Escuela Politécnica Superior de Elche

CONTROL DE ROBOTS Y SISTEMAS SENSORIALES

PRÁCTICAS DE CONTROL VISUAL DE ROBOTS
Taller 1. Introducción al controlador Eyebot

Prof. Luis Miguel Jiménez

Departamento de Ingeniería
Área de Ingeniería de Sistemas y Automática

1 OBJETIVO

El objetivo de este documento es mostrar una introducción a la programación del robot SoccerBot basado en le controlador Eyebot. Se recomienda consultar el servidor web (http://lorca.umh.es/isa/es/temas/eyebot) donde se encuentra documentación adicional.

Se describirán brevemente en primer lugar, la tarjeta controladora y el robot. Posteriormente se describirá en el entorno de desarrollo, la descarga de programas y un ejemplo básico. Por último se describe la librería de programación ROBIOS y el simulador EyeSim.

2 MATERIAL EMPLEADO

La práctica se realizará en grupos de tres personas, disponiendo de un PC con S.O. Windows 95/98/2000/XP, el entorno de desarrollo Robios 6.0, y un Robot SoccerBot. La documentación y el software se puede encontrar en el servidor http://lorca.umh.es/isa/es/temas/eyebot

2.1 ¿Qué es Eyebot?

Eyebot es una tarjeta controladora orientada al control de robots móviles. Está basada en el microcontrolador de 32 bits Motorota 68332 (figura 1). Incorpora una CPU de 32 bits bastante potente que permite manejar una cámara de video digital, dispone así mismo de un módulo de entradas temporizadas TPU con 16 E/S totalmente configurables. Las características más destacables son:

CARACTERÍSTICAS DEL CONTROLADOR EYEBOT MK3

- Microcontrolador Motorola 68332 25MHz 32bit
- 1MB RAM
- 512KB Flash-ROM (para el S.O. + programas de usuario)
- 1 puerto paralelo
- 2 puertos serie
- 8 entradas digitales
- 8 salidas digitales
- 8 entradas analógicas
- Interface para cámara color y monocromo
- LCD gráfico (128x64 pixels)
- 4 botones de entrada
- Botón de reset , interruptor de alimentación
- Altavoz para salida de audio
- Micrófono para entrada de audio
- Indicador de nivel de batería
- Drivers de potencia para 2 motores DC incluidos en la tarjeta
- Circuito background debugger BDM incluido en la tarjeta
- Conectores "plug-and-play" para motores, sensores, cámara, puerto serie 6 sensores infrarrojos, 2 sensores de colisión, 6 entradas analógicas libres
- Cable serie para descarga de programas estándar: 9pin RS232
- Módulo de comunicación inalámbrica vía radio integrado.

