Universidad Politécnica de Madrid
Departamento de Arquitectura y Tecnología de Sistemas Informáticos

FACULTAD DE INFORMÁTICA

UPM-DATSI

Práctica DSCD

Control de un robot industrial

Francisco Manuel SÁNCHEZ MORENO
Juán ZAMORANO FLORES

Resumen
En el presente documento se describe la práctica a realizar para la asignatura de Diseño de Sistemas de Control Discreto. Versiones actualizadas de este documento se podrán encontrar en la página web de la asignatura (http://laurel.datsi.fi.upm.es/~ssoo/DSCD/)

Palabras clave
Tiempo Real, Control de Procesos

Octubre 2002

F.I. ✍Campus de Montegancedo ✍ Boadilla del Monte ✍28660 Madrid ✍ESPAÑA ✍Tel: + 34 91 3367387 ✍Fax: + 34 91 3367373 ✍http://www.datsi.fi.upm.es

Indice de Contenidos

1 Introducción

1.1 Objetivo

El objetivo de esta práctica es el desarrollo completo de una aplicación de control mediante computador. Para ello se empleará el sistema operativo RTEMS [1] aplicando conceptos fundamentales de tiempo real (ingeniería del software), control de procesos y diseño con microcontroladores.

El presente documento describe el sistema físico a controlar, las diversas tareas prácticas a llevar a cabo por los alumnos y la memoria que estos deben entregar.

1.2 Evaluación

Tanto los trabajos prácticos, como la memoria final serán realizados por grupos de dos, a menos que algún alumno desee explícitamente realizar el trabajo individualmente. Estos trabajos son explicados en el punto 3 del presente documento. Al final de curso se podría plantear la posibilidad de poner un examen teórico para evaluar los conocimientos adquiridos por los alumnos. En el caso de haberlo, la forma de evaluación será 50% la nota de la práctica y 50% la nota del examen. Cuando la práctica esté resuelta, los alumnos deberán quedar con cualquiera de los profesores de la asignatura para mostrarle su funcionamiento así como explicarle el proceso que se ha seguido para su consecución. La evaluación de la práctica tendrá en cuenta la exposición ante el profesor y la calidad de la memoria entregada.

1.3 Lugar de trabajo

Se han instalado dos puestos de trabajo en la sala canalobre [2] situada en el sótano del edificio 4. Cada grupo se le reservan cuatro horas semanales en las que tendrá preferencia, pero podrá acceder a su puesto cuando esté libre. El único requisito es que se identifique con el carnet de la universidad ante los encargados del centro de cálculo, que tienen un listado con los alumnos matriculados. Estos serán los que abran la puerta dentro de sus horarios laborales.

Dada que este aula no cuenta con vigilancia, los alumnos serán responsables de los equipos. Se ruega extremar las precauciones, y **cerrar la puerta con llave siempre que la sala quede sola**, aunque la ausencia sea momentánea.

? **¡ATENCIÓN! Cerrar la puerta con llave siempre que la sala quede sola, aunque la ausencia sea momentánea**

1.4 Bibliografía

En principio no se fijan prerequisitos, pero es aconsejable haber cursado alguna de las siguientes asignaturas; Control de Procesos, Sistemas de Tiempo Real y Diseño con Microcontroladores. Al no tener esta asignatura contenido teórico la bibliografía es la que se recomiende emplear en estas asignaturas [9] [10] [11] [12] y [13]. Adicionalmente, son necesarios los manuales de los productos Software y Hardware empleados; Sistema operativo Linux (Red Hat 6.2) [3] , sistema operativo de tiempo real RTEMS [1], Robot Industrial Fischer [4]

2 Introducción

2.1 Objetivo

El objetivo de esta práctica es el desarrollo completo de una aplicación de control mediante computador. Para ello se empleará el sistema operativo RTEMS [1] aplicando conceptos fundamentales de tiempo real (ingeniería del software), control de procesos y diseño con microcontroladores.

El presente documento describe el sistema físico a controlar, las diversas tareas prácticas a llevar a cabo por los alumnos y la memoria que estos deben entregar.

