UNIVERSIDAD POLITÉCNICA DE CARTAGENA
Escuela Técnica Superior de Ingeniería Industrial

Diseño de un robot autónomo bioinspirado utilizando un microcontrolador PIC

Titulación: Ingeniero Técnico Industrial especializado en Automática y Electrónica Industrial
Alumno: Miguel Ángel García Barba
Directores: Juan Suardíaz Muro
Juan Antonio López Riquelme

Cartagena 2012

ÍNDICE

Agradecimientos a:

Juan, por haber acogido mi proyecto con tanto entusiasmo y atención, además de por la ayuda prestada.

A mis padres, que me han apoyado siempre, sabiéndome guiar por el camino adecuado y sin los cuales este proyecto nunca habría sido factible.

Y a todas aquellas personas que con su respaldo personal al proyecto han hecho que saliese adelante lo mejor posible.

De veras gracias.

Miguel Ángel García Barba.

CAPÍTULO 1

INTRODUCCIÓN

La idea de crear un artilugio que, en gran medida, pueda recrear los movimientos y formas de una criatura de la naturaleza llama la atención y puede parecer complejo y sofisticado. Solo con la ayuda de un microcontrolador será posible desarrollar esta tarea, dotándolo no solo de los movimientos básicos de un hexápodo, sino también de la capacidad de sentir su entorno y dar una respuesta a estos sentidos, en concreto de la luz y la proximidad de objetos. Además el color de los ojos (balizas de estado) permitirá conocer lo que está pensando y hacia donde ser dirige dicho robot.

1.1 OBJETIVOS DEL PROYECTO.

Los dispositivos de hoy día son cada vez más avanzados. La tecnología se supera año tras año, y esto no atañe solo a coches más potentes o materiales más ligeros y fuertes, sino ,sobre todo, a la tecnología electrónica. La electrónica nos permite manejar electrones de formas jamás soñadas haciendo que cualquier cosa que se nos ocurra sea posible. Los teléfonos móviles, antes eran eso, meros teléfonos portátiles, ahora se han convertido en mucho más; son dispositivos cuya función de llamada quizás represente la parte menos importante de todo lo que hacen. Hace unos años nadie podría haber imaginado que los ordenadores pudieran ser tan pequeños y extremadamente potentes como lo son ahora; y lo que es más, los robots (autómatas antiguamente programados mecánicamente) que son capaces de realizar hoy día cualquier función y tarea para los seres humanos, ya sea por peligrosa o simplemente por ser tediosa.
Los diseños actuales se basan en la naturaleza ya que presentan mayores ventajas y solventan más dificultades, que los nuevos diseños presentados. Esta es la vía de actuación que se ha elegido para el presente proyecto fin de carrera, el construir un robot inspirado en los movimientos de los hexápodos (bioinspirado), de tal forma que realice sus tareas de una manera mucho más eficiente y con los movimientos que caracterizan a este tipo de seres.

1.2 RESUMEN DE LA MEMORIA.

Las fases de las que se compone el proyecto serán plasmadas a continuación a modo de resumen para ir comprendiendo el alcance de cada capítulo:

- **Capítulo 2. Estado del arte.** En él, haremos una breve historiografía de los hechos más importantes que han acontecido en la historia de la robótica. Se darán también unas nociones básicas de la robótica actual y su porvenir para sumergirnos de lleno en el tema que se trata, los robots, y empezar a entender de primera mano su funcionamiento y utilidades. Abordaremos en este capítulo también unos dispositivos electrónicos llamados microcontroladores, los cuales, como veremos a lo largo de todo el proyecto, nos proporcionarán la base sobre la que trabajar para la programación del robot.

- **Capítulo 3. Metodología y materiales.** Es un capítulo esencialmente teórico, donde se marcarán los procedimientos a la hora de abordar el proyecto del robot hexápodo, y donde se especificarán sus fases para su correcto desarrollo. Por otro lado se perfilarán los materiales a usar, tanto del *Hardware* como del *Software,* sin especificar lo usado, sino dando una visión más global de los elementos imprescindibles necesarios para proceder a las fases antes citadas.

- **Capítulo 4. Análisis de alternativas.** Podría considerarse una continuación del capítulo 3, pues profundizamos y decidimos los materiales usados; se ha separado para dar la idea de unidad a cada capítulo e individualizar, en este, los requisitos exactos para el proyecto y las distintas alternativas que los cumplen, para más tarde dar elección a cada una de ellas. Es un capítulo donde se establecen los componentes exactos a usar en el robot hexápodo (junto a sus correspondientes referencias), así pues, el resto de proyecto se basará en dichos componentes para el modelado del robot final.

- **Capítulo 5. Diseño.** Con este capítulo iniciamos ya la construcción del proyecto y más en concreto el diseño del robot, tanto lo referido a la parte de *software* como de *hardware,* explicando las peculiaridades de cada diseño y mostrando cómo se han creado las funciones que permiten a los servos coordinarse para recrear los movimientos del hexápodo y permitir su movilidad.

- **Capítulo 6. Implementación.** Capítulo especifico de la construcción física del robot donde además se implementa el código en C del programa principal. Se trata de un capítulo con carácter ampliamente didáctico pues en esencia puede considerarse como un tutorial de la construcción paso a paso del hexápodo.Se van mostrando fotos para que la lectura vaya acompasada de documentos gráficos y se pueda comprender más fácilmente.

- **Capítulo 7. Pruebas.** Una vez finalizada la construcción del hexápodo, se pasa a este capítulo donde abarcamos las pruebas, tanto estáticas como dinámicas. En las estáticas se procede al ajuste de los sensores, tanto en su direccionalidad como en los valores de disparo del software, todo ello observando la respuesta que nos ofrecen unos LED a los estímulos exterrnos; finalmente en las pruebas dinámicas se somete al robot a los distintos entornos reales ante los que se puede encontrar, para solventar posibles errores y que le funcionamiento del robot sea el adecuado.

- **Capítulo 8. Conclusiones y trabajo futuro.** Se trata de uno de los capítulos finales donde se detalla una parte importante del proyecto como puede ser el presupuesto; mostramos además las conclusiones obtenidas del capítulo de pruebas, arrojando los resultados y entornos para los cuales el robot es más apto para discurrir realizando sus tareas. Se dan unos esbozos de trabajos futuros con el robot, sobretodo en el campo de la exploración y mostrando el futuro de los diseños bioinspirados.

- **Capítulo 9. Planos.** Podría considerarse un anexo. Capítulo en el cual introducimos los diseños físicos y dimensiones del hexápodo, distinguiendo el esquemático, la placa física y las dimensiones de las distintas partes de la estructura.

- **Capítulo 10. Listado de componentes.** Último capítulo en el cual se muestra la lista de todos los componentes y materiales usados para la construcción del proyecto que se trata, para de este modo en el caso de querer hacer otro, solo tendríamos que remitirnos a ella para comenzar; además se ha incluido un *"pinout"*, es decir, una definición del uso de los pines del microcontrolador y de esta forma proceder a su conexionado con el resto de periféricos.

CAPÍTULO 2

ESTADO DEL ARTE

En este capítulo se muestran algunos hitos del ser humano, relacionados con el avance tecnológico que se produce en la actualidad, se hace notoria la idea de que la robótica ya forma parte nuestra y nos sería imposible vivir sin ella; así mismo se verá como el basar nuestros diseños en formas de la naturaleza ayuda a mejorar muchos aspectos de ellos y como una de las vías de investigación más potentes es la robótica, con la cual es posible hacer multitud de tareas. Otro sub-apartado más que interesante para comprender el interior de los robots es el del microcontrolador, en el cual se expondrá cómo es y cómo funciona, indicando algunos datos adicionales, de forma que se pueda vislumbrar las infinitas posibilidades que ofrecen a la hora de abordar proyectos de este calibre.

Los puntos a continuación mostrados son unas pequeñas pinceladas para recrear la idea de lo que es un robot y lo que implica, de manera que se pueda entender mejor el proyecto que se trata.

2.1 ROBOTS AUTÓNOMOS.

Los robots (del checo robota: sirviente) ya aparecieron por el siglo IV antes de Cristo, el griego Arquias de Tarento construyó un ave mecánica que funcionaba con vapor [Wikipedia 01].

La esencia de un robot, es que pueda mantenerse o realizar la acción para la que está programada por sí solo, es decir que sea autónomo y no requiera de la presencia humana para funcionar. Los artilugios antiguos cuyo funcionamiento era esencialmente mecánico fallaban mucho y no eran lo suficientemente fiables como para encomendarles tareas delicadas, que pudieran influir en la vida de alguna persona. Con el surgir de los dispositivos eléctricos y más adelante los electrónicos, los circuitos que controlaban los actuadores de los robots iban siendo cada vez más fiables y precisos. Hoy día el avance es tal que permite a los robots ser completamente autónomos.

La autonomía antes citada permite a los robots trabajar prácticamente en cualquier campo, siendo uno de los más importantes el campo de la exploración. El robot será capaz de navegar por sí solo, por su entorno esquivando objetos y solventando los posibles problemas que se desarrollen a lo largo de su tarea [Vehículos autónomos]. El campo de la exploración sería la base para el resto de campos aplicados a los robots, como pueden ser, revisión de sistemas, espionaje, mapear zonas, exploración planetaria, transitar campos de combate dando apoyo sanitario o como un combatiente más etc. El campo de la exploración es realmente potente y está basado en la autonomía y buen funcionamiento del robot. De ahí que la importancia de este proyecto resida en crear un robot que sea autónomo y capaz de desenvolverse por su entorno sin problema.

2.2 ROBOTICA INSPIRADA EN LA NATURALEZA.

Desde los tiempos más antiguos cuando el hombre trataba de imitar a los pájaros para volar, se fabricaba unas alas similares a estos; no obstante, como muchos intentos fallidos dieron a entender, esta no era la forma de lograrlo, lo que no significaba que los diseños naturales fuesen erróneos, lo erróneo era la forma de aplicarlos.

Así pues, como más tarde se vio, cuando se aplican movimientos, formas o comportamientos derivados de animales, insectos o plantas de la naturaleza a nuestros diseños, conseguimos realizar tareas complejas o lentas de una forma mucho más rápida y eficiente.

En lo referente al diseño del robot, la configuración de las seis patas ofrece un robot todo-terreno, rápido y recursivo a la hora de solventar los desniveles presentes en cualquier entorno de exploración. Otra ventaja a destacar es la capacidad de transportar cierto peso como carga útil en algún tipo de misión, como pueda ser llevar los equipos necesarios para alguna investigación, o en modelos más grandes el posible transporte de personas heridas por terrenos escarpados y de difícil acceso para unidades móviles con ruedas. Cuantas más patas tenga nuestro robot, es decir, más puntos de apoyo, mayor carga será capaz de mover, con la desventaja del mayor consumo de las baterías. Habría, por tanto, que llegar a un compromiso entre estas dos premisas obteniendo el diseño más eficiente.

2.3 ROBOTS ENTRE NOSOTROS.

Tras la segunda revolución industrial en los trabajos manuales que requerían de más fuerza se empezaron a emplear máquinas que ayudaban en estas tarea: sujetando mejor las piezas, haciendo más fuerza, girando más rápido, etc.

Los trabajos han ido evolucionando y con ellos la maquinaría; la precisión requerida en muchos de ellos solo dejaba la opción de trabajo a las máquinas cuya velocidad y exactitud superaba en gran medida a la mano humana. Estas máquinas más avanzadas y cuyo sistema de control es prácticamente electrónico podrían considerarse robots. Pero no solo es posible encontrar estos "robots" en oficios o en la industria de manufactura de productos, es posible encontrarlos en muchas aplicaciones del día a día; sin ir más lejos cualquier reloj analógico fabricado hoy día cuenta con un dispositivo de reloj que está hecho electrónicamente.

En el hogar también encontramos numerosos dispositivos electrónicos que facilitan las tareas domesticas, como puede ser un microondas, cuyo microchip controla la generación de la radiación microondas y el tiempo de duración; los lavavajillas, las lavadoras, frigoríficos, todos estos electrodomésticos poseen un microcontrolador que controla las variables de cada proceso, ajustando su funcionamiento lo mejor posible y corrigiendo las posibles desviaciones producidas por perturbaciones externas (apertura de la puerta de un frigorífico). Un ejemplo más cercano a lo que serian aplicaciones robóticas domesticas son los novedosos robots aspiradoras [Robots entre nosotros' aspiradora], cuya función es la de barrer el suelo y absorber el polvo; son el ejemplo más intuitivo de robots al servicio del ser humano. Dicha aspiradora no es más que una plataforma motorizada con ruedas cuyos sensores captan el entorno en el que se encuentra, permitiendo sortear obstáculos, evitando caídas y escalones [Robots entre nosotros' sube-escaleras] y siguiendo un algoritmo que le permite recorrer la habitación por completo al tiempo que va limpiando el suelo.

Otro campo donde cada vez más se aplica es en la robótica de la medicina, donde su milimétrica exactitud hace posible intervenciones jamás imaginadas sin la ayuda de los brazos robóticos (cuyo pulso será siempre más estable que el de cualquier experto cirujano) [Robots entre nosotros' cirujano].

Como es posible concluir, los robots están por todos lados y lo que es más, no solo se pueden encontrar en hogares, fábricas, hospitales, comercios, talleres, etc. sino que han sido empleados también fuera de nuestra atmosfera para explorar el universo y otros planetas [Robots entre nosotros' universo], llevando a cabo tareas de investigación, recogiendo muestras, sin poner en peligro vida humana alguna.

Queda así clara la idea de que la robótica más que nunca se encuentra integrada en nuestras vidas y quehaceres diarios; aunque esto no acaba aquí, el imparable desarrollo exponencial de la tecnología va a posibilitar de aquí a unos años que aplicaciones que ahora parezcan impensables o imposibles formen parte de nuestra vida, como lo han hecho los teléfonos móviles, las televisiones o los coches.

2.4 ROBOTICA LA APUESTA SEGURA.

La robótica es una muy importante vía de desarrollo, investigaciones y proyectos que en un futuro darán sus frutos para poder seguir avanzando por otros caminos y de esta forma descubrir nuevos productos.

Si algo es seguro es que la robótica se irá implantando cada vez más según vaya pasando el tiempo; sin ir más lejos, ya se están hablando de combatientes robóticos en futuras guerras, en las cuales los robots serán los que las libren en los campos de batalla, permitiendo así la perdida de menos vidas humanas y de que duren menos tiempo. Actualmente se utilizan los llamados *drones* [Wikipedia'00] no tripulados o UAV (*Unmanned Aerial Vehicle*) en castellano Vehículo Aéreo no Tripulado, con estos tipos de vehículos robóticos la misiones de exploración y espionaje se han convertido en un simple paseo y en los últimos modelos se les ha dotado de armas para ejecutar objetivos sin necesidad de esperar a que vengan los cazas, ahorrando dinero y aumentando en efectividad.

Otra vía en la que se están integrando los robots son las misiones subacuáticas, existiendo robots que imitan las formas y movimientos de las carpas de modo que realizan tareas de inspección de fondos marinos, del casco de los barcos o submarinos, de una forma muy sencilla. Otros robots en consonancia con la naturaleza son unos capaces de escalar árboles, aún hay pocos pero se está logrando que consigan desplazarse por paredes totalmente verticales pudiendo así escalar edificios para posibles misiones de reparación, o limpieza de cristales.

En conclusión, los "Robots" van a por todas y son una inercia a seguir y aún por desarrollar, con lo que si queremos apostar por el caballo ganador, hagámoslo por ellos.

2.5 MICROCONTROLADOR.

Un controlador es un dispositivo empleado para el gobierno de procesos o sistemas. Siempre han existido controladores, desde los más antiguos que basaban su funcionamiento en la mecánica de unos simples engranajes a los más modernos cuyo control está implementado por un micro computador, es decir por microcontroladores [Microcontroladores'01]. El objetivo era el mismo, no obstante, la forma de lograrlo ha ido progresando y avanzando con los años. Antes de los más modernos microcontroladores que tienen todos los periféricos integrados, se trabajaba con microprocesadores a los cuales se les acoplaba periféricos de entrada salida (E/S), de memoria, convertidores, temporizadores (*timers*), etc. Todos estos módulos eran externos a la propia pastilla del microcontrolador, con lo que las placas de circuito impreso ocupaban mucho más y tenían más posibilidad de fallo.

Como antes se ha comentado el microcontrolador es en sí un micro computador, un circuito integrado programable cuya función es la de controlar una tarea determinada y por su ínfimo peso y tamaño es apto para cualquier aplicación portátil o que requiera de bajo consumo, tal es el caso de los móviles, robots, maquinaria en la industria o los más sofisticados satélites de comunicaciones.

Ya que el microcontrolador es un computador dedicado, en su memoria solo reside el programa de la aplicación a realizar. Todos sus periféricos les permiten una gran versatilidad a la hora de contar con ellos en numerosas aplicaciones, pudiendo utilizar sus E/S para captar la información proveniente de sensores o para aplicar señales de salida a algún tipo de actuador.

Atendiendo a sus periféricos, ya que constituyen una parte esencial del microcontrolador y por la cual se seleccionará unos u otros, habrá que decir que no son modificables por el usuario, es decir no se pueden añadir o quitar periféricos, en consecuencia se encuentran integrados en el propio chip de nuestro integrado. Atendiendo a la aplicación para la que vaya destinado será preciso usar uno que con sus capacidades y precio resulte el más rentable económicamente y permita aprovechar mejor sus recursos, no empleando los microcontroladores de gamas superiores que implementen otros periféricos a los cuales no les daremos uso. Algunos de los periféricos más típicos son:

- Convertidor analógico-digital o digital-analógico.
- Comparadores analógicos.
- Modulador por ancho de pulsos (PWM).
- Puertos de comunicación serie; I²C, USB…
- Temporizadores (*Timers*).
- Perro guardián (*Watchdog*).
- Modo bajo consumo (*Sleep*).
- Puertos entrada salida (E/S).

Las principales ventajas en el uso de los microcontroladores son:

- Aumento de la fiabilidad: por la integración de todos los periféricos en la misma pastilla del integrado.
- Gestión eficiente de procesos: no es necesario el empleo de grandes computadoras para la gestión de procesos.
- Reducción del tamaño, consumo y coste: La mayor integración implica no solo una reducción lógica de tamaño, sino también una disminución del consumo y por tanto, del coste, no solo energético sino monetario, por usar menos materiales en la fabricación del integrado.

2.5.1 Construcción.

Respecto a la fabricación de los microcontroladores cabe decir que prácticamente en su mayoría están fabricados con tecnología CMOS (*Complementary Metal Oxide Semiconductor*) ya que poseen unas velocidades de conmutación lo suficientemente elevadas para permitir el correcto funcionamiento de los microcontroladores, además de una de una casi idónea inmunidad al ruido, así como un bajo consumo.

2.5.2 Introducción a las posibles arquitecturas.

Atendiendo a su arquitectura, el microcontrolador implementa una arquitectura Harvard (a continuación descrita), no obstante esta no es la única arquitectura existente en micro computadoras o microcontroladores.

2.5.2.1 Arquitectura Von Neuman

Comenzando con la arquitectura Von Neumann, se enumerarán sus características principales: se dispone de una sola memoria principal donde se almacenan los datos e instrucciones de forma conjunta; para acceder a dicha memoria existe un único bus de comunicaciones. Esta arquitectura supone algunas ventajas como son los accesos a tablas almacenadas en la memoria ROM y una serie de instrucciones ortogonales (se puede utilizar cualquier modo de direccionamiento en cualquier instrucción). Los tipos de buses son: bus de datos por donde se transmite la información desde la CPU a las diferentes posiciones de memoria o viceversa y el bus de direcciones, usado para identificar la posición de memoria a la que se está accediendo. Lo que nos supone un único sistema de buses es el inconveniente de tener que ejecutar secuencialmente las instrucciones para acceder a las direcciones de memoria y ejecutarlas regresando desde/hacia la memoria de datos. Además otra desventaja de este tipo de arquitectura podría ser que algún registro como el contador de programa se corrompiera y apuntara a una dirección de memoria errónea, ejecutándose una instrucción no deseada, lo que podría suponer un fallo del sistema.

2.5.2.2 Arquitectura Harvard

La arquitectura Harvard se diferencia esencialmente de la Von Neumann en poseer dos memorias independientes, una para las instrucciones y la otra para los datos. Tener implementadas las dos memorias de forma independiente supone poder acceder a las dos memorias de forma simultánea, pudiendo realizar operaciones de lectura y escritura en ambas memorias de forma independiente. Este tipo de arquitectura es la que implementan los microcontroladores PIC, la memoria de programa suele ser por tanto de tipo ROM, OTP, EPROM, EEPROM o FLASH mientras que la memoria de datos es de tipo RAM. De esta forma las tablas de datos pueden ser almacenadas en la memoria de programa para evitar su pérdida en el caso de un fallo en la alimentación. La arquitectura Harvard supone el que se puedan realizar transferencias de datos de forma solapada con los ciclos de codificación de instrucciones, es decir, la siguiente instrucción puede ser cargada en la memoria del programa mientras se está ejecutando una instrucción. Una desventaja podría ser el tener que usar instrucciones especiales para acceder a valores en memoria RAM y ROM haciendo la programación un poco complicada (sobre todo si usamos un lenguaje ensamblador, con los lenguajes de alto nivel este problema ya viene solventado por funciones).

2.5.2.3 Computador RISC

En la arquitectura computacional, *RISC* (del inglés *Reduced Instruction Set Computer* = Computación de Juego de Instrucciones Reducidas) es un tipo de microprocesador cuyas características fundamentales se detallan a continuación:

- Instrucciones de tamaño fijo y presentadas en un reducido número de formatos.

- Sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

- Disponen de muchos registros de propósito general.

El objetivo de diseñar máquinas con esta arquitectura es posibilitar la segmentación y el paralelismo en la ejecución de instrucciones y reducir los accesos a memoria, de ahí que dicha arquitectura esté implantada en los microcontroladores (cuyo modelo de memoria es *Harvard*, lo que viene bien al computador RISC pues el conjunto de instrucciones y el conjunto de datos están físicamente separados).

RISC es una filosofía de diseño de CPU para computadora que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse. Terminología más moderna se refiere a esos diseños como arquitecturas de carga-almacenamiento [Wikipedia'02].

2.5.3 Microcontroladores PICs.

Centrándonos en el microcontrolador que se va a usar, como se trata de un microcontrolador de la compañía ***Microchip*** [Microchip], vamos a dar algunos datos históricos de este tipo de microcontroladores PIC:

Los PIC son una familia de microcontroladores tipo *RISC* fabricados por *Microchip Technology Inc. y* derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de *General Instrument.*

El nombre actual no es un acrónimo. En realidad, el nombre completo es *PICmicro*, aunque generalmente se utiliza como *Peripheral Interface Controller* (controlador de interfaz periférico).

El PIC original se diseñó para ser usado con la nueva CPU de 16 bits CP16000. Siendo en general una buena CPU, ésta tenía malas prestaciones de entrada y salida, y el PIC de 8 bits se desarrolló en 1975 para mejorar el rendimiento del sistema quitando peso de E/S a la CPU. El PIC utilizaba microcódigo simple almacenado en ROM para realizar estas tareas; y aunque el término no se usaba por aquel entonces, se trata de un diseño *RISC* que ejecuta una instrucción cada 4 ciclos del oscilador.

En 1985 la división de microelectrónica de *General Instrument* se separa como compañía independiente que es incorporada como filial (el 14 de diciembre de 1987 cambia el nombre a *Microchip Technology* y en 1989 es adquirida por un grupo de inversores) y el nuevo propietario canceló casi todos los desarrollos, que para esas fechas la mayoría estaban obsoletos. El PIC, sin embargo, se mejoró con EPROM para conseguir un controlador de canal programable.

Hoy en día multitud de PICs vienen con varios periféricos incluidos (módulos de comunicación serie, UARTs, núcleos de control de motores, etc.) y con memoria de programa desde 512 a 32.000 palabras (una palabra corresponde a una instrucción en lenguaje ensamblador, y puede ser de 12, 14, 16 ó 32 bits, dependiendo de la familia específica de *PICmicro*) [Wikipedia'03].

Algunas de las características principales de los Microcontroladores PIC son:

- Área de código y de datos separadas (Arquitectura *Harvard*).
- Un reducido número de instrucciones de longitud fija.
- La mayoría de las instrucciones se ejecutan en un solo ciclo de ejecución (4 ciclos de reloj), con ciclos de único retraso en las bifurcaciones y saltos.
- Un solo acumulador (W), cuyo uso (como operador de origen) es implícito (no está especificado en la instrucción).
- Todas las posiciones de la *RAM* funcionan como registros de origen y/o de destino de operaciones matemáticas y otras funciones.
- Una pila de hardware para almacenar instrucciones de regreso de funciones.

- Una relativamente pequeña cantidad de espacio de datos direccionable (típicamente, 256 bytes), extensible a través de manipulación de bancos de memoria.
- El espacio de datos está relacionado con el CPU, puertos, y los registros de los periféricos.
- El contador de programa está también relacionado dentro del espacio de datos, y es posible escribir en él (permitiendo saltos indirectos).

El número de productos que integran en su funcionamiento un microcontrolador, crece exponencialmente día tras día. La industria informática acapara gran parte de los microcontroladores y microprocesadores que se fabrican. Todos los periféricos del ordenador (faxes, impresoras, teclados, ratones, monitores, etc.) pasan por el microcontrolador interno del PC que debe de controlarlos.

Electrodomésticos tan usuales en los hogares como son las lavadoras, o los hornos microondas incorporan también numerosos microcontroladores. Lo mismo sucede en los sistemas de alarma o de vigilancia de los edificios, cuyo cerebro es un microcontrolador que maneja las comunicaciones entre sensores cámaras, dispositivos de grabación, etc. También es posible encontrar estos dispositivos en ascensores, sistemas de climatización en edificios o en cualquier reloj o termómetro digital…

Son de uso común y están en los dispositivos móviles que se utilizan para realizar llamadas, o en cualquier tipo de comunicación ya sea cableada o inalámbrica, cuya misión es la de controlar el envió y recepción de las tramas de datos, con su correspondiente tratamiento para extraer la información.

La instrumentación y la electro-medicina son dos campos idóneos para la implantación de estos circuitos integrados. Uno de los sectores con mayor aplicación es en las industrias automovilísticas que los aplica al control de la climatización, seguridad (cerradura electrónica), sistema de control pirotécnico de los airbags, frenos ABS, etc. Otro ámbito muy importante en el empleo de los microcontroladores son los circuitos integrados, pudiendo albergar en una única pastilla un computador embebido con todos los periféricos necesarios para la tarea que haya sido desarrollado.

CAPÍTULO 3

METODOLOGÍA Y MATERIALES

3.1 METODOLOGÍA.

En esta sección se detallan los pasos seguidos para definir, diseñar e implementar el proyecto, es decir, el robot hexápodo. En el sistema de diseño se suelen seguir una serie de pautas. Que, dependiendo del objetivo del proyecto, llegarán más o menos lejos, el paso final siempre sería su comercialización y siempre se comenzará con la definición básica del proyecto y los requisitos que deberá de cumplir.

El proceso de diseño se ha estructurado en las siguientes fases: especificación de requisitos, establecimiento de la arquitectura del sistema, desarrollo del hardware y software, test y debug, fabricación, experimentos y pruebas.

- ***Especificación de requisitos:*** En esta fase se establecerán los requisitos mínimos que, al menos, deba cumplir el proyecto, es decir, lo que deberá de hacer el robot; ofrece una idea general del proyecto, en la que se irá profundizando conforme se avance en las fases. Deberán de establecerse tanto parámetros relacionados íntimamente con el robot como son dimensiones, peso, consumo y forma como el coste monetario máximo del proyecto final.

- ***Establecimiento de la arquitectura del sistema:*** La base de todo este proyecto gira en torno al microcontrolador y a la construcción mecánica del hexápodo. Así pues, un paso crítico será definir el tipo de PIC y los actuadores mecánicos más adecuados que sean capaces de accionar y mover al robot.

- ***Desarrollo del hardware y software:*** Una vez establecidas las pautas y la configuración básica que va a tener el robot a diseñar, se procede a la fase de diseño, donde se da forma a todas estas ideas. En el diseño quedarán establecidas las dimensiones, propiedades y materiales a usar en la construcción física del robot, teniendo que hacer una serie de pruebas para la elección de de los mejores materiales y, sobre todo, los servomotores que mejor se adapten a los requerimientos exigidos al diseño. También se plasmará en esta fase el diseño de la placa de circuito impreso, en la cual se habrán previsto todos los conectores y clavijas para la comunicación con el mundo exterior y verificación del entorno mediante los sensores.

En esta fase se llevará a cabo numerosas remodelaciones de los diseños, estas remodelaciones deben de ser ejecutadas en este punto y no más lejos, ya que las posibles modificaciones realizadas sobre el modelo real supondrían un costo mayor que si se hubiesen previsto sobre los planos.

- ***Test y depuración:*** Una vez finalizado el proceso de diseño del hardware y software, se lleva a cabo un proceso de simulación, para ver si el comportamiento obtenido es el deseado, antes de proceder a la implementación. Para ello es posible recurrir a programas de diseño gráfico, en los cuales mediante reconstrucciones dimensionales del robot, se observen si existen o no colisiones en las patas al andar o de la existencia de algún otro problema. Para el testeo y debugging del software se emplean programas de simulación de microcontroladores con los cuales es posible determinar si el código creado es capaz de excitar a los servomotores y llevarlos a las posiciones especificadas, todos estos tests se acompañan de la revisión y solución de los errores, hasta lograr que lo diseñado se adecue exactamente a las expectativas del proyecto.

  Una cosa a tener en cuenta en la fase de pruebas del software es que en el simulador, debido a que no es de por si un microcontrolador, no es 100% fiable con lo que más adelante tras la fase de implementación se deberán de realizar otras pruebas para comprobar el correcto funcionamiento de todo el sistema.

- ***Fabricación:*** Ya acabando se procede a la fabricación o implementación física del robot. Lo primero a tratar deberá de ser las partes físicas, con lo que se monta toda la estructura y se le da el acabado deseado; sobre esta estructura se probará en una placa protoboard con el microcontrolador, los movimientos básicos y si todo es correcto, se procederá a la fabricación de la placa final. Todo ello montado y perfectamente conectado dará el resultado final del robot, sin olvidar que falta una última fase de pruebas en la que corregiremos los pequeños fallos de implementación para dejar al robot perfectamente operativo.

