UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA
LABORATORIO DE SISTEMAS INTELIGENTES

EVOLUCIÓN DE LA ARQUITECTURA DE UN MICROROBOT (EUROBOT)

PROYECTO FIN DE CARRERA
INGENIERÍA INFORMÁTICA

Tutor: José María Armingol
Alumno: David Álvarez Izquierdo

Índice

Índice de ilustraciones

1 Introducción y objetivos

El Laboratorio de Sistemas Inteligentes (LSI) de la Universidad Carlos III de Madrid compite anualmente en el concurso internacional de microrobótica Eurobot [3].
Durante los 3 años de duración de este proyecto, se ha implementado el software de los tres microrobots participantes en dicha competición en las ediciones de 2008, 2009 y 2010.

Ilustración 1 - Robots representantes del LSI en Eurobot 2008, 2009 y 2010

La programación de estos microrobots abarca desde el diseño global de la arquitectura software hasta la creación de estrategias de actuación de los microrobots. Además, se ha implementado el software de control de cada uno de los dispositivos hardware que los componen y la comunicación entre estos dispositivos.

Los objetivos se marcan anualmente, es decir, para cada edición de Eurobot. En las fases previas de diseño de la arquitectura global del sistema, se fijan las metas que se estima oportuno que deba cumplir el microrobot a construir. Estas metas no solo influirán en la estrategia de actuación del microrobot, sino que también determinarán en cierta medida la arquitectura hardware de los dispositivos que compondrán el sistema de control, la cantidad de actuadores y sensores que se emplearán, la construcción física de la estructura del microrobot y la disposición de todos estos dispositivos en la estructura.

En estos años de trabajo en la programación de distintos microrobots, se ha ido confeccionando una ***base reutilizable de código***, que pueda ser utilizada por futuros alumnos, de manera que éstos puedan enfocar su esfuerzo en avanzar hacia metas más altas en el diseño de estrategias complejas de actuación.

El presente ***documento*** pretende ser el ***manual de uso*** de esta base reutilizable. Va dirigido a alumnos encargados de la ***parte informática***, por lo que se hace especial hincapié en detallar minuciosamente todo aspecto relativo al software del microrobot, abstrayendo por otra parte aquellos detalles técnicos considerados fuera del ámbito de la informática.

Se hace ***única y exclusivamente*** mención a características técnicas relativas a la electrónica y mecánica que se hayan considerado ***imprescindibles*** para el correcto entendimiento del funcionamiento global del sistema.

2 Estado del arte

La historia de la robótica ha estado unida a la construcción de "artefactos", que trataban de materializar el deseo humano de crear seres a su semejanza y que lo descargasen del trabajo.

Desde el siglo XIII, se han llevado a cabo importantes avances en el mundo de la robótica, aunque es en el siglo XIX cuando han acaecido los más importantes, que han permitido que la robótica se acerque al cumplimiento de sus objetivos iniciales.

Durante este siglo XXI la robótica y las técnicas de Inteligencia Artificial, previsiblemente experimentarán el desarrollo definitivo que dote a las máquinas de autonomía en el trabajo y de ciertos rasgos de inteligencia en la interacción social.

2.1 Eurobot

Competición internacional de robots autónomos seleccionados previamente en ciertas pruebas clasificatorias nacionales celebradas en cada país. Todas las etapas del concurso transcurren con la misma normativa (ver apartado "***9.1 Normativa del concurso Eurobot***" de este documento).

El concurso está destinado a la participación de centros de estudios o asociaciones de jóvenes que realicen su propio proyecto. Pretende ser un punto de encuentro para intercambiar conocimientos científicos, tecnológicos y experiencias personales.

Eurobot fue creado por la asociación de promoción de la ciencia y la tecnología Planète Sciences [4], la productora televisiva VM Group y la ciudad donde se organizó la primera edición, La Ferté Bernard [5]. Actualmente existe un comité internacional formado por varios órganos de gobierno y gestión encargados del desarrollo de la competición en el futuro.

Cada país puede estar representado por un máximo de tres equipos en la competición internacional de Eurobot. Por eso, cada país que participe y en el que haya más de tres equipos candidatos, se hace necesaria la organización de una prueba previa de clasificación.
En el caso de España, esta prueba de clasificación es la Copa de España de Robots, celebrada de forma conjunta a la competición Alcabot-Hispabot [6].

2.1.1 Ediciones anteriores

1998. Fútbol

La competición fue similar a un partido de fútbol pero con más de una pelota. Dos robots enfrentados tenían que marcar al rival el máximo número de tantos posibles. Se presentaron nueve equipos de cinco países diferentes.

1999. Ataque al castillo

El escenario se encontraba dividido en dos partes con dos puentes. En cada lado se levantaban dos castillos realizados con cilindros de madera. Cada robot tenía que tirar los cilindros del equipo contrario, bien con bolas de tenis dispuestas en el terreno de juego o bien por contacto directo. Se presentaron nueve equipos de cinco países diferentes. En la figura se puede ver una representación del escenario del citado año.

Ilustración 2 - Eurobot 1999, Ataque al castillo

2000. Parque de atracciones

En este año, el terreno se presentaba de manera irregular, con diez globos, cinco azules y cinco amarillos. Cada equipo tenía que explotar los globos del equipo contrario en el lado opuesto a su terreno de juego. No se permitía el uso de proyectiles y el tamaño máximo del robot no permitía explotar los globos desde lejos. En la figura se puede ver una representación del escenario del citado año. Se presentaron doce equipos de siete países distintos.

Ilustración 3 - Eurobot 2000, Parque de atracciones

2001. Odisea en el espacio

La competición constaba de dos robots participantes que tenían que "conquistar planetas" (representados por cilindros). Para conquistar estos "planetas" debían de situar una bandera encima de ellos. En la figura, se puede ver una representación del escenario del citado año. Se presentaron diecinueve equipos de doce países distintos.

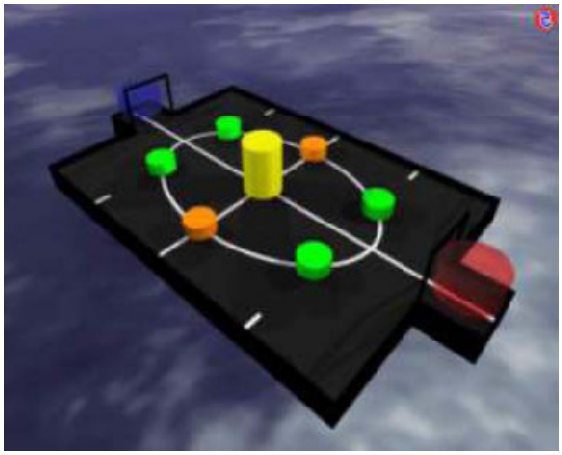


Ilustración 4 - Eurobot 2001, Odisea en el espacio

2002. Billar aéreo

En un terreno de juego liso, había dispuestas ocho bolas rojas y cuatro negras siguiendo una simetría central. Cada robot tenía que situar las bolas negras en las troneras de su lado y las rojas en el lado opuesto. En la figura se puede ver una representación del escenario del citado año. Se presentaron veintisiete equipos de diecisiete países distintos.

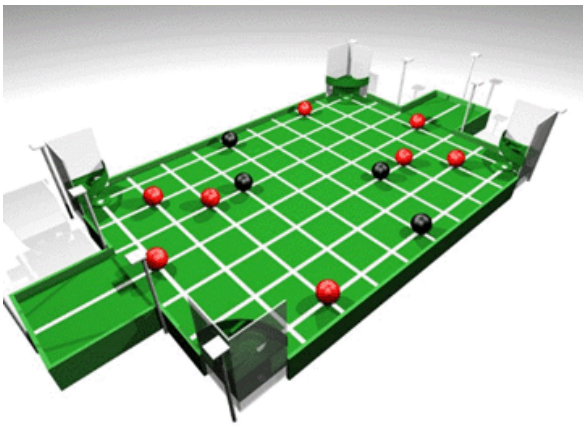


Ilustración 5 - Eurobot 2002, Billar aéreo

2003. Cara o cruz

En este año en el tablero de juego se situaban fichas de dos colores (rojo y verde) y también fichas de un único color (también rojas y verdes) Cada uno de los dos robots participantes debía intentar mostrar la mayor cantidad de fichas con la cara de su color asignado hacia arriba pasado un minuto y medio. En la figura se puede ver una representación del escenario del citado año. Se presentaron treinta y dos equipos de diecinueve países distintos.

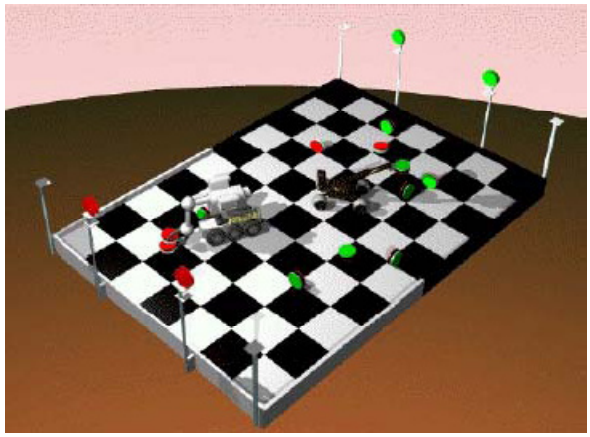


Ilustración 6 - Eurobot 2003, Cara o cruz

2004. Rugby de cocos

En el 2004, los robots se reunieron en una 'isla tropical'. El reto: jugar a rugby con los cocos de las palmeras. Cocos con forma de balón de rugby, por supuesto.

El propósito del juego era recogerlos y llevarlos al área de gol del contrario, o arrojarlos a través de las dos palmeras del oponente. Se presentaron cuarenta y un equipos de veintiún países distintos.

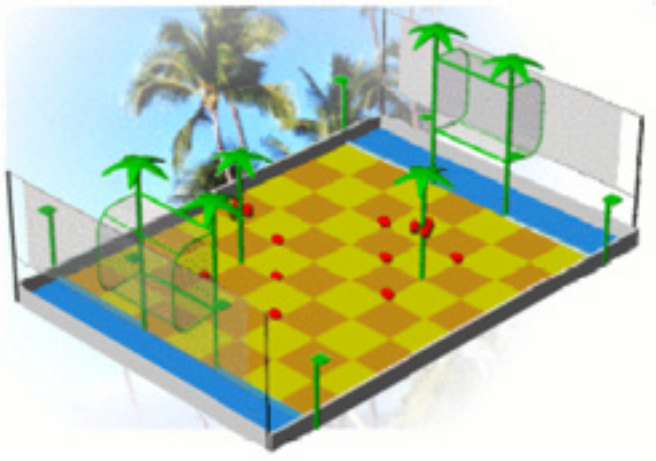


Ilustración 7 - Eurobot 2004, Rugby de cocos

2005. Juego de bolos

La prueba consiste, como su nombre indica en un juego de bolos. Al comienzo del juego cada equipo tiene en su campo unos bolos que debe defender a la vez que tiene que intentar tirar los bolos del contrincante. Tras un minuto y medio de juego, gana el equipo que haya tirado más bolos del contrincante. Para hacer el juego más interesante, entre los dos campos hay un foso atravesado por dos puentes que pueden variar su posición de una partida a otra.
Cada equipo puede tener hasta dos robots cumpliendo las restricciones de tamaño que se indican en las normativas. Participaron cincuenta equipos de más de veinte países.

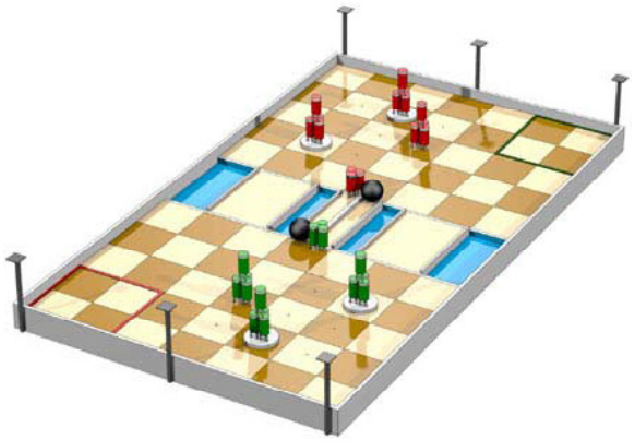


Ilustración 8 - Eurobot 2005, Juego de bolos

2006. Un divertido golf

En esta ocasión el campo tiene multitud de hoyos donde los robots de cada equipo deben introducir pelotas de golf "blancas". Además, existen pelotas "negras" que pueden introducirse en los hoyos del campo contrario, para evitar que el contrincante puntué. Las pelotas de golf saltan al terreno de juego al ser cerrado un contacto situado en unos postes sobre el terreno de juego.
El tiempo total para la prueba es de un minuto y medio y cada equipo solo puede utilizar un robot.

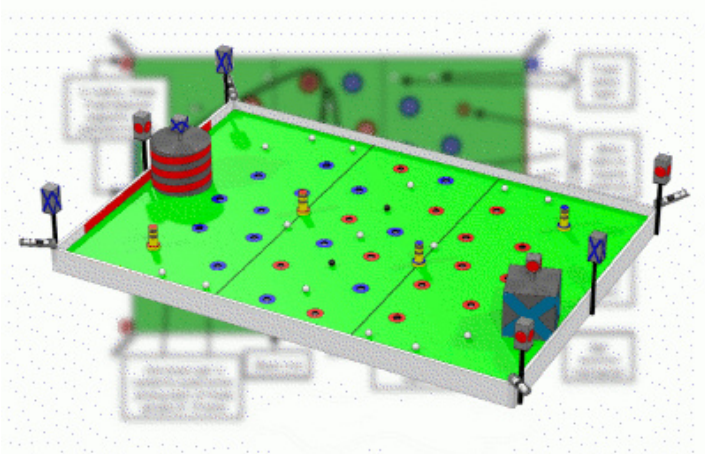


Ilustración 9 - Eurobot 2006, Un divertido golf

2007. Rally de reciclado

La prueba de 2007 consiste en clasificar una serie de desechos (botellas de plástico, latas de refrescos y pilas) cada uno en su cesta correspondiente para su reciclado posterior. Gana el robot que sea más rápido clasificando los desechos.

El tiempo total para la prueba es de un minuto y medio y cada equipo solo puede utilizar un robot.

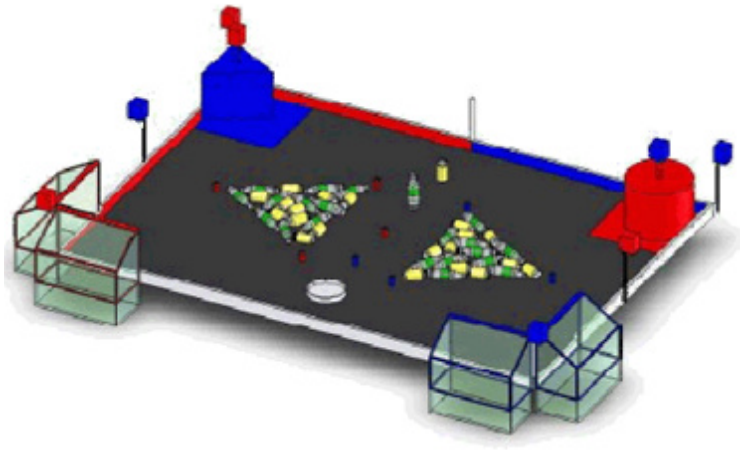


Ilustración 10 - Eurobot 2007, Rally de reciclado

2008. Misión a Marte

El equipo que consiga recolectar la mayor cantidad posible de pruebas de vida en Marte gana. Las pruebas son representadas por pelotas de Hockey de color azul y rojo. Estas pruebas darán más puntuación a su equipo si son almacenadas con hielo (representado por pelotas de color blanco), ya que asegura una mejor conservación de las mismas. Deben depositar las pruebas en los distintos contenedores existentes a lo largo del campo de juego.

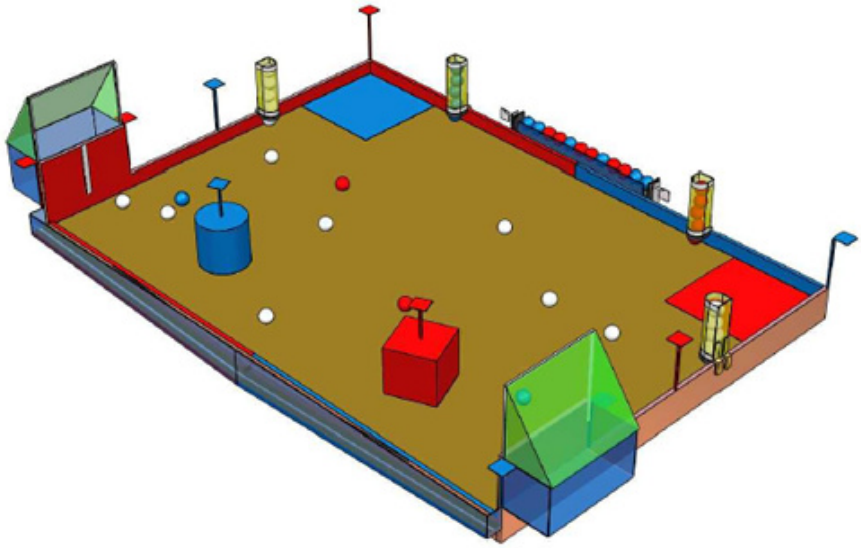


Ilustración 11 - Campo de juego Eurobot 2008

2.1.2 Eurobot 2008

El equipo representante de la Universidad Carlos III de Madrid en la edición de Eurobot de 2008, estaba inicialmente compuesto por 4 estudiantes de Ingeniería Técnica Industrial de especialidad en electrónica (Ignacio Albillo, Roberto Apéstigue, José Luis Martín y Pablo Escribano). Estos alumnos, trabajaron duramente para diseñar y construir la arquitectura mecánica y electrónica del microrobot.

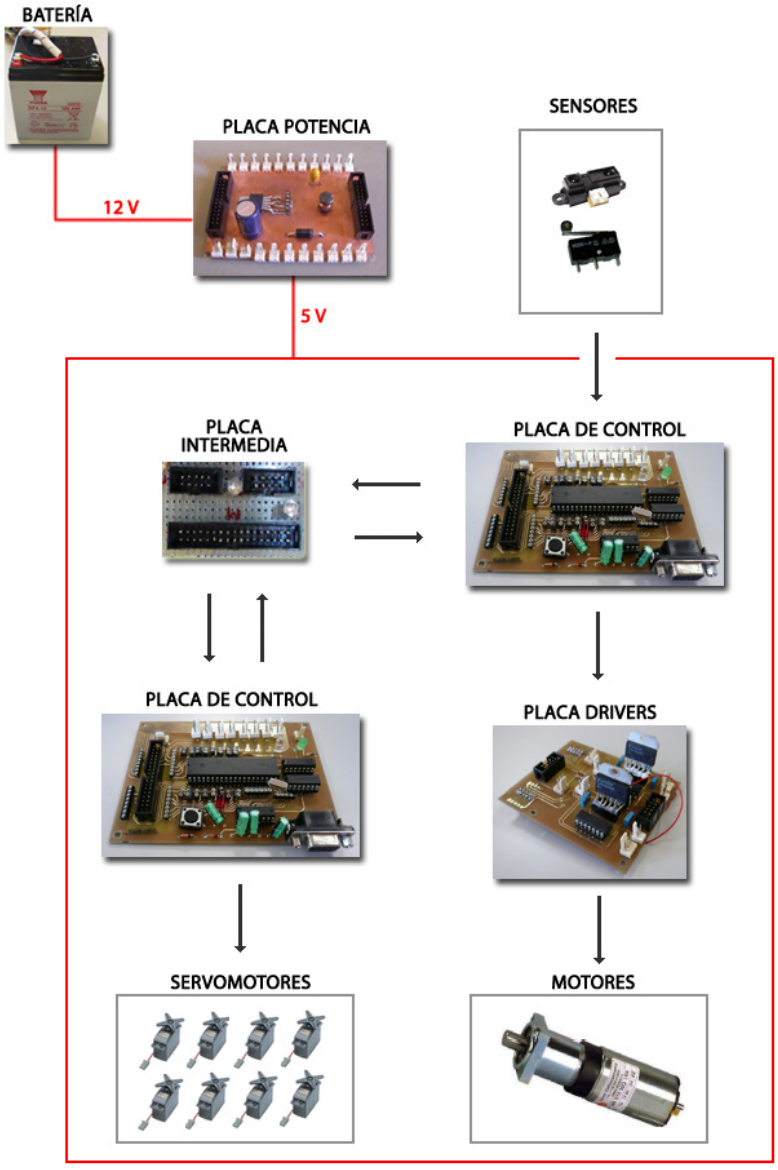


Ilustración 12 - Arquitectura hardware diseñada por el equipo de Eurobot 2008

Tal fue el compromiso de estos estudiantes, que consiguieron construir físicamente el microrobot semanas antes de la fecha en la que el concurso tendría lugar, contando por lo tanto con un margen de tiempo más que suficiente para programar la estrategia de actuación del microrobot. Para ayudar a estos alumnos en la creación del código que implementara esa estrategia, José María Armingol, tutor de este proyecto, decidió incorporar estudiantes de Ingeniería Informática.

3 Arquitectura hardware

Durante el periodo que abarca este proyecto (2008 a 2010), se han llevado a cabo mejoras tanto en el diseño hardware como en el diseño software de la arquitectura del microrobot. Se mencionan ***brevemente*** a continuación los cambios más relevantes realizados sobre la arquitectura hardware.

Adquisición de una placa de control que posibilite la programación multihilos

Se procede a la compra de un ordenador integrado en un circuito de placa (dispositivo conocido como Single Board Computer [7]) para ampliar considerablemente los horizontes computacionales. En este documento es denominada "***placa Linux***".

Desventajas: Gasto económico
Ventajas: Sistema Operativo Linux. Programación multihilo. Amplia memoria tanto volátil (RAM) como permanente (disco duro). Multitud de puertos de comunicaciones (Ethernet, puerto serie, puerto USB). Multitud de entradas analógicas/digitales. Entorno de programación Eclipse [33].

Fusión de dos placas de control en una única

Se fusionaron las dos placas de control existentes en la arquitectura inicial en una única placa que controlara tanto los servomotores como los motores de corriente continua. Esta nueva placa es denominada "***placa de servo-motores***".

Desventajas: Horas de trabajo en la creación de un único código para controlar tanto servomotores como motores.
Ventajas: Ahorro de energía y espacio al eliminar la existencia de una placa.

Eliminación de las placas de potencia

Se incorporan las funciones de alimentación del sistema en la antigua placa intermedia de conexiones. La denominada "***placa intermedia***" será la encargada de la transformación del voltaje de salida de la batería (12 voltios) en los voltajes requeridos por el resto del sistema. Además, servirá de conexión de control y datos entre la placa de servo-motores y la placa de drivers.

Desventajas: Horas de trabajo en el diseño y construcción de la nueva placa intermedia
Ventajas: Ahorro espacio al eliminar la existencia de una placa.

3.1 Esquema general de conexión de dispositivos (Eurobot 2010)

La arquitectura de dispositivos hardware empleada en el microrobot para Eurobot 2010 queda representada en la ***Ilustración 13***.

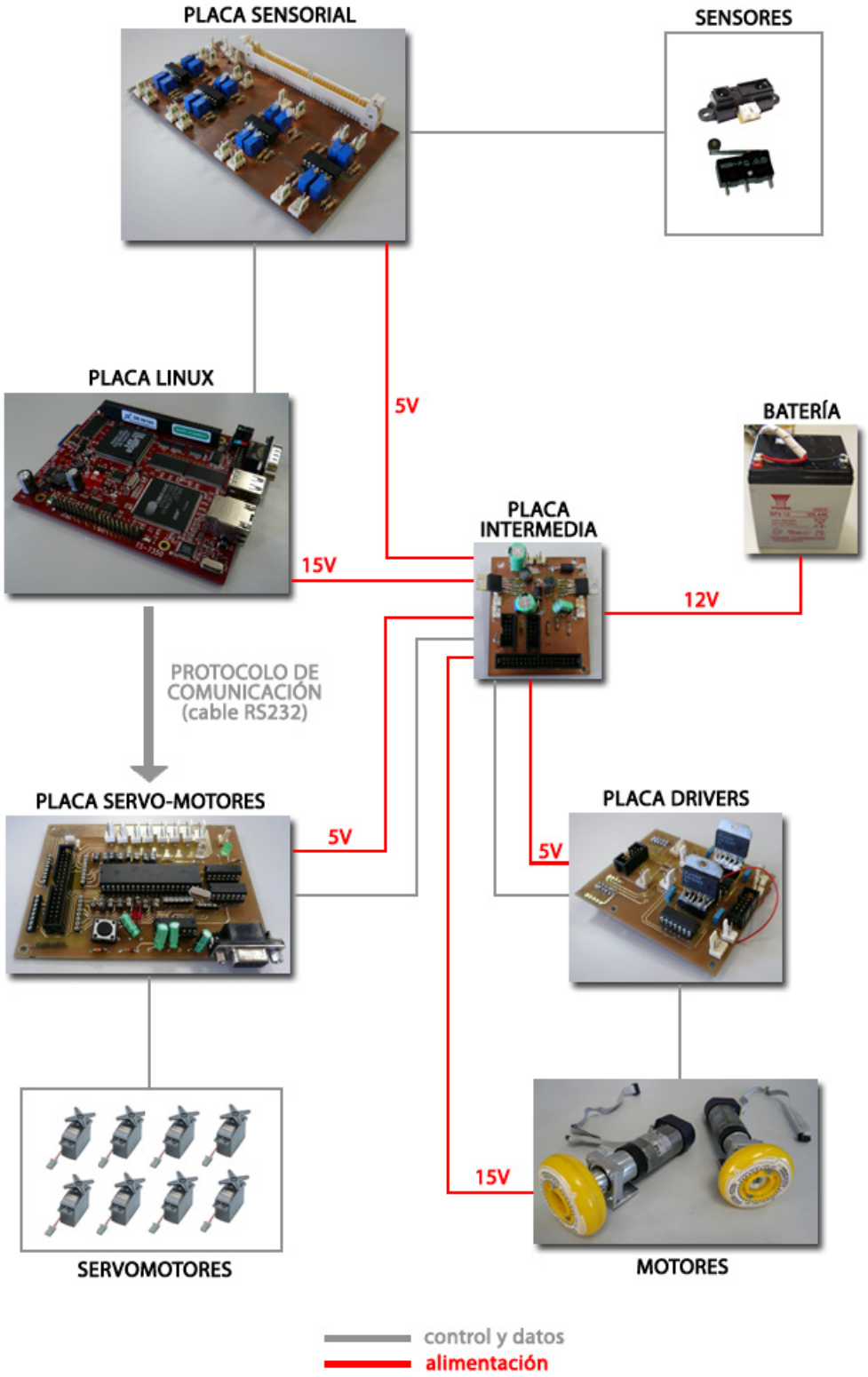


Ilustración 13 - Esquema general de conexión de dispositivos

3.2 Placa Linux

Ordenador integrado en el circuito de una placa. Este tipo de dispositivos es conocido como Single Board Computer [7]. En el presente documento será denominada "***Placa Linux***". Sus principales características técnicas son las siguientes:

- CPU ARM9 a 200MHz
- 32MB SDRAM de memoria
- 5K LUT FPGA
- Conector flexible de 64-pin tipo PC/104
- Puerto Ethernet 10/100
- 2 puertos USB 2.0
- Ranura para tarjeta SD
- 3 puertos RS-232, 2 puertos TTL COM
- Cabeza de 40 pines con salidas y entradas de tipo ADC, SPI, I2C, DIO...
- Soporta temperaturas entre -40° y +70°C
- Alimentada con voltajes desde 5 hasta 28VDC
- Arranca Linux 2.6 en aproximadamente 1 segundo.

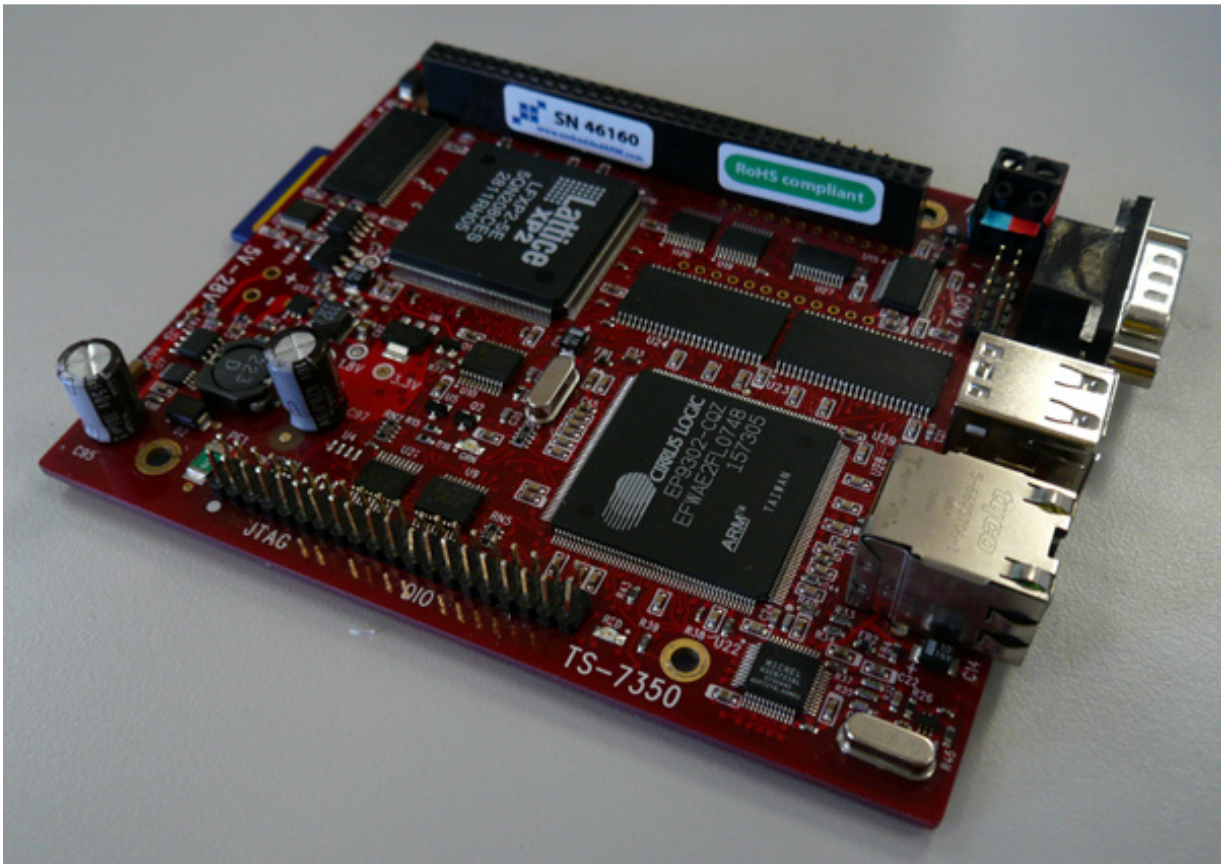


Ilustración 14 – Single Board Computer

Está dotada de distintos tipos de ***conectores*** que permiten conectarla al resto de placas que forman el sistema: Ethernet [8], puerto serie (RS232) [9], USB [10], multitud de entradas analógicas/digitales.

Ethernet
Utilizado para conectar con un ordenador de sobremesa o portátil, con el fin de establecer una comunicación entre ellos mediante el protocolo Telnet [11].

Puertos USB
Posibilita la ampliación del almacenamiento permanente mediante la conexión de memorias Flash [12].

Entradas analógicas/digitales
Utilizadas para la introducción al sistema de información del entorno, ya sea proveniente de interruptores, anilla de arranque, sensores infrarrojos, sensores de final de carrera o cualquier otro dispositivo de percepción.

Puerto serie (RS232)
Empleado para comunicar esta placa con la placa de servo-motores.

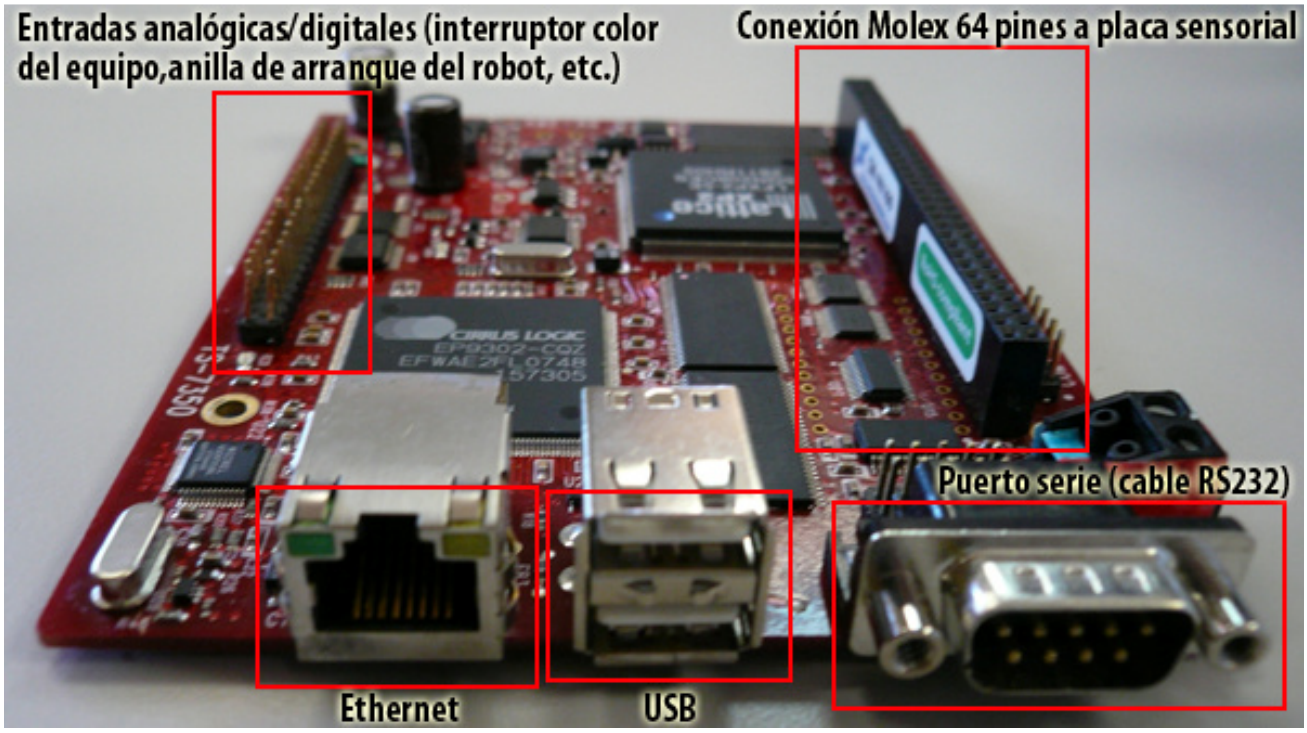


Ilustración 15 - Conexiones de la placa Linux

3.3 Placa de servo-motores

Placa diseñada para el control de servomotores y motores, equipada con un microcontrolador ***DS89C450-MNG*** y un circuito integrado para versión de niveles TTL-RS232 para poder establecer una comunicación serie entre el microcontrolador y un ordenador para la programación.

