Proyecto integrado

Ciclo grado superior en desarrollo de proyectos de productos electrónicos

INSTITUTO: I.E.S. POLITÉCNICO "JESÚS MARÍN"
ALUMNO: JORGE GRANDA ALONSO
CURSO: 2008/2010

Memoria

Planos

Pliego de condiciones.

Presupuesto.

Memoria

1. PREÁMBULO.

1.1 Objetivos.

Dado el carácter académico del proyecto integrado dentro del ciclo de grado superior de desarrollo de proyectos de productos electrónicos, con este mismo se pretenden alcanzar los siguientes objetivos o capacidades terminales:

-Idear soluciones técnicas de aplicaciones electrónicas (analógicas, digitales y microprogramables). A partir de las especificaciones funcionales, utilizando la documentación técnica y/o base de datos de soluciones estándar disponible, seleccionando los componentes y materiales de fiabilidad y coste requerido y establecido.

-Elaborar un programa de control para el dispositivo microprogramable de la aplicación utilizando el lenguaje adecuado y aplicando las técnicas de programación más adecuadas.

-Realizar el montaje del prototipo, simulación y diseño por ordenador de los circuitos y módulos que componen la aplicación, utilizando los medios disponibles y aplicando los conocimientos adecuados. Determinar con precisión las pruebas que se han de realizar en el prototipo (estáticas, funcionales, de fiabilidad y calidad) teniendo en cuenta el tipo de aplicación y los medios disponibles.

-Documentar técnicamente el proyecto de una aplicación electrónica de creación de robot luchador de minisumo para campeonato de MADRIDBOT 2010, incluyendo los planos, lista de materiales, programas debidamente comentados, cálculos, pruebas, y ajustes y demás elementos necesarios para la construcción del prototipo correspondiente a la aplicación electrónica que se desarrolla en este caso.

-Realizar la planificación y gestión del proyecto correspondiente a una aplicación electrónica, realizando la previsión de tiempos y costos, y coordinando las distintas fases establecidas para su adecuado desarrollo.

1.2 ASPECTOS A TENER EN CUENTA.

Al tratarse de un proyecto de carácter técnico-didáctico, los puntos a tener en cuenta son muchos, destacando los siguientes:

-Estudio de las características de los componentes en relación calidad/precio.

-Estudio del manejo del software de programación y simulación.

-Correcto diseño y configuración de los programas.

-Correcto diseño de planos.

-Realización de simulaciones para depurar errores.

-Creación de un diseño teniendo en cuenta las características necesarias de nuestro circuito de control.

-Minimizar el uso de puentes en el diseño.

-Ajustar el mini robot a los requerimientos y normativas específicas de dicha categoría.

-Reducir la placa todo lo posible.

-Asegurar una buena respuesta del mini robot tanto en ataque como en defensa.

-Buscar los materiales adecuados para el prototipo del chasis.

-Seleccionar el diseño más competitivo dado el carácter de este proyecto.

1.4 DATOS DE PARTIDA.

Para comenzar con la realización de este proyecto debemos tener en cuenta el fin para el cual se va a crear, que no es sino una competición de robótica, en la cual se pueden desarrollar varios tipos de diseño de mini robot dependiendo de la modalidad escogida; en este caso escogimos la categoría de “minisumo”, debido a que fue la que más nos llamo la atención en cuanto a competitividad y programación. Como en cada categoría, hay una serie de normativas y reglas que nos van a condicionar el diseño y explicaremos más adelante.

Para realizar este mini robot debemos basarnos en el uso de los microcontroladores PIC y los diferentes periféricos de E/S que necesitemos conectar, para ello debemos crear un software de control en lenguaje C que realizaremos mediante el editor de textos MPLAB y compilador CCS, este software deberá adaptarse al uso de los sensores que decidamos poner en nuestro mini robot, en nuestro caso serán sensores de distancia por infrarrojos y sensores fotosensibles activos. En este programa también tendremos que tener en cuenta los motores y el tipo que se use (de corriente continua ó servomotores trucados), para este diseño optaremos por los servomotores debido a su tamaño compacto y al torque que tienen.

En cuanto al chasis, tenemos total libertad para el uso de cualquier tipo de material que sea útil para nuestro diseño, en el caso de este proyecto se utilizan materiales “reciclados” para conseguir las máximas prestaciones de diseño al menor precio posible.

Como antes comentábamos, unos de los datos de partida más importantes son las normativas y reglas a las que se tiene que ajustar nuestro diseño y las cuales, debido a que este diseño es para la modalidad de minisumo son las siguientes:

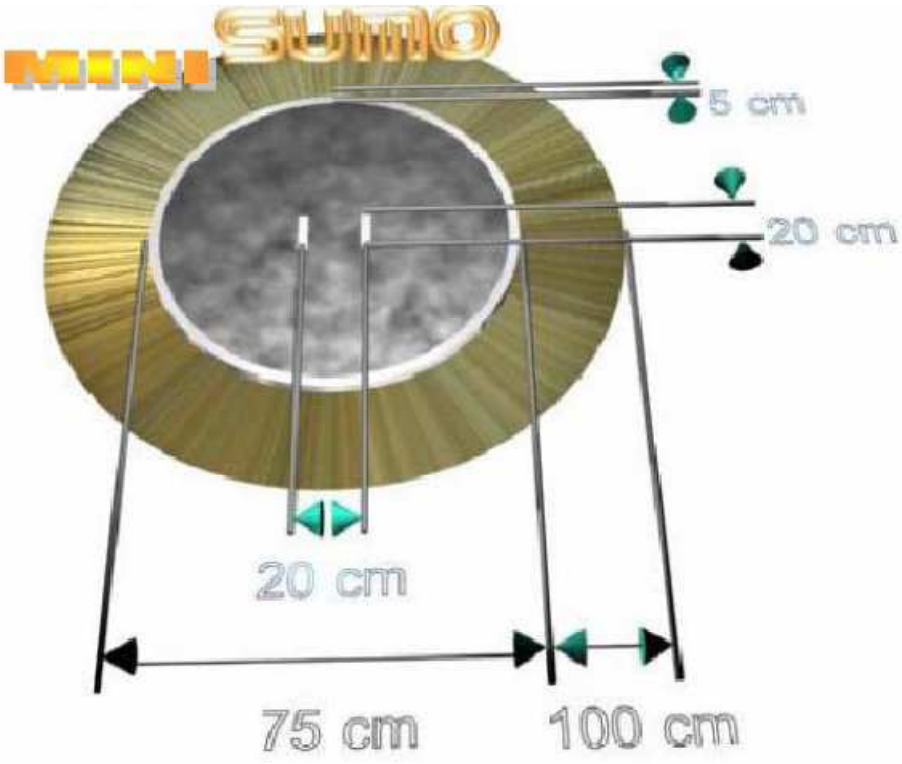
-EN CUANTO AL MINI ROBOT:

1. Los Robots deberán tener unas dimensiones tales que quepan dentro de una caja de 10x10 cm. con los sensores de contacto (micro interruptores) plegados en caso que se utilicen.
2. La altura podrá ser cualquiera. No se permite diseñar el mini robot de forma que cuando empiece el juego se separe en diferentes piezas, es decir, en 2 o más robots independientes o dejando piezas desperdigadas; el Robot que lo haga, perderá el combate. Sí que se permite desplegar estructuras una vez iniciado el combate SIEMPRE Y CUANDO AL FINALIZAR EL TIEMPO DEL COMBATE EL ROBOT SEA CAPAZ DE RECOJERLAS AUTONOMAMENTE DE MODO QUE **TIENE QUE MEDIR UN MAXIMO DE 10x10 UNA VEZ FINALIZADO.**
3. El peso máximo de los Robots será de 500 gramos incluyendo todas las partes.
4. Los Robots habrán de ser completamente autónomos, no permitiéndose comunicación alguna con el exterior, ni por parte de alimentación ni de su radiocontrol.
5. Los robots deberán diseñarse de forma que comiencen a moverse una vez pasados 5 segundos desde la activación de los mismos.

-En cuanto al ring o recinto de combate o dohyo:

1. El Ring será circular, de color negro, de 75 cm. de diámetro y situado a una altura de 5 cm. respecto al suelo.
2. Señalando el límite exterior del Ring, habrá una línea blanca circular de 5 cm. De ancho.
3. La tolerancia de todas las medidas indicadas anteriormente será del ±5 %.
4. Habrá como mínimo 1 m alrededor del Ring, el que seguirá vacío de cualquier obstáculo durante los combates. Este espacio puede ser de cualquier color excepto blanco.

Detalle de la apariencia física del ring o recinto de combate:



1.5 Especificaciones del proyecto.

-Tenemos que realizar un estudio del funcionamiento del microcontrolador PIC16F876/7A, conocer sus características, como se programa y qué condiciones son necesarias conocer.

-Realizamos la programación del mismo mediante la ayuda de software editor de texto MPLAB, teniendo en cuenta los requerimientos de nuestro diseño.

-Realizaremos un análisis de todos los componentes para llegar a la selección de los adecuados, para ello iremos probando las diferentes posibilidades mediante el uso de simuladores como el PROTEUS. De esta manera podremos encontrar los componentes de las mejores características para nuestro diseño, que sean flexibles y que sean los más económicos posibles.

-Diseñamos el hardware que deberá soportar nuestro software. Realizamos diseños de las PCBs que compone nuestro mini robot y haciendo atención en el diseño de los conectores exteriores para los bus de conexión entre placas, conectores de alimentación y programación, interruptores de encendido, etc. Usamos SPRINT-LAYOUT para el diseño.

-Se procede con la impresión del fotolito del diseño de pistas, teniendo en cuenta que el fotolito de pistas se coloca en negativo.

-Usando placa fotosensible positiva mono capa, insolamos la cara de componentes, quedando marcado el dibujo del fotolito en la parte cubierta de cobre y posteriormente se pasa al baño con ácidos.

-Con un baño en proporción de 25%-25%-50%, se revela la PCB en 5 minutos, haciendo atención en mover la placa durante el proceso para una mejor actuación de los ácidos.

-Al finalizar el baño de ácido, se enjuaga la PCB y se comprueban posibles fallos de calidad en el diseño.

-Realizamos los orificios para todos los componentes que lo necesiten y con el diámetro adecuado para no forzar el patillaje de los componentes al ser introducidos y viendo que nos quede superficie de soldadura donde aplicar la cantidad necesaria de estaño.

-Realizamos un control en la calidad del diseño, comprobando los contactos y posibles defectos tras la realización de los orificios.

-Montamos los componentes de manera que podamos ir probando el funcionamiento por bloques para depurar posibles fallos. Prestar atención con el aporte de estaño y en tiempos de soldadura para evitar soldaduras frías y otros posibles fallos.

-Realizamos montaje completo y realizamos pruebas reales de funcionamiento.

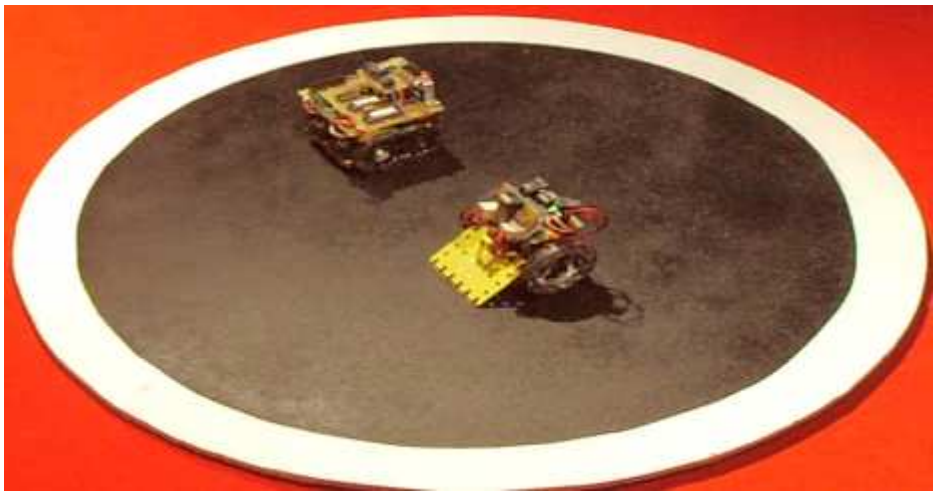
-Seleccionamos materiales adecuados para el montaje del chasis del robot y que se acoplen de la mejor manera posible junto con el sistema de tracción y las placas de control.

-Seleccionamos lugar para los sensores de manera más sencilla y eficaz, y preparamos el lugar con las sujeciones adecuadas.

-Ensamblamos todos los componentes al chasis y comenzamos con las pruebas de funcionamiento.

-Realizamos rectificaciones y mejoras necesarias para llegar al punto en que cubramos las necesidades requeridas y deseadas para nuestro diseño.

-Realizamos pruebas de funcionamiento en el tatami, como son permanecer en el tatami sin salirse, localizar objeto dentro del tatami, empujar objetos fuera del tatami, y prueba para comprobar cantidad de peso capaz de mover.



2. MEMORIA DESCRIPTIVA.

Para llevar a cabo este proyecto debemos tener unas nociones y conocimientos mínimos que se han debido adquirir a lo largo del curso académico y los cuales nos van a ser necesarios para seguir un procedimiento adecuado, bien planificado y programado dentro del tiempo que dura nuestro curso. Se ha de estudiar la viabilidad de nuestro sistema, tenemos que saber realizar los cálculos necesarios para la compensación de los circuitos, se han de realizar los diseños de prueba pertinentes y simularlos para comprobar posibles fallos, para posteriormente llevar nuestro proyecto a un nivel físico. Se debe conocer también el método de programación en C, ya que el sistema se gobierna con programación de este tipo mediante el uso de editores de texto y compiladores de los que se debe tener manejo.

2.1 SOFTWARE.

Para realizar el programa principal de nuestro mini robot utilizamos programación en lenguaje tipo C y el programa MPLAB. Con este editor de texto es con el que vamos a realizar el programa, y el proceso de cómo llevarlo a cabo se explica más adelante (**anexo 5.1**). Nuestro diseño parte de un programa que necesitamos para gobernar los diferentes periféricos del mini robot como son:

- **Leds** (que tenemos dos y usaremos a libre elección ya que nos son obligatoriamente necesarios para un correcto funcionamiento), **motores** (que al ser servomotores tienen una programación más compleja que los motores de continua) y **sensores** (que tenemos de dos tipos diferentes, los fotosensores activos y sensores de infrarrojos).

En primer lugar probamos el diseño de la placa creando un programa para comprobar el correcto funcionamiento antes de crear el algoritmo completo del robot, para ello realizaremos un programa de encendido y apagado de uno o los dos leds (**anexo 5.3**). Una vez que ese programa funcione correctamente en nuestro diseño, podemos corroborar que el diseño del circuito está correcto y podemos seguir con los siguientes programas más complejos.

El siguiente programa que vamos a realizar será el que gobierna el movimiento de nuestros dos servomotores (**anexo 5.4**) y es más complejo que para motores de continua, ya que se gobiernan mediante PWM (pulse width modulation),y al estar trucados, tenemos que indicarle al circuito electrónico del motor el punto o los grados que queremos que gire 0º-90º-180º, (a mayor grado mayor velocidad, ya que busca la posición más alejada a mayor velocidad sin llegar nunca al tope puesto que no está), marcando los grados de los extremos (0º-180ª) el motor girará al máximo de potencia en un sentido o en otro, si marcamos 90º, el motor se quedará parado. Y sabiendo esto lo tenemos que diseñar teniendo en cuenta que los motores se encuentran en posiciones opuestas y tendrán el giro opuesto; e ir creando el programa de manera que nuestros motores giren los dos en un mismo sentido (giro sobre si mismo izquierda/derecha), que los motores giren en sentidos opuestos (adelante/atrás), con estos cuatro movimientos se puede dotar al robot de total agilidad sobre el tatami.

El Programa se lleva a cabo con una serie de configuraciones iniciales y una instrucción llamada “set_pwmX_dutty ( )”, y el valor que tenemos que darle en este caso no es el de los grados, sino que hay que realizar una serie de cálculos que se adjuntan en el apartado de cálculos (**3.3**).

Ahora que tenemos el programa para los movimientos del robot, pasamos al siguiente paso que es realizar el programa para que entren en funcionamiento los fotosensores activos CNY70 (**anexo 5.5**). Este programa es más sencillo, ya que estos sensores son digitales y te entregan una señal digital pura y según lo hemos configurado con el PIC nos dará, al leer negro un “0” lógico y al leer blanco un “1” lógico. Sabiendo esto, es muy fácil realizar este programa utilizando una simple condición en la que según los valores o estados que elijamos para los sensores, el programa nos realice una u otra instrucción. Por ejemplo, si se sale por la parte delantera, leerán los sensores delanteros blanco y los traseros negro, es decir, los dos delanteros ofrecen señal digital 1,1; y los dos traseros señal digital 0,0; pues si se cumple esta condición damos al PIC la instrucción de retroceder. De esta manera completamos el programa con todas las combinaciones posibles que puedan tener los cuatro sensores y que veamos viables para nuestro diseño, para así introducir una instrucción que se ajuste a nuestra necesidad en cada caso.

Para la siguiente parte del programa necesitamos integrar los dos sensores por infrarrojos GP2D12 que nos entregan a los puertos del PIC una señal analógica, por lo que cuando estos detectan un objeto entre 0–80cm te ofrecen un rango tensión, el cual vamos a utilizar a la hora de programar para controlar el movimiento de nuestro robot según detecte con el sensor delantero o con el sensor trasero, según el valor que usemos en nuestro programa el robot reaccionara al encontrar el obstáculo más o menos cerca. Para realizar el programa es necesario escribir las configuraciones necesarias de los puertos analógicos a usar y de las conversiones de los valores y posteriormente las condiciones del estado de cada sensor para que realice una maniobra u otra. (**anexo 5.6**).

Una vez que tenemos una idea de cómo se realiza la programación de cada dispositivo solo tenemos que proceder a interrelacionar todos los sensores con el movimiento de los motores mediante el uso de las condiciones e instrucciones que sean necesarias para dar con el resultado deseado (**anexo 5.7**), de modo que el robot sea capaz de detectar cualquier objeto que se encuentre a su alrededor dentro de la distancia programada (a ser posible la más ajustada al diámetro del tatami de juego), para que cuando lo encuentre, si lo hay, trate de empujarlo hacia el borde del circulo, y lo eche fuera sin llegar a salirse del tatami el mismo.

El último detalle del programa es introducir un retardo inicial al comenzar el programa, de una duración de 5 segundos según el reglamento. Se puede realizar con una simple instrucción de retardo con la duración deseada.

Una vez compilado nuestro programa en MPLAB lo simulamos con PROTEUS para comprobar el buen funcionamiento del programa con el microcontrolador y los periféricos de entrada y de salida, si aún no tenemos el circuito montado físicamente para poder realizarlo en el prototipo.

El hardware se diseña mediante el uso de otro software llamado SPRINT-LAYOUT, que nos ayuda al diseño de las PCB, incluyendo en sus librerías todos los encapsulados necesarios para realizar un diseño correcto de nuestro prototipo, también nos deja realizar encapsulados editados por nosotros.

2.2 HARDWARE.

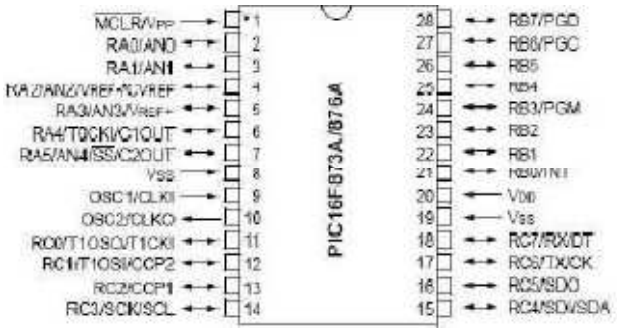
Para poder continuar con nuestro proyecto debemos crear un diseño hardware para nuestro software del minisumo antes diseñado. Por ello nos ayudaremos de los esquemas realizados para implementar nuestro diseño en una PCB, para esto nos ayudamos del software de diseño de PCB llamado SPRINT-LAYOUT, el que nos facilita el diseño de nuestro circuito y la creación de nuestro fotolito.

Para el diseño de este robot minisumo, debido a las normas en cuanto a un tamaño reducido y la disposición elegida vamos a realizar un diseño de PCB en dos partes, una PCB será la general y otra será la de los sensores digitales.

En la principal se ubican el microcontrolador y su circuito oscilador con cristal de cuarzo, todos los conectores de sensores analógicos y motores que van a los puertos directamente, así como los dos leds indicadores. También están el conector del bus de datos para conectar con la otra sección, los conectores para la alimentación de motores y control, (que usan tensiones diferentes y por lo tanto baterías diferentes para mayor autonomía y potencia), el conector RJ11 para poder programar el PIC de manera más cómoda, los interruptores de encendido, switches y pulsador de reset.

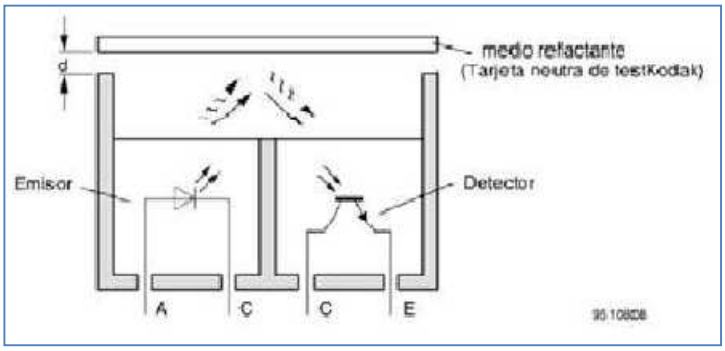
En la PCB de sensores tenemos el convertidor A/D Trigger Schmitt, los seis conectores para conectar sensores digitales (solo se usarán 4), el conector de bus de datos para conectar con la placa principal y resistencias limitadoras de corriente.

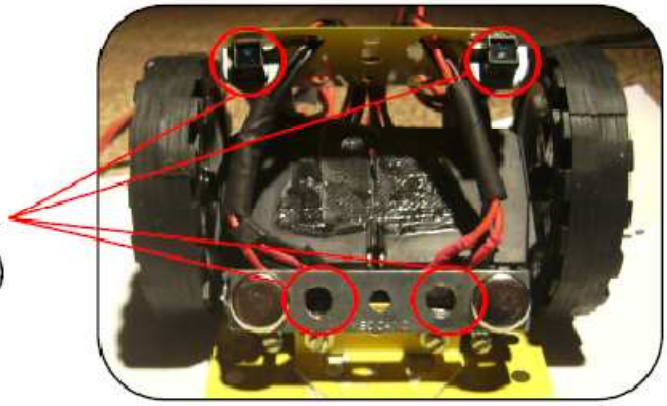
Para el control del hardware del minisumo hemos utilizado un microcontrolador PIC16F876A por su buena capacidad a la hora de gestionar programas, reducido tamaño y su bajo coste, y que consta de tres puertos de expansión PORTA, PORTB y PORTC, donde tenemos entradas suficientes donde conectar los periféricos analógicos y digitales de entrada/salida para que el minisumo trabaje de forma independiente; este PIC tiene entradas con conversores analógicos-digitales para poder recibir señales analógicas (AN0-AN4) y utilizarlas. También tiene salida para programación del PWM donde poder conectar nuestros servomotores (CCP1,CCP2).



Para que nuestro minisumo se mueva independientemente, sin salirse del tatami y localizando al oponente, usamos dos tipos de sensores:

Sensores CNY70 - Estos sensores son los encargados de evitar que el robot se salga del tatami. Son fotosensores activos y funcionan emitiendo luz por un fotodiodo y recibiendo la intensidad de la luz emitida que es reflejada por el suelo, con un fototransistor, dependiendo de la intensidad de la luz que recibe este componente, éste entrará en saturación si es la cantidad adecuada la que se refleja, es decir cuando es blanco; y entra en estado de corte cuando es negra la superficie que lea o sea una distancia muy corta. El sensor da una tensión correspondiente a unos niveles lógicos de 1 (blanco) ó 0 (negro), que son recibidos por los puertos del PIC y tratados por el software implementado en el mismo.
Las características son: Una construcción compacta, una señal de salida alta, un coeficiente de temperatura bajo, y una buena respuesta de trabajo en distancias de 5 a 10 mm.
Por eso es importante reservar un lugar cercano al suelo en la estructura de nuestro robot donde colocar estos sensores. En nuestro caso los colocaremos a una distancia de 5mm del suelo y dispuestos de manera que quedan dos sensores en la parte delantera del robot y dos en la parte trasera del robot, como se puede observar en la foto.