- ***Experimentos y pruebas:*** Una vez el robot esté construido lo someteremos a pruebas que muestren su correcto funcionamiento en el entorno para el que ha sido diseñado, es decir, la luz y la interacción con los posibles obstáculos que se encuentre en su camino. Todas estas pruebas nos harán ir perfilado los posibles fallos iniciales de diseño (no previstos) hasta obtener al robot terminado y listo para la tarea que se le encomendé.

3.2 MATERIALES.

3.2.1 Hardware.

El *hardware* empleado en primer lugar es el propio ordenador con el que se ha creado el código, se trata de un portátil con núcleo intel i5, 4 Gb de RAM y 2 Gb dedicados de tarjeta grafica Nvidia, estos son los aspectos más relevantes y que mas implicación tendrían en este proyecto.

En segundo lugar, otra pieza importantísima sería el programador del microcontrolador; se trata de un programador USB PIC K150 [Programador] cuyo zócalo es capaz de albergar gran cantidad de microcontroladores, de hasta 40 patillas, cuenta con un *software* específico, que se detallará en la sección de *software*.

Para las pruebas del circuito electrónico antes de su implementación física en placa de circuito impreso se ha utilizado una placa protoboard de uso común con cablecillos de par trenzado a modo de pistas y puentes.

El hardware obvio que se ha empleado por supuesto es la placa de circuito impreso con los componentes, sensores y el microcontrolador, además del modelo del robot hexápodo con los doce servos y sus doce conectores.

Otras herramientas empleadas han sido: polímetro, para realizar el testeo de la placa base final y su correcto funcionamiento, un osciloscopio, con el que visualizar todas las señales de salida del microcontrolador así como las señales provenientes de los sensores, además de lámparas para comprobar el funcionamiento del robot en entornos de oscuridad siguiendo a una fuente lumínica y diversos objetos con multitud de formas y colores para comprobar la capacidad de esquivar y continuar su camino del robot.

3.2.2 Software.

Se ha empleado muy diverso *software* para la realización de cada una de las etapas del proyecto, como son programación, pruebas, sistema operativo y diseño.

Para la programación del PIC se ha usado un compilador en C para PICs ***PCWH CCS*** versión 4.104, cuya potencia del lenguaje C se aprovecha para compilarlo a lenguaje máquina y programar gran variedad e PICs, las pruebas del software antes de implementarlo han sido realizadas en el ***Proteus ISIS***, programa en el cual se han modelado los servos y el sistema de control de los sensores, y hemos probado ciertas configuraciones para comprobar que la respuestas dada a los servos es la correcta; el sistema operativo usado ha sido el ***Windows 7 Home Edition***, simplemente resaltar que permite gran compatibilidad con los programas antes ejecutados en Windows XP y por último los programas de diseño empleados en el robot se dividen en dos: el destinado a dimensionar la estructura del robot y acotarlo(***Auto CAD*** 2007) y el empleado para hacer el esquemático para luego pasarlo a placa de circuito impreso llevando a cabo la colocación de los componentes y su ruteo (***Orcad PCB*** 9.2). Vamos a describir más detalladamente cada uno de estos programas:

- **Compilador CCS**

Este compilador de lenguaje C ha sido desarrollado por CCS Inc. [Compilador] para permitir la programación de los microcontroladores de una forma más amena mediante el uso del lenguaje C. Dicho *software* cuenta con extensas y librerías y funciones especificas lo que ahorra mucho trabajo a la hora de realizar un programa, lo que permite gran versatilidad para cualquier tipo de proyecto. Dichas funciones trabajan sobre los periféricos del microcontrolador como pueden ser: convertidor A/D, configuración de los *timers*, atender a las E/S, etc.

Incluye también numerosas librerías y ejemplos sobre, control de servos, módulos LCD, teclados, convertidores A/D… El software puede ser integrado en MPLAB y usa 1, 8, 16, 32 bit de tipo entero y 32 bits de tipo flotante.

- **Simulador Proteus ISIS**

Este simulador ampliamente utilizado, permite probar infinidad de circuitos (sobre todo de tipo digital) pudiéndose incorporar en los diseños microcontroladores donde previamente hayamos cargado el programa y configurado su reloj para hacerlos correr y comprobar su funcionamiento.

Por tratarse de un simulador, los datos que arroja no son 100% exactos y puede que no se ajuste al comportamiento real del microcontrolador, por lo que una vez simulado y comprobado que funciona, se deberá de en un integrado real donde modificar los pequeños detalles que se le hayan escapado al simulador para que el programa corra perfectamente [Simulador].

- **Windows 7**

Sistema operativo ampliamente difundido en la actualidad, es la evolución del “Windows XP” y “Windows vista”, incorpora una interfaz gráfica de usuario agradable e intuitiva, permitiendo un cómodo manejo del PC. Los programas que corrían en Windows XP lo deberán de hacer en este sistema operativo, no obstante muchas veces aparecen incompatibilidades y habría entonces que ejecutarlo en una máquina virtual de XP.

- **Auto CAD**

Autodesk AutoCAD es un programa de diseño asistido por computadora para dibujo en dos y tres dimensiones [Autocad de Autodesk]. Actualmente es desarrollado y comercializado por la empresa Autodesk. El término AutoCAD surge como creación de la compañía Autodesk, teniendo su primera aparición en 1982. AutoCAD es un software reconocido a nivel internacional por sus amplias capacidades de edición, que hacen posible el dibujo digital de planos de edificios o la recreación de imágenes en 3D. Elegido por numerosos arquitectos, Ingenieros y diseñadores industriales. Desglosando su nombre, se encuentra que Auto hace referencia a la empresa creadora del software, Autodesk y CAD a Diseño Asistido por Computadora (por sus siglas en inglés).

- **Orcad PCB**

Orcad es un paquete de programas de entre los cuales se encuetran Capture y Layout como los más destacados [Diseño circuitos].

Orcad Capture permite el diseño de esquemas electrónicos en los cuales hayamos incorporado componentes propios mediante la creación de librerías (procesos muy sistemático y fácil que hace que trabar en Orcad sea cómodo ya que usas tus propios componentes); simplemente colocando los símbolos de los componentes y ruteándolos obtendremos el esquemático, y lo que es más, permite la opción de poder simular dicho circuito, obteniendo tensiones, intensidades, potencias, además de gráficas en las cuales simules el circuito con diferentes valores de un componente para ver cual resulta más adecuado para el diseño. En definitiva, si se trata de un diseño electrónico, sobretodo un circuito analógico, Capture es tu aliado.

Orcad Layout, contenido en el paquete Orcad, es el programa que transforma el esquemático realizado en capture al diseño de la placa real; se disponen de varias opciones en cuanto a colocación de componentes y ruteo, no obstante lo mas aconsejado es desatender la colocación que nos proporciona Layout y disponer los componentes de forma que optimicemos las dimensiones de la placa lo máximo posible. En lo referido al ruteo, de tratarse de placas complejas la opción de *autorute* puede ser buena opción, interviniendo sobe el ruteado final para completarlo al 100%. Se elija la opción que se elija, debe de ser tal que nos proporcione la placa que mejor se adapte a las expectativas de diseño.

También en Layout se pueden crear librerías, siendo esto necesario, pues le introduciremos las dimensiones reales de los componentes a utilizar para no encontrar problemas a la hora de su colocación y soldadura y que de este modo encajen perfectamente.

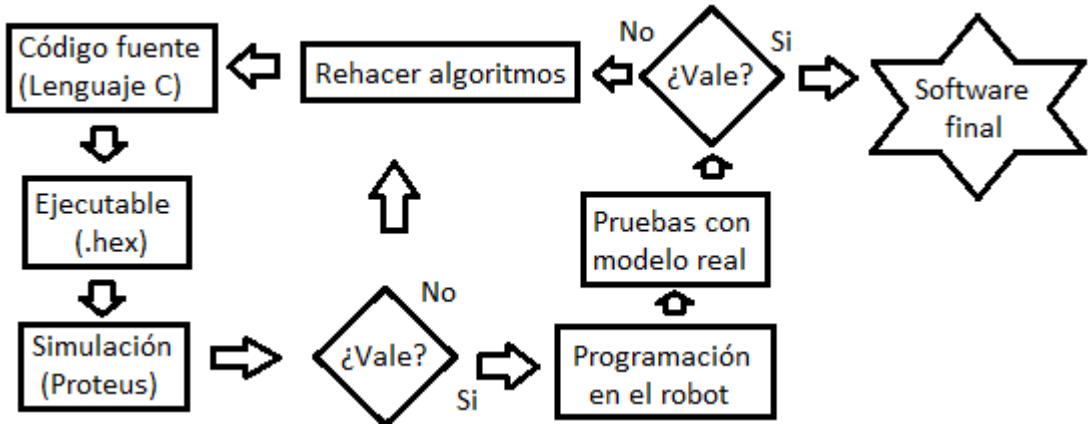


Figura 2.1: *Diagrama de flujo para los pasos del desarrollo software.*

CAPÍTULO 4

ANÁLISIS DE ALTERNATIVAS

En esta sección se entra en más profundidad las características que se deberán de aportar al hexápodo, para posteriormente y en base a eso elegir y seleccionar las distintas alternativas que mejor se adecuen a los requisitos, tanto desde el punto de vista del *hardware* (tipo de microcontrolador, servos, sensores y estructura) como del *software* (Lenguaje de programación y software de desarrollo).

.

4.1 REQUISITOS.

Los requisitos cara al robot podrían ser cuatro: fuerza de las patas, algoritmo solvente, estructura, baterías potentes, sensores efectivos.

La masa del robot va a estar entorno a los 1,2 kg (exactamente 1,18Kg) con lo que deberá de contar con unos servomotores lo suficientemente fuertes y potentes como para no sólo levantar ese peso, sino hacer que se transporte de una forma ágil y fluida. Ya que es un hexápodo contará con 6 patas, esto reparte los 1,18 Kg de masa entre 6, tocando a unos 196.66 gramos a cada servo, lo que conlleva una importante carga en las patas, y más aun cuando realizando uno de sus pasos, se encuentre sobre 3 de sus 6 patas, lo que supone una masa de 393,33 gramos por servo.

Esto no es todo, no nos valen solo unas patas fuertes; como nuestro robot deberá de sortear obstáculos de una forma más o menos rápida, los servos le deberán de aportar esa velocidad extra que requiere, debiendo de buscar unos servos que nos reúnan ambas características: velocidad y fuerza.

Respecto al material de la estructura se debe de pensar en alguno que sea rígido pero liviano y resistente, además deberá de permitir cierta flexibilidad para que no parezca tan tosco el robot cuando camine.

A este respecto, como queremos un robot futurista y que se desenvuelva en entornos de baja luminosidad, le incluiremos algún tipo de iluminación para resaltar su silueta e incrementar el misterio que lo envuelve al caminar sólo por lugares hostiles.

Por tanto, si conseguimos que dicha luz se transmita por la estructura en sí, lograremos el efecto deseado, debiendo de ser pues dicha estructura, transparente o translucida.

Pasando al algoritmo a implementar, debe de ser tal que permita al robot alcanzar la fuente lumínica sin ningún problema (eficaz) y en el menor tiempo posible (eficiente) atendiendo siempre a los posibles obstáculos que se encuentre a su paso; para ello no sólo debe de estar el programa bien hecho, sino que los sensores a utilizar deben de ser los adecuados y el microcontrolador tiene que tener ciertos periféricos como dispositivos E/S y convertidores A/D para el procesado de las señal de los sensores.

En las baterías, el hecho de tener que controlar 12 servos implica un altísimo consumo, ya que en esencia el servo es un motor de corriente continua y esto multiplicado por doce nos da unas prestaciones de batería, difícilmente dadas por cualquiera; de este modo deberemos de incorporarle una fuente de alimentación lo suficientemente potente para aguantar los requerimientos energéticos de nuestro robot.

Por último nos centramos en los sensores, los dividiremos en dos grupos, los que responden a la luz visible y los que lo hacen a la luz infrarroja. Los de luz visible deben de tener una respuesta lo más rápida posible a los cambios de luz, es decir su tiempo de respuesta tiene que ser bajo, esto monitorizará de una forma más eficaz el entorno respondiendo adecuadamente a las posibles perturbaciones que se nos presenten, además su salida debe de poder tener una escala grande para poder precisar más los rangos y umbrales a los que responderá el *software* que implementemos. Atendiendo a los de infrarrojos para esquivar objetos, si en los de luz visible pedíamos una respuesta rápida, en estos requeriremos una respuesta aún más rápida, ya que se emplean para una aplicación más crítica en la que está en peligro la integridad del robot, además deben de tener un rango amplio de medición para poder detectar objetos a distintas distancias.

4.2 EVALUACIÓN Y ELECCIÓN DE ALTERNATIVAS.

Una vez analizados los requisitos que deben de cumplir cada uno de los elementos de nuestro robot, pasamos a valorar las posibles opciones indicando los pros y los contras, para más tarde elegir la más adecuada y desarrollar la implementación entorno a esa elección.

4.2.1 Microcontrolador.

Antes de seleccionar un microcontrolador, tenemos que tener claro los requisitos que dicho microcontrolador debe de tener. Nos vamos a centrar en los microcontroladores de *Microchip Technology Inc.* [Microchip] por ser de los más populares al poseer una amplia gama, entre los que destacan la gama enana, media y alta, además de que al poseer una estructura interna similar, conociendo la programación de uno de ellos podemos llevar a cabo la programación de cualquiera de los otros, lo que nos supone una gran ventaja para el desarrollo de proyectos futuros.
Comencemos a describir las características principales:

- *Entradas/Salidas**:***

  En los conectores de los servos encontramos 3 cables: el marrón es la masa, el rojo es la alimentación y el naranja es la señal de entrada.

  Ya que vamos a controlar doce servos, necesitaremos poder controlar doce señales diferentes, aquí radica uno de los problemas más importantes a la hora de escoger el PIC a usar; el problema no reside en contar con un microcontrolador con doce salidas PWM, ya que dicha codificación la vamos a implementar nosotros por software (es uno de los aspectos importantes de este proyecto, el controlar los doce servos sin emplear estos periféricos del microcontrolador), el problema estaría en contar con las patillas de entrada/salida suficientes, ya que deberemos de disponer de al menos doce E/S, cinco patillas para el muestreo de los sensores, dos para introducir el modo y dos patillas para las balizas de estado de los ojos, además sería bueno disponer de alguna más para futuras aplicaciones. Con lo que en total deberemos de tener como poco 12+5+2+2=21 puertos de E/S.

- *Consumo:*

  Como nuestro robot cuenta con una alimentación autónoma, es decir, baterías, es importante conseguir un consumo global lo más bajo posible, esto se consigue disminuyendo el consumo particular de cada uno de los componentes; el disponer de un microcontrolador con el consumo lo más pequeño posible es una ventaja que quedará reflejada en el tiempo de trabajo del robot. Además un menor consumo conlleva un menor calentamiento del microcontrolador, lo que alargará su vida y no pondrá en peligro ningún otro componente adyacente debido al calor.

- *Memoria:*

  Ya que en el PIC vamos a implementar los movimientos y pasos que se deben de ejecutar para ir a la fuente lumínica el espacio en la memoria de programa debe de ser lo suficientemente grande, para no solo crear estas funciones de andar, sino también para la implementación del algoritmo que nos identifique la luz y nos guie a ella. Según vayamos creando el programa se hará una estimación de la memoria requerida, pero con 7K de memoria cumplimos más que de sobra con nuestras expectativas, dejándonos espacio para posibles ampliaciones de los movimientos.

- *Procesado:*

  Puede ser necesario que el microcontrolador realice cálculos críticos en un tiempo limitado, así pues la velocidad de procesado de la información que nos aportan los sensores es muy importante viniendo determinada no solo por la construcción *hardware* del PIC sino también por el cristal de cuarzo que coloquemos; de este modo una velocidad de 4 MHz (T= 1/4.000.000 = 0.25µs) nos da un tiempo de ejecución de instrucción de 4T es decir de 1µs, suficientemente pequeño para que se realice un buen análisis de datos.

  La precisión es también importante, con lo que si no es suficiente con un microcontrolador de 8 bit, habrá que recurrir a uno de 16 o 32 bits (no obstante para nuestro proyecto no se requerirán anchos de palabra tan grandes).

- *Convertidor A/D:*

  Como la funcionalidad del robot es desenvolverse con soltura por entornos oscuros, la información proveniente de los sensores es crítica, siendo necesario buen módulo convertidor Analógico-Digital para captar con precisión la mayor cantidad de información posible.

4.2.2 Estructura.

La estructura es lo que le va a dar forma a nuestro robot, tratándose de un hexápodo, tenemos que hacerla de forma tal que recree en la medida de lo posible la forma de este.

Como anteriormente comentamos las características que deben de sobresalir en el material empleado han de ser, la rigidez pero con cierto grado de flexibilidad y gran resistencia, debe de ser ligero y transparente o translucido para los efectos lumínicos que vamos a crear.

4.2.3. Servos.

La elección del servo es una de las piezas claves del proyecto, el robot no puede moverse sin los motores que accionan sus patas, y estos actuadores son los servos. El peso del robot (1.2Kg) no debe de ser un impedimento que nos eche atrás a la hora de continuar con el proyecto por no encontrar servos potentes o cuyo costo nos los haga inasequibles; lo que tenemos que hacer es evaluar los servos disponibles en el mercado y llevar a cabo una autentica tarea de ingeniero destinado a compras, es decir, comparar características frente a calidad y precios.

Los datos de los requerimientos de los servos se han obtenido empíricamente probando distintos tipos de servo con diferentes características, llegando a las conclusiones siguientes: requerimos unos servos con un torque superior a 3.5 Kg/cm^2 como mínimo, velocidades nunca inferiores a 0.21seg/60º y con un consumo no excesivamente elevado (recordemos que la fuente de alimentación del robot son unas baterías y cuanto menos sea el consumo particular de los componentes, menor lo será el consumo global haciendo que le robot pueda trabajar durante más tiempo), además dichos servos no pueden pesar demasiado ya que le costaría más al robot realizar los movimientos e incrementaría su masa total. También las dimensiones de los mismos deben de estar en armonía con el tamaño de nuestro robot, unos servos muy pequeños les darían una apariencia enclenque a las patas y unos muy grandes lo harían tosco y pesado.

Además, y esto ya es cuestión estética, el plástico del que esté hecha la carcasa del servo debe de permitir la adherencia de pintura en espray, ya que el aspecto final del robot, incluirá partes transparentes con partes blancas.

Los servos son controlados mediante PWM (Pulse Wide Modulation = Modulación por Ancho de Pulso) por lo que deberemos de construir un *software* (un algoritmo) capaz de traducir los ángulos requeridos en cada momento a PWM con lo que una parte importante del proyecto es conseguir una programación eficiente y que cumpla con las expectativas que se le requieren.

4.2.4 Baterías.

Otro elemento crítico, las baterías. A lo largo de esta exposición estamos haciendo especial hincapié en reducir el consumo a lo mínimo, espero que en esta sección quede más claro el porqué de la insistencia en esta idea.

Las baterías (comúnmente llamadas pilas) son un elemento capaz de almacenar en energía química la energía eléctrica para su posterior uso. El que sean recargables indica que dicha reacción química puede ser reversible mediante la aplicación de una diferencia de potencial, para su posterior transformación de nuevo en energía eléctrica.

Como toda transformación de energía no es perfecta y por cada ciclo de carga y descarga la batería va perdiendo calidad y capacidad, permitiendo un número finito de veces su carga y descarga. Además otro aspecto de las baterías es su capacidad de producir energía eléctrica, como se ha dicho antes dicha energía eléctrica deriva de una reacción química, con lo que cuanta mayor energía eléctrica queramos, mayor deberá de ser dicha reacción química y mayor deberá de ser el tamaño de la batería. Por tanto es importante encontrar un compromiso entre la energía que nos va a aportar y el tamaño y peso de la batería.

La diferencia de potencial con la que funciona nuestro robot está entre los 4.8 y 6 voltios, con lo que si bien no existiese una batería que nos diese esa tensión, deberíamos de poner en serie tantas células como hiciesen falta hasta alcanzar dicho umbral. Además, por muy escaso que queramos que sea nuestro consumo, debemos de sumar el de los distintos elementos del sistema dándonos un consumo no inferior a los 2.000mA en sus pasos más suaves (es claro que por poco que puedan consumir los mejores motores de corriente continua, si tenemos 12 trabajando, esto será notorio a la hora de calcular la intensidad requerida).

En resumen, harán falta unas baterías cuya intensidad no baje de los 3.000mAh y si empleamos 4 células estas deberán de tener de 1.2 a 1.5 V/célula.

4.2.5 Luces.

La interacción por entornos oscuros puede parecer curiosa e incluso misteriosa (Los ruidos de maquinaria o de dispositivos eléctricos en la oscuridad acechando han sentado las bases de muchas obras de ciencia ficción) no obstante, nuestra finalidad no es causar temor en los operarios o personal que trabajen con el hexápodo con lo que vamos a dotar al robot de una serie de luces que nos alerten de su presencia, para no solo darle un aspecto más vistoso y ser estéticamente vanguardista, sino para observar su desplazamiento y ver como ejecuta los movimientos e interacciona con su entorno.

Como siempre volvemos al tema del bajo consumo, y no es que solo porque tengamos que depender de de unas baterías limitadas, sino que como ahora se está viendo, hay que apostar por la defensa del planeta y una de esas formas es el producir equipos lo más respetuosos posibles; una forma de respetar al medio ambiente es mediante el consumo moderado de los dispositivos, por tanto, el robot que aquí vemos también podría ser considerado una forma de acercarse a ese nuevo camino, el camino de las cosas eficientes y respetuosas por el medio ambiente, si se cambia de mentalidad, adaptémonos a este cambio para seguir en liza en este mundo tan competitivo, es la ley del más fuerte, si no te adaptas mueres.

Bajo ese precepto las luces van a ser de bajo consumo y ya que se trata de un dispositivo esencialmente electrónico que mejor que unas luces LED (Light Emitting Diodo), habrá que colocarle resistencias limitadoras en serie, pero eso ya es cuestión de la construcción *hardware,* que se trata más adelante.

4.2.6 Sensores.

- *Lumínicos:*

Queremos unos sensores cuya salida sea lo más lineal posible, con un amplio rango de valores y con una respuesta más o menos rápida. La linealidad es debido a la forma de procesado de la señal, si suponemos una señal cuya relación voltaje/intensidad lumínica sea lineal, podremos definir mejor los rangos para los cuales el programa detalla cada acción, además así podremos proporcionar mejor la respuesta del robot a la señal recibida por el sensores, ya que esta va en función a la cantidad de luz incidente.

El obtener una respuesta rápida es fundamental si la tarea que se le encomienda al robot es la de seguimiento del foco de luz, no obstante no es crítica, ya que la velocidad de actuación de los servos va a ser siempre muy superior a la velocidad de respuesta de cualquier sensor lumínico.

- *Infrarrojos:*

El tiempo de respuesta de estos sensores si es crítico, como se tratan de unos sensores de proximidad (detectan y miden la distancia de un objeto al sensor) el conocer lo más rápidamente posible la existencia de este objeto es fundamental para evitar que el robot choque, se desoriente, o se le entorpezcan las patas con el objeto detectado. Con lo que el tiempo de retardo debemos de procurar que sea de cualquier modo inferior a 50 ms.

Otro aspecto importante de sensores es su salida, como nuestro microcontrolador pose cinco convertidores A/D el obtener una señal analógica proporcional a la distancia del objeto nos sería de gran ayuda, ya que el procesado de estas señales por los convertidores, es sencillo y el software a implementar nos ofrecería muchas más ventajas que si usáramos otro tipo de salidas.

4.2.7 Herramientas informáticas.

Este sub-apartado no es más que una recopilación de todo el *software* a usar (ya que ha sido descrito en profundidad en la sección de *software* del punto 3.2.2)

El sistema operativo empleado es el del ordenador portátil que se tuvo disponible para realizar el proyecto, y se trata del **Windows 7 Home Edition**, que alberga una interfaz grafica de usuario agradable e intuitiva, permitiendo correr programas que corrían en versiones anteriores de Windows como el "XP".

Para el desarrollo del programa que contiene el microcontrolador se emplea un **compilador CCS** de C a código máquina ya que es uno de los más versátiles del mercado y posee una gran cuantía de librerías y dispositivos a programar.

Las simulaciones se realizan con el **Proteus ISIS** uno de los programas que permiten incorporar microcontroladores y cargarles el programa para comprobar su correcto funcionamiento junto a algunos periféricos. Es de gran utilidad sobre todo para proyectos con microcontroladores sencillos.

El diseño del esquemático y de la placa de circuito impreso se llevó a cabo con el **Orcad PCB**, un programa de diseño y simulación electrónica, para nuestro propósito solo empleamos los paquetes para el diseño, destacando el Capture y el Layout Plus.

La estructura y el robot en general, fueron diseñados por el **Auto CAD**, *software* de diseño gráfico afamado mundialmente y empleado por diversos profesionales del sector de la ingeniería.

4.3 Elección final.

Una vez realizado el estudio de las posibles alternativas, y habiendo valorado los requisitos mínimos que debe cumplir cada parte del robot, pasamos a seleccionar la mejor solución.

- Microcontrolador:

Microchip dispone de tres gamas de microcontroladores PIC, la enana, media y alta. Para los requerimientos de nuestro proyecto la gama enana se nos queda corta y la alta nos es excesiva, ya que como dijimos en capítulos anteriores a la hora de seleccionar un PIC deberá de ser aquel en el cual se desaprovechen los menores periféricos posibles, ya que la relación coste/uso sería muy alta.

Sea cual fuese nuestra elección, sería buena en el sentido de aprender a usar este tipo de microcontrolador de *Microchip* ya que cualquier PIC se programa como el resto, es decir emplean las mismas instrucciones (los de gama superior tienen más), rangos de voltaje similares, bajo consumo, e incluso su compilación en el programa que usamos es igual, solo que cada PIC lo definiremos con distinto nombre.

Nos quedamos pues con la gama media en el cual usaremos los 16F87xA por tener la cantidad de puertos necesarios para su conexión con el mundo exterior y servomotores además de contar con los módulos convertidores A/D necesarios. Otra ventaja de los 16F87xA es la memoria de programa interna, que oscila entre 7 y 14 k, nos proporciona las líneas de código suficientes para implementar los movimientos del robot así como los algoritmos necesarios para el tratamiento de las señales provenientes de los sensores. Se requieren de osciladores de cuarzo externo, esta ventaja nos proporciona una oscilación más precisa que si contase con oscilador interno.

La letra F nos indica que la memoria de programa es de tipo Flash, es una memoria no volátil, de bajo consumo y que ese puede escribir y borrar eléctricamente. Su funcionamiento es similar a las memorias ROM o RAM, pero consume menos y es más pequeña. A diferencia de la ROM, la memoria Flash es programable en el propio circuito, además es más rápida y de mayor densidad que la EEPROM.

La alternativa Flash es más recomendable que la EEPROM cuando se necesita gran cantidad de memoria de programa no volátil. Es más rápida y tolera mayor cantidad de ciclos de escritura/borrado.

Por fin seleccionamos el microcontrolador **PIC 16F876A**, cuyas características principales se detallan a continuación:

- ✓ 5 canales de conversión analógica-digital (convertidor A/D de 10bit = buena resolución).
- ✓ Memoria de programa 14.3 Kbyte, que permiten 8192 instrucciones.
- ✓ 22 Puertos de Entrada/Salida.
- ✓ Memoria EEPROM de 256 bytes.
- ✓ Comunicación serie mediante I²C.
- ✓ 2 Módulos de Modulación por ancho de pulso (PWM).
- ✓ 2 Comparadores Analógicos.
- ✓ 3 Timers.
- ✓ Oscilador hasta 20 MHz

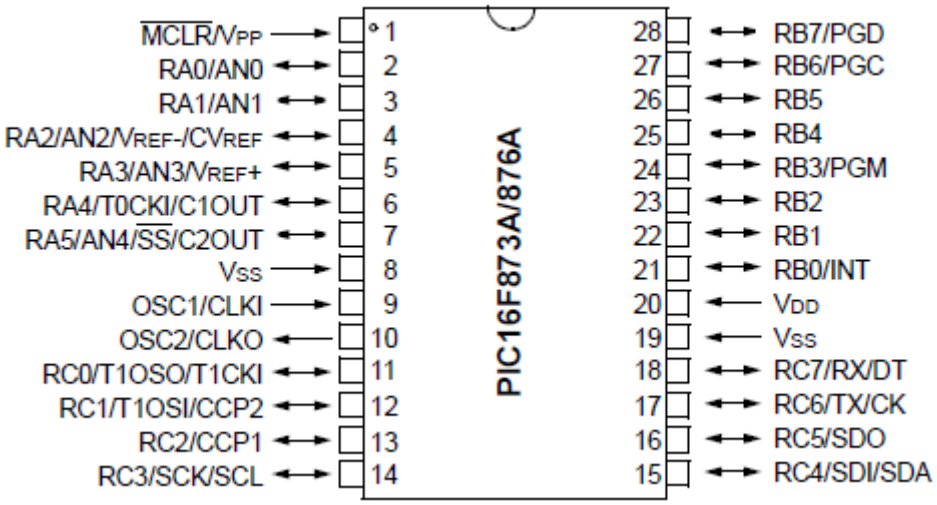


Figura 4.1: *Patillaje PIC 16F876A*

Aunque la empresa *Microchip* proporciona una serie de herramientas de desarrollo totalmente gratuitas y potentes como el entorno de desarrollo MPLAB y el compilador de C C18, se ha optado por emplear en la programación del firmware del PIC el conocidísimo compilador CCS, ya que a diferencia de las herramientas ofrecidas por *Microchip*, este incorpora bibliotecas para el uso de diferentes dispositivos permitiendo de esta forma reducir el tiempo de desarrollo de proyectos, tanto los más sencillos como los más complejos. Además si se desea el compilador CCS puede ser integrado con MPLAB.