2.2 Evaluación

Tanto los trabajos prácticos, como la memoria final serán realizados por grupos de dos, a menos que algún alumno desee explícitamente realizar el trabajo individualmente. Estos trabajos son explicados en el punto 3 del presente documento. Al final de curso se podría plantear la posibilidad de poner un examen teórico para evaluar los conocimientos adquiridos por los alumnos. En el caso de haberlo, la forma de evaluación será 50% la nota de la práctica y 50% la nota del examen. Cuando la práctica esté resuelta, los alumnos deberán quedar con cualquiera de los profesores de la asignatura para mostrarle su funcionamiento así como explicarle el proceso que se ha seguido para su consecución. La evaluación de la práctica tendrá en cuenta la exposición ante el profesor y la calidad de la memoria entregada.

2.3 Lugar de trabajo

Se han instalado dos puestos de trabajo en la sala canalobre [2] situada en el sótano del edificio 4. Cada grupo se le reservan cuatro horas semanales en las que tendrá preferencia, pero podrá acceder a su puesto cuando esté libre. El único requisito es que se identifique con el carnet de la universidad ante los encargados del centro de cálculo, que tienen un listado con los alumnos matriculados. Estos serán los que abran la puerta dentro de sus horarios laborales.

Dada que este aula no cuenta con vigilancia, los alumnos serán responsables de los equipos. Se ruega extremar las precauciones, y **cerrar la puerta con llave siempre que la sala quede sola**, aunque la ausencia sea momentánea.

? **¡ATENCIÓN! Cerrar la puerta con llave siempre que la sala quede sola, aunque la ausencia sea momentánea**

2.4 Bibliografía

En principio no se fijan prerequisitos, pero es aconsejable haber cursado alguna de las siguientes asignaturas; Control de Procesos, Sistemas de Tiempo Real y Diseño con Microcontroladores. Al no tener esta asignatura contenido teórico la bibliografía es la que se recomiende emplear en estas asignaturas [9] [10] [11] [12] y [13]. Adicionalmente, son necesarios los manuales de los productos Software y Hardware empleados; Sistema operativo Linux (Red Hat 6.2) [3] , sistema operativo de tiempo real RTEMS [1], Robot Industrial Fischer [4]

3 Sistema físico

3.1 Puesto de trabajo

En la figura 1 se reproduce esquemáticamente uno de los puestos de práctica, que está compuesto por los siguientes elementos:

- Computador de desarrollo con sistema operativo Linux.
- Computador de ejecución (*target*) donde se ejecutará la aplicación a desarrollar.
- Robot Industrial *Fischer* y electrónica asociada

Se aconseja **extremar las precauciones** ya que en la electrónica de potencia **hay 220v** y si se manipula indebidamente se puede producir una descarga al alumno o una avería del equipo.

? **En el circuito de control del robot hay 220 voltios. ¡PELIGRO! de descarga o cortocircuito**

