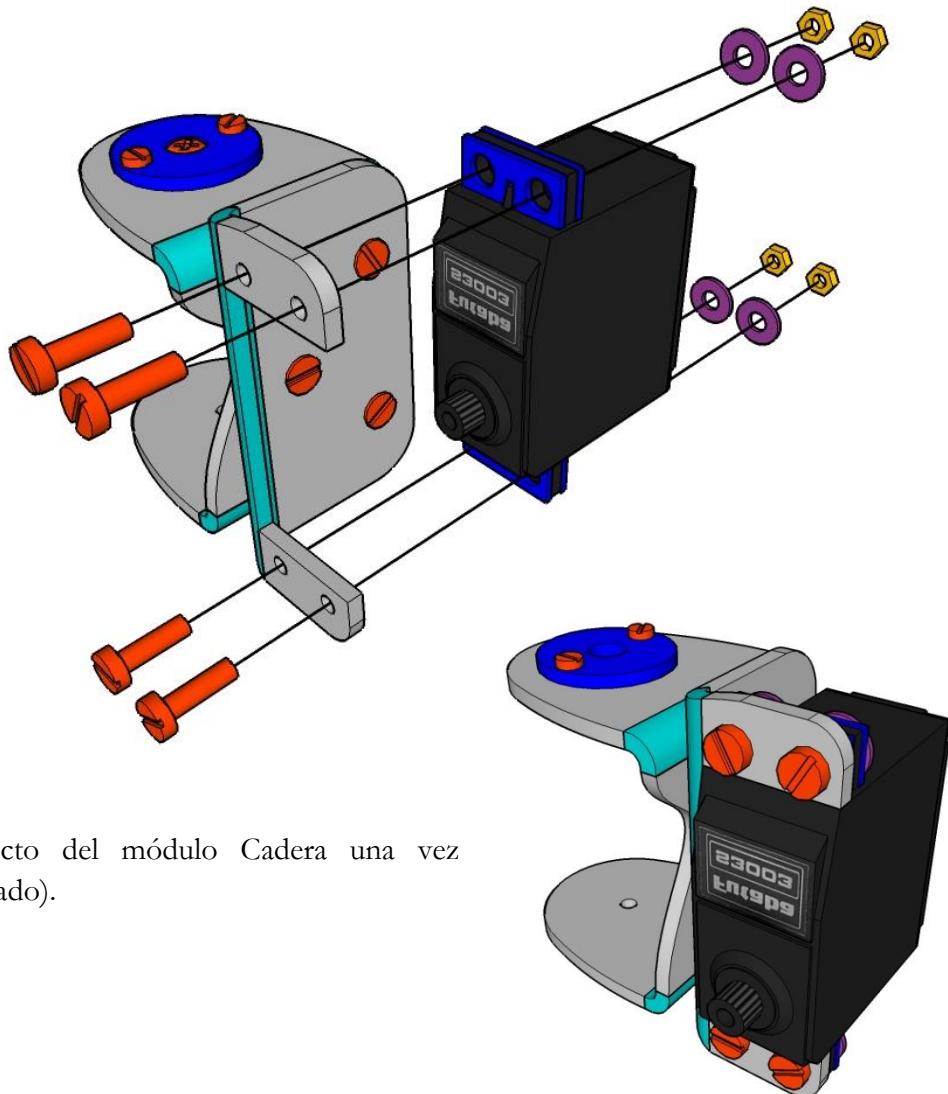


de manera que sólo podrá colocarse en aquellas piezas que muestren suficiente holgura para introducir los dos ejes del servo en sus respectivos orificios. En aquellas piezas en que al doblarse haya quedado más estrecha la distancia entre [O75] y [O76], la pieza [E1] deberá instalarse más adelante, cuando los ejes del servo hayan sido introducidos en estos orificios. En estos casos el servo se introduce empezando por el eje trasero, y desde una trayectoria de inserción de más de 30° respecto de la línea que definen los orificios de los ejes, por lo que puede hacerse necesario incluso realizar un avellanado en [O75] por la cara interna de P03 para aumentar el ángulo de inclinación del eje que admite el orificio.

2. Acto seguido se acoplan las piezas P03, Conector Cadera – Fémur, y P04, Conector Cadera – Tórax, mediante tres tornillos [T7] con sus tuercas, y a través de los orificios [O79 – O81] y [O72 – O74].

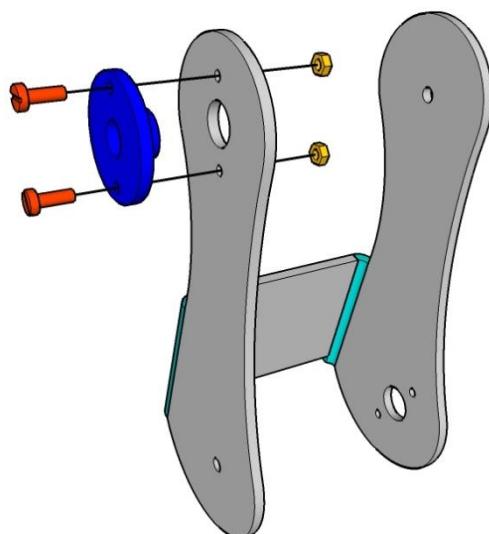


3. Para terminar el módulo Cadera se acopla un servo [S2] a la pieza P04 mediante cuatro tornillos [T4] con sus tuercas, cuatro arandelas [A1] y a través de los orificios [O82 – O85].

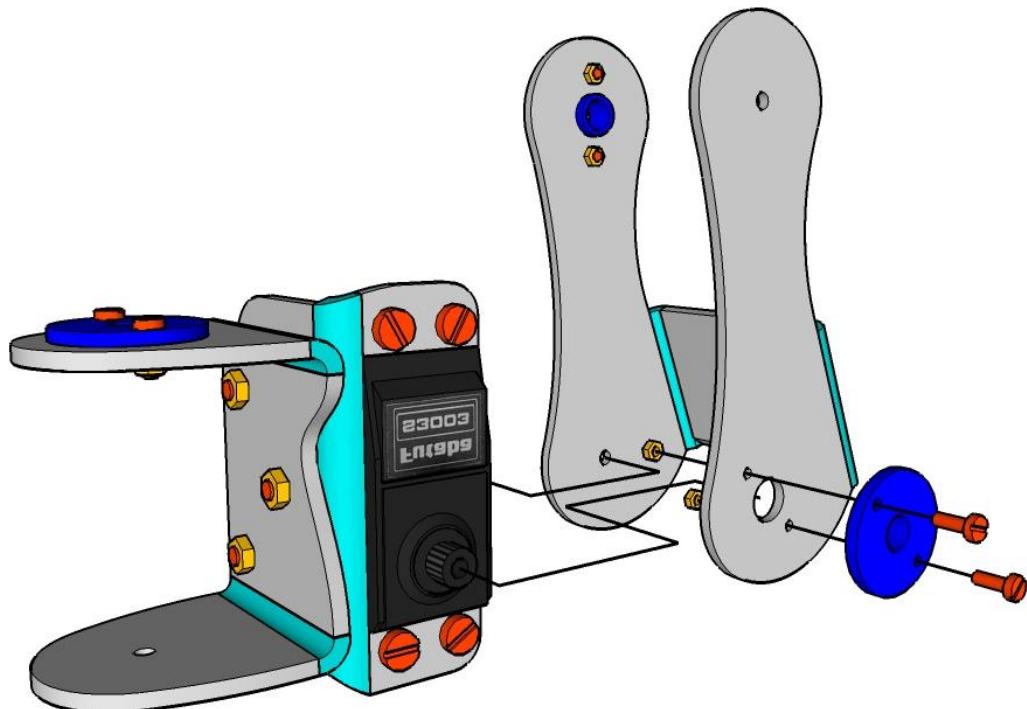


(Aspecto del módulo Cadera una vez montado).

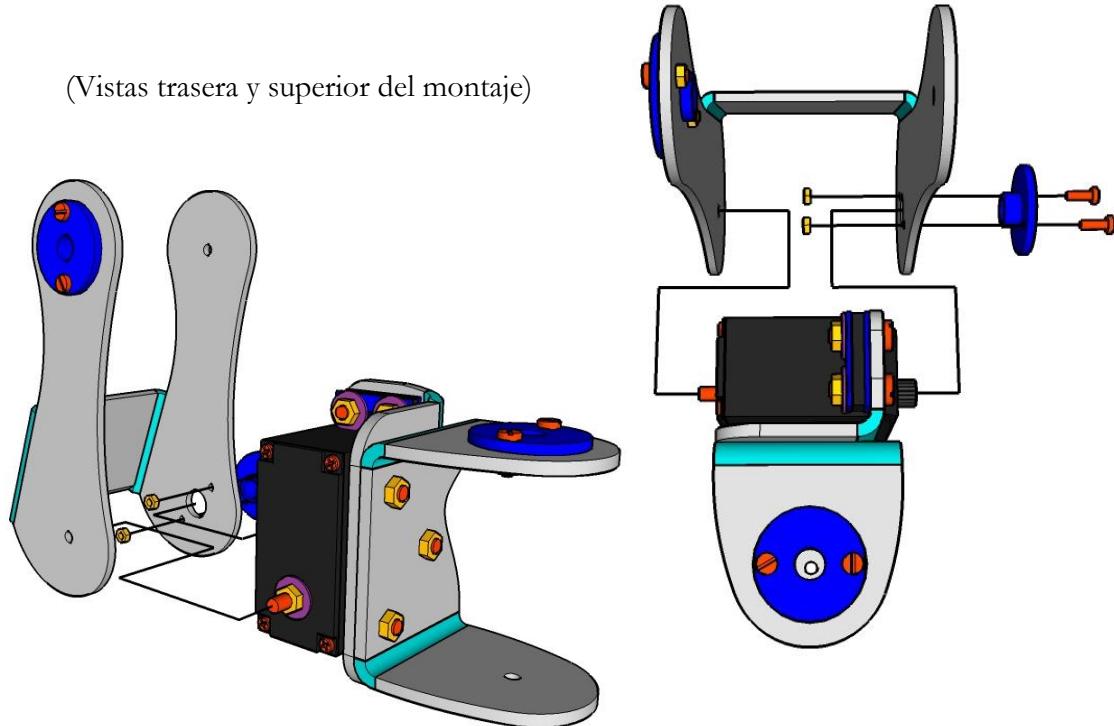
4. Por su parte el módulo Fémur sólo lo componen la pieza P05 y dos soportes [E1] que se conectarán a los ejes motrices de los servos [S2] y [S3]. No obstante, por ahora sólo se acopla una de las piezas [E1] en el orificio [O88] y mediante dos tornillos [T1] con sus tuercas a través de [O90 – O91]. En este caso se coloca [E1] antes que el servo, por lo que la pieza puede quedar ajustada al orificio. Por el contrario, la pieza que se instalará en [O89] debe quedar holgada, porque se introducirá el eje del servo en el orificio antes que esta pieza.



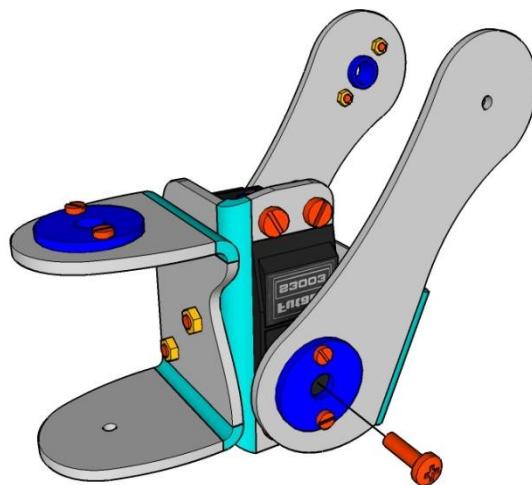
- Empezando por el eje trasero, y realizando un avellanado del lado interior de [O87] si fuera necesario, tal y como sucedió con la pieza P03, se insertan los ejes del servo [S2] en los orificios [O87] y [O89] de P05. Acto seguido, el servo queda atrapado en la pieza P05 acoplándole al eje motriz del servo, y a través de [O89], un soporte [E1] sujeto por dos tornillos [T1] con sus tuercas a través de [O92 – O93]. Esta operación es algo tediosa, ya que, una vez colocado el servo en su lugar, el espacio disponible para enroscar cada tuerca al tornillo [T1] será únicamente el que separa el frontal del servo de la cara interna de P05.



(Vistas trasera y superior del montaje)

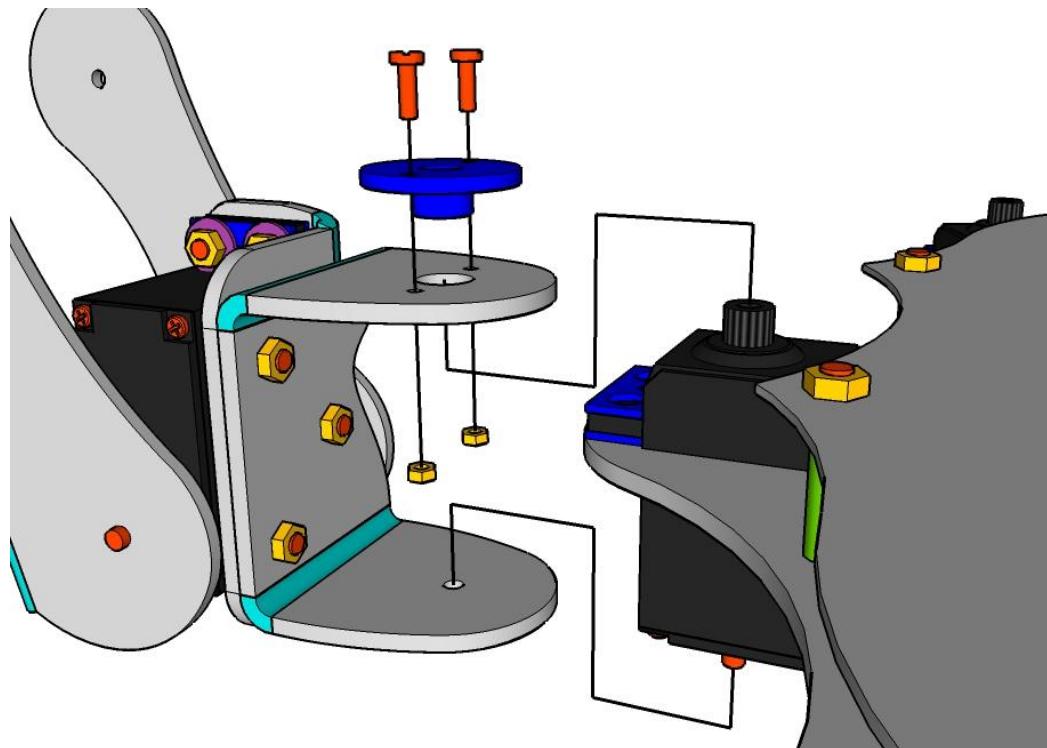


6. Por último, un tornillo [T0] asegura [E1] al servo.

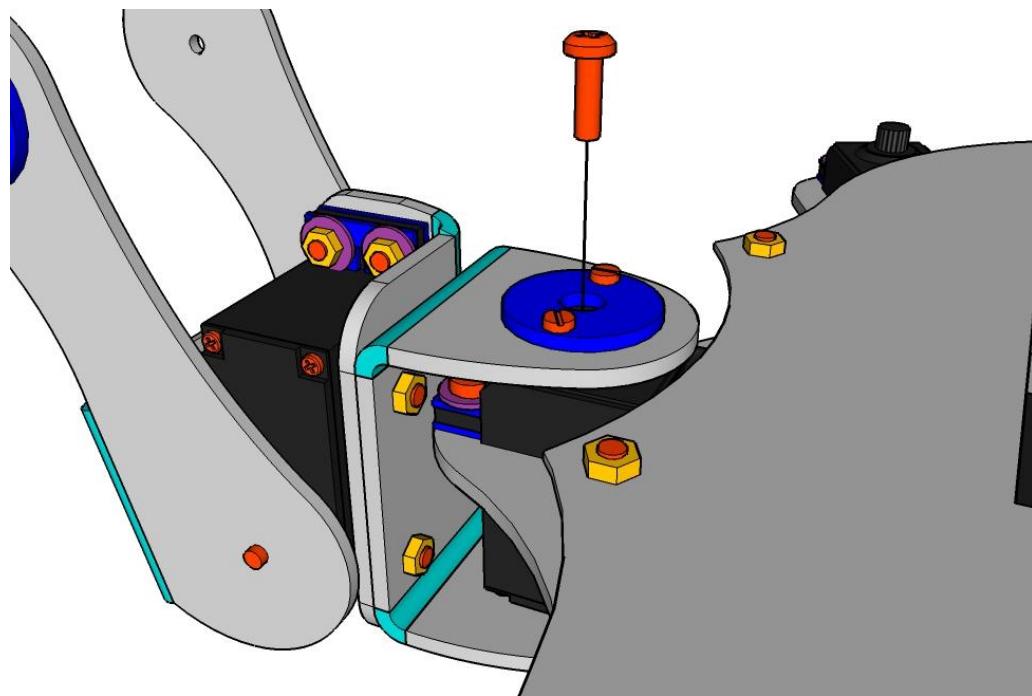


- Montaje de (Cadera + Fémur) + Tórax:

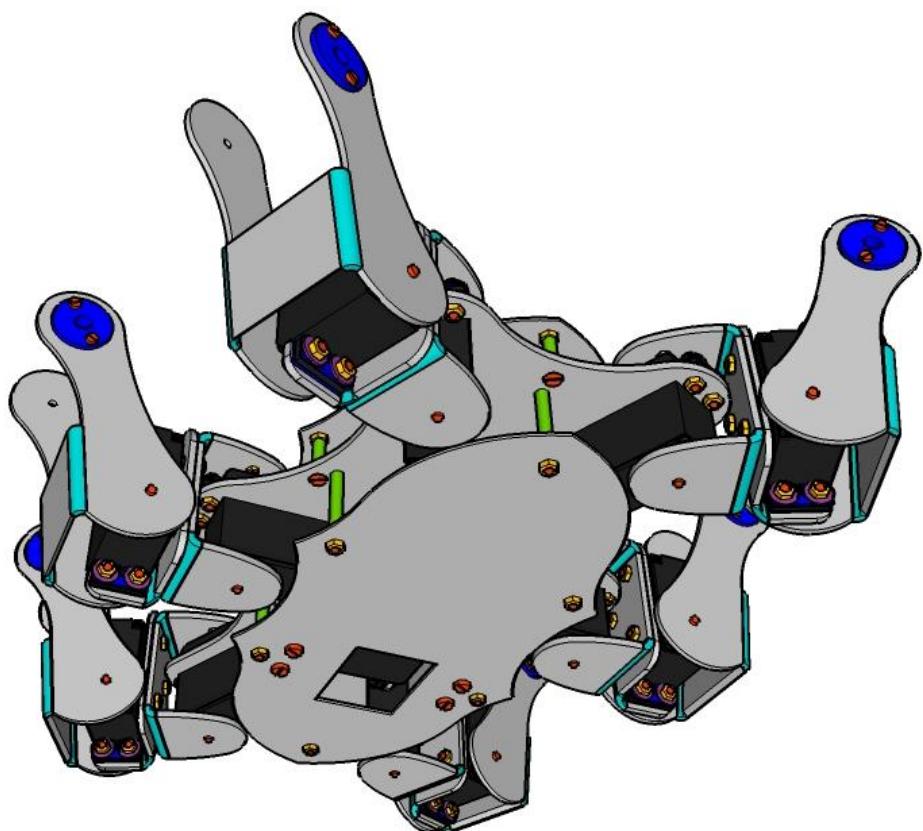
1. El montaje comienza con la inserción, comenzando por el eje trasero e insertándolo desde una trayectoria diagonal, de uno de los servos [S1] instalados en el Tórax entre los orificios [O75] y [O76] de la pieza P03. En este momento puede ser necesario desacoplar el soporte [E1] que podríamos haber acoplado anteriormente, o incluso de avellanar la cara interior del orificio [O75] para lograr que el eje trasero se inserte completamente antes de insertar el eje motriz. Si esto ocurriera, a continuación habría que encerrar el servo en la pieza acoplando [E1] al eje motriz del servo y fijándolo con los dos tornillos [T1] y sus tuercas a P03.



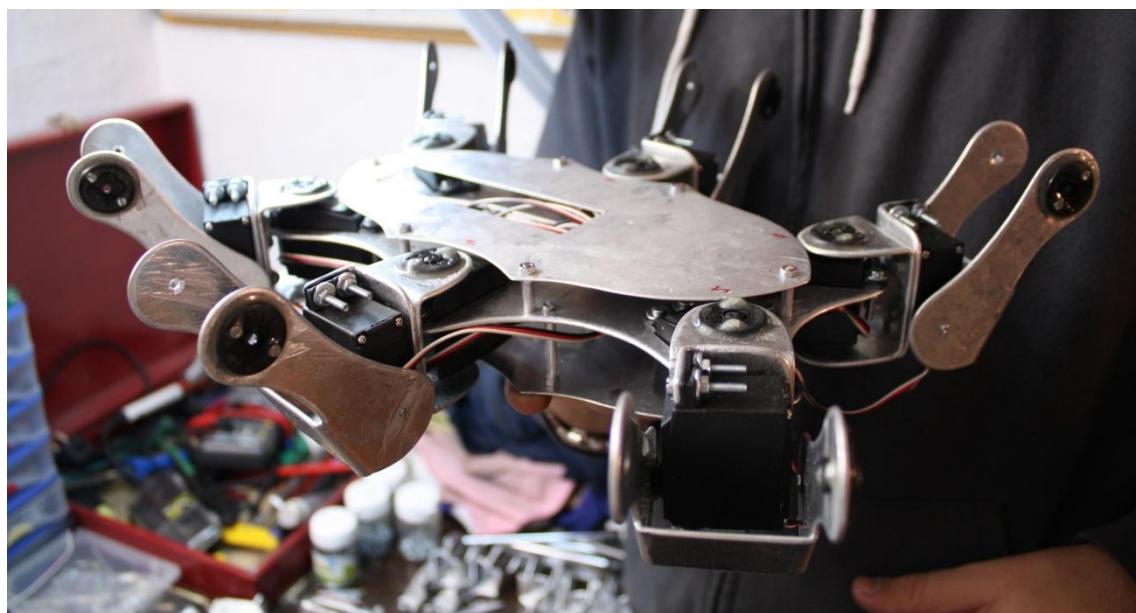
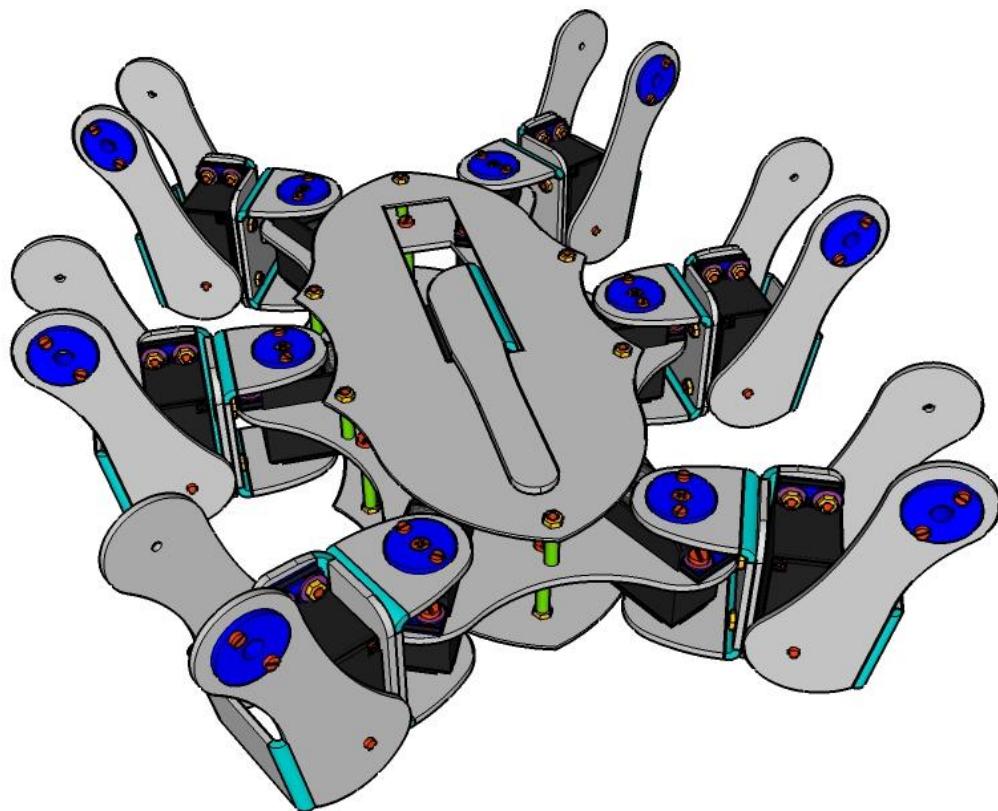
2. Por último, basta con usar un tornillo [T0] para fijar [E1] al eje motriz del servo y repetir todo el proceso seis veces hasta acoplar cada una de las seis patas aún incompletas al Tórax.



(Imagen inferior del montaje completado).



(Imagen superior del montaje terminado. Es interesante compararla con la imagen real del robot en esta etapa de montaje para ver que la construcción real del aparato cumple hasta este punto con todas las especificaciones de diseño).



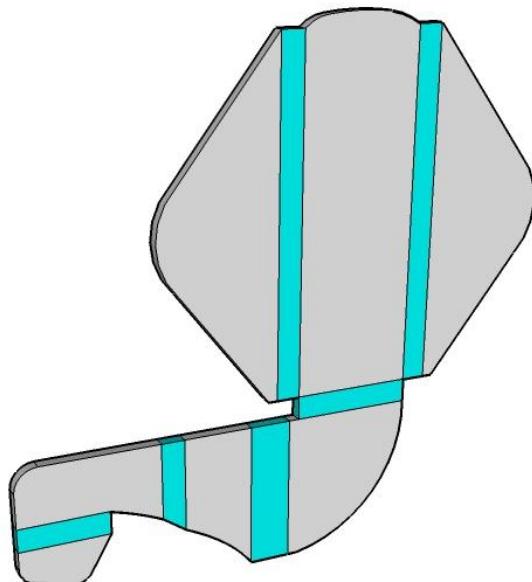


- Recuento de piezas de los 6 módulos Cadera + Fémur (Total 282 componentes: 18 piezas, 6 servos, 42 elementos blandos, 36 arandelas, 96 tornillos y 84 tuercas):
  - P03: 6 Conectores Cadera - Tórax
  - P04: 6 Conectores Cadera - Fémur
  - P05: 6 Fémur
  - [S2]: 6 Servos “Fémur”
  - [E0]: 24 Protectores de goma para servo Futaba
  - [E1]: 18 Soportes de plástico para eje Futaba
  - [A1]: 36 Arandelas normales M3
  - [T0]: 12 Tornillos (6 en el montaje (Cadera + Fémur) + Tórax)
  - [T1]: 36 Tornillos y 36 Tuercas
  - [T4]: 24 Tornillos y 24 Tuercas
  - [T7]: 24 Tornillos y 24 Tuercas

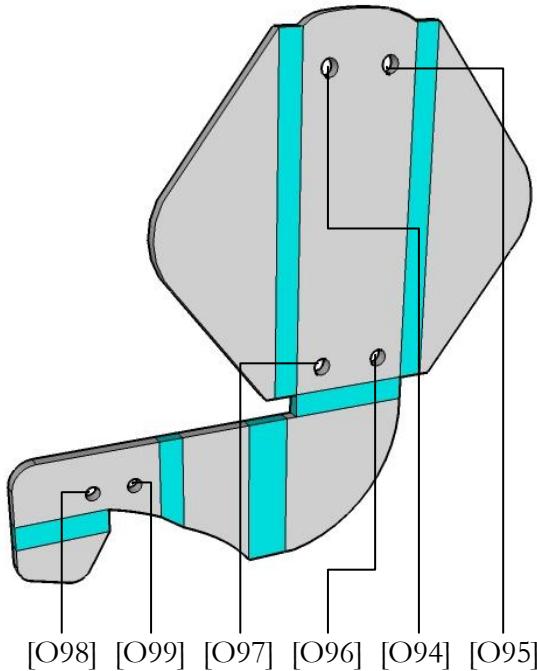
- **Construcción 2 de las patas (Tibias):** Esta etapa del montaje dejará el robot terminado con todas las partes de su esqueleto a excepción de la Cabeza. La construcción de las Tibias es más compleja y lenta que las etapas de montaje vistas hasta ahora, ya que integra piezas más pequeñas y exige mayor precisión en la fabricación de las mismas, de modo que se emplea casi tanto tiempo en construir y montar las Tibias como en el resto del robot. Las Tibias también son simétricas en dos grupos de tres, y cada una de ellas está formada por tres piezas de aluminio que a menudo no encajarán a la perfección entre sí, de modo que habrá que seleccionar y agrupar aquellas que sean más afines para obtener los mejores resultados, y compararlas en algunos pasos para acertar la posición exacta de dobleces y orificios. A causa de esta imprecisión, muchas de las piezas pequeñas como muelles, separadores y tornillos deberán instalarse de forma previa para medirse, cortarse a medida y volverse a instalar. De este modo, y aunque esta guía de montaje supone piezas perfectas y cortadas a medida para no resultar caótica y repetitiva, la realidad tras cada etapa del montaje incluye decenas de pasos intermedios que aquí se omiten.

- **P06 – Parte Fija de la Tibia (aluminio 2mm):**

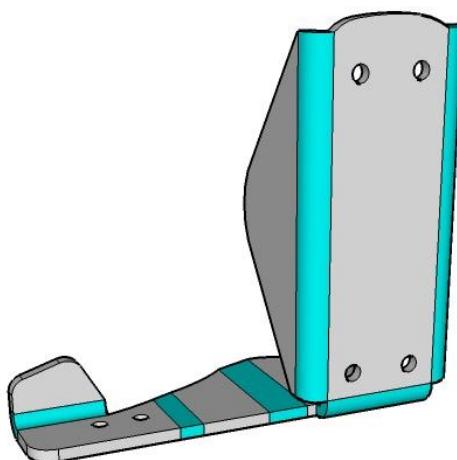
1- Corte de la preforma



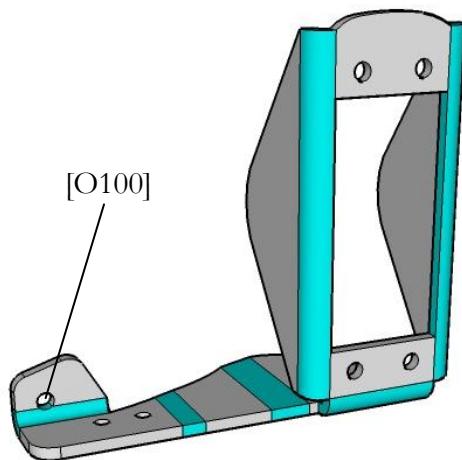
2- Perforación de los orificios de Ø3mm [O94 – O97] y de los orificios de Ø2mm [O98 – O99].



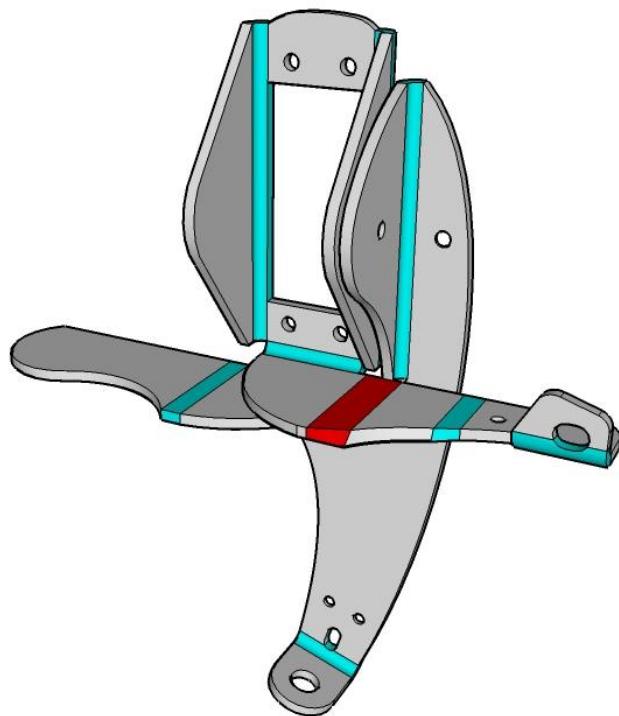
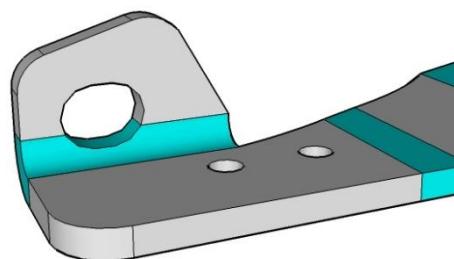
1- Primer doblado de la pieza en tres de sus aletas.



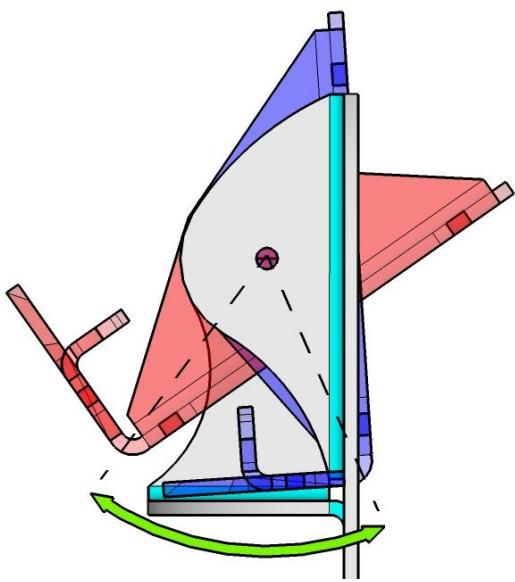
2- Corte del contorno interior que albergará el servo y perforación del orificio de Ø3mm [O100].



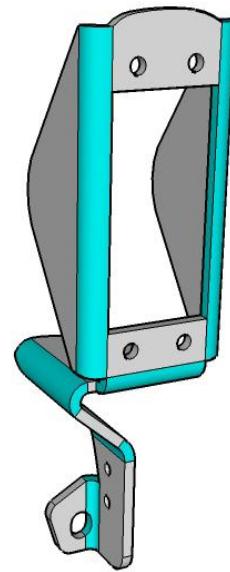
2- Con una fresa de Ø3mm se agranda y alarga el orificio [O100] hasta que éste tome la forma de un óvalo de unos 4'5mm en su diámetro corto.



3- El siguiente paso es el doblado de las dos franjas que aún quedaban por doblar, pero para realizar el doblez en la primera de ellas (marcada en rojo en la imagen) es necesario hallar su posición exacta comparando individualmente cada pieza P06 con la pieza P08 con la que ha de encajar. Además, esta pieza P08 deberá hallarse por lo menos en su etapa 4 de fabricación para resultar útil en este proceso. Se entiende pues, que el proceso de fabricación de las piezas P06 y P08 debe llevarse en paralelo, y que ambos juegos de piezas deben agruparse de dos en dos para garantizar que cada pareja encaja de forma óptima.

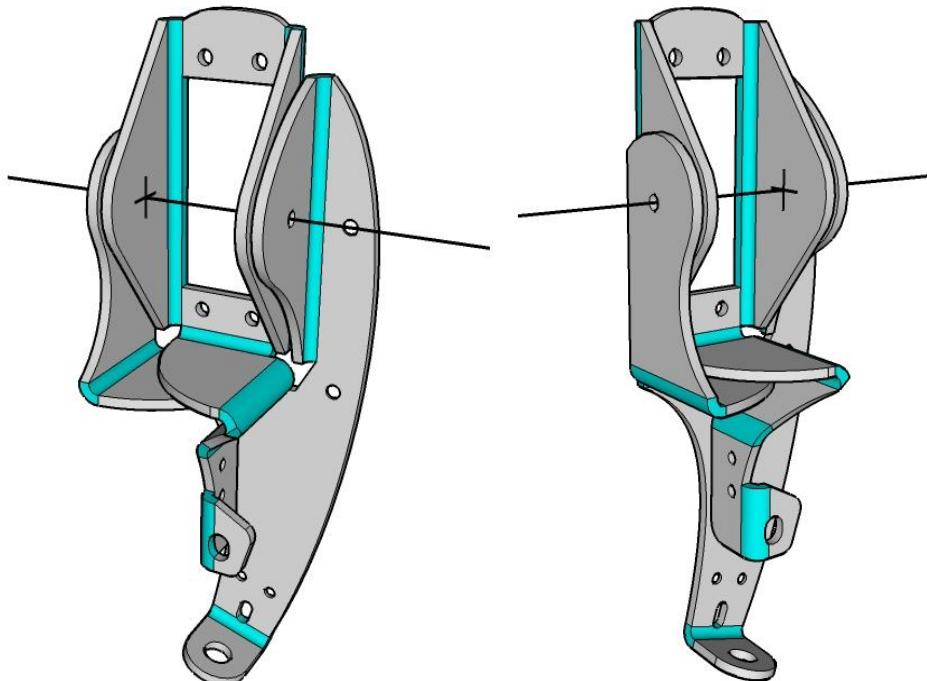


La imagen de la derecha muestra los parámetros a tener en cuenta en cuanto a la posición vertical correcta de P06 respecto a P08. P06 ha de poder pendular hacia atrás lo suficiente como para que la pieza pueda manipularse durante su montaje (en rojo) y también ha de poder pendular hacia delante unos pocos grados (en azul), suficientes como

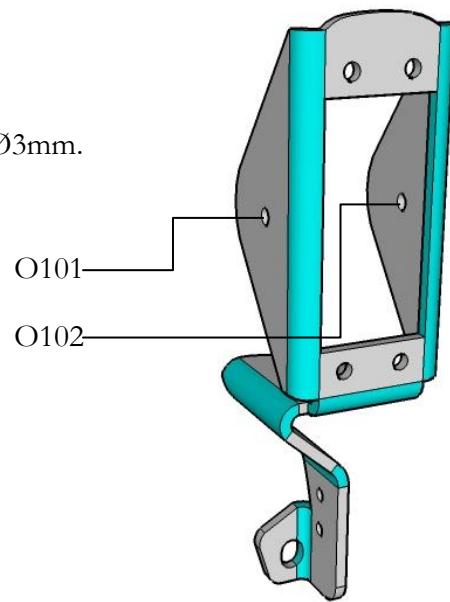


para que el sensor de obstáculo que irá instalado en la aleta de la pieza P06 pueda comprimir su lengüeta contra el cuerpo de P08. Si P06 queda demasiado alta o demasiado baja respecto a P08, estos dos movimientos pueden quedar obstaculizados. La imagen de la derecha muestra el aspecto de la pieza una vez doblada.

- 4- El siguiente paso de nuevo requiere la interacción de las piezas P06 y P08, aunque esta vez será necesario que P08 se halle al menos en su etapa 8 de fabricación. Colocando las dos piezas en la dimensión vertical correcta, según explica el paso anterior, se usan los orificios de P08 [O112] y [O113], previamente alineados, para marcar los dos orificios en P06 que definirán el eje en que esta pieza pendulará sobre P08.

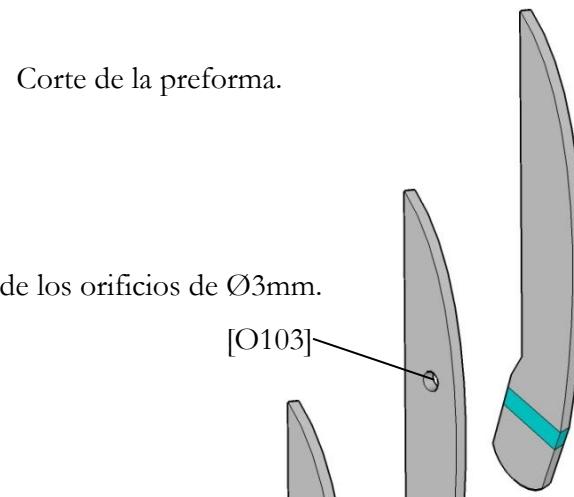


5- Por último, se perforan los dos orificios de Ø3mm.



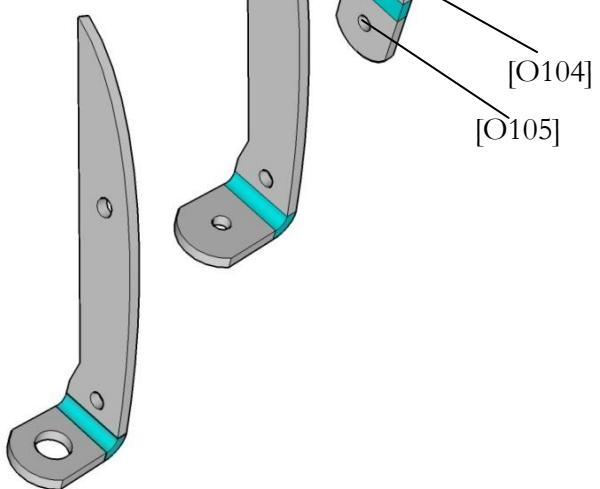
○ **P07 – Soporte Superior del Pié (aluminio 2mm):**

1- Corte de la preforma.



2- Perforación de los orificios de Ø3mm.

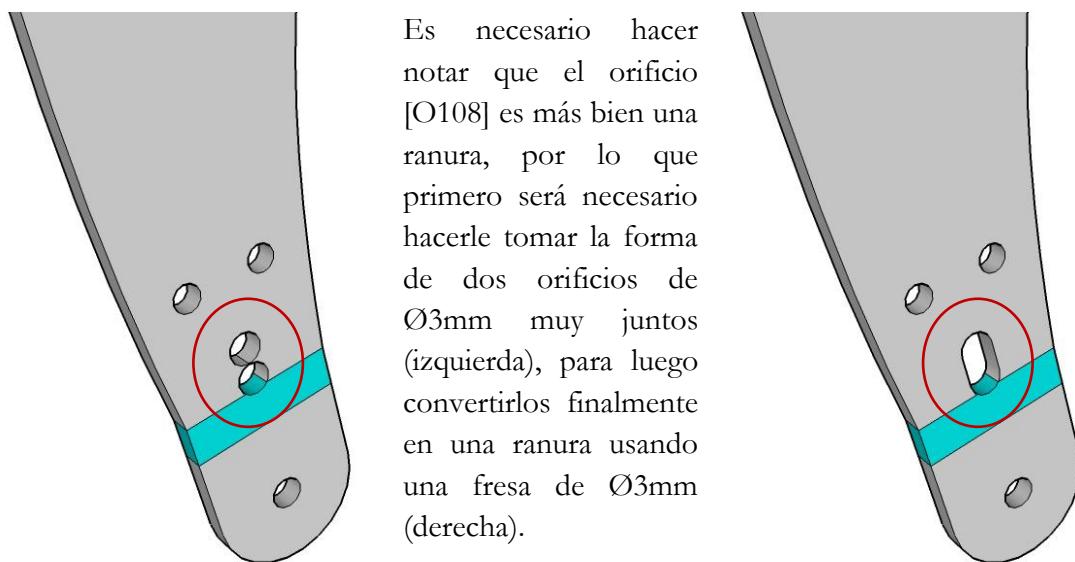
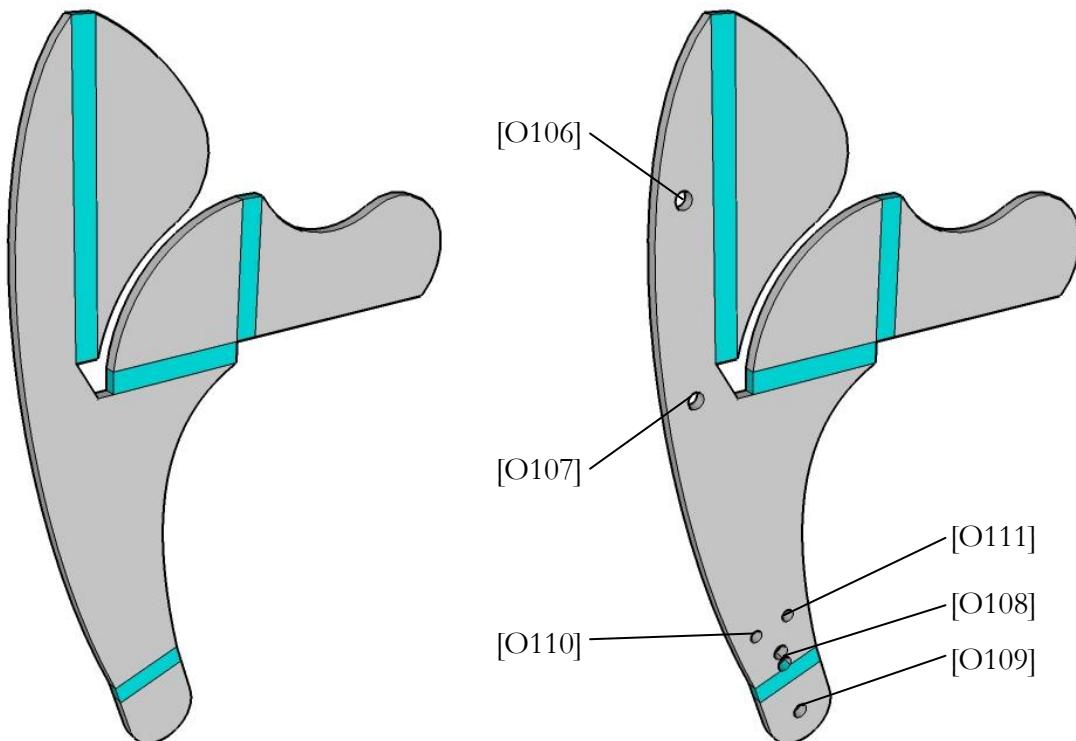
3- Doblado de la pieza.



4- Se agranda [O105] hasta los Ø6mm.