- *Estructura:*

Tras catalogar y comparar diferentes relaciones calidad/precio dentro de los requerimientos del material que constituiría la estructura (recordemos que debía de ser rígida y resistente, pero a la vez debía de tener cierta flexibilidad, ser ligera y transparente o translucida), nos decidimos por emplear **metraquilato**. Este se compro en grandes planchas, para maximizar la función del usos/precio para futuras construcciones con este mismo material. Existen diferentes grosores de metraquilato, queremos la máxima ligereza posible, pero a la vez que nos resista el material los movimientos del hexápodo, por lo que no se pueden emplear grosores por debajo de 2.5 mm, de este modo se decidió duplicar dicha cantidad para asegurarnos la resistencia de la estructura, sin sacrificar la flexibilidad ni el peso (como veremos en capítulos posteriores, la forma de la estructura permite proteger los dispositivos internos del robot sin incrementar notablemente el peso).

- *Servos:*

Con todo lo especificado en los requerimientos de capítulos anteriores, conseguimos encontrar un servo que reunía todas las características a un precio asombrosamente bajo (el cual sería un aspecto importantísimo si quisiéramos en un futuro lanzar el producto al mercado; el precio individual de los componentes nos establece el precio final y cuanto más bajo sea este precio, mayor competitividad alcanzaremos en el mercado, batiendo a la competencia y haciéndonos con las ventas).

El servo usado fue el **HK15138** [Caracteristicas'Servos] un servo analógico estándar cuyas características se detallarán a continuación:

- ✓ Masa: 38 gramos
- ✓ Velocidad cada 60º: 0.21 segundos a 4.8v y 0.17segundos a 6v.
- ✓ Torque: 3.8Kg/cm² a 4.8 v y 4.3 Kg/cm² 6v.
- ✓ Alimentación de 4.8v a 6v.
- ✓ Las dimensiones se detallarán en la sección de "planos".

Como vemos los requisitos mínimos son de sobra suplidos, la fuerza y velocidad de estos servos nos van a recrear los movimientos lo más natural y fluidamente posible, con lo que nuestra tarea por la parte mecánica ha acabado aquí.

- *Baterías.*

Hemos escogido las baterías lo más livianamente posible sin perder la capacidad mínima necesaria, es decir, en el mercado podemos encontrar baterías con capacidades de hasta 30.000mAh ¿Y porque no se usan estas baterías si suplen de sobra con la energía consumida en todo momento? [Caracteristicas'Baterías], pues bien dichas baterías están pensadas para equipos normalmente estáticos y suelen ser de plomo-ácido (como las baterías de los coches) son baterías que pesan muchas veces decenas de Kg, con lo que el comparar el peso con la capacidad de cada batería es importantísimo, sin contar con los cientos de Euros que nos costarían. Existen numerosos tipos de baterías y de entre todos ellos nos hemos decantado por las de Níquel y metal-hidruro (Ni-MH) en concreto las **HHR-450A-1Z** [Datasheet adjunto en "Anexos Datasheets"] cuyas características destacamos a continuación:

- ✓ Índice de descarga medio: 4.500mAh.
- ✓ Tensión nominal por célula: 1.2v.
- ✓ Masa: 60 gramos.
- ✓ Tamaño: Fat 5/4ª.
- ✓ Dimensiones: Ø: 18.2 mm x 67mm.

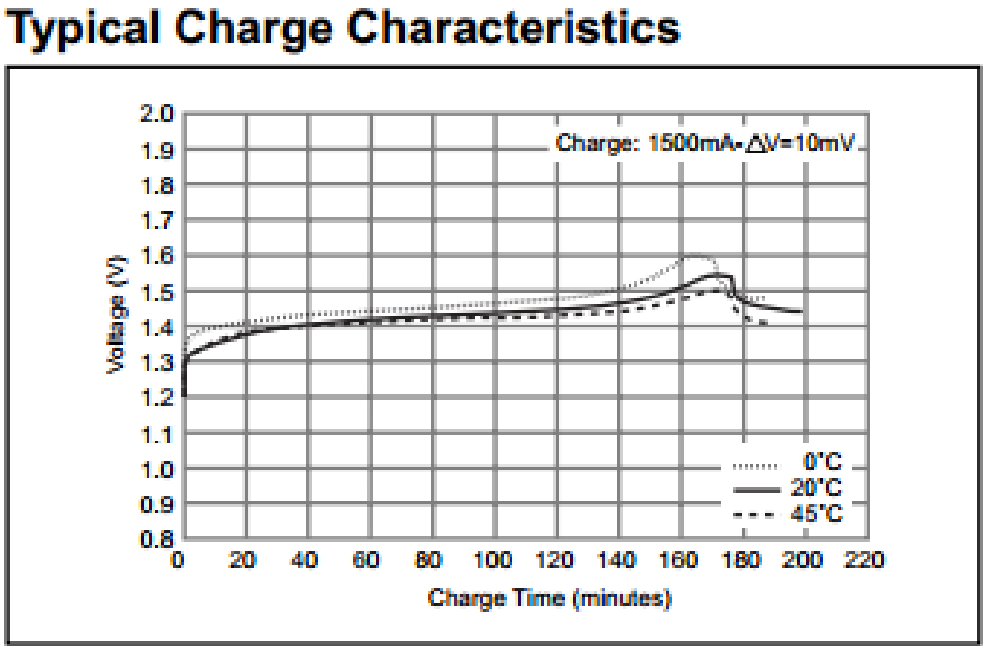


Figura 4.2: *Curva de carga de una célula*

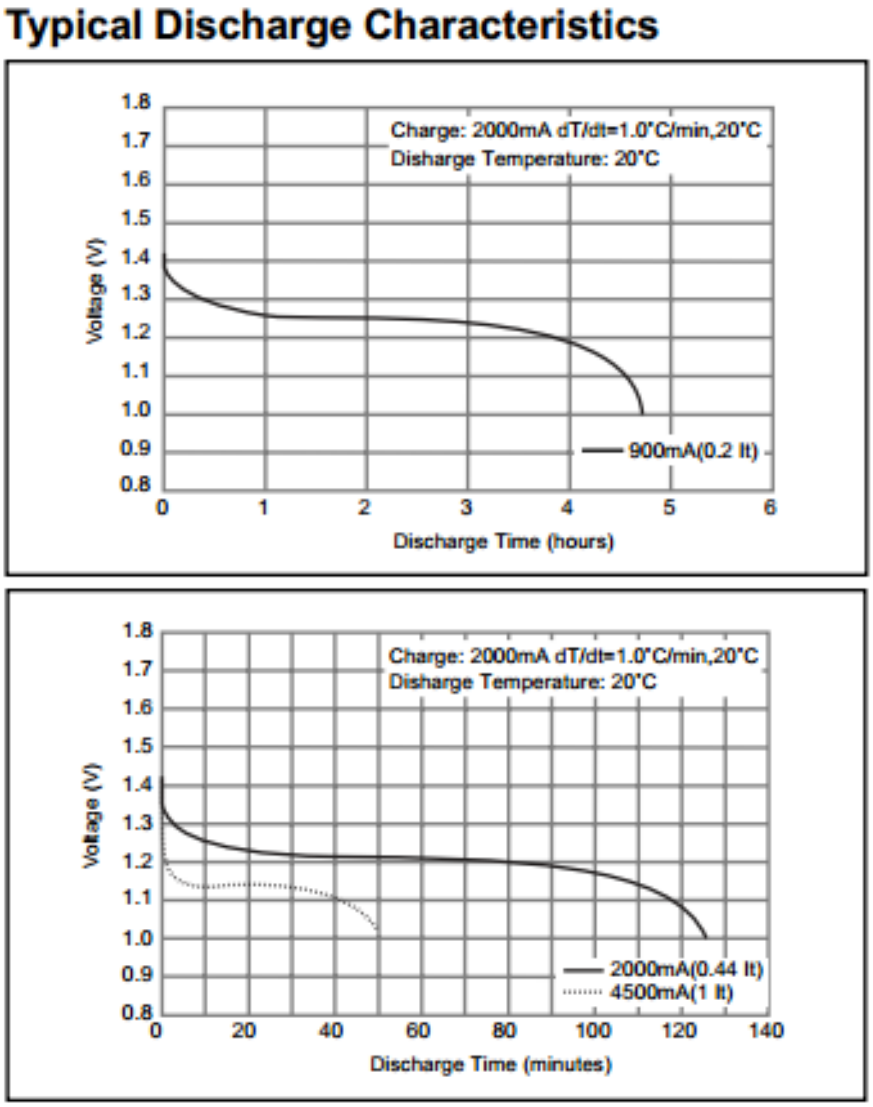


Figura 4.3 y 4.4: *Curvas de descarga correspondientes a una célula con distintos requerimientos de intensidad*

Debido al uso de 12 servos y a que cada uno contiene un motor de corriente continua, el consumo global del robot en momentos de alta actividad motriz puede ser elevado, requiriéndose de una fuente de alimentación que nos proporcione suficiente potencia para aguantar las exigencias del robot y nos supla con las necesidades energéticas de este.

La popularización del aeromodelismo ha hecho que se desarrollen en gran medida todo tipo de baterías, siendo las más novedosas las de Li-Po (Litio y Polímero). Dichas baterías nos aportan índices de descargas increíblemente altos, de decenas de amperios, sin sufrir daños; como dice la famosa frase “la energía ni se crea ni se destruye” con lo que cuanta más intensidad demanden los equipos, antes se descargan las baterías, no obstante para aplicaciones que requiriesen de altos picos de intensidad son perfectas. Se valoró el uso de estas baterías o más en concreto las de Litio, no obstante el incremento considerable del costo del robot descartó la idea.

- *Luces:*

Como se ha especificado se tratarán de luces LED, ya que el diseño de nuestro robot va a ser futurista y tendrá partes transparentes y otras blancas, le colocaremos luces **LED Azules (465nm).**

Como se va a mover por entornos sombríos, una forma de que se presente más majestuoso es dotándolo de luces LED que le den un aspecto más futurista. Dichas luces LED serán azules en las extremidades y una blanca en el cuerpo.

Deberemos de lograr que dicha iluminación propia del robot no interfiera en los sensores luminosos que este posee. De este modo se dispondrán los LED de forma que ayudados de las leyes físicas de la reflexión y refracción conduzcamos el haz lumínico lo más verticalmente posible hacia el suelo (**Figura 4.5**), y así encontramos en cada una de las patas un LED azul enfocando hacia el suelo, esto, sumado con el resto de las patas, crea un halo que envuelve al robot y recrea la sensación de que estuviese flotando. El LED blanco de las estructura tienen la misión de darle volumen al robot y de mostrar más acuciadamente las costillas que constituyen dicha vaporosa estructura. De este modo lograremos hacer nuestro hexápodo más vistoso y agradable a la vista.

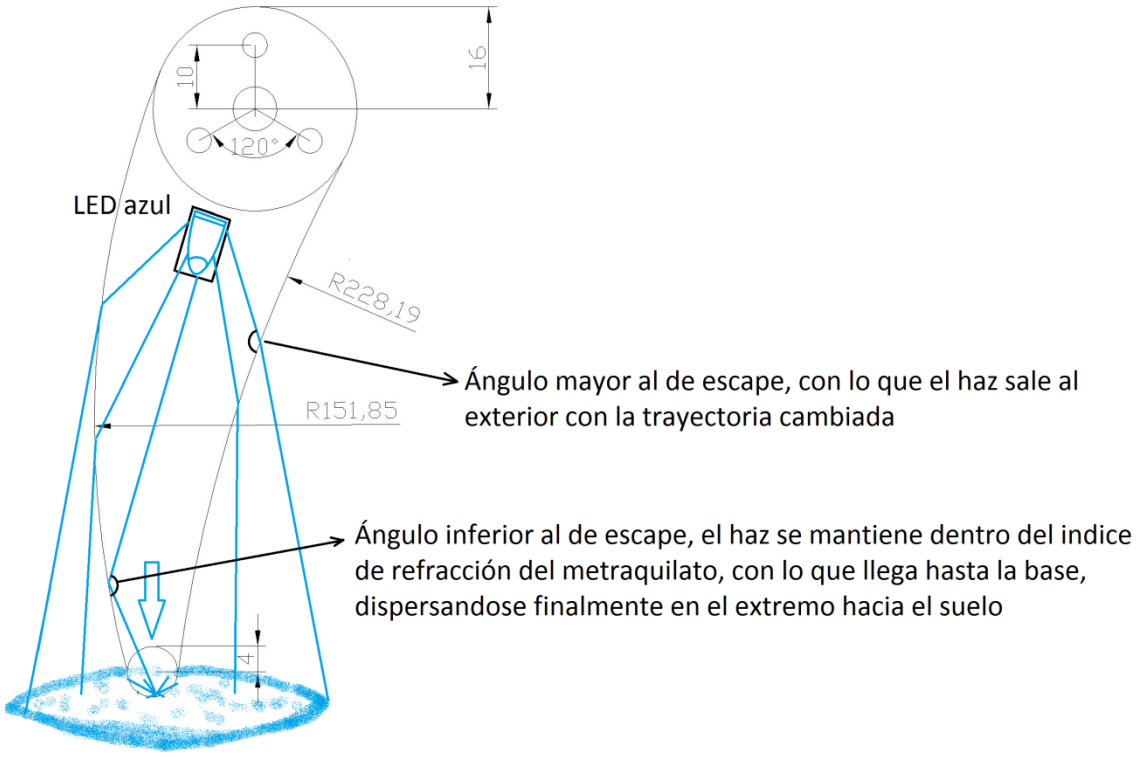


Figura 4.5: *Detalle de la refracción de la luz del LED en las patas.*

- Sensores:

Para los de luz visible, se estuvieron barajando las posibilidades de fotodiodos, fototransistores y LDRs, los fotodiodos requerían de un circuito de acondicionamiento que encarecía el precio final y complicaba el esquemático; existen poca variedad de fototransistores que respondan bien a la luz visible, la mayoría responden a la luz infrarroja, y también requeriría de un circuito de acondicionamiento más o menos complejo, y finalmente en las LDRs encontramos la solución, ya que poseen gran sensibilidad a la luz visible, con un simple divisor resistivo podemos linealizar su comportamiento y obtener una salida en tensión fácilmente procesable y por tratarse en esencia de resistencias, son de fácil acceso y bajo coste. En concreto elegiremos una cuyo valor óhmico nominal esté entorno a 100 KΩ.

La elección de los sensores de infrarrojos ha sido de las más complicadas del proyecto, ha supuesto horas y horas de búsqueda en datasheets de componentes, tales como fotodiodos y fototransistores con sus circuitos de acondicionamiento, de realizar pruebas con ellos y circuitos amplificadores, produciéndose muchos fallos y errores porque la luz ambiente falseaban la medida con lo que se recurrió a un decodificador de tonos para captar las señales emitidas por los infrarrojos; finalmente no se obtuvieron los resultados deseados, por tanto se optó por adquirir un módulo comercial, cuyo precio merece la pena si lo comparamos con el resto de horas que deberíamos de haber echado para obtener finalmente el circuito deseado. Además el propio sensor de medición infrarrojo encapsula el emisor, receptor y los circuitos para el acondicionamiento de las señales en un tamaño reducido y liviano, dándonos una salida analógica (era lo deseado) en función a la distancia que se encuentre el objeto de 0.4 a 2.75 V, las características técnicas se adjuntan a continuación junto a una gráfica de salida:

Módulo comercial de *SHARP* (**GP2Y0A02YK**) [Datasheet adjunto en “Anexos Datasheets”] cuyo rango de medida está entre veinte y ciento cincuenta centímetros.

Caracteristicas más importantes:

- ✓ Salida analógica.
- ✓ Rango: 20 a 150 cm.
- ✓ Tiempo de respuesta: 39ms.
- ✓ Retraso de tiempo al conectar: 44ms.
- ✓ Consumo medio: 33mA.
- ✓ Alimentación: 4.5 a 5.5 V.

Basados en la emisión infrarroja de un haz de luz direccional que es captado por un fototransistor tras rebotar en el objeto. Salida analógica cuyos valores se detallan en la **Figura 4.6.**

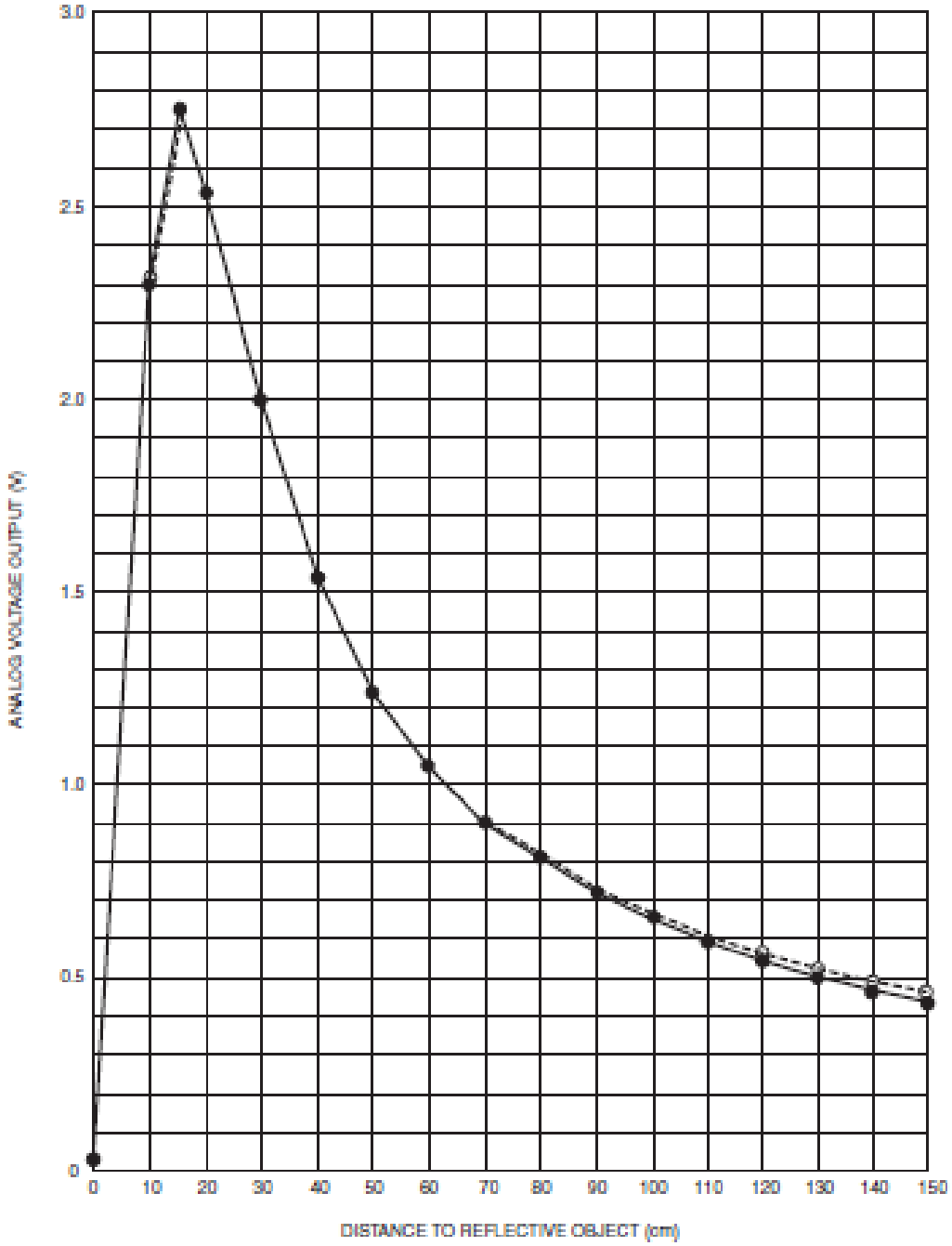


Figura 4.6: *Distancia frente a tensión de salida*

El tiempo de respuesta es de 39ms, tiempo inferior a los 50ms máximos que habíamos prefijado, además la salida es analógica y con un amplio rango de valores de media, con lo que se cumplen sobradamente con las características especificadas.

CAPÍTULO 5

DISEÑO

5.1 DISEÑO DEL HARDWARE.

Al tratarse el robot de una especie de ser hexápodo, estará dotado de seis patas (con dos grados de libertad: lo que se considerará la articulación del hombro y del codo, más adelante se tratarán dichos grados de libertad), un cuerpo central (donde irá alojado el controlador de los Servo Motores y las baterías) y la cabeza (donde encontraremos los sensores y balizas de estado).

Se deberá de conseguir una estructura ligera pero resistente, que permita una mecánica fluida del movimiento de las patas, no deben de entorpecerse entre sí, y al estar inspirado en la naturaleza se podrán solventar obstáculos y dificultades que se presenten, con mayor facilidad.

5.1.1 Servos.

Esencialmente el Hardware está formado por el cuerpo y la estructura en si del robot, con lo que lo primero a describir serán los servos, actuadores que permiten la movilidad del robot.

Un servomotor (también llamado servo) es un dispositivo similar a un motor de corriente continua, que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición. En resumen, es un motor eléctrico con la capacidad de ser controlado en posición.

Los servos se utilizan frecuentemente en sistemas de radio control y en robótica. Es posible modificar un servo para obtener un motor de corriente continua que, si bien ya no tiene la capacidad de controlar la posición en la que se encuentra su eje, conserva la fuerza, velocidad y baja inercia propia de estos dispositivos.

El servomotor está formado por un motor de continua, una caja reductora de engranajes y un circuito de control, que regula la posición (se basa en un potenciómetro conectado al eje de salida para captar la posición en la que se encuentra el mismo y en función a eso actuar sobre el motor).

La corriente que requiere depende del tamaño del servo y depende también principalmente del par que aplica y que puede exceder de un amperio si el servo está enclavado, pero que no es muy alto si se está moviendo libremente.

Como se ha explicado anteriormente su funcionamiento está basado en la modulación por ancho de pulso (PWM) para controlar la posición del eje de salida, controlando el movimiento del motor de corriente continua. La mayoría trabaja a una frecuencia de unos 50 Hz, de este modo las señales PWM tendrán un periodo de 20 milisegundos. Controlando el tiempo que esta activa la señal controlamos la posición del servo, lo que es lo mismo, la electrónica dentro del servomotor responderá al ancho de la señal modulada dando la orden de girar el motor en un sentido o en el otro.

En este caso caso alejándonos de los estándares, se ha hecho una tabla donde se muestra el tamaño del ancho del pulso positivo correspondiente a las diferentes y más importantes posiciones que puede adquirir el eje del servo de nuestro robot. Se ha realizado acoplando un círculo graduado al eje para observar las posiciones que adquiere el mismo al enviarle los diferentes anchos de pulso (es un método totalmente experimental, para estos servos en concreto, lo que quiere decir que para ser más exactos en el empleo de otros servos, habría que realizar esta calibración de nuevo con los nuevos servos)

Se pasa a explicar detalladamente los experimentos realizados para obtener los tiempos de la modulación de ancho de pulso:

En primer lugar se obtuvo un circulo graduado (en divisiones de 10 grados), es decir dividido en ángulos; vienen representados los 360 grados que corresponden a una vuelta completa, no obstante por construcción interna, un servo no puede girar su eje más de 270º y en el caso del robot en particular dicho movimiento quedará restringido tan solo a 180º (por cuestiones de la movilidad de las patas)

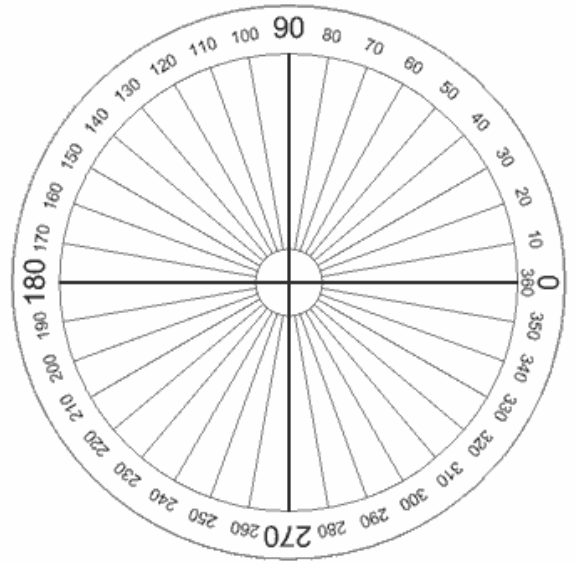


Figura 5.1: *Circulo graduado empleado para las pruebas*

Se dispuso el círculo graduado sobre el servo de pruebas, al que se le aplicarían los tiempos para calcular exactamente los grados correspondientes de la modulación por ancho de pulso (por necesidades de construcción hubo que acoplarle al círculo un cartón para aumentar su rigidez y aguantar la batería de pruebas que se iban a realizar sobre él, **Figura 5.2**).

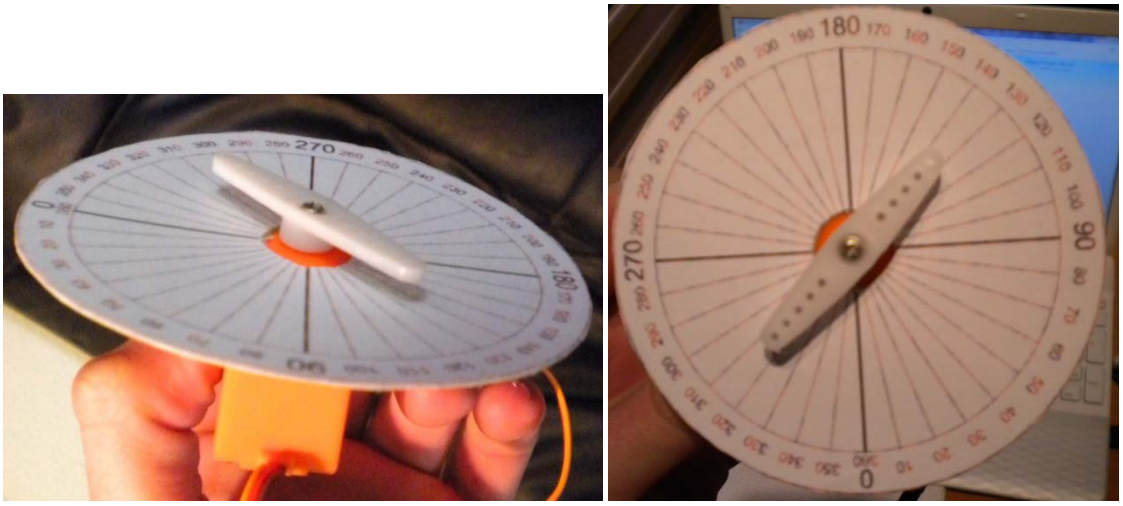


Figura 5.2: *Montura real del círculo graduado usado para cálculo de los tiempos de ancho de pulso positivo*

Como se mencionó, cada periodo de la *PWM* dura 20 ms, en estos 20ms debemos de introducir un pulso positivo (1 lógico) y un pulso negativo, en este caso a tierra (0 lógico), la configuración de estos tiempos será lo que posicione al eje; partiendo del clásico 1.5 ms alto y 18.5 ms bajo (configuración estándar de los tiempos para 90º [Wikipedia 05]) se obtienen unos grados, reflejados en el circulo graduado y con la ayuda de la corona (brazo del servo) como indicador (**Figura 5.2**).

De este modo modificando el tiempo que permanece la señal a 1 lógico (en este caso 5 voltios) y aplicando de tiempo de 0 lógico (en este caso 0 voltios) 20ms menos el tiempo de 1 lógico que se haya aplicado, se irán anotando los tiempos y los grados obtenidos en una tabla (**Figura 5.3**), que será luego de ayuda a la hora de crear las funciones de andar del hexápodo.

Se ha de destacar, que como se trata de un servo analógico lineal, se ha querido demostrar que la teoría no siempre se acompasa con la práctica y ello queda reflejado en la columna de Diferencia (**Figura 5.3**) donde se muestran los microsegundos de diferencia entre unos saltos de grados y otros (siempre a saltos de 10 en 10 grados) donde lo normal es que la diferencia siempre fuese la misma (en este caso 120 µs) y no siempre, como vemos, es así.

Grados	1 Lógico	0 Lógico	Diferencia
0º	550 µs	19.450 µs	0 µs
10º	600 µs	19.400 µs	150 µs
20º	650 µs	19.350 µs	50 µs
30º	770 µs	19.230 µs	120 µs
40º	880 µs	19.120 µs	110 µs
50º	1000 µs	19.000 µs	120 µs
60º	1120 µs	18.880 µs	120 µs
70º	1240 µs	18.760 µs	120 µs
80º	1340 µs	18.660 µs	100 µs
90º	1470 µs	18.530 µs	130 µs
100º	1590 µs	18.410 µs	120 µs
110º	1710 µs	18.290 µs	120 µs
120º	1830 µs	18.170 µs	120 µs
130º	1950 µs	18.050 µs	120 µs
140º	2070 µs	17.930 µs	120 µs
150º	2190 µs	17.810 µs	120 µs
160º	2300 µs	17.700 µs	110 µs
170º	2400 µs	17.600 µs	100 µs
180º	2510 µs	17.490 µs	110 µs

Figura 5.3: *Tiempos y grados para la codificación por ancho de pulsos*

5.1.2. Patas.

Para la forma de las patas se estuvieron diseñando varias formas, hasta que se dio con aquella que estéticamente armonizaba al robot en tamaño y forma. Esencialmente se trata de dos círculos unidos por tangentes exteriores. Un círculo pequeño representa el apoyo de la pata con el suelo y uno más grande para hacer ver la articulación de la pata (codo). Se estuvo pensando en el adaptarle unos calzos a forma de gomas en las bases de las patas para que el agarre con suelo fuese mayor, no obstante finalmente se descartó esta posibilidad por los siguientes motivos:

- La salida de la luz de las patas perdía uniformidad y creaba sombras.

- Perdíamos el característico ruido al caminar del hexápodo (esto podría ser una ventaja, no obstante para el modelo de demostración queda más espectacular que realice ese sonido característico al chocar el metraquilato contra el suelo).