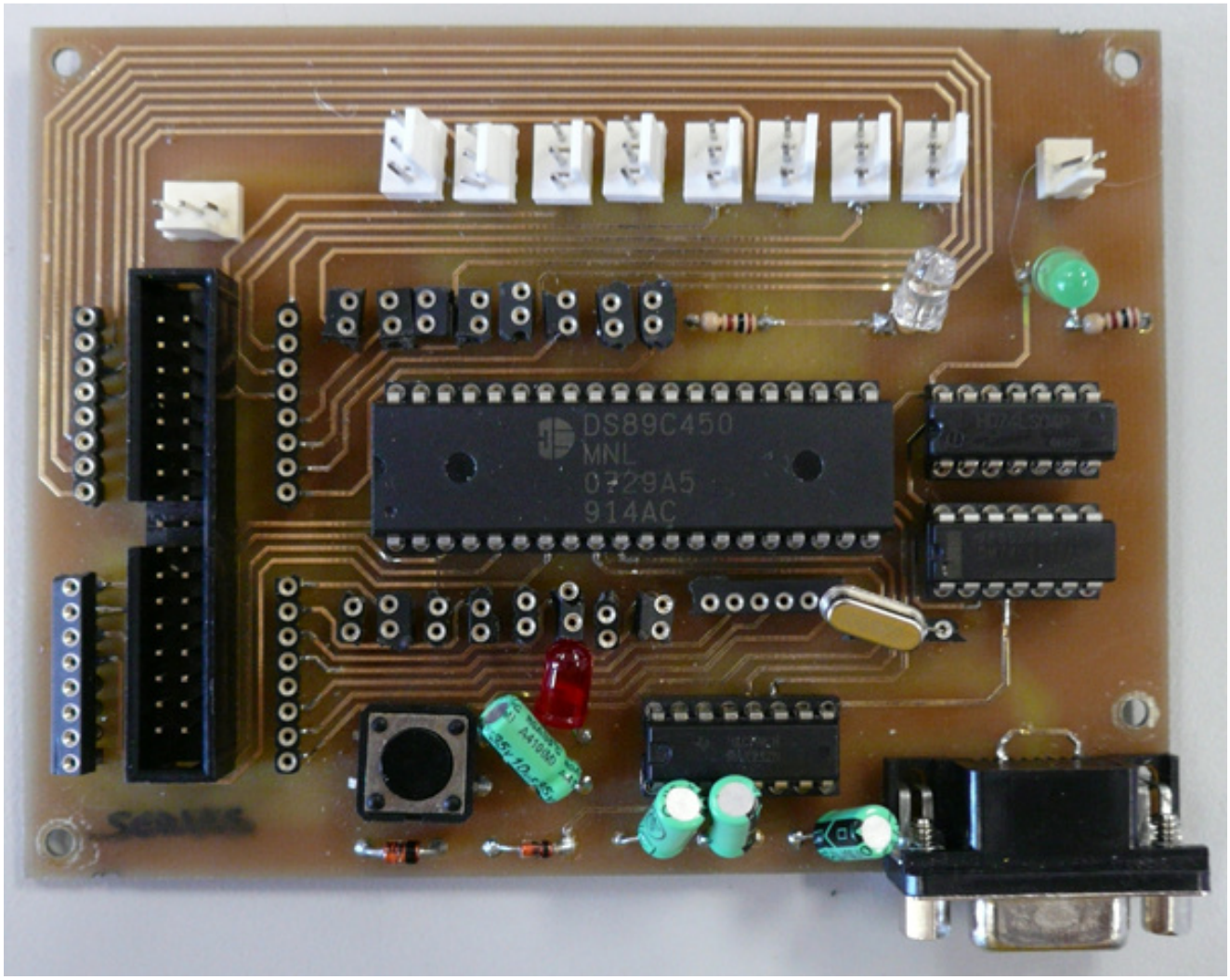


Ilustración 16 - Placa de servo-motores

La placa está equipada con varios diodos leds indicadores del estado de funcionamiento, toma de ***alimentación de 5 V***, botón de reset asíncrono y todos los puertos de entrada/salida del microcontrolador son accesibles mediante los correspondientes conectores. La conexión con los actuadores (servos y motores) se realiza mediante cable plano para evitar errores de conexión y minimizar el cruce de cables en el interior del microrobot.

En el apartado "***5. Manejo de la placa de servo-motores***" de este documento se ha confeccionado un detallado tutorial de manejo de las aplicaciones necesarias para trabajar con esta placa de servo-motores.

La placa de servo-motores se conecta a la placa Linux por medio del puerto serie usando un cable RS232 [9]. Este cable también se emplea para conectar la placa de servo-motores con un ordenador.

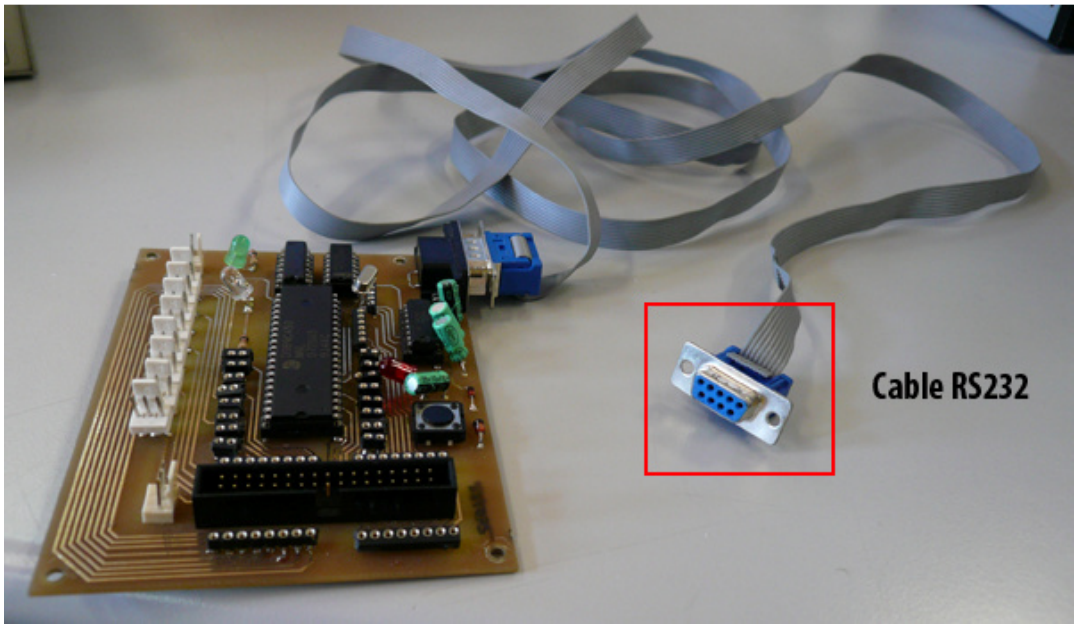


Ilustración 17 - Conexión de la placa de servo-motores mediante cable RS232

La placa de servo-motores se conecta además con la placa intermedia, mediante un cable para Molex de 32 pines.

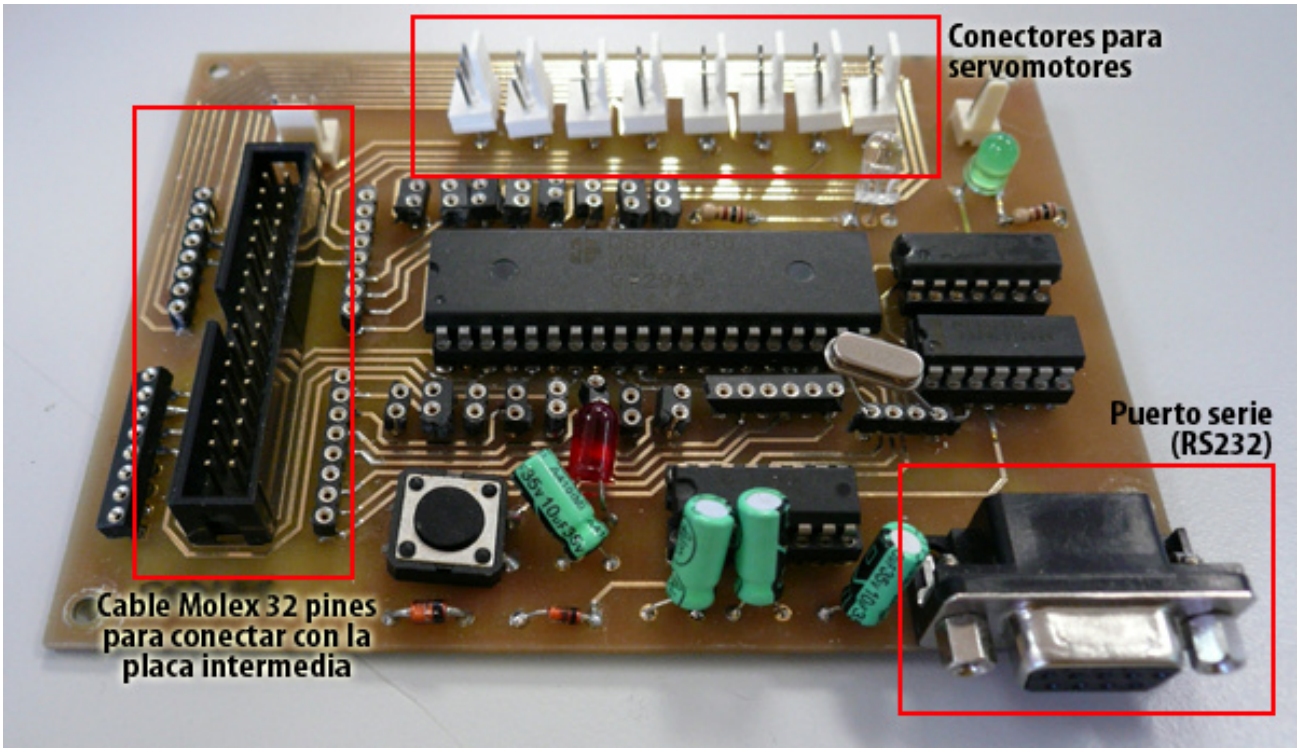


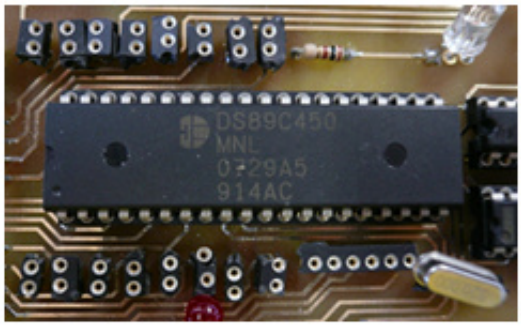
Ilustración 18 - Conexiones de placa de servo-motores con el resto de placas

3.3.1 Microcontrolador DS89C450-MNG

Este microcontrolador está basado en la familia Intel 8051 [13].

FEATURES

- **High-Speed 8051 Architecture**
  One Clock-Per-Machine Cycle
  DC to 33MHz Operation
  Single Cycle Instruction in 30ns
  Optional Variable Length MOVX to Access Fast/Slow Peripherals
  Dual Data Pointers with Automatic Increment/Decrement and Toggle Select
  Supports Four Paged Memory-Access Modes



- **On-Chip Memory**
  16kB/64kB Flash Memory
  In-Application Programmable
  In-System Programmable Through Serial Port
  1kB SRAM for MOVX
- **80C52 Compatible**
  8051 Pin and Instruction Set Compatible
  Four Bidirectional, 8-Bit I/O Ports
  Three 16-Bit Timer Counters
  256 Bytes Scratchpad RAM
- **Power-Management Mode**
  Programmable Clock Divider
  Automatic Hardware and Software Exit
- **ROMSIZE Feature**
  Selects Internal Program Memory Size from 0 to 64kB
  Allows Access to Entire External Memory Map
  Dynamically Adjustable by Software
- **Peripheral Features**
  Two Full-Duplex Serial Ports
  Programmable Watchdog Timer
  13 Interrupt Sources (Six External)
  Five Levels of Interrupt Priority
  Power-Fail Reset
  Early Warning Power-Fail Interrupt
  Electromagnetic Interference (EMI) Reduction

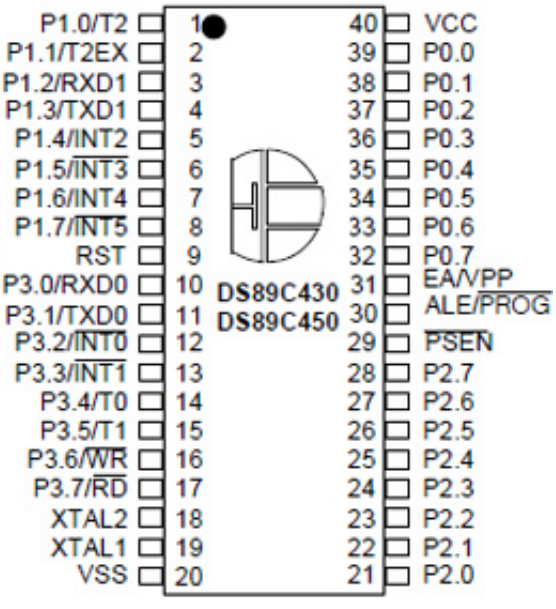


Ilustración 19 - Características del microcontrolador DS89C450-MNG

3.4 Placa intermedia

3.4.1 Alimentación del sistema

La alimentación de todo el sistema corre a cargo de la batería. Este dispositivo está únicamente conectado a la placa intermedia, que es la encargada de distribuir el voltaje entre los distintos elementos que componen el microrobot.

Cada elemento requiere distinto voltaje: ***5 voltios*** (placa de servo-motores, placa de drivers, placa del sistema sensorial) o ***15 voltios*** (motores, placa Linux). Es por este motivo, que la placa intermedia deberá transformar los ***12 voltios*** de entrada que proporciona la batería, a los distintos voltajes de salida demandados por los componentes del sistema para su correcto funcionamiento.

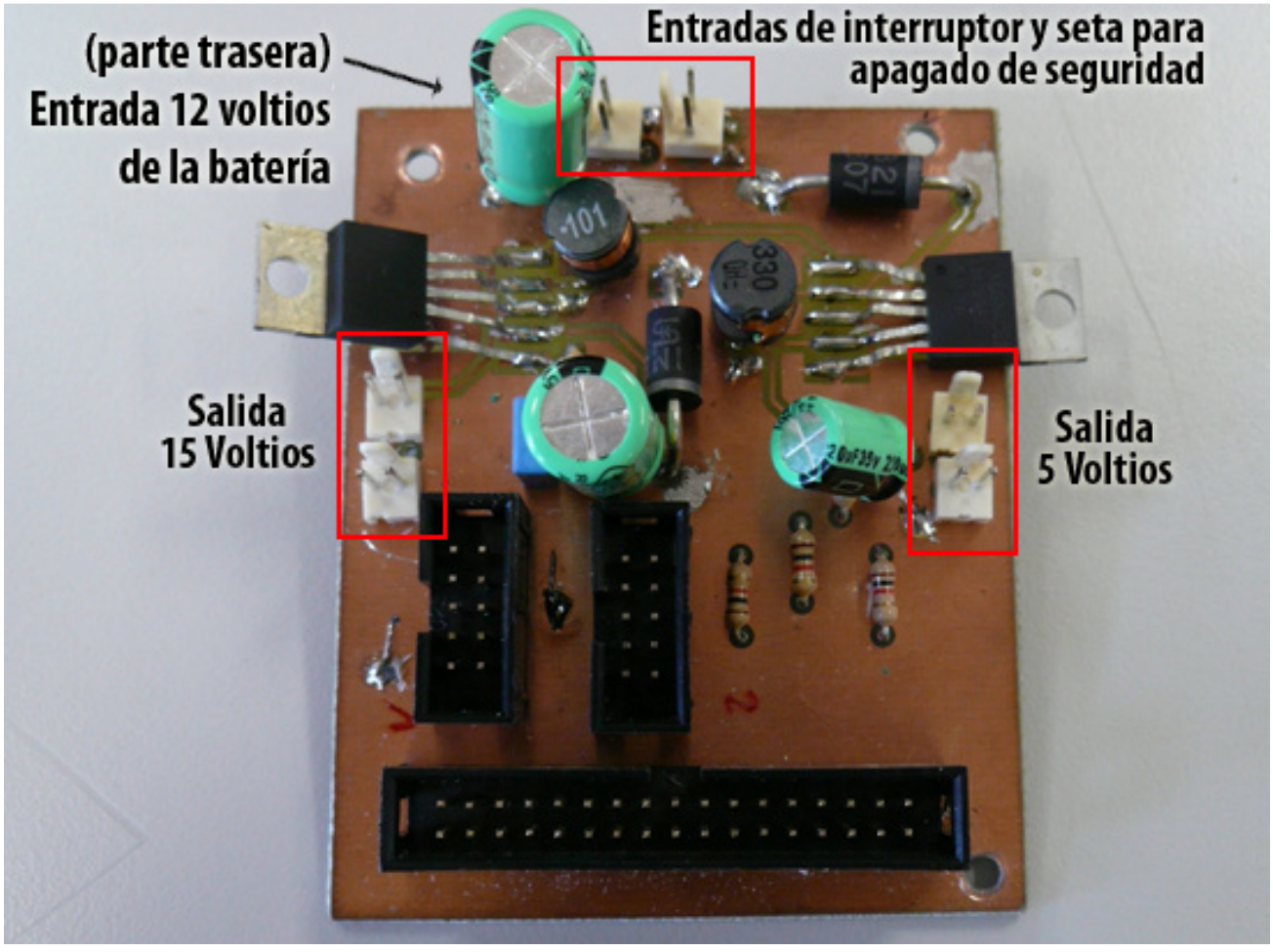


Ilustración 20 - Voltaje de entrada y salida de la placa intermedia

3.4.2 Flujo de datos e información de control del sistema

Además de la distribución de la alimentación del sistema, la placa intermedia sirve de enlace entre la placa de servo-motores y la placa de drivers. Por ella atraviesa la información de control que la placa de servo-motores debe hacer llegar a la placa de drivers para el movimiento de los motores. El desplazamiento de cada motor será cuantificado por un encoder, que enviará una interrupción a la placa de drivers por cada pulso de giro de la rueda que detecte (ver apartado "***3.7.1. Encoders***") Este flujo de interrupciones se transferirá desde la placa de drivers hasta la placa de servo-motores, a través de la placa intermedia.

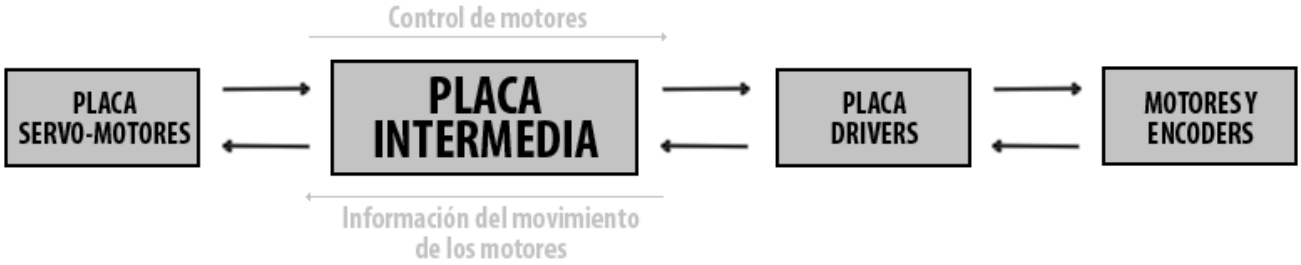


Ilustración 21 - Flujo de datos e información de control a través de la placa intermedia

Las conexiones de las que la placa intermedia está provista para cumplir sus funciones de intermediario entre las distintas placas del sistema quedan representadas en la ***Ilustración 22***.

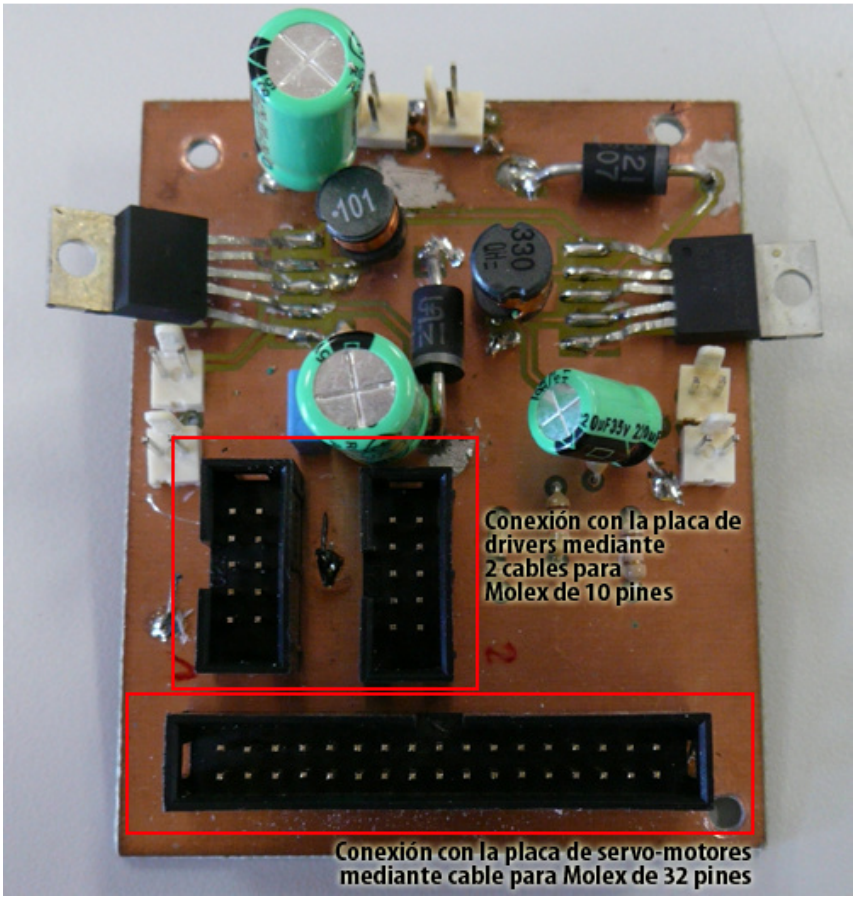


Ilustración 22 - Conexiones de placa intermedia

3.5 Placa del sistema sensorial

Este elemento del sistema se encarga de abstraer y simplificar la tarea de detección del entorno que lleva a cabo el microrobot. Esta información de entrada al sistema de control será la percibida por los sensores de infrarrojos (ver apartado "***3.9.1 Sensores infrarrojos***") y los sensores de final de carrera (ver apartado "***3.9.2 Sensores de final de carrera***"); será tratada y posteriormente enviada a la placa Linux.

Ilustración 23 - Entrada y salida de la placa del sistema sensorial

La señal digital, salida de los sensores final de carrera (bumpers) será transmitida tal cual a la placa Linux, mientras que la señal analógica, salida de los sensores de infrarrojos será convertida a señal digital antes de llegar a la placa Linux. Esta conversión se lleva a cabo utilizando un circuito disparador de Schmitt [14], que asegura que la respuesta de salida del circuito de un sensor no cambie a menos que el objeto detectado se encuentre fuera del intervalo de distancias recogido. Este intervalo varía en función de las resistencias del circuito.

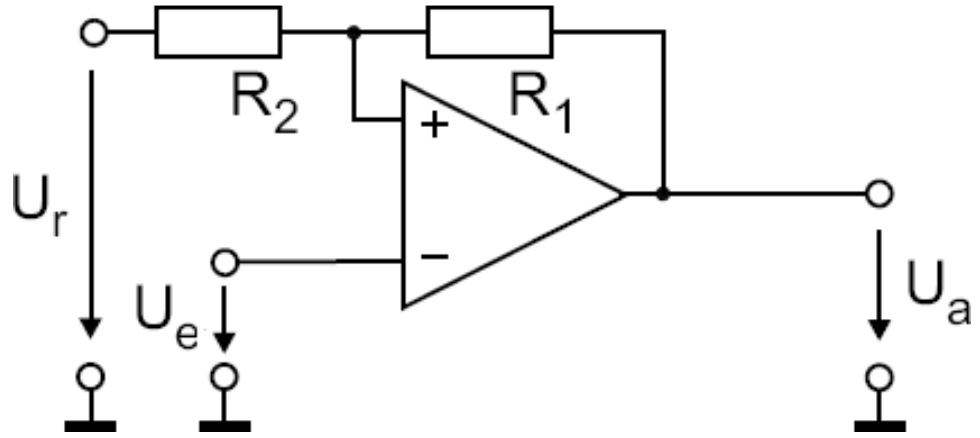


Ilustración 24 – Circuito Disparador de Schmitt

En cada circuito disparador, en lugar de una resistencia fija, en sustitución de la resistencia R2 (ver ***Ilustración 24***), se ha utilizado un potenciómetro para poder así regular el valor de la resistencia.

Modificando la resistencia de uno de los potenciómetros que forman la placa del sistema sensorial puede cambiarse el intervalo de distancias de detección de objetos del sensor de infrarrojo asociado a ese potenciómetro.

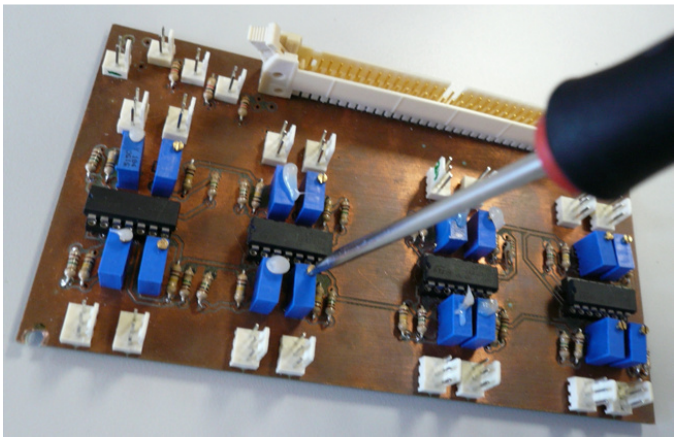


Ilustración 25 - Regulación del potenciómetro asociado a un sensor

La placa del sistema sensorial, está dotada de suficientes entradas para poder conectar 4 sensores de final de carrera y 16 sensores infrarrojos. Además, consta de un conector Molex de 64 pines para ser conectada a la placa Linux.

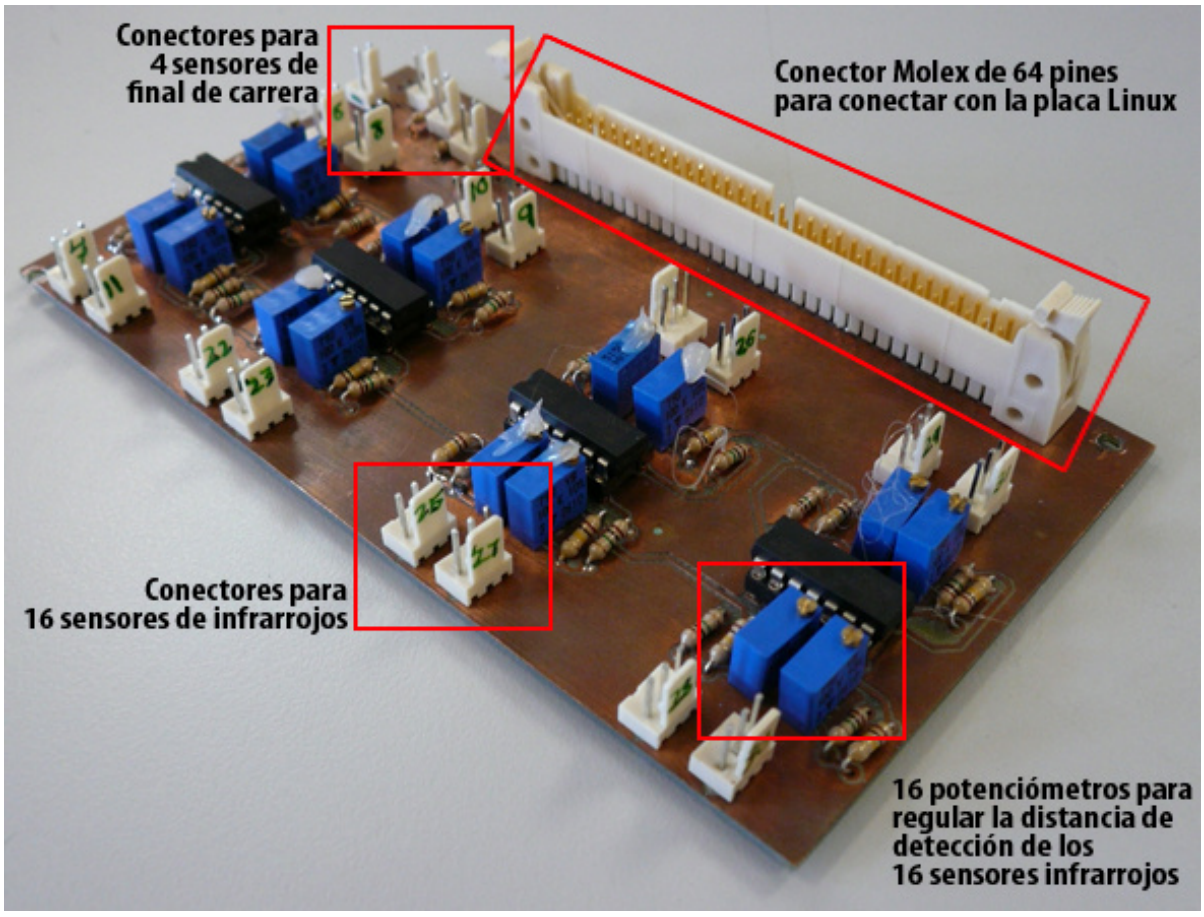


Ilustración 26 - Conexiones de la placa del sistema sensorial

3.6 Placa de drivers

Los motores empleados en el sistema de tracción son de corriente continua [15]. En este tipo de motores, el ***sentido de giro*** está determinado por el sentido de la corriente que lo atraviesa. Por otra parte, la ***velocidad*** de giro del motor viene determinada por la tensión media aplicada entre sus terminales, independientemente de la polaridad, girando a mayor velocidad según se aumente la tensión aplicada. La placa de servo-motores es la encargada de ordenar el movimiento de los motores, haciendo llegar estas órdenes a la placa de drivers, que se encarga de llevarlas a cabo invirtiendo (o no) el sentido de la polaridad de la corriente que atraviesa cada motor para determinar el sentido de giro de cada uno de ellos. De igual modo, aplica mayor o menor tensión entre los terminales de cada motor para moverlo a mayor o menor velocidad.

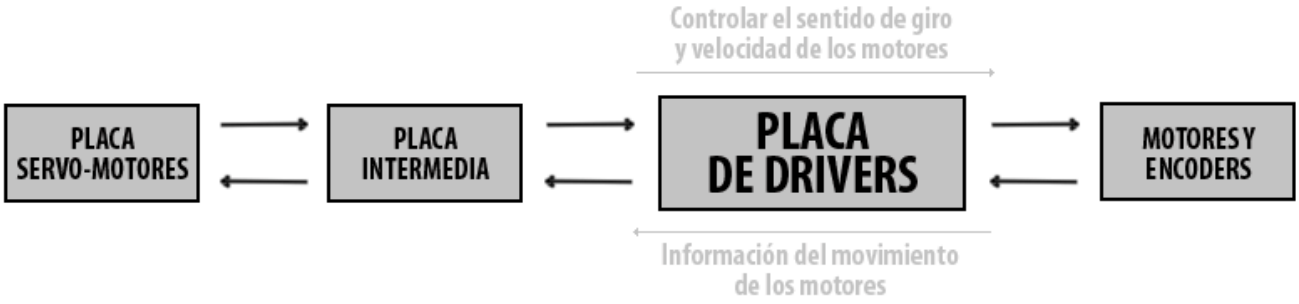


Ilustración 27 - Control de los motores que realiza la placa de drivers

Esta placa está dotada, por tanto, de los conectores necesarios para poder ser conectada a la placa intermedia y a ambos motores (ver ***Ilustracion 28***).

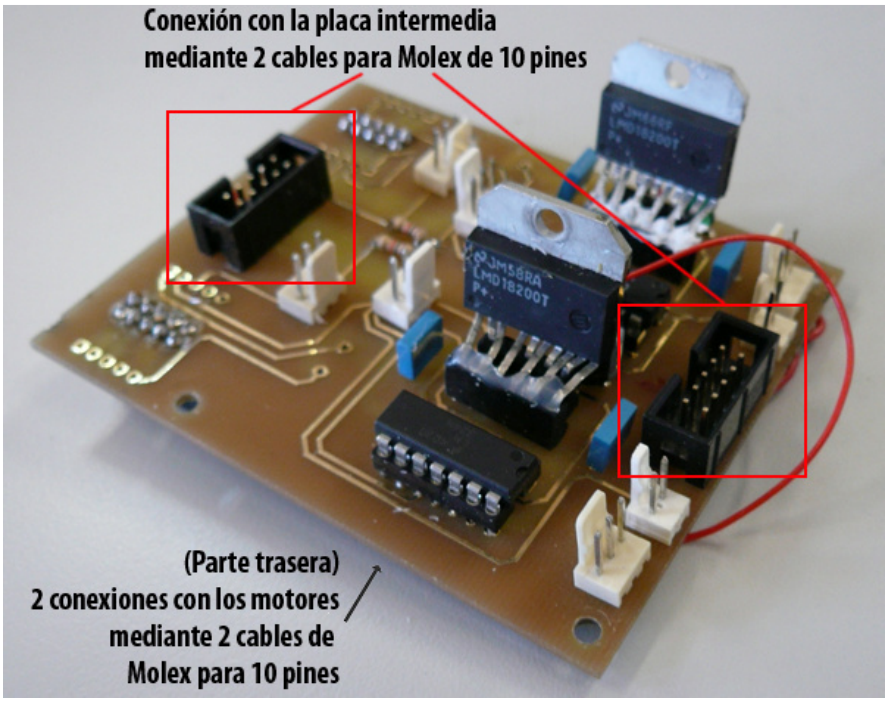


Ilustración 28 - Conexiones de la placa de drivers

3.7 Motores

Han sido empleados motores de ***corriente continua Bernio modelo MR 615 30 Q*** con reductora 1/16 [16]. Esta combinación de motor y reductora ofrece las características de par y velocidad necesarias para mover el robot dentro de unos niveles de consumo de energía moderados. Pueden alimentarse con voltajes comprendidos entre ***12 y 24 voltios***. Se ha determinado su alimentación en un voltaje de ***15 voltios***, cantidad que cumple los requerimientos de potencia y consumo fijados.

Type	Ratio	L mm	* R.P.M. no load min^{-1}	* R.P.M. S 1 min^{-1}	S1 Torque Nm	* R.P.M. S2 min^{-1}	S2 Torque Nm	Max Torque Nm	I max A
615 30Q 1/16	16	101	315	260	0,51	221	0,88	3,1	5.4

Ilustración 29 - Características de motor Bernio MR 615 30Q con reductora 1/16

En los motores de corriente continua, el ***sentido de giro*** estará determinado por el sentido de la corriente que lo atraviesa, por lo tanto, solamente será necesario invertir la polaridad entre sus terminales si queremos girar en sentido contrario. Por otra parte, la ***velocidad*** del giro del motor vendrá determinada por la tensión media aplicada entre sus terminales, independientemente de la polaridad, girando a mayor velocidad según se aumente la tensión aplicada.

Ilustración 30 - Motor Bernio MR 615 30Q

Ilustración 31 - Motores Bernio con las ruedas utilizadas acopladas

3.7.1 Encoders

Sensores ópticos situados en el eje del motor, de manera que al girar generen una serie de pulsos, haciendo llegar esta información a la placa de servo-motores, encargada del control de los motores. Se emplean para controlar el movimiento del robot. Se encargan de vigilar tanto la distancia recorrida por las ruedas, como la velocidad de giro de las mismas.

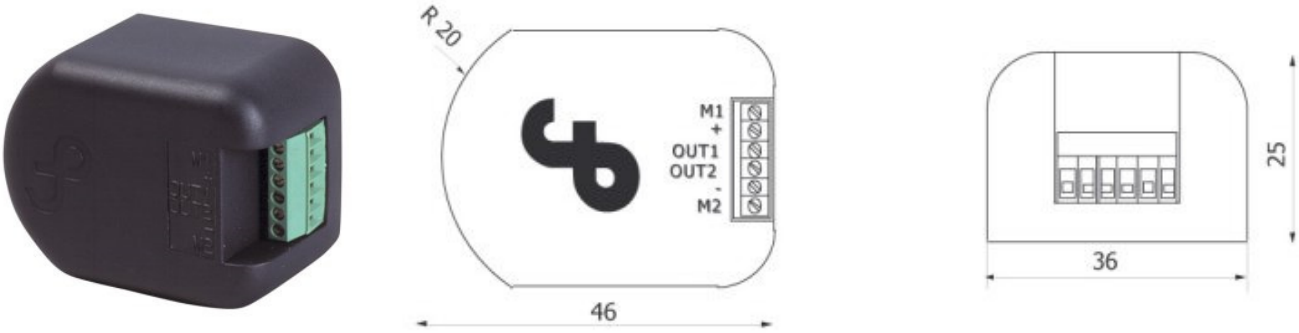


Ilustración 32 - Encoder Bernio EB 50

Funcionan en modo unidireccional (50 pulsos por vuelta) o en modo bidireccional (25 pulsos por vuelta). Se puede alimentar con voltajes comprendidos entre 6 y 24 Vcc.

Midiendo el radio (r) de la rueda utilizada con los motores, podemos obtener la longitud de la circunferencia (l) de la misma.

$$\ell = \pi \cdot 2r$$

Esta información será de gran utilidad para posicionar al robot, ya que, cada vez que el encoder cuente X número de pulsos (donde X es el número de pulsos por vuelta), sabremos que la rueda habrá dado una vuelta completa, es decir, el robot se habrá desplazado "l" unidades de desplazamiento en la dirección de esa rueda.

Ilustración 33 - Calculo del radio de la rueda utilizada

3.8 Servomotores

3.8.1 Características principales

Velocidad: 0.23 seg/60 grados (260 grados/seg)
Par de salida: 3.2 Kg-cm (0.314 N·m)
Dimensiones: 40.4 x 19.8 x 36 mm
Peso: 37.2 gr
Frecuencia PWM: 50Hz (20ms)
Rango giro: 180 grados

3.8.2 Control

Los servos se controlan aplicando una **señal PWM** [17] por su cable de control. Las señales PWM (Pulse Width Modulation, Modulación por anchura de pulso) son digitales (pueden valer 0 ó 1) y permiten que **usando un único pin de un microcontrolador podamos posicionar el servo**. Esto es una gran ventaja, porque si por ejemplo, disponemos de un microcontrolador con 8 pines de salida, podremos posicionar 8 servos.