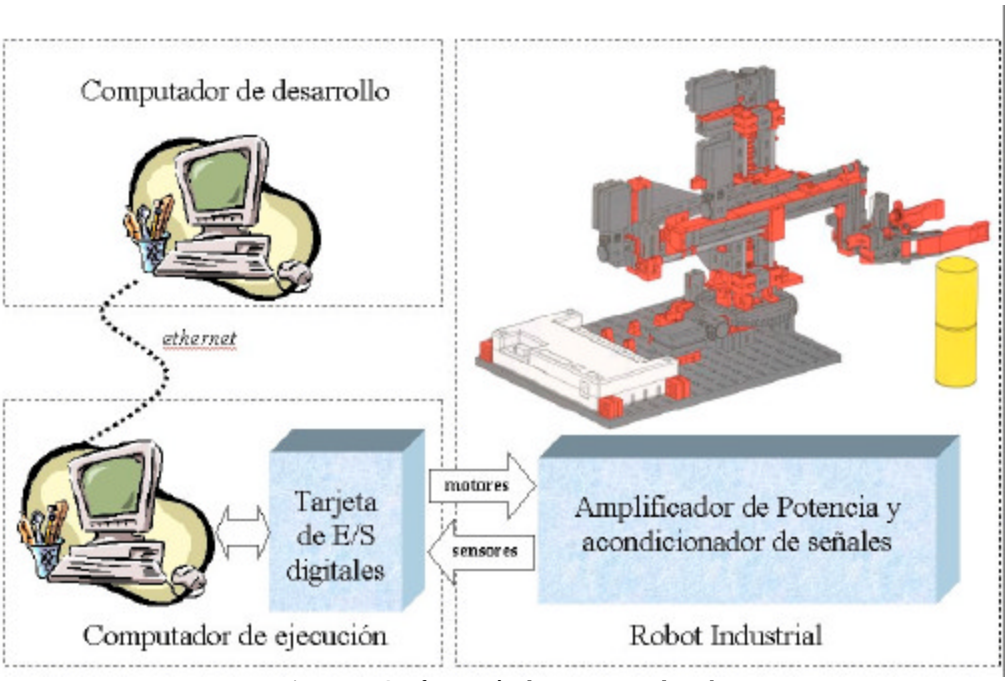


Figura 1: Configuración de un puesto de trabajo

3.2 Computador de desarrollo

Es un PC compatible con microprocesador AMD Athlon ™ a 1200 MHz con 256 Mbytes de memoria RAM, 20 Gigas de disco duro, tarjeta de red y sistema operativo Linux (RedHat 6.2). Se abrirá una cuenta a cada grupo de prácticas. En este PC se realizará el desarrollo de los programas que se lanzarán en el computador de ejecución. Se podrá emplear una interfaz de RTEMS para en ADA o en C. El compilador será el GCC de GNU y la herramienta de depuración el GDB de GNU. Los alumnos deberán estar familiarizados con estas herramientas y con el entorno

de programación en UNIX. Este PC dispone además de una tarjeta de red que es por donde se comunicará con el computador de ejecución.

3.3 Computador de ejecución

Es un PC compatible 486DX4 ™ de 100MHz con 8 Mbytes. No tiene disco duro, ya que el arranque lo hará desde la tarjeta de red que tiene y ejecutará el programa que se ha realizado en el PC de desarrollo. Cada vez que se modifique el programa se deberá reiniciar el PC de ejecución, de esta manera cargará automáticamente el nuevo programa.

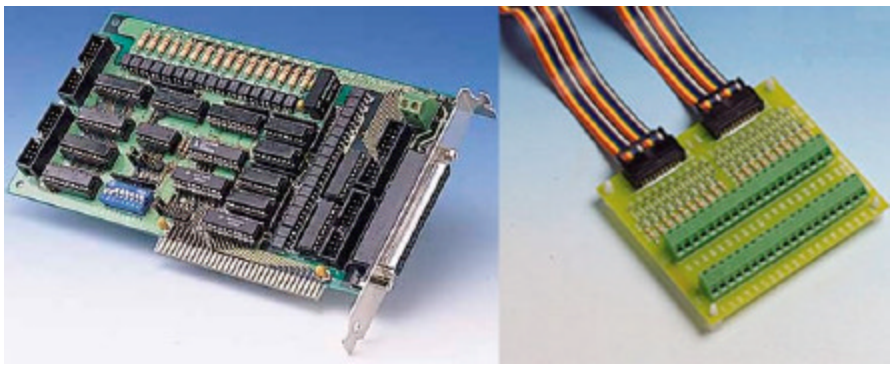


Figura 2: Tarjeta de entradas y salidas digitales y de bornas

El computador de ejecución tiene en una de sus ranuras de expansión una tarjeta ISA de entradas y salidas digitales, la PCL-730 de Advantech [5]. Esta cuenta con 32 entradas y 32 salidas digitales de las que tan solo se emplean 8 salidas y 8 entradas para controlar el robot.

Para realizar el cableado entre la tarjeta de entradas y salidas y el robot se emplea una tarjeta de bornas, la PCLD-780 en la que se pueden añadir filtros analógicos para eliminar ruido. En la figura 2 se aprecia estas dos tarjetas.

