ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO INDUSTRIAL

Título del proyecto:

PROYECTO ARDUINO: SUMO ROBÓTICO

Guillermo Barbadillo Villanueva

Alfredo Pina Calafí

Pamplona, 10 de Septiembre de 2012

Índice general

Parte I

Estado del Arte

Capítulo 1

Objeto y Alcance del Proyecto

El proyecto consiste en diseñar y construir un robot para competir en la categoría de MiniSumo Robótico.

Además se va a generar la documentación suficiente para que un futuro grupo de robótica tenga la base para construir un robot de similares características.

Los pasos a seguir en el desarrollo del proyecto serían los siguientes:

1. Estado del arte. Estudiar el reglamento de la competición, analizar los mejores robots construidos hasta la fecha, ver cuales son los puntos importantes a la hora de construir un robot ganador...
2. Elección de componentes. Rastrear el mercado para comprobar cuales son las opciones disponibles para cada elemento del robot. Comparar las especificaciones de los componentes, ver los requerimientos necesarios...
3. Diseño del robot. Una vez hemos elegido los componentes diseñar la estructura del robot (chasis, carrocería y rampa). Estudiar los materiales disponibles y sus técnicas constructivas. Diseñar el circuito eléctrico del robot.
4. Montaje del robot. Llevar a cabo todo lo planeado y documentarlo adecuadamente para que pueda ser utilizado en proyectos venideros.
5. Puesta a punto. Diseñar una serie de programas para comprobar el correcto funcionamiento de cada componente del robot.
6. Programación. Crear el software que integre todos los elementos del robot y le confiera un comportamiento inteligente
7. Mejoras. Participar en campeonatos de sumo y con la experiencia ver lo que se podría mejorar.

Capítulo 2

Sumo Robótico

El sumo de robots es una competición en la cual dos robots luchan por echar al otro y evitar ser echados de un circulo de juego llamado dohyo.

2.1. Categorías

Hay dos criterios principales de clasificación: el tipo de robots y su modo de funcionamiento. Atendiendo al tipo de robot se pueden clasificar en:

- MicroSumo. Robots de menos de 100g de peso y con un área menor de 5x5cm
- MiniSumo. Robots de menos de 500g de peso y con un área menor de 10x10cm
- Sumo. Robots de menos de 3kg de peso y con un área menor de 20x20cm
- Humanoides. Robots bípedos normalmente construidos con kits comerciales

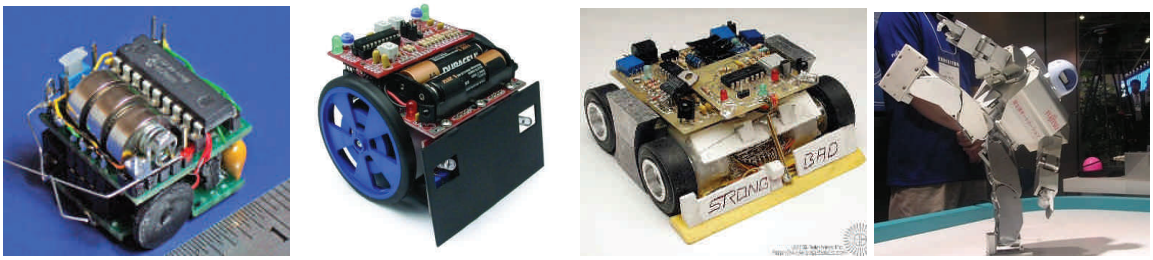


Figura 2.1: Microsumo, Minisumo, sumo y humanoides

Atendiendo al modo de funcionamiento:

- Autónomos. El robot funciona sin recibir ninguna orden del exterior
- Radiocontrol. El robot puede recibir instrucciones de su creador.

Capítulo 3

Historia del Sumo Robótico

La modalidad de Sumo Robótico fue inventada en Japón a finales de los años ochenta por Hiroshi Nozawa, presidente de Fuji Software. Su objetivo era que los estudiantes se interesasen en la robótica.

El primer torneo de exhibición se realizó en 1989 y participaron 33 robots. El primer campeonato oficial fue en 1990 y lucharon 147 robots en él.

Desde entonces el crecimiento del sumo robótico en Japón ha sido imparable. Más de 4000 robots participaron en la Liga de 2001.

A comienzos de los años noventa el sumo robótico fue introducido en Estados Unidos por Mato Hattori. Hattori grabó una cinta con los mejores momentos de la tercera Liga de sumo. Esta cinta llegó a manos de Bill Harrison que se involucró para difundirlo en su país. Harrison posteriormente fue el que inventó la clase de Minisumo. Esta clase es la que se ha hecho más popular ya que existen al menos el doble de robots de minisumo que de sumo estándar ya que su reducido tamaño facilita la construcción y ahorra costes.

Figura 3.1: Bill Harrison, inventor de la clase Minisumo

Desde principios de los años noventa los torneos de sumo se han expandido por todo el mundo y el número de robots ha crecido exponencialmente.

Capítulo 4

Reglamento

En este capítulo se resumen las reglas más importantes del reglamento de Sumo y MiniSumo.

4.1. Dohyo

El tamaño del dohyo es distinto para sumo y minisumo. No obstante comparten que el borde debe ser de 5cm de blanco brillante y el interior debe ser de negro mate. Debe estar situado a una altura de 5cm sobre el suelo.

Categoría	Diámetro del dohyo (cm)
Sumo	154
MiniSumo	77

Durante el combate no puede haber nada a menos de un metro de distancia del dohyo. Esto se hace para evitar que los robots detecten objetos fuera del ring y se confundan.

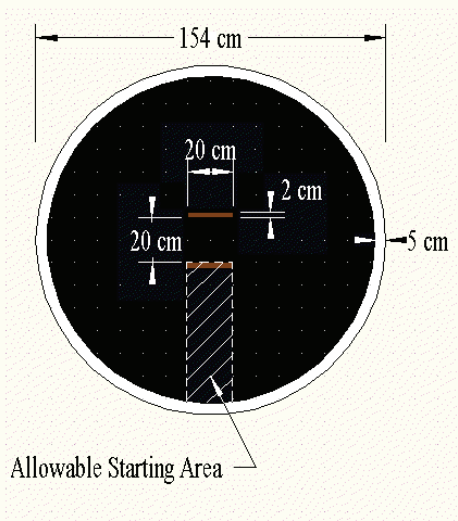


Figura 4.1: Dohyo de sumo

4.2. Peso y Tamaño

El peso máximo permitido depende de la categoría.

Categoría	Peso máximo (g)	Área máxima (cm)
Sumo	3000	20x20
MiniSumo	500	10x10

Es importante señalar que no hay restricciones en la altura del robot.

4.3. Restricciones al diseño

- El robot no tendrá partes que puedan dañar el dohyo
- El robot no puede disparar proyectiles.
- El robot no puede tener un dispositivo que lo fije al dohyo y le impida moverse.
- No se puede diseñar el robot para que se divida en diferentes piezas al empezar el combate.
- Se pueden desplegar estructuras siempre que el robot pueda recogerlas autónomamente de modo que este dentro del área máxima una vez finalizado el combate.

4.4. Combate

- El combate consiste en 3 asaltos de una duración máxima de 3 minutos.
- El asalto comienza 5 segundos después de que se de la señal. El robot que se mueva antes de 5 segundos pierde el combate.
- Gana el combate el robot que consiga echar a su adversario del dohyo sin caerse él.
- El asalto se repetirá si los robots quedan trabados indefinidamente o si los dos robots tocan el suelo a la vez
- Un robot pierde el combate si se le cae alguna pieza

4.5. Elección: Minisumo

Hemos elegido realizar un robot de minisumo porque debido a sus dimensiones más reducidas supone un mayor reto para el diseño y resulta más barata su construcción.

Capítulo 5

Principales Competiciones de Sumo Robótico

La lista de competiciones de sumo crece cada año. En este capítulos vamos a tratar de nombrar a las más importantes tanto del panorama nacional como del internacional.

5.1. Competiciones en España

5.1.1. Cosmobot

Cosmobot[1] se realiza en CosmoCaixa en Alcobendas (Madrid). Se lleva realizando desde 2008 y en 2011 tuvo lugar el fin de semana del 19-20 de Marzo. La competición es de sumo estándar y dice ser la mejor de España. Los premios son muy buenos: 1500 euros para el ganador y 300 para el octavo.

5.1.2. Deustobot

En Deustobot[2] tienen la categoría de Sumo y MiniSumo. Se realizó el 31 de Marzo de 2011 en el Auditorio de la Universidad de Deusto (Bilbao). Los premios son 250 euros para el primero y 150 para el segundo.

5.1.3. Alcabot

Alcabot[3] tiene solo categoría de Sumo. Se realizó del 4-8 de Abril de 2011 en la Universidad de Alcalá (Madrid). Los premios fueron 400 euros para el primero y 100 para el segundo.

5.1.4. Robolid

Robolid[4] 2012 ya está anunciado y tendrá lugar 26-27 de Abril. Tiene categorías de sumo y minisumo y se realiza en Valladolid. Los premios son 300 euros para el primero y 100 para el tercero.

[1] http://www.roboticspot.com/cosmobot/index.php
[2] http://www.roboticspot.com/cosmobot/index.php
[3] http://asimov.depeca.uah.es/robotica/course/view.php?id=18
[4] http://www.robolid.net/

5.1.5. Malakabot

Malakabot[5] se celebró en Málaga el 5 de Mayo de 2011. Parece que no hay premios o al menos no aparecen en la página web.

5.1.6. Robotus

Robotus se celebró en Sevilla el 15-16 de Mayo de 2011. La página web ya no está operativa pero parece que solo hay competición de sumo.

5.1.7. Cruo

Cruo[6] se realiza en la Universidad de Oviedo. En 2011 tuvo lugar el 29-30 de Julio. Tienen solo categoría de sumo y los premios son 400 euros para el primero y 100 para el cuarto.

5.1.8. CrJet

CrJet[7] se realizó en la Universidad Politécnica de Cataluña (Barcelona) el 8 de Octubre de 2011. Tiene categorías de sumo y minisumo y los premios son de 500 euros para el primer clasificado y 100 para el cuarto.

5.1.9. Robolot

Robolot[8] se realizó el 15-16 de Octubre en Cataluña. Hay categoría de Minisumo y el premio es de 300 euros. La página web tiene material didáctico.

5.1.10. AESSBot

AESSBot[9] tuvo lugar el 18-19 de Noviembre de 2011 en el Museo Marítimo de Barcelona. Tiene categorías de sumo y minisumo. Los premios 600 euros para el primero y 150 para el tercer clasificado.

5.1.11. Cuadro Resumen

Competición	Fechas	Lugar	Sumo	MiniSumo
CosmoBot	19-20 Marzo	Madrid	1500	-
DeustoBot	30-31 Marzo	Bilbao	250	250
AlcaBot	4-8 Abril	Madrid	400	-
Robolid	26-27 Abril	Valladolid	300	300
Malakabot	5 Mayo	Málaga	?	?
Robotus	15-16 Mayo	Sevilla	?	-
Cruo	29-30 Julio	Oviedo	400	-
CrJet	8-9 Octubre	Barcelona	500	500
Robolot	15-16 Octubre	Cataluña	-	300
AESSBot	18-19 Noviembre	Barcelona	800	600
			4150	1950

Vemos que hay más competiciones de Sumo y que los premios son mayores. Esto está en consonancia con el hecho de que construir el robot de sumo es mas costoso que el de minisumo.

[5] http://www.malakabot.com
[6] http://cruo.es/
[7] http://www.concursroboticajet.etseiat.upc.edu/
[8] www.robolot.org/
[9] http://aess.upc.es/aessbot/

5.2. Competiciones Internacionales

5.2.1. RoboGames

Son las Olimpiadas de la robótica, la máxima competición internacional. Tiene un montón de competiciones distintas como Kung-fu, Fútbol, Guerra...

5.2.2. All Japan Robot Sumo Tournament

Probablemente la competición de sumo robótico de más nivel del mundo.

Capítulo 6

Análisis de Vídeos de Campeonatos de Sumo

En este capítulo vamos a analizar los vídeos de campeonatos de minisumo que podemos encontrar en Internet. El objetivo es fijarse en los los mejores robots y en los más originales.

6.1. Robolid 2010[1]

Hay dos robots que destacan claramente sobre el resto: Mini-JMV y Sansón, ambos construidos por el equipo D.P.E.Bots. Vamos a analizarlos a continuación.

Mini-JMV.

Este es probablemente uno de los mejores robots de minisumo como lo acreditan los campeonatos que ha ganado.

Tiene dos sensores de distancia en la parte frontal, uno en cada lateral y no parece que tenga ninguno en la parte trasera. Es un robot que encuentra al adversario rápido y se mueve hacia él más rápidamente aún. Sus ruedas son pequeñas y están recubiertas por una especie de goma marrón. Aparentemente solo tiene dos ruedas. Su diseño es muy compacto y bajo. La carcasa cubre únicamente el techo y la parte trasera y frontal, dejando los laterales al aire. En la parte delantera la carcasa tiene forma de rampa con una curvatura que cambia cerca del suelo, y además tiene dos aberturas para que los sensores de distancia puedan escanear lo que tiene delante. En las fotos se puede ver la versión final (arriba) y una versión más preliminar (abajo). Vemos que ha evolucionado la carcasa y los recubrimientos de las ruedas. La carcasa es de plástico y la rampa es de metal.

[1] http://www.youtube.com/watch?v=hlax5Qm_OMI

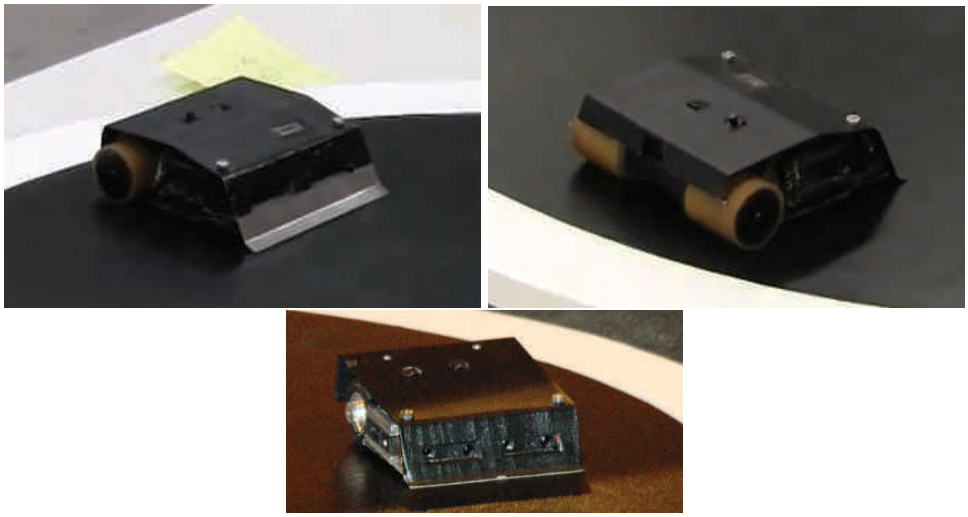


Figura 6.1: Distintas vistas del Mini-JMV

Sansón

Este robot quedó segundo por detrás de Mini-JMV. Su diseño es completamente distinto pero alcanza también un rendimiento muy bueno. Esto nos indica que no hay un solo camino para diseñar un robot campeón.

Este robot tiene dos grandes ruedas que también están recubiertas por la misma goma o resina que tenía el Mini-JMV. Es más lento que el anterior. Repite la rampa frontal agujereada para los sensores de distancia pero parece que no tiene ninguno en los laterales ni en la parte de atrás. Parece que en la parte delantera tiene unos bumpers para detectar el contacto con otro robot. Es un diseño mucho más alto que el robot anterior.

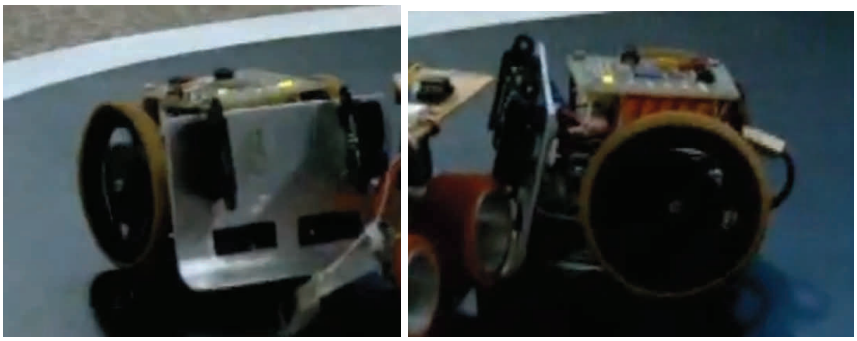


Figura 6.2: Imágenes de Sansón

6.2. DeustoBot VIII 2009[2]

En este concurso casi todos los robots eran idénticos. En comparación con otros campeonatos los robots eran bastante más lentos. Estaban realizados con el kit SumoBot de Parallax[3]. El ganador del concurso lo fue simplemente porque era capaz de empujar con un poco más de fuerza que los demás. Sin embargo este concurso nos deja un interesante diseño que vamos a comentar a continuación.

Chuck Norris

Las reglas de minisumo nos dicen que el robot debe ocupar un área máxima de 10x10cm. Sin embargo una vez comenzado el combate el robot puede desplegarse y ocupar más área siempre que sea capaz de volver a la situación inicial una vez acabado el combate. No obstante en ninguno de los vídeos online hemos podido observar como los robots se repliegan. Lo más común es que tengan una plancha en la parte delantera que se deja caer como una especie de puente levadizo y que sirve para que el robot adversario se suba encima y pueda ser expulsado más fácilmente del ring.

Pero el robot que presentamos ahora se expandía al comenzar el combate. Fue eliminado en la primera ronda pero eso no resta nada a su originalidad. En las fotos se puede ver como se transforma.

Figura 6.3: Transformación del robot Chuck Norris

6.3. DeustoBot IX 2010[4]

En este campeonato vemos de nuevo en acción a los robots de D.P.E.Bots que ya hemos tratado anteriormente: Mini-JMV y Sansón. De nuevo quedan 1º y 2º. De los otros robots no hay nada que reseñar.

[2] http://www.youtube.com/watch?v=deuFnTpAc_c
[3] http://www.parallax.com/Store/Robots/AllRobots/tabid/128/CategoryID/3/List/0/SortField/0/Level/a/ProductID/316/Default.aspx
[4] http://www.youtube.com/watch?v=TFhFqj2Y51c

6.4. Canadian National Robot Games 2010[5]

Este campeonato nos muestra unos cuantos robots de diseños muy curiosos.

Robot Torero

Este robot utiliza la regla de poder expandirse de una forma distinta. Despliega en un lateral una pequeña plancha que hace las veces de la capa de toreo. El robot adversario detectará la capa y pensará que es el robot. Esto puede darnos alguna ventaja en el combate.

Robot Piramidal

Este robot era muy compacto y tenía la forma de una pirámide. Le quedaba aún tiempo de desarrollo ya que fallaba el contacto de las ruedas con el suelo y se movía muy lento.

Figura 6.4: Fotografías de los robots del Canadian National Robot Games

Robot Placa

Este robot utilizaba la placa donde iban montados los sensores y controladores como chasis. Tenía cuatro ruedas y era muy rápido.

Robot de las Grandes Ruedas

Este robot tenía unas ruedas enormes que ocupaban toda su longitud.

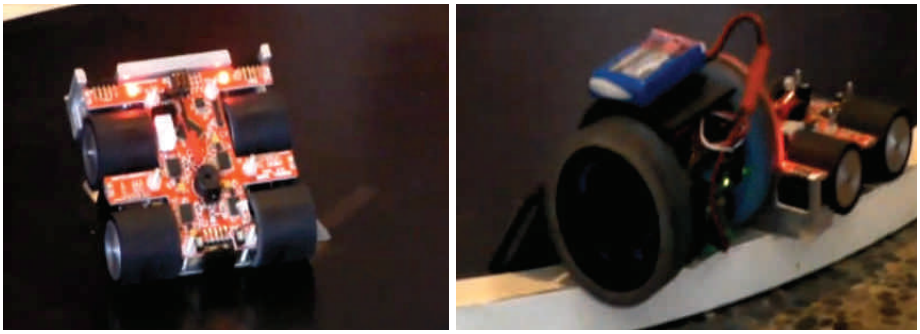


Figura 6.5: Fotografías de los robots del Canadian National Robot Games

[5] http://www.youtube.com/watch?v=TyHmevF1GBI

6.5. Conclusiones

- La primera conclusión que se puede extraer es que hoy por hoy no hay un diseño que sea superior al resto y que se haya impuesto.
- Podemos ver que la mayoría usa dos ruedas de gran diámetro, sin embargo el mejor robot que hemos visto utiliza ruedas pequeñas.
- Algo que parece claro es que con dos ruedas es suficiente y que muy pocos utilizan cuatro ruedas. No parece que se deriven ventajas importantes por utilizar cuatro ruedas.
- La velocidad se ha revelado como un factor muy importante. Cuanto más rápido sea el robot mayores serán sus opciones de ganar el combate siempre que este bien controlado y no se salga del tablero.
- La fuerza que puede ejercer hacia delante también es un factor muy importante. Cuando dos robots chocan frontalmente si uno es capaz de ejercer más fuerza que el otro ganará el combate sin duda. Los principales factores que determinan la fuerza ejercida son: la potencia de los motores, la adherencia de las ruedas con el suelo y la geometría (No es lo mismo empujar una pared vertical que empujar una pared inclinada. Al empujar la pared inclinada parte de la fuerza se pierde en la componente vertical)
- No se ha visto ningún robot ganador que tenga una plancha delantera desplegable. Parece que son más efectivos los diseños compactos en los que la rampa está integrada. Además el hecho de tener que utilizar un mecanismo para recogerla complica el diseño. En todo caso se utilizará alguna estructura similar a la del robot "torero".
- Finalmente concluimos que no va a ser nada fácil construir un robot ganador porque vamos a partir completamente de cero y existen estructuras como el D.P.E.Bots que son realmente muy potentes. Sin embargo vamos a intentar disputarles la supremacía.

Capítulo 7

Aspectos clave del robot de sumo

7.1. Rampa

La rampa es el punto de contacto con el adversario. El robot que consigue deslizar su rampa bajo el contrario consigue reducir la fuerza de empuje del contrario. La mayoría de las veces el que consiga deslizarse por debajo ganará el combate.

7.2. Fuerza

Si ningún robot consigue deslizarse bajo el otro el robot que tenga más fuerza podrá expulsar al otro y ganará el combate.

7.3. Detección

Ser capaz de detectar al contrario es indispensable para echarle del tatami. De la misma forma es vital poder detectar los bordes del tatami para no salirse de él. Los sensores tienen una importancia muy grande en un robot de sumo.

7.4. Velocidad

Tanto la velocidad de movimiento como la de detectar al contrario son parámetros que pueden dar una ventaja competitiva suficiente para ganar el combate.

7.5. Adaptable

Si un robot de sumo tiene distintos modos de comportamiento podrá adaptarse a su enemigo. Esto es una gran ventaja.

	Rampa	Fuerza	Detección	Velocidad	Adaptable
Chasis	x				
Motores		x		x	
Sensores			x		
Microcontrolador			x	x	x

Parte II

Elección de Componentes

Capítulo 8

Motores

8.1. Tipos de motores

Vamos a centrarnos únicamente en los motores eléctricos.

8.1.1. Servo modificado

Los servomotores son muy utilizados en robótica. Tienen un control electrónico que les permite mantener una posición y ejercer una fuerza considerable. Es necesario desmontarlos y modificarlos para que puedan rotar continuamente. Son utilizados en algunos robots de sumo pero tienen la desventaja de que no son muy rápidos, por lo que es necesario utilizar ruedas grandes que elevan el centro de masas del robot.

8.1.2. Motores de corriente continua

Los motores de corriente continua proporcionan una velocidad de giro proporcional al voltaje aplicado. Tienen velocidades de giro muy altas por lo que es necesario utilizar cajas reductoras para adaptar la velocidad a un rango utilizable.

8.1.3. Motores paso a paso

Los motores paso a paso tienen una gran precisión y repetibilidad en cuanto al posicionamiento. Por ello son más caros que los motores de corriente continua.

8.2. Funcionamiento de un motor de corriente continua

8.2.1. Parámetros del motor

Los parámetros más importantes de un motor de corriente continua son:

- Torque: es la fuerza que ejerce el motor
- Velocidad: la rapidez con la que gira el motor
- Corriente: la intensidad que necesita el motor para funcionar

En la siguiente gráfica se puede ver como se relacionan los tres parámetros:

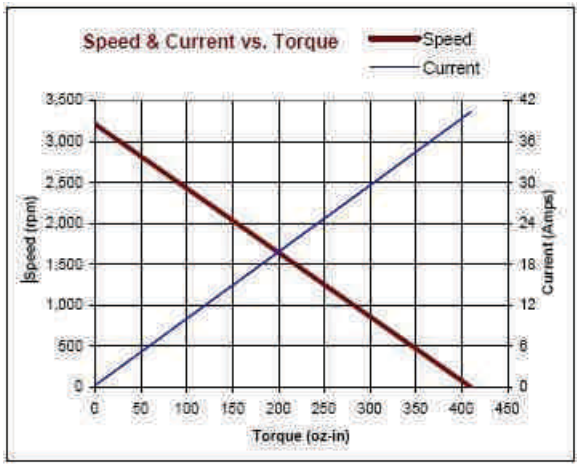


Figura 8.1: Velocidad y corriente frente a Torque

Aunque realmente no es así, se considera que el motor tiene un funcionamiento lineal y esto se aproxima bastante a la realidad.

Como se ve en la gráfica la velocidad de giro del motor depende linealmente del torque. Si el motor no tiene que ejercer fuerza alguna y gira libremente su velocidad será máxima. Conforme vayamos pidiéndole que ejerza una fuerza mayor la velocidad irá bajando hasta que deje de girar. Ese punto en el que el motor no gira se le llama de stall.

Cuando el motor gira libremente el consumo de corriente es mínimo, pero no nulo como parece en la gráfica. Conforme requerimos más fuerza del motor este requerirá a su vez más corriente hasta alcanzar el máximo en el punto de stall.

Queda un cuarto parámetro del motor: el voltaje. Se suele hacer la aproximación de que todas las características anteriores son proporcionales al voltaje. Si usamos el doble de voltaje el motor girará a doble velocidad, necesitará el doble de corriente y podrá ejercer el doble de fuerza. Naturalmente el voltaje que se suministra al motor debe estar dentro de unos límites para evitar que resulte dañado.

8.2.2. Especificaciones del motor

Las características que suelen suministrar los fabricantes de motores son las siguientes:

- Voltaje al que se suministran las especificaciones
- Velocidad de giro libre
- Corriente de giro libre
- Torque de stall
- Corriente de stall

Con esos cinco datos podemos construir la gráfica anterior y determinar completamente el funcionamiento del motor.

8.2.3. Punto de funcionamiento del motor

Las fórmulas que relacionan el torque con la velocidad de giro y la corriente son realmente sencillas.

$$N(T) = N_l \cdot \frac{T_s - T}{T_s} \tag{8.1}$$

Siendo N la velocidad de giro, N_l la velocidad de giro libre, T el torque aplicado y T_s el torque de stall.

$$I(T) = I_l + (I_s - I_l) \cdot \frac{T}{T_s} \tag{8.2}$$

Siendo I la corriente, I_l la corriente de giro libre e I_s la corriente de stall.

Por último recordar que todos los parámetros son proporcionales al voltaje:

$$T(V) = T_0 \cdot \frac{V}{V_0} \quad | \quad I(V) = I_0 \cdot \frac{V}{V_0} \quad | \quad N(V) = N_0 \cdot \frac{V}{V_0} \tag{8.3}$$

8.3. Criterios para la elección

La elección del conjunto motor-reductor-rueda determina completamente la velocidad a la que se desplazará el robot y la fuerza que podrá efectuar. Es por ello que es una decisión crítica a la hora de diseñar un robot competitivo.

Las especificaciones que buscamos que cumpla nuestro robot es que se mueva a una velocidad determinada y que sea capaz de ejercer una fuerza de empuje determinada.

8.3.1. Velocidad

Para determinar la velocidad a la que se desplazará el robot es necesario saber la velocidad de giro del motor y el radio de la rueda.

$$V = 2\pi RN \tag{8.4}$$

Siendo V la velocidad del robot, R el radio de la rueda y N la velocidad de giro de la rueda.

8.3.2. Fuerza

La fuerza que puede ejercer nuestro robot depende del torque del motor y de la tracción de las ruedas. La tracción determina la fuerza máxima que puede ejercer el robot. Si el robot intenta ejercer una fuerza mayor que la tracción las ruedas patinarán. La fuerza de torque debe poder ser mayor que la de tracción o el motor quedará trabado y muy posiblemente se dañará.

En resumen: la tracción determina la fuerza máxima que podemos ejercer y el torque determina si la ejercemos o no.

8.3.2.1. Tracción

La tracción es equivalente a la fuerza de rozamiento del robot con el suelo. La fuerza de rozamiento se modeliza así:

$$F_R = \mu \cdot P \tag{8.5}$$

Siendo F_R la fuerza de rozamiento, μ el coeficiente de rozamiento y P el peso.

Como vemos cuanto mayor sea el peso mayor será la fuerza que podamos ejercer. Por ello el robot deberá estar en el límite de peso permitido. Por otra parte se puede ver lo beneficioso que es conseguir que nuestro robot se cuele por debajo del contrario absorbiendo parte de su peso. De esa forma conseguimos aumentar nuestra tracción y disminuir la suya.

El coeficiente de rozamiento en la física clásica siempre está entre cero y uno y no depende del área de contacto. Esto es cierto cuando se consideran materiales inelásticos en superficies inelásticas como acero en acero.

Pero deja de ser cierto con materiales elásticos como la goma, donde se pueden alcanzar coeficientes de rozamiento de hasta 3.

Materiales	μ_e	μ_d
Acero // Hielo	0.028	0.09
Acero // Acero	0.15	0.09
Caucho // Cemento seco	1	0.8
Caucho // Cemento húmedo	0.3	0.25
Caucho // Madera	0.7	0.6

Cuadro 8.1: Algunos valores del coeficiente de rozamiento estático y dinámico

Estos valores son más bien teóricos ya que el fabricante no da el coeficiente de rozamiento de su rueda. Para los cálculos consideraremos que el coeficiente entre nuestras ruedas y el dohyo es de 1. Aunque probablemente sea menor como muestra la tabla no hay ningún problema si nos pasamos, lo habría si nos quedásemos cortos.

8.3.2.2. Torque

La fuerza de torque que puede ejercer cada motor se calcula utilizando el momento o torque del motor y el radio de la rueda.

$$F = \frac{T}{R} \tag{8.6}$$

Siendo F la fuerza ejercida, T el torque del motor y R el radio de la rueda.

8.3.3. Elección

Como hemos visto el radio de la rueda y las características del motor se interrelacionan a la hora de calcular las prestaciones del conjunto. Vamos a ver cual es el punto de funcionamiento del conjunto motor-rueda suponiendo que sabemos las características del motor y la fuerza que va a tener que ejercer.

Conocidos: motor N_l y T_s, fuerza de trabajo F. Desconocidos: radio de la rueda R .

$$T = F \cdot R \tag{8.7}$$

$$V = 2\pi \cdot R \cdot N \tag{8.8}$$

$$N = N_l \cdot (1 - \frac{T}{T_s}) \tag{8.9}$$

Las tres ecuaciones anteriores ya las conocíamos, las vamos a combinar para hallar la velocidad en función del radio de la rueda. Luego lo derivaremos para ver cual es el radio apropiado para trabajar a velocidad máxima dada una fuerza de trabajo determinada

$$V = 2\pi \cdot R \cdot N_l \cdot (1 - \frac{F \cdot R}{T_s}) \tag{8.10}$$

$$\frac{dV}{dR} = 2\pi \cdot N_l \cdot (1 - \frac{F \cdot R}{T_s}) - 2\pi \cdot N_l \cdot \frac{F \cdot R}{T_s}) = 2\pi \cdot N_l \cdot (1 - \frac{2 \cdot F \cdot R}{T_s}) \tag{8.11}$$

$$\frac{dV}{dR} = 0 \Longleftrightarrow 2\pi \cdot N_l \cdot (1 - \frac{2 \cdot F \cdot R}{T_s}) = 0 \Longleftrightarrow F \cdot R = \frac{T_s}{2} \tag{8.12}$$

Como vemos si queremos la máxima velocidad de un motor dado una fuerza de trabajo y una rueda tendremos que hacerlo trabajar en un torque que sea la mitad del de stall.

Por lo tanto elegida la rueda, velocidad de trabajo y fuerza de trabajo deberemos buscar un motor que tenga:

- Torque de stall doble o mayor que el de trabajo
- Velocidad de giro doble o mayor que la de trabajo

Esto supone que la fuerza de trabajo debe estar perfectamente calculada ya que el motor no podrá hacer más del doble de esa fuerza. Si el motor va a trabajar en dos puntos de trabajo con fuerzas distintas habrá que optimizarlo para la fuerza mayor.

Así vemos que no podemos optimizar el funcionamiento del motor para dos puntos de trabajo muy diferentes. Para nuestro proyecto esto supone que no podemos optimizar nuestro robot para competir en sumo y minisumo. Aunque un minisumo puede vencer a un sumo como muestra el siguiente vídeo[1] deberemos centrarnos y optimizar nuestro robot para el combate de minisumo.

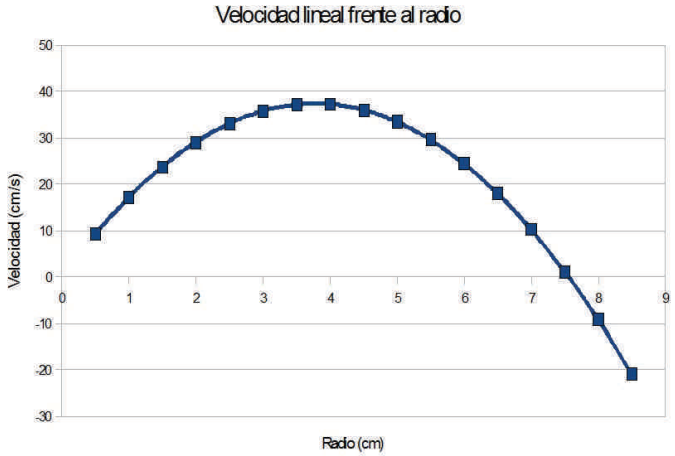


Figura 8.2: Simulación en Excel de la velocidad lineal frente al cambio de rueda para una fuerza de trabajo fijada en un motor 50:1 FingerTech “Gold Spark”

8.4. Especificaciones

Nuestro robot va a tener dos puntos de funcionamiento:

1. Cuando se mueva en solitario por el dohyo buscando a su presa. Tendrá que hacer una fuerza igual o menor a su peso: 500g
2. Cuando este empujando al otro robot. Tendrá que hacer una fuerza máxima igual al peso de los dos robots: 1kg

Tendremos que optimizar el funcionamiento para cuando este empujando al otro robot, para una fuerza de 1kg. Pero no nos interesa la velocidad a la que empuje sino la velocidad a la que se mueve solo. Como la velocidad del motor es linealmente dependiente de la fuerza:

$$\frac{V_{500}}{2-0{,}5} = \frac{V_{1000}}{2-1} \tag{8.13}$$

$$V_{1000} = \frac{V_{500}}{1{,}5} \tag{8.14}$$

Tenemos ya un método para calcular el motor necesario.

[1] http://www.youtube.com/watch?v=XvGka8oue_Y

Vamos a establecer las especificaciones que queremos que cumpla nuestro robot.

El tablero mide $75cm$ de diámetro. Vamos a buscar que pueda ir de un extremo a otro en un segundo lo que hace una velocidad de $75cm/s$. Esta será una de las ventajas de nuestro robot ya que los robots que utilizan servos como por ejemplo el kit Sumobot de Parallax se mueve mucho más lento, del orden de $15cm/s$. Si luego resulta que es demasiado rápido para ser controlado adecuadamente podremos hacer funcionar el motor con un voltaje menor.

$$N_{500} = \frac{V}{2\pi R} = \frac{75cm/s}{2\pi \cdot R}\frac{60s}{1min} = \frac{708rpm \cdot cm}{R} \tag{8.15}$$

$$N_{1000} = \frac{472rpm \cdot cm}{R} \tag{8.16}$$

$$N_l = \frac{944rpm \cdot cm}{R} \tag{8.17}$$

Por si acaso resultase una velocidad excesiva hemos podido calcular la velocidad de dos robots bastante rápidos.

- El primero es el Cobra[2] de Fingertech. Tiene motores Spark 50:1 funcionando a 11.1V. Esto nos da una velocidad de giro libre de 349rpm que será muy próxima a la real ya que el robot tiene cuatro motores. Sus ruedas tienen un diámetro de 2.86cm lo que hace que su velocidad sea de 52cm/s. No parece que sea tan rápido que no se pueda controlar, al contrario se ve muy estable[3].
- El segundo es Pyrostriker[4]. Utiliza los mismos motores pero con una reducción de 86:1 y el mismo voltaje lo que da una velocidad de giro libre de 203rpm. Como el diámetro de sus ruedas es de 3.8cm la velocidad final[5] es de 40cm/s.

Por lo tanto no parece que una velocidad de 75cm/s sea una locura.

La fuerza requerida será la suma de dos robots de minisumo que son $1kg$. Hay que tener en cuenta que al tener dos motores la fuerza se reparte por igual entre los motores.

$$T_{1000} = F \cdot R = 0{,}5kg \cdot R \tag{8.18}$$

$$T_s = 1kg \cdot R \tag{8.19}$$

Como vemos las características del motor dependen de la rueda utilizada. En la siguiente tabla se dan valores para distintas ruedas.

[2] http://www.fingertechrobotics.com/proddetail.php?prod=ft-kit-cobra-4wd-chassis
[3] http://www.youtube.com/watch?v=0sPd4nHu2wY
[4] http://letsmakerobots.com/node/26219
[5] http://www.youtube.com/watch?v=421NEFVahlA

Radio (cm)	$N_l\,(rpm)$	$T_s\,(kg \cdot cm)$
0,25	3776	0,25
0,5	1888	0,5
1	944	1
1,5	629	1,5
2	472	2
2,5	378	2,5
3	315	3
3,5	270	3,5
4	236	4
4,5	210	4,5
5	189	5

Cuadro 8.2: Torque y velocidad del motor con distintas ruedas

Como vemos tendremos dos criterios a la hora de elegir el motor. Si coinciden será perfecto, sino habrá que llegar a una solución de compromiso. No podemos perder fuerza de empuje así que habrá que sacrificar velocidad. Como curiosidad el radio es igual en número al torque (aunque no en unidades).

8.5. Motores disponibles

A la hora de elegir el motor para nuestro robot también tenemos que tener en cuenta el tamaño y el peso. Cuanto menores sean más fácilmente podremos integrarlos con los otros componentes.

A continuación se muestra una tabla en la que se pueden ver todos los motores encontrados que tienen una longitud menor de 5cm. Si fuesen más largos no podríamos colocarlos en paralelo. A la derecha se puede ver el radio que tendría que tener la rueda en función de la velocidad de giro del motor y en función del torque para cumplir las especificaciones.

Si el radio de la velocidad de giro es mayor que el del torque significa que no se pueden cumplir ambos requisitos a la vez. Si son iguales significa que se cumplen los dos requisitos a la vez. Si el radio del torque es mayor significa que podríamos poner unas especificaciones más exigentes.

Motor	Ratio	Peso	Largo	Ancho	N_l 7,4V	T_s 7,4V	R RPM	R Torque
		(g)	(mm)	(mm)	(rpm)	(kg-cm)	cm	cm
Micro Metal Gearmotor HP	5	10	24	10	7400	0,12	-	-
Micro Metal Gearmotor HP	10	10	24	10	3700	0,37	0,25	0,5
Micro Metal Gearmotor HP	30	10	24	10	1233	0,74	1	0,5
Micro Metal Gearmotor HP	50	10	24	10	771	1,73	1,5	1,5
Metal Gearmotor 15.5Dx30L	35	19	30	15	567	1,23	1,5	1
Metal Gearmotor 20Dx41L	29	41	41	20	543	2,22	2	2
Micro Metal Gearmotor HP	100	10	24	10	395	2,22	2,5	2
Micro Metal Gearmotor HP	150	10	24	10	247	3,95	4	4
Metal Gearmotor 20Dx42L	73	41	42	20	222	5,3	4,5	5
Micro Metal Gearmotor HP	210	10	24	10	173	4,44	5	4,5
Metal Gearmotor 15,5Dx30L	115	19	30	15	173	3,08	5	3
Micro Metal Gearmotor HP	250	10	24	10	148	5,3	6	5
Micro Metal Gearmotor HP	298	10	24	10	123	6,17	8	6
Metal Gearmotor 20Dx44L	154	41	44	20	111	10,61	9	10
Micro Metal Gearmotor HP	1000	10	24	10	39	11,1	-	11

Cuadro 8.3: Motores disponibles en Pololu

Motor	Ratio	Peso	Largo	Ancho	N_l 7,4V	T_s 7,4V	R RPM	R Torque
		(g)	(mm)	(mm)	(rpm)	(kg-cm)	cm	cm
FingerTech "Gold Spark"	20	28	40,3	15,5	582	0,93	2	1
FingerTech "Gold Spark"	35	28	40,3	15,5	333	1,63	3	1,5
FingerTech "Gold Spark"	50	28	40,3	15,5	233	2,33	4	2,5
FingerTech "Gold Spark"	63	28	40,3	15,5	185	2,94	5	3
FingerTech "Gold Spark"	86	28	40,3	15,5	136	4	7	4
FingerTech "Gold Spark"	115	28	40,3	15,5	101	5,35	9	5
FingerTech "Gold Spark"	150	28	40,3	15,5	78	6,98	12	7

Cuadro 8.4: Motores disponibles en Fingertech

Motor	Ratio	Peso	Largo	Ancho	N_l 7,4V	T_s 7,4V	R RPM	R Torque
		(g)	(mm)	(mm)	(rpm)	(kg-cm)	cm	cm
HP Mini Metal Gear Motor	35	15	35	15	871	0,83	1	1
Mini Metal Sealed Gear Motor	100	11	25	14	296	3,35	3	3
Mini Metal Sealed Gear Motor	298	11	25	14	97	4,24	-	4

Cuadro 8.5: Motores disponibles en Solarbotics

8.6. Motores que cumplen con las especificaciones

En este apartado vamos a enumerar los motores que cumplen con nuestros requisitos junto con la velocidad que tendrían moviéndose solos y la fuerza de empuje en el punto de máxima potencia. La fuerza de empuje será la del robot entero contando los dos motores.

Ninguno de los motores de FingerTech cumplían con nuestras exigentes especificaciones, si hubiésemos decidido usar cuatro motores hubieran sido validos ya que la fuerza se hubiera repartido más. También los podríamos haber empleados usando un voltaje superior a 7.4V.

Motor	Ratio	Largo	N_l 7,4V	T_s 7,4V	Radio	Fuerza robot	Velocidad
		(mm)	(rpm)	(kg-cm)	(cm)	(kg)	(cm/s)
HP Mini Metal Gear Motor	35	35	871	0,83	1	0,83	63,56
Mini Metal Sealed Gear Motor	100	25	296	3,35	3	1,12	72,12

Cuadro 8.6: Motores de Solarbotics que se acercan a las especificaciones

Motor	Ratio	Largo	N_l 7,4V	T_s 7,4V	Radio	Fuerza robot	Velocidad
		(mm)	(rpm)	(kg-cm)	(cm)	(kg)	(cm/s)
Micro Metal Gearmotor HP	10	24	3700	0,37	0,5	0,74	128,22
Micro Metal Gearmotor HP	30	24	1233	0,74	0,75	0,99	72,29
Micro Metal Gearmotor HP	50	24	771	1,73	1,5	1,15	94,74
Metal Gearmotor 15.5Dx30L	35	30	567	1,23	1,25	0,99	55,42
Metal Gearmotor 20Dx41L	29	41	543	2,22	2	1,11	88,01
Micro Metal Gearmotor HP	100	24	395	2,22	2	1,11	64,01
Micro Metal Gearmotor HP	150	24	247	3,95	4	0,99	77,1
Metal Gearmotor 20Dx42L	73	42	222	5,3	5	1,06	88,8
Micro Metal Gearmotor HP	210	24	173	4,44	4,5	0,99	60,72
Micro Metal Gearmotor HP	250	24	148	5,3	5	1,06	59,2

Cuadro 8.7: Motores de Pololu que se acercan a las especificaciones

Capítulo 9

Ruedas

9.1. Ruedas comerciales

A continuación se muestran las ruedas encontradas que pueden ser utilizadas en un robot de minisumo. El precio se refiere al par de ruedas.

Rueda	Peso	Eje	Diámetro	Radio	Ancho	Precio
	(g)	(mm)	(mm)	(mm)	(mm)	($)
GM10 Wheel	2	-	25	12,5	5,7	5
GM Series Plastic Wheels	-	7	69	34,5	7,62	8
1-1/4" Rubber Wheels - Internal Set Screw	12	3	31	15,5	13,2	8
2-5/8" Diameter Servo Wheel	-	-	69	34,5	7,62	8
Rubber Wheels	-	1,5	9	4,5	7	2

Cuadro 9.1: Ruedas de Solarbotics

Rueda	Peso	Eje	Diámetro	Radio	Ancho	Precio
	(g)	(mm)	(mm)	(mm)	(mm)	($)
FingerTech Cobra Wheels	12,6	3	28,6	14,3	21,6	18,65
Lite Flites	8,51	3,97	44,5	22,2	19,1	9
Lite Flites	10,49	3,97	50,8	25,4	19,1	9
Lite Flites	12,76	3,97	57,2	28,6	19,1	9
Lite Flites	15,31	3,97	63,5	31,8	19,1	9
Lite Flites	18,71	3,97	69,9	34,9	19,1	9
Lite Flites	22,11	3,97	76,2	38,1	19,1	9
Lectra Lites	4,25	3,18	38,1	19,1	12,7	9
Lectra Lites	6,24	3,18	44,5	22,2	12,7	9
Lectra Lites	7,37	3,18	50,8	25,4	12,7	9
Lectra Lites	9,07	3,18	57,2	28,6	12,7	9
Rubber Wheel	7	2,38	25,4	12,7	9,7	6
Rubber Wheel	13	2,38	31,8	15,9	10,9	6
Rubber Wheel	23	3,18	38,1	19,1	14,0	6

Figura 9.1: Ruedas de FingerTech

Rueda	Peso	Eje	Diámetro	Radio	Ancho	Precio
	(g)	(mm)	(mm)	(mm)	(mm)	($)
Pololu 42x19 Idler Wheel/ Sprocket	20	3,5	42	21	19	6,98
Pololu Wheel 90x10	24	3	90	45	10	9,95
Pololu Wheel 80x10	20	3	80	40	10	9,25
Pololu Wheel 70x10	13	3	70	35	10	8,49
Pololu Wheel 60x8	10	3	60	30	8	7,95
Pololu Wheel 32x7	3	3	32	16	7	6,98
Tamiya Slick Tire	-	3	31	15,5	10	2,3
Tamiya Spike Tire Set	-	3-4	65	32,5	26	10,5
Tamiya Narrow Tire Set	-	3	58	29	16	7,25
Tamiya Off-Road Tires	-	3	50	25	30	3,6
Tamiya Truck Tire Set	-	3	36	18	16	4,1
Tamiya Sports Tire Set	-	3-4	46	23	25	6,5

Cuadro 9.2: Ruedas de Pololu

9.2. Ruedas de alta tracción

De todas las ruedas anteriores solo hay unas de alta tracción y se puede ver fácilmente por su precio que es el doble: las Cobra Wheels de Fingertech.

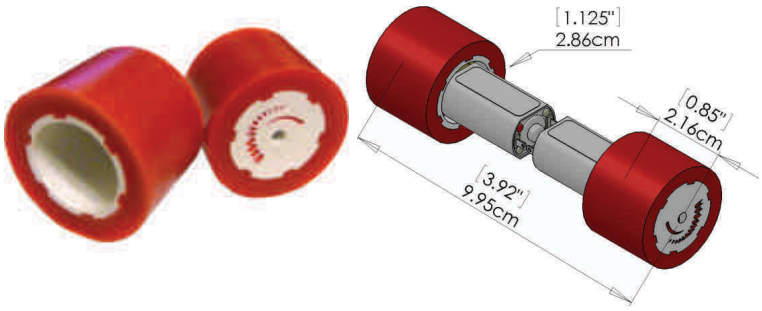


Figura 9.2: Cobra Wheels de Fingertech

Están realizadas con poliuretano de dureza Shore A20, lo que les confiere el mejor compromiso entre dureza y tracción. Un material con dureza Shore A5-10 tendrá más tracción pero será más blando y no resistirá el uso continuo. Por el contrario un material con dureza Shore A40-80 será más resistente pero proporcionará menos tracción.

Se recomienda limpiar las ruedas con alcohol antes de cada combate para tener la máxima tracción. Como son muy adherentes con gran facilidad cogen la suciedad del suelo y su rendimiento disminuye.

Hemos encontrado una única solución alternativa para conseguir ruedas de alta tracción. Se trata de las tiras de Pololu de alta tracción. Tienen un diámetro de 66m y una anchura de 7mm. Su precio es de 4$ la tira por lo que las dos ruedas nos saldrían por un precio parecido al de las Cobra Wheels.

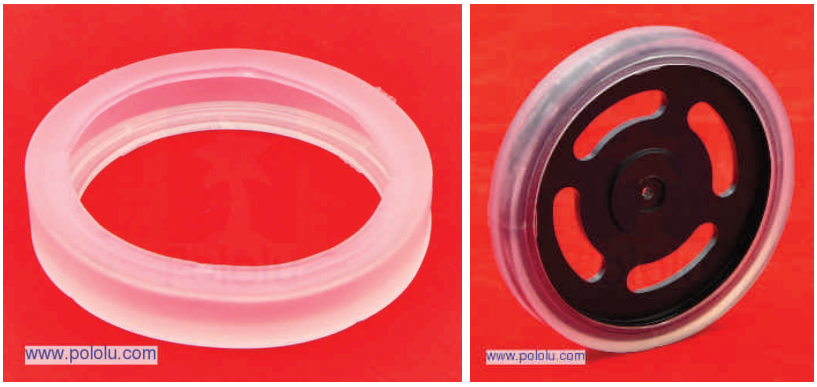


Figura 9.3: Pololu High-Traction Sticky Tire

Como vemos la oferta de ruedas de alta tracción es muy reducida. Por ello como las ruedas son una parte vital del robot la amplía mayoría de los robots avanzados customizan sus propias ruedas. A continuación vamos a ver como conseguir ruedas de gran tracción de cualquier tamaño.