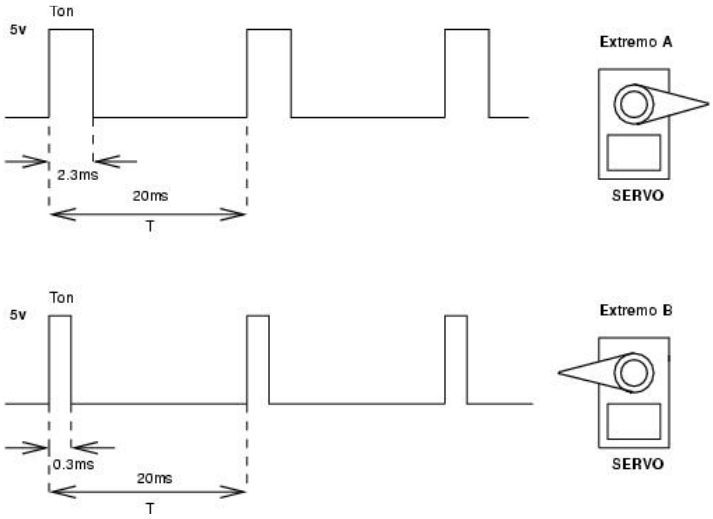


Ilustración 34 - Control de los servo-motores

Para posicionar el servo hay que aplicar una señal periódica, de **50Hz** (**20ms** de periodo). La anchura del pulso determina la posición del servo. Si la anchura es de **2.3ms**, el servo se sitúa en un extremo y si la anchura es de **0.3ms** se sitúa en el opuesto. Cualquier otra anchura entre 0.3 y 2.3 sitúa el servo en una posición comprendida entre un extremo y otro. Por ejemplo, si queremos que se sitúe exactamente en el centro, aplicamos una anchura de 1.3ms.
Cuando se deja de enviar la señal, el **servo entra en un estado de reposo**, y por tanto se podrá mover con la mano. Mientras se le aplique la señal, permanecerá fijado en su posición, haciendo fuerza para permanecer en ella.

3.8.3 Conexión

El servo dispone de un conector al que llegan **tres cables**. El **rojo** es el de alimentación (5 voltios). El **negro** es masa. Y el **blanco** es el de la señal de control.

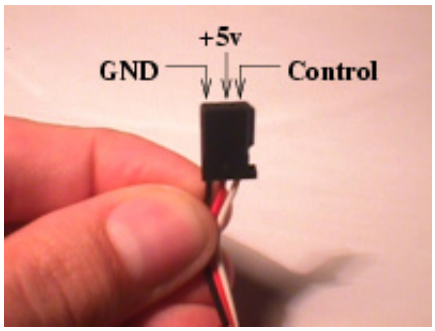


Ilustración 35 - Cableado del servo-motor

En la placa de servo-motores pueden conectarse hasta 8 servomotores. En todos ellos, la parte del conector correspondiente al cable blanco debe ir más próxima al microcontrolador.

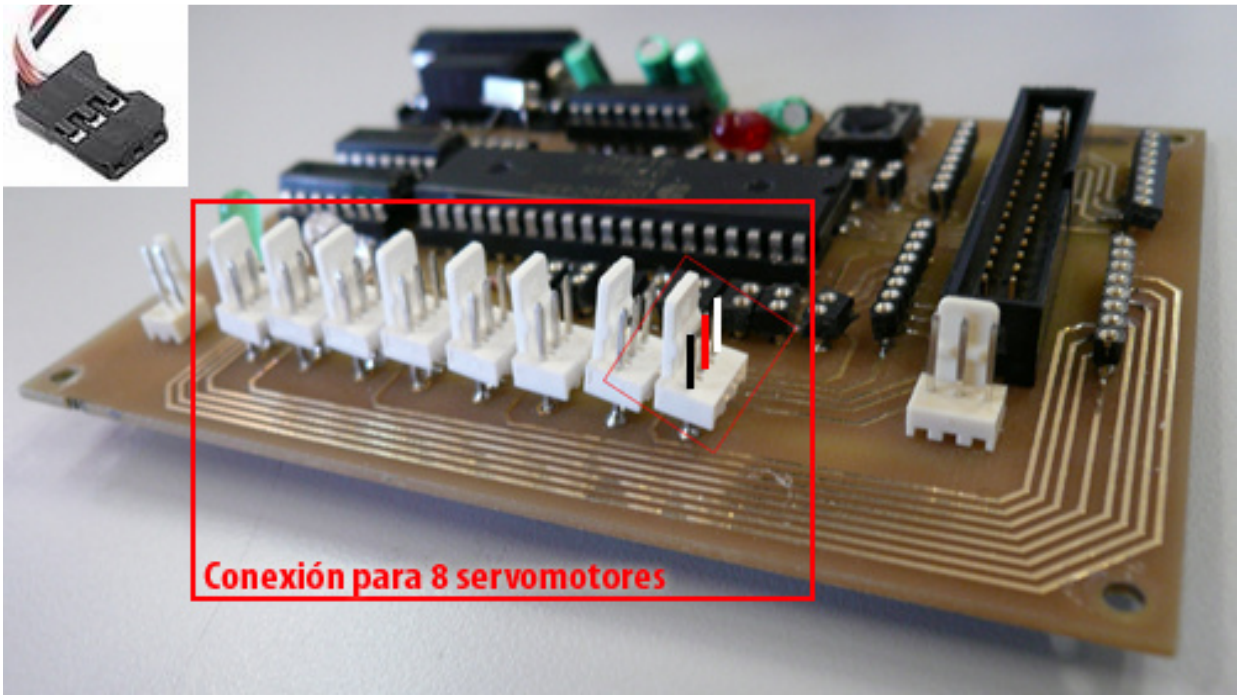


Ilustración 36 - Conexión de un servomotor a la placa de servo-motores

3.8.4 Planos

Se han empleado servomotores Futaba s3003 [18]. El plano de uno de ellos, con cotas en mm puede contemplarse en la ***Ilustracion 37***.

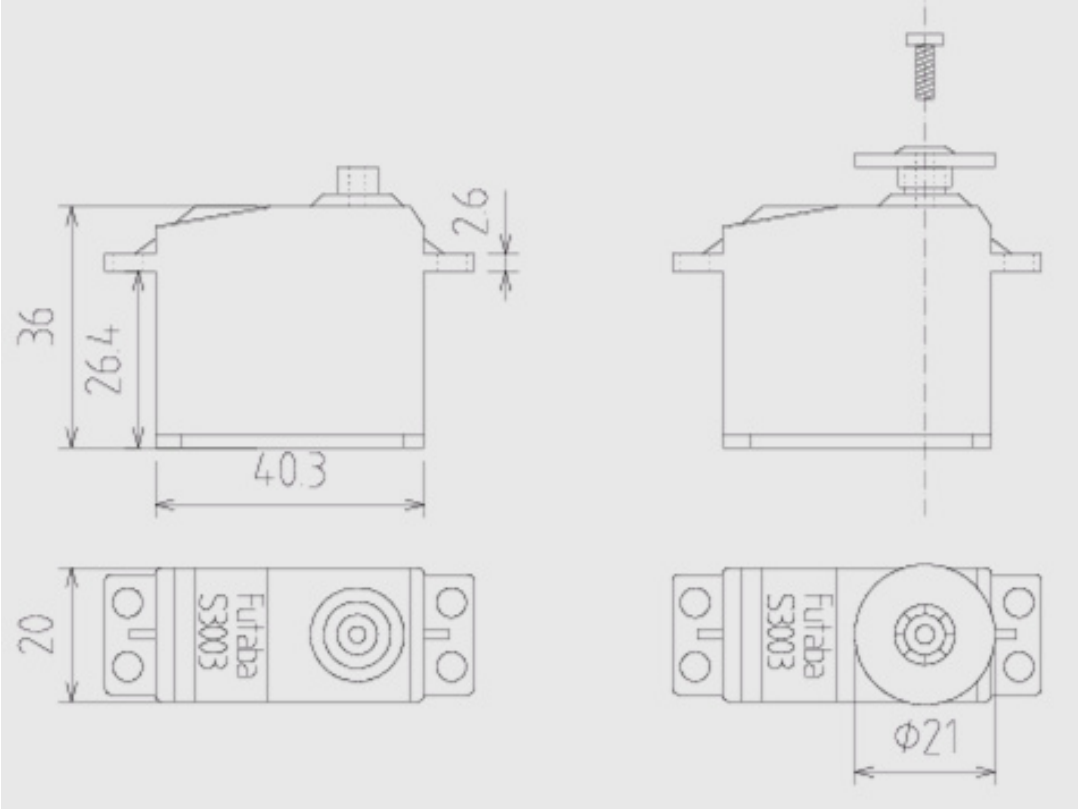


Ilustración 37 – Planos de un servomotor Futaba s3003

Ilustración 38 - Imagen de un servomotor Futaba s3003

3.9 Sensores

3.9.1 Sensores infrarrojos

El microrobot está equipado con sensores infrarrojos de la marca Sharp, modelo GP2D12 [19]. Estos sensores están conectados a la placa del sistema sensorial. De ella reciben una alimentación de 5 voltios (se puede alimentar con voltajes comprendidos entre 0.3 y 7 voltios) y a ella le envían su salida analógica.

Ilustración 39 - Sensor de infrarrojo Sharp GP2D12

Detecta objetos ubicados a distancias comprendidas entre ***10 y 80 cm***. La salida analógica del sensor dependerá de la distancia a la que se encuentre el objeto. Por lo tanto, aportan información de la distancia de otros objetos al microrobot. La salida analógica queda representada en función de la distancia al objeto en la gráfica de la ***Ilustración 40***.

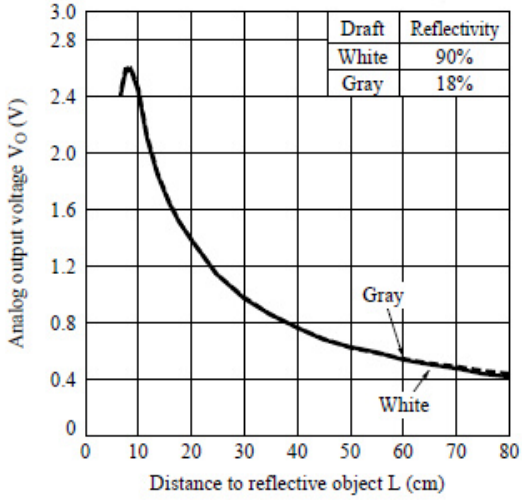


Ilustración 40 - Relación entre la salida del sensor y la distancia al objeto detectado

3.9.2 Sensores de final de carrera

También denominados como "sensores de contacto", "sensores de límite" o "bumpers". Son elementos sensoriales que permiten detectar colisiones una vez éstas se han producido. Son conmutadores de dos posiciones (activado o desactivado) que disponen de un muelle que les permite regresar a la posición de reposo (estado desactivado) si no existe elemento alguno que les mantenga pulsados (estado activado).

Se han utilizado sensores modelo SS5GL2D [20].

Ilustración 41 - Sensores de final de carrera SS5GL2D

La placa del sistema sensorial posee conexiones para poder conectarle hasta 4 sensores de final de carrera.

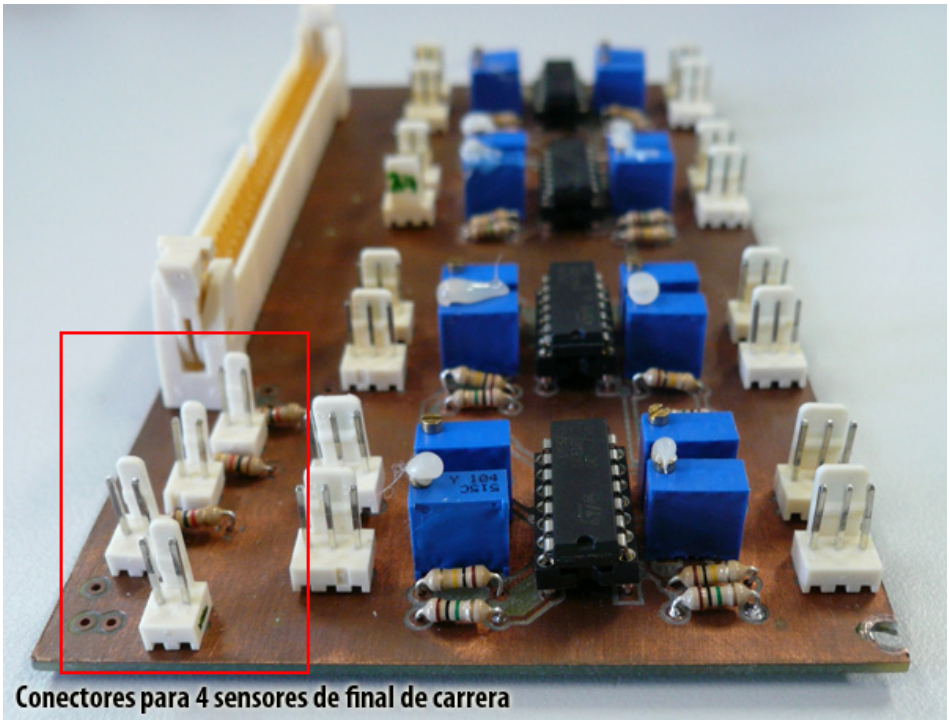


Ilustración 42 - Conectores para 4 sensores de final de carrera en la placa del sistema sensorial

3.10 Batería

Para alimentar el sistema completo, se hace uso de una batería de ***12 V*** y ***4 Ah*** de ***plomo-ácido*** (modelo NP4-12). Las baterías de plomo-ácido [21] se caracterizan por su bajo coste económico, un ciclo de autodescarga largo (pudiendo llegar hasta un año), un tiempo de carga lento (ocho horas para la batería escogida) y un tiempo de vida que oscila entre 200 y 300 ciclos de carga/descarga.

Ilustración 43 - Batería de plomo-ácido de 12 V y 4 Ah

Este tipo de batería, tiene un peso de 1.5kg, es por este motivo que se recomienda situarlas en la parte baja del microrobot para aumentar la estabilidad del mismo.

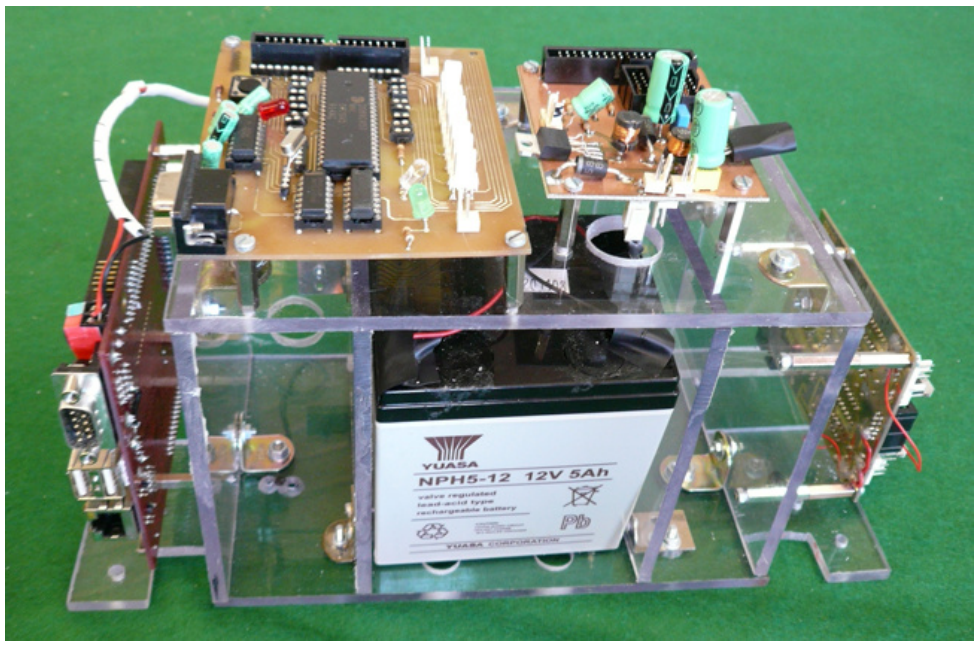


Ilustración 44 – Batería situada entre las placas de control del sistema

El voltaje de salida de la batería cuando está cargada es aproximadamente 12 voltios. Cualquier voltaje de salida inferior a esta cifra indica que no está cargada completamente. La placa intermedia incorpora un circuito de elevación de voltaje desde los 12 voltios de salida de la batería hasta los 15 voltios que requieren algunos elementos del sistema. Este circuito asegura una salida constante de 15 voltios siempre y cuando el voltaje de entrada sea superior a 7 voltios. Esta es la cifra a partir de la cual se considera que la batería ha quedado descargada. El proceso de carga de la batería se debería llevar a cabo con una fuente de alimentación programable, siguiendo el siguiente protocolo de carga.

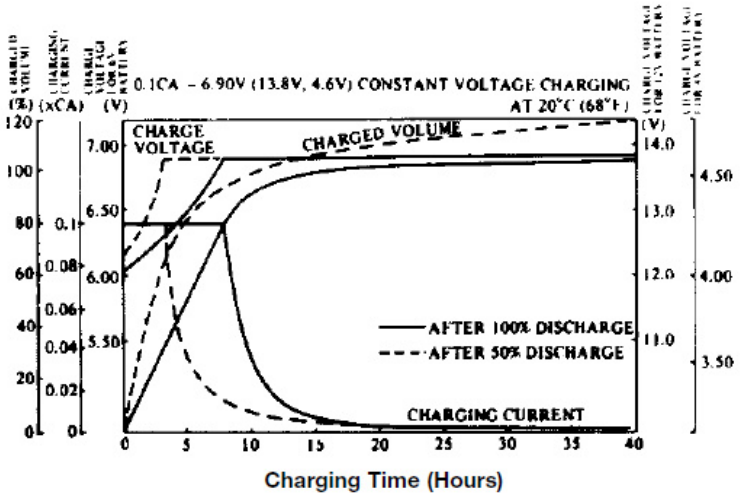


Ilustración 45 - Proceso de carga de la batería

Al finalizar el proceso, cuando el consumo de amperaje se aproxime a 0 amperios, la batería puede considerarse cargada.

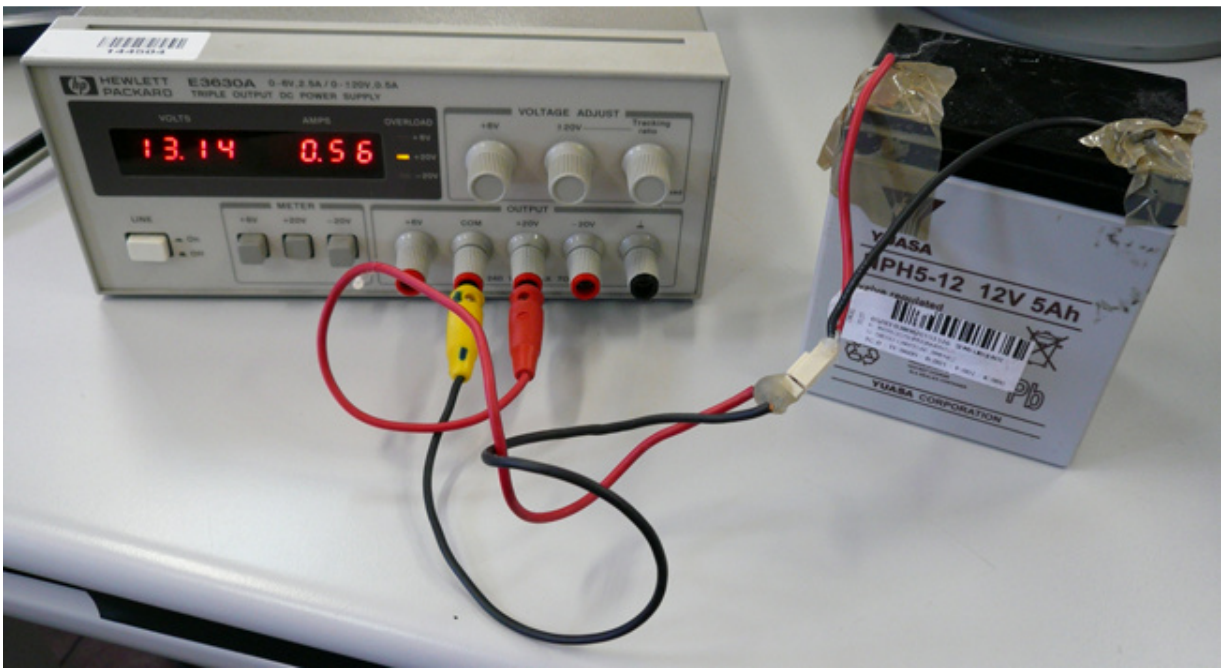


Ilustración 46 - Proceso de carga de la batería

4 Arquitectura software (Eurobot 2010)

4.1 Esquema general

El esquema de componentes software que forman la arquitectura de control del sistema queda representado en la ***Ilustración 47***.

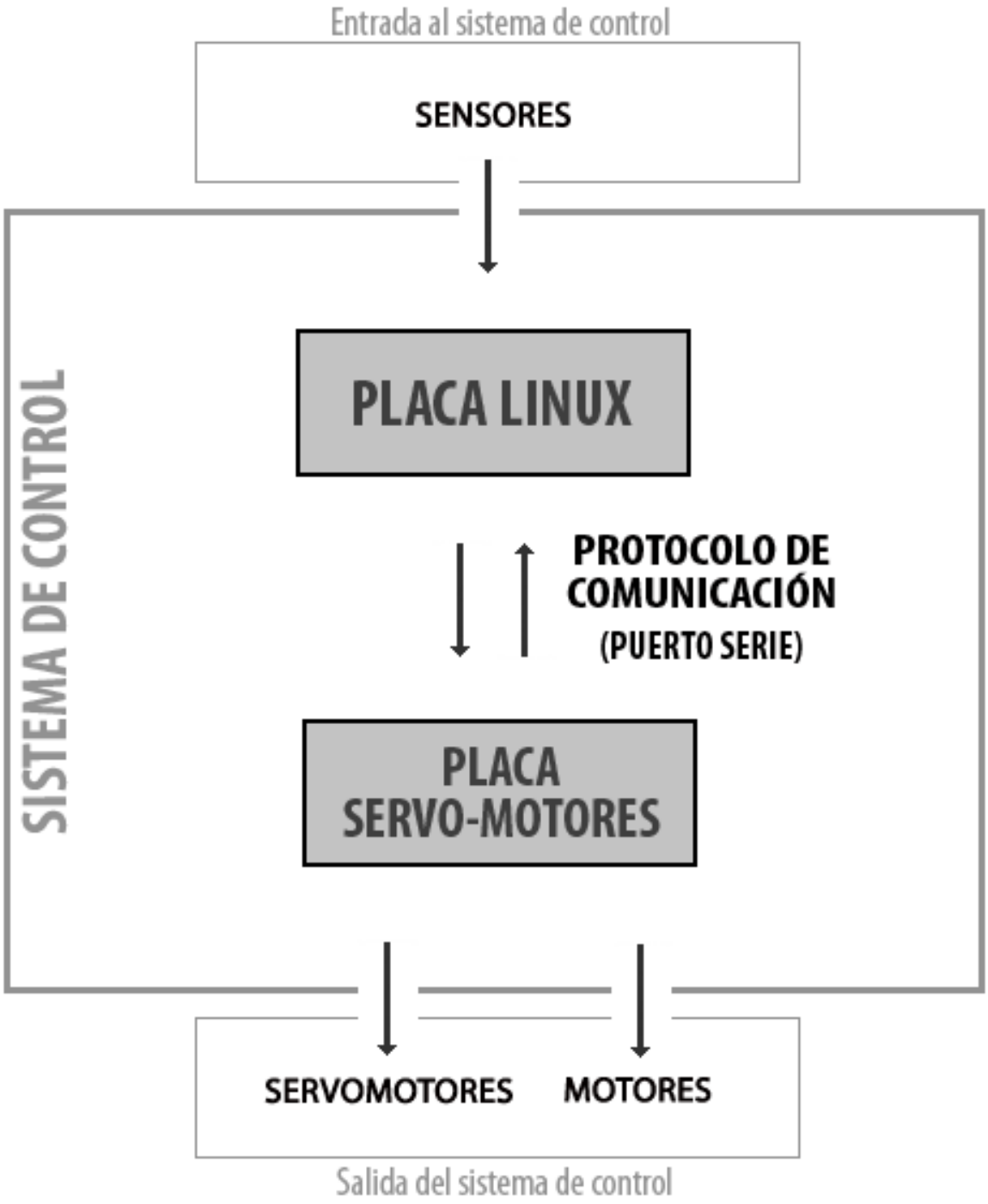


Ilustración 47 - Esquema general software

4.2 Placa Linux

Esta placa alberga el software encargado del control principal del microrobot. Al implementarse estos programas para un dispositivo capaz de operar sobre un sistema operativo Linux Debian [22], se posibilita la creación de distintos componentes que se ejecuten simultáneamente. El lenguaje de programación utilizado es el lenguaje C [23], lo que permite desarrollar tanto funciones de alto nivel (por ejemplo, funciones que constituyen una estrategia) como funciones de bajo nivel (por ejemplo, lectura de una entrada digital).

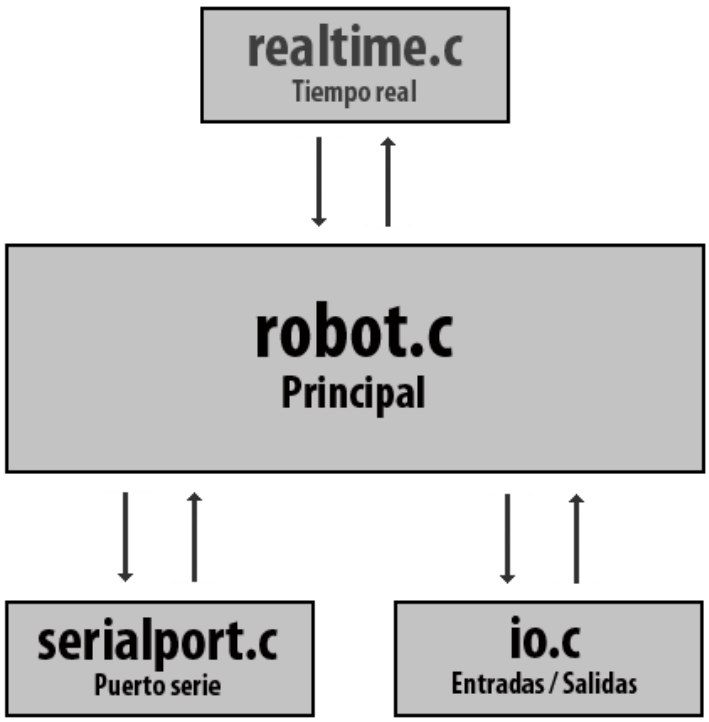


Ilustración 48 - Diagrama de componentes del software de la Placa Linux

Tiempo real (realtime.c): Recoge la funcionalidad que permite establecer prioridades entre los distintos componentes que se ejecutan simultáneamente en la placa Linux, todo ello para asegurar que ciertas funciones urgentes se ejecuten a tiempo.

Principal (robot.c): Componente principal del software de esta placa. Contiene las funciones que hacen uso del resto de componentes. Implementa las estrategias de actuación del robot.

Puerto serie (serialport.c): Componente que recoge la funcionalidad necesaria para poder establecer una comunicación con la placa de servo-motores. Implementa el protocolo de comunicaciones establecido

Entrada / Salida (io.c): Engloba las funciones de lectura y escritura de las entradas y salidas de la placa.

4.2.1 Estrategia (Eurobot 2010)

Una de las estrategias diseñadas estaba única y exclusivamente basada en los dos sensores laterales del microrobot "***Flux Capacitor***" (Eurobot 2010).

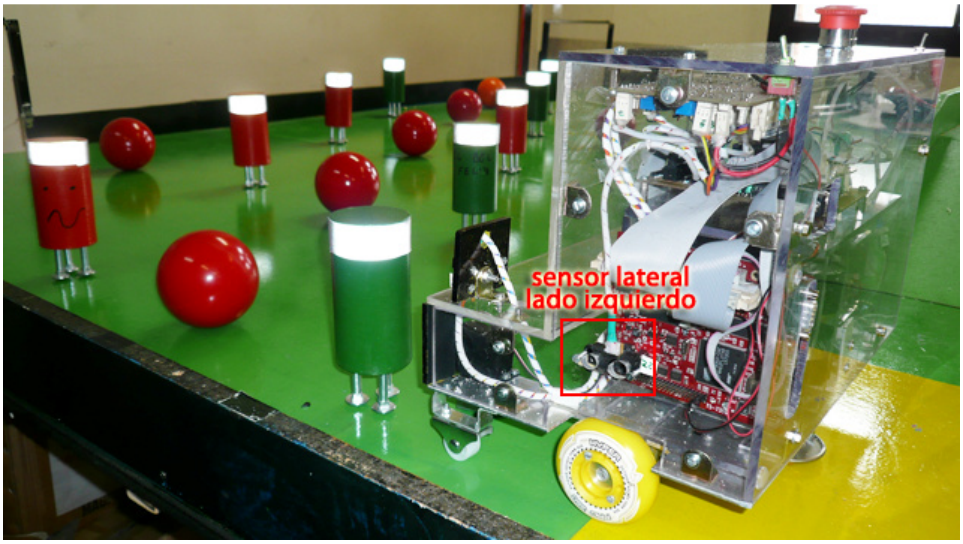


Ilustración 49 - Sensor lateral izquierdo

Atendiendo a la detección de objetos llevada por estos sensores, se sigue una trayectoria curva desde el área de inicio hasta el contenedor ubicado en el lado opuesto del campo de juego.

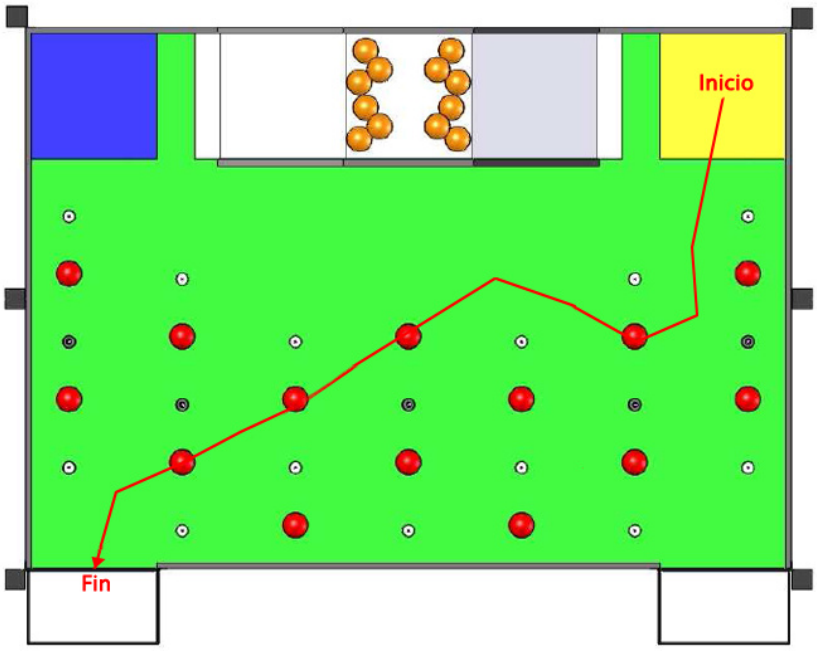


Ilustración 50 - Trayectoria seguida por el microrobot (Eurobot 2010)

Conseguir llegar del inicio al fin de este camino no resultó tarea fácil, ya que sobre el terreno de juego se encontraban elementos fijados permanentemente al campo. Los sensores laterales fueron utilizados para trazar una trayectoria curva que permitiera esquivar los citados obstáculos.

4.3 Placa servo-motores

La aplicación software implementada para ejecutarse en esta placa se encarga de las tareas de control de servomotores y motores. En primer paso será adjudicar las salidas del microcontrolador: 4 para el control de motores (2 para el sentido de giro de los motores y 2 para la señal pwm [17] que determine la velocidad de giro de cada motor) y 8 para controlar los servomotores.

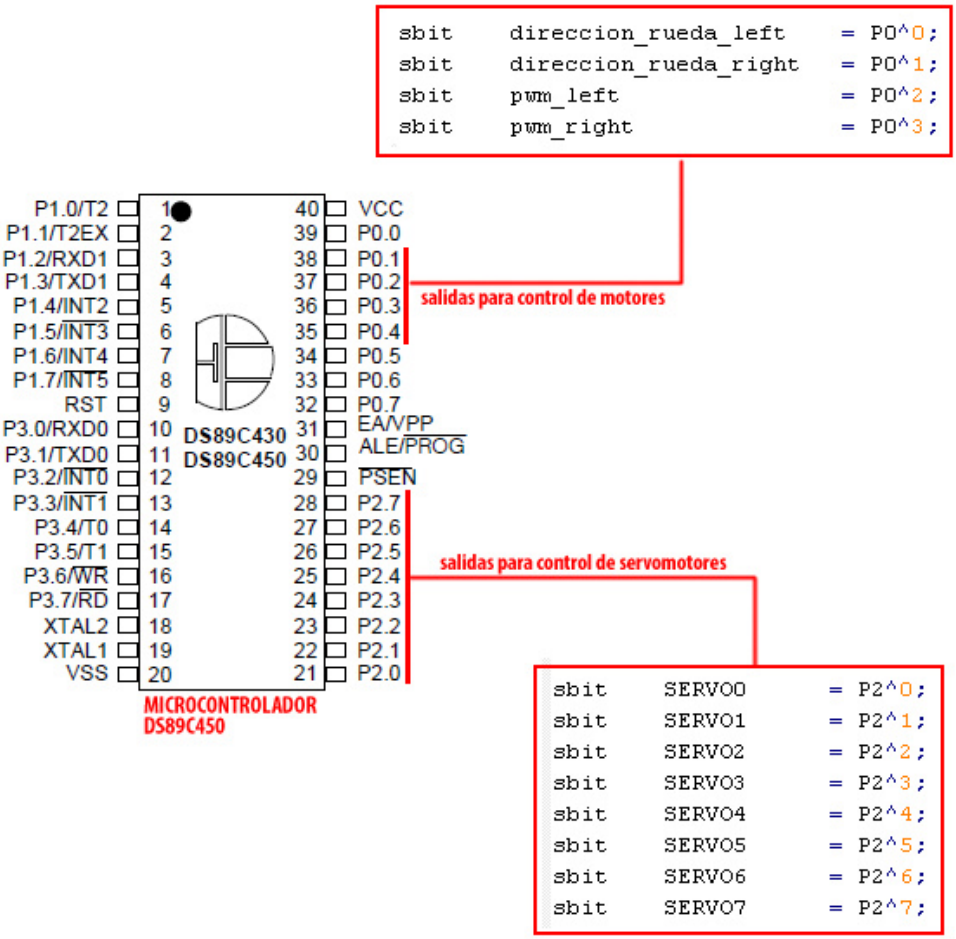


Ilustración 51 - Asignación de las salidas del microcontrolador

La función ***reset*** permite configurar las distintas opciones de la placa de servo-motores con sus parámetros por defecto. Algunas de las variables que se configuran son:

- Frecuencia del Timer a ***1 / 0.0102ms***
- Periodo de la señal pwm = ***20 ms***
- Tasa de transferencia del puerto serie = ***57600 baudios / segundo***
- Posiciones de inicio de los servomotores
- Estado de freno en ambos motores
- Inicialización de contadores (contador de pulsos de cada rueda, contador del Timer)

4.3.1 Control de servomotores

Se hace uso del timer para programar un algoritmo que permita modificar la anchura del pulso de la señal pwm generada en cada pin de salida del microcontrolador. Variando la anchura del pulso conseguimos cambiar la posición del servomotor asociado a esa salida del microcontrolador.

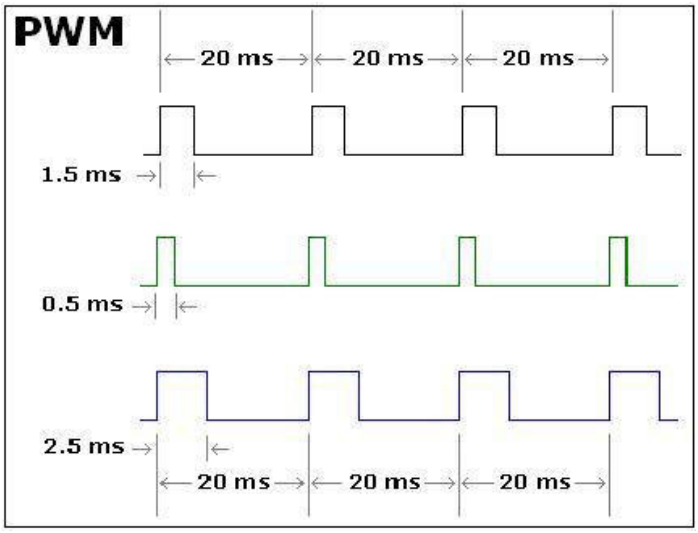


Ilustración 52 - Variación de la anchura de pulso de la señal pwm

En el caso concreto de los servomotores empleados, ***Futaba s3003*** [18], la anchura de pulso mínima será de 0.3 ms (ángulo de 0º), y la máxima de 2.3ms (ángulo de 180º).