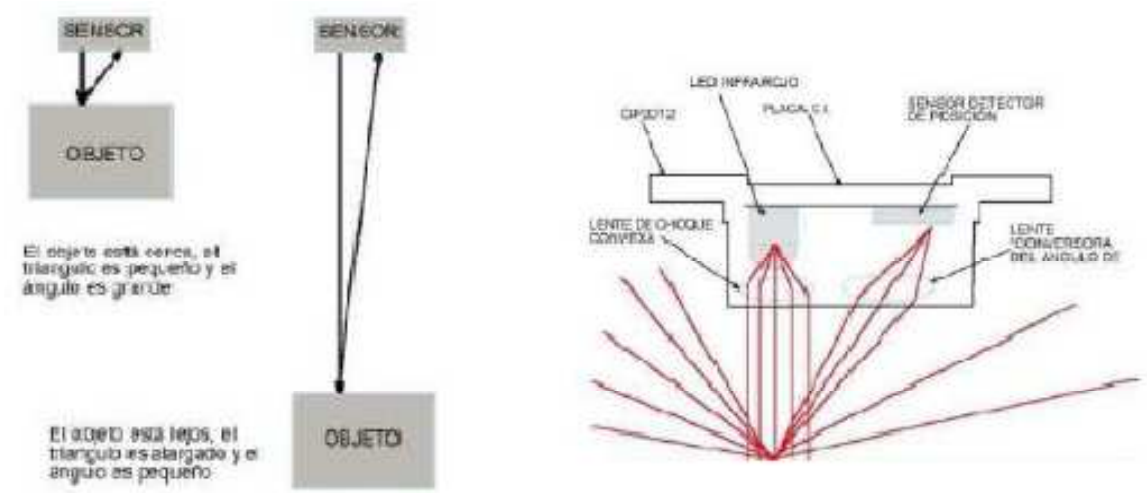
Como se puede apreciar los sensores quedan ajustados a la superficie del suelo lo máximo posible, pero sin que lleguen a tocar ya que podrían dar problemas.

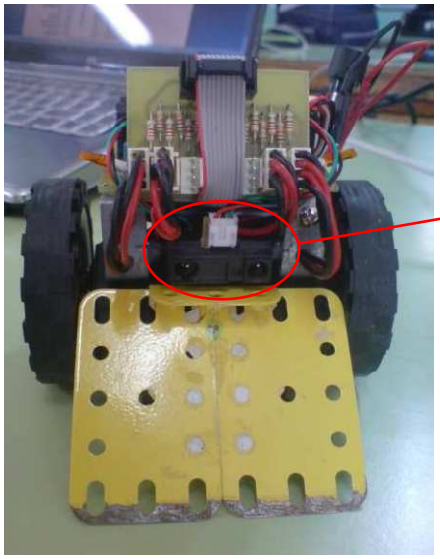
Para sujetar los sensores al chasis del robot usaremos silicona caliente o bien pegamento de dos componentes, la ventaja de usar la silicona caliente, a pesar de que es menos resistente, es que en caso de rotura de algún sensor, la sustitución por otro es más rápida y cómoda ya que es más fácil de quitar.

Sensores infrarrojos GP2D12- Son sensores que miden la distancia a través de la triangulación del haz de luz infrarroja que envía el emisor y del ángulo en que es recibido dicho haz en el otro extremo del sensor por el receptor. Ofrece según la distancia de entre 10-80cm unas tensiones de, entre 0 y 3V, y según la distancia a que este el objeto se recibirá una tensión dentro de este rango, a mayor distancia, menor voltaje; a menor distancia, mayor voltaje.

La información de la distancia se extrae midiendo el ángulo recibido. Si el ángulo es grande, el triangulo formado por el ángulo es ancho y el objeto esta cerca. Si el ángulo es pequeño, el el triangulo formado es estrecho y el objeto está lejos.

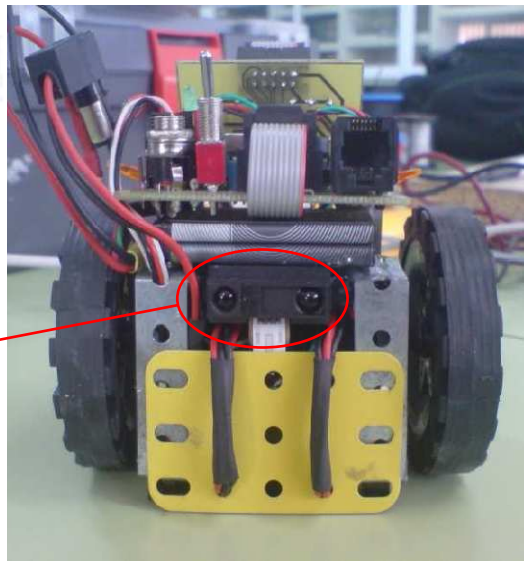
Estos sensores son los más adecuados para nuestra necesidad, además de su coste que no es elevado, tienen una buena respuesta y además el alcance máximo de estos sensores está por encima de nuestra necesidad que será de unos 75 cm, el diámetro del tatami de combate. Estos sensores los colocaremos en la parte delantera y trasera de nuestro robot para tener cubierto mayor espacio.



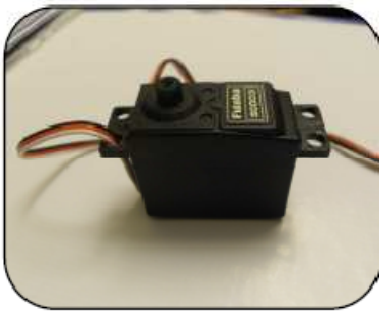


Sensor delantero

Sensor trasero



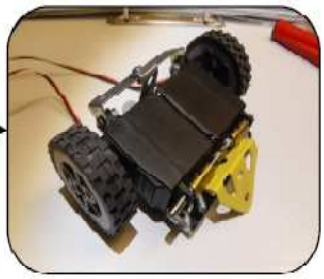
Servomotores Futaba S3003. Estos son los motores que dotarán de tracción a nuestro robot minisumo, este tipo de motores poseen una caja reductora integrada con gran torque, tienen el giro limitado en 180º y se activan por medio de PWM (pulse width modulation), según sea la duración de este pulso (0.3s-1.2s-2.4s) el motor se posiciona en un extremo(0º) o en otro(180º), con un pulso medio se posiciona en el centro(90º). Lo que no es compatible para nuestro diseño es que tiene el giro limitado, por lo que debemos "trucarlo" para que pueda girar los 360º completos (ver **anexo4.2**). Lo que si mantenemos es la reductora y la electrónica del servo para así poder tener un circuito más reducido, ya que si dejásemos el motor sin la electrónica que trae, sería un motor de corriente continua normal y corriente, y deberíamos conectarlo al microcontrolador por medio de un driver que nos diera más potencia haciendo mayor nuestra circuitería.



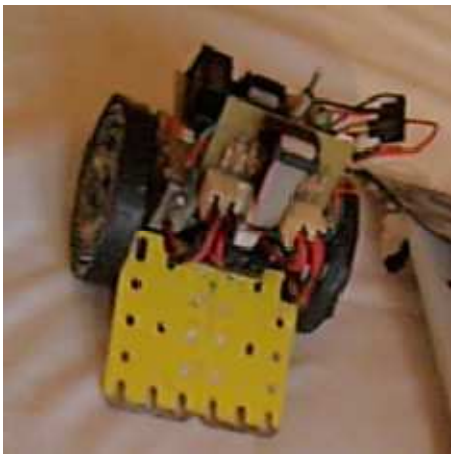
Además la carcasa de estos motores la vamos a poder usar como estructura base de nuestro minirobot, ensamblando estos dos en posiciones opuestas y mediante el uso de chapas metálicas obtenidas a partir de juegos de construcción podemos conseguir un chasis básico que no se sale de las medidas requeridas y es bastante robusto, y donde posteriormente mediante la ayuda de más chapas y plaquitas metálicas completaremos toda la estructura de nuestro robot según nos vaya pidiendo el diseño para poner los sensores, baterías, placas de control y sensores, rampa, etc.
Las medidas máximas que tenemos que respetar que son de 10x10cm, las debemos tener presentes en todo momento así como el peso máximo de 500gr, para no tener que desmontar ni cambiar demasiado el diseño cuando de forma inesperada.

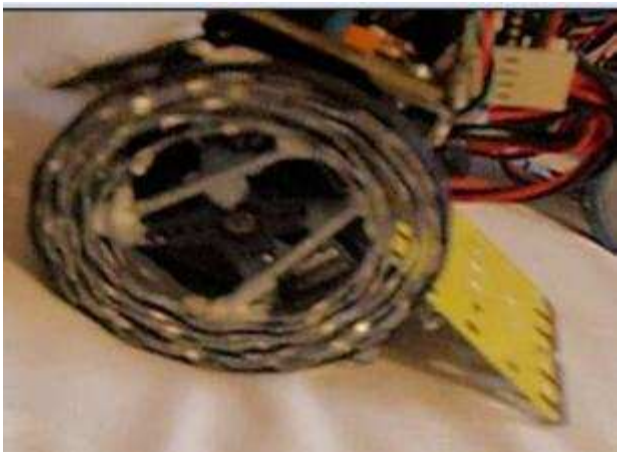
En el eje central de los servomotores colocaremos las ruedas que mejor se adapten a nuestro diseño en cuanto a diámetro, grosor y material de composición ya que si pesan más nos darán más tracción. Los servomotores traen un kit de varios adaptadores para el eje así que podremos utilizar que mejor nos venga para nuestro diseño.





En la siguiente foto se puede observar el detalle de las ruedas que se pusieron al minirobot en último momento, estas ruedas fueron fabricadas a partir de brida metálica enrrollada y compactada mediante pegamento de dos componentes. Se le introdujeron dos ejes metálicos donde poder acoplar el adaptador del eje de los servomotores. Con estas ruedas conseguimos ajustar las medidas de nuestro robot al milímetro y dotamos del mayor peso a las ruedas teniendo así mayor tracción y mayor peso del conjunto del robot, haciendo más robusto e inamovible nuestro minirobot dentro del tatami de combate.





Sistema de alimentación.

Para el diseño de nuestro robot minisumo optamos por la decisión de poner dos baterías de diferente potencial y voltaje, debido a que la sección de control se puede alimentar perfectamente con unas baterías recargables de NiMh +5v de unos 700mA. El problema es que si alimentásemos los dos servomotores con estas baterías también, estos funcionarían casi a la mitad de su rendimiento, reduciendo así la movilidad y capacidad de reacción de nuestro robot y también la duración de nuestras baterías sería mucho más limitada.

Si quisiéramos usar una sola batería para todo nuestro robot tendríamos otra opción que sería obtener una batería de un voltaje cercano al máximo de nuestros servomotores y con una buena carga, y con esto diseñar un regulador de tensión que nos diera también los 5 voltios necesarios para la sección de control de nuestro robot.

En nuestro caso utilizaremos dos tipos de baterías distintos:

-Para la sección de control alimentamos con batería recargable de NImH de +4.8v de 750mAh. Esta batería la obtenemos de conectar 4 baterías de 1.2v tipo AAA en serie.

-Los servomotores los vamos a alimentar con una batería de LIion de 7.2V y una carga de 850mAh, (esta batería se ha fabricado con dos baterías de móvil de 3.6v cada una y conectadas en serie, para la carga de las mismas se ha dejado unos cables de fácil conexionado y acceso para cargarlas por separado. El uso de estas baterías recicladas de teléfonos móviles usados fue una de las ideas para dar este nombre al robot (“RECICLEBOT”).

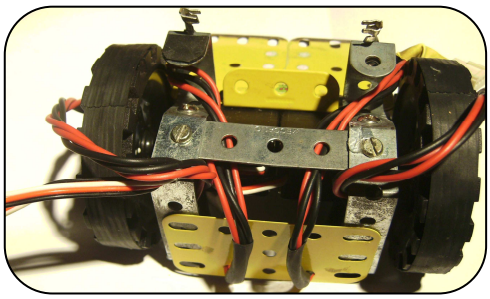
2.3 PLANIFICACIÓN.

En este apartado vamos a planificar la realización de nuestro prototipo de minirobot luchador de minisumo. Este prototipo lo vamos a realizar partiendo de un estudio de los diferentes programas realizados en lenguaje C, y que necesitamos para controlar nuestro sistema gobernado por microcontrolador y los periféricos que necesitamos incorporarle como son los sensores, motores, etc.

Con estos parámetros de funcionamiento, normativas y reglas debemos realizar un hardware que se adapte a nuestras necesidades y que haga competitivo nuestro minirobot luchador de minisumo, estableciendo los diseños de PCB correspondientes a cada módulo, su simulación y la implementación física de cada uno, así como el chasis que deberá soportar todos nuestros componentes dentro de las medidas y peso establecido. Todo este proceso se planifica de la siguiente forma:

- Nuestro prototipo comienza con el acercamiento al funcionamiento de fotosensores activos, sensores infrarojos, sensores de ultrasonido, motores de continua, servomotores y de sus características de funcionamiento y requerimientos para usarlos gobernados por los microcontroladores. También debemos conocer el funcionamiento de los microcontroladores, en especial el microcontrolador PIC 16F876A que es el que usaremos; y el proceso que hay que realizar para su programación. para ello vamos a utilizar el software de editor de texto MPLAB y con el realizaremos los programas en C para controlar nuestro minirobot luchador de minisumo gobernado por microcontrolador.
- Comenzamos en primer lugar a diseñar el programa que consiste en configurar el PIC para que podamos poner a funcionar todos los periféricos que necesitamos en nuestro robot como son sensores, motores y leds, diseñamos los programas necesarios individualmente y los vamos probando por separado en los MONIBOTS (robot experimentales del aula que están montados y listos para programar), si no disponemos de robots de prueba podremos realizar simulaciones mediante PROTEUS, aunque son menos exactas.
- Lo siguiente que se realiza es un programa que gobierne nuestro minirobot y el cual contenga todo lo necesario para que nuestro robot sea completamente independiente sobre el tatami de lucha, a esto llegamos uniendo todos los programas realizados para cada periférico y ensamblándolos todos juntos en uno solo. Tenemos que prestar mucha atención en este proceso a las configuraciones necesarias para que nuestro programa funcione correctamente.

- Comprobar el correcto funcionamiento del programa creando un circuito de simulación para depurar posibles errores tanto de software como de hardware. Para ello utilizamos los robots de que disponemos en clase, que ya están montados y podemos implementarle nuestro programa diseñado de forma más cómoda y rápida que mediante simulador PROTEUS donde podemos diseñar el circuito en ordenador y programarlo en su opción de simulación.
- Durante la fase de diseño del programa podemos también ir adquiriendo los diferentes componentes que nos harán falta para el diseño de nuestro chasis, para ir tomando medidas y poder ir ajustando el resto de diseños a nuestra necesidad según prototipo del chasis. Vamos montando posibles opciones de configuración del chasis para depurar errores y sacar nuevas ideas.
- Una vez que tengamos el programa final ya diseñado o de forma paralela a su diseño y al del chasis, podemos realizar los diseños de las PCB guiándonos del circuito de simulación y de la documentación adquirida sobre microcontroladores. Debemos prestar atención en el diseño para no dejar ningún componente sin implementar. Y teniendo en cuenta los ajustes necesarios para adaptar el circuito a nuestros requerimientos y medidas. En este paso debemos tener cuidado al tratarse de un diseño biplaca en las posiciones correctas de los conectores del bus de datos para no cambiar por error el emparejado de los pines.
- Se ajustan medidas para que la placa se adapte lo mejor posible a nuestro chasis y se implementa la PCB mediante proceso de insolación, revelado y ataque de ácido.
- Ajustamos los detalles para sujeción de las PCB en nuestro chasis de la manera más cómoda posible para poder tener acceso a los mandos como interruptores y conectores una vez ensamblada la PCB al chasis.
- Comprobar la calidad de la implementación del diseño de las PCB realizado y volver a realizarlos si fuera necesario para subsanar errores y fallos de procedimento.
- Perforar todos los pads con las medidas apropiadas para cada componente, comprobando correcto estado de las superficies de los pads del circuito.
- Hacer inspección de calidad del perforado de los pads y las superficies de soldadura, para que no surjan problemas posteriores a la hora de soldar los componentes en su lugar, como malos contactos.
- Insertar los componentes y soldar prestando atención en los tiempos de soldadura para así evitar las falsas soldaduras y soldaduras frías. Una vez montadas ambos circuitos, el de control y el de sensores. Tras terminar de montar ambas placas realizaremos una inspección para comprobar el correcto estado de las soldaduras y que no haya falsos puentes.



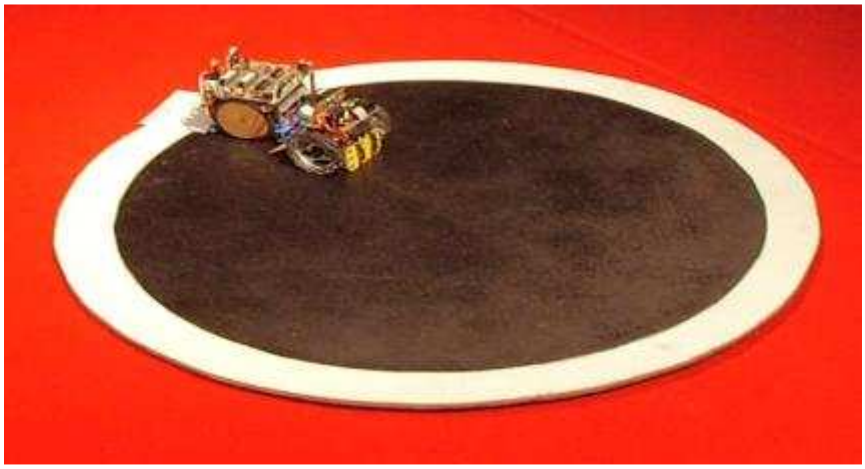
- Es momento de comprobar el funcionamiento de los circuitos, para ello probamos a programar nuestro circuito con los diferentes programas existentes ya realizados. Es recomendable empezar programando un programa sencillo como parpadeo de un led de la PCB para asegurarnos del correcto funcionamiento de la PCB. Interconectamos ambas placas y probamos funcionamiento.

- Para este paso tendremos que tener a nuestro alcance todos los elementos de hardware necesarios para que el robot funcione como son, las baterías que nos hacen falta para alimentar el sistema de control y los motores, el chasis casi al completo terminado para poder incorporarle las placas y hacer los ajustes en las sujeciones, debemos tener los sensores ya acoplados al chasis y los servomotores listos también para su uso instalados en chasis.

- Con todo el robot montado o casi montado, procedemos a realizar las primeras pruebas de funcionamiento. Programamos nuestro robot y realizamos ajustes para adaptar los resultados a los esperados. Debemos comprobar que el robot no se salga del tatami, que sea capaz de detectar un objeto que esté en el alcance programado y reaccione como deseamos, y que tenga la tracción suficiente para poder desplazar el mayor peso posible.

- Como ampliación del programa básico de minisumo se realiza un programa alternativo que se activa con los switch de la PCB para que nuestro robot realice un rastreo únicamente con los sensores de distancia por infrarojos, también realizaremos las pruebas pertinentes.

- Una vez realizados todos los ajustes podemos comenzar a variar las estrategias de combate en cuestión de movimientos de ataque y defensa. Para ello lo mejor es probar el robot con otros robots dentro del tatami.



2.4 PROGRAMACIÓN.

La programación de nuestro proyecto está condicionada por el calendario escolar dado el carácter didáctico de la asignatura y se realizará de la siguiente manera:

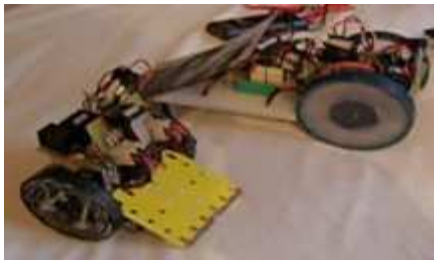
-Tenemos una primera etapa, que es la más extensa, en la que se realiza un acercamiento al funcionamiento y manejo de los microcontroladores y su programación, Para aprender a programarlos realizamos un acercamiento al entorno MPLAB y CCS. Este proceso necesita de un mes y de material didáctico informático software y hardware.
*El que posea conocimientos básicos reduce esta etapa al máximo.

-La segunda etapa es la de planteamiento del programa y búsqueda de una solución mediante programación en lenguaje C para controlar el microcontrolador, simulación del programa y ajuste de los componentes necesarios. Este proceso necesita mucha atención por parte del diseñador para poder realizar el programa y depurarlo correctamente mediante la realización de simulaciones en PROTEUS o con el uso de los robots de que se disponen en el aula. Nos hace falta material de software de programación y de simulación, así como un PC.

-La tercera etapa consiste en la selección y recopilación de componentes con las características más adecuadas a nuestro diseño, realización del chasis del robot, pedido de material, implementación del circuito en software y en PCB y comprobación de posibles fallos. En este proceso dependemos del tiempo de entrega del material, y de una semana para la realización del diseño de la PCB y su implementación física, libre de posibles fallos. Para esta etapa necesitamos todos los componentes, software de diseño PCB, un PC, acetato, PCB fotosensible positiva de doble capa, insoladora, reveladores y ácidos, así como todos los componentes que vayamos a utilizar en la construcción de nuestro chasis y todas las herramientas.

-En la cuarta etapa se perforan los pads para los componentes, se comprueban las superficies de soldadura, se limpia, se comienza a insertar componentes en su lugar pero por bloques de funcionamiento para comprobación del correcto estado, se procede con la soldadura, se comprueba el estado de las soldaduras y contactos. Esta etapa es la más rápida, en menos de una semana debemos tener todos los componentes montados y comprobados para poner en funcionamiento el robot durante un periodo de prueba. Necesitamos todos los componentes, taladro y accesorios, brocas, utensilios o productos para limpiar restos, soldador, desoldador y aleación de estaño.

-En la quinta y última etapa tenemos que realizar las comprobaciones del funcionamiento del robot totalmente montado e interactuando con otros robots para poder observar los resultados de nuestra programación y realizar todos los cambios pertinentes en cuanto a hardware y software.



3. CÁLCULOS.

3.1 Cálculo de la intensidad de los puertos del PIC.

Debemos saber que los puertos del microcontrolador PIC son capaces de entregar o recibir un corriente fija que es de:

-**25mA por pin y 200mA por puerto** cuando recibe (entrada).
-**25mA por pin y 200mA por puerto** cuando ofrece (salida).

Sabiendo esto debemos verificar que las intensidades que tendremos en cada pin y en cada puerto son las soportadas por el microcontrolador.

En el caso de necesitar poner motores de corriente continua, esta corriente es insuficiente y se debe intercalar entre el microcontrolador y los motores, un driver de potencia para dar la corriente necesaria y unos diodos de protección para la los picos de la bobina del motor en los cambios de giro.

3.2Cálculo de limitación de intensidad en los leds.

Los diodos leds se limitan en intensidad para que no se produzcan daños por exceso de calor y de continuidad en la barrera iónica.

En este caso vamos a limitar la intensidad de los leds sobre los 20mA, y el led luce con buen brillo, para ello calculamos:

Vin = +5v. $R = \frac{V}{I}$; $R = \frac{5}{0.020}$; $R = 250\Omega$.

La resistencia comercial que se escoge es de **220Ω**, y nos da intensidad de 22mA.

3.3Cálculo del PWM para servos.

Un servo es un motor controlado por una electrónica que lee el PWM y que se encarga de mover al motor dependiendo de lo que ha leído.

El servo, o mejor dicho la electrónica del servo coloca al motor en cada posición dependiendo del tiempo en que el pulso que le inyectamos permanece en alto. Si el tiempo que dura en estado alto dura exactamente **1.5 milisegundos** entonces el Servo va y se coloca en el **centro de su recorrido**, si dura exactamente **0.5 milisegundos** el servo retrocede desde el punto medio unos 90º y se coloca en su **extremo izquierdo** y si, por último, dura exactamente **2.5 milisegundos** el servo avanza desde el punto medio unos 90º y se coloca en su **extremo derecho**. Al tiempo en que permanece en alto un pulso le llamamos **Duty Cicle.**

Con duraciones intermedias del tiempo en que permanece el pulso en alto, o Duty Cicle, el servo se posiciona en puntos intermedios de su recorrido.

Para que el servo responda correctamente a estos distintos Duty Cicles los pulsos deben llegarle al servo con una periodicidad, o **frecuencia constante**, uno tras otro, separados 20 milisegundos cada uno del siguiente, cada flanco de subida debe estar separado del siguiente flanco de subida los mismos 20 milisegundos; por lo tanto cada ciclo alto-bajo dura siempre exactamente 20 milisegundos y lo que variamos es la relación entre el tiempo que está en alto y en bajo.

Decir que los pulsos están separados unos de otros 20 milisegundos es exactamente lo mismo que decir que se envían con una frecuencia de 50 Herzios, ya que 50Hz son 50 pulsos por segundo y por lo tanto 1000 milisegundos (que tiene un segundo) dividido entre 50 son exactamente eso: 20 milisegundos. O sea aplicamos la formula,

f (frecuencia en Herzios) = **1 / t** (Periodo en Segundos).

Ahora lo que tenemos que hacer es saber cómo podemos controlar estos tiempos en nuestro PIC para poner en alto (disparar el pulso) y en bajo (apagarlo) con la cadencia adecuada, siguiendo la tabla de tiempos descrita más arriba.