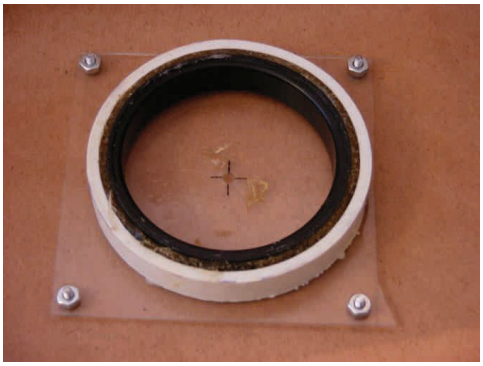
9.3. Ruedas de alta tracción customizadas

En el capítulo de recursos utilizados se pueden encontrar varios tutoriales[1] en los que se inspira esta sección.

- Material. Necesitamos Poliuretano para moldeo que normalmente se vende en dos botes A y B. Para utilizarlo es necesario mezclarlo a partes iguales. También se necesita un spray de cera antiadherente. Sirve para rociar el molde y evitar que el poliuretano se quede pegado.

- Realizar el molde. Se puede emplear cualquier material con tal de que no sea poroso para que no se produzcan burbujas en el poliuretano.

[1] http://www.davehylands.com/Robotics/Marauder/Making-Tires/index.html



- Rociar el molde con el spray antiadherente

- Moldear la pieza

Capítulo 10

Controlador del motor

10.1. Puente en H

Si queremos que el robot pueda moverse hacia adelante y hacia atrás debemos ser capaces de cambiar la polaridad del voltaje que entregamos al motor. Esto se consigue mediante un puente en H.

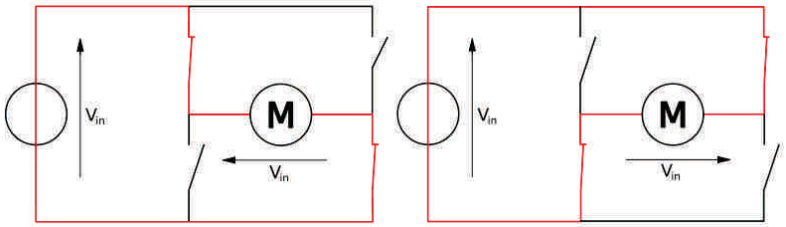


Figura 10.1: Esquema del puente en H

Como se puede ver en la figura el puente en H consiste básicamente de cuatro interruptores los cuales se abren o se cierran adecuadamente para entregar al motor un voltaje positivo o negativo.

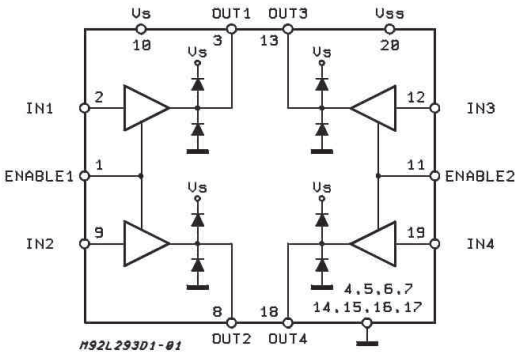
10.1.1. L293D

Existe un circuito integrado llamado L293D que tiene en su interior 2 puentes en H y que es perfecto para nuestro robot. Existen otros integrados con puentes de H pero es necesario montarlos en un circuito con diodos rectificadores, el L293D los lleva dentro. Puede proporcionar una corriente de pico de 1,2A por cada canal y 600mA continuamente. Si esto no es suficiente se pueden soldar dos en paralelo para duplicar las prestaciones.

El tiempo que tarda en abrir y cerrar la salida es menor de 2 microsegundos, lo que nos permite operar a frecuencias de hasta 500kHz. Para evitar problemas lo mejor seria utilizar frecuencias al menos un orden menor, 50kHz.

El L293D tiene un pin de entrada en el que tenemos que proporcionar el voltaje de salida.

Cada canal tiene dos pines de control Input, un pin de Enable y dos pines de salida. Así es como fácilmente controlamos dos motores.



TRUTH TABLE (one channel)

Input	Enable (*)	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance
(*) Relative to the considered channel

Figura 10.2: Diagrama y tabla de verdad del L293D

Cuando Enable está desactivado no se suministra ninguna tensión y en la salida tenemos una alta impedancia. Este modo no nos interesa en el combate porque no frena al motor sino que lo deja moverse libremente. Podría interesarnos cuando estemos en el menú de configuración del robot y queramos moverlo libremente con la mano.

Cuando el Enable está activado podemos elegir mediante los Input si en los pines de salida tenemos la máxima tensión o tierra. Si en uno tenemos tensión y en el otro tierra el motor girará. Si ponemos la misma tensión en ambos, ya sea tierra o la tensión de alimentación, el motor se frenará.

10.1.2. L293

El L293 es otra versión que no incorpora los diodos rectificadores pero que puede manejar una corriente mayor. Puede proporcionar una corriente de pico de $2A$ por cada canal y $1A$ continuamente.

Es importante señalar que tanto el L293 como el L293D tienen una cierta caída de voltaje debido a que su tecnología esta basada en transistores bipolares. Se pueden perder hasta 1.7V en los drivers basados en tecnología de transistores bipolares. Existen alternativas como los drivers basados en tecnologías FET que tienen perdidas de entre 0.2 y 0.4V pero son más caros.

10.1.3. MC33886VW

El MC33886VW es un circuito integrado que tiene un único puente en H. Puede suministrar corrientes de hasta 5A y está hecho con tecnología FET por lo que la caída de tensión en el driver es mucho menor. La resistencia aparente es de $120m\Omega$ y por eso retiene tan poco voltaje.

Puede controlarse mediante PWM de frecuencia de hasta 10kHz.

La desventaja es que su encapsulado tiene una distancia entre pines de 1.27mm en lugar de los 2.54mm estándar. Por lo tanto habría que buscar algún zócalo especial para poder ensamblarlo en la placa de prototipado.

10.1.4. Cuadro resumen

Modelo	Peak Current (A)	Output Current (mA)	Diodos rectificadores	Precio
L293NEE4	2	1000	No	2.83
L293DNE	1.2	600	Si	3.16
2xL293DNEE4	2.4	1200	Si	6.32
MC33886VW	8	5000	Si	13 (2)

Por lo tanto habrá que ver los requerimientos de corriente que tenga el motor y la rueda escogida. Si podemos usaremos preferentemente el L293D ya que evita tener que montar los diodos rectificadores.

A la hora de comprarlos encontramos que ambos están descatalogados. Sin embargo se pueden encontrar modelos equivalentes más modernos cuyos nombres son los mismos pero se han añadido algunas letras más como L293DNEE4 o L293NE. Las especificaciones son prácticamente las mismas.

10.1.5. Elección: 2xL293DNEE4

Finalmente hemos elegido utilizar dos L293D por simplicidad. De esta forma no tenemos que utilizar diodos que compliquen el circuito y hagan necesaria una placa de prototipado mayor. El MC33886VW nos hubiera dado un mejor rendimiento pero a un costo mayor y obligándonos a utilizar un zócalo para adaptarlo a la placa.

10.2. PWM Modulación por Ancho de Pulso

Una vez que ya controlamos el sentido del giro podemos empezar a pensar como controlar la velocidad. Como ya se ha dicho anteriormente la velocidad de giro del motor depende del voltaje aplicado. A mayor voltaje mayor será la velocidad de giro.

Nosotros disponemos de una batería que nos da un voltaje constante. Ahora imaginemos que tenemos un interruptor que conecta la batería y el motor y que lo abrimos y cerramos a intervalos iguales. La tensión media que tendría el motor sería la mitad de la batería. Ahora imaginemos que los intervalos no tienen porque ser iguales y que abrimos y cerramos el interruptor tan rápidamente que no somos capaces de percibirlo. En eso consiste el Pulse Width Modulation (PWM).

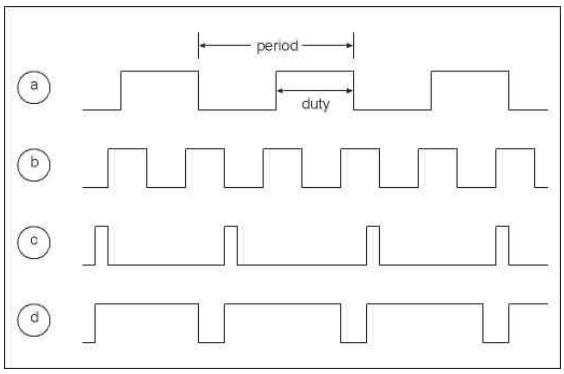


Figura 10.3: Gráfica explicativa del PWM

Cuanto mayor sea el ancho del pulso mayor será el voltaje medio.

$$V = \frac{\tau}{T} V_0 \tag{10.1}$$

Siendo τ el ancho de pulso, T el periodo y V_0 la tensión de la batería.

10.3. Utilización conjunta del PWM y del puente en H

Utilizando las dos técnicas conjuntas podremos alcanzar un grado de control de los motores muy altos. Podremos controlar a la vez la velocidad y el sentido de giro de los motores.

Capítulo 11

Microcontrolador

Como microcontrolador vamos a utilizar una placa de Arduino, concretamente una Duemilanove[1] ya que estaba disponible en el laboratorio.

Arduino es una plataforma de hardware libre, basado en un placa con un microcontrolador y un entorno de desarrollo. Las principales ventajas de Arduino son:

- Se programa en el lenguaje de programación Processing/Wiring que es muy similar a C.
- Es una plataforma que disfruta de una gran popularidad y que va en aumento. Esto hace que haya una gran cantidad de información disponible en internet.
- Es una plataforma realmente potente, se pueden hacer muchísimas cosas con Arduino.

11.1. Arduino Duemilanove

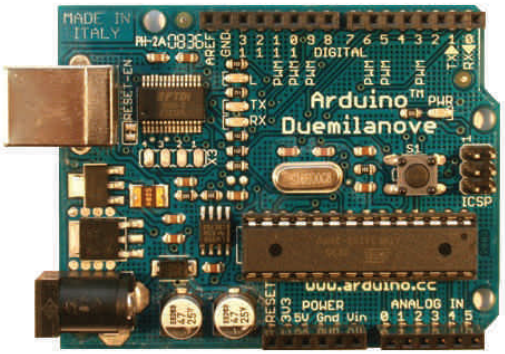


Figura 11.1: Arduino Duemilanove

- Arduino Duemilanove es una placa basada en el microcontrolador ATmega328. Este microcontrolador se hace funcionar a 16MHz por lo que puede llegar a ejecutar 16 millones de operaciones por segundo ya que la mayoría de sus instrucciones se realizan en un solo ciclo de reloj. Esto debería ser más que suficiente para poder controlar nuestro robot adecuadamente.

[1] http://arduino.cc/en/Main/arduinoBoardDuemilanove

- La placa tiene 14 pines digitales que pueden funcionar como entrada o salida. De ellos 6 pueden utilizarse como salidas PWM.
- La placa tiene también 6 entradas analógicas.
- Arduino Duemilanove opera a 5V pero el voltaje de entrada recomendado está entre 7-12V, pudiendo llegar a 20V. La Duemilanove usa el MC33269 como regulador de tensión para conseguir los 5V necesarios para su funcionamiento. Este regulador tiene un "DropOut" de 1V, esto es, que para conseguir los 5V necesita estar alimentado con 6V, como mínimo
- Los pines de 5V son capaces de suministrar y absorber una corriente de 40mA.
- La placa tiene un pin de 5V que podemos utilizar para alimentar los sensores. Este pin no tiene la restricción de los 40mA, está conectado al MC33269 y puede dar hasta 800mA. Esta es una de las grandes ventajas de Arduino, que nos proporciona una fuente de 5V muy potente con la que alimentar los sensores.

11.2. Otras placas Arduino

11.2.1. Arduino Nano

Arduino Nano[2] es una versión reducida de la Duemilanove pero con la desventaja de que no puede usar alimentación externa y debe ser alimentada con cable USB Mini-B. Sus dimensiones son 18,5x43,2mm.

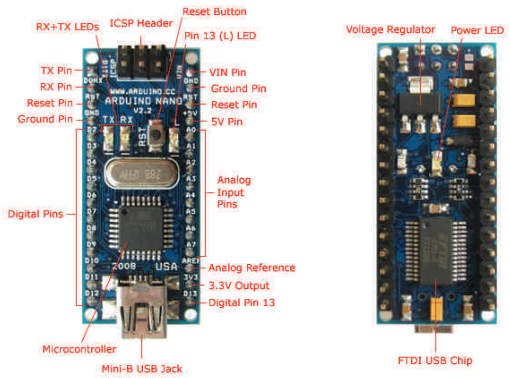


Figura 11.2: Arduino Nano

11.2.2. Arduino Mini

Arduino Mini[3] es otra versión reducida de la Duemilanove pero en este caso admite alimentación externa. Para programarla es necesario conectarle un adaptador de Mini USB. Esta podría ser una opción a utilizar si no disponemos de espacio en el robot para la Duemilanove

[2] http://www.arduino.cc/en/Main/ArduinoBoardNano
[3] http://www.arduino.cc/en/Main/ArduinoBoardMini

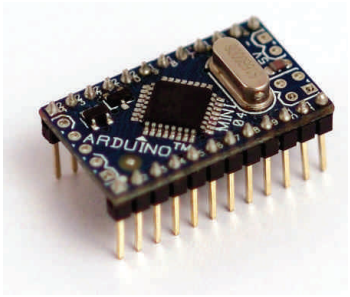


Figura 11.3: Arduino Mini

11.2.3. Arduino Pro

Arduino Pro[4] es simplemente una versión de bajo costo de Arduino que viene sin conector USB ni pines. No es una versión mejorada como parece indicar su nombre.

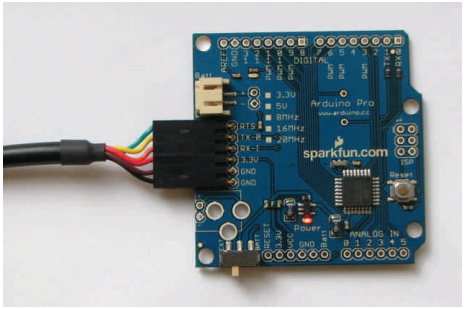


Figura 11.4: Arduino Pro

11.2.4. Arduino Mega

Arduino Mega[5] es una versión con muchas más entradas que la Duemilanove. Tiene 16 entradas analógicas y 54 digitales.

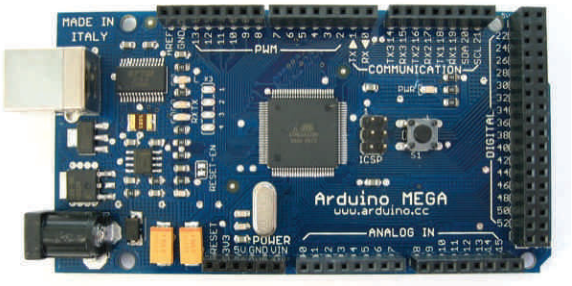


Figura 11.5: Arduino Mega

[4] http://arduino.cc/en/Guide/ArduinoPro
[5] http://arduino.cc/en/Main/ArduinoBoardMega

Capítulo 12

Pantalla LCD

Hemos decidido utilizar una pantalla LCD para poder interactuar con el microcontrolador. La idea es que mediante la pantalla y unos botones podamos escoger estrategias de combate, ver los valores de los sensores, emitir mensajes para ver lo que está haciendo el microcontrolador durante el combate y poder depurar el software...

Utilizar una pantalla LCD dotará a nuestro robot de una mayor versatilidad y potencia.

Los únicos requisitos que pedimos a la pantalla es que su tamaño no exceda el del reglamento de minisumo, es decir, que no sea más larga que 10cm. Por eso vamos a utilizar una pantalla LCD de 16 caracteres y 2 líneas que había en el laboratorio. Además tenemos un módulo SerLCD de SparkFun que facilita enormemente su utilización. Tan solo tenemos que conectar dos cables de alimentación y otro al pin 1 de la placa Arduino para enviar el texto.

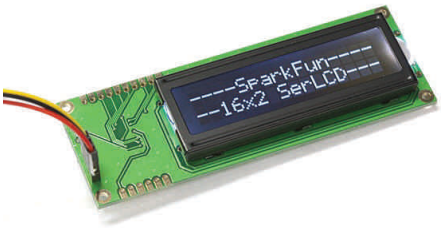


Figura 12.1: Serial Enabled 16x2 LCD - White on Black 5V

Capítulo 13

Sensores de Borde

Un robot de sumo que no pueda detectar el borde del tatami no tendrá ninguna posibilidad ante un robot competitivo. Conseguir que el robot se mueva por el tatami sin salirse sólo es lo mínimo que le podemos pedir al robot. Existen dos aproximaciones al problema de detectar el borde.

- Detectar el desnivel. El reglamento marca que el dohyo tiene que estar 5cm elevado sobre el suelo. Por lo tanto es posible utilizar sensores mecánicos que detecten este desnivel.
- Detectar el cambio de color. El reglamento marca que el dohyo tiene que ser negro salvo un anillo blanco de 5cm en el borde. Este cambio de color puede ser detectado y utilizado para evitar que el robot abandone el ring.

13.1. Sensores Mecánicos

No son comúnmente utilizados pero hay que hacer notar que existe esa posibilidad. Se trata de diseñar un actuador que perciba el desnivel en el borde y active un interruptor.

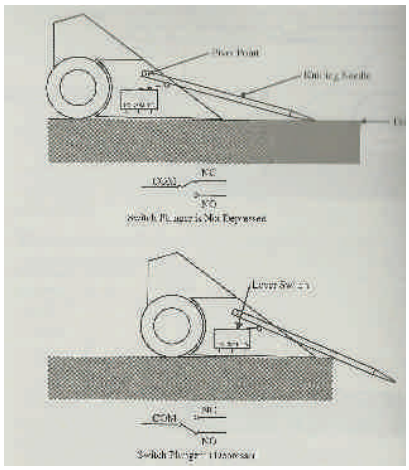


Figura 13.1: Posibles sensores mecánicos

13.2. Sensores Ópticos

Como sabemos el negro y el blanco reflejan la luz de forma muy distinta. Este es el hecho que usaremos para detectar el borde del dohyo. Tenemos tres posibles opciones.

1. Fotorresistencia. Existen resistencias que varían su valor en función de la luz recibida. Utilizaríamos un LED para iluminar el suelo y una resistencia para medir la luz reflejada. Este método tiene el inconveniente de que el cambio de valor de la resistencia no es rápido y un robot veloz no podrá detectar el borde a tiempo.

2. Fototransistor. En este caso la corriente que atraviesa el fototransistor depende de la luz recibida. La ventaja que tiene es que es mucho más rápido que la fotorresistencia.

3. Infrarrojos. Con las fotorresistencias y los fototransistores podemos usar luz visible e infrarroja. Esto tiene el inconveniente de que la medida del sensor puede verse alterada por la luz de la sala. Esto lo solucionan los sensores de infrarrojos ya que no son sensibles a la luz visible. La desventaja es que no podemos apreciar a simple vista si el sensor está funcionando en caso de problema. Sin embargo tenemos una solución muy a mano: las cámaras digitales son sensibles a la luz infrarroja.

13.3. Elección: Sensor Infrarrojo CNY70

Habiendo considerado las distintas posibilidades vemos que la mejor opción por efectividad y simplicidad son los sensores de infrarrojos. Existen múltiples opciones en el mercado y nosotros elegimos la que probablemente sea más usada en los minisumos por su simplicidad, robustez y bajo precio: el sensor CNY70.

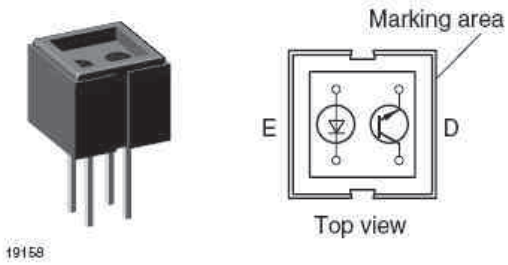


Figura 13.2: Sensor CNY70

El CNY70 tiene unas dimensiones realmente reducidas de $7x7x6mm$. Debe estar a una distancia del suelo inferior a 5mm para funcionar con normalidad, además tiene un filtro de luz visible.

Cuando el fototransistor no recibe la luz se encuentra en corte y no permite el paso de la corriente. Por el contrario cuando recibe la luz reflejada se satura y permite el paso de corriente. Podemos utilizar un circuito como el siguiente para obtener el estado del fototransistor.

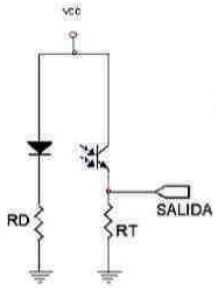


Figura 13.3: Circuito posible con el CNY70

Con este sencillo circuito si el sensor esta sobre la zona negra el fototransistor se encontrará en corte y la tensión en la salida será nula. Por el contrario al llegar al borde el color blanco reflejará la luz contra el transistor que comenzará a conducir y a tendremos una tensión alta en la salida. Podríamos haber hecho que respondiera al revés pero teniendo en cuenta que la mayor parte del tiempo los sensores estarán sobre el negro así ahorramos corriente.

Con esta configuración únicamente tendremos que llevar al sensor 3 cables ya que la alimentación será común a dos de sus cuatro patillas.

En la hoja de características del CNY70 vemos que la máxima corriente permitida por el LED es de 50mA y que la corriente que circule por el transistor depende de la que circule por el LED. Un valor adecuado de corriente sería 20mA, por eso se suele usar una resistencia de 220Ω con una fuente de 5V. La resistencia del transistor basta con que sea lo suficientemente elevada para que cuando entre en conducción absorba la mayoría de la tensión por lo que 10-47kΩ será suficiente.

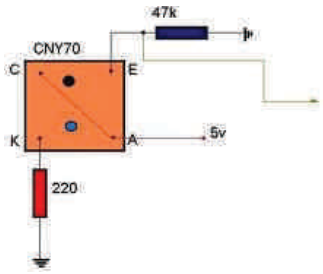


Figura 13.4: Valores de las resistencias

13.4. Sabotear los Sensores Infrarrojos de Suelo

Hay dos posibilidades principales para engañar a los sensores infrarrojos de suelo: utilizar un emisor de infrarrojos o simular el borde de la pista con el color blanco.

13.4.1. Simular el borde de la pista

Es posible colocar en el frontal de nuestro robot una plancha plana y delgada de color blanco pegada al suelo. De esta manera si logramos colarla debajo de nuestro adversario sus sensores detectarán que se encuentra en el borde y se irá hacia atrás sin siquiera tener que empujarle.

Esto puede evitarse haciendo una corrección en el software del robot. Por ejemplo haciendo que si los sensores de proximidad nos dicen que el adversario está delante nuestra y los de suelo nos dicen que estamos en el borde ignoremos los sensores de suelo. Esto puede ser un poco peligroso por eso

consideramos que la mejor opción será crear una estrategia que haga esta corrección y activarla solo si nos enfrentamos a un robot de estas características.

13.4.2. Emisor de Infrarrojos

En teoría sería posible utilizar un potente emisor de infrarrojos que iluminase los bajos del robot adversario para aumentar la luz que reciben sus sensores y hacerle creer que está en el borde. Si el presupuesto lo permite sería buena idea probarlo ya que si funciona es sencillísimo de implementar y se convierte en un arma muy potente. Habría que tener en cuenta la longitud de onda de los sensores del adversario, típicamente suele ser $880nm$ o $950nm$.

Capítulo 14

Sensores de Proximidad

Los sensores de proximidad son utilizados para localizar a nuestro adversario en el ring. Hay dos tipos de sensores de proximidad: los sensores de infrarrojos y los de ultrasonidos. Los dos basan su funcionamiento en el mismo método: un emisor de ondas y un receptor.

14.1. Sensores Infrarrojos

Dentro de esta familia de sensores destacan los Sharp GP2. Son sin duda los más utilizados en minisumo. Son compactos y en su interior incluyen ya el emisor y el receptor de infrarrojos.

Su funcionamiento se basa en que si hay un obstáculo cercano el ángulo de rebote será más grande que si el obstáculo está más lejos. Esto lo podemos ver claramente en la siguiente figura.

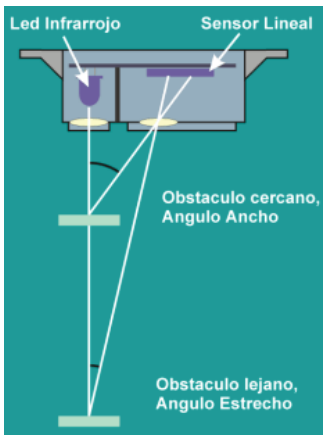


Figura 14.1: Funcionamiento de los sensores Sharp GP2

Los hay de dos tipos: binarios y analógicos.

14.1.1. Sensores binarios

Estos sensores dan una salida positiva si detectan algo a una distancia menor de la preestablecida. Si esta más lejos la salida es nula. Como vemos no son útiles para estimar la distancia, sirven para saber si hay algo delante o no. Por eso no es recomendable colocarlos en la parte frontal del robot ya que no podremos apuntar bien hacia el adversario. Pero en cambio si que pueden ser útiles encarados

hacia la parte trasera o los laterales para que sepamos hacia donde tiene que girar nuestro robot. A continuación en una tabla se muestran los dos principales sensores binarios.

Sensor	Rango (cm)	Preset (cm)	Precio (€)
GP2Y0D805Z0F	4-6	5	6.73
GP2Y0D340K	10-60	40	8.46

En nuestro caso nos interesa más el GP2Y0D340K dado que el diámetro del dohyo es de 80cm.

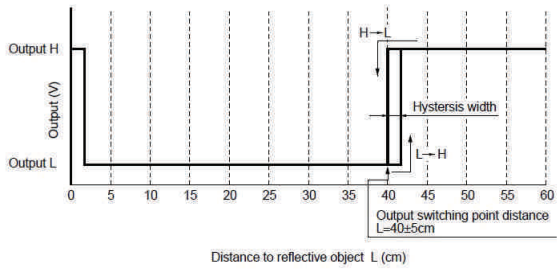


Figura 14.2: Salida del GP2Y0D340K frente a la distancia del obstáculo

Los sensores binarios presentan la ventaja de ser más pequeños que sus hermanos analógicos. Prácticamente son la mitad de largos.

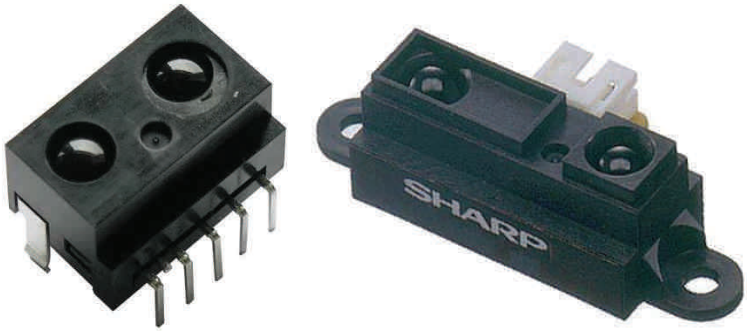


Figura 14.3: Sensores digitales y analógicos

14.1.2. Sensores Analógicos

Estos sensores dan como salida un voltaje que es mayor cuanto más cerca este el obstáculo. En la siguiente tabla podemos ver los principales sensores analógicos.

Sensor	Rango de medida (cm)	Precio (€)
GP2Y0A41SK0F	4-30	9.63
GP2Y0A21YK0F	10-80	10.1
GP2Y0D02YK0F	20-150	15.16

El sensor GP2Y0A21Y sería perfecto para nuestro minisumo dado su rango $10 - 80cm$. El único fallo que tiene es que para distancias menores de 8cm disminuye el voltaje de salida como puede verse en la gráfica siguiente.

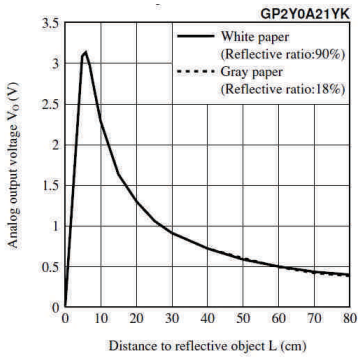


Figura 14.4: Salida del GP2Y0A21Y frente a la distancia del obstáculo

Habría que tenerlo en cuenta en el diseño para que nunca estuviese el robot enemigo completamente pegado al sensor. También podría usarse un sensor GP2Y0A41S de refuerzo pero vemos que adolece del mismo defecto.

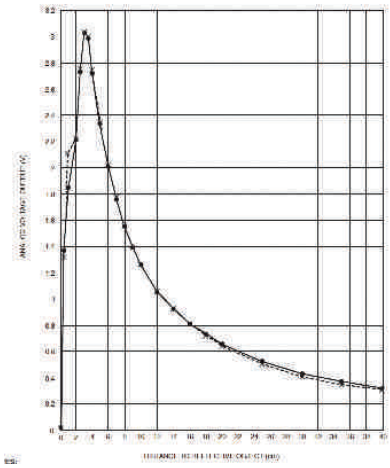


Figura 14.5: Salida del GP2Y0A41S frente a la distancia del obstáculo

En caso de elegir estos sensores habrá que realizar pruebas para ver que tensiones obtenemos al colocar el robot enemigo justo delante.

Para leer el valor de los sensores utilizaríamos el conversor analógico-digital del microprocesador.

Finalmente en la siguiente figura se puede ver el tamaño de estos sensores. Todos tienen un encapuchado parecido.

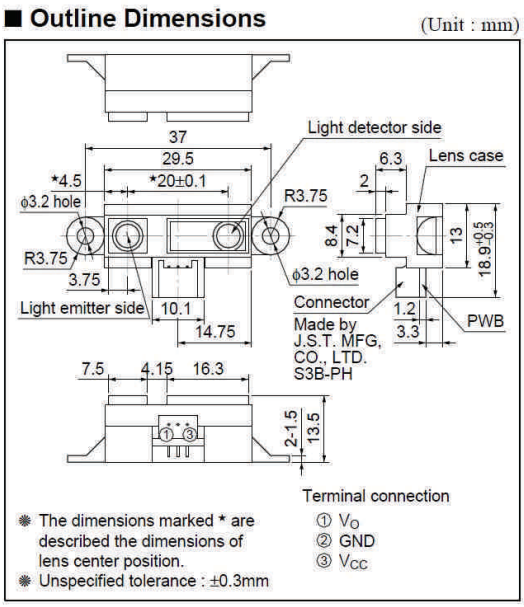


Figura 14.6: Dimensiones de la familia GP2

14.2. Sensores de Ultrasonidos

Los sensores de ultrasonidos funcionan con un método distinto a los de infrarrojos. Emiten un sonido y luego miden el tiempo que tarda en rebotar y volver la señal. Por ello necesitan una electrónica más compleja que los sensores de infrarrojos. Es por ello que no hemos sido capaces de encontrar un modelo compacto como los anteriores, además de tener un tamaño mayor que los penaliza.

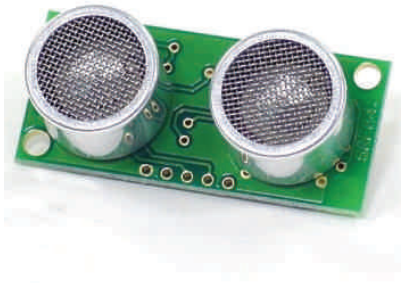


Figura 14.7: Sensor de ultrasonidos

Otra desventaja es que el rango de medida es mucho mayor. Esto en otras aplicaciones puede ser positivo pero en nuestro caso no es útil ya que las medidas podrían ser perturbadas por elementos ajenos al tatami.

Sensor	Rango (cm)
LV-MaxSonar -EZ Sensor Line	645
DUR5200 Ultrasonic Range Sensor	4-340
Parallax PING))) Ultrasonic Sensor	2-300

14.3. Elección: Sensores de Infrarrojos

Hemos decidido utilizar los sensores infrarrojos por las siguientes razones:

- Su diseño es mucho más compacto y pequeño que el de los sensores de ultrasonidos
- Su salida es un voltaje proporcional a la distancia. No necesitamos ningún circuito para adaptar la señal, podemos conectarla directamente al microcontrolador lo que facilita mucho las cosas.
- Podemos utilizar sensores analógicos o binarios según las necesidades y según la disponibilidad de pines en el microcontrolador.
- Únicamente necesitamos tres cables para cada sensor. Dos para la alimentación y uno para la salida.

14.4. Sabotear los sensores de proximidad

La única forma de poder alterar las medidas de los sensores de proximidad es diseñar el robot para que refleje el mínimo de las ondas recibidas. Esto se puede conseguir de dos maneras: haciendo un robot con diseño antiradar y utilizando materiales que absorban las ondas.

14.4.1. Robot Antiradar

Diseñando el robot para que sea lo más bajo posible y para que no tenga caras perpendiculares al suelo sino que sean curvas. El equivalente lo tenemos en los aviones antiradar.

Figura 14.8: Avión Antiradar

14.4.2. Materiales Absorbentes

Utilizar materiales y colores que absorban las ondas. Buscando en internet encontramos que el Si-O presente en el cristal y el cuarzo y el C-C absorben la luz infrarroja. También nos dice que el plexiglass absorbe la radiación infrarroja. La información no es muy fiable así que habrá que hacer pruebas.

Por otro lado vemos que los materiales porosos absorben más sonido a medida que aumenta la frecuencia. Esto nos vendría bien ya que los ultrasonidos tienen frecuencias altas. Encontramos un material llamado Espuma acústica que absorbe el sonido.

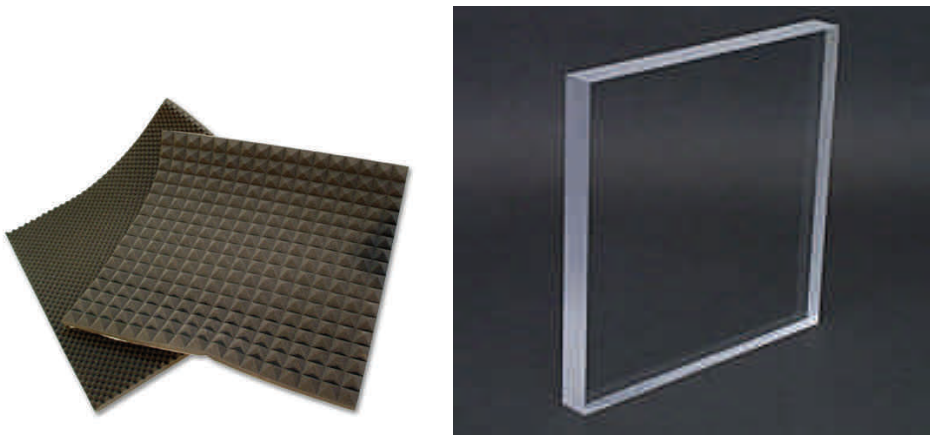


Figura 14.9: Espuma acústica y plexiglass

Como vemos no podemos usar ambos tratamientos a la vez. Es preferible utilizar el de los sensores infrarrojos ya que son más utilizados.

Este tema se trata más adelante en profundidad en la parte de Diseño en el capítulo de Robot Antiradar.

Capítulo 15

Sensores de Contacto

Podemos utilizar sensores mecánicos conocidos como Bumpers para detectar cuando estamos en contacto con el robot adversario.

15.1. Bumper

El bumper es un conmutador de 2 posiciones con muelle de retorno a la posición de reposo y con una palanca de accionamiento.

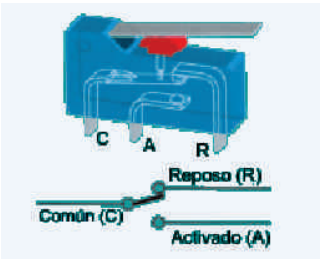


Figura 15.1: Imagen de un bumper

Funcionamiento: En estado de reposo la pata común (C) y la de reposo (R) están en contacto permanente hasta que la presión aplicada a la palanca del bumper hace saltar la pequeña pletina acerada interior y entonces el contacto pasa de la posición de reposo a la de activo (A), se puede escuchar cuando el bumper cambia de estado, porque se oye un pequeño clic, esto sucede casi al final del recorrido de la palanca.

15.2. Soluciones comerciales

Encontramos varias opciones en la tienda electrónica de Pololu. El dispositivo es el mismo y lo único que cambia es la palanca de accionamiento. Cuestan menos de un dolar por lo que son una opción interesante para corregir los problemas que presentan los sensores de infrarrojos Sharp en las distancias cortas.

Figura 15.2: Distintos bumpers de la tienda Pololu

15.3. Sabotear los Sensores de Contacto

No existe modo alguno de sabotear los sensores de contacto debido a la simpleza de su funcionamiento. Si están bien colocados en cuanto choquemos contra el contrario se activarán. La única posibilidad de que fallen es que se dañen y queden siempre activados.

Capítulo 16

Alimentación del robot

Lo ideal es alimentar los motores con una tensión entre 6-9V. Alimentarlos con tensiones superiores disminuiría su tiempo de vida y no queremos que se nos estropeen durante el campeonato de sumo.

El problema que tenemos haciendo esto es que todos los sensores suelen funcionar con una alimentación de 5V. Podemos solucionar esto gracias a que la placa Arduino tiene un regulador de tensión para conseguir los 5V necesarios. Por lo tanto utilizaremos la batería para alimentar los motores y la placa Arduino y usaremos la placa Arduino para alimentar a los sensores.

16.1. Requisitos de la batería

Hay tres requerimientos que pedimos a la batería: Voltaje, Capacidad y dimensiones reducidas

16.1.1. Voltaje

El voltaje necesario para alimentar la placa Arduino debía estar entre 7-12V y el de los motores entre 3-9V. Cuanto mayor sea el voltaje del motor mayores serán sus prestaciones como hemos visto en el capítulo de motores. Pero alimentar el motor con un voltaje excesivamente superior a su voltaje nominal afecta negativamente a la vida del motor y a la fiabilidad. La fiabilidad en un robot de campeonato es algo primordial por eso no alimentaremos los motores con un voltaje superior a 9V (el voltaje nominal de los motores examinados suele ser de 6V).

Por lo tanto buscaremos una batería que nos proporciones entre 7-9V. De no ser posible es preferible tener un voltaje mayor de 9V que uno menor de 7V ya que con un voltaje menor la placa Arduino podrá presentar problemas de funcionamiento.

16.1.2. Capacidad

El reglamento de minisumo establece que los combates son a tres asaltos de 3 minutos cada uno. Sin embargo ningún asalto dura tanto tiempo. Para llegar a una duración más acorde con la realidad vamos a tomar los tiempos del IX Campeonato de Euskadi de Microbots a través de un vídeo de Youtube[1]. Mediremos solo el tiempo que están moviéndose, obviando los 5 segundos iniciales.

Asalto	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Media
Duración (s)	4	14	4	4	3	7	4	5	7	8	8	3	4	12	6	3	7	3	3	5,1

Como podemos ver la media de duración de un asalto es de unos 5 segundos. Para tener en cuenta que luego se tarda un rato en parar el robot vamos a doblar el valor para los cálculos: supondremos que en cada asalto el robot está moviéndose 10 segundos de media.

[1] http://www.youtube.com/watch?v=TFhFqj2Y51c

Consideraremos también para los cálculos que el robot está encendido entre combate y combate aunque los motores no estén funcionando. Supondremos que se realiza un asalto cada minuto. Por lo tanto estará 10 segundos moviéndose y 50 segundos parado cada minuto.

Los requerimientos de corriente son los siguientes:

Componente	Consumo (mA)	Número	Consumo total (mA)	Coeficiente tiempo	Consumo Real (mA)
CNY70	25	4	100	1	100
GP2Y0A21YK0F	30	5	150	1	150
Pantalla LCD	160	1	160	1	160
Arduino	20	1	20	1	20
Motor	750	2	1500	0,16	240
		Total	1930	Total	670

Vemos que el consumo en un campeonato será de unos 700 mA. Por lo tanto usando una batería de 2000 mAh deberíamos tener una autonomía en torno a las 3 horas.

Sin embargo si tenemos los motores moviéndose todo el rato el consumo de corriente aumenta mucho. Con la misma batería de 2000 mAh la autonomía se quedaría en poco más de 1 hora. Esto tendremos que tenerlo en cuenta cuando estemos programando y probando el robot.

16.1.3. Dimensiones

Las dimensiones de la batería son primordiales a la hora de diseñar un robot de minisumo. Su tamaño determinará en buena medida la estructura del robot y la posición de los demás componentes.

Buscaremos por lo tanto baterías con alta densidad energética.

16.2. Tipos de Baterías

16.2.1. Baterías de Niquel-Cadmio

Estas baterías pueden tolerar corrientes de descarga muy grandes sin sufrir daños. Ademas pueden ser cargadas mediante carga rápida. Probablemente son la mejor opción calidad-precio.

Sin embargo no soportan descargas parciales entre ciclos de carga. Para cargarlas tienen que estar totalmente descargadas o se dañan reduciendo su rendimiento.

16.2.2. Baterías de Niquel-Hidruro de Metal

Tienen una densidad de energía entre un 30 y un 40 % superior a las de Ni-Cd. No presentan los problemas de memoria de las baterías Ni-Cd.

La desventaja es que requieren de cargadores especiales y que su ciclo de vida es aproximadamente un tercio del de las Ni-Cd.

16.2.3. Baterías de Ion Litio

Tienen una densidad de energía que dobla a la de una batería de NiCd. Son ideales para las aplicaciones en las que el peso y la dimensión son importantes.

Una desventaja que presentan es que empiezan a degradarse pasado un año de funcionamiento. Además tienen unas condiciones de carga muy específicas, necesitan un cargador especial porque si no pueden explotar.

16.2.4. Baterías de Polímero Ion Litio

Son parecidas a las anteriores pero son más ligeras y pueden tener cualquier forma.

Tipo de batería	Voltaje (V)	Densidad Energía (Wh/kg)	Densidad Energía (Wh/l)	Ciclos
Niquel-Cadmio	1.2	40-60	50-150	1500
Niquel-Hidruro de Metal	1.2	30-80	140-300	500-1000
Ion Litio	3.6	150-250	250-360	1200
Polímero de Ion Litio	3.7	130-200	300	500-1000

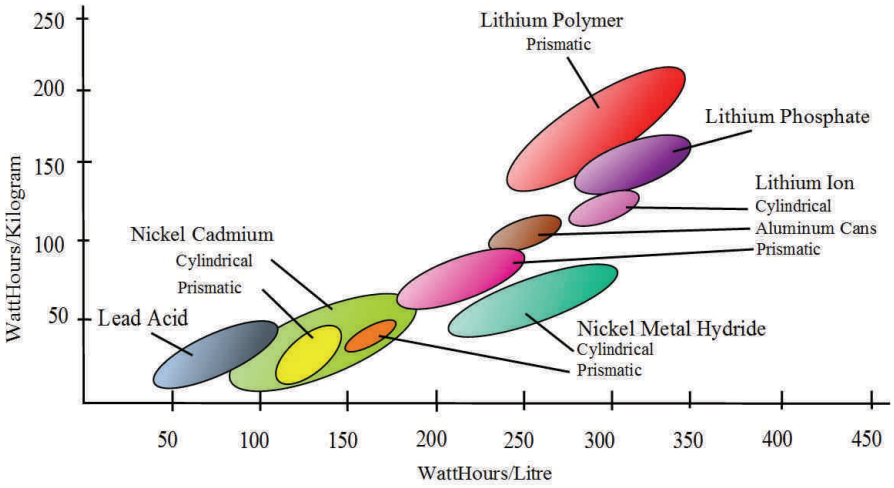


Figura 16.1: Gráfico de la densidad másica y volumétrica de la energía de distintas baterías

Elegiremos las baterías de Polímero Ion Litio por mayor densidad energética. Para colocarnos en el rango de de los 7-9V utilizaremos dos celdas en serie que nos proporcionaran 7.4V.

16.3. Soluciones Comerciales

Resumo a continuación en una tabla las distintas posibilidades. Se muestran solo baterías con un tamaño aceptable para el robot de minisumo.

Batería	Tienda	V (V)	C (mAh)	D (mm)	Precio
BATERIA LIPO 1000MAH	Bricogeek	3,6	1000	55x35x5	9,9
BATERIA LIPO 2000MAH	Bricogeek	3,6	2000	65x55x5	13,55
Lipo Battery Pack for Lama	DealExtreme	7,4	1200	65x33x18	8,96
Lipo Battery Pack for 4-CH R/C Helicopters	DealExtreme	7,4	1200	88x32x11	8,59
ZIPPY Flightmax 2S1P 15-25C	HobbyKing	7,4	1300	74x37x13	4,92
ZIPPY Flightmax 2S3P Receiver Pack	HobbyKing	7,4	2100	59x37x27	9,75
B-Grade 2S3P Receiver Pack	HobbyKing	7,4	2100	56x35x27	4,84
Rhino 2S 20C Lipoly Pack	Fingertech	7,4	750	67x35x10	6,58
15005 LiPo 2S 1P 13C	AdvantageHobby	7,4	1320	65x35x16	13,08
1550 LiPo 2S 10C GPV-1 Molex	AdvantageHobby	7,4	1250	64x32x13	13,02
1559 LiPo 2S 10C Rx Flat Rx	AdvantageHobby	7,4	1600	84x31x16	19,53
1560 LiPo 2S 2C 10C Hump Rx	AdvantageHobby	7,4	1200	57x31x21	13,02
BATERÍAS DE LI-PO PLANAS DE 7.4V	Shoptronica	7,4	1000	53.5x30x14	8,05

Una lista de cargadores es la siguiente

Cargador	Tienda	Intensidad (mA)	Precio
Mystery Balance Charger for 2/3-Cell Lithium Ion/Polymer Batteries	DealExtreme	800	7,2
Turnigy balancer & Charger 2S-3S	HobbyKing	800	3,46
Turnigy 2S 3S Balance Charger. Direct 110/240v Input	HobbyKing	650	8,81
HobbyKing E4 Balance Charger	HobbyKing	100-4500	9,71
IMAX B6 Charger/Discharger 1-6 Cells (COPY)	HobbyKing	1000-5000	12,62
CARGADOR BALANCEADOR BATERIAS DE LI-PO 2S-3S	Shoptronica	1200	9,62

16.4. Elección: Shoptronica

Finalmente hemos decidido comprar los materiales en Shoptronica debido que al ser una tienda nacional la entrega será más rápida, además que la batería se adapta perfectamente a nuestras necesidades. Como son baterías de 1000 mAh tendremos una autonomía de una media hora, por ello encargaremos dos baterías para tener absoluta seguridad.

16.5. Comprobación del estado de la batería

Una de las funcionalidades de nuestro robot será la de que comprobará el estado de la batería e indicará cuando debe ser cargada. Esto lo haremos utilizando el conversor analógico-digital de la placa Arduino. El conversor tiene un rango de medida 0-5V por lo que no podrá medir directamente el voltaje de la batería. Lo que haremos será utilizar dos baterías iguales formando un divisor de tensión y medir el voltaje entre ambas. La tensión de la batería será el doble de la tensión medida.

Parte III

Diseño del Robot

Capítulo 17

Locomoción

La elección del método que usará el robot de sumo para moverse por el tatami determinará completamente el diseño del robot. A continuación se enumeran las distintas posibilidades.

17.1. Ruedas

Es la opción más utilizada para los robots de sumo. Esto es así porque es el método más simple y eficiente para mover un vehículo sobre una superficie. Se puede establecer una clasificación en cuanto al número de ruedas.

17.1.1. Robot con dos ruedas

Un robot con dos ruedas necesitará un punto de apoyo adicional para mantener el equilibrio. Este punto adicional puede ser una rueda loca o la rampa frontal del robot. Una de las principales ventajas de usar dos ruedas es que si impedimos que el robot avance la parte delantera del robot se elevará como si fuese un caballo encabritado. Esto puede ser muy útil para volcar al robot adversario.

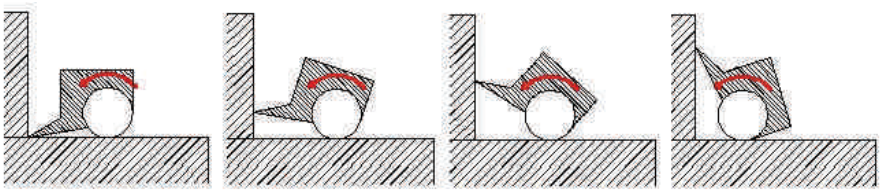


Figura 17.1: Diagrama explicativo de como se eleva la parte delantera del robot al bloquear el avance

La otra ventaja consiste en que siempre tendremos el frontal del robot a ras de suelo. Como veremos esto es muy importante cuando dos robots chocan frontalmente.

17.1.2. Robot con cuatro ruedas

La principal fortaleza del robot de cuatro ruedas es que tiene una mayor tracción que el de dos, ya que todo el peso recae sobre las ruedas(en el de dos parte del peso caía sobre la rampa frontal). Si se usan cuatro motores la fuerza que podrá hacer el motor también será mayor, o se podrán usar motores menos potentes para conseguir la misma fuerza.

Sin embargo al usar cuatro ruedas tendremos menos sitio para los otros elementos del motor. También será más difícil conseguir que la cuña delantera quede a nivel de suelo. Por último y también muy importante es que el robot de cuatro ruedas será lógicamente más caro de construir que el de dos.

17.1.3. Robot con más de cuatro ruedas

Comparten las ventajas y desventajas de los de cuatro ruedas pero amplificadas.