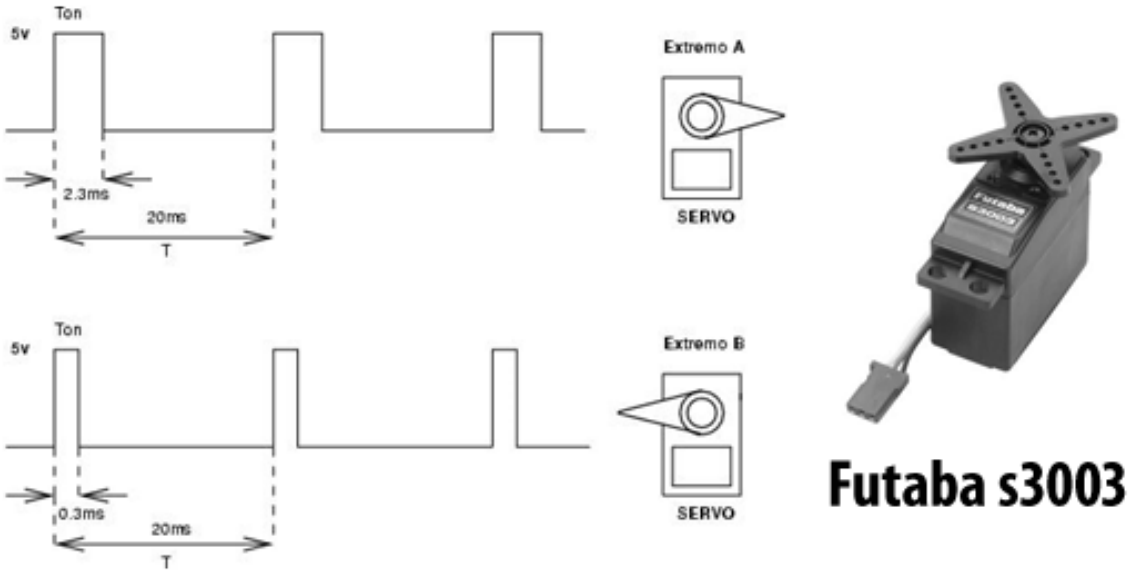


Ilustración 53 - Intervalo de anchura de pulso en servomotor Futaba s3003

El pseudocódigo del algoritmo que genera en la salida correspondiente del microcontrolador la señal pwm que permite controlar la posición de un servomotor concreto es el siguiente:

En cada interrupción provocada por el timer (cada 0.0102ms)…

Si el contador del timer < 20 ms (frecuencia de la pwm)
Si contador < valor (en ms) del angulo deseado en el servomotor
salida correspondiente del microcontrolador = 1 (5 voltios)
Fin si
Si contador < valor (en ms) del angulo deseado
salida correspondiente del microcontrolador = 0 (0 voltios)
Fin si
Fin si
Contador del timer = contador del Timer + 1
Si el contador del timer > 20 ms (ha completado un ciclo de pwm)
Asignar nuevo valor del angulo del servomotor (por si ha cambiado)
Fin si

Este pseudocódigo corresponde al control de un único servomotor. En cada interrupción del timer, se llevará a cabo este algoritmo 8 veces, una por cada uno de los servomotores a controlar. (ver apartado "***9.2 Código del programa de la placa de servo-motores***" de este documento).

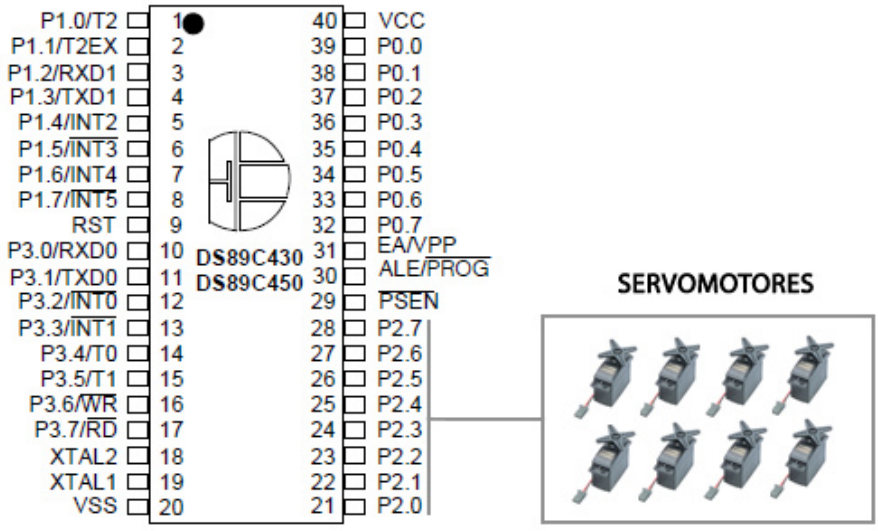


Ilustración 54 - Salidas del microcontrolador asociadas a 8 servomotores

La función ***controlar_servos*** abstrae esta tarea. Requiere dos parámetros de entrada; el primero indica el identificador del servomotor (0 a 7) al que se desea cambiar de posición (0 a 255, siendo 0 un ángulo de 0º y 255 uno de 180º)

4.3.2 Control de motores

El sentido de giro de cada motor será controlado mediante una señal cuadrada [24] (una por cada motor). En cambio, la velocidad de los motores de corriente continua será una señal pwm [17]. Se hace uso del timer [25] de la placa para programar un algoritmo que permita modificar la anchura del pulso de la señal pwm generada en los pines de salida del microcontrolador asociadas a las señales de control de los dos motores (una para el motor izquierdo y otra para el derecho). Variando la anchura del pulso conseguimos variar la velocidad de la orden de movimiento enviada a los motores.

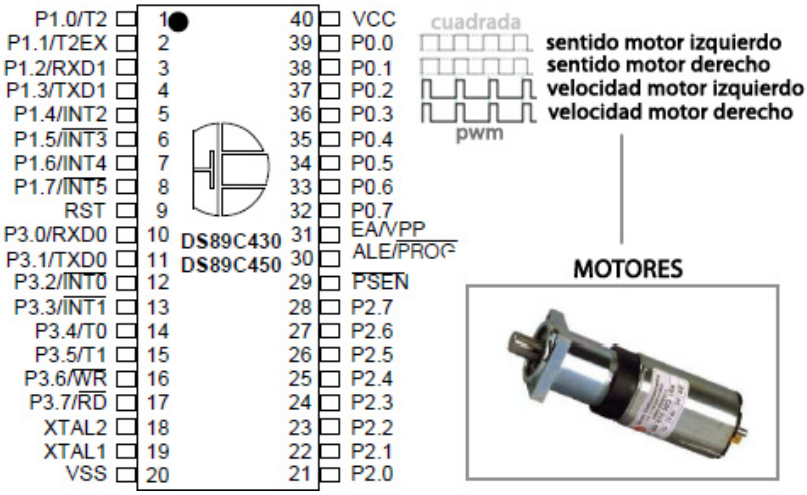


Ilustración 55 - Salidas del microcontrolador asociadas a los motores

El pseudocódigo del algoritmo que genera en la salida correspondiente del microcontrolador la señal pwm que permite controlar la velocidad del motor es el siguiente:

En cada interrupción provocada por el timer (cada 0.0102ms)…
Si el contador del timer < 20 ms (frecuencia de la pwm)
    Si contador < valor (en ms) de la velocidad deseada para ese motor
        salida correspondiente del microcontrolador = 1 (5 voltios)
    Fin si
    Si contador < valor (en ms) de la velocidad deseada para ese motor
        salida correspondiente del microcontrolador = 0 (0 voltios)
    Fin si
Fin si
Contador del timer = contador del Timer + 1
Si el contador del timer > 20 ms (ha completado un ciclo de pwm)
    Asignar nuevo valor de la velocidad (por si ha cambiado)
Fin si

Las funciones que abstraen estas acciones de control de los motores que ofrece la placa de servo-motores pueden ser ejecutadas haciendo uso del protocolo de comunicación establecido.

4.4 Protocolo de comunicación

Se detalla a continuación la especificación completa del protocolo que permite establecer una comunicación entre la placa Linux y la placa de servomotores. Puede utilizarse tanto para intercambiar datos como para enviar información de control. Desde el punto de vista de la placa de servo-motores, es una conexión pasiva, ya que es la placa Linux la encargada de comenzar cualquier intercambio de información. La placa de servo-motores permanece pues a la espera, leyendo en el puerto serie cualquier byte de entrada proveniente de la placa Linux. Cuando recibe un nuevo byte, dependiendo del valor en código ASCII [26] de éste, ejecutará una función u otra. Una vez finalice la ejecución de la función correspondiente, enviará de vuelta el byte de asentimiento, que en el presente caso se ha establecido en el carácter "**#**".

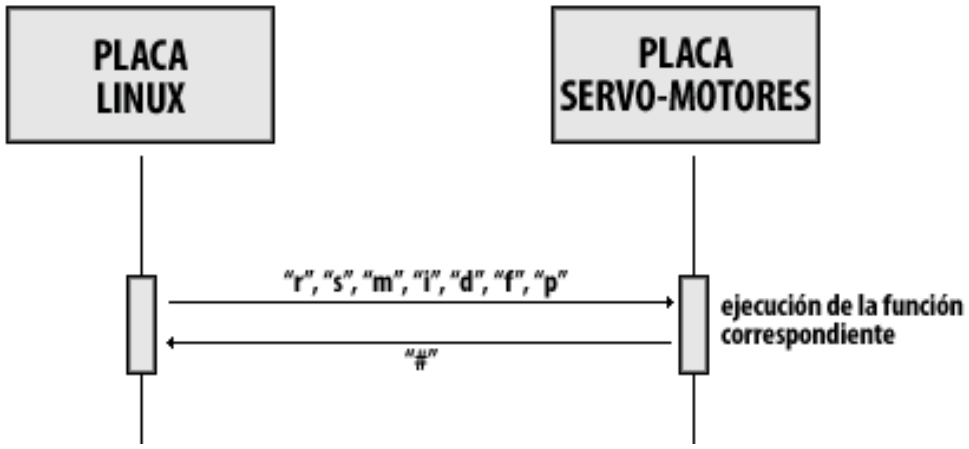


Ilustración 56 - Diagrama de ejecución de la comunicación entre placas

Los únicos bytes recibidos que se tendrán en cuenta son:

- ***Carácter "r"***: ejecutará la función "***reset***" que configura la placa de servo-motores con los parámetros por defecto.
- ***Carácter "s":*** ejecutará la función "***controlar_servos***", empleada para modificar la posición de un servomotor. Esta función leerá por el puerto serie dos nuevos bytes, el primero de ellos será el identificador del servo que se quiere cambiar de posición (valor entre 0 y 7), el segundo byte corresponde a la nueva posición que debe establecerse en ese servomotor (valor entre 0 y 255).
- ***Carácter "m":*** ejecutará la función "***controlar_motores***", utilizada para activar el movimiento de los motores. Esta función leerá por el puerto serie dos nuevos bytes, el primero corresponderá a la velocidad de movimiento de la rueda izquierda y el segundo a la velocidad de la rueda derecha. Se han establecido 26 posibles valores de velocidad, representadas por las letras del alfabeto (A-Z) siendo A la menor velocidad y Z la mayor. Si estas letras se reciben en mayúscula (A-Z), debe tomarse el sentido de ese motor hacia delante. En caso de recibirse en minúscula (a-z) deberá establecerse el sentido del motor hacia atrás.

- ***Carácter "i":*** ejecutará la función "***consultar_encoder_izquierdo***", que solicita el envío de la cuenta (desde la última consulta) del número de pulsos que ha contabilizado el encoder izquierdo. La placa de servo-motores enviará esta información codificada en dos bytes.
- ***Carácter "d":*** ejecutará la función "***consultar_encoder_derecho***", que solicita el envío de la cuenta (desde la última consulta) del número de pulsos que ha contabilizado el encoder derecho. La placa de servo-motores enviará esta información codificada en dos bytes.
- ***Carácter "f":*** ejecutará la función "***cambiar_factor_correccion***" que permite aumentar/disminuir el desplazamiento de un motor en función del otro, para poder corregir tendencias de giro (causadas por imperfecciones en la construcción física del microrobot) en trayectorias que deberían ser rectas. Esta función leerá por el puerto serie dos nuevos bytes, el primero correspondiente al factor de corrección de la rueda izquierda, y el segundo al de la rueda derecha. Estos valores serán siempre positivos, ya que para obtener un valor negativo es suficiente con enviarlo para la rueda contraria.
- ***Carácter "p":*** ejecutará la función "***muévete_pasos***" que permite ordenar el movimiento hacia delante de ambos motores durante un determinado número de pulsos. La función leerá por el puerto serie un nuevo byte, correspondiente al número de pasos que deben desplazarse los motores.

```
while (1) {
    scanf("%c", &accion);

    if(accion == 'r') {
        reset();
        printf("%c", CARACTER_ACK);
    } else if (accion == 's') {
        controlar_servos ();
        printf("%c", CARACTER_ACK);
    } else if (accion == 'm') {
        controlar_motores ();
        printf("%c", CARACTER_ACK);
    } else if (accion == 'i') {
        consultar_encoder_izquierdo();
    } else if (accion == 'd') {
        consultar_encoder_derecho();
    } else if (accion == 'f') {
        cambiar_factor_correccion();
        printf("%c", CARACTER_ACK);
    } else if (accion == 'p') {
        muevete_pasos();
        printf("%c", CARACTER_ACK);
    }
}
```

Ilustración 57 – Rutina de la comunicación pasiva desde la placa de servo-motores

5 Manejo de la placa de servo-motores

5.1 Aplicación gráfica para controlar servomotores y motores

Se ha implementado el protocolo de comunicaciones en una aplicación gráfica para facilitar el uso de la placa de servo-motores, no sólo por parte de los alumnos informáticos sino también para el resto de componentes del equipo, alumnos pues de otras titulaciones (Ingeniería Técnica especialidad de mecánica o electrónica).

Esta aplicación ha sido implementada en lenguaje de programación ***C#*** [27], para Sistema Operativo ***Windows XP*** [28]. Ha sido denominada "***ServoController***" y puede encontrarse instalada y lista para ejecutar en el ordenador utilizado para el desarrollo del presente proyecto, ubicado en el Laboratorio de Sistemas Inteligentes, aula 1.3.B16 del edificio Betancourt, de la Universidad Carlos III de Madrid.

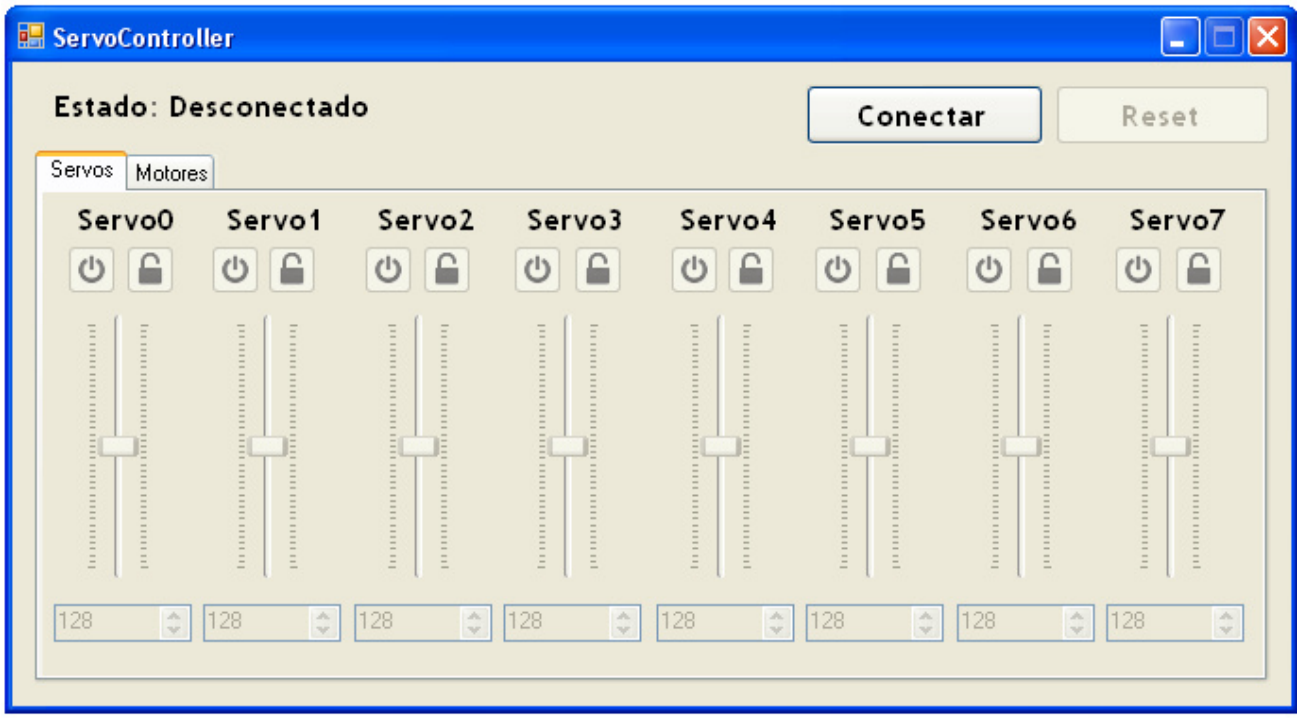


Ilustración 58 - Interfaz gráfica de la aplicación implementada: ServoController

Las acciones o casos de uso que cualquier usuario puede llevar a cabo en esta aplicación quedan representadas en la ***Ilustración 59***.

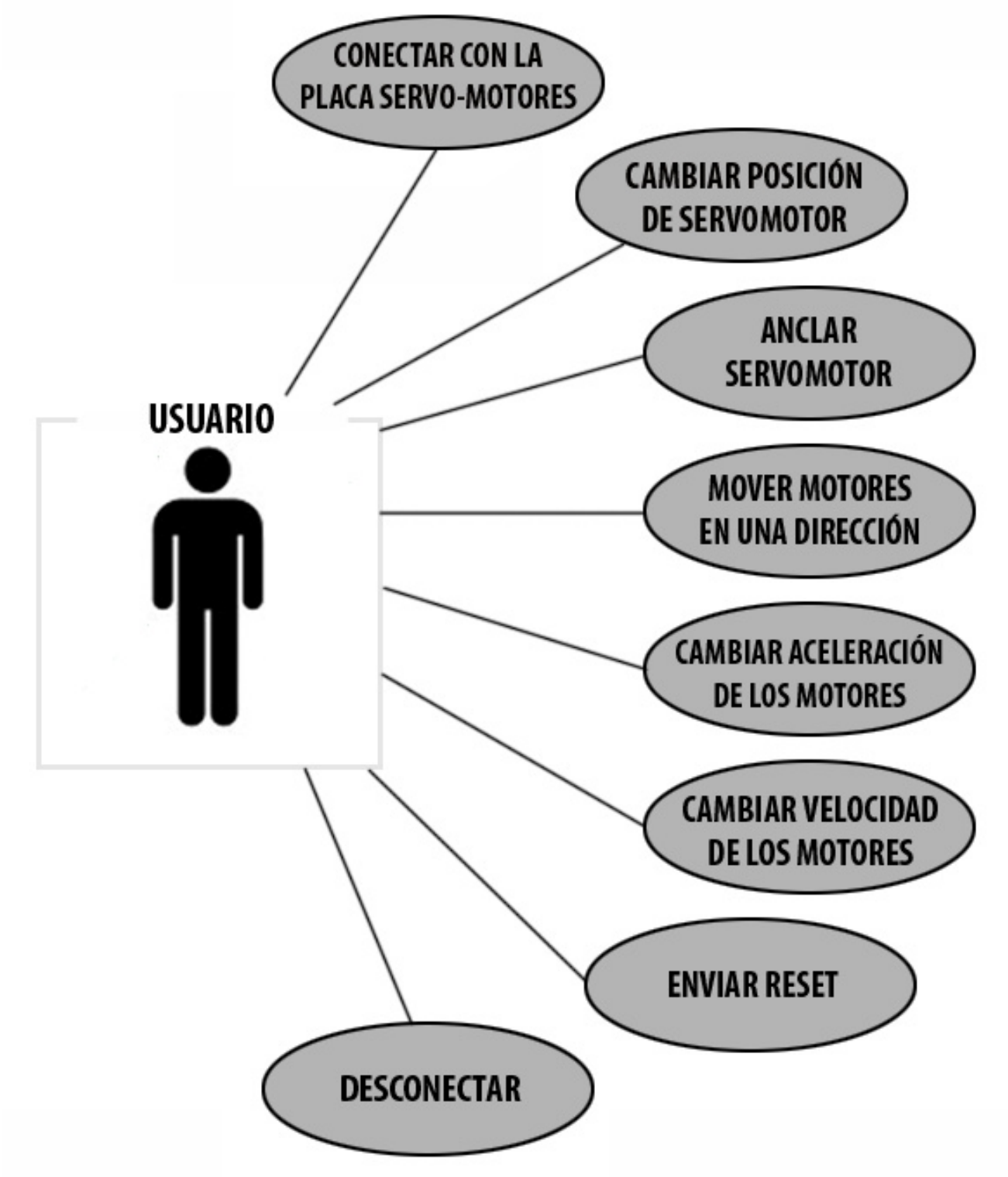


Ilustración 59 - Acciones o casos de uso sobre la aplicación Servocontroller

La realización de algunas de estas acciones implica el envío de una orden a la placa de servo-motores, mientras que otras de ellas no.

Al cambiar la posición de un servo, activar el movimiento de los motores o acción de reset, se enviará la orden correspondiente a la placa de servo-motores.

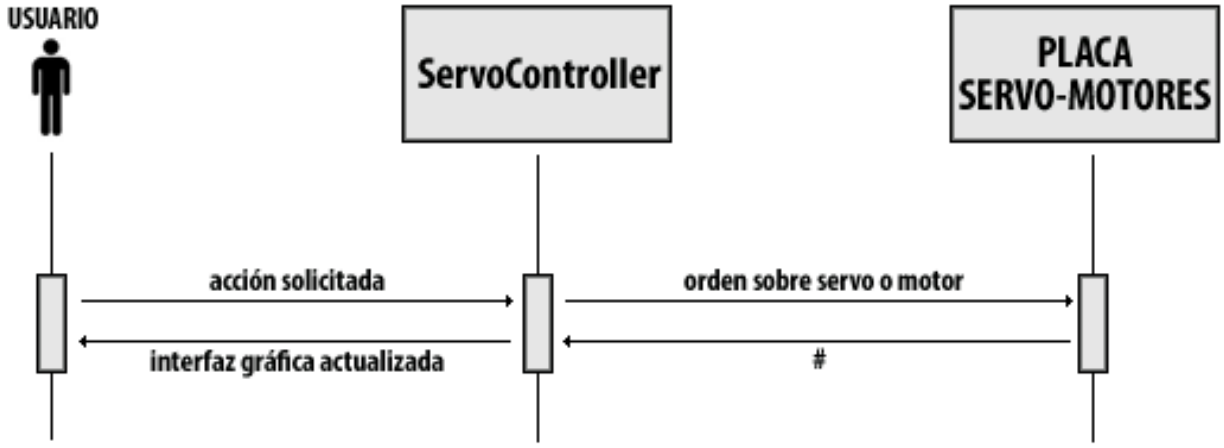


Ilustración 60 - Diagrama de ejecución de acciones que impliquen el envío de una orden

Al anclar un servomotor, al configurar la aceleración o la velocidad de los motores, no se enviará orden alguna.

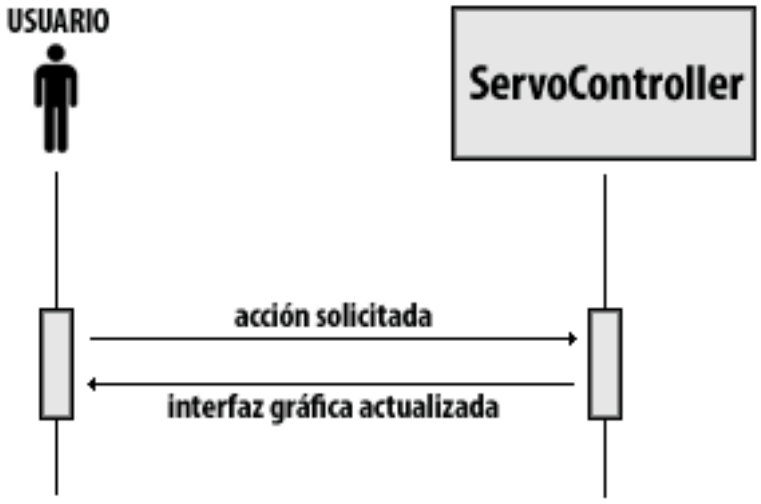


Ilustración 61 - Diagrama de ejecución de acciones que no impliquen envío alguno

5.1.1 Conectar con la placa de servo-motores

Para comenzar la comunicación entre el ordenador y la placa de servo-motores, hacer uso del botón "***Conectar***" ofrecido en la interfaz gráfica de la aplicación. El estado cambiará de "***Desconectado***" a "***Conectado***" si la conexión consigue establecerse correctamente.

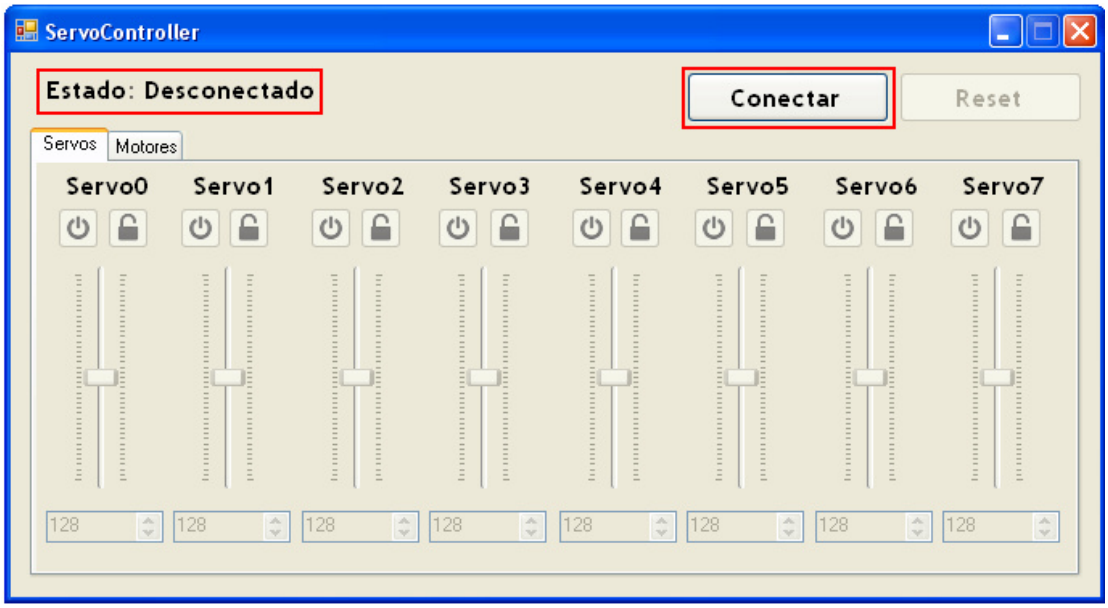


Ilustración 62 - Acción "Conectar" en aplicación ServoController

5.1.2 Cambiar posición de servomotor

Los 8 servomotores que pueden controlarse con la placa están numerados de 0 a 7. Para controlar uno de ellos, es necesario previamente activarlo utilizando el botón establecido para tal efecto.

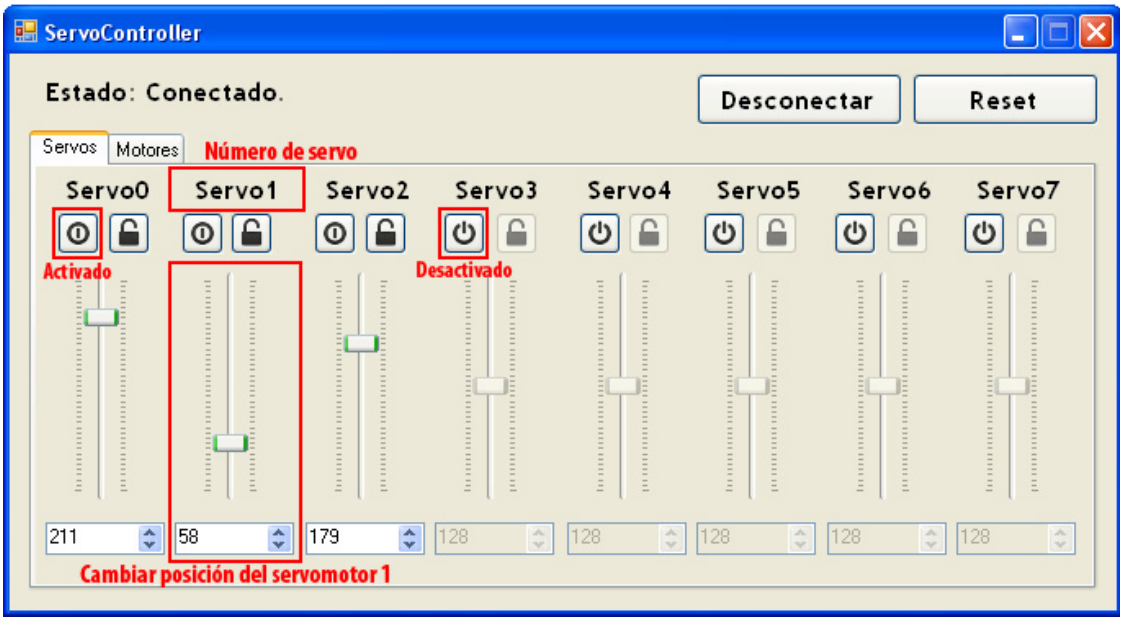


Ilustración 63 - Interfaz de control para cambiar posición de un servomotor

Una vez activado, quedarán habilitados los controles de cambio de posición de ese servomotor. Se ofrecen dos tipos de control: ***barra deslizante*** e ***introducción numérica***. El rango de posiciones de un servomotor Futaba s3003 [18] es de 0º a 180º. El rango de valores para introducir en el control de posición de un servomotor en la aplicación ServoController va de 0 a 255, correspondiendo el valor 0 a la posición 0º y el valor 255 a la posición 180º. Valores intermedios dentro de este intervalo corresponderán a posiciones con ángulos comprendidos entre los ángulos mínimo (0º) y máximo (180º).

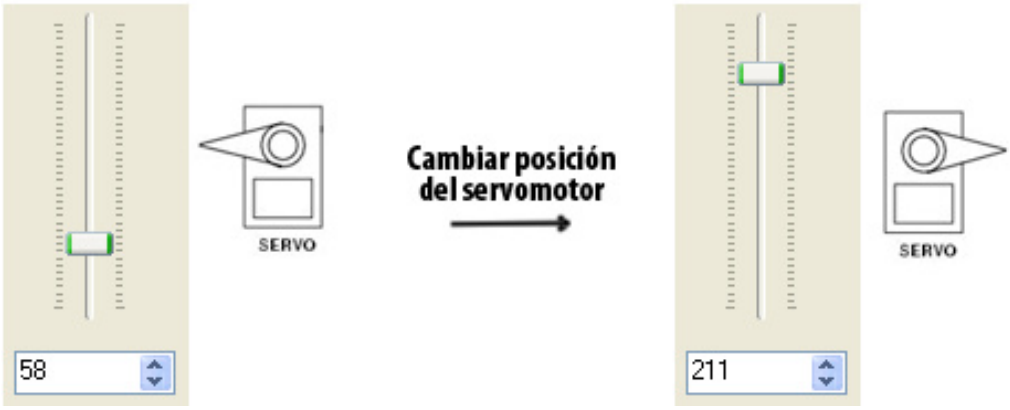


Ilustración 64 - Control de servomotores mediante barra deslizante o introducción numérica

5.1.3 Anclar servomotor

Algunos elementos del microrobot requieren el desplazamiento simultáneo de varios servomotores. Es por este motivo que se ha incluido en la aplicación ServoController la funcionalidad necesaria para poder cambiar la posición de varios servomotores al unísono. Basta con activar el botón de anclaje existente en el control de cada servomotor.

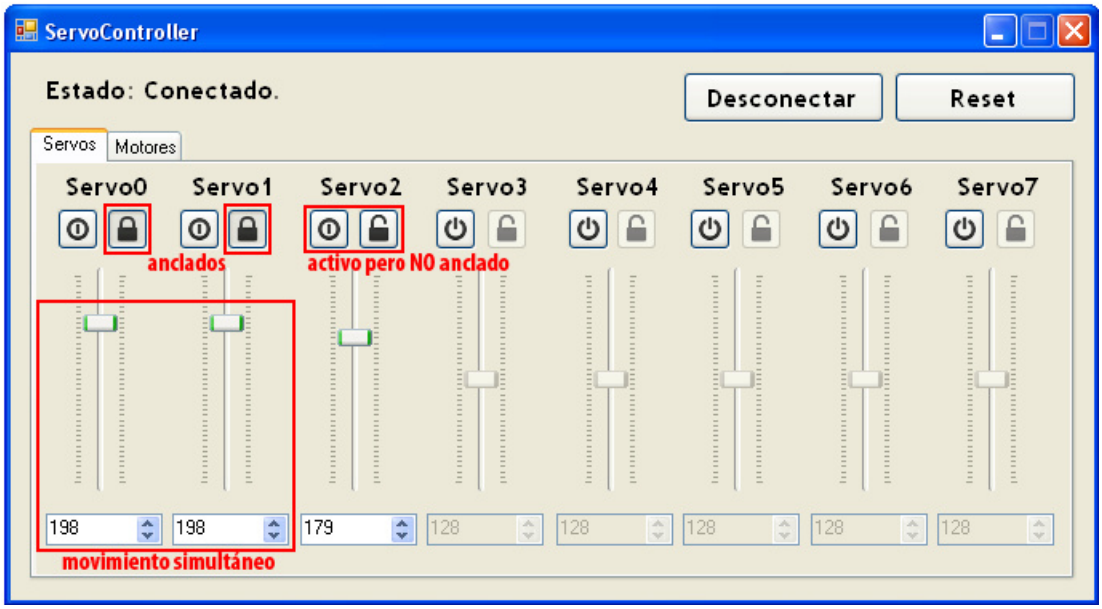


Ilustración 65 - Anclado de varios servomotores para movimiento simultáneo

5.1.4 Mover motores en una dirección

Para acceder al control de motores, hay que hacer click sobre la etiqueta de control de motores.

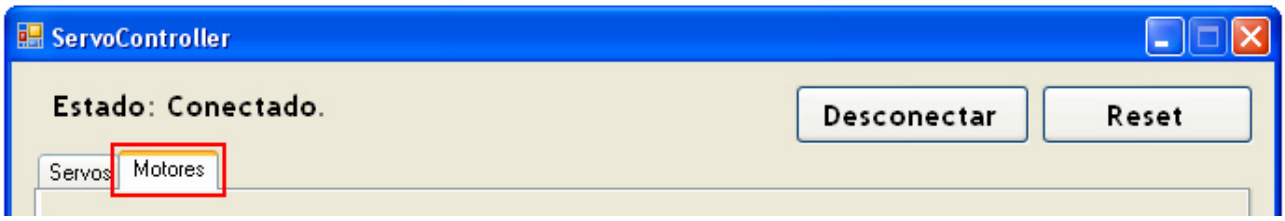


Ilustración 66 - Etiqueta de control de motores

Una vez situados en el control de motores, para activar el movimiento en una u otra dirección, basta hacer uso de la flecha correspondiente. Se ofrecen 4 direcciones de movimiento: adelante, atrás, izquierda y derecha. El movimiento se realiza a la velocidad y aceleración establecidas en el momento de pulsar en una de las flechas.

- ***Adelante***: envía la orden para el movimiento de ambas ruedas hacia adelante
- ***Atrás***: ordena el movimiento de ambas ruedas hacia atrás.
- ***Derecha***: ordena el movimiento de la rueda derecha hacia adelante y de la rueda izquierda hacia atrás.
- ***Izquierda:*** ordena el movimiento de la rueda izquierda hacia adelante y de la rueda derecha hacia atrás.