Para ello usamos el **TIMER0** del PIC que me va a servir de reloj para saber cuándo y durante cuánto tiempo tengo que tener mi pulso en alto. Hemos elegido un divisor, o preescaler, del TIMER0 de 1:16.

Asumiendo que tenemos nuestro PIC funcionando con un cristal de 4.00Mhz entonces el TIMER0 funcionando a 1:16 hace saltar la *Interrupción por Desbordamiento de Timer*, tambien conocida como **RTCC**, cada 4.096 milisegundos.

Esto es lo mismo que decir que TIMER0 tarda 4.096 milisegundos en contar desde 0 a 255 y que al llegar a 255 pasar de nuevo a 0 hace saltar la RTCC.

Esto significa que cada paso de contador del TIMER0, a lo que llamamos un **tick de reloj**, tarda 4.096 / 256 = 0.016 milisegundos. Esto me da una pauta bastante fácil de calcular que consiste en que cada 5 RTCC completas tengo 5 * 4.096 = 20.48 milisegundos que es un poco más de lo que necesito, que son 20 milisegundos exactos.

Además sabiendo que cada tick de reloj ocupa 0.016 milisegundos podemos traducir los tiempos de anchos de pulsos descritos anteriormente en ticks de nuetro reloj particular: así **0.5 milisegundos** son lo mismo que esperar **31 ticks de reloj**, **1.5 milisegundos** equivalen a **93 ticks de reloj** y **2.5 milisegundos** son **155 ticks de reloj**. (Recordad que llamamos tick de reloj al tiempo que tarda TIMER0 en contar 1 más).

4. ANEXOS DE HARDWARE.

Hay que tener una información técnica mínima para el correcto desarrollo de los diseños, como puede ser patillaje de los componentes, características eléctricas, rangos máximos de trabajo, etc. A continuación se exponen todos los necesarios.

4.1 MICROCONTROLADOR PIC 16F876A.

Los microcontroladores PIC son en el fondo procesadores similares a otros tipos, como por ejemplo la familia de los microprocesadores X86, 80486, Pentium y muchos otros que usan una arquitectura interna del tipo Von Neumann.

En este tipo de arquitectura los datos y la memoria del programa se encuentran en el mismo espacio de direcciones.

En realidad un microprocesador y un microcontrolador no son la misma cosa. Los PICs son microcontroladores, es decir, una unidad que posee en su interior al microprocesador y a los elementos indispensables para que pueda funcionar como una minicomputadora en un solo chip.

Un microprocesador es solamente la unidad central de procesos o CPU, la memoria, los puertos y todos los demás periféricos son exteriores. La programación de un microprocesador es, por lo tanto, una tarea compleja porque deben controlarse todos estos dispositivos externos.

Un microcontrolador integra la CPU y todos los periféricos en un mismo chip.

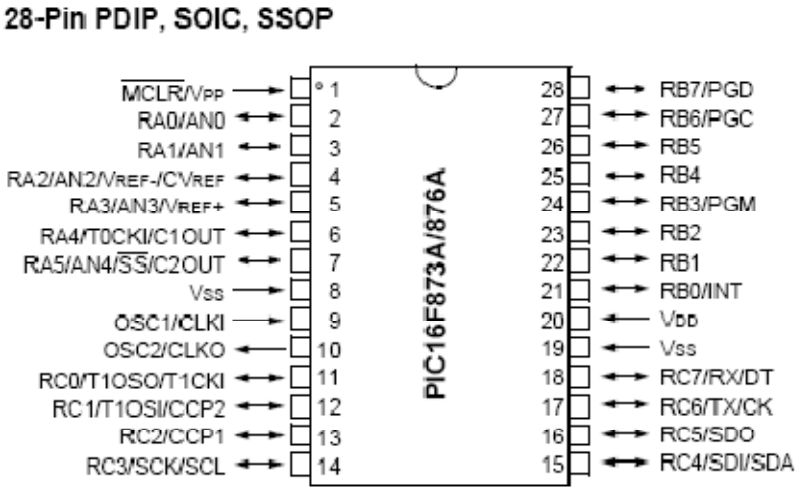
ARQUITECTURA SIMPLIFICADA DEL PIC16F84

Un microcontrolador como cualquier circuito integrado analógico tiene entradas, salidas y algunos componentes exteriores necesarios para procesar las señales de entrada y convertirlas en las señales de salida. El 16F876A requiere un cristal de 4MHz con dos condensadores y como mínimo una resistencia para el reset. Por supuesto necesita una tensión de fuente de 5V (VDD) aplicada con respecto al terminal de masa (VSS). Posee tres puertos de entrada/salida, PORT A, PORT B y PORT C, cuyos terminales son marcados de RA0 a RA5, de RB0 a RB7 y de RC0 a RC7. Estos puertos pueden ser programados como entrada o de salida. El pin 1 opera como reset pero también cumple funciones de carga de memoria de programa cuando es excitado con pulsos de 15V.

-Los terminales 12 y 13 (CCP2 y CCP1), son los que nos ofrecen el PWM del microcontrolador y es donde deberán ir conectados los servomotores.

-Este PIC posee varios pines que aceptan entrada de señales analógicas y que están en el puerto A. (AN0,AN1,AN2,AN3 y AN4), a estos pines podremos conectar los sensores analógicos GP2D12.

-Los pines 28,27 y 24 (PGD,PGC,PGM) son donde conectaremos el conector RJ11 para realizar la conexión del microcontrolador con el exterior para su programación por el debuger.



Función de los diferentes pines del microcontrolador 16F876A.

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI	9	6		ST/CMOS(3)	Oscillator crystal or external clock input.
OSC1			I		Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS.
CLKI			I		External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO	10	7		—	Oscillator crystal or clock output.
OSC2			O		Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O		In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	26		ST	Master Clear (input) or programming voltage (output).
$\overline{MCLR}$			I		Master Clear (Reset) input. This pin is an active low Reset to the device.
VPP			P		Programming voltage input.
					PORTA is a bidirectional I/O port.
RA0/AN0	2	27		TTL	
RA0			I/O		Digital I/O.
AN0			I		Analog input 0.
RA1/AN1	3	28		TTL	
RA1			I/O		Digital I/O.
AN1			I		Analog input 1.
RA2/AN2/VREF-/CVREF	4	1		TTL	
RA2			I/O		Digital I/O.
AN2			I		Analog input 2.
VREF-			I		A/D reference voltage (Low) input.
CVREF			O		Comparator VREF output.
RA3/AN3/VREF+	5	2		TTL	
RA3			I/O		Digital I/O.
AN3			I		Analog input 3.
VREF+			I		A/D reference voltage (High) input.
RA4/T0CKI/C1OUT	6	3		ST	
RA4			I/O		Digital I/O – Open-drain when configured as output.
T0CKI			I		Timer0 external clock input.
C1OUT			O		Comparator 1 output.
RA5/AN4/$\overline{SS}$/C2OUT	7	4		TTL	
RA5			I/O		Digital I/O.
AN4			I		Analog input 4.
$\overline{SS}$			I		SPI slave select input.
C2OUT			O		Comparator 2 output.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT	21	18		TTL/ST(1)	
RB0			I/O		Digital I/O.
INT			I		External interrupt.
RB1	22	19	I/O	TTL	Digital I/O.
RB2	23	20	I/O	TTL	Digital I/O.
RB3/PGM	24	21		TTL	
RB3			I/O		Digital I/O.
PGM			I		Low-voltage (single-supply) ICSP programming enable pin.
RB4	25	22	I/O	TTL	Digital I/O.
RB5	26	23	I/O	TTL	Digital I/O.
RB6/PGC	27	24		TTL/ST(2)	
RB6			I/O		Digital I/O.
PGC			I		In-circuit debugger and ICSP programming clock.
RB7/PGD	28	25		TTL/ST(2)	
RB7			I/O		Digital I/O.
PGD			I/O		In-circuit debugger and ICSP programming data.
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI	11	8		ST	
RC0			I/O		Digital I/O.
T1OSO			O		Timer1 oscillator output.
T1CKI			I		Timer1 external clock input.
RC1/T1OSI/CCP2	12	9		ST	
RC1			I/O		Digital I/O.
T1OSI			I		Timer1 oscillator input.
CCP2			I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	13	10		ST	
RC2			I/O		Digital I/O.
CCP1			I/O		Capture1 input, Compare1 output, PWM1 output.
RC3/SCK/SCL	14	11		ST	
RC3			I/O		Digital I/O.
SCK			I/O		Synchronous serial clock input/output for SPI mode.
SCL			I/O		Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	15	12		ST	
RC4			I/O		Digital I/O.
SDI			I		SPI data in.
SDA			I/O		I^2C data I/O.
RC5/SDO	16	13		ST	
RC5			I/O		Digital I/O.
SDO			O		SPI data out.
RC6/TX/CK	17	14		ST	
RC6			I/O		Digital I/O.
TX			O		USART asynchronous transmit.
CK			I/O		USART1 synchronous clock.
RC7/RX/DT	18	15		ST	
RC7			I/O		Digital I/O.
RX			I		USART asynchronous receive.
DT			I/O		USART synchronous data.
VSS	8, 19	5, 6	P	—	Ground reference for logic and I/O pins.
VDD	20	17	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

Características eléctricas del microcontrolador 16F876A.

17.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

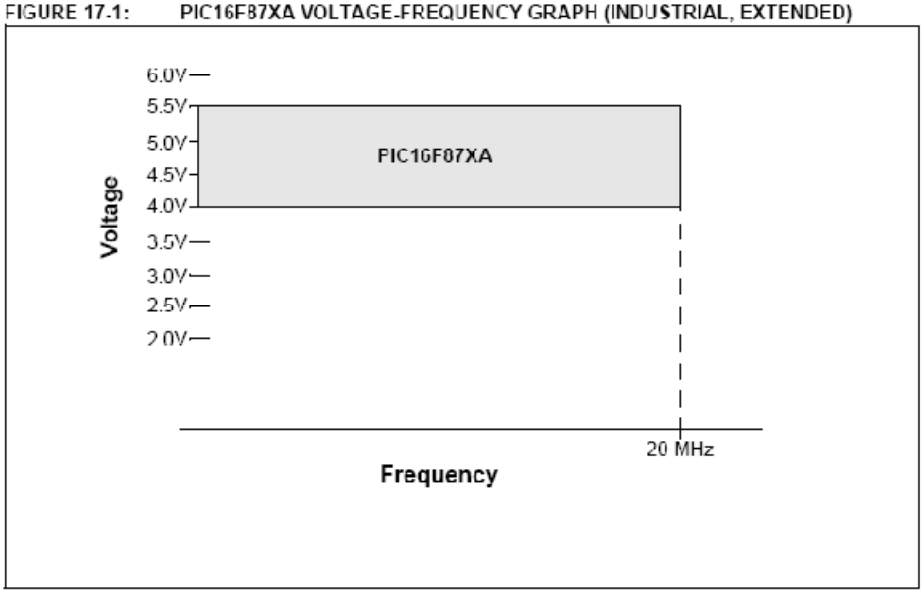
Ambient temperature under bias	-55 to +125°C
Storage temperature	-65°C to +150°C
Voltage on any pin with respect to Vss (except VDD, $\overline{\text{MCLR}}$, and RA4)	-0.3V to (VDD + 0.3V)
Voltage on VDD with respect to Vss	-0.3 to +7.5V
Voltage on $\overline{\text{MCLR}}$ with respect to Vss **(Note 2)**	0 to +14V
Voltage on RA4 with respect to Vss	0 to +8.5V
Total power dissipation **(Note 1)**	1.0W
Maximum current out of Vss pin	300 mA
Maximum current into VDD pin	250 mA
Input clamp current, IIK (VI < 0 or VI > VDD)	±20 mA
Output clamp current, IOK (VO < 0 or VO > VDD)	±20 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	25 mA
Maximum current sunk by PORTA, PORTB and PORTE (combined) **(Note 3)**	200 mA
Maximum current sourced by PORTA, PORTB and PORTE (combined) **(Note 3)**	200 mA
Maximum current sunk by PORTC and PORTD (combined) **(Note 3)**	200 mA
Maximum current sourced by PORTC and PORTD (combined) **(Note 3)**	200 mA

Note 1: Power dissipation is calculated as follows: Pdis = VDD x {IDD - Σ IOH} + Σ {(VDD - VOH) x IOH} + Σ(VOL x IOL)

2: Voltage spikes below Vss at the $\overline{\text{MCLR}}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{\text{MCLR}}$ pin rather than pulling this pin directly to Vss.

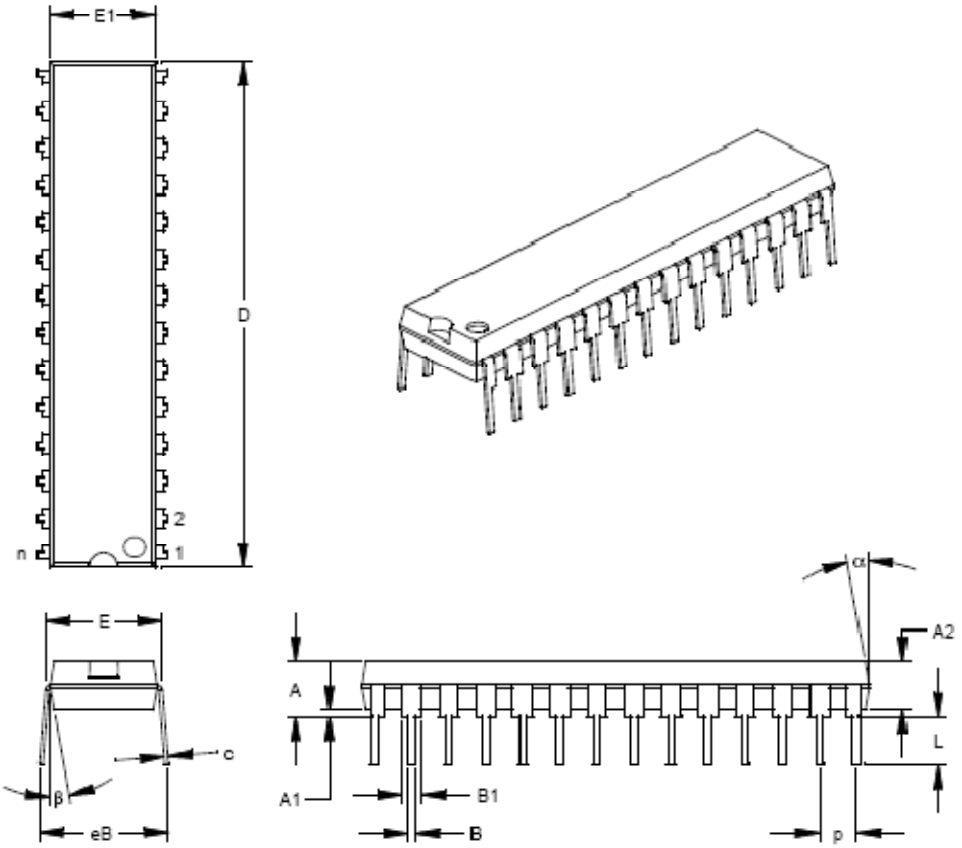
3: PORTD and PORTE are not implemented on PIC16F873A/876A devices.

FIGURE 17-1: PIC16F87XA VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL, EXTENDED)



Dimensiones reales del microcontrolador 16F876A.

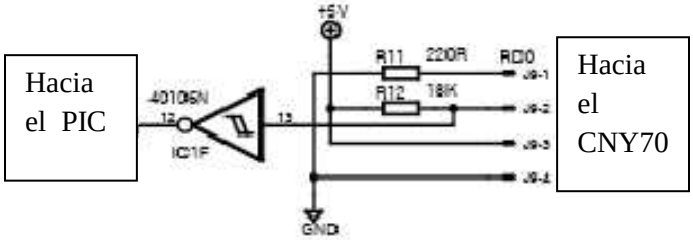
28-Lead Skinny Plastic Dual In-line (SP) – 300 mil (PDIP)



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.140	.150	.160	3.56	3.81	4.06
Molded Package Thickness	A2	.125	.130	.135	3.18	3.30	3.43
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.300	.310	.325	7.62	7.87	8.26
Molded Package Width	E1	.275	.285	.295	6.99	7.24	7.49
Overall Length	D	1.345	1.365	1.385	34.16	34.67	35.18
Tip to Seating Plane	L	.125	.130	.135	3.18	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.040	.053	.065	1.02	1.33	1.65
Lower Lead Width	B	.016	.019	.022	0.41	0.48	0.56
Overall Row Spacing §	eB	.320	.350	.430	8.13	8.89	10.92
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

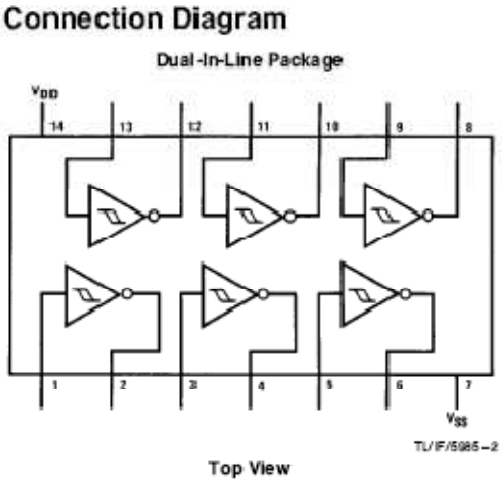
4.2 TRIGGER SCHMITT 40106.

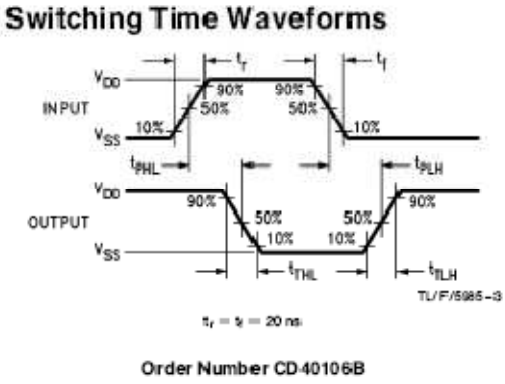
Conversor de señales analógicas en señales digitales puras, con salida inversora. Este dispositivo lo utilizaremos en la conexión de los fotosensores activos para obtener la señal digital más pura. Con un solo Trigger Schmitt tenemos para conectar hasta seis sensores. Usaremos este dispositivo para los sensores CNY70 y lo conectaremos de la siguiente manera al microcontrolador:



Las conexiones marcadas con el 1,2,3,4, son las que van a las patillas de los CNY70 y su correspondencia es:

1-Cátodo del diodo.
2-Colector del fototransistor.
3-Ánodo del diodo emisor.
4-Emisor del fototransistor.





Aspecto físico.

Características eléctricas del Trigger Schmitt 40106N.

Absolute Maximum Ratings (Notes 1 & 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

DC Supply Voltage (V_{DD})	−0.5 to +18 V_{DC}
Input Voltage (V_{IN})	−0.5 to V_{DD} + 0.5 V_{DC}
Storage Temperature Range (T_S)	−65°C to +150°C
Power Dissipation (P_D)	
Dual-In-Line	700 mW
Small Outline	500 mW
Lead Temperature (T_L) (Soldering, 10 seconds)	260°C

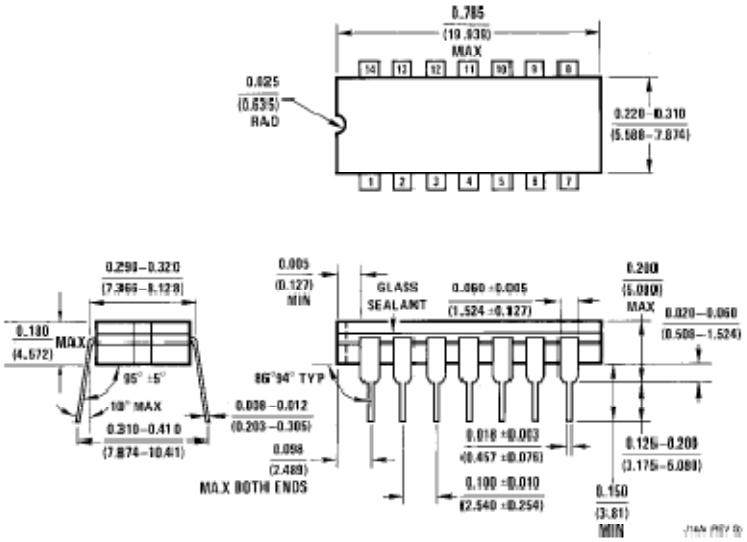
Recommended Operating Conditions (Note 2)

DC Supply Voltage (V_{DD})	3 to 15 V_{DC}
Input Voltage (V_{IN})	0 to V_{DD} V_{DC}
Operating Temperature Range (T_A)	
CD40106BM	−55°C to +125°C
CD40106BC	−40°C to +85°C

DC Electrical Characteristics CD40106BM (Note 2)

Symbol	Parameter	Conditions	−55°C		+25°C			+125°C		Units
			Min	Max	Min	Typ	Max	Min	Max	
I_{DD}	Quiescent Device Current	V_{DD} = 5V, V_{IN} = V_{DD} or V_{SS}		1.0			1.0		30	μA
		V_{DD} = 10V, V_{IN} = V_{DD} or V_{SS}		2.0			2.0		60	μA
		V_{DD} = 15V, V_{IN} = V_{DD} or V_{SS}		4.0			4.0		120	μA
V_{OL}	Low Level Output Voltage	$\|I_O\|$ < 1 μA								
		V_{DD} = 5V		0.05			0.05		0.05	V
		V_{DD} = 10V		0.05			0.05		0.05	V
		V_{DD} = 15V		0.05			0.05		0.05	V
V_{OH}	High Level Output Voltage	$\|I_O\|$ < 1 μA								
		V_{DD} = 5V	4.95		4.95	5		4.95		V
		V_{DD} = 10V	9.95		9.95	10		0.95		V
		V_{DD} = 15V	14.95		14.95	15		14.95		V
V_{T-}	Negative-Going Threshold Voltage	V_{DD} = 5V, V_O = 4.5V	0.7	2.0	0.7	1.4	2.0	0.7	2.0	V
		V_{DD} = 10V, V_O = 9V	1.4	4.0	1.4	3.2	4.0	1.4	4.0	V
		V_{DD} = 15V, V_O = 13.5V	2.1	6.0	2.1	5.0	6.0	2.1	6.0	V
V_{T+}	Positive-Going Threshold Voltage	V_{DD} = 5V, V_O = 0.5V	3.0	4.3	3.0	3.6	4.3	3.0	4.3	V
		V_{DD} = 10V, V_O = 1V	6.0	8.6	6.0	6.8	8.6	6.0	8.6	V
		V_{DD} = 15V, V_O = 1.5V	9.0	12.9	9.0	10.0	12.9	9.0	12.9	V
V_H	Hysteresis (V_{T+} − V_{T-})	V_{DD} = 5V	1.0	3.6	1.0	2.2	3.6	1.0	3.6	V
		V_{DD} = 10V	2.0	7.2	2.0	3.6	7.2	2.0	7.2	V
		V_{DD} = 15V	3.0	10.8	3.0	5.0	10.8	3.0	10.8	V
I_{OL}	Low Level Output Current (Note 3)	V_{DD} = 5V, V_O = 0.4V	0.64		0.51	0.88		0.36		mA
		V_{DD} = 10V, V_O = 0.5V	1.6		1.3	2.25		0.9		mA
		V_{DD} = 15V, V_O = 1.5V	4.2		3.4	8.8		2.4		mA
I_{OH}	High Level Output Current (Note 3)	V_{DD} = 5V, V_O = 4.6V	−0.64		−0.51	−0.88		−0.36		mA
		V_{DD} = 10V, V_O = 9.5V	−1.6		−1.3	−2.25		−0.9		mA
		V_{DD} = 15V, V_O = 13.5V	−4.2		−3.4	−8.8		−2.4		mA
I_{IN}	Input Current	V_{DD} = 15V, V_{IN} = 0V		−0.10		-10^{-5}	−0.10		−1.0	μA
		V_{DD} = 15V, V_{IN} = 15V		0.10		10^{-5}	0.10		1.0	μA

Physical Dimensions inches (millimeters)

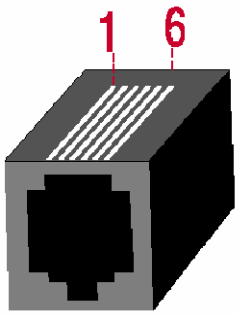


Ceramic Dual-In-Line Package (J)
Order Number CD40106BMJ or CD40106BCJ
NS Package Number J14A

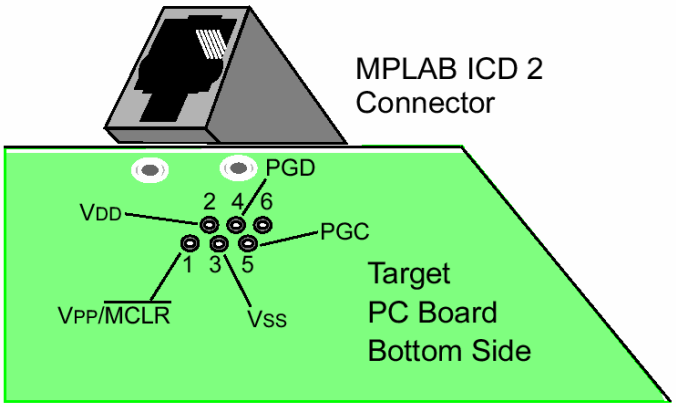
4.3 Conexion del RJ11 al PIC.