17.2. Orugas

Es posible utilizar orugas como las que utilizan los tanques para mover nuestro robot. Es un método visualmente muy atractivo y que proporciona una gran tracción. Sin embargo es complicado encontrar unas orugas buenas y además el giro del robot es más lento y consume más energía. Si el terreno fuese irregular sería una opción estupenda pero al ser completamente plano la sencillez de las ruedas hace que sean más elegidas.

Figura 17.2: Orugas para usar en robot de sumo

17.3. Patas

Se han construido algunos robots que se mueven caminando pero la tecnología está lejos de llegar a ser competitiva.

17.4. Modo de dirección

Una vez visto que las ruedas son la mejor opción para mover nuestro robot nos encontramos con que existen dos principales formas para controlar la dirección.

17.4.1. Dirección Ackerman

La dirección Ackerman es la que utilizan los coches. Las ruedas traseras se encargan de la propulsión y las delanteras giran para dirigir el vehículo.

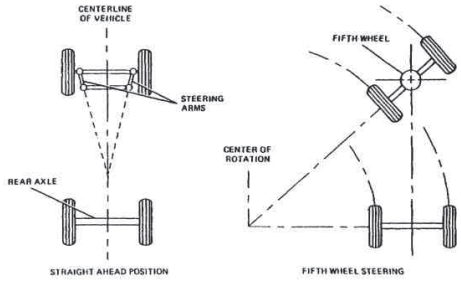


Figura 17.3: Esquema de la dirección Ackerman

17.4.2. Dirección Diferencial

La dirección diferencial consiste en que las ruedas de cada lado tienen un motor independiente. Para avanzar en línea recta giran a la misma velocidad y para cambiar la dirección giran a velocidades distintas.

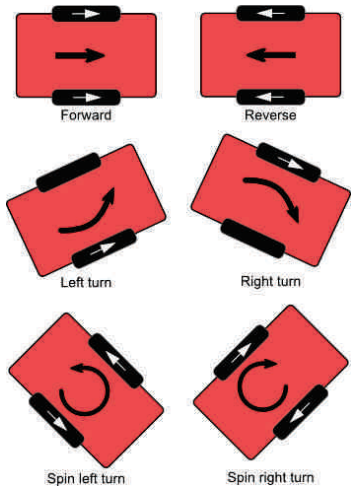


Figura 17.4: Esquema de la dirección Diferencial

17.5. Elección: Dos Ruedas

Hemos decidido utilizar finalmente dos ruedas como método de desplazamiento. Es cierto que utilizar cuatro ruedas proporciona más tracción pero si diseñamos una buena rampa frontal el robot adversario será elevado y solo tocará el suelo con las ruedas traseras. Utilizar dos ruedas va a conseguir que reduzcamos costes en el robot y nos va a obligar a realizar un esfuerzo de diseño en la rampa delantera.

Lógicamente teniendo dos ruedas vamos a utilizar la dirección diferencial.

Capítulo 18

Teoría de Rampas

El objetivo del robot de minisumo es empujar a su adversario fuera del terreno de juego. La rampa delantera será la que contacte con el robot contrario. Como vamos a ver a continuación el éxito del robot depende en gran medida de realizar un buen diseño con la rampa.

La rampa puede diseñarse con dos enfoques distintos:

1. Intentar que nuestro adversario suba por la rampa haciendo que todo el peso posible de su robot recaiga sobre el nuestro. De esta forma restamos tracción al contrario y la ganamos para nosotros.

2. Hacer una rampa muy reflectante para que cuando el robot adversario suba sobre ella sus sensores de suelo se activen como si estuviera en el borde. Entonces el robot adversario retrocederá y usaremos su propia fuerza para empujarle fuera del ring. Una debilidad de este enfoque es que se puede programar el robot para que no caiga en esta trampa, de hecho nosotros lo vamos a hacer. Simplemente hay que hacer que ignore los sensores de borde delanteros si tiene delante a su adversario. La segunda debilidad es que si al retroceder se baja de la rampa nada nos asegura que en el siguiente encontronazo vayamos a conseguir que el robot adversario suba sobre nuestra rampa. Podría pasar que nosotros subiésemos sobre la suya y perdiéramos el combate.

Vamos a usar el primer enfoque en el que diseñaremos la rampa para conseguir que el robot adversario suba sobre el nuestro de tal manera que le robemos tracción.

18.1. Fuerza horizontal y elevación del morro

Vamos a estudiar como debe ser la rampa del robot. Para ello vamos a utilizar un modelo simplificado con forma de cuña como el que se puede ver en la siguiente imagen.

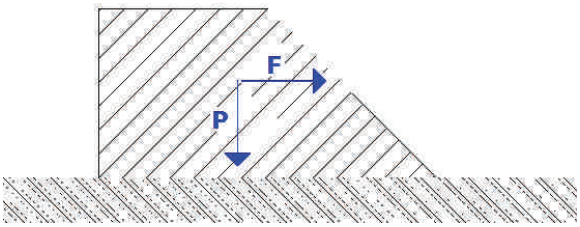


Figura 18.1: Modelo de cuña

Como vemos la cuña ejerce su peso contra el suelo y una fuerza horizontal hacia adelante. La fuerza horizontal será proporcional al peso porque como sabemos:

$$F = \mu \cdot P \tag{18.1}$$

Siendo μ el coeficiente de rozamiento.

En primer lugar podemos observar que si la parte delantera de la cuña se eleva la fuerza horizontal que realiza el robot disminuye por dos razones.

- La primera es que parte del peso del robot recae sobre el objeto que lo está elevando, por lo que las ruedas perderán tracción al recibir menos peso y la fuerza que pueden realizar disminuirá.
- La segunda es que al cambiar la dirección de la fuerza la componente horizontal inevitablemente disminuirá.

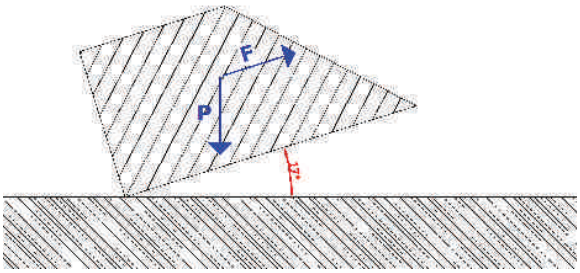


Figura 18.2: Robot con la parte delantera elevada

En este caso tendríamos que la fuerza horizontal sería:

$$H = F \cdot cos(\alpha) \tag{18.2}$$

Por lo que cuanto mayor sea el ángulo de inclinación menor será la fuerza horizontal que pueda ejercer el robot.

Por lo tanto vemos que si dos robots tiene igual fuerza pero uno consigue colar su rampa por debajo del otro podrá vencerle con facilidad. Hay que añadir que al elevar el robot adversario parte de su peso recae sobre el nuestro aumentando nuestra tracción. Por lo tanto vemos que conseguir colarnos debajo del contrario o evitar que se cuelen debajo de nosotros es vital para poder ganar el combate.

Veamos cuales son los parámetros que determinan quien eleva y quien es elevado. Fácilmente podemos observar que la pendiente es determinante.

18.2. Pendiente

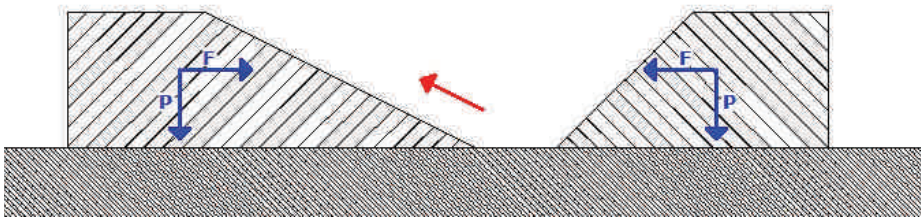


Figura 18.3: Choque entre cuñas de distinta pendiente

Como se aprecia en el dibujo cuanto menor sea la pendiente hará que sea más fácil que una cuña suba sobre otra.

Llevado al extremo la pendiente es nula con lo que tendríamos una lámina horizontal muy fina.

Es importante señalar que las consideraciones sobre la pendiente se aplican solo a la punta de la rampa. Después la pendiente podrá y deberá aumentar.

Según parece la federación de sumo ha definido como cortante todo aquello que tenga un espesor menor de 0.005 pulgadas o lo que es lo mismo 0.127mm. Habrá que buscar una placa metálica de espesor un poco mayor para realizar la rampa.

18.3. Pegada al suelo

Otro factor muy importante es que nuestra rampa vaya pegada al suelo. Como se aprecia en el siguiente dibujo si una de las cuñas no tiene contacto con el suelo estará completamente vendida frente a la otra.

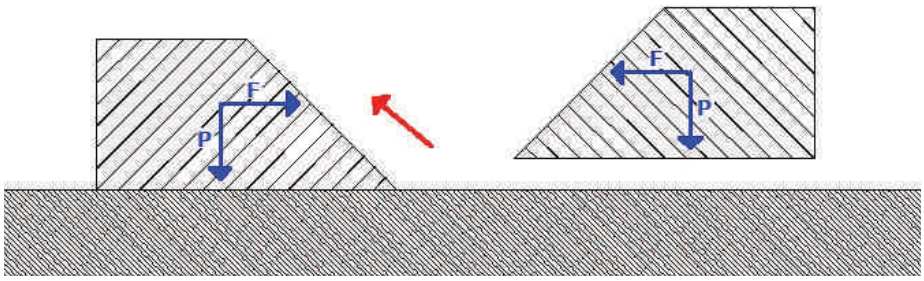


Figura 18.4: Cuña sin contacto con el suelo

18.4. Peso sobre la rampa

Es obvio que si tenemos dos rampas iguales y a una le colocamos más peso encima será la que tenga más probabilidades de quedar debajo cuando choquen ya que es la opción que minimiza la energía.

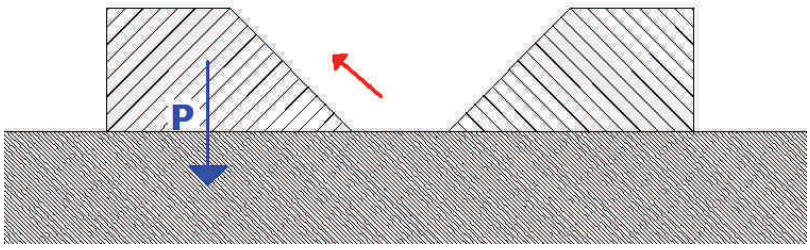


Figura 18.5: Choque entre cuñas con distinto peso

Esto hace que los robots de dos ruedas tengan una cierta ventaja sobre los de cuatro o más a la hora de introducir su rampa por debajo del adversario. Si un robot tiene dos ruedas necesariamente parte de su peso recaerá sobre la rampa. En cambio un robot de cuatro ruedas tendrá la mayor parte de su peso sobre las ruedas y muy poco o nada recaerá sobre la rampa. Sin embargo debido a esto el robot de cuatro ruedas puede proporcionar una mayor tracción que uno de dos.

Vemos que ninguna configuración es superior a la otra ya que las dos tienen ventajas y debilidades. Por ello nos hemos decantado por la opción de dos ruedas ya que es más económica y obliga a agudizar el ingenio para diseñar una buena rampa delantera.

18.5. Centro de gravedad del robot

La posición del centro de gravedad del robot influye en la facilidad con la que es levantada su parte delantera.

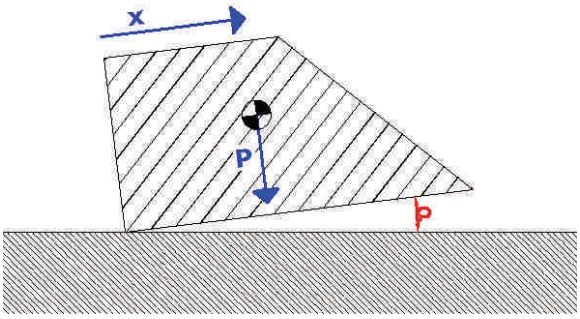


Figura 18.6: Centro de gravedad de un bloque

La variación de energía potencial resultante de elevar un ángulo pequeño la parte delantera es:

$$\triangle E = P \cdot x \cdot sen(\alpha) \tag{18.3}$$

Por lo tanto cuanto más cercano a la rampa este el centro de gravedad más difícil será que otro robot levante al nuestro.

18.5.1. Centro de gravedad regulable

Aquí vamos a introducir una innovación a la competición de sumo robótico. Vamos a realizar un robot en el que la posición del lastre y o la batería sea regulable. De esta forma podremos modificar la posición del centro de gravedad del robot.

Si nos enfrentamos a un robot con dos ruedas es presumible que será más difícil colar nuestra rampa debajo de la suya por lo que acercaremos el centro de gravedad a la rampa aún a costa de perder algo de tracción.

Cuando nos enfrentemos a un robot de cuatro ruedas retrasaremos el centro de gravedad ya que será más fácil colarnos por debajo.

Además si en el primer asalto vemos que necesitamos más peso adelante o atrás podremos modificarlo antes de que se inicie el segundo asalto. De esta forma conseguimos una ventaja muy importante sobre el resto de los robots que tienen un centro de gravedad fijo.

En ningún modelo de internet se ha visto algo parecido por lo que puede que sea algo totalmente nuevo.

18.6. Perfil de la rampa

El perfil de la rampa tiene que conjugar tres aspectos.

- Ganar altura para que el robot contrario pierda la horizontalidad
- Ganar profundidad para absorber el peso del robot contrario
- Evitar que el robot contrario pueda subir por completo encima del nuestro. La ventaja se consigue haciendo que el robot contrario tenga un punto de apoyo sobre el suelo y otro sobre nuestro robot, se pierde si deja de estar en contacto con el suelo.

18.7. Conclusión

Por lo tanto las características que debe cumplir la rampa son:

- La pendiente debe ser lo menor posible en la punta de la rampa. lo mejor es una lámina horizontal.
- La punta de la rampa debe ir perfectamente pegada al suelo
- Cuanto más peso recaiga sobre la rampa mejor, pero no olvidemos que es a costa de perder tracción.
- Cuanto más cercano a la rampa este el centro de gravedad mejor
- El perfil debe ganar altura y profundidad e impedir que el robot contrario pueda subirse encima del nuestro

Capítulo 19

Métodos Constructivos

19.1. Plástico

Hoy en día hay cientos de plásticos de uso común. Para la construcción de robots los plásticos más usados son: plástico acrílico, PVC y policarbonato. A continuación vamos a explicar como se realizan las principales operaciones de mecanizado con el plástico.

19.1.1. Doblado

Para doblar el plástico se calienta la zona a doblar y luego se dobla simplemente empujando con la mano. Para conseguir ángulos se dobla sobre un trozo de madera.

Se pueden ver ejemplos en los siguientes vídeos.
http://www.youtube.com/watch?v=iUTNcQlHiho
http://www.youtube.com/watch?v=ydpQdQXN0D4

Figura 19.1: Doblado del plástico con una fuente de calor

19.1.2. Pegado

Se pueden pegar piezas de plástico fácilmente utilizando cemento acrílico. Es un pegamento que se parece más al proceso de soldadura porque lo que hace es fusionar las piezas de plástico a unir.

En el siguiente vídeo se puede ver como se realiza el proceso de pegado.
http://www.youtube.com/watch?v=hT6Ow_cBTps

Figura 19.2: Pegado del plástico con cemento acrílico

19.1.3. Cortado

El cortado del plástico es muy sencillo ya que puede hacerse simplemente con un cúter. Primero se marca el corte con la ayuda de una regla y luego se profundiza usando solamente el cúter. Cuando la profundidad es suficiente se dobla la pieza hasta cortarla. El cúter usado en el vídeo es un cúter de garfio.

El siguiente vídeo ilustra la facilidad con la que se corta el plástico.
http://www.youtube.com/watch?v=jCeHx-vvJ7k

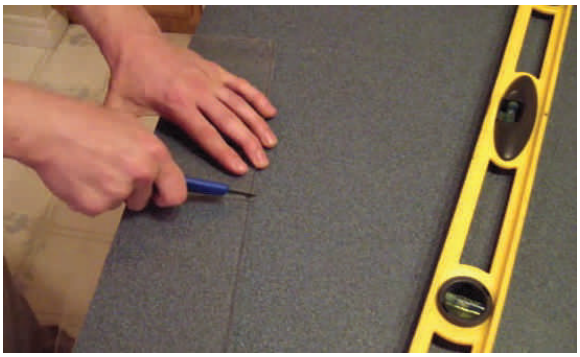


Figura 19.3: Cortado del plástico con cúter

19.1.4. Agujereado

El agujereado también es muy simple ya que el plástico no es un material duro. Se puede hacer con cualquier taladro.
http://www.youtube.com/watch?v=eKXhq04ipPU

Figura 19.4: Taladrado del plástico

19.1.5. Material Necesario

- Cemento Acrílico[1]. El bote son 6 dólares
- Hoja de acrílico[2]. La máxima finura es 1.54mm. Una hoja de 20x20 cuesta 10 dolares
- Cúter
- Fuente de calor. Podría servir un secador de pelo.
- Taladro. Uno manual sería lo más práctico ya que el espesor va a ser pequeño.
- Lija

19.2. Aluminio

De todos los metales disponibles el aluminio es el mejor para la construcción de robots. Es muy resistente y es el más ligero de los metales comúnmente disponibles. Además es relativamente barato.

19.2.1. Doblado

Si la pieza es delgada se puede doblar sujetándola a un tornillo de banco y doblándola con la mano. Si es más gruesa es necesario emplear una máquina especial para doblado.

http://www.youtube.com/watch?v=9DNHNY9BCRQ

http://www.youtube.com/watch?v=yAlqX3gFUuE

[1] http://www.tapplastics.com/shop/product.php?pid=130&

[2] http://www.tapplastics.com/shop/product.php?pid=334&

Figura 19.5: Doblado del aluminio

19.2.2. Pegado

Para el pegado la opción más sencilla es utilizar pegamentos tipo Nural 21. Se consigue una soldadura metálica en frío de gran resistencia.

Figura 19.6: Nural 21

19.2.3. Cortado

Para cortar el metal es necesario una sierra o una máquina de mecanizado.
http://www.youtube.com/watch?v=c4DT0Iv24RE
http://www.youtube.com/watch?v=Ytczt75-tqw

Figura 19.7: Cortado del aluminio

19.2.4. Agujereado

Se puede agujerear el metal con un taladro si se usa la broca adecuada. Si no es necesario utilizar maquinaria tipo CNC.

http://www.youtube.com/watch?v=D4q8UhbpaG0
http://www.youtube.com/watch?v=bhqPUGzXNRE

Figura 19.8: Taladrado del aluminio

19.2.5. Material

- Placa de aluminio. Puede ser de 1mm de espesor.
- Pegamento Nural 21
- Dobladora
- Sierra
- Taladro eléctrico

19.3. Comparación

	Plástico	Aluminio
Doblado	A mano con calor	Máquina o tornillo de banco
Pegado	Cemento acrílico	Nural 21
Cortado	Cúter	Sierra o CNC
Agujereado	Taladro manual	Taladro o CNC
Grosor mínimo	1.5 mm	0.4 mm

Cuadro 19.1: Tabla comparativa del plástico y del aluminio

19.4. Elección: Aluminio

Tras realizar unas pruebas sobre una chapa de aluminio de 0.4 mm de espesor comprada en Pinturas Iturrama se ha decidido utilizar este material. La delgadez facilita en gran medida el trabajo del material. Se puede doblar fácilmente con la ayuda de un tornillo de banco y puede cortarse haciendo varias pasadas con el cúter.

Capítulo 20

Robot Antiradar

Para fabricar un robot antiradar podemos influir principalmente sobre dos aspectos del robot: la geometría y el material utilizado.

Aunque finalmente no se aplicaron estos principios en nuestro robot incluimos la información para robots futuros.

20.1. Geometría

La geometría determinará de que manera se refleja la luz infrarroja que emiten los sensores de los robots adversarios. Dependiendo de la forma del robot reflejará más o menos la luz infrarroja que emita el contrario.

Imaginemos que tenemos una linterna y delante nuestra un espejo en posición vertical. Si apuntamos al espejo la luz rebotará y nos iluminará a nosotros por completo. Ahora imaginemos que seguimos apuntando al espejo pero que este empieza a inclinarse hacia atrás. Conforme se vaya inclinando la luz que rebota y nos llega a nosotros irá disminuyendo, y será nula cuando el espejo este en posición horizontal.

De la misma forma sucederá con nuestro robot. Si tenemos un robot que es un cubo reflejará perfectamente la radiación del contrario. Pero si tenemos un robot con forma de pirámide reflejará menos cuanto más plana sea la pirámide. Lo mismo sucede si tenemos un robot muy alto frente a uno muy bajo, el robot más alto tendrá una superficie mayor y por lo tanto podrá reflejar más que el robot bajo.

Habría que hacer pruebas inclinando un plano frente a los sensores para poder cuantificar la reducción de la radiación.

Otra prueba que habría que hacer es como responden los sensores ante superficies de distinto acabado. Es muy probable que los sensores respondan peor ante superficies irregulares, agujereadas o con un perfil de montañas como el cartón que frente a una superficie totalmente lisa. Habría que comprobarlo y cuantificarlo.

20.2. Material

En óptica la reflectividad es la fracción de la radiación incidente que es reflejada por una superficie. La reflectividad depende del ángulo de incidencia, del material y de la longitud de onda.

Teniendo en cuenta únicamente la dirección la mayoría de las superficies se pueden clasificar en reflexión especular y reflexión difusa.

- Para las superficies de reflexión especular la reflectividad será cercana a cero excepto en los apropiados ángulos de reflexión. Esto podríamos utilizarlo para colocar una superficie especular inclinada de forma que no refleje nada contra el contrario.

- Para las superficies de reflexión difusa la reflectividad es uniforme, la radiación se refleja igual en todos los ángulos.

Por otra parte si tenemos en cuenta únicamente el material podríamos buscar tablas de reflectividad pero no las encontraríamos. Podemos encontrar en cambio tablas de emisividad[1], que es la magnitud inversa a la reflectividad. Un material muy oscuro tendrá una emisividad de 1 y uno muy reflectante tendrá una emisividad cercana a 0.

Material	Emisividad
Aluminio	0.02
Latón	0.03
Plata	0.01
Acero	0.08
Papel	0.93
Pintura negra	0.96
Madera	0.8
Goma	0.86

Cuadro 20.1: Tabla con valores de emisividad

Como se puede ver hay dos aproximaciones para resolver el problema.

1. Usar un material que refleje mucho pero lo refleje en otra dirección
2. Usar un material que refleje lo menos posible

Habrá que hacer pruebas para ver cual de los dos es mejor. Es importante señalar que podemos transformar una superficie de baja emisividad como el aluminio a una de baja emisividad simplemente pintándola, pero no podemos hacer lo contrario tan fácilmente.

20.3. Avión Antiradar

Los aviones antiradar más famosos son el F-117A y el B-2.

Figura 20.1: F-117A y B-2

Ambos aviones, a pesar de su extraña apariencia y de sus diferencias, están basados en un concepto primordial: evitar que las ondas de radar regresen a la base que las emitió. ¿Cómo hacen esto? De

[1] http://www.monarchserver.com/TableofEmissivity.pdf

manera muy simple. Como primer paso, están recubiertos de un material llamado RAM ("Radar Absorving Material", o Material absorbente de radar), que los unifica con un color oscuro.

El segundo paso es lograr que la figura del avión no resulte llamativa a las ondas radar. Esto se logra diseñando especialmente el fuselaje del aparato para que el eco sea el mínimo posible. Los dos aparatos "furtivos", el F-117A y el B-2, no tienen superficies completamente verticales. Esto es, justamente, porque en las áreas verticales como las presentes en la cola de los aviones es donde se producen los ecos más fuertes, al igual que en los ángulos más agudos. Por eso las líneas de estos aviones, sean curvas o rectas, nunca dejan ver ángulos cerrados.

Como vemos los aviones antiradar siguen los mismos principios que deben usarse en un robot. Puede encontrarse más información sobre los aviones espía en el siguiente enlace[2].

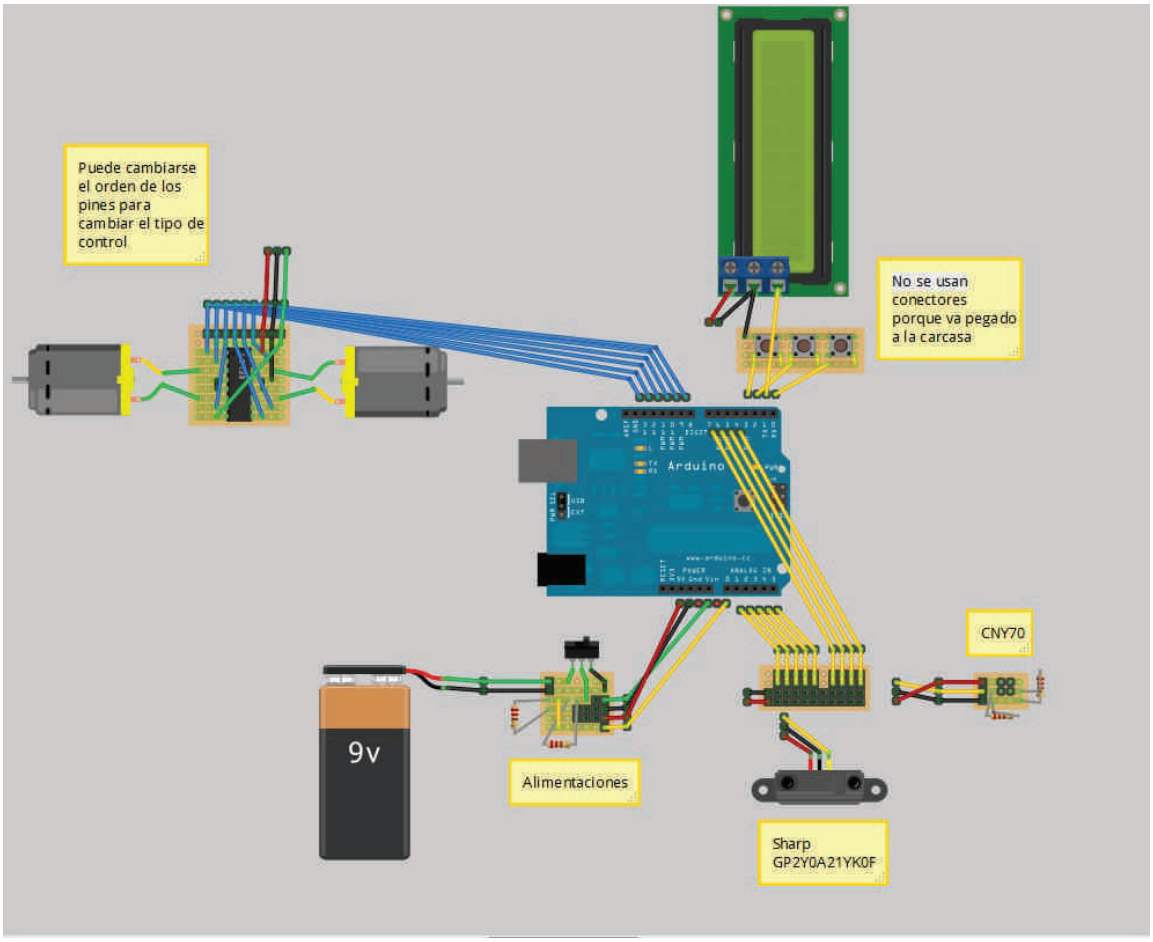
[2] http://cssbl.com/aire/stealth.htm

Capítulo 21

Cableado

En este capítulo se muestra el esquema de conexiones y los distintos componentes eléctricos que se van a utilizar para realizarlo.

21.1. Esquema Eléctrico



Cada sensor CNY70 va a tener su propia placa en la que estarán las dos resistencias del circuito de acondicionamiento. De esa placa saldrán 3 cables que llegan a una placa donde se juntan los cables de todos los sensores: los Sharp y los CNY70.

Recordamos que tenemos 5 sensores Sharp y 4 CNY70.

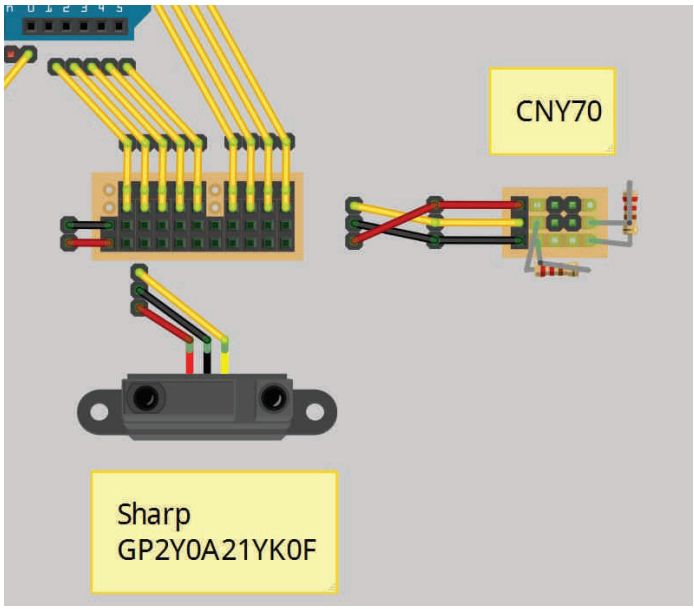


Figura 21.1: Vista de las conexiones de los sensores

Toda la alimentación confluye en una placa en la que también está el interruptor del robot. De esta forma no tenemos que colocar una placa sobre la placa Arduino ahorrando espacio.

En esta placa tenemos también el divisor de resistencias que nos permite medir el voltaje de la batería desde la placa Arduino.

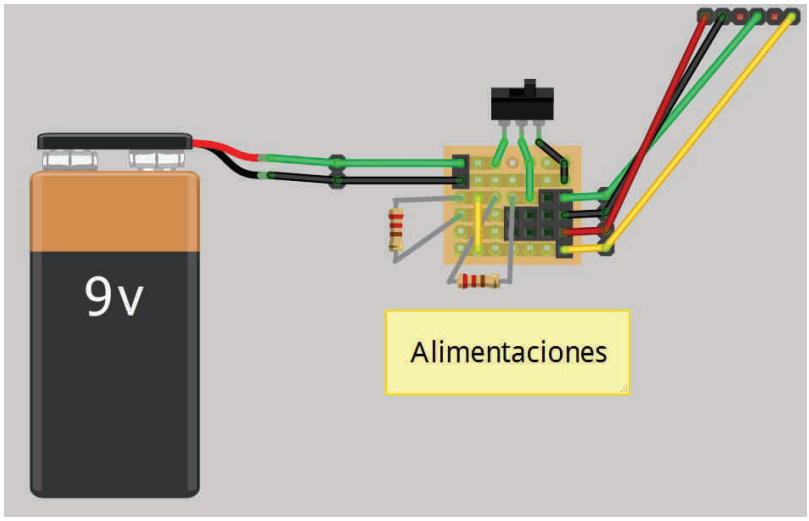


Figura 21.2: Esquema de la alimentación

El control de los motores se realiza en otra placa distinta.

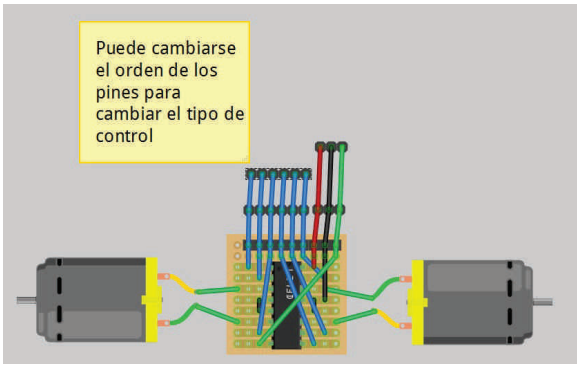


Figura 21.3: Vista del control de los motores

Usaremos una última placa para sostener los botones de control del robot.

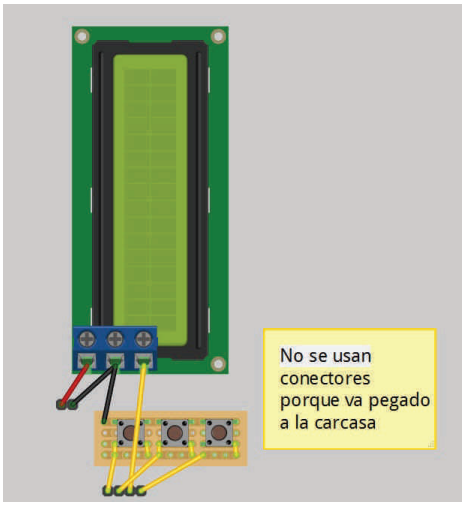


Figura 21.4: Esquema de la pantalla y los botones

21.2. Conectores Sensores Sharp

Los sensores de distancia Sharp tienen un conector JSP PH. Se venden cables con el conector ya incorporado. Los utilizaremos para poder simplificar el proceso de montado y centrarnos en otras tareas más importantes. Están disponibles en Pololu[1] y en Bricogeek[2].

[1] http://www.pololu.com/catalog/product/117
[2] http://www.bricogeek.com/shop/herramientas-de-prototipado/265-cable-para-sensor-sharp.html

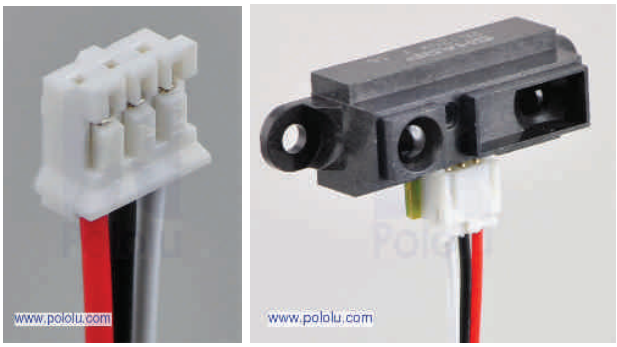


Figura 21.5: Conectores Sensores Sharp

21.3. Conectores genéricos. Para placa de prototipado y Arduino.

Es curioso que resulte difícil encontrar las cosas si son muy simples como los conectores. Por eso voy a poner las características a marcar en Farnell para poder encontrar unos conectores genéricos para placa. Hay que estar atento porque a la derecha salen productos asociados que pueden simplificar mucho nuestra búsqueda.

Características de los conectores genéricos.

- Connector type: Wire to board.
- Pitch spacing: 2.54mm.
- No. of Rows: 1
- Gender: Header para macho, receptable para hembra
- Contact Termination: Through hole Vertical

Esta pareja podría servir. Visto lo visto es la mejor, buen precio y buena calidad.
http://es.farnell.com/fischer-elektronik/sl1-025-36z/macho-pin-2-54-mm-36vias/dp/9729038?Ntt=9729038
http://es.farnell.com/fischer-elektronik/bl1-36z/macho-hembra-2-54-mm-36vias/dp/9728856?Ntt=9728856

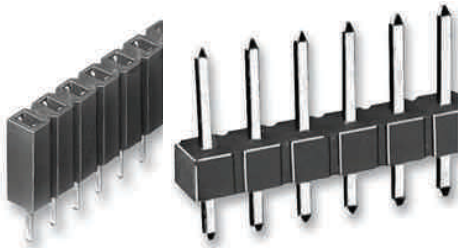


Figura 21.6: Conectores genéricos cable a placa

Alternativa para header:
http://es.farnell.com/molex/90120-0761/header-2-54mm-1way/dp/1756963
Alternativa para receptable:
http://es.farnell.com/fischer-elektronik/bl5-36z/header-socket-2-54mm-36way/dp/9728910

21.4. Resistencias para los sensores CNY70

Necesitamos cuatro resistencias de 220Ω y otras cuatro de 10kΩ. Como queremos montar las resistencias en una pequeña placa de prototipado junto a los sensores necesitamos resistencias más pequeñas de lo normal. El espaciamiento entre agujeros en la placa de prototipado es de 2.54mm, por lo que las resistencias deben ser un poco más grandes que eso. Para ello las más apropiadas son las resistencias de la serie 1206 como se puede ver en la siguiente tabla. El nombre de las series indica la longitud y anchura en pulgadas, en este caso 1206 sería 0.12 pulgadas de largo y 0.06 pulgadas de ancho.

Serie	Longitud (mm)	Anchura (mm)
0603	1.6	0.8
0805	2	1.25
1206	3.2	1.6
1210	3.2	2.6
2010	5	2.5

En Farnell tenemos por ejemplo las siguientes resistencias[3] de 220Ωque soportan una potencia de 330mW lo que permitiría una corriente de hasta 66mA, muy superior a los 20mA que van a circular realmente.

La resistencia de 10k disipará menos potencia ya que como sabemos la potencia disipada por una resistencia es $P = \frac{V^2}{R}$. Esto nos da una potencia máxima de 2.5mW. En Farnell la más barata[4] es de 125mW así que nos viene de sobra.

Figura 21.7: Resistencias de la serie 1206

21.5. Interruptor

Para evitar que se gaste la batería lo mejor sería poner un interruptor entre la batería y la placa Arduino y los motores. Lo más importante al elegir el interruptor es ver la corriente que puede soportar. Recordamos que en condiciones muy adversas los motores pedirán 800mA cada uno. Por lo tanto teniendo un interruptor que pueda suministrar 2A será suficiente.

En las hojas de características vemos que aguantan mucha más corriente a alterna que a continua. Hay interruptores de varios tipos.

[3] http://es.farnell.com/koa/sg73p2bttd2200f/resistor-surge-220-ohm-1-1206/dp/1400167
[4] http://es.farnell.com/multicomp/mc1206s4f1002t5e/resistor-thick-film-10kohm-125mw/dp/1632523

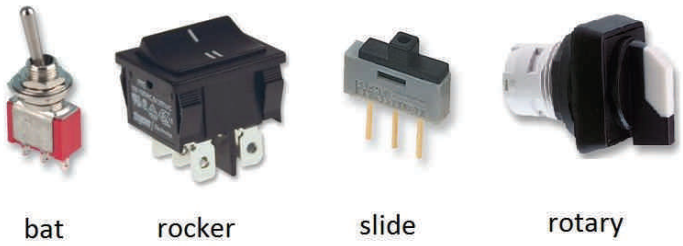


Figura 21.8: Distintos tipos de interruptores

El otro parámetro importante a la hora de elegir el interruptor es el tamaño. El interruptor no puede ir en los laterales porque correríamos el riesgo de que en una colisión el robot adversario apagase el nuestro. Por ello deberá ir en la parte superior en la que ya están la pantalla LCD y los botones de control del motor o podría ir también en la base. Por eso necesitamos que sea lo más pequeño posible para que altere mínimamente el diseño del robot.

Farnell

Modelo	Tipo	Corriente (A)	Largo (mm)	Ancho (mm)	Precio (E)
MULTICOMP - R13-608C1-02-BB-5A	Slide	16A	37.2	22.4	0.93
C & K COMPONENTS - 1101M2S3CQE2	Slide	6A	12.7	6.6	4.35
KNITTER-SWITCH - MFP106D	Slide	4A	13	7	2.69
MARQUARDT - 1811.1102-01	Toogle	10A	21	15	2.49
MULTICOMP - MC36328-001-01	Rocker	6A	22.7	15	0.56
CHERRY - PRK22J5DBBNN	Rocker	3A	15	10	1.26
TE CONNECTIVITY / ALCOSWITCH	Bat	5A	12.7	6.8	3.08

Como se puede ver en la tabla la opción que mejor conjuga tamaño y precio es KNITTER-SWITCH - MFP106D[5]. Puede verse una fotografía del interruptor a continuación.

Figura 21.9: KNITTER-SWITCH - MFP106D

21.6. Cables

Los cables van a ir conectados a las placas por medio de conectores. Por lo tanto no necesitamos cable rígido, necesitamos cable formado por hilos. Para evitar confusiones en el conexionado lo mejor es utilizar un código de colores.

[5] http://es.farnell.com/knitter-switch/mfp106d/slide-switch-spdt/dp/807527

- Rojo. 5V
- Verde. 7,4V
- Negro. Tierra
- Amarillo. Sensores y pines digitales de Arduino.

La opción más barata en Farnell es comprar bobinas de 7m. Cada bobina cuesta 3,68E. Por suerte en el laboratorio hay cable suficiente y no necesitamos comprar.

Para otra ocasión abajo están los enlaces.
http://es.farnell.com/nte-electronics/wh26-00-25/hook-up-wire-25ft-26awg-cu-black/dp/1662108
http://es.farnell.com/nte-electronics/wh26-02-25/hook-up-wire-25ft-26awg-cu-red/dp/1662110
http://es.farnell.com/nte-electronics/wh26-05-25/hook-up-wire-25ft-26awg-cu-green/dp/1662115
http://es.farnell.com/nte-electronics/wh26-04-25/hook-up-wire-25ft-26awg-cu-yellow/dp/1662113

21.7. Tubo Termoretráctil

Para fortalecer y proteger la unión del cable con los conectores es necesario utilizar tubo termoretráctil. Como los cables a emplear son de 26AWG esto significa que el diámetro interno es de 0.4mm. Por lo tanto podemos usar tubo termoretráctil de 1.5mm de diámetro interno[6] para tener en cuenta el grosor de la goma del cable.

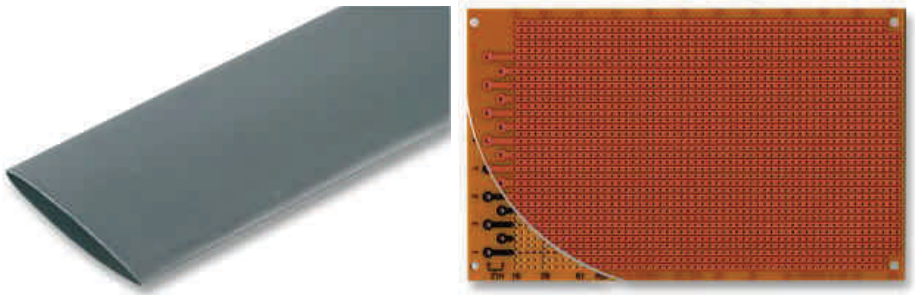


Figura 21.10: Tubo termoretráctil y placa de prototipado

21.8. Placa Prototipado

La placa de prototipado es donde montamos los conectores, driver del motor o los sensores CNY70. Compraremos una de tamaño estándar y la cortaremos para formar las pequeñas placas que hemos diseñado anteriormente.

Elegimos la más barata[7] que encontramos en Farnell.

[6] http://es.farnell.com/pro-power/hs510-1-22m/heatshrink-3-1-1-5mm-blk-1-22m/dp/1259194
[7] http://es.farnell.com/roth-elektronik/re526-hp/buch-mikrosystemtechnik/dp/1172120

Capítulo 22

Condensadores

A la hora de diseñar un circuito eléctrico se realizan muchas simplificaciones. Una de ellas es suponer que las fuentes de tensión proporcionan un voltaje constante, sin variaciones y sin ruido.

Sin embargo cuando nos trasladamos a la realidad vemos que eso no es así. La tensión proporcionada por la batería tiene ruido eléctrico como todos los componentes reales. La tensión sufre variaciones debidas por ejemplo a que la intensidad requerida en cada momento es distinta. Hay muchos componentes conectados a la batería y la intensidad que necesitan varía con el tiempo, por ejemplo el motor necesitará más corriente conforme mayor sea el par que tenga que efectuar. Por último la batería no es ideal y perfecta y las imperfecciones generarán variaciones en el voltaje.

Vemos que por distintas razones no vamos a tener una tensión constante para alimentar nuestros componentes. Los componentes han sido diseñados para trabajar con una tensión constante por lo que las variaciones van a afectar a su rendimiento y por extensión al del robot. El robot presentará comportamientos indeseados debido a las variaciones de voltaje.

Por suerte para mitigar estos cambios podemos recurrir a los condensadores. Un condensador se puede idealizar como una resistencia que varia con la frecuencia de la tensión. Si la tensión es continua el condensador tendrá una impedancia infinita, si la tensión tiene una frecuencia muy alta la impedancia del condensador será muy baja.

De esta forma colocando un condensador entre la tensión de alimentación y la tierra podemos atenuar las variaciones de tensión. El condensador mantendrá la componente continua de la tensión y absorberá la componente variable.

Cuanto mayor sea la capacidad del condensador mayor será su poder para mantener la tensión continua.

22.1. Sensores Sharp GP2Dxxx

Los sensores Sharp GP2D son muy utilizados en robótica como sensores de distancia debido a su disponibilidad y su bajo precio. Sin embargo estos sensores fueron diseñados para otras tareas menos exigentes como por ejemplo el tirado automático de la cadena en un inodoro.

Por ello al utilizar estos sensores en un robot encontramos que la señal que proporcionan es bastante ruidosa.

La solución es tan sencilla como colocar un condensador entre VCC y GND del sensor. El condensador debe estar lo más cerca posible del sensor, si esta lejos se mitiga su efecto.

Un condensador de 100nF eliminará los picos y un segundo condensador de entre $10 - 100\mu F$ servirá para eliminar la mayor parte de los rebotes de la señal.

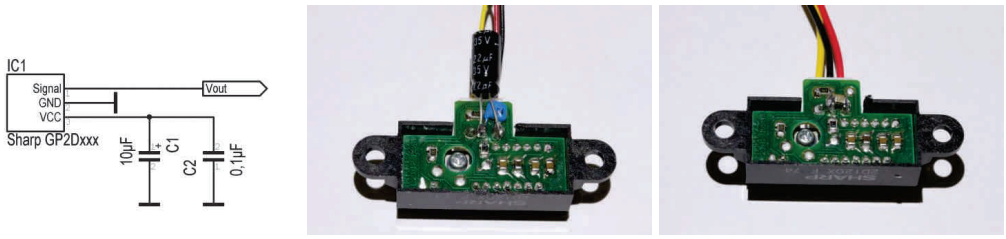


Figura 22.1: Esquema y montaje, el segundo con condensadores SMD

En el siguiente vídeo[1] se puede apreciar la mejora en la señal al introducir los condensadores.

22.2. Driver Motores L293D

A la hora de utilizar el L293D también es recomendable utilizar condensadores.

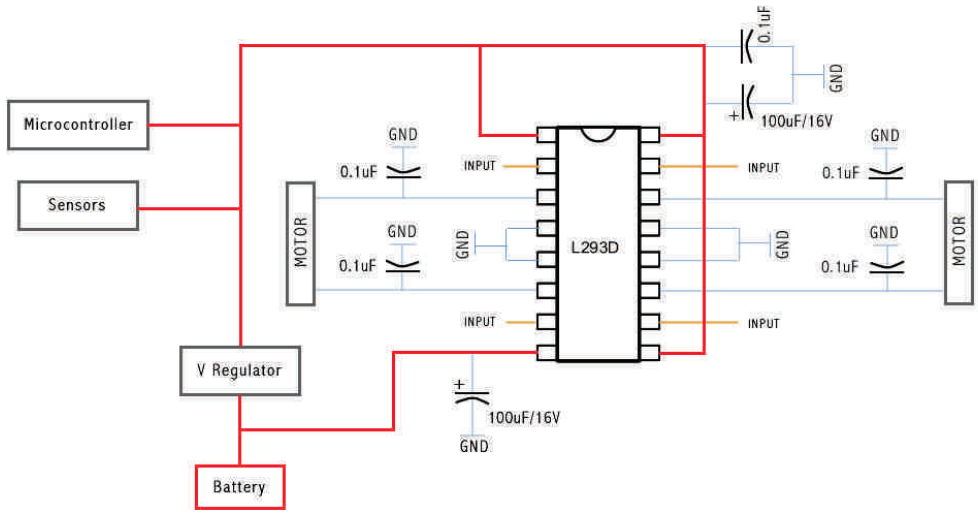


Figura 22.2: Circuito con los condensadores

Los condensadores de $100\mu F$ sirven para estabilizar la tensión, tanto la de la batería como la del regulador de voltaje de 5V. Los condensadores de $0{,}1\mu F$ absorben los picos de corriente en los bornes del motor.

22.3. Batería y Regulador de Tensión

Finalmente es altamente recomendable colocar condensadores en los bornes de la batería y del regulador de tensión. Aunque ya los hemos colocado antes de los componentes que los necesitan, ponerlos en las fuentes de tensión hará que mejore la calidad de la tensión suministrada.

El siguiente esquema muestra los condensadores a utilizar.

[1] http://www.youtube.com/watch?v=fgmQ4G8u1pg

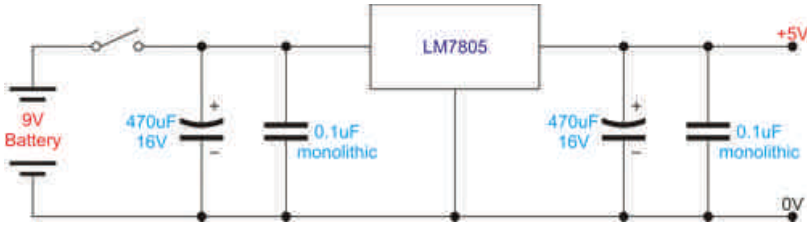


Figura 22.3: El regulador de tensión puede ser cualquiera

Los condensadores grandes ayudan al regulador cuando hay variaciones grandes de la intensidad como al encenderlo. Los condensadores pequeños eliminan el ruido y los picos.

22.4. Condensadores necesarios

Capacidad	Sensores Sharp	Driver	Alimentación	Total
$0{,}1\mu F$	5	5	2	12
$10\mu F$	5	0	0	5
$100\mu F$	0	2	0	7
$470\mu F$	0	0	2	2

Capítulo 23

Peso del Robot

En este capítulo vamos a contabilizar el peso de los elementos fijos para calcular cuanto peso disponemos para elaborar la estructura.

Componente	Peso (g)	Unidades	Total (g)
150:1 Micro Metal Gearmotor	10	2	20
Pololu Wheel 80x10	20	2	40
Pololu Gearmotor Bracket	5	2	10
Arduino Duemilanove	26	1	26
Sparkfun 16x2 serLCD	30	1	30
CNY70	1	4	4
GP2Y0A21Y	4	5	20
Batería LIPO 7,4V (estimado)	60	1	60
		Total	210

Por lo tanto nos quedan 290 gramos para realizar la estructura. Lo más seguro es que nos sobre y debamos utilizar unos lastres.