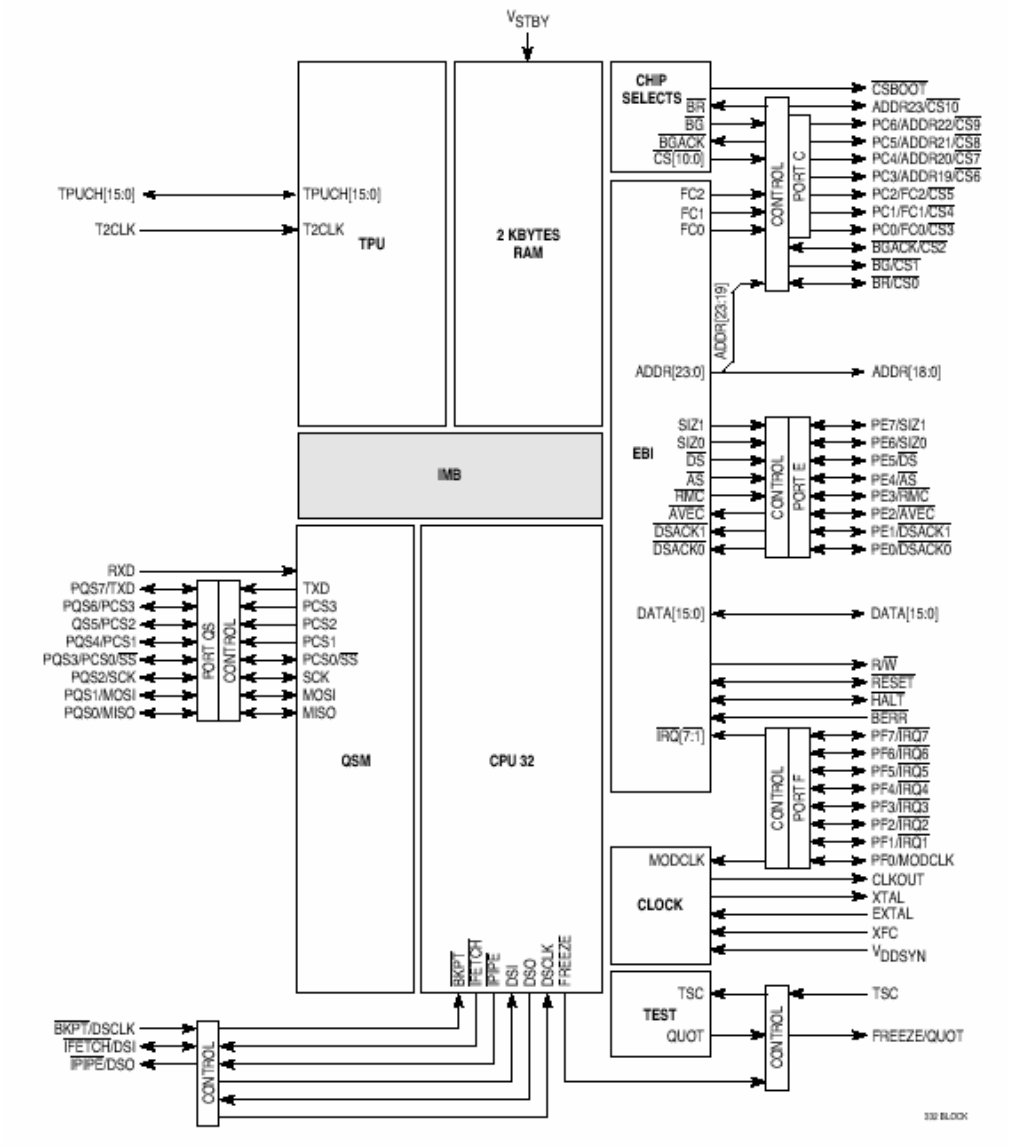


Figura 1 Microcontrolador Motorola 68332

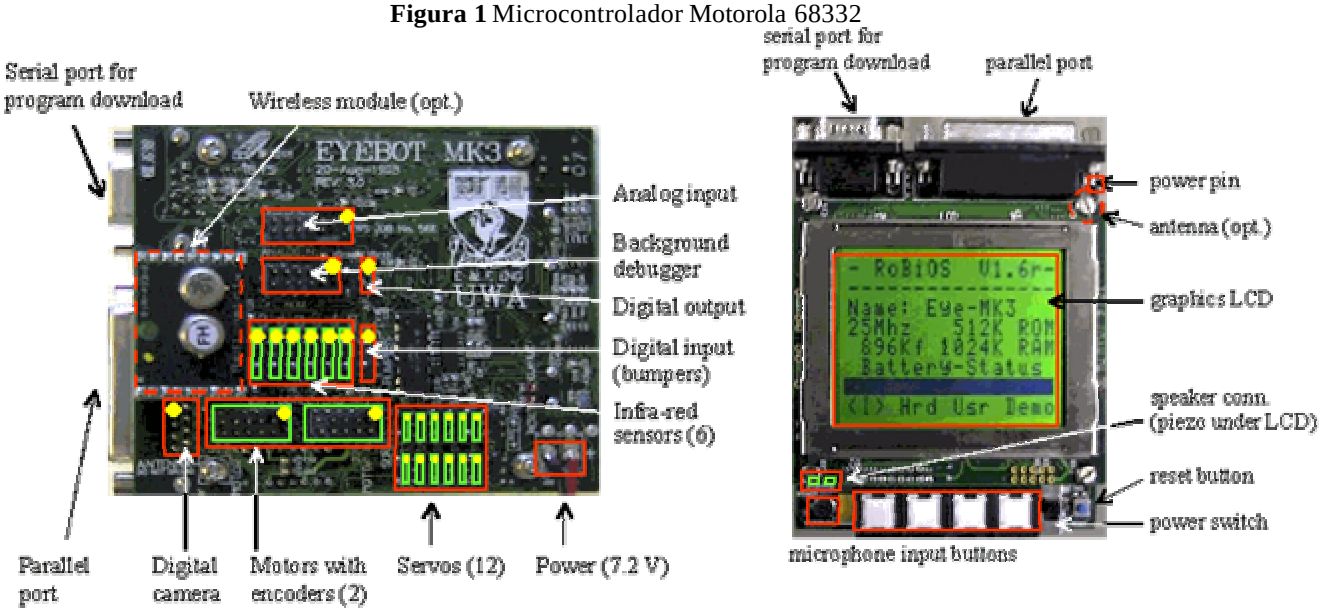


Figura 2. Descripción de los conectores de la tarjeta Eyebot

2.2 Conectores de la tarjeta Eyebot

En la figura 2 se describen los conectores para los diferentes dispositivos. El pin marcado de amarillo en la figura corresponde al pin 1 del conector:

- **Cámara:** permite conectar la cámara digital en color EyeCam (80x60 pixels de resolución con 24bits de profundidad de color). El pin 1 debe conectarse normalmente al cable rojo (siempre los salientes del conector hacia arriba).
- **LCD** : el display permite representar en blanco y negro texto (8x16 caracteres) y gráficos (64x128 pixels).
- **Fuente de Alimentación**: Se dispone de dos pines marcados "+" (cable rojo) y "-" (cable negro). La alimentación puede variar entre 7 y 9 voltios. El robot dispone de una batería recargable Li-Ion de 7,2V
- **Conector Serie**: Utiliza un conector estándar DB-9 con señalización RS-232. Permite conectar el robot con un PC para descarga de programas. El cable debe realizar una conexión 1-a-1 sin cruzar (cable plano).
- **Conectores para motores DC y encoders:** se dispone de dos conectores para motores DC con sus correspondientes encoders en cuadratura. Los drivers de los motores están integrados en la misma tarjeta. El conector es directamente compatible con los conectores usados por diferentes fabricantes de motores.
- **Conectores para Servos:** se dispone de 12 conectores de 3 pines cada uno (algunos de ellos están ya ocupados) para la conexión directa de servomotores controlados mediante modulación PWM. Corresponden a las patillas 0-11 de la TPU. La mayoría de los servos estándar son compatibles con bs pines utilizados (pin 1 corresponde al cable amarillo de señal)

  Nota:

  - Cuando se usan los motores DC, los conectores de servos 1-6 no pueden ser usados
  - La configuración de el periodos PWM, los ángulos y límites puede ser configurados mediante el fichero HDT.

- **Conector Infrarrojos:** se dispone de 6 conectores, de 4 pines cada uno, para sensores de distancia infrarrojos (PSD) Sharp GP2D02. Proporcionan una medida en 8 bits de la distancia (10-70 cm) a un obstáculo. El robot dispone de tres de estos sensores. El pin 1 debe conectarse al cable negro.
- **Conector Altavoz:** la tarjeta dispone de un pequeño altavoz y de un conector para la conexión de un altavoz externo para mejorar la calidad del sonido.
- **Micrófono:** Se dispone de un micrófono miniatura en la parte frontal.

- **Conectores para Extensiones**: Se dispone de 3 conectores adicionales para añadir nuevos dispositivos:
  - Conector de entradas digitales (2 entradas, 4 pines)
  - Conector de salidas digitales (3 entradas, 3 pines)
  - Conector de entradas analógicas (6 entradas, 10 pines)

- **Background Debugger** (BDM): permite depurar a bajo nivel programas desde un PC. En caso de que se corrompa la BIOS almacenada en FLASH este es el único método de volverla a descargar en la tarjeta.

DETALLES DE LOS CONECTORES DEL CONTROLADOR EYEBOT MK3

Serial Download (9 pin)
Standard RS-232 serial port

1 -
2 Tx
3 Rx
4 -
5 GND
6 -
7 CTS
8 RTS
9 -

DC Motor and Encoder Connectors (2 times 10 pin)
Motors are mapped to TPU channels 0..1
Encoders are mapped to TPU channels 2..5

1 Motor +
2 Vcc
3 Encoder channel A
4 Encoder channel B
5 GND
6 Motor -
7 --
8 --
9 --
10 --

Analog Input Connector (10 pin)
Microphone is mapped to analog input 0
Battery level gauge is mapped to analog input 1

1 Vcc
2 Vcc
3 analog input 2
4 analog input 3
5 analog input 4
6 analog input 5
7 analog input 6
8 analog input 7
9 analog GND
10 analog GND

Digital Input Connector (4 pin)
Infrared PSDs are mapped to digital input 0..5
The remaining 2 inputs have been configured with pull-up resistors and GND-pins

1 digital input 6 (pull-up 10K)
2 GND
3 digital input 7 (pull-up 10K)
4 GND

Digital Output Connector (3 pin)
Motors are mapped to digital output 0..3
Infrared PSD pulse is mapped to digital output 4

1 digital output 5
2 digital output 6
3 digital output 7

Digital Camera (16 pin)
6 Pin connector requires 1:1 connection
(cable with female:female) to EyeCam digital color camera

(Pin 1 =>CABLE ROJO)

Infrared Connectors (6 times 4 pin)
Sensor outputs are mapped to digital input 0..5

1 GND *(CABLE NEGRO)*
2 Vin (pulse)
3 Vcc
4 Sensor output (digital)

Servo Connectors (12 times 3 pin)
Servo signals are mapped to TPU channels 2..13

1 Signal (CABLE AMARILLO)
2 Vcc
3 GND

Parallel Port (25 pin)
Standard parallel port

Background Debugger Connector (10 pin)
Standard Motorola background debugger, connects to PC parallel port

2.3 El robot SoccerBot

Se trata de un robot basado en la tarjeta Eyebot diseñado originalmente en la Universidad de Perth (Australia) para participar en la competición oficial de la RoboCup. Se trata de una plataforma bastante avanzada que permite el uso de sensores de visión para el control de robots. El robot está constituido por (figura 3):

- La tarjeta controladora Eyebot mk3 con módulo de comunicación vía radio.
- 2 motores DC con reductora y encoders diferenciales
- Un servomotor para orientación de la cámara
- 3 sensores infrarrojos PSD Sharp GP2D02
- Una cámara EyeCam CCD en color de 24 bits y resolución 80*60 pixels
- Batería recargable Li-Ion de 7,2V