3.4 Robot industrial

Este bloque está compuesto por el robot y por una tarjeta de potencia acondicionadora de las señales. Se ha elegido el robot industrial que comercializa la compañía de Juegos educativos Fisher. El robot representado en la figura 3 tiene cuatro grados de libertad; giro sobre su base, elevación de su brazo, fondo o estiramiento del brazo y por último la pinza, para asir objetos. Por cada grado de libertad se cuenta con dos interruptores normalmente abiertos, uno para detectar la posición inicial o final de carrera y el otro para detectar la posición.

La tarjeta acondicionadora de señal ha sido desarrollada como resultado del proyecto fin de carrera [6]. Está basada en el circuito integrado L-293B de Thomson. Su función básica es alimentar los motores de corriente continua que funcionan con 9 voltios. Los sensores se conectan directamente a la placa de entradas y salidas del computador de ejecución sin ningún tratamiento.

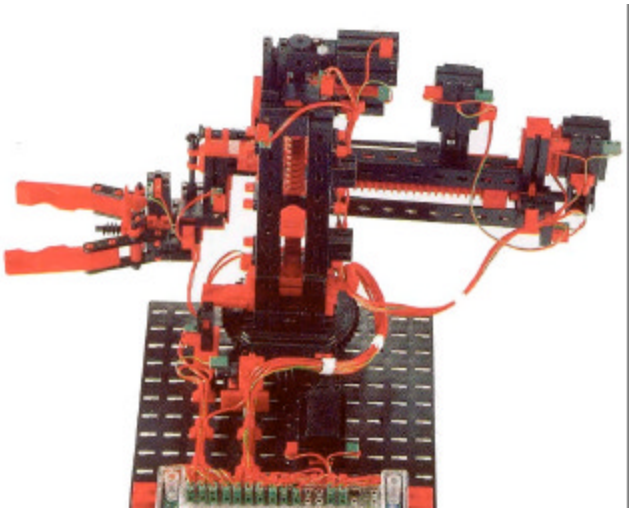


Figura 3: Robot industrial empleado (cortesía Fisher)

Dado que cada motor tiene tres estados posibles (parado, giro a la izquierda y giro a la derecha) éstos se codifican con al menos dos bits. Como se tienen cuatro motores se emplea un byte completo (8 bits) para codificar los motores. Como se tiene dos interruptores por grado de libertad también se necesita un byte completo (8 bits) para la lectura de la información en la que se encuentren. Este octeto de entradas como de salida se ha configurado en la posición 0x202 del computador de ejecución. Así para actuar sobre los

motores y para obtener el estado de los sensores se deberá escribir o leer de esta posición respectivamente. En la figura 4 se refleja la correspondencia de cada bit para los motores y sensores.

Debido a las características constructivas del robot y según la figura 4, se puede conseguir los movimientos descritos en la tabla 1 escribiendo el byte apropiado en la dirección 0x202h. El sensor final de carrera se buscará en el proceso de inicialización (*homing*) para que el programa pueda determinar donde se encuentra cada articulación. En todo el tiempo de vida en que el programa esté en ejecución éste sabrá donde se encuentra cada eje contando los pulsos. El robot está construido de tal forma que por cada vuelta que da el eje correspondiente se generan cuatro pulsos. Estos deberán por tanto ser contados y decontados adecuadamente en función del movimiento.

Tabla 1: Valores a escribir en la dirección 0x202 y movimientos del robot que se obtienen

0x01	Giro antihorario	0x04	Fondo atrás	0x10	Altura arriba	0x40	Pinza abrir
0x02	Giro horario	0x08	Fondo adelante	0x20	Altura abajo	0x80	Pinza cerrar