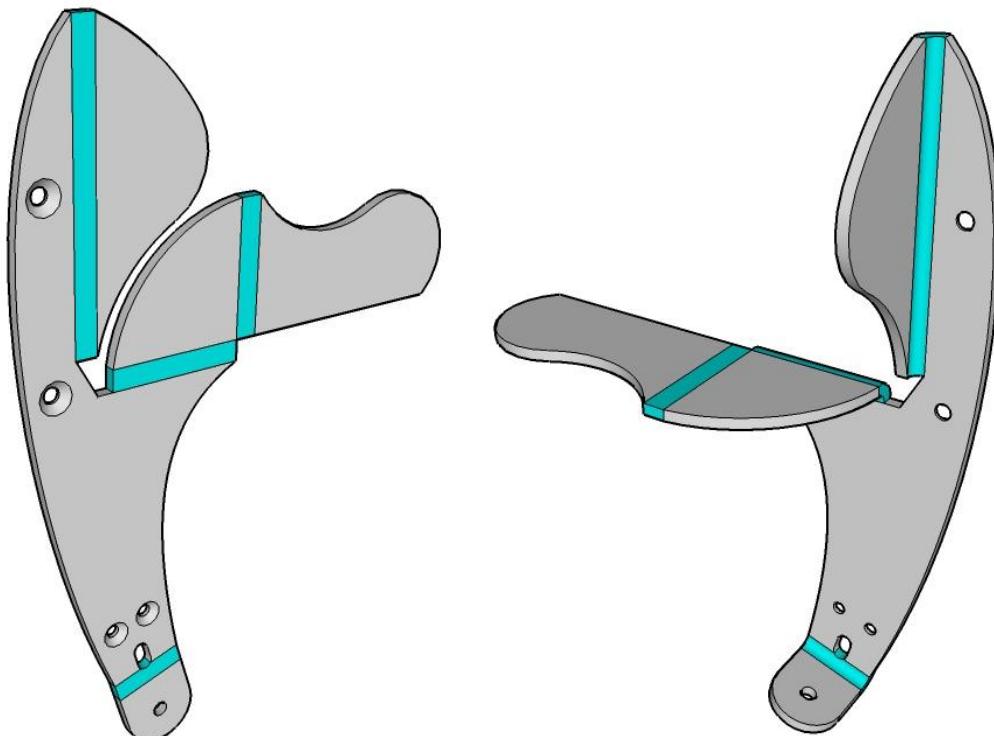
○ P08 – Parte Móvil de la Tibia (aluminio 2mm):

- 1- Corte de la preforma.
- 2- Perforación de los orificios de Ø3mm [O106 – O109], i de los de Ø2mm [O110 – O111].

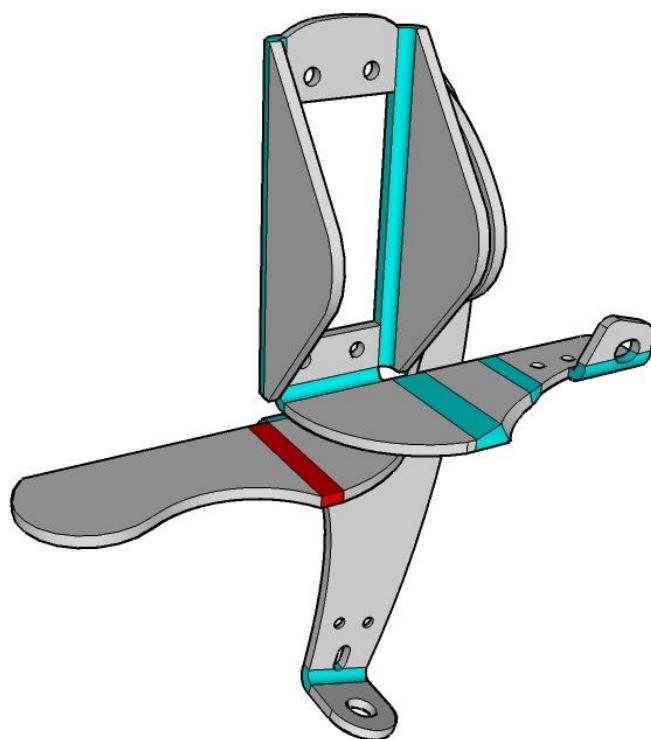


Es necesario hacer notar que el orificio [O108] es más bien una ranura, por lo que primero será necesario hacerle tomar la forma de dos orificios de Ø3mm muy juntos (izquierda), para luego convertirlos finalmente en una ranura usando una fresa de Ø3mm (derecha).

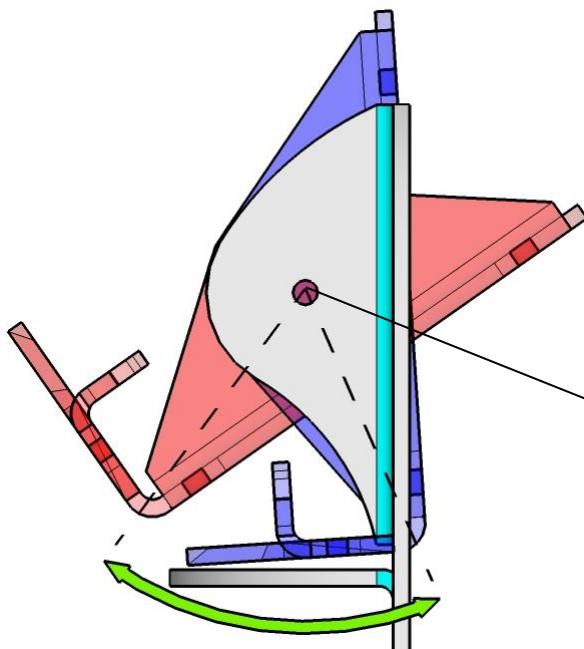
- 3- Avellanado de los orificios de Ø3mm [O106 – O107] y [O110 – O111]. 4- Primer doblado de la pieza (vista trasera).



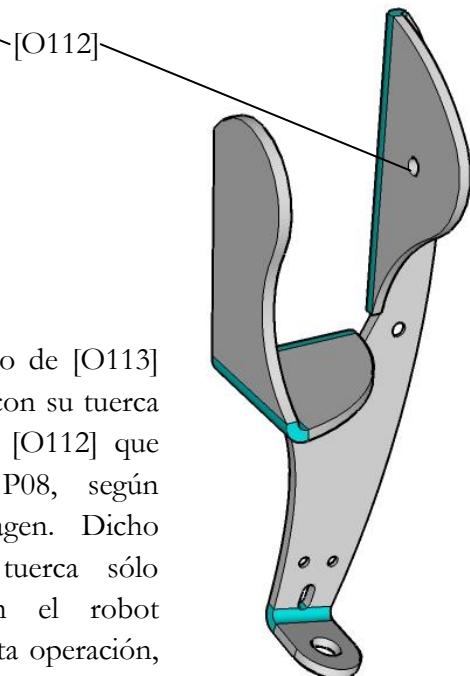
- 5- Se agranda [O109] hasta los Ø6mm.



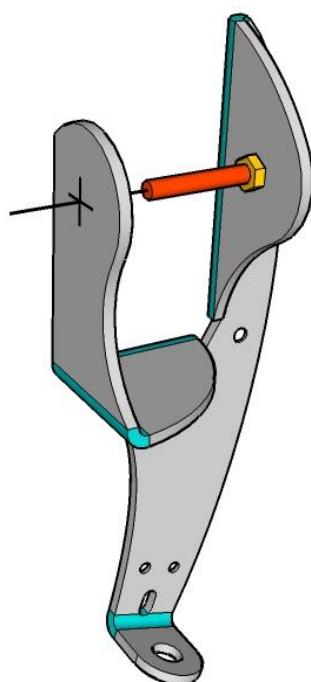
- 6- Este paso ocurre a la vez que se ejecuta la etapa 6 de fabricación de la pieza P06, sólo que en este caso la superposición de las piezas sirve para calcular la posición exacta de la franja de doblez marcada en rojo sobre P08.



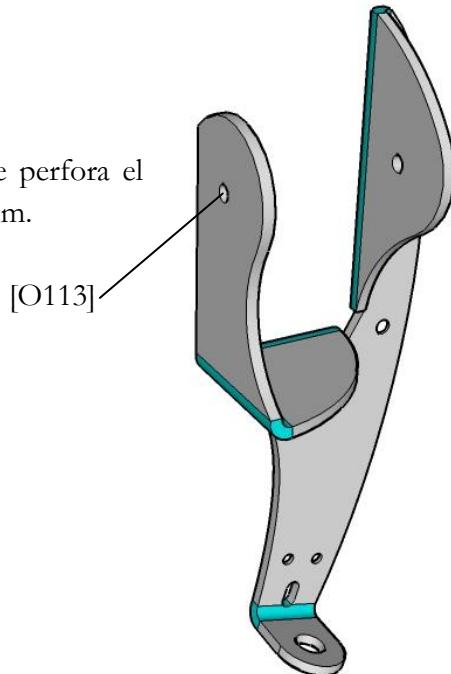
Además, aprovechando que durante la etapa 6 de la fabricación de P06 se calcula el alineamiento vertical de dicha pieza sobre P08 (izquierda), se marca y posteriormente se perfora en esta última el orificio de Ø3mm [O112]. Finalmente, se procede a doblar P08 (imagen inferior).



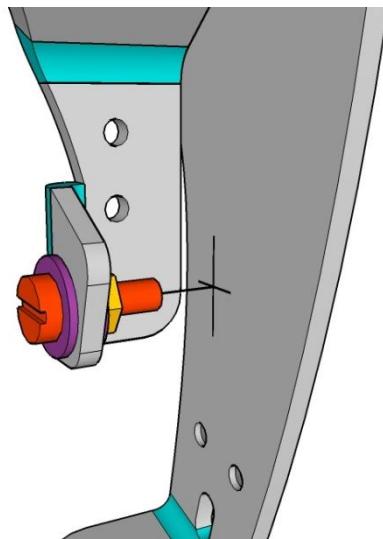
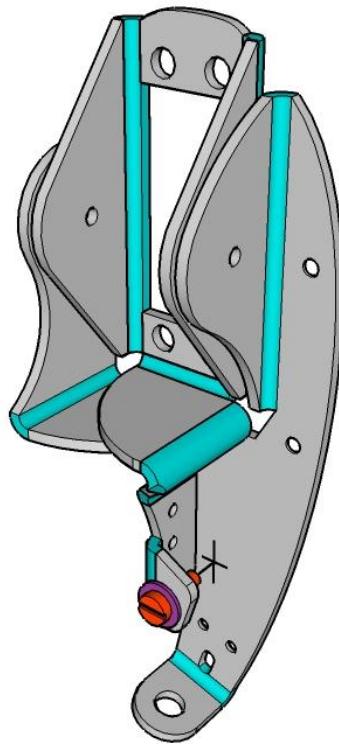
- 7- A continuación se alinea el emplazamiento de [O113] con el de [O112] usando un tornillo [T4] con su tuerca para proyectar una línea perpendicular a [O112] que interseca con P08, según muestra la imagen. Dicho tornillo y su tuerca sólo permanecerá en el robot mientras dure esta operación, por lo que no figura en el despiece.



- 8- Acto seguido se perfora el orificio de Ø3mm.

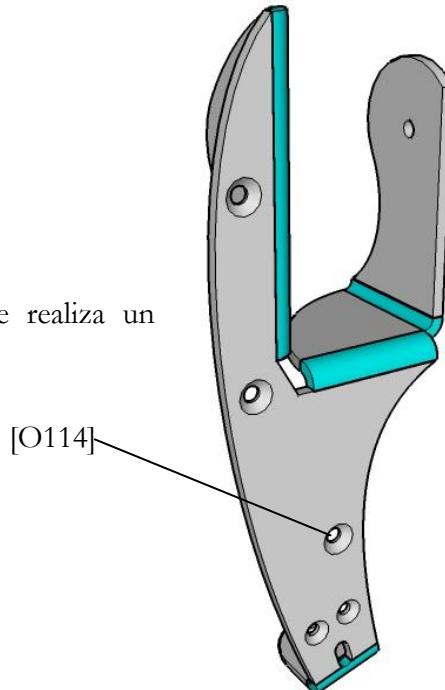


- 9- Finalmente, es necesario volver a interactuar con la pieza P06 para marcar la posición exacta del último orificio de P08. En esta ocasión hará falta que P06 esté completamente terminada, y que P06 y P08 queden fijadas la una a la otra en perfecta alineación de los orificios [O112 – O113] y [O101 – O102]. Se usa un tornillo [T3] con su tuerca y dos arandelas [A1], que de nuevo no figurarán en el despiece, para proyectar de la forma más centrada posible la perpendicular a [O100] sobre P08.



(Vista general del montaje arriba y detalle a la izquierda)

- 10- Se perfora el orificio de Ø3mm y se le realiza un posterior avellanado por su cara externa.

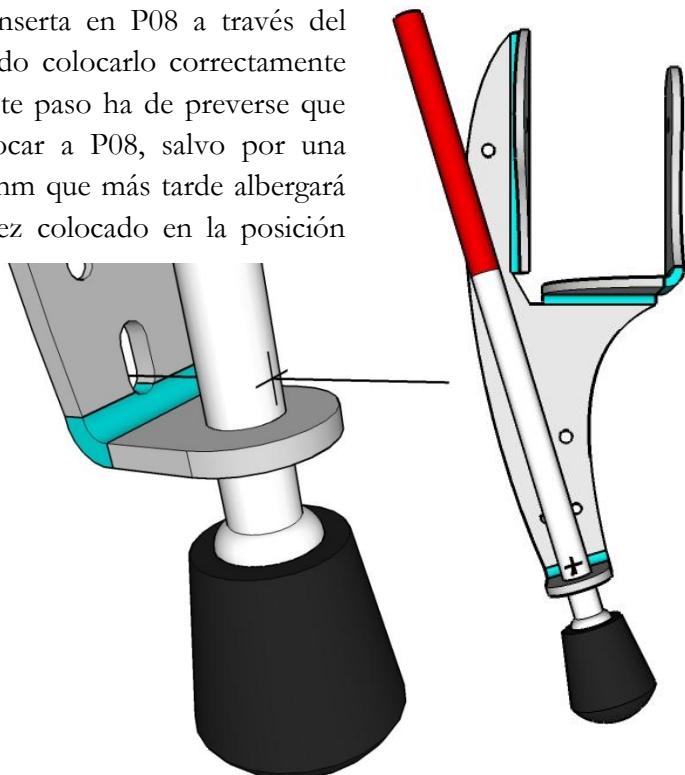


- P09 – Pié (Tubo de latón cromado de Ø6mm):

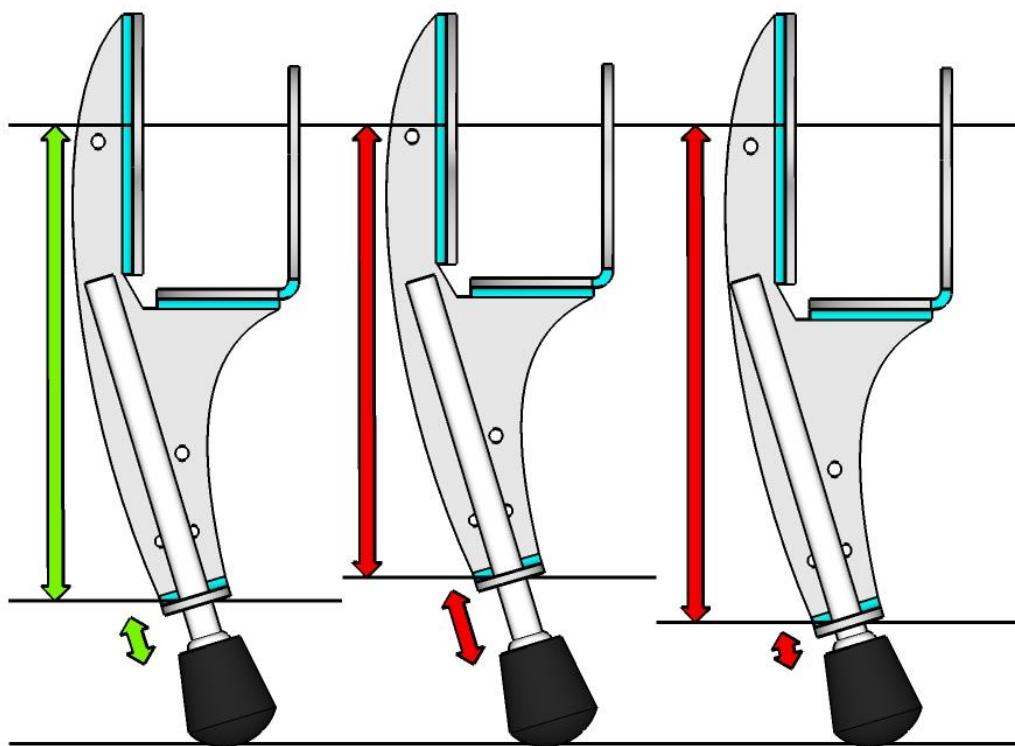
- 1- La parte principal del pié es un tubo de latón cromado de Ø6mm que termina en uno de sus extremos en una masa metálica, también de latón, en forma de bola. Este es el único componente del esqueleto del robot, a parte de la tornillería y otras piezas de unión, que no ha sido completamente moldeado a mano, sino que procede de un sencillo toallero extensible que comercializa la marca Ikea. Se inserta esta terminación en forma de bola metálica en el interior de un tajo hueco de goma antideslizante [E4] relleno de cola de epoxi, tras limar el metal para darle rugosidad y mejorar la adherencia del adhesivo. En las siguientes 12 horas el tajo de goma debe permanecer inmóvil para no quedar torcido al secarse la cola. Para conseguirlo los seis pies, correspondientes a las seis patas, se secan en una improvisada prensa que los sujetá en la posición deseada.



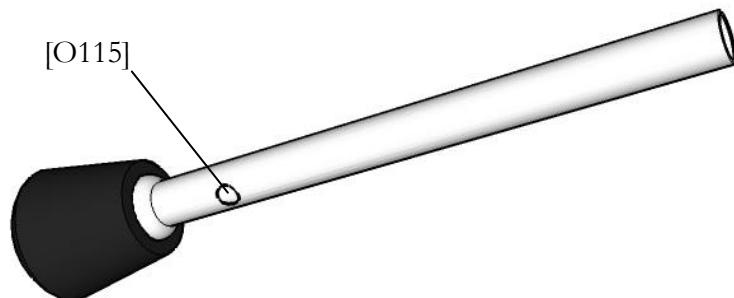
- 2- Tras el secado, P09 se inserta en P08 a través del orificio [O109], procurando colocarlo correctamente en sentido vertical. En este paso ha de preverse que [E4] no debe llegar a tocar a P08, salvo por una distancia de entre 5 y 10mm que más tarde albergará el muelle del pié. Una vez colocado en la posición correcta se marca el trozo superior sobrante del tubo (en rojo) y se determina la posición exacta de [O115], comparando la altura de la marca con el extremo inferior de la ranura [O108].



Este paso debe realizarse de forma conjunta en las seis Tibias a la vez. El motivo es que el orificio [O115] determinará la distancia con la que el pie sobresale de P08, y dicha distancia debe compensar los defectos de doblado de P08. Dicho de otra forma, las seis piezas resultantes de la fabricación de las seis P08 no serán idénticas, sino que en cada una habrá diferencias en la distancia que separa la aleta que contiene el orificio [O109] del eje que cruza los orificios [O112 – O113]. Estas diferencias deben ser compensadas por la disposición de los pies, ya que de lo contrario aquellas Tibias que hayan quedado más cortas podrían no tocar el suelo cuando el robot se hallara erguido. Esto se traduciría en una mala funcionalidad de las mismas y en una indeseable distribución del peso del robot sobre sus patas. Para solucionarlo, la posición correcta de todos los pies se marca a la vez manteniendo constante en todos ellos la distancia entre el eje [O112 – O113] y el extremo inferior del taco de goma [E4].

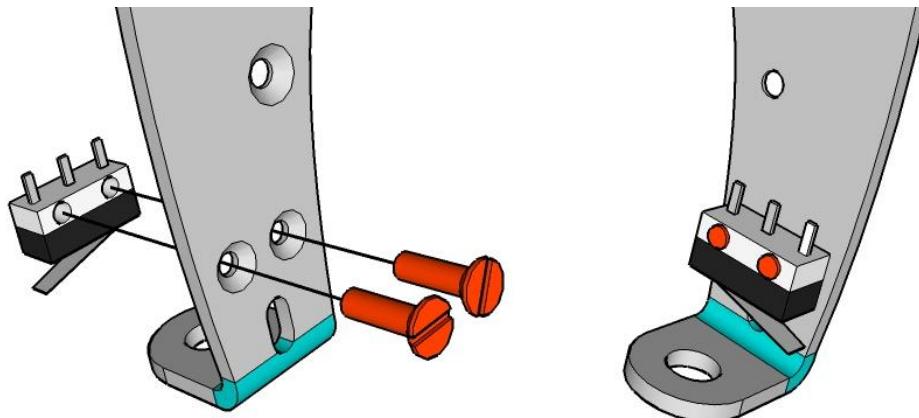


- 3- Finalmente se corta el sobrante del tubo con la cortadora de disco y se perfora el orificio de Ø2'5mm.

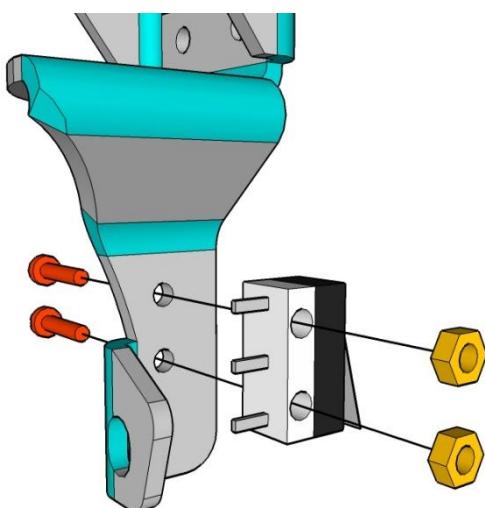
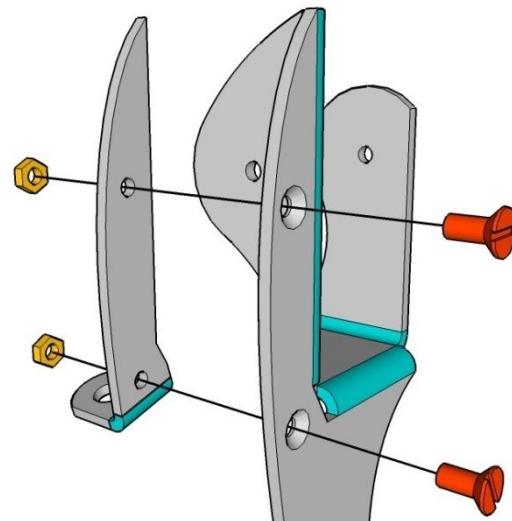


- Montaje de las Tibias:

- 1- El montaje empieza con la instalación del sensor de contacto [E3] en la pieza P08 mediante dos tornillos [T6] y a través de los orificios [O110 – O111]. Estos tornillos no usan tuerca, ya que roscan a presión en los orificios de la estructura de plástico de [E3], pero el tornillo sobrante debe ser cortado al ras para no obstaculizar la inserción del pié más adelante.



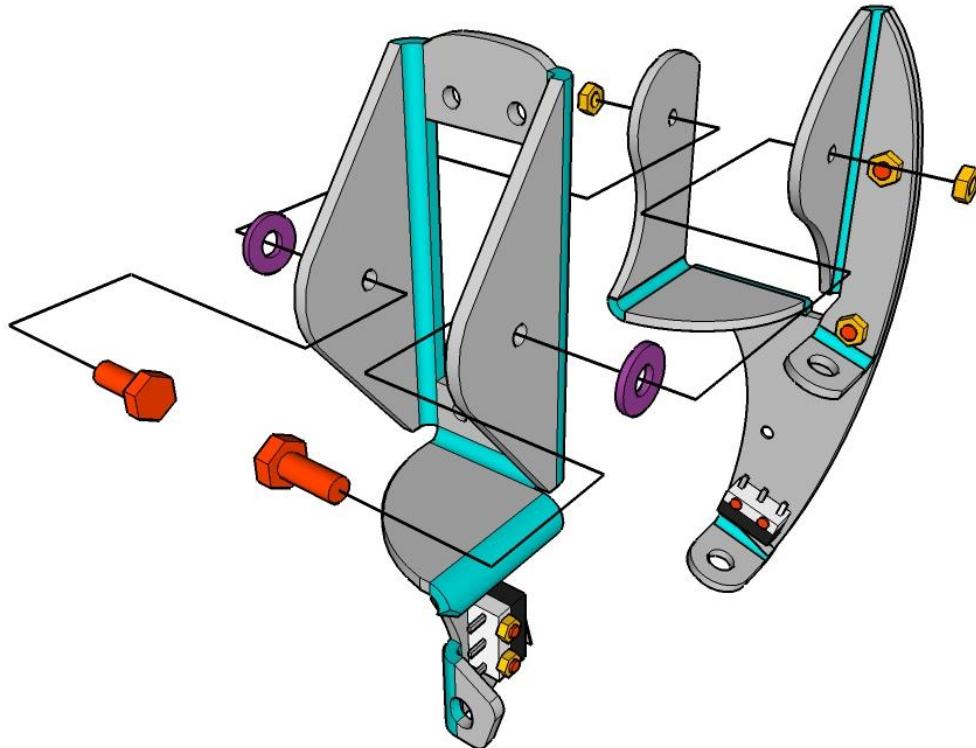
- 2- Usando dos tornillos [T7] con sus tuercas se acopla la pieza P07 a P08 a través de los orificios [O106 – O107] y [O103 – O104].



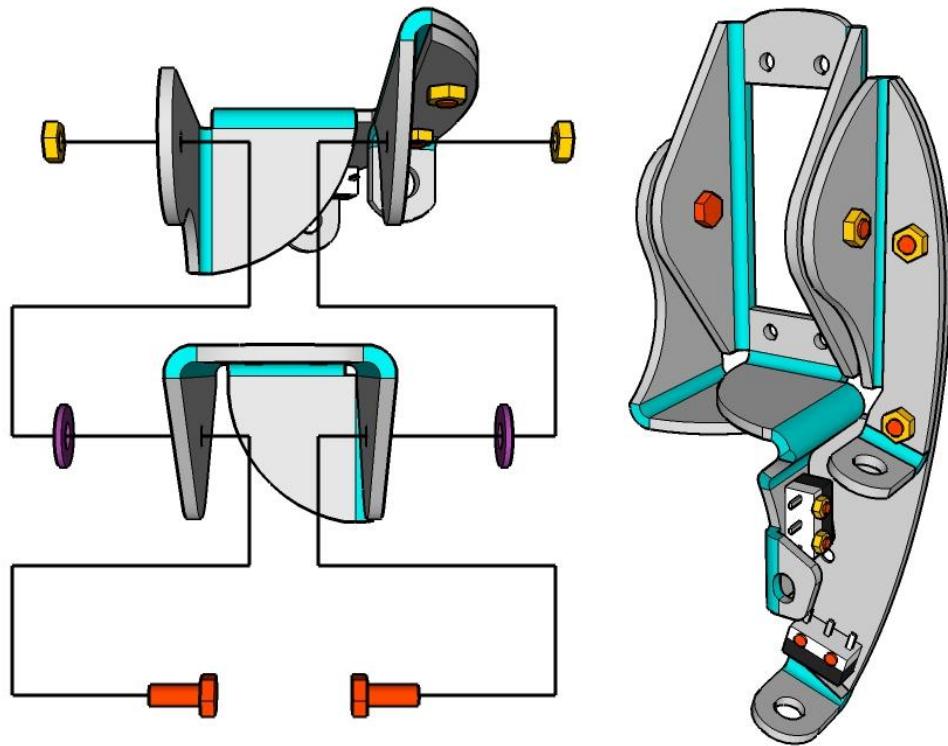
- 3- Por otro lado, se usan dos tornillos [T1] con sus tuercas para fijar un sensor [E3] a la pieza P06, a través de los orificios [O98 – O99].

4- Llegados a este punto, las piezas P08 (+P07) y P06 ya están listas para unirse. Dicha unión se realiza mediante dos tornillos [T9] que se introducen de dentro a fuera, y cuando las piezas P06 y P08 están alineadas en su posición correcta, a través de los orificios [O101 – O102] de P06, y que a su vez atraviesan dos arandelas [A1], que hacen de separador entre P06 y P08, para finalmente atravesar, también de dentro a fuera, los orificios [O112 – O113] de P08. Por último dos tuercas cierran la unión por ambos lados, pero no se aprietan en absoluto, puesto que cualquier presión en el eje de giro de estas dos piezas puede producir un rozamiento que dificultaría su movimiento. Ha de tenerse en cuenta que la pieza P08 pendula sobre P06 para presionar sobre el sensor de contacto instalado en esta última cuando la trayectoria de la pata interfiere con un obstáculo. Así pues, el sensor debe activarse con una presión mínima, por lo que la dureza de su retorno elástico también será mínima, de manera que P08 no retornará a su posición inicial si su unión con P06 no ofrece una resistencia mecánica despreciable.

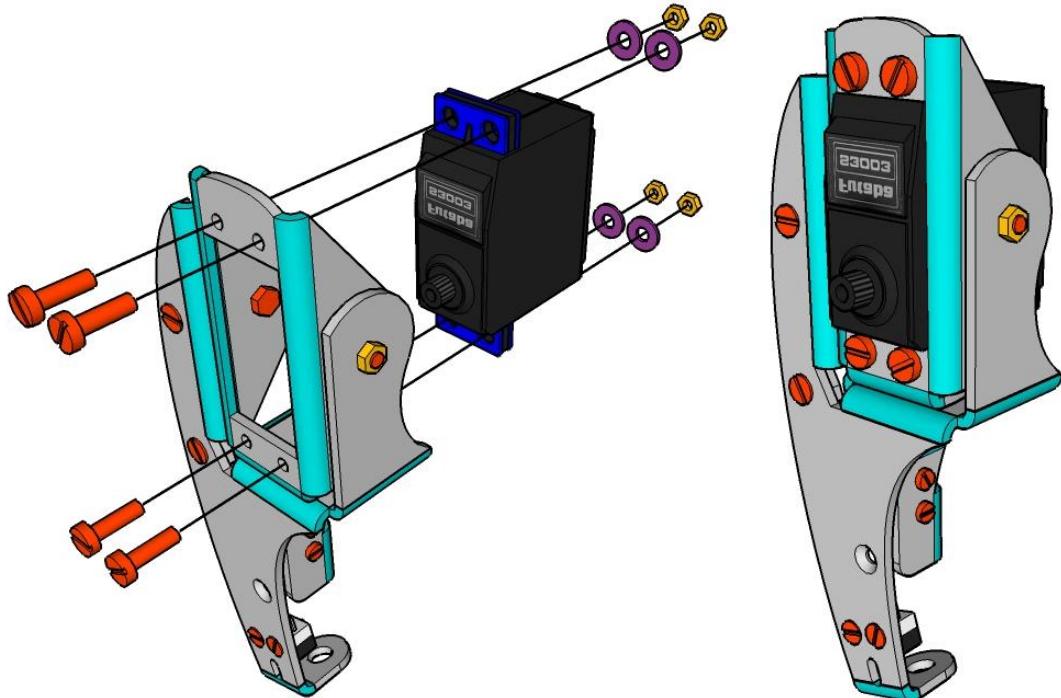
No obstante, al estar tan flojas las tuercas de los [T9], éstas podrían desenroscarse y caerse a causa de las vibraciones inherentes al funcionamiento de los motores y del movimiento del robot, por lo que tuerca y tornillo deben ir fijados al esqueleto con cola de epoxi. Dicha cola se aplica de forma normal a la cabeza de cada tornillo [T9] para unirla a P06, pero debe aplicarse con mucho cuidado en el lado de la tuerca, pues debe fijarla a P08 por su contorno, pero sin entrar en contacto en ningún momento con el tornillo. Si no se respeta esta premisa, tuerca y tornillo quedarán completamente unidos, y la unión será inservible.



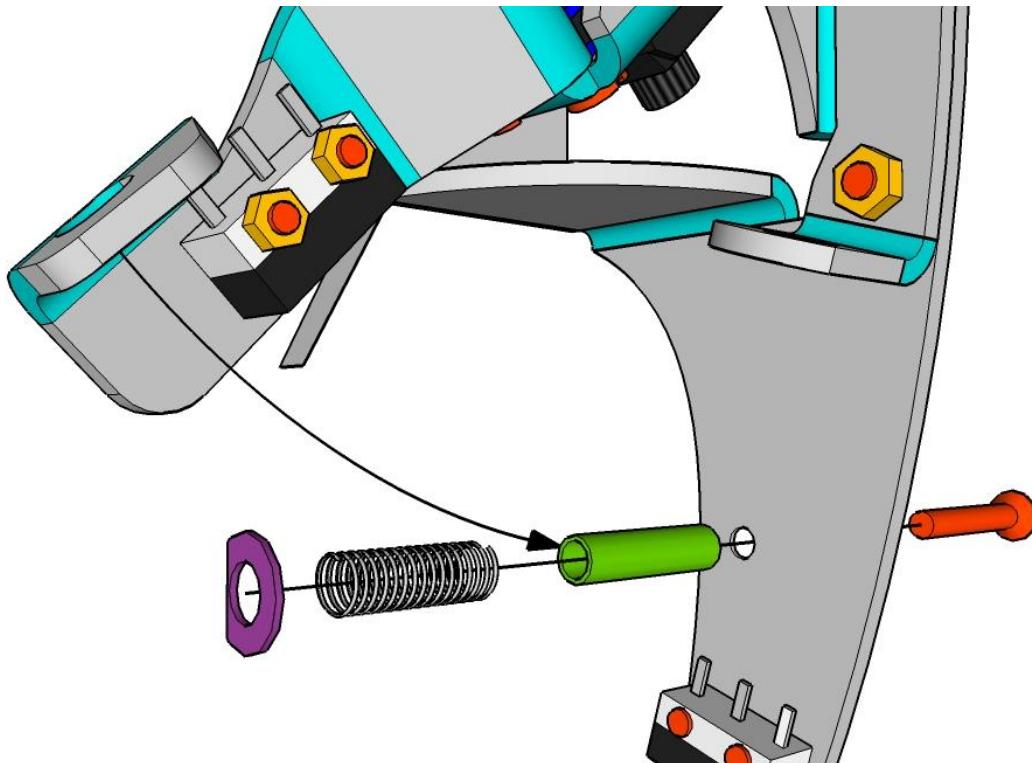
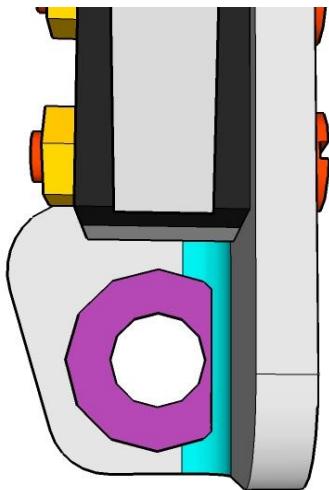
(Vista superior y aspecto terminado del montaje).



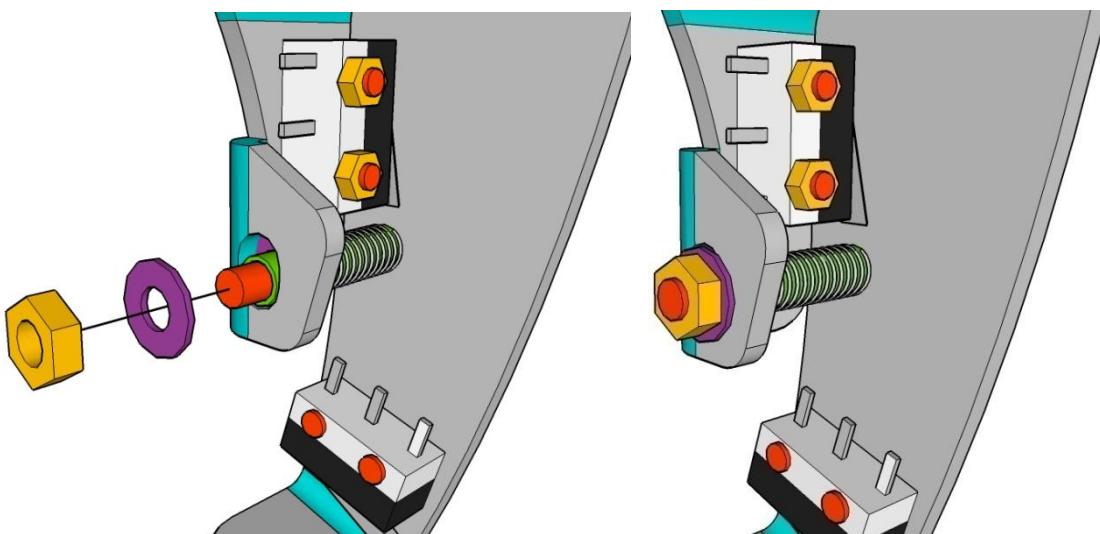
- 5- Una vez acopladas las dos piezas y comprobado el buen funcionamiento del eje de movimiento, ya no será necesario operar en el espacio interior de P06, por lo que puede acoplarse el servo [S3], que ocupará ese mismo espacio. Para hacerlo se usan cuatro tornillos [T4] con sus tuercas, y cuatro arandelas [A1] para sujetar el servo a P06 a través de los orificios [O94 – O97].



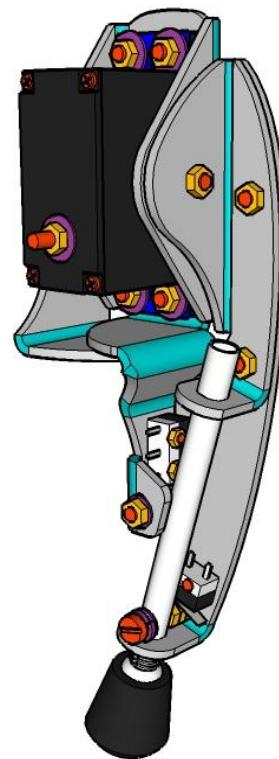
- 6- A continuación se procede a instalar el retorno elástico del sensor de obstáculo. Este retorno elástico es un conjunto de seis piezas pequeñas que consiguen que P08 retorne a su posición inicial cuando la pata deja de hacer presión contra un obstáculo, y además hacen de tope a P08, evitando que pendule libremente sobre P06 más allá de su posición deseada. Su instalación se lleva a cabo en dos pasos: en este primer paso la aleta inferior de P06 se separa de P08 y se hace pasar un tornillo [T8] a través de [O114], que se enfunda con un separador [Y2] desde el otro lado de P08. Sobre [Y2] se coloca un muelle [E6] que se encierra con una arandela [A1] modificada. Dicha arandela (imagen de la izquierda) tiene un cierto arco de su circunferencia recortado para encajar suavemente con la forma de la aleta inferior de P06, y además su diámetro inferior encaja casi perfectamente con el diámetro exterior del tubo de aluminio que hace de separador, de modo que cuando desciende la aleta inferior de P06, la arandela puede moverse a lo largo de [Y2], comprimiendo a [E6] mientras que [Y2] y el sobrante de [T8] pasan libremente a través de [O100].



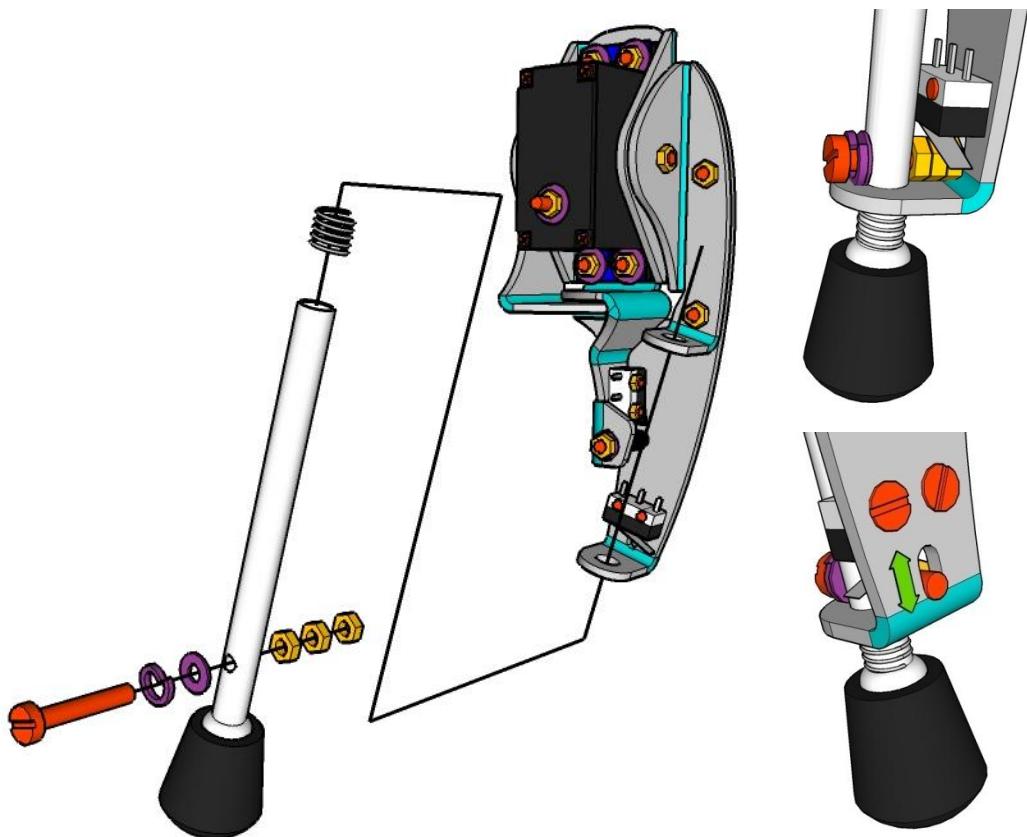
- 7- El segundo paso sirve para evitar que la aleta inferior de P06 pueda volver a separarse de P08, de modo que se fija el retorno elástico con la tuerca de [T8], atrapando a la vez a una arandela [A0] entre la tuerca y el borde del separador [Y2]. La arandela [A0] tiene un diámetro interior inferior al de [Y2], por lo que éste no puede atravesarla, pero a la vez tiene un diámetro exterior superior al de [O100], por lo que el orificio tampoco puede superar la arandela y la pieza queda atrapada entre un tope y un muelle. Finalmente, [T8], [Y2] y [A0] quedan fijados a P08 comprimidos por la tuerca, mientras que [E6], [A1] y P06 pueden desplazarse por [Y2], limitados por [A0] por un lado y por la compresión de [E6] por el otro.



- 8- Finalmente, el montaje de la Tibia termina con la instalación del pié P09 (vista de la Tibia terminada a la derecha). Para hacerlo se hace pasar el extremo abierto del tubo de P09 por el interior de un muelle [E5] hasta que éste descance sobre la base del tubo, introducido en la boca del taco de goma. A continuación se inserta P09 por el mismo extremo a través de los orificios [O109] y [O105] de la Tibia, y se comprime [E5] contra [O109] hasta que [O115] se alinee con [O108]. Entonces se introduce un tornillo [T2] equipado con una arandela de presión [A2] y una arandela [A0] a través de [O115] y de tres de sus tuercas, para terminar introduciéndose en [O108]. De este modo, [O108] permite un ascenso y descenso de [T2] suficiente como para que sus tuercas puedan apretar la lengüeta del sensor [E3] cuando la pata proyecta a P09 contra el suelo. Además, [T2] no puede sellarse con cola de epoxi, puesto que es necesario desmontarlo para poder extraer P09 cuando sea necesario realizar reparaciones o sustituciones de

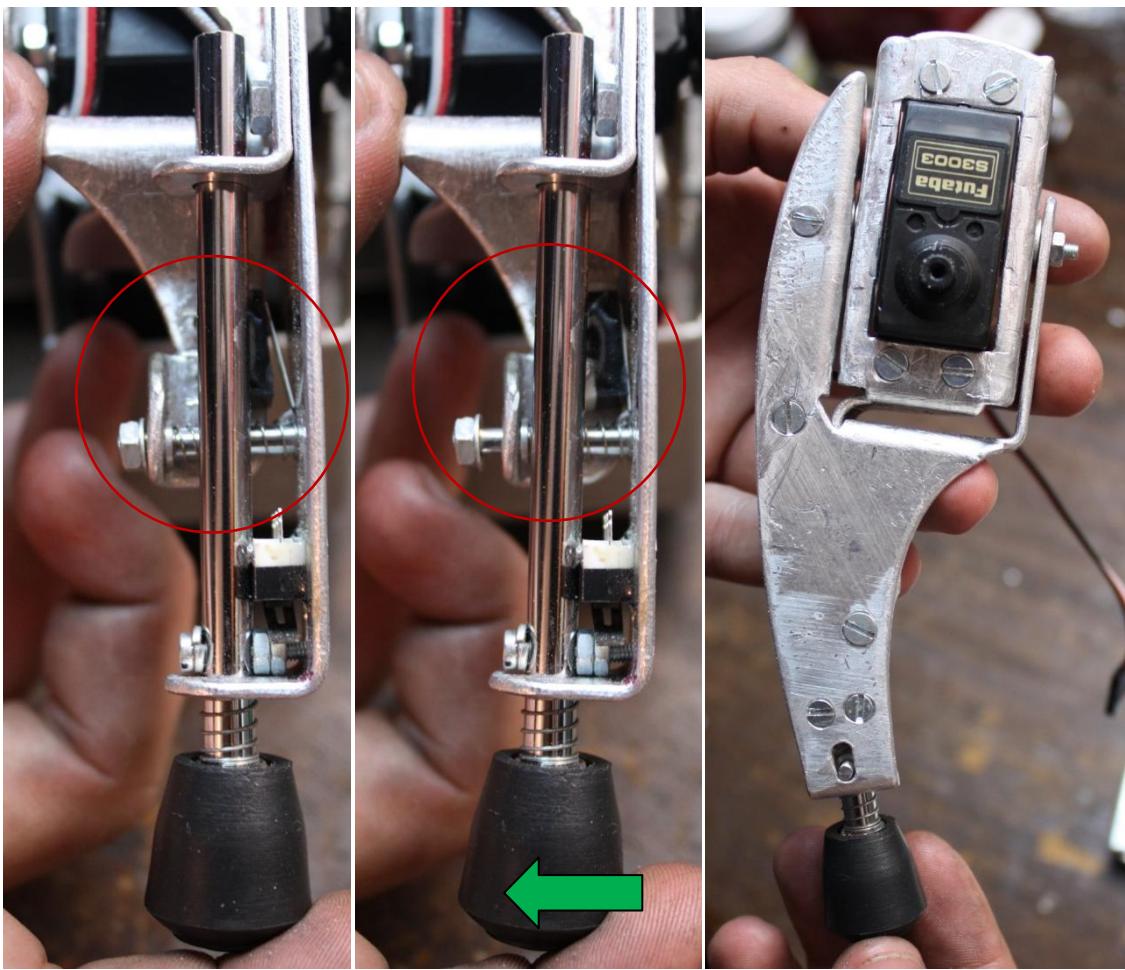


piezas en la Tibia. Por eso se coloca una arandela de presión en [A2] que mantendrá las tuercas siempre apretadas hasta que se desee aflojarlas.