Capítulo 24

Análisis de los Prototipos

En este capítulo vamos a ir analizando los distintos diseños. Vamos a señalar las debilidades y fortalezas de cada uno para ir marcando la dirección a seguir en el siguiente prototipo. De esta forma mediante un proceso iterativo podremos llegar a un diseño correcto.

Vamos a empezar el análisis por el Mark II ya que el primero era muy preliminar y faltaban algunos componentes como las baterías. En primer lugar comentaremos las debilidades porque nos mostrarán que es lo que debemos reforzar de cara al siguiente diseño. Después ya comentaremos las fortalezas que queremos conservar en el siguiente modelo.

Los criterios que se han seguido a la hora de diseñar son los siguientes:

- Rápido
- Flexible
- Inteligente
- Estético

24.1. Mark II

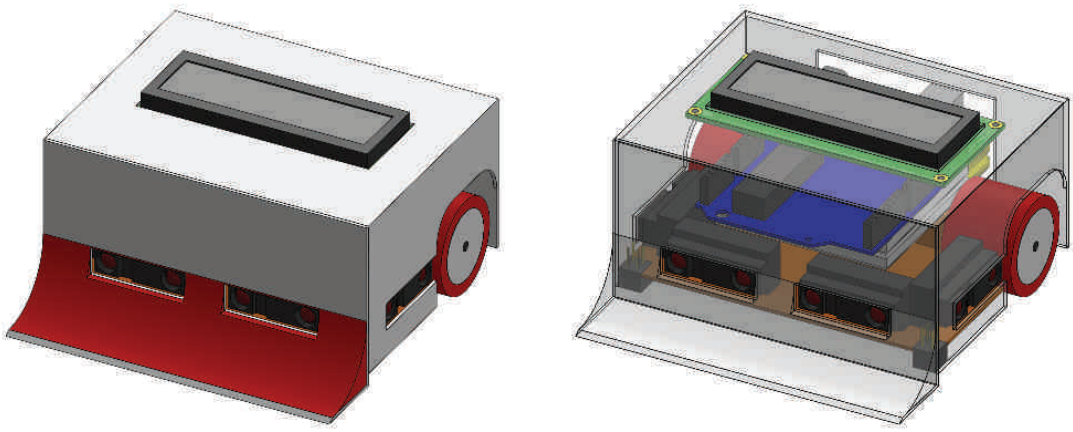


Figura 24.1: Vista delantera del Mark II

24.1.1. Debilidades

- La altura es mayor de lo que me gustaría. El punto más alto del robot se sitúa a 54.8mm sobre el suelo. Si tenemos en cuenta que el diámetro de las ruedas es de 28.6mm vemos que hay 26mm de altura que podrían ser eliminados. Eliminando la pantalla LCD podríamos reducir la altura en 11mm, pero no es una opción admisible.
- La distancia teórica entre las batería y las ruedas es de 1.44mm. Habrá que asegurar la rigidez del montaje para evitar que se toquen. Una solución podría ser utilizar unas ruedas menos anchas. Otra podría ser utilizar baterías más pequeñas.
- La carcasa exterior es demasiado cuadriculada, habría que buscar un diseño más atractivo. Tratar de redondear las formas, estilizar el robot.
- El espacio no está bien aprovechado. Hay mucho hueco en el interior de la carcasa

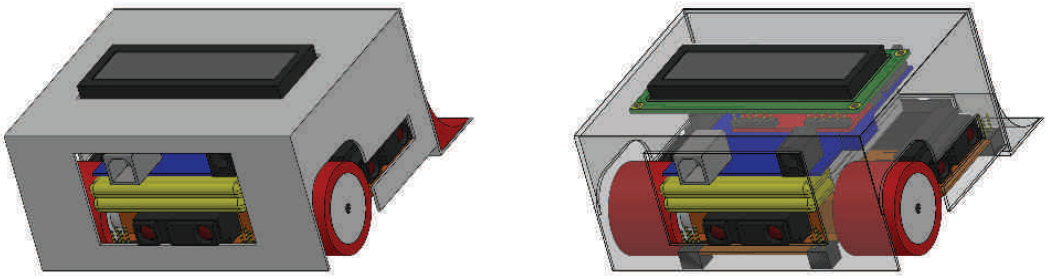


Figura 24.2: Vista trasera del Mark II

24.1.2. Fortalezas

- Se reservan 23mm para la rampa frontal, lo que está bastante bien. El diseño de la rampa podrá ser más agresivo cuanto más espacio tenga.
- La batería puede extraerse fácilmente, tiene el puerto Usb para programar la placa Arduino también perfectamente accesible.
- No parece que vaya a haber problemas con el cableado, hay espacio suficiente para que pasen los cables.
- Hay un hueco en el interior en el que meter lastre y poner el puente en H para los motores.
- La pantalla podría atornillarse al techo y utilizar cableado largo
- Los sensores de distancia delanteros podrían retrasarse si no detectasen al robot contrario cuando está muy cerca.
- El montaje una vez construido el chasis no sería complicado
- Podría hacerse que la rampa delantera fuera intercambiable

24.2. Mark III

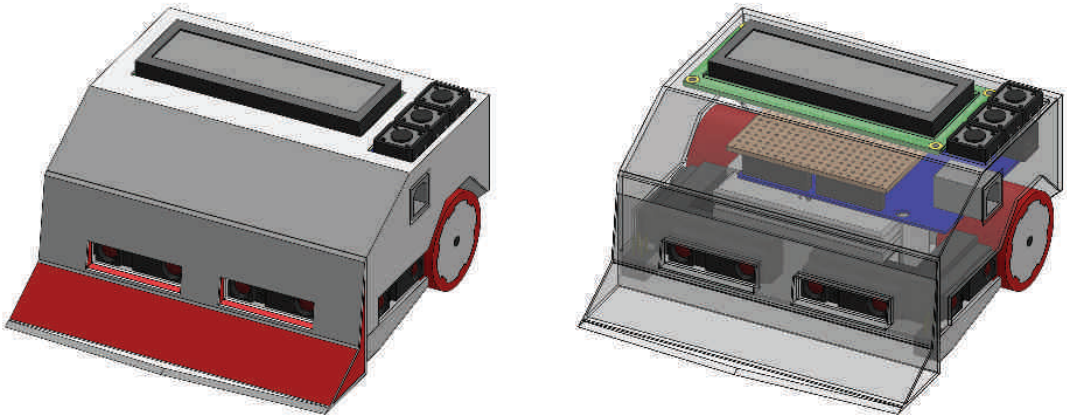


Figura 24.3: Vista delantera del Mark III

24.2.1. Debilidades

- La altura sigue siendo mayor de lo que me gustaría. El punto más alto del robot se sitúa a 54.9mm sobre el suelo.
- La distancia teórica entre las batería y las ruedas sigue siendo de 1.44mm. Habrá que asegurar la rigidez del montaje para evitar que se toquen. Una solución podría ser utilizar unas ruedas menos anchas. Otra podría ser utilizar baterías más pequeñas.

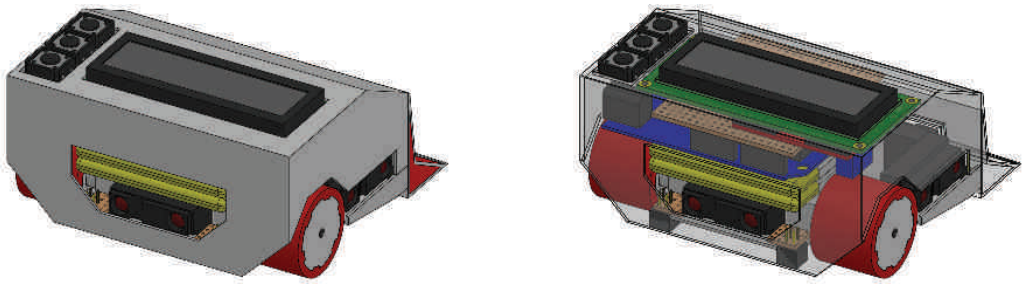


Figura 24.4: Vista trasera del Mark III

24.2.2. Fortalezas

- El espacio interior está mejor aprovechado, todo está más comprimido y el espacio vacío es menor.

- Las formas del robot se han estilizado, se ha perdido la forma cúbica del anterior prototipo.
- El acceso a la batería y al USB es inmediato
- Se consigue reservar 27mm para la rampa frontal, 4 más que el diseño anterior.

24.3. Mark IV

En este prototipo se cambia el concepto. Se va a diseñar un robot que comience en posición vertical y se mueva hacia atrás para que caiga y se coloque horizontal. De esta forma como no hay restricciones de altura podemos hacer un robot más largo de lo normal.

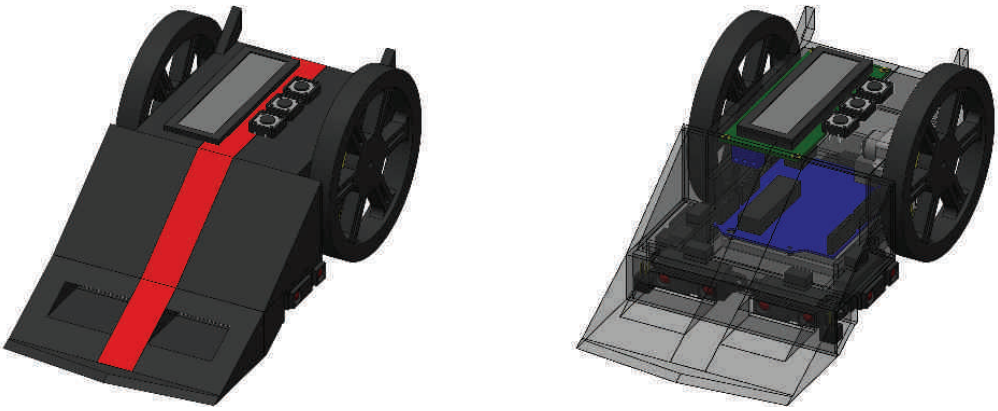


Figura 24.5: Vista delantera del Mark IV

24.3.1. Debilidades

- El espacio interior no está aprovechado al máximo. Para poder poner la placa Arduino debajo de los motores es necesario desplazarla hacia la parte delantera del robot, dejando un espacio vacío atrás. También hay un espacio vacío entre los motores y la parte trasera.
- La pantalla LCD no cabe extendida entre las dos ruedas, hay que girarla y se pierde la simetría. Podría buscarse una pantalla más pequeña.
- Quizá sea excesivamente largo.
- La posición de la batería no está clara, no esta seguro que sea fácil extraerla.
- Al comenzar el combate tiene que ponerse en posición horizontal perdiendo un poco de tiempo
- Las ruedas no son muy gruesas y eso hace que proporcionen una tracción menor que los prototipos anteriores

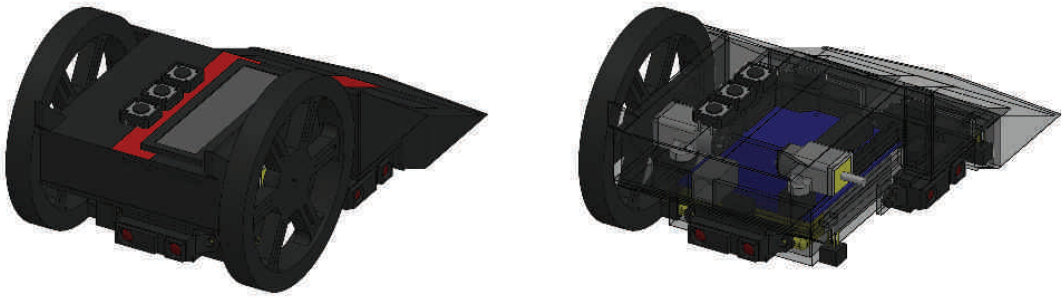


Figura 24.6: Vista trasera del Mark IV

24.3.2. Fortalezas

- El cableado va a ser sencillo, las conexiones encajan de una manera natural y no hay que hacer vueltas extrañas ni agujeros para poder conectar los sensores.
- Al comenzar en posición vertical no hay restricciones al espacio que puede ocupar la rampa delantera. Se podrá experimentar con total libertad hasta encontrar la mejor solución.

24.4. Mark V

Este prototipo sigue el mismo concepto que el anterior: un robot que se gire al comienzo de del combate. Sin embargo el diseño fue mucho más fluido y todo parecía encajar naturalmente.

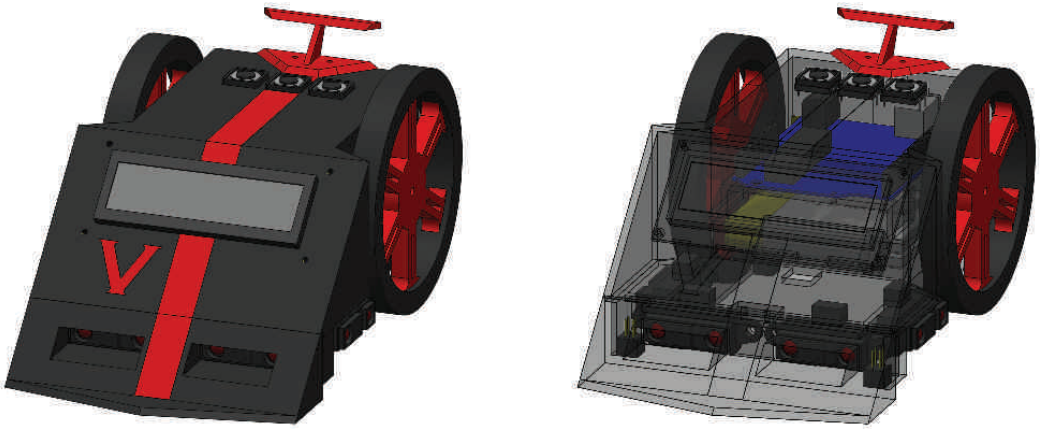


Figura 24.7: Vista delantera del Mark V

24.4.1. Debilidades

- Quizá sea excesivamente largo

- Al comenzar el combate tiene que ponerse en posición horizontal perdiendo un poco de tiempo
- Las ruedas no son muy gruesas y eso hace que la tracción proporcionada sea menor que prototipos anteriores

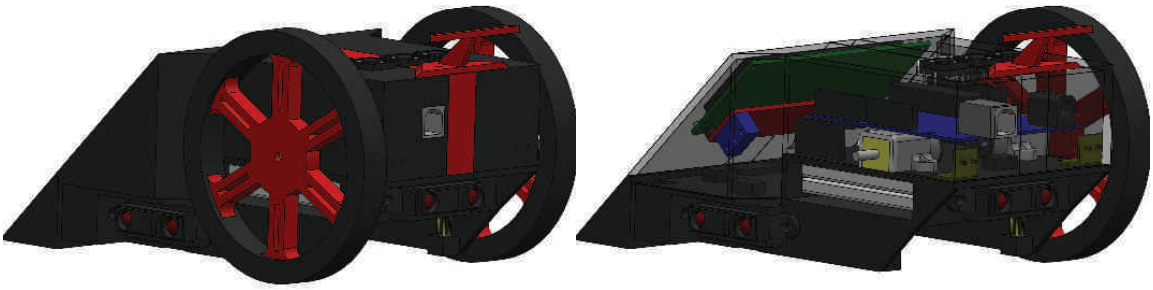


Figura 24.8: Vista trasera del Mark V

24.4.2. Fortalezas

- El cableado va a ser sencillo, las conexiones encajan de una manera natural y no hay que hacer vueltas extrañas ni agujeros para poder conectar los sensores.
- Al comenzar en posición vertical no hay restricciones al espacio que puede ocupar la rampa delantera. Se podrá experimentar con total libertad hasta encontrar la mejor solución.
- Hay un hueco habilitado en los bajos para poder colocar los lastres fácilmente
- El alerón trasero que sirve para apoyar el robot cuando está en posición vertical también servirá para coger el robot si se sale del ring.
- La nueva posición de la pantalla permite reubicar la placa Arduino en una mejor posición
- La carcasa tiene unas bisagras para poder operar fácilmente el interior del robot.
- Se han realizado las uniones de las piezas mediante tornillos
- La batería puede extraerse fácilmente por un lateral simplemente quitando la rueda

24.5. Mark VI

Este prototipo es una evolución directa del anterior. El principal cambio es la utilización de dos ruedas en cada lado para solucionar el mayor problema de los dos prototipos anteriores, la falta de tracción. Para ello se ha tenido que adelgazar el chasis entre las ruedas.

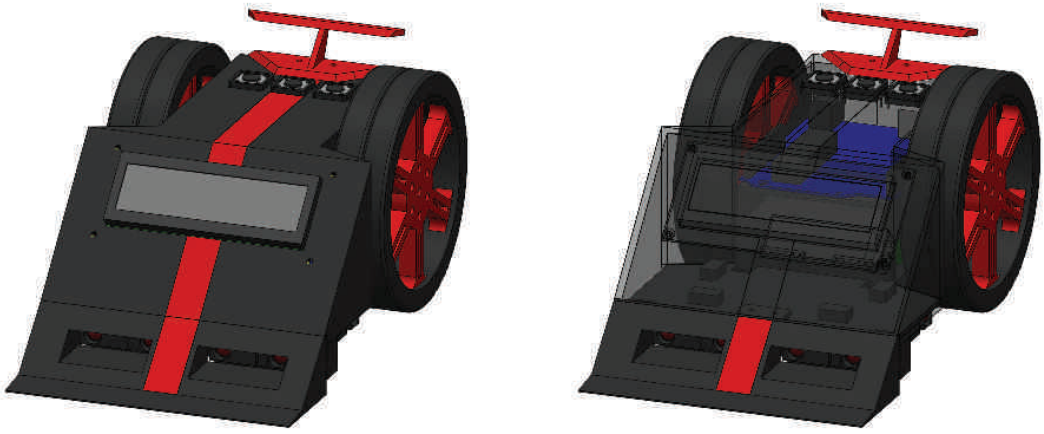


Figura 24.9: Vista delantera del Mark VI

24.5.1. Debilidades

- Quizá sea excesivamente largo
- Al comenzar el combate tiene que ponerse en posición horizontal perdiendo un poco de tiempo
- La capacidad de la batería se ha reducido a la mitad debido al estrechamiento del chasis. No obstante la autonomía sigue siendo elevada y el cambio de batería es muy rápido y sencillo. Sin embargo quizá sea posible colocar una segunda batería en los bajos del robot.

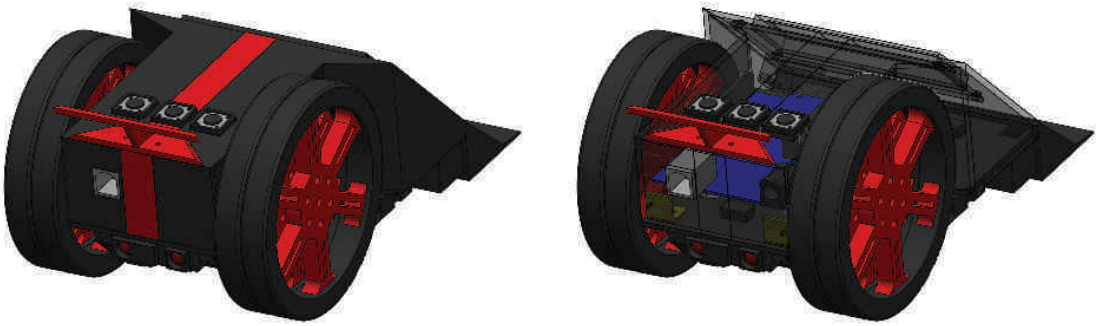


Figura 24.10: Vista trasera del Mark VI

24.5.2. Fortalezas

- Todas las fortalezas del modelo anterior las comparte este
- En los bajos del robot se ha habilitado un mecanismo para colocar lastre o una segunda batería. Este mecanismo es móvil y permite modificar la posición del centro de gravedad del robot.

Capítulo 25

Presupuesto

Sensores de distancia	*Unidades*	*Precio Unidad*	*Total*
Sharp GP2Y0A21YK0F	5	12	60
		Total	60

Sensores de Infrarrojos	*Unidades*	*Precio Unidad*	*Total*
Infrarrojos CNY70	4	0,75	3
		Total	3

Interfaz del robot	*Unidades*	*Precio Unidad*	*Total*
Serial Enabled 16x2 LCD	1	24,95	24,95
TE CONNECTIVITY - FSM101	3	0,38	1,14
		Total	26,09

Motor y ruedas	*Unidades*	*Precio Unidad*	*Total*
150:1 Micro Metal Gearmotor HP	2	13	26
Pololu Gearmotor Bracket Extended Pair	1	4	4
Pololu Wheel 80x10mm Pair	2	7	14
L293D	2	5	5
		Total	49

Microcontrolador	*Unidades*	*Precio Unidad*	*Total*
Arduino Duemilanove	1	23,6	23,6
		Total	23,6

Chasis y carrocería	*Unidades*	*Precio Unidad*	*Total*
Chapa Aluminio 0,4 mm grosor	2	3,9	7,8
Nural 21	1	6	6
Tornillos, arandelas y tuercas	1	10	10
		Total	23,8

Circuitería	*Unidades*	*Precio Unidad*	*Total*
Cable JST para sensor SHARP	5	1,4	7
MACHO, PIN, 2,54 mm, 36VÍAS	1	1,37	1,37
MACHO, HEMBRA, 2,54 mm, 36VÍAS	1	2,69	2,69
220 OHM 1 % 1206, 330mW	10	0,07	0,69
10 kOHM 1 % 1206, 125mW	10	0,01	0,12
KNITTER-SWITCH - MFP106D	1	2,69	2,69
ROTH ELEKTRONIK - RE526-HP	1	3,62	3,62
PRO POWER - HS510-1.22M - 3:1, 1,5 mm	1	0,54	0,54
		Total	18,72

Batería	*Unidades*	*Precio Unidad*	*Total*
Batería LIPO 1000 mAh 7,4V	2	8,05	16,1
Cargador batería LIPO 2S-3S	1	9,62	9,62
		Total	25,72

		Total	229,63

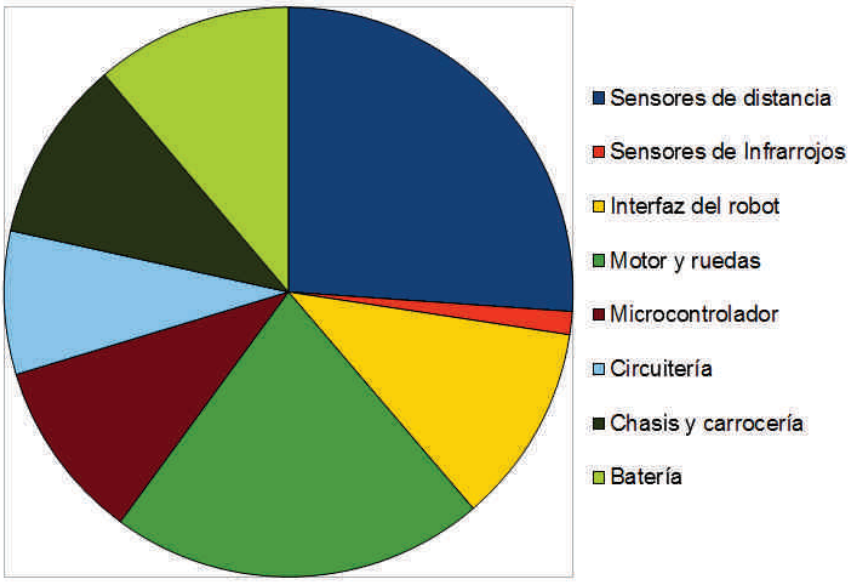


Figura 25.1: Distribución del gasto

Parte IV

Montaje del Robot

Capítulo 26

Chasis y carcasa

Para elaborar el primer prototipo se decidió utilizar chapa de aluminio de 0.4mm de espesor. Se utilizó porque se podía doblar y cortar con facilidad, sin necesidad de maquinaria. De esta forma al tratarse de un prototipo se podían introducir cambios sencillamente gracias a su facilidad de mecanizado. Las chapas se compraron en Pinturas Iturrama y tienen un tamaño aproximado de 25x50cm y un precio de 3.9 euros.

26.1. Pasos para construir una pieza

- El primer paso consiste en diseñar la pieza. Para ello lo mejor es utilizar un programa de CAD y crear modelos 3D de los componentes que va a tener el robot como pueden ser sensores, motores, batería... De esta manera tenemos referencias de los tamaños de los componentes y se puede diseñar el chasis adecuadamente. Hay que buscar la simplicidad, cuanto más simple sean las piezas más fácil será construirlas.
- Una vez las piezas han sido diseñadas hay que pensar como construirlas. Para ello hay que tener en cuenta que como estamos trabajando con metal podemos realizar las siguientes operaciones: cortado, doblado, agujereado y pegado. Probablemente habrá que dividir la pieza en otras más sencillas que luego pegaremos. Es importante fijarse si podremos realizar todos los dobles que deseamos o si por el contrario la pieza tiene demasiados dobles y es necesario dividir la pieza en partes. Como resultado de este proceso obtendremos los planos a partir de los que fabricaremos las piezas.
- Ahora imprimimos los planos con el ordenador y los pegamos a la chapa de aluminio utilizando pegamento de tubo, el pegamento que se utiliza en el colegio. La ventaja de usar este pegamento es que es instantáneo, no tenemos que esperar, proporciona una buena fijación y sobretodo es fácil de retirar una vez la pieza ha sido terminada. Un consejo es que si hemos pegado el plano el día anterior y aún no hemos mecanizado la pieza volvamos a pegarlo porque la unión se debilita pasadas unas horas. También es importante imprimir los planos a escala 1:1 si no queremos tener disgustos cuando tengamos el robot montado y no mida lo deseado. En el pegado tratar de aprovechar el espacio al máximo.

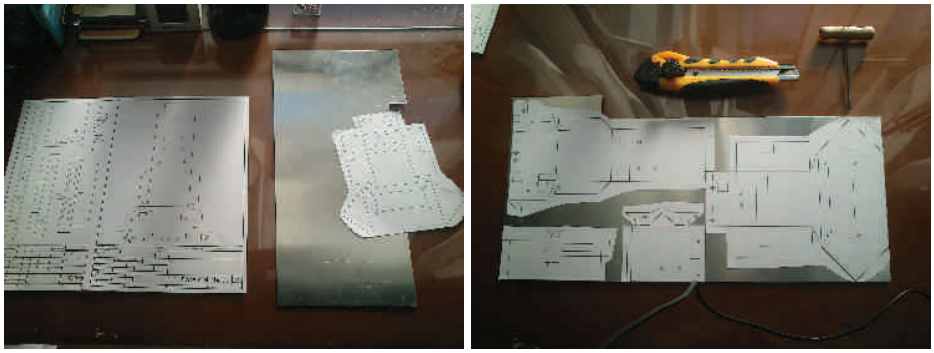


Figura 26.1: Impresión y pegado de los planos

- Ahora vamos a cortar las piezas. Como ya se ha dicho anteriormente la chapa de aluminio escogida es muy sencilla de mecanizar. Para ello nos ayudamos de un cúter y de una regla. Con ellos vamos marcando los perfiles de las piezas. Realizamos varias pasadas para que la cuchilla penetre bien en el aluminio. Para evitar equivocaciones es buena idea pintar los interiores de las piezas para diferenciarlas del material que hay que retirar.

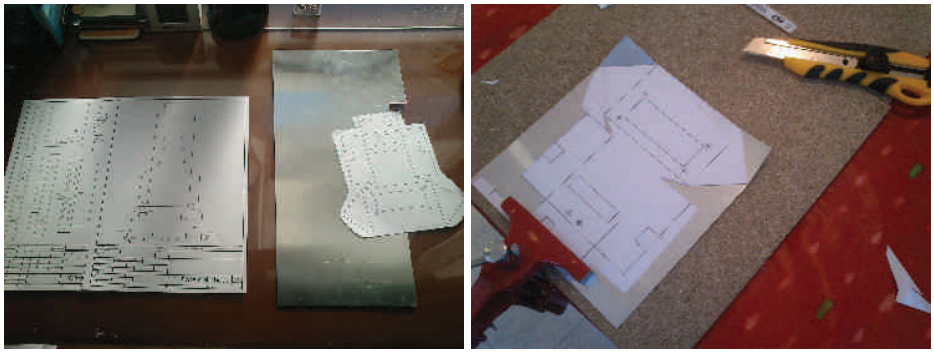


Figura 26.2: Marcamos con el cúter los perfiles de las piezas en la chapa

- Una vez hayamos marcado el perfil de la pieza el corte se realiza doblando la chapa por la zona marcada. Si la profundidad de la marca es adecuada la chapa se doblará sin problemas y finalmente se partirá por la zona marcada. Si el corte es en el interior de la pieza, p.e. en la carcasa el agujero que hay para el cable USB, no podremos ayudarnos del doblado. En este caso tendremos que completar el corte utilizando el cúter y un punzón o taladro manual. Lo primero que haremos será agujerear las esquinas del corte. Después colocaremos la chapa sobre dos tablones de madera dejando un hueco pequeño entre ellos. Colocaremos la chapa para que la zona a cortar este encima del hueco. Luego con el cúter ayudándonos de los agujeros realizados y del marcado anterior recortaremos la pieza.

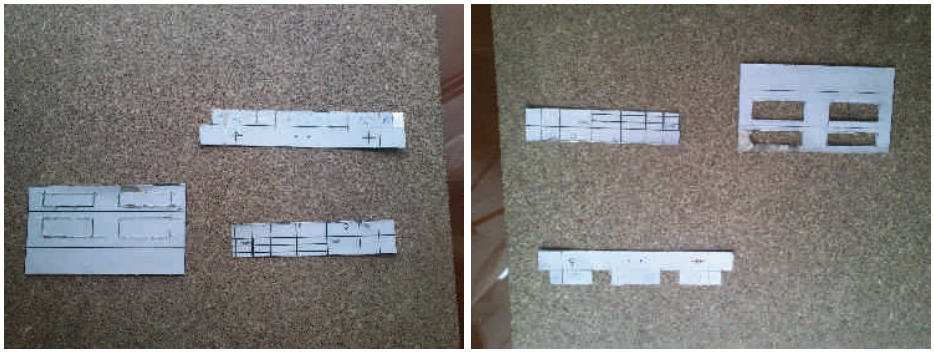


Figura 26.3: Doblamos para cortar por las zonas marcadas

- Agujereado. Ayudándonos de un punzón con el diámetro deseado realizaremos los agujeros indicados en los planos. Cuando se realizan los agujeros es frecuente que en el lado contrario de la chapa se acumule parte del material retirado. Es aconsejable retirarlo utilizando una lija, el cúter o simplemente colocando la chapa entre dos tablones y ejerciendo presión con un tornillo de banco o dándole un golpe con un martillo.

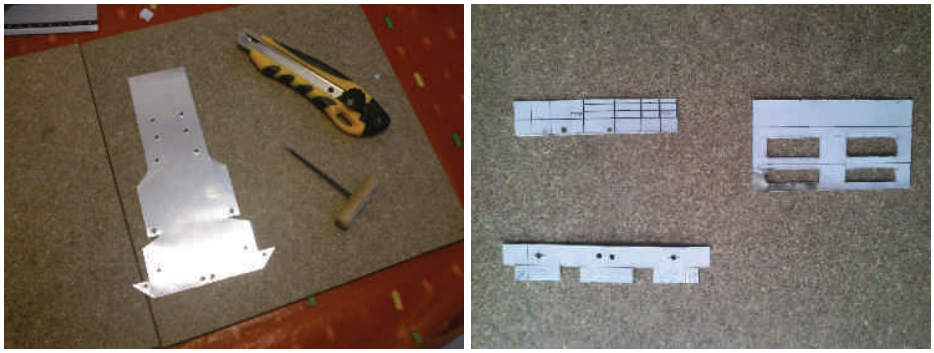


Figura 26.4: Agujereado

- Doblado. El doblado se realiza con la ayuda de un tornillo de banco y varios tablones de madera. En primer lugar se coloca la chapa entre dos tablones dejando asomar la zona a doblar y se fija la posición con el tornillo de banco. Seguidamente con la ayuda de otro tablón se dobla la chapa. Es un proceso sencillo y que no requiere apenas fuerza. Ángulos mayores de 90^{o} son difíciles de conseguir, es mejor invertir el diseño para tener un ángulo de menos de 90^{o}.

Figura 26.5: Doblamos para conseguir las geometrías deseadas

- Pegado. Finalmente pegamos las distintas piezas para formar la pieza diseñada inicialmente. Para ello utilizamos el pegamento Nural 21 que está especialmente diseñado para uniones metálicas y se conoce como "soldadura en frío". Las claves para obtener un buen resultado es conseguir que las piezas queden bien sujetas mientras el pegamento se va secando. Para ello lo mejor es utilizar pinzas. El tiempo de endurecimiento está en torno a las 12 horas. Por lo tanto es conveniente realizar un pegado temprano a la mañana y otro a la noche. Hay que añadir que el pegamento Nural 21 no solo sirve para pegar sino que también puede reparar grietas o cortes indeseados y también llenar huecos.

Figura 26.6: Proceso de pegado

- Aislamiento. Es una buena idea recubrir con cinta aislante el interior de las partes que puedan ser susceptibles de estar en contacto con la electrónica. De esta forma nos evitamos posibles cortocircuitos que hagan que nuestro robot no funcione correctamente o que aumenten el consumo de la batería.

26.2. Decoración de la carcasa

Tras seguir los pasos anteriores ya tenemos el esqueleto del robot operativo. Pero también hay que cuidar la apariencia externa ya que es lo primero que juzgará cualquier persona que lo vea.

Para conseguir un buen acabado vamos a imprimir una cubierta realizando los siguientes pasos:

- Utilizando un software como Autocad dibujamos la cubierta utilizando las medidas de la carcasa.
- Luego importamos la cubierta a un programa como Photoshop y le añadimos los detalles que deseemos. En nuestro caso hemos decidido que el robot luzca los colores del logotipo de la Upna:

blanco, negro y rojo. También dado que el sumo es un deporte japonés hemos colocado en el frontal una mascara de samurai. También hemos incluido el nombre del robot y el logotipo de la Upna.

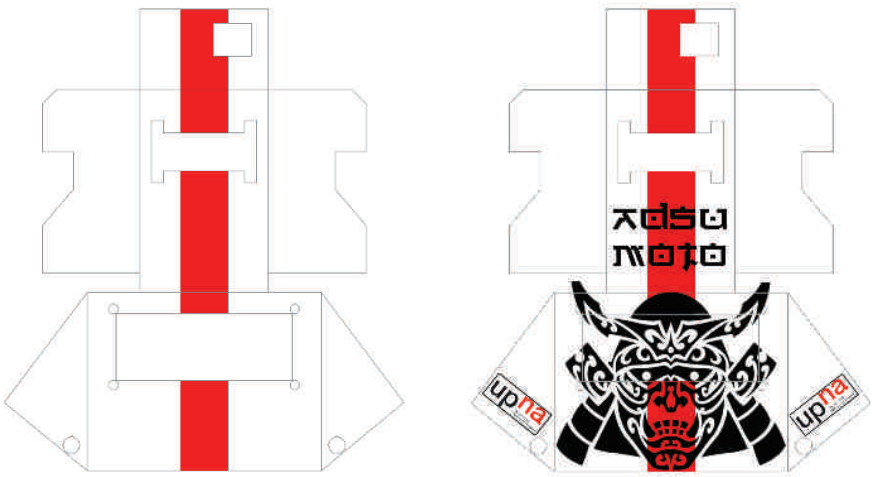


Figura 26.7: A la izquierda el diseño de Autocad y a la derecha los detalles añadidos en Photoshop

- Imprimimos la cubierta a una hoja de papel y lo recortamos.
- Pegamos la cubierta a la carcasa utilizando abundante pegamento.
- Imprimamos la cubierta con un spray de barniz brillante para conseguir un acabado realmente profesional. Realizamos varias capas delgadas en vez de una gorda. Este paso es fundamental porque al utilizar el barniz ya no da la impresión de ser papel, sino que parece plástico.

A continuación se puede ver como mejora el aspecto del robot al añadirle la cubierta. Sin duda es mucho más profesional.

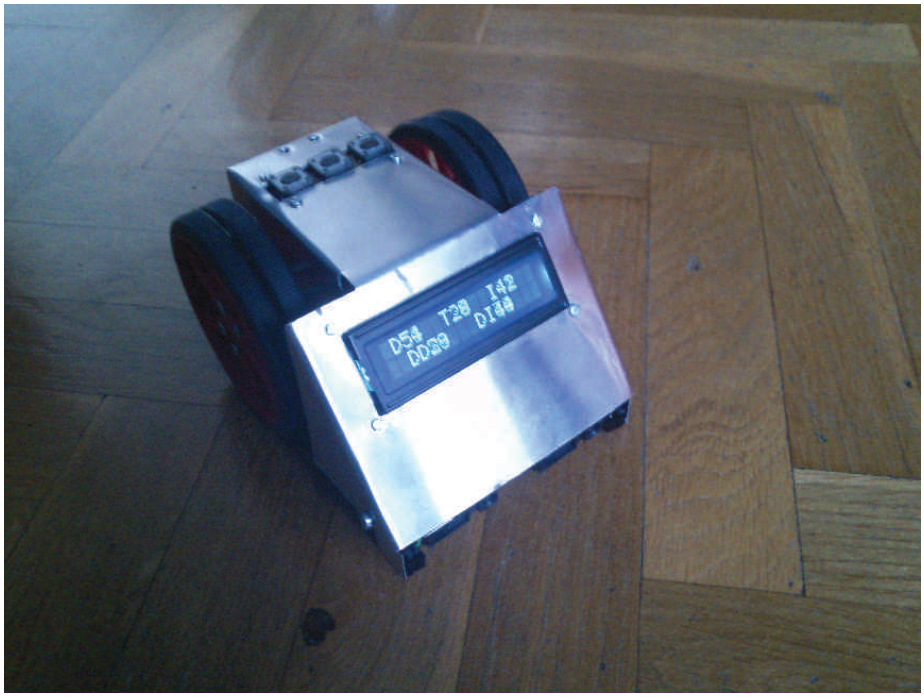


Figura 26.8: El robot con la carcasa sin decorar

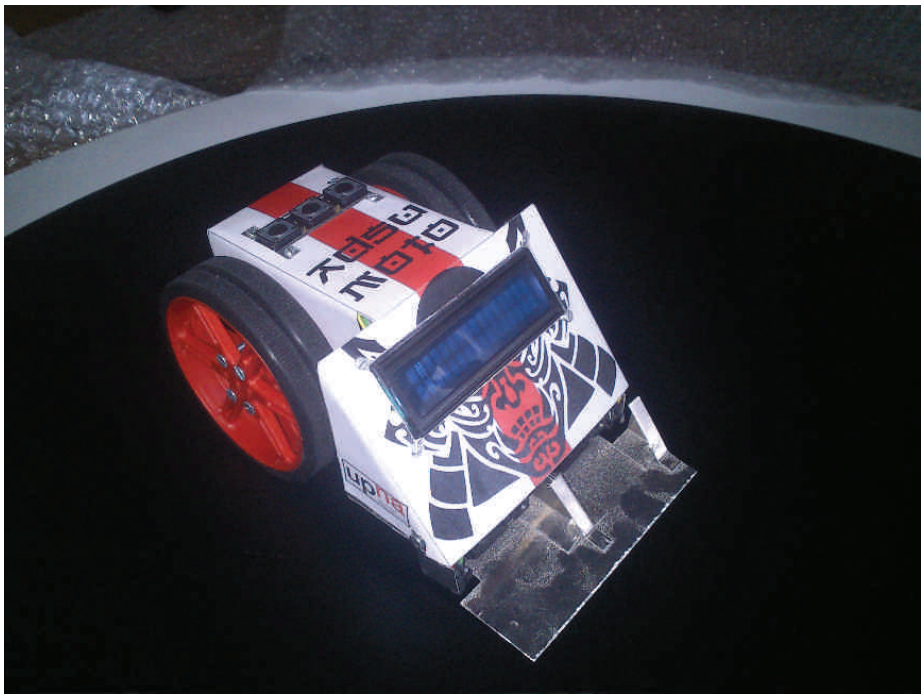


Figura 26.9: Adsumoto con la cubierta ya añadida

Capítulo 27

Electrónica

27.1. Placa base

A la placa base llegan los cables de los sensores y alberga también los controladores de los motores. Su construcción se realiza siguiendo estos pasos:

- En primer lugar hay que diseñar la placa: elegir la ubicación de los componentes, conexiones entre ellos... Para ello lo más sencillo es utilizar algún software de diseño de PCB. En nuestro caso hemos utilizado Fritzing[1] ya que es libre y además tiene varios modos de visionado en los que se incluye una vista esquemática y una vista real muy útil para diseñar la placa. Este programa es muy recomendable si vamos a trabajar con Arduino.
- Una vez hayamos diseñado la placa lo primero que tenemos que hacer es recortar la placa de prototipado para conseguir el tamaño deseado. Para ello utilizaremos una sierra de mano. No se necesita una sierra de dientes grandes ya que se corta con mucha facilidad.

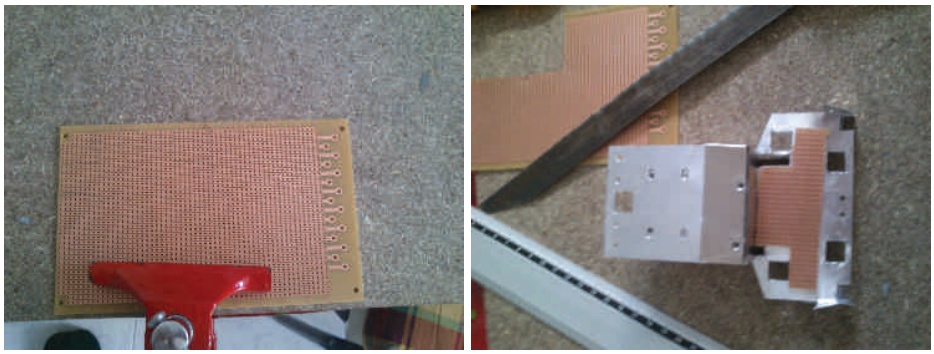


Figura 27.1: Recortar la placa de prototipado

- Ahora tenemos que cortar las calles de la placa donde no queramos que exista conexión. Una buena forma es usar un lápiz para marcar aquellas pistas que hay que separar y luego utilizar un cúter para cortar la unión metálica. Es obligatorio comprobar luego que realmente se ha conseguido la separación. Para ello utilizamos un polímetro con la función de continuidad que realiza un pitido cuando dos puntos están conectados.

[1] http://fritzing.org/

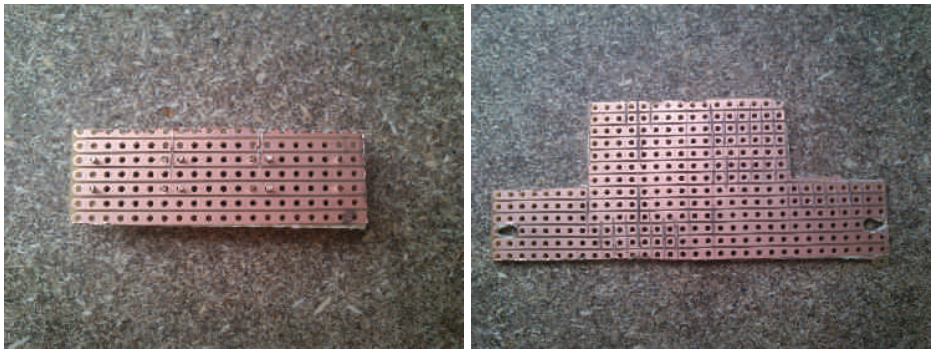


Figura 27.2: Cortar las pistas que no necesitan estar unidas

- Soldar. Ahora vamos colocando los componentes en la placa según su tamaño de menor a mayor y los vamos soldando. Soldamos también las pistas que queremos que estén conectadas. Es buena idea sacar una foto a la placa y colorearla para que nos sirva de guía.

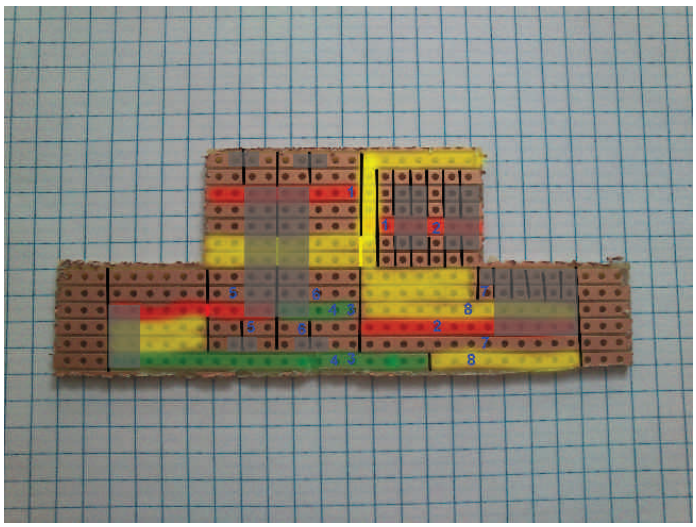


Figura 27.3: Fotografía de la placa coloreada

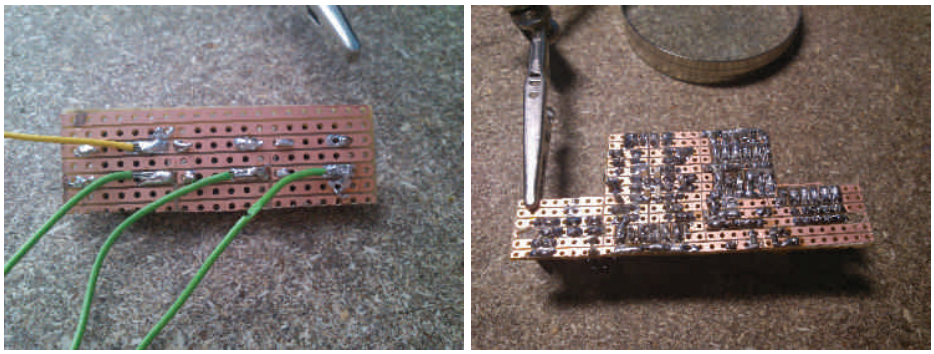


Figura 27.4: Soldado

- Comprobación de las conexiones. Comprobamos que todo ha sido realizado según el diseño utilizando nuevamente el polímetro. Comprobamos que no hay uniones indeseadas y que las deseadas se han realizado de forma correcta. Reparar todos los fallos.
- Finalmente proteger la soldadura pegando cinta aislante por encima. Además como luego se colocará la placa en el chasis metálico con la cinta aislante evitamos que se creen contactos indeseados.

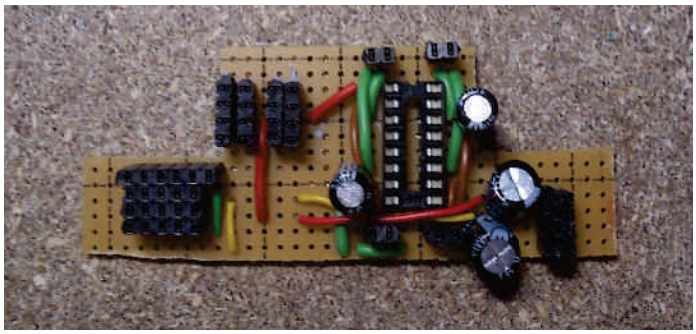


Figura 27.5: Vista final de la placa con la parte inferior protegida por la cinta aislante

Capítulo 28

Variaciones sobre el Diseño Inicial

En este capítulo se comentan las variaciones que hubo que introducir en el prototipo para asegurar su correcto funcionamiento.

28.1. Rueda loca

El reglamento de sumo robótico establece que los robots no pueden dañar el dohyo sobre el que se realiza el combate. Provocar desperfectos al área de combate se considera una violación de juego que implica la pérdida del combate. Por ello para que el robot sea competitivo debe evitar a toda costa el dañar el dohyo.

Al comenzar nuestro robot en posición vertical y tener que caer para colocarse horizontalmente se produce un impacto contra el suelo. Normalmente ese impacto es amortiguado por la rampa delantera que tiene una cierta flexibilidad que permite absorber la energía del golpe. Pero a veces el chasis llegaba a tocar el suelo rayandolo un poco. Por eso se decidió colocar una rueda loca a una altura un poco más baja que el chasis para evitar que el chasis pudiese llegar a tocar tierra.

En condiciones normales la rueda loca no toca el suelo por lo que no altera la distribución de fuerzas en contacto con el suelo. Solo en la puesta en horizontal cuando la rampa flexiona levemente llega a tocar el suelo y frena completamente la caída. Al ser esférica no produce ningún daño al área de juego.

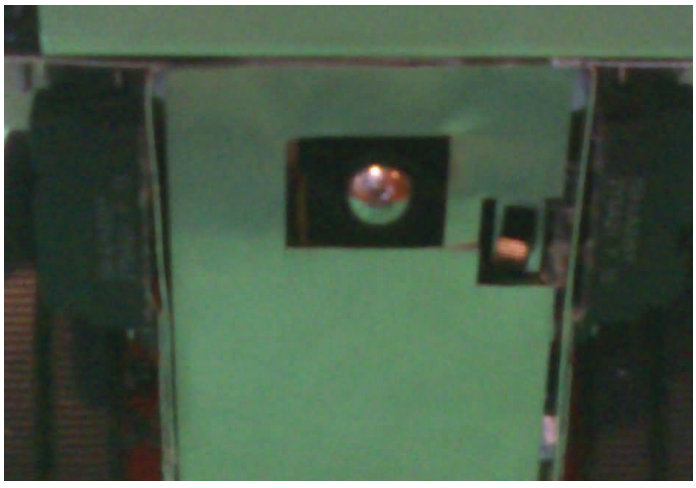


Figura 28.1: Fotografía de la rueda loca colocada en los bajos del robot

28.2. Interruptor para la pantalla LCD

Desde el principio la pantalla presentó problemas de sincronización con la placa Arduino. En ocasiones representa símbolos extraños en lugar de las palabras que le envía el microcontrolador. Se ha intentado buscar la causa del problema pero aparece y desaparece de forma completamente aleatoria. Sin ningún motivo aparente a veces funciona bien y otras funciona mal.