- Problemas a la hora de adherir las gomas al metraquilato (dificultad de encontrar un adhesivo que no degradase la goma y a la vez pegase en el metraquilato)

A continuación se adjuntan algunos de los diseños hasta llegar a la pata final:

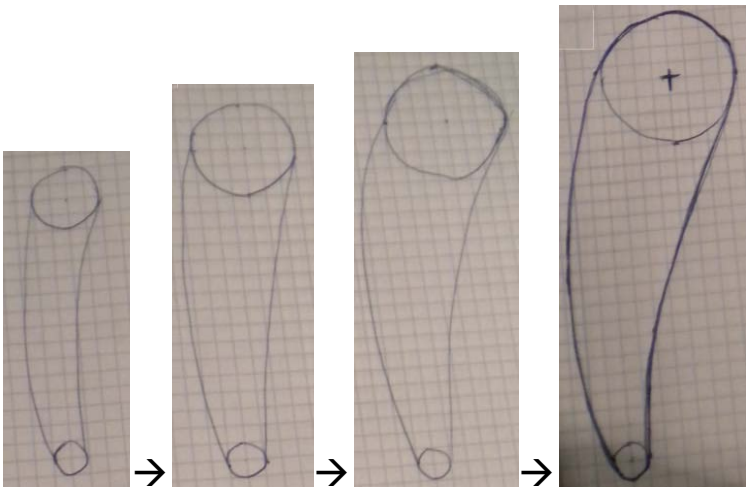


Figura 5.4: *Pata final: fue incrementando en tamaño y mejorando en forma.*

La estructura debe de ser similar a la de un hexápodo, no obstante todas las soluciones que se dieron eran complicadas y poco factibles ya que se le debía de dar curvatura al plástico y hacer formas difíciles sin la ayuda de las máquinas apropiadas (evidentemente no se disponen de dichas máquinas) por lo que finalmente y tras una tormenta de ideas (brain stroming) se optó por un novedoso tipo de estructura, una carcasa que no lo es como tal, ya que no cubre por completo al robot, simplemente le da forma, la figura de un hexápodo.

Del mismo modo que la marquetería trabaja sobre luterma (madera blanda) para darle forma y si es en 3D para luego ensamblar las piezas, así se hizo con el metraquilato, recortando las piezas que conformarían el armazón abierto del robot. El basar la estructura en el uso de "costillas" reporta numerosas ventajas antes imposibles sin el uso de esta técnica:

- ✓ La protección de los circuitos interiores es completa, no permitiendo que reciban daño alguno por algún elemento exterior al robot (siempre y cuando el objeto sea de mayor tamaño que el tamiz que deja la estructura).

- ✓ Considerable disminución del peso de la estructura con el considerable incremento de la vida de las baterías y menor esfuerzo llevado a cabo por los servos.

- ✓ Nos recrea un robot vaporoso pero resistente; idealiza un robot "atlético" y ágil dando la sensación junto a las luces LED de encontrarnos ante algo fuera de este planeta (es la idea que de vez en cuando damos a lo largo del texto, no solo queremos un robot funcional, sino que también nos cause impresión, que guste y sea vanguardista en cuanto a forma y efectos lumínicos).

5.1.3 Ojos.

Se han instalado unos ojos a modo de balizas de señalización, dichos LED muestran el estado del robot. Cuenta pues con unos LED bicolor a cada lado de la cabeza, los colores pueden ser rojo o verde. La finalidad de dichos ojos (aparte del mero hecho de conferirle a nuestro robot un aspecto más animal) es el de indicar hacia donde se está moviendo en cada momento.

Ambos LEDs serán verdes cuando el desplazamiento sea hacia adelante y cuando un LED se pusiera rojo estaría indicando que la luz se encuentra por el lado del ojo rojo o que el objeto que está esquivando se encuentra del lado contrario del ojo rojo, realizando el giro hacia la luz o esquivando el objeto para volver a desplazarse hacia adelante con ambos ojos verdes. Con esto se extrae información de lo que está procesando el microcontrolador y que funciones está ejecutando en cada momento, para que de este modo, se monitorice si los servos llevan a cabo la función para la que están encomendados.

El circuito de los LED es el siguiente: a la patilla central (cátodo común) se le acoplará una resistencia limitadora de corriente, de tal modo que cuando apliquemos una diferencia de potencial de unos 5v la corriente máxima que circulará por los LED sea de unos 7mA (queremos que el consumo sea el menor posible), con lo que aplicando la ley de Ohm y aproximando que la caída de tensión de cada diodo es de unos 2.4v, la resistencia será la siguiente: $V_{resistencia}=V_{CC}-V_{LED}$ → 5-2.4=2.6v. Como la ley de Ohm dice: V=R*I → R=V/I→Resistencia=2.6/0.007=371Ω que aproximando a un valor comercial, será 390Ω.

El resto del control de los ojos (y para emplear el menor número de puertos E/S del microcontrolador) se llevará a cabo con un **inversor disparador Scmitt 40106**, donde colocaremos la patilla del LED verde a la entrada del inversor y la patilla roja a la salida, ocupamos solo 2 inversores y dejamos el resto disponible para futuras aplicaciones.

La entrada del inversor junto con la patilla del LED verde, se conectarán a una línea de E/S del microcontrolador que controlará el encendido del LED bicolor, de la siguiente forma: cuando ejecuta el movimiento de andar mandará un 1 lógico, iluminando el LED verde y como el LED rojo tiene en su patilla 0v por la inversión de la puerta inversora NOT, no se iluminará; de modo contrario cuando el robot esté ejecutando un giro, la patilla de entrada del inversor se pondrá a 0 lógico, permaneciendo el LED verde apagado e iluminándose el rojo (por la inversión antes citada).

5.1.4 Sensores.

Los sensores empleados son LDR (Light Dependent Resistor = Resistencia dependiente de la luz) y medidores de distancia por infrarrojos.

LDRs: se tratan de unas resistencias cuyo valor óhmico varía en función de la cantidad e luz que incide sobre su película fotosensible (normalmente este valor disminuye cuanta más cantidad de luz recibe). Existen LDRs de muchos valores nominales, no obstante sin realizar cálculos se trabajará con unas de unos 100KΩ ya que esto creará un divisor resistivo por el que apenas circule intensidad y consuma la menor energía posible.

Como hemos indicado vamos a realizar un divisor resistivo, en la rama superior (con la alimentación) situaremos nuestra LDR y en la inferior una resistencia de 47 KΩ, la elección de dicho valor, la vamos a detallar a continuación:

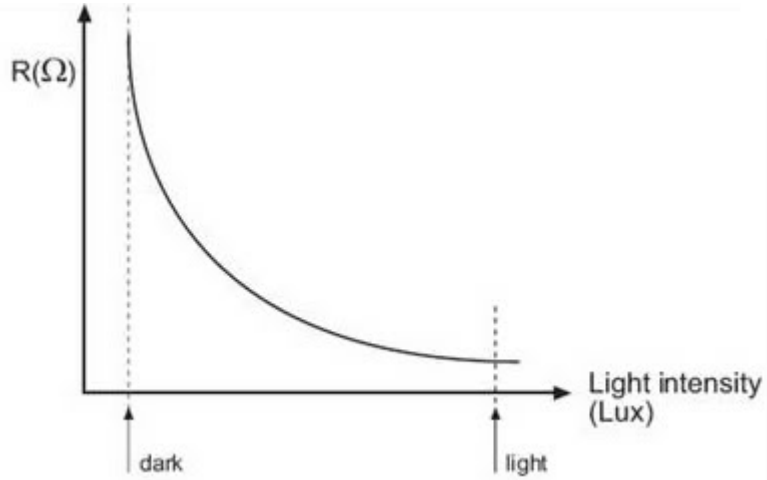


Figura 5.5: *Curva de respuesta de la LDR ante la luz*

La respuesta de la LDR frente a la luz es una respuesta no lineal, siendo esta curva prácticamente exponencial descendente como se muestra en la gráfica adjunta.

Al no ser lineal una de las resistencias del divisor, cuando esta varíe, su salida tampoco será lineal, siendo una exponencial creciente; no obstante existe una forma de linealizar dicho divisor y no es más que colocarle en la otra rama una resistencia equivalente a la resistencia que tiene nuestra LDR en su punto de trabajo. Suponiendo un punto de trabajo ni de mucha luz, ni de mucha oscuridad (en penumbra) el valor de la LDR rondará entorno a los 50KΩ, siendo este el valor elegido para nuestra resistencia.

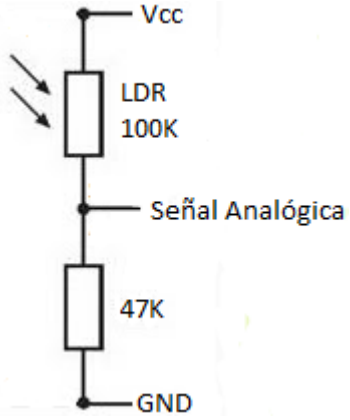


Figura 5.6: *Divisor resistivo*

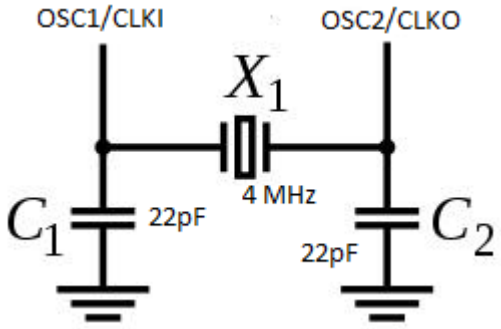
Dichos sensores se encuentran alojados en la cabeza, a modo de antenas, y uno en lo que podría considerarse la boca de nuestro hexápodo. Como hemos dicho se tratan de LDRs cuyo valor resistivo nos informará de la cantidad de luz en el ambiente, así como de su procedencia y de este modo llegar a ella lo más eficientemente posible (recordar que esa es la finalidad del robot). Las antenas están dispuestas de tal forma que consigamos triangular perfectamente la posición de la luz mediante un posterior procesado de las señales provenientes de estas; encontraríamos dos en lo alto de la cabeza y un poco retrasadas para captar la información procedente de los lados y de atrás y el sensor inferior enfocando hacia adelante captando la luz que procede de esa dirección.

Medidores de distancia infrarrojos: se tratan de los módulos comerciales GP2Y0A02YK que poseen tres pines, alimentación masa y señal con lo que no requerirá de circuito de acondicionamiento alguno. Su colocación deberá de ser tal que se detecten que se encuentren en el frente y a los lados, de este modo el robot caminará sin chocar con obstáculo alguno.

5.1.5 Placa de circuito principal.

Se pasa finalmente a diseñar lo que podría considerarse el cerebro del robot, o la placa de circuito impreso principal. En la placa se deben de introducir todos los componentes necesarios para el perfecto funcionamiento del robot y además otras utilidades que permitan la ampliación de las funciones en un futuro.

En primer lugar el microcontrolador PIC, la señal de reloj es externa, con lo que se deberá de prever un espacio en la placa para el cristal de cuarzo y sus condensadores (dichos condensadores cerámicos se conectan entre masa y las entradas de reloj para estabilizar la señal, si su valor es muy pequeño perdemos estabilidad y si es muy grande el arranque es más lento, habrá que llegar a un compromiso que aporte buena estabilidad y un bajo tiempo de arranque).



Se recomiendan condensadores de 15pF, no obstante se emplearon unos de 22pF para ganar en estabilidad ya que el tiempo de arranque no es crítico en este proyecto, se quiere que el ciclo de instrucción dure 1 µs con lo que el cristal de cuarzo será de 4 MHz (T=0.25 µs).

Figura 5.7: *Montura oscilador*

En la placa también se incluirá un inversor disparado Schmitt (40106) para el control de las luces LED de los ojos (de esta forma se emplean la mitad de puertos del microcontrolador necesarios para encender el color correspondiente a cada movimiento); la idea es que con un 1 lógico se encienda el LED verde y con un 0 lógico el LED rojo, esto se lleva a cabo como se especifica en la **Figura 5.8,** habrá que hacer esto con otro puerto más para el otro ojo; además quedan disponibles 4 inversores libres para cualquier otra aplicación futura que necesite de la inversión digital de alguna señal.

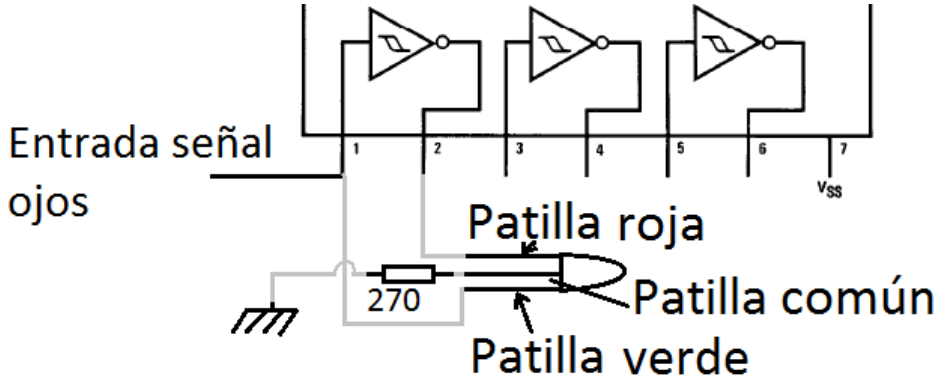


Figura 5.8: *Conexión LED bicolor a inversor*

Se añade a la placa de circuito impreso un condensador de gran capacidad (1000µF) para reducir el efecto en el microcontrolador de las posibles bajadas de tensión repentinas en las baterías por una alta actividad motriz de los servos.

Por último y no menos importante los pines de conexión que irán distribuidos por todas las Entradas/Salidas del PIC (muchos de ellos con alimentación y masa para los servos). Se contará con doce arrays de tres pines para las patillas de la B0 a la B5 y de la C0 a la C5 para su conexión con los conectores JR de los servos, dos arrays de siete pines para las entradas y salidas del inversor, donde además hay dos pines que representarán la masa y alimentación, dos arrays de 2 pines, para las patillas C6, C7, B6 y B7, por último un array de siete pines para las entras analógicas provenientes de los sensores y la entrada de *reset* del PIC. Todos estos pines proporcionan una fácil conexión/desconexión para la sustitución de los módulos, sensores, servos… que hacen que el robot tenga un *hardware* totalmente abierto y rápidamente modificable.

5.2 DISEÑO DEL SOFTWARE.

En este apartado se abarcará la programación del PIC 16F876A, debiendo de prestar gran atención para comprender el funcionamiento de los algoritmos empleados.

5.2.1 Mecánica de funcionamiento.

Para empezar, se explicará cómo funciona el robot y cómo ejecuta los movimientos de las patas para andar.

El movimiento del hexápodo está basado en el equilibrio estático, es decir, solo tres de sus seis patas estarán en el aire a la vez, permitiendo mantener el equilibrio al robot con las tres patas restantes.

Poseerá dos flancos de tres patas, por cada paso dos de las patas de un flanco y una del contrario se elevarán, avanzarán y se posaran, para volver a su posición inicial permitiendo al robot avanzar un paso; para el siguiente movimiento, se levantaran dos patas del flanco contrario al inicial y una del opuesto, realizando el mismo movimiento antes descrito; esto producirá otro paso. Encadenando dichos pasos el robot transportará en línea recta, de una forma más o menos eficiente, pudiendo superar terrenos no necesariamente uniformes, gracias a su configuración de hexápodo.

Cada pata tendrá dos grados de libertad, uno que le permitirá subir y bajar la pata (codo) y otro, uniendo la pata con el cuerpo (hombro), que le permitirá avanzar o retrasar la pata respecto del cuerpo.

Estas dos articulaciones convenientemente sincronizadas hacen que la pata realice un paso, como se muestra a continuación (imágenes de la izquierda corresponden a una vista en planta e imágenes de la derecha corresponden a una vista de alzado desde atrás del robot en un costado y hacia adelante:

1º Levanta la pata a la vez que la sitúa perpendicular al cuerpo.

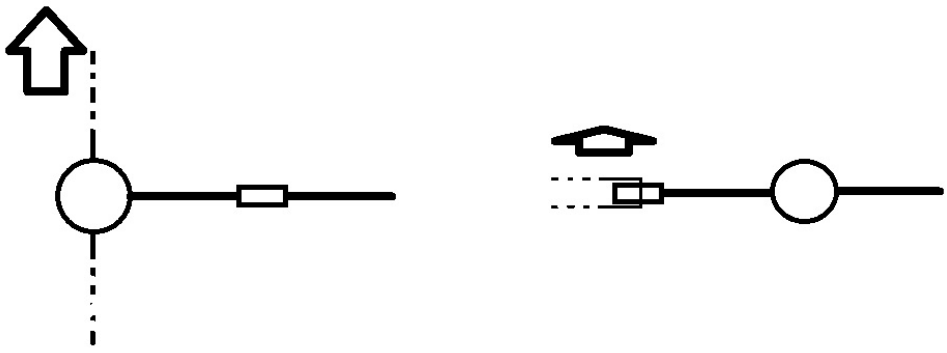


Figura 5.9: *paso 1*

2º Avanza la pata a la vez que baja.

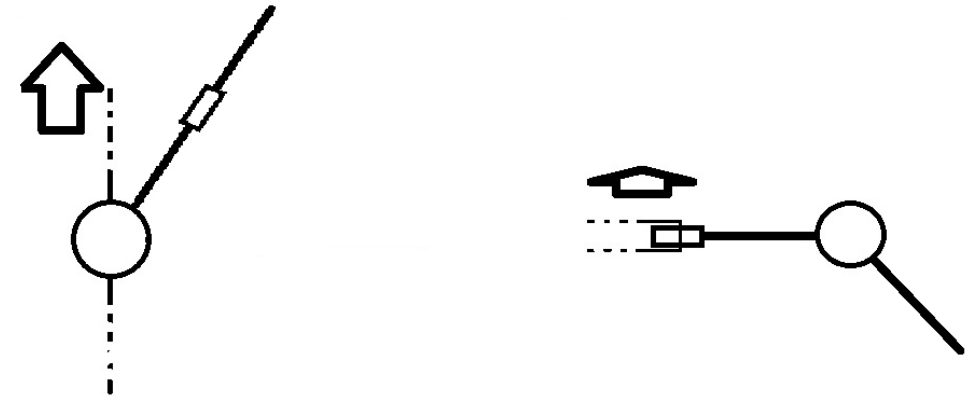


Figura 5.10: *paso 2*

3º Retrocede la pata hasta que se sitúa perpendicular al cuerpo.

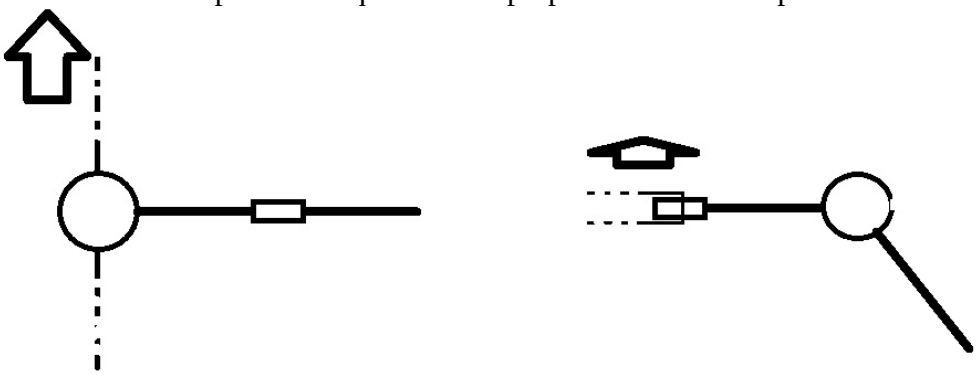


Figura 5.11: *paso 3*

4º Vuelve a retroceder la pata para permitir más movimiento.

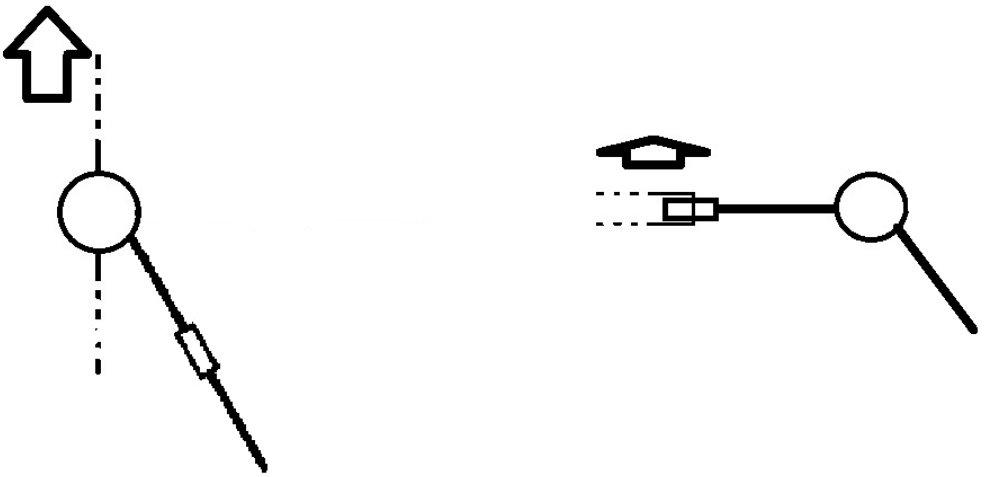


Figura 5.12: *paso 4*

En el 3º paso, las tres patas restantes que permiten el equilibrio estático, deberán de moverse como se especifica en el paso 1º, para permitir que el robot avance y no frenar su movimiento. La concatenación de dichos movimientos recrea la peculiar forma de andar de un animal hexápodo.

Dichos movimiento de "Andar" girar a la "Izquierda", "Izquierda tanque" y girar a la "Derecha", "Derecha tanque" se muestran con mayor exactitud en el *apartado 5.2.3. Funciones.*

5.2.2 Simultaneidad.

Quizá este apartado se trate del más importante de todo el proyecto, no por ello va a ser largo y complejo, sino que se tratará de explicar cómo se consigue una ejecución simultanea de los movimientos de los servos de la forma más rápida y clara posible.

El truco de conseguir una ejecución de los servos de forma multi-hilo reside en la creación del propio código y en el conocimiento del funcionamiento interno de los servos. Por tanto queda claro que no se van a emplear funciones o subrutinas ya creadas en las librearías del compilador CCS, sino que se van a crear unas propias, desechando los puertos del microcontrolador destinados al PWM.

Los servos funcionan mediante modulación por ancho de pulso, es decir, variando el tiempo de subida de la señal (1 lógico) controlamos el giro que realiza dicho servo. En el apartado *5.1.1. Servos* se encuentr la tabla que relaciona dicho tiempo de 1 lógico con los grados que queramos en el eje de salida del servo, esto será de vital importancia a la hora de construir las señales de salida.

Lo que se hace es iniciar el movimiento de los servos a la vez con un 1 lógico en sus respectivas líneas de entrada, e ir bajándolas a 0 lógico conforme pase su tiempo (para los grados que se quieran en el servo) y así hasta poner a 0 todas las líneas, el tiempo de subida a 1 lógico y de bajada a 0 lógico total es de 20 milisegundos.

Se muestra un gráfico de ejemplo donde se entenderá mejor lo que se hace y como se lleva a cabo el control de los servos:

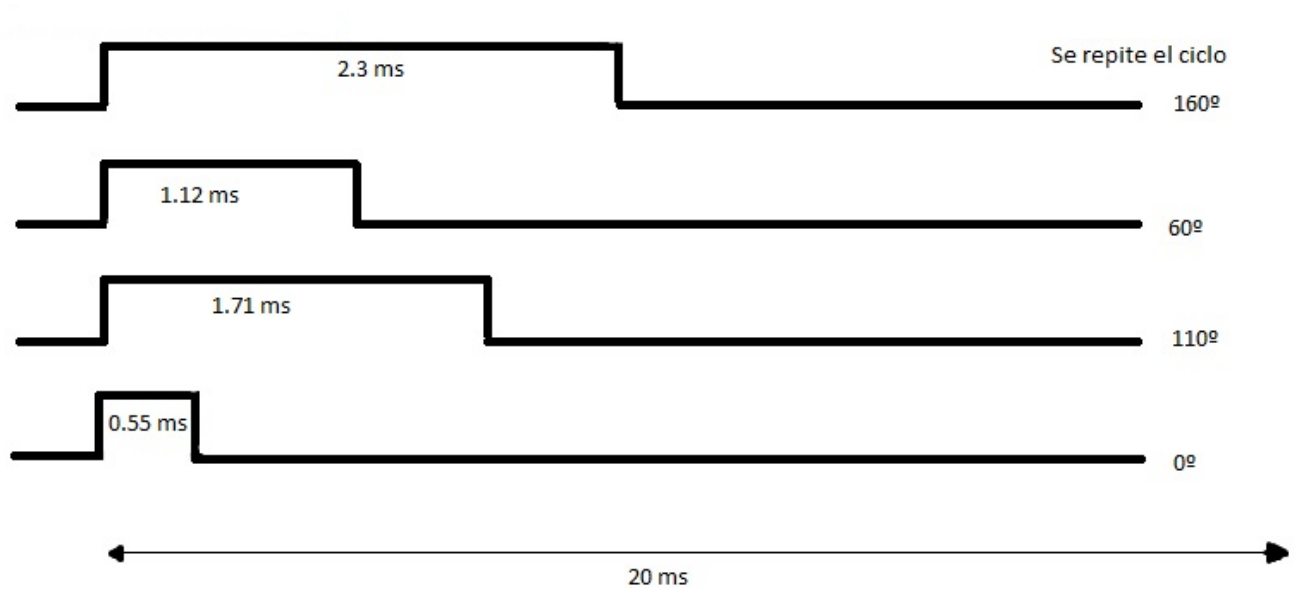


Figura 5.13: *Ejemplo señales PWM*

En este ejemplo se controlan 4 servos y se quiere que cada uno alcance una posición diferente, pero que se muevan a la vez; como un programa se ejecuta de forma secuencial, si se mueve uno a uno cada servo cuando se haya movido el primero y se esté moviendo el siguiente, como ya no recibe PWM el primer servo, su eje se puede girar (por el propio peso del robot o la resistencia de la aplicación que estemos realizando), con lo que solo se podrá manejar un servo cada vez dejando el resto de servos libres.

De la forma que se plantea en el gráfico se pueden controlar las líneas de salida que se quieran (siempre y cuando se dispongan de los suficientes puertos de E/S). Se pone a 1 lógico todas las salidas, y dependiendo de la posición que se quiera alcanzar en cada servo, se irán poniendo a 0 lógico su correspondiente señal, hasta que todas estén a 0; una vez están a 0 se espera el tiempo necesario para que se completen los 20 ms del ciclo completo. Este ciclo se repetirá las veces que se quiera para conseguir tener los servos en las posiciones deseadas el tiempo necesario (por ejemplo: como cada ciclo dura 20 ms, si queremos que todos los servos permanezcan en esa posición 2 segundos, tendremos que repetir el ciclo→2/0.02=100 veces, para luego pasar a la posición siguiente).

De este modo controlando el tiempo que la señal permanece a 1 lógico se logra que los servos se posicionen donde se desee el tiempo que se quiera.

Con esta forma de programar software se rompen con muchas de las ideas con las que se programaban servos por multiplexado u otros métodos, así se hace de una forma rápida y sencilla, solo hay que saber qué posición deben de alcanzar los servos y por código aplicar los tiempos entre los 1s y 0s lógicos necesarios (dichos tiempos quedan reflejados en la tabla del capítulo *5.1.1. Servos*.

5.2.3 Funciones.

A continuación se tratan de lleno las funciones o subrutinas de los movimientos del robot, incluyendo su respectivo código en C con tablas y dibujos que indican la posición de las patas (sus grados) para que el robot se desplace. Cada función cuenta con 4 pasos, en cada paso se espera un tiempo para permitir que los servos se posicionen (se repiten los ciclos hasta que están todos en su posición) tras ese paso, se reposicionarán los servos con el siguiente, y así sucesivamente hasta llegar al final; si se quisiera repetir este modo de andar constantemente, simplemente habría que volver al paso inicial y seguir ejecutando.

Se ha de decir que se probó un paso (detallado en un video de la sección multimedia) el cual incrementaba la complejidad respecto a los finalmente implementados y que por su ineficiencia se descartó. Se trataba de un paso que levantaba secuencialmente las patas una a una, recreando ciertos modos de caminar de insectos hexápodos, pero que finalmente se suprimió por no obtener los resultados deseados.

En el siguiente dibujo se describe la estructura hexápoda básica con los conectores para las distintas patillas del microcontrolador y los puertos correspondientes.

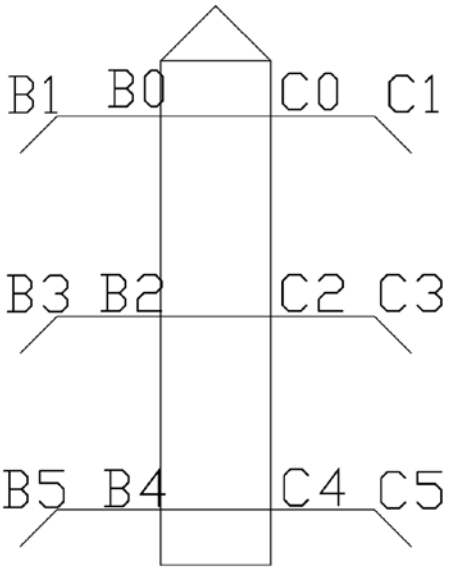


Figura 5.14: *Detalle puerto microcontrolador-servo*

5.2.3.1 Posición de Pie

Pasos de la función Posición de Pie:

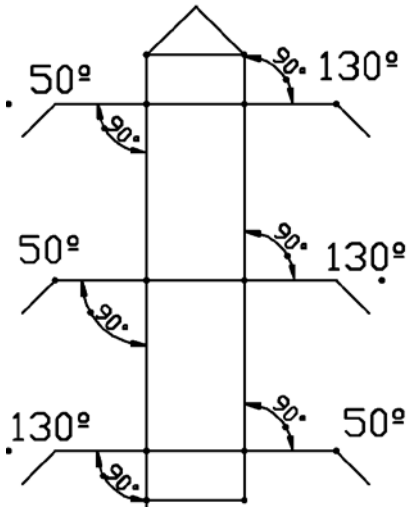


Tabla con los grados correspondientes a cada señal con su respectivo paso:

Posición de Pie												
	B5	B4	B3	B2	B1	B0	C5	C4	C3	C2	C1	C0
1	130	90	50	90	50	90	50	90	130	90	130	90

Código de la función:

```
for(i=0;i<=cuenta;i++){
        output_b(0b11111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b11110101); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(470);
        output_b(0b11100000);// Se posicona a 90º
        output_c(0b00001010);// Se posicona a 90º
        delay_us(480);
        output_c(0b00000000);   //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);        }
```

5.2.3.2 Andar

El robot se transporta en línea recta.