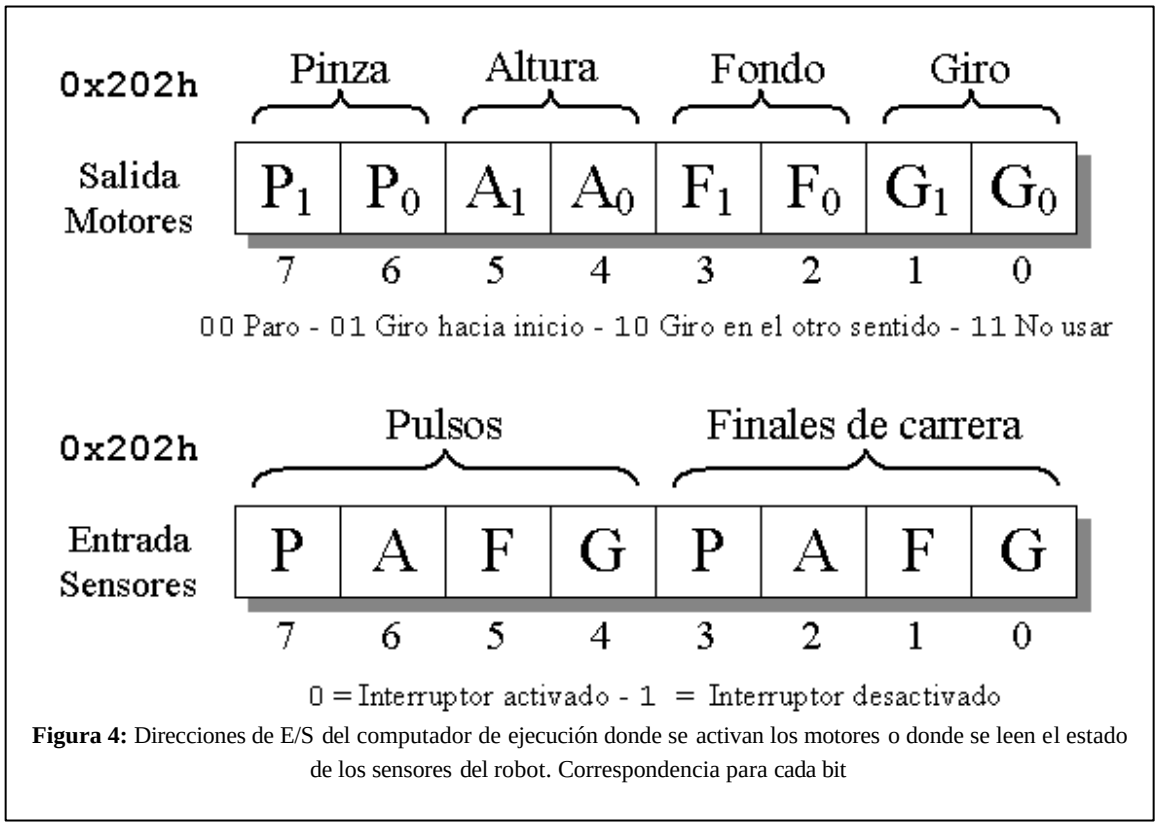


Figura 4: Direcciones de E/S del computador de ejecución donde se activan los motores o donde se leen el estado de los sensores del robot. Correspondencia para cada bit

4 Trabajo práctico

4.1 Iniciación

Antes de mover el robot los alumnos deben asegurarse de haber leído con todo detalle el presente cuadernillo de práctica y haber consultado las posibles dudas con los profesores de la asignatura.

? **SE RECOMIENDA leer atentamente el presente cuadernillo antes de manipular el robot**

Para familiarizarse con el entorno de desarrollo cruzado, antes de comenzar a mover el robot el alumno deberá compilar algún programa que saque mensajes por pantalla y requiera de la interacción del usuario a través del terminal. Podrá desarrollar programas empleando la interfaz de ADA o de C que ofrece RTEMS, aunque se recomienda que el alumno utilice ambas.

El siguiente paso será compilar y ejecutar los ejemplos que se añaden en el apéndice A. Con estos el alumno podrá manipular el robot y experimentará con los diversos movimientos actuando manualmente a través del terminal del computador de ejecución. Se recomienda activar manualmente los distintos interruptores y ver el estado que se lee de estos.

Cuando se mueva los motores hay que tener cuidado cuando se llega al final del recorrido ya que los motores se fuerzan y pueden dañarse. Si el programa no responde por la razón que fuera, apagar el computador de ejecución, ya que de esta manera se parará el motor. Asegurarse que el eje que se active pueda moverse con libertad y que no hay nada que lo obstaculiza. Si se observará alguna anomalía en el funcionamiento de algún motor o sensor, o en cualquier otro equipo avisar inmediatamente al profesor o en su defecto mandar un correo electrónico explicando el suceso y señalando el grupo y la hora en que ocurrió.