El conexionado entre los sensores y la tarjeta Eyebot es el siguiente:

- El servo de orientación de cámara se conecta a la salida de servos S11 (ver figura 2)
- Los sensores PSD se conectan al puerto infrarrojos PSD0 (sensor izquierdo), PSD1 (sensor frontal) PSD2 (sensor derecho)

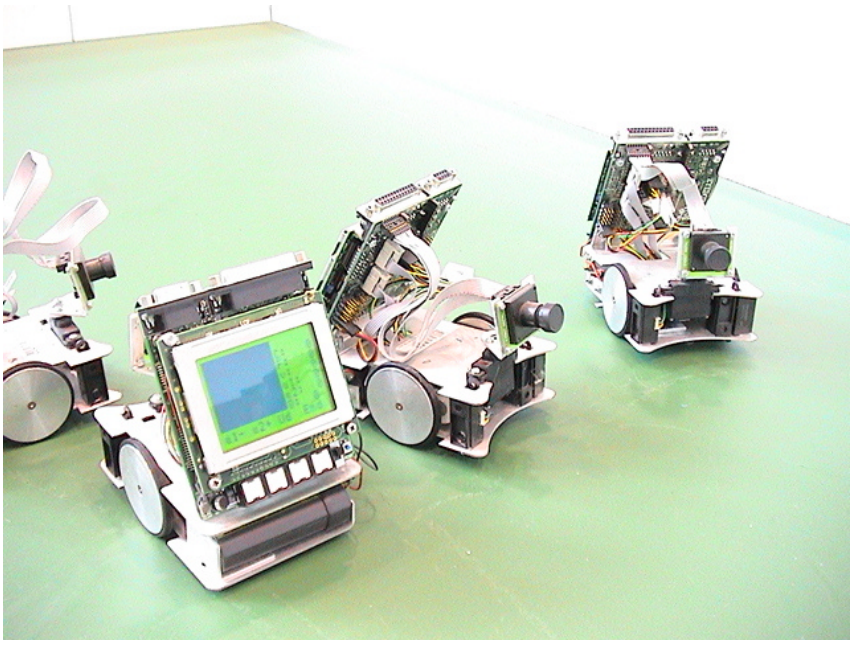


Figura 3. Robots SoccerBot disponibles en el departamento

Como primer ejercicio vamos a probar cada uno de los sensores y actuadores del robot utilizando las rutinas de test del propio sistema operativo ROBIOS.

2.4 Test del robot SoccerBot

1) Encenderemos el robot (interruptor situado al lado derecho de los botones (figura 2). La pantalla nos muestra la configuración del robot (figura 4.a). En la última fila tenemos el menú de opciones controlado por lo cuatro botones. Pulsaremos el botón Hrd. Se nos mostrará la siguiente pantalla (figura 4.b). Pulsaremos el botón HDT y accedermos a la configuración del hardware del robot (figura 4.c)

Figura 4. a) pantalla de inicio, b) configuración hardware c) HDT

2) En la pantalla se nos muestran los tipos de sensores y actuadores configurados. La tecla Nxt nos permite desplazarlos hacia abajo en la lista (el elemento seleccionado tiene el cursor a la derecha). La tecla + permite, sobre el tipo seleccionado, especificar un elemento concreto (por ejemplo el motor derecho o izquierdo). Por último la tecla Tst ejecuta el test. En la figura 5 se muestran varios ejemplos:

Figura 5. a) Test PSD, b) Test servomotor cámara c) Test motor DC

El test PSD proporciona la distancia al obstáculo calibrada en centímetros (D) y el valor que proporciona el sensor (R)
El test de servos y motores nos permiten accionar poco a poco cada uno de los motores.

Para salir de cada pantalla pulsaremos el botón END que nos permite regresar a la pantalla anterior.

2.5 Ejecutar un programa

Como segundo ejercicio ejecutaremos un programa de demostración ya almacenado en la memoria Flash-ROM:

1) Volveremos a la pantalla de inicio pulsando END repetidas veces (figura 6.a). Pulsaremos el botón Usr y accedemos a la gestor de programas (figura 6.b). Tenemos tres opciones Ld , Run y Rom . Para cargar un programa almacenado en la ROM pulsaremos el botón Rom, accediendo al listado de programas almacenados (figura 6.c)

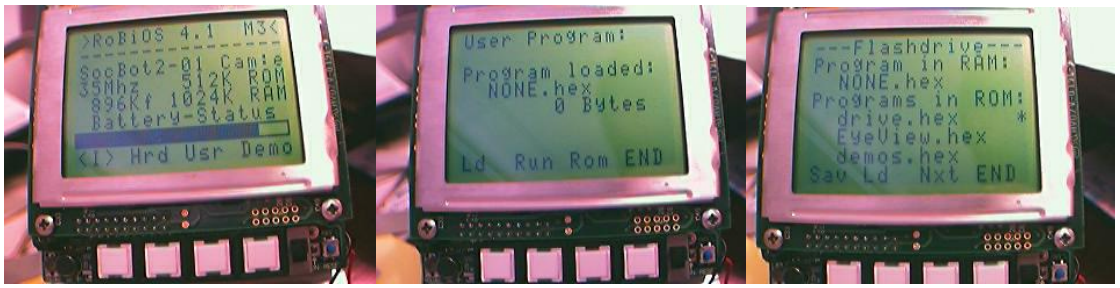


Figura 6. a) pantalla de inicio, b) gestor de programas c) programas en ROM

2) En la pantalla se nos muestran en la parte superior el programa que está actualmente en memoria RAM (solo podemos ejecutar el programa que esté en RAM). Debajo nos muestra los tres programas que pueden estar en la memoria no volátil FLASH-ROM (128K por programa). La tecla Nxt nos permite desplazarlos hacia abajo en la lista (el elemento seleccionado tiene el cursor a la derecha). Pulsaremos la tecla Nxt hasta que seleccionemos el programa "demos.hex". Una vez posicionado el cursor pulsaremos la tecla Ld

3) Pulsaremos la tecla END para volver al menú anterior (figura 7.a). Ejecutaremos el programa mediante el botón Run . El programa dispone de una serie de demos de todos los sensores y actuadores que podemos probar (figura 7.b) . El cursor se desplaza con los botones + y - . Ejecutaremos la demo de captura de video para probar la cámara pulsando Sel en la primera opción, y dentro de ella la opción Color. En la figura 7.c se muestra la imagen capturada umbralizada.

Figura 7. a) gestor de programas, b) demos c) demo cámara color

Como puede observarse las imágenes se pueden representar en pantalla pero solo en dos niveles de intensidad debido a las limitaciones de la misma, por lo que es difícil ajustar el enfoque de la cámara. En próximos apartados veremos cómo transferir las imágenes en color (figura 8) al PC mediante el puerto serie lo que nos permitirá diseñar los algoritmos de visión de forma cómoda sobre un PC. Pero primero debemos aprender a compilar y descargar programas en el robot.

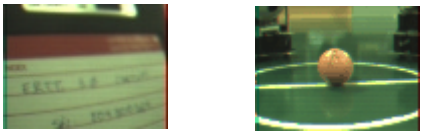


Figura 8. Imágenes capturadas con la cámara EyeCam y transferidas al PC mediante el programa EyeView

2.6 Programación sobre ROBIOS

La programación de la tarjeta Eyebot se pude realizar en varios lenguajes: ensamblador, C y C++. En nuestro caso utilizaremos un compilador de ANSI-C cruzado. El término ***cruzado*** se refiere a que se utilizará una plataforma diferente para realizar la compilación (PC-x86) a la del código ejecutable final (68332).