Para conectar nuestro robot con el exterior mediante el MPLAB ICD2 o ICD3 y así poder programarlo tantas veces nos sea necesario de la forma más cómoda, debemos incluir en nuestro diseño un conector RJ-11(cable telefónico) el cuál debe ir conectado correctamente al microcontrolador. Para esta necesidad el PIC posee varios pines destinados a este fin y son los pines 28,27 y 24 (PGD,PGC,PGM). Hay que conectar los pines del RJ-11 de la siguiente manera para que funcione correctamente.

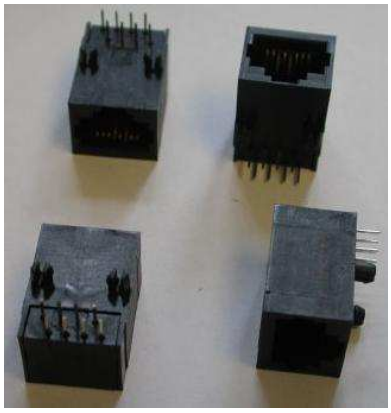
-**Pin1**, a la patilla MCLR del PIC(pin 1).
-**Pin2**, conectamos a Vcc (+5v).
-**Pin3**, conectamos a Vss (masa).
-**Pin4**, a la patilla PGD del PIC (pin 28).
-**Pin5**, a la patilla PGC del PIC (pin 27).
-**Pin6**, a la patilla PGM del PIC (pin 24).



Es recomendable añadir al diseño un condensador de 10nF en paralelo entre el pin que va a Vcc (+5v) y el pin que va Vss (masa) para estabilizar la tensión que le llega.



El aspecto físico del conector :



4.3 Método para trucar servomotores FUTABA S3003.

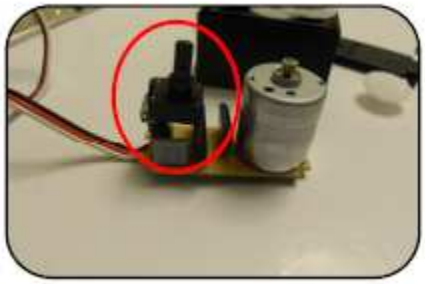
Para el diseño de este minisumo he seleccionado dos servomotores de la marca FUTABA S3003, este tipo de motores poseen una caja reductora integrada con gran torque, tienen el giro limitado en 180º y se activan por medio de PWM (pulse width modulation), según sea la duración de este pulso (0.3s-1.2s-2.4s) el motor se posiciona en un extremo (0º) o en otro (180º), con un pulso medio se posiciona en el centro (90º). Lo que no es compatible para nuestro diseño es que tiene el giro limitado, por lo que debemos "trucarlo" para que pueda girar los 360º completos. Lo que si mantenemos es la reductora y la electrónica del servo para poder controlar nuestros motores mediante el PWM y sin tener que incorporar driver de potencia para conectarlos al PIC. Este tipo de motores podemos conectarlos directamente al microcontrolador reduciendo el tamaño del diseño.

Para trucar el giro del motor realizamos los siguientes pasos:

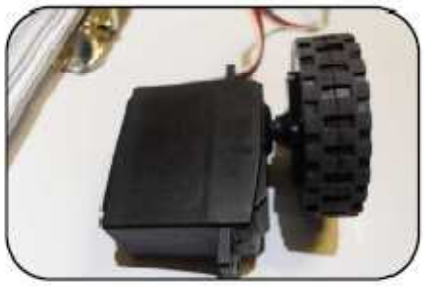
1. Lo primero es desmontar la carcasa superior que cubre el motor y que está fijada mediante cuatro tornillos, con cuidado de no descolocar los engranajes al sacarla.

2. Ahora tenemos que eliminar el tope que hay en el engranaje principal, en el que engancha la rueda, éste limita físicamente al motor a girar 180º. Para cortarlo utilizamos una cuchilla o herramienta cortante para cortarla al nivel del resto del engranaje y así no dé con el tope de la carcasa.

3. El siguiente paso consiste en centrar el potenciómetro existente en la circuitería poniéndole un punto de pegamento para fijarlo en esa posición, y también tenemos que dejarlo un poco más debajo de su posición para que no se encaje dentro del eje central de giro.



4. Una vez centrado el potenciómetro volvemos a montar todos los engranajes, colocamos la carcasa y ponemos los cuatro tornillos. Las ruedas las acoplamos pegando uno de los tipos de base que trae el servo a la parte trasera de la rueda.



Para conectar los servos al microcontrolador, lo podemos hacer directamente lo que simplifica el diseño. Nosotros elegiremos conectores llamados JUMPERS ya que se ajustan a las medidas del conector, serán de tres contactos que son los que salen del motor y se conectarán de la siguiente manera:

-**cable blanco**, cable portador del impulso PWM.
-**cable rojo**, cable de tensión positiva (+8).
-**cable negro**, cable de masa (0v).

Para poder desconectar la tensión de los motores mientras no se usan o se está programando, hemos incorporado otro jumper que anula la tensión de +8v cuando se cambia el jumper link de posición.

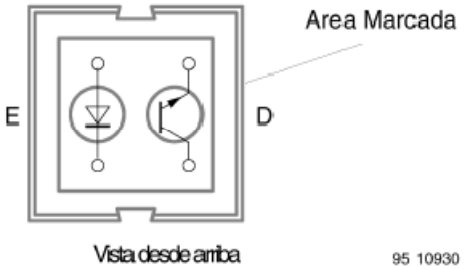
4.4 Fotosensor activo CNY70.

Este tipo de sensor es el que utilizaremos para que nuestro robot permanezca dentro del tatami circular de combate ya que cambia la señal digital que envía de 1 a 0 según este sobre blanco o sobre negro.

El CNY70 es un sensor óptico reflexivo que tiene una construcción compacta dónde el emisor de luz y el receptor se colocan en la misma dirección para detectar la presencia de un objeto utilizando la reflexión del infrarrojo sobre el objeto. La longitud de onda de trabajo es 950nm. Y el detector consiste en un fototransistor. Su distancia de trabajo ronda desde los 5mm a los 10mm, aunque es recomendable ajustarlo lo máximo posible al mínimo.

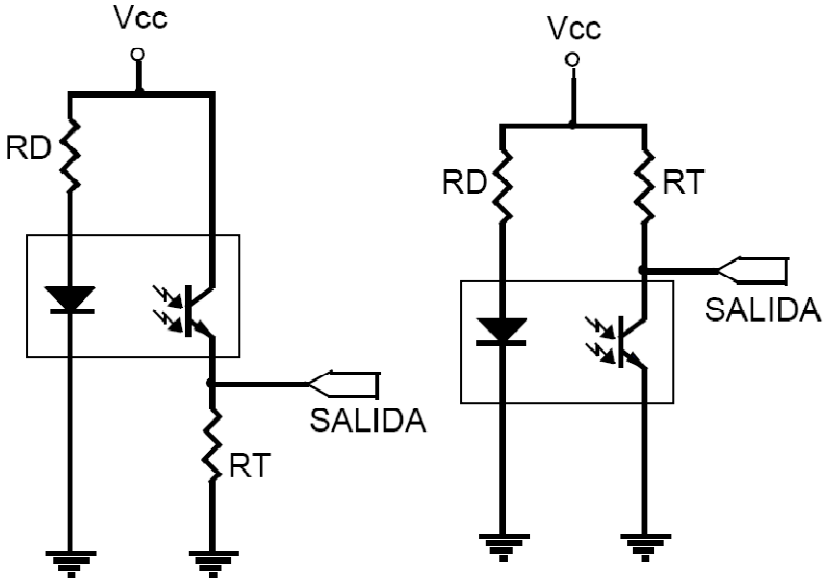
A la hora de conectar estos sensores debemos tener mucho cuidado en no confundir los pines, ya que esto puede ocasionar la rotura del sensor.Para poder diferenciar cada pin se puede observar el siguiente esquema:

1-Cátodo del diodo.
2-Colector del fototransistor.
3-Ánodo del diodo emisor.
4-Emisor del fototransistor.



Según conectemos este sensor al microcontrolador, nos dará señal digital en estado alto (1 lógico) con blanco o con negro, para ello es necesario cambiar la configuración del diseño de la siguientes dos maneras posibles:

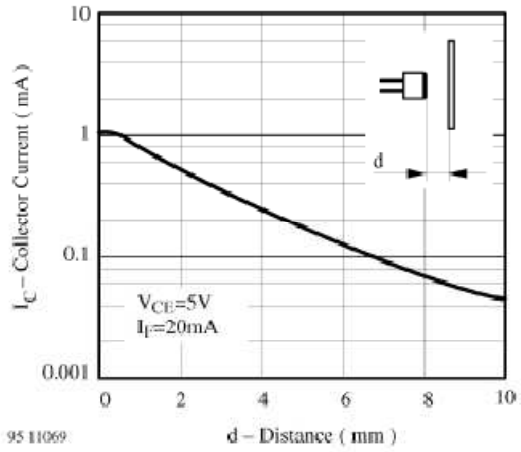
El circuito (a) entrega a la salida un nivel bajo cuando no refleja el haz infrarrojo y un nivel alto cuando encuentra un material sobre el que refleja el haz. El circuito (b) entrega un nivel alto cuando el haz no refleja y un nivel bajo cuando se detecta un material reflectante. Si la señal se quiere introducir a un microcontrolador es conveniente hacer pasar las salidas a través de un circuito Trigger Schmitt que conforme las señales.



Las características eléctricas de este dispositivo son las siguientes:

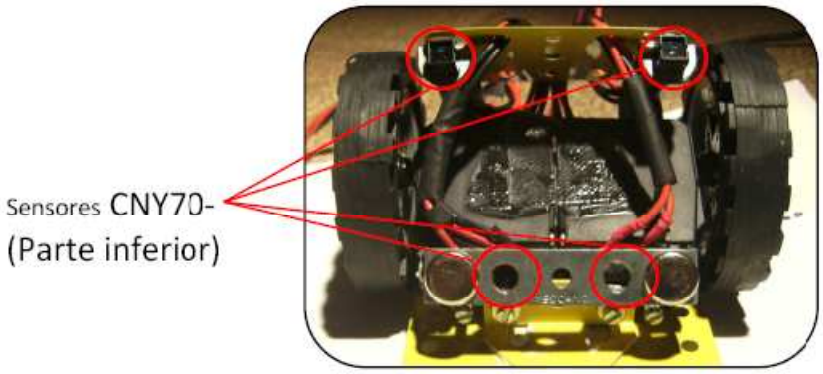
Entrada (Emisor)				
Parámetro	**Condiciones de Test**	**Símbolo**	**Valor**	**Unidades**
Tensión Inversa		V_R	5	V
Corriente Directa		I_F	50	mA
Corriente directa de Sobretensión	$T_p < 10\ \mu s$	I_{FSM}	3	A
disipación de Potencia	$T_{amb} \leq 25ºC$	P_V	100	mW
Temperatura de la unión		T_j	100	ºC

Salida (Detector)				
Parámetro	**Condiciones de Test**	**Símbolo**	**Valor**	**Unidades**
Tensión Colector Emisor		V_{CEO}	32	V
Tensión Colector Emisor		V_{ECO}	7	V
Corriente de Colector		I_C	50	mA
Disipación de Potencia	$T_{amb} \leq 25ºC$	P_V	100	mW
Temperatura de la unión		T_j	100	ºC



La corriente que entrega el colector según la distancia a la que se encuentra el sensor del objeto se describe en el gráfico de la izquierda.

Nota: A la hora de soldar este tipo de sensores se debe tener mucha atención en cuanto a los tiempos de soldadura y a las temperaturas máximas ya que son extremadamente sensibles y puede ocasionar averías en los sensores, ocasionando un mal funcionamiento del sistema.

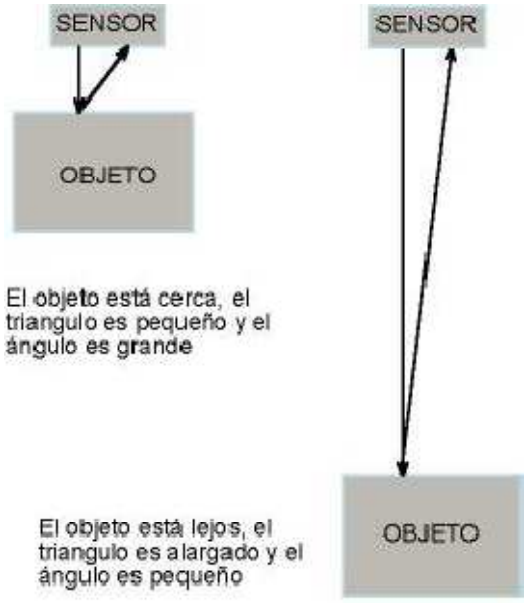


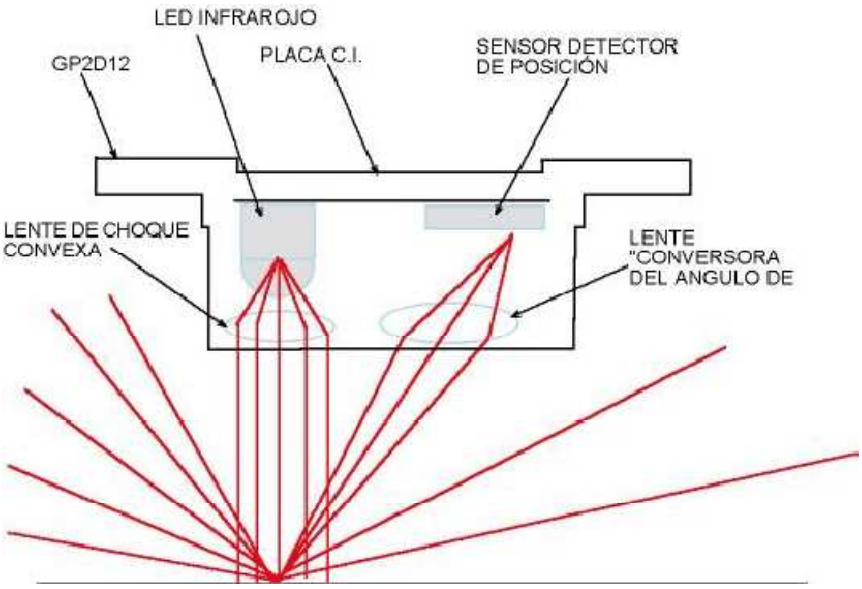
4.5 Sensor infrarrojo GP2D12.

Los IR Sharp GP2DXX son una familia de sensores de infrarrojos para la detección y medida de distancia a los objetos. Este sensor nos entrega una señal de valor analógico que varía entre los 0 y los 3 voltios dependiendo de la distancia a la que se encuentre el objeto y su alcance llega desde los 10 a los 80cm.

Estos dispositivos emplean el método de triangulación utilizando un pequeño Sensor Detector de Posición (PSD) lineal para determinar la distancia o la presencia de los objetos dentro de su campo de visión. Básicamente su modo de funcionamiento consiste en la emisión de un pulso de luz infrarroja, que se transmite a través de su campo de visón que se refleja contra un objeto o que por el contrario no lo hace. Si no encuentra ningún obstáculo, el haz de luz no refleja y en la lectura que se hace indica que no hay ningún obstáculo. En el caso de encontrar un obstáculo el haz de luz infrarroja se reflecta y crea un triangulo formado por el emisor, el punto de reflexión (obstáculo) y el detector.

La información de la distancia se extrae midiendo el ángulo recibido. Si el ángulo es grande, entonces el objeto está cerca, por que el triangulo es ancho. Si el ángulo es pequeño, entonces el objeto está lejos, por que el triangulo formado es estrecho. Por lo tanto, si el ángulo es pequeño, quiere decir que el objeto está lejos, porque el triangulo es largo y delgado. En la figura podemos verlo más claro.



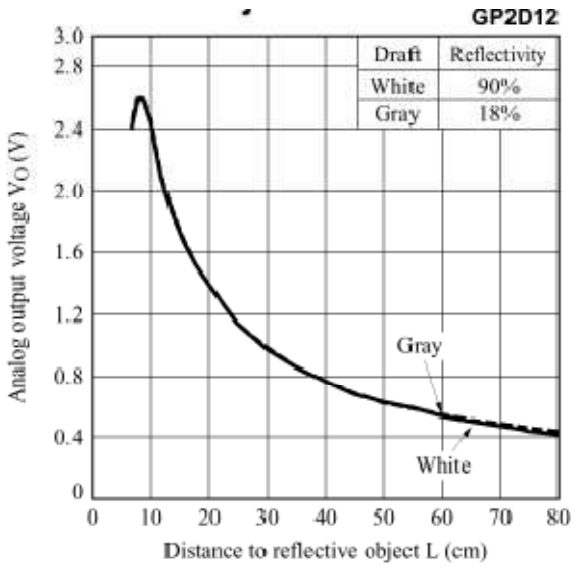


Éste dispositivo semiconductor entrega una salida cuya intensidad es proporcional a la posición respecto al centro (centro eficaz) de la luz que incide en él. En el caso del GP2D12, los rendimientos de PSD la salida es proporcional a la posición del punto focal. Esta señal analógica tratada es la que se obtiene a la salida del sensor.

Esto quiere decir que mientras más cerca este el objeto, más intensidad recibe el transistor receptor y por lo tanto más tensión tendremos en la salida analógica del sensor. Y mientras más lejos esté el obstáculo, menos intensidad recibirá el transistor y por lo tanto menos tensión tendremos a la salida analógica del mismo.

-Para conectar este tipo de sensores necesitamos de un convertidor analógico digital, que en este caso será el que tiene el microcontrolador en algunos de sus pines del PUERTO A, así ahorramos más espacio en el diseño y podemos tener un rango de valores analógicos que introducir en el programa del microcontrolador.

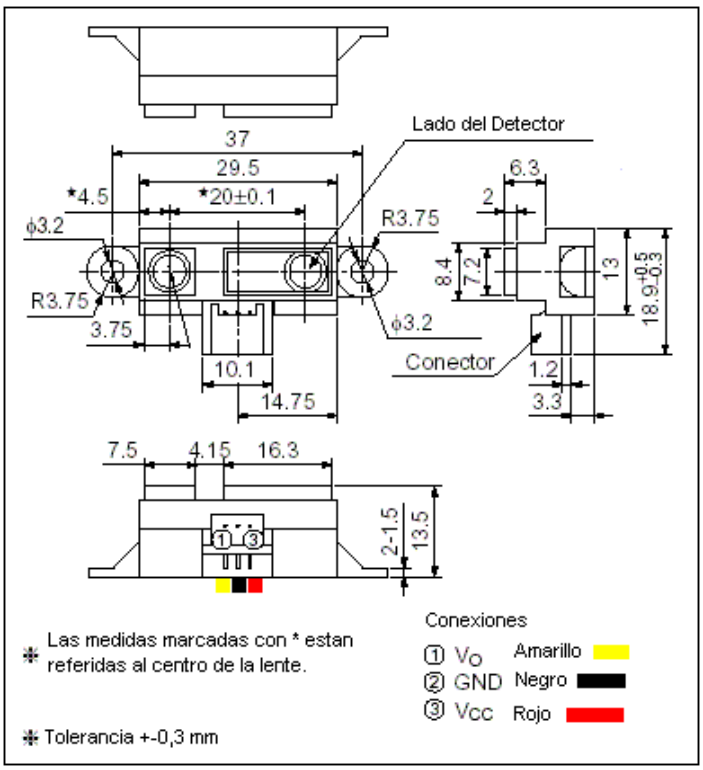
Gráfico de la disminución de la tensión en función de la distancia.



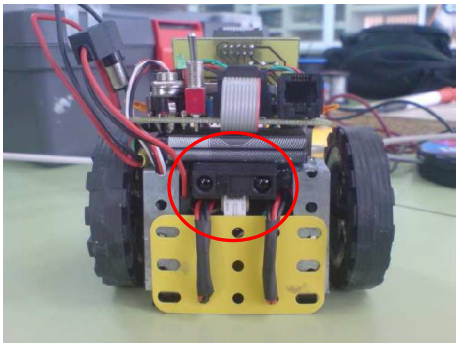
Este sensor tiene características eléctricas que debemos conocer:

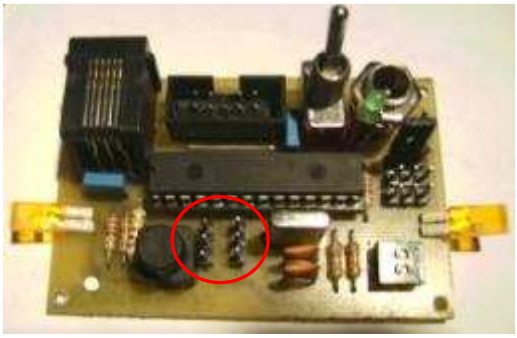
GP2D12 (Ta=25˚C)

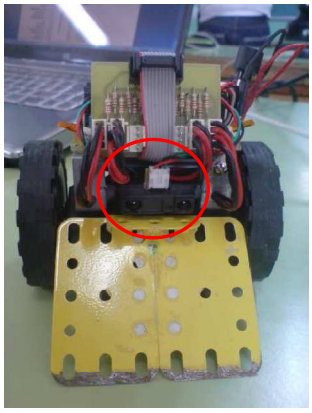
Parameter	Symbol	Rating
Supply voltage	Vcc	4.5 to 5.5V
Dissipation current	Icc	MAX.35mA
Measuring range	L	10 to 80cm
Output type	—	Analog output
Operating temperature	Topr	-10 to +60˚C



Nota: Al crear el diseño de la PCB tendremos en cuenta el posicionamiento de los conectores para simplificarlo y el conector que pondremos en la placa serán unos jumpers de tres contactos que encajan casi a la perfección con este tipo de conector.







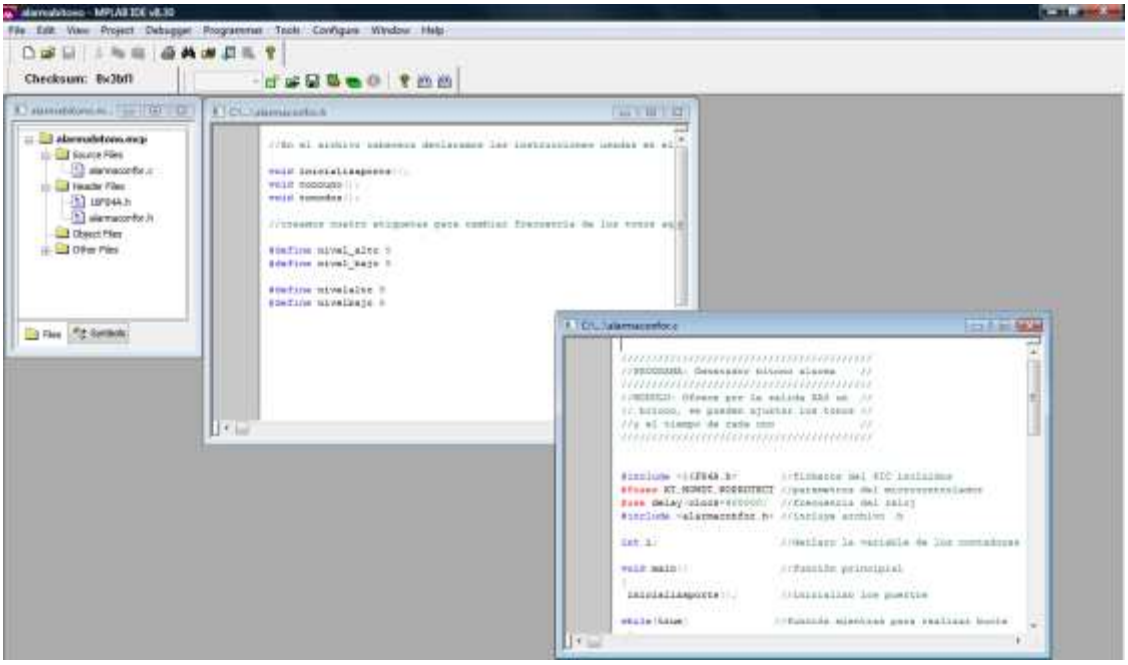
5. ANEXOS DE SOFTWARE.

5.1 TUTORIAL MPLAB.

El editor de texto que utilizamos es el MPLAB IDE v8.30, que es un software que funciona con Windows y nos ayuda a desarrollar aplicaciones para dispositivos digitales y microcontroladores. Este software dispone de un entorno para desarrollar los códigos de programación que posteriormente ensamblaremos y programaremos en el microchip.