? **APAGAR EL COMPUTADOR DE EJECUCIÓN INMEDIATAMENTE si cualquiera de los ejes no avanza, llegan al límite o se mueven anormalmente**

4.2 Determinación de los límites

Como se explicó en el punto 2.4, el robot no dispone de un sensor absoluto que informe al sistema en cada momento donde se encuentra cada eje. En su lugar cuenta con sensores relativos, con los que se determina la posición respecto a un punto conocido. Para ello se emplean dos interruptores por eje. Esto es una técnica muy frecuente en la robótica industrial, otros tipos de sensores empleados en robótica pueden encontrarse en [7]. Uno de estos dos interruptores señala cuando se llega al final de un extremo, mientras que el otro contabiliza un número de pulsos por cada vuelta de motor. En este caso se generan cuatro pulsos por vuelta. Se deberá determinar los límites de cada eje medidos en pulsos. Para ello el alumno deberá realizar un programa que realice las siguientes acciones:

1. Girar el motor en sentido de avance hacia el final de carrera.
2. Cuando el programa detecte que se ha pulsado éste, poner un contador de pulsos a cero e iniciar el giro en el sentido contrario.
3. Cuando el alumno vea que el robot llega al otro extremo parará el motor, el número de pulsos contados será utilizado de ahora en adelante como límite de ese eje para el resto de los programas que se desarrollen.
4. Repetir el proceso para los siguientes ejes.

Se recomienda no trabajar con el robot cerca de los límites.

4.3 Realización de movimientos combinados

En este apartado el alumno hará un programa para que el robot manipule cualquiera de los objetos que se dispone u otros que el alumno aporte, siempre que éstos sean de poco peso. La estrategia a seguir será partiendo de los objetos de determinadas posiciones fijas el robot cambiará piezas o las pondrá una encima de otra. La acción a realizar será elegida por el grupo de práctica con la única restricción de que se deben mover los cuatro grados de libertad de forma combinada.

4.4 Problema de las torres de Hanoi

PROBLEMA

Se tienen 3 estacas numeradas 1, 2 y 3. Supóngase que en la estaca 1 están apilados de mayor a menor n discos de distinto tamaño. Hágase un programa que indique la manera de pasar los discos a la estaca 3, teniendo en cuenta que después de cualquier movimiento, los discos deben quedar apilados en alguna de las tres estacas y ningún disco puede quedar encima de otro de menor tamaño.

El "rompecabezas" conocido como la Torre de Hanoi fue presentado por el matemático francés Edouard LUCAS en 1883 [8], Lucas introdujo el problema románticamente contando la siguiente leyenda: En el principio de los tiempos, en lo alto de la Torre de Brahma, se encontraban tres finas agujas de diamante con 64 discos de oro puro descansando en la primera de ellas. Entonces, Dios encargó a un grupo de monjes que transfirieran los 64 discos a la tercer aguja de acuerdo a las reglas que se han dado en el enunciado del problema. Los monjes trabajan desde entonces día y noche sin descanso para completar su tarea; cuando ello ocurra, la Torre se derrumbará y el mundo habrá terminado.

En este caso el grupo deberá de resolver el famoso problema conocido como “Las torres de Hanoi” pasando de la situación de los tres discos de la figura 5-A a la 5-B.

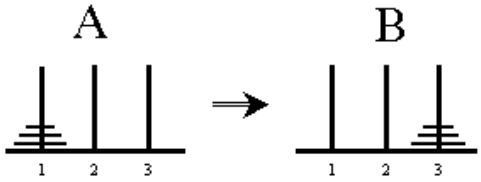


Figura 5: Problema de las torres de Hanoi

4.5 Realización de la memoria

El grupo deberá entregar una memoria describiendo todo el trabajo realizado, organigrama y toda la documentación necesaria para la comprensión del funcionamiento de sus programas. Será importante señalar la bibliografía consultada o punteros a internet.