Adicionalmente al lenguaje C estándar, utilizaremos un Sistema Operativo que está almacenado en la Flash-ROM del robot denominado ***ROBIOS***.

ROBIOS es un sistema operativo de tiempo real para sistemas empotrados que ocupa apenas 128 Kb. Está orientado al control de robots por lo que incorpora el manejo de las unidades de entrada salida disponibles en el Motorota 68332 así como los drivers para los sensores y actuadores utilizados en el robot. El entorno es fácilmente ampliable mediante el fichero de descripción de hardware HDT, incluso se pueden añadir nuevas llamadas (funciones de bajo nivel) al sistema.

El uso de los recursos básicos se realiza mediante llamas al sistema de forma parecida al S.O Unix. Para ello basta con compilar incorporando la cabecera ***eyebot.h*** y enlazar la librería ***robios***. Realmente no tenemos que preocuparnos de estos detalles ya que se proporcionan varios ficheros '**bat'** que automatizan todo el proceso de compilación.

Adicionalmente existen librerías de código C generadas en diferentes universidades que utilizan este mismo robot y que podemos incorporar a nuestra aplicación. Destacar entre ellas las librerías de procesamiento de imágenes.

El entorno de desarrollo está formado por los siguientes módulos:

- ***Compilador GNU y utilidades:*** se trata de herramientas en línea de comandos (MS-DOS) que incluyen los compiladores de C y C++. Así mismo se dispone de programas para transferencia de la aplicación mediante puerto serie. Se ejecuta en un PC bajo MS-DOS, Windows 95/98/2000/XP o Linux

- ***Sistema operativo para el microcontrolador ROBIOS:*** Gestiona todo el hardware de la tarjeta e implementa un planificador multitarea. Dispone así mismo de un programa de configuración, test y descarga por el puerto serie manejado mediante la pantalla y los botones. Está almacenado en la memoria Flash del Robot pudiendo ser actualizado fácilmente.

- ***Simulador del Robot EyeSim:*** se dispone de un simulador completo de la tarjeta Eyebot en el que se pueden definir diferentes robots y escenarios. Permite compilar los mismos programas que se ejecutan en el robot y probarlos offline, disponiendo de una simulación de los actuadotes y los sensores (incluida la cámara). Está disponible para Windows y Linux.

3 UTILIZACIÓN DEL ENTORNO DE PROGRAMACIÓN

Se presenta en este apartado una introducción al manejo del entorno de programación de la tarjeta Eyebot utilizando el sistema operativo ROBIOS. Se recomienda estudiar la documentación existente en el servidor web así como los ejemplos disponibles.

3.1 Instalando el software.

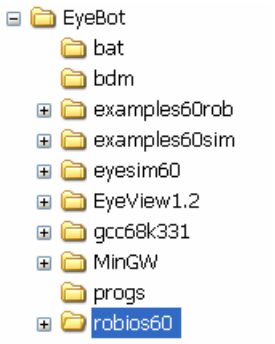
En caso de no tener instalado el software en el PC utilizado debemos ejecutar los siguientes pasos:

1) Mediante un navegador accederemos a la página web del Eyebot (http://lorca.umh.es/isa/es/temas/eyebot). En la página principal en el apartado Instalación de software encontraremos un fichero comprimido (***rob60win.exe***) que contiene el software con un instalador. Descargaremos el fichero en un directorio temporal y lo ejecutaremos siguiendo las instrucciones del instalador.

2) Nos habrá creado una carpeta (***c:\Eyebot***) y dos accesos directos a las consolas de compilación y simulación (*Robios prompt*, *Eyesim prompt*). Estos enlaces arrancan una consola en modo texto con las variables de entorno debidamente configuradas.

3) Desde la misma página, descargaremos e instalaremos el programa ***EyeView12.zip*** en la carpeta ***c:\Eyebot\EyeView1.2***. Este programa permite transmitir y visualizar las imágenes de la cámara del robot en un PC.

4) Ejecutaremos el programa ***c:\Eyebot\MinGW-3.1.0.exe***. Realizará la instalación del compilador gcc para compilar las simulaciones en el PC. Al finalizar debemos tener una estructura de directorios similar a la mostrada a continuación:



5) En el directorio ***bat*** se encuentran los ficheros de arranque de las consolas, ***examples60rob*** contiene ejemplos compilados para el robot, ***examples60sim*** contiene ejemplos compilados para ejecutar con el simulador en el PC, ***eyesim60*** contiene el simulador, en ***gcc68k331*** se encuentra el compilador cruzado completo para el 68332. En el directorio ***robios60*** están todas las librerías, comandos y ficheros HDT. Cabe destacar el contenido de dos subdirectorios:

c:\Eyebot\robios60\cmd -> en el se encuentran los ficheros '.bat' para compilar y transferir los programas

c:\Eyebot\robios60\doc -> en el se encuentran la documentación de ROBIOS

6) Por último crearemos el subdirectorio ***c:\Eyebot\progs***, en el ubicaremos nuestros programas. Podemos descargar de la página web los programas de ejemplo de estos talleres (***ejemplos.zip***) e instalarlos en esta carpeta. Se recomienda instalar un buen editor para programación (en la misma página web se puede descargar ***Crimson Editor*** que es gratuito).

Nota*: la instalación elige la carpeta c: de forma fija. Si queremos instalarlo en otro disco o directorio debemos moverlo manualmente y editar 9 ficheros .bat para indicar la nueva ubicación (start-sim.bat, start-rob.bat, versions.bat, run-rob.bat, run-sim.bat, gccsim.bat, gcc68.bat, gcchdt.bat, specs) También deberemos cambiar los enlaces de los dos iconos y la asociación de ficheros de windows (.hex .hx .sim)*

3.2 Compilando una aplicación:

Para probar cada una de las etapas que se irán describiendo vamos a crear un programa denominado **ejemplo1.c**:

Nota: se supone que se conoce la programación en C

Para editar el programa podemos utilizar cualquier editor de textos pero recomendamos utilizar un buen editor de programación como **CrimsomEditor** o **EditPlus** (instalado en los PCs del laboratorio). Este editor nos permite también automatizar los procesos de compilación y descarga desde el propio programa.

El código que debemos teclear es el siguiente:

```
/* Ejemplo1.c */

#include <eyebot.h>

#define MENSAJE "Hola Mundo\n"

int main()
{
       LCDPutString(MENSAJE);
       return 0;
}
```

Este programa es muy sencillo y se limita a imprimir en la pantalla del Eyebot el mensaje *"Hola mundo"*.

Para compilar un programa abriremos en primer lugar un ventana de la consola MS-DOS mediante el enlace  (***Robios prompt)***. En esta consola realizaremos todas las operaciones:

1) Nos moveremos al directorio donde esté el programa : *cd c:/Eyebot/progs*

2) Si queremos compilar el fichero *ejemplo1.c* ejecutaremos el comando
c:> gcc68 ejemplo1.c –o ejemplo1.hex

3) Si el código no contiene errores habrá generado el fichero ***ejemplo1.hex*** que es el código ejecutable que deberemos descargar en el robot.