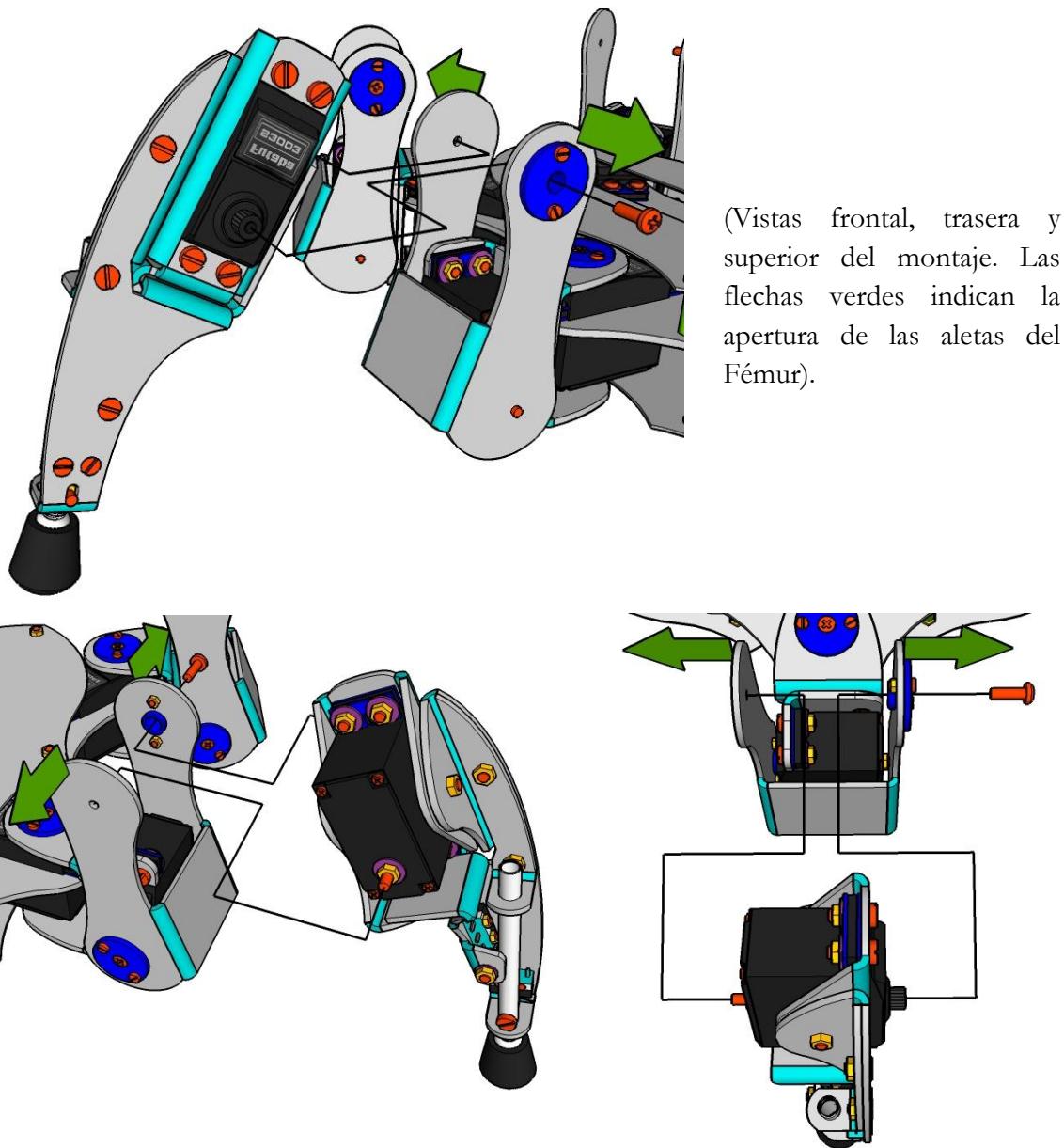
Hecho esto las Tibias están terminadas, y sólo queda ver si funcionan correctamente comprobando que los sensores dan señal y que los retornos elásticos confieren a las partes móviles de la Tibia un movimiento suave pero firme. Las siguientes imágenes muestran el acabado real de la Tibia y el funcionamiento de sus dos sensores puesto a prueba.





- Montaje de las Tibias + Patas + Tórax:

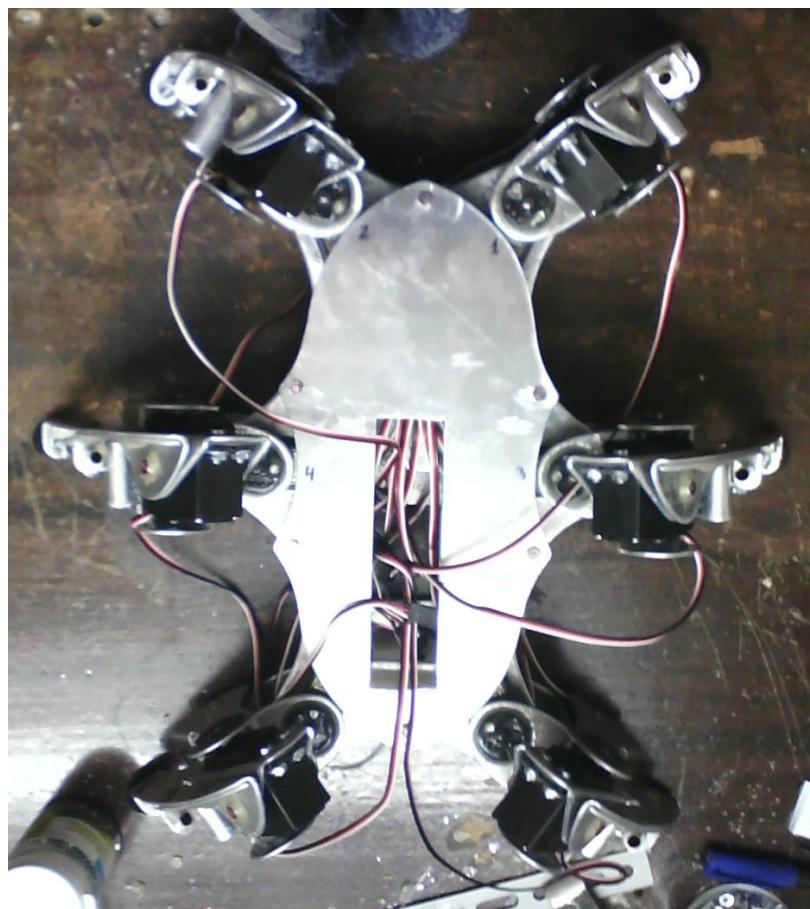
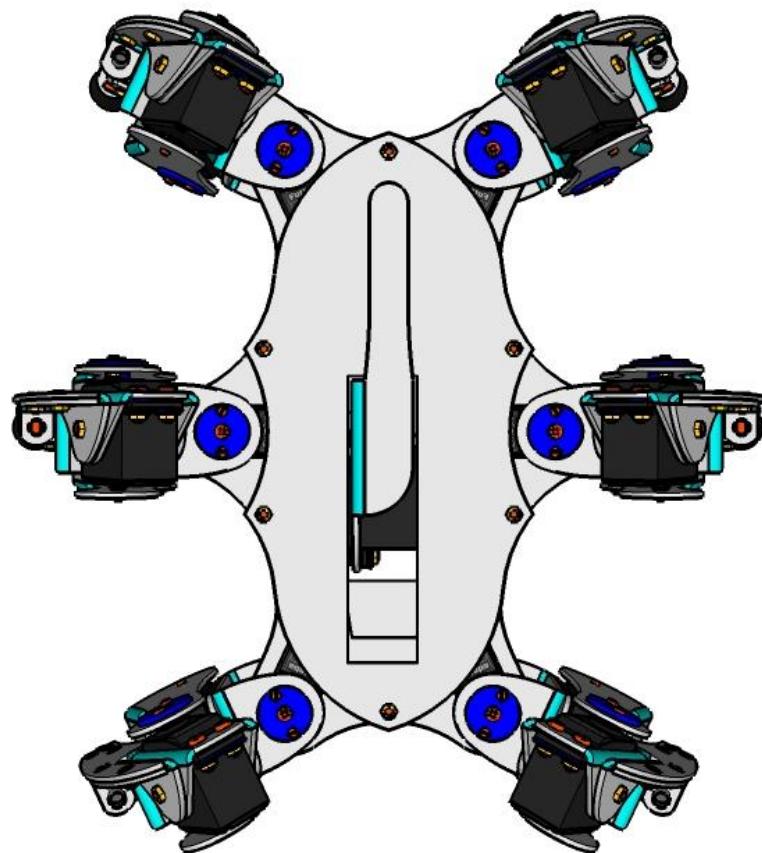
- 1- Al finalizar esta etapa de montaje el esqueleto del robot estará completamente terminado a excepción de la Cabeza. Para acoplar las Tibias al conjunto Tórax + Cadera + Fémur habrá que forzar ligeramente las aletas del Fémur a separarse para albergar el servo [S3] de la Tibia. Este proceso ha de realizarse con moderación, ya que el aluminio tiene un coeficiente de elasticidad muy bajo, y separar los extremos de esta pieza en exceso podría provocar su deformación, que por otro lado no se podría corregir una vez instalada la Tibia. El eje trasero de [S3] se introduce primero en [O86] desde una trayectoria diagonal y después el eje motriz se acopla a la pieza [E1] instalada previamente en el orificio [O88]. Hecho esto [S3], y por tanto la Tibia entera, quedan encerrados en el Fémur. Por último se asegura la unión apretando un tornillo [T0] al eje motriz de [S3].

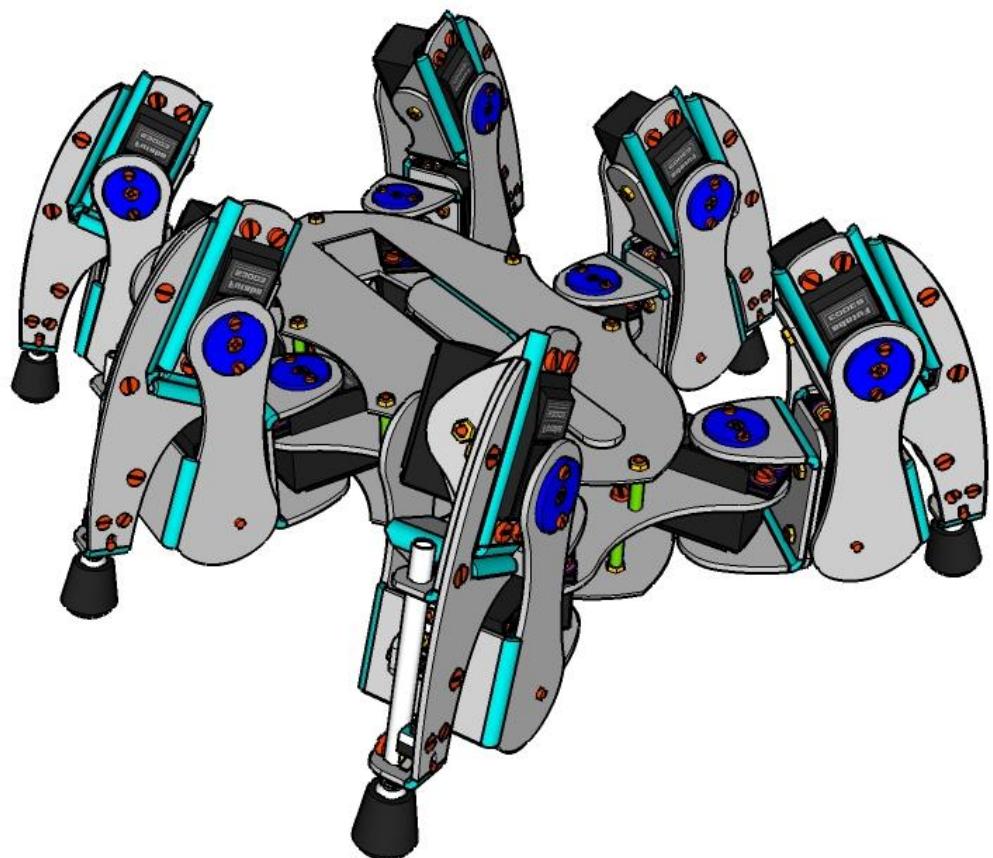


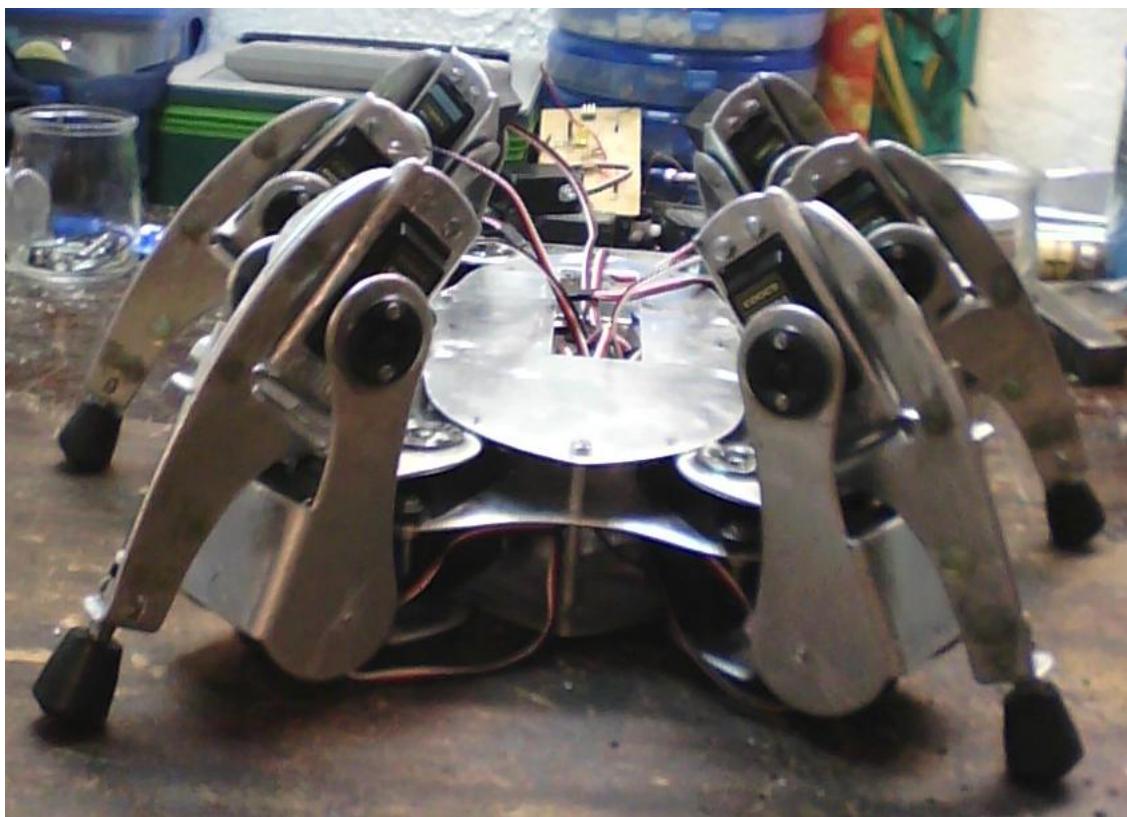
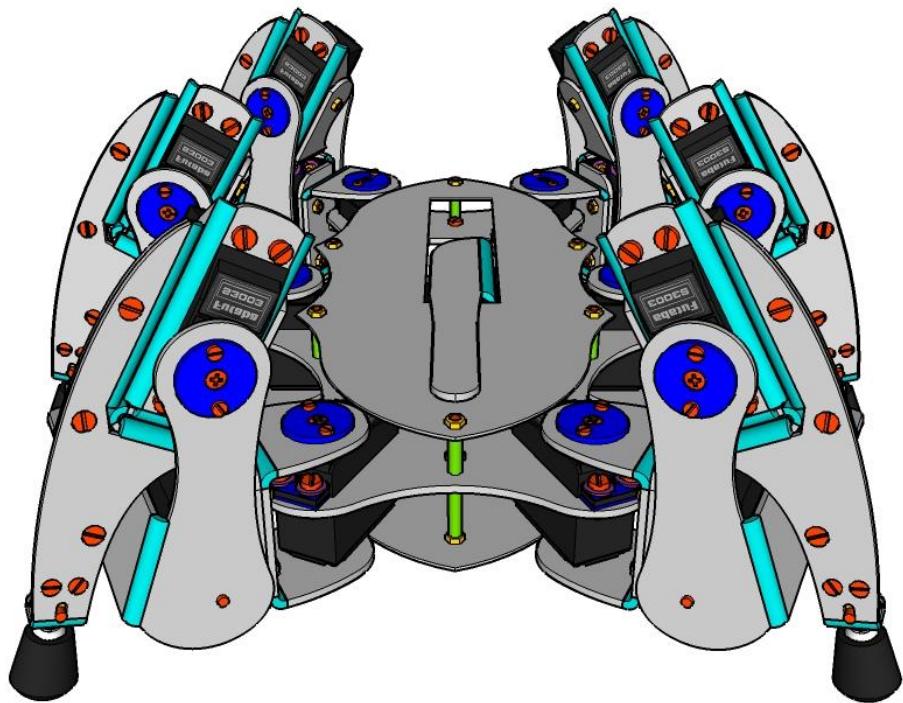
El proceso se repite con las seis patas hasta obtener el esqueleto completo (a excepción de la cabeza). Las siguientes imágenes muestran su aspecto final desde varios ángulos.

Además, llegados a este punto resulta interesante comparar el modelo virtual con el esqueleto real para comprobar que se han cumplido todas las especificaciones de diseño, por eso cada vista renderizada del robot va acompañada de su fotografía análoga. Las imágenes hablan por sí solas: el diseño original se ha recreado de forma satisfactoria, todas las Tibias son perfectamente funcionales y las articulaciones cubren todo el rango de movimiento para el que se diseñaron.

(Hay que apuntar que, como se ha apuntado al inicio de este capítulo, el cuello del robot finalmente no se montará en esta etapa del proyecto, por lo que en las fotografías que siguen el Cuello aún no ha sido instalado en el robot, aunque el diseño lo contemple).







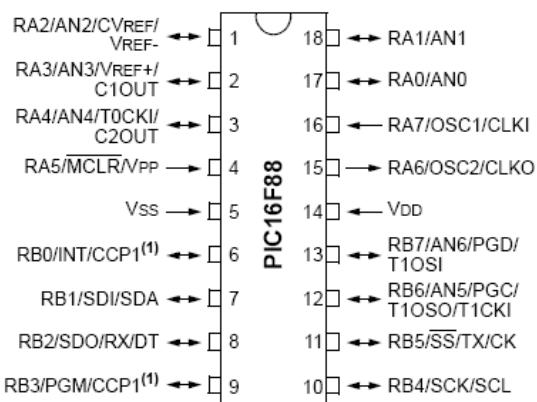
- Recuento de piezas de las 6 Tibias (Total 348 componentes: 24 piezas, 6 servos, 54 elementos blandos, 6 separadores, 72 arandelas, 96 tornillos y 90 tuercas):
  - P06: 6 Partes Fijas de la Tibia
  - P07: 6 Soportes Superiores del Pié
  - P08: 6 Partes Móviles de la Tibia
  - P09: 6 Pies
  - [S3]: 6 Servos “Tibia”
  - [E0]: 24 Protectores de goma para servo Futaba
  - [E3]: 12 Sensores de contacto de tipo Mini-Bumper
  - [E4]: 6 Topes huecos de goma antideslizante
  - [E5]: 6 Muelles de Ø6'5mm
  - [E6]: 6 Muelles de Ø4'5mm
  - [Y2]: 6 Separadores de tubo de aluminio de Ø4mm
  - [A0]: 12 Arandelas normales M2'5
  - [A1]: 54 Arandelas normales M3
  - [A2]: 6 Arandelas de presión M2'5
  - [T0]: 6 Tornillos
  - [T1]: 12 Tornillos y 12 tuercas
  - [T2]: 6 Tornillos y 18 tuercas
  - [T4]: 24 Tornillos y 24 tuercas
  - [T6]: 12 Tornillos
  - [T7]: 18 Tornillos y 18 tuercas
  - [T8]: 6 Tornillos y 6 tuercas
  - [T9]: 12 Tornillos y 12 tuercas

## 5 Diseño electrónico

### 5.1 PIC16f88

El microcontrolador escogido para este proyecto es uno de los más versátiles de entre los integrados pequeños de la familia 16f de Microchip. Entre sus características más importantes destacan:

- Integrado PDIP de 18 pines
- 16 pines de Entrada/Salida genéricos
- 7 Kbytes de memoria Flash de programa (4096 instrucciones de un Word)
- 368 bytes de memoria SRAM
- 256 vbytes de memoria EEPROM
- Modo oscilador HS de hasta 20 MHz
- 7 Canales de conversión ADC de 10 bits
- Módulo SSP con soporte hardware para I<sup>2</sup>C Slave y SPI
- Módulo AUSART (en operación RS-232 usa un oscilador interno)
- Dos módulos Timer de 8 bits con disparo de interrupción
- Un módulo Timer de 16 bits con disparo de interrupción
- Módulo Capture/Compare/PWM de 16 y 10 bits
- Dos fuentes de interrupción externa: RB0 y cambio en el Puerto B[4:7]
- In-Circuit Serial Programming (ICSP)
- 100 000 ciclos típicos de lectura/escritura en la memoria Flash
- 1 000 000 ciclos típicos de lectura/escritura en la memoria EEPROM
- Set de sólo 35 instrucciones RISC



**Note 1:** The CCP1 pin is determined by the CCPMX bit in Configuration Word 1 register.

**Figura F. 32: Patillaje del PIC16f88**  
(Fuente: PIC16f88 DataSheet)

La cantidad de funciones que puede realizar cada patilla, expresada en su diagrama de conexionado, pone de manifiesto la aglomeración de sistemas que agrupa este integrado. Las tres placas electrónicas que se desarrollan en este proyecto hacen uso de todas y cada una de estas patillas, así como de las principales funciones de sus módulos SSP y Timer1, y de sus dos fuentes de interrupción externa, además de varias de las internas, entre otros.

## 5.2 Placa “Patas” grupo A

### 5.2.1 Características

La controladora de cada pata es una pequeña placa electrónica de doble cara de no más de 12'6 cm<sup>2</sup> (45mm x 28mm) que ofrece las siguientes características:

- Conector de alimentación de 3 pines: V<sub>SS</sub>, V<sub>DD</sub>, y V<sub>SERVO</sub>, que corresponden a 0V, 5V y 6V (típicamente, aunque puede variar).
- Tres conectores de tres pines (V<sub>SS</sub>, V<sub>SERVO</sub> y PWM) para los servos compatibles con el conector Futaba.
- 3 LEDs indicadores: detección del suelo, detección de obstáculo y estado de ON/no error.
- PIC16f88.
- Xtal oscilador de 20MHz.
- Botón de RESET.
- Conector principal de 16 pines mediante el que se accede a las patillas de programación, alimentación, sensores, comunicación I<sup>2</sup>C y expansión para sensor de infrarrojos de cada pata.

El mapa del circuito electrónico de esta placa puede consultarse en el anexo de Planos.

La placa “Patas” grupo A está diseñada para conectarse a las patas 1, 3 y 6. Esto se debe a que la estructura mecánica del cuerpo del robot es simétrica en dos ejes, por lo que las patas 1, 3 y 5 son simétricas a las 2, 4 y 6, del mismo modo que las patas 1 y 2 son simétricas a las 5 y 6. De esta manera, 2 y 4, y 1 y 3 son simétricas entre sí de forma normal, mientras que 5 y 6 lo son de forma inversa. Así pues, para que las placas puedan encajar con la estructura del robot, deberá haber un grupo A de placas diseñado para tres de las patas, y otro grupo B diseñado para las patas simétricas a las primeras.

La placa está diseñada para instalarse detrás de los servomotores acoplados al Tórax Base del robot, en el espacio entre esta pieza y la Tapa Inferior y orientando su conector principal hacia la Tapa Inferior. Así mismo, la placa tiene todos sus componentes electrónicos soldados a la cara TOP de su serigrafía, excepto los conectores de alimentación y servos, los LEDs y el botón de RESET, todos ellos expresamente situados en la cara BOTTOM y en las bandas laterales de la placa. De este modo, la placa ofrece una zona llana donde sólo hay soldaduras, justo encima de su hendidura inferior de 3 x 20mm, de manera que puede usarse esta área despejada para encolar la placa a la pared trasera del servo acoplado al Tórax Base. Con esto se consigue que todos los componentes electrónicos que forman la placa estén orientados hacia el interior del robot, excepto los elementos mencionados, que estarán orientados hacia el exterior, o hacia la pata. De esta forma, podrá consultarse el estado de los LEDs, apretar el botón de RESET o conectar la alimentación y los servos a la placa desde el exterior del robot.

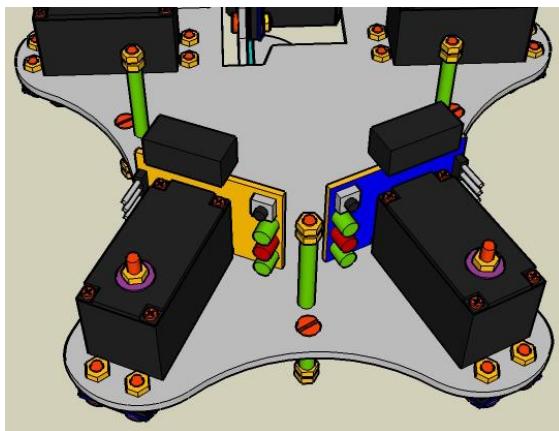


Figura F. 33: Vista exterior de dos placas simétricas  
(Fuente: Proyecto MIHRO)

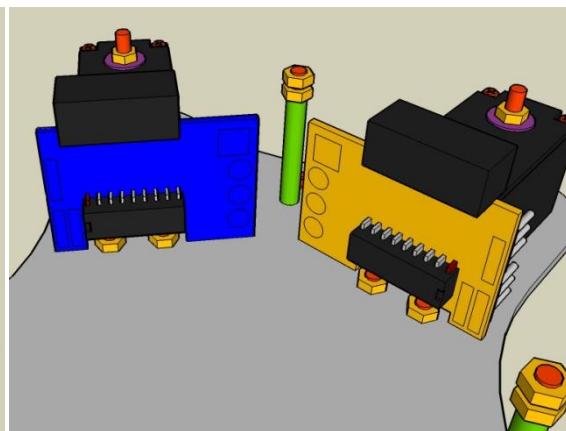


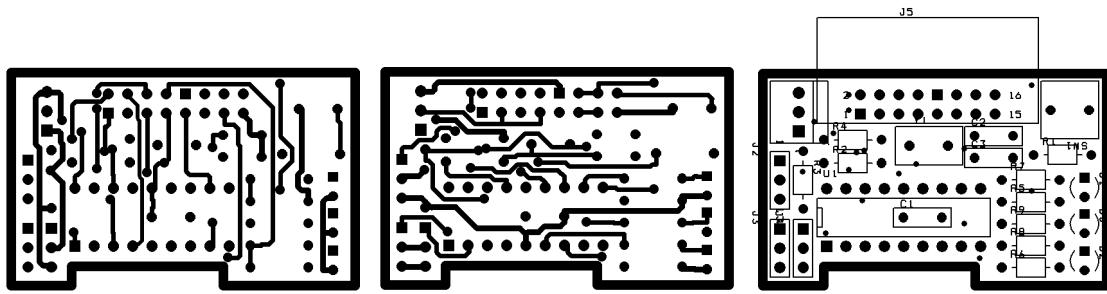
Figura F. 34: Vista interior de dos placas simétricas  
(Fuente: Proyecto MIHRO)

### 5.2.2 Lista de componentes

- Placa de cobre sobre fibra de vidrio de doble cara y de 50 micras de espesor.
- PIC 16f88 en encapsulado PDIP18
- Zócalo PDIP 18 patillas
- Resistencia 0'25W: 1KΩ (x3)
- Resistencia 0'25W: 10KΩ (x3)
- Resistencia 0'25W: 47KΩ
- Resistencia 0'25W: 220Ω
- Resistencia 0'25W: 330Ω
- Condensador cerámico 15pF (x2)
- Condensador cerámico 100nF
- Xtal oscilador 20MHz
- LED verde 3mm (x2)
- LED rojo 3mm
- Pulsador mini normal
- Conector hembra doble fila 2'54mm para cable plano 16pin
- Conector macho doble fila 2'54mm acodado 16pin
- Conector macho poste 2,54mm 3pin
- Conector hembra 2,54mm 3pin
- Conector en tira macho 2'54mm bajo perfil 20pin (sólo se usan 9 y en grupos de 3)
- Cable plano 16 hilos
- Cable rojo 0'4mm
- Cable negro 0'7mm
- Cable naranja 0'7mm

### 5.2.3 Serigrafías TOP y BOTTOM

El circuito de la placa “Patas” ha sido diseñado usando el programa OrCAD Capture, incluido en el paquete de software OrCAD 16.0. La serigrafía de la placa “Patas” ha sido diseñada mediante el software OrCAD Layout, incluido en el mismo paquete. La serigrafía de esta pata está diseñada para estamparse sobre una placa de cobre de doble cara y de 50 micras de espesor. Todas las pistas tienen un ancho constante de 0'6 mm, excepto las pistas de  $V_{SS}$  y  $V_{SERVO}$ , que tienen un ancho de 0,8 mm. Todos los orificios tienen un diámetro de 0'8 mm y disponen de un *pad* de 1,5 mm de diámetro. El estampado de las pistas y las perforaciones sobre la placa ha sido realizado mediante una máquina fresadora especializada con un cabezal de 0'4 mm de diámetro.



Figuras F. 35, F. 36 y F. 37: Serigrafías Top, Bottom y SST, a escala real, de la Placa “Patas” grupo A  
(Fuente: Proyecto MIHRO)

En las tres imágenes (aparecen ampliadas en el anexo de Planos) puede verse que el perfil rectangular de la placa tiene una hendidura de 3 x 20mm en su costado inferior, justo debajo del espacio que ocupa el PIC. Esta hendidura sirve para evitar que la placa quede levantada por encima de las tuercas de los tornillos que sujetan el servo tras el que se instalará esta placa. También evita que dichas tuercas puedan entrar en contacto con las pistas o las soldaduras de la placa, pudiendo provocar así algún puente o cortocircuito.

La mayor dificultad en la fabricación de estas placas ha sido sin duda el proceso de soldado. Las placas, además de ser pequeñas y apiñadas, exigen que los componentes electrónicos que las integran se suelden por sus dos caras. Esto es relativamente sencillo en el caso de resistencias o condensadores, pero se vuelve una tarea ardua en el caso de los conectores, pues todos ellos disponen de un zócalo que tapa, al instalar el componente sobre la placa en su posición correcta, los *pads* en los que deben soldarse sus patillas. Más difícil aún resulta la soldadura de los zócalos que han de albergar a los microcontroladores (estos integrados son sensibles a la temperatura y se estropean, por lo que nunca se sueldan directamente a la placa), pues la soldadura debe realizarse tanto en la cara por la que sobresalen las patillas del zócalo al colocarlo en su lugar como en la misma cara en la que se asienta el zócalo, teniendo que soldar con gran habilidad debajo del mismo, y sin ninguna oportunidad de error. Otros componentes, como el botón de RESET, el Xtal oscilador, los LEDs o el conector de 16 pines, también resultan extremadamente difíciles de soldar, por lo que la tarea de soldadura ha acabado por ser una de las más complejas de todo el proyecto en cuanto a la habilidad manual que exige.

#### 5.2.4 Operación

El acceso a los conectores, los indicadores y el RESET de la placa “patas” se realiza desde el exterior y, excepto en el caso del conector principal, desde el lateral del robot.

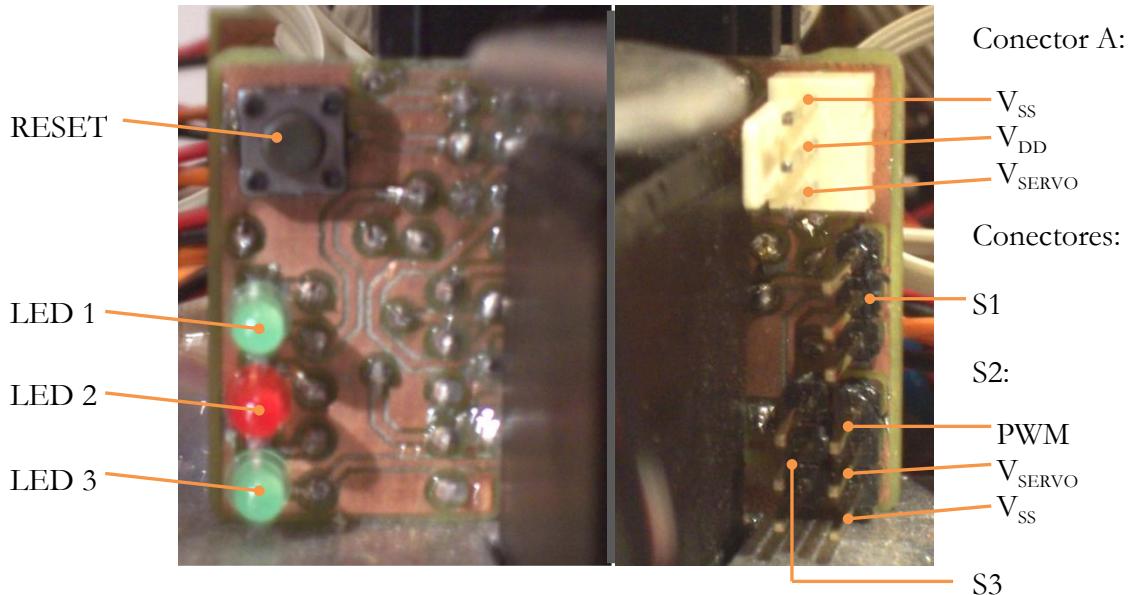


Figura F. 38: Elementos 1 de la placa “Patas”  
(Fuente: Proyecto MIHRO)

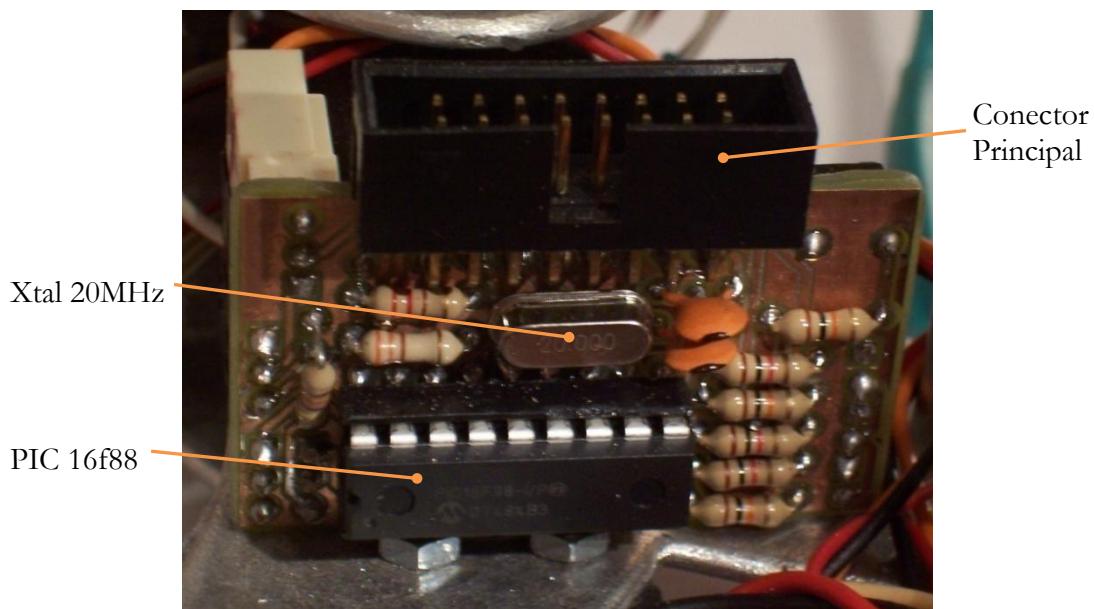


Figura F. 39: Elementos 2 de la placa “Patas”  
(Fuente: Proyecto MIHRO)

Hay que tener en cuenta que LED 1 y LED 3 son, respectivamente, los indicadores de detección de obstáculo y contacto con el suelo. Así mismo, S1, S2 y S3 corresponden, por orden, a los servomotores de “Cadera”, “Fémur” y “Tibia”. Los tres tienen el mismo patillaje, pero diferenciándose en que cada uno recibe su propia señal PWM.

El conector principal ofrece acceso a las principales características de la placa. No obstante algunas de ellas, como el botón de RESET, son incompatibles con la tarjeta programadora empleada en este proyecto, por lo que habrá dos conectores, uno normal y otro de programación, que permitirán conectar las patillas que sean necesarias en cada caso. Este botón de RESET está inicialmente desconectado de la patilla MCLR del microcontrolador, pero tanto ésta como el botón disponen de un pin en el conector principal, de manera que el botón puede conectarse al PIC puenteando ambas patillas físicamente en el conector. De esta manera, al conectar la programadora, la patilla MCLR estará conectada sólo a la programadora, mientras que al conectar el robot de forma normal MCLR volverá a conectarse con el botón de RESET.

Por último ha de mencionarse que los últimos tres pines del conector principal no se usan (los últimos dos pines, si contamos con que uno de ellos lo usa la tarjeta programadora). Estos pines corresponden a una posible expansión aún no implementada de las funciones de la pata, que consiste en introducir un tercer plano de detección de obstáculos mediante un dispositivo óptico, como un optoacoplador reflexivo. La inclusión de un tercer plano de detección en las patas permitiría al robot detectar obstáculos también cuando avanzara de forma vectorial. No obstante, la implementación mecánica de este sistema es extremadamente compleja, si se pretenden conservar los criterios de robustez del robot, por lo que se ha optado por una solución óptica en vez de mecánica. La placa “patas” está preparada, mediante estos pines y las resistencias de  $47\text{K}\Omega$  y  $220\Omega$ , para digitalizar una señal analógica procedente de un optoacoplador reflexivo genérico situado en la Tibia (los valores de las resistencias corresponden a un CNY70) y cuantificar así la proximidad de un obstáculo.

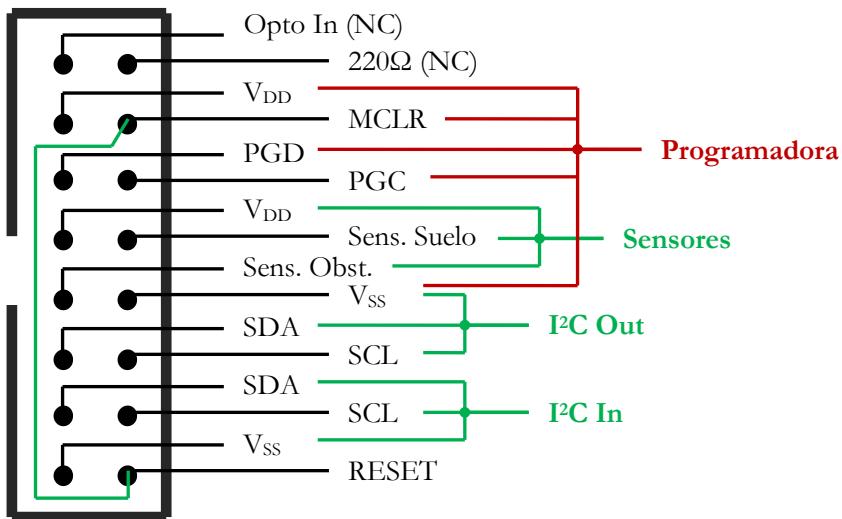
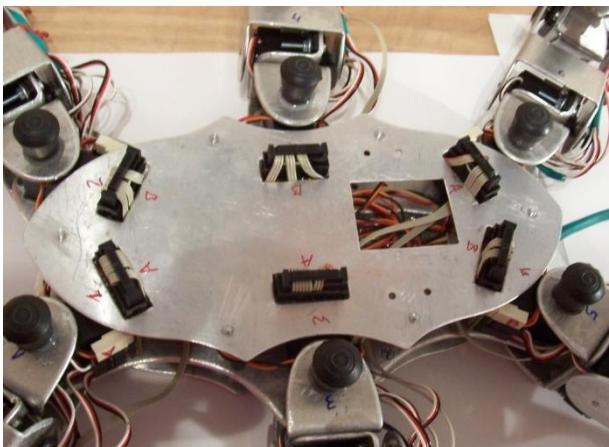


Figura F. 40: Diagrama de pines del Conector Principal del grupo A  
(Fuente: Proyecto MIHRO)

En el diagrama, las líneas verdes corresponden al conector normal, mientras que las rojas se refieren al conector de la programadora. Además, el diagrama también es válido para el conector hembra.

Para programar las placas “patas” grupo A es necesario tener en cuenta ciertas consideraciones:



**Figura F. 41: Acceso a los Conectores Principales**  
(Fuente: Proyecto MIHRO)

- Los conectores principales de cada pata son accesibles desde la parte inferior del robot, a través de la Tapa Inferior del Tórax.
- Para programar una placa del grupo A habrá que disponer de un cable adaptado para la función de programación, según el diagrama de pines de la página anterior (este cable no será compatible con las placas “pata” del grupo B).

- Una vez conectado el cable a los terminales de la programadora, se desconecta el conector principal normal de la placa que se desea programar, y se introduce el conector de programación.



**Figura F. 42 : Extracción del conector**  
(Fuente: Proyecto MIHRO)



**Figura F. 43: Conexión de la programadora.**  
(Fuente: Proyecto MIHRO)

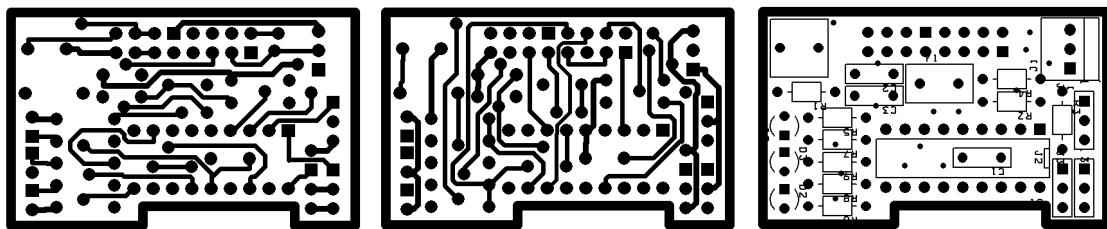
- Es imperativo desconectar el cable de alimentación de la placa que se desea programar. Esto se debe a que la programadora está diseñada para acceder a un PIC colocado en un zócalo aislado, por lo que se asume que la tensión de programación sólo alimentará al PIC. Si esta tensión de programación, conectada a V<sub>DD</sub>, encuentra otros elementos que provoquen un consumo de corriente, la programadora no podrá mantener la tensión de programación y el proceso fallará irremisiblemente.
- Por último, si la programación se ha realizado con éxito, se desconecta el cable de programación, se devuelve el Conector Principal de la placa a su sitio y se reconecta el cable de alimentación.

## 5.3 Placa “Patas” grupo B

### 5.3.1 Características

Esta placa es prácticamente idéntica a su análoga del grupo A, excepto en el hecho de que su serigrafía es el “reflejo” horizontal de la serigrafía de la placa del grupo A. Esto significa que la placa del grupo B dispone a su izquierda todos los elementos que la placa del grupo A dispone a su derecha, y viceversa. De este modo se soluciona el problema de la disposición simétrica de las placas, comentado en el apartado 5.2.1. A pesar de ello, ambas tienen similar forma e idéntico tamaño, integran los mismos componentes electrónicos, proceden del mismo circuito “Patas” y mantienen el mismo patillaje en todos los conectores excepto en el Conector Principal. Entonces, el motivo de que se haya desarrollado una nueva serigrafía para esta placa en vez de usar una versión espejada de la serigrafía del grupo A reside en el hecho de que el PIC, un componente integrado y con un patillaje definido, no se puede espejar. La única forma de construir esta placa directamente de la serigrafía del grupo A sería cambiar la cara de la placa en que se suelda el PIC (dándole así la vuelta, cosa que orientaría sus patillas en el sentido correcto), pero esto haría que el PIC ocupase el espacio destinado a encollar la placa a la parte trasera del servo, por lo que esta solución queda descartada.

La única solución entonces es desarrollar una nueva serigrafía basada en el mismo circuito, e integrar en ella los mismos componentes electrónicos que los del grupo A (con la excepción de los LEDs: el grupo A tiene dos verdes y uno rojo, mientras que el grupo B tiene dos rojos y uno verde). El resultado es el siguiente:



Figuras F. 44, F. 45 y F. 46: Serigrafías Top, Bottom y SST, a escala real, de la Placa “Patas” grupo B  
(Fuente: Proyecto MIHRO)

(El anexo Planos dispone de una versión ampliada de estas serigrafías para su mejor visualización).

Aunque el proceso de programación y la operación con esta placa son los mismos que con la placa de grupo A, hay que hacer notar que el patillaje del Conector Principal de esta placa es distinto, por lo que las “Patas” grupo B necesitarán, entre otras cosas, un cable de programación propio.