Se ha investigado en Internet en los diversos foros existentes sobre Arduino pero no se ha hallado nada parecido. Lo más probable es que haya algún fallo en la pantalla, ya que no fue comprada sino que se utilizó una que se encontraba en el Laboratorio de Robótica.

Como el problema parece ser de sincronización se colocó un interruptor en el cable de comunicación entre la pantalla y Arduino. De esta forma siempre se puede conseguir la sincronización jugando con el interruptor de alimentación y el de comunicación.

28.3. Plomos

Tras completar la construcción del robot el peso era de 400g. Pero las reglas de minisumo permiten un peso de hasta 500g y como se ha visto anteriormente para que el robot sea competitivo tiene que pesar lo máximo posible.

Por ello se procedió a lastrar el robot hasta alcanzar los 500g. La opción elegida fue utilizar plomos de pesca y distribuirlos en el interior del robot pegándolos con pegamento termofusible. El plomo es uno de los elementos más densos siendo su densidad más de 11 veces superior a la del agua.

Figura 28.2: Plomos utilizados para aumentar el peso del robot

28.4. Simplificación de formas

Como se estudia en la Tecnología de Fabricación, a menudo piezas de muy distintas formas pueden cumplir el mismo cometido. Por ello cuando fue necesario realizar la construcción de un segundo chasis debido a fallos irreparables en el primero se procedió a realizar una simplificación de las formas de las piezas para facilitar su fabricación.

Se vio que la realización de agujeros interiores era mucho más costosa que realizar un agujero exterior. También se minimizó el número de cortes necesario para hacer cada pieza. La construcción del primer chasis tomó 15 horas, la del segundo redujo el tiempo a la mitad gracias a las mejoras introducidas.

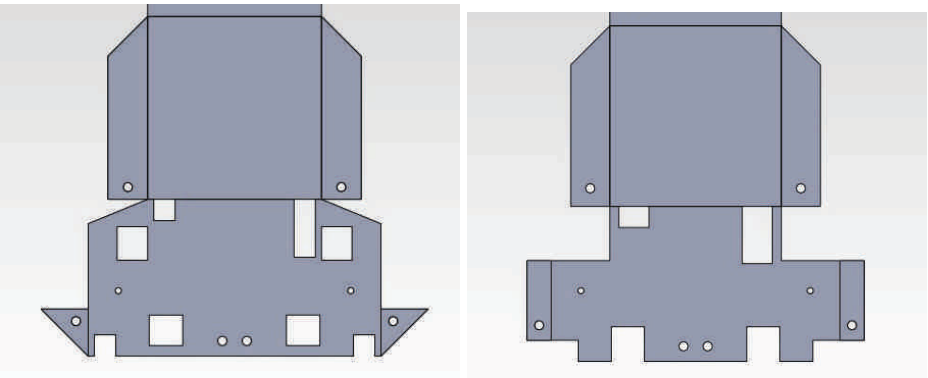


Figura 28.3: A la izquierda el diseño inicial y a la derecha el diseño simplificado para agilizar la construcción

28.5. Altavoz

Se añadió un pequeño altavoz en la la base para marcar los 5 segundos antes de que comience el combate. También puede ser utilizado para la depuración del programa haciendo que emita un ruido distinto según cada situación.

28.6. Mecanismo de regulación del centro de gravedad

Finalmente se desechó la idea de colocar un mecanismo que permitiese modificar el centro de gravedad debido a la complejidad de su construcción. Sin embargo la idea en si es muy buena y debería ser explorada en futuros diseños.

Parte V

Puesta a Punto

Capítulo 29

Sistema Operativo

Para poder controlar el robot, cambiar su modo de comportamiento, ejecutar distintos programas... se va a crear una especie de sistema operativo sobre el que construiremos todo el resto de la programación.

Todo el código fuente del programa puede encontrarse en la parte final del trabajo en el Anexo.

29.1. Arduino

Al programar la placa de Arduino tenemos dos partes bien diferenciadas: una secuencia de inicio y un bucle que se repite indefinidamente.

29.1.1. Secuencia de Inicio

En la secuencia de inicio configuraremos los pines de Arduino como entradas o salidas. También cargaremos los valores de algunas variables. En la programación esta parte va en *setup*.

29.1.2. Bucle

El funcionamiento de Arduino consiste en repetir una y otra vez el mismo programa. No se queda esperando a que pulsemos un botón sino que está constantemente comprobando si el botón está pulsado o no. Tendremos que tener en cuenta esto a la hora de diseñar el sistema operativo. En la programación esta parte va en *loop*.

29.2. Estructura del Sistema

El sistema operativo se ha construido utilizando una serie de variables de estado que dirigen el flujo del programa. Las principales variables de estado son las siguientes:

- Modo: esta variable determina si el robot está funcionando en el modo de combate o en el de menú. Por lo tanto solo puede tomar dos valores.
- Estrategia: esta variable determina de que forma se va a comportar el robot en el modo combate. Algunas de las estrategias son por ejemplo: Fluida, Suicida, Cobarde, Vueltas...
- Nivel: esta variable determina si estamos eligiendo un subprograma en el menú o si lo estamos ejecutando. Lógicamente tiene solo dos valores.

- Menú: esta variable determina nuestra posición en el menú. El menú tiene bastantes programas para ejecutar como: ver el voltaje de la batería, comprobar los valores de los sensores, cambiar la velocidad del robot...

Mediante las variables anteriores podemos conseguir que el sistema operativo funcione como muestra el siguiente esquema:

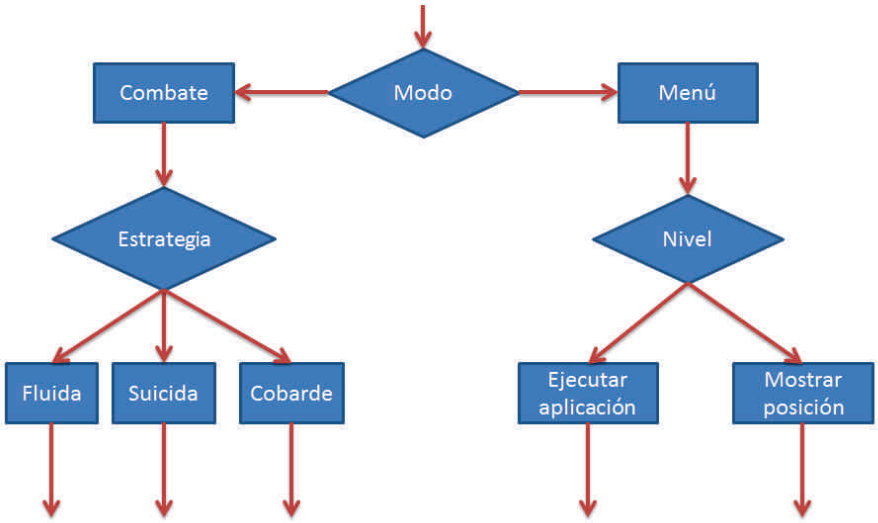


Figura 29.1: Esquema de funcionamiento del sistema operativo

De esta forma podemos añadir nuevas estrategias y nuevas aplicaciones fácilmente. Tan solo tenemos que añadir un nuevo valor a las variables de estado correspondientes. Por lo tanto el sistema operativo es la base sobre la que edificamos todo lo demás.

Por poner un ejemplo si tuviésemos seleccionado el modo de combate y la estrategia suicida tan solo se ejecutarían las partes del programa coloreadas de verde.

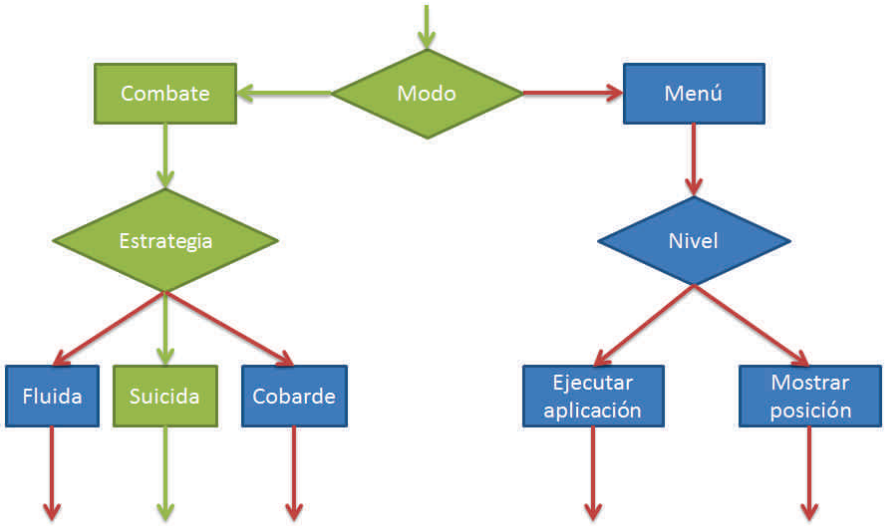


Figura 29.2: Ejemplo para modo de combate y estrategia suicida. Las partes coloreadas en verde son las que se ejecutan.

29.3. Interfaz de Usuario

El robot tiene una pantalla LCD de 16x2 caracteres y tres botones mediante los cuales podemos configurarlo.

La pantalla nos muestra nuestra posición en el menú y también nos permite ver por ejemplo los valores que están midiendo los sensores cuando activamos la aplicación correspondiente.

Los tres botones tienen las funciones de Salir, Intro y Siguiente. Con ellos podemos cambiar los valores de las variables de estado y de esa forma dirigimos el flujo del programa según nuestra voluntad.

Figura 29.3: Fotografia de la interfaz de usuario

Capítulo 30

Sensores de Distancia

Los sensores de distancia empleados en el robot han sido los GP2Y0A21YK0F como ya se ha comentado anteriormente. Estos sensores son analógicos, se basan en luz infrarroja y tienen un rango de medida de 10-80 cm lo que es ideal para nuestro robot.

A la hora de diseñar el robot no disponíamos de estos sensores en el laboratorio por lo que la única información fiable que teníamos era la hoja de características. Por eso cuando los montamos en el robot nos encontramos con algunas sorpresas que ahora vamos a relatar para ayudar a resolver problemas futuros.

30.1. Asimetría

Uno de los mayores inconvenientes de estos sensores es que no son simétricos. Las lecturas de un objeto situado a la derecha son distintas de las de un objeto situado a la izquierda. En la siguiente gráfica se puede ver una representación aproximada de los valores que proporciona el sensor.

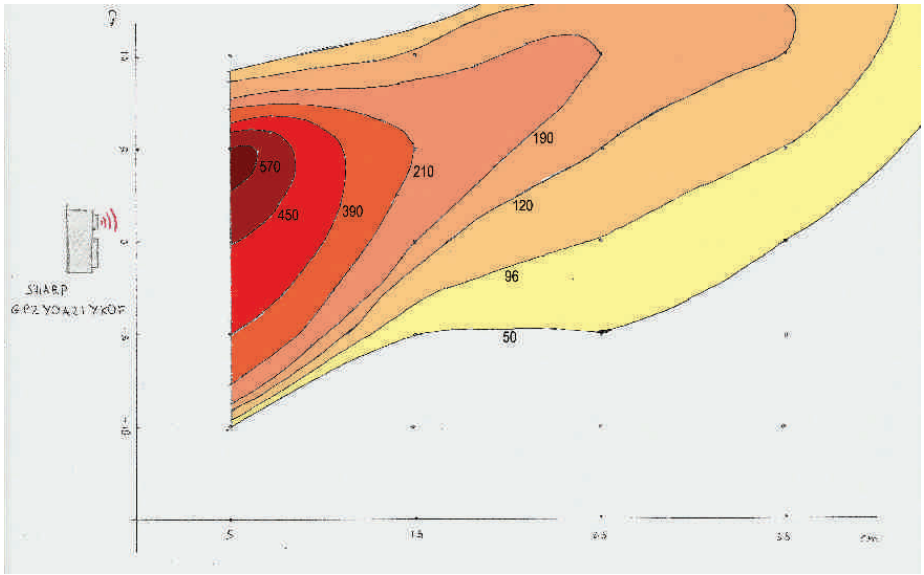


Figura 30.1: Representación gráfica de la salida del sensor GP2Y0A21YK0F

Esta asimetría ha hecho que programar al robot para que rastree a su enemigo con los dos sensores

delanteros sea bastante complicada. Pero tiene una solución muy sencilla: colocar un sensor boca abajo. De esta forma las lecturas de los sensores delanteros si serán simétricas y la programación se simplificará mucho. En la siguientes gráficas se puede ver como hay que colocar los sensores para obtener un resultado simétrico en su conjunto, podríamos obtener otra configuración simétrica girando los dos sensores a la vez. En nuestro robot no se pudo hacer este cambio pero en futuros deberá hacerse así.

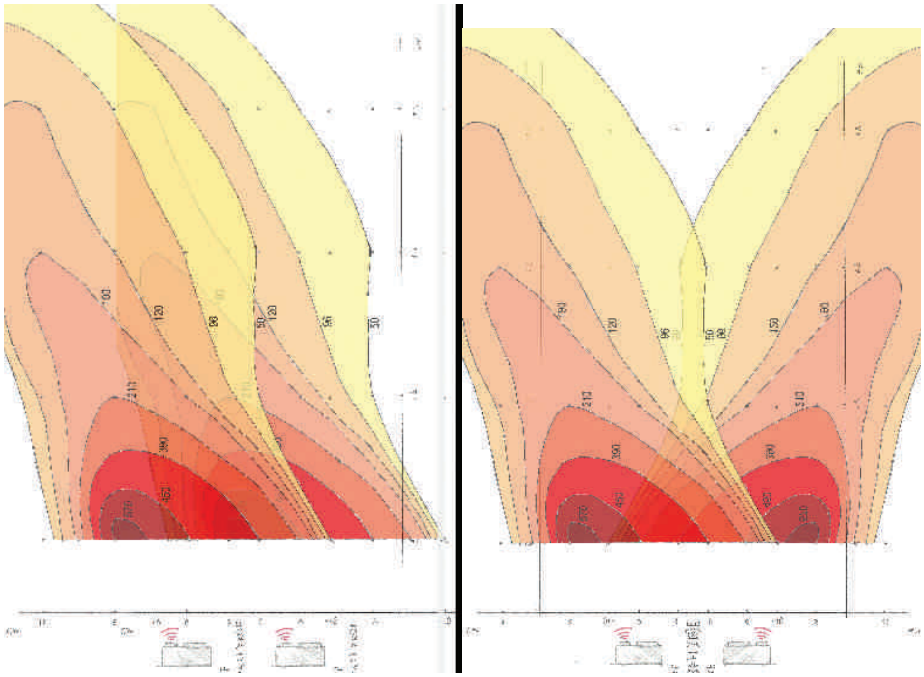


Figura 30.2: A la izquierda una mala colocación de los sensores de distancia y a la derecha la colocación correcta

30.2. Ruido

Los sensores de distancia utilizados han sido diseñados para funcionar en inodoros automáticos. Por lo tanto las condiciones en las que los estamos utilizando difieren mucho de aquellas para las que fueron diseñados. Esto hace que el ruido que obtenemos en la medida del sensor sea considerable y es necesario hacer un tratamiento por software para evitar que nuestro robot se comporte inadecuadamente al recibir información incorrecta de los sensores.

La primera medida es la de colocar un umbral de detección. Toda lectura por debajo de ese umbral será ignorada. Para determinar el umbral hay que examinar cuidadosamente las lecturas del sensor. En nuestro caso situábamos el umbral en 200, en una escala en la que 1023 son 5V, lo que sería aproximadamente 1V.

La segunda medida es no considerar que el sensor ha detectado algo hasta que tenemos varias lecturas seguidas por encima del umbral. De esta forma eliminamos casi completamente el ruido porque la frecuencia del ruido suele ser alta y por lo tanto el tiempo en el que superará el umbral será corto.

El siguiente esquema resume el tratamiento que se le ha dado al ruido de los sensores GP2Y0A21YK0F.

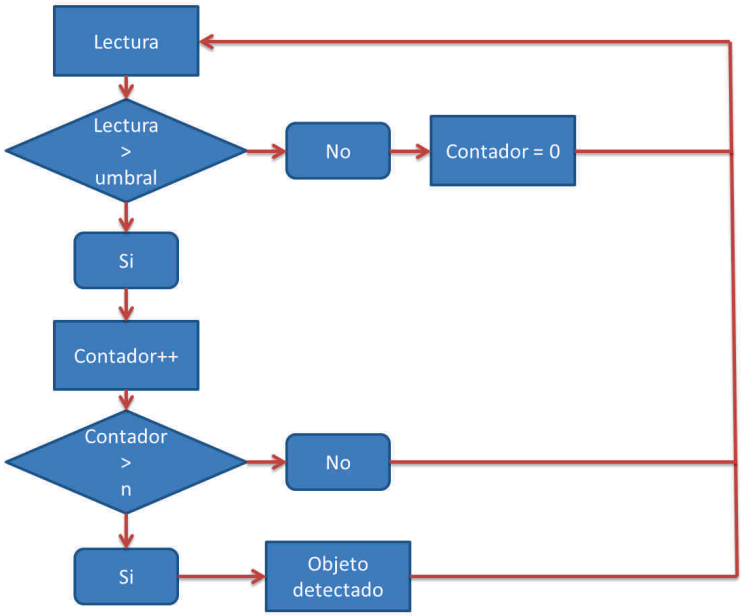


Figura 30.3: Esquema de tratamiento del ruido por software

Capítulo 31

Transiciones entre movimientos

A la hora de probar las diferentes estrategias se vio que había momentos en los que el robot se encabritaba levantando la parte delantera. Esto lo dejaba en una posición muy vulnerable ya que si en ese momento nos atacaba el contrario se colaría fácilmente por debajo y nos empujaría sin esfuerzo hacia el exterior del dohyo. Por eso se realizó un estudio que desembocó en la estrategia Fluida, llamada así porque las transiciones entre los movimientos se realizan fluidamente y sin que se levante apenas la rampa delantera.

31.1. Situaciones conflictivas

En primer lugar se hizo un estudio exhaustivo para ver en que transiciones se producía el encabritamiento. Las principales situaciones en las que se encabritaba eran:

- Moviéndose hacia atrás → Moverse hacia adelante
- Parado → Moverse hacia adelante
- Girando hacia adelante → Moverse hacia adelante

31.2. Resolución del problema

Después de realizar varias pruebas se llegó a una solución que consistía en que para hacer una transición fluida entre movimientos se debería pasar por una serie de estados intermedios como describe el siguiente esquema.

Figura 31.1: Estados de transición

El tiempo mínimo de transición que se mostró exitoso fue de 300ms, dedicando 100ms a cada una de las tres partes de la transición. Cuando se mira al robot no se aprecia que que vaya más lento y que frene.

Capítulo 32

Aplicaciones

En este capítulo se describen las aplicaciones que pueden ser accedidas desde el menú. Puede verse una demostración de las aplicaciones en el siguiente vídeo[1].

32.1. Aplicaciones de Seguridad

Batería Esta no es una aplicación que se pueda acceder mediante el menú, pero cada vez que se ejecuta el bucle del programa comprueba que el estado de la batería es el correcto. Cuando es necesario recargarla muestra una señal de alarma por pantalla.

32.2. Aplicaciones de Configuración

Este grupo de aplicaciones sirve para modificar las variables que rigen el comportamiento del motor.

Estrategia Está aplicación nos permite seleccionar entre las distintas estrategias que tiene almacenadas el robot como: Fluida, Suicida...

Velocidad Está aplicación nos permite cambiar la velocidad entre unos valores fijados: 128, 160, 192, 224, 255. Cuanto mayor es el número mayor es la velocidad.

Figura 32.1: Vista de la aplicación de Velocidad

Sensor Trasero Con esta aplicación podemos activar o desactivar el uso del sensor trasero. Cuando el sensor trasero está activo y detecta algo lo que el robot piensa es que el enemigo está detrás y se gira para enfrentarlo. Pero puede suceder que el sensor trasero se active porque el robot enemigo eleve

[1] http://www.youtube.com/watch?v=Rer1351Ze9A

nuestra parte frontal haciendo que el sensor trasero detecte el suelo. En esa situación trataría de girar y esto haría que fuese muy fácil empujarle hacia atrás. Por eso es útil tener la opción de desactivarlo frente a ciertos oponentes.

Luz Esta aplicación permite activar o desactivar la luz de la pantalla LCD. De esta forma se puede ahorrar batería.

Reset A veces puede ser útil resetear la placa de Arduino y eso es lo que hace está aplicación.

32.3. Aplicaciones de Prueba

Prueba Estrategia Está aplicación hace que el robot se comporte como si estuviera en modo Combate durante 2 segundos. De esta forma podemos probar que tal funcionan las distintas estrategias en determinadas situaciones y podemos detectar los fallos y optimizarlo cómodamente.

Leer Sharp Está aplicación nos muestra por la pantalla las lecturas de los sensores de distancias. Nos permite comprobar que todo va bien antes del combate.

Figura 32.2: Vista de la aplicación de Leer Sharp

Radar Está aplicación utiliza los dos sensores delanteros para mostrar en la pantalla donde está el objeto que tiene delante.

Figura 32.3: Vista de la aplicación de Radar

CNY70 Está aplicación nos muestra por la pantalla las lecturas de los sensores de borde. Muestran un 0 si la superficie debajo es negra y un 1 si es blanca.

Figura 32.4: Vista de la aplicación de CNY70

Batería Nos muestra el voltaje de la batería y de esta forma podemos inferir cuanto tiempo más disponemos para usar el robot. Cuando la batería está completamente cargada puede llegar a 8.3V y es necesario recargarla cuando llega a unos 7.3V.

Figura 32.5: Vista de la aplicación de Batería

Prueba Motor Esta aplicación realiza una secuencia con los motores que nos permite comprobar si hay fallos. Fue especialmente útil cuando se empezaron a desgastar los agujeros para el eje de las ruedas de plástico.

Prueba Huir Esta aplicación sirve para ver el comportamiento del robot cuando llega al borde del dohyo.

Caída Forzada Esta aplicación sirve para comprobar que el paso de posición vertical a horizontal funciona correctamente.

Banco de pruebas Esta aplicación sirve para meter código nuevo y ver como funciona. Si es útil después se saca del banco de pruebas y se crea una nueva aplicación.

32.4. Aplicaciones Extra

Coche Fantástico Esta aplicación muestra por pantalla la animación típica de la serie de televisión El Coche Fantástico

Sonido Con esta aplicación hacemos que el altavoz reproduzca sonido.

Capítulo 33

Duración de la batería

Anteriormente en la parte de Elección de Componentes se había estimado que el consumo del robot en parado seria de unos 450 mA y con los motores funcionando al máximo podria llegar casi a los 2000 mA.

Teniendo en cuenta que nuestras baterías tienen una capacidad de 1000 mAh fácilmente se calcula que la duración de la batería en parado sería de unas 2 horas y con el robot funcionando a máxima potencia duraría media hora.

Pero es aconsejable hacer una prueba real para no encontrarnos con sorpresas. Por eso realizamos una prueba de descarga de la batería con el robot parado y mostramos los resultados a continuación.

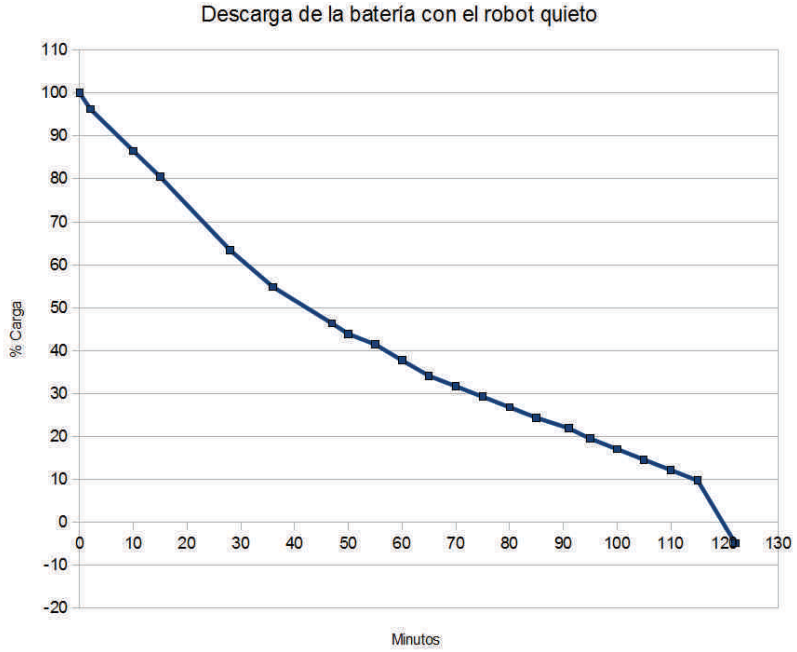


Figura 33.1: Gráfica de descarga de la batería

Como se ve la duración fue aproximadamente de dos horas tal y como se había calculado.

No se realizó una prueba de funcionamiento continuo del robot porque ese no es su modo de funcionamiento normal. En un campeonato hay tiempo de parada entre cada asalto y cada combate. Por eso para evitar posibles averías por sobrecalentamiento en los motores o en la electrónica por

funcionar durante media hora al máximo se decidió no hacer esa prueba.

Lo que se puede añadir es que en la fase de programación y puesta a punto el robot aguantaba toda la mañana o la tarde con una única batería. Obviamente no estaba todo el rato funcionando y se apagaba mientras no se usaba. También que durante el campeonato de Robolid también aguantó perfectamente con una batería y no hubo que recurrir a la segunda.

En conclusión: la autonomía del robot es más que suficiente para el propósito con el que fue diseñado.

Parte VI

Programación

Capítulo 34

Estrategia Fluida

Esta es la estrategia estándar que utiliza Adsumoto. Su nombre viene de que es una evolución de una estrategia anterior en la que se incorporan las transiciones de movimientos para que los cambios sean fluidos. Puede verse una demostración en el siguiente vídeo[1].

34.1. Funcionamiento básico

El esquema que muestra el funcionamiento es el siguiente:

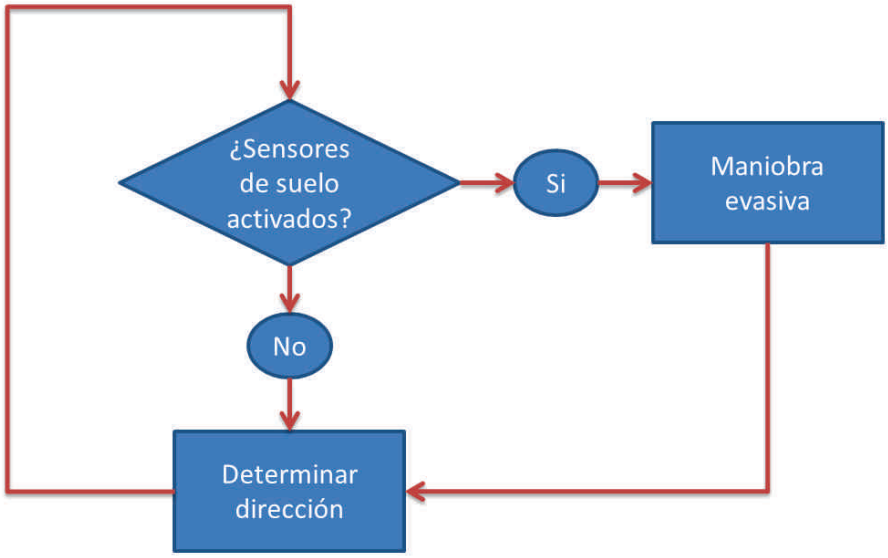


Figura 34.1: Esquema de la Estrategia Fluida

Como se ve en primer lugar comprobamos que no estamos en el borde del dohyo y si es así procedemos a detectar donde está el adversario y movernos hacia él. Esto hace que prioricemos el mantener el robot en el área de juego sobre encontrar al contrario. Lo primero es sobrevivir y después ya buscaremos al contrario.

[1] http://www.youtube.com/watch?v=ClTWrSS48wY

34.2. Sensores de Suelo

Esta función examina los sensores de suelo y envía un mensaje a las siguientes funciones indicando los sensores que han sido activados.

El orden en el que se examinan los sensores es muy importante como vamos a explicar a continuación. En primer lugar comprobamos si tenemos activados los dos sensores derechos o izquierdos. Esta es la situación más conflictiva ya que si diéramos prioridad a los sensores delanteros y traseros

34.3. Maniobra Evasiva

Esta función recibe la información sobre que sensores de suelo han detectado el borde y actúa en consecuencia. El orden en el que se reacciona frente a los sensores es importante como vamos a ver a continuación.

34.3.1. Sensores Laterales Activados

Si por ejemplo tenemos el sensor delantero izquierdo y el sensor trasero izquierdo activado significa que el borde del dohyo se encuentra a la izquierda del robot. Para evitar salirnos debemos girar hacia la derecha, ya que si seguimos hacia delante o hacia detrás nos saldremos del dohyo.

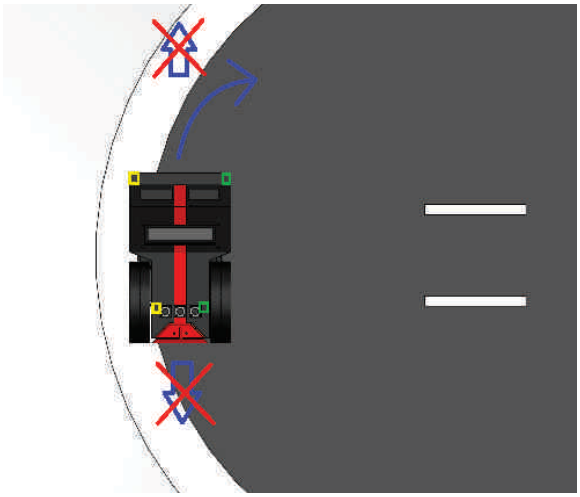


Figura 34.2: Sensores laterales activados

34.3.2. Sensores Delanteros Activados

Si tenemos el sensor delantero izquierdo o delantero derecho activado significa que el borde se encuentra en frente del robot. Por lo tanto daremos marcha atrás para evitar salirnos y seguidamente daremos la vuelta para encarar hacia el centro del dohyo.

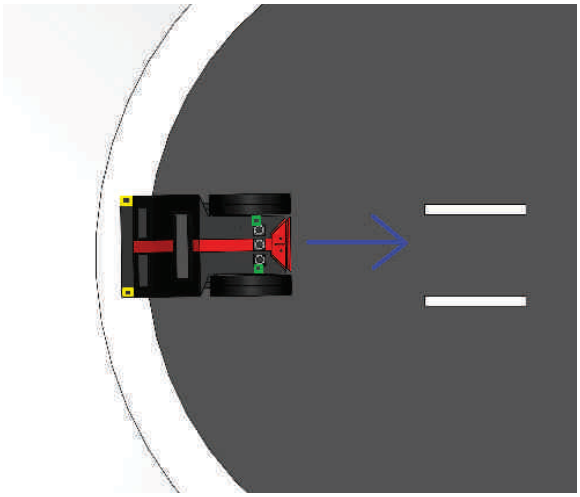


Figura 34.3: Sensores delanteros activados

34.3.3. Sensores Traseros Activados

Si tenemos el sensor trasero izquierdo o trasero derecho activado significa que el borde se encuentra detrás del robot. Por lo que tendremos que movernos hacia adelante para evitar que el robot se caiga.

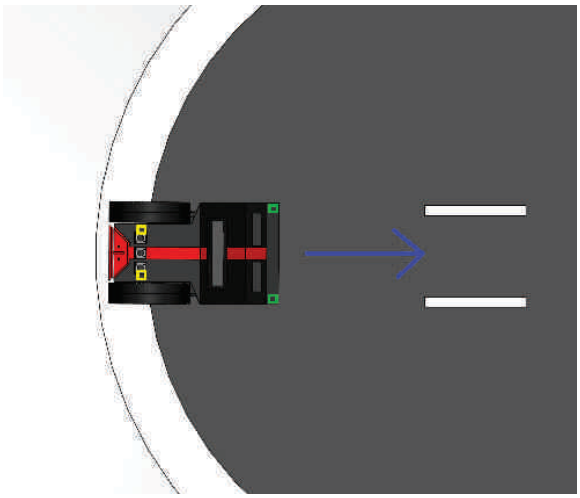


Figura 34.4: Sensores traseros activados

34.4. Determinar Dirección

Si llegamos a esta función significa que el robot no está en peligro de caer por el borde y lo que debemos hacer es rastrear al adversario con los sensores de distancia para poder ir a por él. Por lo tanto lo primero que hace la función es leer los sensores de distancia y luego actúa en consecuencia.

34.4.1. Enemigo Delante

Si el enemigo está delante tenemos dos sensores de distancia para poder apuntar y dirigir el robot hacia él. Hay una subfunción que toma los valores leídos a los sensores de distancia delanteros y da las ordenes necesarias a los motores para orientar correctamente el robot hacia el enemigo. Debido a la asimetría de los sensores ya comentada en un capítulo anterior la función empleada para poder apuntar es bastante compleja y fue necesario realizar numerosas pruebas y modificaciones hasta alcanzar un resultado satisfactorio.

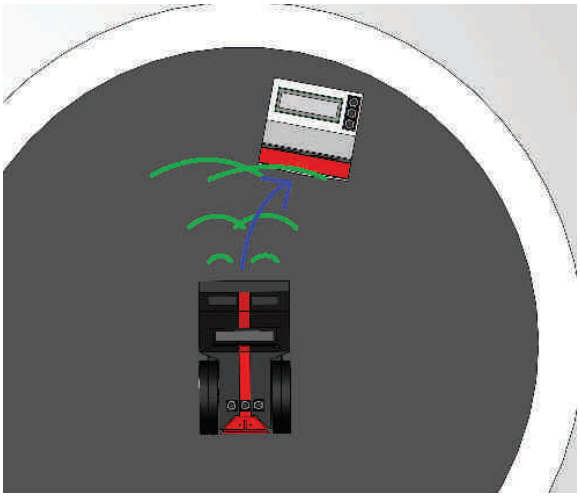


Figura 34.5: Enemigo delante

34.4.2. Enemigo en un Lateral

En cada lateral del robot tenemos un sensor de distancia. Si por ejemplo se activa el sensor derecho sabremos que el enemigo se encuentra a la derecha de nuestro robot y deberemos girar a la derecha para poder encararle frontalmente.

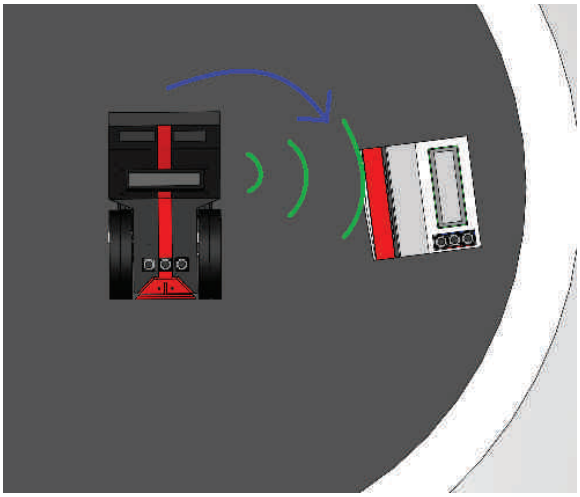


Figura 34.6: Enemigo en un lateral

34.4.3. Enemigo Detrás

También tenemos un sensor trasero de distancia para saber cuando el enemigo ha conseguido colocarse en nuestra retaguardia. Deberemos hacer que el robot gire rápidamente para dejar de ser vulnerables a un ataque trasero.

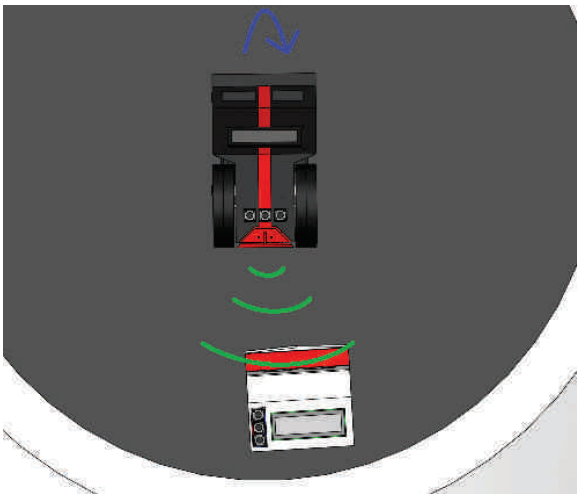


Figura 34.7: Enemigo detrás

Capítulo 35

Estrategia Suicida

Esta es una estrategia hermana de la anterior Fluida. La única variación que presenta es como reacciona cuando detecta que el borde del dohyo está delante.

Existen robots de sumo que despliegan estructuras planas blancas a ras de suelo. De esta forma cuando el robot adversario se sitúa encima de esas estructuras piensa que está en el borde y retrocede. Cuando esto ocurre el robot que ha desplegado las estructuras tiene todas las de ganar ya que ni siquiera tiene que empujar al enemigo, este se mueve solo hacia atrás.

Para evitar que nos suceda esto hemos creado la estrategia Suicida. Cuando el robot detecta que el borde del dohyo está delante no se moverá hacia atrás automáticamente. Primero leerá los sensores de distancia delanteros. Si estos no detectan nada significará que realmente estamos en el borde del dohyo y deberemos dar marcha atrás. Pero si detectan algo delante será el robot que trata de engañarnos y seguiremos hacia delante para empujarle.

Esta estrategia se utilizará solo ante robots adversarios que puedan llegar a engañarnos ya que al tener que leer los sensores de distancia necesitamos un mayor tiempo de procesamiento y por lo tanto la frecuencia a la que se ejecuta el programa es menor.

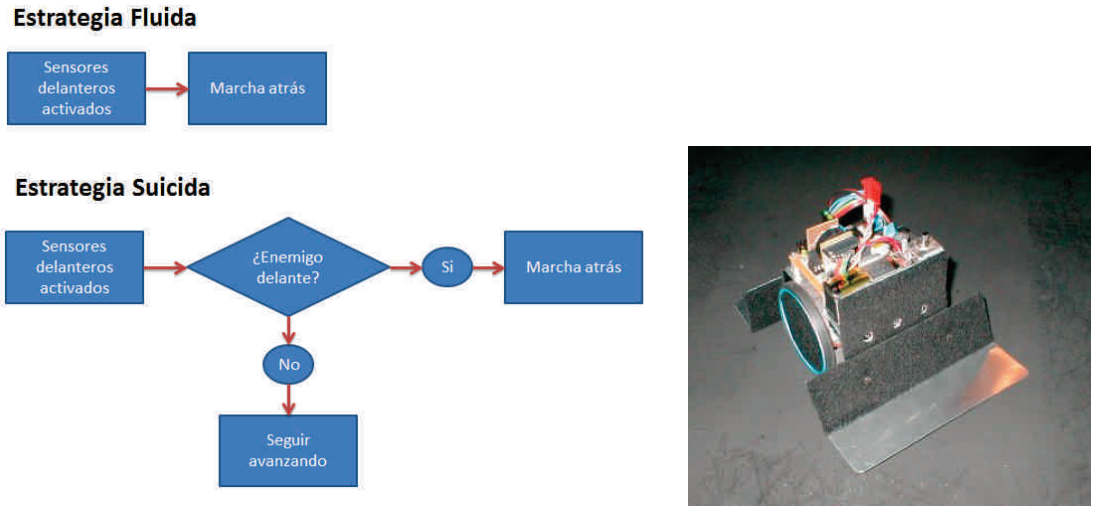


Figura 35.1: Comparativa entre las dos estrategias y robot con rampa horizontal brillante

Capítulo 36

Estrategia Vueltas

Esta estrategia consiste en utilizar el robot como un siguelíneas. La línea que debe seguir es la separación entre la zona negra y blanca del dohyo.

Por lo tanto lo que hace el robot es buscar el borde del dohyo y después dar vueltas alrededor del terreno de juego. Si nuestro robot es más rápido que el contrario no podrán alcanzarnos mientras estemos girando. Lógicamente no podemos estar todo el combate rehuyendo el cuerpo a cuerpo, llegado un momento deberíamos cambiar a una estrategia que busque chocar al contrario.

Lo mejor sería comenzar dando vueltas al ring y cuando detectásemos al contrario dejar de dar vueltas y tratar de movernos para atacarle por un lateral o por la retaguardia.

Puede verse la estrategia Vueltas en el siguiente vídeo[1].

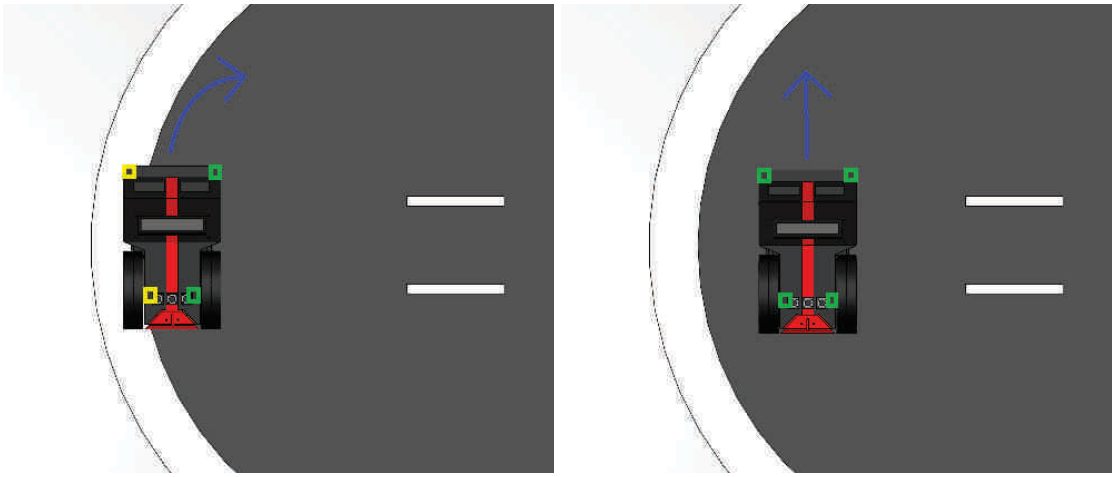


Figura 36.1: Esquemas del funcionamiento de la estrategia

[1] http://www.youtube.com/watch?v=4mYK0H5tTQM

Capítulo 37

Estrategia Cobarde

Esta no es una estrategia de combate propiamente dicha sino un modo de funcionamiento para demostrar las posibilidades del robot.

En este modo el robot utiliza los sensores de distancia para escanear sus alrededores en busca de amenazas. Si detecta algo en las proximidades huirá pero siempre tratando de no perder de vista a su enemigo. Retrocederá como un animal asustado pero plantando cara por si tuviese que atacar.

Figura 37.1: El robot retrocede sin perder la cara al enemigo

Este modo de funcionamiento podría evolucionar en una estrategia de combate si además usáramos los sensores de suelo para evitar salirnos del dohyo. En ese caso sería una estrategia que consistiría en evitar al contrario a toda costa buscando que se le gaste la batería o bien que se salga del dohyo.

Puede verse la estrategia en el siguiente vídeo[1].

[1] http://www.youtube.com/watch?v=x7Dk51UHPv8

Capítulo 38

Estrategia WallRace

Esta no es una estrategia de combate propiamente dicha sino un modo de funcionamiento para demostrar las posibilidades del robot.

Esta modo de combate consiste en que el robot avance hacia adelante manteniéndose cerca de la pared. Si estamos en una habitación cerrada el robot se dedicará a dar vueltas a la habitación manteniendo la distancia con la pared más o menos constante.

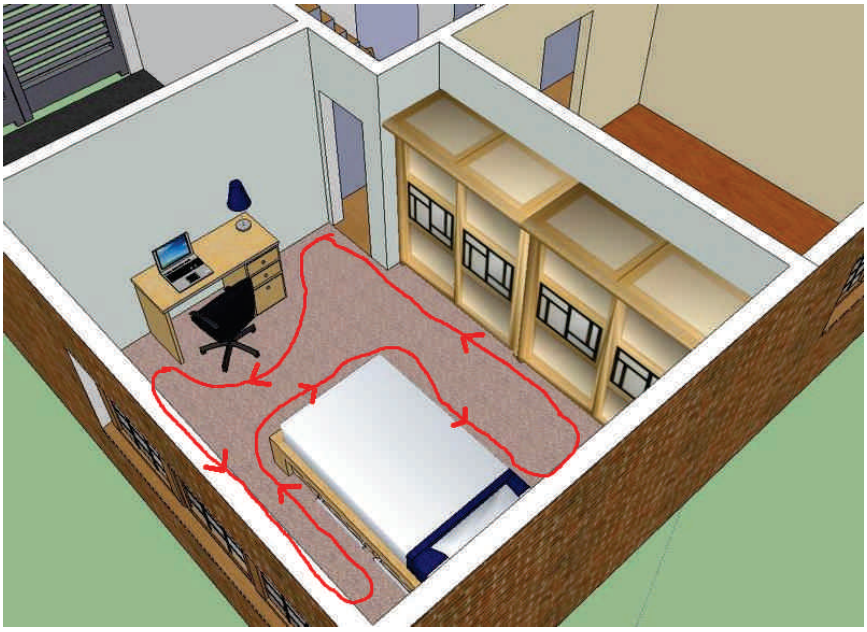


Figura 38.1: Comportamiento del robot en una habitación

Este modo se ha creado basándose en un artículo de la página de robots LetsMakeRobots[1]. Puede verse una demostración en el siguiente vídeo[2].

[1] http://letsmakerobots.com/node/928
[2] http://www.youtube.com/watch?v=YqaCvLFgOtg

Capítulo 39

Memoria en Arduino

Para programar el robot hemos utilizado una placa de Arduino Duemilanove, que según sus especificaciones tiene una memoria de 32 KB de los cuales 2 KB son utilizados por el gestor de arranque (bootloader). Por lo tanto teóricamente nos quedarían 30 KB para poder programar el robot.

Sin embargo en la práctica hemos hallado problemas de funcionamiento cada vez que superábamos la barrera de los 25 KB. Esta cota deberá ser tenida en cuenta en futuros proyectos con Arduino ya que puede ser escasa para determinadas aplicaciones.

Esto ha hecho que hayamos tenido que optimizar el código para poder almacenar todos los modos de funcionamiento que deseábamos. Una de las cosas que más memoria ocupa es el texto que enviamos a la pantalla LCD. También es muy recomendable repasar cada cierto tiempo el código para ver que cosas ya no estamos utilizando, que funciones se pueden mejorar... Por ejemplo la versión 26 del programa ocupaba 24338 bytes y ya estaba cerca del límite por lo que la versión 27 se limitó a mantener las mismas funcionalidades pero optimizando el uso de la memoria, se consiguió que ocupase 19602 bytes por lo que se ahorró unos 5 KB de memoria.

Parte VII

Campeonatos, Mejoras y Conclusión

Capítulo 40

Participación en Robolid

Para poner a prueba el robot y conocer de primera mano a otros aficionados a la robótica se decidió acudir a la competición de Robolid.

Robolid es un concurso de robótica organizado por la Asociación de Microbótica de la Universidad de Valladolid (AMUVA). Hay distintas categorías como siguelíneas, velocistas y sumo. Se han celebrado ya diez ediciones del concurso por lo que es uno de los más importantes a nivel nacional.

40.1. Robolid 2012

Lo primero que hay que decir de Robolid es que se respira un gran ambiente, no es una competición sino un encuentro de aficionados para probar los robots y compartir los conocimientos. La mayoría de los participantes eran ya veteranos con entre 3 y 7 años de experiencia por lo que el nivel era muy alto. Concretamente en la categoría de minisumo el único debutante que había era yo. Además la mayoría competía en varias categorías o incluso algunos tenían varios robots en cada categoría. También es reseñable la presencia de "escuderías", grupos de aficionados que se unen para crear y presentar conjuntamente sus robots.

La competición se trata de presentar como un espectáculo para que el público este entretenido: hay un comentarista narrando los combates, se utiliza un proyector para mostrar los resultados de las pruebas...

Figura 40.1: Vistazo de Robolid 2012

Había mucha profesionalidad en la construcción de los robots. Por ejemplo un robot de la categoría de sumo grande tenía un chasis construido con corte láser que había costado 300 euros. Había también un robot que tenía un inhibidor de señal de infrarrojos que cegaba los sensores de distancia de sus enemigos. Había también un participante que activaba y desactivaba sus robots mediante un mando a distancia.

40.2. Adsumoto

Adsumoto era uno de los robots más llamativos que participaron. Era el único que comenzaba el combate en posición vertical, el único que tenía una pantalla LCD, altavoz... Además tenía el aspecto exterior más cuidado y profesional de todos. Había algunos robots que tenían los cables completamente al aire frente a la carcasa decorada de Adsumoto.

40.3. Campeonato de Minisumo

A la categoría de Minisumo se presentaron un total de 9 robots, por lo que se decidió hacer grupos de 3 robots en los que se enfrentarían para determinar quien pasaba a la ronda final. A Adsumoto le tocó competir contra dos robots del mismo aficionado: Xabi, un catalán que llevaba 5 años construyendo robots y que sería el ganador del torneo con uno de los robots que luchó contra Adsumoto.

Como se ve Adsumoto tuvo como rivales a algunos de los robots más fuertes del torneo y esa fue una de las causas de que perdiera los dos combates. Las otras fueron las ruedas y la secuencia de inicio y se discuten a continuación. Aunque es pura especulación porque solo realizó dos combates, de haberse realizado un campeonato con más enfrentamientos para determinar la posición de cada robot Adsumoto habría quedado en la mitad de la tabla.

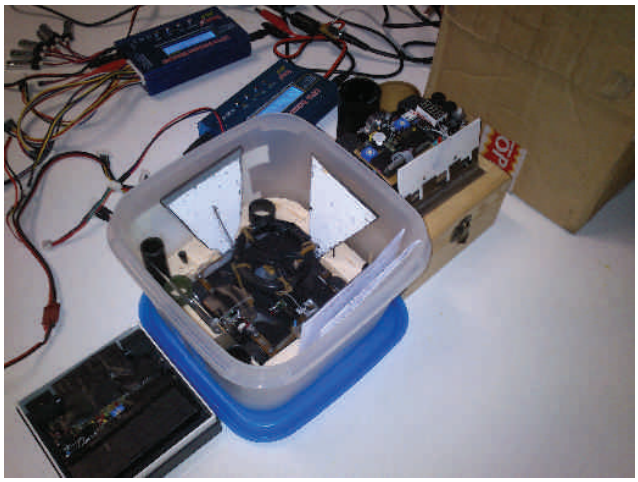


Figura 40.2: Algunos robots participantes en el campeonato

40.4. Ruedas

Uno de los pocos aspectos que era llevado con secretismo era el de las ruedas. Al contrario que nuestro robot que utilizaba unas ruedas comerciales estándar los demás utilizaban ruedas fabricadas por ellos mismos. El proceso de fabricación y el material utilizado eran secretos que guardaban celosamente cada aficionado o cada escudería.