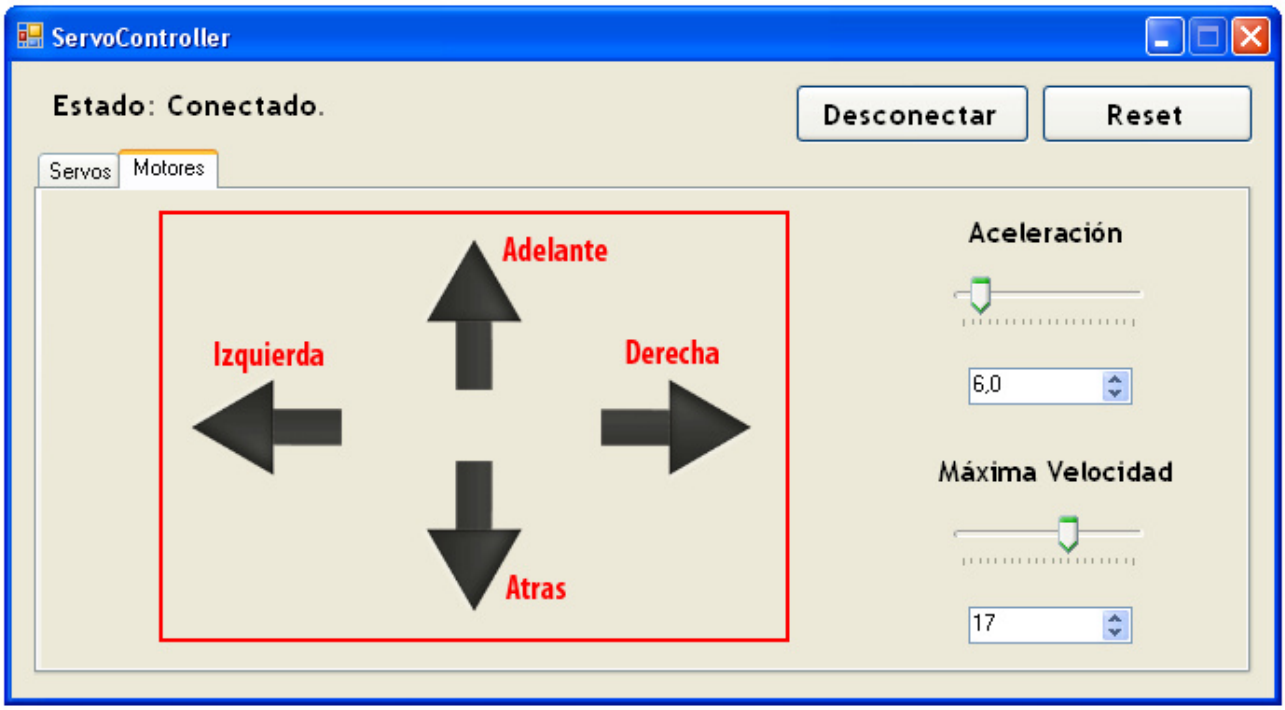


Ilustración 67 - Flechas de control de movimiento de motores

5.1.5 Cambiar velocidad de los motores

La velocidad de los motores determina el desplazamiento por unidad de tiempo del microrobot. La aplicación ServoController ofrece la posibilidad de modificar este parámetro.

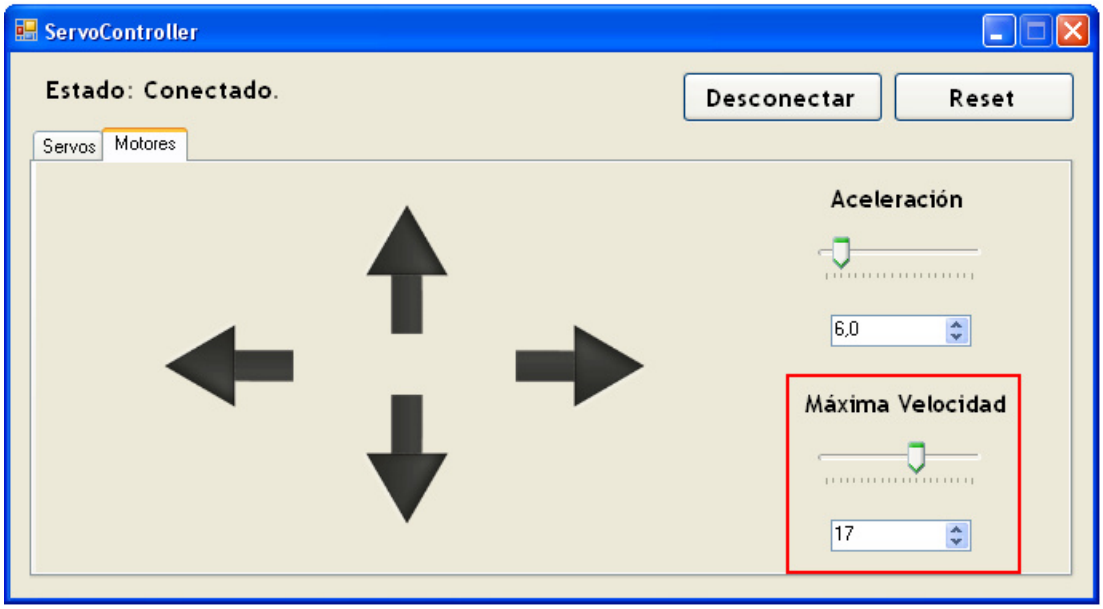


Ilustración 68 - Modificar parámetro velocidad

5.1.6 Cambiar aceleración de los motores

La aceleración de los motores determina el aumento de la velocidad por unidad de tiempo del microrobot en su desplazamiento. La aplicación ServoController ofrece la posibilidad de modificar este parámetro.

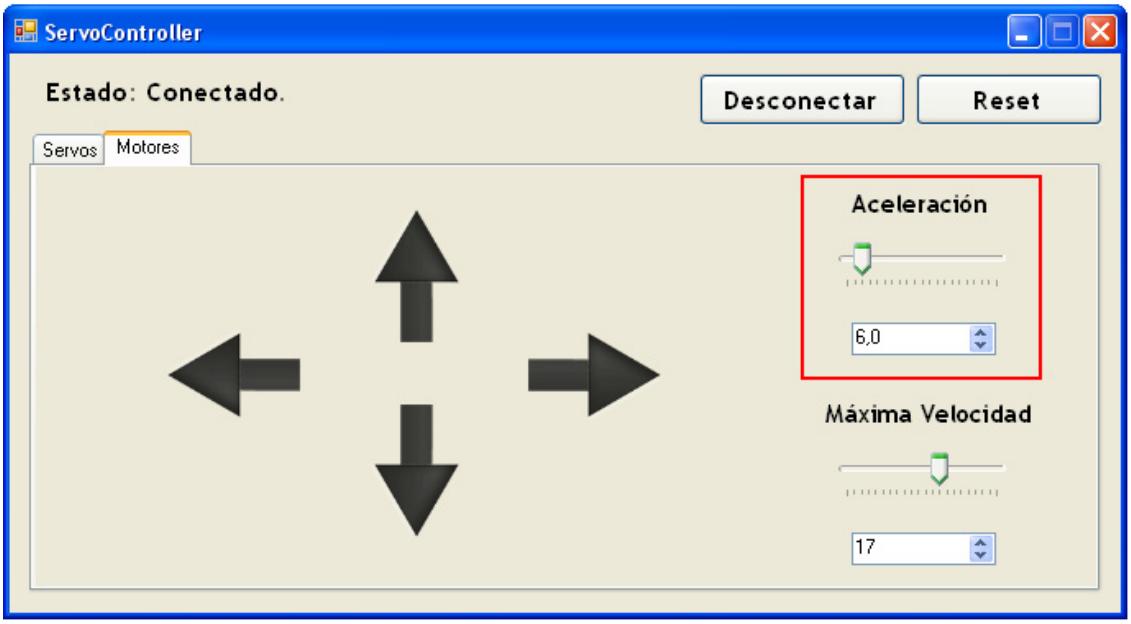


Ilustración 69 - Modificar parámetro de aceleración

5.1.7 Enviar Reset

La placa de servo-motores puede llevar a cabo una acción de reinicio en la que restablece su configuración a los parámetros por defecto. El protocolo de comunicaciones contempla la posibilidad de enviar una orden de reset a la placa de servo-motores.

La aplicación ServoController ofrece también esta posibilidad, habilitando un botón para tal efecto, que puede utilizarse en cualquier momento desde cualquier sección del programa.

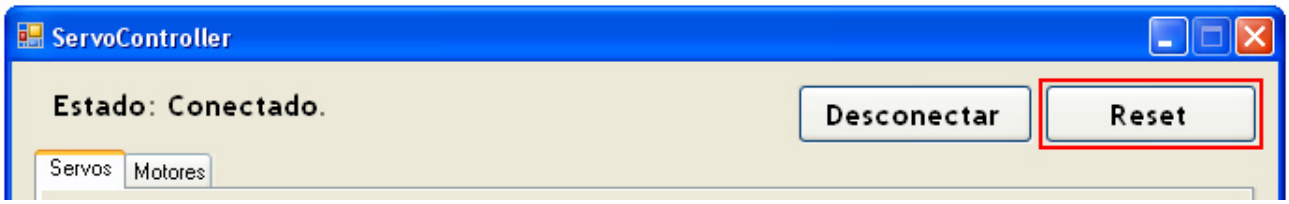


Ilustración 70 - Reset de la placa de servo-motores

5.1.8 Desconectar

Para finalizar la comunicación entre el ordenador y la placa de servo-motores, en cualquier momento puede hacerse uso del botón "***desconectar***" ubicado en la parte superior derecha de la interfaz gráfica de la aplicación ServoController.

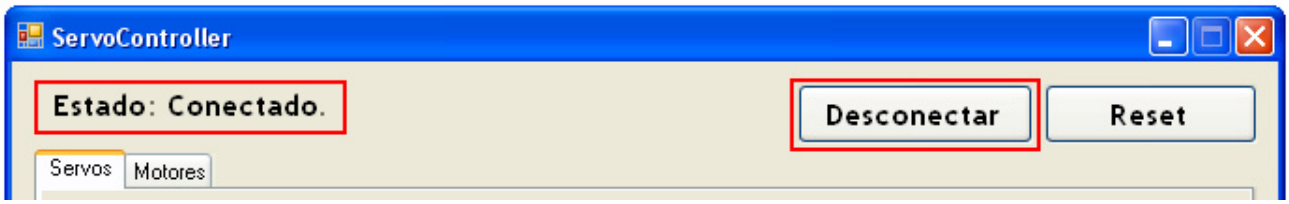


Ilustración 71 - Finalizar la conexión entre el ordenador y la placa de servo-motores

5.2 Manual de uso de herramientas software asociadas

En este apartado se pretende componer un ***manual*** que permita a cualquier alumno encargado de la programación del robot en futuras ediciones de Eurobot adquirir fácilmente el conocimiento básico suficiente para poder hacer uso de las herramientas de software necesarias para trabajar con los distintos elementos programables que forman el robot.

En el apartado "***3.3 Placa de servo-motores***" del presente documento se encuentra una especificación detallada de las características de la placa de servo-motores. Para poder trabajar con esta placa será necesario instalar previamente en el ordenador a utilizar, los programas ***Keil µvision2*** [29] y ***Microcontroller Tool Kit (MTK2)*** [30].

5.2.1 Cómo crear un programa para la placa de servo-motores

Se ha utilizado el entorno de programación ***Keil µvision2*** [29].

Una vez abierto el programa, el ***primer paso*** será crear un nuevo proyecto.

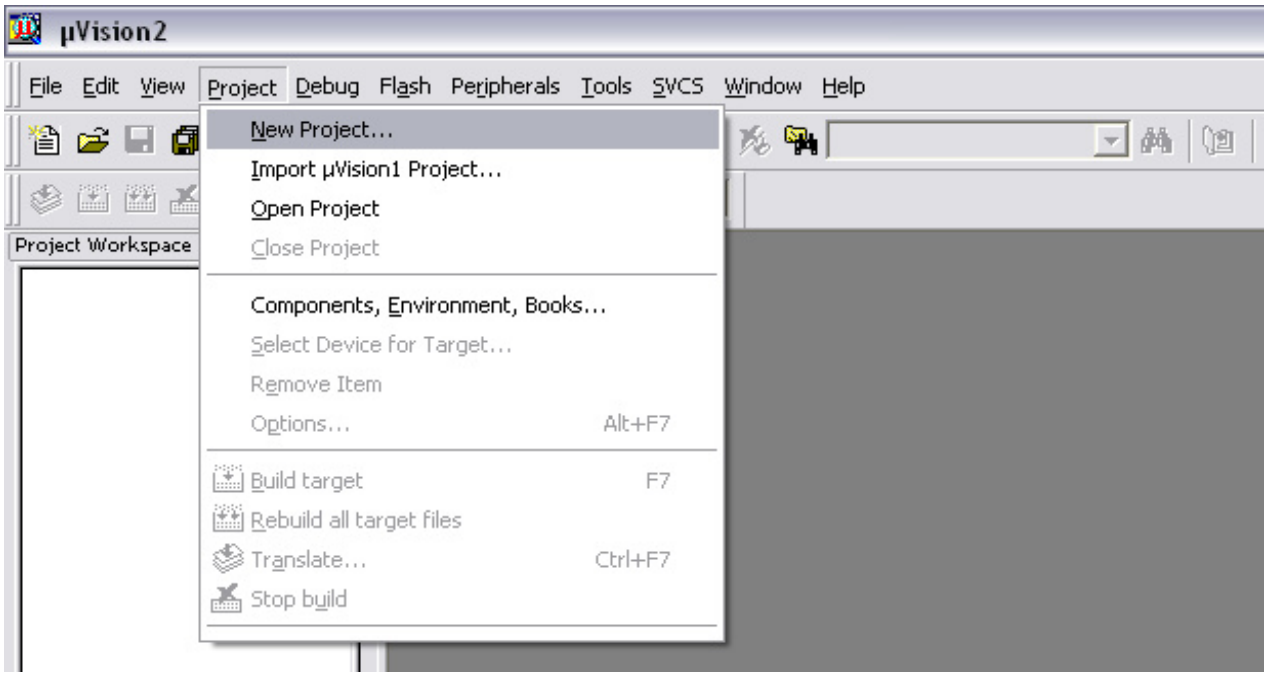


Ilustración 72 - Crear un nuevo proyecto

Después hay que asignarle un nombre al proyecto.

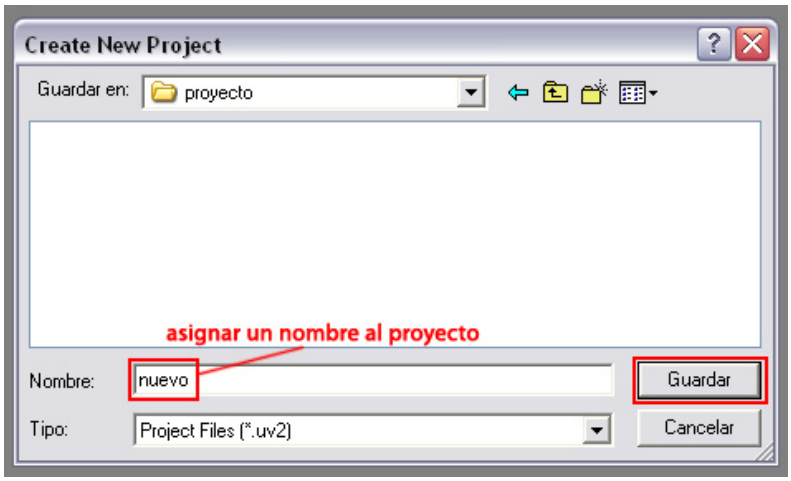


Ilustración 73 - Asignar nombre al proyecto creado

El siguiente paso es elegir el tipo de microcontrolador para el que se va a implementar el programa. En el presente proyecto se hace uso del microcontrolador de la marca ***Dallas Semiconductor***, modelo ***DS89C450*** [31].

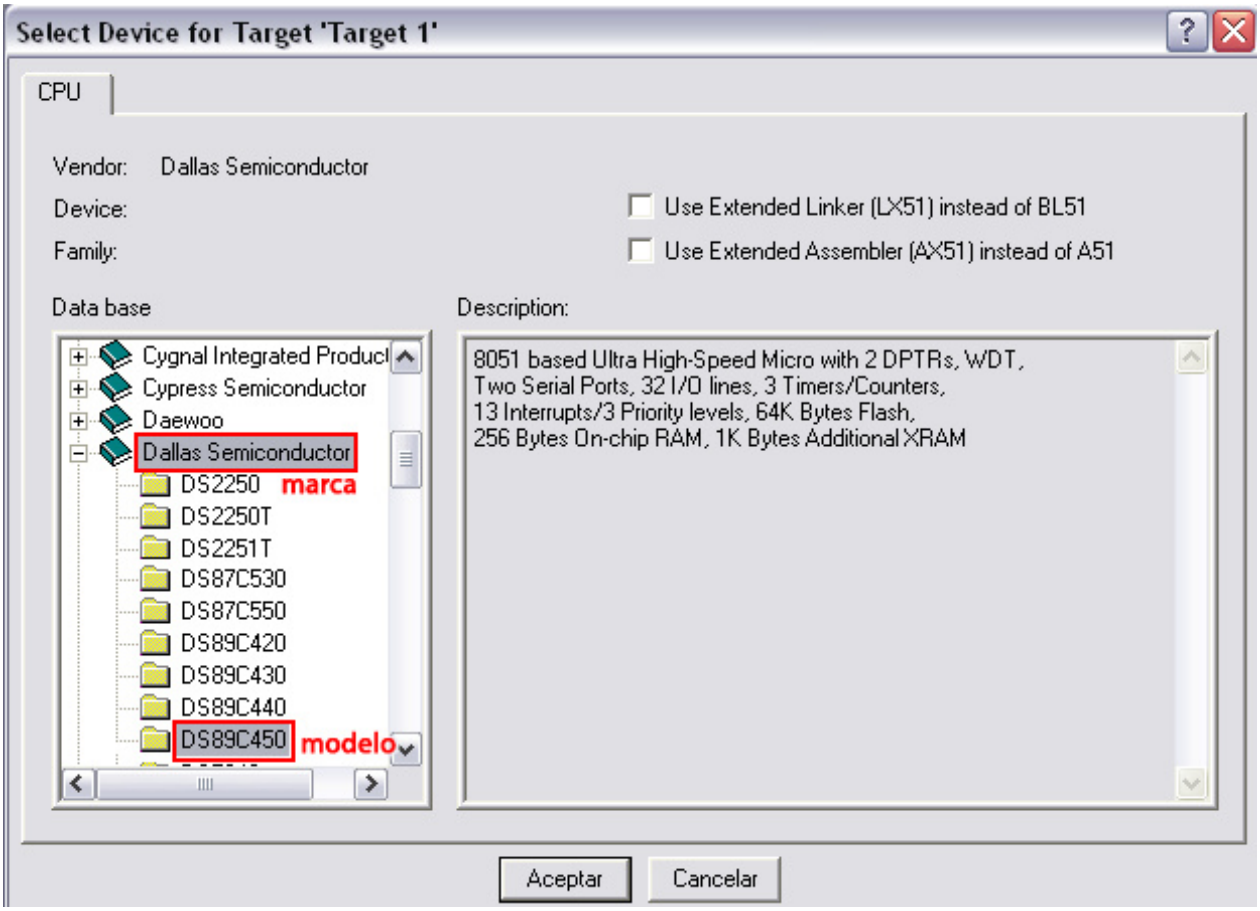


Ilustración 74 - Indicar marca y modelo del microcontrolador empleado

Llegados a este punto, el proyecto queda creado. En el entorno de programación será denominado como "***Target 1***"

Antes de comenzar con la implementación, será necesario configurar las opciones del mismo para que al compilar el programa, se obtenga la traducción a ensamblador en formato hexadecimal. Para ello, es necesario hacer click secundario sobre ***Target 1*** y hacer click sobre el apartado de opciones.

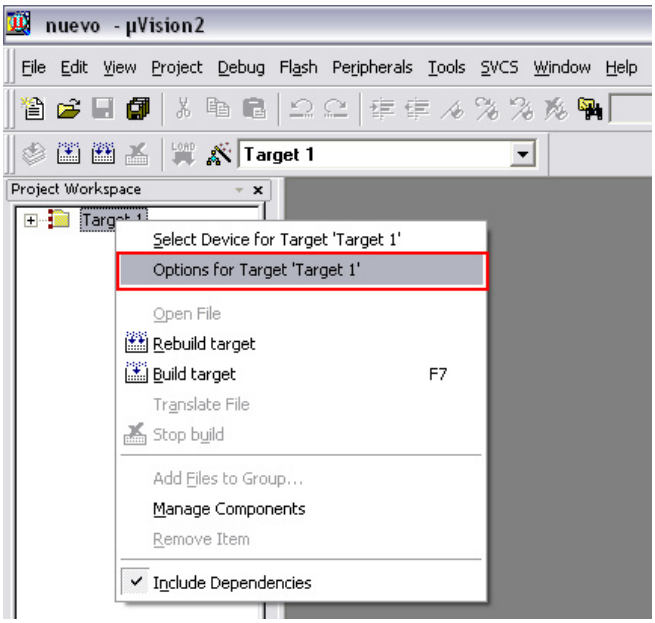


Ilustración 75 - Configurar opciones

En el apartado de opciones, dentro de la etiqueta "***Output***", marcar la opción "***Create HEX File***"

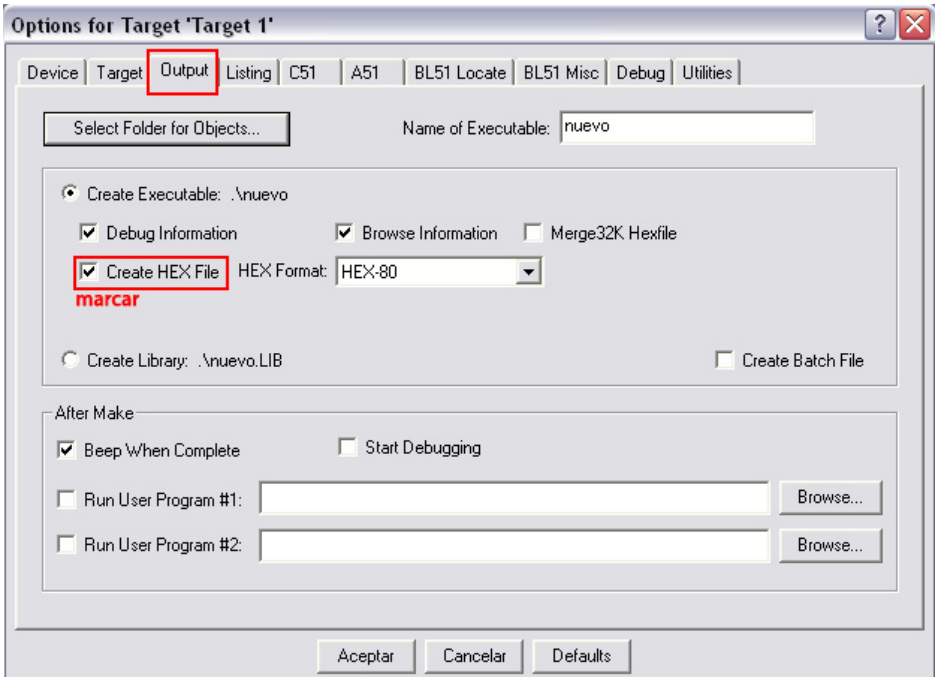


Ilustración 76 – Obtención del ejecutable en formato hexadecimal

Una vez configurado correctamente el proyecto, se procede a añadir el archivo donde se incluirá el código en lenguaje de programación C [23] que formará el programa a implementar.
Hacer click en el botón de ***crear nuevo fichero*** e incluir el código en el espacio correspondiente.

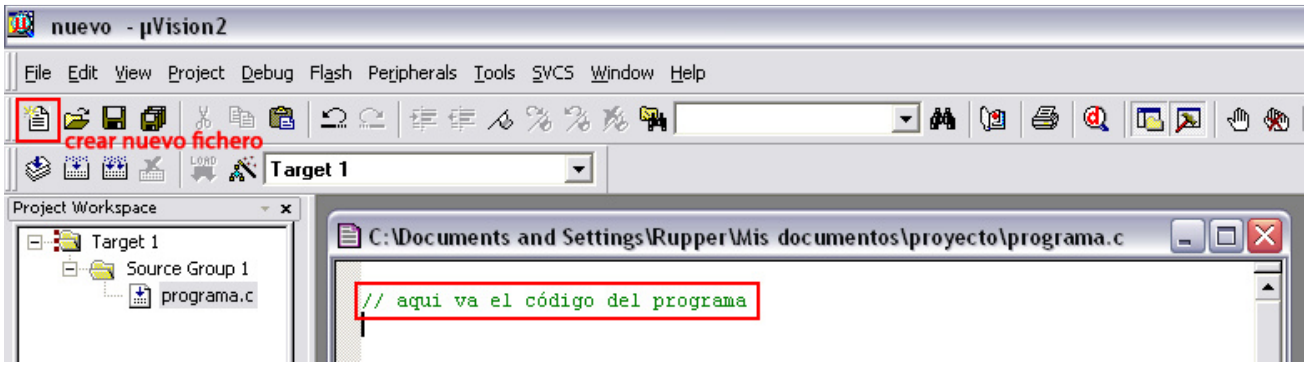


Ilustración 77 - Crear un nuevo fichero donde incluir código

Hacer click en el botón de ***guardar*** fichero, elegir un nombre para el fichero en ***lenguaje de programación C.***

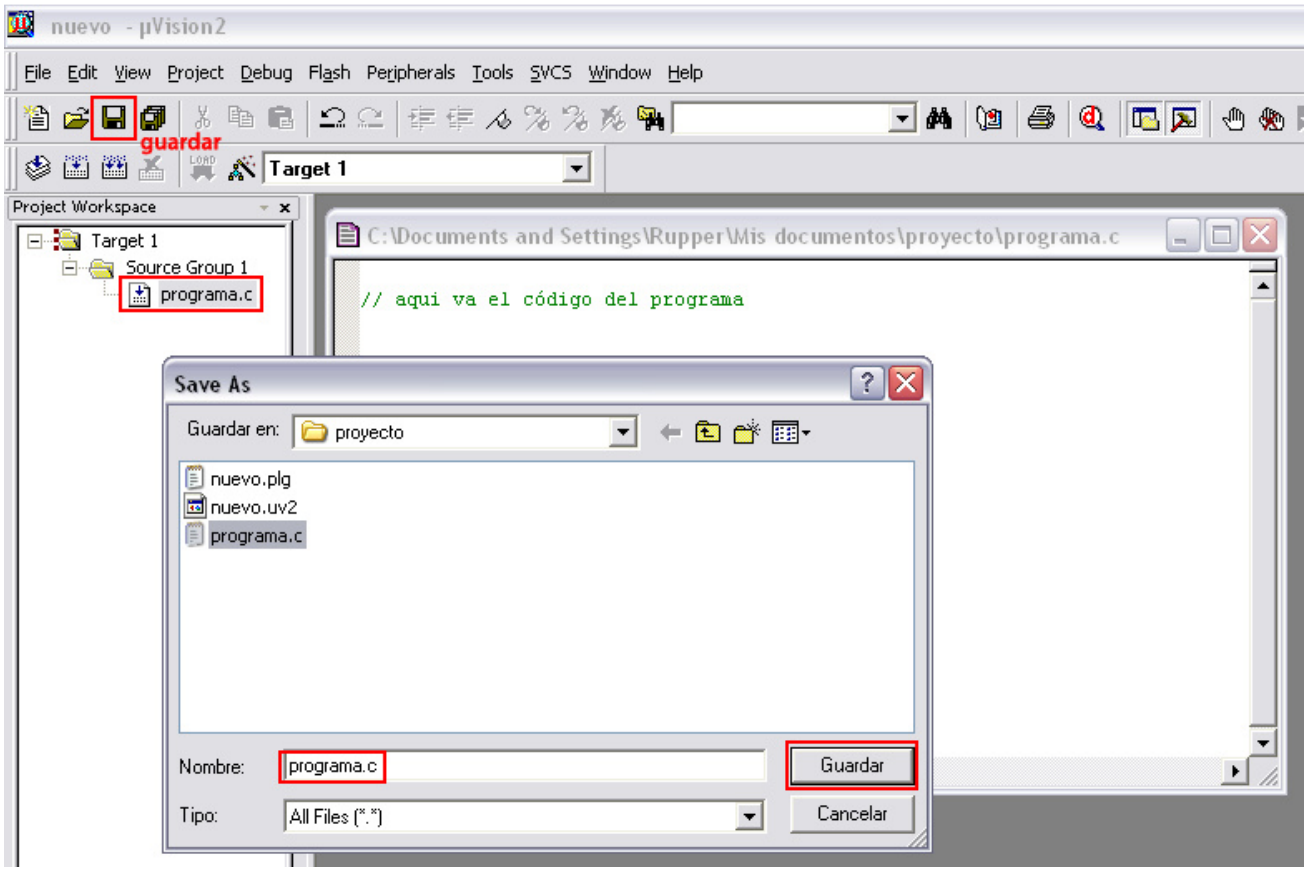


Ilustración 78 - Indicar nombre y tipo del fichero creado

En el fichero que ha sido denominado "***programa.c***" se incluirá el código del programa.

5.2.2 Cómo compilar un programa para la placa de servo-motores

Para obtener el ejecutable en formato hexadecimal (fichero .hex) de un proyecto creado previamente se hace uso del programa ***Keil µvision*** [29]. Para ello se utilizan los botones para compilación. Se usará uno u otro dependiendo de si se quiere compilar un fichero (terminación .c) de código concreto o el proyecto completo, es decir, todos los ficheros contenidos en el mismo.

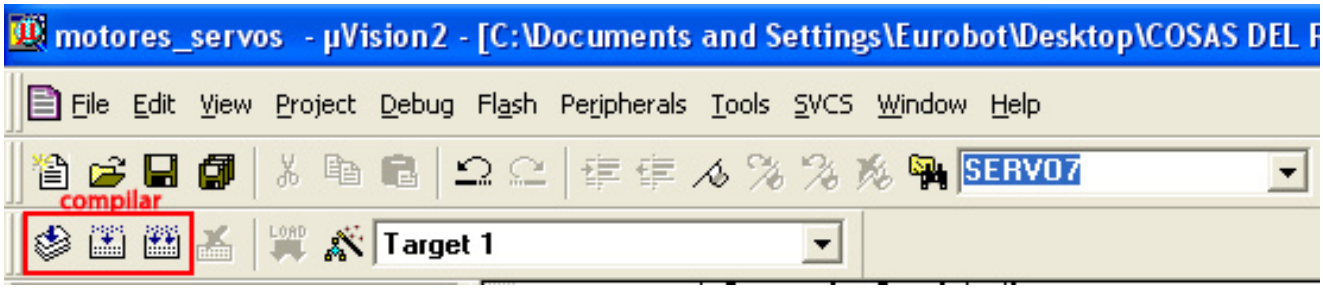


Ilustración 79 - Botones para compilar en el entorno de programación Keil µvision

Al pulsar cualquiera de estos botones, comenzará el proceso de compilación (ya sea de uno o varios ficheros de código). Pueden darse los siguientes casos:

- Exista algún error de programación en el código
- No existan errores de programación en el código, pero existan advertencias
- No existan ni advertencias ni errores de programación en el código

Un fichero de código contiene algún error de programación

Se avisará en la parte inferior de la pantalla de los tipos de fallos cometidos, así como la ubicación dentro del fichero (número de línea de código en la que se encuentra) de los mismos. El ejecutable **NO** será generado.

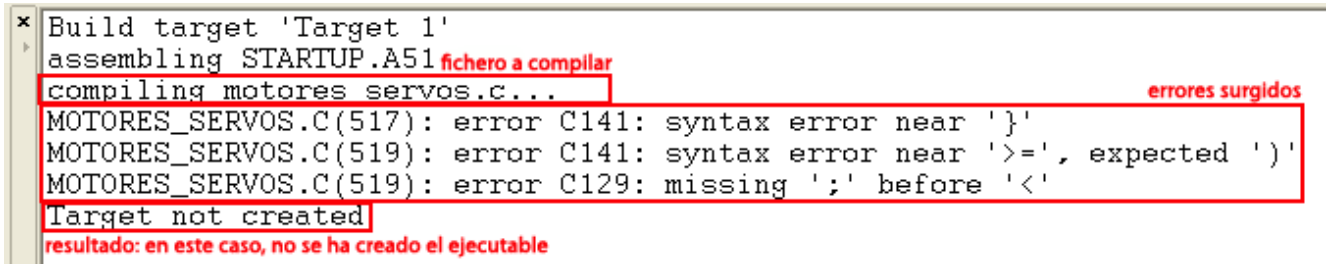


Ilustración 80 - Compilación fallida por errores de programación

Algún fichero de código contiene advertencias de programación

Se informará en la parte inferior de la pantalla de las advertencias surgidas. Las advertencias no implican que el código sea incorrecto, sino que tiene pequeños fallos leves. El ejecutable **SÍ** será generado.

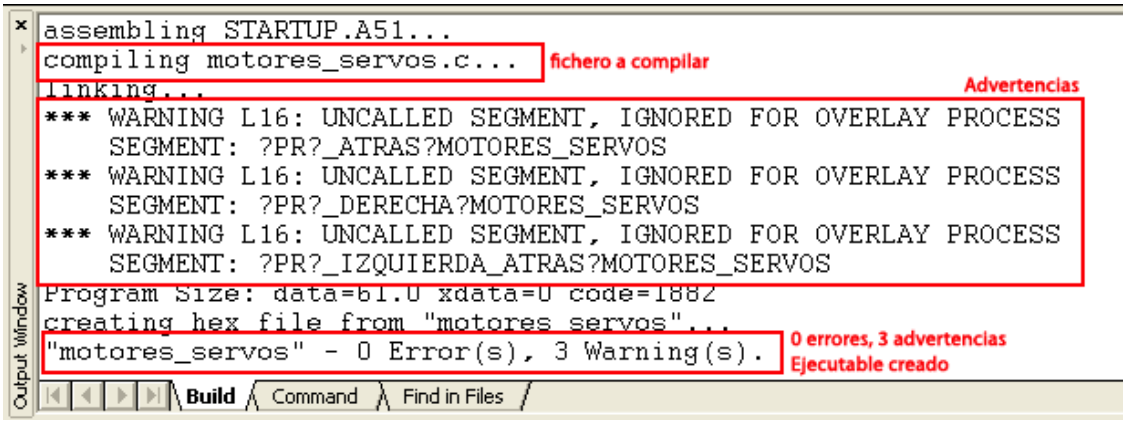


Ilustración 81 - Compilación completa, aunque con advertencias de programación

Para comprobar que el proceso de compilación ha concluido satisfactoriamente, puede accederse a la carpeta donde está ubicado el proyecto de programación creado y comprobar que se ha generado un fichero ejecutable en formato hexadecimal (.hex)

Ilustración 82 - Ejecutable generado en la carpeta correspondiente

Todo el código es correcto en cuanto a la gramática del lenguaje de programación C.
Se avisará en la parte inferior de la pantalla de la inexistencia de fallos. No existen errores ni advertencias. En este caso, también se genera el ejecutable.

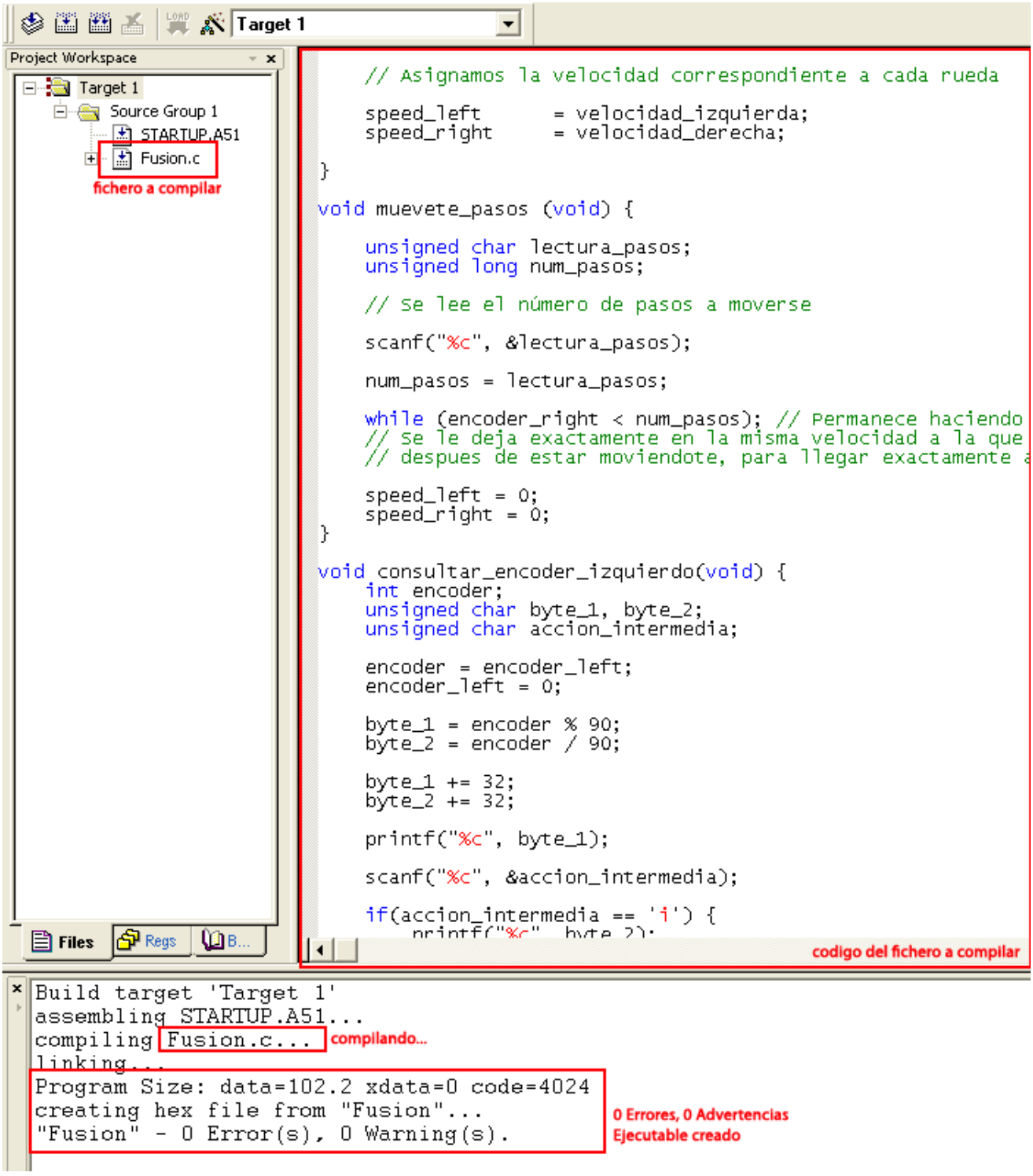


Ilustración 83 - Compilación completada, sin fallos ni advertencias

5.2.3 Cómo arrancar la placa de servo-motores

Esta placa se alimenta con un voltaje de ***5 voltios***, que se obtendrá con una ***fuente de alimentación,*** en la fase de programación.