Si se echa un vistazo al subdirectorio *c:\Eyebot\robios60\cmd\Windows* se puede observar que existen más alternativas para compilar y enlazar programas:

gcc68 -> compila programas de uno o varios ficheros, admite todas las opciones y modificadores del compilador gcc
g++68 -> compila código c++ (realmente es una llamada al anterior)
gas68 -> compila código ensamblador
gcchdt -> compila los ficheros de configuración del hardware HDT
dl -> descargar el código en el robot (probablemente deba configurar el puerto serie editando el fichero)
srec2bin -> permite convertir a formato binario y comprimir los ficheros .hex

Si nuestra aplicación está constituida por varios ficheros es recomendable el uso de ficheros de automatización de la compilación de tipo ***makefile***. Estos ficheros se compilan con el comando ***make*** y la sintaxis es la misma que en UNIX. Se proporcionan ejemplos en el subdirectorio *examples60rob*.

3.3 Descargando el programa

Para descargar un programa al robot conectaremos el cable serie (figura 9) desde el robot a un puerto del PC (debemos conocer si es el puerto serie 1 o 2)

En el robot ejecutaremos lo siguiente:

1) Volveremos a la pantalla de inicio pulsando END repetidas veces (figura 8.a). Pulsaremos el botón Usr y accedemos a la gestor de programas (figura 8.b). Tenemos tres opciones Ld, Run y Rom. Para descargar un programa por el puerto serie en memoria RAM pulsaremos el botón Ld, (figura 8.c)

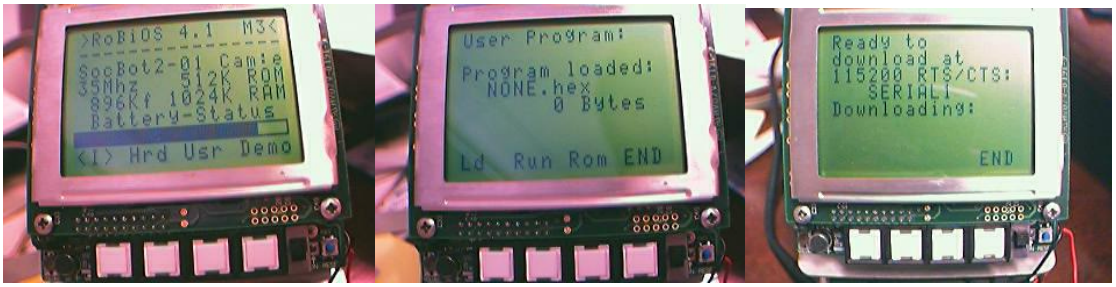


Figura 8. a) pantalla de inicio, b) gestor de programas c) descarga

2) En la pantalla se nos muestra la configuración del puerto serie. En este momento el robot queda esperando la transferencia del programa desde el PC. Si quisiéramos cancelar la transmisión basta con pulsar la tecla END

3) En este punto podemos tener problemas de transmisión, en tal caso deberemos repetir todo el proceso. Si persistieran los problemas podemos reducir la velocidad de comunicación. Si todo el proceso es correcto, tendremos cargado en memoria RAM el programa. Ejecutaremos el programa mediante el botón Run.

4) El programa en RAM se perderá al apagar el robot por lo que si queremos conservarlo debemos almacenar en la memoria Flash-ROM. Para ello basta con pulsar la tecla Rom, seleccionar el slot a usar de la memoria Flash, y almacenarlo pulsando Sav (figura 6.c)

En el PC ejecutaremos los siguiente:

1) En la consola cambiaremos al directorio donde esté el programa : *cd c:/Eyebot/progs*

2) Ejecutaremos el comando: *dl ejemplo1.hex*

Si todo es correcto se habrá transferido el programa.

Nota: desde el PC no es posible tener confirmación de que la transferencia fue correcta. Debemos comprobarlo en la pantalla del robot

Nota: *si queremos que el programa se ejecute automáticamente al encender el robot debemos almacenarlo en Flash-ROM con el nombre* ***startup.hex***.

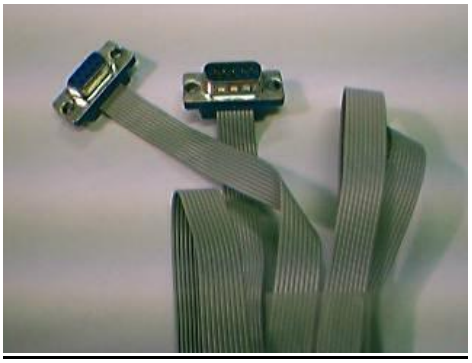


Figura 9. cable serie

Comentarios sobre el programa de ejemplo:

- Aunque se puede utilizar la librería estándar de entrada salida para manejar la pantalla <*stdio.h*>, no es recomendable ya que incrementa de forma considerable el tamaño de los programas. Es preferible utilizar la funciones específicas de ROBIOS que se describen de forma detallada en la documentación anexa.

- La función LCDPutString() imprime una cadena por la pantalla.

- Al terminar el programa nos muestra el tiempo consumido en la ejecución.

Experimentación:

- Se recomienda experimentar con otras funciones como la lectura de las cuatro teclas disponibles mediante las funciones:

  LCDMenu();

  KEYGet();

  KEYRead();

3.4 Problemas de transferencia

Se han detectado casos de problemas de transferencia con Windows 2000,XP debido a las diferencias en el puerto serie de los diferentes PCs. Si experimentas de forma continuada estos problemas debes hacer lo siguiente:

1) Utilizar ficheros de programas ASCII en lugar de ficheros comprimidos:
   Cuando se realiza la compilación podemos usar el comando gcc68 que crea un fichero ASCII o aplicar sobre el fichero resultado el comando ***srec2bin*** que crea un fichero binario comprimido (extensión .hx).

2) Reducir la velocidad de transferencia por defecto 115.200 bps. Para ello debemos hacer lo siguiente:

- En el PC debemos utilizar editar el comando ***dl*** cambiando la velocidad de transferencia a 57600 bps. Si queremos, podemos utilizar directamente el comando ***transhex*** indicando todos los parámetros:

```
TransHex 1.5
(c) 1999-2000 Michael Kasper, Michael Kapp, University of Kaiserslautern
Usage: TransHex [/C<Port>] [/B<Baud>|/D<Divisor>]
          [/IRMATE] [/NOHANDSHAKE] <File>
Defaults: /C1 /B115200
```

En el robot configuraremos la velocidad de la siguiente forma:

1) Volveremos a la pantalla de inicio pulsando END repetidas veces (figura 10.a). Pulsaremos el botón Hdr y accedemos la configuración hardware (figura 10.b). Pulsaremos el botón Set apareciéndonos la pantalla de selección, pulsaremos el botón Ser, para configurar el puerto serie (figura 10.c).