Pasos de la función Andar:

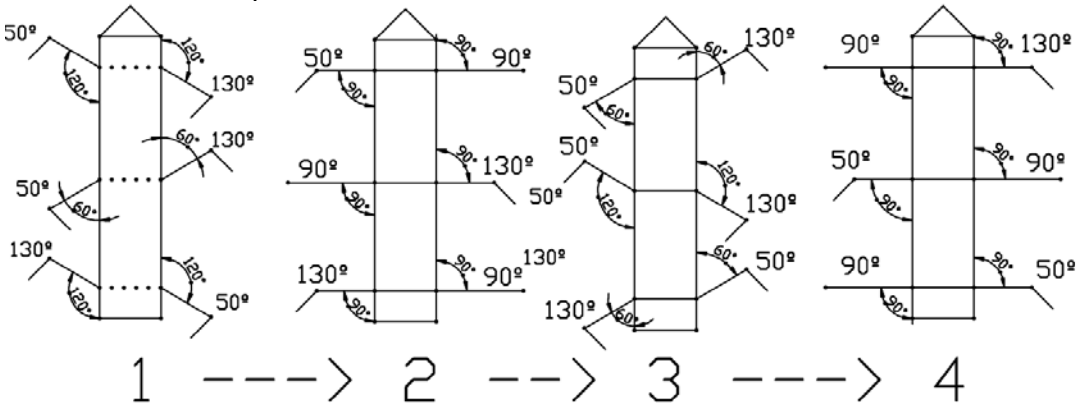


Tabla con los grados correspondientes a cada señal con su respectivo paso:

Secuencia Andar												
	B5	B4	B3	B2	B1	B0	C5	C4	C3	C2	C1	C0
1	130	120	50	60	50	120	50	120	130	60	130	120
2	130	90	90	90	50	90	90	90	130	90	90	90
3	130	60	50	120	50	60	50	60	130	120	130	60
4	90	90	50	90	90	90	50	90	90	90	130	90

Código de la función:

```
for(i=0;i<=t1;i++){
        output_b(0b11111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b11110101); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(120);
        output_low(PIN_B2);  //Se posiciona a 60º
        output_low(PIN_C2);  //Se posiciona a 60º
        delay_us(710);
        output_b(0b11100000);   //Se posiciona a 120º
        output_c(0b00001010);   //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);   //Se posiciona a 130º
        output_low(PIN_B5);  //Se posiciona a 130º
        delay_us(18050);        }
```

```
for(i=0;i<=t1;i++){
        output_b(0b11111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b11111101); //Se posiciona a 50º
        delay_us(470);
        output_b(0b11100000);  //Se posiciona a 90º
        output_c(0b00001000);  //Se posiciona a 90º
        delay_us(480);
        output_low(PIN_C3);   //Se posiciona a 130º
        output_low(pin_B5);   //Se posiciona a 130º
        delay_us(18050);       }

for(i=0;i<=t1;i++){
        output_b(0b11111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b11110101); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(120);
        output_b(0b11100100);  //Se posiciona a 60º
        output_c(0b00001110);  //Se posiciona a 60º
        delay_us(710);
        output_low(PIN_B2);   //Se posiciona a 120º
        output_low(PIN_C2);   //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);   //Se posiciona a 130º
        output_low(PIN_B5);  //Se posiciona a 130º
        delay_us(18050);       }

for(i=0;i<=t1;i++){
        output_b(0b11111111);
        output_c(0b00111111);
        delay_us(1000);
        output_low(PIN_B3); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(470);
        output_b(0b11000000);  //Se posiciona a 90º
        output_c(0b00000010);  //Se posiciona a 90º
        delay_us(480);
        output_c(0b00000000);   //Se posiciona a 130º
        delay_us(18050);       }
```

5.2.3.3 Izquierda

El robot anda hacia adelante mientras va girando a la izquierda.

Pasos de la función Izquierda:

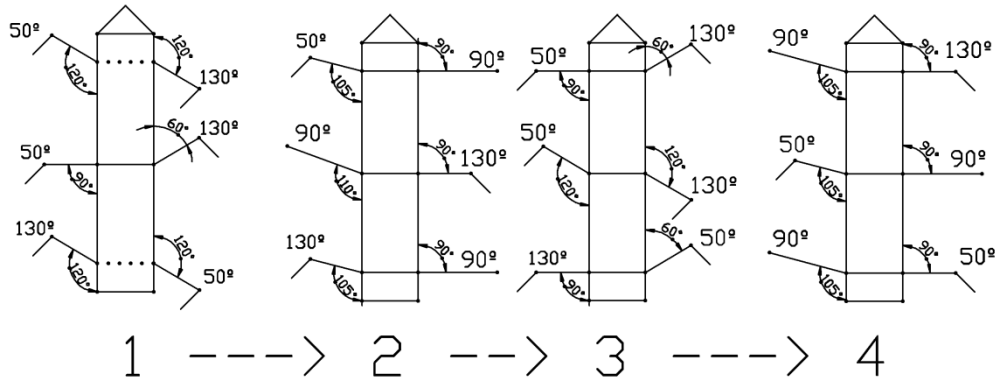


Tabla con los grados correspondientes a cada señal con su respectivo paso:

Secuencia Izquierda												
	B5	B4	B3	B2	B1	B0	C5	C4	C3	C2	C1	C0
1	130	120	50	90	50	120	50	120	130	60	130	120
2	130	105	90	105	50	105	90	90	130	90	90	90
3	130	90	50	120	50	90	50	60	130	120	130	60
4	90	105	50	105	90	105	50	90	90	90	130	90

Código de la función:

```
for(i=0;i<=t2;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b10110101);       //Se posiciona a 50º
        output_low(PIN_C5);
        delay_us(120);
        output_low(PIN_C2);         //Se posiciona a 60º
        delay_us(350);
        output_low(PIN_B2);         //Se posicona a 90º
        delay_us(360);
        output_b(0b10100000);   //Se posiciona a 120º
        output_c(0b00001010);   //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);   //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);        }
```

```
for(i=0;i<=t2;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b10111101); //Se posiciona a 50º
        delay_us(470);
        output_low(PIN_B3);  //Se posiciona a 90º
        output_c(0b00001000);  //Se posiciona a 90º
        delay_us(180);
        output_b(0b10100000);   //Se posiciona a 105º
        delay_us(300);
        output_low(PIN_C3);  //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);       }

for(i=0;i<=t2;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b10110101); //Se posiciona a 50º
        output_low(PIN_C5);
        delay_us(120);
        output_c(0b00001110);  //Se posiciona a 60º
        delay_us(350);
        output_b(0b10100100);   //Se posiciona a 90º
        delay_us(360);
        output_low(PIN_B2);   //Se posiciona a 120º
        output_low(PIN_C2);   //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);   //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);       }

for(i=0;i<=t2;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_low(PIN_B3); //Se posiciona a 50º
        output_low(PIN_C5);
        delay_us(470);
        output_b(0b10010101);  //Se posiciona a 90º
        output_c(0b00000010);  //Se posiciona a 90º
        delay_us(180);
        output_b(0b10000000);   //Se posiciona a 105º
        delay_us(300);
        output_c(0b00000000);   //Se posiciona a 130º
        delay_us(18050);       }
```

5.2.3.4 Izquierda Tanque

El robot gira sobre su eje hacia la izquierda.

Pasos de la función Izquierda Tanque:

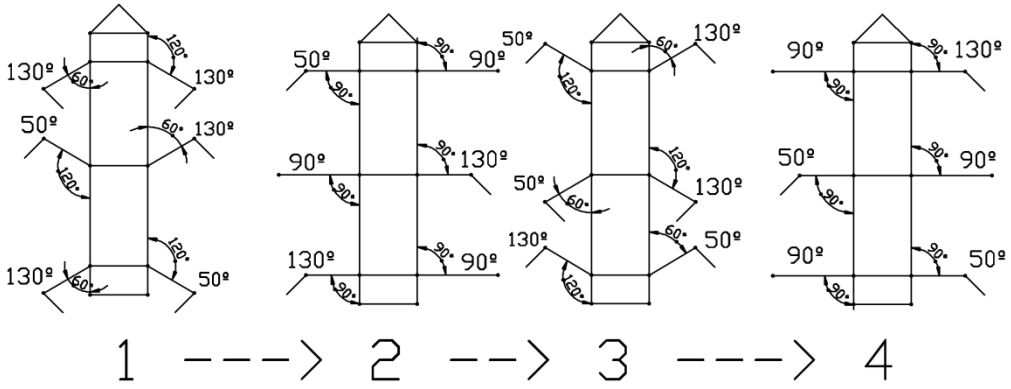


Tabla con los grados correspondientes a cada señal con su respectivo paso:

Secuencia Izquierda Tanque												
	B5	B4	B3	B2	B1	B0	C5	C4	C3	C2	C1	C0
1	130	60	50	120	50	60	50	120	130	60	130	120
2	130	90	90	90	50	90	90	90	130	90	90	90
3	130	120	50	60	50	120	50	60	130	120	130	60
4	90	90	50	90	90	90	50	90	90	90	130	90

Código de la función:

```
for(i=0;i<=t4;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b00110101); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(120);
        output_b(0b10100100);  //Se posiciona a 60º
        output_low(PIN_C2);  //Se posiciona a 60º
        delay_us(710);
        output_b(0b00100000);   //Se posiciona a 120º
        output_c(0b00001010);   //Se posiciona a 120º
        delay_us(120);
        output_low(PIN_B5);   //Se posiciona a 130º
        output_c(0b00000000);  //Se posiciona a 130º
        delay_us(18050);       }
```

```
for(i=0;i<=t4;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_low(PIN_B1); //Se posiciona a 50º
        delay_us(470);
        output_b(0b00100000);  //Se posiciona a 90º
        output_c(0b00001000);  //Se posiciona a 90º
        delay_us(480);
        output_low(PIN_B5);   //Se posiciona a 130º
        output_low(PIN_C3);   //Se posiciona a 130º
        delay_us(18050);       }

for(i=0;i<=t4;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b00110101); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(120);
        output_low(PIN_B2);  //Se posiciona a 60º
        output_c(0b00001110);  //Se posiciona a 60º
        delay_us(710);
        output_b(0b10100000);   //Se posiciona a 120º
        output_low(PIN_C2);   //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);   //Se posiciona a 130º
        output_low(PIN_B5);     //Se posiciona a 130º
        delay_us(18050);       }

for(i=0;i<=t4;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_low(PIN_B3); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(470);
        output_b(0b00000000);  //Se posiciona a 90º
        output_c(0b00000010);  //Se posiciona a 90º
        delay_us(480);
        output_low(PIN_C1);   //Se posiciona a 130º
        delay_us(18050);       }
```

5.2.3.5 Derecha

El robot anda hacia adelante mientras va girando a la derecha.

Pasos de la función Derecha:

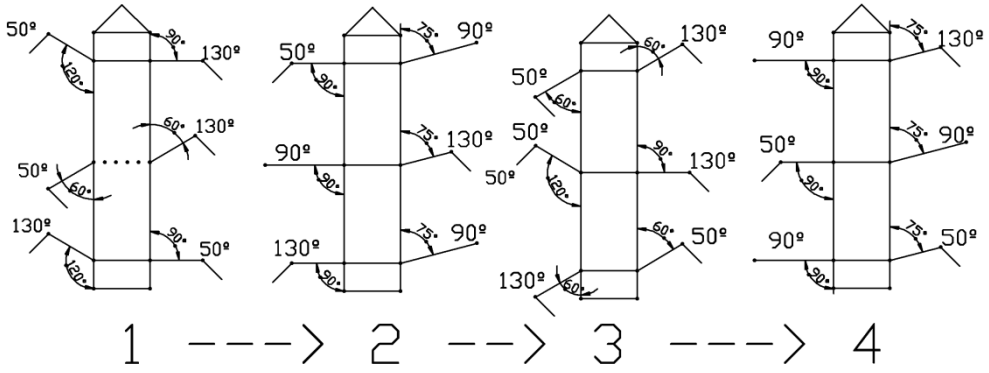


Tabla con los grados correspondientes a cada señal con su respectivo paso:

Secuencia Derecha												
	B5	B4	B3	B2	B1	B0	C5	C4	C3	C2	C1	C0
1	130	120	50	60	50	120	50	90	130	60	130	90
2	130	90	90	90	50	90	90	75	130	75	90	75
3	130	60	50	120	50	60	50	60	130	90	130	60
4	90	90	50	90	90	90	50	75	90	75	130	75

Código de la función:

```
for(i=0;i<=t3;i++){
        output_b(0b01111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b01110101); //Se posiciona a 50º
        output_low(PIN_C5);
        delay_us(120);
        output_low(PIN_C2);  //Se posiciona a 60º
        output_low(PIN_B2);  //Se posiciona a 60º
        delay_us(350);
        output_c(0b00001010);        //Se posicona a 90º
        delay_us(360);
        output_b(0b01100000);   //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);   //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);        }
```

```
for(i=0;i<=t3;i++){
        output_b(0b01111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b01111101); //Se posiciona a 50º
        delay_us(290);
        output_c(0b00101010);  //Se posiciona a 75º
        delay_us(180);
        output_b(0b01100000);  //Se posiciona a 90º
        output_c(0b00001000);  //Se posiciona a 90º
        delay_us(480);
        output_low(PIN_C3);  //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);       }

for(i=0;i<=t3;i++){
        output_b(0b01111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b01110101); //Se posiciona a 50º
        output_low(PIN_C5);
        delay_us(120);
        output_b(0b01100100);  //Se posiciona a 60º
        output_c(0b00001110);  //Se posiciona a 60º
        delay_us(350);
        output_low(PIN_C2);   //Se posiciona a 90º
        delay_us(360);
        output_b(0b01100000);   //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);   //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);       }

for(i=0;i<=t3;i++){
        output_b(0b01111111);
        output_c(0b00111111);
        delay_us(1000);
        output_low(PIN_B3); //Se posiciona a 50º
        output_low(pin_c5);
        delay_us(290);
        output_c(0b00001010);   //Se posiciona a 75º
        delay_us(180);
        output_b(0b01000000);  //Se posiciona a 90º
        output_low(PIN_C3);  //Se posiciona a 90º
        delay_us(480);
        output_c(0b00000000);   //Se posiciona a 130º
        delay_us(18050);       }
```

5.2.3.6 Derecha Tanque

El robot gira sobre su eje hacia la derecha.

Pasos de la función Derecha Tanque:

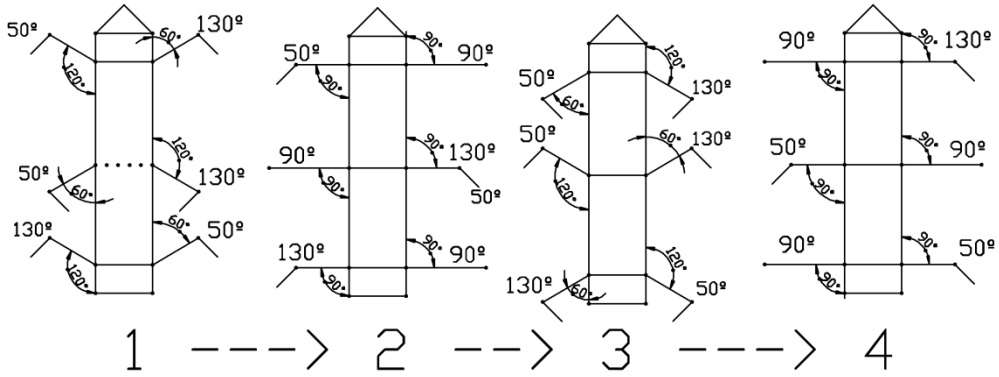


Tabla con los grados correspondientes a cada señal con su respectivo paso:

Secuencia Derecha Tanque												
	B5	B4	B3	B2	B1	B0	C5	C4	C3	C2	C1	C0
1	130	120	50	60	50	120	50	60	130	120	130	60
2	130	90	90	90	50	90	90	90	130	90	90	90
3	130	60	50	120	50	60	50	120	130	60	130	120
4	90	90	50	90	90	90	50	90	90	90	130	90

Código de la función:

```
for(i=0;i<=t5;i++){
        output_b(0b01111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b00110101); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(120);
        output_low(PIN_B2);  //Se posiciona a 60º
        output_c(0b00001110);  //Se posiciona a 60º
        delay_us(710);
        output_b(0b01100000);   //Se posiciona a 120º
        output_low(PIN_C2);   //Se posiciona a 120º
        delay_us(120);
        output_low(PIN_B5);   //Se posiciona a 130º
        output_c(0b00000000);  //Se posiciona a 130º
        delay_us(18050);       }
```

```
for(i=0;i<=t5;i++){
        output_b(0b01111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b00111101); //Se posiciona a 50º
        delay_us(470);
        output_b(0b11100000);  //Se posiciona a 90º
        output_c(0b00001000);  //Se posiciona a 90º
        delay_us(480);
        output_low(pin_B5);   //Se posiciona a 130º
        output_low(PIN_C3);   //Se posiciona a 130º
        delay_us(18050);       }

for(i=0;i<=t5;i++){
        output_b(0b01111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b00110101); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(120);
        output_b(0b01100100);  //Se posiciona a 60º
        output_low(PIN_C2);  //Se posiciona a 60º
        delay_us(710);
        output_low(PIN_B2);   //Se posiciona a 120º
        output_c(0b00001010);   //Se posiciona a 120º
        delay_us(120);
        output_low(PIN_B5);     //Se posiciona a 130º
        output_c(0b00000000);   //Se posiciona a 130º
        delay_us(18050);       }

for(i=0;i<=t5;i++){
        output_b(0b01111111);
        output_c(0b00111111);
        delay_us(1000);
        output_low(PIN_B3); //Se posiciona a 50º
        output_low(PIN_C5);  //Se posiciona a 50º
        delay_us(470);
        output_b(0b00000000);  //Se posiciona a 90º
        output_c(0b00000010);  //Se posiciona a 90º
        delay_us(480);
        output_low(PIN_C1);   //Se posiciona a 130º
        delay_us(18050);       }
```

5.2.4 Convertidor analógico digital.

Otra parte fundamental del software es el transformar las señales analógicas provenientes de los sensores a datos binarios para su correcto procesado por el programa que se incluye en el PIC. Para ello se emplean los cinco convertidores A/D que tiene el microcontrolador.

Mediante software se especifica los bits que queramos que tenga la señal binaria convertida, siempre y cuando no pasemos del rango máximo que para el 16F876A son de 10 bits; como se quiere obtener la máxima información proveniente del exterior, se configura el convertidor a 10 bits, obteniendo valores comprendidos entre 0 y 1023 (2^{10}) en total 1024 valores, más que de sobra para marcar los límites y que el software los procese y actué en consecuencia llevando alrobot al lugar con más luz.

Se emplean unas funciones que vienen definidas en el compilador CCS para el mejor manejo de los módulos analógicos digitales, como son:

- setup_adc_ports(): se establecen los puertos que vamos a usar como analógicos.
- setup_adc(): Indica si activamos o no el convertidor, y si así es se especifica el reloj usado para el muestreo del convertidor A/D.
- set_adc_channel(x): se establece el canal (x) del que se quiera realizar la conversión.
- read_adc(): se lleva a cabo la lectura del puerto correspondiente (previamente se ha debido de esperar un tiempo mínimo de 10 µs desde que se establece el puerto a leer y la obtención del valor.

El uso de estas funciones se explicará más adelante en el sub-apartado de implementación del *software* donde se mostrará el programa al completo y como se tratan las señales analógicas para comandar los servos y que nos lleven a la fuente lumínica lo más rápidamente posible.

CAPÍTULO 6

IMPLEMENTACIÓN

Con todos los diseños planteados, solo queda llevar a cabo su concreción física en el caso del *hardware* y su concreción a nivel de 1s y 0s en el caso del *software*; quizá durante este proceso los diseños sufran alguna modificación, ocasionada por la mejora de algún aspecto que se ha pasado por alto durante la fase de diseño.

6.1 IMPLEMENTACIÓN DEL HARDWARE.

Se comienza con la construcción de las articulaciones (dos servos), para ello y puesto que solo existe la articulación del hombro y codo, es decir una en el eje Z (hombro) y otra en el eje Y (codo), simplemente pegamos los servos formando un ángulo recto con sus anclajes y cuerpo, más tarde los pintamos con espray de pintura blanca (RAL-9010) del siguiente modo:

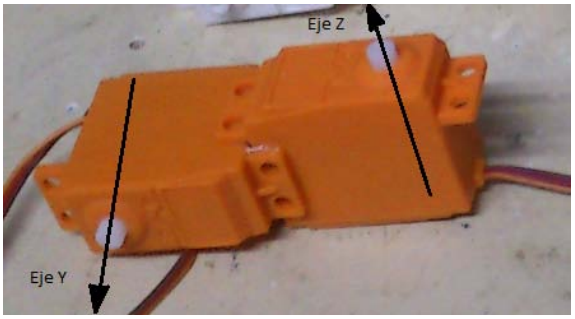


Figura 6.1: *Fotografía pata derecha*

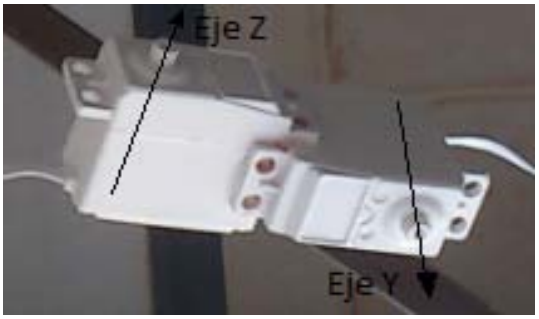


Figura 6.2: *Fotografía pata izquierda*

El pegamento empleado es de tipo binario, epoxi especial para plásticos.

Las patas se implementaron primero en madera de luterma a modo de prototipo para ver si se adecuaban las dimensiones, más tarde se hizo otra de metraquilato para ir probando la forma de cortar este material (que por sus características termoplásticas requiere de gran cuidado en su manejo en el corte, puede que el plástico derretido por la fricción del corte se adhiera a la hoja de la sierra y partamos la pieza que se esté realizando). Las dimensiones del cuerpo también se fueron probando con luterma hasta obtener las óptimas en la que las patas no se entorpecían entre sí al caminar.

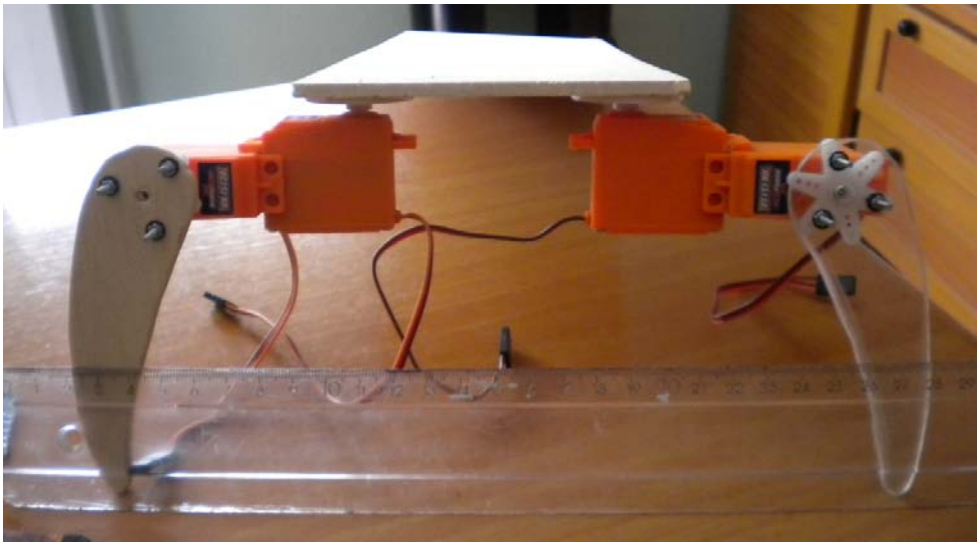


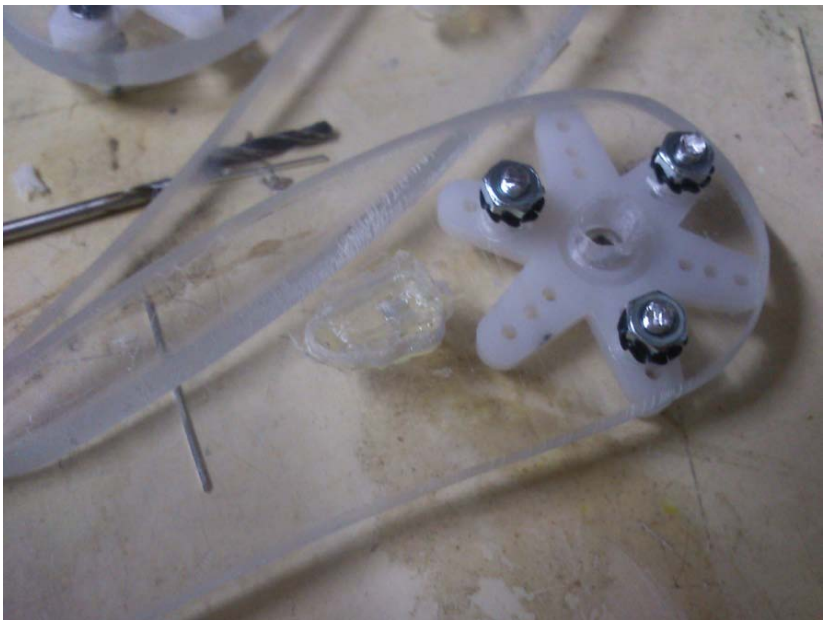
Figura 6.3: *Primeros prototipados del robot*

Una vez se ha adquirido experiencia en este prototipo de pata de metraquilato para cortar dicho material, se cortan otras 6 patas idénticas que serán las que formen nuestro robot final.

Figura 6.4: *Patas finales recortadas (en metraquilato)*

Además se deberán de incluir los orificios para la colocación de los LED y para atornillar las coronas (piezas que se atornillan al eje de los servos y que son las que transmiten el movimiento a la pata). Los tornillos serán de Ø=2.5mm y se emplearán también unas arandelas dentadas para su mejor sujeción con el metraquilato.

Para alimentar las luces LED se desarmó cada servo correspondiente a cada pata y se aprovechó las entradas de alimentación y masa del servo para realizar la conexión con el LED. Colocando una resistencia de de 270Ω a masa y ánodo del servo mediante cablecillos blancos y conectando directamente el ánodo del servo a la alimentación. De este modo se ahorra cableado y en posibles problemas que pudieran dar tantos cables sueltos cerca de las patas.



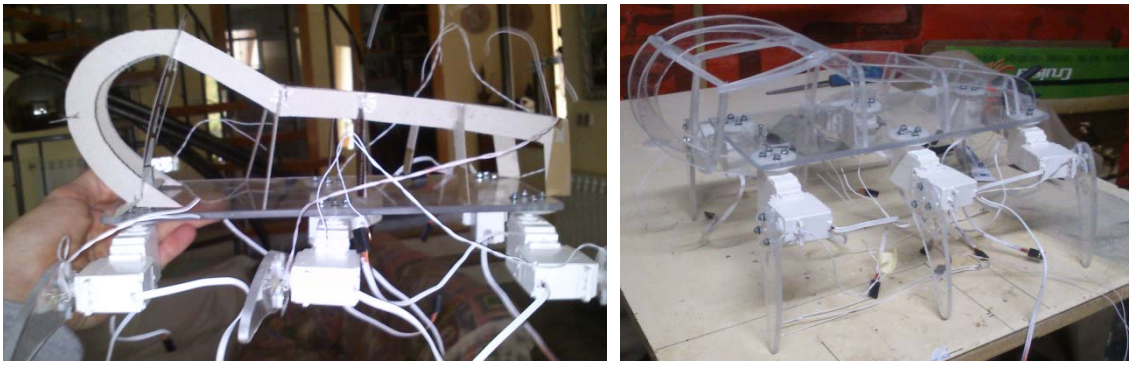
Figuras 6.5 y 6.6: *Detalles de las patas: anclaje con las coronas de los servos y LED*

Se procede a la sustitución de la plataforma de luterma por una de metraquilato que será ya la definitiva, hay que medir la posición en las que irán las coronas (el término corona del servo se refiere a la pieza que se atornilla al servo y transmite el movimiento al elemento que esté conectada a esta) de la articulación del hombro para conseguir una simetría exacta (esto se reflejará más tarde en lo recto que se desplace el robot) y se realizan los agujeros para su colocación con los tornillos y las arandelas dentadas.

Figura 6.7 y 6.8: *Montura patas a la base y detalle de base de metraquilato con las coronas ancladas*

Se está ya en disposición para hacer la estructura, una de las partes que requirió de mayor empleo de tiempo en el diseño y que modificación tras modificación fue mejorando hasta adaptar la forma que se precisaba. Como se comentó anteriormente está formada por "costillas" y que a modo de puzle se van montando para dar volumen a la coraza del robot, es más ligera que la opción de cubrirlo por completo, y con esto se consigue una protección de los circuitos similar. Además si existiera la posibilidad de que algún componente se sobrecalentase, de este modo tendría también más ventilación.

Al principio todo modelo de estructura se implementó en cartón, el cual era fácilmente modificable y de rápida construcción. Una vez se obtuvo el modelo que satisfacía las expectativas del proyecto, se pasó a su modelado en metraquilato, obteniéndose un muy buen resultado, y dando ahora verdadera sensación de que de que se trata de un auténtico hexápodo.



Figuras 6.9 y 6.10: *Fases de creación de la estructura, desde los primeros prototipos en cartón hasta el diseño final en metraquilato*

Además se le incluyeron ya los LED bicolor a modo de ojos en la cabeza. Toda la estructura fue pegada con epoxi transparente dando la sensación de que formaba una sola pieza, además fue sujeta con tornillos, los cuales permitían con gran facilidad el desmontar la estructura de la base para posibles reparaciones o transporte.

Pasando a la placa de circuito impreso, fue realizada mediante microfresado como se había especificado en los planos (ir a capítulo *9 Planos*), la forma de sujeción de la placa al robot es mediante tornillos, los cuales atraviesan la placa en sus esquinas.