Esta ha sido una de las causas principales de la derrota de Adsumoto. Las ruedas que utilizaban los contrarios tenían mucha más tracción y pese a que el peso lo tenían peor distribuido eran capaces de ejercer una fuerza mayor y ganaban el combate. Además las ruedas de Adsumoto tenían dibujo frente a las ruedas lisas que utilizaban el resto de participantes.

Para la futura construcción de un robot de sumo debe ponerse especial énfasis en conseguir unas ruedas muy adherentes.

40.5. Secuencia de Inicio

La otra gran debilidad era la secuencia de inicio del robot. El inicio de Adsumoto estaba basado en la lectura de los sensores, primero debía colocarse horizontalmente y luego mirar donde estaba el adversario.

Por contra los robots adversarios tenían un interruptor giratorio que permitía elegir la dirección en la que queríamos que saliese el robot. Aunque esta técnica deja menos libertad al robot resultó ser más rápida y efectiva.

Figura 40.3: TE CONNECTIVITY - 1-1825007-6 - SWITCH, ROTARY DIP

Por ello sería buena idea implementar esa opción en nuestro robot ya que de esa forma a la vez que se pone horizontal giraría para orientarse hacia el contrario y no perdería tiempo.

40.6. Conclusión

La participación en Robolid fue una gran experiencia que sirvió para resaltar las debilidades y fortalezas de Adsumoto y poder marcar la dirección de su futura evolución. Además el intercambio de conocimiento con los participantes fue muy enriquecedor.

Capítulo 41

Fortalezas de Adsumoto

En este capítulo se describen las principales fortalezas de Adsumoto para orientar la construcción de robots futuros.

41.1. Pantalla LCD

La pantalla LCD es probablemente la mayor fortaleza de Adsumoto frente a sus rivales. La pantalla le permite una flexibilidad a la hora de elegir estrategia o configurar el comportamiento que no se puede conseguir mediante el uso únicamente de interruptores. Además permite un depurado del programa mucho más rápido y eficiente que sin utilizar pantalla.

41.2. Distribución del Peso

Adsumoto tenía la mayor parte del peso distribuido sobre las ruedas. Por ello si hubiéramos tenido unas ruedas competitivas sin duda habría sido el robot que más fuerza hubiera podido ejercer.

41.3. Acabado Exterior

Aunque no supone ninguna ventaja en el combate conseguir un robot con un buen acabado siempre es bueno de cara a buscar patrocinadores, realizar presentaciones en público...

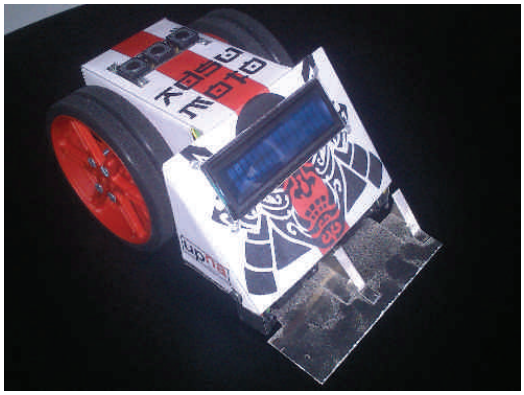


Figura 41.1: Imagen de Adsumoto

Capítulo 42

Mejorando a Adsumoto

En este capítulo se enumeraran las posibles mejoras que harían de Adsumoto un robot campeón.

42.1. Ruedas

La mejora de las ruedas sería probablemente la que elevaría más el rendimiento de Adsumoto.

En primer lugar es necesario comprar ruedas de alta tracción como las Cobra Wheels de Fingertech comentadas en la parte de Elección de Componentes o moldear nuestras propias ruedas utilizando siliconas. Es absolutamente primordial para construir un robot que pueda luchar por la victoria.

En segundo lugar se recomienda evitar las ruedas que tengan los bujes de plástico. Las ruedas de Adsumoto tenían los bujes de plástico y se han desgastado mucho con el contacto con el eje metálico del motor hasta el punto de que giraba el motor y la rueda no lo hacía. Esto supuso un gran problema y hubo que utilizar pegamento como solución temporal para resolverlo.

Una buena idea sería utilizar un dinamómetro para poder medir la fuerza que ejerce el robot y comparar distintas ruedas.

Figura 42.1: Fotografía de la rueda donde se aprecia el desgaste del buje

Como esta mejora suponía un fuerte desembolso económico y no aportaba nada nuevo al proyecto se desestimó llevarla a cabo.

42.2. Sensores de Distancia

Como ya se ha comentado anteriormente los sensores de distancia utilizados son asimétricos. Esto hace que con la colocación utilizada el rastreo sea complicado. En una futura evolución de Adsumoto debería usarse otra disposición de los sensores para conseguir la simetría.

42.3. Acelerómetro

La utilización de un acelorómetro similar al que utilizan los móviles para saber si están en posición vertical u horizontal sería muy interesante en un robot como Adsumoto. Ayudaría a corregir de manera natural la posición del robot cuando se encabrita, serviría para controlar fácilmente la caída del robot cuando pasa de la posición vertical a la horizontal.

42.4. Secuencia de Inicio

Como se ha indicado anteriormente, es necesario mejorar la secuencia de inicio del robot para que a la vez que cae se oriente hacia su enemigo.

42.5. Centro de Gravedad Modificable

Otra mejora sería la introducción de un lastre que pudiera ser movido por un actuador como un servomotor. De esta forma el robot podría modificar la posición del centro de gravedad durante el combate según la situación en la que se encontrase. Por ejemplo si se encabrita movería el peso hacia adelante y si tiene que empujar al contrario lo movería hacia atrás.

42.6. Placa base

Otra gran mejora sería la de diseñar e imprimir una placa base. De esta forma los sensores se colocan directamente sobre la placa base sin utilizar cables por lo que reduce enormemente el espacio necesario. Además se eliminan las posibilidades de fallo debido a que se suelte o se queme algún cable.

Capítulo 43

Ideas para Nuevos Robots

A raíz de la participación en Robolid se nos ocurrieron nuevos conceptos de robot de sumo que mostramos a continuación. Algunos son un poco fantasiosos pero los presentamos porque podrían conducir a otras ideas nuevas.

43.1. Roscobot

El roscobot sería un robot con forma de cilindro que giraría sobre si mismo a gran velocidad. Así el contrario no podría empujarle sino que se deslizaría sobre la carcasa y se vería empujado por el giro. De esta forma el roscobot podría rodear al contrario y situarse en la retaguardia, cuando aprovecharía la ventaja para empujarle fuera del terreno de juego.

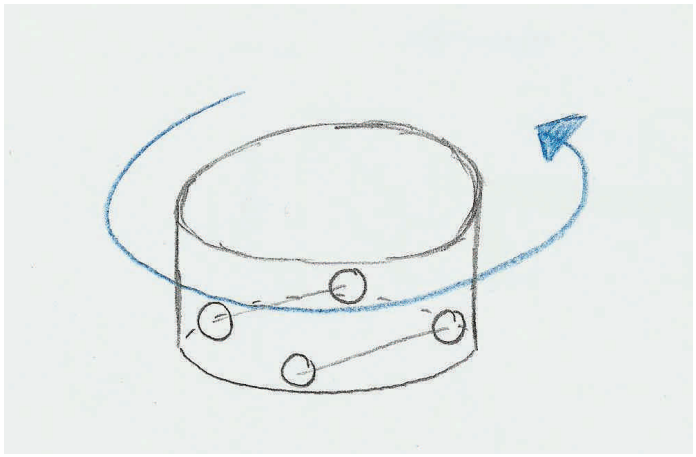


Figura 43.1: Esquema del robot

43.2. Robot Pescador

Se trataría de un robot volador, por ejemplo un quadcoptero, que tendría una red para capturar a su enemigo, elevarlo y sacarlo fuera del terreno de juego. Tendría que tener un motor muy potente para poder elevar al contrario, debería ser capaz de levantar su peso y otros 500g.

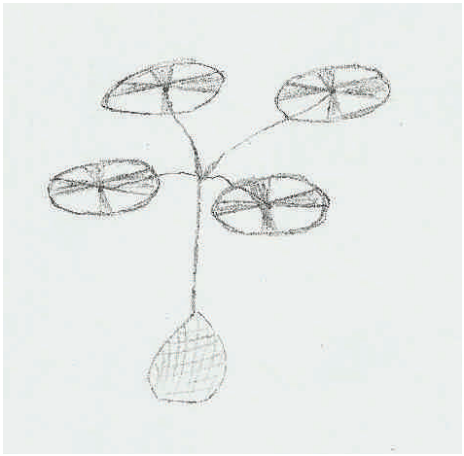


Figura 43.2: Esquema del robot

43.3. Robot Recogedor

Sería un robot que dispondría de una pala similar a la de una excavadora. Cuando chocase contra el contrario desplegaría la pala y levantaría al adversario. Una vez en el aire dejarlo fuera del ring sería fácil.

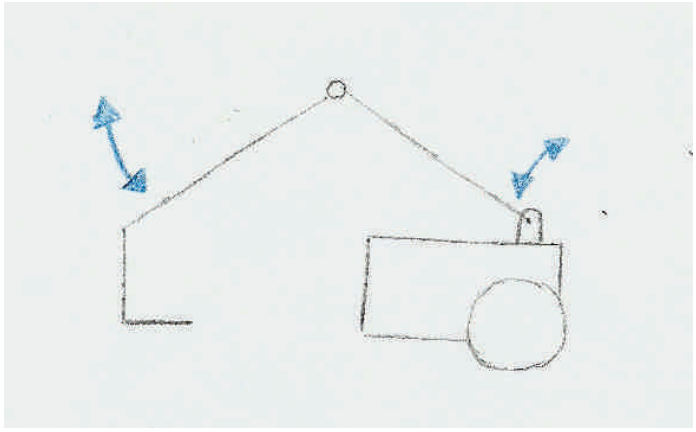


Figura 43.3: Esquema del robot

43.4. Robot Giratorio

Este robot es un poco distinto del roscobot. En este caso el robot se mueve normalmente y su chasis dispone de un motor que lo hace girar a gran velocidad para impedir que se acerquen los contrarios.

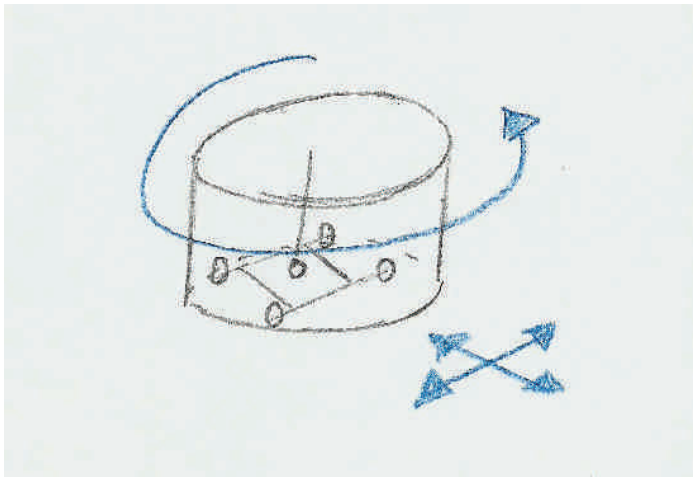


Figura 43.4: Esquema del robot

43.5. Robot Reptador

Este robot avanzaría reptando por el suelo. Tendría una gran área de contacto con el suelo lo que haría que fuese difícil de mover por sus adversarios. Sería un robot lento pero eficaz. Podría estar formado por 3 bloques para poder apoyarse en dos puntos mientras el tercero avanza para coger un nuevo punto de apoyo.

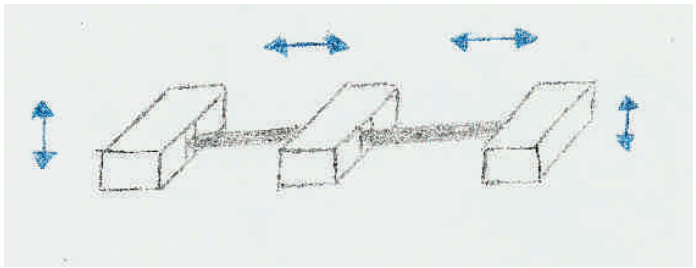


Figura 43.5: Esquema del robot

Capítulo 44

Adsumoto en Internet

Desde el principio Internet ha sido la principal fuente de información a la hora de diseñar y construir a Adsumoto. Si se ha podido construir un robot competitivo partiendo desde cero ha sido en gran medida gracias al trabajo de otros aficionados que han compartido generosamente sus conocimientos en la red.

Es necesario resaltar la página de LetsMakeRobots en la que los mienbros suben los robots que han construido. Hasta el momento hay más de 2000 robots subidos lo que da una idea de la cantidad de información disponible.

Para agradecer la ayuda se decidió subir a Adsumoto a LetsMakeRobots[1] una vez estuviese terminado.

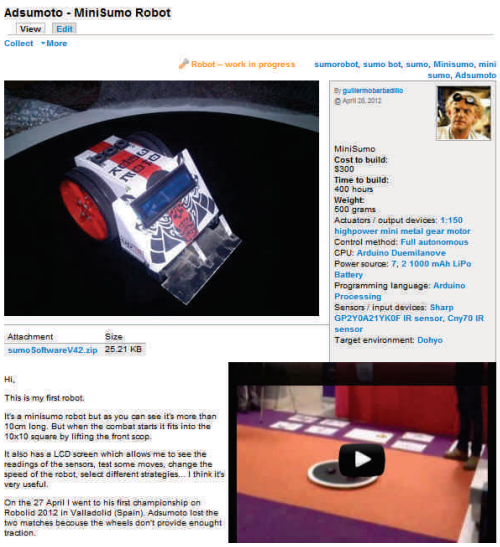


Figura 44.1: Captura de la página de Adsumoto en LetsMakeRobots

A la hora de mostrar el robot lo mejor son los vídeos, por ello se creo un canal de Youtube en el que ir subiendo los videos de la evolución de Adsumoto. La repercusión ha sido realmente sorprendente. Por ejemplo el vídeo más visto tiene 405 reproducciones y ha sido visto en países como Estados Unidos, India, Canadá, Alemania... A día 16 de Agosto entre todos los vídeos de Adsumoto sumaban más de 1200 reproducciones.

[1] http://letsmakerobots.com/node/32435

Capítulo 45

Conclusión

La conclusión principal es que el proyecto ha sido realmente entretenido e instructivo.

El tener una gran componente lúdica ha hecho que trabajar en el Proyecto Fin de Carrera no haya supuesto ningún esfuerzo sino al contrario, ha sido más como un hobby que como un trabajo. Esto ha facilitado en gran medida que se haya podido compaginar el 5º curso de Ingeniería Industrial con la realización del PFC.

Las competencias que se han trabajado en el proyecto son las siguientes:

- Electrónica. Se ha tenido que seleccionar los componentes a utilizar, diseñar los circuitos para que todo funcionase, buscar y comprar los componentes en la página de un proveedor, construir y soldar los circuitos, analizar los circuitos para encontrar los fallos...
- Diseño y Mecánica. Se ha trabajado con programas de diseño en 3d para crear el chasis del robot y comprobar que todas las piezas encajaban en el modelo antes de construirlo. Se han contemplado distintos métodos para construir el robot y se ha utilizado el más sencillo y eficaz para construir un robot robusto.
- Programación. Se ha creado un sistema operativo desde cero para el robot y se han creado multitud de aplicaciones para aprovechar su potencial
- Documentación. Se ha realizado una gran labor de búsqueda de información sobre los robots de sumo. Motores, ruedas, sensores, baterías disponibles en el mercado han sido analizados y se han seleccionado los que mejor cumplían con nuestros objetivos.

Al finalizar el proyecto creo que he aprendido a diseñar un aparato desde cero y a construirlo exitosamente.

Parte VIII

Anexos

Capítulo 46

Recursos Utilizados

46.1. Recursos Bibliográficos

Miles, Pete (2002). Robot Sumo. The Official Guide. McGraw-Hill Osborne Muy buen libro que cubre todos los aspectos en el proceso de diseñar y construir un robot de sumo. Los capítulos de sensores, los estilos de locomoción, el de las baterías..

Miles, Pete y Carrol, Tom (2002). Build your own Combat Robot. McGraw-Hill Prof Med/Tech Este esta más orientado a construir robots más grandes y mortíferos pero si no se dispone del primero sirve muy bien de apoyo.

Arráiz Gil, Javier y Senosiain Miquélez, Vicente (2006). Diseño y construcción de un microbot para competiciones de sumo. Upna Es para la categoría grande de sumo, sirve como ejemplo. Los esquemas para utilizar el CNY70 están sacados de ahí. También tiene una tabla con valores del coeficiente de rozamiento.

46.2. Recursos Electrónicos

http://blog.alvaro-calvo.com/categorias/general/informatica/robots/ Tiene vídeos muy buenos sobre como hace pruebas en su robot. La idea del menú está sacada de él. También tiene vídeos de su participación en distintas competiciones.

http://www.tarribot.com/ Construye un robot bastante sencillo a partir de una caja de CDs. Está muy bien la parte en la que explica lo del chip L293D que se utiliza en los motores.

http://letsmakerobots.com/ Imprescindible para cualquiera que quiera hacer robots. Hay robots a montones y dan explicaciones de como se hizo, material utilizado... En la categoría de minisumo hay un fabricante muy bueno "thesaxmachine". Además permite registrarte y ver si hay usuarios cerca de ti.

Figura 46.1: LetsMakeRobots

http://www.er-online.co.uk/minisumo/index.php Una página inglesa que está muy bien. Sobretodo las explicaciones sobre física y sensores.

http://www.huv.com/miniSumo/ Unos sumos y minisumos con un acabado muy profesional.

http://www.x-robotics.com/sensores.htm Esta página explica los sensores utilizados en sumo y minisumo.

http://www.rchelicopterfun.com/rc-lipo-batteries.html Una guía muy buena sobre las baterías Lipo.

http://www.balticrobotsumo.org/default.asp?itemID=247&itemTitle=Cast

http://www.davehylands.com/Robotics/Marauder/Making-Tires/index.html

http://www.pdxbot.com/howto/making_sticky.php?link_id=14 Tutoriales para realizar tiras de poliuretano para las ruedas.

http://letsmakerobots.com/node/22781 Como se pueden mejorar los sensores Sharp GP2Dxxx.

http://letsmakerobots.com/node/2074 Condensadores que se deben utilizar en el L293D.

http://letsmakerobots.com/node/3880 Condensadores en la batería y el regulador de tensión

http://www.balticrobotsumo.org/ Una página muy buena en la que cuenta como se construyeron varios robots.

http://cssbl.com/aire/stealth.htm Información sobre los aviones antiradar.

http://www.youtube.com/playlist?list=PL2EBF373920E95C08&feature=mh_lolz Una lista de reproducción de Youtube con vídeos de sumo robótico.

http://www.youtube.com/user/guillermobarbadillo Mi canal de Youtube con todos los vídeos de Adsumoto.

46.3. Tiendas

- Pololu
- Fingertech
- Solarbotics
- Parallax
- Tamiya
- Farnell
- Superrobotica
- HobbyKing. Para baterías

- DealExtreme
- TapPlastics. Acrílico y poliuretano.
- Shoptronica

46.4. Arduino

http://www.arduino.cc/ Toda la información sobre Arduino. Instrucciones, ejemplos, foros...

http://fritzing.org/ Fritzing es un software libre que permite hacer esquemáticos de las conexiones con Arduino de manera muy fácil y vistosa

Margolis, Michael (2010). Arduino CookBook. O'Reilly Media Un libro que tiene casi todo lo que hay que saber sobre Arduino. Muy buenos ejemplos y muy bien explicados.

Capítulo 47

Instrucciones de Uso del Robot

En este capítulo se narran distintos consejos a la hora de manejar el robot.

47.1. Encendido

El robot se enciende con el interruptor situado en la base.

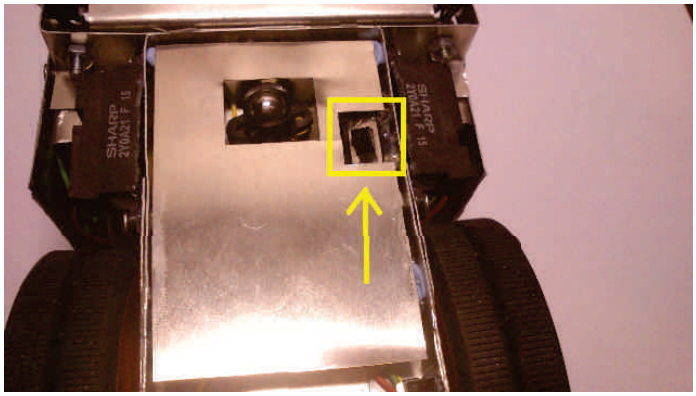


Figura 47.1: Interruptor de encendido del robot

Pero a veces hay problemas con la sincronización de la pantalla LCD. Por eso se ha puesto un segundo interruptor que abre o cierra el pin de comunicación de la pantalla.

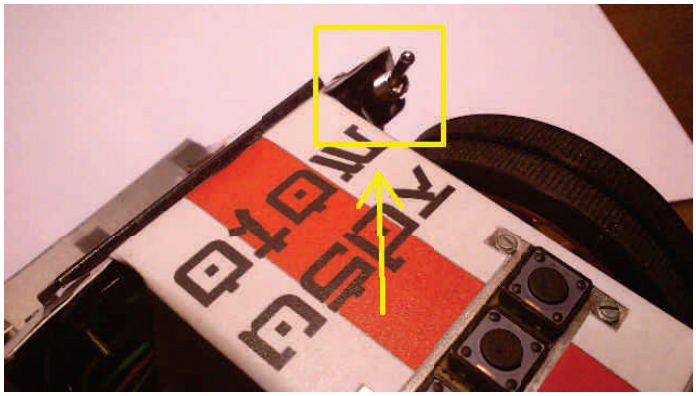


Figura 47.2: Interruptor de la pantalla LCD

Si encendemos el robot y la pantalla no se ha sincronizado hay que seguir los siguientes pasos.

1. Apagar el robot
2. Desconectar la pantalla
3. Encender el robot
4. Esperar. A veces funciona esperando 5 segundos, a veces con medio. Es aleatorio
5. Conectar la pantalla
6. Si no se ha sincronizado repetir los pasos anteriores. Al final acaba sincronizándose bien.

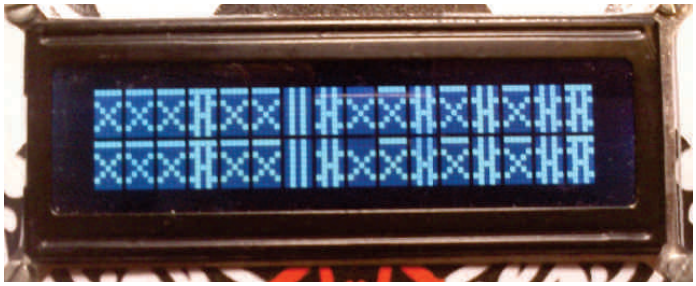


Figura 47.3: Imagen de la pantalla mal sincronizada

47.2. Abrir Carcasa

Para abrir la carcasa es necesario retirar primero los tornillos laterales.

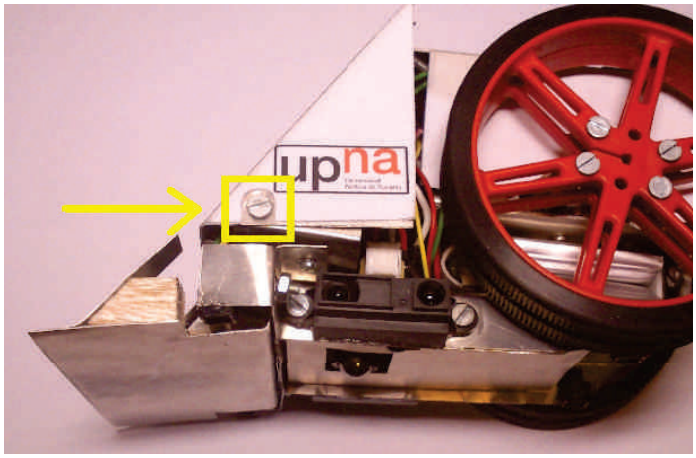


Figura 47.4: Tornillos laterales que sujetan la carcasa

47.3. Quitar ruedas

Para dañar lo menos posible los bujes de plástico de las ruedas lo mejor es retirarlas ayudándose de un destornillador como indica la imagen.

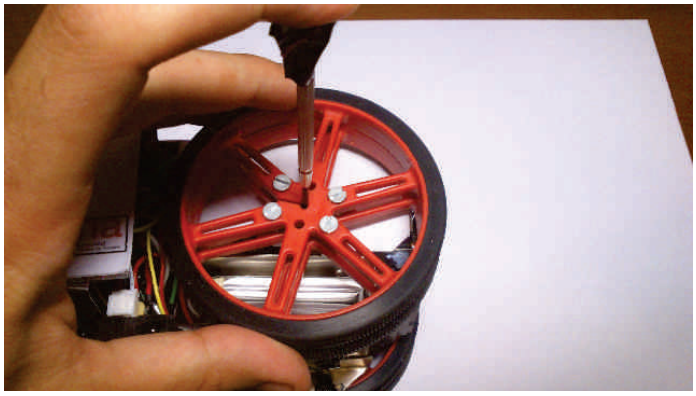


Figura 47.5: Las ruedas se retiran ayudándose de un destornillador

47.4. Cambiar la batería

Para cambiar la batería hay que retirar la rueda y luego sacar la batería. Lo mejor es retirar la rueda que no está junto al interruptor de la pantalla.

47.5. Recargar la batería

El cargador de la batería falla a veces. Por eso hay que usar pequeños trucos como utilizar el cargador 3S en vez del 2S o no conectar la batería hasta el final.

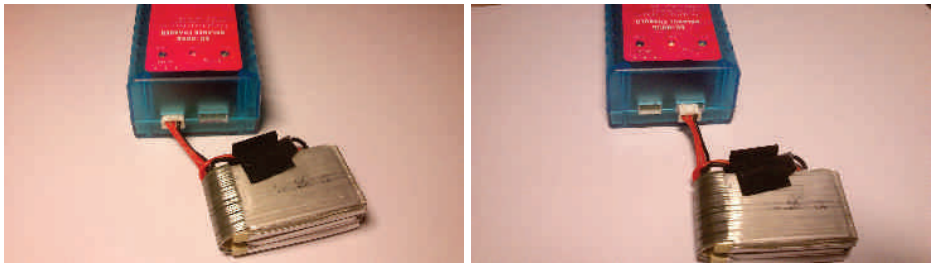


Figura 47.6: Cargador de las baterías

47.6. Programar el robot

Para programar el robot debemos conectar en primer lugar el cable USB entre el robot y el ordenador. Después abrimos la aplicación de Arduino y cargamos el programa que queramos.

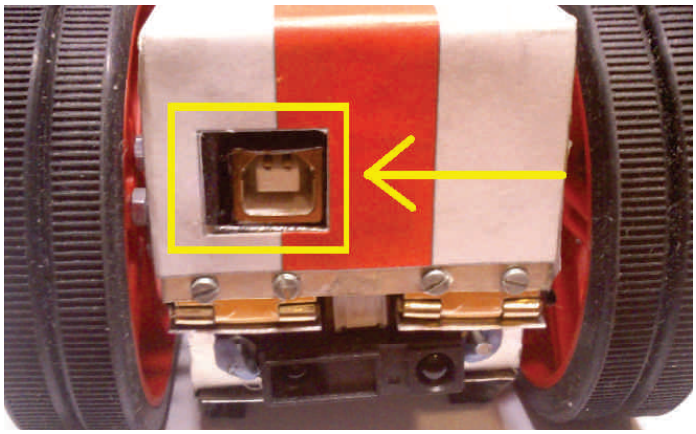


Figura 47.7: Clavija para el cable USB de Adsumoto

Chapter 48

Código Fuente

A continuación se muestra el código fuente de la programación del robot. Se ha utilizado el entorno de desarrollo de Arduino y se ha dividido el código en pestañas para tenerlo más organizado.

Para poder mostrar el código Arduino tiene la opción de imprimir cada pestaña por separado. Para que quepa bien es necesario cambiar el tamaño de la letra a 10 y reiniciar el programa. Los diferentes archivos son:

1. Estructura del programa
2. Tipos
3. Menú
4. Pulsación de botón
5. Transiciones
6. Funciones escribir LCD
7. Control Motores
8. Estrategia Fluida
9. Estrategia Suicida
10. Estrategia Cobarde
11. Vueltas al Ring
12. Wallrace
13. Bonus

```
/*
ESTRUCTURA DEL PROGRAMA
*/

//*****************************************************************************
//   LIBRERÍAS UTILIZADAS
//*****************************************************************************
#include <serLCD.h>
#include <SoftwareSerial.h>
#include "types.h"

//*****************************************************************************
//  PINES DE ARDUINO
//*****************************************************************************
//Botones
#define btnSiguiente 2
#define btnIntro 1
#define btnAtras 0
//Pantalla
#define pantallaLCD 3
//Sensores Sharp
#define    sharpI    3//Izquierda
#define   sharpDI    2//Delantero izquierda
#define sharpDD 1
#define sharpD 0
#define sharpT 4
//Batería
#define bateria 5
//Sensores CNY70
#define CNY70DI 4
#define CNY70DD 6
#define  CNY70TD  7
#define  CNY70TI  5
//Pines para control del motor
#define motorI1 9
#define motorI2 8
#define motorD1 11
#define motorD2 12
#define altavoz 10


//*****************************************************************************
//     VARIABLES
//*****************************************************************************


//Valores predeterminados de las variables
eModo  modo=MENU;
//Iniciamos en modo menu mientras no llegue el campeonato
eMenu   menu=ESTRATEGIA;
//define la posicion del menu
eNivel   nivel=FUERA;
//define si estamos mirando el menu o dentro de la aplicacion del menu
eLuz   luz=ENCENDIDA;
//define si la luz de la pantalla LCD esta encendida o apagada
```

```
eEstrategia   estrategia=COBARDE;
//define la estrategia que adopta el robot
eEstrategia  menuEstrategia=FLUIDO;
//define la posicion del menu de estrategia
eMovimientos  movimientoAnterior=ADELANTE;
//guarda el movimiento anterior del robot

//Objeto para controlar el lcd
serLCD lcd(pantallaLCD);

//Variables de funcionamiento del programa
int  delayBoton  =  200;
//delayBoton determina cuanto esperamos tras pulsar el boton en el menu
//para evitar rebote
int  delayCombate  =  10;
//delayCombate determina cuanto tiempo pasa en el modo combate para comprobar
//que hemos pulsado el boton realmente y no es un fallo.

unsigned long     tiempoGirando;
//tiempoGirando es una variable para no girar indefinidamente
unsigned long     tiempoLento;
//tiempoLento sirve para evitar que levantemos el morro con cambios
//de velocidad bruscos
unsigned long     tiempoAtras;
//tiempoAtras sirve para evitar que levantemos el morro con cambios
//de velocidad bruscos
int tiempoExtra=0;

int     velocidadMaxima=128;
//De 0-255  Me voy a mover en 128-160-192-224-255
byte velocidadAuxiliar=velocidadMaxima;
byte velocidadMinima=80;

//Las dos siguientes variables establecen el tiempo de contragolpe en el
// giro de los motores
int   delayGiroLocoI;
int  delayGiroLocoD;
int   tiempoSalvacion=100;
//Este es el tiempo que retrocedemos al detectar que estamos en el borde
int   limiteGiro;
//Esta variable sirve para que no estemos girando eternamente sino que
// pasado un rato vayamos recto.
int randomGiro;

//Variables contadores para eliminar el ruido
int contadorDD=0;
int contadorDI=0;
int contadorD=0;
int contadorT=0;
int contadorI=0;

//Variables motores
unsigned int motorI;
unsigned int motorD;
```

```
//Variable para activar 0/ desactivar 1 el sensor trasero.
boolean  sensorTrasero=0;//Lo pongo activado de serie

//Variable para hacer lo del coche fantastico
int contadorFantastico=7;
boolean direccionFantastico=1;
int numeroAleatorio=random(4);
//*****************************************************************************
//   SETUP
//*****************************************************************************
//En setup incluimos aquello que queremos que haga solo al
//principio del programa, por ejemplo la configuración de los pines
void setup () {
  //Configuramos los pines de sensores y botones como input
  pinMode(btnSiguiente,INPUT);
  pinMode(btnIntro,INPUT);
  pinMode(btnAtras,INPUT);
  pinMode(CNY70DI,INPUT);
  pinMode(CNY70DD,INPUT);
  pinMode(CNY70TI,INPUT);
  pinMode(CNY70TD,INPUT);
  //Configuramos los pines de control del motor como salidas
  pinMode(motorI1,OUTPUT);
  pinMode(motorI2,OUTPUT);
  pinMode(motorD1,OUTPUT);
  pinMode(motorD2,OUTPUT);
  // Conectamos las resistencias internas a los botones, de esta
  //manera no usamos resistencias. Conectamos el boton al pin y a GND
  digitalWrite(btnSiguiente, HIGH);
  digitalWrite(btnIntro, HIGH);
  digitalWrite(btnAtras, HIGH);
  //Iniciamos la variable aleatoria leyendo la batería.
  randomSeed(analogRead(5));
  //Ajustamos los motores
  moverFrenar();
    switch (velocidadMaxima) {
    case 128: delayGiroLocoI=130; delayGiroLocoD=160;
       limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=100
      break;
    case 160: delayGiroLocoI=140; delayGiroLocoD=170;
       limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=100
      break;
    case 192: delayGiroLocoI=150; delayGiroLocoD=180;
       limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=125
      break;
    case 224: delayGiroLocoI=152; delayGiroLocoD=180;
       limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=200
      break;
    case 255: delayGiroLocoI=155; delayGiroLocoD=180;
       limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=275
      break;
  }
  //Enviamos un saludo
  mensajeInicio4();
}
```

```
//*****************************************************************************
//   LOOP
//*****************************************************************************
void loop () {
  switch (modo) {
    case COMBATE:
    switch (estrategia) {
      //case AGRESIVA:     estrategiaAgresiva() ; break;
      case VUELTAS:       estrategiaVueltas();break;
      case FLUIDO:        estrategiaFluido();break;
      case SUICIDA:       estrategiaSuicida();break;
      case WALLRACE:      estrategiaWallRace();break;
      case COBARDE:       estrategiaCobarde();break;
      default: borrarEscribirLCD(" FALLO EN "," ESTRATEGIA"); delay(500); lcd.clear()
        break;
    }
    pulsacionBoton();
    break;

    case MENU:
    switch (nivel) {
      case  FUERA:  mostrarMenu();break;
      case  DENTRO:  ejecutarAplicacion();break;
      default: borrarEscribirLCD(" FALLO EN "," NIVEL"); delay(500);
    }
    comprobarBateria();
    pulsacionBoton();
    break;

    default: borrarEscribirLCD(" FALLO EN "," MODO"); delay(500);
  }
}
```

```
/* TIPOS */
//Defino aquí las enumeraciones porque si no no se pueden usar como funcion

//Ahora voy a definir las enumeraciones que utilizaremos como variables
//globales de control de flujo
enum eModo {COMBATE, MENU};
//eModo Enumeración Modo, asi puedo usar modo como variable
//Cuando se añade una nueva opción al menu hay que meterla
//en aplicaciones menu y tambien en la pulsacion de boton.
enum eMenu {ESTRATEGIA, SHARP, CNY70, BATERIA, LUZ, FANTASTICO,RESET,RADAR,
PRUEBAMOTOR,PRUEBAHUIR,PRUEBAESTRATEGIA,SHARP2,GIROLOCOD,GIROLOCOI,VERTICAL
HORIZONTAL,HORIZONTAL2,VELOCIDAD,BANCOPRUEBAS,SONIDO,SENSORTRASERO};
enum eNivel {FUERA, DENTRO};
enum eLuz{ENCENDIDA, APAGADA};
enum eEstrategia{ AGRESIVA,VUELTAS,FLUIDO,SUICIDA,WALLRACE,COBARDE};
enum eCNY70 {DD,DI,TD,TI,Z,D,I};
enum eMovimientos {ADELANTE,ATRAS,DERECHA,IZQUIERDA,ATRASDERECHA,
ATRASIZQUIERDA,INDETERMINADO,MEDIAVUELTA,QUIETO,ENCARANDO};
```

```
/*
MENU
En esta pestaña voy a definir las distintas funciones que vamos a utilizar en el menu
La nomenclatura sera menuNombreFuncion, p.e. menuEstrategia
Tambien voy a meter aqui las funciones que muestran el escritorio
o ejecutan la aplicacion*/

//Esta funcioin ejecuta la aplicacion del menu que este seleccionada en ese momento.
void ejecutarAplicacion() {
  switch (menu) {
     case ESTRATEGIA:       elegirEstrategia();break;
     case SHARP:            menuSharp();break;
     case SHARP2:           menuSharp2();break;
     case CNY70:            menuCNY70();break;
     case BATERIA:          menuBateria();break;
     case LUZ:              menuLuz();break;
     case FANTASTICO:       menuFantastico();break;
     case RESET:            softwareReset();break;
     case RADAR:            menuRadar();break;
     case PRUEBAMOTOR:      menuPruebaMotor();break;
     case PRUEBAHUIR:       menuPruebaHuir();break;
     case PRUEBAESTRATEGIA:  menuPruebaEstrategia();break;
     case VERTICAL:         menuVertical();break;
     case HORIZONTAL2:      menuHorizontal2();break;
     case VELOCIDAD:        menuVelocidad();break;
     case BANCOPRUEBAS:     menuBancoPruebas();break;
     case SONIDO:           menuSonido();break;
     case SENSORTRASERO:    menuSensorTrasero();break;
     default:              borrarEscribirLCD(" FALLO EN "," MENU");
  }
}

//Esta funcion muestra la posicion del menu en la que nos encontramos
void mostrarMenu() {
  goTo(6);
  lcd.print("MENU");
  switch (menu) {
        case ESTRATEGIA:    goTo(16);lcd.print("^  ESTRATEGIA  >"); break;
        case SHARP:         goTo(16);lcd.print("^  LEER SHARP  >"); break;
        case SHARP2:        goTo(16);lcd.print("^  LEER SHARP2 >"); break;
        case CNY70:         goTo(16);lcd.print("^  LEER CNY70  >"); break;
        case BATERIA:       goTo(16);lcd.print("^   BATERIA    >"); break;
        case LUZ:           goTo(16);lcd.print("^     LUZ      >"); break;
        case FANTASTICO:    goTo(16);lcd.print("^ C.FANTASTICO >"); break;
        case RESET:         goTo(16);lcd.print("^    RESET     >"); break;
        case RADAR:         goTo(16);lcd.print("^    RADAR     >"); break;
        case VERTICAL:      goTo(16);lcd.print("^  VERTICAL    >"); break;
        case HORIZONTAL2:   goTo(16);lcd.print("^CAIDA FORZADA >"); break;
        case VELOCIDAD:     goTo(16);lcd.print("^  VELOCIDAD   >"); break;
        case BANCOPRUEBAS:  goTo(16);lcd.print("^BANCO PRUEBAS >"); break;
        case SONIDO:        goTo(16);lcd.print("^    SONIDO    >"); break;
        case SENSORTRASERO: goTo(16);lcd.print("^SENSOR TRASERO>"); break;
        case PRUEBAMOTOR:             goTo(16);lcd.print("^ PRUEBA MOTOR >"); break;
        case PRUEBAHUIR:              goTo(16);lcd.print("^ PRUEBA HUIR  >"); break;
        case PRUEBAESTRATEGIA:        goTo(16);lcd.print("^ PRUEBA ESTRA >"); break;
```

```
        default:            borrarEscribirLCD(" FALLO EN "," MENU");
      }
}

//Esta función permite seleccionar entre las estrategias disponibles
void elegirEstrategia (){
  goTo(3);lcd.print("ESTRATEGIA");
  switch   (menuEstrategia)    {
        case VUELTAS:       goTo(16);lcd.print("^    VUELTAS   >"); break;
        case FLUIDO:        goTo(16);lcd.print("^    FLUIDO    >"); break;
        case SUICIDA:       goTo(16);lcd.print("^   SUICIDA    >"); break;
        case WALLRACE:      goTo(16);lcd.print("^  WALL RACE   >"); break;
        case COBARDE:       goTo(16);lcd.print("^  COBARDERR   >"); break;
        default:            borrarEscribirLCD(" FALLO EN ","MENU ESTRATEGIA");
  }
  if (digitalRead(btnSiguiente)==LOW) {
    switch               (menuEstrategia)                              {
        case VUELTAS:        menuEstrategia=FLUIDO;break;
        case FLUIDO:         menuEstrategia=SUICIDA;break;
        case SUICIDA:        menuEstrategia=WALLRACE;break;
        case WALLRACE:       menuEstrategia=COBARDE;break;
        case COBARDE:        menuEstrategia=VUELTAS;break;
        default:            borrarEscribirLCD(" FALLO EN ","MENU ESTRATEGIA");
    }
    delay(delayBoton);
  }
  if (digitalRead(btnIntro)==LOW) {
    switch               (menuEstrategia)                              {
        case VUELTAS:       estrategia=VUELTAS; lcd.clear();
          delay(500); borrarEscribirLCD("ELEGIDA ESTRTGIA",">>>  VUELTAS <<<");
          delay(1000); break;
        case FLUIDO:        estrategia=FLUIDO; lcd.clear();
          delay(500); borrarEscribirLCD("ELEGIDA ESTRTGIA",">>>  FLUIDO  <<<");
          delay(1000); break;
        case SUICIDA:       estrategia=SUICIDA; lcd.clear();
          delay(500); borrarEscribirLCD("ELEGIDA ESTRTGIA",">>> SUICIDA  <<<");
          delay(1000); break;
        case WALLRACE:      estrategia=WALLRACE; lcd.clear();
          delay(500); borrarEscribirLCD("ELEGIDA ESTRTGIA",">>>WALL RACE <<<");
          delay(1000); break;
        case COBARDE:       estrategia=COBARDE; lcd.clear();
          delay(500); borrarEscribirLCD("ELEGIDA ESTRTGIA",">>> COBARDERR <<<");
          delay(1000); break;
        default:            borrarEscribirLCD(" FALLO EN ","MENU ESTRATEGIA");
    }
    delay(delayBoton);
    lcd.clear();
  }


}
```

```
// Esta función lee el valor de los sensores Sharp y lo muestra por pantalla.
//  Esta optimizada para que los valores salgan siempre en la misma posición
void menuSharp () {
  lcd.clear();
  int lecturaDI=analogRead(sharpDI);
  goTo(25);
  if (lecturaDI>99) lcd.print("DI"+String(lecturaDI));
  else
    if (lecturaDI>9) lcd.print("DI "+String(lecturaDI));
    else lcd.print("DI  "+String(lecturaDI));

  int lecturaDD=analogRead(sharpDD);
  goTo(18);
  if (lecturaDD>99) lcd.print("DD"+String(lecturaDD));
  else
    if (lecturaDD>9) lcd.print("DD "+String(lecturaDD));
    else lcd.print("DD  "+String(lecturaDD));

  int lecturaI=analogRead(sharpI);
  goTo(11);
  if (lecturaI>99) lcd.print("I"+String(lecturaI));
  else
    if (lecturaI>9) lcd.print("I "+String(lecturaI));
    else lcd.print("I  "+String(lecturaI));

  int lecturaT=analogRead(sharpT);
  goTo(6);
  if (lecturaT>99) lcd.print("T"+String(lecturaT));
  else
    if (lecturaT>9) lcd.print("T "+String(lecturaT));
    else lcd.print("T  "+String(lecturaT));

  int lecturaD=analogRead(sharpD);
  goTo(1);
  if (lecturaD>99) lcd.print("D"+String(lecturaD));
  else
    if (lecturaD>9) lcd.print("D "+String(lecturaD));
    else lcd.print("D  "+String(lecturaD));
  delay(100);
}

void menuSharp2 () {
  int lecturaDI=analogRead(sharpDI);
  goTo(26);
  if (lecturaDI>200) lcd.print("XXX"); else lcd.print("   ");
  int lecturaDD=analogRead(sharpDD);
  goTo(18);
  if (lecturaDD>200) lcd.print("XXX"); else lcd.print("   ");
  int lecturaI=analogRead(sharpI);
  goTo(12);
  if (lecturaI>200) lcd.print("XXX"); else lcd.print("   ");
  int lecturaT=analogRead(sharpT);
  goTo(6);
  if (lecturaT>200) lcd.print("XXX"); else lcd.print("   ");
  int lecturaD=analogRead(sharpD);
```

```
  goTo(1);
  if (lecturaD>200) lcd.print("XXX"); else lcd.print("   ");
  //delay(25);
}

//La funcion lee el valor de los sensores CNY70 y lo muestra por pantalla.
void menuCNY70 () {
  lcd.clear();
  byte lecturaDI=digitalRead(CNY70DI);
  goTo(25);
  lcd.print("DI="+String(lecturaDI));
  byte lecturaDD=digitalRead(CNY70DD);
  goTo(18);
  lcd.print("DD="+String(lecturaDD));
  byte lecturaTI=digitalRead(CNY70TI);
  goTo(9);
  lcd.print("TI="+String(lecturaTI));
  byte lecturaTD=digitalRead(CNY70TD);
  goTo(2);
  lcd.print("TD="+String(lecturaTD));
  delay(50);
}

//Esta funcion lee el voltaje de la batería y lo muestra por pantalla.
void menuBateria () {
  int lectura  =analogRead(bateria);
  //float voltaje = lectura*5/1023*2*1000;  //En milivoltios
  //Estoy teniendo algún problema de desbordamiento. Voy a usar un truco matematico.
  //En vez de utilizar 1023 voy a usar 1024
  //10000/1024=10^4/2^10=5^4/2^6=625/64
  //La precision va a ser de 10mV aproximadamente
  int  voltaje=lectura*25/32*25/2;//Aqui estoy perdiendo precision porque multiplico
  //por 25 y divido entre 32. Hay que evitar las contracciones. 0.781*12.5
  //Pero utilizando 1023 la perdida de precisión sería aun mayor porque lo mejor
  // que puedo hacer es 20/31=0.645
  //El orden de la operación es importante para que no se desborde o se pierda precis
  //Hay que cuidar que no tengamos algo mayor de 32000 y dividir al final
  // para no perder precisión.
  // voltaje=lectura/64*625 <> voltaje=lectura*25/64*25 <> voltaje=lectura*25/32*25/2
  int voltios = divisionEntera(voltaje,1000);
  int milivoltios = voltaje - 1000*voltios;
  if (voltaje>7200) {
    if (milivoltios>99) borrarEscribirLCD("BATERIA: "+String(voltios)+
      "."+String(milivoltios)+"V","VOLTAJE OK");
    else {
      if (milivoltios>9) borrarEscribirLCD("BATERIA: "+String(voltios)+
        ".0"+String(milivoltios)+"V","VOLTAJE OK");
      else borrarEscribirLCD("BATERIA: "+String(voltios)+".00"+
        String(milivoltios)+"V","VOLTAJE OK");
    }
  }
    else {
      if (milivoltios>99) borrarEscribirLCD("BATERIA: "+String(voltios)+
        "."+String(milivoltios)+"V","RECARGAME YA");
      else {
```

```
        if (milivoltios>9) borrarEscribirLCD("BATERIA: "+String(voltios)+
          ".0"+String(milivoltios)+"V","RECARGAME YA");
        else borrarEscribirLCD("BATERIA: "+String(voltios)+".00"+
          String(milivoltios)+"V","RECARGAME YA");
    }
    }
  delay(50);
}

//Esta función comprueba que el voltaje de la batería es correcto
// y si no lo es muestra un mensaje por pantalla
void comprobarBateria () {
  int lectura =analogRead(bateria);
  int voltaje=lectura*25/32*25/2;
  int voltios = divisionEntera(voltaje,1000);
  int milivoltios = voltaje - 1000*voltios;
  if (voltaje<7250) {
    goTo(0);
    lcd.print("!B!");
    goTo(13);
    lcd.print("!B!");
    delay(1);
    }
}



//Enciende o apaga la luz de la pantalla LCD
void menuLuz() {
  if (luz==ENCENDIDA) {
    lcd.setBrightness(1);
   luz=APAGADA;
   }
  else {
  lcd.setBrightness(30);
  luz=ENCENDIDA;
  }
  nivel=FUERA;
}

//Muestra por pantalla el efecto clasico de coche fantastico
void menuFantastico() {
//  for (int i=0;i<16;i++) cocheFantastico2(i);
//  for (int i=15;i>=0;i--) cocheFantastico2(i);
  cocheFantastico2(contadorFantastico);
  delay(4);
  if (contadorFantastico==0) direccionFantastico=1;
  if (contadorFantastico==15) direccionFantastico=0;
  if (direccionFantastico) contadorFantastico++;
  else contadorFantastico--;
}

//Muestra por pantalla la posicion de lo que haya delante del robot.
void menuRadar(){
  lcd.clear();
```

```
int lecturaDI=analogRead(sharpDI);
int lecturaDD=analogRead(sharpDD);
//Escribimos los valores por pantalla
if (lecturaDD>99) {
  goTo(1);
  lcd.print("DD:"+String(lecturaDD));
}
else {
  if (lecturaDD>9) {
    goTo(1);
    lcd.print("DD: "+String(lecturaDD));
  }
  else {
    goTo(1);
    lcd.print("DD:  "+String(lecturaDD));
  }
}

if (lecturaDI>99) {
  goTo(9);
  lcd.print("DI:"+String(lecturaDI));
}
else {
  if (lecturaDD>9) {
    goTo(9);
    lcd.print("DI: "+String(lecturaDI));
  }
  else {
    goTo(9);
    lcd.print("DI:  "+String(lecturaDI));
  }
}

//Ahora tenemos que determinar la posicion
//Primero pongo un umbral de sensibilidad, si no lo cumple me salgo
if ((lecturaDI<200)&&(lecturaDD<200)) {
  goTo(21);
  lcd.print("??");
  delay(50);
  return;
}
int  posicion;//posicion tiene que estar al final entre 0 y 15.
//Ahora tengo que distinguir los casos de izquierda o derecha
if (lecturaDD>lecturaDI) {
  //Tenemos tres franjas, una en la que lecturaDI es muy pequeña,
  //otra en la que va creciendo y una última en la que son muy parecidos.
  //Esta podría ser la condición de muy derecha
  if  (lecturaDI<10)  {//En la derecha los valoresDI bajan muy rápido,
   // no es necesario un umbral
    posicion=0;
  }
  else {
    posicion=7-7*(lecturaDD-lecturaDI)/(lecturaDD+lecturaDI);
  }
}
```

```
  else {
    if  (lecturaDD<60)   {//Cuando tenemos algo muy a la izquierda del
    //sensor oscila mucho, por eso el umbral lo ponemos mucho más alto aquí
      posicion=14;
    }
    else {
      posicion=7+7*(lecturaDI-lecturaDD)/(lecturaDD+lecturaDI);
    }
  }

  goTo(posicion+16);
  lcd.print("##");
  delay(50);
}

//Esto es para comprobar que los motores están bien conectados
//y funcionan correctamente
void menuPruebaMotor() {
  delay(2000);  //Esperamos un poco antes de empezar la prueba
//Ahora hacemos una prueba de los dos motores conjuntamente
  goTo(0);
  lcd.print("IZQ ATRAS");
  motorIAtrasPWM(80);
  delay(200);
  moverFrenar();
  delay(1000);
  goTo(0);
  lcd.print("DER ATRAS");
  motorDAtrasPWM(80);
  delay(200);
  moverFrenar();
  delay(1000);
  goTo(0);
  lcd.print("IZQ ADEL");
  motorIAdelantePWM(80);
  delay(200);
  moverFrenar();
  delay(1000);
  goTo(0);
  lcd.print("DER ADEL");
  motorDAdelantePWM(80);
  delay(200);
  moverFrenar();
  delay(1000);

  nivel=FUERA;
  lcd.clear();
}

void  menuPruebaHuir()   {
  //Primero activo motores y eso y muestro los valores de los sensores
  moverFrenar();

  menuCNY70();
  delay(2000);
```

```
  //Ahora activo el mecanismo de huida
  eCNY70 estado=sensoresSuelo();
  switch (estado) {
    case   Z:break;
    default: maniobraEvasivaFluido(estado);
  }
  //Lo repito por si estaba todo un lado en lo blanco
  estado=sensoresSuelo();
  switch (estado) {
    case   Z:break;
    default: maniobraEvasivaFluido(estado);
  }
  moverFrenar();
  nivel=FUERA;
  lcd.clear();
}

void menuPruebaEstrategia() {
  menuSharp();
  delay(2000);
  int herzios=0;
  movimientoAnterior=QUIETO;
  contadorDD=0;
  contadorDI=0;
  contadorD=0;
  contadorI=0;
  contadorT=0;
  tiempoLento=millis();
  unsigned long  tiempoEstrategia=millis();
  while (millis()-tiempoEstrategia<2000) {
    switch (estrategia) {
      //case AGRESIVA:     estrategiaAgresiva() ; break;
      case VUELTAS:       estrategiaVueltas();break;
      case FLUIDO:        estrategiaFluido();break;
      case SUICIDA:       estrategiaSuicida();break;
      case WALLRACE:      estrategiaWallRace();break;
      case COBARDE:       estrategiaCobarde();break;
      default: borrarEscribirLCD(" FALLO EN "," ESTRATEGIA");
        delay(500); lcd.clear(); break;
    }
    herzios++;
  }
  nivel=FUERA;
  menu=ESTRATEGIA;
  moverFrenar();
  delay(1000);
  borrarEscribirLCD(" HERZIOS: "," "+String(herzios/2));
  delay(2000);
  lcd.clear();
}



//Modo para poner el robot vertical.
void menuVertical() {
```

```
  borrarEscribirLCD("CAMBIO A MODO","  VERTICAL");
  delay(2000);
  moverFrenar();

  moverAtrasPWM(255);
  delay(75);
  moverAdelantePWM(255);
  delay(500);
  //Ahora está levantado y lo que hay que hacer es frenarlo
  //sin que se caiga. Primero frenamos el motor izquierdo
for (byte i=0;i<128;i++) {
     motorIAdelantePWM(255-2*i);
    delay(1);
  }

  for (byte i=0;i<128;i++) {
     motorIAtrasPWM(2*i);
    delay(1);
  }

  for (byte i=255;i>30;i--) {
    girarIzquierdaPWM(i,0);
    delay(25);
  }

  moverFrenar();
  nivel=FUERA;
  lcd.clear();
}



//Modo de caída forzada. Tratamos de que llegue al suelo lo más rápido posible.
void menuHorizontal2() {
  borrarEscribirLCD("CAMBIO A MODO","  HORIZONTAL");
  delay(2000);

  caidaForzada();


  moverFrenar();
  nivel=FUERA;
  lcd.clear();
}



//Selección de velocidad
void menuVelocidad() {
  switch (velocidadMaxima) {
    case 128: borrarEscribirLCD("CAMBIANDO A","VELOCIDAD: 160");
       cambiarVelocidad(160);break;
    case 160: borrarEscribirLCD("CAMBIANDO A","VELOCIDAD: 192");
       cambiarVelocidad(192);break;
    case 192: borrarEscribirLCD("CAMBIANDO A","VELOCIDAD: 224");
```

```
        cambiarVelocidad(224);break;
      case 224: borrarEscribirLCD("CAMBIANDO A","VELOCIDAD: 255");
        cambiarVelocidad(255);break;
      case  255:
      default: borrarEscribirLCD("CAMBIANDO A","VELOCIDAD: 128");
        cambiarVelocidad(128);break;
    }
    delay(1000);
    nivel=FUERA;
    lcd.clear();
}

void cambiarVelocidad(int nuevaVelocidad) {
      velocidadMaxima=nuevaVelocidad;//De 0-255  Me voy a mover en 128-160-192-224-25
    velocidadAuxiliar=velocidadMaxima;

    switch (velocidadMaxima) {
      case 128:  delayGiroLocoI=130;  delayGiroLocoD=160;
        limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=100
        break;
      case 160:  delayGiroLocoI=140;  delayGiroLocoD=170;
        limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=100
        break;
      case 192:  delayGiroLocoI=150;  delayGiroLocoD=180;
        limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=125;
        break;
      case 224:  delayGiroLocoI=152;  delayGiroLocoD=180;
        limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=200
        break;
      case 255:  delayGiroLocoI=155;  delayGiroLocoD=180;
        limiteGiro=170-velocidadMaxima*2/5; randomGiro=limiteGiro; tiempoSalvacion=275
        break;
    }
}



//Una opcion del menu para probar cosas nuevas
void menuBancoPruebas(){
    delay(2000);

    borrarEscribirLCD("EMPUJANDO A 255","");
     moverAdelantePWM(80);delay (120);
     moverAdelantePWM(255);delay(400);
    moverFrenar();

    delay(1000);
    moverFrenar();
    nivel=FUERA;
    lcd.clear();
}



//Distintas opciones para el menu de sonido
```

```
void menuSonido() {
  int lecturaT=analogRead(sharpT);
  int umbral=200;
  if (lecturaT<umbral) {
  }
  else {
    tone(altavoz,(lecturaT-umbral)*25+100,5);
    delay(5);
  }
}



//Aplicacion que enciende o apaga el uso del sensor Trasero
void menuSensorTrasero() {
  if (sensorTrasero) {
    sensorTrasero=0;
    borrarEscribirLCD("ENCENDIENDO EL","SENSOR TRASERO");
  }
  else {
    sensorTrasero=1;
    borrarEscribirLCD("APAGANDO EL","SENSOR TRASERO");
  }
  delay(1000);
  nivel=FUERA;
  lcd.clear();
}
```

```
/*
PULSACIÓN DE BOTÓN
Esta función va a cambiar el valor de las variables globales de flujo
en función de los botones pulsados y del contexto
*/
void pulsacionBoton (){

  //Boton siguiente
  if (digitalRead(btnSiguiente)==LOW) {
    switch (modo) {
      case COMBATE:
      //Para no salir del modo combate accidentalmente comprobamos
      //que de verdad se ha pulsado el boton
      delay(delayCombate);
      if (digitalRead(btnSiguiente)==LOW) transicionMenu();
      break;

      case MENU:
      //El boton siguiente solo actua si no estamos dentro de una aplicacion
      if (nivel==FUERA) {
        //enum eMenu {ESTRATEGIA, SHARP, CNY70, BATERIA};
        switch (menu) {
          case ESTRATEGIA:  menu=VELOCIDAD;break;
          case VELOCIDAD:  menu=PRUEBAESTRATEGIA;break;
          case PRUEBAESTRATEGIA:  menu=SENSORTRASERO;break;
          case SENSORTRASERO:  menu=SHARP;break;
          case SHARP:  menu=SHARP2;break;
          case SHARP2:  menu=RADAR;break;
          case RADAR:  menu=CNY70;break;
          case CNY70:  menu=BATERIA;break;
          case BATERIA:  menu=PRUEBAMOTOR;break;
          case PRUEBAMOTOR:  menu=PRUEBAHUIR;break;
          case PRUEBAHUIR:  menu=VERTICAL;break;
          case VERTICAL:  menu=HORIZONTAL2;break;
          case HORIZONTAL2:  menu=BANCOPRUEBAS;break;
          case BANCOPRUEBAS:  menu=LUZ;break;
          case LUZ:  menu=FANTASTICO;break;
          case FANTASTICO:  menu=SONIDO;break;
          case SONIDO:  menu=RESET;break;
          case RESET:  menu=ESTRATEGIA;break;
        }
        delay(delayBoton);
      }
      break;
    }
  }
  //Boton Intro
  if (digitalRead(btnIntro)==LOW) {
    switch (modo) {
      case COMBATE:
      //Para no salir del modo combate accidentalmente comprobamos
      //que de verdad se ha pulsado el boton
      delay(delayCombate);
      if (digitalRead(btnIntro)==LOW) transicionMenu();
      break;
```

```
        case MENU:
        //El boton Intro solo actua si no estamos dentro de una aplicacion
        if (nivel==FUERA) {
          nivel=DENTRO;
          lcd.clear();
          delay(delayBoton);
        }
        break;
      }
    }
    //Boton Atras
    if (digitalRead(btnAtras)==LOW) {
      switch (modo) {
        case COMBATE:
        //Para no salir del modo combate accidentalmente comprobamos
        //que de verdad se ha pulsado el boton
        delay(delayCombate);
        if (digitalRead(btnAtras)==LOW) transicionMenu();
        break;

        case MENU:
        switch (nivel) {
          case  DENTRO:
          nivel=FUERA;
          lcd.clear();
          delay(delayBoton);
          break;

          case FUERA:
          modo=COMBATE;
          transicionCombate();
          break;

        }
        break;
      }
    }

}
```

```
/*
TRANSICIONES
En esta pestaña desarrollamos las transiciones
del modo menu al combate y viceversa
Tambien los mensajes de inicio de programa
*/

//Transicion a modo combate. Esperamos los 5 segundos
//y luego colocamos el robot en posición horizontal
void transicionCombate () {
  secuenciaInicio2();
}

void secuenciaInicio2() {
//  unsigned long time=millis();
  tone(altavoz,1200,100);
  unsigned long pitido=millis();
  lcd.clear();
  for (int i=0;i<=15;i++) {
    for (int j=0;j<58;j++) {
      delay(5);
      if ((millis()-pitido)>1000) {
        pitido=millis();
        tone(altavoz,1200,100);
      }
    }
    goTo(i); lcd.print(">");
    goTo(16+i); lcd.print(">");
    delay(4);
  }
  movimientoAnterior=ADELANTE;
  lcd.clear();
  tiempoLento=millis();
  delay(12);
  tone(altavoz,400,400);

  caidaForzada();
}

//Transicion a modo menu.
//Damos a las variables globales los valores adecuados
void transicionMenu () {
  moverFrenar();
  modo=MENU;
  menu=ESTRATEGIA;
  nivel=FUERA;
  lcd.clear();
  velocidadMaxima=velocidadAuxiliar;
  delay(500);  //Este dalay es para evitar que volvamos otra vez al
  //modo combate por tener demasiado tiempo apretado el boton salir.
}

void mensajeInicio4() {
  lcd.clear();
  delay(500);
  borrarEscribirLCD(" SUMO  SOFTWARE "," VERSION V42");
  delay(500);
  lcd.clear();
}
```

```
void caidaForzada() {
   //Movemos hacia atras para hacer caer
    moverAdelantePWM(255);delay(5);
  moverAtrasPWM(255);
  delay(143);
  moverAdelantePWM(255);
  delay(75);
}
```

```
/*
FUNCIONES ESCRIBIR LCD
Estas funciones se utilizan para trabajar con las serLCD,
No hay que usar lcd.println porque surgen simbolos raros en pantalla
Me dice que en Arduino 1.0 no se puede hacer    lcd.print(0xFE, BYTE  );
se hace simplemente Serial.write(0xFE);
No se pueden escribir más de 16 caracteres por línea

*/


//Esta funcion escribe dos lineas.
//Si no llegamos a los 16 caracteres y habia algo no sera borrado.
//Es más rápida que borrar la pantalla y luego escribir.
void escribirLCD (String  linea1,String linea2) {
  lcd.selectLine(1);
  lcd.print(linea1);
  lcd.selectLine(2);
  lcd.print(linea2);
}

//Esta función primero borra la pantalla y luego escribe,
//es más lenta por eso deberá ser usada solo en el menú
//Nunca usarla en combate porque perderemos eficiencia
void borrarEscribirLCD (String  linea1,String linea2) {
  lcd.clear();
  lcd.selectLine(1);
  lcd.print(linea1);
  lcd.selectLine(2);
  lcd.print(linea2);
}

void goTo(int posicion) {
  if (posicion<16) {
    lcd.setCursor(1,posicion+1);
  }
  else {
    lcd.setCursor(2,posicion-15);
  }
}
```

```
/*
CONTROL MOTORES
En esta pestaña voy a definir las funciones necesarias para controlar los motores
Las funciones creadas son
moverAdelante()
moverAtras()
moverFrenar()
girarDerecha()
girarIzquierda()

moverAdelantePWM(byte duty)
moverAtrasPWM(byte duty)
girarDerechaPWM(byte duty,int radio)
girarIzquierdaPWM(byte duty,int radio)
girarDerechaAtrasPWM(byte duty,int radio)
girarIzquierdaAtrasPWM(byte duty,int radio)

girarDerechaFluido(byte duty)
girarIzquierdaFluido(byte duty)
girarDerechaAtrasFluido(byte duty)
girarIzquierdaAtrasFluido(byte duty)


motorIAtrasPWM(duty);
motorDAtrasPWM(duty);
motorIAdelantePWM(duty);
motorDAdelantePWM(duty);


*/

//*****************************************************************************
//Las siguientes funciones mueven el robot a máxima velocidad
//*****************************************************************************
void moverAdelante() {
  motorIAdelante();
  motorDAdelante();
}
void moverAtras() {
  motorIAtras();
  motorDAtras();
}
void moverFrenar() {
  motorIFrenar();
  motorDFrenar();
}
void girarDerecha(){
  motorIAdelante();
  motorDAtras();
}
void girarIzquierda(){
  motorIAtras();
  motorDAdelante();
}
```

```
//*****************************************************************************
//Las siguientes funciones mueven el robot a la velocidad deseada
//*****************************************************************************
void moverAdelantePWM(byte duty) {
  motorIAdelantePWM(duty);
  motorDAdelantePWM(duty);
}
void moverAtrasPWM(byte duty) {
  motorIAtrasPWM(duty);
  motorDAtrasPWM(duty);
}
//Gira a la derecha a una velocidad determinada por el duty
//con el radio de giro deseado
void girarDerechaPWM(byte duty,int  radio)   {//el radio se da en cm
  //duty determian la velocidad de giro de la rueda izquierda,
  //  tenemos que calcular la derecha
  int   N2=duty*(2*radio-8)/(2*radio+8);
  //8 son los 8 cm que hay entre rueda y rueda.
  byte duty2;
  motorIAdelantePWM(duty);
  if (N2>=0){
    duty2=N2;
    motorDAdelantePWM(duty2);
  }
  else {
    duty2=-N2;
    motorDAtrasPWM(duty2);
  }

}

//Gira a la derecha a una velocidad determinada por el duty
//con el radio de giro deseado
void girarIzquierdaPWM(byte duty,int  radio)   {//el radio se da en cm
  //duty determian la velocidad de giro de la rueda izquierda,
  //tenemos que calcular la derecha
  int   N2=duty*(2*radio-8)/(2*radio+8);//8 son los 8 cm que hay entre rueda y rueda
  byte duty2;
  motorDAdelantePWM(duty);
  if (N2>=0){
    duty2=N2;
    motorIAdelantePWM(duty2);
  }
  else {
    duty2=-N2;
    motorIAtrasPWM(duty2);
  }

}

void girarDerechaAtrasPWM(byte duty,int  radio)   {//el radio se da en cm
  //duty determian la velocidad de giro de la rueda izquierda,
  //tenemos que calcular la derecha
  int   N2=duty*(2*radio-8)/(2*radio+8);//8 son los 8 cm que hay entre rueda y rueda.
  byte duty2;
```

```
  motorIAtrasPWM(duty);
  if (N2>=0){
    duty2=N2;
    motorDAtrasPWM(duty2);
  }
  else {
    duty2=-N2;
    motorDAdelantePWM(duty2);
  }

}


void girarIzquierdaAtrasPWM(byte duty,int  radio)   {//el radio se da en cm
  //duty determian la velocidad de giro de la rueda izquierda,
  //tenemos que calcular la derecha
  int   N2=duty*(2*radio-8)/(2*radio+8);//8 son los 8 cm que hay entre rueda y rueda.
  byte duty2;
  motorDAtrasPWM(duty);
  if (N2>=0){
    duty2=N2;
    motorIAtrasPWM(duty2);
  }
  else {
    duty2=-N2;
    motorIAdelantePWM(duty2);
  }

}


//*****************************************************************************
//Las siguientes funciones mueven los motores a máxima velocidad
//(Son usadas por las anteriores funciones)
//*****************************************************************************
//Hace girar el motor izquierdo a toda velocidad hacia delante
void motorIAdelante() {
  digitalWrite(motorI1,HIGH);
  digitalWrite(motorI2,LOW);
}
//Hace girar el motor derecho a toda velocidad hacia delante
void motorDAdelante() {
  digitalWrite(motorD1,HIGH);
  digitalWrite(motorD2,LOW);
}
//Hace girar el motor izquierdo a máxima velocidad hacia atras
void motorIAtras() {
  digitalWrite(motorI1,LOW);
  digitalWrite(motorI2,HIGH);
}
//Hace girar el motor derecho a máxima velocidad hacia atras
void motorDAtras() {
  digitalWrite(motorD1,LOW);
  digitalWrite(motorD2,HIGH);
}
```

```
//Frena el motor izquierdo
void motorIFrenar() {
  digitalWrite(motorI1,LOW);
  digitalWrite(motorI2,LOW);
}
//Frena el motor izquierdo
void motorDFrenar() {
  digitalWrite(motorD1,LOW);
  digitalWrite(motorD2,LOW);
}



//*****************************************************************************
//Las siguientes funciones mueven los motores utilizando PWM
//(Son usadas por las anteriores funciones)
//*****************************************************************************
//Hace girar el motor izquierdo hacia adelante a una velocidad proporcional al duty.
//Si duty=0 esta parado, si es 255 va a tope.
void motorIAdelantePWM(byte duty) {
  analogWrite(motorI1,duty);
  digitalWrite(motorI2,LOW);
}
//Hace girar el motor derecho hacia adelante a una velocidad proporcional al duty.
//Si duty=0 esta parado, si es 255 va a tope.
void motorDAdelantePWM(byte duty) {
  analogWrite(motorD1,duty);
  digitalWrite(motorD2,LOW);
}
//Hace girar el motor izquierdo hacia atras a una velocidad proporcional al duty.
//Si duty=0 esta quieto, si 255 va a tope.
void motorIAtrasPWM(byte duty) {
    duty=255-duty;//Restamos para que cuando duty de entrada sea 255, este el I1 a 0.
  analogWrite(motorI1,duty);
  digitalWrite(motorI2,HIGH);
}
//Hace girar el motor derecho a máxima velocidad hacia atras
void motorDAtrasPWM(byte duty) {
  duty=255-duty;
  analogWrite(motorD1,duty);
  digitalWrite(motorD2,HIGH);
}

//*****************************************************************************
//Las siguientes funciones sirven para suavizar el cambio de velocidad de los motores
//y evitar que se levante el morro
//*****************************************************************************

void arrancadoMotor1(byte n) {
  byte paso=velocidadMaxima/n;
  for (byte i=1;i<=n;i++) {
    moverAdelantePWM(paso*n);
    delay(1);
  }
}
```

```
void arrancadoMotor2(byte n) {
  byte velocidadUmbral=80;
  byte paso=(velocidadMaxima-velocidadUmbral)/n;
  for (byte i=1;i<=n;i++) {
    moverAdelantePWM(velocidadUmbral+paso*n);
    delay(1);
  }
}

void arrancadoMotor3(byte  n,byte d) {
  byte velocidadUmbral=90;
   moverAdelantePWM(velocidadUmbral);delay(d);
  byte paso=(velocidadMaxima-velocidadUmbral)/n;
  for (byte i=1;i<=n;i++) {
    moverAdelantePWM(velocidadUmbral+paso*n);
    delay(1);
  }
}


//*****************************************************************************
//Las siguientes funciones mueven el robot con movimientos fluidos
//*****************************************************************************

//Gira a la derecha a una con el motor derecho parado
void girarDerechaFluido(byte duty)  {
  motorIAdelantePWM(duty);
  motorDFrenar();
}

void girarIzquierdaFluido(byte  duty)   {
  motorDAdelantePWM(duty);
   motorIFrenar();
}

void girarDerechaAtrasFluido(byte duty) {
  motorIAtrasPWM(duty);
  motorDFrenar();
}

void girarIzquierdaAtrasFluido(byte duty) {
  motorDAtrasPWM(duty);
  motorIFrenar();
}
```

```
/*
ESTRATEGIA FLUIDA
Aquí voy a desarrollar una estrategia que haga movimientos más fluidos.
En vez de girar con radio 0 girar con más radio. Una rueda parada y la otra a tope.
Trabajamos sobre la base de la estrategia agresiva (ya descatalogada)

*/


void estrategiaFluido() {
  eCNY70 estado=sensoresSuelo();
  switch (estado) {
    case Z:  determinarDireccionFluido();break;
    default: maniobraEvasivaFluido(estado);
  }
}


//Funcion sensoresSuelo
eCNY70 sensoresSuelo() {
  //Primero leemos todos los sensores y guardamos sus valores
  boolean cny70DD=digitalRead(CNY70DD);
  boolean cny70TD=digitalRead(CNY70TD);
  boolean cny70DI=digitalRead(CNY70DI);
  boolean cny70TI=digitalRead(CNY70TI);
  //Ahora comprobamos que no esté todo el lado, porque si es así
  //y vamos para atras nos caemos
  if  (cny70DD&&cny70TD)return D;
  if  (cny70DI&&cny70TI)return I;
  //Ahora que no haya sensores delanteros activados
  if  (cny70DD)return DD;
  if  (cny70DI)return DI;
  //Luego los traseros. Se podría hacer que no siempre se leyesen
  //con un random por ejemplo
  if  (cny70TD)return     TD;//Los quito porque creo que dan problemas
  if  (cny70TI)return TI;
  //Si no hay que devolver Z
  return Z;
}


void determinarDireccionFluido() {
  int   umbral=200;//Normalmente vale 200
  unsigned long tiempo;
  velocidadMaxima=velocidadAuxiliar;
  //Primero comprobamos si hay algo delante
  int lecturaDD=analogRead(sharpDD);
  int lecturaDI=analogRead(sharpDI);
  if (lecturaDD>umbral)  contadorDD++;else contadorDD=0;
  if (lecturaDI>umbral)  contadorDI++;else contadorDI=0;
  if ((contadorDD>4)||contadorDI>4) {

    //Este bloque sirve para que no se encabrite el robot al pasar
    //de una situación con los motores
    //girando hacia atras a otra con los motores girando hacia adelante
```

```
  tiempo=millis();
  if (tiempo-tiempoAtras<300) {
    if (tiempo-tiempoAtras<200) {
      if (tiempo-tiempoAtras<100) {
        //Si ha pasado muy poco tiempo seguimos hacia atras pero más lentamente
        moverAtrasPWM(velocidadMinima);
        return;
      }
      //Despues ya frenamos
      moverFrenar();
      return;
    }
    else {
      //Antes de movernos hacia adelante al maximo avanzamos un poco mas lento
      velocidadMaxima=velocidadMinima;
    }
  }
  //Ponemos los contadores a cero para que no se produzcan detecciones fantasmas
  contadorD=0; contadorI=0; contadorT=0;
  //Por ultimo comprobamos si estamos en alguna transición
  //y tenemos que ir más lentos
  if (millis()-tiempoLento<110) velocidadMaxima=velocidadMinima;
  movimientoAnterior=ENCARANDO;
  gobiernaMotores(lecturaDD,lecturaDI);
  return; //Salimos porque ya hemos emitido la orden para los motores
}
//Si llegamos aquí es que no hay nada delante, hay que buscar en los otros lados
//Ahora hay que comprobar si hay algo a los lados o atras
eMovimientos movimientoActual=INDETERMINADO;
int lecturaI=analogRead(sharpI);
int lecturaD=analogRead(sharpD);
int lecturaT=analogRead(sharpT);
if (lecturaD>umbral) contadorD++;else contadorD=0;
if (lecturaI>umbral) contadorI++;else contadorI=0;
if (lecturaT>umbral+100+500*sensorTrasero) contadorT++;else contadorT=0;
//El orden de los siguientes if hará que se comporte de una forma u otra
if (contadorI>4) movimientoActual=IZQUIERDA;
if (contadorD>4) movimientoActual=DERECHA;
if   (contadorT>18)   movimientoActual=MEDIAVUELTA;
//Aqui es que de la vuelta, me da igual para un lado o para otro
switch (movimientoActual) {
  case IZQUIERDA:
    //Este bloque sirve para que no se encabrite el robot
    // al pasar de una situación con los motores girando hacia
    // atras a otra con los motores girando hacia adelante
     tiempo=millis();
    if (tiempo-tiempoAtras<300) {
      if (tiempo-tiempoAtras<200) {
        if (tiempo-tiempoAtras<100) {
          moverAtrasPWM(velocidadMinima);
          return;
        }
        moverFrenar();
        return;
      }
```

```
      else {
        velocidadMaxima=velocidadMinima;
      }
    }
    //Giramos a la izquierda e iniciamos la variable tiempoGirando
     girarIzquierdaFluido(velocidadMaxima); movimientoAnterior=IZQUIERDA;
    //borrarEscribirLCD(" DETECTADO "," IZQUIERDA");
    tiempoGirando=millis();  tiempoExtra=0;break;
  case DERECHA:
    //Este bloque sirve para que no se encabrite el robot
    // al pasar de una situación con los motores girando hacia
    // atras a otra con los motores girando hacia adelante
     tiempo=millis();
    if (tiempo-tiempoAtras<300) {
      if (tiempo-tiempoAtras<200) {
        if (tiempo-tiempoAtras<100) {
          moverAtrasPWM(velocidadMinima);
          return;
        }
        moverFrenar();
        return;
      }
      else {
        velocidadMaxima=velocidadMinima;
      }
     }
    //Giramos a la derecha e iniciamos la variable tiempoGirando
     girarDerechaFluido(velocidadMaxima); movimientoAnterior=DERECHA;
    //borrarEscribirLCD(" DETECTADO "," DERECHA");
    tiempoGirando=millis();tiempoExtra=0; break;
  case MEDIAVUELTA:
   //Este bloque sirve para que no se encabrite el robot
    // al pasar de una situación con los motores girando hacia
    // atras a otra con los motores girando hacia adelante
    tiempo=millis();
    if (tiempo-tiempoAtras<300) {
      if (tiempo-tiempoAtras<200) {
        if (tiempo-tiempoAtras<100) {
          moverAtrasPWM(velocidadMinima);
          return;
        }
        moverFrenar();
        return;
      }
      else {
        velocidadMaxima=velocidadMinima;
      }
    }
     //Giramos a la derecha, aunque podríamos girar igualmente a la izquierda
     //y calculamos el tiempo extra
     girarDerechaFluido(velocidadMaxima); movimientoAnterior=DERECHA;
    //borrarEscribirLCD(" DETECTADO "," ATRAS");
    tiempoGirando=millis();tiempoExtra=limiteGiro+random(randomGiro); break;
  //Si llegamos aquí quiere decir que el movimientoActual es INDETERMINADO,
  // los sensores no ven al contrario
```

```
default: {
  //Este switch es para parar de girar si llevamos mucho rato
  switch (movimientoAnterior) {
    case  ADELANTE:break;
    case  ENCARANDO:break;
    //Si llegamos aquí es que estamos girando
    default:
    //La siguiente condición mira si hemos pasado el tiempo asignado para girar
    if ((millis()-tiempoGirando)>limiteGiro+tiempoExtra) {
          movimientoAnterior=ADELANTE;
      //Esto marca el inicio del tiempo de transición de giro a adelante
      tiempoLento=millis();
      //Esto se hace para que no se ejecute la condición de transición
      //atras-adelante. Queremos transicion giro-adelante que es la anterior
        tiempoAtras=tiempoAtras-300;
      //borrarEscribirLCD(" ADELANTE "," BASTA GIRAR");
    }
    break;
  }
  //Ahora continuamos con el movimiento que teníamos antes
  switch (movimientoAnterior) {
    case IZQUIERDA:           girarIzquierdaFluido(velocidadMaxima);break;
    case ATRASIZQUIERDA:
      //Para pasar de atras-adelante girando hay que hacer esta transicion
      //Atras-girarAtrasFluido-girarAtrasPWM-Adelante
      if (millis()-tiempoAtras<100)  girarIzquierdaAtrasFluido(velocidadMaxima);
      else  girarIzquierdaAtrasPWM(velocidadMaxima,0);break;
    case DERECHA:             girarDerechaFluido(velocidadMaxima);break;
    case  ATRASDERECHA:
      //Para pasar de atras-adelante girando hay que hacer esta transicion
      //Atras-girarAtrasFluido-girarAtrasPWM-Adelante
      if (millis()-tiempoAtras<100)  girarDerechaAtrasFluido(velocidadMaxima);
      else  girarDerechaAtrasPWM(velocidadMaxima,0);break;
    case    ENCARANDO:
      //El caso de ENCARANDO tiene que tener en cuenta que le ha podido llegar
      //una situación de velocidades capadas por transición
      //Por lo tanto debe examinar si sigue estando en transición y si no lo
      //está elevar las velocidades manteniendo el ratio
      //Podría ser por una situación de tiempoAtras o de tiempoLento
      tiempo=millis();
      //Primero comprabamos si seguimos en tiempo de transición
      if    ((tiempo-tiempoAtras<300)||(tiempo-tiempoLento<110))    {
        motorIAdelantePWM(motorI); motorDAdelantePWM(motorD);
      }
      else {
        //Luego comprobamos si tenemos las velocidades capadas o no
        if ((motorI==velocidadAuxiliar)||(motorI==velocidadAuxiliar)){
          //Estos ifs son para evitar que se quede dando vueltas como un tonto
          if  (motorI<velocidadAuxiliar)  motorI++;
          if (motorD<velocidadAuxiliar) motorD++;
          motorIAdelantePWM(motorI); motorDAdelantePWM(motorD);
        }
        else {
          if (motorI>motorD) {
              motorD=(motorD+1)*velocidadAuxiliar/motorI;//El +1 es por si es cer
```

```
                motorI=velocidadAuxiliar;
                motorIAdelantePWM(motorI); motorDAdelantePWM(motorD);
              }
              else {
                  motorI=(motorI+1)*velocidadAuxiliar/motorD;//El +1 es por si es cero
                motorD=velocidadAuxiliar;
                motorIAdelantePWM(motorI); motorDAdelantePWM(motorD);
              }
            }
          }
           break;
        //case MEDIAVUELTA:         girarDerechaPWM(velocidadMaxima,0); break;
        case QUIETO:               moverFrenar();break;
        default:
          //Comprobamos que no estamos en transición y nos movemos adelante
          if (millis()-tiempoLento<110)     velocidadMaxima=velocidadMinima;
          moverAdelantePWM(velocidadMaxima);
      }
                      }
   }
}


//Esta función va a asignar un duty a cada motor en función
//de las lecturas de los sensores
//Hay que tener en cuenta que para entrar aquí ya han pasado la criba del umbral
void gobiernaMotores (int  xDD,int xDI) {

  //Ahora tengo que distinguir los casos de izquierda o derecha
  if (xDD>xDI) {
    //Tenemos tres franjas, una en la que lecturaDI es muy pequeña,
    //otra en la que va creciendo y una última en la que son muy parecidos.
    //Esta podría ser la condición de muy derecha
    //Hacemos que gire a la derecha
    if  (xDI<10)  {//En la derecha los valoresDI bajan muy rápido,
    //no es necesario un umbral
      motorIAdelantePWM(velocidadMaxima); motorI=velocidadMaxima;
      motorDAdelantePWM(0); motorD=0;
    }
    else {
      motorD=96-96*(xDD-xDI)/(xDD+xDI);
      //Esto sería cuando lo tengamos delante o a la izquierda
      //Hacemos que gire ligeramente a la izquierda
      if (motorD>90) {
        motorDAdelantePWM(velocidadMaxima); motorD=velocidadMaxima;
        motorI=velocidadMaxima*90/96;
        motorIAdelantePWM(motorI);
      }
      //Esto es cuando lo tenemos a la derecha
      //HAcemos que gire a la derecha
      else {
        motorIAdelantePWM(velocidadMaxima); motorI=velocidadMaxima;
        motorD=velocidadMaxima*motorD/96;
        motorDAdelantePWM(motorD);
      }
```

```
    }
  }
  else {
    //Primero ponemos el motor derecho a tope
    motorDAdelantePWM(velocidadMaxima); motorD=velocidadMaxima;
    //Esto es muy a la izquierda
    if  (xDD<60)  {//Cuando tenemos algo muy a la izquierda del sensor oscila mucho,
    //por eso el umbral lo ponemos mucho más alto aquí
      motorIAdelantePWM(0); motorI=0;
    }
    //Esto es entre muy a la izquierda y a la izquierda
    else {
      motorI=96-96*(xDI-xDD)/(xDD+xDI);
      //Esta condición la pongo como giro mínimo, concuerda con lo programado antes
      if (motorI>90) {
        motorI=velocidadMaxima*90/96;
        motorIAdelantePWM(motorI);
      }
      else {
        motorI=velocidadMaxima*motorI/96;
        motorIAdelantePWM(motorI);
      }
    }
  }

}


//Es igual que la otra solo que añadiendo la contabilizción de tiempo atras
void maniobraEvasivaFluido(eCNY70 sensorActivado) {
  switch (sensorActivado) {
    //Primero los sensores delanteros
    case  DD:case DI:
    moverAtrasPWM(255);
    delay(tiempoSalvacion);  //Esto es porque se estaba saliendo un poco
    while (digitalRead(CNY70DD)||digitalRead(CNY70DI)) {
      moverAtrasPWM(255);
    }
    //moverFrenar();  //Esto es para evitar las sacudidas
    if (sensorActivado==DD) {
      movimientoAnterior=ATRASDERECHA;
      //borrarEscribirLCD(" ATRAS "," DERECHA");
      tiempoGirando=millis();
      tiempoExtra=115;
    }
    else {
      movimientoAnterior=ATRASIZQUIERDA;
      //borrarEscribirLCD(" ATRAS "," IZQUIERDA");
      tiempoGirando=millis();
      tiempoExtra=115;
    }
    tiempoAtras=millis();  //Hacemos esto para que no se levante al girar
    break;
    //Ahora los casos en los que tenemos todo un lado en lo blanco
    //Lo que hago es sacar la parte delantera para que pueda ir luego para adelante.
```

```
    //Volvería a entrar solo con el sensor trasero activado.
    case D:
    while (digitalRead(CNY70DD)) {
      girarIzquierdaFluido(velocidadAuxiliar);
    }
    movimientoAnterior=ADELANTE;
    break;
    case I:
    while (digitalRead(CNY70DI)) {
      girarDerechaFluido(velocidadAuxiliar);
    }
    movimientoAnterior=ADELANTE;
    break;

    //luego ya los traseros
    case  TD:case TI:
    while (digitalRead(CNY70TD)||digitalRead(CNY70TI)) {
      moverAdelantePWM(velocidadAuxiliar);
    }
    movimientoAnterior=ADELANTE;
    break;
    default: break;
  }
}
```

```
/*
ESTRATEGIA SUICIDA
La estrategia suicida es aquella que busca echar al enemigo a toda costa.
Si tenemos al enemigo delante aun en el borde seguiremos empujando
Esto no parece muy recomendable pero podría pasar qeu el adversario tuviese
una rampa plana blanca para activar nuestros cny70. En ese caso sin la
estrategia suicida sería muy dificil vencerle.
La base va a ser la agresiva y la fluida
*/
void estrategiaSuicida() {
  eCNY70 estado=sensoresSueloSuicida();
  switch (estado) {
    case Z:  determinarDireccionFluido();break;
    default: maniobraEvasivaFluido(estado);
  }
}

//Funcion sensoresSuelo
eCNY70 sensoresSueloSuicida() {
  //Primero leemos todos los sensores y guardamos sus valores
  boolean cny70DD=digitalRead(CNY70DD);
  boolean cny70TD=digitalRead(CNY70TD);
  boolean cny70DI=digitalRead(CNY70DI);
  boolean cny70TI=digitalRead(CNY70TI);
  //Ahora comprobamos que no esté todo el lado,
  //porque si es así y vamos para atras nos caemos
  if  (cny70DD&&cny70TD)return D;
  if  (cny70DI&&cny70TI)return I;
  //Ahora que no haya sensores delanteros activados
  if (cny70DD) {
    int lecturaDD=analogRead(sharpDD);
    int lecturaDI=analogRead(sharpDI);
    if  ((lecturaDD>300)||(lecturaDI>300))return Z;
    else return DD;
  }
  if  (cny70DI)  {
    int lecturaDD=analogRead(sharpDD);
    int lecturaDI=analogRead(sharpDI);
    if  ((lecturaDD>300)||(lecturaDI>300))return Z;
    else return DI;
  }
  //Luego los traseros. Se podría hacer que no siempre
  //se leyesen con un random por ejemplo
  if  (cny70TD)return     TD;//Los quito porque creo que dan problemas
  if  (cny70TI)return TI;
  //Si no hay que devolver Z
  return Z;
}
```

```
/*
ESTRATEGIA COBARDE
Aqui se encuentra la estrategia cobarde que huye de todo lo que
tenga cerca, se basa en otras estrategias anteriores
*/

void estrategiaCobarde() {
  cocheFantastico2(contadorFantastico);
  if (contadorFantastico==0) direccionFantastico=1;
  if (contadorFantastico==15) direccionFantastico=0;
  if (direccionFantastico) contadorFantastico++;
  else contadorFantastico--;

  determinarDireccionCobarde();
}


void determinarDireccionCobarde() {
  int   umbral=200;//Normalmente vale 200
  unsigned long tiempo;
  velocidadMaxima=velocidadAuxiliar;

  //Primero leemos todos los sensores
  int lecturaDD=analogRead(sharpDD);
  int lecturaDI=analogRead(sharpDI);
  int lecturaI=analogRead(sharpI);
  int lecturaD=analogRead(sharpD);
  int lecturaT=analogRead(sharpT);
  if (lecturaDD>umbral)  contadorDD++;else contadorDD=0;
  if (lecturaDI>umbral)  contadorDI++;else contadorDI=0;
  if (lecturaD>umbral)  contadorD++;else contadorD=0;
  if (lecturaI>umbral)  contadorI++;else contadorI=0;
  if (lecturaT>umbral)  contadorT++;else contadorT=0;
  //Ahora vemos cuales están activados
  boolean bA=(contadorDD>4)||(contadorDI>4);
  boolean bD=(contadorD>4);
  boolean bI=(contadorI>4);
  boolean bT=(contadorT>4);


  //Rodeado o no hay nadie
  if ((bD&&bI&&bA&&bT)||!(bD||bI||bA||bT)) {
    //Transicion
    tiempo=millis();
    if (tiempo-tiempoAtras<100) {
      //Si ha pasado muy poco tiempo seguimos hacia atras pero más lentamente
      moverAtrasPWM(velocidadMinima);
      return;
    }
    movimientoAnterior=QUIETO;
    tiempoLento=millis(); tiempoAtras=0;
      moverFrenar();
    //Esto podría sustituirse por movimientos aleatorios típicos de un animal
    return;
  }
```

```
  //Enemigo detrás
  if (!bA&&bT) {
    //Transicion Atras->Adelante
    tiempo=millis();
    if (tiempo-tiempoAtras<300) {
      if (tiempo-tiempoAtras<200) {
        if (tiempo-tiempoAtras<100) {
          //Si ha pasado muy poco tiempo seguimos hacia atras pero más lentamente
          moverAtrasPWM(velocidadMinima);
          return;
        }
        //Despues ya frenamos
        moverFrenar();
        return;
      }
      else {
        //Antes de movernos hacia adelante al maximo avanzamos un poco mas lento
        velocidadMaxima=velocidadMinima;
      }
    }
    //Transicion quieto->adelante
    if (millis()-tiempoLento<110) velocidadMaxima=velocidadMinima;
    movimientoAnterior=ADELANTE;
    moverAdelantePWM(velocidadMaxima);
    return;
  }

  //Enemigo delante
  if (bA&&!bT) {
    movimientoAnterior=ATRAS;
    tiempoAtras=millis(); tiempoLento=0;
    moverAtrasPWM(velocidadMaxima);
    return;
  }
  //Enemigo a los lados
  if (bD&&bI) {
    movimientoAnterior=ATRAS;
    moverAtrasPWM(velocidadMaxima);
    return;
  }

  //Enemigo en un lado
  if (bI) {
    movimientoAnterior=ATRASDERECHA;
    girarDerechaAtrasFluido(velocidadMaxima);
  }
  else {
  //if (bD) {
    movimientoAnterior=ATRASIZQUIERDA;
    girarIzquierdaAtrasFluido(velocidadMaxima);
  }

}
```

```
void movimientoAleatorio() {
  int i=random(4);
  if (i==0) numeroAleatorio=random(8);
  switch(numeroAleatorio) {
    case 0:  girarDerechaAtrasFluido(50);break;
    case 1:  girarIzquierdaAtrasFluido(50);break;
    case 2:  girarDerechaFluido(50);break;
    case 3:  girarIzquierdaFluido(50);break;
    default: moverFrenar();
  }
}
```

```
/*
VUELTAS AL RING
Aquí voy a definir la estrategia que me permita dar vueltas al ring
entre la parte blanca y la negra
Me voy a apoyar sobre la base de la estrategia agresiva
*/

void estrategiaVueltas() {
  //Primero vemos como están los sensores del suelo
  eCNY70 estado=sensoresSuelo();
  //Luego actuamos en consecuencia
  vueltasRing(estado);
}

void vueltasRing(eCNY70 sensorActivado) {
  //Con radios de 8 y 30 funciona bien, se queda dentro del circulo sin salir
  switch (sensorActivado) {
    //Caso 1. Solo sensor delantero activado
    case DD:
      girarIzquierdaPWM(velocidadMaxima,10);
      break;
    //Caso 2. Solo sensor trasero activado
    case TD:
      girarIzquierdaPWM(velocidadMaxima,32);
      break;
    //Caso 3. Ambos sensores activados
    case D:
      girarIzquierdaPWM(velocidadMaxima,10);
      break;
    //Caso 4. Ningún sensor activado
    default:
          girarIzquierdaPWM(velocidadMaxima,32);//(75-10)/2=32.5
      break;
  }
}
```

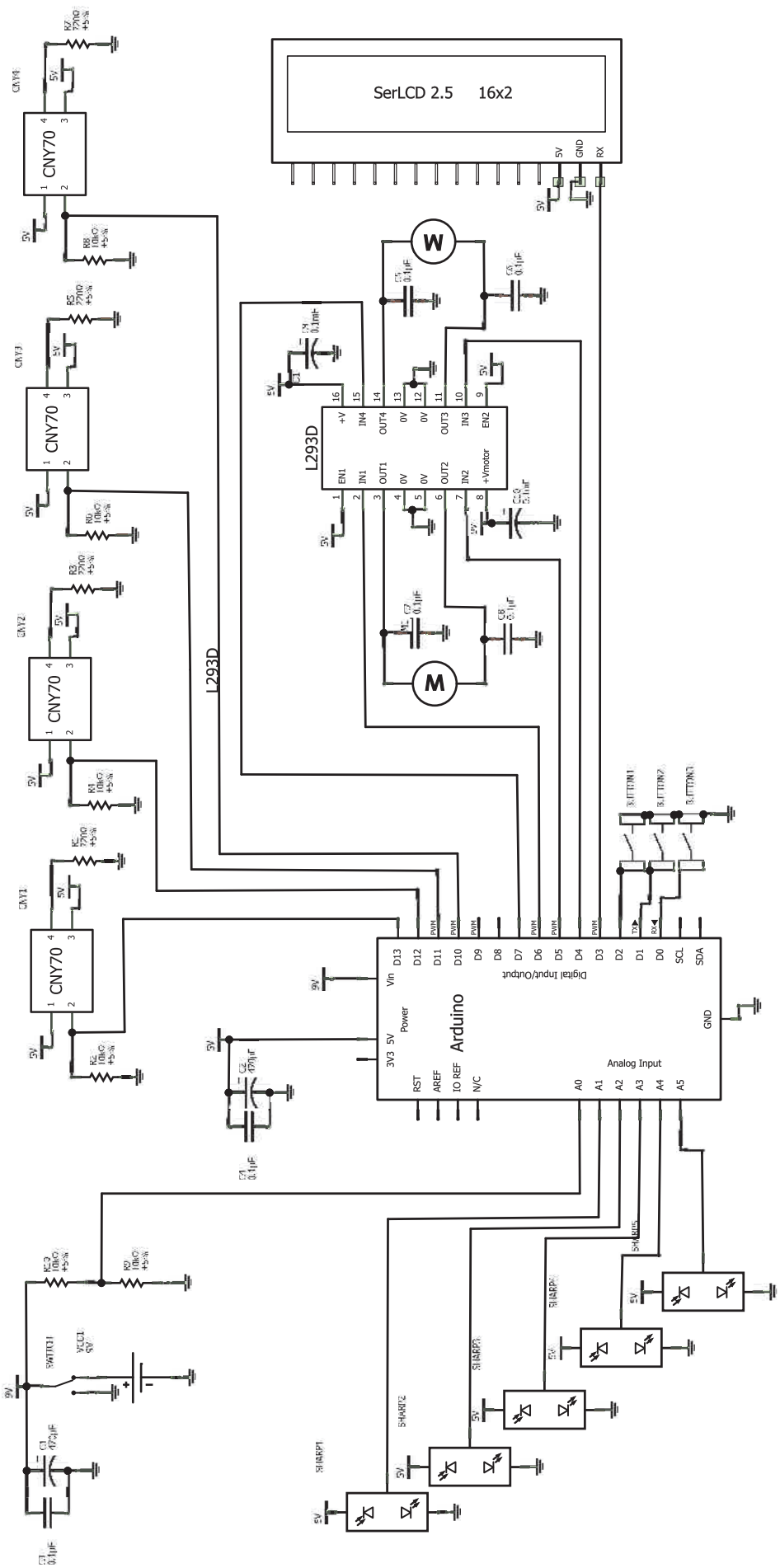
```
/*
WALLRACE
Aquí voy a desarrollar el modo de funcionamiento de ir andando cerca de la pared,
me voy a apoyar en la estrategia fluido

*/
void estrategiaWallRace() {
  cocheFantastico2(contadorFantastico);
  if (contadorFantastico==0) direccionFantastico=1;
  if (contadorFantastico==15) direccionFantastico=0;
  if (direccionFantastico) contadorFantastico++;
  else contadorFantastico--;

  determinarDireccionWallRace();
}


void determinarDireccionWallRace() {
  //Variables que determinan el funcionamiento del programa
  int   umbral=200;//Normalmente vale 200
  int distanciaPared=300;
  int radioMaximo=40;

  velocidadMaxima=velocidadAuxiliar;
  //Primero comprobamos si hay algo delante
  int lecturaDD=analogRead(sharpDD);
  int lecturaD=analogRead(sharpD);
  if  (lecturaDD>umbral)  contadorDD++;else contadorDD=0;
  //Si tenemos un obstaculo delante giramos a la izquierda y salimos
  if ((contadorDD>4)) {
    girarIzquierdaFluido(velocidadMaxima);
    return;
  }
  //Si no hay nada delante tratamos de mantenernos junto a la pared
  int   diferencia=lecturaD-distanciaPared;//Esto estará entre -300 y 300 aproximadame
   diferencia=diferencia/10;//Ahora entre 30 y -30
  //Caso de que estamos en el rango querido
  if ((diferencia<5)&&(diferencia>-5)) {
    moverAdelantePWM(velocidadMaxima);
    return;
  }
  if (diferencia>0) {
    girarIzquierdaPWM(velocidadMaxima,radioMaximo-diferencia);
  }
  else {
    girarDerechaPWM(velocidadMaxima,radioMaximo+diferencia);
  }
}
```

```
/*
BONUS
En esta pestaña voy a poner cosas extra como por ejemplo el efecto coche fantástico
*/


void cocheFantastico2 (int n) {
  switch (n) {
    case 0:
    goTo(0);
    lcd.print("# ");
    goTo(16);
    lcd.print("# ");
    break;
    case 15:
    goTo(14);
    lcd.print(" #");
    goTo(30);
    lcd.print(" #");
    break;
    default:
    goTo(n-1);
    lcd.print(" # ");
    goTo(n+15);
    lcd.print(" # ");
  }
}

int divisionEntera(int  dividendo,int divisor) {
  int resto=dividendo%divisor;
  return((dividendo-resto)/divisor);
}

// Restarts program from beginning but does not reset the peripherals and registers
void softwareReset()
{
asm volatile ("  jmp 0");
}
```

Capítulo 49

Esquema eléctrico

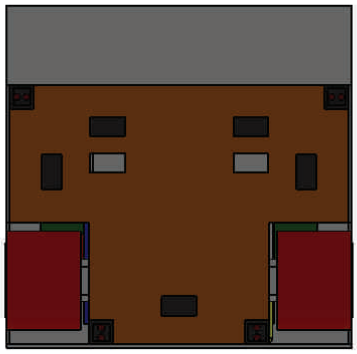
A continuación se muestra el esquema eléctrico del robot. Ha sido elaborado utilizando el software de Fritzing.