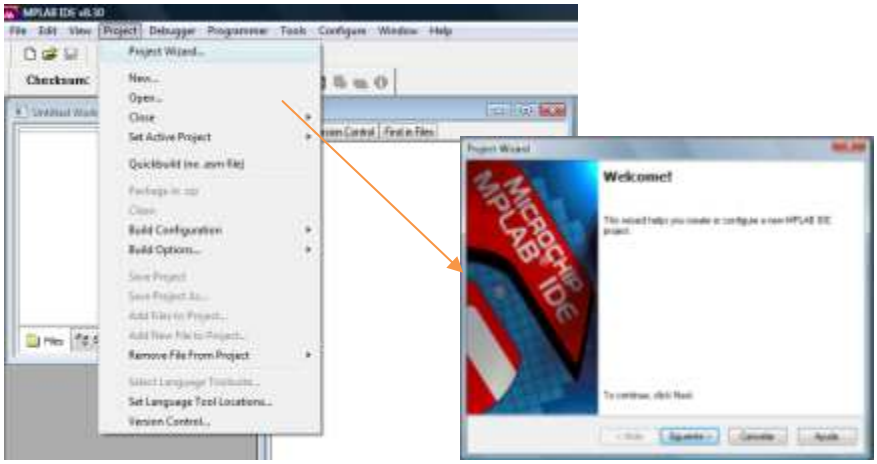
MPLAB dispone de gran cantidad de herramientas que simplifican el proceso de aplicación del código, ya lo escribamos en ensamblador, en C, o en lenguaje básico; y lo convierte en códigos ejecutables por los microchips a programar. En este caso vamos a usar el lenguaje C debido a su mayor simplicidad frente a otros códigos.

La forma de utilización no es compleja; lo primero que se hace es **crear un espacio de trabajo** donde crear y guardar los diferentes archivos de nuestro programa. A los archivos se les adjunta una etiqueta o terminación según el tipo (.c , .h , .asm ,) para así identificarlos. En el tipo de código C, tenemos que crear dos archivos, **un "archivo.c" y un "archivo.h"**; que son el principal y el de cabecera respectivamente. Aquí desarrollaremos el programa en lenguaje C, poniendo en la cabecera el nombre del programa y lo que realiza para poder identificarlo, después desarrollamos el programa.

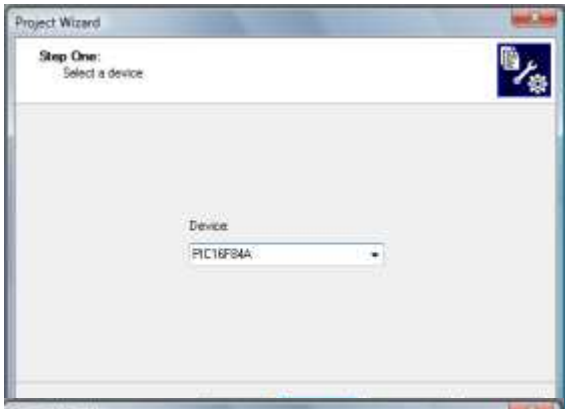


Pantalla del editor de texto con los archivos .c y .h creados y guardados en su carpeta.

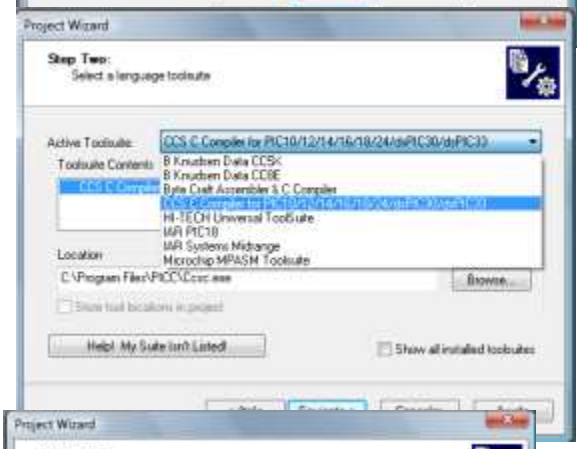
Para crear nuestro programa en el editor de texto MPLAB IDE debemos seguir un procedimiento determinado; en primer lugar se tiene que crear un nuevo proyecto donde guardar los archivos qué necesitemos; para ello el programa consta de un asistente para su creación (Project Wizard). Solo tenemos que seguir los pasos que nos marca:



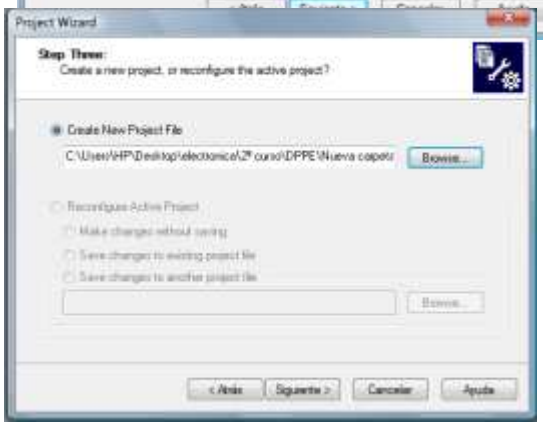
En el primer paso debemos elegir el dispositivo para el cuál creamos el programa, en nuestro caso elegimos el PIC16F876A y pinchamos en “siguiente”.



El segundo paso nos ayuda a elegir el programa con el que compilaremos el programa, pudiendo elegir de una lista desplegable todos los instalados en el ordenador; nosotros elegiremos el CCS C compiler. Pinchamos en “siguiente”.



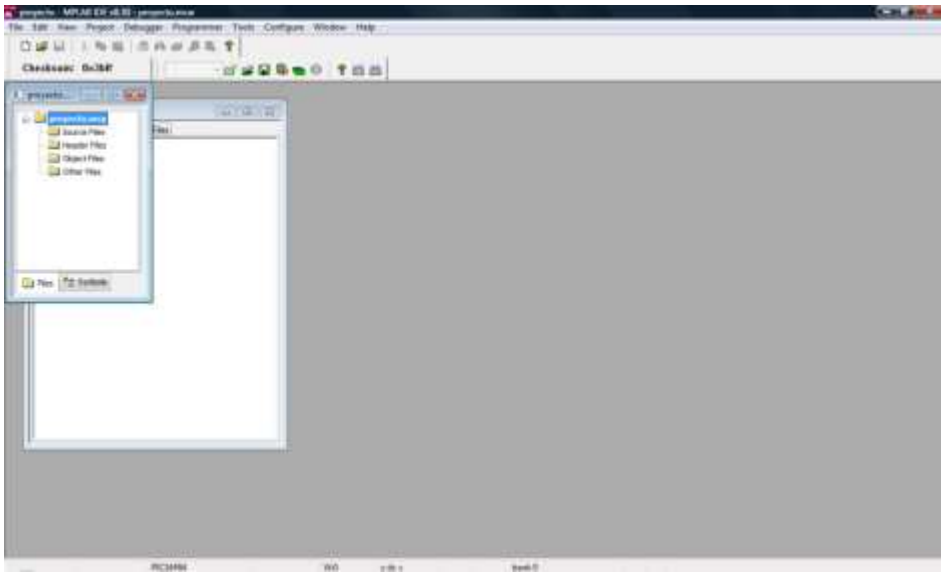
En el tercer paso nos obliga a guardar el proyecto en el disco duro o en la ubicación que queremos; para ello pinchamos en “browse” (vistazo), que nos despliega una ventana del administrador de archivos donde elegir dónde guardarlo. Pinchamos en “siguiente”.



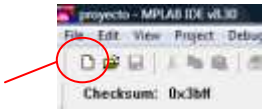
En el cuarto paso nos da la opción de agregar a nuestro nuevo proyecto archivos ya existentes. En nuestro caso no tenemos archivos que agregar. Pinchamos en “siguiente”.

Nuestro nuevo proyecto ya está creado, solo tenemos que finalizar.

Una vez finalizado se nos abre la ventana principal del proyecto.



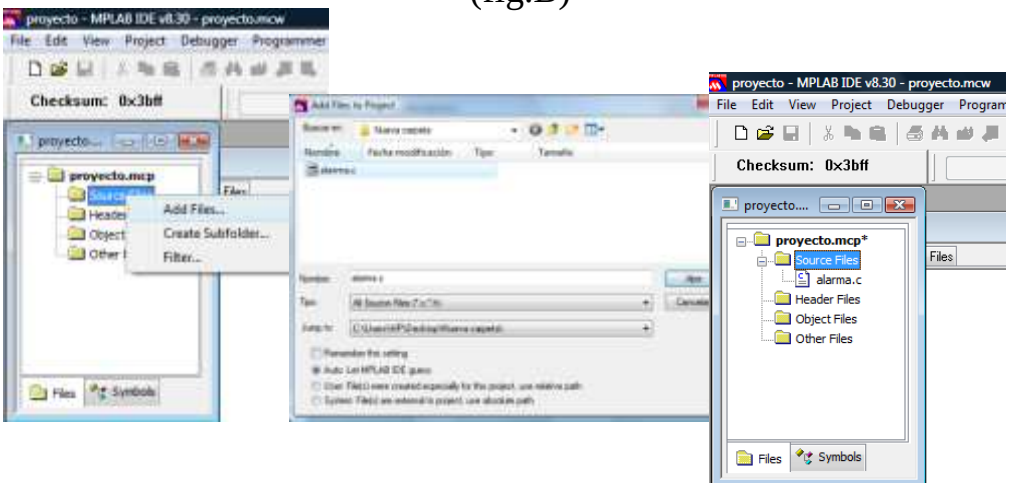
Ahora tenemos que crear los archivos principal (source) y de cabecera (header).



Primero creamos un archivo nuevo: , pinchamos en **“file / save as”** y lo guardamos en la misma carpeta que nuestro proyecto con un nombre identificativo y con la terminación **“.c”** (fig. A) y lo agregamos a la carpeta de archivos principales (source files): para ello pinchamos botón derecho en **“Source files / add files”** y en el menú que se despliega buscamos el archivo **“.c”** y aceptamos (fig.B), el archivo aparecerá en la ventana de **“Source files”** del proyecto:

(fig.A)

(fig.B)



Para crear el archivo de cabecera (header file) tenemos que seguir el mismo procedimiento, guardando este archivo con el mismo nombre que el anterior pero con la terminación “.h” y agregándolo en la carpeta de archivos de cabecera ”Header files” .

Al finalizar tendremos los dos archivos, “archivo.c” y “archivo.h” listos para escribir los códigos del programa:

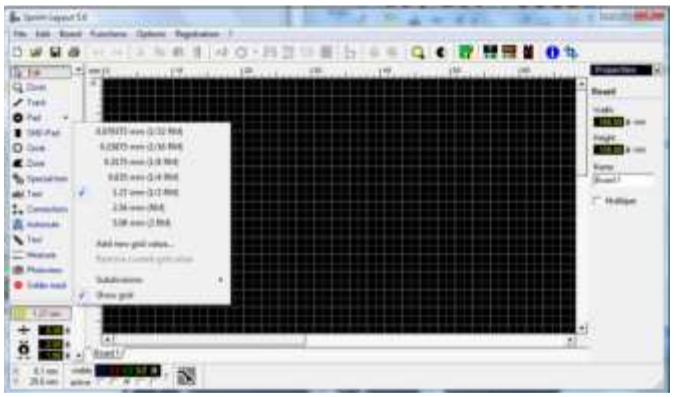
5.2 Manejo de sprint layuot.

Este software nos simplifica el proceso de diseño de nuestra PCB mediante el uso de librerías de encapsulados de componentes, medidas de pads y pistas; se puede trabajar en distintas capas y posee herramienta para comprobar continuidad de las pistas. El manejo es muy sencillo.

Para realizar nuetra PCB, debemos iniciar el programa con el archivo “layout.exe” y seguimos el siguiente proceso:

1. Primero debemos seleccionar las dimensiones de la PCB, en la barra de menus “board”>>”properties”.

2. Después seleccionamos el tamaño del grid que nos ayuda en la precisión del posicionamiento de los componentes.

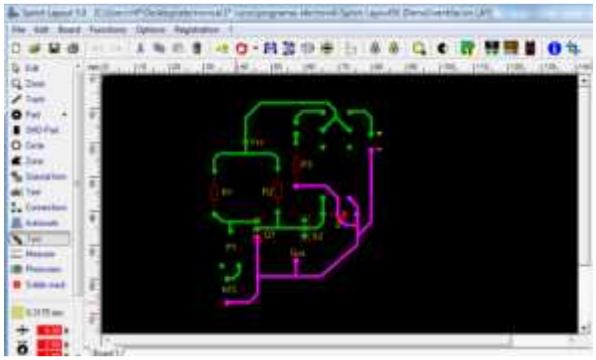


3. Elección de la capa de trabajo entre las siete que ofrece el programa: C1, S1, C2, S2, O, I1, I2. Cada capa sería como una trasparencia superpuesta a la otra.

4. Ahora solo nos queda seleccionar los componentes e ir posicionándolos según nos convenga, para ello elegimos en el menú la opción “makros” que nos despliega una librería con los encapsulados comerciales de los componentes; desde aquí se eligen y se arrastran a nuestro proyecto.

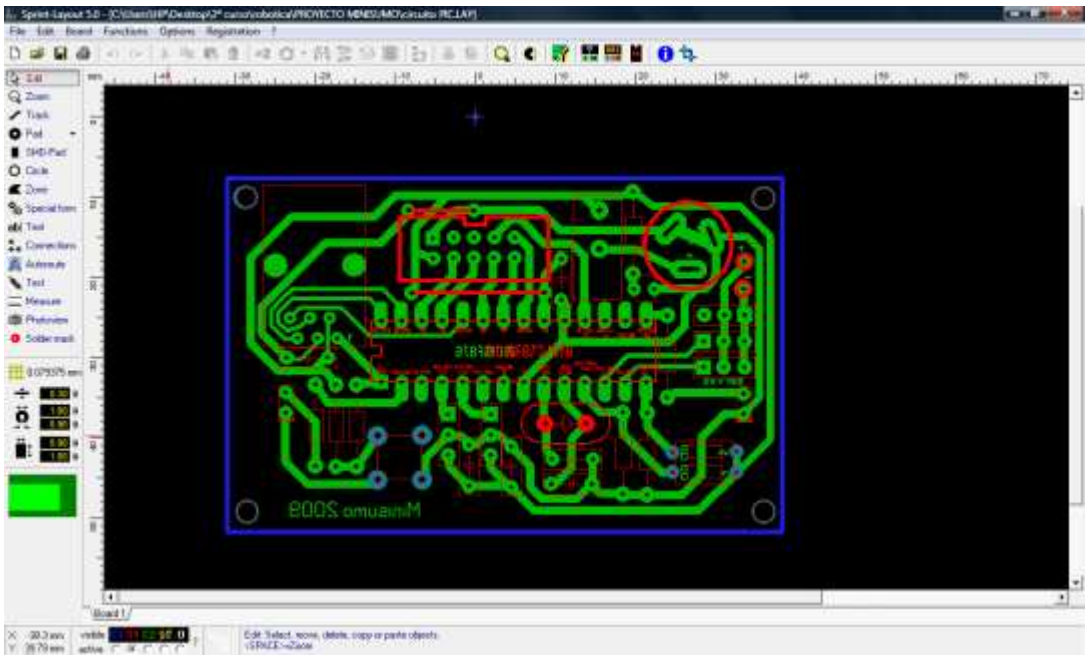
5. En este paso tenemos que conectar los componentes entre sí, para ello seleccionamos “Track”, elegimos el ancho y posicionando el puntero sobre el lugar de comienzo pinchamos y arrastramos, apareciendo nuestra pista; de aquí la llevamos hasta donde se quiera.

6. Ahora comprobamos la correcta conexión de todos los componentes con la función de “Tester” que posee el programa.



7. Una vez revisado procedemos a imprimir nuestro circuito en papel de acetato para crear nuestro fotolito.

Para la realización de este diseño a veces es necesario editar tus propios componentes de manera que coincidan con los componentes que tenemos, por eso es importante para simplificar el trabajo y no tener que deshacer el diseño, el tener todos los componentes en estado físico o en su defecto las medidas exactas de los componentes que compongan nuestro diseño. De esta manera podremos realizar el diseño con total tranquilidad de que los elementos coincidan. En nuestro caso nos hizo falta crear diseños de componentes como el del conector RJ-11, que no se encuentra en la librería del programa entre otros.



5.3 PROGRAMA INTERMITENCIA EN LED.

Archivo “leds.c”

```
//CABECERA DEL PROGRAMA

//////////////////////////////////////////////////////////
//Programa: Intermitencia led                           //
//Plataforma HW: Simulador Proteus (16F877A)            //
//Fecha: 13/10/2009                                     //
//Versión: 0.1                                          //
//Programador: Jorge Granda Alonso                      //
//Descripción: enciende-apaga leds placa minisumo       //
//////////////////////////////////////////////////////////

//PARÁMETROS DE CONFIGURACIÓN

#include <16F876A.h>                    // incluimos librerias del 16f876a
#fuses XT,NOWDT,NOPROTECT,NOLVP         //configuraciones para el pic
#use delay(clock=4000000)               //frecuencia del reloj
#include <leds.h> //incluye archivo .h  //incluimos el archivo ".h"

//COMIENZO DEL PROGRAMA

void main()                             //función principal
   {
      inicializaports();                //función creada para incicializar puertos

    while(1)                            //mientras sea verdad, condicion infinita
    {
     luces();                           //instruccion de intermitencia etiquetada
    }
   }                                    //fin del programa

// DEFINICION DE LAS FUNCIONES

void luces()
{
output_high(pin_a0);                   //seleccionamos los pines en donde están conectados
output_high(pin_c3);                   //los leds y los ponemos a nivel alto y despues a nivel
delay_ms(timi);                        //bajo, con un retraso entre un estado y otro, para crear
output_low(pin_a0);                    //intermitencia de los leds, sino estarian siempre encendidos.
output_low(pin_c3);
delay_ms(timi);
}
void inicializaports()
{
set_tris_c(0x00);                      //configuramos los puertos como salida ya que necesitamos que
set_tris_a(0x00);                      //el pic entregue tensión, por lo que ponemos 0x(hexadecimal)
}                                      //00(numero hexadecimal que es 0000 0000 en binario)
```

Archivo “leds.h”

```
//DECLARAMOS LAS FUNCIONES UTILIZADAS

void inicializaports();

void luces1();

// DEFINIMOS LAS ETIQUETAS

#define timi 100
```

Antes de comenzar a escribir los códigos del programa hay que introducir una cabecera para el programa, donde indicar un nombre y lo que realiza dicho programa.

A lo largo del código del programa es recomendable ir haciendo comentarios para poder entender y hacer entender con mayor facilidad el programa.

Al comienzo ponemos los archivos y parámetros que necesita el PIC.

Declaramos variables y las definimos y comenzamos con el programa principal.

Después se definen las instrucciones utilizadas, en “.c”, y se declaran en “.h”.

En el archivo “.h” también se declaran las posibles etiquetas utilizadas para simplificar el programa. En cualquier caso con las etiquetas creadas se puede cambiar el valor de cada variable solamente cambiando el valor de estas etiquetas.

Para que la parte principal del programa sea más simple, he realizado el programa con bucle infinito y una instrucción etiquetada.

5.4 PROGRAMA MOVIMIENTO SERVOS.

```
//CABECERA DEL PROGRAMA

//////////////////////////////////////////////////////
//Programa: Movimiento servos                       //
//Plataforma HW: Simulador Proteus (16F876A)        //
//Fecha: 13/10/2009                                 //
//Versión: 0.1                                      //
//Programador: Jorge Granda Alonso                  //
//Descripción: mueve motores segun valor            //
//////////////////////////////////////////////////////

//PARÁMETROS DE CONFIGURACIÓN

#include <16F876A.h>
#use delay(clock=4000000)
#fuses NOWDT,HS,NOPUT,NOBROWNOUT,NOCPD,NOWRT,NODEBUG,NOLVP,NOPROTECT

//COMIENZO DEL PROGRAMA

void main()                                  //función principal
{
    setup_counters(RTCC_INTERNAL,RTCC_DIV_1);  //configuracion de los contadores y divisores
    setup_timer_1(T1_DISABLED);              //desabilitamos el contador de timer_1
    setup_ccp1(CCP_PWM);                     //configuramos los puertos del pic para PWM
    setup_ccp2(CCP_PWM);
    setup_timer_2(T2_DIV_BY_4,255,1);        //el contaodr del timer_2 lo configuramos divisiones de 4
                                             //hasta 255 y comienzo en 1.

   while(1)                                  //condición bucle infinito
   {
      set_pwm1_duty(0);                      //instruccion que manda el impulso marcado a el puerto 1 del PWM
      set_pwm2_duty(155);                    //instruccion que manda el impulso marcado a el puerto 2 del PWM
   }
}
//Los valores que se deben introducir y como se deben introducir para que realice los movimientos deseados se explican
//en el apartedo de cálculos 3.3 del proyecto.
//En este programa no es necesario incluir el archivo ".h" ya que no tenemos que definir niguna función ni etiqueta.
//Los valores para los movimientos son:   0 - 33 -155. Con los valores que aparecen en este programa el robot se
//movería hacia adelante o hacia detras dependiendo de la configuración del circuito y los motores.
```

El programa que se presenta corresponde al necesario para poder mover los motores servo en una dirección según los valores que le vayamos introduciendo en las instrucciones para cada impulso pwm en cada motor, que se llaman **set_pwm1_duty(X)**; y **set_pwm2_duty(X)**; sustituiremos la X por el valor de los grados que queramos que gire, es decir los ticks de reloj. Que son en nuestro caso 0 – 92 – 155.

Con estas dos instrucciones controlamos nuestros motores. Y para poder realizar más movimientos en un mismo programa, solo debemos introducir las condiciones que queramos que se cumplan en cada caso para introducir después la instrucción con un valor diferente para que nuestro robot realice un movimiento u otro.

Este programa no necesita de archivo ".h" ya que no tenemos funciones que declarar ni etiquetas que definir.

5.5 PROGRAMA SENSORES CNY70.

Archivo “sensores.c”

```
//CABECERA DEL PROGRAMA

//////////////////////////////////////////////////////
//Programa: sensores cny70                          //
//Plataforma HW: Simulador Proteus (16F876A)        //
//Fecha: 13/10/2009                                 //
//Versión: 0.1                                      //
//Programador: Jorge Granda Alonso                  //
//Descripción: activa motores segun estado de sensores //
//////////////////////////////////////////////////////

//PARÁMETROS DE CONFIGURACIÓN

#include <16F876A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP,NOPUT,NOBROWNOUT,NOCPD,NOWRT,NODEBUG
#device adc=10
#use delay(clock=4000000)
#include <sensores.h>                //incluye archivo .h

void main()
{
      setup_counters(RTCC_INTERNAL,RTCC_DIV_1);
      setup_timer_1(T1_DISABLED);
      setup_ccp1(CCP_PWM);                        //configuraciones para PWM
      setup_ccp2(CCP_PWM);
      setup_timer_2(T2_DIV_BY_16,255,1);

      inicializaports();                          //configuracion de los puertos

       if(!sens5&&!sens2&&!sens1&&!sens6)         //condicion de los 4 sensores CNY70
     {
     set_pwm1_duty(155);                         //instruccion para los motores
     set_pwm2_duty(0);
     }
    if((!sens5||!sens2)&&(sens1||sens6))         //sensores delanteros 2(der) y 5(izq)
     {                                           //sensores traseros 6(izq) y 1(der)
      set_pwm1_duty(155);
      set_pwm2_duty(0);                          //155-0 alante rapido      al contrario hacia atras
     }

    if((sens5||sens2)&&(!sens1||!sens6))
     {
      set_pwm1_duty(0);
      set_pwm2_duty(155);
     }
    if(sens5&&sens2&&sens1&&sens6)
     {
      set_pwm1_duty(92);
      set_pwm2_duty(92);
     }
}
```

Archivo “sensores.h”

```
//DECLARAMOS FUNCIONES
void inicializaports();

//etiquetamos los sensores segun el puerto en que esten colocados
#define sens1  input(pin_c5)
#define sens2  input(pin_c6)   //sensores delanteros 1(izq) y 6(der)
#define sens3  input(pin_b2)     //sensores traseros 2(der) y 5(izq)
#define sens4  input(pin_b1)
#define sens5  input(pin_c7)
#define sens6  input(pin_c4)

//setup_ccp2(CCP_PWM); // Seleccionamos PWM para el módulo CCP2.
//setup_timer_2(T2_DIV_BY_1,255,1); // Para cristal de 4 mhz, T2_DIV a 1. Para calcular el periodo:
// T = (4/frecuencia cristal)*T2_DIV*(PR2+1)
// T = (4/4000000)*1*(255+1) = 0.000256seg f=3.90625 MHz
// Para calcular el tiempo en nivel alto, dado que usamos un LONG, sería:
// Ancho_Impulso = duty_PWM1*T2_DIV*(1/frecuencia del cristal)
// Para saber maximo valor para duty_PWM1, duty cycle -> 100% :
// 0.000256seg/(1*(1/4000000)) = 1024
// Max_duty_PWM1=1023
```

En este programa ya hacemos uso de los sensores cny70, que como nos dan una señal digital a nivel alto o bajo según lo que lean, y por el puerto en el que esté cada uno configurado, solo tenemos que crear las condiciones con los cuatro sensores (2 atrás y 2 adelante). Para simplificar el programa creamos etiquetas para llamar a cada sensor de modo que quedarían como **sens1,sens2,sens3,sens4**. En cada condición pondremos los estados que deberán tener los sensores para que se cumpla la condición y por lo tanto pase a ejecutar la instrucción que abarca.