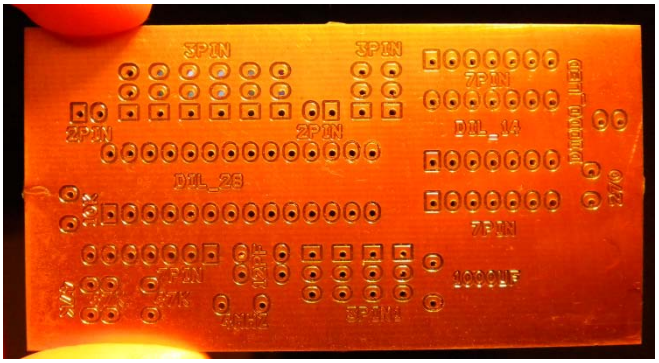


Figura 6.11: *Parte superior (Top)*

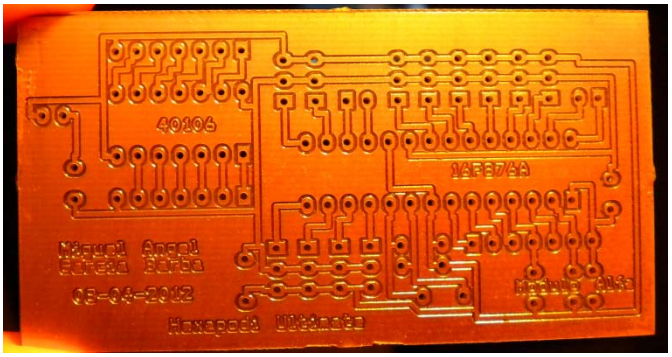


Figura 6.12: *Parte inferior (Bottom)*

Los conectores para los servos se han externalizado en dos arrays de conectores puestos en los flancos del robot, esto se ha hecho para una mejor conexión/desconexión de los mismos.

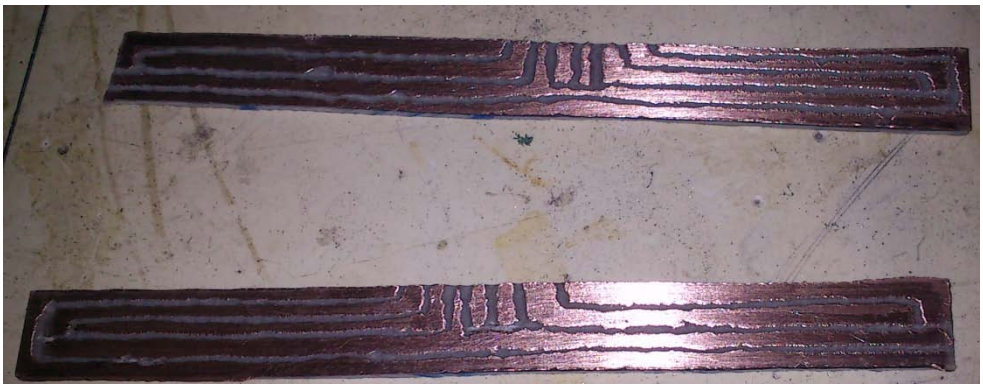
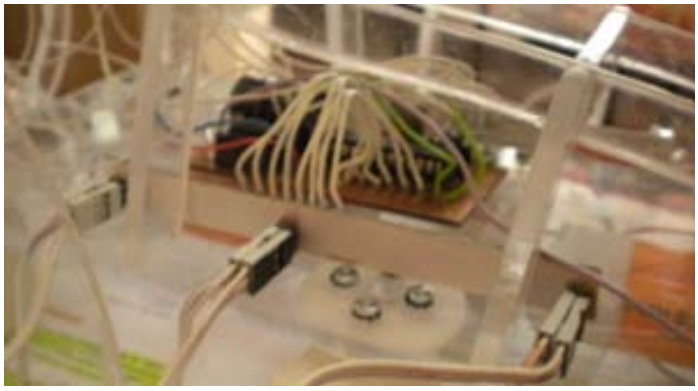


Figura 6.13: *Placas para array de conectores*



Figuras 6.14: *Array de conectores sobre el robot, observamos la mejora que supone en cuanto a accesibilidad para el conexionado con los servos*

El condensador para estabilizar la tensión en el microcontrolador es de 1000 µF, valor más que suficiente para que todo funcione correctamente.

Además se ha colocado una pequeña placa de circuito impreso en la parte trasera del robot, donde se controlanr las funciones y otros aspectos del robot, como: el jack de carga de 6v para las baterías, botón de *reset* del sistema, un interruptor con dos posiciones, a la derecha activa el robot y a la izquierda lo desactiva permitiendo la carga de las baterías (las cuales han sido dispuestas en la parte inferior de la estructura, haciendo que le punto de equilibrio este lo más cercano al suelo posible aumentando la estabilidad del robot). Por último un array de interruptores donde solo son hábiles los dos de la izquierda (1=puerto C7, 2=puerto C6), permitiendo mediante los 4 posibles estados de los interruptores (2^2) seleccionar el modo en el que va a trabajar el robot, siendo:

- ✓ 00: esquivar objetos (solo habilitados los infrarrojos)
- ✓ 01: seguimiento de luz (solo habilitadas las LDRs)
- ✓ 10: seguimiento de luz con sorteo de obstáculos (ambos sensores activados).
- ✓ 11: no implementado, aplicación futura (control remoto con mando).

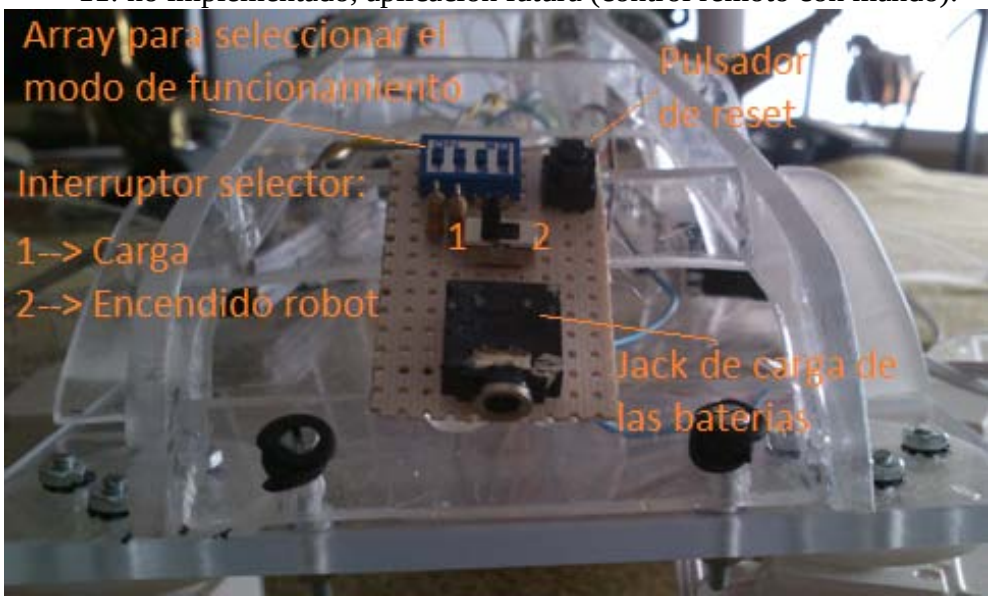


Figura 6.15: *Detalle de la placa de control*

Los sensores lumínicos (LDRs) se sitúan a modo de antenas en la cabeza y simulando la boca en la parte delantera de la cabeza. Los sensores de infrarrojos por otro lado también han sido acoplados a la cabeza a cada lateral, llevando a cabo el testeo de la proximidad de objetos según la imagen siguiente:

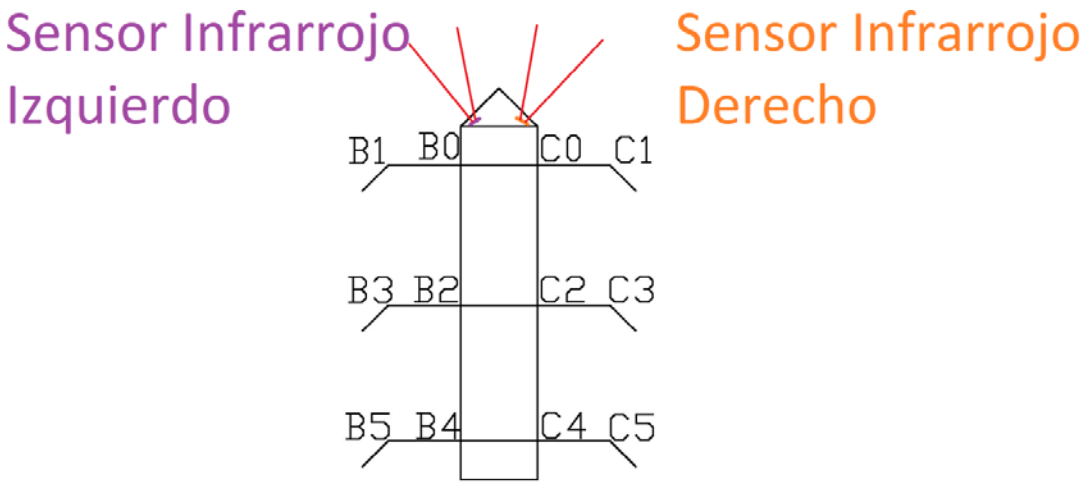


Figura 6.16: *Dirección de los rayos infrarrojos*

Por último destacar que se le ha dotado de un botón de *reset* (se conecta al primer pin del array donde están las entradas analógicas), es un *reset* que se activa por nivel bajo con lo que lo conectaremos entre masa y la entrada de *reset* del microcontrolador.

Se han extendido los puntos de conexión de masa y alimentación mediante dos conectores hembra de 4 pines, a los cuales se le conectarán cables auxiliares como son los de alimentación de los sensores, ojos, etc.

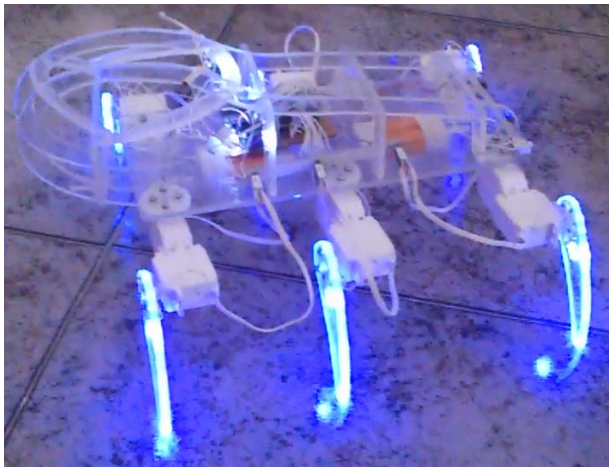


Figura 6.17: *Resultado final*

6.2 IMPLEMENTACIÓN DEL SOFTWARE.

En primer lugar se aborda la lógica de funcionamiento, lógica implementada en el PIC para procesar la información proveniente de los sensores y tratarla de forma que se actúe sobre las subrutinas o funciones exactas y de esta forma alcanzar de la forma más eficiente la fuente lumínica.

Comenzando con los sensores de luz (son tres) y habiendo muchas combinaciones posibles, se reducen a tres, que son: sensor 1 y 2 con más luz que 3, sensor 3 y 2 con más luz que 1 y sensor 1 y 3 con más luz que 2.

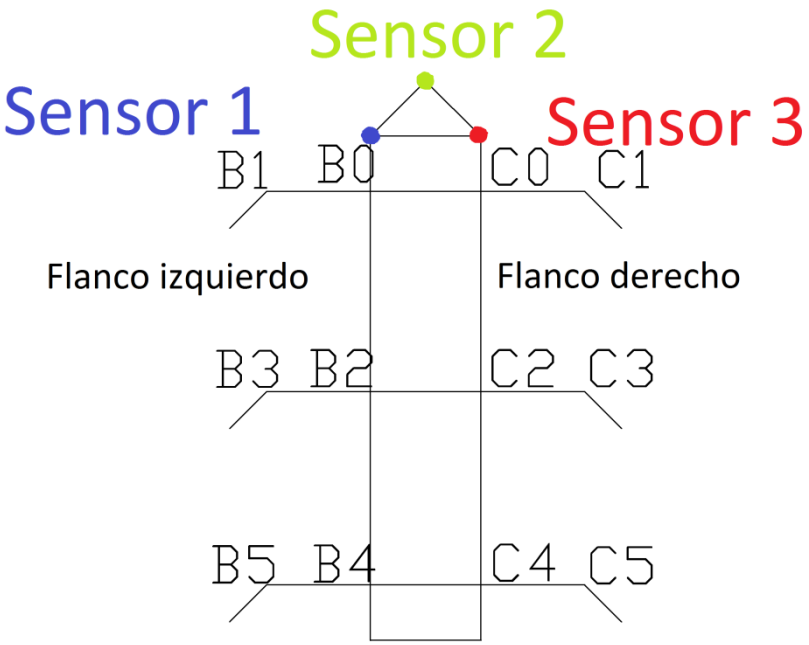


Figura 6.18: *Especificación del emplazamiento de los sensores lumínicos*

- Sensor 1 y 2 con más luz que 3:

La luz procede más del flanco izquierdo que del derecho con lo que lo primero que se evalúa es ver si la luz se encuentra en el frente; si no es así la luz proviene entonces de la izquierda, por lo que habrá que saber como de ladeada se encuentra, es decir, si se encuentra muy a la izquierda el robot girara en modo “izquierda Tanque”, no obstante si se encuentra solo ligeramente hacia la izquierda con la subrutina de “izquierda” simplemente bastaría.

- Sensor 3 y 2 con más luz que 1:

La luz procede más del flanco derecho que del izquierdo con lo que lo primero que se evalúa es ver si la luz se encuentra en el frente; si no es así la luz proviene entonces de la derecha, por lo que habrá que saber como de ladeada se encuentra, es decir, si se encuentra muy a la derecha el robot girara en modo "derecha Tanque", no obstante si se encuentra solo ligeramente hacia la derecha con la subrutina de "derecha" simplemente nos bastaría.

- Sensor 1 y 3 con más luz que 2:

La luz procede de atrás ya que el sensor delantero no detectaría apenas nada. Se procede a evaluar cual de los dos sensores detecta más luz, para que el robot con la subrutina de "izquierda Tanque" o "derecha Tanque" se gire lo más rápidamente hacia ella.

Todos estos algoritmos quedan simplificados en el flujograma de la **Figura 6.18** donde se explica la lógica implementada en el seguimiento de la fuente de luz.

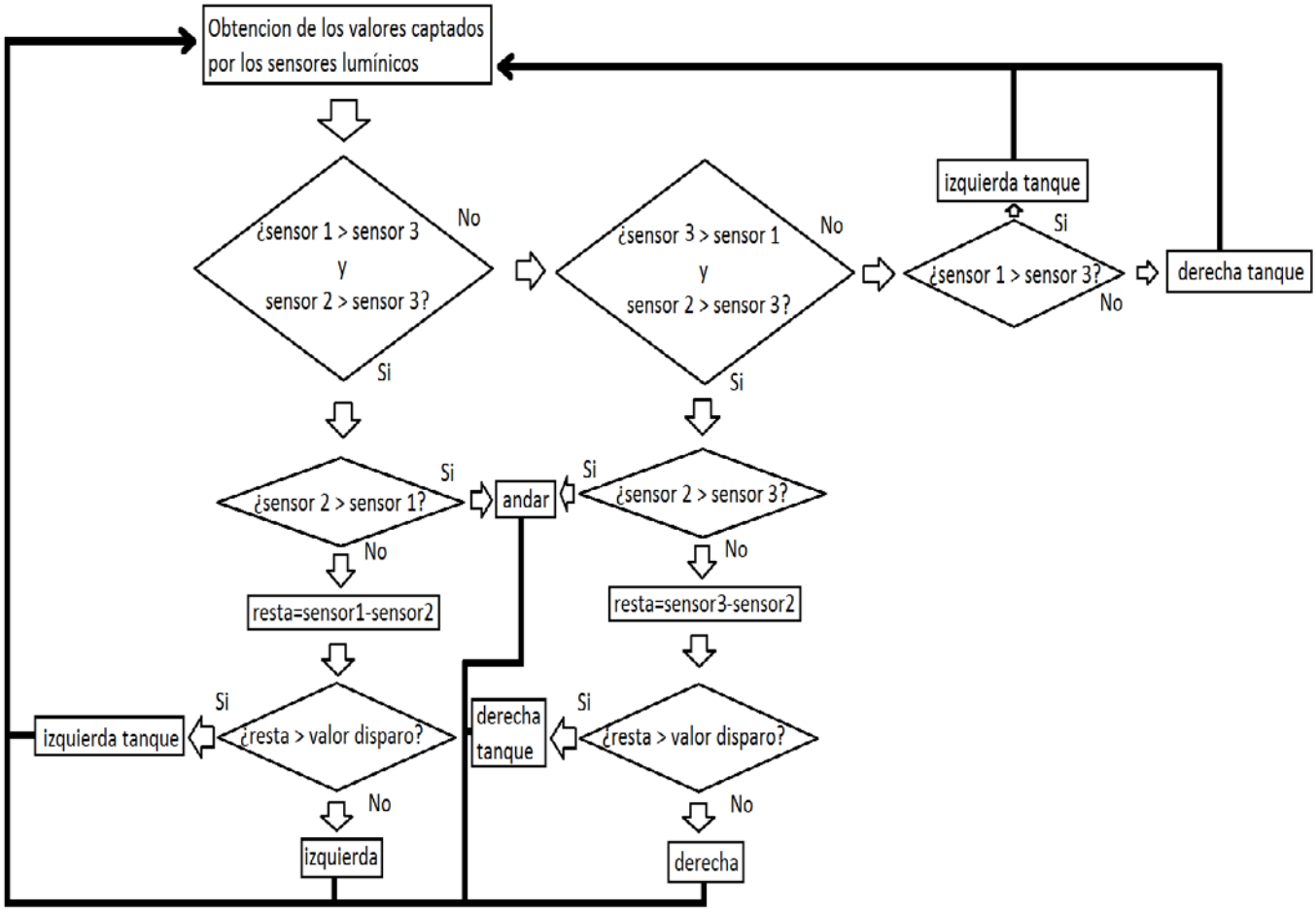


Figura 6.19: *Flujograma para el programa del tratamiento de las LDRs*

Pasamos ahora a explicar la lógica de los sensores infrarrojos:

Los sensores de infrarrojos dan una salida analógica en función de la distancia del sensor al objeto, con lo que analizando la tensión de cada sensor, se puede conocer la distancia que esta el objeto de nuestro robot. Para ello se establece un umbral, como la señal analógica se va a procesar y a traducir a datos digitales cuyo valor irá de 0 a 1023, para una distancia de unos 50 cm (1.3v) el umbral se situará en unos 200 puntos sobre 1023 que serían los 5v; con lo que cada vez que un sensor sobrepase ese umbral (es decir, el objeto se ha acercado más de 50cm) se lleva a cabo la función correspondiente a girar en modo tanque más un paso de andar, para escapar de esa proximidad al objeto.

En el flujograma de la **Figura 6.19** se muestra la lógica antes descrita de una forma más simplificada y a partir de la cual se creará el código.

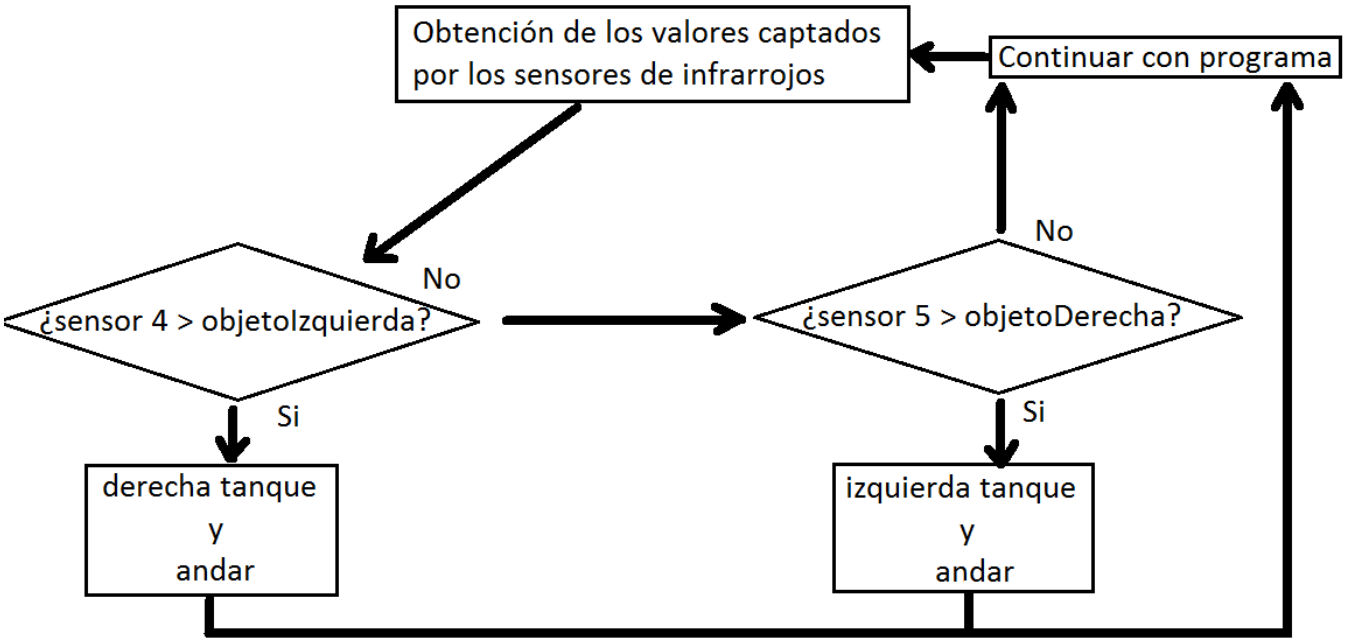


Figura 6.20: *Flujograma del algoritmo para esquivar objetos*

Para el código implementado, se expican algunos conceptos que aquí aparecen:

- Cualquier término relacionado con 1 se refiere al sensor LDR de la izquierda.
- Cualquier término relacionado con 2 se refiere al sensor LDR del frente.
- Cualquier término relacionado con 3 se refiere al sensor LDR de la derecha.
- Cualquier término relacionado con 4 se refiere al sensor infrarrojo de la izquierda.
- Cualquier término relacionado con 5 se refiere al sensor infrarrojo de la derecha.
- Offset"x"= variable cuyo valor se le asigna para corregir las posibles desviaciones producidas por la imperfección de las LDRs y de los sensores infrarrojos, en resumen, por su no paridad.
- valorDisparo"x"= El valor de la resta entre el sensor 2 y cualquiera de los otros que indicará si la luz está muy ladeada, activando en consecuencia el modo "tanque" correspondiente.
- objeto(Izquierda/Derecha)= valor para el cual, a partir de ahí para arriba saltará la subrutina correspondiente para esquivar el objeto.

Una vez explicado como pensaría el robot, se muestra la parte código que realiza estas funciones y que ha sido obtenido partiendo de sus correspondientes flujogramas:

Código para las LDRs:

```
set_adc_channel(0);                 //La lectura analógica se va a realizar en el puerto 0.
  delay_us(20);
  lectura1=read_adc() + offset1;      //lectura1 coge el valor del sensor 1 mas el offset que le hayamos establecido.

  set_adc_channel(1);                 //La lectura analógica se va a realizar en el puerto 1.
  delay_us(20);
  lectura3=read_adc() + offset3;      //lectura2 coge el valor del sensor 2 mas el offset que le hayamos establecido.

  set_adc_channel(3);                 //La lectura analógica se va a realizar en el puerto 3.
  delay_us(20);
```

```
  lectura2=read_adc() + offset2;          //lectura3 coge el valor del sensor 3 mas el offset que le hayamos establecido.

//Hemos realizado la asignación de los valores leídos por los sensores a cada una de las variables de los sensores, con esta información pasamos a analizar lo que está viendo el robot para actuar en consecuencia:
  delay_us(20);
  //Tiempo de espera de 20 µs para que se estabilice el programa y apartir de aquí encontramos las condiciones de cada movimiento.
  if(lectura1>lectura3 && lectura2>lectura3){        //luz en la izquierda
    if(lectura2>lectura1){
      Andar ();                 //adelante
      goto inicio;}
    else{
      resta=lectura1-lectura2;
      if(resta>valorDisparo1){
        Izquierda_Tanque ();          //tanque izquierda
        goto inicio;}
      else{
        Izquierda ();          //gira izquierda
        goto inicio;}
    }
  }

  if(lectura3>lectura1 && lectura2>lectura1){        //luz en la derecha
    if(lectura2>lectura3){
      Andar ();                 //adelante
      goto inicio;}
    else{
      resta=lectura3-lectura2;
      if(resta>valorDisparo2){
        Derecha_Tanque ();          //tanque derecha
        goto inicio;}
      else{
        Derecha ();          //gira derecha
        goto inicio;}
    }
  }

  else{
    if(lectura1>lectura3){                    //Luz por atras a la izquirda
      Izquierda_Tanque ();              //tanque izquierda
      goto inicio;}
    else{                               //Luz por detras a la derecha
      Derecha_Tanque ();                //tanque derecha
      goto  inicio;}
  }
```

Código para los sensores infrarrojos:

```
set_adc_channel(3);                    //La lectura analógica se va a realizar en el puerto 0.
  delay_us(20);
  lectura4=read_adc() + offset4;       //lectura1 coge el valor del sensor 1 mas el offset que le hayamos establecido.

  set_adc_channel(4);                  //La lectura analógica se va a realizar en el puerto 1.
  delay_us(20);
  lectura5=read_adc() + offset5;       //lectura2 coge el valor del sensor 2 mas el offset que le hayamos establecido.


//Hemos realizado la asignación de los valores leídos por los sensores a cada una de las variables de los sensores, con esta información pasamos a analizar lo que está viendo el robot para actuar en consecuencia:
  delay_us(20);
  //Tiempo de espera de 20 µs para que se estabilice el programa y a partir de aquí encontramos las condiciones de cada movimiento.
    if (lectura4> objetoIzquierda){    //Si el objeto se encuentra más cerca de lo establecido por el flanco Izquierdo
    Derecha_Tanque();
    Andar();
    goto  inicio;}

  if (lectura5> objetoDerecha){        //Si el objeto se encuentra más cerca de lo establecido por el flanco Derecho
    Izquierda_Tanque();
    Andar();
    goto  inicio;}
```

Si no se comprende alguna parte del código no hay más que volver a ver los pasos a seguir para el tratamiento del software, lo único que se ha hecho es implementar dichos pasos, es decir el flujograma, en lenguaje C, entendible por el compilador.

Sin más, en el **"Anexo Código"**, se muestra todo el código (programa completo) empleado para el programa del robot, con sus funciones, definiciones de periféricos, etc. los comentarios quedarán destacados en azul, mientras que el código propiamente dicho estará en verde oscuro

CAPÍTULO 7

PRUEBAS

Con el robot ya terminado se pasa a comprobar su correcto funcionamiento. Para ello se dividen las pruebas en dos: las estáticas y las dinámicas. La finalidad de cada una de las pruebas es comprobar el funcionamiento de cada una de las partes del proyecto. Con las pruebas estáticas se verifica que los sensores lumínicos reaccionen bien a la posición de la luz y los de infrarrojos a la proximidad de un obstáculo, con las dinámicas se observan los movimientos del robot para ver si se desenvuelve con soltura en su medio según le van llegando datos a través de los sensores.

7.1 PRUEBAS ESTÁTICAS.

Estas pruebas se emplean para comprobar que, por un lado:

-Los sensores de luz triangulan correctamente la posición de la luz, si no es así simplemente se reorientan las LDRs hasta conseguirlo.

-Los sensores infrarrojos se encuentran dentro del rango de detección deseado y que apuntan en la dirección adecuada, de no ser así, al igual que con las LDRs, se reorientaría la dirección del rayo infrarrojo y para el rango de actuación se modificaría el valor de disparo del software, hasta que reaccione a la distancia deseada.

El software que se introduce en el microcontrolador PIC por su puesto no es el mismo que ejecuta el robot es su modo normal de funcionamiento, es un programa de prueba en el cual se emplean indicadores LED que señalizan la función a usar; así hay 5 LED bien diferenciados: dos para las funciones de Izquierda Tanque y Derecha Tanque, otros dos para Izquierda y Derecha y uno para Andar; del siguiente modo: Andar→B0, Izquierda→B1, Derecha→B2, Izquierda Tanque→B3, Derecha Tanque→B4

- Para las LDRs:

Se alimenta una protoboard donde estarán los LED y el PIC; en las entradas irán las salidas de las LDRs; balanceando el robot de izquierda a derecha frente a una fuente de luz se observa como los LED se van encendiendo, hay que comprobar que se encienden los LED cuando corresponde, como es el caso de las funciones de Izquierda Tanque y Derecha Tanque donde habrá que, de forma empírica, ir cambiando los valores de disparo hasta obtener el deseado.