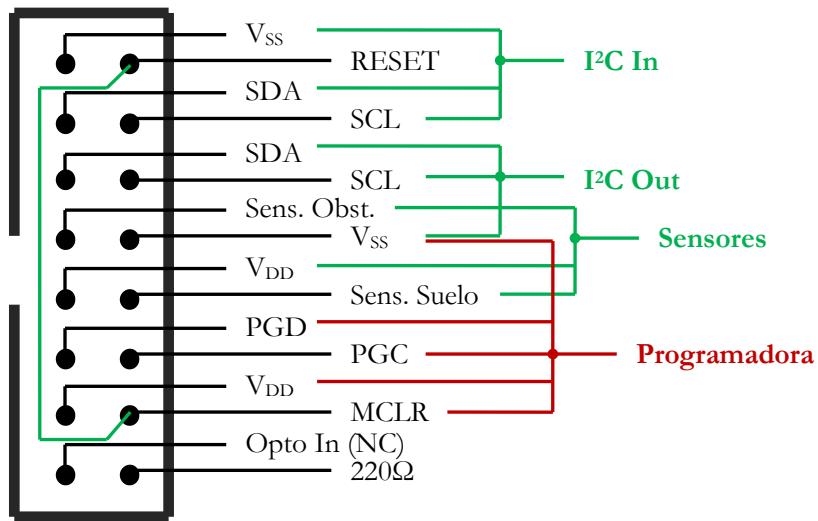


Figura F. 47: Diagrama de pines del Conector Principal del grupo A  
(Fuente: Proyecto MIHRO)

Otra diferencia entre la placa de grupo A y de grupo B que hay que poner de manifiesto es la distribución de los conectores de los servos. En el caso de la placa de grupo A, los servos de “Cadera”, “Fémur” y “Tibia” se conectan a los conectores S1, S2 y S3, respectivamente. Esto es, de arriba abajo y de fuera a dentro. Por el contrario, los conectores de los servos de la placa de grupo B se cuentan de dentro a fuera y de abajo arriba, por lo que, en el diagrama del apartado 5.2.4 el servo “Cadera” correspondería a S3, el servo “Fémur” a S2 y el servo “Tibia” a S1.

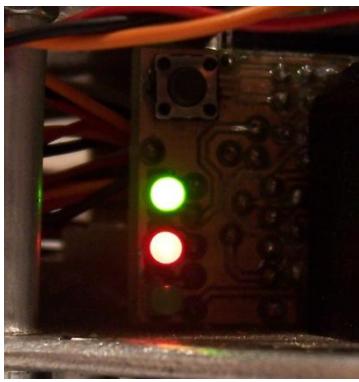


Figura F. 48: LEDs de estado  
(Fuente: Proyecto MIHRO)

Como información adicional, se puede decir que tanto la placa “patas” de grupo A como la de grupo B presentan un consumo medio de 12mA, algo menos si los motores están alimentados y en reposo. También es interesante mencionar que los LEDs de estado se han incluido en la placa con el fin de detectar fácilmente fallos en el funcionamiento de los sensores de contacto o de la misma electrónica, y también con el fin de ayudar en el desarrollo del software durante las etapas de programación. Con el mismo objeto se ha incluido el botón de RESET en el diseño de estas placas, pues en la aplicación final carecen de mayor uso, ya que en pocas circunstancias se puede dar el caso de que estas placas deban

reiniciarse manualmente, y en tal caso bastaría con desconectar la alimentación y volverla a conectar. En cualquier caso, en el momento de la programación se puede activar o desactivar la función que realiza este botón actuando sobre el flag de configuración MCLRE (Memory Clear Enabled), pero en caso de que esté activado ha de tenerse en cuenta que estas placas no deberán funcionar si no tienen su conector principal conectado, pues de no tenerlo el pin de MCLR quedaría “al aire”, de modo que su funcionamiento se volvería impredecible y podría provocar reinicios del programa no deseados.

## 5.4 Placa Entrenadora

### 5.4.1 Características

Aunque originalmente se diseñó para realizar experimentos sobre la locomoción del tercer prototipo de pata que se desarrolló, tal y como explica el capítulo 3, finalmente esta placa se ha adaptado para encajar en la estructura de MIHRO y se ha programado para tomar el rol de “cerebro” o Master del bus I<sup>2</sup>C. Entre sus principales características destacan:

- Dos conectores de alimentación de dos pines, uno para V<sub>DD</sub> y V<sub>SS</sub> y otro para V<sub>SERVO</sub> y V<sub>SS</sub>.
- Dos conectores para los sensores de suelo y obstáculo.
- Tres conectores de tres pines (V<sub>SS</sub>, V<sub>SERVO</sub> y PWM) para los servos compatibles con el conector Futaba.
- Tres botones.
- Cinco LEDs indicadores: tres verdes para los botones, y dos para las señales RX y TX de la comunicación RS-232, uno verde y otro rojo.
- PIC16f88 con dos ranuras de expansión para darle un uso diferente a cualquiera de sus patas.
- Xtal oscilador de 20MHz.
- Botón de RESET.
- Dos conectores para el bus I<sup>2</sup>C. Sólo se usa uno de ellos, pues el otro debe quedar abierto para futuras expansiones.
- Un conector de tres pines para la comunicación RS-232.
- Dos resistencias *pull-up* conectadas a SCL y SDA. Al estar estas resistencias en el Master, el bus queda configurado para cualquier esclavo que se le conecte.
- Dos *arrays* de cinco *jumpers* cada uno, uno de programación y otro de trabajo. Si se conecta el *jumper* de programación y se desconecta el de trabajo, las patillas de programación del PIC quedan conectadas únicamente a la programadora. De lo contrario, estas patillas interactúan con el circuito de forma normal.
- Driver MAX232, conversor de niveles TTL a RS-232.

El mapa del circuito electrónico de esta placa puede consultarse en el anexo de Planos.

En el cambio de Entrenadora a Master, esta placa no ha sufrido ningún cambio en su circuito gracias a las ranuras de expansión que abrazan el microcontrolador. Por ejemplo, una de las tareas más importantes que lleva a cabo esta placa es la lectura del pequeño teclado numérico desde el que el usuario introduce las órdenes a ejecutar por el robot. Dicho teclado consta de once botones que deben ser leídos en once pines diferentes del PIC. Esto es imposible en primera instancia, ya que el diseño inicial de la placa otorga una función definida a cada una de las patillas del PIC, pero mediante la ranura de expansión y el control de dirección de los puertos del PIC es posible cambiarlo y adaptarlo al teclado.

#### 5.4.2 Lista de componentes

- Placa de topos para prototipos de baquelita
- PIC 16f88 en encapsulado PDIP18
- Zócalo PDIP 18 patillas
- MAX232N encapsulado en PDIP16
- Zócalo PDIP 16 patillas
- Resistencia 0'25W: 1K $\Omega$  (x5)
- *Array* de resistencias 0'25W: 10K $\Omega$  (x8)
- Resistencia 0'25W: 330 $\Omega$
- Condensador cerámico 15pF (x2)
- Condensador cerámico 100nF
- Condensador electrolítico 1uF (x4)
- Xtal oscilador 20MHz
- LED verde 6mm (x4)
- LED rojo 6mm
- Pulsador mini normal (x4)
- Conector macho acodado 2,54mm 2pin (x5) + hembras
- Conector macho acodado 2,54mm 3pin (x4) + hembras
- Conector en tira hembra 2'54mm 20pin (sólo se usan 18)
- Conector en tira macho acodado 2'54mm bajo perfil 20pin (tres trozos de tres pin)
- Conector en tira macho poste 2'54mm bajo perfil doble línea 20pin (se usan 2x2x5)
- Cables varios

#### 5.4.3 Serigrafías TOP y BOTTOM

Al contrario de lo que cabía esperar, ya que este circuito se diseñó en una etapa muy temprana del proyecto, la placa entrenadora ha dado muy buen resultado, demostrando ser estable, fiable y duradera. No obstante, es una placa hecha a mano y apenas diseñada por ordenador, por lo que no existen mapas de su serigrafía más precisos que el mapa que se dibujó usando SketchUp para guiar su fabricación. La placa está físicamente construida sobre una placa de topos para prototipos de baquelita, y utiliza un sistema de jumpers para conmutar entre las tareas de programación y trabajo normal: cuando la placa está recibiendo un uso normal, un microrruptor de cinco vías, adaptado para puentear con todos sus interruptores cerrados un array de cinco jumpers, permite que llegue al PIC su alimentación normal y conecta los pines MCLR, RB7 y RB6 al resto del circuito. Por el contrario, cuando se pretende programar el microcontrolador, el microrruptor debe puentear los pines del segundo array de jumpers para que sólo los pines de programación se conecten a la programadora.

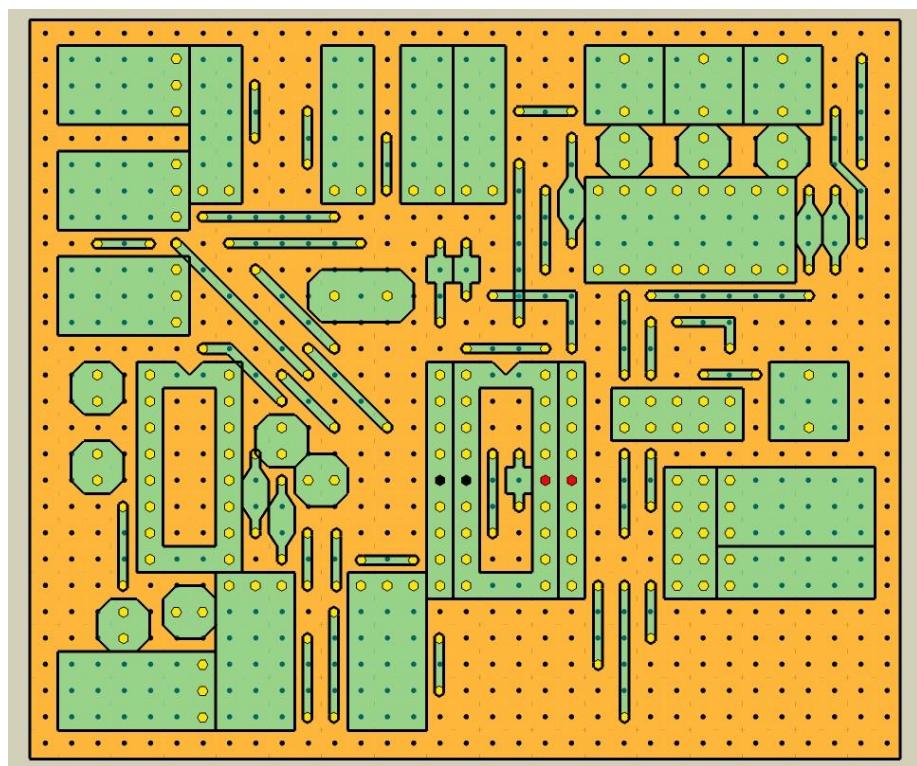


Figura F. 49: Serigrafía Top de la placa Master  
(Fuente: Proyecto MIHRO)

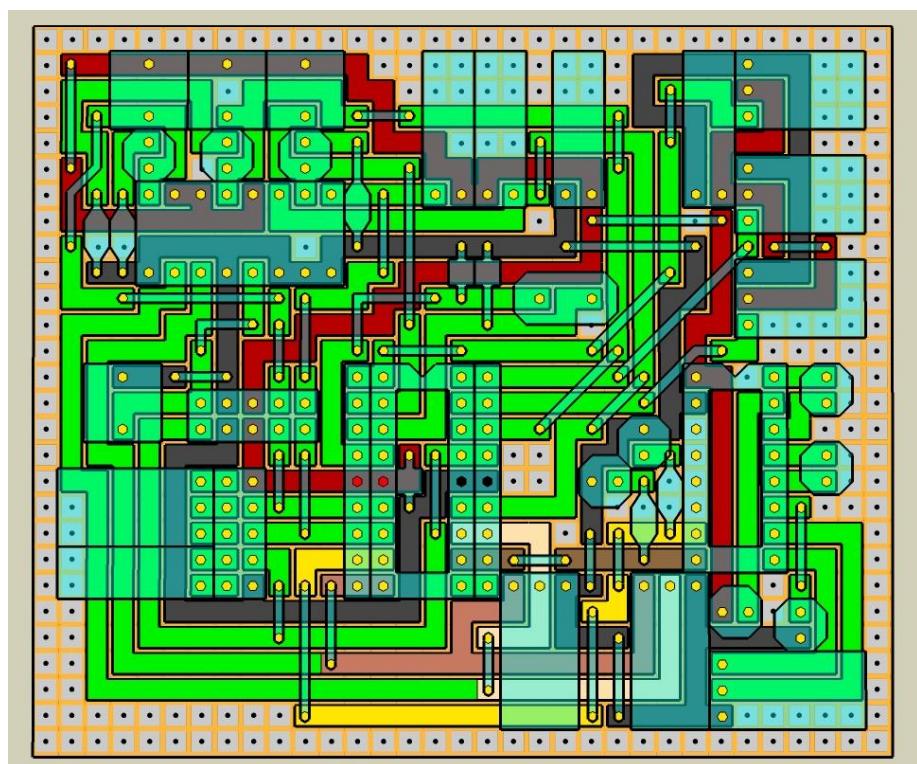


Figura F. 50: Serigrafía Bottom de la placa Master  
(Fuente: Proyecto MIHRO)

#### 5.4.4 Operación

La placa Entrenadora, de acuerdo con el uso específico que recibe en su aplicación final sobre el robot, sólo necesita de la conexión de sus conectores de alimentación e I<sup>2</sup>C. El resto de sus conectores están pensados para su uso como controladora del tercer prototipo de pata en la maqueta para la que se diseñó, o como herramienta de comunicación con un PC mediante RS-232, aunque esta aplicación no ha llegado a implementarse en este proyecto.

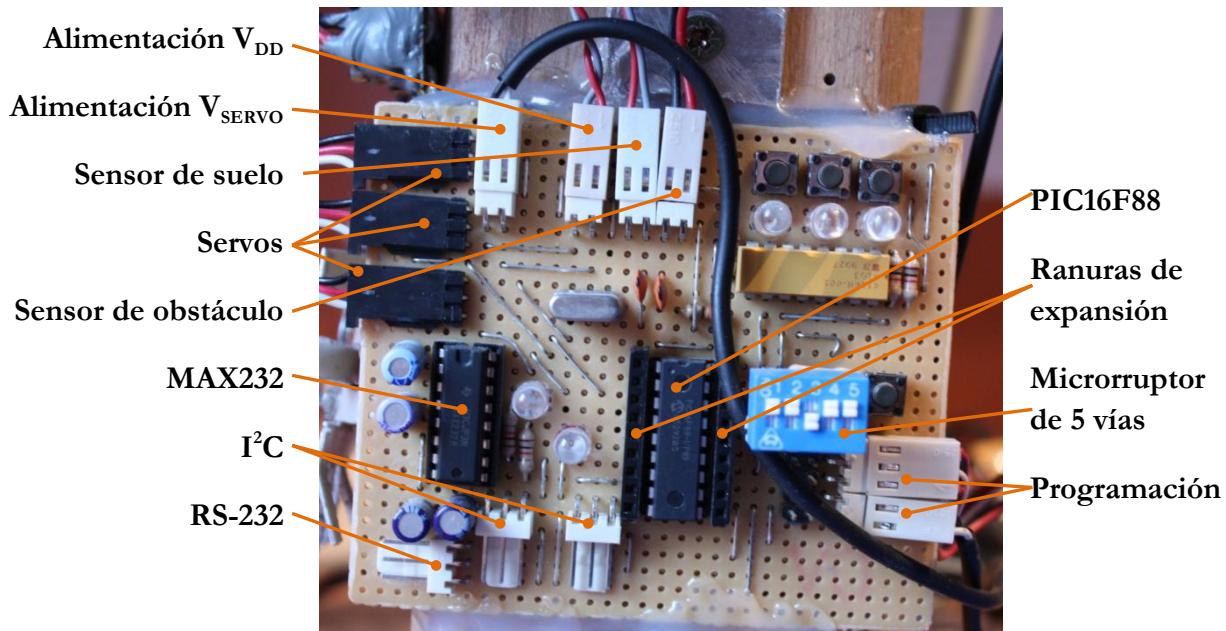


Figura F. 51: Elementos de la placa Entrenadora  
(Fuente: Proyecto MIHRO)

Cuando el microrruptor de 5 vías está conectado sobre el array de jumpers de trabajo normal, su tercera vía actúa como interruptor de alimentación, aunque hay que tener en cuenta que sólo conecta o desconecta la alimentación del PIC, por lo que si la tercera vía del microrruptor está abierta en esta posición, aunque el PIC deje de funcionar, el resto del circuito sigue recibiendo alimentación. Por otro lado, cuando el microrruptor da soporte a la función de programación desde el array de jumpers correspondiente, es la primera de sus cinco vías la que controla la tensión de programación que recibe el PIC.

Además, la aplicación final de esta placa hace uso de los conectores de expansión del PIC para implementar el control desde un teclado numérico o un mando parecido al de algunas videoconsolas. Este no es un teclado matricial, por lo que su lectura requiere de tantas patillas del PIC como botones se deseen leer. En esta aplicación sólo se dispone de 11 patillas que puedan dar soporte a la lectura de los botones del teclado, por lo que, aunque el teclado tiene 12 botones, uno de ellos (\*) no muestra conectividad con el circuito. No obstante, los 11 botones disponibles ofrecen la posibilidad de un control completo sobre todas las funciones del robot, según resume la tabla siguiente:

Teclado	Mando	Dígito 1	Dígito2	Dígito3	Dígito4
No botón	No botón	Stop (←)	-	-	-
0	4	+ 0	+0	+0	"acu"=0
1	-	+ 100	+10	+1	Speed
2	-	+ 200	+20	+2	Centro
3	1 (L1, Sel.)	(N.I.)(←)	+30	+3	Rango
4	L2 (3)	action,4=1(←)	+40	+4	Zanco
5	Up	action,1=1(←)	+50	+5	Altura
6	R2	action,5=1(←)	+60*/[Err(←)]	+6**/[Err(←)]	M1
7	Left	action,2=1(←)	+70*/[Err(←)]	+7**/[Err(←)]	M2
8	Down	action,1=0(←)	+80*/[Err(←)]	+8**/[Err(←)]	M3
9	Right	action,3=1(←)	+90*/[Err(←)]	+9**/[Err(←)]	Slave ***
#	Start (2)	I <sup>2</sup> C Send(←)	Err(←)	Err(←)	Mode

(←): Permanece en el mismo dígito

\* : No si el primer dígito es 2

\*\* : No si el primer dígito es 2 y el segundo 5

\*\*\* : Si es 0, se dirige a todos los esclavos

Tabla T. 5: Órdenes manuales mediante teclado o mando  
(Fuente: Proyecto MIHRO)



Figura F. 52: Mando y teclado  
(Fuente: Proyecto MIHRO)

No obstante, el mando, mucho más cómodo para el control de las funciones dinámicas del robot, no ofrece la posibilidad de introducir todos los valores numéricos disponibles desde el teclado, por lo que este último será preferible para las tareas de control estático del robot. Las órdenes introducidas desde el mando o el teclado se interpretan en el PIC maestro, que produce una serie de envíos de datos por el bus I<sup>2</sup>C a los distintos esclavos, consiguiendo que éstos actúen en consecuencia.

## 5.5 Otros elementos de hardware

### 5.5.1 Programadora PIPO2

El PIC16F88 incorpora un módulo llamado ICSP (In-Circuit Serial Programming) que le permite gestionar la escritura de su propia memoria de programa o datos durante el proceso de programación. De este modo, lo único que debe hacer el programador para grabar un código fuente en el microcontrolador es enviar los datos en código máquina por un canal serie, provocando a la vez los niveles de tensión adecuados para este proceso. Para resolver esta tarea existen montones de circuitos y dispositivos en el mercado que ofrecen diferentes niveles de características adicionales complementarias al proceso de grabación. No obstante, llama la atención esta PIPO2, basada en un circuito parecido llamado LudiPIPO, por su enorme simplicidad y efectividad.

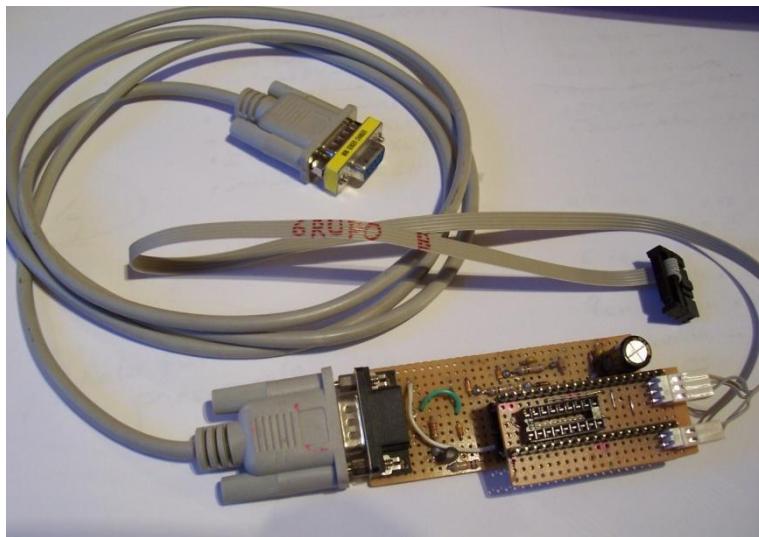


Figura F. 53: Programadora PIPO2  
(Fuente: Proyecto MIHRO)

La programadora PIPO 2, cuyo esquema y documentación están incluidos en el anexo de Planos y en el CD respectivamente, sólo realiza la tarea de adecuación de los niveles de tensión requeridos para grabar cualquiera de los PICs incluidos en su lista de dispositivos compatibles (en la cual se encuentran además memorias EEPROM y algún otro integrado), tomando los datos y la alimentación de un puerto serie convencional. De esta forma, con la ayuda de un software genérico, se puede usar el puerto serie de un PC para programar un gran número de microcontroladores PIC usando esta tarjeta como puente entre el PC y las cinco patillas de programación del PIC ( $V_{DD}$ , PGD, PGC,  $V_{PP}$  y  $V_{SS}$ ). Pero lo que más sorprende de este circuito es que puede montarse en apenas un rato por un precio no superior a 10€, por lo que constituye un recurso efectivo y rápido para resolver la grabación de estos microcontroladores. Además, el Club de Robótica Autónoma de la EUPMT lleva ya algunos años usando este circuito y los que lo han montado (correctamente) han comprobado que es duradero, robusto y fiable.

### 5.5.2 Otros módulos ayudantes

Además de la electrónica mencionada hasta ahora, se hace necesario incluir en el robot algunos circuitos pequeños y simples que suplan de forma provisional la carencia de la placa de alimentación. El más importante de ellos es un pequeño circuito regulador de tensión basado en un L7805 que convierte un nivel mínimo de 7,5V, sometido a las inestabilidades inherentes a la fuente de alimentación de laboratorio que se está usando, en un nivel estable de 5V ( $V_{DD}$ ). Aunque su complejidad es mínima y sus funciones básicas, el robot no podría realizar ni una sola acción sin este circuito. Los otros dos módulos adicionales que usa MIHRO son un distribuidor de los conectores de alimentación de cada placa y un soporte con dos LEDs indicadores para los interruptores de alimentación generales del robot.

### 5.5.3 Diseño previo de la placa de alimentación

Aunque no forma parte del desarrollo de ninguna de las etapas incluidas en este proyecto, se realizó un diseño preliminar de la placa de alimentación que acabará gestionando la energía del robot. Su esquemático puede consultarse en el anexo de Planos. A grandes trazos, lo que pretende este circuito es modularizar la distribución de la energía, de forma que, por un lado, se pueda activar o desactivar cualquiera de los módulos esclavos del robot, y por otro, se pueda regular dinámicamente la tensión de alimentación que llega en cada momento a los motores de cada pata por separado. Para lograrlo, se sigue una estrategia que consiste en reducir la tensión que llega a aquellas patas que estén avanzando, y por tanto que no estén soportando el peso del robot, a la vez que se aumenta la tensión que llega a las patas que están en el suelo, que necesitan ejercer un par motor suficiente como para sostener y arrastrar el peso del robot a lo largo de cada paso. Esto se consigue mediante los dos integrados AD5263BRUZ50, dos potenciómetros digitales de cuatro canales y de 50 K $\Omega$  de fondo de escala, que pueden gobernarse por I<sup>2</sup>C, y que permiten regular, mediante la variación de cada una de sus resistencias, la intensidad de la patilla de ajuste de un regulador de tensión variable LM317, capaz de entregar hasta 1,5A a cada pata.

Además, 2 de las 24 patillas de estos potenciómetros son salidas digitales controlables desde algunos de sus registros internos accesibles mediante la comunicación I<sup>2</sup>C. Esto ofrece la interesante posibilidad de controlar hasta 4 señales digitales en la placa de alimentación mediante el envío de órdenes por el bus desde el Master. De este modo, es posible controlar algunos elementos críticos de esta placa, como relés o transistores MOSFET, que gestionen la conexión o desconexión de sus diferentes módulos sin tener que incluir ningún microcontrolador en la misma. Con esto se lograría, entre otras cosas, ampliar su diseño con un módulo gestor de la carga de una batería controlado desde el Master, o incluir un conmutador que permitiera escoger entre una fuente de carga doméstica fija o una célula fotovoltaica, cuya inclusión en este proyecto está planificada para sus últimas etapas de desarrollo.

# 6 Programación

## 6.1 Herramientas de programación

La tarea de programación del robot se divide a su vez en tres sub-tareas (edición, compilado y grabación) que han sido realizadas mediante programas gratuitos descargados de Internet:

- **EditPlus 2:** Este sencillo programa constituye una interesante herramienta de edición de código fuente en ensamblador si se configura adecuadamente. Lo que hay que hacer sencillamente es linkar ciertos parámetros del panel de preferencias del programa a un archivo de sintaxis, que nosotros mismos podemos escribir o modificar, para poder ver los distintos elementos del código en pantalla coloreados para su mejor distinción. Con esto el programa permite visualizar en una lista ayudante la correcta escritura de las 35 instrucciones que componen el lenguaje ensamblador (sin macros) de la familia 16F de Microchip, así como diseñar nuestros propios *templates* con cabeceras, fragmentos de código, etc., además de sus otras características genéricas.

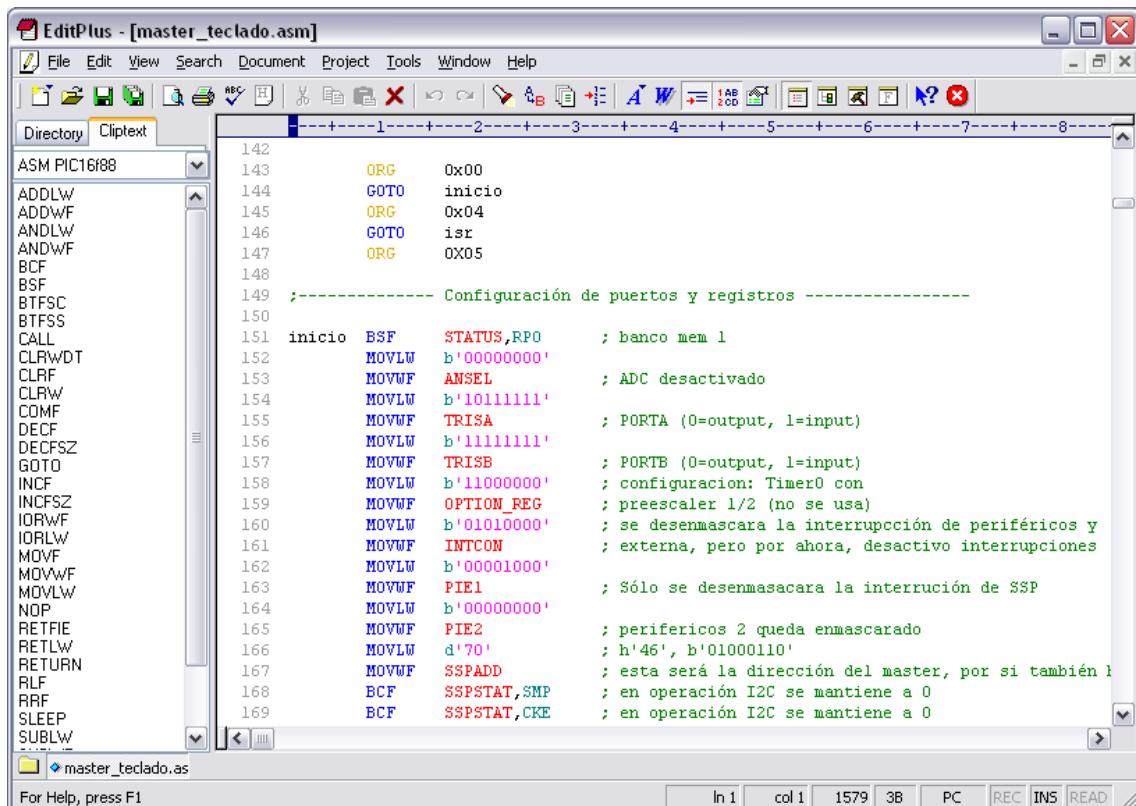


Figura F. 54: Interfaz del editor EditPlus2  
(Fuente: Proyecto MIHRO)

- **MPASM:** Compilador para lenguaje ensamblador proporcionado por Microchip. Su función es la de traducir el texto guardado en un archivo de código fuente .asm a un archivo .hex de valores hexadecimales (código máquina), que contiene la información que deberemos grabar en el PIC. Tras la compilación se generan los archivos secundarios de error, listado y referencia cruzada, además del archivo.hex.

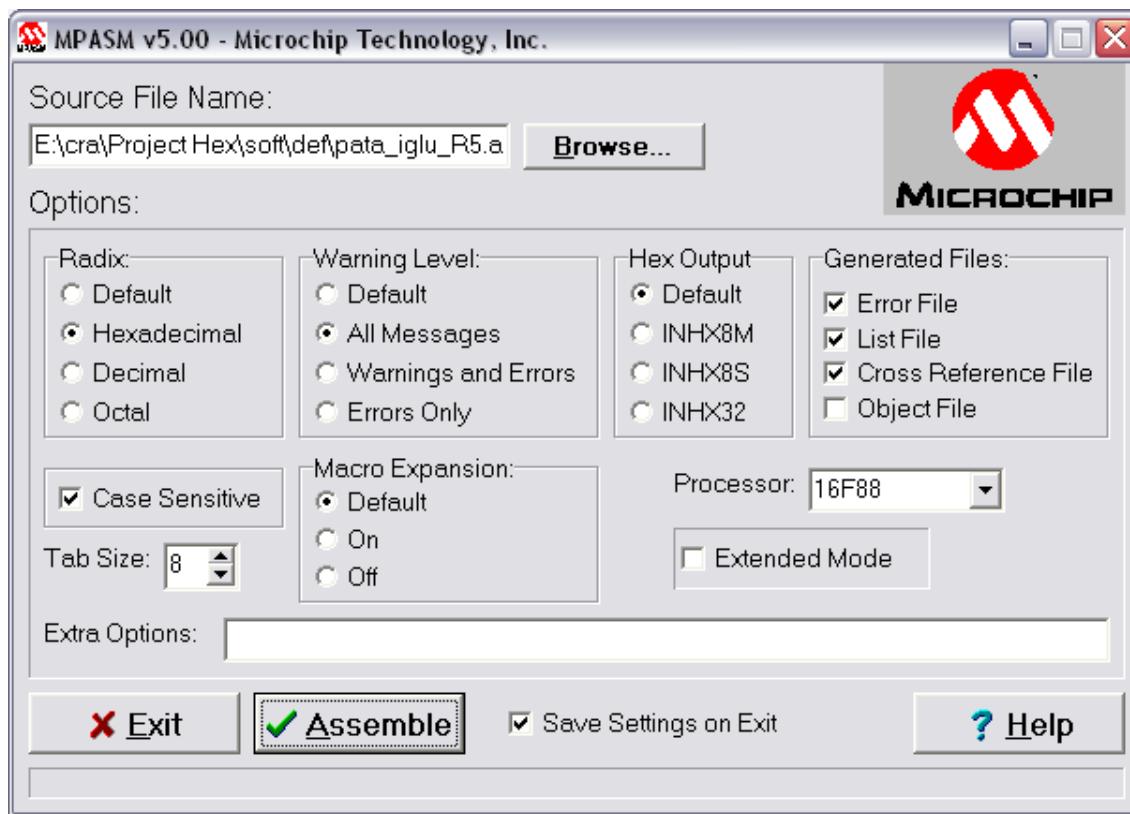


Figura F. 55: Interfaz del compilador MPASM  
(Fuente: Proyecto MIHRO)

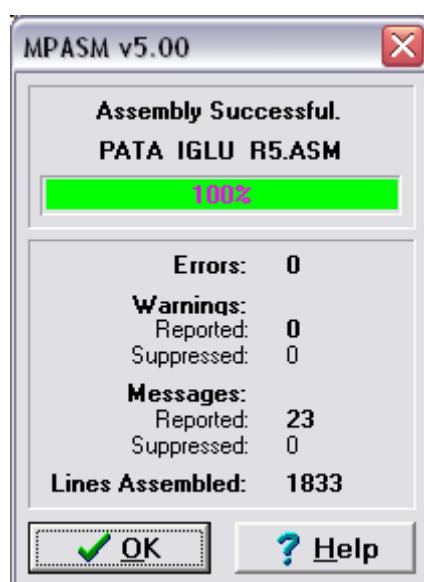


Figura F. 56: Resultado de la compilación  
(Fuente: Proyecto MIHRO)

- **WinPic800:** Este es el software grabador de PICs que se usó en la última edición del “Curset d’Estiu de Robòtica” de la EUPMT. Su interfaz, sencilla y directa, permite grabar el PIC, leerlo, borrarlo y verificarlo, pero también testar el hardware programador, y reconocer el dispositivo que se pretende programar. Otras opciones interesantes son, por ejemplo, la posibilidad de modificar los *fuses* de configuración del PIC antes de programarlo, sobrescribiendo la configuración que hubiéramos programado en el código fuente, o la de leer el contenido en código máquina de un PIC e interpretarlo para reconvertirlo de nuevo a código ensamblador.

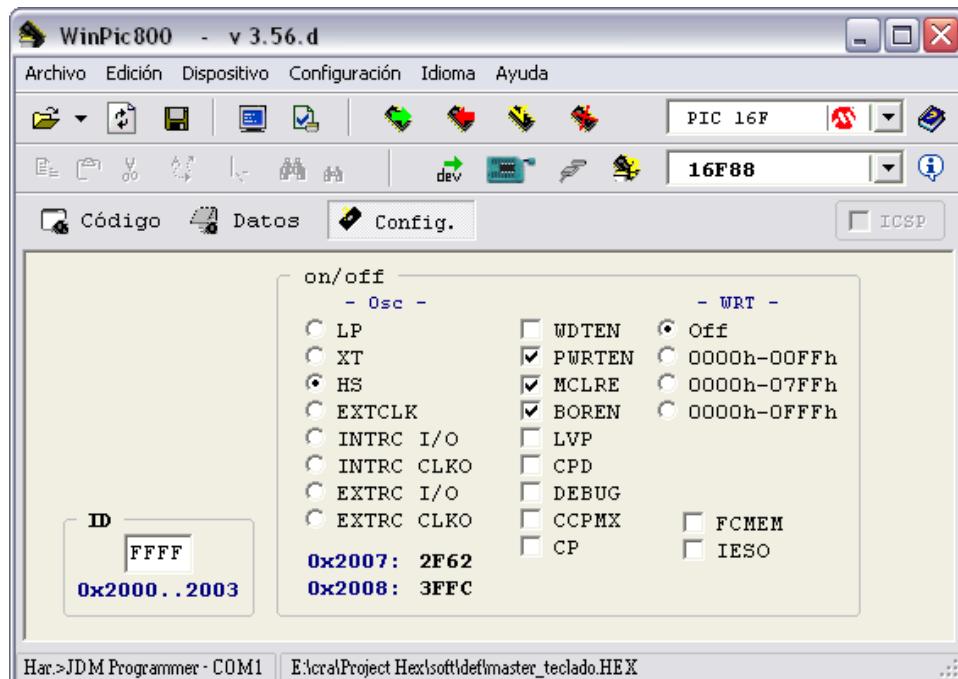


Figura F. 57: Interfaz de WinPic800  
(Fuente: Proyecto MIHRO)

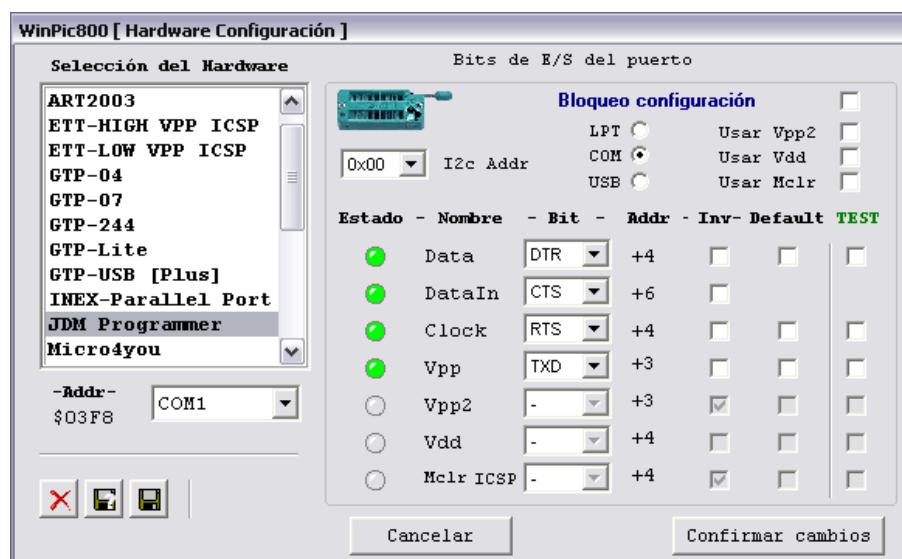


Figura F. 58: Configuración de WinPic800 para programar con PIPO2  
(Fuente: Proyecto MIHRO)

## 6.2 Programas

Resulta complicado dejar una constancia fehaciente del trabajo de programación realizado en este proyecto, pues la extensión del código fuente desarrollado, así como el carácter críptico inherente al lenguaje ensamblador hacen difícil tanto la tarea de adjuntarlo en esta memoria como la de explicar su funcionamiento detallado. Finalmente, la estrategia adoptada para solucionar el problema ha sido la de aunar ambas tareas escribiendo un código fuente muy comentado, casi a cada línea que se preste a ello, y adjuntar en el anexo de Código fuente los programas del Master y de uno de los esclavos. Hay que tener en cuenta que, aunque son muy similares, los seis programas esclavos desarrollados mantienen sustanciosas diferencias entre sí, como por ejemplo la declaración de las variables limitantes de sus movimientos o las rutinas de corrección de los motores 2 y 3 en las etapas finales de cada paso. No obstante, la totalidad del software definitivo escrito suma más de 10.000 líneas de código, por lo que finalmente se optó por incluir en el proyecto escrito, y sólo en un documento anexo, dos de los programas desarrollados (los dos más distintos entre sí). Así mismo, esta memoria sólo pretende dar a conocer el funcionamiento genérico y global de estos dos programas, dejando las explicaciones más concretas en manos de los comentarios intercalados en el código.

### 6.2.1 Programa “pata\_iglu\_R3.asm”

Este es el programa de control de la pata número 3 del robot, y presenta una equivalencia superior al 90% con los otros dos programas correspondientes a las otras dos patas derechas del robot. El funcionamiento general del programa se basa en el uso de variables de *flags*, o banderas, que permiten planificar y definir todas las divergencias que existen en el programa, de manera que cada módulo del código se pueda ejecutar correctamente aunque se acceda a él desde casi cualquier posición aleatoria del programa completo. El motor del programa es la interrupción del Timer1, que hace que todas las tareas del mismo, excepto las inicializaciones, se repitan cada 20ms. Este es un período suficientemente rápido como para que parezca que el robot responde a sus estímulos en tiempo real, pero lo cierto es que el refresco de acciones y variables se produce “sólo” 50 veces por segundo. No obstante, esta circunstancia obliga al programa a controlar, mediante los flags mencionados, el acceso a cualquiera de sus procesos para bloquear su ejecución más de una vez por cada período.

**Configuración:** a destacar que el PIC funciona en el modo oscilador HS con un Xtal de 20MHz. Esto nos deja, teniendo en cuenta que el Ciclo de Instrucción de esta máquina consta de 4 ciclos de reloj, en un C.I.= 200ns. Además, el programa activa las características de *Power-Up Timer*, *Brown-out Reset* y *MCLR Enabled* (aunque la aplicación esta última sólo cobra sentido durante las etapas de puesta en marcha del robot). Otras características como el *Watch Dog Timer*, o la protección de código han sido voluntariamente desactivadas.

**Bucle principal:** mantiene al procesador ocupado en un bucle infinito después de haber terminado todas las tareas que debe realizar en cada período de 20ms. Se mantiene permanentemente testeando los flags de desvío a las rutinas “ciclo” y “mofs”, pero dado que “mofs” sólo puede activarse al terminar la rutina “ciclo”, sólo la activación de esta rutina, o una interrupción, podrán sacar al programa de este bucle. De hecho, de forma genérica el programa sale de este bucle para atender a la interrupción por overflow del Timer1, configurado para contar periodos de 20ms, que precisamente tiene como única tarea (la interrupción) la activar el flag de acceso a la rutina “ciclo”.

**Rutina “ciclo”:** Esta rutina de sólo 18 líneas es la más importante del programa, pues se encarga de generar, una vez cada 20ms, el pulso de control PWM de la posición de cada uno de los tres servos implicados en esta pata. La estrategia que sigue para hacerlo es simple: se parte de la premisa de que la patilla conectada a la señal PWM del servo en cuestión muestra un nivel bajo en su salida, así que se pone en nivel alto para iniciar el pulso. A continuación se llama a un retraso “ciego” de tantos milisegundos como se le indique a través de la variable de entrada “del”. Cuando el tiempo ha terminado, el programa regresa a la línea siguiente a la llamada del retraso, en la que se devuelve el estado de la patilla al nivel bajo y se prosigue con el control del siguiente servo. Una vez completado el pulso correspondiente a la posición deseada en cada servo, se bloquea el acceso a la rutina “ciclo”, para no repetir el pulso antes del próximo período de 20ms, y se activa el acceso a la variable “mofs”. Finalmente, el programa retorna al bucle principal.

Hay que hacer notar que el tiempo que el procesador transcurre en la rutina de retraso es tiempo de cálculo perdido, pues éste sólo se dedica a contar instrucciones con el objeto de perder tiempo de forma controlada. Si se tiene en cuenta que el ancho de pulso máximo que se le pueden enviar a estos servos es 2'28ms, el peor caso de pérdida de tiempo de cálculo se produciría cuando los tres motores desearan esta posición extrema, por lo que el procesador perdería un tiempo por valor de 6'84ms. De este modo, se puede determinar que este sistema permitiría el control de 8 servomotores de este tipo como máximo por cada microcontrolador, a costa de limitar el tiempo para el procesamiento de otras tareas.

**Rutina “mofs”:** Aunque es muchísimo más larga, esta rutina sólo tarda en ejecutarse una fracción centesimal de lo que tarda “ciclo”. Su cometido es la gestión de la fluctuación controlada de las variables m1, m2 y m3, de las que se nutre “ciclo” para determinar la posición en que debe posicionarse cada servo. Para ello, esta rutina se nutre a su vez de un buen número de variables que condicionan el comportamiento dinámico de la pata, aunque las más importantes y determinantes son “action” y “mode”.

La variable 'action' contiene 8 banderas:

action,0 : 1= Go	0= Stop
action,1 : 1= Atrás	0= Delante
action,2 : 1= Giro Izquierda	0= No Giro
action,3 : 1= Giro Derecha	0= No Giro
action,4 : 1= Rotación Izq.	0= No Rotación
action,5 : 1= Rotación Dch.	0= No Rotación
action,6 : 1= Ida del paso	0= No Ida
action,7 : 1= Vuelta del paso	0= No vuelta

Dichas banderas son modificadas por el Master mediante la rutina de comunicación I<sup>2</sup>C, que se trata más tarde, y se usan para coordinar la acción motora que deberá realizar la pata. Por eso, los bits [2:5] sólo se emplean en el programa del Master, mientras que el resto de bits sirven para ordenar al esclavo un movimiento de avance o retroceso que se sincroniza con el del resto de las patas mediante la activación y desactivación controlada de los bits [6:7].

La variable 'mode' contiene 8 banderas:

mode,0 : 1= Modo Dinámico	0= Modo Estático
mode,1 : 1= Modo levantamiento	0= No Modo
mode,2 : 1= Modo Coger	0= No Modo
mode,3 : 1= Modo Inicial	0= No Modo
mode,4 : 1= Modo Programación	0= No Modo
mode,5 : 1= Obstáculo detec.	0= No Obstáculo
mode,6 : 1= (N.I.)	0= (N.I.)
mode,7 : 1= (N.I.)	0= (N.I.)

Mode, por el contrario, actúa a un nivel más alto, determinando el tipo de acción motora que deberá realizar la pata entre:

- Modo estático: La pata no ejecuta el control evolutivo de las variables m1, m2 y m3, sólo copia en ellas las consignas que le envía el Master para mantener cada motor en una posición fija. Mediante este modo, el usuario puede introducir por teclado el valor exacto de la posición que desea que adquiera un motor del robot.
- Modo Dinámico: La pata ejecuta un control evolutivo de las variables m1, m2 y m3 diseñado para reproducir la secuencia de movimientos de un paso. Dicha secuencia recibe el nombre de “iglú” por la forma que dibujan sus motores 1 y 2 trabajando al unísono, y su principal característica es que está diseñada para reducir la inercia en los límites de su rango de movimiento, consiguiendo así evitar las oscilaciones, optimizar el paso, reducir el consumo y alargar la vida de los motores. La secuencia “iglú” se divide en 7 etapas controladas por 8 variables que definen puntos clave de su movimiento en los ejes de m1, m2 y m3. Dichas variables están a su vez controladas por las variables *speed*, *centro*, *rango*, *zanco* y *altura*, que constituyen las 5 consignas definitorias del paso. Así pues, mediante el envío de estas 5 variables, el Master controla todas las características de cada paso de la pata. También hay que decir que cada etapa se nutre de las rutinas “fluc” y “find\_m3” para ejecutar, respectivamente, una evolución progresiva de la posición de los motores tanto en su sentido de avance como de retroceso, y una corrección del valor de m3 que le permite definir una línea recta en su movimiento de retroceso (mejorando así la estabilidad en el avance del robot).
- Modo inicial: lleva al robot a una posición conocida de inicio que mantiene durante un par de segundos
- Modo levantamiento: hace que el robot se levante con sus seis patas. Normalmente se llega a este modo justo después del modo inicial.
- Modo programación: hace que las patas del robot desciendan con las tibias recogidas, cosa que facilita la operación de darle la vuelta al robot para programarlo.