5 Apéndices

5.1 Apéndice A

5.1.1 Ejemplo de programa de manejo del robot escrito en C

El siguiente programa imprime un menú en pantalla solicitando al usuario a que ponga en marcha o pare cualquiera de los grados de libertad del robot. En cada iteración se muestra el estado de los interruptores.

```
#include <stdio.h>
#include <unistd.h>

#define DIR_PUERTO_SALIDA  0x202
#define DIR_PUERTO_ENTRADA 0x202
#define STOP               0x00
#define GIRO_ANTIHORARIO   0x01
#define GIRO_HORARIO       0x02
#define FONDO_ATRAS        0x04
#define FONDO_ADELANTE     0x08
#define ALTURA_ARRIBA      0x10
#define ALTURA_ABAJO       0x20
#define PINZA_ABRIR        0x40
#define     PINZA_CERRAR       0x80

void ImprimeMenu(void);
void ImprimeEntradas(void);
char LeerPuerto();

/************************ FUNCION PRINCIPAL ************************/
void  *POSIX_Init (void  *argument)
{
 int c;

 while(1)
 {
   ImprimeMenu();
   ImprimeEntradas();
   scanf("%d",&c);
   switch (c)
   {
     case 0: exit(0);                                          break;
     case 1: outport_byte(DIR_PUERTO_SALIDA,GIRO_ANTIHORARIO); break;
     case 2: outport_byte(DIR_PUERTO_SALIDA,GIRO_HORARIO    ); break;
     case 3: outport_byte(DIR_PUERTO_SALIDA,FONDO_ATRAS);      break;
     case 4: outport_byte(DIR_PUERTO_SALIDA,FONDO_ADELANTE);   break;
     case 5: outport_byte(DIR_PUERTO_SALIDA,ALTURA_ARRIBA);    break;
     case 6: outport_byte(DIR_PUERTO_SALIDA,ALTURA_ABAJO);     break;
     case 7: outport_byte(DIR_PUERTO_SALIDA,PINZA_ABRIR);      break;
     case 8: outport_byte(DIR_PUERTO_SALIDA,PINZA_ABAJO);      break;
     case 9: outport_byte(DIR_PUERTO_SALIDA,STOP);             break;
     default : break;
```

```
   }
 }
} /* FIN función principal */

void ImprimeEntradas()
{
  char val='x';

  inport_byte(DIR_PUERTO_ENTRADA,val);
  if (val&0x80) printf("1"); else  printf("0");
  if (val&0x40) printf("1"); else  printf("0");
  if (val&0x20) printf("1"); else  printf("0");
  if (val&0x10) printf("1"); else  printf("0");
  if (val&0x08) printf("1"); else  printf("0");
  if (val&0x04) printf("1"); else  printf("0");
  if (val&0x02) printf("1"); else  printf("0");
  if (val&0x01) printf("1"); else  printf("0");
  printf("\n");
}

void ImprimeMenu(void)
{
  printf(" 1=GIRO ANTIHORARIO   2=GIRO HORARIO\n");
  printf(" 3=FONDO ATRAS        4=FONDO ADELANTE\n");
  printf(" 5=ALTURA ARRIBA      6=ALTURA ABAJO\n");
  printf(" 7=PINZA ABRIR        8=PINZA CERRAR\n");
  printf(" 9=STOP               0=EXIT\n");
}
```

5.1.2 Ejemplo de programa del manejo del robot escrito en ADA

El siguiente programa mueve el brazo en altura y calcula los milisegundos por cada paso en ese eje. Utiliza el paquete Robot_IO que se lista a continuación.