El programa ejemplo es el siguiente:

```
#include <16F876A.h>        //Esto nos crea un archivo donde se especifican ciertas variables y comandos usados por el programa.
#device adc=10

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                  //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP       //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                         //No EE protection
#FUSES NOWRT                         //Program memory not write protected
#FUSES RESERVED                      //Used to set the reserved FUSE bits
#USE delay(clock=4000000)            //Tipo de reloj usado (4 Mhz)
#USE  FAST_IO(b)                     //Seremos nosotros quien establezcamos las patillas de entrada salida

void main() {

  setup_adc_ports(AN0_AN1_AN3);          //Establece los puertos que van a ser ADC
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);        //Deshbilitados timers...
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);


  set_tris_b(0b00000000);                //Puerto B todo salidas.

  output_b(0b00000000);                  //Nos aseguramos que las salidas estan a 0


  long lectura1, lectura2, lectura3, resta;
```

```
  long offset1=0;                    //factor de correccion para la LDR 1
  long offset2=0;                    //factor de correccion para la LDR 2
  long offset3=0;                    //factor de correccion para la LDR 3

  long valorDisparo1=300;                //rango de tension para diferenciar un giro
brusco con uno suave, flanco izquierdo
  long valorDisparo2=300;                //rango de tension para diferenciar un giro
brusco con uno suave, flanco derecha

  while(1){

  inicio:

  set_adc_channel(0);              //La lectura analógica se va a realizar en el puerto 0.
  delay_us(20);
  lectura1=read_adc() + offset1;             //lectura1 coge el valor del sensor 1 mas el
offset que le hayamos establecido.

  set_adc_channel(1);              //La lectura analógica se va a realizar en el puerto 1.
  delay_us(20);
  lectura3=read_adc() + offset3;             //lectura3 coge el valor del sensor 3 mas el
offset que le hayamos establecido.

  set_adc_channel(3);              //La lectura analógica se va a realizar en el puerto 3.
  delay_us(20);
  lectura2=read_adc() + offset2;             //lectura2 coge el valor del sensor 2 mas el
offset que le hayamos establecido.

  delay_us(20);


  if(lectura1>lectura3 && lectura2>lectura3){      //luz en la izquierda
    if(lectura2>lectura1){
      output_b(0b00000001);            //adelante
      goto inicio;}
    else{
      resta=lectura1-lectura2;
      if(resta>valorDisparo1){
        output_b(0b000001000);        //tanque izquierda
        goto inicio;}
      else{
        output_b(0b00000010);        //gira izquierda
        goto inicio;}
    }
  }
```

```
  if(lectura3>lectura1 && lectura2>lectura1){       //luz en la derecha
    if(lectura2>lectura3){
      output_b(0b00000001);              //adelante
      goto inicio;}
    else{
      resta=lectura3-lectura2;
      if(resta>valorDisparo2){
        output_b(0b00010000);          //tanque derecha
        goto inicio;}
      else{
        output_b(0b00000100);          //gira derecha
        goto inicio;}
    }
  }

  else{
    if(lectura1>lectura3){                    //Luz por atras a la izquirda
      output_b(0b00001000);             //tanque izquierda
      goto inicio;}
    else{                                 //Luz por detras a la derecha
      output_b(0b00001000);               //tanque derecha
      goto  inicio;}
  }
  }

}
```

- Para los Infrarrojos:

Se alimenta una protoboard donde estarán los LED y el PIC; en las entradas irán las salidas de los sensores infrarrojos; balanceando un objeto alrededor del robot observamos el área de seguridad que se define por los sensores de proximidad infrarrojos. Para ello se observa que LEDs se encienden en función del flanco por el que se acerque el objeto, al igual que con el programa de las LDRs hay que tener en cuenta la iluminación de los LED ya que esto marcará la función que se va a realizar en cada momento con lo que se deberá de llevar a cabo una tarea correctora (modificando el ángulo del rayo infrarrojo o el valor de disparo) en el caso de que la función activada no sea la que se corresponda con la situación actual de los sensores respecto a los objetos.

```
#include <16F876A.h>        //Esto nos crea un archivo donde se especifican ciertas variables y comandos usados por el programa.
#device adc=10

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                  //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP        //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                        //No EE protection
#FUSES NOWRT                        //Program memory not write protected
#FUSES RESERVED                     //Used to set the reserved FUSE bits
#USE delay(clock=4000000)           //Tipo de reloj usado (4 Mhz)
#USE  FAST_IO(b)                    //Seremos nosotros quien establezcamos las patillas de entrada salida

void main()
{
  setup_adc_ports(ALL_ANALOG);          //Establece los puertos que van a ser ADC
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);        //Deshbilitados timers...
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);

  set_tris_b(0b00000000);               //Puerto B todo salidas.
  output_b(0b00000000);                 //Nos aseguramos que las salidas estan a 0..
```

```
  long lectura4, lectura5;
  long offset4=0;
  long offset5=0;

  long objetoIzquierda=200;
  long objetoDerecha=200;

  inicio:

  set_adc_channel(3);              //La lectura analógica se va a realizar en el puerto 3.
  delay_us(20);
  lectura4=read_adc() + offset4;                    //lectura4 coge el valor del sensor infrarrojo izquierdo mas el offset que le hayamos establecido.

  set_adc_channel(4);              //La lectura analógica se va a realizar en el puerto 4.
  delay_us(20);
  lectura5=read_adc() + offset5;                    //lectura5 coge el valor del sensor infrarrojo derecho mas el offset que le hayamos establecido

  delay_us(20);



    if (lectura4> objetoIzquierda){       //Si el objeto se encuentra más cerca de lo establecido por el flanco Izquierdo
    Derecha_Tanque();
    Andar();
    goto  inicio;}

  if (lectura5> objetoDerecha){           //Si el objeto se encuentra más cerca de lo establecido por el flanco Derecho
    Izquierda_Tanque();
    Andar();
    goto  inicio;}

   }
```

Estos programas son muy similares al programa principal del robot en el que procesamos las señales de los sensores, es debido a que en vez de actuar sobre las funciones y que el robot se desplace, se actua sobre los LED, siendo esta respuesta más rápida y procediendo a una depuración sobre la posición de los sensores instantánea.

Una vez se ha pasado con éxito esta prueba, se procede a las pruebas dinámicas, en las que se comprueba cómo se desplaza el hexápodo esquivando objetos para lograr alcanzar su objetivo, “la luz”.

7.2 PRUEBAS DINÁMICAS.

En este punto es donde se comprueba realmente si el robot cumple con lo definido en el proyecto, se somete a una serie de pruebas, entre las que se incluye, distintos entornos y escenarios.

Primero se comprobó en un banco de pruebas (se sostenía al robot sin tocar el suelo) que al incidir la luz de distintas formas sobre sus sensores las patas efectuaban los distintos movimientos para los que estaba programado robot y que corresponden a sus diferentes funciones. De igual modo se hizo enfrentando objetos al robot para comprobar que el movimiento de las patas lo hacía huir de dicha proximidad.

Todo debe de ir y funcionar correctamente, con lo que ahora habrá que someter al robot a distintos escenarios para comprobar que se ejecuta la función correspondiente.

- ✓ Para los lumínicos (espacios abiertos): entornos totalmente oscuros, con distintos puntos de luz, con un único punto, punto de luz móvil etc.

- ✓ Para los infrarrojos: intentar desplazarse de un lada a otro de un área con obstáculos, salir de un laberinto de paredes, situarlo en medio de muchos objetos y ver como los sortea, etc.

- ✓ Conjunto (lumínicos e infrarrojos): Esta es la prueba definitiva y vale cualquier tipo de entorno, ya sea con mucha o poca luz, rodeado de objetos, luces detrás de objetos… será el entorno más complejo que se pueda encontrar, no obstante con la ayuda de los dos tipo de sensores se solventará sin mayores dificultades.

Todo ello para detallar en que entornos el funcionamiento del robot es más favorable, esto quedará reflejado en el capítulo de *8 Conclusiones.*

En la carpeta Multimedia, encontramos videos donde se muestran algunas de las pruebas realizadas para comprobar el correcto funcionamiento del robot.

CAPÍTULO 8

CONCLUSIONES Y TRABAJO FUTURO

8.1 PRESUPUESTO.

En lo referido al presupuesto, hay que decir que se gastó más de lo que requeriría la construcción del robot, esto es lógico ya que en todo prototipo el mayor empleo del tiempo, de probaturas con diferentes materiales, sensores, lógicas, etc. hace que el prototipo se encarezca; es un encarecimiento necesario que hace obtener el mejor producto posible, para así y partiendo de este robot final se comience la posible cadena de producción o producción en masa del producto, el cual es infinitamente más barato que el prototipo original. A cada producto habrá que añadirle un pequeño porcentaje en su precio para alcanzar lo gastado en el empleo de los recursos para el prototipo, es decir rentabilizar la inversión inicial.

No se va a detallar uno a uno los componentes usados y que finalmente no han dado resultado, o las lógicas implementadas que no han hecho más que suponer un retraso en la construcción del proyecto, simplemente se va a estimar de una forma más o menos intuitiva el peso económico a restar del prototipo para obtener el valor real del robot. El presupuesto desglosado del robot, por otra parte, si que se mostrará indicando que precio corresponde a qué concepto; de esta forma se procede:

Para las mermas producidas por el empleo de materiales que resultaron dañados o no adecuados para el proyecto, así como a consecuencia de emplear componentes de alta calidad y que al no poder adquirirse sueltos tuvieron que ser comprados en lotes, ascienden a unos 50€. En un proceso de creación de un producto, el perder tan solo 50€ en el prototipo, nos asegura una rentabilidad y un éxito casi garantizado por parte de la empresa que lo lleve a cabo, ya que los beneficios se obtendrían a partir de la 2ª unidad vendida más o menos (estimando que el precio del robot (el cual se desglosará más adelante) sea de unos 240€ y los vendamos por 280€).

Pasemos al desglose del precio del robot:

- Partes mecánicas:

Nº	***Componente***	***Precio unitario***	***Precio final***
12	Servo HK15138	2.95$ = 2.35€	28.2€
1	Placa metraquilato 1m² 5mm de grosor	24€	24€
39	Tornillo	0.16€	6.24€
39	Tuerca	0.10€	3.90€
39	Arandela	0.06€	2.34€
3	Arandela goma	0.05€	0.15€
2	LED bicolor	1.4€	2.8€
6	LED azul	0.90€	5.40€
1	Epoxi	6.50€	6.50€
4	Baterías **HHR-450A-1Z**	12.50€	50€
1	Interruptor	0.70€	0.70€
1	Jack de carga	0.50€	0.50€
1	Array 2 microrruptores	2.10€	2.10€
3	LDR	1.50€	4.50€
2	Sensor infrarrojo **GP2Y0A02YK**	17.85€	35.70€
3	Envío y transporte	6€	18€
1	Espray pintura RAL-1090	8.90€	8.90€
Costo total partes mecánicas			199.93€

- Circuitería:

Nº	***Componente***	***Precio unitario***	***Precio final***
1	Microcontrolador PIC 16F876A	4.62€	4.62€
1	Oscilaor de cuarzo XT	3.60€	3.60€
2	Zócalos	0.15€	0.30€
59	Pines	0.03€	1.77€
10	Resistencias	0.07€	0.70€
2	Condensadores cerámicos	0.60€	1.20€
1	Condensador electrolítico	0.30€	0.30€
1	Inversor 40106	0.40€	0.40€
1	LED blanco	1.40€	1.40€
1	Fabricación placa circuito impreso	12€	12€
…	Cables para el conexionado	Estimación: 3€	3€
1	Rollo Estaño 60% - Plomo 40%	3.40€	3.40€
Costo total circuitería			32.69

Costos fraccionarios	***Costo***
Costo total partes mecánicas	199.93€
Costo total circuitería	32.69€
Costo final del producto	232.62€

A simple vista es de notar la diferencia entre ambos conceptos en cuanto a los costos de los materiales; siempre se ha dicho que la electrónica no es cara, si no la mano de obra o en nuestro caso las partes mecánicas cuyo precio es un 584.36% el de la circuitería.

Obviamente en los cálculos del presupuesto, no se han incluido el costo extra que supondría la mano de obra y horas dedicadas por tratarse de un Proyecto Fin de Carrera que actualmente no persigue objetivo comercial alguno, ni se han incluido los gastos por energía eléctrica, instalaciones ni uso del software. Todo ello debería de ser incluido en el caso que queramos comercializarlo, aplicándole un porcentaje al precio final donde se verían reflejados los honorarios que percibiesen los ingenieros encargados del proyecto.

8.2 CONCLUSIONES.

Las pruebas han arrojado luz acerca de los entornos más aptos para que se desenvuelva el robot: es capaz de seguir una fuente lumínica por un entorno oscuro y a la vez esquivando los objetos que se encuentra a su paso (modo 3→C7=1, C6=0), además en lugares con distintos puntos luminosos el robot siempre irá al que mayor intensidad lumínica presente (modo 2→C7=0, C6=1), cambiando su rumbo si dicho punto desaparece y aproximándose al que irradie más luz en ese momento.

Omitiendo la funcionalidad de seguir luz podemos dejar que nuestro robot camine sin más, sorteando los objetos que se presenten en su camino de una forma eficiente (modo 1→C7=0, C6=0), pudiendo cubrir así largas distancias, y cuando sus baterías se agoten, recurrir a la función de búsqueda de luz (modo 3→C7=1, C6=0), para situarse en el entorno más iluminado y cargar así unos posibles paneles solares que se le acoplen, para continuar su tarea exploradora más tarde (una vez se haya abastecido).

En definitiva no solo se trata de un robot todo-terreno por poseer 6 patas que le permiten una gran movilidad por todo tipo de terrenos, sino que mediante el reconocimiento del entorno que llevan a cabo los sensores, se pueden realizar multitud de tareas exploradoras con una extrema soltura sin preocuparse por que el robot pueda encallar o perderse. Los infrarrojos le libraran de colisión alguna con los obstáculos que pueda encontrarse por su camino y las LDRs lo guían hacia la luz, pudiendo por *software* en un futuro, programar para poder detectar presencia por las sombras que se formen etc.

8.3 TRABAJO FUTURO.

El potencial de este robot hexápodo experimental es alto en cuanto a las posibilidades de exploración y autonomía. Como se han dejado libres algunos puertos y periféricos del microcontrolador es fácilmente ampliable a otra función o a nuevos sensores. Podrían incluso sustituirse los sensores lumínicos y de infrarrojos por una cámara y mediante el procesado de imágenes captar todo el entorno que rodease a nuestro hexápodo, pudiéndose desenvolver con mayor soltura y habilidad, a la par de distinguir colores y grados de luminosidad; todo ello unido a unos eficientes paneles solares, harían de este hexápodo un elemento indispensable en la exploración de por ejemplo planetas, en los que la autonomía de los dispositivos es fundamental (las baterías que posee le permiten una autonomía, a máxima capacidad, de unas dos horas y media.

Figura 8.1: *Recreación ficticia de la labor a desempeñar por este tipo de dispositivos robóticos en el campo de la exploración espacial.*

Si soporta las condiciones de un entorno hostil como podría ser, por ejemplo, la exploración de Marte, también podría ser usado como robot de avanzadilla en los combates, localizando victimas e indicando su localización con un modulo GPS, detectando minas, observando más de cerca los puntos flacos del enemigo, etc. Es decir, podría ser usado como un dron cuya misión sea reconocimiento de áreas en conflicto, no poniendo en peligro la vida de persona alguna que estuviera destinada para esta labor.

Es evidente que la vía de los robots autónomos e inspirados en la naturaleza (bioinspirados) sea tan atractiva y se le encuentren tantas utilidades; la era de la robótica no empezó hace mucho (comparándola con otras tecnologías actuales, como la hidrodinámica de los barcos) con lo que aun queda mucho por avanzar y aprender, por descubrir y crear, es un mundo sin límites, el único límite lo impondrá tu capacidad de imaginar.

CAPÍTULO 9

PLANOS

Se especifican en este capítulo los planos para el completo montaje y construcción de el robot hexápodo.

9.1 CIRCUITO ELECTRONICO.

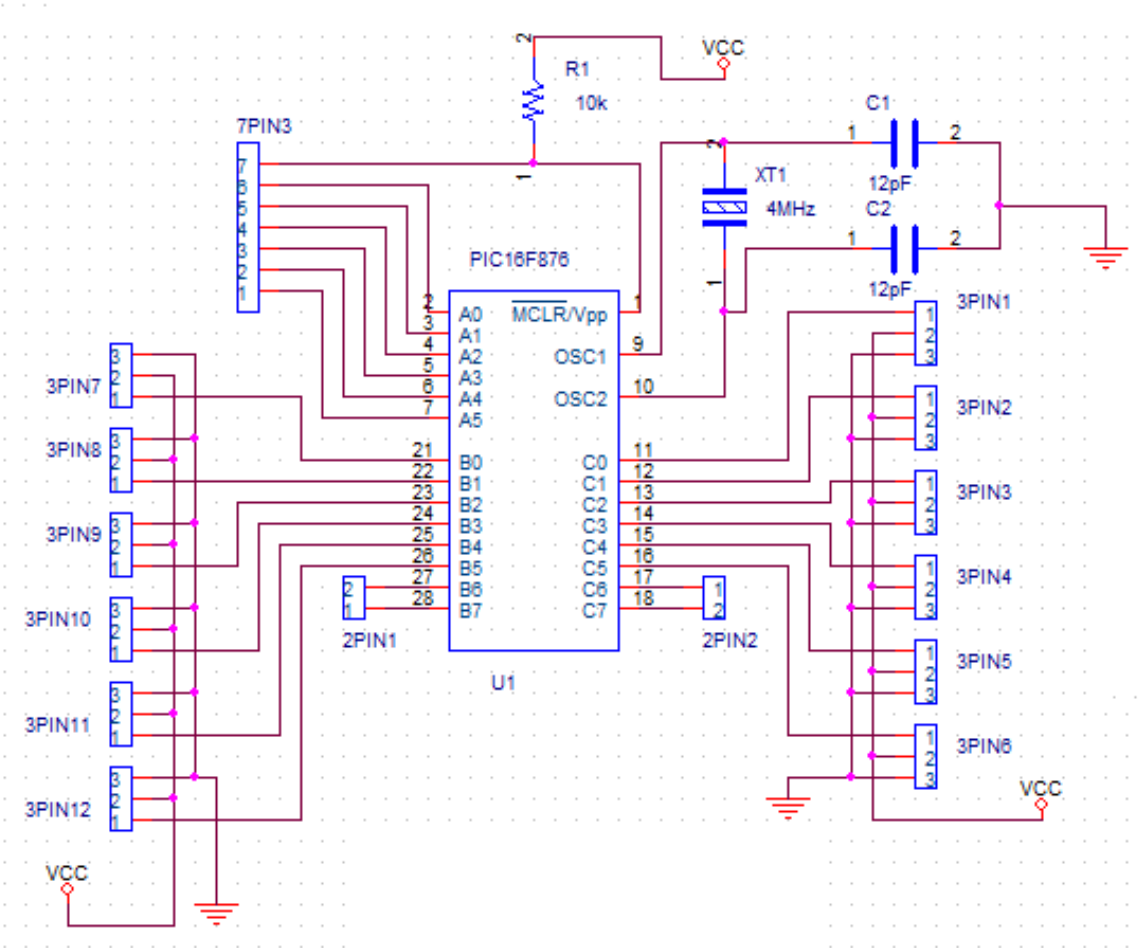


Figura 9.1: *Detalle del esquemático del microcontrolador*

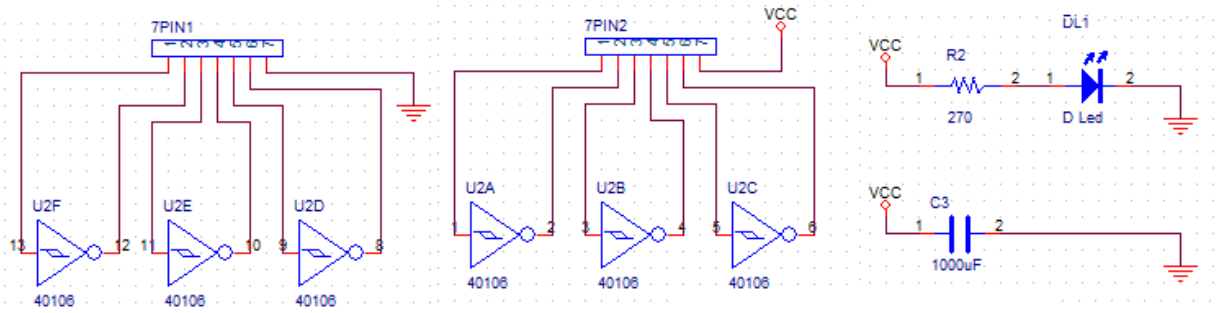


Figura 9.2: *Detalle del esquemático del inversor junto al LED y condensador*

9.2 LAYOUT DE LA PLACA.

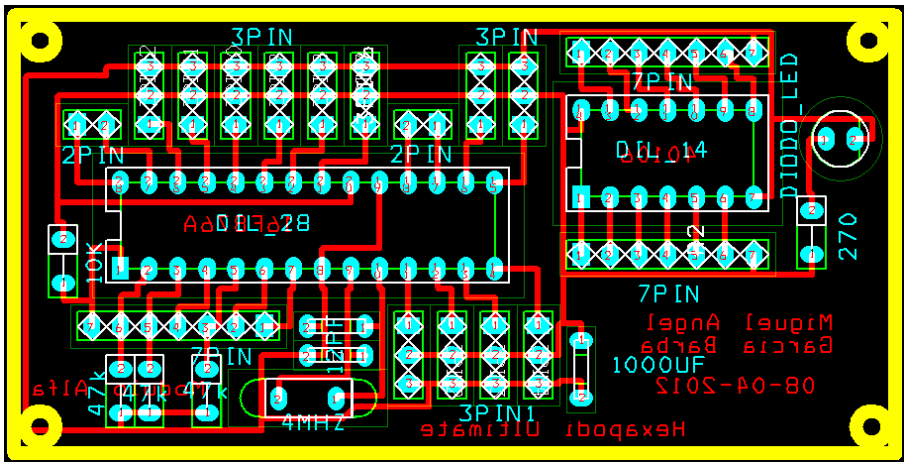


Figura 9.3: *Placa completa*

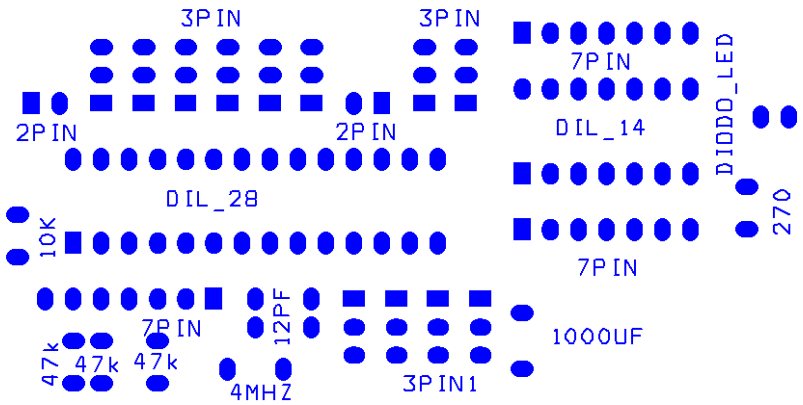


Figura 9.4: *Capa superior (TOP)*

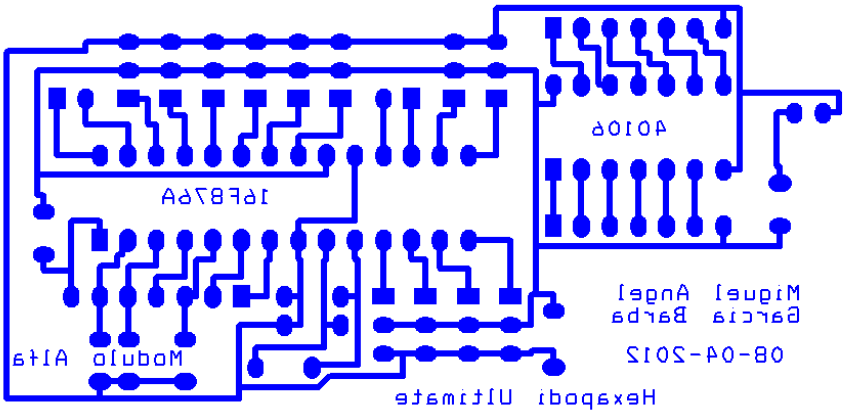


Figura 9.5: *Capa Inferior (BOTTOM)*

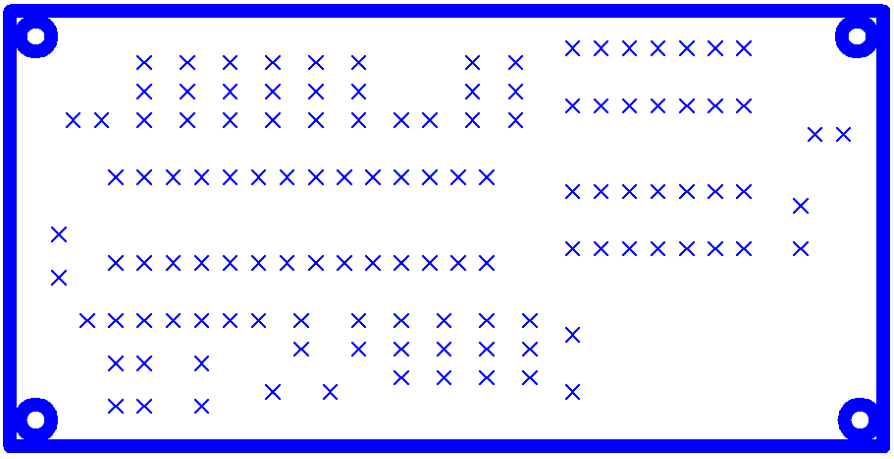


Figura 9.6: *Capa de los taladros (DRILL)*

9.3 PLANOS FÍSICOS DEL ROBOT.

En este sub-apartado se mostrarán los planos físicos del robot, su vista en planta así como los detalles del diseño de las patas y servos. Además cabe destacar que se le ha dotado de una estructura en forma de "costillas" (aportan resistencia al robot sin incrementar en gran medida su peso final) que permitirá proteger a los dispositivos internos del robot (el cerebro y las baterías) así como de soporte para la montura de los distintos sensores y balizas de estado.

Toda la estructura ha sido diseñada en metraquilato, para jugar con los juegos de luces que nos permiten los diodos LED y dotar de estilo al robot, creando algo novedoso e ingenioso.

- *ESTRUCTURA*

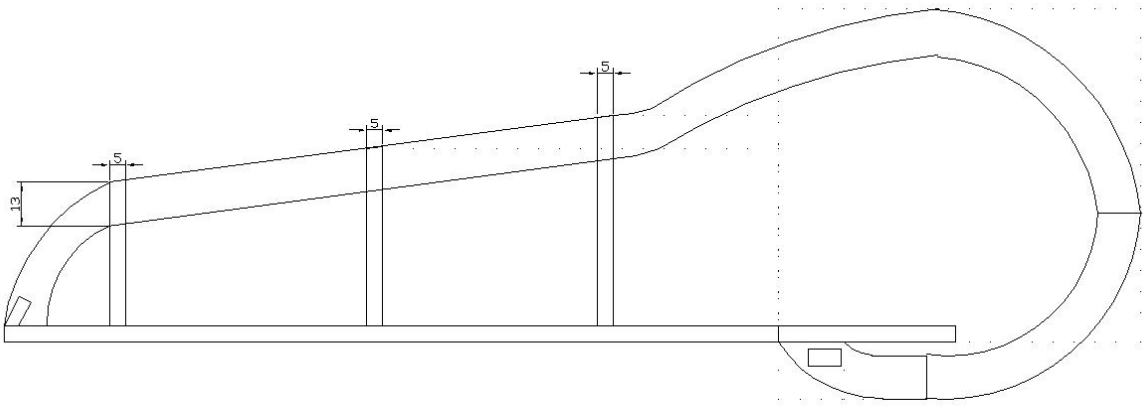


Figura 9.7: *Perfil*

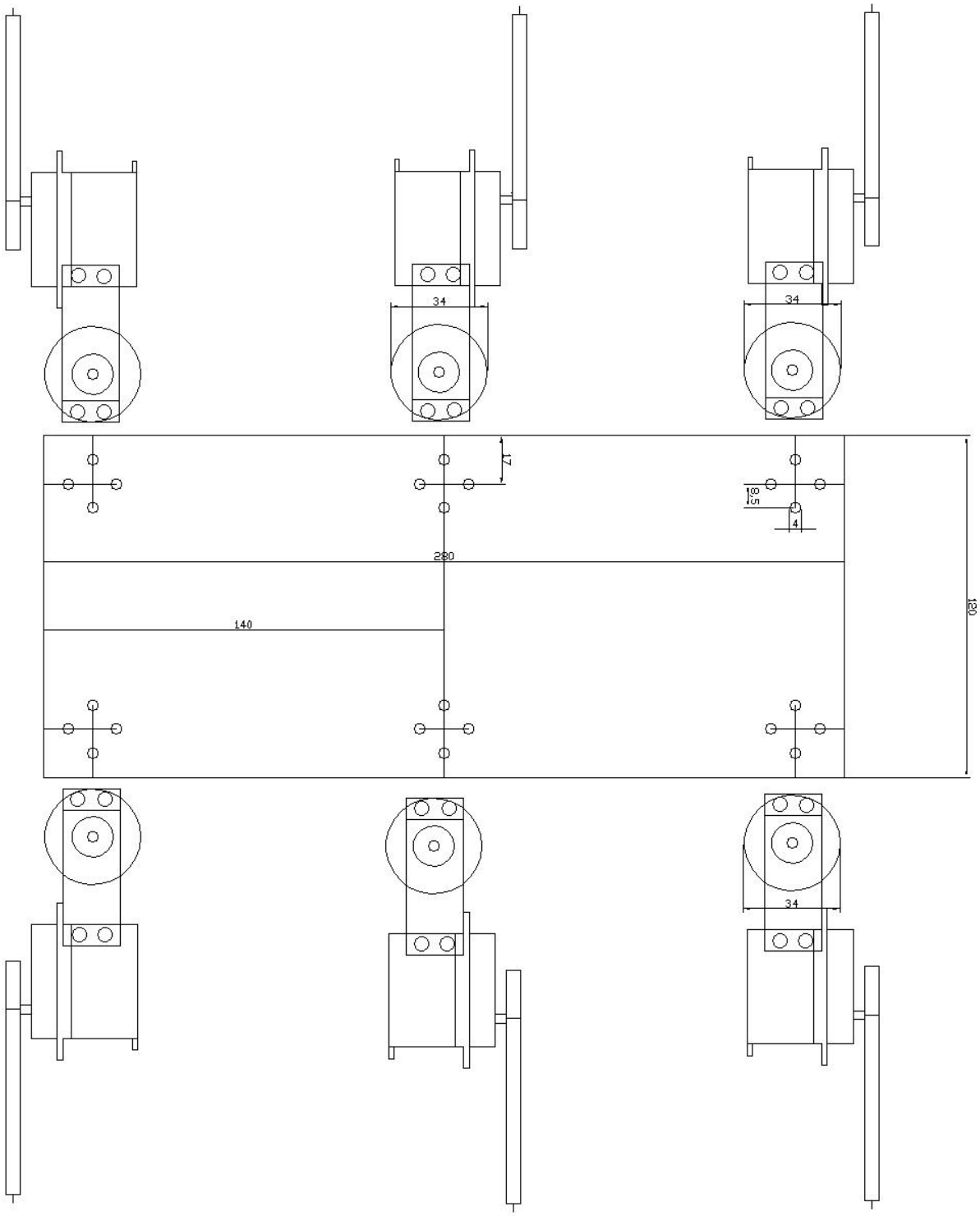


Figura 9.8: *Planta*

- PATAS

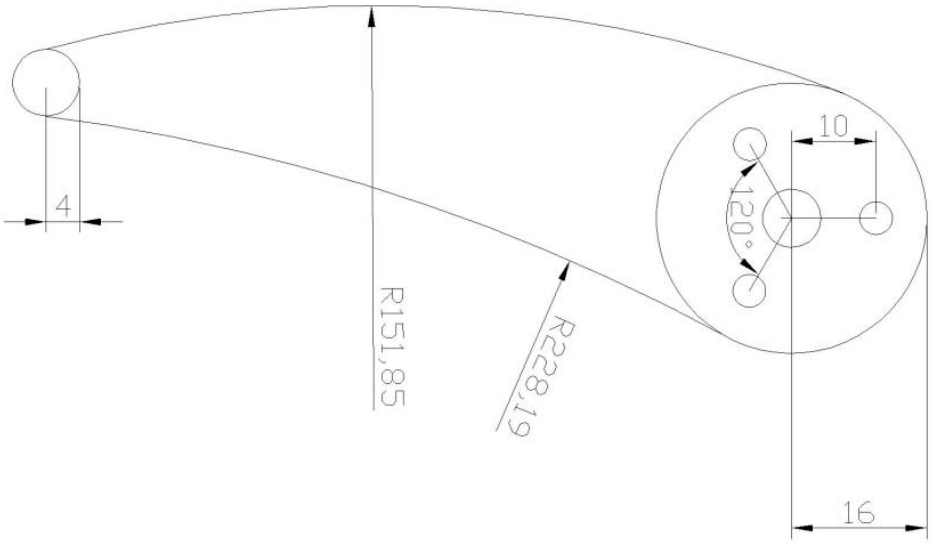


Figura 9.9: *Dimensiones pata*

- DIMENSIONES SERVOS

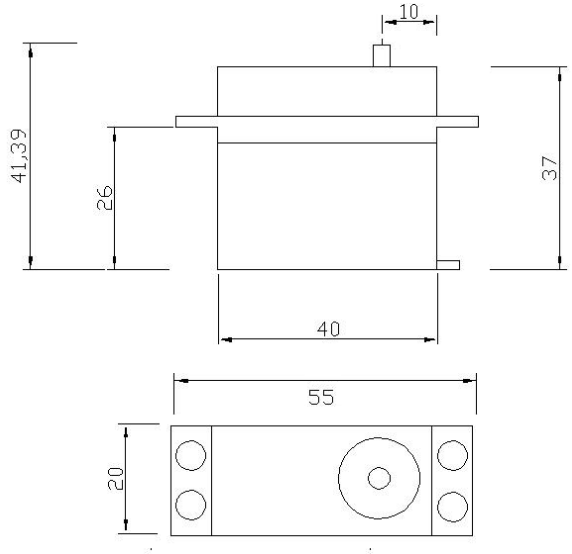


Figura 9.10: *Servo Real*

Weight (g)	38
Torque (kg)	4.3
Speed (Sec/60deg)	0.17
A(mm)	41
B(mm)	40
C(mm)	37
D(mm)	20
E(mm)	55
F(mm)	26

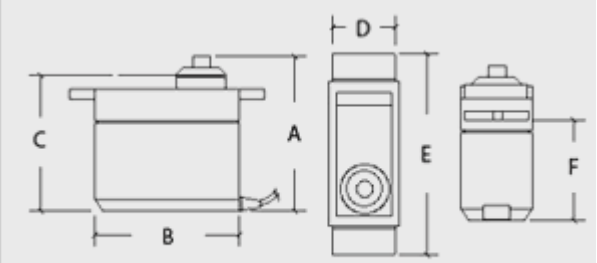


Figura 9.11: *Tabla Datasheet Servo*

CAPÍTULO 10

LISTADO DE COMPONENTES

9.1 COMPONENTES.

- Plancha de Metraquilato:
  - 5mm de grosor.