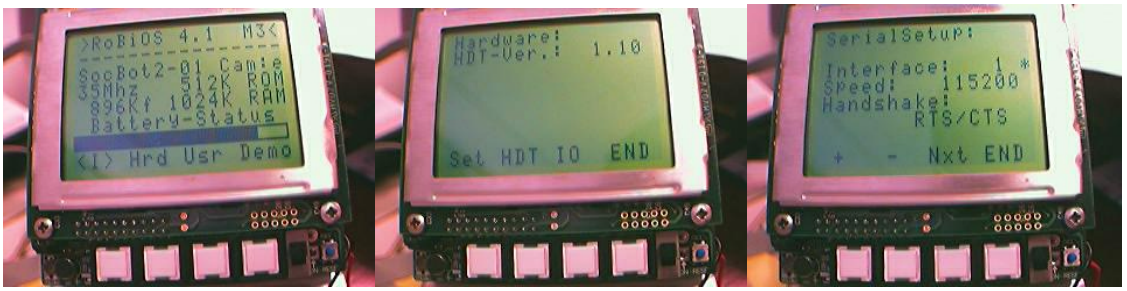


Figura 10. a) pantalla de inicio, b) gestor de programas c) descarga

2) En la pantalla se nos muestra la configuración del puerto serie. Utilizaremos el método habitual de desplazar el cursor sobre la opción deseada pulsando la tecla Nxt. Podemos cambiar la configuración del parámetro seleccionado con las teclas + y -. Una vez terminado, pulsando la tecla END retrocederemos en el árbol de menús.

Nota:

La configuración del robot volverá al valor inicial al apagar el interruptor. Si queremos mantenerla de forma permanente deberemos recompilar el fichero HDT de configuración de hardware y actualizar la Flash-ROM. (No se debe modificar el fichero HDT sin estudiar previamente su estructura, un error puede bloquear al sistema operativo).

3.5 El segundo programa en C

Para familiarizarnos con el proceso descrito anteriormente vamos a crear un segundo programa denominado ***ejemplo2.c***, en el que utilizaremos funciones adicionales del S.O. ROBIOS.

El código que debemos teclear es el siguiente:

```
/* Ejemplo2.c */

#include "eyebot.h"

int main ()
{
    int k;

    LCDClear();
    LCDMenu(" 1 "," 2 "," 3 "," 4 ");

    LCDPrintf("Hola soy: \n %s no. %d\n",
              OSMachineName(), OSMachineID() );
    LCDPrintf(" Version: %s\n %.3f MHz\n",
              OSVersion(),OSMachineSpeed()*1e-6 );
    LCDPrintf(" Tipo: %d\n",OSMachineType());
    k = KEYGet();
    LCDPrintf("Tecla presionada: %d !\n", k);

    return 0;
}
```

Este programa obtiene la información del sistema a través de las funciones proporcionadas por ROBIOS. También se muestra la lectura del teclado. Para compilarlo invoca el comando *`gcc68 ejemplo2.c –o ejemplo2.hex`*, debe generar un fichero llamado ***ejemplo2.hex***, que es el ejecutable correspondiente. Para descargarlo en el Eyebot conéctalo al puerto serie, ponlo a recibir el fichero y en el PC ejecuta el comando *`dl ejemplo2.hex`*.

En el siguiente apartado se presenta de forma general las posibilidades de la librería ROBIOS, se recomienda repasar la documentación adicional presente en el servidor web.

En las prácticas siguientes se irán estudiando aspectos concretos de esta librería. En el ejemplo se muestra el uso de las siguientes funciones:

`LCDClear()->` Borra la pantalla
`LCDMenu()->` Muestra la barra del menú indicando la función de los botones.

`LCDPrintf()->` similar al *printf()* de la librería estándar pero precisa menos memoria.

`OSMachineName(), OSMachineID() OSVersion(),OSMachineSpeed()`

➔ Obtiene información del fichero HDT

`KEYGet()->` Espera la pulsación de un botón y devuelve el número asociado

3.6 La librería ROBIOS

La librería ROBIOS implementada en la memoria Flash-ROM incorpora funciones para el manejo básico de los sensores, actuadores y los procesos que constituyan nuestra aplicación. Su uso consume muy poca memoria en nuestro programa ya que su código ya está almacenado en el S.O.

En el servidor web y en los ejemplos de la instalación se pueden encontrar así mismo librerías externas que pueden ser enlazadas en nuestra aplicación

En el documento anexo (***library.pdf***) se describe cada una de las funciones disponibles. ROBIOS proporciona recursos para el manejo de :

- Procesamiento de imágenes
- Entrada desde el teclado
- Salida por pantalla
- Captura y procesamiento de imágenes desde la cámara en color
- Funciones del sistema
- Multitarea
- Intercomunicación entre procesos (semáforos)
- Temporizadores
- Comunicación por puerto serie
- Lectura de audio
- Sensores Infrarrojos PSD
- Servos y Motores DC
- API de control de velocidad y posición
- Bumpers, sensores infrarrojos
- Entradas /salidas digitales
- Puerto paralelo
- Entrada analógica
- Comunicación vía radio
- Comunicación mediante mandos remotos infrarrojos.

Notas:

NO debe utilizarse la función ***exit()*** ya que bloquea el S.O.

Códigos de retorno de las funciones:

Por defecto (si no se indica lo contrario) las funciones devuelven 0 si es correcto y un valor no nulo si ha habido algún error. Otros valores posibles están indicados en la documentación de las funciones.

Uso básico de los recursos del robot:

Para utilizar cualquier recurso (motor, sensor,…) es preciso en primer lugar inicializarlo. La función de inicialización es del tipo "***xxxInit()***", donde "xxx" depende del tipo de dispositivo. En caso de que puedan existir varios dispositivos del mismo tipo hay que indicar en la función de inicialización a cual nos referimos, es el parámetro llamado "***semantics***". El nombre de cada dispositivo está fijado en el fichero HDT mediante una constante (se resumen a continuación). La función de inicialización nos devuelve un manejador (básicamente un entero) que será usado en las funciones de lectura o escritura sobre dicho recurso.

Por último cuando ya no es necesario, se debe liberar mediante la función "***xxxRelease()***". Esta función puede tener o no parámetros para indicar qué dispositivo deseamos liberar. Veamos un ejemplo:

```
/* psd.c */
#include "eyebot.h"

int main ()
{
     int val;
    PSDHandle psd_handle;

     LCDMode(SCROLLING|NOCURSOR);
     LCDClear();
     LCDMenu("","","","END");

    psd_handle = PSDInit(PSD_FRONT); // Inicializa el sensor PSD frontal

    if(!psd_handle)
    {
       // no olvidar la comprobación de errores
       OSError("\nPSDInit Error!\n", psd_handle, FALSE);
      return(-1);
    }

    //  Funciones de lectura del sensor
    PSDStart(psd_handle, TRUE);  // pone en marcha la lectura continua

    while (KEYRead() != KEY4)
    {
       val=PSDGet(psd_handle);    // lee el valor actual
       LCDSetPos (3, 0);   // posición en pantalla (fila 3, col. 1)
       LCDPrintf("PSD_FRONT: %3d", val);       // muestra el valor por pantalla
    }
    PSDStop();                   // para los sensores PSD

    //  Libera el recurso
    PSDRelease();

    return 0;
}
```

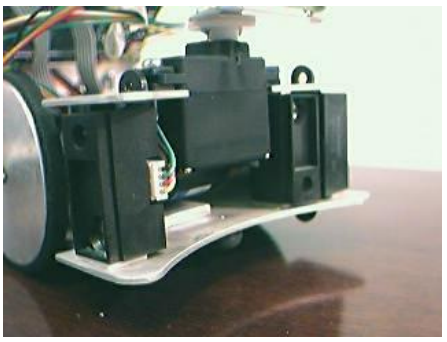
Cualquier dispositivo que utilicemos requerirá un proceso similar aunque las funciones y los parámetros utilizados siempre son específicos y debemos consultar la documentación.