En este programa se realizan las condiciones siguientes:

-Si los **4 sensores** leen **negro** el robot avanza.

-Si los **delanteros** leen **negro** y los **traseros** leen **blanco**, el robot avanza.

-Si los **delanteros** leen **blanco** y los **traseros** leen **negro**, el robot retrocede.

-Si los **4 sensores** leen blanco, el robot se para.

Con este programa bien planteado el robot no debe salirse del tatami de combate.

5.6 PROGRAMA SENSORES GP2D12.

Archivo “sensors.c”

```
//CABECERA DEL PROGRAMA

//////////////////////////////////////////////////////////
//Programa: Sensores GP2D12                             //
//Plataforma HW: Simulador Proteus (16F877A)            //
//Fecha: 13/10/2009                                     //
//Versión: 0.1                                          //
//Programador: Jorge Granda Alonso                      //
//Descripción: Comprueba funcionamiento de sensores     //
//////////////////////////////////////////////////////////

//PARÁMETROS DE CONFIGURACIÓN

#include<16f876a.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP,NOPUT,NOBROWNOUT,NOCPD,NOWRT,NODEBUG
#device adc=10
#use delay(clock=4000000)
#include<gp2d12.h>

//DECLARACIÓN DE VARIABLES

long valor_leido;
float voltios;

//COMIENZO DEL PROGRAMA

void main()                          //funcion principal
{
 setup_adc_ports(AN0_AN1_AN3);       // RA3 como entrada analógica
 setup_adc(ADC_CLOCK_INTERNAL);      // Fuente de reloj para conversión interno

 inicializaports();                  //configuracion de los puertos

 while(true)                         //condicion bucle infinito
 {
  set_adc_channel(3);                    // Habilitación del canal AN3
  delay_ms(20);                          // Retardo para  asegurar la conversión
  valor_leido=read_adc();                //conversion para el lectura del pic
  voltios= 5.0 * valor_leido/ 1024.0;    //conversion del pic para los voltios

  output_low(PIN_A0);                    //ponemos leds a cero por si empiezan encendidos
  output_low(PIN_c3);

   if (voltios<1)                        //condicion, si el sensor ofrece menos de 1 voltio
    {                                    //es decir si detecta objeto a media distancia o
    output_high(PIN_C3);                 //no detecta objeto

    }
   else
    {                                    //sino, si detecta objeto
    output_high(PIN_A0);

    }

 }
}

//DEFINICION DE FUNCIONES

 void inicializaports()
   {
    set_tris_a(0x1E);                 //CONFIGURACION DE LOS OUERTOS ENTRADA O SALIDA
    set_tris_b(0xFF);
    set_tris_c(0x00);
   }
```

Archivo “sensors.c”

```
void inicializaports();
```

En este programa se especifica como configurar un sensor infrarrojo analógico gp2d12. Este tipo de sensores tiene salida analógica por lo que debemos configurar los puertos analógicos en que se vaya a ubicar dicho sensor o sensores.

Primero debemos definir las variables que va a necesitar el pic para comunicarse con el sensor y hacer las conversiones.

Para activar los puertos analógicos basta con dos instrucciones de “setup”, que se detallan en el programa y que son instrucciones estándar.

Una vez activado el puerto analógico del sensor, se debe elegir el canal del sensor, dejar un intervalo de tiempo para que el pic haga la conversión del canal que está leyendo y después las instrucciones para el cálculo del voltaje recibido con otras dos instrucciones que se detallan en el programa.

Una vez todo configurado se realizan las condiciones, que al ser por el puerto analógico, serán directamente el rango de tensiones que nos ofrece el sensor según la distancia al objeto de 0.5 a 2.5 voltios. Según la tensión recibida realizara una instrucción u otra.

Este programa controla 1 solo sensor. Si detecta objeto enciende led izquierdo, sino detecta objeto enciende el led derecho.

5.7 PROGRAMA COMPLETO MINISUMO.

Archivo “principal.c”

```
//CABECERA DEL PROGRAMA

////////////////////////////////////////////////////////////////////
///////////////PROGRAMA PRINCIPAL DE ROBOT MINISUMO/////////////////
////////////////////////////////////////////////////////////////////
//FECHA: 11-02-2010/////////////////////////////////////////////////
//VERSION: 0.1//////////////////////////////////////////////////////
//PROGRAMADOR: JORGE GRANDA ALONSO//////////////////////////////////
////////////////////////////////////////////////////////////////////
//DISEÑO PROGRAMA PARA PODER DESARROLLAR DIFERENTES FACETAS SEGUN////
//////////////////LA POSICION DE LOS MICROSWITCH/////////////////////
////////////////////////////////////////////////////////////////////


//PARÁMETROS DEL PIC

#include <16F876A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP,NOPUT,NOBROWNOUT,NOCPD,NOWRT,NODEBUG
#device adc=10
#use delay(clock=4000000)
#include <Principal.h> //incluye archivo .h

//VARIABLES

int16 valor_leido,valor;
float delante,detras;

//COMIENZO DEL PROGRAMA


void main()
   {

      wait_5_segundos();                          //funcion de espera reglamentaria

      setup_adc_ports(AN0_AN1_AN3);               // configuracion puertos analogicos
      setup_adc(ADC_CLOCK_INTERNAL);              // Fuente de reloj para conversión interno

      setup_counters(RTCC_INTERNAL,RTCC_DIV_1);   //configuraciones para PWM
      setup_timer_1(T1_DISABLED);
      setup_ccp1(CCP_PWM);
      setup_ccp2(CCP_PWM);
      setup_timer_2(T2_DIV_BY_16,255,1);

      inicializaports();                         //funcion configuracion de los puertos


          while(minisumo)                       //etiqueta para posicion de los micro switch
             {
              config_sensors();                 //funcion configura sensores analogicos AN2 y AN3
              busqueda();                       //funcion busqueda oponente
              ataque();                         //funcion ataque oponente
              defensa();                        //funcion defensa
              wait();                           //funcion paro, para coger el robot
             }
          while(rastreo)                        //etiqueta para posicion de los micro switch
             {
              config_sensors();                 //funcion configura sensores analogicos AN2 y AN3
              paseo();                          //funcion paseo libre
             }
    }
```

Este es el programa completo que gobierna nuestro minisumo. Después de haber visto parte por parte cada módulo, los hemos combinado para que todos los periféricos del minisumo estén interrelacionados. Para ello nos servimos de funciones creadas por nosotros y etiquetas que nos simplifican mucho el programa principal. Que es el que está a la izquierda. En las páginas siguientes se van definiendo las funciones usadas y que también forman parte del archivo “principal .c”

Después de la cabecera, parámetros del PIC y variables usadas comenzamos el programa. Lo primero que tiene que aparecer es el retardo obligatorio de todos los robots minisumo con su función etiquetada. Después ponemos las configuraciones iniciales de los sensores analógicos y del PWM para los motores, además de la configuración de los puertos (I/O).

A continuación, según la posición de los micro switches de la PCB principal, el programa escoge un camino u otro, **RASTREO** es para usar robot en cualquier superficie y que no choque con nada, y **MINISUMO** es para que realice el programa de lucha sobre tatami.

En el archivo “.h” se declaran etiquetas y funciones. Y la explicación de cada función se puede ver en el programa con los comentarios que lo detallan.

Definición de funciones del archivo "principal.c".

```
void inicializaports()                          //configuramos los puertos como entrada o salida
   {                                            //según nos convenga para activar todos los perifericos
    set_tris_a(0x1F);                           //de entrada o salida. Sensores-leds-motores.
    set_tris_b(0xFF);
    set_tris_c(0xF0);
   }
void ataque()                                  //funcion con los parámetros para que el robot ataque al objetivo
{
 if(delante>0.4&&detras<0.5)                   //si sensor analógico delantero detecta y trasero no.
   {
    output_high(pin_c3);                       //enciende led
    if(!sens5&&!sens2&&!sens1&&!sens6)         //y si los cny70 estan los cuatro en negro
     {
     set_pwm1_duty(155);                       //minisumo avanza rapido
     set_pwm2_duty(0);
     }
    if((!sens5||!sens2)&&(sens1||sens6))       //si los cny70 traseros detectan blanco
     {
      set_pwm1_duty(155);                      //minisumo avanza rapido
      set_pwm2_duty(0);
     }
    if((sens5||sens2)&&(!sens1||!sens6))       //si los cny70 delanteros detectan blanco
     {
      set_pwm1_duty(0);                        //minisumo retrocede rapido
      set_pwm2_duty(155);
     }
    if(sens5&&sens2&&sens1&&sens6)             //si los cuatro cny70 detectan blanco
     {
      set_pwm1_duty(92);                       //minisumo se para
      set_pwm2_duty(92);
     }
  }
}


void busqueda()                                //funcion con los parámetros para que el robot busque al objetivo
{
 if(delante<0.5&&detras<0.5)                   // si ningun sensor analógico detecta objeto.
  {
   if(!sens5&&!sens2&&!sens1&&!sens6)          //y si cuatro cny70 detectan negro
     {
     set_pwm1_duty(92);                        //minisumo gira sobre si mismo
     set_pwm2_duty(155);
     }
   if((!sens5||!sens2)&&(sens1||sens6))        //y si los cny70 traseros detectan blanco
     {
      set_pwm1_duty(155);                      //minisumo avanza rapido
      set_pwm2_duty(0);
     }
    if((sens5||sens2)&&(!sens1||!sens6))       //y si los cny70 delanteros detectan blanco
     {
      set_pwm1_duty(0);                        //minisumo retrocede rapido
      set_pwm2_duty(155);
     }
   if(sens5&&sens2&&sens1&&sens6)              //y si los cuatro cny70 detectan blanco
     {
      set_pwm1_duty(92);                       //minisumo se para
      set_pwm2_duty(92);
     }
  }
}
void defensa()                                 //funcion con los parámetros para que el robot se defienda del objetivo
{
 if(delante<0.5&&detras>0.8)                   //si sensor analógico trasero detecta objeto.
  {

   output_high(pin_c3);                        //enciende led.
   if(!sens5&&!sens2&&!sens1&&!sens6)          //y si cuatro cny70 detectan negro
     {
      set_pwm1_duty(155);                      //minisumo gira sobre si mismo en otro sentido
      set_pwm2_duty(92);
     }
   if((!sens5||!sens2)&&(sens1||sens6))        //y si los cny70 traseros detectan blanco
     {
      set_pwm1_duty(155);                      //minisumo avanza rapido
      set_pwm2_duty(0);
     }
    if((sens5||sens2)&&(!sens1||!sens6))       //y si los cny70 delanteros detectan blanco
     {
      set_pwm1_duty(0);                        //minisumo retrocede rapido
      set_pwm2_duty(155);
     }
    if(sens5&&sens2&&sens1&&sens6)             //y si los cuatro cny70 detectan blanco
     {
      set_pwm1_duty(92);                       //minisumo se para
      set_pwm2_duty(92);
     }

  }
}
void wait()                                    //funcion de parada para coger el robot
{
 if(delante>1&&detras>1)                       //si ambos sensores analogicos detectan objeto
  {
   set_pwm1_duty(92);                          //minisumo se para
   set_pwm2_duty(92);
  }
}
```

```
void config_sensors()                        //funcion para configurar ambos sensores analógicos
{
   set_adc_channel(3);          // Habilitación del canal AN3
   delay_us(40);                // Retardo para  asegurar la conversión
   valor_leido=read_adc();
   delante=5.0*valor_leido/1024.0;

   set_adc_channel(2);          // Habilitación del canal AN2
   delay_us(40);                // Retardo para  asegurar la conversión
   valor=read_adc();
   detras=5.0*valor/1024.0;
}
void paseo()                                 //funcion para usar robot libremente
{
 if(delante>0.8&&detras<1)                   //si sensor analogico delantero detecta objeto
   {
    set_pwm1_duty(155);                      //minisumo gira sobre si mismo
    set_pwm2_duty(92);
   }
 if(delante<1&&detras<1)                     //si ningun sensor analogico detecta
   {
    set_pwm1_duty(155);                      //minisumo avanza
    set_pwm2_duty(0);
   }
 if(delante<1&&detras>1)                     //si sensor analogico trasero detecta objeto
   {
    set_pwm1_duty(92);                       //minisumo gira en sentido opuesto
    set_pwm2_duty(155);
   }
 if(delante>1&&detras>1)                     //si ambos sensores analogicos detectan
   {
    set_pwm1_duty(97);                       //minisumo se para
    set_pwm2_duty(97);
   }
}

void wait_5_segundos()                      //funcion de espera reglamentaria
{
 output_high(pin_c3);                       //enciende led naranja
 delay_ms(5000);                            //espera 5 segundos
 output_low(pin_c3);                        //apaga led naranja
}
```

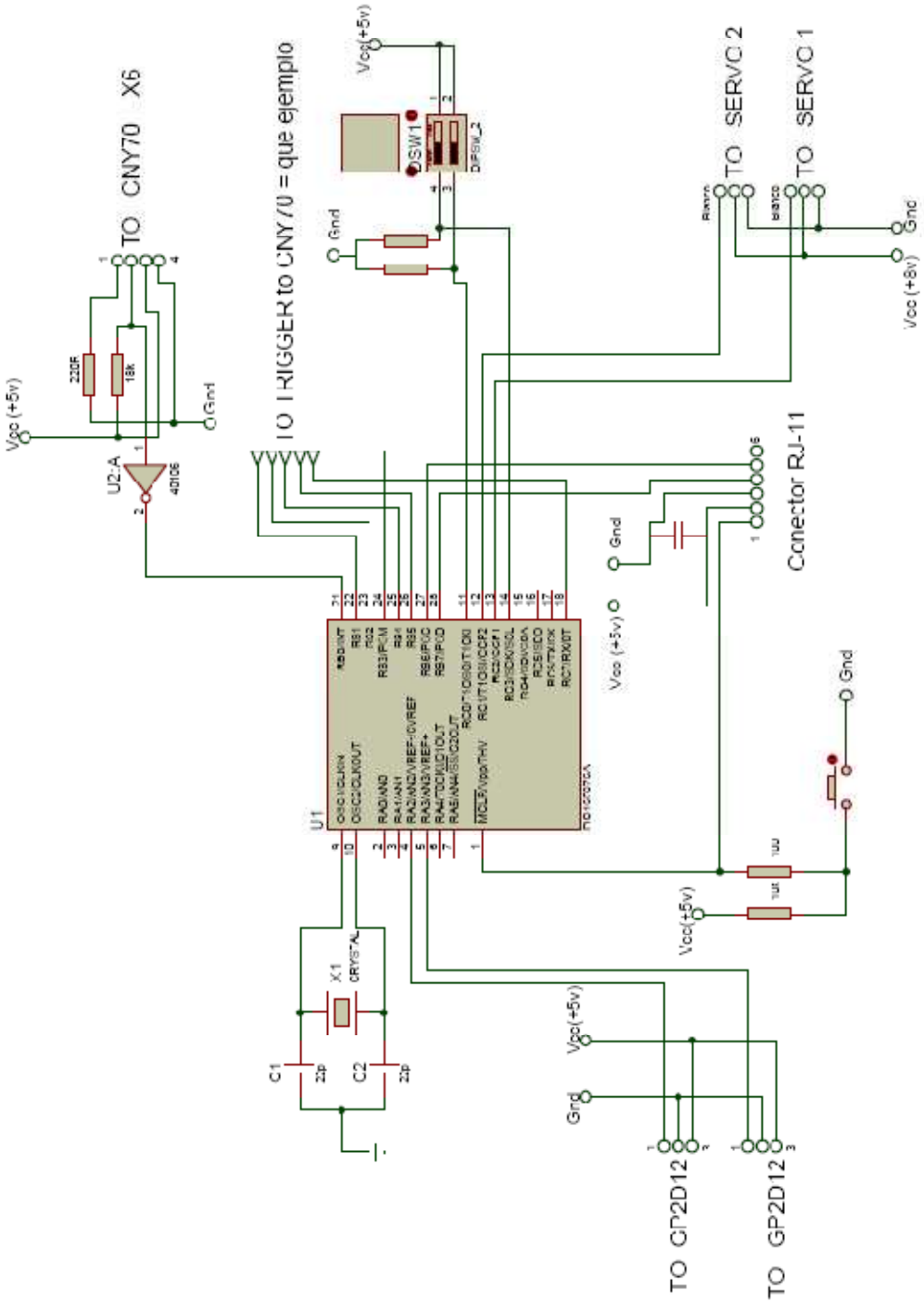
Archivo “principal.h”

```
//DECLARACIÓN DE FUNCIONES

void inicializaports();
void ataque();
void busqueda();
void defensa();
void wait();
void paseo();
void config_sensors();
void wait_5_segundos();

//DECLARACIÓN DE ETIQUETAS

#define rastreo !input(pin_a4)&&!input(pin_a5)      //ambos micro switch a 0.
#define minisumo input(pin_a4)&&input(pin_a5)       //ambos micro switch a 1.
#define time 200
#define timi 2000
#define sens1  input(pin_c5)                        //valor de cada etiqueta de
#define sens2  input(pin_c6)                        //los sensores digitales
#define sens3  input(pin_b2)
#define sens4  input(pin_b1)
#define sens5  input(pin_c7)
#define sens6  input(pin_c4)


//sensores CNY70 traseros sens2(der) y sens5(izq)
//sensores CNY70 delanteros sens1(izq) y sens6(der)
//const pwm ticks_PULSO_MINIMO  =  0;
//const pwm ticks_PULSO_MEDIO   =  94;
//const pwm ticks_PULSO_MAXIMO  = 155;
```

6. Esquema

Circuito

general

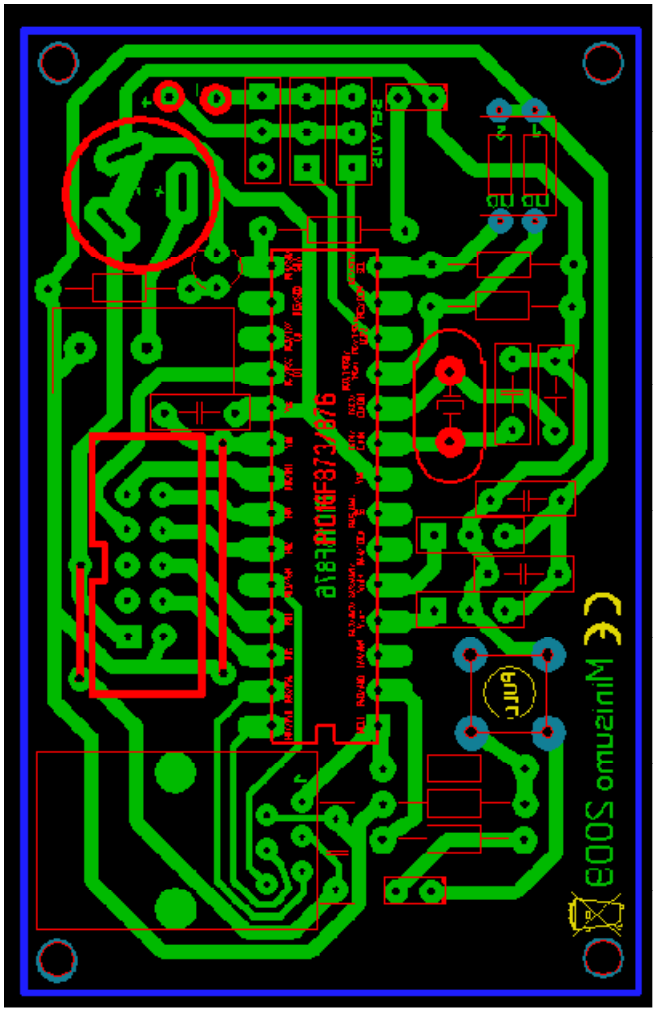


Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 1
Denominación del dibujo	6.1 Esquema circuito general				

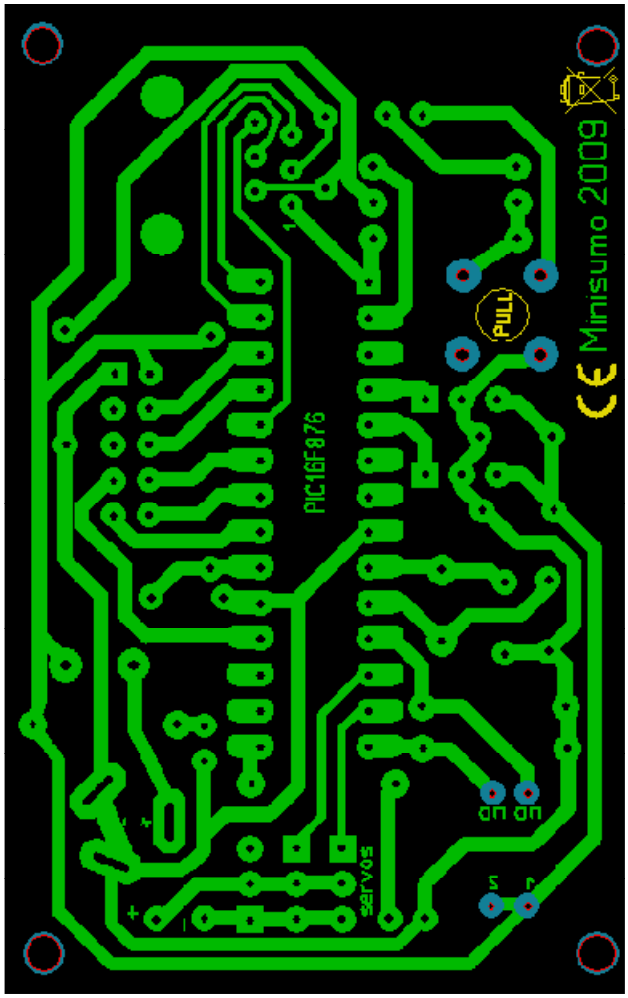
7. Planos

de la

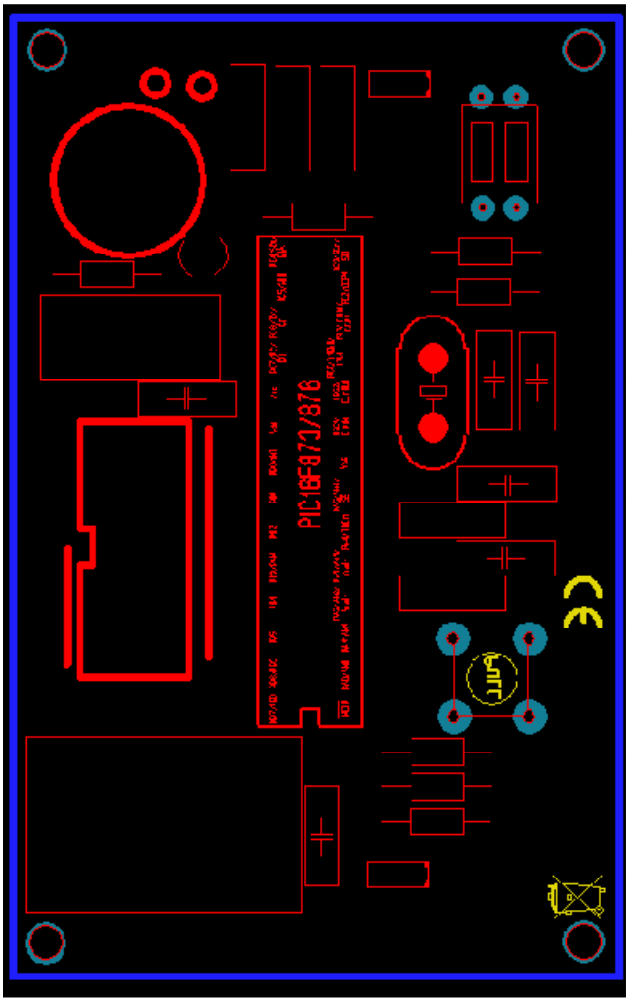
PCB



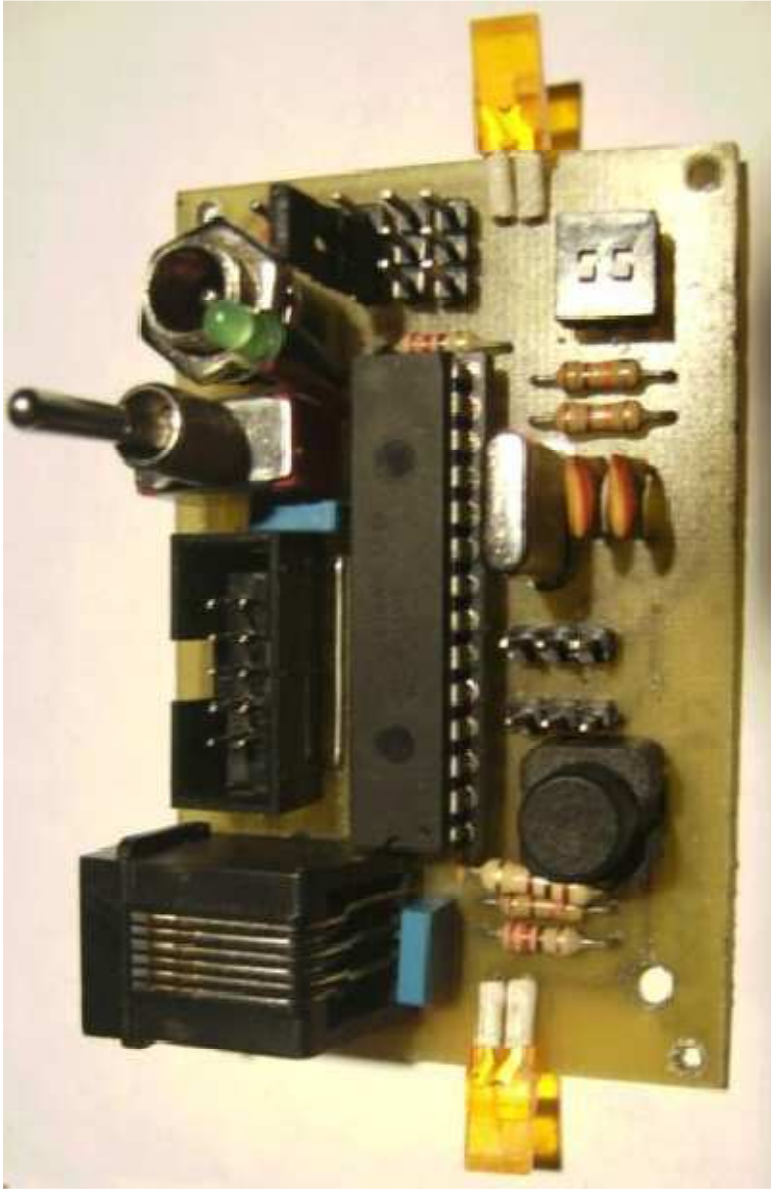
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 2
Denominación del dibujo	7.1 Plano PCB control completo				