- x36 +x3 tornillos:
  - métrica zincados.
  - Ø 2.5mm.
  - L 20mm.

- x36 +x3 tuercas:
  - Ø (interno) 2.5mm.

- x36 +x3 arandela:
  - dentada de acero pavonado.
  - Ø (interno) 3mm.

- x12 servos HK15138:
  - Torque: 3.8Kg/cm² @ 4.8v, 4.3Kg/cm² @ 6v.
  - Peso: 38 g.
  - Velocidad: 0.21seg/60º @ 4.8v, 0.17seg/60º @ 6v.
  - Voltaje: 4.8v-6V.
  - Conector: tipo JR.
  - Dimensiones: especificadas en los diseños AutoCAD.

- Coronas servos:
  - Las de sus respectivos servos.

- Pintura:
  - Blanca, Ral-1090.

- x3 LDR:
  - Valor nominal 100KΩ.

- x2 Sensores infrarrojos:
  - GP2Y0A02YK.
  - Rango efectivo: 20-150cm.
  - Voltaje: 4.5-5.5V.
  - Salida en voltaje analógica.

- x1Cristal de cuarzo:
  - Oscilación de 4MHz.

- x2 Condensadores cerámicos:
  - Valor nominal: 22pF.

- Arrays de pines:
  - x12 de tres.
  - x3 de siete.
  - x2 de dos.

- Condensador Electrolítico:
  - Valor nominal: 1000µF.

- Diodos LED:
  - x6 Azules.
  - x1 Blanco.
  - x2 Bicolor (rojo-verde)

- Zócalos:
  - x1 de 28 pines torneado.
  - x1 de 14 pines planos.

- x1 microcontrolador PIC 16F876A:
  - 5 canales de conversión analógica-digital (convertidor A/D de 10bit = buena resolución).
  - Memoria de programa 14.3 Kbyte, que permiten 8192 instrucciones.
  - 22 Puertos de Entrada/Salida.
  - Memoria EEPROM de 256 bytes.
  - Comunicación serie mediante I²C.
  - 2 Módulos de Modulación por ancho de pulso (PWM).
  - 2 Comparadores Analógicos.
  - 3 *Timers*.

- x1 inversor disparador Shmitt:
  - 6 puertas inversoras.

- Resistencias:
  - x1 de 10KΩ.
  - x7 de 270 Ω.
  - x5 de 47KΩ.

- Cables:
  - Multihilo (aislante blanco).
  - Monohilo (aislante blanco).

- x4 Baterías HHR-450A-1Z:
  - Índice de descarga medio: 4500mAh.
  - Tensión nominal por célula: 1.2v.
  - Masa: 60 gramos.
  - Tamaño: Fat 5/4ª.
  - Dimensiones: Ø: 18.2 mm x 67mm.

- x1 Interruptor:
  - Tres patillas (carga y funcionamiento).

- x1 jack de carga:
  - Ø 3.5mm.

- x1 array de microrruptores:
  - x2 microrruptores en batería.

- Carrete de estaño:
  - 60% estaño-40% plomo.+ Flux.

9.2 PINOUT.

En cada una de las articulaciones de las patas, irá alojado un Servo Motor, pudiendo controlar exactamente el grado de inclinación de las articulaciones mediante la modulación por ancho de pulso, PMW (*Pulse Modulating Whith*). Dicha modulación será encargada al microcontrolador, donde el puerto B se ocupará de las patas del flanco izquierdo y el puerto C de las patas del flanco derecho. Habiendo 3 patas por cada flanco y con dos articulaciones por cada una de ellas, el microcontrolador será capaz de dirigir 12 Servos con sus 12 correspondientes salidas independientes de modulación por ancho de pulso; de esta forma definimos así los pines del microcontrolador:

PIN	FUNCIÓN	PIN	FUNCIÓN
1	Entrada Master Clear.	15	Hombro derecho 3.
2	Entrada analógica LDR1.	16	Codo derecho 3.
3	Entrada analógica LDR3.	17	Entrada modo.
4	Entrada analógica LDR3.	18	Entrada modo.
5	Entrada analógica Infrarrojos Izq.	19	GND.
6	Entrada/Salida auxiliar digital.	20	VCC.
7	Entrada analógica Infrarrojos Der.	21	Hombro izquierdo 1.
8	GND.	22	Codo izquierdo 1.
9	Señal de reloj 1.	23	Hombro izquierdo 2.
10	Señal de reloj 2.	24	Codo izquierdo 2.
11	Hombro derecho 1.	25	Hombro izquierdo 3.
12	Codo derecho 1.	26	Codo izquierdo 3.
13	Hombro derecho 2.	27	Control ojo izquierdo.
14	Codo derecho 2.	28	Control ojo derecho.

Los pines 27 y 28 son usados para indicar el estado de los LEDs bicolor de los ojos, un 1 lógico en dichos pines, indica la iluminación del LED verde, y un 0 lógico la iluminación del LED rojo, el control se complementa con el inversor de disparador de Smicht.

BIBLIOGRAFÍA

Referencias bibliográficas

Listado de referencias a libros:

[1] "Microcontroladores PIC. Diseño práctico de aplicaciones". José Mª Angulo Usategui, Ignacio Angulo Martínez. Ed. McGrawHill.

[2] "Microcontroladores PIC". Christian Tavernier. Ed. Paraninfo.

[3] Manual de uso del compilador "CCS PIC C".

[4] Datasheets de los componentes

Listado de referencias a artículos:

[5] Habib, M.K.; Watanabe, K.; Izumi, K.
"Biomimetics Robots From Bio-inspiration to Implementation".
The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON). Nov. 5-8, 2007, Taipei, Taiwan. Pg 143-148

[6] Andrew Conn, Stuart Burgess, Rick Hyde and Chung Seng Ling.
"From Natural Flyers to the Mechanical Realization of a Flapping Wing Micro Air Vehicle". International Conference on Robotics and Biomimetics. December 17 - 20, 2006, Kunming, China. Pg 439-444

[7] Masaki Takahashi, Member, IEEE, Takafumi Suzuki, Francesco Cinquegrani, Rosario Sorbello, Member, IEEE, and Enrico Pagello, Member, IEEE.
"A Mobile Robot for Transport Applications in Hospital Domain with Safe Human Detection Algorithm ".International Conference on Robotics and Biomimetics. December 19 -23, 2009, Guilin, China. Pg 1543-1548

[8] Paolo Arena, Member, IEEE, Luigi Fortuna, Member, IEEE, and Marco Branciforte. "Reaction–Diffusion CNN Algorithms to Generate and Control Artificial Locomotion". IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: FUNDAMENTAL THEORY AND APPLICATIONS, VOL. 46. 2, FEBRUARY 1999

[9] Kosuge, K.; Hirata, Y.
"Human-Robot Interaction".
International Conference on Robotics and Biomimetics. August 22 - 26, 2004, Shenyang, China.

[10] Vehículos autónomos.
http://alt1040.com/2011/10/inteligencia-artificial-vehiculos-autonomos

[11] Robots entre nosotros' sube-escaleras.
http://www.youtube.com/watch?v=3I_KoCeAdEw

[12] Robots entre nosotros' aspiradora.
http://www.youtube.com/watch?v=XG1bmCWnl70

[13] Robots entre nosotros' cirujano.
http://www.youtube.com/watch?v=peNjTQh0avg

[14] Robots entre nosotros' universo.
http://www.youtube.com/watch?v=BudlaGh1A0o

[15] Microcontroladores'01.
http://www.monografias.com/trabajos12/microco/microco.shtml#APLICAC

Los artículos correspondientes a IEEE Explorer, se adjuntan en la carpeta ***"Articulos"***

Listado de referencias a direcciones URL:

[16] Microchip.
www.microchip.com

[17] Programador.
http://www.ebay.es/itm/PIC-USB-Microcontroller-Development-programador-Programmer-ICSP-k150-/180739295799?pt=LH_DefaultDomain_186&hash=item2a14e6ca37#ht_5584wt_905

[18] Compilador.
http://www.ccsinfo.com/content.php?page=compilers

[19] Simulador.
http://www.marcombo.com/Descargas/9788426714954-COMPILADOR%20C%20CCS%20Y%20SIMULADOR%20PROTEUS%20PARA%20MICROCONTROLADORES%20PIC/descargar_primer_capitulo_libro_compiladorccs_simulador_proteus.pdf

[20] Autocad de Autodesk.
http://www.autodesk.es/adsk/servlet/pc/index?siteID=455755&id=14626579

[21] Diseño circuitos.
http://www.cadence.com/products/orcad/orcad_pcb_designer/pages/default.aspx

[22] Caracteristicas'Baterías.
http://www.tiposde.org/cotidianos/420-tipos-de-baterias/

[15] Página web de datasheets.
www.alldatasheet.com/

[16] Página web de RS.
www.amidata.com/

****

[17] Caracteristicas'Servos.
En el capítulo de planos.

ANEXO CÓDIGO

```
#include <16F876A.h> //Esto nos crea un archivo donde se especifican ciertas variables y comandos usados por el programa.
#device adc=10

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage progming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected
#FUSES RESERVED                 //Used to set the reserved FUSE bits
#USE delay(clock=4000000)       //Tipo de reloj usado (4 MHz)
#USE  FAST_IO(b)      //Seremos nosotros quien establezcamos las patillas de entrada salida
#USE  FAST_IO(c)      //Seremos nosotros quien establezcamos las patillas de entrada salida


Stand_Pose(long t0);   //t0= segundos que queramos que permanezca el robot en posición de pie
Andar();
Derecha();
Izquierda();
Izquierda_Tanque();
Derecha_Tanque();

void main()
{

  setup_adc_ports(ALL_ANALOG);              //Establece los puertos que van a ser ADC
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);        //Deshabilitados timers...
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);


  set_tris_b(0b00000000);                 //Puerto B todo salidas.
  set_tris_c(0b11000000);   //Puerto C todo salidas menos el C6 y C7 para seleccionar el modo.
  output_b(0b00000000);                   //Nos aseguramos que las salidas están a 0..
  output_c(0b00000000);                   //

  long lectura1, lectura2, lectura3,lectura4, lectura5, resta;
  long offset1=0;                         //factor de corrección para la LDR 1
  long offset2=0;                         //factor de corrección para la LDR 2
  long offset3=0;                         //factor de corrección para la LDR 3
  long offset4=0;                         //factor de corrección para infrarrojo izquierdo
  long offset5=0;                         //factor de corrección para infrarrojo derecho
```

```
  long valorDisparo1=300;//rango de tensión para  giro brusco o suave flanco izquierdo.
  long valorDisparo2=300;  //rango de tensión para giro brusco o suave, flanco derecha
  long objetoIzquierda=200;
  long objetoDerecha=200;

  delay_ms(2000);
  Stand_Pose(2); //El robot se levanta durante 2 segundos antes de iniciar el programa.

  inicio:

//*****************************************************************************
  //Tabla de entradas --> funciones
  //   C6        C7      función
  //
  //   0         0       esquivar objetos
  //   0         1       buscar luz
  //   1         0       busacar luz & esquivar objetos
  //   1         1       aplicación futura: control remoto mediante un mando a distancia

//*****************************************************************************

//        LÓGICA DE ACTIVACIÓN DE LOS MODOS DE FUNCIONAMIENTO
  // Se activará buscar luz cuando sean diferentes las entradas, es decir es una funcion XOR entre C6 y C7
  // Se activara esquivar objetos siempre que C7 sea 0



  //**************ADQUISICION DE DATOS POR LOS SENSORES************
  set_adc_channel(0);                   //La lectura analógica se va a realizar en el puerto 0.
  delay_us(20);
  lectura1=read_adc() + offset1;   //lectura1 valor del sensor 1 mas el offset ajustado.

  set_adc_channel(1);                   //La lectura analógica se va a realizar en el puerto 1.
  delay_us(20);
  lectura3=read_adc() + offset3;              //lectura3 valor del sensor 3 mas el offset ajustado.

  set_adc_channel(2);                   //La lectura analógica se va a realizar en el puerto 2.
  delay_us(20);
  lectura2=read_adc() + offset2;              //lectura2 valor del sensor 2 mas el offset ajustado.

  set_adc_channel(3);                   //La lectura analógica se va a realizar en el puerto 3.
  delay_us(20);
  lectura4=read_adc() + offset4;//lectura4 valor del sensor 4 mas el offset ajustado.

  set_adc_channel(4);                   //La lectura analógica se va a realizar en el puerto 4.
  delay_us(20);
  lectura5=read_adc() + offset5;              //lectura5 valor del sensor 5 mas el offset ajustado.

  delay_us(20);
```

```
//*****************************************************************************
   if(!input(PIN_C6)==1){
//**********************LÓGICASENSORES INFRARROJOS**********************

  if (lectura4> objetoIzquierda){//Objeto más cerca de lo establecido, flanco Izquierdo
    Derecha_Tanque();
    Andar();
    goto  inicio;}

  if (lectura5> objetoDerecha){       //Objeto más cerca de lo establecido, flanco Derecho
    Izquierda_Tanque();
    Andar();
    goto  inicio;}          }

//*****************************************************************************
  if ((!input(PIN_C6) && input(PIN_C7))||(input(PIN_C6) && !input(PIN_C7))==1){
  //*****************LÓGICASENSORES LUMÍNICOS*****************************
  if(lectura1>lectura3 && lectura2>lectura3){       //luz en la izquierda
    if(lectura2>lectura1){
      Andar ();               //adelante
      goto inicio;}
    else{
      resta=lectura1-lectura2;
      if(resta>valorDisparo1){
        Izquierda_Tanque ();          //tanque izquierda
        goto inicio;}
      else{
        Izquierda ();          //gira izquierda
        goto inicio;}
    }    }
  if(lectura3>lectura1 && lectura2>lectura1){       //luz en la derecha
    if(lectura2>lectura3){
      Andar ();               //adelante
      goto inicio;}
    else{
      resta=lectura3-lectura2;
      if(resta>valorDisparo2){
        Derecha_Tanque ();          //tanque derecha
        goto inicio;}
      else{
        Derecha ();          //gira derecha
        goto inicio;}
    }    }
  else{
    if(lectura1>lectura3){                    //Luz por atras a la izquirda
      Izquierda_Tanque ();              //tanque izquierda
      goto inicio;}
    else{                                 //Luz por detras a la derecha
      Derecha_Tanque ();                //tanque derecha
      goto  inicio;}
  }  }
//*****************************************************************************
  Andar();
  goto inicio;}       //FIN DE PROGRAMA
```

```
//*****************************SUBRUTINAS*******************************
Stand_Pose(long t0){                    //Establecemos la función que hace quedarse al robot en pie.
  int i;
  long cuenta=t0/0.02;        //El tiempo se introduce en segundos, conversión para su procesado.

      for(i=0;i<=cuenta;i++){
        output_b(0b11111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b11110101); //Se posiciona a 50º
        output_low(PIN_C5);    //Se posiciona a 50º
        delay_us(470);
        output_b(0b11100000); // Se posicona a 90º
        output_c(0b00001010); // Se posicona a 90º
        delay_us(480);
        output_c(0b00000000);             //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);       }            }


Andar () {                          //Establecemos la función que hace andar hacia adelante al robot.
  int i;
  int t1=7;
      for(i=0;i<=t1;i++){
        output_b(0b11111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b11110101); //Se posiciona a 50º
        output_low(PIN_C5);    //Se posiciona a 50º
        delay_us(120);
        output_low(PIN_B2);    //Se posiciona a 60º
        output_low(PIN_C2);    //Se posiciona a 60º
        delay_us(710);
        output_b(0b11100000);             //Se posiciona a 120º
        output_c(0b00001010);              //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);              //Se posiciona a 130º
        output_low(PIN_B5);     //Se posiciona a 130º
        delay_us(18050);
      }


      for(i=0;i<=t1;i++){
        output_b(0b11111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b11111101); //Se posiciona a 50º
        delay_us(470);
        output_b(0b11100000);             //Se posiciona a 90º
        output_c(0b00001000); //Se posiciona a 90º
        delay_us(480);
        output_low(PIN_C3);    //Se posiciona a 130º
        output_low(pin_B5);     //Se posiciona a 130º
        delay_us(18050);}
      for(i=0;i<=t1;i++){
```

```
      output_b(0b11111111);
      output_c(0b00111111);
      delay_us(1000);
      output_b(0b11110101); //Se posiciona a 50º
      output_low(PIN_C5);    //Se posiciona a 50º
      delay_us(120);
      output_b(0b11100100);          //Se posiciona a 60º
      output_c(0b00001110);          //Se posiciona a 60º
      delay_us(710);
      output_low(PIN_B2);     //Se posiciona a 120º
      output_low(PIN_C2);    //Se posiciona a 120º
      delay_us(120);
      output_c(0b00000000);          //Se posiciona a 130º
      output_low(PIN_B5);    //Se posiciona a 130º
      delay_us(18050);
    }

    for(i=0;i<=t1;i++){
      output_b(0b11111111);
      output_c(0b00111111);
      delay_us(1000);
      output_low(PIN_B3);    //Se posiciona a 50º
      output_low(PIN_C5);    //Se posiciona a 50º
      delay_us(470);
      output_b(0b11000000);  //Se posiciona a 90º
      output_c(0b00000010);          //Se posiciona a 90º
      delay_us(480);
      output_c(0b00000000);          //Se posiciona a 130º
      delay_us(18050);
    }
}


Derecha (){                          //Establecemos la función que hace girar a la derecha al robot.
  int i;
  int t3=7;

    for(i=0;i<=t3;i++){
      output_b(0b01111111);
      output_c(0b00111111);
      delay_us(1000);
      output_b(0b01110101); //Se posiciona a 50º
      output_low(PIN_C5);
      delay_us(120);
      output_low(PIN_C2);    //Se posiciona a 60º
      output_low(PIN_B2);    //Se posiciona a 60º
      delay_us(350);
      output_c(0b00001010); // Se posicona a 90º
      delay_us(360);
      output_b(0b01100000);          //Se posiciona a 120º
      delay_us(120);
      output_c(0b00000000);          //Se posiciona a 130º
      output_low(PIN_B5);
      delay_us(18050);     }
```

```
for(i=0;i<=t3;i++){
          output_b(0b01111111);
          output_c(0b00111111);
          delay_us(1000);
          output_b(0b01111101); //Se posiciona a 50º
          delay_us(290);
          output_c(0b00101010);          //Se posiciona a 75º
          delay_us(180);
          output_b(0b01100000);          //Se posiciona a 90º
          output_c(0b00001000);          //Se posiciona a 90º
          delay_us(480);
          output_low(PIN_C3);    //Se posiciona a 130º
          output_low(PIN_B5);
          delay_us(18050);
        }
        for(i=0;i<=t3;i++){
          output_b(0b01111111);
          output_c(0b00111111);
          delay_us(1000);
          output_b(0b01110101); //Se posiciona a 50º
          output_low(PIN_C5);
          delay_us(120);
          output_b(0b01100100);          //Se posiciona a 60º
          output_c(0b00001110);          //Se posiciona a 60º
          delay_us(350);
          output_low(PIN_C2);    //Se posiciona a 90º
          delay_us(360);
          output_b(0b01100000);          //Se posiciona a 120º
          delay_us(120);
          output_c(0b00000000);          //Se posiciona a 130º
          output_low(PIN_B5);
          delay_us(18050);
        }

        for(i=0;i<=t3;i++){
          output_b(0b01111111);
          output_c(0b00111111);
          delay_us(1000);
          output_low(PIN_B3);    //Se posiciona a 50º
          output_low(pin_c5);
          delay_us(290);
          output_c(0b00001010);          //Se posiciona a 75º
          delay_us(180);
          output_b(0b01000000);          //Se posiciona a 90º
          output_low(PIN_C3);    //Se posiciona a 90º
          delay_us(480);
          output_c(0b00000000);          //Se posiciona a 130º
          delay_us(18050);
           }
}
```

```
Izquierda () {                    //Establecemos la función que hace girar a la izquierda al robot.
   int i;
   int t2=7;

      for(i=0;i<=t2;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b10110101); //Se posiciona a 50º
        output_low(PIN_C5);
        delay_us(120);
        output_low(PIN_C2);    //Se posiciona a 60º
        delay_us(350);
        output_low(PIN_B2);            // Se posicona a 90º
        delay_us(360);
        output_b(0b10100000);          //Se posiciona a 120º
        output_c(0b00001010);          //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);          //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);
      }

      for(i=0;i<=t2;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b10111101); //Se posiciona a 50º
        delay_us(470);
        output_low(PIN_B3);    //Se posiciona a 90º
        output_c(0b00001000);          /Se posiciona a 90º
        delay_us(180);
        output_b(0b10100000);          //Se posiciona a 105º
        delay_us(300);
        output_low(PIN_C3);    //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);}

      for(i=0;i<=t2;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b10110101); //Se posiciona a 50º
        output_low(PIN_C5);
        delay_us(120);
        output_c(0b00001110);          //Se posiciona a 60º
        delay_us(350);
        output_b(0b10100100);          //Se posiciona a 90º
        delay_us(360);
        output_low(PIN_B2);    //Se posiciona a 120º
        output_low(PIN_C2);    //Se posiciona a 120º
        delay_us(120);
        output_c(0b00000000);          //Se posiciona a 130º
        output_low(PIN_B5);
        delay_us(18050);}
```

```
      for(i=0;i<=t2;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_low(PIN_B3);    //Se posiciona a 50º
        output_low(PIN_C5);
        delay_us(470);
        output_b(0b10010101);          //Se posiciona a 90º
        output_c(0b00000010);          //Se posiciona a 90º
        delay_us(180);
        output_b(0b10000000);          //Se posiciona a 105º
        delay_us(300);
        output_c(0b00000000);          //Se posiciona a 130º
        delay_us(18050);}


Izquierda_Tanque () {                  //Establecemos la función que hace girar al robot en modo
tanque a la izquirda.
  int i;
  int t4=7;
      for(i=0;i<=t4;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_b(0b00110101); //Se posiciona a 50º
        output_low(PIN_C5);    //Se posiciona a 50º
        delay_us(120);
        output_b(0b10100100);          //Se posiciona a 60º
        output_low(PIN_C2);    //Se posiciona a 60º
        delay_us(710);
        output_b(0b00100000);          //Se posiciona a 120º
        output_c(0b00001010);          //Se posiciona a 120º
        delay_us(120);
        output_low(PIN_B5);    //Se posiciona a 130º
        output_c(0b00000000);          //Se posiciona a 130º
        delay_us(18050);
      }


      for(i=0;i<=t4;i++){
        output_b(0b10111111);
        output_c(0b00111111);
        delay_us(1000);
        output_low(PIN_B1);    //Se posiciona a 50º
        delay_us(470);
        output_b(0b00100000);          //Se posiciona a 90º
        output_c(0b00001000);          //Se posiciona a 90º
        delay_us(480);
        output_low(PIN_B5);    //Se posiciona a 130º
        output_low(PIN_C3);    //Se posiciona a 130º
        delay_us(18050);
      }
```

```
      for(i=0;i<=t4;i++){
      output_b(0b10111111);
      output_c(0b00111111);
      delay_us(1000);
      output_b(0b00110101); //Se posiciona a 50º
      output_low(PIN_C5);   //Se posiciona a 50º
      delay_us(120);
      output_low(PIN_B2);   //Se posiciona a 60º
      output_c(0b00001110);         //Se posiciona a 60º
      delay_us(710);
      output_b(0b10100000);         //Se posiciona a 120º
      output_low(PIN_C2);   //Se posiciona a 120º
      delay_us(120);
      output_c(0b00000000);         //Se posiciona a 130º
      output_low(PIN_B5);           //Se posiciona a 130º
      delay_us(18050);
    }
    for(i=0;i<=t4;i++){
      output_b(0b10111111);
      output_c(0b00111111);
      delay_us(1000);
      output_low(PIN_B3);   //Se posiciona a 50º
      output_low(PIN_C5);   //Se posiciona a 50º
      delay_us(470);
      output_b(0b00000000);  //Se posiciona a 90º
      output_c(0b00000010);         //Se posiciona a 90º
      delay_us(480);
      output_low(PIN_C1);   //Se posiciona a 130º
      delay_us(18050);
    }
}


Derecha_Tanque () {           //Establecemos la función que hace andar hacia adelante al robot.
  int i;
  int t5=7;
    for(i=0;i<=t5;i++){
      output_b(0b01111111);
      output_c(0b00111111);
      delay_us(1000);
      output_b(0b00110101); //Se posiciona a 50º
      output_low(PIN_C5);   //Se posiciona a 50º
      delay_us(120);
      output_low(PIN_B2);   //Se posiciona a 60º
      output_c(0b00001110);         //Se posiciona a 60º
      delay_us(710);
      output_b(0b01100000);          //Se posiciona a 120º
      output_low(PIN_C2);    //Se posiciona a 120º
      delay_us(120);
      output_low(PIN_B5);   //Se posiciona a 130º
      output_c(0b00000000);         //Se posiciona a 130º
      delay_us(18050);
    }
```

```
    for(i=0;i<=t5;i++){
      output_b(0b01111111);
      output_c(0b00111111);
      delay_us(1000);
      output_b(0b00111101); //Se posiciona a 50º
      delay_us(470);
      output_b(0b11100000);          //Se posiciona a 90º
      output_c(0b00001000);          //Se posiciona a 90º
      delay_us(480);
      output_low(pin_B5);    //Se posiciona a 130º
      output_low(PIN_C3);    //Se posiciona a 130º
      delay_us(18050);
    }


    for(i=0;i<=t5;i++){
      output_b(0b01111111);
      output_c(0b00111111);
      delay_us(1000);
      output_b(0b00110101); //Se posiciona a 50º
      output_low(PIN_C5);    //Se posiciona a 50º
      delay_us(120);
      output_b(0b01100100);          //Se posiciona a 60º
      output_low(PIN_C2);    //Se posiciona a 60º
      delay_us(710);
      output_low(PIN_B2);     //Se posiciona a 120º
      output_c(0b00001010);          //Se posiciona a 120º
      delay_us(120);
      output_low(PIN_B5);            //Se posiciona a 130º
      output_c(0b00000000);           //Se posiciona a 130º

      delay_us(18050);
    }

    for(i=0;i<=t5;i++){
      output_b(0b01111111);
      output_c(0b00111111);
      delay_us(1000);
      output_low(PIN_B3);    //Se posiciona a 50º
      output_low(PIN_C5);    //Se posiciona a 50º
      delay_us(470);
      output_b(0b00000000);          //Se posiciona a 90º
      output_c(0b00000010);          //Se posiciona a 90º
      delay_us(480);
      output_low(PIN_C1);    //Se posiciona a 130º
      delay_us(18050);
    }
}
```