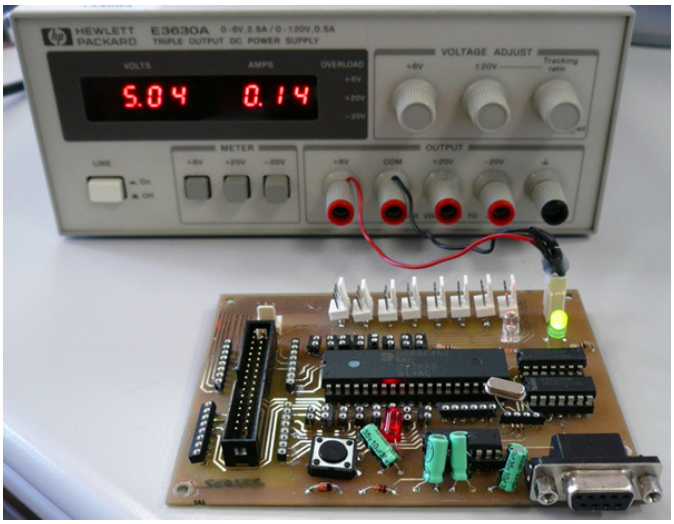


Ilustración 84 - Placa de servo-motores alimentada con una fuente de alimentación

En la fase de pruebas o de competición, cuando la placa de servo-motores ya se encuentra ubicada dentro del robot, se obtendrá el voltaje conectándola a una placa (ver apartado "***3.4 Placa intermedia"*** de este documento) que transforma los ***12 voltios*** de salida de la ***batería***, a los ***5 voltios*** requeridos por la placa de servo-motores.

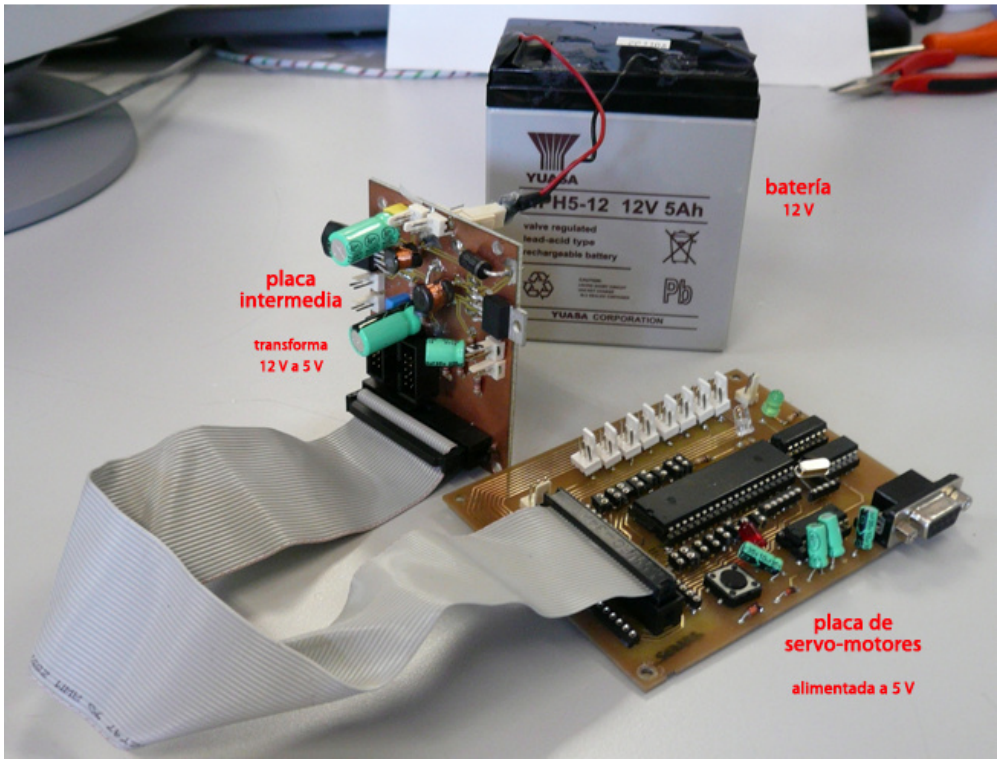


Ilustración 85 - Placa de servo-motores alimentada con batería de 12 V

5.2.4 Cómo cargar el ejecutable de un programa en la placa de servo-motores

Para cargar en la placa el ejecutable de un programa, se hace uso del programa ***Microcontroller Tool Kit*** (***MTK2)*** de ***Dallas Semiconductor*** [30].

El primer paso será conectar el ordenador a la placa de servo-motores mediante un ***cable RS232*** [9]. Arrancar la placa de servo-motores será el siguiente paso a llevar a cabo (ver apartado "***5.2.3. Cómo arrancar la placa de servo-motores***" de este documento).

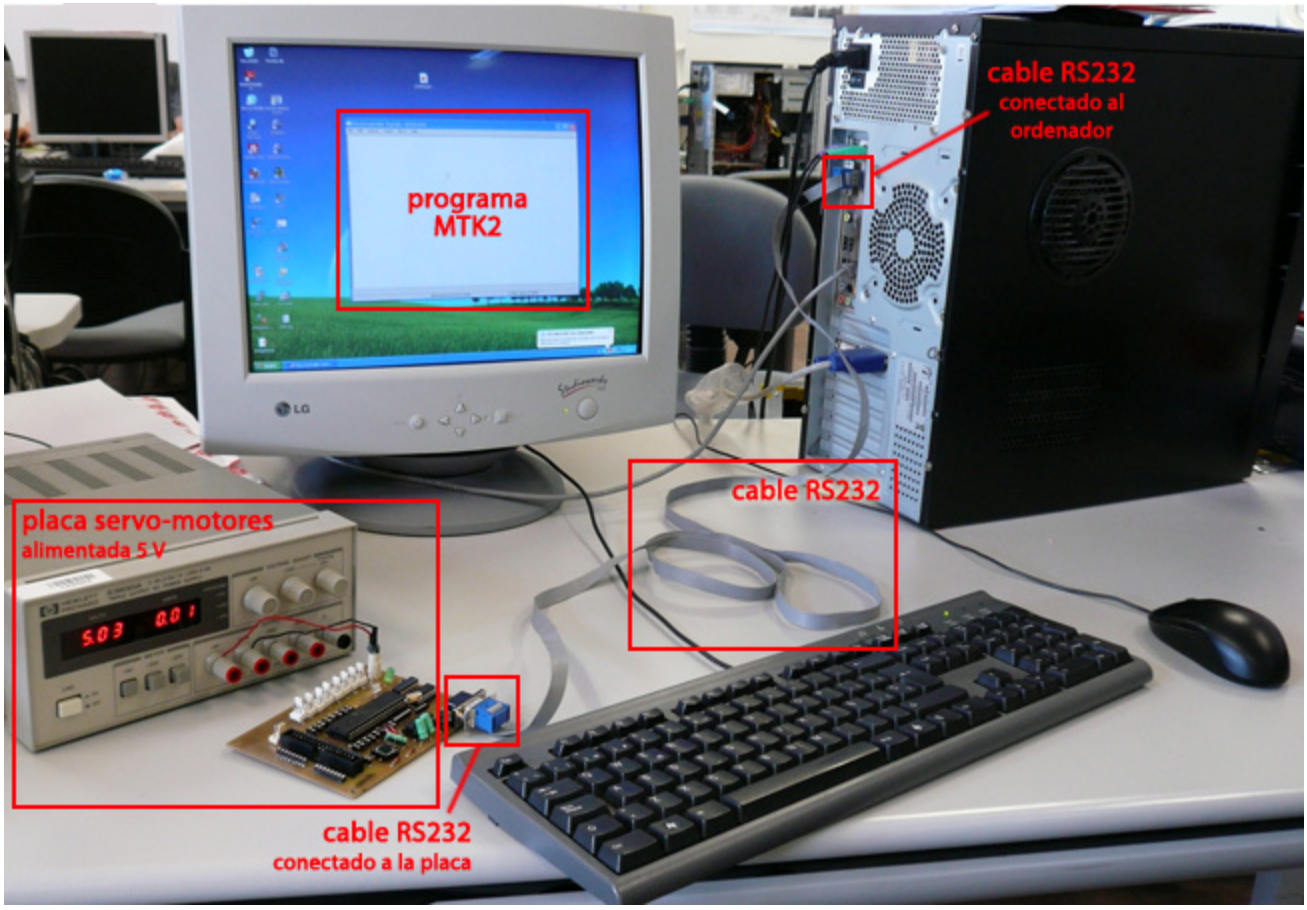


Ilustración 86 - Conexiones necesarias para cargar un programa en la placa de servo-motores

Al abrir el programa en el ordenador ***por primera vez***, será necesario elegir el modelo de microcontrolador empleado, que en este caso será el modelo ***DS89C450***

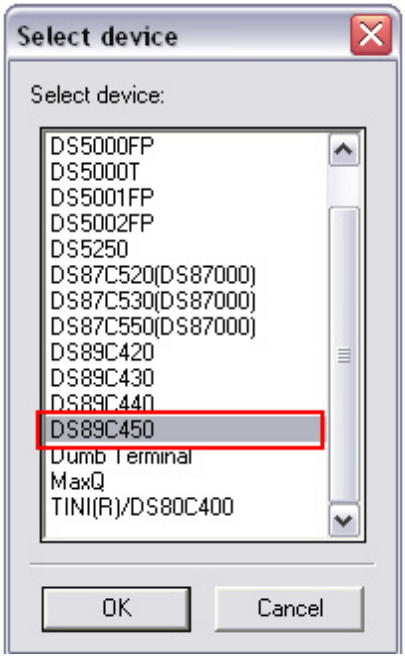


Ilustración 87 - Elección inicial de la marca y modelo del microcontrolador usado

Para ***cargar un programa*** en la placa de servo-motores, deberá abrirse el ***puerto serie (COM1)*** a ***9600 baudios por segundo***. Por lo tanto, el primer paso será configurar el puerto a 9600 baudios por segundo, y después abrirlo.

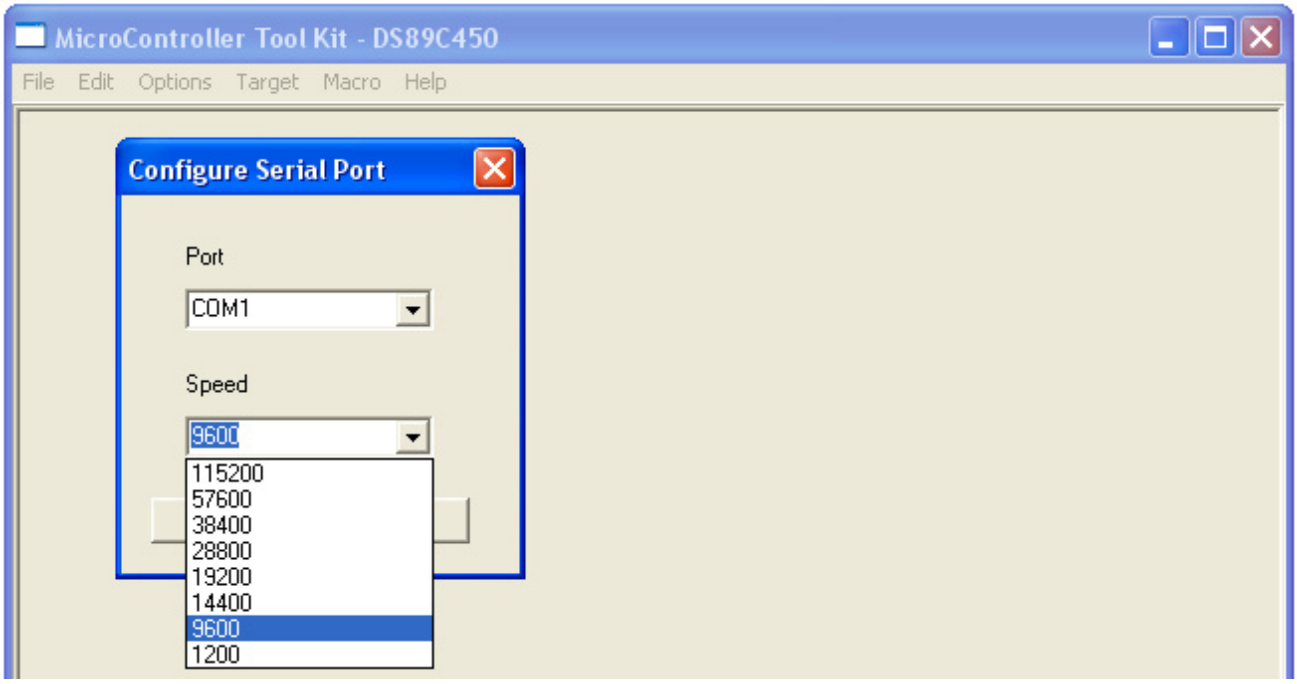


Ilustración 88 - Configurar puerto serie a 9600 baudios por segundo

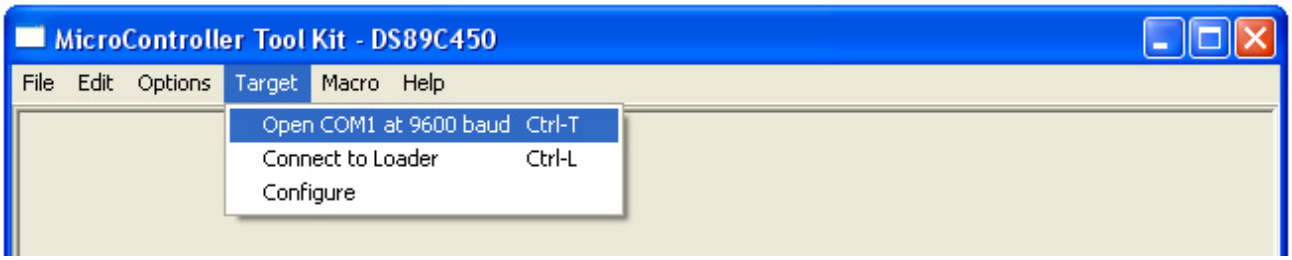


Ilustración 89 - Abrir puerto serie a 9600 baudios

Si el puerto se abre correctamente, el fondo del programa pasará del color gris claro anterior a color blanco. Llegados a esta situación, es el momento de conectarse al cargador del programa.

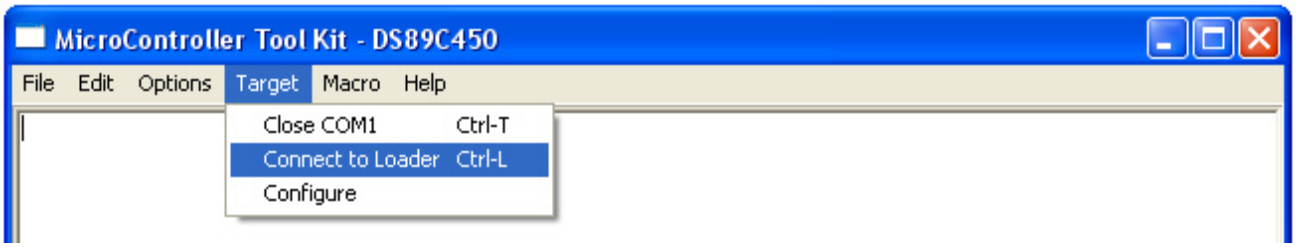


Ilustración 90 - Conectarse al cargador

Si la conexión al cargador se ha realizado correctamente, deberá mostrarse por pantalla el mensaje que aparece en la ***Ilustración 91***.

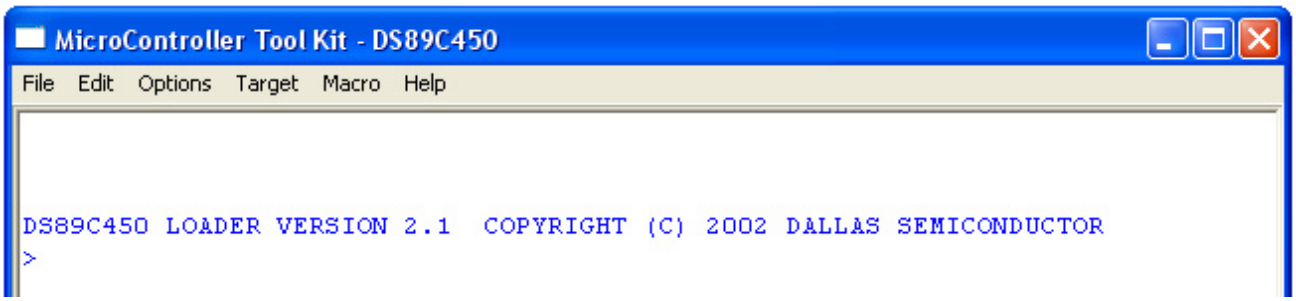


Ilustración 91 - Cargador conectado correctamente

En caso de que la conexión no se haya realizado satisfactoriamente, el mensaje mostrado se muestra en la ***Ilustración 92***.

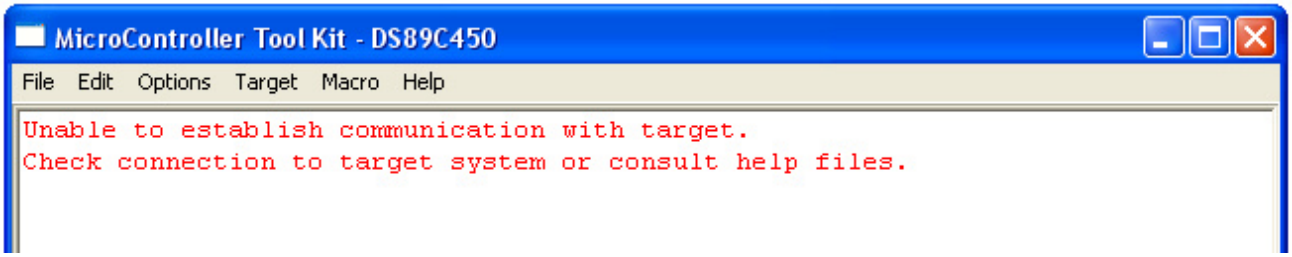


Ilustración 92 - Conexión con el cargador fallida

Una vez la conexión al cargador haya sido establecida correctamente, se procede a la carga del ejecutable (en formato hexadecimal) en la placa de servo-motores.

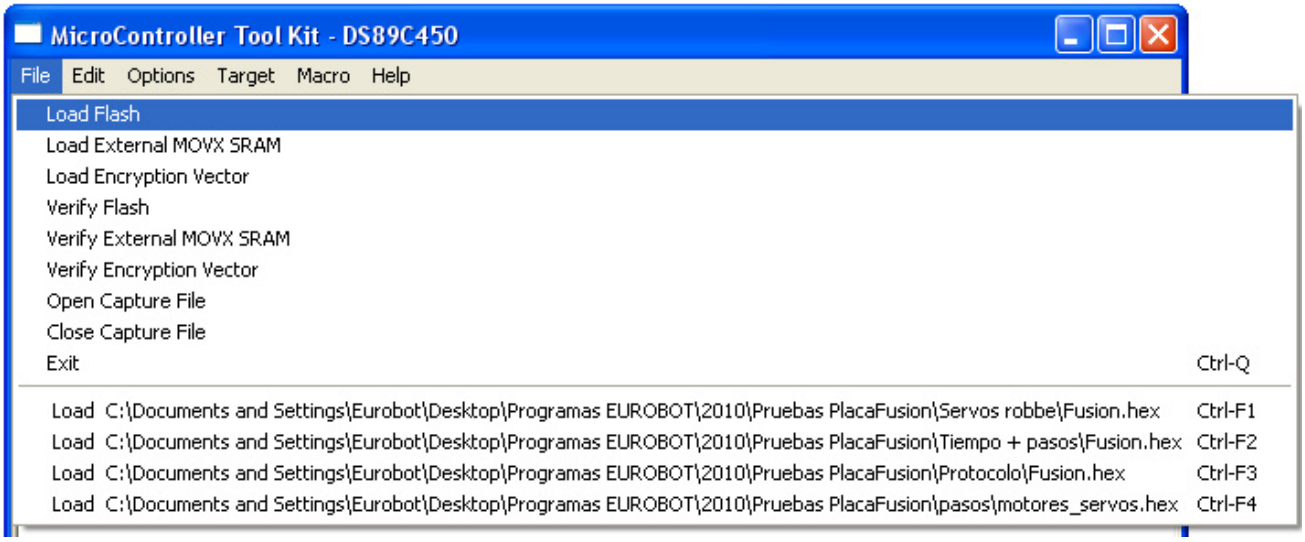


Ilustración 93 - Cargar ejecutable en la placa de servo-motores

Se accede a la carpeta que contiene el ejecutable (***fichero .hex***) generado en la fase de compilación del programa (ver apartado "***5.2.2. Cómo compilar un programa para la placa de servo-motores***" de este documento).

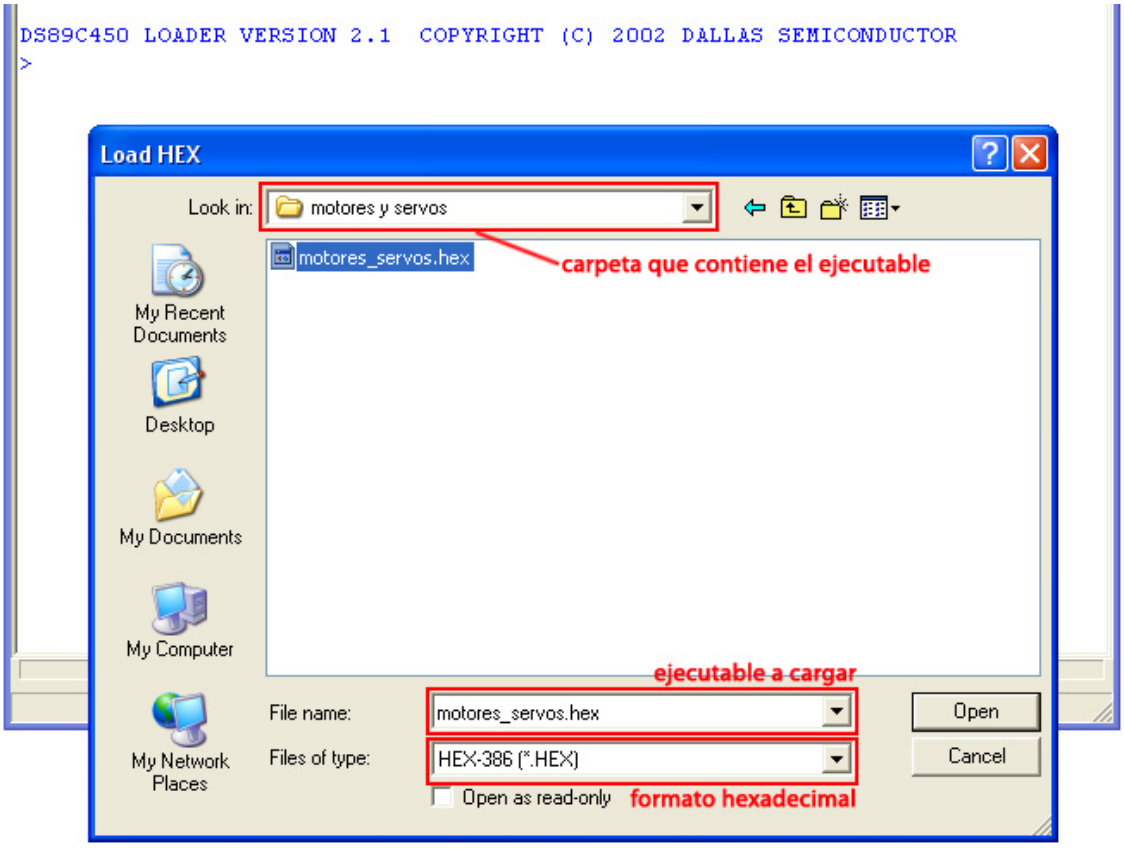


Ilustración 94 - Seleccionar el ejecutable a cargar en la placa

Al iniciar el proceso de carga del ejecutable en la placa, si existe algún fallo, por ejemplo en la conexión, no se procederá a realizar la operación, mostrándose por pantalla el mensaje de la ***Ilustración 95***.

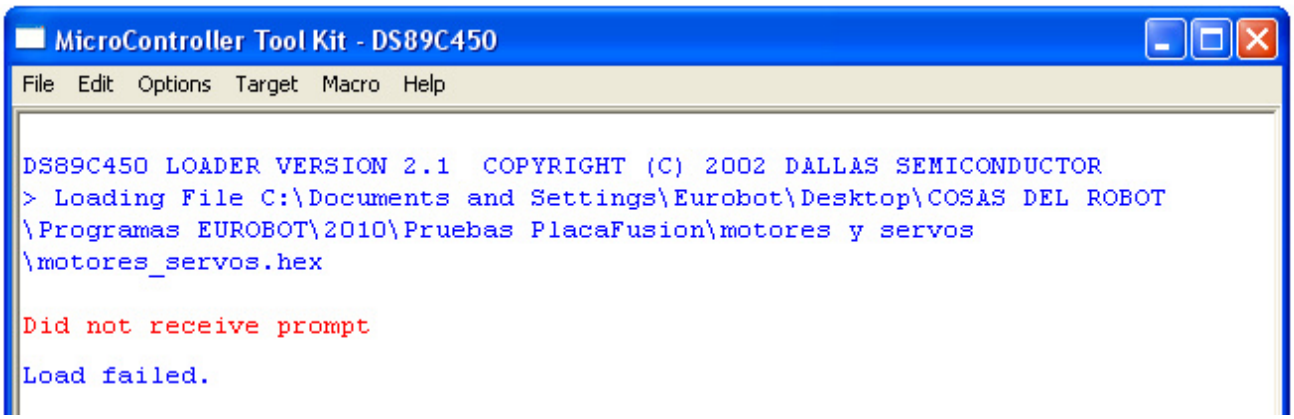


Ilustración 95 - Fallo en el proceso de carga del ejecutable en la placa

Si no existe fallo alguno que impida el proceso de carga, la primera acción que realizará el cargador será eliminar el ejecutable existente en la placa de servo-motores. Una vez finalizada esta tarea, comenzará la operación de carga del nuevo ejecutable, mostrándose por pantalla el mensaje que aparece en la ***Ilustración 96***.

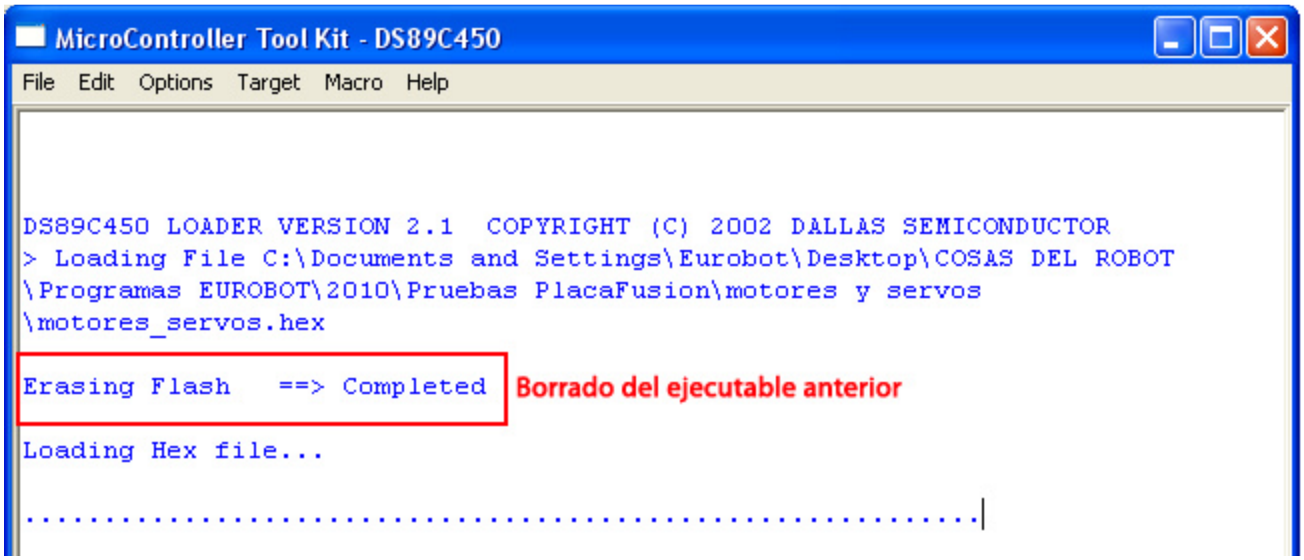


Ilustración 96 - Comienzo del proceso de carga del ejecutable en la placa

Si durante el proceso de carga sucede alguna situación de fallo (por ejemplo, la desconexión del cable RS232 o el apagado de la placa de servo-motores), la acción de carga se verá interrumpida, mostrándose por pantalla el mensaje de la ***Ilustración 97***.

```
DS89C450 LOADER VERSION 2.1  COPYRIGHT (C) 2002 DALLAS SEMICONDUCTOR
> Loading File C:\Documents and Settings\Eurobot\Desktop\COSAS DEL ROBOT
\Programas EUROBOT\2010\Pruebas PlacaFusion\motores y servos
\motores_servos.hex

Erasing Flash   ==> Completed

Loading Hex file...

................................................
Error on writing the line :

:10006E00120690E4FBFAF9F8C312066740EE120688


Error writing file: expected G and received

Error on line number : 51


Load failed.
```

Ilustración 97 - Fallo en mitad del proceso de carga del ejecutable en la placa

Si no ocurriera ningún problema, el proceso de carga terminaría satisfactoriamente, mostrándose por pantalla el mensaje de la ***Ilustración 98***.

```
DS89C450 LOADER VERSION 2.1  COPYRIGHT (C) 2002 DALLAS SEMICONDUCTOR
> Loading File C:\Documents and Settings\Eurobot\Desktop\COSAS DEL ROBOT
\Programas EUROBOT\2010\Pruebas PlacaFusion\motores y servos
\motores_servos.hex

Erasing Flash   ==> Completed

Loading Hex file...

................................................................................
..................................................

Load complete.

>
```

Ilustración 98 - Carga del ejecutable en la placa terminada correctamente

Llegados a este punto, el programa queda cargado en la placa de servo-motores.

5.2.5 Cómo comunicar un ordenador con la placa de servo-motores

Una vez cargado correctamente el ejecutable del programa implementado en la placa de servo-motores, se puede llevar a cabo una comunicación entre un ordenador y la placa de servo-motores que permita probar y depurar el protocolo de comunicación diseñado para comunicar la placa de servo-motores con la placa Linux (ver apartado "***4.4 Protocolo de comunicaciones***" de este documento). Esta comunicación puede realizarse de distintas maneras:

Consola del programa MTK2 en ordenador con Sistema Operativo Windows

El programa ***Microcontroller Tool Kit*** [30] ofrece la posibilidad de conectar el ordenador a la placa de servo-motores y enviar/recibir bytes, de manera que pueda simularse el comportamiento de una comunicación real entre la placa de servo-motores y otro dispositivo. Esta conexión debe establecerse a ***57600 baudios por segundo***, siguiendo los siguientes pasos:

1º Cerrar el puerto serie (en caso de que esté abierto)

Ilustración 99 - Cerrar el puerto serie en programa MTK2

2º Configurar el puerto serie a 57600 baudios por segundo.

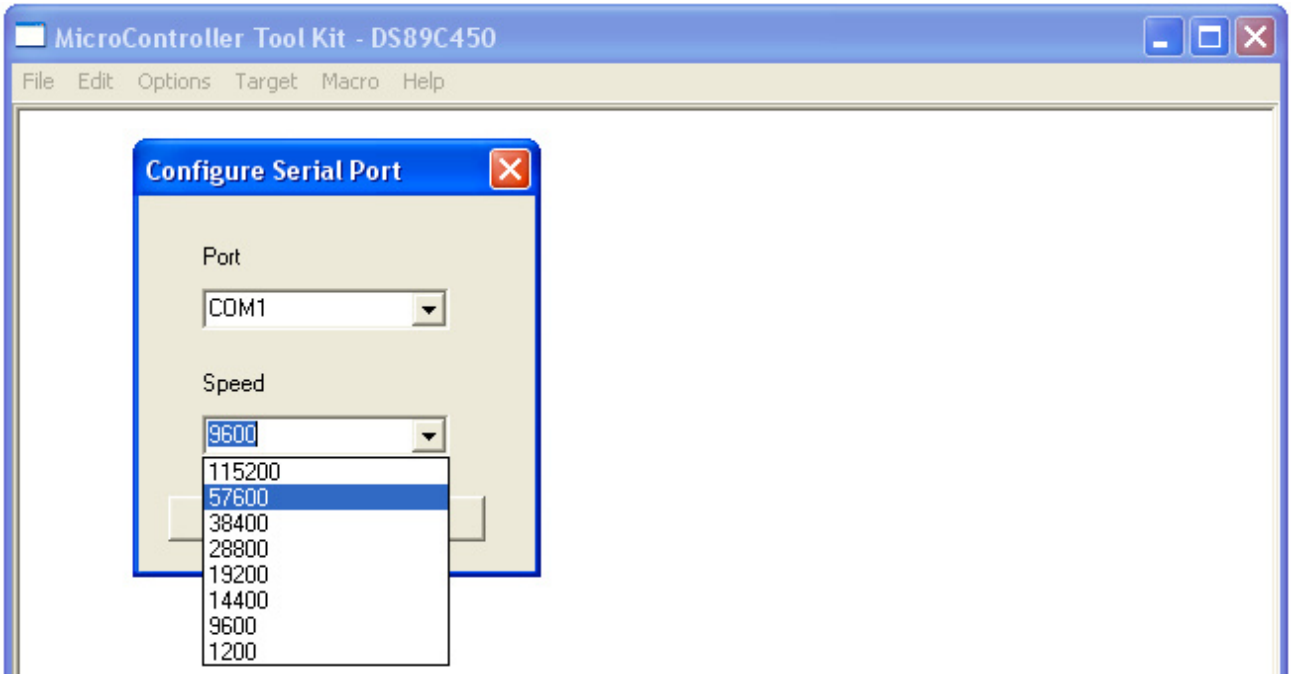


Ilustración 100 - Configurar el puerto serie a 57600 baudios por segundo

3º Abrir el puerto serie a 57600 baudios por segundo

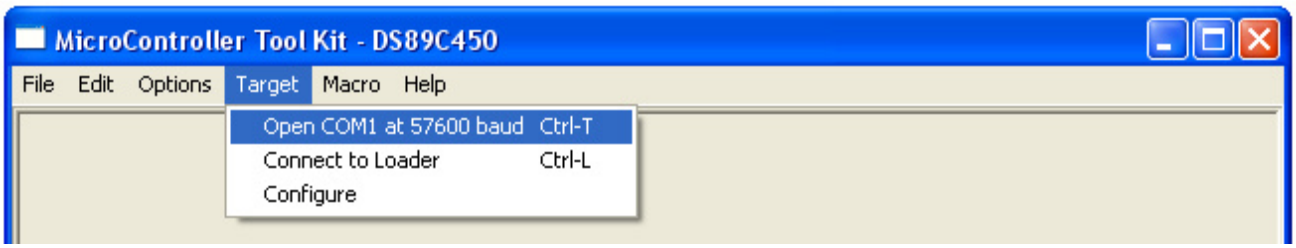


Ilustración 101 - Abrir puerto serie a 57600 baudios por segundo

Si el puerto queda abierto correctamente, el fondo de color gris claro del programa cambiará a color blanco. En este momento, toda tecla pulsada en el teclado (el byte de código ASCII equivalente [26]) será enviada por el puerto serie [9] desde el ordenador hasta la placa de servo-motores. Así mismo, se mostrarán por pantalla todos los bytes recibidos por el puerto serie en el ordenador, ya sean bytes recibidos por el eco de los datos que ha enviado, o datos recibidos que hayan sido enviados por la placa de servo-motores.