- Modo coger: Cuando el robot se halla a una distancia conocida de un objeto que desea recoger y cargar, ejecuta una secuencia de movimiento con sus patas delanteras que lo agarra, lo eleva y lo deposita sobre su cuerpo mientras el robot se sostiene sobre sus cuatro patas traseras. Esta característica aún no ha sido implementada.

**Rutina “delay”:** esta es la rutina de retraso mencionada anteriormente. Consiste en la ejecución de un primer tiempo de retardo de 0'48ms, correspondientes al ancho del pulso mínimo que se le puede mandar al servo. Acto seguido ejecuta tantas veces como se le haya indicado a través de la variable “del” una rutina de exactamente 7,2us. De esta manera se puede controlar el intervalo entre la posición de 0° (0'48ms) y de 180° (2'28ms) con 250 de los 255 valores posibles de una variable de 8 bits, consiguiendo así controlar la posición de cada servo con una precisión de  $180^\circ / 250 = 0,72^\circ$ .

**I<sup>2</sup>C Slave:** a esta rutina sólo se llega cuando ocurre un evento en el módulo SSP que produce una interrupción (y ésta está desenmascarada). Lo que hace es discernir entre 5 situaciones distintas que determinan su comportamiento:

- Recepción de un dato en escritura, el último byte es una dirección.
- Recepción de un dato en escritura, el último byte es un dato.
- Recepción de un dato en lectura, el último byte es una dirección.
- Recepción de un dato en lectura, el último byte es un dato.
- Finalización / Reset de la comunicación I<sup>2</sup>C.

Para más detalles, consultar el apartado correspondiente a este fragmento del código en el anexo de Código fuente.

**Sistema Interruptivo:** Este programa responde a las interrupciones de: evento en el módulo SSP, overflow del Timer1, flanco de subida en el pin RB0/INT y cambio en el pin RB4 (esta fuente de interrupción debe enmascararse cuando está desenmascarada la interrupción por evento en el módulo SSP).

### 6.2.2 Programa “master\_teclado.asm”

Este programa, al contrario que el anterior, no funciona a intervalos regulares, sino de forma continua. A grandes rasgos, su funcionamiento se basa en la interacción entre la placa Master y el teclado que lleva conectado: de forma normal el programa se mantiene atrapado en el bucle principal, pero sale del mismo cuando el usuario pulsa una tecla, y va a parar a la rutina de gestión de esa tecla. Dicha rutina tiene cuatro estados posibles que le indican si la tecla se ha pulsado cuando la máquina esperaba un dígito 1, 2, 3 o 4. Por ejemplo, si se pulsa un número cuando el estado del programa correspondía al dígito 1, la tecla presionada conllevará la ejecución del código escrito en el espacio destinado al dígito 1 de la rutina de dicha tecla, y acto seguido el programa cambiará al estado de dígito 2.

Además, existen normas para presionar una u otra tecla en cada uno de los dígitos. Por ejemplo, si se presiona la secuencia de teclas “1”-“4”-“3”-“1” (consultar la tabla T.5 del apartado 5.5.4), el código de dígito 1 de la tecla “1” hará que una variable “acu” valga 100, el código de dígito 2 de la tecla “4” hará que “acu” se incremente en 40, el código de dígito 3 de la tecla “3” hará que “acu” se incremente en 3 puntos, obteniendo el valor 143 en “acu”, y por último el código de dígito 4 de la tecla “1” hará que dicho valor se almacene en la variable *speed*. De este modo, el código se protege contra su mal uso, evitando que se introduzcan valores erróneos. Por ejemplo, si intentamos introducir un 5 en el primer dígito se iluminará un LED rojo gestionado desde la rutina “led4”, indicándonos que ha habido un error. El error, por otro lado, es lógico, pues cualquier cifra que empezara por 5 en su campo de las centenas debería valer al menos 500, y lo máximo que pueden cuantificar las variables de 8 bits es 255 (FFh). Otro ejemplo de cifra errónea sería la introducción de la secuencia “2”-“7”-“X” o “2”-“5”-“7”; en el primer caso, el siete no llegaría a entrar, pues cualquier número superior a 270 supera ya de por sí a 255, por lo que el sistema ni siquiera nos dejaría llegar al tercer dígito. En el segundo caso, el 5 entraría pero el sistema nos daría error al introducir el 7 final.

Algunas teclas, no obstante, se toleran en situaciones erróneas para darles otros usos. Es el caso, por ejemplo, de los números superiores a 2 en el primer dígito, todos incorrectos según lo explicado, pero que en este caso sirven para manejar la rutina “movs”, activando y desactivando los flags de la variable “action” que le indican qué tipo de secuencia debe sincronizar el Master en todas las patas para producir unos u otros movimientos. Por ejemplo, si se le da la orden de avanzar en línea recta el primer dígito la rutina mandará las consignas de movimiento dinámico a cada pata para que ejecuten una secuencia de locomoción triangular normal (esto significa que todas las patas avanzan y retroceden la misma distancia, pero las patas 1, 4 y 5 lo hacen a destiempo de las 2, 3 y 6, de forma que cuando unas avanzan las otras retroceden). Por otro lado, si se le da la orden de girar a la derecha mientras se avanza, le estará indicando a la rutina que ejecute la secuencia de locomoción triangular, pero reduciendo el rango de las patas del lado derecho del robot respecto de las del izquierdo, consiguiendo así que el robot se ladee.

Otros botones ejecutan tareas más concretas de almacenamiento y envío de los datos, como la tecla “#” que envía los datos almacenados por I<sup>2</sup>C al esclavo cuya dirección se haya almacenado usando la tecla “9”, o a todos ellos si dicho valor es 0.

A parte de esto, sólo queda por decir que el Master no usa ninguna fuente de interrupción, aunque disponga de algunas programadas para futuras revisiones de este programa, y que su rutina más compleja, la implementación de la comunicación I<sup>2</sup>C en modo Single Master completamente por software, exige un análisis demasiado concreto como para abordarlo en este apartado, por lo que deberá ser consultado en el apartado correspondiente del anexo de Código fuente.

## 7 Resultado

La integración de todos sus sistemas le confieren a MIHRO su aspecto final.

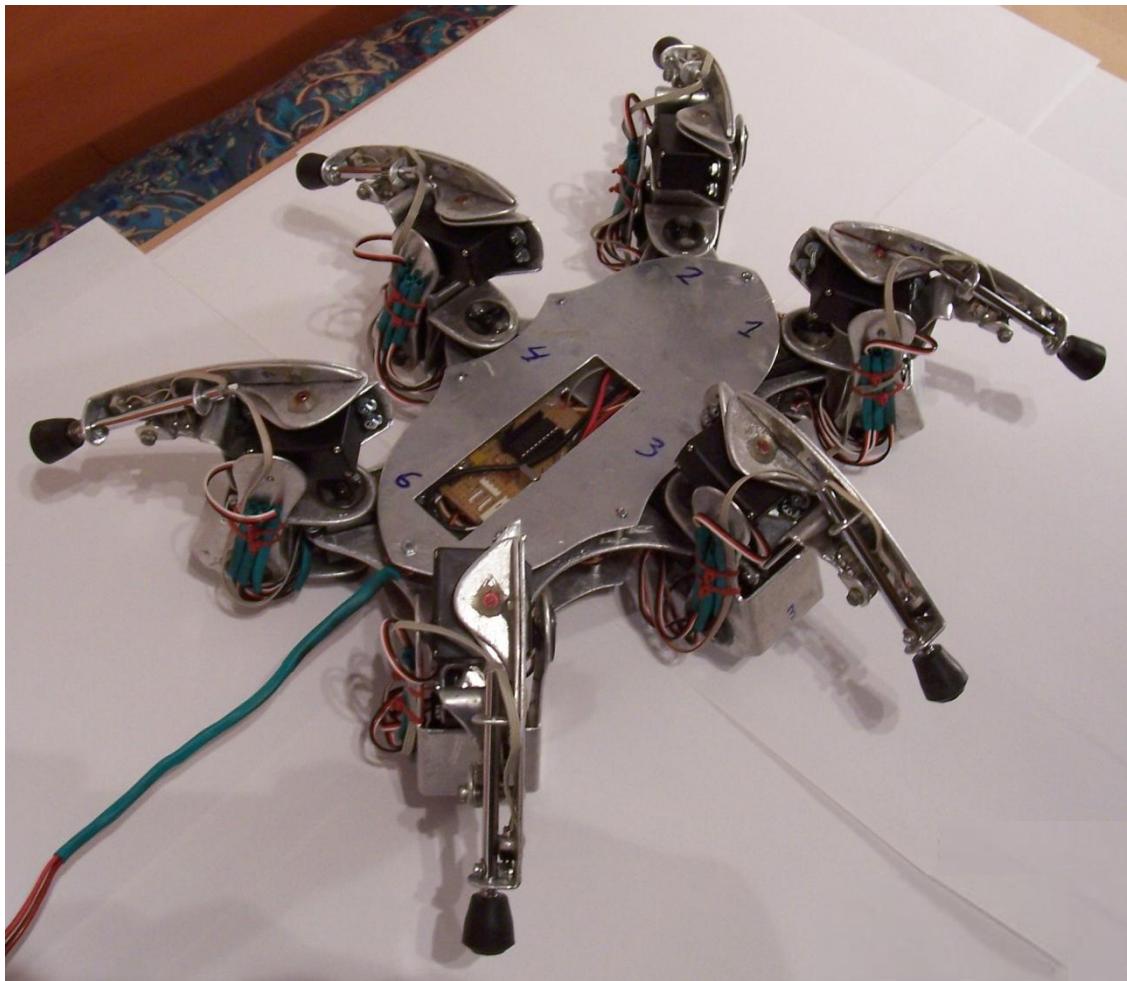


Figura F. 59: Aspecto final de MIHRO  
(Fuente: Proyecto MIHRO)

Puesto que el resultado final no puede evaluarse con una imagen, sino que exige ver al robot en funcionamiento, se han adjuntado algunos vídeos demostrativos en el CD que muestran algunos de los principales experimentos que ha llevado el desarrollo de MIHRO, así como la demostración de su operación y de sus principales habilidades.

## 8 Conclusión

Este proyecto perseguía desde un principio unos objetivos que exigían una cantidad de trabajo que excedía ampliamente las expectativas y la disponibilidad de tiempo de esta asignatura, por eso, como explican los primeros capítulos de esta memoria, el trabajo se fragmentó en etapas diseñadas para abordar el desarrollo de partes y características concretas del robot, obteniendo al final de cada una un resultado visible y evaluable. Con este sistema, se determinó que los objetivos principales de este proyecto corresponderían a las tres primeras etapas de su desarrollo global, y a partir de ahí se iría ampliando el proyecto con tantos objetivos de las etapas siguientes como el tiempo permitiera. No obstante, llegados al final de la asignatura ha quedado patente que el trabajo de estas tres primeras etapas fue subestimado, pues la consecución de sus objetivos solamente ha dejado tiempo para comenzar el desarrollo de uno de los puntos de la cuarta etapa, el diseño de la placa de alimentación, sin llegar siquiera a completarlo. A pesar de ello, la consecución de estos objetivos principales ya supone un logro para el proyecto, pues su desarrollo ha resultado ser una tarea ardua y compleja que ha exigido una dedicación muy superior a lo esperado.

La parte grata de esta circunstancia es que se han adquirido montones de conocimientos de gran valor sobre todas las materias y procesos incluidos en este trabajo. Se concluye entonces que se han asumido con éxito todos los objetivos principales de este proyecto, incluyendo los que se referían a la consideración del mismo como herramienta de aprendizaje y preparación para el futuro profesional:

- Se ha aprendido a desarrollar aplicaciones que requieran el uso de microcontroladores PIC, incluyendo los recursos electrónicos y las técnicas necesarias para programarlos.
- Se ha aprendido a programar y gobernar un sistema de comunicación electrónica basada en el bus I<sup>2</sup>C.
- Se ha adquirido experiencia y habilidad en la soldadura de circuitos electrónicos de dificultad avanzada.
- Se han asumido valiosos métodos y herramientas útiles para el diseño de estructuras mecánicas.
- Se han adquirido grandes conocimientos, experiencia y habilidad en la construcción y montaje de mecanismos y piezas articuladas.
- Se ha aprendido a diseñar circuitos y serigrafías usando OrCAD Capture y OrCAD Layout.
- Se ha aprendido a dominar el lenguaje ensamblador de la familia 16F de Michochip, incluyendo la programación de algunos de los principales módulos periféricos y del sistema interruptivo de estos dispositivos.
- Se han adquirido técnicas de programación que permiten el control dinámico de servomotores pequeños mediante señales PWM.

- Se han adquirido hábitos de programación y ordenación de las variables y funciones de un programa en ensamblador que evitan los problemas típicos de mal funcionamiento derivados de estrategias algorítmicas deficientes.
- Se han asumido técnicas de búsqueda, gestión y ordenación de la información, así como de su correcta expresión en forma de documentos técnicos.
- Se han adquirido valiosos conocimientos sobre robótica general, locomoción de robots y desarrollo de aplicaciones de este campo.

Y muchos más.

Además, todo este aprendizaje y la correcta finalización de esta primera etapa del proyecto permitirán encarar desde una posición mucho más cómoda el gran trabajo que queda aún por hacer. Y ese trabajo comenzará lo antes posible con el desarrollo de la etapa cuatro, sobre todo en lo referente a la placa de alimentación y la inclusión de la batería, dos parámetros que dotarán al robot de autonomía energética y harán que éste pueda prescindir de su actual cordón umbilical. También será un cambio inminente, en cuanto se reúnan los fondos para asumir el coste, la sustitución de los actuales servomotores “Tibia” y “Fémur” por otros más potentes, mejorando así la estabilidad del robot, sus propiedades motrices y su capacidad de carga.

Como último punto de esta conclusión y de esta memoria descriptiva, quisiera dejar en el aire una propuesta dirigida a cualquier persona interesada en realizar un proyecto de este tipo, o incluso en seguir los pasos de éste mismo. La propuesta consiste en desarrollar un sistema de comunicación para que varios robots como éste se sincronicen o se organicen para realizar tareas en colaborativas. Si se coge como modelo el sistema desarrollado para MIHRO, su bus interno tendría que extenderse a un canal de comunicación inalámbrico que le permitiría interactuar con las variables de control internas de otro robot como él, permitiendo a ambos compartir información procedente de sus respectivos sistemas sensoriales, enviarse órdenes o variables de control, etc. De esta forma se extendería la capacidad de detección un robot a la del otro, permitiendo a ambos colaborar para realizar tareas como sostener y transportar objetos más pesados o menos manejables, explorar zonas más amplias o realizar tareas de seguimiento, vigilancia o rastreo cubriendo varios puntos de acción o ángulos de visión simultáneamente.

## 9 Bibliografía

### 9.1 Proyectos de Final de Carrera EUPMT

- [1] Carlos Vivancos Gómez, *Robot insecte intel·ligent*. Mataró, EUPMT 2000.
- [2] Isaac Hernàndez Izquierdo, *Construcció d'un Robot Quadrúped*. Mataró, EUPMT 2002.
- [3] Marc Rivas Capafons, *Disseny i creació d'un microbot insect autònom*. Mataró, EUPMT 2005.
- [4] Keila Belando Baca, *Robot hexàpode "Mordor"*. Mataró, EUPMT 2005.
- [5] Jordi Estrella Domene, *Diseño y Construcción de un Hexápodo Autónomo*. Mataró, EUPMT 2007.

### 9.2 Libros

- [1] Joseph L. Jones, Anita M. Flynn, Bruce A. Seiger, *Mobile Robots. Inspiration to Implementation*. Segunda edición. Natick, Massachusetts, A K Peters 1999.
- [2] José M<sup>a</sup> Angulo Usategui, Susana Romero Yesa, Ignacio Angulo Martínez, *Microbótica. Tecnología, aplicaciones y montaje práctico*. Segunda edición. Madrid, Paraninfo Thomson Learning 2001.
- [3] José M<sup>a</sup> Angulo Usategui, Susana Romero Yesa, Ignacio Angulo Martínez, *Microcontroladores PIC. Diseño práctico de aplicaciones. Segunda parte*. Madrid, Mc Graw Hill 2000.
- [4] José M<sup>a</sup> Angulo Usategui, Ignacio Angulo Martínez, *Microcontroladores PIC. Diseño práctico de aplicaciones. Primera parte*. Tercera edición. Madrid, Mc Graw Hill 2003.

## 9.3 Data Sheets y Notas de Aplicación

- [1] Microchip Technology Inc. *AN515. Communicating with the I<sup>2</sup>C Bus Using the PIC16C5X*. DS00515D. 1993.
- [2] National Semiconductor Corporation. *LM78XX Series Voltage Regulators*. 1995.
- [3] National Semiconductor Corporation. *LM117/LM317A/LM317 3-Terminal Adjustable Regulator*. 1996.
- [4] Microchip Technology Inc. *AN578. Use of the SSP Module in the I<sup>2</sup>C Multi-Master Environment*. DS00578B. 1997.
- [5] Microchip Technology Inc. *AN554. Software Implementation of I<sup>2</sup>C Bus Master*. DS00554C. 1997.
- [6] Vishay Semiconductor GmbH. *CNY70. Reflective Optical Sensor with Transistor Output*. 83751. 2000.
- [7] Microchip Technology Inc. *PIC16F84A Data Sheet*. DS35007B. 2001.
- [8] Maxim Integrated Products. *MAX220 – MAX249. +5V-Powered, Multichannel RS-232 Drivers/Receivers*. 2001.
- [9] Analog Devices, Inc. *AD5263. Quad, 15 V, 256-Position, Digital Potentiometer with Pin Selectable SPI/I<sup>2</sup>C*. 2003.
- [10] Microchip Technology Inc. *PIC16F87/88 Data Sheet*. DS30487C. 2005.
- [11] Microchip Technology Inc. *PIC16F87/88 Rev. B1 Silicon Errata*. DS80171K. 2006.
- [12] Microchip Technology Inc. *AN734. Using the PIC Devices' SSP and MSSP Modules for Slave I<sup>2</sup>C Communication*. DS00734B. 2008.

## 9.4 Artículos y otras publicaciones

- [1] Enric Celaya, José Luis Albarral. *Implementation of a hierarchical walk controller for the LAURON III hexapod robot.* IRI, Institut de Robòtica i Informàtica Industrial (UPC-CSIC), 2003.
- [2] Néstor Sorli Martínez de Oraa, Pedro Fernández Gómez. *Robot Hexápodo.* Escola Universitària Politècnica de Vilanova i la Geltrú (UPC), 2003.
- [3] Sergi Gomis Gascó (Club de Robòtica Autònoma). *Curs de Robòtica (rev.2).* EUPMT, 2006.
- [4] A. Schneider, U. Schmucker. *Force Sensing for Multi-legged Walking Robots: Theory and Experiments – Part 1: Overview and Force Sensing.*  
(Fuente: *Mobile Robots, Moving Intelligence.* Jonas Buchli, 2006.)
- [5] Alejandro Arango Saavedra, Héctor Fabio Satizábal Mejía, Humberto Loaiza Correa. *Diseño e implementación de un robot móvil hexápodo.* Universidad del Valle, Colombia, 2006.
- [6] Shota Fujii, Kenji Inoue, Tomohito Takubo, Tatsuo Arai. Climbing up onto Steps for Limb Mechanism Robot “ASTERISK”. Universidad de Osaka, 2006.
- [7] Alberto Sánchez Espinosa, Carlos López Cardelo. *Diseño de un Robot Hexápodo. Hardware y Software de Control.* Escola Universitària Politècnica de Vilanova i la Geltrú (UPC).

## 9.5 Páginas web

- [1] CRA (Club de Robòtica Autònoma) de la EUPMT.  
<http://www.eupmt.es/cra>
- [2] Farnell, distribuidor de productos electrónicos a nivel mundial.  
<http://es.farnell.com>
- [3] RS-Amidata, distribuidor de productos electrónicos a nivel mundial.  
<http://es.rs-online.com/web>

- [4] Microchip, fabricante de los microcontroladores PIC.  
<http://www.microchip.com>
- [5] Futaba, fabricante de servomotores de modelismo.  
<http://www.futaba-rc.com>
- [6] WinPic800, software empleado para programar los PICs.  
<http://www.winpic800.com>
- [7] Google SketchUp, software empleado para el diseño mecánico en 3D.  
<http://sketchup.google.com>
- [8] EditPlus 2, software empleado para escribir el código fuente.  
<http://www.editplus.com>
- [9] Cadence OrCAD, paquete de software empleado para diseñar la electrónica.  
<http://www.cadence.com/orcad>
- [10] Mundobot, web dedicada al robot Melanie.  
<http://mundobot.com/projects/melanie/spmelanie>
- [11] Mike Smyth's Hexapod Robot.  
<http://home.ctlnet.com/~robotguy67/hexapod/hexapod>
- [12] Koji's Hexapod Robot.  
[http://www2.plala.or.jp/k\\_y\\_yoshino/6legs/6legs\\_top\\_e](http://www2.plala.or.jp/k_y_yoshino/6legs/6legs_top_e)
- [13] Micromagic Systems, web del creador de robots Matt Denton.  
<http://www.micromagicsystems.com>
- [14] Vídeos de los robots de Matt Denton.  
<http://www.youtube.com/user/winchymatt>
- [15] Projekt Marvin.  
<http://www.vreal.de>
- [16] Trossen Robotics, robot Phoenix.  
<http://forums.trossenrobotics.com/showthread.php?t=1352>
- [17] Trossen Robotics, robot Oxyopus.  
<http://forums.trossenrobotics.com/gallery/showimage.php?i=101>
- [18] Robot Bill-Ant.  
<http://biorobots.cwru.edu/projects/billant>

## Bibliografía

---

- [19] Robot RHex.  
<http://www.rhex.web.tr>
- [20] Robot Arachnobot.  
<http://www.pdjinc.com/arachno>
- [21] Arai Lab., Universidad de Osaka, robot Asterisk.  
[http://www-arailab.sys.es.osaka-u.ac.jp/research/limbgroup/e\\_index](http://www-arailab.sys.es.osaka-u.ac.jp/research/limbgroup/e_index)
- [22] IRI (Institut de Robòtica i Informàtica Industrial, UPC), robot Lauron III.  
[http://www-iri.upc.es/groups/arobots/legged\\_robots/robots](http://www-iri.upc.es/groups/arobots/legged_robots/robots)
- [23] Robots Argentina, completa web sobre robótica.  
<http://robots-argentina.com.ar/robots>
- [24] Robot AMOS-WD06  
<http://www.chaos.gwdg.de/~poramate/AMOSWD06>
- [25] Walking Machines, gran catálogo de robots caminadores  
<http://www.walking-machines.org>
- [26] Robotis, fabricante del kitt Biloid.  
<http://www.robotis.com>
- [27] Robot Shop, tienda especializada en materiales para la robótica.  
<http://www.robotshop.ca>
- [28] SuperRobótica, tienda especializada en materiales para la robótica  
<http://www.superrobotica.com>

[Página en blanco, Contraportada]

## Bibliografía

---

[Página en blanco, para que los anexos vaya en página par]



Escola Universitària  
Politècnica de Mataró

**Ingeniería Técnica Industrial: Especialidad Electrónica Industrial**

**PROYECTO MIHRO**  
**(MOBILE INTELLIGENT HEXAPOD ROBOT)**  
**VOLUMEN II**

**IGNACIO PEDROSA LOJO**  
**JULIÁN HORRILLO TELLO**

OTOÑO 2008





Escola Universitària  
Politècnica de Mataró



**VOLUMEN II**  
**PRESUPUESTO**

**IGNACIO PEDROSA LOJO**  
**JULIÁN HORRILLO TELLO**



## Índice de presupuesto

1	Desglose de material	VI
2	Mediciones	VII
3	Cuadro de precios	VII
4	Presupuesto parcial	VIII
5	Presupuesto global	IX

# 1 Desglose de material

Concepto	Cantidad	Precio Unitario (€)	Total (€)
<b>Componentes mecánicos</b>			
Chapa de aluminio 2'5mm (50x50cm)	1	26,00	26,00
Chapa de aluminio 2mm (50x50cm)	1	22,00	22,00
Chapa de aluminio 1mm (30x30cm)	1	15,00	15,00
Tornillería (20 unidades)	13	2,50	32,50
Tubo de aluminio para separadores	1	2,50	2,50
Tacos de goma antideslizante	12	0,60	7,20
Muelles	12	0,20	2,40
Tubos de latón cromado	6	7,20	43,20
Araldit Standard	2	5,70	11,40
Araldit Rápido	1	5,60	5,60
<b>Componentes electromecánicos</b>			
Sensores de contacto Mini-bumper	12	1,25	15,00
Servos Standard Futaba S3003	22	10,00	220,00
Teclado numérico	1	3,00	3,00
Pad videoconsola	1	8,50	8,50
<b>Componentes electrónicos</b>			
Placa de cobre de doble cara	6	4,89	29,34
Placa de topos de Baquelita (100x170)	4	5,29	21,16
Estaño para soldadura 1mm	2	3,52	7,04
Estaño para soldadura 0,5mm	1	3,79	3,79
PIC16F88	10	5,32	53,21
PIC16F84A	2	3,61	7,22
MAX232	1	1,22	1,22
L7805	2	0,19	0,39
Zócalos	10	-	0,43
Xtal oscilador 20MHz	8	0,38	3,04
Resistencias	76	0,16	12,16
Condensadores	32	-	1,59
LEDs	26	-	1,38
Pulsadores	13	0,22	2,86
Interruptores	5	-	3,61
Conectores	87	-	27,57
Cables (paquetes de 1 metro)	12	-	14,48
Otros (miscelánea)	-	-	1,32
<b>Subtotal</b>			<b>606,11</b>

## 2 Mediciones

Ref.	Descripción	Partes iguales
1	Horas de proyecto destinadas a la búsqueda de información y antecedentes	30
2	Horas de proyecto destinadas al diseño electrónico y mecánico	100
3	Horas de proyecto destinadas a la fabricación y montaje de los elementos mecánicos y electrónicos	70
4	Horas de proyectos destinadas al desarrollo del software	50
5	Horas de proyecto destinadas a la puesta en marcha de todo el conjunto	30
6	Horas de proyecto destinadas a la elaboración escrita de la memoria del proyecto	20
7	Soporte para el almacenamiento de información y copias de seguridad (CDs)	6
8	Impresión de los documentos del proyecto.	255
9	Impresión de documentos de soporte y versiones preliminares	400
10	Encuadernación de los documentos del proyecto.	2
11	Desplazamientos varios (Onda Radio BCN)	1
12	Minutos de conexión telefónica a Internet	1800

## 3 Cuadro de precios

Ref.	Unidades	Precio unitario (€)
1	Horas	25
2	Horas	50
3	Horas	25
4	Horas	50
5	Horas	50
6	Horas	25
7	CDs	0,5
8	Hojas en calidad	0,45
9	Hojas en borrador	0,15
10	Encuadernación	7
11	Billetes de Cercanías (T10 3 zonas)	19,80
12	Minutos	0,03

## 4 Presupuesto parcial

Ref.	Descripción	Partes iguales	Precio unitario (€)	Importe (€)
1	Horas de proyectista destinadas a la búsqueda de información y antecedentes	30	25	750
2	Horas de proyectista destinadas al diseño electrónico y mecánico	100	50	5.000
3	Horas de proyectista destinadas a la fabricación y montaje de los elementos mecánicos y electrónicos	70	25	1.750
4	Horas de proyectistas destinadas al desarrollo del software	50	50	2.500
5	Horas de proyectista destinadas a la puesta en marcha de todo el conjunto	30	50	1.500
6	Horas de proyectista destinadas a la elaboración escrita de la memoria del proyecto	20	25	500
7	Soporte para el almacenamiento de información y copias de seguridad (CDs)	6	0,5	3
8	Impresión de los documentos del proyecto.	255	0,45	114,75
9	Impresión de documentos de soporte y versiones preliminares	400	0,15	60
10	Encuadernación de los documentos del proyecto.	2	7	14
11	Desplazamientos varios (Onda Radio BCN)	1	19,80	19,80
12	Minutos de conexión telefónica a Internet	1800	0,03	54
<b>Subtotal</b>				<b>12.265,55</b>

## 5 Presupuesto global

Concepto	Cantidad (€)
Total Material	606,11 €
Total Presupuesto Parcial	12.265,55 €
Total	12.871,66 €
IVA (16%)	2.059,47 €
<b>Total Presupuesto</b>	<b>14.931,13 €</b>

La elaboración del presente proyecto genera unos gastos de catorce mil novecientos treinta y uno con trece euros.





Escola Universitària  
Politècnica de Mataró

**VOLUMEN II**

**PLANOS**

**IGNACIO PEDROSA LOJO  
JULIÁN HORRILLO TELLO**

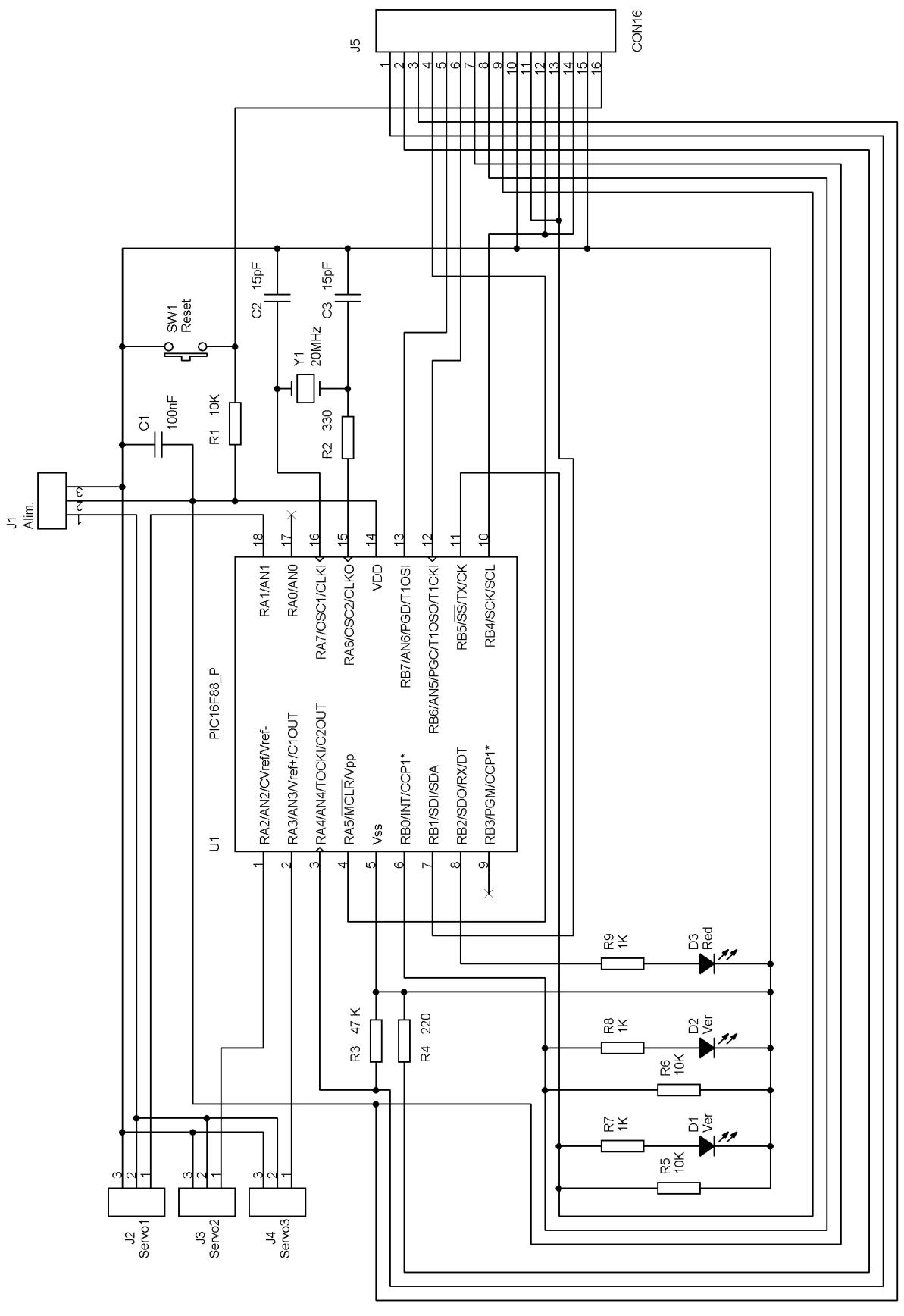


## Índice de planos

<b>1</b>	<b>Esquemáticos</b>	<b>IV</b>
1.1	Esquemático de las placas “Patas”	IV
1.2	Esquemático de la placa Entrenadora	VI
1.3	Esquemático de la placa de Alimentación	VIII
1.4	Esquemático de la programadora PIPO2	X
<b>2</b>	<b>Serigrafías</b>	<b>XII</b>
2.1	Placa Entrenadora, TOP	XII
2.2	Placa Entrenadora, BOTTOM	XIII
2.3	Placa “Patas”, grupo A, TOP	XIV
2.3.1	Escala 1:1	XIV
2.3.2	Escala 3:1	XIV
2.4	Placa “Patas”, grupo A, BOTTOM	XV
2.4.1	Escala 1:1	XV
2.4.2	Escala 3:1	XV
2.5	Placa “Patas”, grupo A, SST	XVI
2.5.1	Escala 1:1	XVI
2.5.2	Escala 3:1	XVI
2.6	Placa “Patas”, grupo B, TOP	XVII
2.6.1	Escala 1:1	XVII
2.6.2	Escala 3:1	XVII
2.7	Placa “Patas”, grupo B, BOTTOM	XVIII
2.7.1	Escala 1:1	XVIII
2.7.2	Escala 3:1	XVIII
2.8	Placa “Patas”, grupo B, SST	XIX
2.8.1	Escala 1:1	XIX
2.8.2	Escala 3:1	XIX
<b>3</b>	<b>Planos mecánicos</b>	<b>XX</b>
3.1	P00: Tórax Base	XX
3.2	P01: Tapa Superior del Tórax	XXII
3.3	P02: Tapa Inferior del Tórax	XXIV
3.4	P03: Conector Cadera-Tórax y P04: Conector Cadera-Fémur	XXVI
3.5	P05: Fémur	XXVIII
3.6	P06: Parte Fija de la Tibia	XXX
3.7	P07: Soporte Superior del Pié y P08: Parte Móvil de la Tibia	XXXII
3.8	P09: Pié y P10: Cuello 1	XXXIV
3.9	P11: Hombro Izquierdo y P12: Hombro Derecho	XXXVI