Nota:
Otra característica importante es que el parámetro del manejador pasado a algunas funciones pude ser ***compuesto***, es decir combinación de varios. Esto es útil cuando queremos aplicar el mismo comando a varios dispositivos simultáneamente. Para ello basta con combinar todos los manejadores con el operado OR de bits ("|")

En la tabla siguiente se resumen las constantes asociadas a los diferentes dispositivos del robot (parámetro "*semantics*") y los tipos de datos para los manejadores. Estas constantes son incluidas automáticamente al incorporar la cabecera "***eyebot.h***"

```
/*****************************************************************/
/*  semantic-constants, give the semantic of an HDT entry        */
/*****************************************************************/

#define     MOTOR_RIGHT          -100
#define     MOTOR_LEFT           -101
#define     VW_DRIVE             -700
#define     PSD_FRONT            -200
#define     PSD_LEFT             -205
#define     PSD_RIGHT            -210
#define     QUAD_LEFT            -400
#define     QUAD_RIGHT           -401
#define     SERVO11              -361

typedef int MotorHandle;
typedef int ServoHandle;
typedef int PSDHandle;
typedef int QuadHandle;
typedef int BumpHandle;
typedef int IRHandle;
typedef int VWHandle;
```

3.6.1 Sensores PSD



Los sensores PSD son sensores de distancia infrarrojos. La medida se realiza por triangulación y permite un rango de trabajo entre los 60-600 mm. El valor de distancia ledido del sensor (*RAW*) se calibra mediante tablas de transformación configuradas en el fichero HDT. La activación y transmisión digital de la distancia se puede realizar de manera continua (ejemplo anterior), o realizar una medida única. En el segundo caso se dispone de funciones para esperar a que termine la medida y obtener el tiempo en el que fue realizada.

A continuación se propone el siguiente ejemplo más complejo de uso de los sensores PSD realizando medida no continua (***Nota***: el simulador no permite este tipo de lectura):

```
/* pds_s.c */

#include "eyebot.h"

int main ()
{
       char    z;
       PSDHandle psd_handle;

       LCDPrintf("    PSD-Demo    ");
       LCDPrintf("----------------");

      /* Init the FRONT-PSD */
       psd_handle = PSDInit(PSD_FRONT);
       if(psd_handle == 0)
       {
              LCDPrintf("\nPSDInit Error!\n");
              OSWait(200);
              return -1;
       }
       LCDMenu("SHOT","","","END");

       while (1)
       {
           z = KEYRead();
           switch (z)
           {
              case KEY1:
                 /* Start PSD-measuring for single shot */
                 if(PSDStart(psd_handle, FALSE))
                     LCDPrintf("\nPSD busy!\n");

                 /* wait for the result */
                 while(!PSDCheck()){};

                 /* print timestamp, distance, raw-value */
                 LCDSetPos(4,0);
                 LCDPrintf("T:     %5d\nD:     %3d     R:     %3d",     PSDGet(0),
PSDGet(psd_handle), PSDGetRaw(psd_handle));
                 break;

              case KEY4:
                     /* release the used PSDs */
                     PSDRelease();
                      return 0;

              default:
                     break;
              }
       }
}
```

3.7 El simulador EyeSim

La programación de aplicaciones mínimamente complejas exige la depuración de errores. Si ha esto se le une la complejidad de la descarga en memoria y los problemas que siempre supone el uso de sensores y accionamiento físicos hace esta tarea laboriosa.

Se dispone de una aplicación de simulación en PC (Windows y Linux) llamada ***EyeSim*** que permite probar las aplicaciones antes de descargarlas en el robot. El simulador incluye una completa representación del entorno 2.5D con obstáculos, así como la cinemática del robot y todos sus sensores incluida la cámara. A los datos sintéticos generados por el simulador se la pueden añadir ruido para comprobar la aplicación de forma más realista. Permite así mismo la ejecución concurrente de varias aplicaciones en diferentes robots dentro del mismo escenario.

La figura siguiente muestra la pantalla del simulador ejecutando una aplicación.

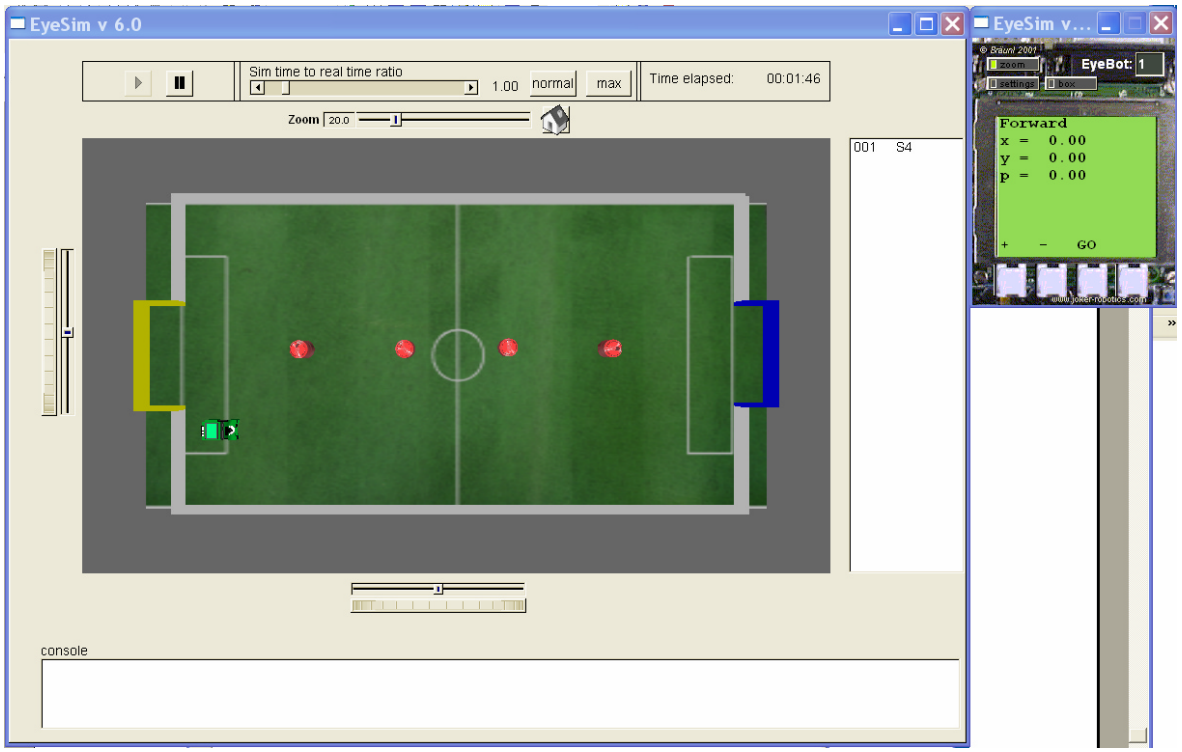


Figura 11. Simulador Eyesim 6

En la pantalla se muestra el escenario con el robot y los obstáculos (Todos los elementos son configurables mediante ficheros). A la derecha se muestra la pantalla del robot con sus botones.

Justo encima de esta pantalla tenemos tres botones adicionales que nos permiten configurar el simulador. El botón ***Settings*** nos da acceso a la ventana de configuración (figura 12). Dispone de tres pestañas, la primera da acceso a la información del robot y escenarios usados los parámetros así como a la configuración de la ubicación del robot (posición y orientación), y la opción de trazar el desplazamiento en pantalla (*trail*).