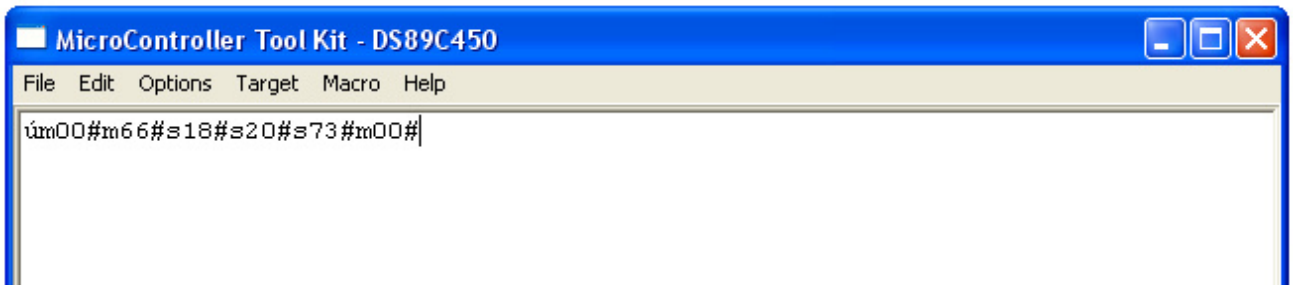


Ilustración 102 - Bytes de comunicación entre el ordenador y la placa de servo-motores

<u>Programa ucom implementado para ordenador con Sistema Operativo Linux</u>

Se ha desarrollado la aplicación "***ucom***" en lenguaje de programación C [23], que permite conectar la placa de servo-motores con ordenadores con Sistema Operativo ***Linux*** [22]. Entre otros, la placa Linux es un dispositivo que usa una versión adaptada del Sistema Operativo Linux, por lo tanto, el programa "***ucom***" también será válido para probar manualmente la comunicación entre la placa Linux y la placa de servo-motores.

El programa solamente requiere que se le indique como parámetro en qué ***puerto serie*** de los que disponga el ordenador ha sido conectado el ***cable RS232*** (si no se le indica, cogerá por defecto el primero de ellos). Después, se configura automáticamente para establecer una comunicación a ***57600 baudios***. A partir de ese momento, toda tecla pulsada será enviada por el puerto serie hacia la placa de servo-motores. Todos los bytes recibidos (de eco o enviados por la placa de servo-motores) serán mostrados por pantalla.

Si la comunicación se establece entre un ordenador (ya sea portátil o de sobremesa) con Sistema Operativo Linux y la placa de servo-motores, es suficiente con abrir una terminal y ejecutar el programa "***ucom***".

En cambio, si la comunicación se establece entre la placa Linux y la placa de servo-motores, será necesario utilizar un ordenador de sobremesa que se conecte (utilizando el protocolo de comunicación Telnet [11]) con la placa Linux para poder ver en la pantalla del ordenador las acciones que se están llevando a cabo (no se dispone de pantalla conectada a la placa Linux directamente, es por este motivo que se hace imprescindible el uso de un ordenador). Para conectar la placa Linux y el ordenador se empleará un cable de ***Ethernet*** [8].

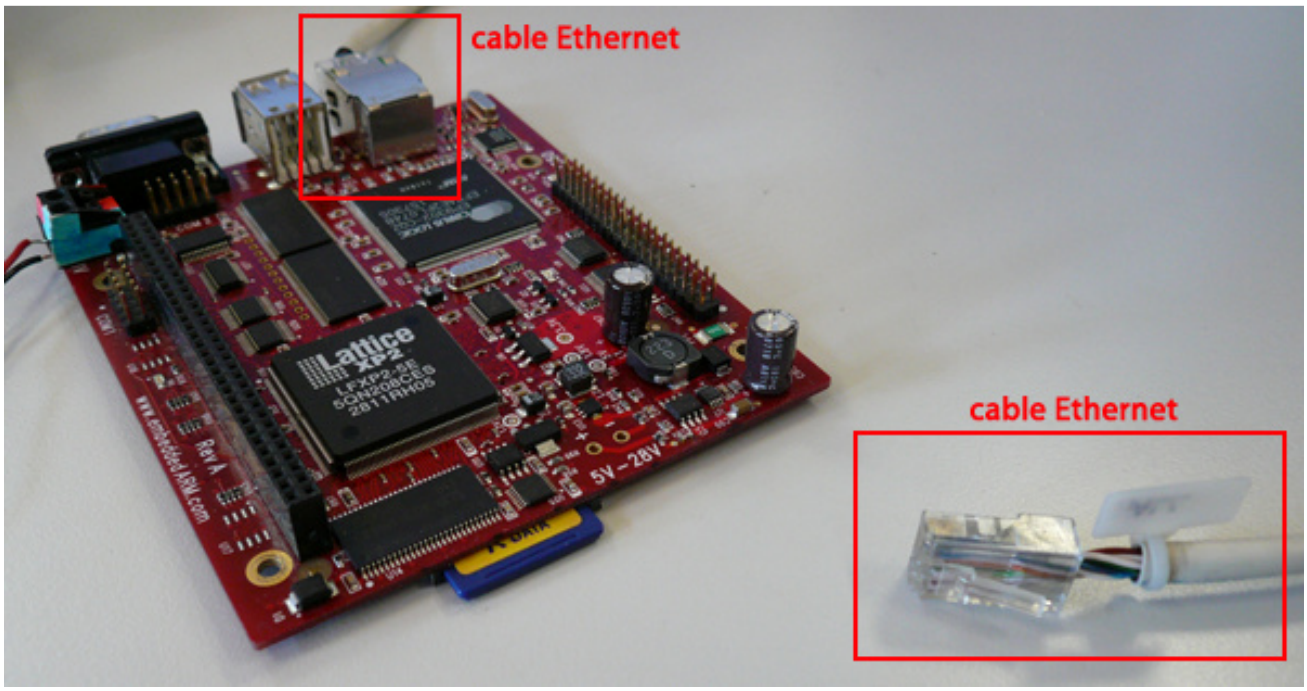


Ilustración 103 - Conexión entre un ordenador y la placa usando un cable Ethernet

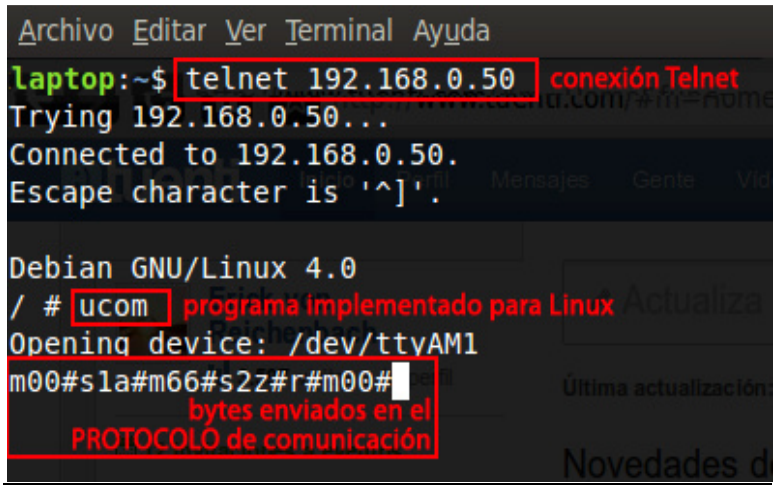


Ilustración 104 - Programa ucom

5.2.6 Cómo resolver situaciones de fallo comunes de la placa de servo-motores

La placa de servo-motores está provista de indicadores leds que permiten informar del estado de la misma. Si dos de estos leds (situados próximos en la placa) se encuentran de color:
Uno de color blanco rojizo y el otro de color amarillo verdoso. La placa no está conectada al cable RS232, o está conectada a 57600 baudios por segundo (comunicación de envío/recepción de bytes entre la placa de servo-motores y otro dispositivo).
Ambos leds de color amarillo verdoso. La placa está conectada al cable RS232 para proceder a cargar el ejecutable de un programa en la placa (placa en modo programación)

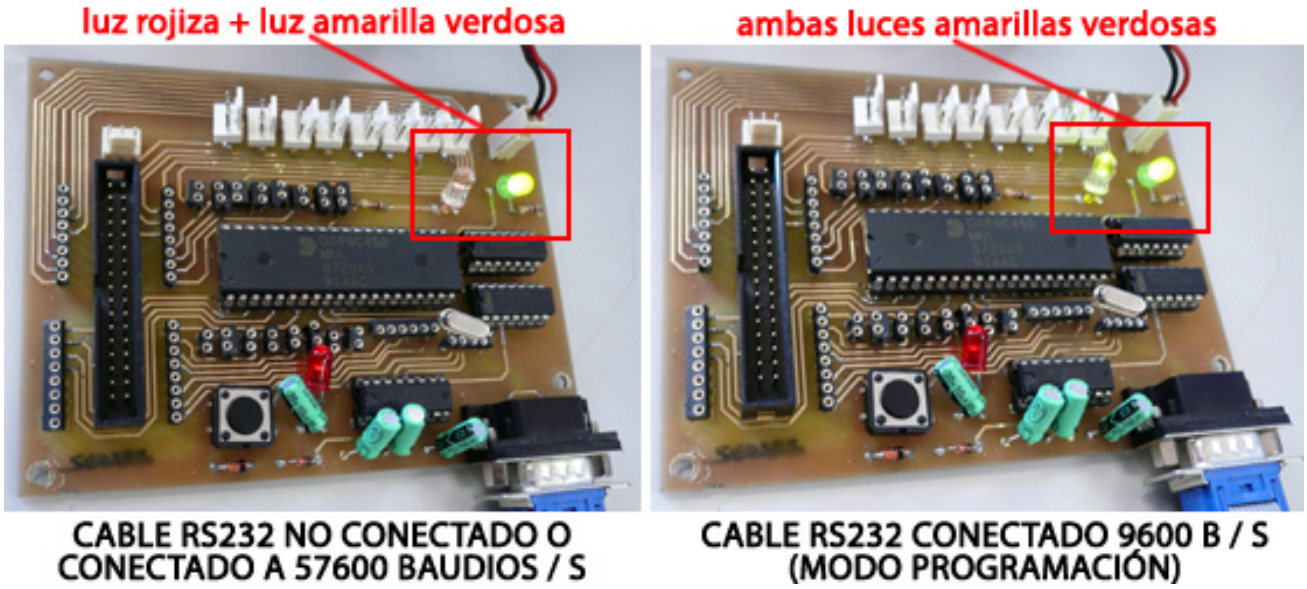


Ilustración 105 - Indicadores leds de la placa de servo-motores

Si las luces de la placa no se corresponden con el estado en el que ésta se encuentra, el procedimiento típico para intentar solventar esta situación errónea es el siguiente:

1º Reiniciar manualmente la placa de servo-motores

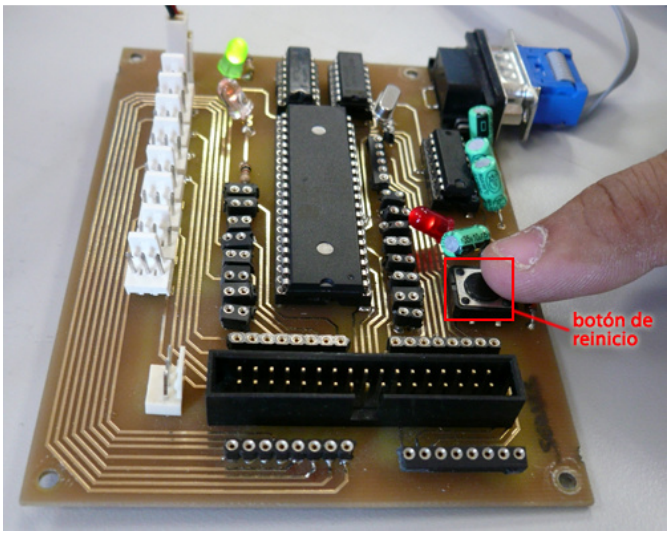


Ilustración 106 - Reinicio manual de la placa de servo-motores

2º Apagar completamente la placa de servo-motores. Para ello, desconectar de la fuente de alimentación que le esté proporcionando los 5 voltios de voltaje que la placa requiere para su correcto funcionamiento.

3º Cuando ninguno de los pasos anteriores fuera efectivo para solventar el problema. Tras cerciorarse minuciosamente de que el error proviene de la placa (y no de ninguno elemento externo). Cambiar el microcontrolador ***DS89C450-MNG*** por otro nuevo.

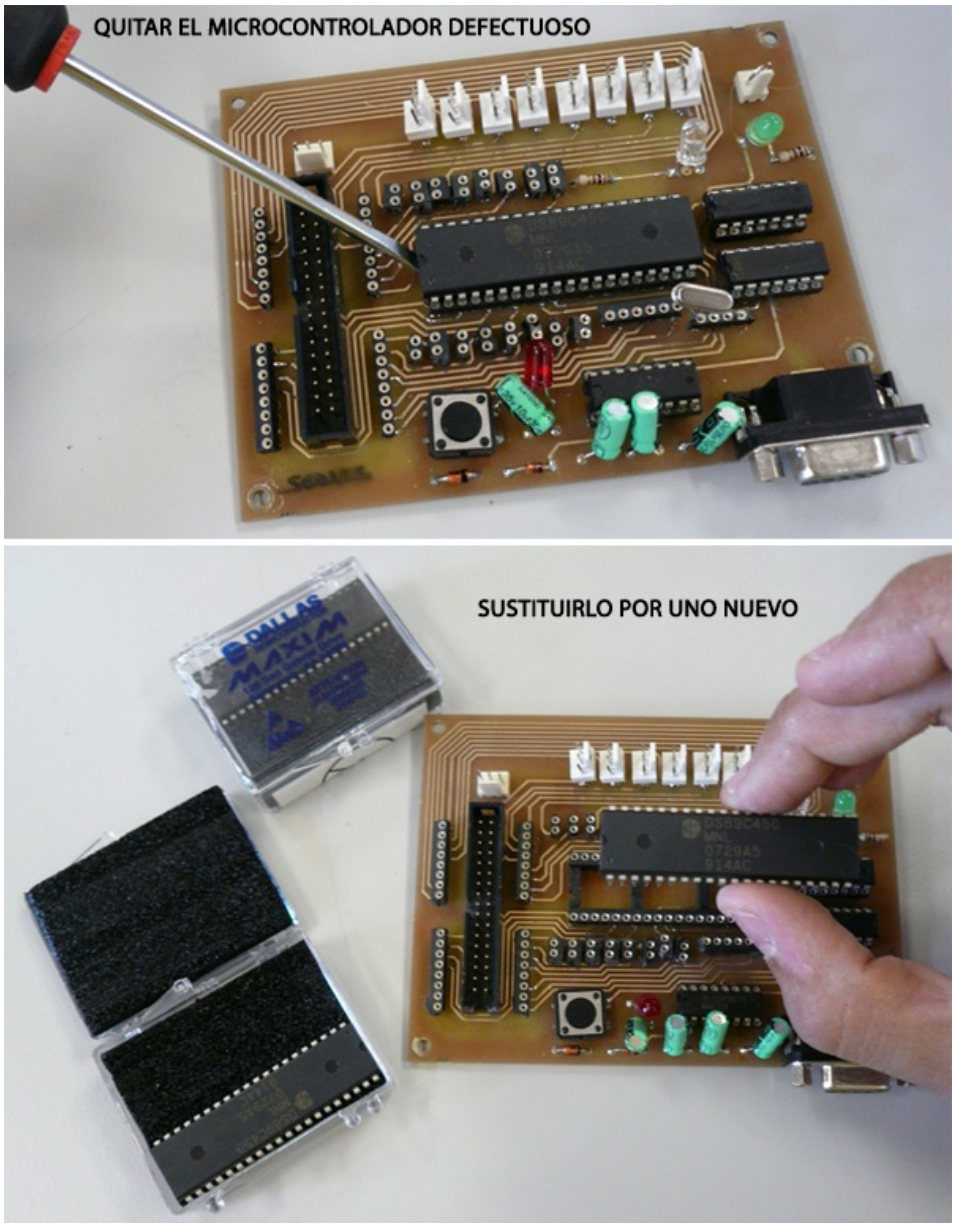


Ilustración 107 - Sustituir microcontrolador defectuoso

4º Si este último paso tampoco tuviera éxito, es probable que la placa se haya estropeado y sea necesario utilizar otra copia (se han construido múltiples copias, que pueden encontrarse en el laboratorio). Ponerse en contacto con el Técnico del Laboratorio.

6 Presupuesto (Eurobot 2010)

En este apartado de detallarán los costes derivados del diseño, construcción y programación del microrobot "**Flux Capacitor**".

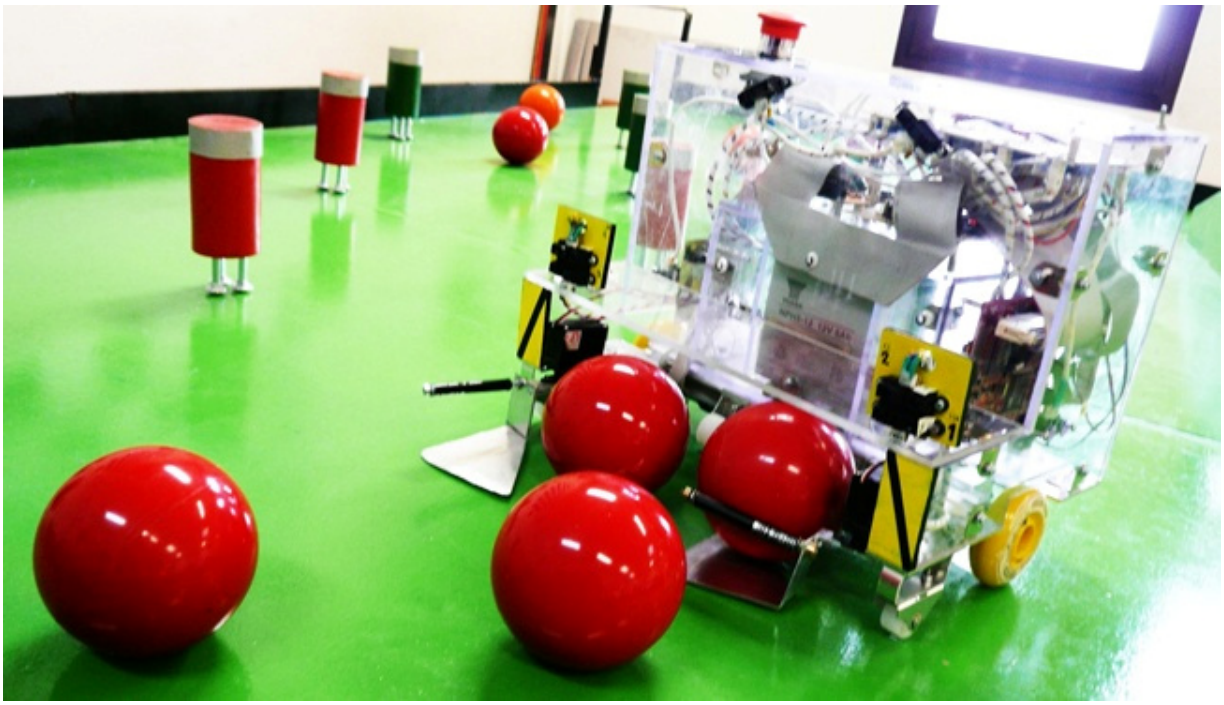


Ilustración 108 - Flux capacitor

6.1 Costes de material

6.1.1 Estructura

Descripción	Coste unitario	Medición	Total
Plancha de aluminio de 1.5 mm	75 €	1m^2	75 €
Lámina de policarbonato	80 €	2m^2	160 €
Escuadras de aluminio de 1.5 mm	0,20 €	30	6 €
Otros materiales de ferretería	-	-	85 €
		Subtotal	**326 €**

6.1.2 Sistema locomotor

Descripción	Coste unitario	Medición	Total
Motor, encoder y reductora	160 €	2	320 €
Ruedas	4,50 €	2	9 €
Ruedas locas	4 €	2	9 €
Soportes y casquillos	35 €	2	70 €
		Subtotal	**408 €**

6.1.3 Actuadores

Descripción	Coste unitario	Medición	Total
Servomotor Futaba s3003	12 €	8	96 €
		Subtotal	**96 €**

6.1.4 Sensores

Descripción	Coste unitario	Medición	Total
Sensor infrarrojo Sharp GP2D12	15 €	10	150 €
Sensor de final de carrera (bumper)	1,20 €	4	4,80 €
		Subtotal	**154,80 €**

6.1.5 Electrónica y alimentación

Descripción	Coste unitario	Medición	Total
Placa de circuito impreso	100 €	4	400 €
Single Board Computer (Placa Linux)	300 €	1	300 €
Componentes electrónicos	-	-	45 €
Batería	31 €	3	93 €
Bobina cable conexionado (0.5 mm^2)	45 €	1	45 €
		Sutbtotal	**883 €**

6.1.6 Campo y elementos del juego

Descripción	Coste unitario	Medición	Total
Tablero de DM	510 €	1	510 €
Listones de madera	2 €	10	20 €
Material de ferretería	-	-	30 €
Pintura	11 €	3	33 €
Pelota de malabares roja (150g)	6 €	6	36 €
Pelota de malabares naranja (300g)	9 €	2	18 €
		Subtotal	**647 €**

Total del Coste de material	**2514,8 €**

6.2 Costes de personal

Descripción	Sueldo mensual	Meses	Total
Ingeniero Técnico Industrial: electrónica	1800 €	9	16200 €
Ingeniero Técnico Industrial: mecánica	1600 €	9	14400 €
Ingeniero Informático	1700 €	9	15300 €
Total del coste de personal			**45900 €**

6.3 Coste global

Subtotal	Cuantía
Coste de material	2514,8 €
Coste de personal	45900,0 €
Total	**48414,8 €**

El presupuesto total es de **Cuarenta y ocho mil cuatrocientos catorce euros con ochenta céntimos de euro.**

7 Conclusiones

En las 3 ediciones en las que se ha concursado, se ha conseguido construir un microrobot competitivo para la Copa de España, logrando en todas ellas puestos clasificatorios para la competición internacional Eurobot. Sin embargo, en la competición europea, la diferencia respecto a los microrobots de equipos de otros países como Francia o Alemania era considerable.

La experiencia dicta que la mejor estrategia a llevar a cabo es la construcción de un microrobot ***robusto*** y ***fiable***, con una tasa de errores muy baja. Cuánta razón tenía ***José María Armingol***, tutor de este proyecto, cuando insistía en "***es mejor hacer pocos puntos, pero hacerlos siempre***".

En la edición de ***2008***, el microrobot "***Sbirro***", representante del LSI-UC3M (Laboratorio de Sistemas Inteligentes de la Universidad Carlos III de Madrid), consiguió el ***primer puesto*** en la Copa de España clasificatoria para representar a España en la competición internacional Eurobot. El concurso tuvo lugar en Mayo de 2008, en ***Heidelberg, Alemania***.

Ilustración 109 - Eurobot 2008. Heidelberg, Alemania

En la edición de ***2009***, el microrobot "***Iron-paco***", representando al equipo LSI-UC3M, consiguió clasificarse en ***segundo puesto*** en la Copa de España clasificatoria para representar a España en Eurobot, que tuvo lugar en Mayo de 2009, en ***La Ferté-Bernard, Francia***.

Ilustración 110 - Eurobot 2009. La Ferté-Bernard, Francia.

En la edición de ***2010***, el microrobot "***Flux Capacitor***", representante del LSI-UC3M, consiguió el ***tercer puesto*** en la prueba clasificatoria en España de Eurobot. El jurado del concurso propuso la formación de una candidatura conjunta compuesta por alumnos de la Universidad Carlos III y de la Universidad de Alcalá. El concurso tuvo lugar en Mayo de 2010, en ***Rapperswil-jona, Suiza***.

Ilustración 111 - Eurobot 2010. Rapperswil-jona, Suiza.

7.1 Contribución

Se propone comenzar una tendencia de trabajo continuado sobre la misma arquitectura, documentando minuciosamente los cambios llevados a cabo y los avances derivados de esas modificaciones.

El presente ***proyecto*** pretende comenzar esa ***base reutilizable tanto de hardware como de software***, que pueda ser utilizada por futuros alumnos, de manera que éstos puedan enfocar su esfuerzo en avanzar hacia metas más altas en el diseño de microrobots más complejos. Este ***documento*** pretende ser el ***manual de uso*** de esta base reutilizable.

7.2 Líneas futuras

Se mencionan brevemente a continuación algunos cambios que no han sido realizados pero que se estima que mejorarían la arquitectura del microrobot:

Programar la placa de servomotores en la FPGA que integra la placa Linux

Integración de todo el software de control en una única placa.

Ventajas: Eliminaría la comunicación entre ambas placas. Menos gasto de energía.

La FPGA permitiría crear un dispositivo con mejores prestaciones que la placa de servo-motores.

Desventajas: Multitud de horas de trabajo para programar la FPGA.

Sistema de posicionamiento basado en motores paso a paso

El posicionamiento con encoders tiene ciertas desventajas. Los motores paso a paso [32] permiten un posicionamiento del microrobot más preciso.

Ventajas: Mejora considerable en el posicionamiento.

Desventajas: Gasto económico

Sistema de apoyo al posicionamiento mediante balizas

El posicionamiento mediante motores paso a paso sería suficiente para conocer la ubicación del microrobot en todo momento. Sin embargo, si se quiere conocer la posición del microrobot rival, sería necesario un sistema de balizas.

Ventajas: Mejora en la creación de estrategias complejas

Desventajas: Gran número de horas de trabajo.

Diseño, especificación y creación de una base de tracción reutilizable anualmente

Las normas que definen las dimensiones máximas de los microrobots en la competición Eurobot no varían de un año a otro. Es por este motivo que la existencia de una base de tracción reutilizable permitiría avanzar en otras tareas, y por lo tanto, a alcanzar mejores objetivos.

Ventajas: Ahorro de tiempo en futuras ediciones para avanzar más en otros aspectos del robot

Desventajas: Horas de trabajo.

8 Referencias

8.1 Recursos bibliográficos

[1] José Ignacio Albillo Arribas, **Diseño Electrónico de un microrobot "Eurobot 2008"**, Universidad Carlos III de Madrid, Octubre de 2008

[2] José Luis Martín Gómez, **Diseño del sistema de control y accionamiento de recogida de muestras de un microrobot "Eurobot 2008"**, Universidad Carlos III de Madrid, Septiembre de 2008

8.2 Recursos electrónicos

[3] Eurobot
http://www.eurobot.org/

[4] Planète Sciences
http://fr.wikipedia.org/wiki/Plan%C3%A8te_Sciences

[5] La Ferté Bernard
http://es.wikipedia.org/wiki/La_Fert%C3%A9-Bernard

[6] Alcabot-Hispabot
http://asimov.depeca.uah.es/robotica/

[7] Single Board Computer
http://en.wikipedia.org/wiki/Single-board_computer

[8] Ethernet
http://es.wikipedia.org/wiki/Ethernet

[9] Puerto serie
http://es.wikipedia.org/wiki/Puerto_serie

[10] USB
http://es.wikipedia.org/wiki/Universal_Serial_Bus

[11] Protocolo de conexión Telnet
http://es.wikipedia.org/wiki/Telnet

[12] Memoria Flash
http://es.wikipedia.org/wiki/Memoria_flash

[13] Intel 8051
http://en.wikipedia.org/wiki/Intel_8051

[14] Disparador de Schmitt
http://es.wikipedia.org/wiki/Disparador_Schmitt

[15] Motor de corriente continua
http://es.wikipedia.org/wiki/Motor_de_corriente_continua

[16] Motores Bernio MR 615 30Q
http://bernio.it/show.php?Cod=mr61530q&Lang=ENG

[17] Modulación por ancho de pulsos. Señal PWM
http://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos

[18] Cuaderno técnico de servomotor Futaba 3003
http://www.iearobotics.com/proyectos/cuadernos/ct3/ct3.html

[19] Hoja de características de sensor Sharp - GP2D12
http://www.alldatasheet.com/datasheet-pdf/pdf/SHARP/GP2D12.html

[20] Sensor de final de carrera SS5GL2D
http://octopart.com/ss5gl2d-omron-917129

[21] Batería de plomo-ácido
http://es.wikipedia.org/wiki/Bater%C3%ADa_de_plomo_y_%C3%A1cido

[22] Linux Debian
http://es.wikipedia.org/wiki/Debian_GNU/Linux

[23] Lenguaje de programación C
http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C

[24] Señal cuadrada
http://es.wikipedia.org/wiki/Onda_cuadrada

[25] Timer de microcontrolador
http://en.wikipedia.org/wiki/Timer#Computer_timers

[26] Código ASCII
http://elcodigoascii.com.ar/el_codigo_ascii/el%20Codigo%20ASCII%20.gif

[27] Lenguaje de programación C#
http://es.wikipedia.org/wiki/C_Sharp

[28] Windows XP
http://es.wikipedia.org/wiki/Windows_XP

[29] Programa Keil µvision2
http://www.keil.com/uvision/

[30] Programa Microcontroller Tool Kit (MTK2) de Dallas Semiconductor
http://files.dalsemi.com/microcontroller/dev_tool_software/mtk/

[31] Microcontrolador DS89C450-MNG
http://datasheets.maxim-ic.com/en/ds/DS89C430-DS89C450.pdf

[32] Motor paso a paso
http://es.wikipedia.org/wiki/Motor_paso_a_paso

[33] Entorno de programación Eclipse
http://es.wikipedia.org/wiki/Eclipse_%28software%29

9 Anexos

9.1 Normativa del concurso Eurobot

9.1.1 Reglamento común a todos los años

Sólo se permite un robot por equipo.
Los robots deben ser completamente autónomos, no se permiten comunicaciones con elementos externos, excepto con las balizas.

Juego limpio
La finalidad del concurso es jugar el máximo número de partidos, de manera amistosa. De esta manera no se permite, y estará penalizado:

- Bloquear al robot contrario el acceso a los elementos del juego, así como impedir deliberadamente su movimiento.
- Diseñar el robot con el fin de confundir al contrario usando colores designados para los elementos del juego, así como partes del tablero.
- Hacer daño intencionado al robot contrario.

Seguridad
Los robots deben evitar tener partes puntiagudas o afiladas que puedan dañar al robot contrario.
Está prohibido el uso de líquidos, productos corrosivos, materiales pirotécnicos, o seres vivos dentro del robot.

Material obligatorio
Un cordón de al menos 50cm que se debe utilizar para arrancar el robot. Este cable debe ser la manera de hacer que el robot comience el partido, y al tirar de él, debe quedar completamente desacoplado del robot.

Botón de parada de emergencia, de al menos 2 cm de diámetro. Debe ser rojo, y al pulsarlo, el robot debe pararse inmediatamente, cortando el suministro de energía a cualquier parte mecánica del robot. Debe situarse en la parte superior del robot, siendo accesible en cualquier momento por el árbitro.

Apagado automático que actúe al transcurrir **90 segundos** tras el comienzo del partido. En ese momento, el robot debe cortar el suministro de energía a todas las partes mecánicas del robot.

Sistema de evasión de obstáculos que impida que el robot colisione contra un robot contrario. No es necesario esquivarlo, simplemente detectarlo a una distancia razonable para no chocarnos con él. Siempre se toma como modelo de robot un cilindro de 30cm por 20 cm de diámetro.

Soporte para baliza
El robot debe disponer de un soporte, donde el robot contrario podrá situar una baliza si así lo desea.
No es obligatorio tenerla, se puede homologar sin ella, pero en caso de que en un partido se compita contra un equipo que desee colocar una baliza en el robot contrario, se descalificará automáticamente al robot que no tiene soporte para baliza.

Dimensiones del robot
El tamaño de la base del robot debe tener dos estados.
El primer estado será el **estado replegado**, y será en el que empezará el robot. En este estado el robot no puede superar los **120cm de diámetro**.
Una vez comienza el partido el robot podrá pasar a un **estado desplegado**, donde su diámetro máximo podrá ser de **140cm**.

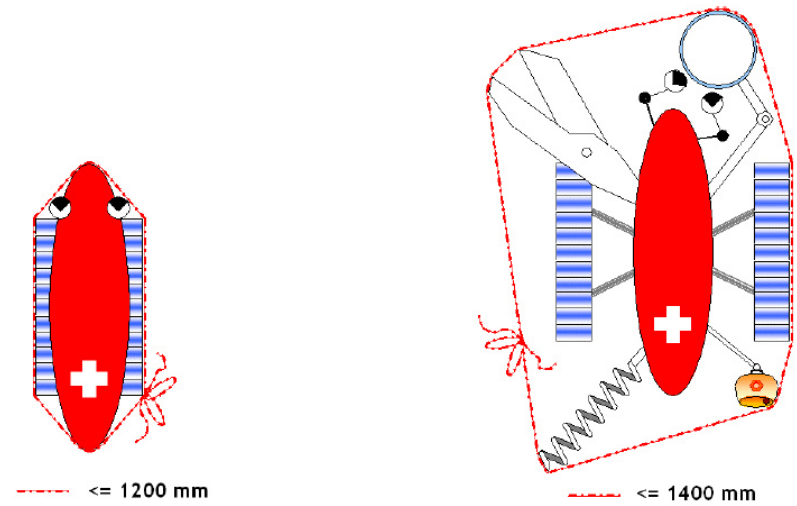


Ilustración 112 - Dimensiones máximas de la base del Robot

En ambos casos, el diámetro se mide usando una cinta métrica, o una tela con las dimensiones indicadas.

En cuanto a la **altura**, el tope máximo permitido es de **35 cm**, pudiendo llegar a 43 cm al añadir el soporte para baliza. Si el robot adversario coloca en este soporte su baliza, la altura del robot llegará a 51 cm.

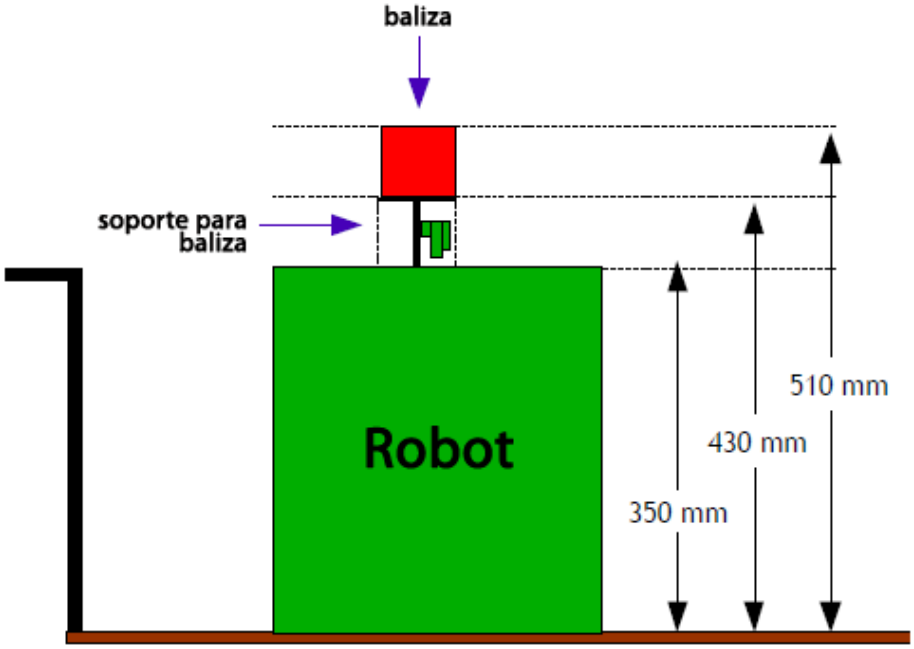


Ilustración 113 - Altura máxima del Robot

Penalizaciones

Cualquier acción incompatible con el espíritu de juego limpio que se fomenta en el concurso, será penalizado por los árbitros. Por ejemplo, los siguientes actos serán castigados:

- Un robot choca violentamente contra otro.
- Cuando un robot sea considerado peligroso para el robot adversario, para el campo de juego o para el público.
- Si un robot impide deliberadamente al robot oponente acceder a algún elemento del campo de juego.
- Cuando un robot arroja continuamente elementos fuera del campo de juego.