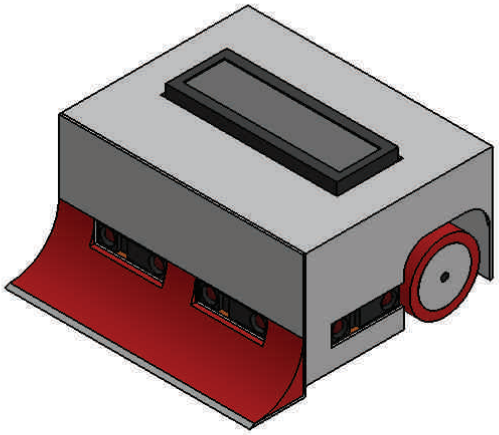


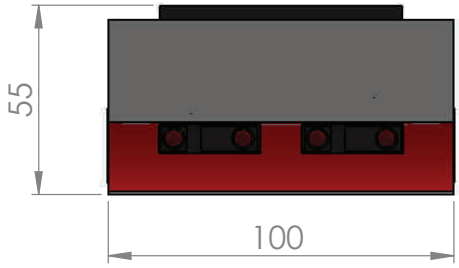
Capítulo 50

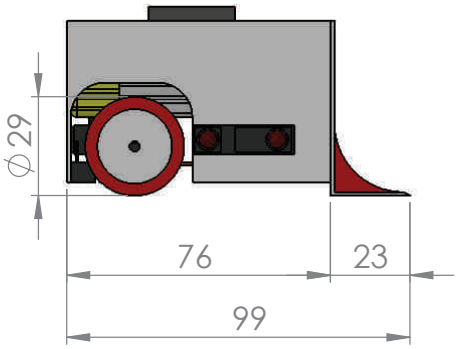
Planos del robot

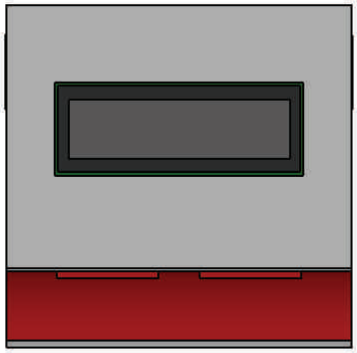
A continuación se pueden ver los distintos prototipos diseñados y por último los planos para construir el chasis y la carrocería de la versión final del robot.

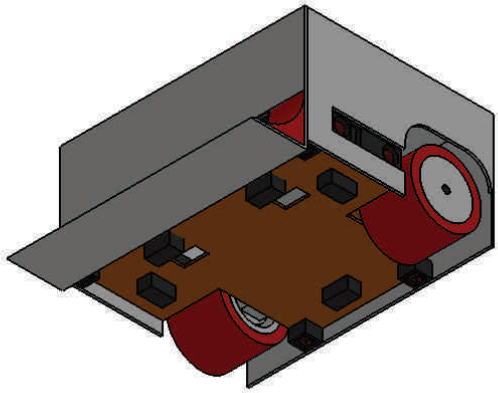




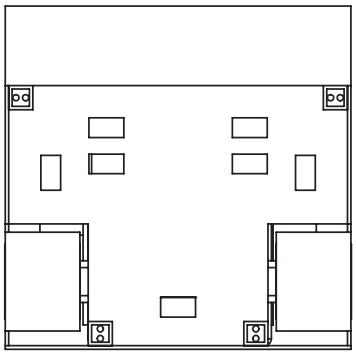


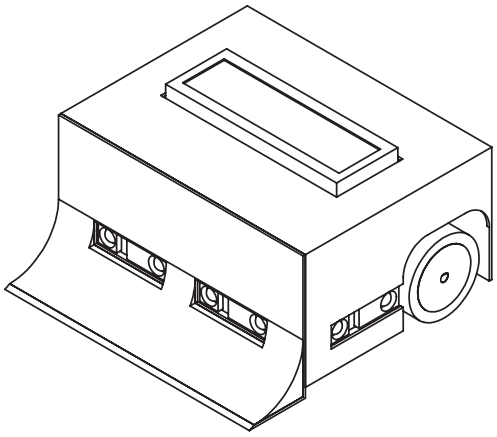


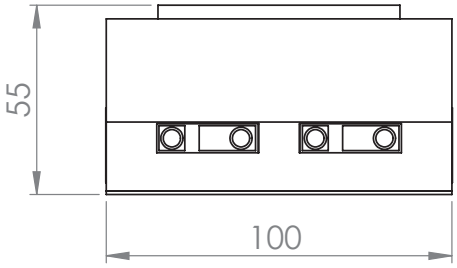


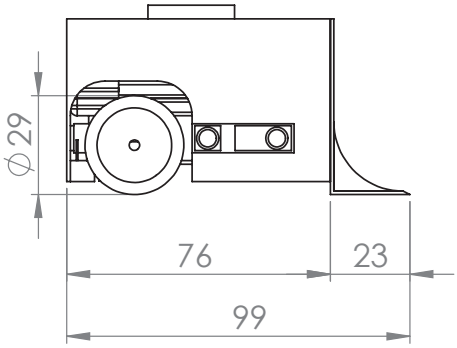


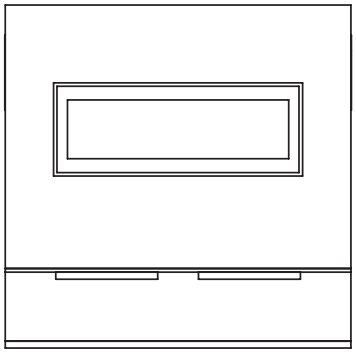
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Mark II	
				PESO:		ESCALA:1:2	HOJA 1 DE 1

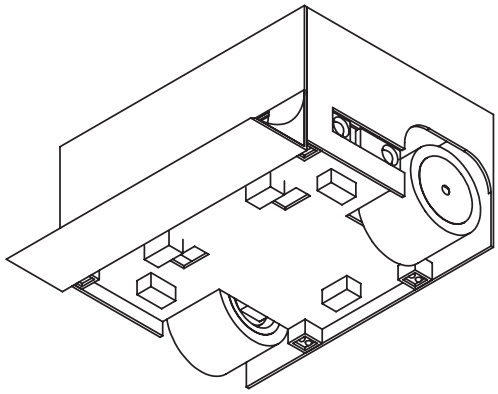




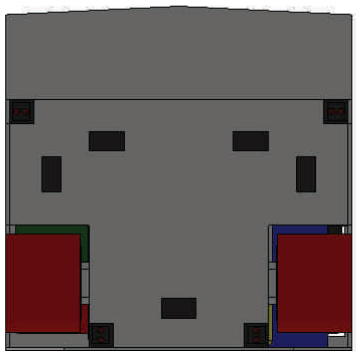


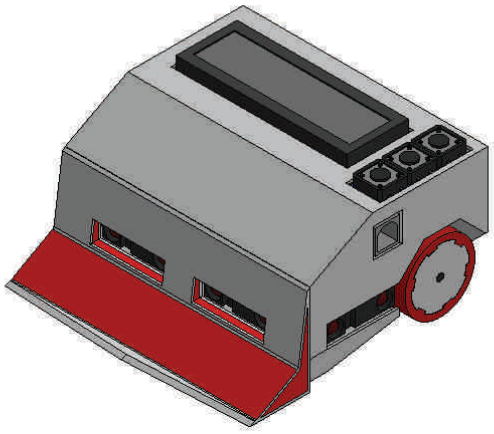


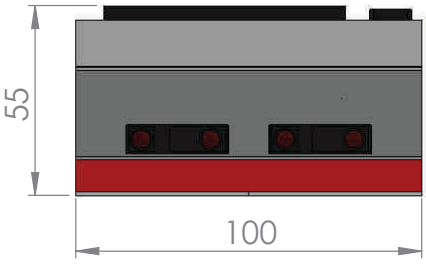


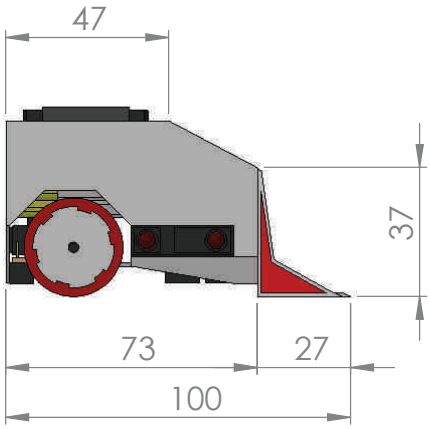


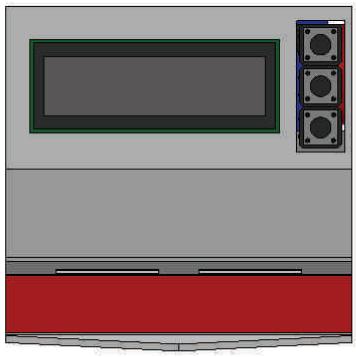
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Mark II	
				PESO:		ESCALA:1:2	HOJA 1 DE 1

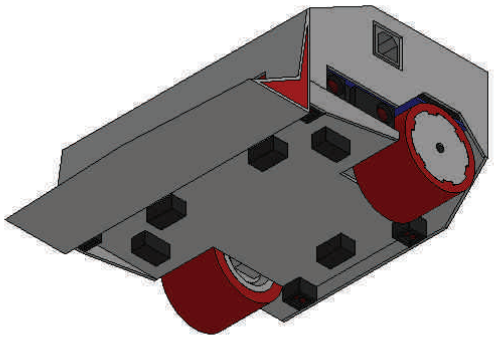




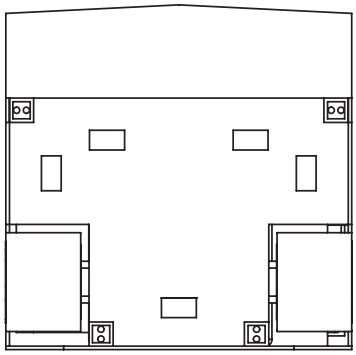


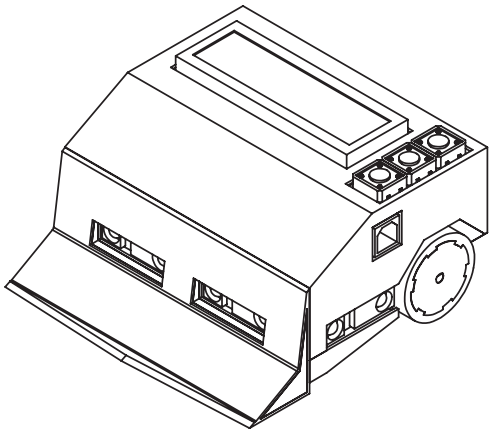


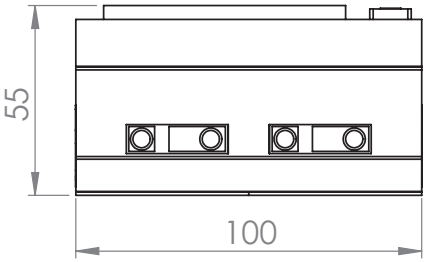


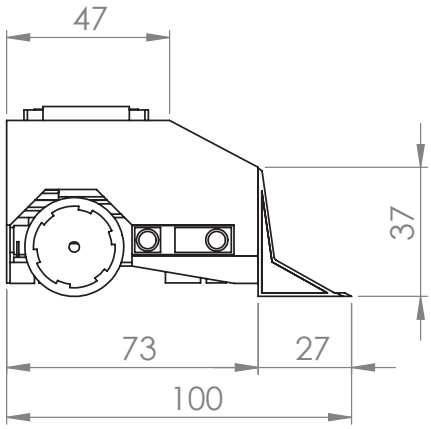


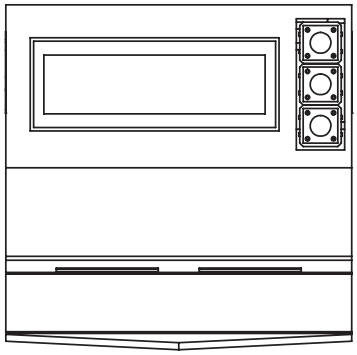
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA		REVISIÓN
	NOMBRE	FIRMA	FECHA				TÍTULO:
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Mark III	
				PESO:		ESCALA:1:2	HOJA 1 DE 1

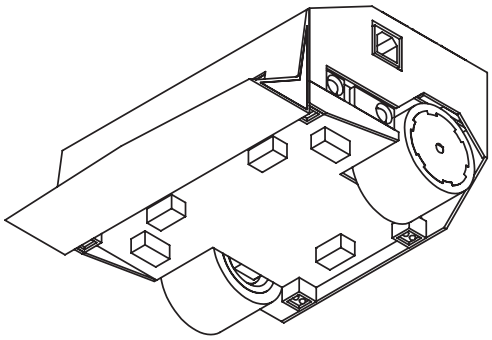






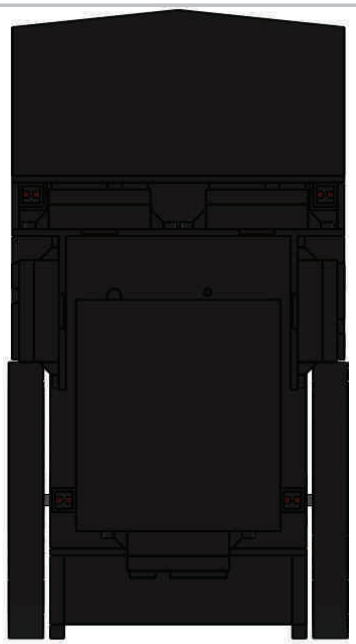


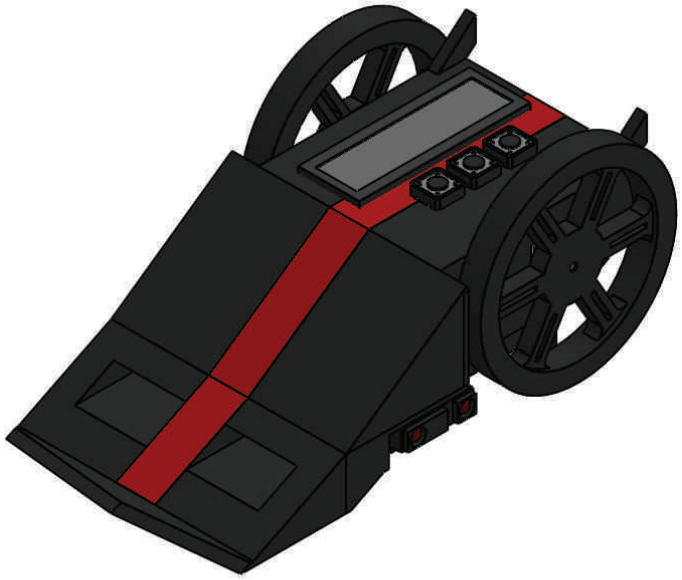


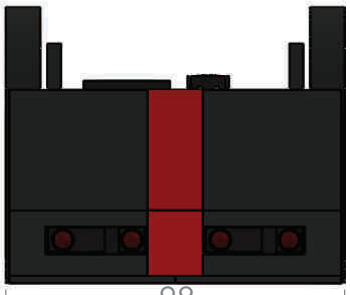


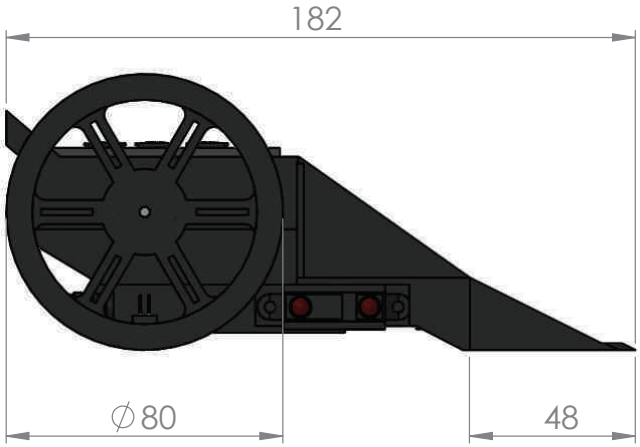
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA		REVISIÓN	
	NOMBRE	FIRMA	FECHA				TÍTULO:
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
				PESO:		ESCALA:1:2	HOJA 1 DE 1

Mark III

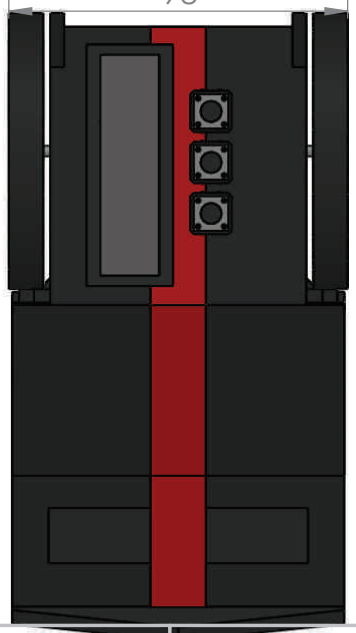


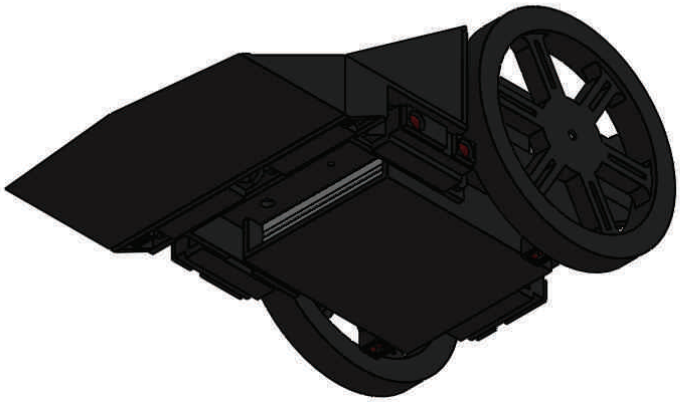




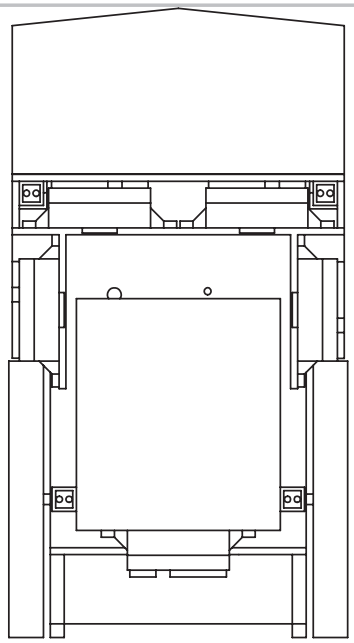


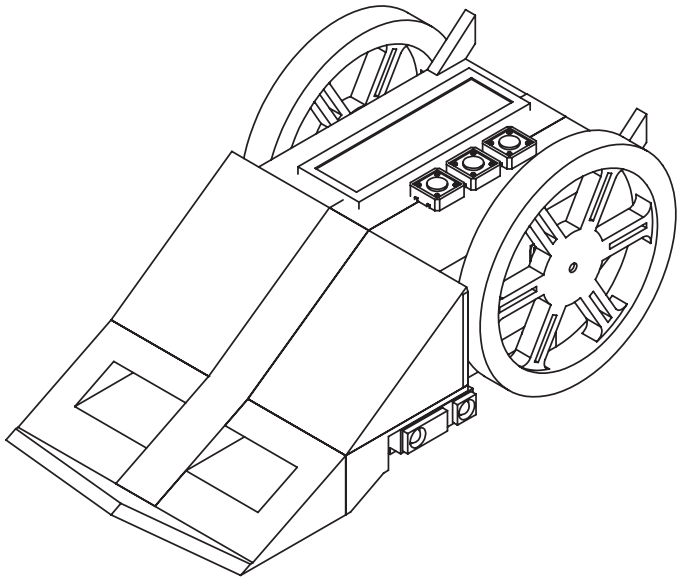
98

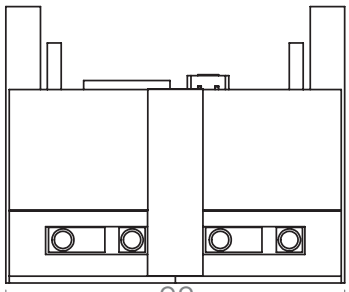


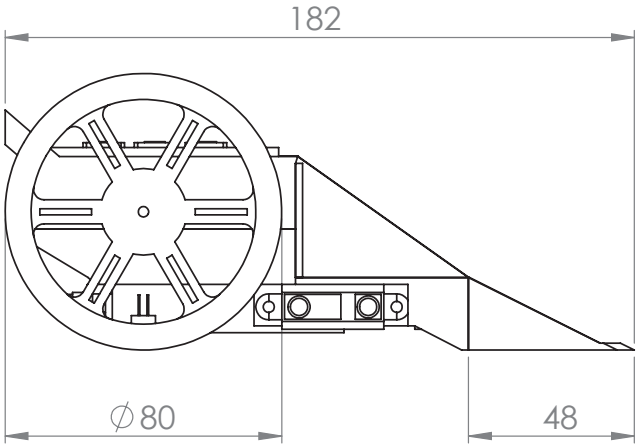


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:				REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Mark IV	
				PESO:		ESCALA:1:2	HOJA 1 DE 1

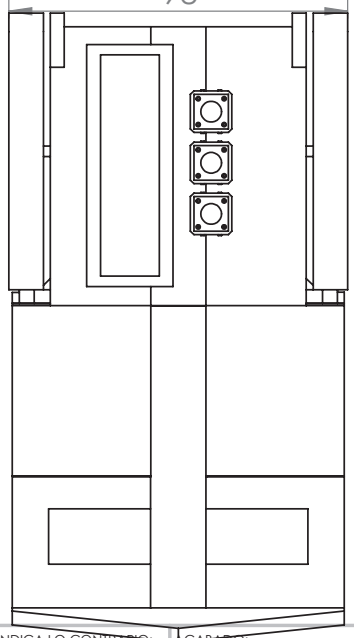


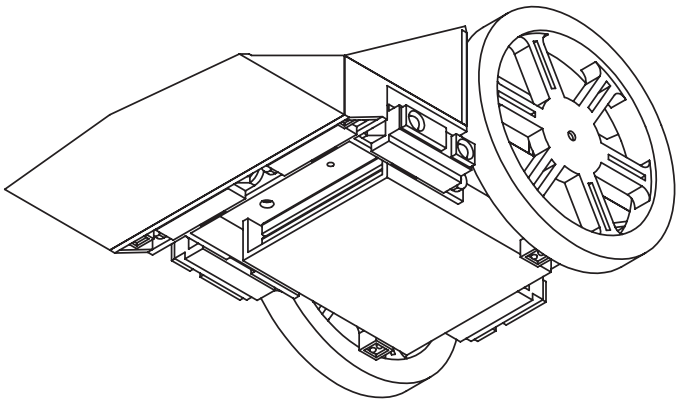




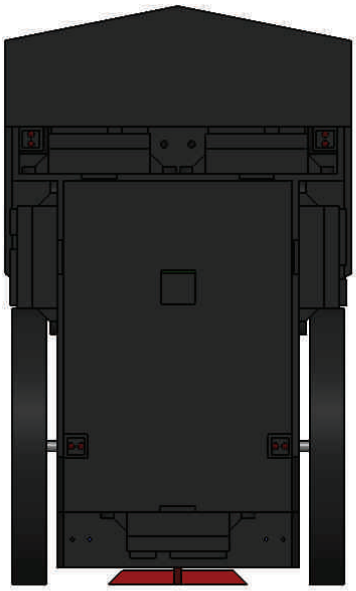


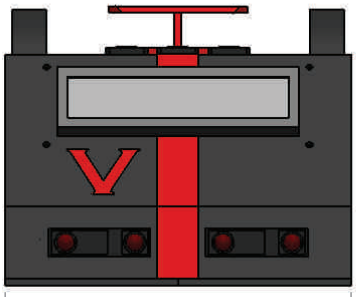
98

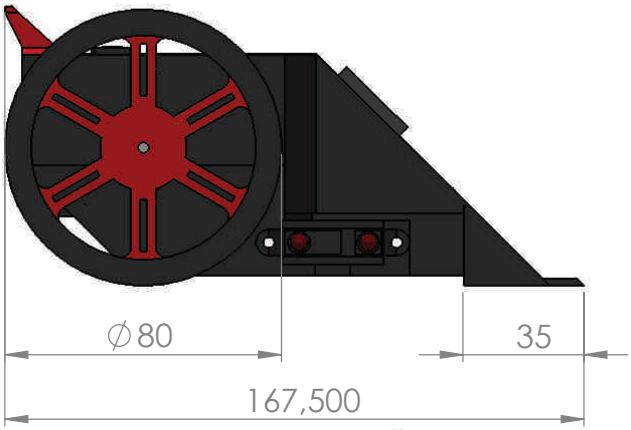


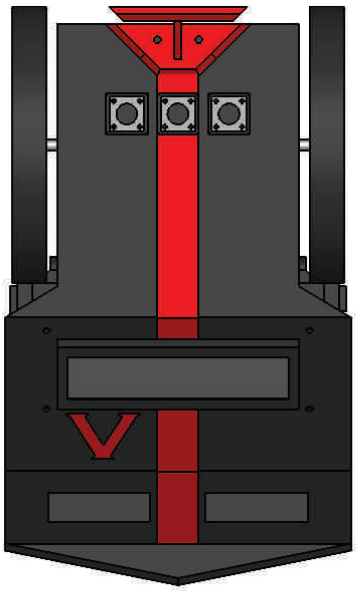


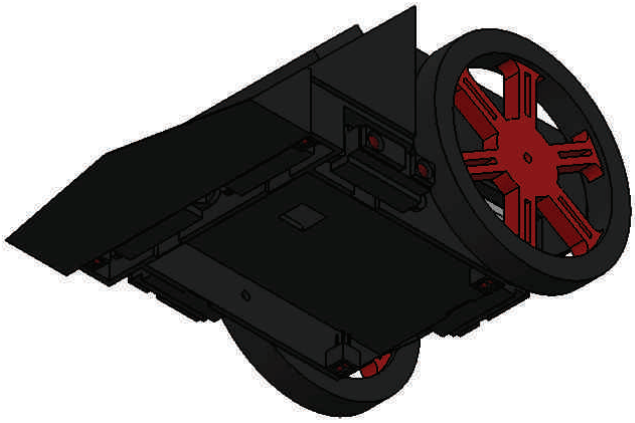
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN		
	NOMBRE	FIRMA	FECHA				TÍTULO:
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Mark IV	
				PESO:		ESCALA:1:2	HOJA 1 DE 1











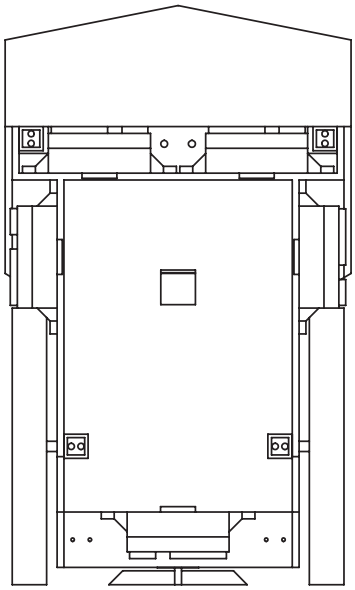
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN

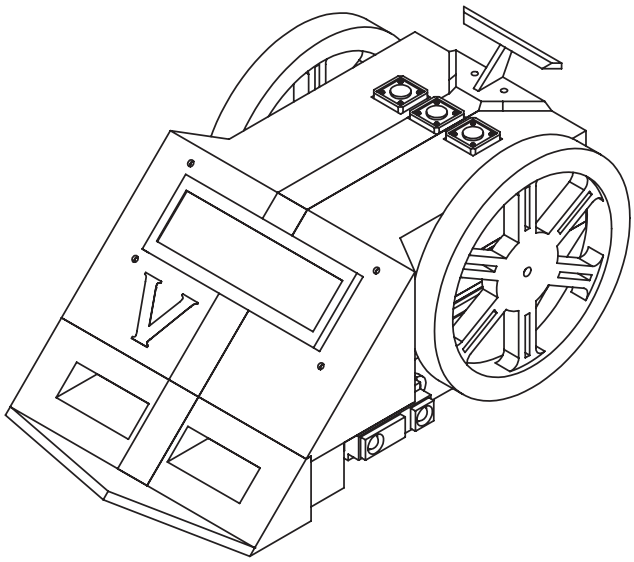
	NOMBRE	FIRMA	FECHA	
DIBUJ.				TÍTULO:
VERIF.				
APROB.				
FABR.				
CALID.				MATERIAL:
				N.º DE DIBUJO: Mark V
				PESO:

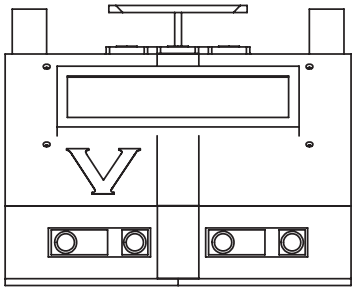
A4

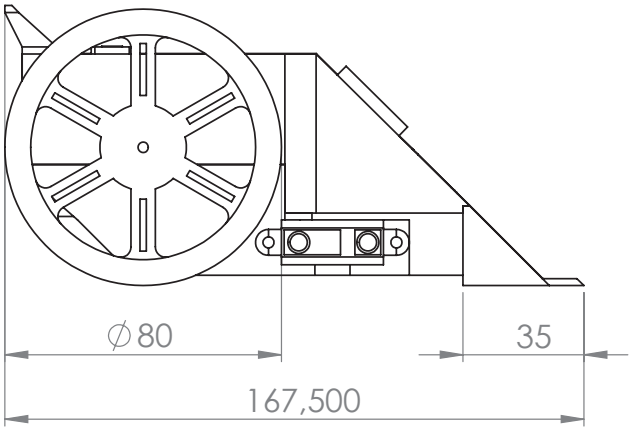
ESCALA:1:2

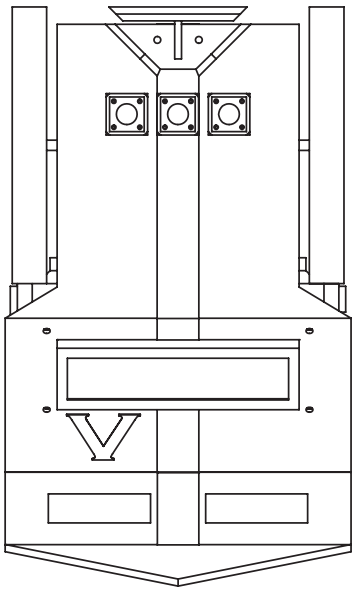
HOJA 1 DE 1

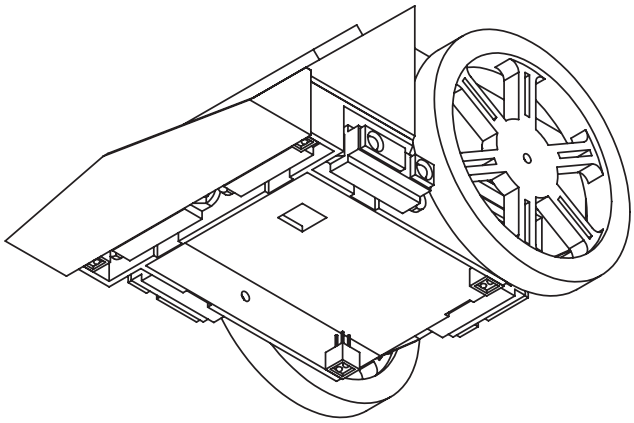




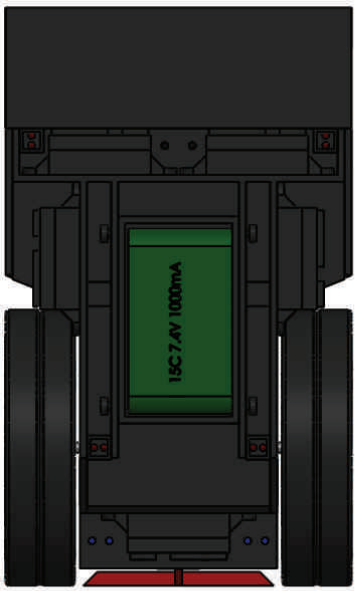


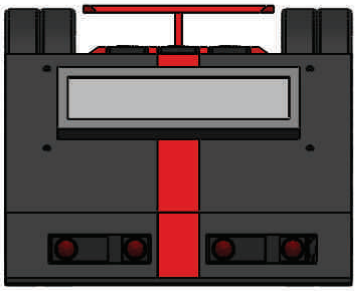


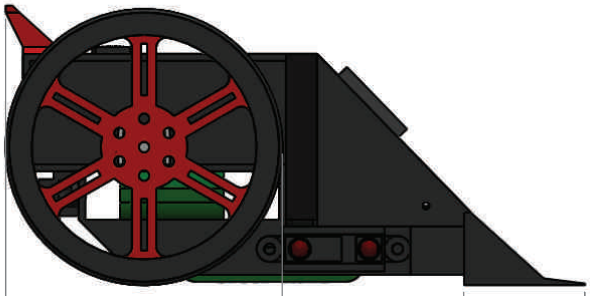


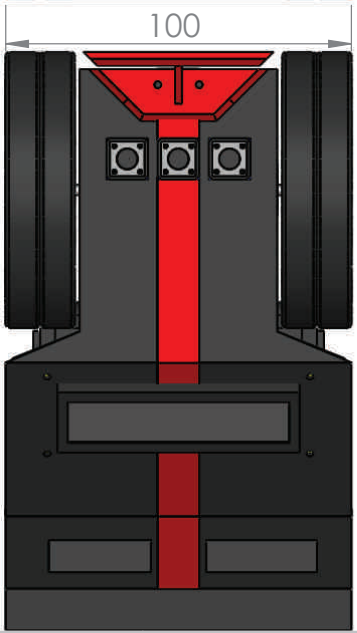


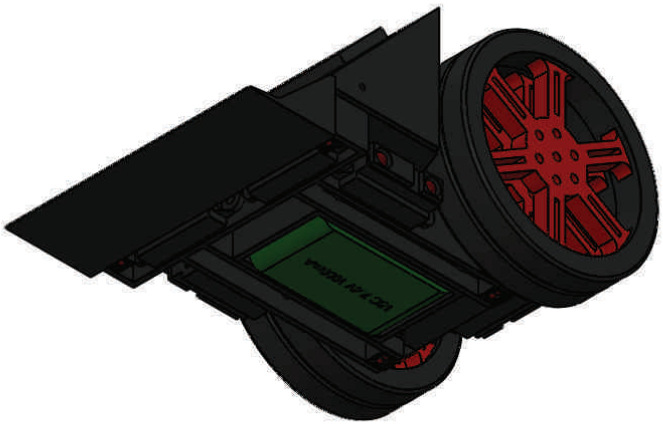
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Mark V	
				PESO:		ESCALA:1:2	HOJA 1 DE 1



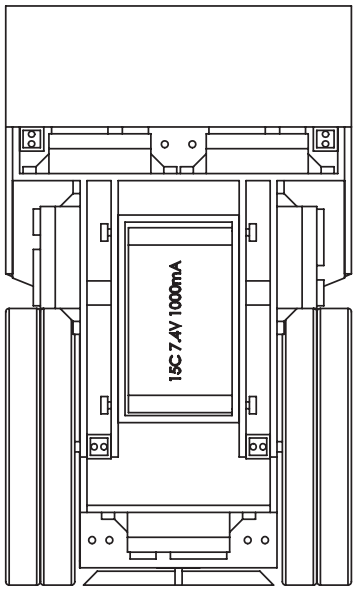


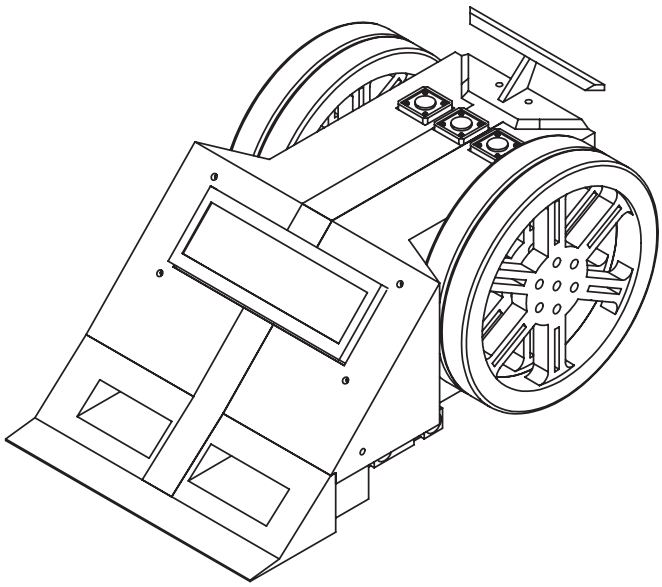


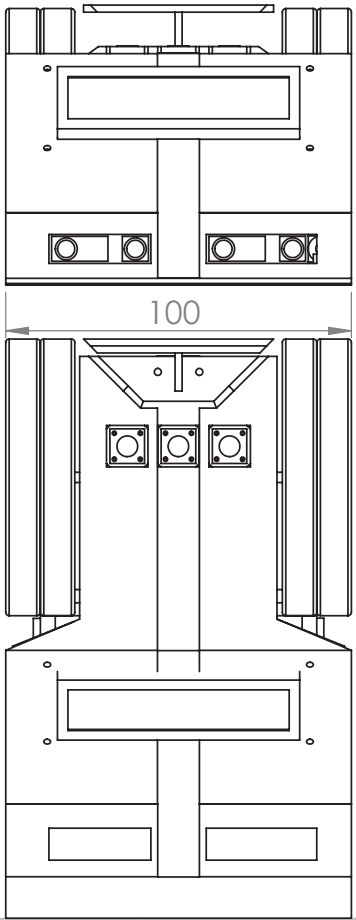


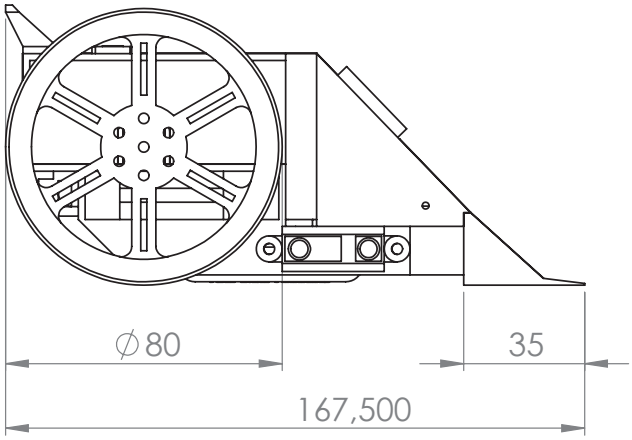


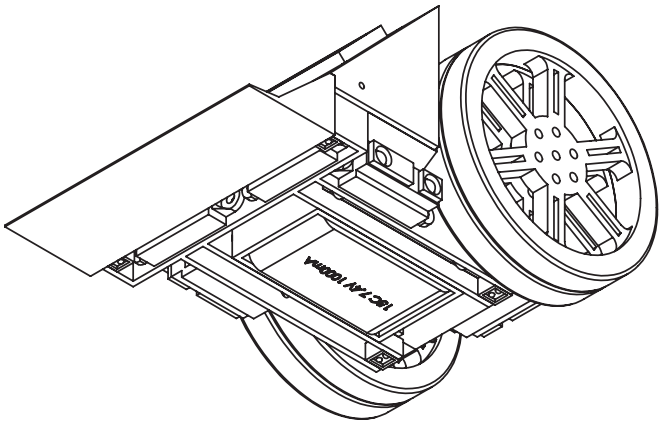
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN		
	NOMBRE	FIRMA	FECHA				TÍTULO:
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
				PESO:		ESCALA:1:2	HOJA 1 DE 1











SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.			
VERIF.			
APROB.			
FABR.			
CALID.			

TÍTULO:

MATERIAL:

N.º DE DIBUJO

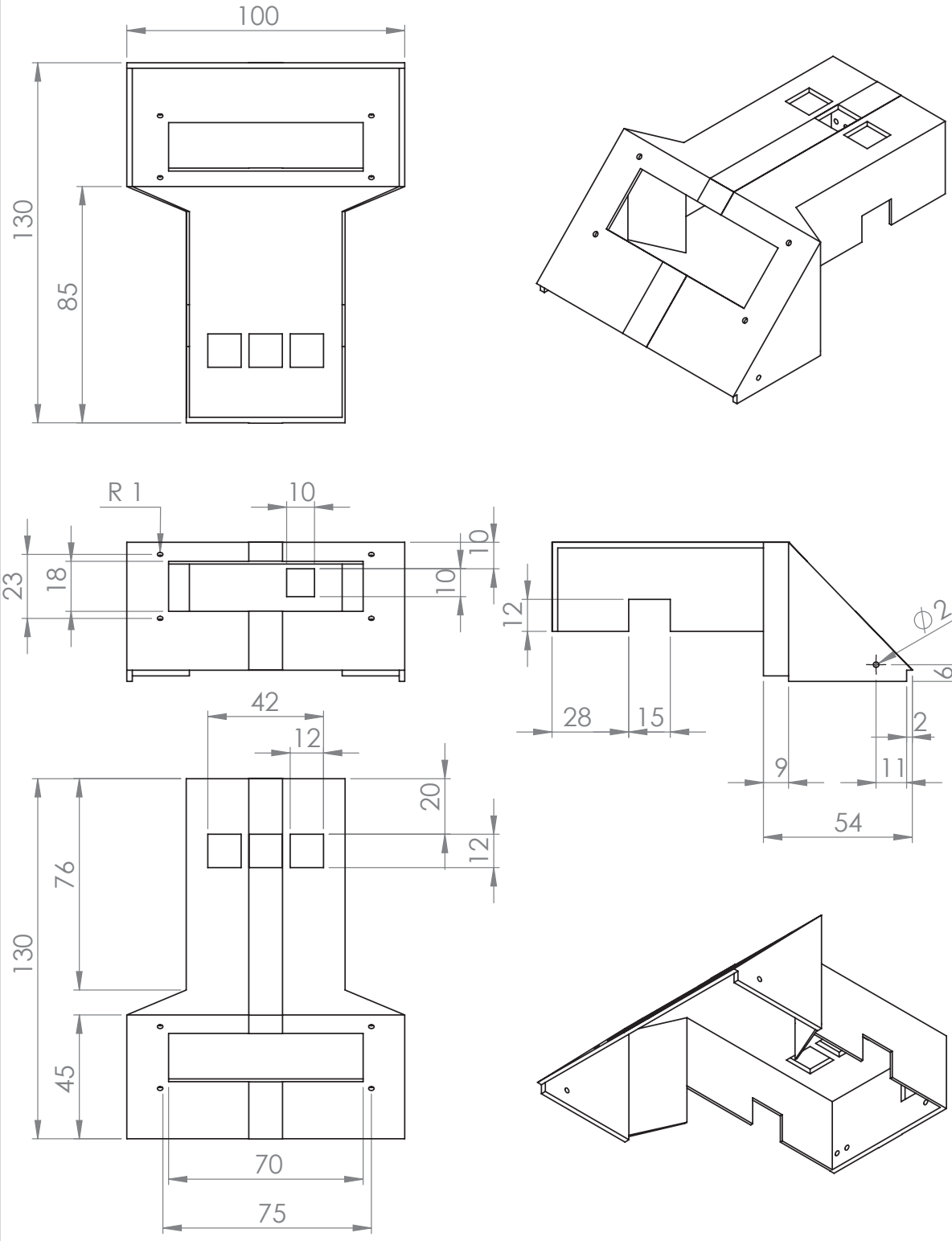
Mark VI

A4

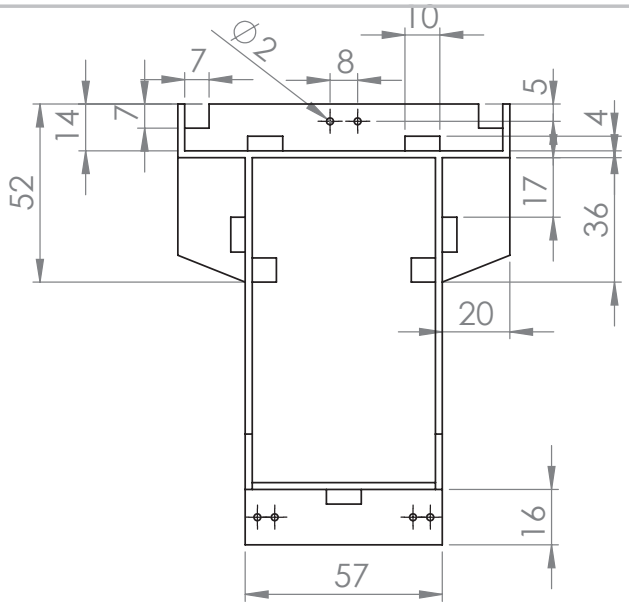
PESO:

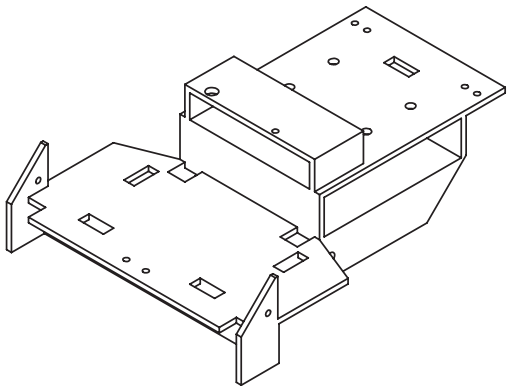
ESCALA:1:2

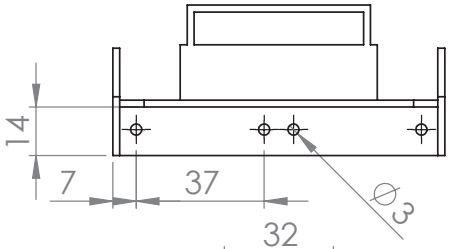
HOJA 1 DE 1

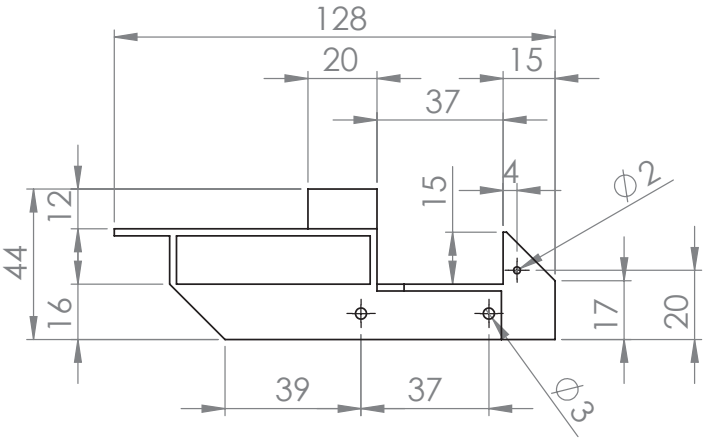


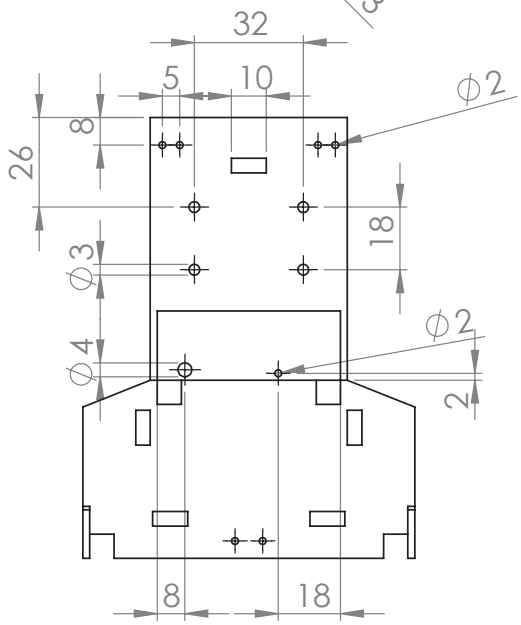
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN	
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Carrocería	
				PESO:		ESCALA:1:2	HOJA 1 DE 1

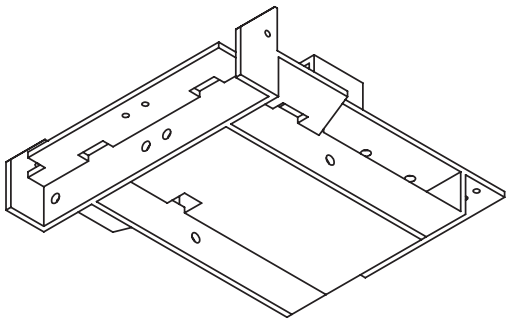












SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN		
	NOMBRE	FIRMA	FECHA				TÍTULO:
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:			N.º DE DIBUJO
							Chasis
							A4
				PESO:		ESCALA:1:2	HOJA 1 DE 1