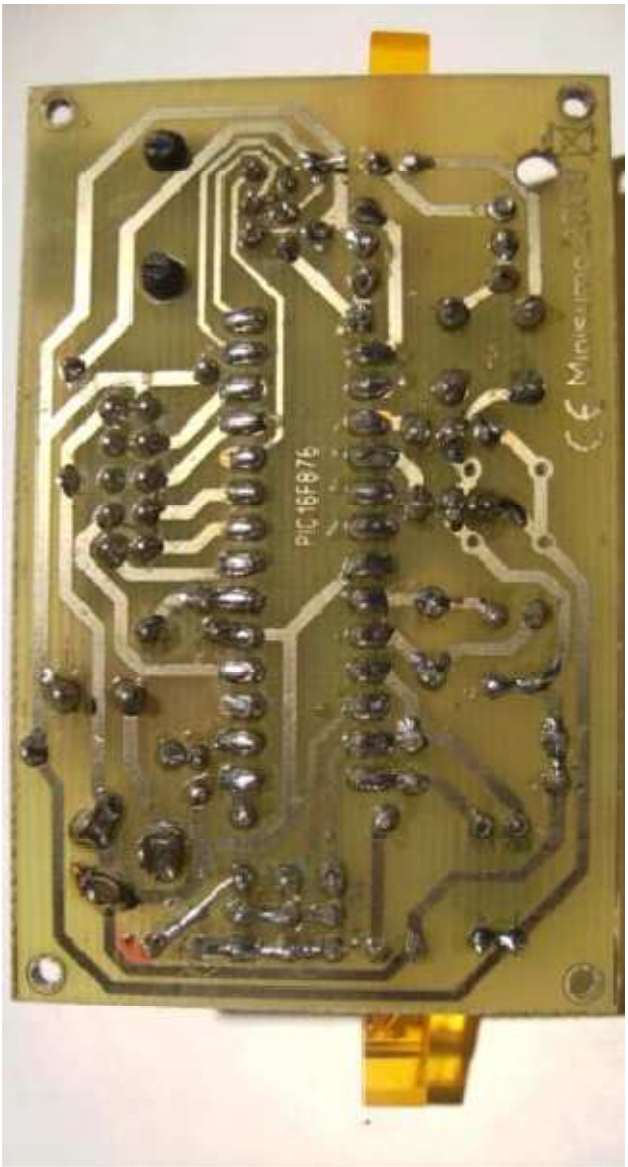
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 3
Denominación del dibujo	7.2 Plano PCB control cara pistas				



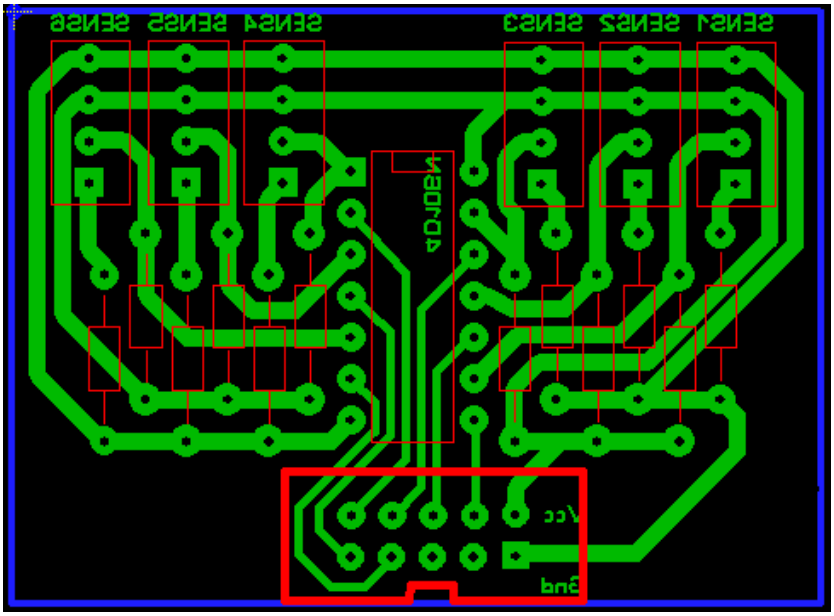
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 4
Denominación del dibujo	7.3 Plano PCB control cara componentes				



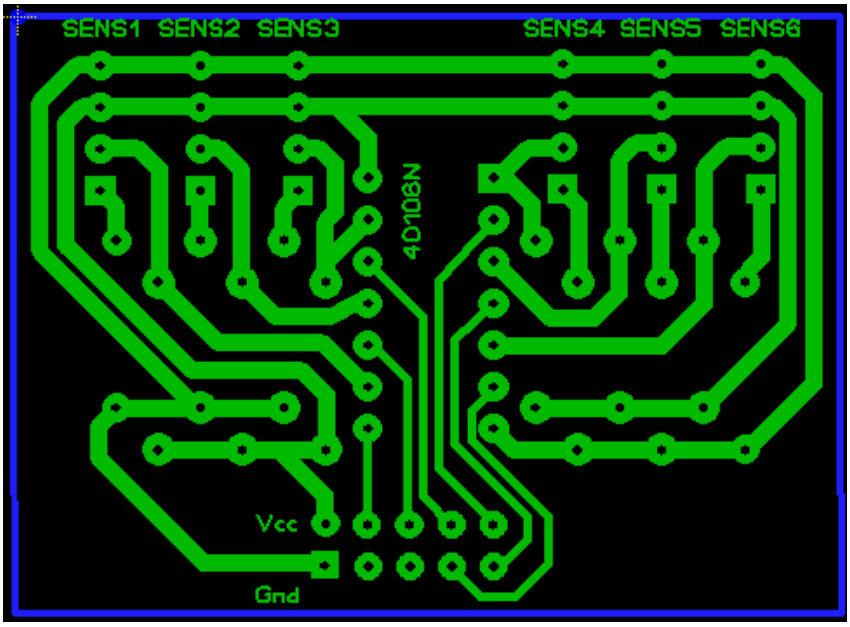
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 5
Denominación del dibujo	7.4 Aspecto real PCB control cara componentes				



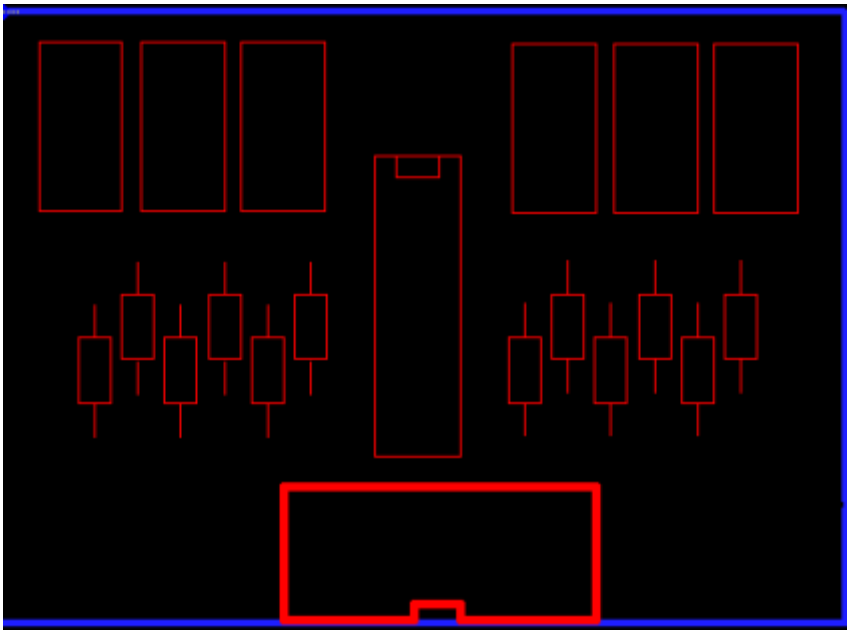
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 6
Denominación del dibujo	7.5 Aspecto real PCB control cara pistas				



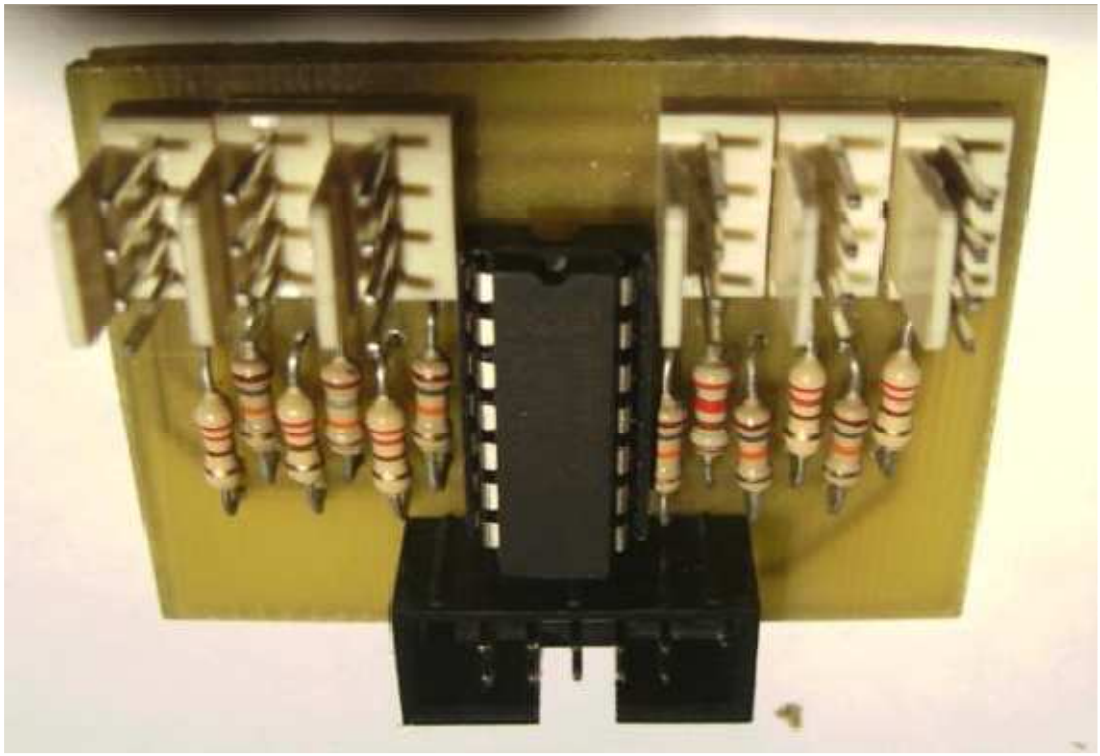
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 7
Denominación del dibujo	7.6 Plano PCB sensores completo				



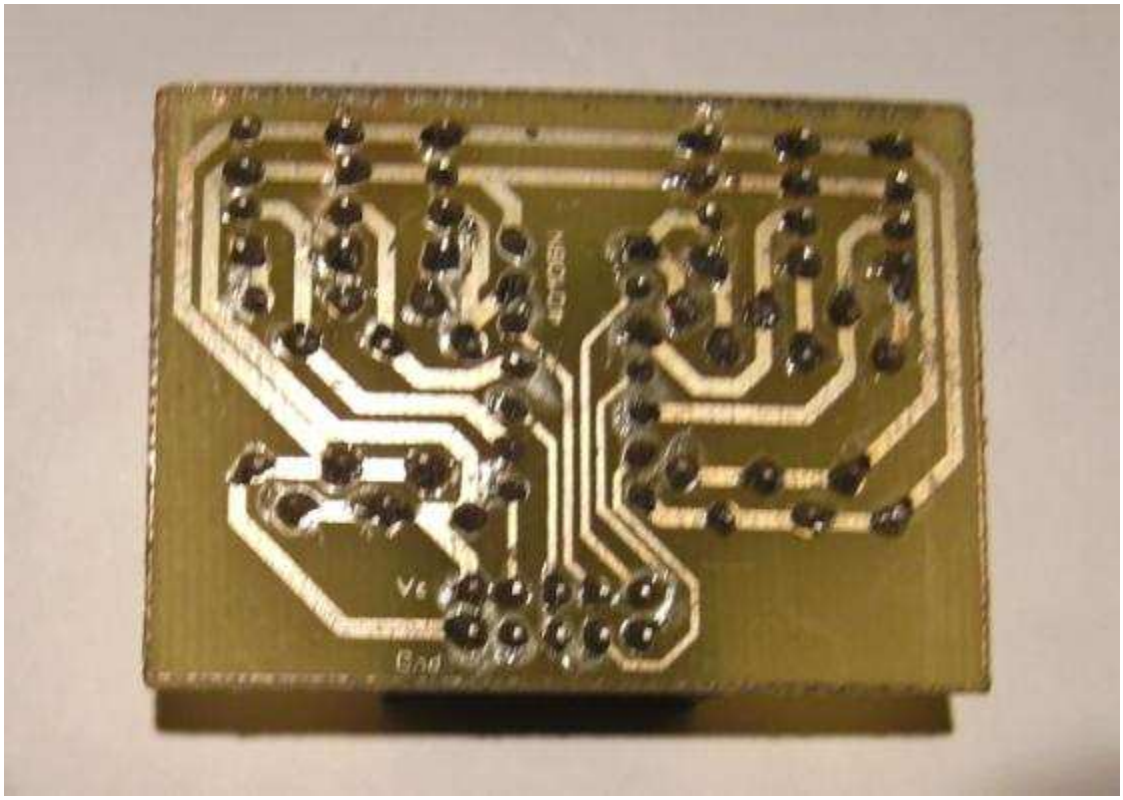
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 8
Denominación del dibujo	7.7 Plano PCB sensores cara pistas				



Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 9
Denominación del dibujo	7.8 Plano PCB sensores cara componentes				



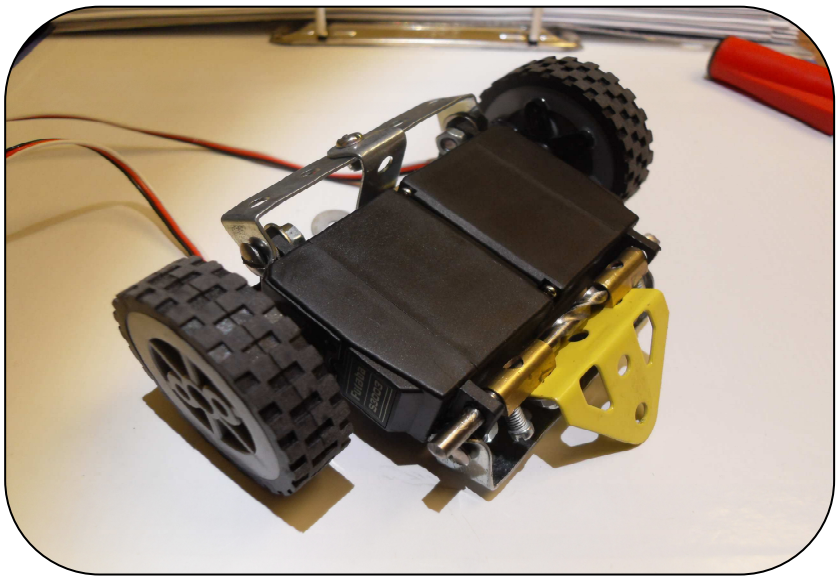
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 10
Denominación del dibujo	7.9 Aspecto real PCB sensores cara componentes				

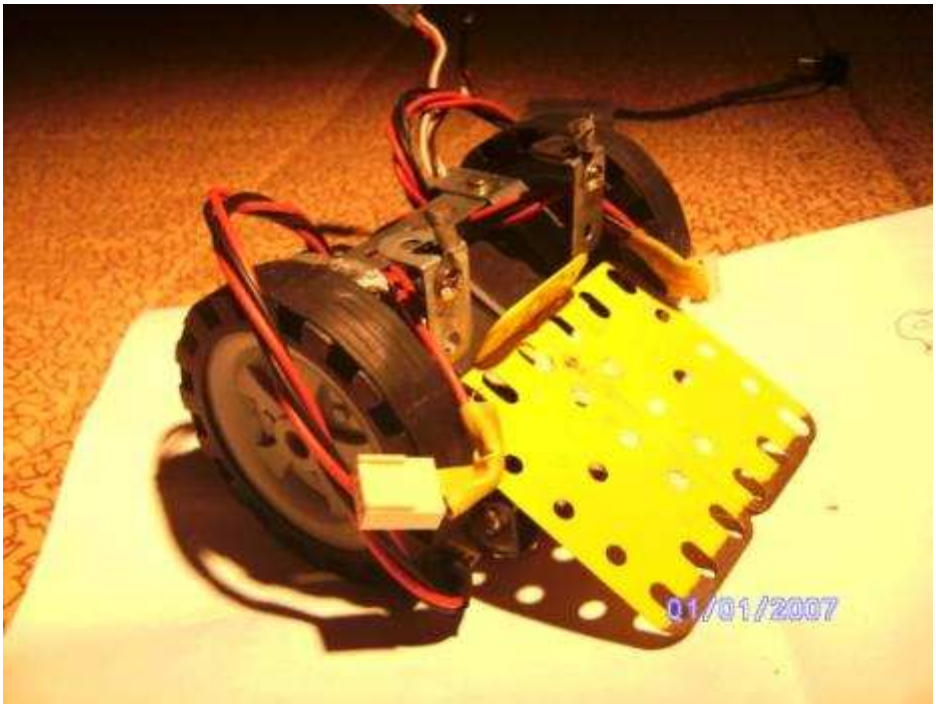


Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 11
Denominación del dibujo	7.10 Aspecto real PCB sensores cara pistas				

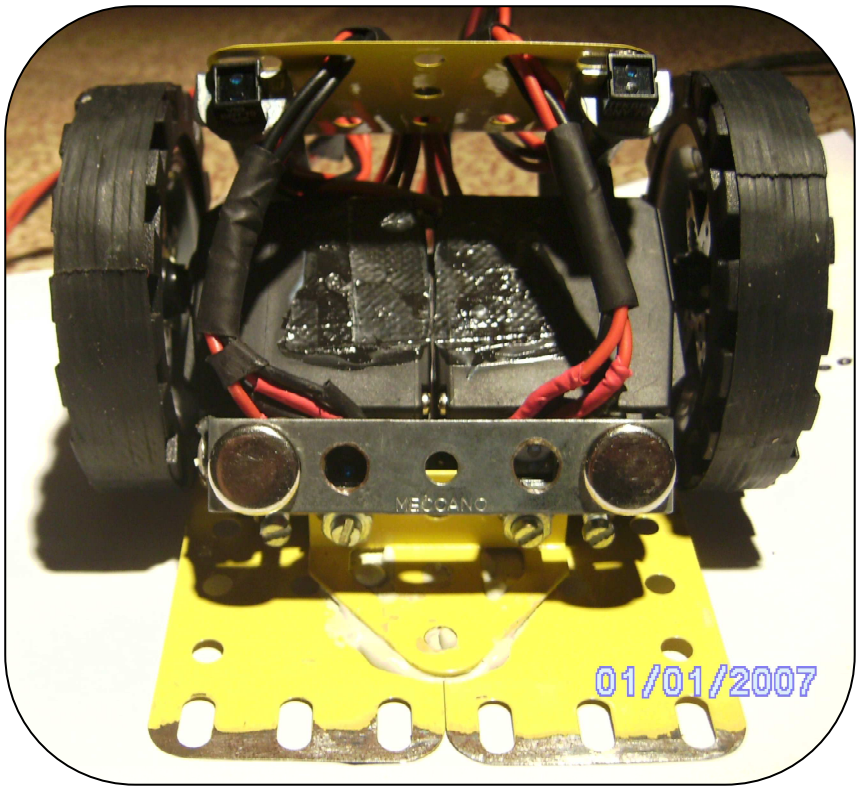
8. Aspecto

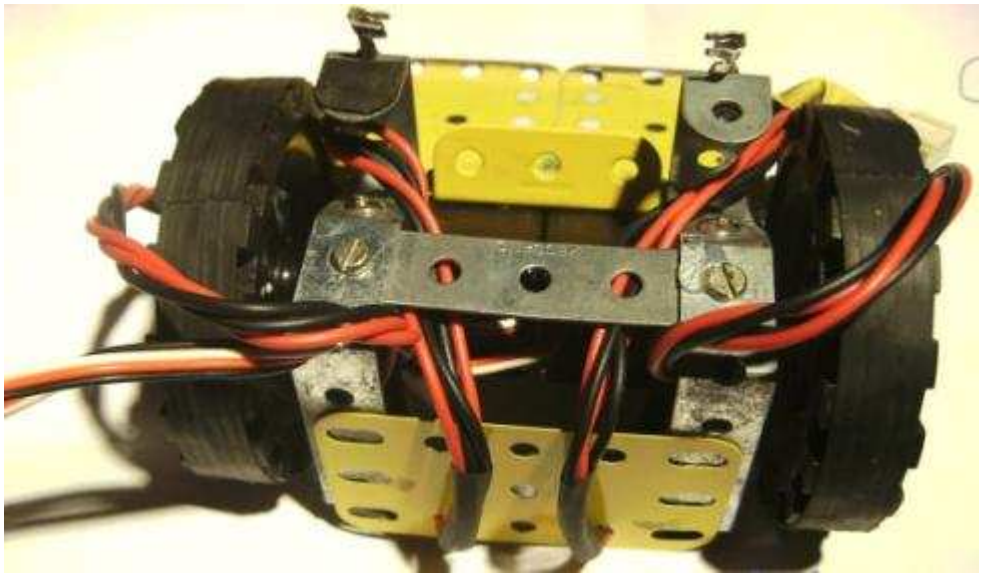
Físico



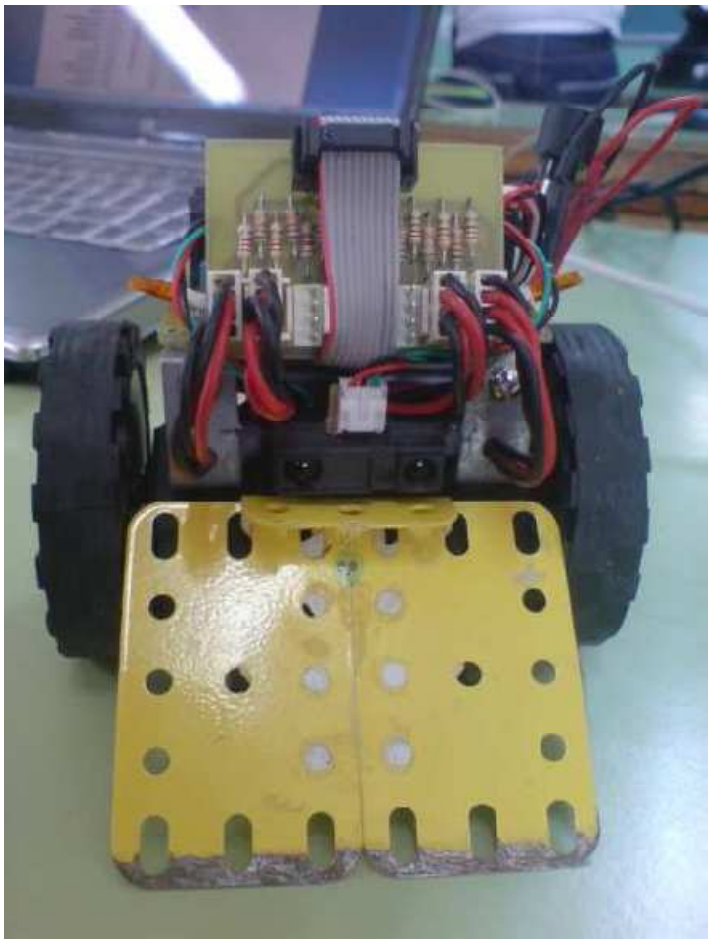


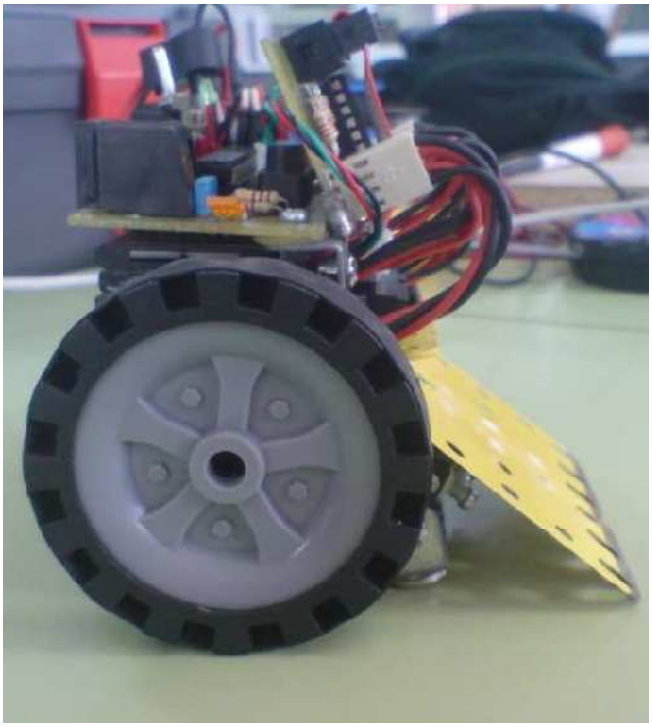
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 12
Denominación del dibujo	8.1 Chasis de minisumo				