Descalificaciones

Un equipo será descalificado **de ese partido**, cuando alguna de estas situaciones ocurra:

- El equipo no llega a tiempo a la sala de espera previa a un partido donde se reúne a los equipos participantes
- El equipo no consigue estar preparado en el campo de juego en menos de 3 minutos
- El robot no llega el soporte para balizas y el equipo adversario solicita colocarle una baliza
- El robot no consigue abandonar completamente el área de inicio

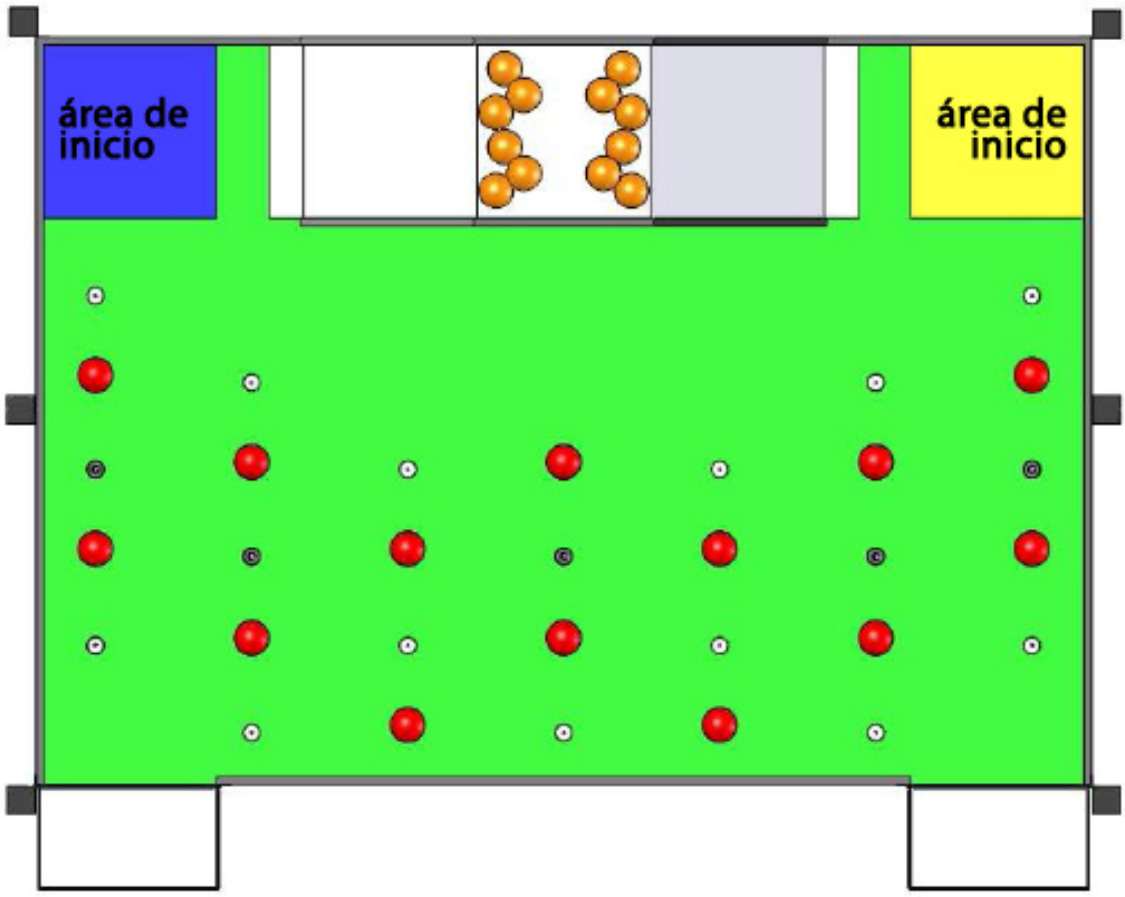


Ilustración 114 - Área de inicio del campo de juego

Un equipo será descalificado **de la competición**, cuando:

- El robot realice reiteradamente la misma acción penalizable
- El equipo tenga un comportamiento inaceptable
- El robot no cumpla con las normas de seguridad establecidas

Tamaño del campo de juego

Los encuentros se llevarán a cabo en un campo rectangular de **210 cm x 300 cm**, pudiendo ser aún mayor en algunas partes para albergar distintos elementos imprescindibles para el juego.

9.1.2 Reglamento específico de 2008 – Misión a Marte

Encontrar pruebas de vida y traerla a La Tierra para analizarlas

Campo de juego

El tablero de juego será de color gris. Se compondrá de:

- ***45 pelotas***. Las pelotas rojas son para el equipo rojo. Las pelotas azules son para el equipo azul. Las pelotas azules y rojas simbolizan las pruebas de vida. Las pelotas blancas son válidas para ambos equipos, simbolizan el hielo.
- ***4 dispensadores verticales***: 2 contienen pelotas blancas, otro contiene pelotas rojas y otro contiene pelotas azules.
- ***1 dispensador horizontal***: contienen 6 pelotas rojas y 6 pelotas azules.
- ***2 contenedores estándar***: uno para pelotas rojas y otro para pelotas azules. Se encuentran al nivel de altura del campo de juego.
- ***2 contenedores helados***: uno para pelotas rojas y otro para pelotas azules. Ubicados en los lados más cortos del campo de juego.
- Las 45 pelotas están distribuidas en el campo de la siguiente manera:
    - 13 sueltas por el tablero de juego (su distribución se realiza por sorteo): 9 blancas, 2 rojas y 2 azules
    - 20 pelotas en los dispensadores verticales
    - 12 pelotas en el dispensador horizontal

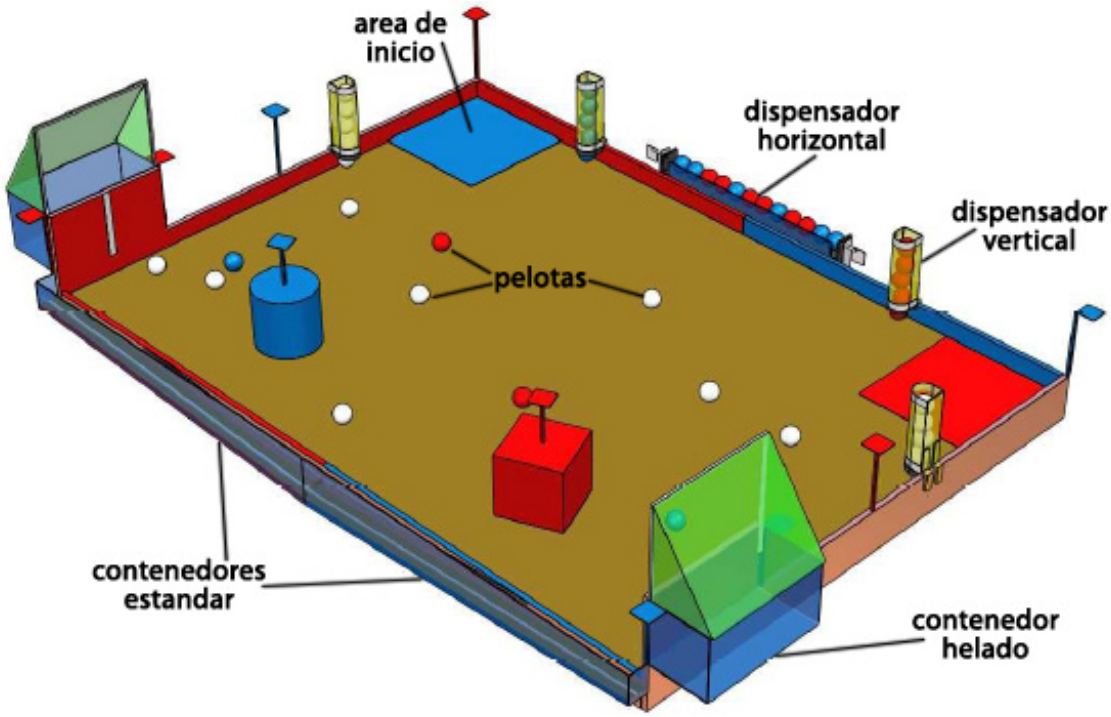


Ilustración 115 - Campo de juego Eurobot 2008

Puntuación

Cada equipo debe recoger las pelotas de su color (o también las blancas) que se encuentran en los dispensadores o sueltas en el tablero de juego y depositarlas en cualquiera de los contenedores (estándar o helado) ***de su color***, es decir, el equipo rojo en los contenedores rojos y el equipo azul en los azules. Al terminar los 90 segundos del partido, se hará el recuento de la siguiente forma:

- Las ***pelotas blancas*** (hielo) que se encuentren en los contenedores azules/rojos sumarán ***1 punto*** para el equipo azul/rojo
- Las ***pelotas azules*** que se encuentren en ***cualquier contenedor estándar*** (ya sea el rojo o el azul) sumarán ***2 puntos*** para el equipo azul.
- Las ***pelotas rojas*** que se encuentren en ***cualquier contenedor estándar*** (ya sea el rojo o el azul) sumarán ***2 puntos*** para el equipo rojo.
- Las ***pelotas azules*** que se encuentren en el ***contenedor helado azul*** sumarán ***2 puntos*** para el equipo azul.
- Las ***pelotas rojas*** que se encuentren en el ***contenedor helado rojo*** sumarán ***2 puntos*** para el equipo rojo.
- Cualquier pelota ubicada en un ***contenedor helado que no sea de su color***, ***restará 1 punto*** al equipo del color de la pelota.

9.1.3 Reglamento específico de 2009 – Templos de Atlantis

Construir las columnas y los templos más altos de Atlantis

Campo de juego

El campo de juego es de color azul. Se compone de:

- ***2 zonas de construcción de nivel 1***. Ubicadas en el lado opuesto a las áreas de inicio.
- ***1 zona de construcción de nivel 2***. Situado entre las dos zonas de nivel 1.
- ***1 zona de construcción de nivel 3***. De forma circular, ubicada en el centro del tablero.

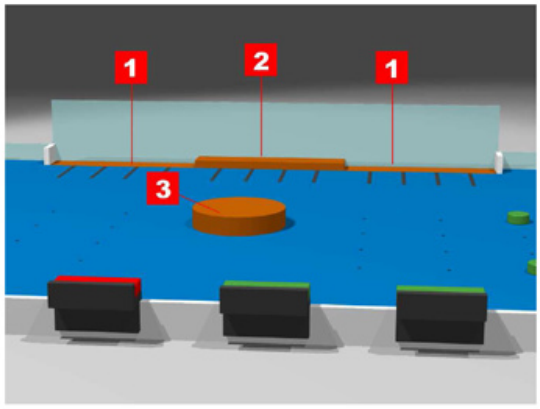


Ilustración 116 - Distintos niveles de zonas de construcción

- ***32 cilindros para construir columnas***: 16 de color rojo (para el equipo rojo) y 16 de color verde (equipo verde)
- ***4 dinteles***: 2 rojos (equipo rojo) y 2 verdes (equipo verde). Cada robot además podrá opcionalmente llevar pre cargado un dintel de su color.
- ***4 dispensadores verticales de cilindros***. Dos de ellos contendrán 8 cilindros de color rojo, y los otros dos 8 cilindros de color verde.
- ***4 dispensadores de dinteles***. Cada uno de ellos contendrá un dintel, dos de ellos un dintel de color rojo, los otros dos de color verde.

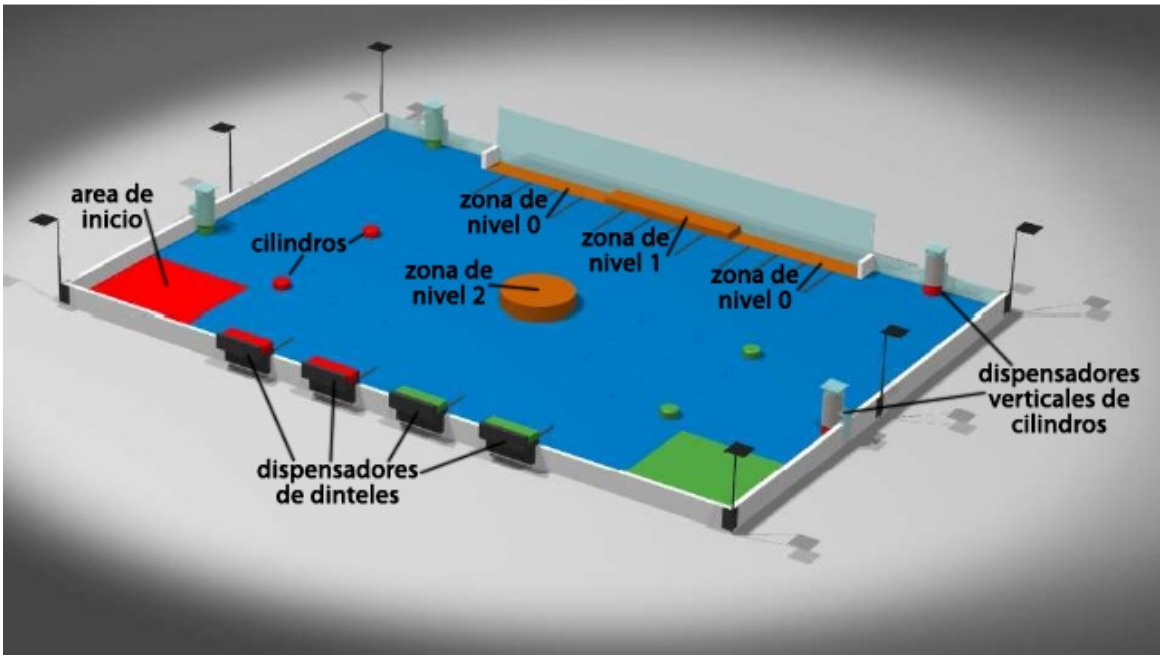


Ilustración 117 - Campo de juego Eurobot 2009

Puntuación

Cada equipo debe construir la mayor cantidad de templos y columnas que le sea posible, utilizando los cilindros y dinteles de su color que se encuentran las distintas partes del campo de juego (dispensadores o tablero).

Las construcciones deben llevarse a cabo en cualquiera de las zonas de construcción. La puntuación obtenida por cada columna o templo dependerá del nivel de la zona donde se haya construido; a más nivel, más puntuación.

Al terminar los 90 segundos del partido, se hará el recuento. Será considerado válido para puntuar:

- Cualquier cilindro que se encuentre completamente dentro de una de las zonas de construcción, apoyado sobre la misma.
- Aquel cilindro que se encuentre completamente dentro de una de las zonas de construcción, apoyado en otro elemento considerado válido para puntuar (otro cilindro o dintel).
- Un dintel que se encuentre completamente dentro de una de las zonas de construcción, apoyado (por uno de sus dos lados más grandes) en al menos dos elementos considerados válidos para puntuar (cilindro o dintel).

No es necesario que los elementos se apoyen sobre otros del mismo color para ser considerados válidos. El cilindro verde de la siguiente ilustración será válido.

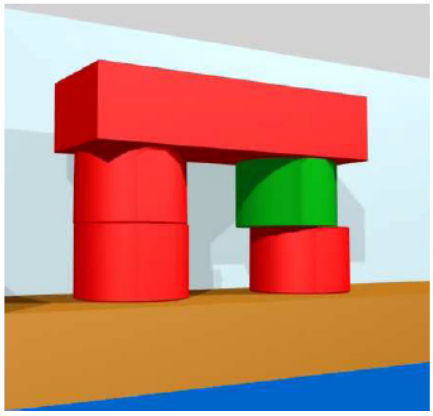


Ilustración 118 - Situación válida

Los distintos elementos considerados válidos, puntuarán de la siguiente manera:

- Cada cilindro válido apoyado directamente sobre una zona de construcción obtendrá tantos puntos como el nivel en el que esté construido. Por ejemplo, un cilindro construido en una zona de construcción de nivel 1 obtendrá 1 punto.
- Cada cilindro válido apoyado sobre otro elemento válido obtendrá tantos puntos como el nivel al que esté construido, más el número de alturas que tenga debajo. Por ejemplo, un cilindro construido en una zona de nivel 3, apoyado sobre otro cilindro que está apoyado directamente sobre la zona de construcción obtendrá 3 + 1 = 4 puntos.
- Cada dintel válido obtendrá la puntuación de forma análoga a los cilindros, multiplicando por 3 la puntuación final. Por ejemplo, un dintel construido en una zona de nivel 2, apoyado sobre dos cilindros que están apoyados directamente sobre la zona de construcción obtendrá (2 + 1) x 3 = 9 puntos

9.1.4 Reglamento específico de 2010 – Alimenta al mundo

Recolectar la mayor cantidad de frutas, hortalizas y semillas

Campo de juego

El tablero de juego es de color verde. Se compone de:

2 contenedores. Uno para cada equipo. Cada equipo deberá depositar en su contenedor correspondiente los distintos alimentos que vaya recolectando.

12 naranjos. Situados en 2 grupos de 6, en la parte alta de una cuesta. Cada grupo está ubicado próximo al área de inicio de cada uno de los equipos. Cada naranjo contiene una naranja. Por lo tanto, habrá 12 naranjas en juego, representadas por pelotas de malabares de color naranja.

14 tomates. Representados por pelotas de color rojo. Distribuidos a lo largo del tablero de juego

18 mazorcas. Clavadas verticalmente en el tablero de juego. 9 de ellas de color blanco, y las otras 9 de color negro. Las mazorcas blancas son fácilmente extraíbles de donde hayan sido clavadas, mientras que las mazorcas negras están atornilladas al suelo, imposibilitando su extracción.

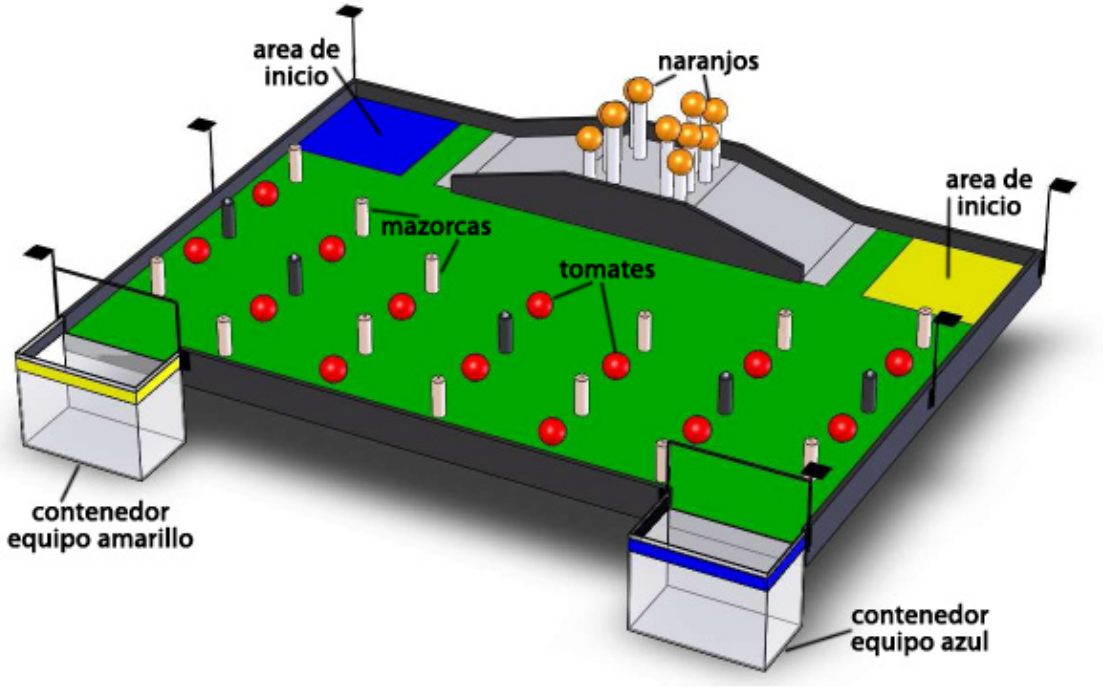


Ilustración 119 - Campo de juego Eurobot 2010

Puntuación

Cada equipo debe recolectar la mayor cantidad de alimentos posibles (tomates, mazorcas o naranjas) que se encuentran en los naranjos o sueltas en el tablero de juego y depositarlas en el contenedor que le corresponde, el ***de su color***.

Al terminar los 90 segundos del partido, se hará el recuento en función del peso de los elementos recolectados, de la siguiente forma:

Los ***tomates*** pesan 150g, por lo tanto, cada tomate contenido en el contenedor de un equipo sumará ***150 puntos*** a ese equipo.

Las ***mazorcas*** pesan 250g, por lo que cada mazorca existente en el contenedor de un equipo aumentará en ***250 puntos*** el marcador de ese equipo.

Las ***naranjas*** pesan 300g. Cada naranja contenida en el contenedor de un equipo le dará una cantidad de ***300 puntos***.

9.2 Código del programa de la placa de servo-motores

```
/*
 * Este programa es capaz de manejar dos motores, identificados como derecho e izquierdo.
 * Para cada uno de los motores se puede indicar su velocidad y su dirección (adelante o atrás)
 *
 * Nada más iniciar el programa los motores estarán parado, con los frenos activos hasta que reciban una orden
 *
 * Programa capaz de manejar hasta 8 servos, ampliable a lo que sea
 * El tipo de servos que maneja son los FUTABA 3003
 *
 * Mediante el puerto serie recibe paquetes de 2 bytes, donde el primer byte
 * es el número de servo + 97, esto se hace para que el servo cero corresponda con la letra 'a' en ASCII. El segundo byte es la posición del servo, un número comprendido entre 0 y 255.
 *
 * Nada más iniciar el micro, no pone ningún servo a ninguna posición, los servos se quedan libres hasta que reciban alguna orden.
 */

#include <stdio.h>
#include <reg420.h>

/* Estas tres definiciones no sirven para nada pero ahí están */
#define MAX_UNSIGNED_INT 65535U
#define MAX_UNSIGNED_LONG 4294967295L
#define CRYSTAL_FREQUENCY 20000000L      //    Value in Hz

// Initialize serial port0 at 57600 baud rate, through the Timer 1 start value
#define TIMER1 245
// Timer 0 interrupt every about 0.0102ms
#define TIMER0 239
// PWM Period: PWM_PERIOD * TIMER0 period = 20ms
#define PWM_PERIOD 1960U

#define MAX_SPEED      26

/* Si ponemos este ángulo el servo no hace fuerza, se queda libre */
#define INITIAL_ANGLE   PWM_PERIOD

/* Información sobre los servos futaba 3003 en
 * http://www.iearobotics.com/proyectos/cuadernos/ct3/ct3.html */

/* Ángulo mínimo 0.3ms / 20ms  ~= 30 / 1960 */
#define MIN_ANGLE      30U
/* Ángulo máximo 2.3ms / 20ms  ~= 226 / 1960 */
#define MAX_ANGLE      226U

/* El número total de servos que controla la aplicación */
#define NUM_SERVOS           8
```

```
//*********************************************************************
//********************  DEFINICIONES      *********************************

/* Cada una de las patitas del micro que van a los motores
   (estas salidas necesitan pull-up's) */

sbit  direccion_rueda_left              = P0^0;
sbit  direccion_rueda_right             = P0^1;
sbit  pwm_left                          = P0^2;
sbit  pwm_right                         = P0^3;

/* Cada servo está enganchado a una patita del micro, aquí le decimos a que
patita está cada servo */

sbit    SERVO0          = P2^0;
sbit    SERVO1          = P2^1;
sbit    SERVO2          = P2^2;
sbit    SERVO3          = P2^3;
sbit    SERVO4          = P2^4;
sbit    SERVO5          = P2^5;
sbit    SERVO6          = P2^6;
sbit    SERVO7          = P2^7;


/* Definiciones que nos ayudan para no pensar si adelante es un 0, si el freno es
un 1, etc. Nótese que las direcciones para la rueda derecha y la izquierda están
invertidas */
#define DIRECCION_DERECHA_ADELANTE      0
#define DIRECCION_DERECHA_ATRAS         1

#define DIRECCION_IZQUIERDA_ADELANTE    1
#define DIRECCION_IZQUIERDA_ATRAS       0

/* Carácter de aceptación en comunicaciones */
#define CARACTER_ACK                    0x23


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

// Factores de ajuste de la velocidad
unsigned int factor_izquierda, factor_derecha;
// Encoders
unsigned long int encoder_left, encoder_right;
// Parámetros de velocidad
unsigned int actual_speed_left, actual_speed_right;
unsigned int speed_left, speed_right;


/* En el vector “actual_angle” está el ángulo que realmente se está pasando
mediante la PWM a cada uno de los servos */
unsigned int actual_angle[NUM_SERVOS];

/* En angle se guarda el valor del ángulo que se quiere pasar a cada servo, cada
vez que termine un ciclo de PWM se asignan a las variables actual_angle el valor
que esté en angle */
unsigned int angle[NUM_SERVOS];

/* Counter sirve para saber en que momento del ciclo PWM está (entre 0 y
PWM_PERIOD)*/
unsigned int counter;
```

```
//*****************************************************************************
//************************    INTERRUPCIONES    *********************************

void timer0 (void) interrupt 1 {

      /* MOTORES */
      if(counter < actual_speed_left) {
            pwm_left = 1;
      } else {
            pwm_left = 0;
      }
      if(counter < actual_speed_right) {
            pwm_right = 1;
      } else {
            pwm_right = 0;
      }

      /* SERVOS */
      if(counter < actual_angle[0]) {
            SERVO0 = 1;
      } else {
            SERVO0 = 0;
      }
      if(counter < actual_angle[1]) {
            SERVO1 = 1;
      } else {
            SERVO1 = 0;
      }
      if(counter < actual_angle[2]) {
            SERVO2 = 1;
      } else {
            SERVO2 = 0;
      }
      if(counter < actual_angle[3]) {
            SERVO3 = 1;
      } else {
            SERVO3 = 0;
      }
      if(counter < actual_angle[4]) {
            SERVO4 = 1;
      } else {
            SERVO4 = 0;
      }
      if(counter < actual_angle[5]) {
            SERVO5 = 1;
      } else {
            SERVO5 = 0;
      }
      if(counter < actual_angle[6]) {
            SERVO6 = 1;
      } else {
            SERVO6 = 0;
      }
      if(counter < actual_angle[7]) {
            SERVO7 = 1;
      } else {
            SERVO7 = 0;
      }

      counter++;
```

```
/* Si entramos aquí significa que estamos al final del ciclo de la PWM
reiniciamos el contador, y asignamos a las variables 'actual' los nuevos valores
*/

      if (counter == PWM_PERIOD) {

            actual_angle[0] = angle[0];
            actual_angle[1] = angle[1];
            actual_angle[2] = angle[2];
            actual_angle[3] = angle[3];
            actual_angle[4] = angle[4];
            actual_angle[5] = angle[5];
            actual_angle[6] = angle[6];
            actual_angle[7] = angle[7];

            actual_speed_left = speed_left;
            actual_speed_right = speed_right;

            counter = 0;
      }

}

/* Cada vez que el encoder de la rueda derecha cuente un pulso en el movimiento
del motor, enviará una interrupción que será recogida en esta entrada. Se llevará
la cuenta del número de pulsos contados en la variable “encoder_right” */

void int0 (void) interrupt 8
{
      encoder_right++;
      EXIF &= 0xEF ;            // Bajada flag interrupcion 2
}

/* Cada vez que el encoder de la rueda izquierda cuente un pulso en el movimiento
del motor izquierdo, enviará una interrupción que será recogida en esta entrada.
Se llevará la cuenta del número de pulsos contados en la variable “encoder_left”
*/

void int1 (void) interrupt 9
{
      encoder_left++;
      EXIF &= 0xDF ;            // Bajada flag interrupcion 3
}
```

```
//*********************************************************************
//**********************      INICIALIZACION    **************************

void ports_init(void)
{
      /* iniciamos patitas de motores */
      direccion_rueda_left          = DIRECCION_IZQUIERDA_ADELANTE;
      direccion_rueda_right         = DIRECCION_DERECHA_ADELANTE;
      pwm_right                           = 0;
      pwm_left                            = 0;

      /* iniciamos patitas de servos */
      SERVO0 = 1;
      SERVO1 = 1;
      SERVO2 = 1;
      SERVO3 = 1;
      SERVO4 = 1;
      SERVO5 = 1;
      SERVO6 = 1;
      SERVO7 = 1;
}

void variables_init(void)
{
      int i;
      /* Inicializamos variables de motores*/
      encoder_left = 0;
      encoder_right = 0;
      actual_speed_left = 0;
      actual_speed_right = 0;
      speed_left = 0;
      speed_right = 0;
      factor_izquierda = 0;
      factor_derecha = 0;

      /* Inicializamos variables de servos */
      for(i = 0; i < NUM_SERVOS; i++) {
            angle[i] = INITIAL_ANGLE;
            actual_angle[i] = INITIAL_ANGLE;
      }
}

/* Inicialización de los parámetros del puerto serie */
void serial0_init(void)
{
      SBUF0 = ' ';            //    Clearing the serial port 0 buffer
      TMOD |= 0x20;           //    Timer 1: 8-bit autoreload from TH1
      TH1 = TIMER1;           //    Timer 1 reload value
      TL1 = TIMER1;           //    Timer 1 initial value
      CKMOD |= 0x10;          //    Use system clock for timer 1 counting clock
      TCON |= 0x40;           //    Enable timer 1
      SCON0 = 0x50;           //    Enable serial port 0 in mode 1
}

/* Inicialización de las interrupciones que usarán los encoders para informar a
la placa de un pulso en el desplazamiento del correspondiente motor */
void interrupts_init(void)
{
      IE |= 0x80;
      EIE |= 0x03 ; // Enable interrupt 2,3 (Para los encoders)
}
```

```
/* Inicialización del timer0 para los servos */
void timer0_init(void)
{
      TMOD |= 0x02;                 //    Timer 0: 8bit autoreload from TH0
      TH0 = TIMER0;                 //    Timer 0 reload value
      TL0 = TIMER0;                 //    Timer 0 initialized
      CKMOD &= 0xFB;
      CKCON &= 0xFB;                //    Use f/12 for timer 0 counting clock
      IE |= 0x02;                   //    Timer0 interrupt enable
      TR0 = 1;                      //    Start Timer0
}



//***********************************************************************
//************************  FUNCTIONS   *********************************


// Función para controlar los servomotores
void controlar_servos (void) {

      unsigned char posicion;
      unsigned char num_servo;

      scanf("%c%c", &num_servo, &posicion);

/* Restamos 48 para que si nos mandan una '0' (48 en ASCII) signifique es el
servo 0 */
      num_servo = num_servo - 48;

/* En caso de que nos manden un número de servo incorrecto, simplemente no
hacemos nada */
      if (num_servo >= 0 && num_servo < NUM_SERVOS ) {

            if(posicion == 0) {
                  angle[num_servo] = INITIAL_ANGLE;
            } else {

/* Esta fórmula hace que el ángulo esté entre MIN_ANGLE y MAX_ANGLE*/
                  angle[num_servo] = ((posicion * (MAX_ANGLE - MIN_ANGLE) / 255)
                                    + MIN_ANGLE);
            }
      }
}
```

```
// Función para controlar los motores
void controlar_motores (void) {
      unsigned char lectura_izquierda, lectura_derecha;
      unsigned long velocidad_izquierda, velocidad_derecha;

      scanf("%c%c", &lectura_izquierda, &lectura_derecha);

      velocidad_izquierda = lectura_izquierda;
      velocidad_derecha = lectura_derecha;


      // Si es una letra mayuscula, será velocidad positiva. Cuanto mayor (a -
            z), más velocidad. Hay 26 distintas

      // RUEDA IZQUIERDA
      if (velocidad_izquierda >= 'A' && velocidad_izquierda <= 'Z') {
            velocidad_izquierda -= 65;
            direccion_rueda_left    = DIRECCION_IZQUIERDA_ADELANTE;
      }

      if (velocidad_izquierda >= 'a' && velocidad_izquierda <= 'z') {
            velocidad_izquierda -= 97;
            direccion_rueda_left    = DIRECCION_IZQUIERDA_ATRAS;
      }

      velocidad_izquierda = velocidad_izquierda * PWM_PERIOD / MAX_SPEED;

      // RUEDA DERECHA
      if (velocidad_derecha >= 'A' && velocidad_derecha <= 'Z') {
            velocidad_derecha -= 65;
            direccion_rueda_right   = DIRECCION_DERECHA_ADELANTE;
      }

      if (velocidad_derecha >= 'a' && velocidad_derecha <= 'z') {
            velocidad_derecha -= 97;
            direccion_rueda_right   = DIRECCION_DERECHA_ATRAS;
      }

      velocidad_derecha = velocidad_derecha * PWM_PERIOD / MAX_SPEED;

      // Se le añaden los factores de correccion de velocidad
      if (velocidad_izquierda > factor_izquierda) {
            velocidad_izquierda -= factor_izquierda;
      }

      if (velocidad_derecha > factor_derecha) {
            velocidad_derecha -= factor_derecha;
      }

      // Asignamos la velocidad correspondiente a cada rueda
      speed_left        = velocidad_izquierda;
      speed_right       = velocidad_derecha;

}
```

```
// Función que permite mover los motores un determinado número de pasos
void muevete_pasos (void) {

	unsigned char lectura_pasos;
	unsigned long num_pasos;

	// Se lee el número de pasos a moverse
	scanf("%c", &lectura_pasos);
	num_pasos = lectura_pasos;

	while (encoder_right < num_pasos);
// Permanece haciendo el movimiento que ya estaba haciendo hasta llegar a los pasos. Se le deja exactamente en la misma velocidad a la que estaba, ya que se supone que este metodo se ejecuta despues de estar moviendote, para llegar exactamente al numero de pasos que estabas buscando

	speed_left = 0;
	speed_right = 0;
}

void consultar_encoder_izquierdo(void) {
	int encoder;
	unsigned char byte_1, byte_2;
	unsigned char accion_intermedia;

	encoder = encoder_left;
	encoder_left = 0;

	byte_1 = encoder % 90;
	byte_2 = encoder / 90;
	byte_1 += 32;
	byte_2 += 32;

	printf("%c", byte_1);
	scanf("%c", &accion_intermedia);

	if(accion_intermedia == 'i') {
		printf("%c", byte_2);
	}
}

void consultar_encoder_derecho(void) {
	int encoder;
	unsigned char byte_1, byte_2;
	unsigned char accion_intermedia;

	encoder = encoder_right;
	encoder_right = 0;

	byte_1 = encoder % 90;
	byte_2 = encoder / 90;
	byte_1 += 32;
	byte_2 += 32;

	printf("%c", byte_1);
	scanf("%c", &accion_intermedia);

	if(accion_intermedia == 'd') {
		printf("%c", byte_2);
	}
}
```

```
// Reinicio de la placa. Configuración con los parámetros iniciales
void reset(void) {
      int i;

      /* Velocidades a 0 */
      speed_left = 0;
      speed_right = 0;

      direccion_rueda_left          = DIRECCION_IZQUIERDA_ADELANTE;
      direccion_rueda_right         = DIRECCION_DERECHA_ADELANTE;

      /* Reiniciamos servos */
      for(i = 0; i < NUM_SERVOS; i++) {
            angle[i] = INITIAL_ANGLE;
      }

      encoder_left = 0;
      encoder_right = 0;
}


/* Para conseguir una trayectoria lo más recta posible en el desplazamiento
adelante del microrobot, esta función permite regular el movimiento de una rueda
respecto a la otra. */
void cambiar_factor_correccion (void) {
      unsigned char new_factor_izquierda, new_factor_derecha;

      scanf("%c%c", &new_factor_izquierda, &new_factor_derecha);

      factor_izquierda = new_factor_izquierda;
      factor_derecha = new_factor_derecha;
}
```

```
//*********************************************************************
//************************    MAIN PROGRAM      ***************************

void main()
{
      unsigned char accion;

/* Inicializamos las patitas del micro que vamos a usar */
      ports_init();

/* Inicializamos todas las variables globales que usa el programa */
      variables_init();

/* Inicializamos el timer0 encargado de generar la PWM para todos los servos */
      timer0_init();

/* Inicializamos todo lo necesario para poder leer datos del puerto serie*/
      serial0_init();


/* Inicilaizamos las interrupciones */
      interrupts_init();
      TR0=1;


/* Bucle principal, simplemente recoge órdenes del Puerto serie y las deriva a la
función correspondiente */

      while (1) {
            scanf("%c", &accion);

            /* RESET */
            if(accion == 'r') {
                  reset();
                  printf("%c", CARACTER_ACK);
            } else if (accion == 's') {
                  controlar_servos ();
                  printf("%c", CARACTER_ACK);
            } else if (accion == 'm') {
                  controlar_motores ();
                  printf("%c", CARACTER_ACK);
            } else if (accion == 'i') {
                  consultar_encoder_izquierdo();
            } else if (accion == 'd') {
                  consultar_encoder_derecho();
            } else if (accion == 'f') {
                  cambiar_factor_correccion();
                  printf("%c", CARACTER_ACK);
            } else if (accion == 'p') {
                  muevete_pasos();
                  printf("%c", CARACTER_ACK);
            }
      }
}
```