# 1 Esquemáticos

## 1.1 Esquemático de las placas “Patas”



Asignatura de PFC
Proyecto MIHRO
Esquemático Placa Patas
Otoño 2008
1
de
3



1

3

4

5

A

D

C

B

A

1

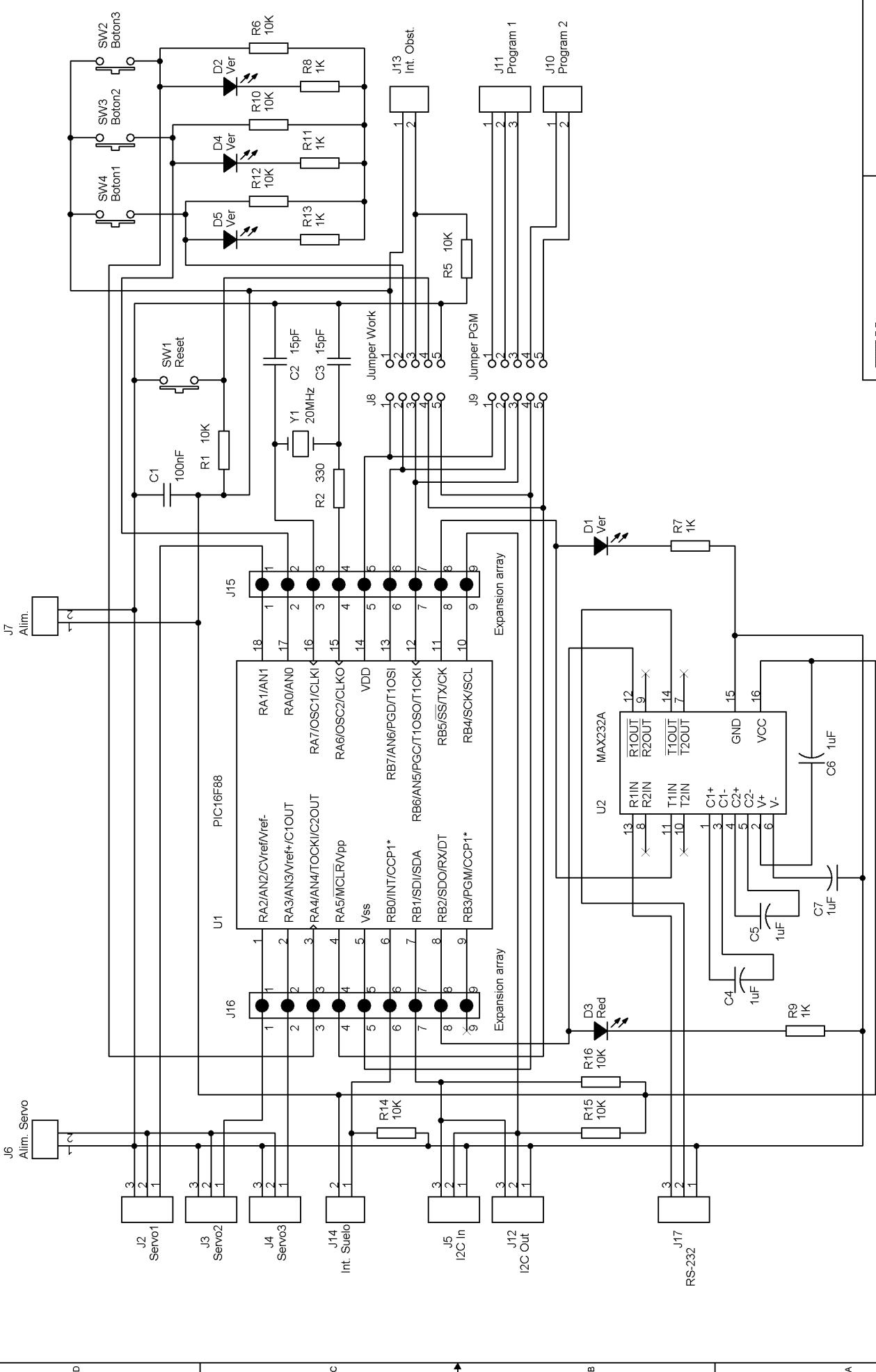
2

3

4

5

## **1.2 Esquemático de la placa Entrenadora**



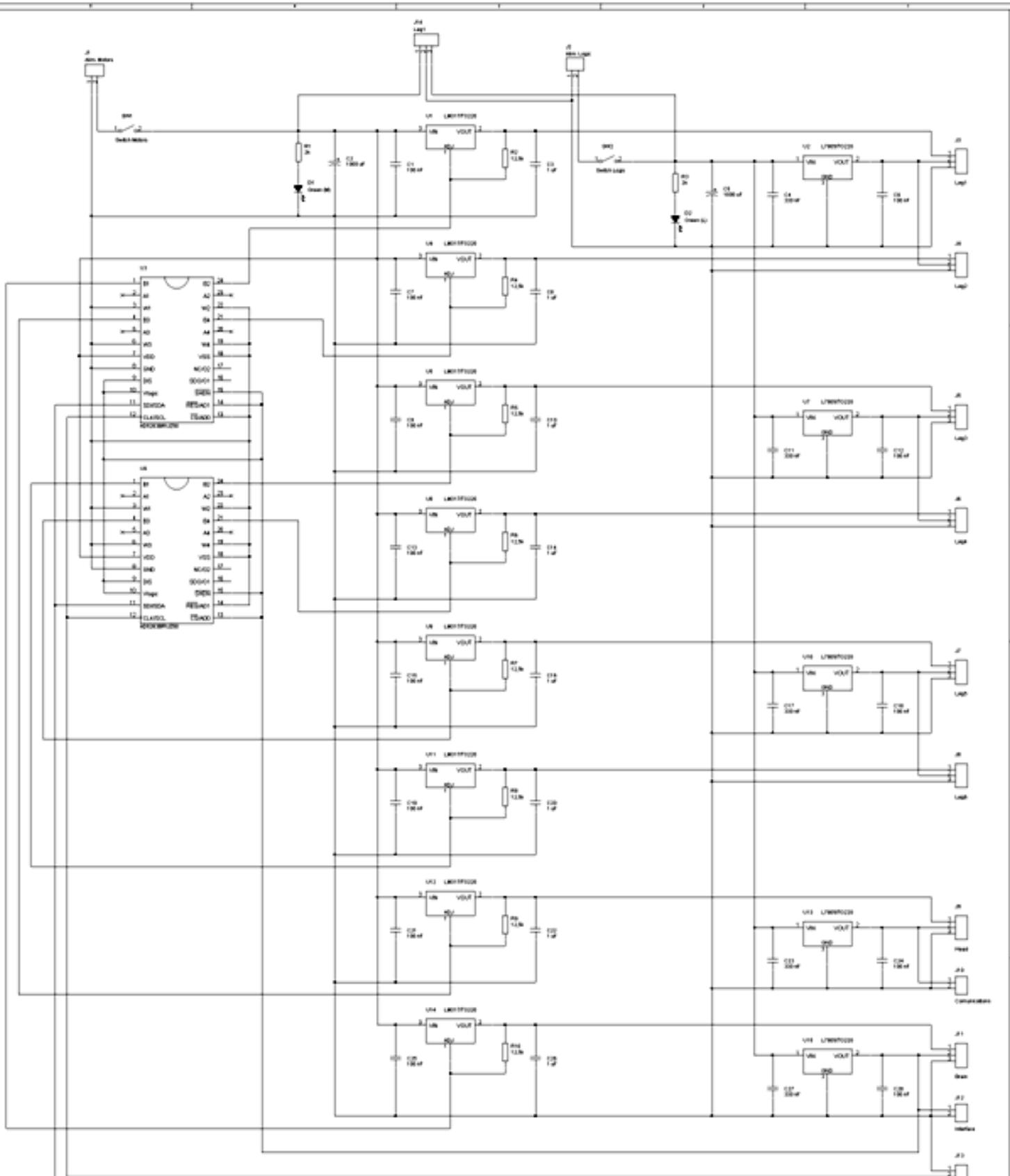
Asignatura de PFC  
Proyecto MIHRO

Esquemático Placa Entrenadora  
Otoño 2008 2 de 3

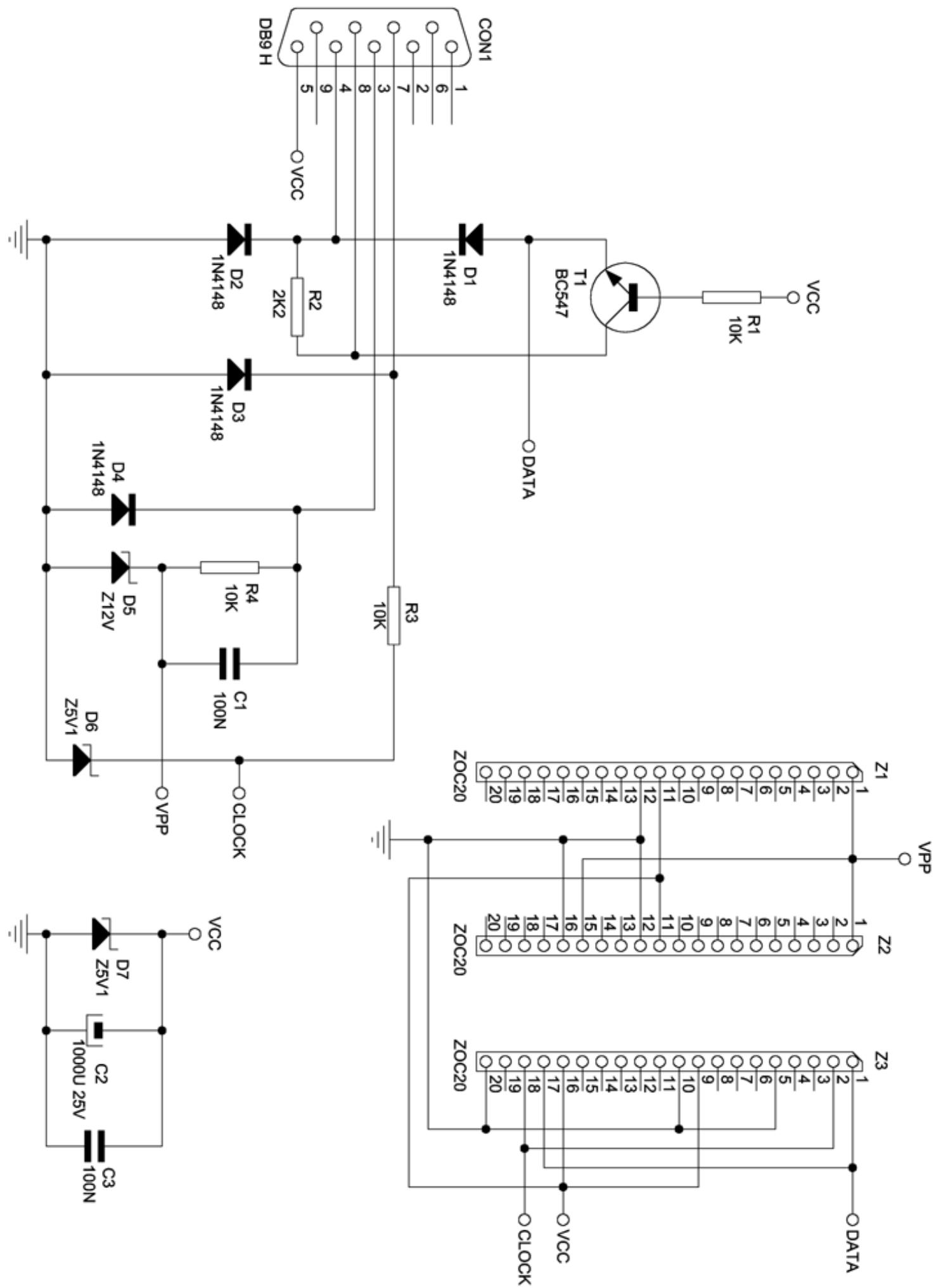


Escola Universitària  
Politècnica de Mataró

### **1.3 Esquemático de la placa de Alimentación**

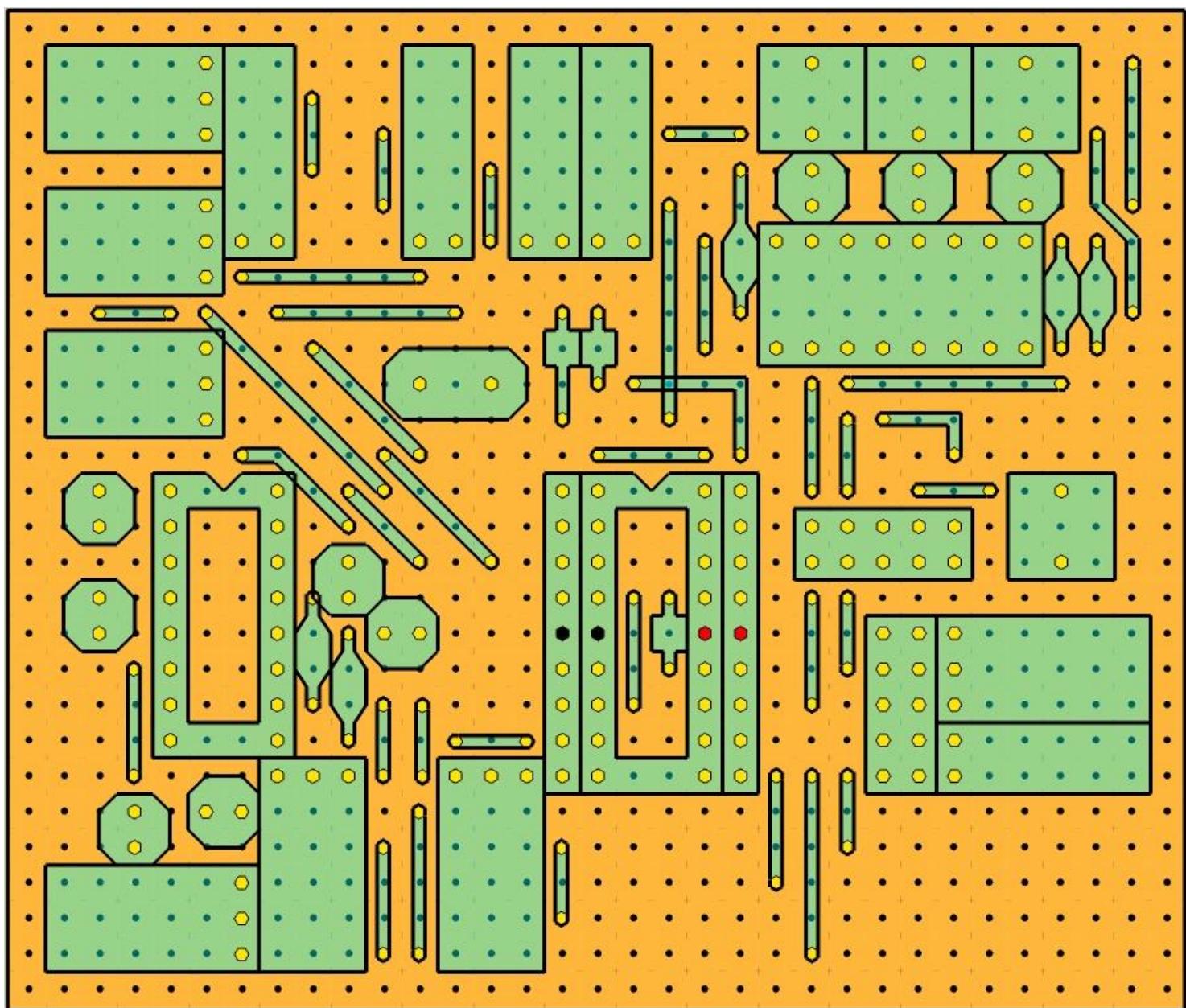


## **1.4 Esquemático de la programadora PIPO2**

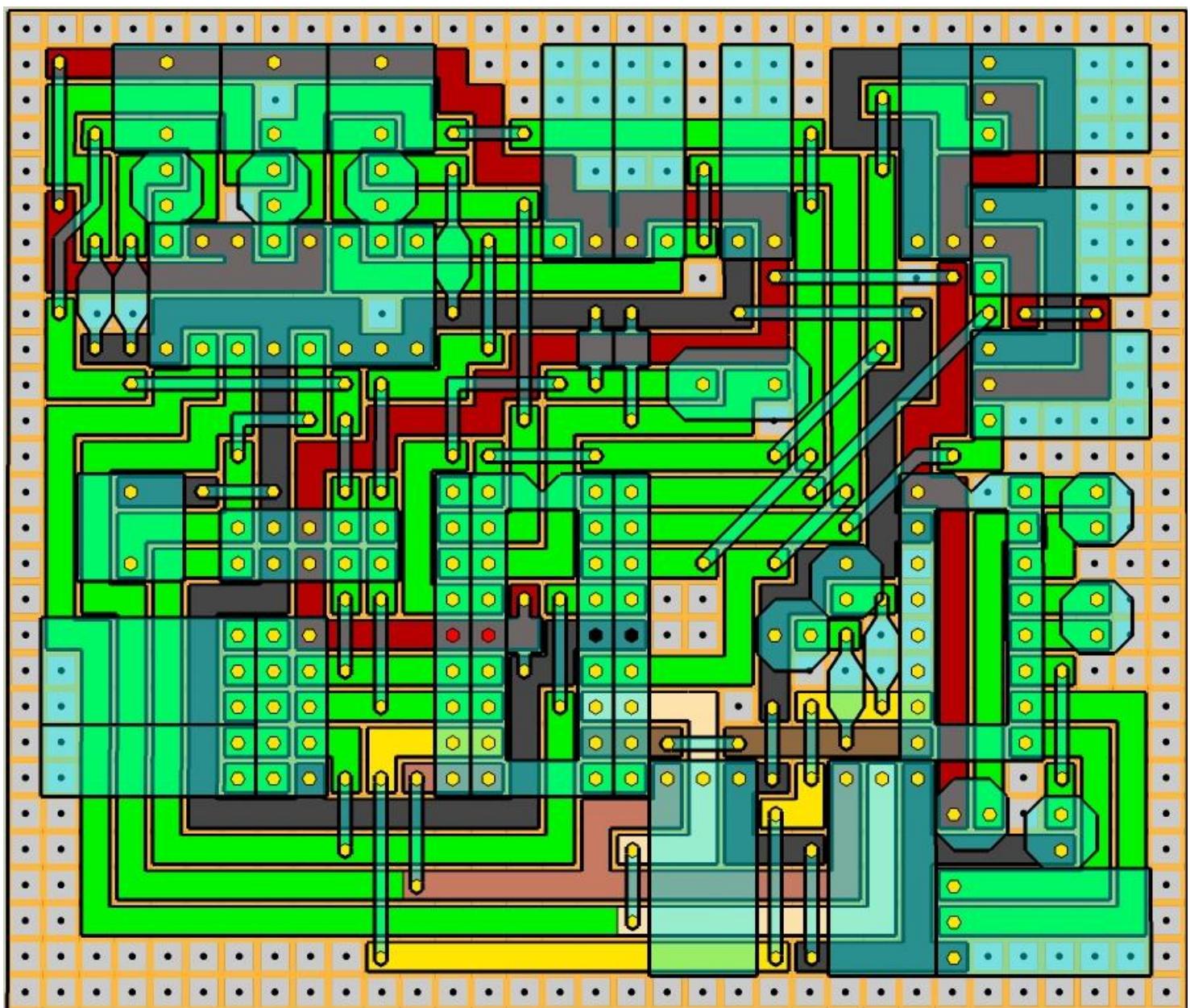


## 2 Serigrafías

### 2.1 Placa Entrenadora, TOP

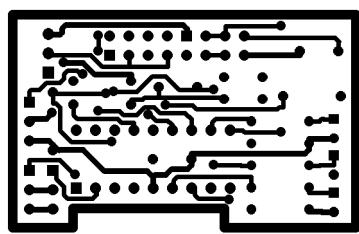


## 2.2 Placa Entrenadora, BOTTOM

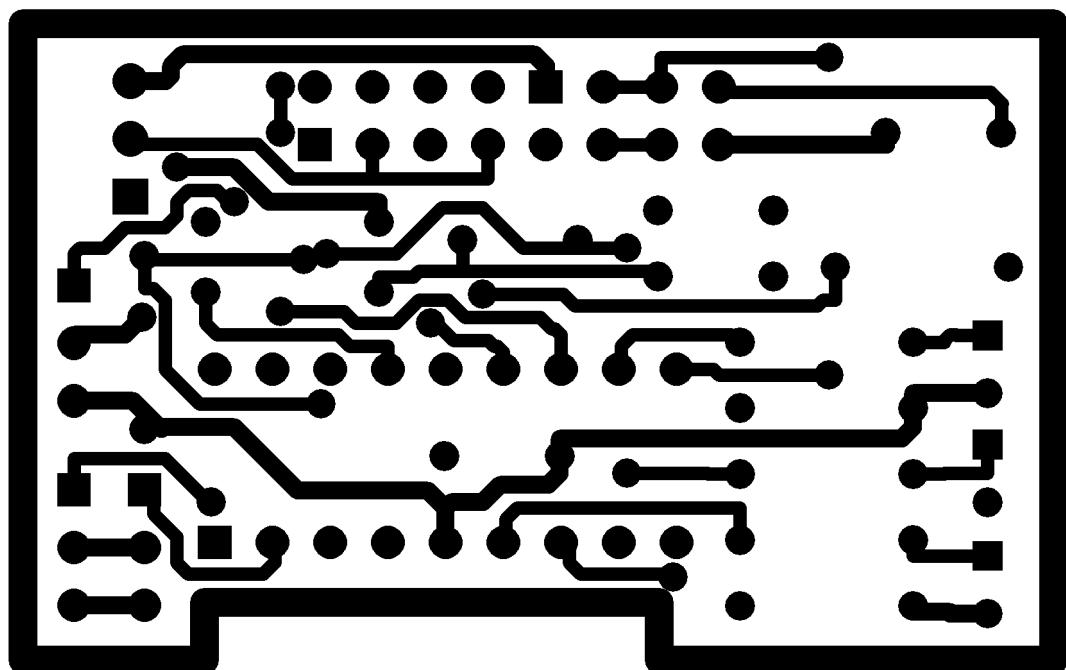


## 2.3 Placa “Patas”, grupo A, TOP

2.3.1 Escala 1:1

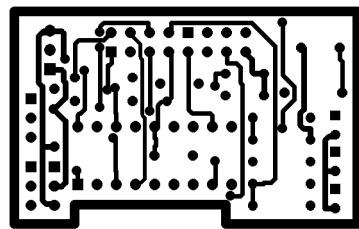


2.3.2 Escala 3:1

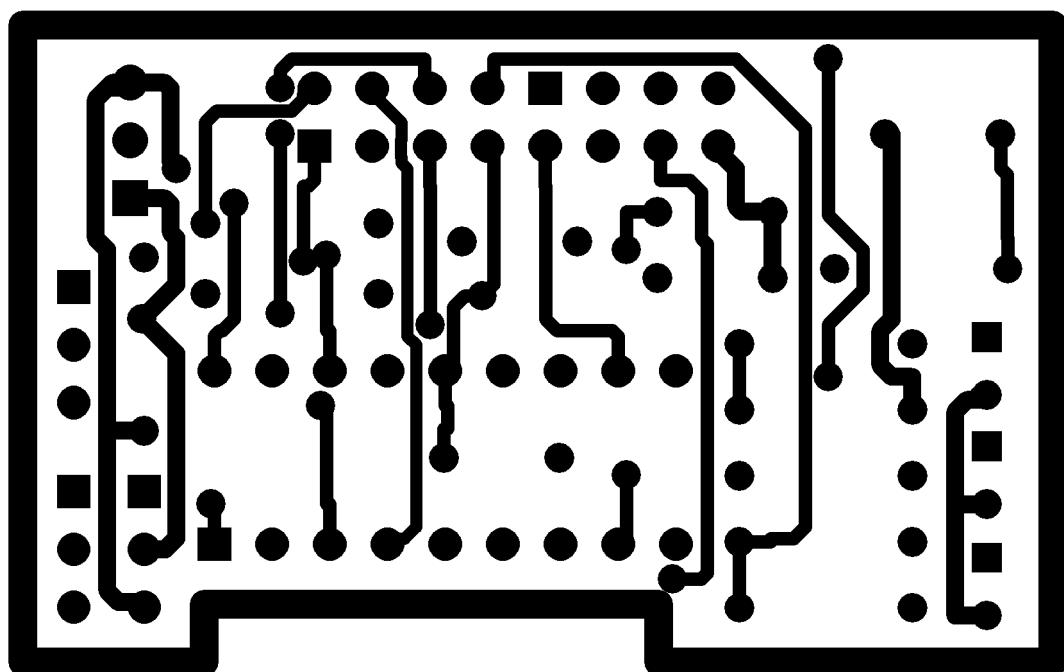


## 2.4 Placa “Patas”, grupo A, BOTTOM

2.4.1 Escala 1:1

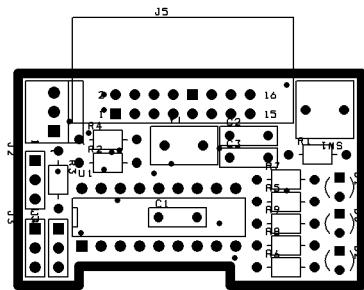


2.4.2 Escala 3:1

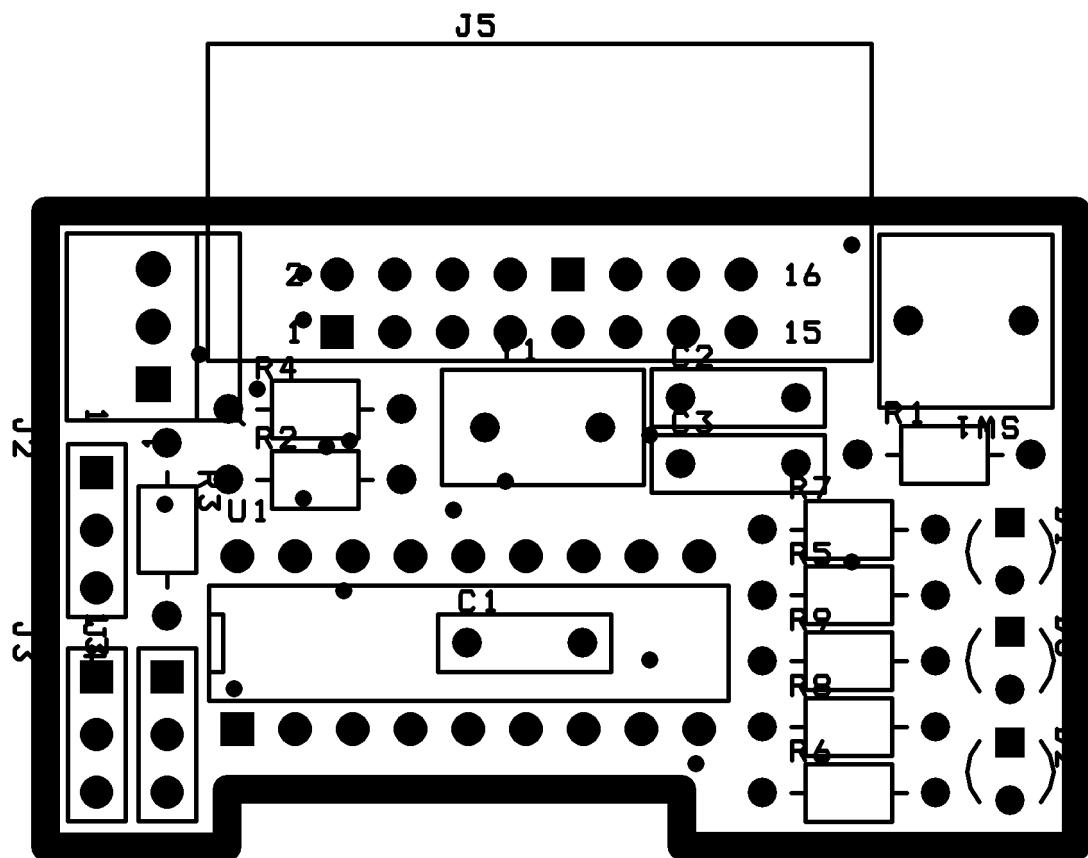


## 2.5 Placa “Patas”, grupo A, SST

### 2.5.1 Escala 1:1

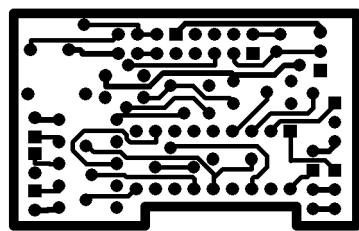


### 2.5.2 Escala 3:1

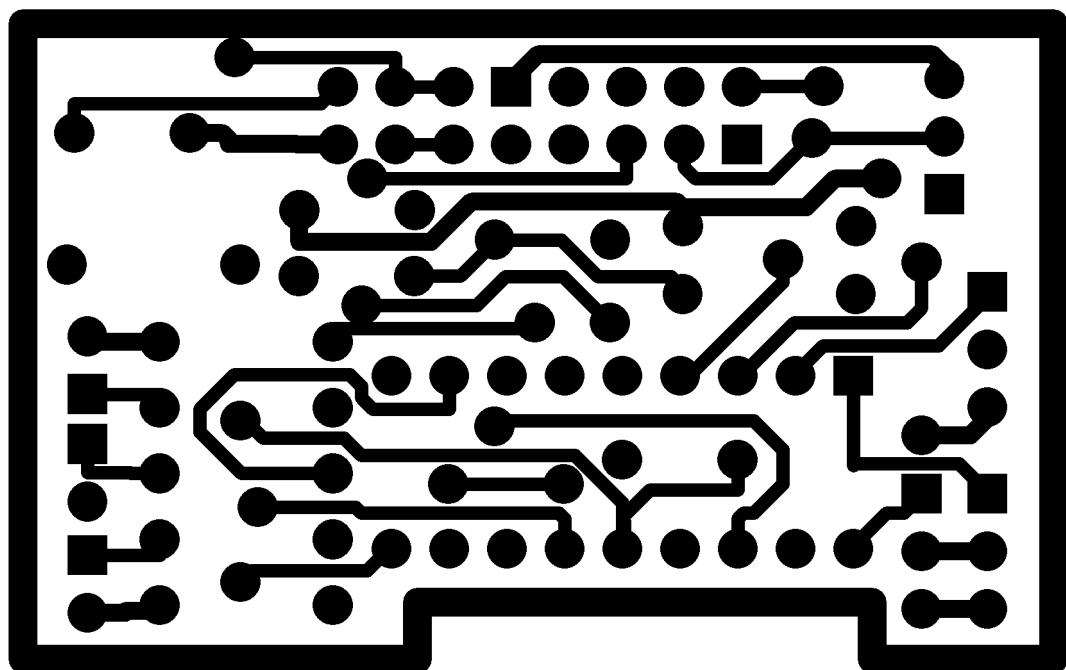


## 2.6 Placa “Patas”, grupo B, TOP

2.6.1 Escala 1:1

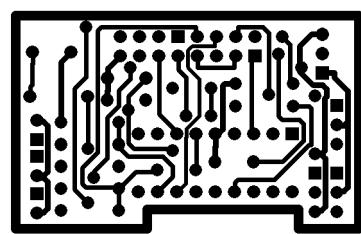


2.6.2 Escala 3:1

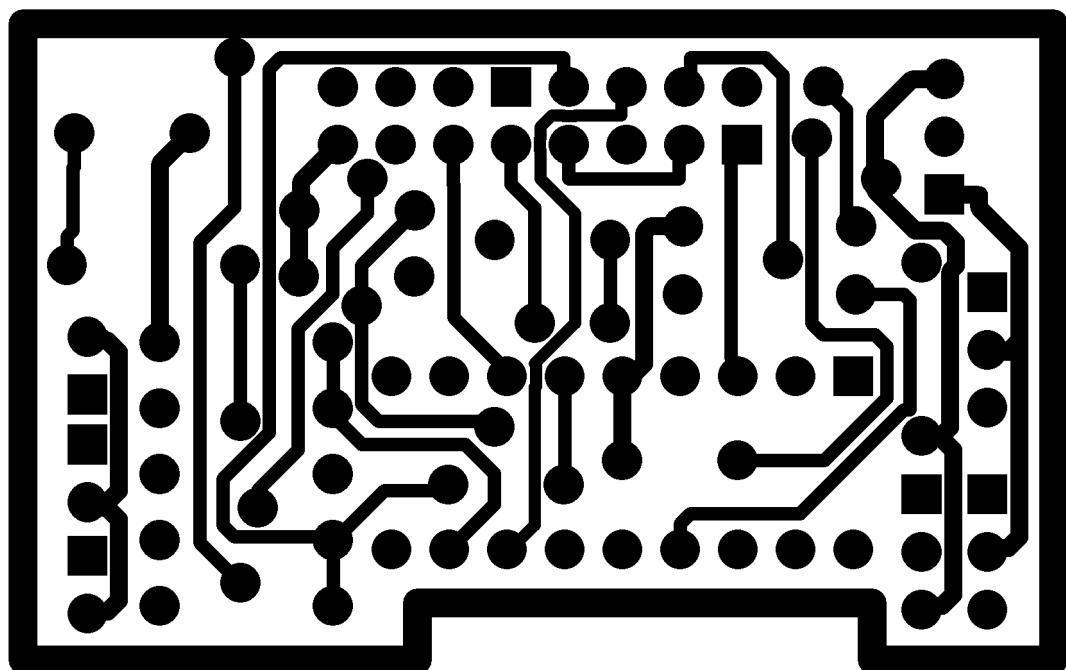


## 2.7 Placa “Patas”, grupo B, BOTTOM

2.7.1 Escala 1:1

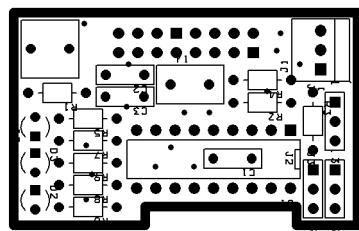


2.7.2 Escala 3:1

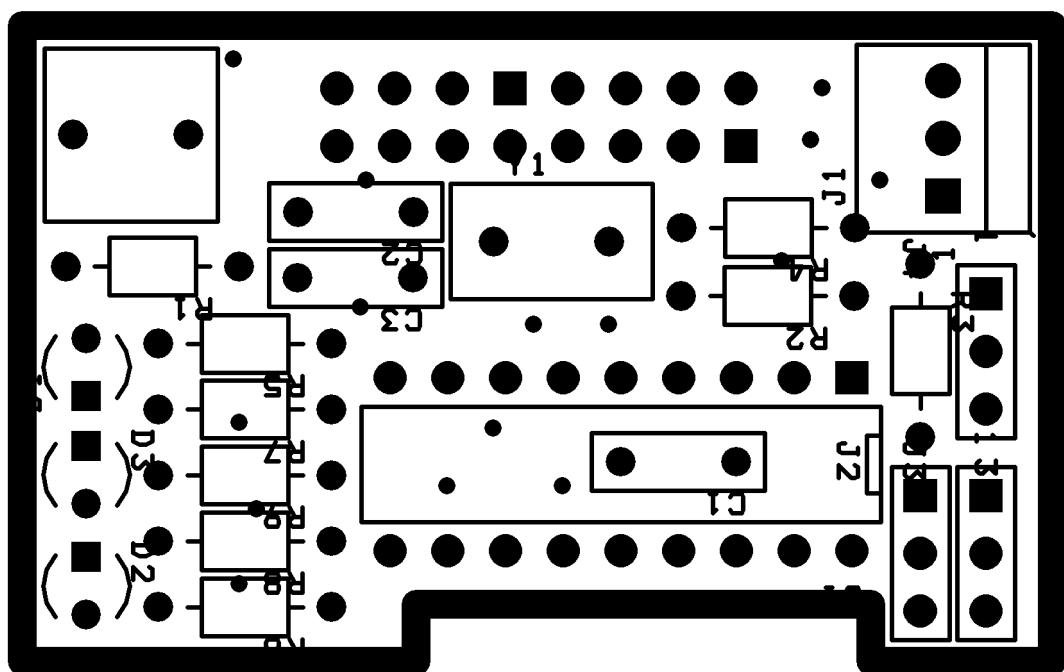


## 2.8 Placa “Patas”, grupo B, SST

### 2.8.1 Escala 1:1



### 2.8.2 Escala 3:1



### 3 Planos mecánicos

#### 3.1 P00: Tórax Base



### **3.2 P01: Tapa Superior del Tórax**



### **3.3 P02: Tapa Inferior del Tórax**



**3.4 P03: Conector Cadera-Tórax y P04: Conector  
Cadera-Fémur**



### **3.5 P05: Fémur**



### **3.6 P06: Parte Fija de la Tibia**



**3.7 P07: Soporte Superior del Pié y P08: Parte  
Móvil de la Tibia**



### **3.8 P09: Pié y P10: Cuello 1**



**3.9 P11: Hombro Izquierdo y P12: Hombro Derecho**







Escola Universitària  
Politècnica de Mataró

**VOLUMEN II**

**CÓDIGO FUENTE**

**IGNACIO PEDROSA LOJO  
JULIÁN HORRILLO TELLO**



# Índice de código fuente

<b>1 Código fuente de la pata 3</b>	<b>IV</b>
<i>Fuses de configuración</i>	V
Variables en memoria	V
Vectores	VII
Configuración de puertos y registros	VII
Inicialización de variables	VIII
Bucle principal	IX
Rutina "ciclo"	IX
Rutina "movs"	X
Rutina "fluc"	XIX
Rutina "find_m3"	XX
Rutina "delay"	XXI
Comunicación I <sup>2</sup> C Slave	XXII
Servicio de interrupción	XXVI
<b>2 Código fuente de la placa Master</b>	<b>XXVIII</b>
<i>Fuses de configuración</i>	XXIX
Variables en memoria	XXIX
Vectores	XXXI
Configuración de puertos y registros	XXXI
Inicialización de variables	XXXII
Bucle principal	XXXIII
Rutina de acción de botones	XXXIV
Rutina "led1"	XLI
Rutina "led2"	XLI
Rutina "led3"	XLII
Rutina "led4"	XLIII
Rutina "led5"	XLIII
Rutina "delay"	XLIV
Comunicación I <sup>2</sup> C Master	XLV
Servicio de interrupción	LII

# 1 Código fuente de la pata 3

Este es un código fuente en fase de pruebas. Su versión definitiva se adjunta en el CD.

```

;----- Encabezado -----
;
; Archivo:      pata_iglu_R3.asm
; Autor:        Ignacio Pedrosa Lojo
; Versión:      1.00
;
; Descripción: Programa de control y comunicación de la pata 3
;                 Incluye el código para la gestión de los motores,
;                 las variables de control del movimiento, la rutina
;                 de retraso adaptada al servo Futaba S3003, la
;                 gestión de la comunicación I2C en modo esclavo y la
;                 gestión del Sistema Interruptivo.
;
; Ensamblado mediante Microchip Assembler v5.00 (MPASM)
; El archivo "P16F88.inc" proporcionado por Microchip debe ser
; incluido en el directorio de trabajo en caso de no usar MPASM v5.00
;
;-----
;

LIST      P=16F88
INCLUDE    "P16F88.INC"
; Registro 1 de Configuración
__CONFIG   _CONFIG1, _CP_OFF
            & _CCP1_RB3           ; CCP1 en el pin RB3
            & _DEBUG_OFF          ; modo depuración OFF
            & _WRT_PROTECT_OFF   ; Protección de escritura OFF
            & _CPD_OFF            ; Code protection OFF
            & _LVP_OFF             ; Low voltage programming OFF
            & _BODEN_ON           ; Brown-out Reset enabled
            & _MCLR_ON             ; Botón de RESET activado (;!)
            & _PWRTE_ON            ; Power-Up timer enabled
            & _WDT_OFF              ; Watch Dog Timer OFF
            & _HS_OSC                ; HS Oscillator Mode

; Registro 2 de Configuración
__CONFIG   _CONFIG2, _IESO_OFF
            & _FCMEN_OFF

;----- Variables en memoria -----
cont      EQU      h'20'
del       EQU      h'21'
cont2     EQU      h'22'
del2      EQU      h'23'
cont3     EQU      h'24'
del3      EQU      h'25'
m1        EQU      h'26'
m2        EQU      h'27'
m3        EQU      h'28'
action    EQU      h'29'
w_temp    EQU      h'2A'
stat_t    EQU      h'2B'
pcl_t    EQU      h'2C'
mot       EQU      h'2D'
mot2      EQU      h'2E'
fst        EQU      h'2F'
speed     EQU      h'30'
centro   EQU      h'31'
rango     EQU      h'32'
zanco    EQU      h'33'
altura   EQU      h'34'
m1_max   EQU      h'35'
m1_min   EQU      h'36'
m2_max   EQU      h'37'

```

```
m2_min EQU h'38'
m3_max EQU h'39'
m3_min EQU h'3A'
umax_m1 EQU h'3B'
umin_m1 EQU h'3C'
usup_m1 EQU h'3D'
uinf_m1 EQU h'3E'
usup_m2 EQU h'3F'
umed_m2 EQU h'40'
uinf_m2 EQU h'41'
upos_m3 EQU h'42'
mot3 EQU h'43'
meta EQU h'44'
var EQU h'45'
comp EQU h'46'
mode EQU h'47'
r_m1 EQU h'48'
r_m2 EQU h'49'
r_m3 EQU h'4A'
mot4 EQU h'4B'

; hasta 7Fh

; la variable 'action' contiene 8 banderas:
; action,0 : 1= Go 0= Stop
; action,1 : 1= Atrás 0= Delante
; action,2 : 1= Giro Izquierda 0= No Giro
; action,3 : 1= Giro Derecha 0= No Giro
; action,4 : 1= Rotación Izq. 0= No Rotación
; action,5 : 1= Rotación Dch. 0= No Rotación
; action,6 : 1= Ida del paso 0= No Ida
; action,7 : 1= Vuelta del paso 0= No vuelta

; la variable 'mode' contiene 8 banderas:
; mode,0 : 1= Modo Dinámico 0= Modo Estático
; mode,1 : 1= Modo levantamiento 0= No Modo
; mode,2 : 1= Modo Cojer 0= No Modo
; mode,3 : 1= Modo Inicial 0= No Modo
; mode,4 : 1= Modo Programación 0= No Modo
; mode,5 : 1= Obstáculo detec. 0= No Obstáculo
; mode,6 : 1= 0=
; mode,7 : 1= 0=

; la variable 'mot' contiene 8 banderas:
; mot,0 = LED indicador
; mot,1 = Recibir Speed / enviar mot2 (I2C)
; mot,2 = Recibir Centro / enviar mode (I2C)
; mot,3 = Recibir Rango / enviar Zanco (I2C)
; mot,4 = rutina ciclo
; mot,5 = rutina movs
; mot,6 = posicionamiento inicial
; mot,7 = secuencia de levantamiento

; la variable 'mot2' contiene 8 banderas:
; mot2,0 = Etapa 0
; mot2,1 = Etapa 1
; mot2,2 = Etapa 2
; mot2,3 = Etapa 3
; mot2,4 = Etapa 4
; mot2,5 = Etapa 5
; mot2,6 = Etapa 6
; mot2,7 = Etapa 7
```

```

; la variable 'mot3' contiene 8 banderas:
; mot3,0 = Fluctuación terminada
; mot3,1 = m1 terminado
; mot3,2 = m2 terminado
; mot3,3 = m3 terminado
; mot3,4 = estado anterior en m1,3
; mot3,5 = flag de aplicar el factor corrector de m3
; mot3,6 = Recibir Zanco / enviar altura (I2C)
; mot3,7 = Recibir Altura

; la variable 'mot4' contiene 8 banderas:
; mot4,0 = Recibir mode
; mot4,1 = Recibir action
; mot4,2 = Recibir r_m1
; mot4,3 = Recibir r_m2
; mot4,4 = Recibir r_m3
; mot4,5 =
; mot4,6 =
; mot4,7 =

;----- Vectores de inicio e interrupción -----
ORG 0x00
GOTO inicio
ORG 0x04
GOTO isr
ORG 0X05

;----- Configuración de puertos y registros -----
inicio BSF STATUS,RPO      ; banco mem 1
        MOVLW b'00000000'
        MOVWF ANSEL      ; ADC desactivado
        MOVLW b'10110000'
        MOVWF TRISA      ; PORTA (0=output, 1=input)
        MOVLW b'00110011'
        MOVWF TRISB      ; PORTB (0=output, 1=input)
        MOVLW b'11000110' ; Pull-ups Off y configuracion Timer0 con
        MOVWF OPTION_REG ; preescalador 1:128 (aunque timer 0 no se usa)
        MOVLW b'11000000' ; se desenmascara la interrupcción de periféricos,
        MOVWF INTCON      ; y general y se =0 los flags
        MOVLW b'00001001'
        MOVWF PIE1        ; se desenmasacara la interrupción de SSP y Timer1
        MOVLW b'00000000'
        MOVWF PIE2        ; perifericos 2 queda enmascarado
        MOVLW d'30'        ; h'1E', b'00011110'
        MOVWF SSPADD      ; esta será la dirección de mi pata. El bit0 es R/W.
BCF SSPSTAT,SMP      ; en operación I2C se mantiene a 0
BCF SSPSTAT,CKE      ; en operación I2C se mantiene a 0

BCF STATUS,RPO      ; banco mem 0
        MOVLW b'00010001' ; Configuración y encendido del Timer1
        MOVWF T1CON      ; preescalador 1:2, según el clock interno
        MOVLW b'00110110' ; configuración del módulo SSP
        MOVWF SSPCON      ; SSPCON,5 = enable bit
                           ; SSPCON,4 = enable clock
                           ; SSPM<3:0> = 0110 Slave mode, 7-bit
        MOVLW b'00000000'
        MOVWF PIR1        ; De momento se borran todas las banderas
        MOVLW b'00000000'
        MOVWF PIR2        ; De momento se borran todas las banderas

```

```
;----- Inicializaciones -----  
  
CLRF PORTA  
CLRF PORTB  
CLRF SSPBUF  
CLRF cont  
CLRF del  
CLRF cont2  
CLRF del2  
CLRF cont3  
CLRF del3  
CLRF mot  
CLRF mot2  
CLRF mot3  
CLRF mot4  
CLRF action  
CLRF w_temp  
CLRF stat_t  
CLRF pcl_t  
CLRF umax_m1  
CLRF umin_m1  
CLRF usup_m1  
CLRF uinf_m1  
CLRF usup_m2  
CLRF umed_m2  
CLRF uinf_m2  
CLRF upos_m3  
CLRF meta  
CLRF var  
CLRF comp  
CLRF mode  
  
MOVlw d'2'  
MOVWF speed ; velocidad inicial = 2  
MOVlw d'135'  
MOVWF centro ; centro incial = 135  
MOVlw d'55'  
MOVWF rango ; rango incial = 55  
MOVlw d'130'  
MOVWF zanco ; zanco incial = 100  
MOVlw d'210'  
MOVWF altura ; altura incial = 210  
  
MOVlw d'210' ; M1 tope superior  
MOVWF m1_max ; M1 tope inferior  
MOVwf m1_min  
MOVlw d'220' ; M2 tope superior  
MOVWF m2_max ; M2 tope inferior  
MOVlw d'20' ; M2 tope superior  
MOVWF m2_min ; M3 tope superior  
MOVlw d'250' ; M3 tope superior  
MOVWF m3_max ; M3 tope inferior  
MOVlw d'15' ; M3 tope inferior  
MOVWF m3_min ;  
  
BSF mot,6 ; bandera de posicionamiento inicial  
BSF mot2,0 ; comenzamos en la etapa 0  
BSF PORTB,2 ; Led de estado ON  
MOVlw d'250' ; +o- = 1 sec para posicionar la pata  
MOVWF fst ; al principio  
  
MOVF centro,W ; m1 empieza en centro  
MOVWF m1  
MOVF m2_max,W ; m2 empieza en m2_max  
MOVWF m2  
MOVF m3_min,W ; m3 empieza en m3_min  
MOVWF m3
```

```

MOVF  centro,W      ; r_m1 empieza en centro
MOVWF r_m1
MOVF  centro,W      ; r_m2 empieza en centro
MOVWF r_m2
MOVF  centro,W      ; r_m3 empieza en centro
MOVWF r_m2

;----- Inicio del programa principal-----
; Está compuesto por cinco módulos principales:
;   · rutina "bucle": permanece a la espera de que se activen los
;     flags de las rutinas "ciclo" y "movs"
;   · rutina "ciclo": gestiona los motores mandando un puls PWM.
;     Puede llegar a durar más de 6 ms
;   · rutina "movs": gestiona la forma en que fluctúan
;     dinámicamente las variables que goviernan el robot
;   · rutina "fluc": permite que una variable se aproxime
;     lentamente, a una meta mayor o menor.
;   · rutina "m3_find": permite aplicar un factor de corrección al
;     movimiento de la tibia que hace que el pie se mueva en línea
;     recta.
;   "movs" se activa cuando acaba "ciclo" y "ciclo" se activa cada 20ms.
;-----

;----- Bucle Principal -----
bucle BTFSC mot,4      ; identificador de "ciclo"
CALL  ciclo
BTFSC mot,5      ; identificador de "movs"
CALL  movs

GOTO  bucle      ; y vuelve a empezar el bucle.

;----- Rutina de ciclo -----
ciclo  MOVF  m1,W      ; carga m1 en W
MOVWF del          ; carga W en "del"
BSF   PORTA,1      ; RA1 = 1
CALL  delay        ; llama la rutina de retardo
BCF   PORTA,1      ; RA1 = 0

MOVF  m2,W      ; carga m2 en W
MOVWF del          ; carga W en "del"
BSF   PORTA,2      ; RA2 = 1
CALL  delay        ; llama la rutina de retardo
BCF   PORTA,2      ; RA2 = 0

MOVF  m3,W      ; carga m2 en W
MOVWF del          ; carga W en "del"
BSF   PORTA,3      ; RA3 = 1
CALL  delay        ; llama la rutina de retardo
BCF   PORTA,3      ; RA3 = 0

BCF   mot,4      ; bajo bandera de ciclo
BSF   mot,5      ; subo bandera de movs

RETURN

```

## Código fuente

---

```
;----- RUTINA DE MOVIMIENTO -----
;

;----- Limitadores -----
movs  MOVF   speed,W      ; si speed > 10; speed = 0
      SUBLW  d'10'          ; 10-speed en W
      BTFSS  STATUS,C       ; si speed < 10; 10-speed > 0; C=1; otra cosa
      CLRWF  speed          ; si speed > 10; 10-speed < 0; C=0; Solución

      MOVF   rango,W        ; si rango > 125; rango = 125
      SUBLW  d'125'         ; 125-rango en W
      BTFSC  STATUS,C       ; si rango < 125; 125-rango > 0; C=1; otra cosa
      GOTO   com1            ; si rango > 125; 125-rango < 0; C=0; Solución
      MOVLW  d'125'
      MOVWF  rango           ; rango = 125

com1  MOVF   rango,W      ; si en centro+rango ha habido un salto de 255 a 0;
      ADDWF  centro,W       ; centro+rango = m1_max
      BTFSS  STATUS,C       ; si en centro+rango ; C=0; otra cosa
      GOTO   com2            ; si en centro+rango ; C=1; solución

      MOVF   rango,W
      SUBWF  m1_max,W
      MOVWF  centro

; centro = m1_max-rango

com2  MOVF   rango,W      ; si centro+rango > m1_max; centro = m1_max-rango
      ADDWF  centro,W       ; centro+rango en W
      SUBWF  m1_max,W
      BTFSC  STATUS,C       ; m1_max-(centro+rango) en W
; si centro+rango>m1_max; m1_max-(centro+rango) < 0;
; C=0: solución
; si centro+rango<m1_max; m1_max-(centro+rango) > 0;
; C=1: otra cosa

      GOTO   com3

      MOVF   rango,W
      SUBWF  m1_max,W
      MOVWF  centro

; centro = m1_max-rango

com3  MOVF   rango,W      ; si en centro-rango ha habido un salto de 0 a 255;
      SUBWF  centro,W       ; centro-rango=m1_min
      BTFSC  STATUS,C       ; centro-rango en W
; si en centro-rango C=0; solución
; si en centro-rango C=1; otra cosa

      GOTO   com4

      MOVF   rango,W
      ADDWF  m1_min,W
      MOVWF  centro

; centro = m1_min+rango

com4  MOVF   rango,W      ; si centro-rango < m1_min; centro = m1_min+rango
      SUBWF  centro,W       ; centro-rango en W
      SUBWF  m1_min,W
      BTFSS  STATUS,C       ; m1_min-(centro-rango) en W
; si centro-rango>m1_min; m1_min-(centro-rango) < 0;
; C=0: otra cosa
; si centro-rango<m1_min; m1_min-(centro-rango) > 0;
; C=1: solución

      GOTO   com5

      MOVF   rango,W
      ADDWF  m1_min,W
      MOVWF  centro

; centro = m1_min+rango

com5  MOVF   m2_min,W     ; si altura > 250 - m2_min; altura = 250-m2_min
      SUBLW  d'250'          ; 250 - m2_min
      SUBWF  altura,W        ; altura - (250-m2_min)
      BTFSS  STATUS,C       ; si altura<(250-m2_min); altura - (250-m2_min) < 0;
; C=0: otra cosa
; si altura>(250-m2_min); altura - (250-m2_min) > 0;
; C=1: solución

      GOTO   com6

      MOVF   m2_min,W
      SUBLW  d'250'
      MOVWF  altura

; altura = 250-m2_min
```

```

com6   BCF    STATUS,C      ; borro el carry para rotar
       RRF    rango,W       ; si zanco < rango/2 + 5; zanco = rango/2 + 5
       ADDLW  d'5'
       SUBWF zanco,W       ; zanco - (5 + rango/2)
       BTFSC STATUS,C      ; si zanco>(5 + rango/2); zanco - (5 + rango/2) > 0;
                           ; C=1: otra cosa
       GOTO   com7          ; si zanco<(5 + rango/2); zanco - (5 + rango/2) < 0;
                           ; C=0: Solución
       RRF    rango,W       ; zanco = 5 + rango/2
       ADDLW  d'5'
       MOVWF zanco

com7   MOVF   altura,W     ; si 250 - altura + zanco > m2_max; altura =
       SUBLW  d'250'        ; = 250 - m2_max + zanco
       ADDWF  zanco,W       ; 250 - altura
       SUBWF  m2_max,W     ; zanco + (250 - altura)
       BTFSC STATUS,C      ; m2_max - (zanco + (250 - altura))
                           ; si m2_max > (zanco + (250 - altura));
                           ; m2_max - (zanco + (250 - altura)) > 0; C=1; otra
cosa    GOTO   umbral        ; si m2_max < (zanco + (250 - altura));
                           ; m2_max-(zanco + (250 - altura)) < 0: C=0; solución
       MOVF   m2_max,W     ; 250 - m2_max
       SUBLW  d'250'        ; zanco + (250-m2_max)
       ADDWF  zanco,W       ; altura = zanco + (250-m2_max)

;----- Umbrales -----
umbral MOVF   rango,W      ; centro + rango
       ADDWF  centro,W     ; umax_m1 = centro + rango
       MOVWF  umax_m1

       MOVF   rango,W      ; centro - rango
       SUBWF  centro,W     ; umin_m1 = centro - rango
       MOVWF  umin_m1

       BCF    STATUS,C      ; Borro Carry (no puede haber carry!)
       RRF    rango,W       ; roto a la derecha para /2 (con carry se suma 128!)
       ADDWF  centro,W     ; centro + rango/2
       MOVWF  usup_m1       ; usup_m1 = centro + rango/2

       BCF    STATUS,C      ; Borro Carry (no puede haber carry)
       RRF    rango,W       ; roto a la derecha para /2 (con carry se suma 128!)
       SUBWF  centro,W     ; centro - rango/2
       MOVWF  uinf_m1       ; uinf_m1 = centro - rango/2

       MOVF   altura,W     ; 250 - altura
       SUBLW  d'250'        ; uinf_m2 = 250 - altura

       MOVF   zanco,W      ; uinf_m2 + zanco
       ADDWF  uinf_m2,W     ; usup_m2 = uinf_m2 + zanco
       MOVWF  usup_m2

       BCF    STATUS,C      ; Borro Carry (no puede haber carry)
       RRF    rango,W       ; roto a la derecha para /2 (con carry se suma 128!)
       SUBWF  usup_m2,W     ; usup_m2 - rango/2
       MOVWF  umed_m2       ; umed_m2 = usup_m2 - rango/2