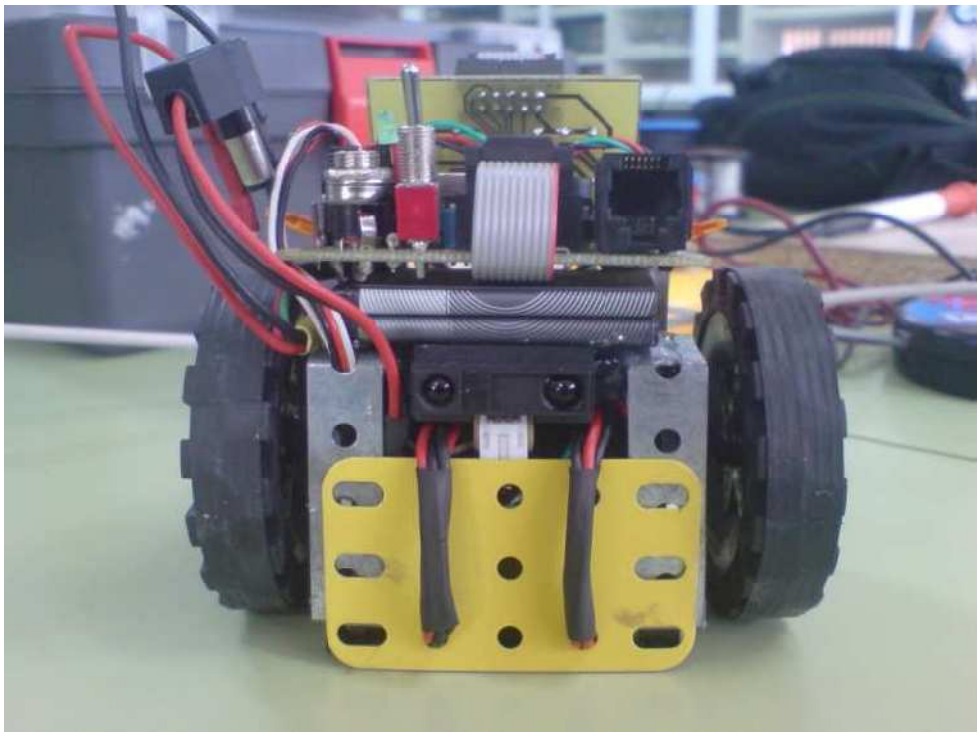




Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 13
Denominación del dibujo	8.2 Chasis de minisumo				

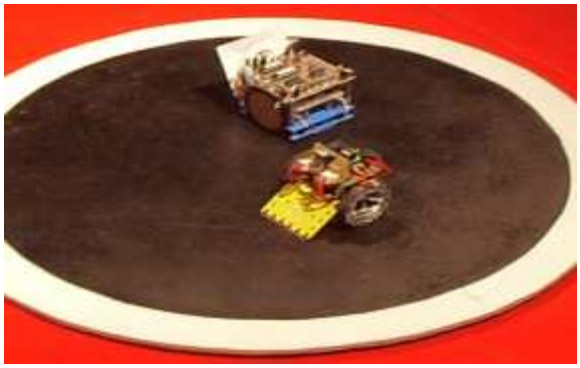


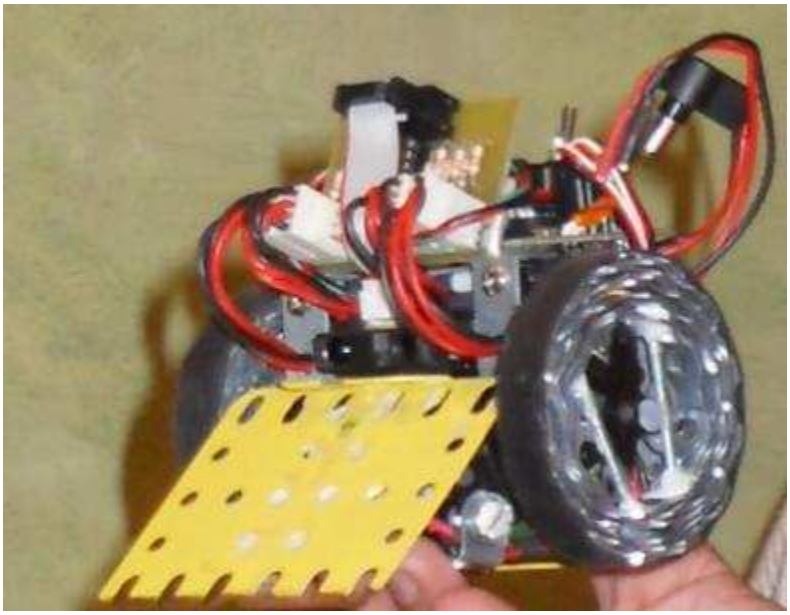




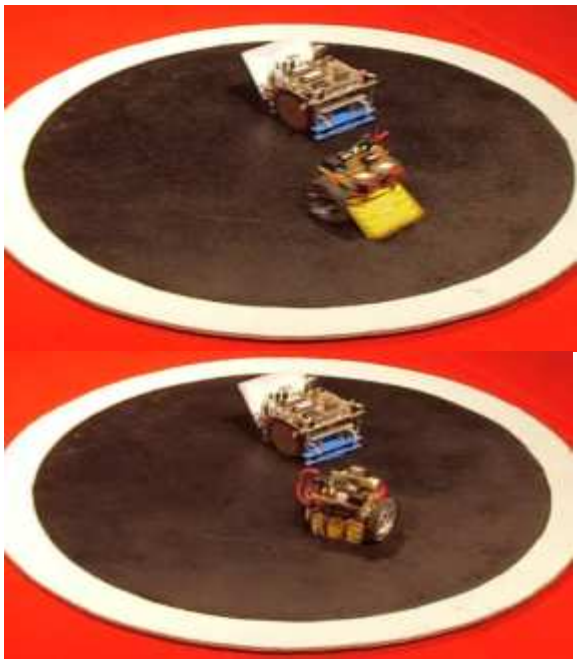
Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	20/05/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 14
Denominación del dibujo	8.3 Robot minisumo completo				

1. INICIO PAUSA 5 SEGUNDOS

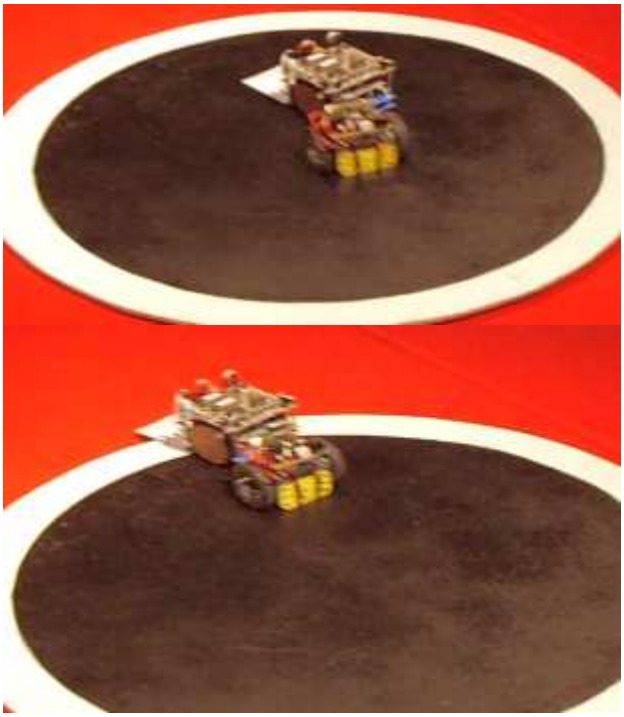




2. BÚSQUEDA DE OPONENTE



3. ATAQUE



4. PERMANENCIA EN TATAMI

Diseño	Fecha	Nombre	Firma	Departamento de electrónica	
	02/06/2010	Jorge Granda Alonso	S28		
Comprobado				Robótica	Lámina nº- 15
Denominación del dibujo	8.4 Robot minisumo completo				

Pliego de condiciones

En este apartado se contemplan las condiciones administrativas y legales que debe cumplir nuestro proyecto, las condiciones de calidad y entrega que deben cumplir los materiales y herramientas, las condiciones en los tiempos de ejecución de las diferentes fases y del proyecto completo y por último las condiciones de pago del proyecto.

9. Condiciones generales (Normativas).

El proyecto consta de unas condiciones generales que las podemos separar en condiciones administrativas y condiciones legales.

Dentro de las condiciones administrativas del proyecto podemos decir que este proyecto ha sido comprobado en todos sus aspectos técnicos y se ha apoyado en las condiciones legales vigentes. Hemos realizado verificaciones o ensayos de tipo eléctrico como aislamiento eléctrico del equipo; se han realizado verificaciones de las condiciones de trabajo del minirobot, de los efectos que tienen las capacidades, de la influencia de los efectos electromagnéticos y de la influencia radio eléctrica y del efecto de la luz natural y artificial sobre el funcionamiento de los sensores.

Este proyecto ha sido aprobado tras concluir las verificaciones técnicas con éxito.

Una vez supervisado por el colegio de ingenieros técnicos industriales y que haya sido aceptado por industria o por el organismo que asume las trasferencias, y solo entonces, podremos afirmar que este proyecto está dentro del marco legal que lo capacita para la puesta en marcha de la circuitería que se describe en todo el ámbito de aplicación.

Dentro del marco legal y teniendo en cuenta que se trata de un equipo electrónico, este proyecto se debe realizar al amparo de la legislación vigente y de acuerdo con lo estipulado en el RBT (reglamento de baja tensión), de acuerdo con lo estipulado en el reglamento de verificaciones eléctricas y teniendo en cuenta las leyes sobre prevención de riesgos laborales.

10. Condiciones de materiales y Equipos.

Las condiciones de los materiales necesarios para este proyecto, se deben adaptar teniendo en cuenta las tensiones a las que trabaja cada sección del robot como la de control y la de tracción. Estas secciones, deben soportar trabajar a estas tensiones y corrientes de tensión, teniendo en cuenta que las características de los componentes sobrepasen los máximos que vaya tener el equipo para poder gozar de margen.

Los materiales como los componentes, se deben ajustar en lo máximo posible a las especificaciones que se muestran en la lista de componentes, ya que previamente se han ajustado a las necesidades de nuestra circuitería, el uso de unos componentes no contemplados puede ocasionar un mal funcionamiento de nuestro sistema de alarma.

Las PCBs de nuestro minirobot, tanto la de control como la de sensores deberá estar fabricada con materiales que resistan las tensiones y las temperaturas de trabajo de nuestro dispositivo. Para ello usaremos placa fotosensible de fibra de vidrio que es más recomendable frente a las de baquelita.

En cuanto a las herramientas y material necesario para el montaje y posterior comprobación de nuestro proyecto sobre robot luchador de minisumo, necesitaremos unas herramientas específicas para poder cumplir nuestro propósito, estas herramientas deberán tener las calidades mínimas exigidas para poder desempeñar su función con total garantía y sin entorpecer el curso del proyecto. Los equipos que necesitemos para la medida y comprobación de las diferentes secciones de nuestro dispositivo de robótica deben estar preparados para soportar las corrientes y tensiones que debemos manejar:

- +8v en continua.
- +5v en continua.
- La intensidad en continua no sobrepasa los 800mA en sección de tracción.
- La intensidad en continua no sobrepasa los 750mA en sección de control.

Estos equipos serán:

- Polímetro Digital.
- Osciloscopio analógico o digital.

En cuanto a la sección de alimentación, tenemos varias opciones viables:

- Una es utilizar una batería común y realizar regulación de la tensión.
- Otra es usar dos baterías que ofrezcan distintas tensiones y aislar los dos circuitos, para evitar cortos y derivaciones.

11. Condiciones de ejecución.

En este apartado debemos contemplar todos los detalles para la correcta ejecución de nuestro proyecto con carácter electrónico. Especificando los tiempos de ejecución de cada bloque y los materiales y recursos humanos necesarios para su desarrollo.

En primer lugar debemos realizar la elección del soporte para nuestro diseño, la PCB, que en nuestro caso va estar compuesta por Fibra de Vidrio ante su mejor comportamiento sobre otros materiales como por ejemplo la baquelita. Esta placa será fotosensible y a una cara ya que queremos revelar solo la cara de pistas.

Estas dos placas que vamos a necesitar para el proyecto tendrán unas medidas mínimas de:

- 7cm. x 4.5cm. la PCB principal.
- 5cm. X 3.6cm la PCB de sensores.

El proceso de ejecución de nuestras PCB de sección de control y de sensores será el siguiente:

- **Realización de los fotolitos-**

  -El fotograbado utiliza una fotomecánica y grabado químico para eliminar el sustrato de la placa de cobre.

  -Se procede con la impresión del fotolito del diseño de pistas previamente diseñado mediante SPRINT-LAYUOT, teniendo en cuenta que el fotolito de pistas se coloca en negativo. Usaremos impresora de chorro y acetatos para impresora para crear los fotolitos.

- **Proceso insolación y atacado-**

  -Usando placa fotosensible positiva mono capa, insolamos la cara de pistas, utilizando una insoladora, y dejando cada fotolito durante 3 minutos, quedando marcado el dibujo de cada fotolito en cada placa y se pasa al baño revelador con sosa cáustica y agua, que durará unos 4 minutos.

  -Con un baño en proporción de 25% agua fuerte - 25% agua oxigenada - 50% agua, se revela la PCB en 5-8 minutos máximo, haciendo atención en el baño de no rozar con exceso una placa con otra para no deteriorar la impresión.

  -Al finalizar el baño de ácido, se enjuaga la PCB y se comprueban posibles fallos de calidad en el diseño.

- **Perforado de la PCB-**

  -Realizamos los orificios para todos los componentes que lo necesiten y con el diámetro adecuado para no forzar el patillaje de los componentes al ser introducidos y viendo que nos quede superficie de soldadura donde aplicar la cantidad necesaria

de estaño. Este proceso necesita de un taladro de precisión con soporte vertical y utilizaremos brocas para metal de diámetros comprendidos entre 0.8mm y 5mm

-Realizamos un control en la calidad del diseño, comprobando los contactos y posibles defectos tras la realización de los orificios.

- **Proceso de estañado-**

-Montamos los componentes de manera que podamos ir probando el funcionamiento por bloques para depurar posibles fallos. Prestar atención con el aporte de estaño y en los tiempos de soldadura para evitar soldaduras frías y otros posibles fallos por malos contactos. En esta etapa nos serviremos de un hilo de estaño de 1mm de diámetro al 60% de aleación de SN y de un soldador de punta sólida de 26W de potencia.

- **Diseño y montaje de chasis-**

-Realizamos chasis con materiales reciclados de otras utilidades como chapas, ruedas, etc. Realizaremos montaje sobre los motores para reducir espacio y debemos adaptar el chasis al peso y medidas reglamentarias, ya que si no sería no apto. Debemos preparar anclajes para las PCB y sensores, dejando estos a 0.5cm máximo de la superficie del suelo.

- **Pruebas y verificaciones-**

-Realizamos montaje completo y realizamos pruebas reales de funcionamiento de cada tarea o programa del minirobot y comprobaremos tensiones, como son:

- Alimentación, donde realizaremos comprobación de las tensiones que nos ofrecen las baterías y en cada componente. Estas tensiones serán de +5V y +8V.
- Sección de control, donde comprobaremos las tensiones y señales que debe haber en cada componente.
- Prueba de programa básico funcionamiento de leds de la PCB.
- Verificar movimientos de los motores.
- Verificar funcionamiento y distancias de los fotosensores activos.
- Verificar funcionamiento de sensores analógicos y programación.
- Prueba de programas independientes de cada sensor.
- Prueba de todos los periféricos ensamblados y en un mismo programa.
- Prueba de permanencia en el tatami.
- Prueba de detección de objeto y expulsión del recinto.
- Prueba de tracción y peso desplazable.
- Prueba de medidas y peso reglamentario.

12. Condiciones económicas.

En este apartado de condiciones contempla las diferentes opciones para mantener nuestro proyecto dentro de los límites de pago, para poder realizar un producto competitivo en el mercado y a la misma vez de calidad.

Gracias a que nuestro diseño se ha realizado con materiales reutilizados el coste es mucho menor que si se adquirieran dichos materiales en establecimientos.

Para poder mantener el proyecto dentro de los márgenes económicos establecidos se deben cumplir todas las fases de la planificación y programación de acuerdo con lo establecido, ya que alguna variación en el desarrollo del proyecto podría ocasionar costes adicionales.

Las condiciones económicas del proyecto deben contemplar las calidades mínimas de los componentes a utilizar, no pudiendo ignorarlas.

Las condiciones de pago de este proyecto pueden ser:

Pago al contado.

Pago a plazos.

Presupuesto

13. Precios unitarios y desglosados.

Los precios se representan en euros y no llevan el IVA incluido.

-PCB-

Componente	Unidades	Precio Unitario	Importe
Pcb fotosensible positiva mono capa	1	6.80	6.80

-RESISTENCIAS-

Componente	Etiqueta	Unidades	Precio Unitario	Importe
Resistencia 10kΩ-1/4w	R	3	0.050	0.15
Resistencia 18kΩ-1/4w	R	6	0.050	0.30
Resistencia 220Ω-1/4w	R	8	0.050	0.40
Resistencia 100Ω-1/2w	R	1	0.080	0.080

-CONDENSADORES-

Componente	Etiqueta	Unidades	Precio Unitario	Importe
Cond. Cerámico27pF / 10v	C	2	0.150	0.300
Cond. Poli.Metal MKT 10nf / 10v	C	2	0.300	0.600

-CIRCUITOS INTEGRADOS-

Componente	Etiqueta	Unidades	Precio Unitario	Importe
Microcontrolador PIC 16F876A DIL28	U1	1	6.20	6.20
Trigger Schmitt inversor 40106 CMOS 6 DIL 14	U6	1	0.350	0.350

-VARIOS-

Componente	Etiqueta	Unidades	Precio Unitario	Importe
Conectores jumpers 20	CN	1	0.40	0.40
Leds planos naranjas	led	2	0	0
Cristal de cuarzo 4MHz	X1	1	0.50	0.50
Micro switch 2contactos	program	1	0.25	0.25
Micro pulsador	RESET	1	0.15	0.15
Interruptor 2 contacto	ON/OFF	1	0.50	0.50
Leds 3mm verde	D1	2	0.150	0.15
conector jack baterías hembra	hembra	1	0.30	0.30
conector jack baterías macho	macho	1	0.50	0.50
Conectro RJ-11 sobre placa	rj11	1	0.30	0.30
Conector bus datos DIL10 hembra	bus	2	0.25	0.50
Conector bus datos DIL10 macho	bus	2	0.25	0.50
Cable plano 10 hilos	hilo	1	0.50	0.50
conector DIL4 sobre placa	sens	6	0.20	1.20
Baterias li-ion 3.6v 800mA	motor	2	0.00	0.00
Baterias NiMh 1.2V 750mA	control	4	0.00	0.00
Servomotores FUTABAS3003	mot	2	8.50	17.00
GP2D12	ana	2	7.50	15.00
CNY70	sens	4	0.80	3.20
Chapas metalicas	mec	-	0.00	0.00

14. Presupuestos parciales de cada módulo.

14.1-Módulo de sensores-				
Componente	**Etiqueta**	**Unidades**	**Precio Unitario**	**Importe**
Trigger Schmitt inversor 40106 CMOS 6 DIL 14	tg	1	0.35	0.35
Conector DIL4	sens1-6	6	0.20	1.20
Conector bus datos DIL10 hembra	con	1	0.25	0.25
Resistencia 10kΩ-1/4w	R	6	0.050	0.30
Resistencia 220Ω-1/4w	R	6	0.050	0.30
			TOTAL	**2.40€**

14.2-Módulo de control-				
Componente	**Etiqueta**	**Unidades**	**Precio Unitario**	**Importe**
Microcontrolador PIC 16F84A DIL18	U1	1	6.20	6.20
Conectores jumpers 20	j	1	0.40	0.40
Conector bus datos DIL10 hembra	bus	1	0.25	0.25
conector jack baterías hembra	J	1	0.40	0.40
Cristal de cuarzo 4MHz	X	1	0.50	0.500
Cond. Cerámico27pF / 50v	C	2	0.150	0.300
Micro switch 2contactos	S	1	0.25	0.25
Resistencia 100Ω-1/4w	R		0.050	0.050
Resistencia 220Ω-1/4w	R	3	0.050	0.150
Micro pulsador	RESET	1	0.15	0.150
Resistencia 10kΩ-1/4w	R	3	0.05	0.150
Leds planos naranjas	L	2	0.00	0.00
Interruptor 2contactos	ON/OFF	1	0.50	0.500
Leds 3mm verde	D2	1	0.150	0.150
Cond. Poli.Metal MKT 10nF / 10v	c	2	0.30	0.60
			TOTAL	**10.05€**

14.3-Recursos materiales-			
Componente	**Unidades**	**Precio Unitario**	**Importe**
Rollo estaño	1	2.50	2.50
Acetato de folio A4	2	0.60	1.20
Servomotores FUTABAS3003	2	8.50	17.00
GP2D12	2	7.50	15.00
Baterias Li-ion y NiMh	2	0.00	0.00
Materiales Chasis	-	0.00	0.00
CNY70	4	0.50	2.00
Agua oxigenada	1	0.60	0.60
Agua fuerte	1	0.60	0.60
Sosa caustica	1	1.50	1.50
		TOTAL	**40.40€**

15. Presupuesto general.

Este es el presupuesto definitivo del proyecto sin tener en cuenta el coste de los recursos humanos como sería la mano de obra del técnico, que se ha estimado en, 5 horas, entre insolación de la PCB, revelado, atacado con ácidos, perforado, inserción de componentes y soldadura y pruebas de funcionamiento, siendo remunerada a 30€ la hora de mano de obra de un técnico, sería un total de 150€.

PRESUPUESTO GENERAL			
Concepto	**Cantidad**	**Precio Unitario**	**Importe**
Pcb Fotosensible mono capa	1	6.80	6.80
Módulo de sensores	1	2.40	2.40
Módulo de control	1	10.05	10.05
Recursos humanos/horas	**5**	**30.00**	**150.00**
Recursos materiales	1	40.40	40.40
		Total Bruto	**209.65€**
		Importe IVA16%	**33.54€**
		Total	**243.19€**