La segunda pestaña nos permute activar en pantalla la línea de lectura de los sensores PSD.

La tercera pestaña da acceso a la configuración de los parámetros de ruido del simulador.

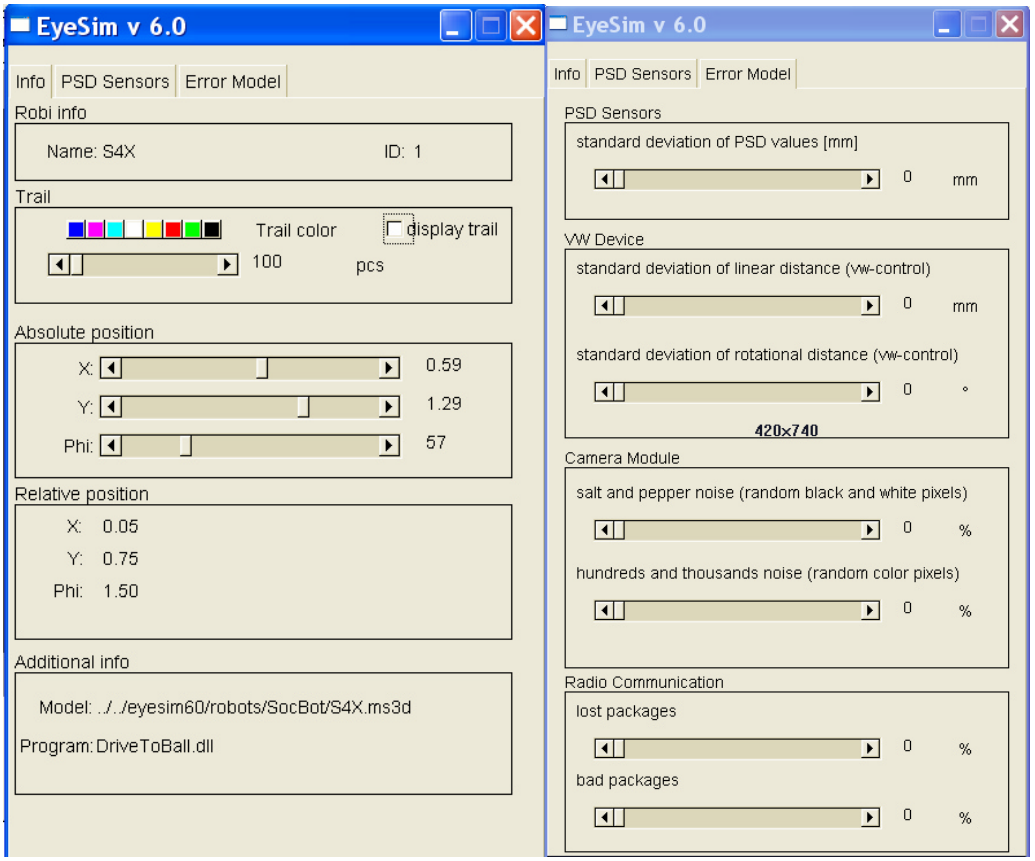


Figura 12. Ventana de configuración del simulador

3.7.1 Compilando la aplicación para el simulador

Para poder ejecutar la aplicación en simulación es necesario compilarla como una ***dll*** con un compilador para plataforma Windows. Cualquier compilador ANSI es válido pero en la instalación se incluye MinGW (Minimal GNU C Compiler) que habremos instalado previamente.

Para compilar la aplicación en modo simulador abriremos en primer lugar un ventana de la consola MS-DOS mediante el enlace  (***Eyesim prompt)***. En esta consola realizaremos todas las operaciones:

1) Nos moveremos al directorio donde esté el programa : *cd c:/Eyebot/progs*

2) Si queremos compilar el fichero *ejemplo2.c* ejecutaremos el comando
c:> gccsim ejemplo2.c –o ejemplo2.dll

3) Si el código no contiene errores habrá generado una librería dinámica de Windows ***ejemplo2.dll*** que es el código ejecutable que deberemos utilizar en el simulador.

Para poder ejecutar la aplicación en simulación es necesario crear adicionalmente varios ficheros que indique la información del robot, el entorno, aspecto gráfico y datos iniciales:

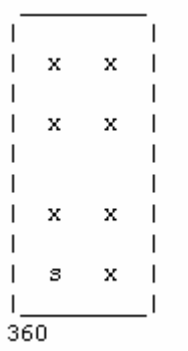
1) Fichero de descripción de la simulación: ***ejemplo2.sim***

Se trata del fichero principal que contiene los enlaces la resto de ficheros utilizados. A continuación se muestra un ejemplo:

```
define sim d:/Robots/Eyebot/eyesim60

# world description file (either one maze one world)
#world %sim%/worlds/worlds/Soccer1999.wld
maze   %sim%/worlds/mazes/maze1.maz

# additional obstacles: x,y,theta, friction parameter, image file
#object  100 100   0  0.0 %sim%/worlds/objects/coke-can/can.ms3d

# robi description file, programa.dll, [posx posy theta], [graphics file]
robi   %sim%/robots/SocBot/SocBot.robi  ejemplo2.dll  0 0 0
```

La instrucción ***define*** permite declarar constantes (en este caso se utiliza para indicar el directorio donde están ubicados los ficheros de datos)

Para indicar el escenario tenemos dos opciones que corresponden a dos formatos diferentes (ver ejemplos):



world: el entorno se crea mediante coordenadas de líneas o rectángulos. Permite crear escenarios complejos en cualquier orientación, y utilizar texturas para su renderizado 3D.

maze: utiliza un formato ASCII gráfico sencillo mediante los caracteres '|' , '_' (solo permite paredes ortogonales). Tiene la posibilidad de indicar uno o varios robots con su orientación (letras s,u,d,l,r), bolas (o) y obstáculos móviles (x), así como las dimensiones de la celda básica.

La instrucción ***object***, cuando utilizamos un entrono definido mediante el comando world, permite añadir objetos 3D diseñados mediante OpenInventor.

Por último el comando ***robi*** debe indicar el robot (o los robots) que se utilizaran para la simulación junto al programa a ejecutar. Como parámetros se debe especificar un fichero de configuración del robot, el programa y la ubicación y orientación inicial.

2) Fichero de descripción del robot: ***socbot.robi***

Describe el modelo cinemática y los sensores del robot: tamaño físico, velocidades máximas, descripción gráfica, sensores PSD disponibles, parámetros de la cámara.

3) Fichero de descripción gráfica 3D del robot: ***socbot.ms3d***

4) Fichero de descripción del entorno: ***maze1.maz*** o ***soccer1999.wld***

Disponemos de varios ficheros de datos disponibles en el directorio del simulador.

3.7.2 Ejecutando la aplicación en el simulador

Para ejecutar la aplicación basta con teclear el siguiente comando en la consola abierta anteriormente:

c:> eyesim ejemplo2.sim

También podemos realizar un doble click del ratón sobre el fichero ***.sim*** para ejecutarlo.

El simulador no permite el accionamiento directo de los motores ni de los encoders, es preciso utilizar el controlador ***V-W*** (esto no es una limitación, ya que dada la fuerte inercia del robot, y el no muy elevado par de los motores, su uso sin este controlador es muy inestable). Tampoco está disponible el uso del servomotor de la cámara.

Como ejemplo se propone ejecutar la simulación de ***wall.sim*** disponible en el fichero de ejemplos (descargable desde la página web).