```

## Código fuente

---

```
;----- Posicionamiento inicial -----
        BTFSS mot,6          ; flag de primer ciclo
        GOTO  eto            ; si no estoy en primer ciclo voy a etapa 0
        BTFSC mot,7          ; flag de secuencia de levantamiento
        GOTO  ini0

una_vez
        MOVF   centro,W      ; m1 empieza en centro
        MOVWF  m1
        MOVF   m2_max,W     ; m2 empieza en m2_max
        MOVWF  m2
        MOVF   m3_min,W     ; m3 empieza en m3_min
        MOVWF  m3

        DECFSZ fst,F        ; Decrementa fst, cuando sea 0...
        GOTO  salir           ; termina la temporización inicial
        BSF   mot,7           ; y comienza la secuencia de levantamiento
        GOTO  salir

;----- Inicialización 0 -----
ini0  BTFSS mot2,0        ; estoy en etapa 0
        GOTO  ini1

        MOVLW d'25'          ; valor que busca m3 en esta etapa
        MOVWF  meta
        MOVF   m3,W
        MOVWF  var
        MOVLW d'0'            ; compensación de velocidad
        MOVWF  comp
        CALL   fluc
        MOVF   var,W
        MOVWF  m3
        BTFSS mot3,0          ; Comprobación de llegada de m3
        GOTO  salir

        BCF   mot3,0          ; Si m3 ha llegado se borra el flag de llegada
        CLRF  mot2            ; Y se activa el flag de siguiente etapa
        BSF   mot2,1
        GOTO  salir

;----- Inicialización 1 -----
ini1  BTFSS mot2,1        ; estoy en etapa 1
        GOTO  salir

        MOVF   uinf_m2,W     ; valor que busca m2 en esta etapa
        MOVWF  meta
        MOVF   m2,W
        MOVWF  var
        MOVLW d'0'            ; compensación de velocidad
        MOVWF  comp
        CALL   fluc
        MOVF   var,W
        MOVWF  m2

        MOVLW d'170'
        SUBWF m2,W            ; m2 - 170
        BTFSC STATUS,C         ; Si m2 > 170, C=1; si m2 < 170, C=0
        GOTO  salir
        MOVF   m2,W
        SUBLW d'220'           ; 250 - m2 - 30 = INV(uinf_m2) - 30 = 220 - m2
        MOVWF  m3

        BTFSS mot3,0          ; Comprobación de llegada de m3
        GOTO  salir
```

```

BCF    mot3,0      ; Si m3 ha llegado se borra el flag de llegada
CLRF   mot2          ; Y se activa el flag de siguiente etapa
BSF    mot2,7      ; se activa la etapa 7
BCF    mot,7
BCF    mot,6      ; ya no es el periodo inicial de posicionamiento
GOTO   salir

;----- Etapas -----
;

; Este apartado ejecuta una secuencia de movimiento que realiza
; un paso de tipo "ingú", según el esquema siguiente.
; Se llama así por la forma que definen el movimiento de los
; motores 1 y 2. Su principal ventaja es que reduce la inercia
; en los extremos del movimiento horizontal, logrando disminuir
; así la oscilación de la pata, y por tanto su consumo
;

;
;      2-----3
;      /     - \
;      /       \
;      /         \
;      1//           \\\4
;      |             |
;      |             |
;      |             |
;      |             |
;      |             |
;      |             |
;      0 | <-----| -----| -----|| 5
;           7       6

;----- Etapa 0 -----
et0    BTFSS  mot2,0      ; estoy en etapa 0
GOTO   et1

MOVF   umin_m1,W      ; valor que busca m1 en esta etapa
MOVWF  meta
MOVF   m1,W
MOVWF  var
MOVLW  d'0'          ; compensación de velocidad
MOVWF  comp
CALL   fluc
MOVF   var,W
MOVWF  m1
BTFSC  mot3,0
BSF    mot3,1
BCF    mot3,0

MOVF   umed_m2,W      ; valor que busca m2 en esta etapa
MOVWF  meta
MOVF   m2,W
MOVWF  var
MOVLW  d'1'          ; compensación de velocidad
MOVWF  comp
CALL   fluc
MOVF   var,W
MOVWF  m2
BTFSC  mot3,0
BSF    mot3,2
BCF    mot3,0

BCF    mot3,5      ; impide aplicar el factor corrector de m3
CLRF   upos_m3      ; borra la cuenta del factor corrector
CALL   find_m3      ; llamo la rutina de cálculo de m3

```

```
BTFSS mot3,1           ; Comprobación de llegada de motores 1 y 2
GOTO salir
BTFSS mot3,2
GOTO salir

BCF mot3,1           ; Si todos han llegado seborran los flags de llegada
BCF mot3,2

CLRF mot2           ; Y se activa el flag de siguiente etapa
BSF mot2,1
GOTO salir

;----- Etapa 1 -----
et1    BTFSS mot2,1       ; estoy en etapa 1
GOTO et2

MOVF uinf_m1,W      ; valor que busca m1 en esta etapa
MOVWF meta
MOVF m1,W
MOVWF var
MOVLW d'0'          ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m1
BTFSC mot3,0
BSF mot3,1
BCF mot3,0

MOVF usup_m2,W      ; valor que busca m2 en esta etapa
MOVWF meta
MOVF m2,W
MOVWF var
MOVLW d'0'          ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m2
BTFSC mot3,0
BSF mot3,2
BCF mot3,0

BCF mot3,5           ; impide aplicar el factor corrector de m3
CLRF upos_m3
CALL find_m3          ; borra la cuenta del factor corrector
                      ; llamo la rutina de cálculo de m3

BTFSS mot3,1           ; Comprobación de llegada de motores 1 y 2
GOTO salir
BTFSS mot3,2
GOTO salir

BCF mot3,1           ; Si todos han llegado seborran los flags de llegada
BCF mot3,2

CLRF mot2           ; Y se activa el flag de siguiente etapa
BSF mot2,2
GOTO salir
```

## Proyecto MIHRO (Mobile Intelligent Hexapod Robot)

---

```
;----- Etapa 2 -----  
et2    BTFSS mot2,2      ; estoy en etapa 2  
GOTO  et3  
  
      MOVF usup_m1,W   ; valor que busca m1 en esta etapa  
      MOVWF meta  
      MOVF m1,W  
      MOVWF var  
      MOVLW d'1'        ; compensación de velocidad  
      MOVWF comp  
      CALL fluc  
      MOVF var,W  
      MOVWF m1  
      BTFSC mot3,0  
      BSF mot3,1  
      BCF mot3,0  
  
      MOVF usup_m2,W   ; valor que busca m2 en esta etapa  
      MOVWF meta  
      MOVF m2,W  
      MOVWF var  
      MOVLW d'0'        ; compensación de velocidad  
      MOVWF comp  
      CALL fluc  
      MOVF var,W  
      MOVWF m2  
      BTFSC mot3,0  
      BSF mot3,2  
      BCF mot3,0  
  
      BCF mot3,5      ; impide aplicar el factor corrector de m3  
      CLRF upos_m3  
      CALL find_m3  
      ; borra la cuenta del factor corrector  
      ; llamo la rutina de cálculo de m3  
  
      BTFSS mot3,1      ; Comprobación de llegada de motores 1 y 2  
      GOTO salir  
      BTFSS mot3,2  
      GOTO salir  
  
      BCF mot3,1      ; Si todos han llegado seborran los flags de llegada  
      BCF mot3,2  
  
      CLRF mot2      ; Y se activa el flag de siguiente etapa  
      BSF mot2,3  
      GOTO salir  
  
  
;----- Etapa 3 -----  
et3    BTFSS mot2,3      ; estoy en etapa 3  
GOTO  et4  
  
      MOVF umax_m1,W   ; valor que busca m1 en esta etapa  
      MOVWF meta  
      MOVF m1,W  
      MOVWF var  
      MOVLW d'0'        ; compensación de velocidad  
      MOVWF comp  
      CALL fluc  
      MOVF var,W  
      MOVWF m1  
      BTFSC mot3,0  
      BSF mot3,1  
      BCF mot3,0
```

```
MOVF umed_m2,W ; valor que busca m2 en esta etapa
MOVWF meta
MOVF m2,W
MOVWF var
MOVLW d'0' ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m2
BTFS C mot3,0
BSF mot3,2
BCF mot3,0

BCF mot3,5 ; impide aplicar el factor corrector de m3
CLRF upos_m3 ; borra la cuenta del factor corrector
CALL find_m3 ; llamo la rutina de cálculo de m3

BTFS S mot3,1 ; Comprobación de llegada de motores 1 y 2
GOTO salir
BTFS S mot3,2
GOTO salir

BCF mot3,1 ; Si todos han llegado se borran los flags de llegada
BCF mot3,2

CLRF mot2 ; Y se activa el flag de siguiente etapa
BSF mot2,4
GOTO salir

;----- Etapa 4 -----
et4 BTFS S mot2,4 ; estoy en etapa 4
GOTO et5
MOVF umax_m1,W ; valor que busca m1 en esta etapa
MOVWF meta
MOVF m1,W
MOVWF var
MOVLW d'0' ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m1
BTFS C mot3,0
BSF mot3,1
BCF mot3,0

MOVF uinf_m2,W ; valor que busca m2 en esta etapa
MOVWF meta
MOVF m2,W
MOVWF var
MOVLW d'1' ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m2
BTFS C mot3,0
BSF mot3,2
BCF mot3,0

BCF mot3,5 ; impide aplicar el factor corrector de m3
CLRF upos_m3 ; borra la cuenta del factor corrector
CALL find_m3 ; llamo la rutina de cálculo de m3

BTFS S mot3,1 ; Comprobación de llegada de motores 1 y 2
GOTO salir
BTFS S mot3,2
```

## Proyecto MIHRO (Mobile Intelligent Hexapod Robot)

---

```
GOTO  salir

BCF  mot3,1      ; Si todos han llegado se borran los flags de llegada
BCF  mot3,2

CLRF  mot2      ; Y se activa el flag de siguiente etapa
BSF  mot2,5
GOTO  salir

;----- Etapa 5 -----
et5   BTFSS  mot2,5      ; estoy en etapa 5
GOTO  et6

MOVF  usup_m1,W    ; valor que busca m1 en esta etapa
MOVWF meta
MOVF  m1,W
MOVWF var
MOVLW d'0'        ; compensación de velocidad
MOVWF comp
CALL  fluc
MOVF  var,W
MOVWF m1
BTFS  mot3,0
BSF  mot3,1
BCF  mot3,0

MOVF  uinf_m2,W    ; valor que busca m2 en esta etapa
MOVWF meta
MOVF  m2,W
MOVWF var
MOVLW d'0'        ; compensación de velocidad
MOVWF comp
CALL  fluc
MOVF  var,W
MOVWF m2
BTFS  mot3,0
BSF  mot3,2
BCF  mot3,0

BSF  mot3,5      ; permite aplicar el factor corrector de m3
CALL  find_m3

BTFS  mot3,1      ; Comprobación de llegada de motores 1 y 2
GOTO  salir
BTFS  mot3,2
GOTO  salir

BCF  mot3,1      ; Si todos han llegado se borran los flags de llegada
BCF  mot3,2

CLRF  mot2      ; Y se activa el flag de siguiente etapa
BSF  mot2,6
GOTO  salir

;----- Etapa 6 -----
et6   BTFSS  mot2,6      ; estoy en etapa 6
GOTO  et7

MOVF  uinf_m1,W    ; valor que busca m1 en esta etapa
MOVWF meta
MOVF  m1,W
MOVWF var
```

## Código fuente

---

```
MOVLW d'1'          ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m1
BTFSC mot3,0
BSF mot3,1
BCF mot3,0

MOVF uinf_m2,W    ; valor que busca m2 en esta etapa
MOVWF meta
MOVF m2,W
MOVWF var
MOVLW d'0'          ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m2
BTFSC mot3,0
BSF mot3,2
BCF mot3,0

BSF mot3,5          ; permite aplicar el factor corrector de m3
CALL find_m3        ; llamo la rutina de cálculo de m3

BTFSS mot3,1        ; Comprobación de llegada de motores 1 y 2
GOTO salir
BTFSS mot3,2
GOTO salir

BCF mot3,1          ; Si todos han llegado seborran los flags de llegada
BCF mot3,2

CLRF mot2            ; Y se activa el flag de siguiente etapa
BSF mot2,7
GOTO salir

;----- Etapa 7 -----
et7   BTFSS mot2,7      ; estoy en etapa 7
GOTO salir

MOVF umin_m1,W      ; valor que busca m1 en esta etapa
MOVWF meta
MOVF m1,W
MOVWF var
MOVLW d'0'          ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m1
BTFSC mot3,0
BSF mot3,1
BCF mot3,0

MOVF uinf_m2,W      ; valor que busca m2 en esta etapa
MOVWF meta
MOVF m2,W
MOVWF var
MOVLW d'0'          ; compensación de velocidad
MOVWF comp
CALL fluc
MOVF var,W
MOVWF m2
BTFSC mot3,0
BSF mot3,2
```

```

BCF    mot3,0

BSF    mot3,5      ; permite aplicar el factor corrector de m3
CALL   find_m3     ; llamo la rutina de cálculo de m3

BTFS  mot3,1      ; Comprobación de llegada de motores 1 y 2
GOTO  salir
BTFS  mot3,2
GOTO  salir

BCF    mot3,1      ; Si todos han llegado se borran los flags de llegada
BCF    mot3,2

CLRF  mot2         ; Y se activa el flag de siguiente etapa
BSF    mot2,0

;-----
;----- Salida de movs -----
salir BCF    mot,5      ; Borro el flag de rutina movs, para
                        ; acceder sólo una vez cada 20ms
RETURN

;-----
;----- Flujo de variable -----
fluc  MOVF   var,W
SUBWF meta,W          ; meta - var
BTFS  STATUS,Z        ; Z=1 si meta = var, Z=0 si meta <> var
GOTO  diff             ; Si meta <> var, voy a diff
BSF   mot3,0           ; si meta = var, flag de finalizar fluctuación
GOTO  sale              ; y salgo de rutina

diff  MOVF   var,W
SUBWF meta,W          ; meta - var
BTFS  STATUS,C        ; C=1 si meta > var, C=0 si meta < var
GOTO  resta

suma  MOVF   speed,W       ; la variable crece hacia la meta
ADDWF comp,W           ; comp + speed
ADDWF var,F             ; comp + speed + var == var
MOVF   var,W
SUBWF meta,W           ; meta - var
BTFS  STATUS,C         ; C=1 si meta > var, C=0 si meta < var
GOTO  sale
MOVF   meta,W
MOVWF  var
BSF   mot3,0             ; si var ya ha llegado a meta, mot3,0=1
GOTO  sale

resta MOVF   speed,W       ; la variable decrece hacia la meta
ADDWF comp,W           ; comp + speed
SUBWF var,F             ; var - (comp + speed) == var
MOVF   var,W
SUBWF meta,W           ; meta - var
BTFS  STATUS,C         ; C=1 si meta > var, C=0 si meta < var
GOTO  sale
MOVF   meta,W
MOVWF  var
BSF   mot3,0             ; si var ya ha llegado a meta, mot3,0=1

sale   RETURN

```

```

;----- Cálculo de m3 -----
find_m3
    MOVF    m2,W           ; valor que busca m3 en esta etapa
    SUBWF  d'220'          ; 250 - m2 - 30 = INV(uinf_m2) - 30 = 220 - m2
    MOVWF  m3

    BTFSS  mot3,5          ; flag para aplicar el factor corrector de m3
    GOTO   listo
    MOVF   centro,W
    SUBWF  m1,W           ; m1 - centro = m1 - 135
    BTFSS  STATUS,C        ; Si m1 > 135, C=1; si m1 < 135, C=0
    GOTO   menor
    BTFSC  m1,3             ; miro el bit 3 pk se modifica a cada múltiplo de 8
    GOTO   bit1
    BTFSS  mot3,4          ; estado anterior del bit
    GOTO   listo
    BTFSC  mot2,6          ; compruevo si estoy en etapa 6
    GOTO   etapa6
    MOVLW  d'4'             ; factor de incremento
    ADDWF  upos_m3,F
    BCF   mot3,4
    GOTO   listo
etapa6 INCF   upos_m3,F
    BCF   mot3,4
    GOTO   listo
bit1  BTFSS  m1,3          ; miro el bit 3 pk se modifica a cada múltiplo de 8
    GOTO   listo
    BTFSC  mot3,4          ; estado anterior del bit
    GOTO   listo
    BTFSC  mot2,6          ; compruevo si estoy en etapa 6
    GOTO   etap6
    MOVLW  d'4'             ; factor de incremento
    ADDWF  upos_m3,F
    BSF   mot3,4
    GOTO   listo
etap6 INCF   upos_m3,F
    BSF   mot3,4
    GOTO   listo
menor BTFSC  m1,3          ; miro el bit 3 pk se modifica a cada múltiplo de 8
    GOTO   bit_1
    BTFSS  mot3,4          ; estado anterior del bit
    GOTO   listo
    BTFSC  mot2,6          ; compruebo si estoy en etapa 6
    GOTO   eta6
    MOVLW  d'4'             ; factor de decremento
    SUBWF  upos_m3,F
    BCF   mot3,4
    GOTO   listo
eta6  DECF   upos_m3,F
    BCF   mot3,4
    GOTO   listo
bit_1 BTFSS  m1,3          ; miro el bit 3 pk se modifica a cada múltiplo de 8
    GOTO   listo
    BTFSC  mot3,4          ; estado anterior del bit
    GOTO   listo
    BTFSC  mot2,6          ; compruebo si estoy en etapa 6
    GOTO   etp6
    MOVLW  d'4'             ; factor de decremento
    SUBWF  upos_m3,F
    BSF   mot3,4
    GOTO   listo
etp6  DECF   upos_m3,F
    BSF   mot3,4
listo MOVF   upos_m3,W      ; aplico el factor corrector a m3
    SUBWF  m3,F            ; m3 = m3 - upos_m3

RETURN

```

```
;  
----- Rutina de retraso -----  
; La variable "del" contiene el valor del  
; retraso según: <del> * 7,2us + 40 * 12us = retraso  
; La rutina tiene 2 partes (delay y delay2)  
; DELAY ejecuta un retraso constante de 40*12us = 0,48ms  
; y no recibe ningún parámetro de entrada  
; DELAY2 ejecuta un retraso segun la variable "del"  
; 1<"del"<250, de máx 7,2us*250 = 1,8ms  
; así el retraso sera 0,4872ms < retraso < 2,28ms  
; que coincide con los límites del servo  
-----  
  
delay MOVLW d'40'          ; 40 ciclos de 12us=0,48ms (mínimo óptimo del servo)  
      MOVWF del3            ; contador grande  
  
delay3 MOVLW d'10'          ; 60 instrucciones X 200 ns = 12us  
     MOVWF cont3           ; contador pequeño  
     NOP  
     NOP  
     NOP  
  
retar3 NOP  
      NOP  
      DECFSZ cont3,F       ; decrementa cont en 1, salta si cont=0  
      GOTO    retar3  
      NOP  
  
      DECFSZ del3,F        ; decrementa del en 1, salta si del=0  
      GOTO    delay3  
      NOP  
  
-----  
  
delay2 MOVLW d'6'          ; 36 instrucciones X 200 ns = 7,2us  
      MOVWF cont            ; contador pequeño  
      NOP  
  
retar  NOP  
      NOP  
      DECFSZ cont,F         ; decrementa cont en 1, salta si cont=0  
      GOTO    retar  
      NOP  
  
      DECFSZ del,F          ; 250 ciclos de 7,2us = 1,8 ms  
      GOTO    delay2          ; 1,8ms + 0,48ms = 2,28 ms (máximo óptimo del servo)  
      NOP  
  
      RETURN
```

```

; -----
;----- Rutina de Esclavo I2C -----
; Esta rutina identifica 5 posibles estados en la comunicación I2C
; + Escritura del master, envío de la dirección del esclavo
; + Escritura del master, envío del dato
; + Lectura del master, envío de la dirección del esclavo
; + Lectura del master, envío del dato desde el esclavo
; + Reset en la comunicación I2c, el master manda un No-ACK
; Después el master envía una secuencia de variables (Speed, Centro,
; Rango, Zanco, Altura) o demanda unas variables de control (mot2,
; mode, Zanco y Altura)
;-----

i2c_com
    BSF STATUS,RP0 ; banco mem 1 (SSPSTAT está en banco 1)
    BTFSC SSPSTAT,S ; lectura del bit Start
    GOTO s_1 ; S = 1
    BTFSC SSPSTAT,P ; S = 0, lectura del bit Paro
    GOTO res_I2C ; P = 1 = condición de Stop, Bus libre
    GOTO exit ; P = 0 = I2C no funcionando

s_1   BTFSS SSPSTAT,D_A ; comprobación Data/Adress para estado de Reset I2C
    GOTO no_res ; D_A = 0, no puede ser Reset I2C
    BTFSC SSPSTAT,BF ; comprobación BF para estado de Reset I2C
    GOTO no_res ; BF = 1, no puede ser Reset I2C
    BCF STATUS,RP0 ; banco mem 0
    BTFSC SSPCON,CKP ; comprobación CKP para estado de Reset I2C
    GOTO res_I2C ; CKP = 1, se cumple Reset I2C
    ; CKP = 0, no puede ser Reset I2C

no_res BSF STATUS,RP0 ; banco mem 1
    BTFSC SSPSTAT,R_W ; lectura del bit Read/Write
    GOTO read ; R_W = 1 = Read
    ; R_W = 0 = Write
    BTFSC SSPSTAT,D_A ; lectura del bit Data/Adress
    GOTO w_dat ; D_A = 1 = escritura con dato
    GOTO w_dir ; D_A = 0 = escritura con dirección
read   BTFSS SSPSTAT,D_A ; lectura del bit Data/Adress
    GOTO r_dir ; D_A = 0 = lectura con dirección (BF no se mira)
    GOTO r_dat ; D_A = 1 = Data (BF se testea en la rutina)

;-----
;----- Escritura con dirección -----
w_dir  BSF STATUS,RP0 ; banco mem 1
    BTFSC SSPSTAT,BF ; Leo BF para ver si el dato ha entrado
    GOTO in_dir ; si 1, voy al dato
    BCF STATUS,RP0 ; banco mem 0
    MOVF SSPBUF,W ; Leo SSPBUF para borrar BF, descarto el dato
    BTFSC SSPCON,SSPOV ; miro overflow
    BCF SSPCON,SSPOV ; si es 1 lo borro
    GOTO fallo
in_dir BCF STATUS,RP0 ; banco mem 0
    MOVF SSPBUF,W ; Leo SSPBUF para borrar BF, descarto el dato
    BSF mot,1 ; A continuación viene por el I2C la secuencia de
    ; actualización de variables
    BCF mot,2 ; Así que borro las banderas, y activo la primera
    BCF mot,3 ; para empezar por speed
    BCF mot3,6
    BCF mot3,7
    BCF mot4,0
    BCF mot4,1
    BCF mot4,2
    BCF mot4,3
    BCF mot4,4
    GOTO exit ; Hecho esto, estoy listo para recibir la
    ; actualización, y salgo

```

```

;----- Escritura con dato -----
w_dat  BSF    STATUS,RP0      ; banco mem 1
       BTFSC SSPSTAT,BF      ; Leo BF para ver si el dato ha entrado
       GOTO   in_dat        ; si 1, voy al dato
       BCF    STATUS,RP0      ; banco mem 0
       MOVF   SSPBUF,W       ; Leo SSPBUF para borrar BF, descarto el dato
       BTFSC SSPCON,SSPOV    ; miro overflow
       BCF    SSPCON,SSPOV    ; si es 1 lo borro
       GOTO   fallo         ; finalmente, llamo a fallo
in_dat  BCF    STATUS,RP0      ; banco mem 0
       BTFSS SSPCON,SSPOV    ; Compruevo que No ha habido Overflow
       GOTO   speed_c       ; si No ha habido Overflow, recibo el primer dato
       BCF    SSPCON,SSPOV    ; Si ha habido overflow, borro el bit de overflow
       MOVF   SSPBUF,W       ; leo SSPBUF para borrar BF, descarto el dato
       GOTO   fallo         ; finalmente, llamo a fallo

speed_c
       BTFSS mot,1          ; Si la recepción ha es buena y toca recibir Speed
       GOTO   cent_c        ; Si no toca Speed, probemos con Centro
       MOVF   SSPBUF,W       ; Si toca Speed, lo cargo del Búffer a la variable
       MOVWF  speed          ; apago mot,1 y enciendo mot,2 para que
       BCF    mot,1          ; el siguiente byte recibido vaya a Centro
       BSF    mot,2          ;
       GOTO   acaba         ;

cent_c
       BTFSS mot,2          ; Toca recibir Centro
       GOTO   rango_c        ; Si no toca Centro, probemos con Rango
       MOVF   SSPBUF,W       ; Si toca Centro, lo cargo del Búffer a la variable
       MOVWF  centro          ; apago mot,2 y enciendo mot,3 para que
       BCF    mot,2          ; el siguiente byte recibido vaya a Rango
       BSF    mot,3          ;
       GOTO   acaba         ;

rango_c
       BTFSS mot,3          ; Toca recibir Rango
       GOTO   zanco_c        ; Si no toca Rango, probemos con Zanco
       MOVF   SSPBUF,W       ; Si toca Rango, lo cargo del Búffer a la variable
       MOVWF  rango          ; apago mot,3 y enciendo mot3,6 para que
       BCF    mot,3          ; el siguiente byte recibido vaya a Zanco
       BSF    mot3,6          ;
       GOTO   acaba         ;

zanco_c
       BTFSS mot3,6         ; Toca recibir Zanco
       GOTO   altur_c        ; Si no toca Zanco, probemos con Altura
       MOVF   SSPBUF,W       ; Si toca Zanco, lo cargo del Búffer a la variable
       MOVWF  zanco          ; apago mot3,6 y enciendo mot3,7 para que
       BCF    mot3,6          ; el siguiente byte recibido vaya a Altura
       BSF    mot3,7          ;
       GOTO   acaba         ;

altur_c
       BTFSS mot3,7         ; Toca recibir Altura
       GOTO   mode_c         ; Si no toca Altura, vamos a mode
       MOVF   SSPBUF,W       ; Si toca Altura, lo cargo del Búffer a la variable
       MOVWF  altura          ; apago mot3,7 y enciendo mot4,0 para que
       BCF    mot3,7          ; el siguiente byte recibido vaya a mode
       BSF    mot4,0          ;
       GOTO   acaba         ;

mode_c
       BTFSS mot4,0         ; Toca recibir mode
       GOTO   actio_c        ; Si no toca mode, vamos a action
       MOVF   SSPBUF,W       ; Si toca mode, lo cargo del Búffer a la variable
       MOVWF  mode          ; apago mot4,0 y enciendo mot4,1 para que
       BCF    mot4,0          ; el siguiente byte recibido vaya a action
       BSF    mot4,1          ;
       GOTO   acaba         ;

```

```

actio_c
    BTFSS mot4,1      ; Toca recibir action
    GOTO m1_c         ; Si no toca action, vamos a r_m1
    MOVF SSPBUF,W     ; Si toca action, lo cargo del Búffer a la variable
    MOVWF action
    BCF mot4,1        ; apago mot4,1 y enciendo mot4,2 para que
    BSF mot4,2        ; el siguiente byte recibido vaya a r_m1
    GOTO acaba

m1_c  BTFSS mot4,2      ; Toca recibir r_m1
    GOTO m2_c         ; Si no toca r_m1, vamos a r_m2
    MOVF SSPBUF,W     ; Si toca r_m1, lo cargo del Búffer a la variable
    MOVWF r_m1
    BCF mot4,2        ; apago mot4,2 y enciendo mot4,3 para que
    BSF mot4,3        ; el siguiente byte recibido vaya a r_m2
    GOTO acaba

m2_c  BTFSS mot4,3      ; Toca recibir r_m2
    GOTO m3_c         ; Si no toca r_m2, vamos a r_m3
    MOVF SSPBUF,W     ; Si toca r_m2, lo cargo del Búffer a la variable
    MOVWF r_m2
    BCF mot4,3        ; apago mot4,3 y enciendo mot4,4 para que
    BSF mot4,4        ; el siguiente byte recibido vaya a r_m3
    GOTO acaba

m3_c  BTFSS mot4,4      ; Toca recibir r_m3
    GOTO exit          ; Si no toca r_m3, salimos
    MOVF SSPBUF,W     ; Si toca r_m3, lo cargo del Búffer a la variable
    MOVWF r_m3
    BCF mot4,4        ; apago mot4,4 y enciendo mot1 para que
    BSF mot1           ; el siguiente byte recibido vaya a Speed

acaba BSF STATUS,RP0   ; banco mem 1
       BTFSC SSPSTAT,BF  ; Leo BF para ver si se ha borrado
       GOTO fallo        ; si sigue siendo 1, error
       GOTO exit          ; si se ha borrado, exit

;----- Lectura con dirección -----
r_dir BCF STATUS,RP0   ; banco mem 0
       BTFSC SSPCON,CKP  ; ¿CKP = 0?
       GOTO fallo        ; CKP = 1, error
       MOVF SSPBUF,W     ; Leo Buffer para borrar BF
       BSF mot1           ; BF = 0, preparo las banderas para envío secuencial
de datos
       BCF mot2
       BCF mot3
       BCF mot3,6
       BCF mot3,7
       BCF mot4,0
       BCF mot4,1
       BCF mot4,2
       BCF mot4,3
       BCF mot4,4
       GOTO mot2R         ; mando mot2 (está en R_dat, pero no importa)

```

## Proyecto MIHRO (Mobile Intelligent Hexapod Robot)

---

```
;----- Lectura con dato -----  
  
r_dat BCF STATUS,RP0 ; banco mem 0  
       BTFSC SSPCON,CKP ; ¿CKP = 0?  
       GOTO fallo ; CKP = 1  
       BSF STATUS,RP0 ; banco mem 1  
       BTFSC SSPSTAT,BF ; CKP = 0, compruebo ¿BF = 0?  
       GOTO r_dat ; BF = 1, me espero a que se vacie  
       BCF STATUS,RP0 ; BF = 0, continua la secuencia de envio (Speed ya  
                      ; está enviado), banco mem 0  
  
mot2R BTFSS mot,1 ; Toca enviar mot2  
       GOTO modeR ; Si no toca mot2, probemos con mode  
       BCF SSPCON,WCOL ; borro el flag de colisión  
       MOVF mot2,W ; Si toca mot2, lo cargo de la variable al Búffer  
       MOVWF SSPBUF  
       BTFSC SSPCON,WCOL ; Hay colisión de datos?  
       GOTO mot2R ; Si la hay vuelvo a intentarlo  
       BSF SSPCON,CKP ; Si no la hay, enciendo CKP para mandar el dato  
       BCF mot,1 ; apago mot,1 y enciendo mot,2 para que  
       BSF mot,2 ; el siguiente byte enviado sea Centro  
       GOTO exit  
  
modeR BTFSS mot,2 ; Toca enviar mode  
       GOTO zancoR ; Si no toca mode, probemos con zanco  
       BCF SSPCON,WCOL ; borro el flag de colisión  
       MOVF mode,W ; Si toca mode, lo cargo de la variable al Búffer  
       MOVWF SSPBUF  
       BTFSC SSPCON,WCOL ; Hay colisión de datos?  
       GOTO modeR ; Si la hay vuelvo a intentarlo  
       BSF SSPCON,CKP ; Si no la hay, enciendo CKP para mandar el dato  
       BCF mot,2 ; apago mot,2 y enciendo mot,3 para que  
       BSF mot,3 ; el siguiente byte enviado sea Rango  
       GOTO exit  
  
zancoR BTFSS mot,3 ; Toca enviar Zanco  
       GOTO alturaR ; Si no toca Zanco, probemos con Altura  
       BCF SSPCON,WCOL ; borro el flag de colisión  
       MOVF zanco,W ; Si toca Zanco, lo cargo de la variable al Búffer  
       MOVWF SSPBUF  
       BTFSC SSPCON,WCOL ; Hay colisión de datos?  
       GOTO zancoR ; Si la hay vuelvo a intentarlo  
       BSF SSPCON,CKP ; Si no la hay, enciendo CKP para mandar el dato  
       BCF mot,3 ; apago mot,3 y enciendo mot3,6 para que  
       BSF mot3,6 ; el siguiente byte enviado sea Zanco  
       GOTO exit  
  
alturaR  
       BTFSS mot3,6 ; Toca enviar altura  
       GOTO exit ; Si no toca altura, salimos  
       BCF SSPCON,WCOL ; borro el flag de colisión  
       MOVF altura,W ; Si toca Altura, lo cargo de la variable al Búffer  
       MOVWF SSPBUF  
       BTFSC SSPCON,WCOL ; Hay colisión de datos?  
       GOTO alturaR ; Si la hay vuelvo a intentarlo  
       BSF SSPCON,CKP ; Si no la hay, enciendo CKP para mandar el dato  
       BCF mot3,6 ; apago mot3,6 y enciendo mot,1 para que  
       BSF mot,1 ; el siguiente byte enviado sea mot2  
       GOTO exit
```

```
;----- Fin de la comunicación -----  
  
res_I2C    BCF    STATUS,RP0 ; banco mem 0  
            MOVF   SSPBUF,W ; Leo SSPBUF para borrar BF, descarto el dato  
            BSF    STATUS,RP0 ; banco mem 1  
            BTFSC SSPSTAT,BF ; Leo BF para ver si se ha borrado  
            GOTO   fallo ; si sigue siendo 1 error  
            BCF    STATUS,RP0 ; banco mem 0  
            BTFSC SSPCON,SSPOV ; miro overflow  
            BCF    SSPCON,SSPOV ; si es 1 lo borro  
            BSF    mot,1 ; Reinicio de la secuencia de envío/recepción  
            BCF    mot,2  
            BCF    mot,3  
            BCF    mot3,6  
            BCF    mot3,7  
            BCF    mot4,0  
            BCF    mot4,1  
            BCF    mot4,2  
            BCF    mot4,3  
            BCF    mot4,4  
            GOTO   exit  
  
;  
-----  
fallo    BCF    STATUS,RP0 ; banco mem 0  
            BCF    PORTB,2 ; Led de estado OFF  
            RETURN  
  
exit     BCF    STATUS,RP0 ; banco mem 0  
            BSF    PORTB,2 ; Led de estado ON  
            RETURN  
  
;  
----- Rutina de Servicio de Interrupción-----  
; (para salvar el reg. status se usa swap pk así no se altera el  
; contenido (status contiene los flag aritméticos). Al volver de  
; la rutina se restablece el orden de los nibbles.)  
;  
; Hay tres fuentes de interrupción:  
; · flanko de rubida de RB0/int (sirve para el control de  
;     adaptación al terreno, actualmente sin implementar)  
; · Módulo SSP  
; · ovrerflow del Timer 1  
;  
  
isr      MOVWF w_temp ; salva registro W en reg. temporal  
            SWAPF  STATUS,W ; salva el reg. status (swap no altera el registro)  
            MOVWF stat_t ; dándole la vuelta  
            MOVF   PCLATH,W  
            MOVWF pcl_t  
  
            BCF    INTCON,GIE ; apago interrupciones  
            BCF    STATUS,RP0 ; banco mem 0  
  
            BTFSC INTCON,RBIF ; pooling de RB7:RB4  
            GOTO   i_rb ; (Actualmente desactivado para preservar  
                        ; el funcionamiento de I2C_com)  
            BTFSC INTCON,INTF ; pooling de RB0/INT  
            GOTO   i_rb0 ; Actualmente no implementado  
  
            BTFSC PIR1,SSPIF ; pooling de evento en el módulo SSP  
            GOTO   i2c_ev  
  
            BTFSC PIR1,TMR1IF ; pooling de overflow Timer1  
            GOTO   i_tmr1  
            GOTO   escape
```

## Proyecto MIHRO (Mobile Intelligent Hexapod Robot)

```
i_rb    MOVF   PORTB,F
        BCF    INTCON,RBIF ; limpiamos el flag
        BSF    INTCON,RBIE ; desenmascaramos la INT
        GOTO   escape

i_rb0   BCF    INTCON,INTF ; limpiamos el flag
        BSF    INTCON,INTE ; desenmascaramos la INT
        GOTO   escape

i_tmrl1 BCF    PIR1,TMR1IF ; limpiamos el flag
        BSF    STATUS,RP0  ; banco mem 1 (PIE1 está en banco 1)
        BSF    PIE1,TMR1IE ; desenmascaramos la INT
        BCF    STATUS,RP0  ; banco mem 0

        MOVLW h'3C'          ; cargo 3C en TMR1H y
        MOVWF  TMR1H          ;
        MOVLW h'B0'          ; cargo B0 en TMR1L para que
        MOVWF  TMR1L          ; juntos hagan 3CB0, que hasta FFFF incrementa
                               ; C350 veces, que son 50000 veces en decimal,
                               ; de manera que el timer se demora
                               ; 200ns x 2 x 50000 = 20ms (2 es el preescaler)
        BSF    mot,4           ; subo bandera de ciclo
        GOTO   escape          ; asi se ejecuta cada 20ms

i2c_ev  BCF    PIR1,SSPIF ; limpiamos el flag
        BSF    STATUS,RP0  ; banco mem 1 (PIE1 está en banco 1)
        BSF    PIE1,SSPIE ; desenmascaramos la INT
        BCF    STATUS,RP0  ; banco mem 0

        CALL   i2c_com        ; acción en caso de evento en el módulo SSP

escape  MOVF   pcl_t,W
        MOVWF  PCLATH
        SWAPF  stat_t,W      ; le da la vuelta al stat_t
        MOVWF  STATUS          ; y lo reestablece en status
        SWAPF  w_temp,F       ; reestablece W desde W_temp
        SWAPF  w_temp,W       ; dandole la vuelta 2 veces

        RETFIE                ; salgo de rutina

;-----
END
;-----
```

(Total: 1.475 líneas)

## 2 Código fuente de la placa Master

Este es un código fuente en fase de pruebas. Su versión definitiva se adjunta en el CD.

```

;----- Encabezado -----
;
; Archivo:      master_teclado.asm
; Autor:        Ignacio Pedrosa Lojo
; Versión:      1.00
;
; Descripción: Programa de control y comunicación de la placa
;               maestra. Incluye el código para la lectura del
;               teclado, el control de los LEDs indicadores, la
;               rutina de retraso, el control del movimiento del
;               robot, la gestión de la comunicación I2C en modo
;               Master y la gestión del Sistema Interruptivo.
;
; Ensamblado mediante Microchip Assembler v5.00 (MPASM)
; El archivo "P16F88.inc" proporcionado por Microchip debe ser
; incluido en el directorio de trabajo en caso de no usar MPASM v5.00
;
;-----
;

LIST      P=16F88
INCLUDE    "P16F88.INC"

; Registro 1 de Configuración
__CONFIG   _CONFIG1, _CP_OFF
            & _CCP1_RB3           ; CCP1 en la patilla RB3
            & _DEBUG_OFF          ; Modo debug OFF
            & _WRT_PROTECT_OFF   ; Protección contra escritura OFF
            & _CPD_OFF            ; Code protection OFF
            & _LVP_OFF             ; Low voltage Programming OFF
            & _BODEN_ON           ; Brown-out reset enabled
            & _MCLR_ON             ; Memmory Clear enabled
            & _PWRTE_ON            ; Power-up timer enabled
            & _WDT_OFF              ; WatchDog Timer OFF
            & _HS_OSC                ; HS Oscillator mode

; Registro 2 de Configuración
__CONFIG   _CONFIG2, _IESO_OFF & _FCMEN_OFF

;----- Variables en memoria -----
cont      EQU      h'20'
del       EQU      h'21'
cont2     EQU      h'22'
del2      EQU      h'23'
cont3     EQU      h'24'
del3      EQU      h'25'
m1        EQU      h'26'
m2        EQU      h'27'
m3        EQU      h'28'
fail      EQU      h'29'
w_temp    EQU      h'2A'
stat_t    EQU      h'2B'
pcl_t    EQU      h'2C'
mot       EQU      h'2D'
mot2      EQU      h'2E'
mot3      EQU      h'2F'
acu       EQU      h'30'
time      EQU      h'31'
dato     EQU      h'32'
send_d    EQU      h'33'
read_d    EQU      h'34'
slave     EQU      h'35'
speed     EQU      h'36'
centro    EQU      h'37'
rango     EQU      h'38'

```

```
zanco      EQU      h'39'
altura     EQU      h'3A'
slv1       EQU      h'3B'
slv2       EQU      h'3C'
slv3       EQU      h'3D'
slv4       EQU      h'3E'
slv5       EQU      h'3F'
slv6       EQU      h'40'
slv_gen    EQU      h'41'
action     EQU      h'42'
mode       EQU      h'43'
r_mot2    EQU      h'44'
r_mode    EQU      h'45'
r_zanco   EQU      h'46'
r_altu    EQU      h'47'

; hasta 7Fh

; la variable 'action' contiene 8 banderas:
; action,0 : 1= Go                      0= Stop
; action,1 : 1= Atrás                   0= Delante
; action,2 : 1= Giro Izquierda          0= No Giro
; action,3 : 1= Giro Derecha            0= No Giro
; action,4 : 1= Rotación Izq.          0= No Rotación
; action,5 : 1= Rotación Dch.          0= No Rotación
; action,6 : 1= Ida del paso           0= No Ida
; action,7 : 1= Vuelta del paso         0= No vuelta

; la variable 'mode' contiene 8 banderas:
; mode,0 : 1= Modo Dinámico          0= Modo Estático
; mode,1 : 1= Modo levantamiento    0= No Modo
; mode,2 : 1= Modo Cojer             0= No Modo
; mode,3 : 1= Modo Inicial           0= No Modo
; mode,4 : 1= Modo Programación      0= No Modo
; mode,5 : 1= Obstáculo detec.      0= No Obstáculo
; mode,6 : 1=                         0=
; mode,7 : 1=                         0=

; la variable 'mot' contiene 8 banderas:
; mot,0 = botón 0
; mot,1 = botón 1
; mot,2 = botón 2
; mot,3 = botón 3
; mot,4 = botón 4
; mot,5 = botón 5
; mot,6 = botón 6
; mot,7 = botón 7

; la variable 'mot2' contiene 8 banderas:
; mot2,0 =
; mot2,1 =
; mot2,2 = primer dígito es 2
; mot2,3 =
; mot2,4 =
; mot2,5 = segundo dígito es 5
; mot2,6 =
; mot2,7 =

; la variable 'mot3' contiene 8 banderas:
; mot3,0 = botón 8
; mot3,1 = primer dígito
; mot3,2 = segundo dígito
; mot3,3 = tercer dígito
; mot3,4 = cuarto dígito
; mot3,5 = recuerdo de ACK=1, NACK=0;
; mot3,6 = botón 9
; mot3,7 = botón A
```

```
;----- Vectores de inicio e interrupción -----  
  
ORG    0x00  
GOTO  inicio  
ORG    0x04  
GOTO  isr  
ORG    0X05  
  
;----- Configuración de puertos y registros -----  
  
inicio BSF    STATUS,RP0      ; banco mem 1  
       MOVLW b'00000000'  
       MOVWF ANSEL          ; ADC desactivado  
       MOVLW b'10111111'  
       MOVWF TRISA           ; PORTA (0=output, 1=input)  
       MOVLW b'11111111'  
       MOVWF TRISB           ; PORTB (0=output, 1=input)  
       MOVLW b'11000000'        ; configuracion: Timer0 con  
       MOVWF OPTION_REG        ; preescalador 1/2 (no se usa)  
       MOVLW b'01010000'        ; se desenmascara la interrupcción de periféricos y  
       MOVWF INTCON            ; externa, pero por ahora, desactivo interrupciones  
       MOVLW b'00001000'  
       MOVWF PIE1              ; Sólo se desenmasacara la interrupción de SSP  
       MOVLW b'00000000'  
       MOVWF PIE2              ; perifericos 2 queda enmascarado  
       MOVLW d'70'              ; h'46', b'01000110'  
       MOVWF SSPADD             ; esta será la dirección del master, (para  
                               ; Multi-Master Mode)  
       BCF    SSPSTAT,SMP        ; en operación I2C se mantiene a 0  
       BCF    SSPSTAT,CKE        ; en operación I2C se mantiene a 0  
       BCF    STATUS,RP0          ; banco mem 0  
       MOVLW b'00111011'        ; configuración del módulo SSP  
       MOVWF SSPCON              ; SSPCON,5 = 1, enable bit  
                               ; SSPCON,4 = 1, enable clock  
                               ; SSPM<3:0> = 1011 Firmware Master Mode (Slave Idle)  
       MOVLW b'00000000'  
       MOVWF PIR1              ; De momento se borran todas las banderas  
       MOVLW b'00000000'  
       MOVWF PIR2              ; De momento se borran todas las banderas
```

```
;----- Inicializaciones -----  
  
CLRF PORTA  
CLRF PORTB  
CLRF SSPBUF  
CLRF cont  
CLRF del  
CLRF cont2  
CLRF del2  
CLRF cont3  
CLRF del3  
CLRF mot  
CLRF mot2  
CLRF mot3  
CLRF w_temp  
CLRF stat_t  
CLRF pcl_t  
CLRF acu  
CLRF time  
CLRF fail  
CLRF dato  
CLRF send_d  
CLRF read_d  
CLRF action  
CLRF mode  
CLRF r_mot2  
CLRF r_mode  
CLRF r_zanco  
CLRF r_altu  
  
MOVLW d'10'          ; dirección esclavo 1  
MOVWF slv1  
MOVLW d'20'          ; dirección esclavo 2  
MOVWF slv2  
MOVLW d'30'          ; dirección esclavo 3  
MOVWF slv3  
MOVLW d'40'          ; dirección esclavo 4  
MOVWF slv4  
MOVLW d'50'          ; dirección esclavo 5  
MOVWF slv5  
MOVLW d'60'          ; dirección esclavo 6  
MOVWF slv6  
CLRF slv_gen          ; dirección esclavo introducido por el usuario  
  
MOVLW d'4'  
MOVWF speed           ; velocidad inicial = 4  
MOVLW d'135'  
MOVWF centro          ; centro incial = 135  
MOVLW d'15'  
MOVWF rango           ; rango incial = 15  
MOVLW d'130'  
MOVWF zanco           ; zanco incial = 100  
MOVLW d'210'  
MOVWF altura          ; altura incial = 210  
  
MOVLW d'135'         ; m1, m2 y m3 comienzan en un valor céntrico  
MOVWF m1  
MOVLW d'135'  
MOVWF m2  
MOVLW d'135'  
MOVWF m3  
  
BSF mot3,1           ; comenzamos con primer dígito  
MOVLW d'250'  
MOVWF time            ; dejo cargado time
```

```
;----- Bucle principal-----  
  
bucle    BTFSC  PORTA,0           ; bot0  
          BSF     mot,0  
          BTFSC  PORTA,1           ; bot1  
          BSF     mot,1  
          BTFSC  PORTA,2           ; bot2  
          BSF     mot,2  
          BTFSC  PORTA,3           ; bot3  
          BSF     mot,3  
          BTFSC  PORTA,4           ; bot4  
          BSF     mot,4  
          BTFSC  PORTB,5           ; bot5  
          BSF     mot,5  
          BTFSC  PORTB,6           ; bot6  
          BSF     mot,6  
          BTFSC  PORTB,7           ; bot7  
          BSF     mot,7  
          BTFSC  PORTB,2           ; bot8  
          BSF     mot3,0  
          BTFSC  PORTB,3           ; bot9  
          BSF     mot3,6  
          BTFSC  PORTB,0           ; botA  
          BSF     mot3,7  
  
          BTFSC  mot,0            ; Llamada bot0  
          CALL   bot0  
          BTFSC  mot,1            ; Llamada bot1  
          CALL   bot1  
          BTFSC  mot,2            ; Llamada bot2  
          CALL   bot2  
          BTFSC  mot,3            ; Llamada bot3  
          CALL   bot3  
          BTFSC  mot,4            ; Llamada bot4  
          CALL   bot4  
          BTFSC  mot,5            ; Llamada bot5  
          CALL   bot5  
          BTFSC  mot,6            ; Llamada bot6  
          CALL   bot6  
          BTFSC  mot,7            ; Llamada bot7  
          CALL   bot7  
          BTFSC  mot3,0           ; Llamada bot8  
          CALL   bot8  
          BTFSC  mot3,6           ; Llamada bot9  
          CALL   bot9  
          BTFSC  mot3,7           ; Llamada botA  
          CALL   botA  
  
          GOTO   bucle           ; y vuelve a empezar el bucle.
```

## Código fuente

---

```
;----- Rutinas de acción de los botones -----  
  
;----- Rutina botón 0 -----  
  
bot0  MOVLW  d'250'      ; Llamaré a un retraso de 2,28ms  
      MOVWF  del          ; para evitar rebotes  
      CALL   delay  
      BTFSC  PORTA,0      ; Salta si RA0=0  
      GOTO   bot0  
  
b0_d1  BTFSS  mot3,1  
      GOTO   b0_d2  
      MOVLW  d'0'          ; Acción para apretar b0 en el primer dígito  
      ADDWF  acu,F  
      BCF    mot3,1  
      BSF    mot3,2  
      CALL   led1  
      GOTO   exb0  
b0_d2  BTFSS  mot3,2  
      GOTO   b0_d3  
      MOVLW  d'0'          ; Acción para apretar b0 en el segundo dígito  
      ADDWF  acu,F  
      BCF    mot3,2  
      BSF    mot3,3  
      CALL   led2  
      GOTO   exb0  
b0_d3  BTFSS  mot3,3  
      GOTO   b0_d4  
      MOVLW  d'0'          ; Acción para apretar b0 en el tercer dígito  
      ADDWF  acu,F  
      BCF    mot3,3  
      BSF    mot3,4  
      CALL   led3  
      GOTO   exb0  
b0_d4  BTFSS  mot3,4  
      GOTO   exb0          ; Acción para apretar b0 en el cuarto dígito  
      CLRWF acu  
      BCF    mot2,2          ; Borro acu  
      BCF    mot2,5          ; Borro las banderas de primer dígito es 2  
      BCF    mot3,4          ; y de segundo dígito es 5  
      BSF    mot3,1          ; borro flag díxito 4 y activo dígitol1  
                          ; para volver a empezar  
  
exb0   BCF    mot,0  
      RETURN  
  
;----- Rutina botón 1 -----  
  
bot1  MOVLW  d'250'      ; Llamaré a un retraso de 2,28ms  
      MOVWF  del          ; para evitar rebotes  
      CALL   delay  
      BTFSC  PORTA,1      ; Salta si RA1=0  
      GOTO   bot1  
  
b1_d1  BTFSS  mot3,1  
      GOTO   b1_d2  
      MOVLW  d'100'         ; Acción para apretar b1 en el primer dígito  
      ADDWF  acu,F  
      BCF    mot3,1  
      BSF    mot3,2  
      CALL   led1  
      GOTO   exb1  
b1_d2  BTFSS  mot3,2  
      GOTO   b1_d3  
      MOVLW  d'10'           ; Acción para apretar b1 en el segundo dígito  
      ADDWF  acu,F  
      BCF    mot3,2
```

```

        BSF    mot3,3
        CALL   led2
        GOTO   exb1
b1_d3  BTFSS mot3,3
        GOTO   b1_d4
        MOVLW d'1'           ; Acción para apretar b1 en el tercer dígito
        ADDWF acu,F
        BCF   mot3,3
        BSF   mot3,4
        CALL  led3
        GOTO  exb1
b1_d4  BTFSS mot3,4
        GOTO  exb1           ; Acción para apretar b1 en el cuarto dígito
        MOVF  acu,W           ; apretar "1" en el dígito 4
        MOVWF speed            ; carga speed con el valor introducido
        MOVLW d'1'
        MOVWF acu             ; cargo 1 en acu
        CALL  led5             ; mando un 1 a led5 para señalizar la carga correcta
        CLRF  acu
        BCF   mot2,2           ; Borro las banderas de primer dígito es 2
        BCF   mot2,5           ; y de segundo dígito es 5
        BCF   mot3,4           ; borro flag dígitos 4 y activo dígito 1
        BSF   mot3,1           ; para volver a empezar

exb1   BCF   mot,1
        RETURN

;----- rutina botón 2 -----
bot2   MOVLW d'250'          ; Idem k bot1
        MOVWF del
        CALL  delay
        BTFSC PORTA,2         ; Salta si RA2=0
        GOTO  bot2

b2_d1  BTFSS mot3,1
        GOTO  b2_d2
        MOVLW d'200'           ; Acción para apretar b2 en el primer dígito
        ADDWF acu,F
        BSF   mot2,2           ; Bandera de primer dígito es dos
        BCF   mot3,1
        BSF   mot3,2
        CALL  led1
        GOTO  exb2
b2_d2  BTFSS mot3,2
        GOTO  b2_d3
        MOVLW d'20'             ; Acción para apretar b2 en el segundo dígito
        ADDWF acu,F
        BCF   mot3,2
        BSF   mot3,3
        CALL  led2
        GOTO  exb2
b2_d3  BTFSS mot3,3
        GOTO  b2_d4
        MOVLW d'2'              ; Acción para apretar b2 en el tercer dígito
        ADDWF acu,F
        BCF   mot3,3
        BSF   mot3,4
        CALL  led3
        GOTO  exb2
b2_d4  BTFSS mot3,4
        GOTO  exb2             ; Acción para apretar b1 en el cuarto dígito
        MOVF  acu,W             ; apretar "2" en el dígito 4
        MOVWF centro            ; carga centro con el valor introducido
        MOVLW d'1'
        MOVWF acu             ; cargo 1 en acu
        CALL  led5             ; mando un 1 a led5 para señalizar la carga correcta
        CLRF  acu

```

```

BCF    mot2,2      ; Borro las banderas de primer dígito es 2
BCF    mot2,5      ; y de segundo dígito es 5
BCF    mot3,4      ; borro flag dígitos 4 y activo dígito 1
BSF    mot3,1      ; para volver a empezar

exb2  BCF    mot,2
      RETURN

;----- Rutina boton 3 -----

bot3  MOVLW d'250'      ; Idem k bot1
      MOVWF del
      CALL delay
      BTFSC PORTA,3      ; Salta si RA3=0
      GOTO bot3

b3_d1 BTFSS mot3,1
      GOTO b3_d2
      CALL led4          ; Mensaje de error
      GOTO exb3          ; Apretar b3 en el primer dígito no tiene sentido

b3_d2 BTFSS mot3,2
      GOTO b3_d3
      MOVLW d'30'
      ADDWF acu,F
      BCF    mot3,2
      BSF    mot3,3
      CALL led2
      GOTO exb3

b3_d3 BTFSS mot3,3
      GOTO b3_d4
      MOVLW d'3'
      ADDWF acu,F
      BCF    mot3,3
      BSF    mot3,4
      CALL led3
      GOTO exb3

b3_d4 BTFSS mot3,4
      GOTO exb3          ; Acción para apretar b3 en el cuarto dígito
      MOVF  acu,W
      MOVWF rango
      MOVLW d'1'
      MOVWF acu
      CALL led5
      CLRF  acu
      BCF    mot2,2
      BCF    mot2,5
      BCF    mot3,4
      BSF    mot3,1      ; Borro las banderas de primer dígito es 2
                           ; y de segundo dígito es 5
                           ; borro flag dígitos 4 y activo dígito 1
                           ; para volver a empezar

exb3  BCF    mot,3
      RETURN

;----- Rutina boton 4 -----

bot4  MOVLW d'250'      ; Idem k bot1
      MOVWF del
      CALL delay
      BTFSC PORTA,4      ; Salta si RA4=0
      GOTO bot4

b4_d1 BTFSS mot3,1
      GOTO b4_d2
      CALL led4          ; Mensaje de error
      GOTO exb4          ; Apretar b4 en el primer dígito no tiene sentido

b4_d2 BTFSS mot3,2
      GOTO b4_d3
      MOVLW d'40'
      ADDWF acu,F        ; Acción para apretar b4 en el segundo dígito

```

```

BCF    mot3,2
BSF    mot3,3
CALL   led2
GOTO   exb4
b4_d3 BTFSS mot3,3
GOTO   b4_d4
MOVLW d'4'          ; Acción para apretar b4 en el tercer dígito
ADDWF acu,F
BCF    mot3,3
BSF    mot3,4
CALL   led3
GOTO   exb4
b4_d4 BTFSS mot3,4
GOTO   exb4          ; Acción para apretar b1 en el cuarto dígito
MOVF   acu,W          ; apretar "4" en el dígito 4
MOVWF  zanco          ; carga zanco con el valor introducido
MOVLW d'1'
MOVWF  acu            ; cargo 1 en acu
CALL   led5            ; mando un 1 a led5 para señalizar la carga correcta
CLRF   acu
BCF    mot2,2          ; Borro las banderas de primer dígito es 2
BCF    mot2,5          ; y de segundo dígito es 5
BCF    mot3,4          ; borro flag dígito 4 y activo dígito 1
BSF    mot3,1          ; para volver a empezar

exb4 BCF    mot4,4
RETURN

;----- Rutina boton 5 -----
bot5 MOVLW d'250'      ; Idem k bot1
MOVWF del
CALL  delay
BTFSC PORTB,5         ; Salta si RB5=0
GOTO  bot5

b5_d1 BTFSS mot3,1
GOTO  b5_d2
CALL  led4            ; Mensaje de error
GOTO  exb5            ; Apretar b5 en el primer dígito no tiene sentido
b5_d2 BTFSS mot3,2
GOTO  b5_d3
MOVLW d'50'
ADDWF acu,F
BTFSC mot2,2          ; Si el primer dígito es 2 (200 + ...)
BSF   mot2,5          ; Bandera de segundo dígito es cinco
BCF   mot3,2
BSF   mot3,3
CALL  led2
GOTO  exb5
b5_d3 BTFSS mot3,3
GOTO  b5_d4
MOVLW d'5'
ADDWF acu,F
BCF   mot3,3
BSF   mot3,4
CALL  led3
GOTO  exb5
b5_d4 BTFSS mot3,4
GOTO  exb5            ; Acción para apretar b1 en el cuarto dígito
MOVF   acu,W          ; apretar "5" en el dígito 4
MOVWF  altura          ; carga altura con el valor introducido
MOVLW d'1'
MOVWF  acu            ; cargo 1 en acu
CALL  led5            ; mando un 1 a led5 para señalizar la carga correcta
CLRF   acu
BCF   mot2,2          ; Borro las banderas de primer dígito es 2
BCF   mot2,5          ; y de segundo dígito es 5

```

## Código fuente

---

```
        BCF    mot3,4      ; borro flag díxito 4 y activo dígito 1
        BSF    mot3,1      ; para volver a empezar

exb5  BCF    mot,5
      RETURN

;----- Rutina boton 6 -----

bot6  MOVLW  d'250'      ; Idem k bot1
      MOVWF  del
      CALL   delay
      BTFSC  PORTB,6     ; Salta si RB6=0
      GOTO   bot6

b6_d1 BTFSS  mot3,1
      GOTO   b6_d2
      CALL   led4      ; Mensaje de error
      GOTO   exb6      ; Apretar b6 en el primer dígito no tiene sentido

b6_d2 BTFSS  mot3,2
      GOTO   b6_d3
      BTFSC  mot2,2     ; Continua si el primer dígito no es 2
      GOTO   over6
      MOVLW  d'60'
      ADDWF  acu,F
      BCF    mot3,2
      BSF    mot3,3
      CALL   led2
      GOTO   exb6

over6 CALL   led4      ; Mensaje de error
      GOTO   exb6

b6_d3 BTFSS  mot3,3
      GOTO   b6_d4
      BTFSC  mot2,5     ; Continua si el segundo dígito no es 5
      GOTO   ovr6
      MOVLW  d'6'
      ADDWF  acu,F
      BCF    mot3,3
      BSF    mot3,4
      CALL   led3
      GOTO   exb6

ovr6  CALL   led4      ; Mensaje de error
      GOTO   exb6

b6_d4 BTFSS  mot3,4
      GOTO   exb6
      MOVF   acu,W
      MOVWF  m1
      MOVLW  d'1'
      MOVWF  acu
      CALL   led5      ; cargo 1 en acu
      CLRF   acu
      BCF    mot2,2     ; mando un 1 a led5 para señalizar la carga correcta
      BCF    mot2,5
      BCF    mot3,4
      BSF    mot3,1     ; Borro las banderas de primer dígito es 2
                        ; y de segundo dígito es 5
                        ; borro flag dígito 4 y activo dígito 1
                        ; para volver a empezar

exb6  BCF    mot,6
      RETURN

;----- Rutina boton 7 -----

bot7  MOVLW  d'250'      ; Idem k bot1
      MOVWF  del
      CALL   delay
      BTFSC  PORTB,7     ; Salta si RB7=0
      GOTO   bot7

b7_d1 BTFSS  mot3,1
      GOTO   b7_d2
```

```

        CALL  led4           ; Mensaje de error
        GOTO  exb7
b7_d2  BTFSS mot3,2      ; Apretar b7 en el primer dígito no tiene sentido
        GOTO  b7_d3
        BTFSC mot2,2      ; Continua si el primer dígito no es 2
        GOTO  over7
        MOVLW d'70'        ; Acción para apretar b7 en el segundo dígito
        ADDWF acu,F
        BCF   mot3,2
        BSF   mot3,3
        CALL  led2
        GOTO  exb7
over7  CALL  led4           ; Mensaje de error
        GOTO  exb7
b7_d3  BTFSS mot3,3      ; Continua si el segundo dígito no es 5
        GOTO  b7_d4
        BTFSC mot2,5      ; Continua si el segundo dígito no es 5
        GOTO  ovr7
        MOVLW d'7'          ; Acción para apretar b7 en el tercer dígito
        ADDWF acu,F
        BCF   mot3,3
        BSF   mot3,4
        CALL  led3
        GOTO  exb7
ovr7   CALL  led4           ; Mensaje de error
        GOTO  exb7
b7_d4  BTFSS mot3,4      ; Acción para apretar b7 en el cuarto dígito
        GOTO  exb7
        MOVF  acu,W
        MOVWF m2
        MOVLW d'1'
        MOVWF acu
        CALL  led5
        CLRWF acu
        BCF   mot2,2
        BCF   mot2,5
        BCF   mot3,4
        BSF   mot3,1      ; cargo 1 en acu
                                ; mando un 1 a led5 para señalizar la carga correcta
                                ; Borro las banderas de primer dígito es 2
                                ; y de segundo dígito es 5
                                ; borro flag dígito 4 y activo dígito 1
                                ; para volver a empezar

exb7   BCF   mot,7
        RETURN

;----- Rutina boton 8 -----
bot8   MOVLW d'250'       ; Idem k bot1
        MOVWF del
        CALL  delay
        BTFSC PORTB,2     ; Salta si RB2=0
        GOTO  bot8

b8_d1  BTFSS mot3,1      ; Mensaje de error
        GOTO  b8_d2
        CALL  led4
        GOTO  exb8
b8_d2  BTFSS mot3,2      ; Apretar b8 en el primer dígito no tiene sentido
        GOTO  b8_d3
        BTFSC mot2,2      ; Continua si el primer dígito no es 2
        GOTO  over8
        MOVLW d'80'        ; Acción para apretar b8 en el segundo dígito
        ADDWF acu,F
        BCF   mot3,2
        BSF   mot3,3
        CALL  led2
        GOTO  exb8
over8  CALL  led4           ; Mensaje de error
        GOTO  exb8
b8_d3  BTFSS mot3,3
        GOTO  b8_d4

```

## Código fuente

---

```
BTFSC mot2,5           ; Continua si el segundo dígito no es 5
GOTO ovr8
MOVLW d'8'
ADDWF acu,F
BCF mot3,3
BSF mot3,4
CALL led3
GOTO exb8
ovr8 CALL led4          ; Mensaje de error
GOTO exb8
b8_d4 BTFSS mot3,4
GOTO exb8
MOVWF acu,W
MOVWF m3
MOVLW d'1'
MOVWF acu
CALL led5
CLRF acu
BCF mot2,2
BCF mot2,5
BCF mot3,4
BSF mot3,1
; Borro las banderas de primer dígito es 2
; y de segundo dígito es 5
; borro flag dígito 4 y activo dígito 1
; para volver a empezar

exb8 BCF mot3,0
RETURN

;----- Rutina boton 9 -----
bot9 MOVLW d'250'        ; Idem k bot1
MOVWF del
CALL delay
BTFSC PORTB,3          ; Salta si RB3=0
GOTO bot9

b9_d1 BTFSS mot3,1
GOTO b9_d2
CALL led4
GOTO exb9
; Mensaje de error
; Apretar b9 en el primer dígito no tiene sentido
b9_d2 BTFSS mot3,2
GOTO b9_d3
BTFSC mot2,2
GOTO over9
MOVLW d'90'
ADDWF acu,F
BCF mot3,2
BSF mot3,3
CALL led2
GOTO exb9
; Continua si el primer dígito no es 2
; Acción para apretar b9 en el segundo dígito

over9 CALL led4          ; Mensaje de error
GOTO exb9
b9_d3 BTFSS mot3,3
GOTO b9_d4
BTFSC mot2,5
GOTO ovr9
MOVLW d'9'
ADDWF acu,F
BCF mot3,3
BSF mot3,4
CALL led3
GOTO exb9
; Continua si el segundo dígito no es 5
; Acción para apretar b9 en el tercer dígito

ovr9 CALL led4          ; Mensaje de error
GOTO exb9
b9_d4 BTFSS mot3,4
GOTO exb9
MOVWF acu,W
MOVWF slv_gen
MOVLW d'1'
MOVWF acu
; cargo 1 en acu
```

## Proyecto MIHRO (Mobile Intelligent Hexapod Robot)

```
CALL led5 ; mando un 1 a led5 para señalizar la carga correcta
CLRF acu
BCF mot2,2 ; Borro las banderas de primer dígito es 2
BCF mot2,5 ; y de segundo dígito es 5
BCF mot3,4 ; borro flag dígito 4 y activo dígito 1
BSF mot3,1 ; para volver a empezar

exb9 BCF mot3,6
RETURN

;----- Rutina boton A -----
botA MOVLW d'250' ; Idem k bot1
MOVWF del
CALL delay
BTFSB PORTB,0 ; Salta si RB0=0
GOTO botA

bA_d1 BTFSB mot3,1
GOTO bA_d2
CALL i2c_wr ; Apretar A en el dígito 1 efectúa una escritura I2C
GOTO exbA ; No se borran banderas, sigue activo dígito 1
bA_d2 BTFSB mot3,2
GOTO bA_d3
CALL led4 ; Mensaje de error
GOTO exbA ; Apretar bA en el segundo dígito no tiene sentido
bA_d3 BTFSB mot3,3
GOTO bA_d4
CALL led4 ; Mensaje de error
GOTO exbA ; Apretar bA en el tercero dígito no tiene sentido
bA_d4 BTFSB mot3,4
GOTO exbA ; Acción para apretar bA en el cuarto dígito
MOVWF acu,W ; apretar "A" en el dígito 4
MOVWF mode ; carga mode con el valor introducido
MOVLW d'1'
MOVWF acu ; cargo 1 en acu
CALL led5 ; mando un 1 a led5 para señalizar la carga correcta
CLRF acu
BCF mot2,2 ; Borro las banderas de primer dígito es 2
BCF mot2,5 ; y de segundo dígito es 5
BCF mot3,4 ; borro flag dígito 4 y activo dígito 1
BSF mot3,1 ; para volver a empezar

exbA BCF mot3,7
RETURN

;
;----- Rutina LED 1 -----
led1 MOVLW d'250' ; Enciende el primer LED
MOVWF del
CALL delay
BTFSB PORTA,0 ; Salta si RA0=0
GOTO led1

BSF STATUS,RP0 ; banco mem 1
MOVLW b'10111110'
MOVWF TRISA ; RA0 outputs
BCF STATUS,RP0 ; banco mem 0

ld1 BSF PORTA,0
MOVLW d'250' ; Llamaré a un retraso de 2,28ms
MOVWF del ; para evitar rebotes
CALL delay

DECFSZ time,F
GOTO ld1
BCF PORTA,0
```

```
BSF    STATUS,RP0      ; banco mem 1
MOVLW b'10111111'
MOVWF TRISA            ; RA0 input
BCF    STATUS,RP0      ; banco mem 0

MOVLW d'250'          ; 250 veces 2,28ms son 0,57s
MOVWF time
RETURN

;----- Rutina LED 2 -----

led2  MOVLW d'250'      ; Enciende el segundo LED
MOVWF del
CALL  delay
BTFS  PORTA,4         ; Salta si RA4=0
GOTO led2

BSF    STATUS,RP0      ; banco mem 1
MOVLW b'10101111'
MOVWF TRISA            ; RA4 output
BCF    STATUS,RP0      ; banco mem 0

ld2   BSF    PORTA,4
MOVLW d'250'          ; Llamaré a un retraso de 2,28ms
MOVWF del              ; para evitar rebotes
CALL  delay

DECFSZ time,F
GOTO  ld2
BCF    PORTA,4

BSF    STATUS,RP0      ; banco mem 1
MOVLW b'10111111'
MOVWF TRISA            ; RA4 input
BCF    STATUS,RP0      ; banco mem 0

MOVLW d'250'          ; 250 veces 2,28ms son 0,57s
MOVWF time
RETURN

;----- Rutina LED 3 -----

led3  MOVLW d'250'      ; Enciende el tercer LED
MOVWF del
CALL  delay
BTFS  PORTB,7         ; Salta si RB7=0
GOTO led3

BSF    STATUS,RP0      ; banco mem 1
MOVLW b'01111111'
MOVWF TRISB            ; RB7 output
BCF    STATUS,RP0      ; banco mem 0

ld3   BSF    PORTB,7
MOVLW d'250'          ; Llamaré a un retraso de 2,28ms
MOVWF del              ; para evitar rebotes
CALL  delay
DECFSZ time,F
GOTO  ld3
BCF    PORTB,7
BSF    STATUS,RP0      ; banco mem 1
MOVLW b'11111111'
MOVWF TRISB            ; RB7 input
BCF    STATUS,RP0      ; banco mem 0

MOVLW d'250'          ; 250 veces 2,28ms son 0,57s
MOVWF time
RETURN
```

## Proyecto MIHRO (Mobile Intelligent Hexapod Robot)

---

```
;----- Rutina LED 4 -----
led4  MOVLW d'250'          ; Enciende y apaga el LED rojo
      MOVWF del
      CALL delay
      BTFSC PORTB,2        ; Salta si RB2=0
      GOTO led4

      BSF STATUS,RP0        ; banco mem 1
      MOVLW b'01111011'
      MOVWF TRISB           ; RB2 output
      BCF STATUS,RP0         ; banco mem 0

      BSF PORTB,2
ld4   MOVLW d'250'          ; Llamaré a un retraso de 2,28ms
      MOVWF del             ; para evitar rebotes
      CALL delay

      DECFSZ time,F
      GOTO ld4
      MOVLW d'250'          ; 250 veces 2,28ms son 0,57s
      MOVWF time

ld44  BCF PORTB,2          ; Llamaré a un retraso de 2,28ms
      MOVLW d'250'          ; para evitar rebotes
      MOVWF del
      CALL delay

      DECFSZ time,F
      GOTO ld44
      MOVLW d'250'          ; 250 veces 2,28ms son 0,57s
      MOVWF time

      BSF STATUS,RP0        ; banco mem 1
      MOVLW b'11111111'
      MOVWF TRISB           ; RB2 input
      BCF STATUS,RP0         ; banco mem 0

      RETURN

;----- Rutina LED 5 -----
led5  MOVLW d'250'          ; Enciende el LED tantas veces como "acu"
      MOVWF del
      CALL delay
      BTFSC PORTB,5        ; Salta si RB2=0
      GOTO led5

      BSF STATUS,RP0        ; banco mem 1
      MOVLW b'01011111'
      MOVWF TRISB           ; RB5 output
      BCF STATUS,RP0         ; banco mem 0

ld53  BSF PORTB,5          ; Enciendo LED
      MOVLW d'250'          ; 250 veces 2,28ms son 0,57s
      MOVWF time

ld5   MOVLW d'250'          ; Llamaré a un retraso de 2,28ms
      MOVWF del
      CALL delay
      DECFSZ time,F
      GOTO ld5

      BCF PORTB,5          ; Apago LED
      MOVLW d'250'          ; 250 veces 2,28ms son 0,57s
      MOVWF time

ld52  MOVLW d'250'          ; Llamaré a un retraso de 2,28ms
```

```

MOVWF del
CALL delay
DECFSZ time,F
GOTO ld52

DECFSZ acu,F
GOTO ld53

BSF STATUS,RP0 ; banco mem 1
MOVLW b'11111111'
MOVWF TRISB ; RB5 input
BCF STATUS,RP0 ; banco mem 0

MOVLW d'250' ; 250 veces 2,28ms son 0,57s
MOVWF time
RETURN

----- Rutina de retraso -----
; La variable "del" contiene el valor del
; retraso según: <del> * 7,2us + 40 * 12us = retraso
; La rutina tiene 2 partes (delay y delay2)
; DELAY ejecuta un retraso constante de 40*12us = 0,48ms
; y no recibe ningún parámetro de entrada
; DELAY2 ejecuta un retraso segun la variable "del"
; 1<"del"<250, de máx 7,2us*250 = 1,8ms
; así el retraso sera 0,4872ms < retraso < 2,28ms
; que coincide con los límites del servo
-----

delay MOVLW d'40' ; 40 ciclos de 12us = 0,48 ms
; (mínimo óptimo del servo)
MOVWF del3 ; contador grande

delay3 MOVLW d'10' ; 60 instrucciones X 200 ns = 12us
MOVWF cont3 ; contador pequeño
NOP
NOP
NOP

retar3 NOP
NOP
DECFSZ cont3,F ; decrementa cont en 1, salta si cont=0
GOTO retar3
NOP

DECFSZ del3,F ; decrementa del en 1, salta si del=0
GOTO delay3
NOP

-----
delay2 MOVLW d'6' ; 36 instrucciones X 200 ns = 7,2us
MOVWF cont ; contador pequeño
NOP

retar NOP
NOP
DECFSZ cont,F
GOTO retar
NOP

DECFSZ del,F ; 250 ciclos de 7,2us = 1,8 ms
GOTO delay2 ; 1,8 ms + 0,48ms = 2,28 ms
; (máximo óptimo del servo)
NOP

RETURN

```

```

;----- Rutina de comunicación Master I2C -----
; Esta rutina planifica las acciones que se pretenden realizar y
; llama a cada procedimiento secuencialmente:
;   + Envía Start
;   + Envía Stop
;   + Envía ACK
;   + Envía NACK
;   + Envía Dato
;   + Envía Dirección para escritura
;   + Envía Dirección para lectura
;   + Esperar y recibir ACK / NACK
;   + Esperar y recibir Dato
; SDA y SCL se controlan mediante los bits de dirección TRIS, ya que
; el nivel del bus se libera en colector abierto (con pull-up), de
; manera que la salida física del pin siempre es 0 (PORTB)
;-----

;----- Secuencia de escritura de direcciones -----

i2c_wr MOVF slv_gen,W
    BTFSC STATUS,Z      ; compruevo si slv_gen es 0
    GOTO resend          ; si lo es, envío los parámetros a todas las patas
    CALL s_send           ; Mando señal Start
    MOVF slv_gen,W       ; Cargo la dirección del esclavo escojido en "slave"
    MOVWF slave
    CALL write_s          ; Mando demanda de escritura
    BTFSS mot3,5          ; compruevo si ha habido ACK
    RETURN                ; No ha habido ACK por 10 veces, error de envío
    CALL vari              ; envio el resto de variables

    RETURN

resend CALL s_send          ; Mando señal Start
    MOVF slv1,W           ; Cargo la dirección del esclavo 1 en "slave"
    MOVWF slave
    CALL write_s          ; Mando demanda de escritura
    BTFSS mot3,5          ; compruevo si ha habido ACK
    RETURN                ; No ha habido ACK por 10 veces, error de envío
    CALL vari              ; envio de variables

    MOVF slv2,W           ; Cargo la dirección del esclavo 2 en "slave"
    MOVWF slave
    CALL write_s          ; Mando demanda de escritura
    BTFSS mot3,5          ; compruevo si ha habido ACK
    RETURN                ; No ha habido ACK por 10 veces, error de envío
    CALL vari              ; envio de variables

    MOVF slv3,W           ; Cargo la dirección del esclavo 3 en "slave"
    MOVWF slave
    CALL write_s          ; Mando demanda de escritura
    BTFSS mot3,5          ; compruevo si ha habido ACK
    RETURN                ; No ha habido ACK por 10 veces, error de envío
    CALL vari              ; envio de variables

    MOVF slv4,W           ; Cargo la dirección del esclavo 4 en "slave"
    MOVWF slave
    CALL write_s          ; Mando demanda de escritura
    BTFSS mot3,5          ; compruevo si ha habido ACK
    RETURN                ; No ha habido ACK por 10 veces, error de envío
    CALL vari              ; envio de variables

    MOVF slv5,W           ; Cargo la dirección del esclavo 5 en "slave"
    MOVWF slave
    CALL write_s          ; Mando demanda de escritura
    BTFSS mot3,5          ; compruevo si ha habido ACK

```

```

RETURN                                ; No ha habido ACK por 10 veces, error de envío
CALL vari                            ; envío de variables

MOVF slv6,W                          ; Cargo la dirección del esclavo 6 en "slave"
MOVWF slave                           ; Mando demanda de escritura
CALL write_s                         ; compruevo si ha habido ACK
BTFS S mot3,5                        ; No ha habido ACK por 10 veces, error de envío
RETURN                                ; envio de variables

CALL vari                            ; No ha habido ACK por 10 veces, error de envío
;----- Secuencia de escritura de los parámetros -----
vari MOVF speed,W                   ; Envío de "speed"
MOVWF send_d                         ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

MOVF centro,W                        ; Envío de "centro"
MOVWF send_d                         ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

MOVF rango,W                         ; Envío de "rango"
MOVWF send_d                         ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

MOVF zanco,W                         ; Envío de "zanco"
MOVWF send_d                         ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

MOVF altura,W                        ; Envío de "altura"
MOVWF send_d                         ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

MOVF mode,W                           ; Envío de "mode"
MOVWF send_d                          ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

MOVF action,W                         ; Envío de "action"
MOVWF send_d                          ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

MOVF m1,W                            ; Envío de "m1"
MOVWF send_d                          ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

MOVF m2,W                            ; Envío de "m2"
MOVWF send_d                          ; Cargo el dato que deseo mandar en "send_d"
CALL data_s                           ; Mando el dato, si NACK repite el envío
BTFS S mot3,5                        ; compruevo si ha habido ACK
RETURN                                ; No ha habido ACK por 10 veces, error de envío

```

```

MOVF  m3,W           ; Envío de "m3"
MOVWF send_d          ; Cargo el dato que deseo mandar en "send_d"
CALL  data_s          ; Mando el dato, si NACK repite el envío
BTFS  mot3,5          ; compruevo si ha habido ACK
RETURN                ; No ha habido ACK por 10 veces, error de envío

CALL  p_send           ; Mando señal de Stop

RETURN

;----- Secuencia de lectura de parámetros -----
i2c_rd
    CALL  s_send         ; Mando señal Start

    MOVF  slv_gen,W      ; Cargo la dirección del esclavo escojido en "slave"
    MOVWF slave           ; Mando demanda de escritura
    CALL  read_s          ; compruevo si ha habido ACK
    BTFS  mot3,5          ; No ha habido ACK por 10 veces, error de envío

    CALL  data_r          ; recibo el dato en read_d
    MOVF  read_d,W        ; el dato recibido es mot2
    MOVWF r_mot2          ; traslado el dato a una variable buffer
    CALL  ack_s           ; ACK para que mande el siguiente dato

    CALL  data_r          ; recibo el dato en read_d
    MOVF  read_d,W        ; el dato recibido es mode
    MOVWF r_mode           ; traslado el dato a una variable buffer
    CALL  ack_s           ; ACK para que mande el siguiente dato

    CALL  data_r          ; recibo el dato en read_d
    MOVF  read_d,W        ; el dato recibido es zanco
    MOVWF r_zanco          ; traslado el dato a una variable buffer
    CALL  ack_s           ; ACK para que mande el siguiente dato

    CALL  data_r          ; recibo el dato en read_d
    MOVF  read_d,W        ; el dato recibido es altura
    MOVWF r_altu           ; traslado el dato a una variable buffer

    CALL  p_send           ; Mando señal de Stop

RETURN

```

## Código fuente

---

```
;----- Mandar secuencia de Start -----  
  
s_send          ; Enviar Start  
    BSF STATUS,RP0      ; banco mem 1 (TRISB está en banco 1)  
    BSF TRISB,1         ; SDA en alto  
    BSF TRISB,4         ; SCL en alto  
    BCF STATUS,RP0      ; banco mem 0  
    BTFSS PORTB,4       ; SCL está bloqueado por esclavo (=0)?  
    GOTO s_send          ; Sí, lo intento hasta que el esclavo lo libere  
    NOP                 ; No, retraso total de 1,6us  
    NOP  
    NOP  
    NOP  
    BSF STATUS,RP0      ; banco mem 1  
    BCF TRISB,1         ; SDA flanco de bajada mientras SCL en alto  
    NOP                 ; retraso de 1,6 us  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    BSF STATUS,RP0      ; banco mem 1  
    BCF TRISB,4         ; SCL en bajo para encadenar con el siguiente pulso  
    BCF STATUS,RP0      ; banco mem 0  
                           ; el código cumple el retraso total de 1,6 us  
    RETURN  
  
;----- Mandar secuencia de Paro -----  
  
p_send          ; Enviar Stop  
    BSF STATUS,RP0      ; banco mem 1 (TRISB está en banco 1)  
    BCF TRISB,1         ; SDA en bajo  
    BSF TRISB,4         ; SCL en alto  
    BCF STATUS,RP0      ; banco mem 0  
    BTFSS PORTB,4       ; SCL está bloqueado por esclavo (=0)?  
    GOTO p_send          ; Sí, lo intento hasta que el esclavo lo libere  
    NOP                 ; No, retraso total de 1,6us  
    NOP  
    NOP  
    NOP  
    BSF STATUS,RP0      ; banco mem 1  
    BSF TRISB,1         ; SDA flanco de subida mientras SCL en alto  
    NOP                 ; retraso de 1,6 us  
    NOP  
    NOP  
    NOP  
    NOP  
    NOP  
    BCF STATUS,RP0      ; banco mem 0  
    RETURN
```

```

----- Mandar dato -----
; Entradas: dato a mandar en "send_d"
-----

data_s MOVF    send_d,W      ; Enviar dato (MSB primero)
        MOVWF   dato          ; el dato se modifica, pero la info está
                                ; a salvo en send_d
        MOVLW   d'8'           ; reinicio contador a 8
        MOVWF   cont2

new_bit
        BTFSC  dato,7         ; ¿el bit es 0?
        GOTO   bit_1          ; el bit es 1
        BSF    STATUS,RPO     ; banco mem 1
        BCF    TRISB,1         ; el bit es 0, SDA = 0
        GOTO   hold

bit_1  BSF    STATUS,RPO     ; banco mem 1
        BSF    TRISB,1         ; el bit es 1, SDA = 1
hold   BCF    TRISB,4         ; me aseguro que SCL en bajo al empezar
        NOP                 ; retraso total de 1,6 us
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP

snd    BSF    STATUS,RPO     ; banco mem 1
        BSF    TRISB,4         ; SCL en alto manda el bit
        BCF    STATUS,RPO     ; banco mem 0
        BTFSS  PORTB,4         ; SCL está bloqueado por esclavo (=0)?
        GOTO   snd             ; Sí, lo intento hasta que el esclavo lo libere
        NOP                 ; No, retraso total de 1,6 us
        NOP
        NOP
        NOP
        NOP
        BSF    STATUS,RPO     ; banco mem 1
        BCF    TRISB,4         ; SCL en bajo para encadenar con el siguiente pulso
        BCF    STATUS,RPO     ; banco mem 0
        RLF    dato,F           ; roto dato a la izquierda
                                ; retraso software mayor de 1'6 us
        DECFSZ cont2,F         ; inicialmente 8
        GOTO   new_bit          ; hasta el bit 0, repito el proceso
                                ; retraso software mayor de 1'6 us
        CALL   ack_r             ; llamo a la comprobación de ACK
        BTFSC  mot3,5            ; compruebo ACK

        RETURN                  ; Si ACK, salgo

        MOVF   send_d,W         ; si NACK, cargo de nuevo la variable y
                                ; repito desde el principio
        MOVWF  dato
        INCF   fail,F
        MOVF   fail,W
        SUBLW  d'10'             ; 10 - fail
        BTFSS  STATUS,Z         ; si fail=10, 10-fail = 0
        GOTO   data_s            ; si fail < 10, vuelvo a intentar
        CLRF   fail              ; si fail = 10, borro fail y
        GOTO   fallo             ; error

```

## Código fuente

---

```
;----- recibir dato -----
; Salidas: dato recibido en "read_d"
;-----

data_r MOVLW d'8'          ; Recibir dato (MSB primero)
        MOVWF cont2         ; reinicio contador a 8

n_bit  NOP                  ; retraso de 1,6 us mientras slave prepara el dato
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        BSF   STATUS,RP0    ; banco mem 1
        BSF   TRISB,4       ; SCL en alto para recibir el bit
        BCF   STATUS,RP0    ; banco mem 0
        BTFSS PORTB,4      ; SCL está bloqueado por esclavo (=0)?
        GOTO  n_bit         ; Sí, lo intento hasta que el esclavo lo libere
        BTFSS PORTB,1      ; No, testeo SDA
        GOTO  b_zero        ; SDA=0
        BSF   read_d,0      ; SDA=1, enciendo el bit 0
        GOTO  decr

b_zero BCF   read_d,0      ; borro el bit 0
decr  DECFSZ cont2,F     ; decremento el contador
        GOTO  corr_0
        GOTO  fin_r

corr_0 RLF   read_d,F     ; y lo corro a la izquierda
                           ; cuando haya corrido 8 veces, el primero
                           ; recibido estará en el bit 7
        NOP                  ; no hace falta más retraso para que se cumpla
                           ; el Ton de SCL
        BSF   STATUS,RP0    ; banco mem 1
        BCF   TRISB,4       ; SCL en bajo para encadenar con el siguiente pulso
        NOP                  ; retraso preventivo de 1us
        NOP
        NOP
        NOP
        GOTO  n_bit

fin_r  NOP                  ; no hace falta más retraso para que
                           ; se cumpla el Ton de SCL
        BSF   STATUS,RP0    ; banco mem 1
        BCF   TRISB,4       ; SCL en bajo para encadenar con el siguiente pulso
        CALL  ack_s

RETURN

;----- Enviar Dirección para escritura -----
; Entradas: Dirección del esclavo en "slave"
;-----


write_s      ; Enviar Dirección para escritura
        BCF   slave,0      ; bit0 de slave para escritura = 0
        MOVF  slave,W
        MOVWF send_d       ; preparo el byte a mandar
        CALL  data_s        ; llamo a enviar el dato

RETURN
```

```

;----- Enviar Dirección para lectura -----
; Entradas: Dirección del esclavo en "slave"
;-----

read_s          ; Enviar Dirección para lectura
    BSF    slave,0      ; bit0 de slave para lectura = 1
    MOVF   slave,W
    MOVWF  send_d       ; preparo el byte a mandar
    CALL   data_s       ; llamo a enviar el dato

    RETURN

;----- leer ACK / mandar NACK -----

ack_r           ; comprueba ACK / envía NACK
    BSF    STATUS,RP0    ; banco mem 1
    BSF    TRISB,1       ; SDA=1 para permitir modificación desde slave
    NOP
    NOP
    NOP
    NOP

puls_9 BSF    STATUS,RP0    ; banco mem 1
        BSF    TRISB,4      ; SCL en alto
        BCF    STATUS,RP0    ; banco mem 0
        BTFSS  PORTB,4      ; SCL está bloqueado por esclavo (=0)?
        GOTO   puls_9       ; Sí, lo intento hasta que el esclavo lo libere
        BTFSS  PORTB,1      ; No, compruevo SDA
        GOTO   hold2         ; SDA=0, ACK
        BCF    mot3,5       ; SDA=1, NACK
        GOTO   hold3
hold2  BSF    mot3,5
        NOP
        ; retraso software de 1'6 us

hold3  BSF    STATUS,RP0    ; banco mem 1
        BCF    TRISB,4       ; SCL en bajo para encadenar con el siguiente pulso
        BCF    STATUS,RP0    ; banco mem 0
        ; retraso software mayor de 1'6 us
    RETURN

;----- Mandar ACK -----

ack_s           ; Enviar ACK
    BSF    STATUS,RP0    ; banco mem 1
    BCF    TRISB,1       ; SDA=0 significa ACK
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP

puls9  BSF    STATUS,RP0    ; banco mem 1
        BSF    TRISB,4      ; SCL en alto
        BCF    STATUS,RP0    ; banco mem 0
        BTFSS  PORTB,4      ; SCL está bloqueado por esclavo (=0)?
        GOTO   puls9       ; Sí, lo intento hasta que el esclavo lo libere
        NOP
        ; retraso total de 1,6 us
        NOP
        NOP
        NOP
        BSF    STATUS,RP0    ; banco mem 1
        BCF    TRISB,4       ; SCL en bajo para encadenar con el siguiente pulso
        BCF    STATUS,RP0    ; banco mem 0

    RETURN

```

## Código fuente

---

```
;----- Mandar NACK -----
nack_s          ; Enviar NACK
    CALL ack_r      ; recibir un ACK pone SDA en nivel alto,
                      ; por lo que es lo mismo que mandar NACK
    RETURN

;----- En caso de Fallo -----
fallo CALL led4      ; mensaje de error
    RETURN

;----- Rutina de Servicio de Interrupción-----
; Para salvar y reestablecer el reg. status se usa swap pk así
; no se altera su contenido, ya que status contiene los flags
; aritméticos. Al volver de la rutina se reestablece el orden
; de los nibbles.
; Este programa no usa ninguna fuente de interrupción, aunque
; queda preparado para asumir en un futuro próximo las
; las fuentes de interrupción:
;   · flanco de rubida de RB0/int
;   · Evento en el módulo SSP (Multi-Master Mode)
;   · overflow del Timer 0
;-----

isr  MOVWF w_temp      ; salva registro W en reg. temporal
     SWAPF STATUS,W    ; salva el reg. status
     MOVWF stat_t       ; dandole la vuelta
     MOVF PCLATH,W
     MOVWF pcl_t

     BCF  INTCON,GIE    ; apago interrupciones
     BCF  STATUS,RP0     ; banco mem 0

;----- Pooling -----
BTFS  INTCON,INTF    ; pooling de RB0/INT
GOTO i_rb0
BTFS  PIR1,SSPIF     ; pooling de evento en el módulo SSP
GOTO i2c_ev
BTFS  PIR1,TMR1IF    ; pooling de overflow Timer1
GOTO i_tmrl
GOTO escape

;----- Interrupción externa -----
i_rb0 BCF  INTCON,INTF    ; limpiamos el flag
      BSF  INTCON,INTE   ; desenmascaramos la INT
      GOTO escape

;----- Timer1 -----
i_tmrl BCF  PIR1,TMR1IF    ; limpiamos el flag
      BSF  STATUS,RP0     ; banco mem 1 (PIE1 está en banco 1)
      BSF  PIE1,TMR1IE    ; desenmascaramos la INT
      BCF  STATUS,RP0     ; banco mem 0

      MOVLW h'3C'        ; cargo 3C en TMR1H y
      MOVWF TMR1H
      MOVLW h'B0'        ; cargo B0 en TMR1L para que
                          ; juntos hagan 3CB0, que hasta FFFF incrementa
                          ; C350 veces, que son 50000 veces en decimal,
                          ; de manera que el timer se demora
                          ; 200ns x 2 x 50000 = 20ms (2 es el preescalador)
      MOVWF TMR1L
      BSF  mot,4          ; subo bandera de ciclo
      GOTO escape         ; así se ejecuta cada 20ms
```

## Proyecto MIHRO (Mobile Intelligent Hexapod Robot)

---

```
;----- Interrupción I2C -----  
i2c_ev BCF    PIR1,SSPIF    ; limpiamos el flag  
      BSF    STATUS,RPO    ; banco mem 1 (PIE1 está en banco 1)  
      BSF    PIE1,SSPIE    ; desenmascaramos la INT  
      BCF    STATUS,RPO    ; banco mem 0  
  
;CALL i2c_com      ; acción en caso de evento en el módulo SSP  
; puesto que es single master, no se usa  
  
;----- Salida de ISR -----  
  
escape MOVF  pcl_t,W  
      MOVWF PCLATH  
      SWAPF stat_t,W    ; le da la vuelta al stat_t  
      MOVWF STATUS        ; y lo reestablece en status  
      SWAPF w_temp,F     ; reestablece W desde W_temp  
      SWAPF w_temp,W     ; dandole la vuelta 2 veces  
  
      RETFIE           ; salgo de rutina  
  
;-----  
END  
;
```

(Total: 1572 líneas)