altura .adb

```
with Robot_IO;
use Robot_IO;

with Calendar;
use Calendar;

with Ada.Integer_Text_IO;
use Ada.Integer_Text_IO;

with Ada.Text_IO;
use Ada.Text_IO;

procedure Altura is

   Numero_de_Pasos : Integer := 0;
   Antes, Despues : Time;
   Entradas_Robot: Tipo_Entradas_Robot := Leer_Entradas;
   Pasos_Altura_Antes : Tipo_Pulsador := Entradas_Robot.Pasos_Altura;
   Salidas_Robot: Tipo_Salidas_Robot := (Parado, Parado, Parado, Parado);

   package Pulsador_IO is new Ada.Text_IO.Enumeration_IO (Tipo_Pulsador);
   use Pulsador_IO;
```

```
begin

    -- Primero lo llevamos hasta el fin de carrera
    While Leer_Entradas.Final_Carrera_Altura = No_Pulsado loop
        Salidas_Robot.Motor_Altura := Arriba;
        Escribir_Salidas (Salidas_Robot);
    end loop;

    --  Despues lo separamos del fin de carrera
    --  el tiempo depende del motor
    Salidas_Robot.Motor_Altura := Abajo;
    Escribir_Salidas (Salidas_Robot);
    delay 10.0;

    --  Ahora lo llevamos otra vez al fin de carrera contando los pulsos
    --  y el tiempo transcurrido. Además mostramos las entradas
    Salidas_Robot.Motor_Altura := Arriba;
    Antes := Clock;
    Escribir_Salidas (Salidas_Robot);

    While Leer_Entradas.Final_Carrera_Altura = No_Pulsado loop

        Entradas_Robot:= Leer_Entradas;

        put (Entradas_Robot.Final_Carrera_Giro);
        put (Entradas_Robot.Final_Carrera_Fondo);
        put (Entradas_Robot.Final_Carrera_Altura);
        put (Entradas_Robot.Final_Carrera_Pinzas);
        new_line;
        put (Entradas_Robot.Pasos_Giro);
        put (Entradas_Robot.Pasos_Fondo);
        put (Entradas_Robot.Pasos_Altura);
        put (Entradas_Robot.Pasos_Pinzas);
        new_line;

        if Entradas_Robot.Pasos_Altura /= Pasos_Altura_Antes then
            Numero_de_Pasos := Numero_de_Pasos + 1;
            Pasos_Altura_Antes := Entradas_Robot.Pasos_Altura;
        end if;
    end loop;

    Salidas_Robot.Motor_Altura := Parado;
    Despues := Clock;
    Escribir_Salidas (Salidas_Robot);

    put (Entradas_Robot.Final_Carrera_Giro);
    put (Entradas_Robot.Final_Carrera_Fondo);
    put (Entradas_Robot.Final_Carrera_Altura);
    put (Entradas_Robot.Final_Carrera_Pinzas);
    new_line;
    put (Entradas_Robot.Pasos_Giro);
    put (Entradas_Robot.Pasos_Fondo);
    put (Entradas_Robot.Pasos_Altura);
    put (Entradas_Robot.Pasos_Pinzas);
    new_line;

    put ("Milisegundos por paso altura = ");
    put (Integer (1000.0 * (Despues - Antes)) / Numero_de_Pasos);
    new_line;
```

```
end Altura;
```

robot_io.ads

```
package Robot_IO is

   Type Tipo_Pulsador is (Pulsado, No_Pulsado);

   Type Tipo_Motor_Giro is (Parado, Giro_Antihorario, Giro_Horario);
   -- Giro_Antihorario es hacia el fin de carrera

   Type Tipo_Motor_Fondo is (Parado, Atras, Adelante);
   --  Atras es hacia el fin de carrera

   Type Tipo_Motor_Altura is (Parado, Arriba, Abajo);
   --  Arriba es hacia el fin de carrera

   Type Tipo_Motor_Pinzas is (Parado, Abrir, Cerrar);
   --  Abrir es hacia el fin de carrera

   Type Tipo_Entradas_Robot is
       record
           Final_Carrera_Giro : Tipo_Pulsador;
           Final_Carrera_Fondo : Tipo_Pulsador;
           Final_Carrera_Altura : Tipo_Pulsador;
           Final_Carrera_Pinzas : Tipo_Pulsador;
           Pasos_Giro : Tipo_Pulsador;
           Pasos_Fondo : Tipo_Pulsador;
           Pasos_Altura : Tipo_Pulsador;
           Pasos_Pinzas : Tipo_Pulsador;
       end record;

   Type Tipo_Salidas_Robot is
       record
           Motor_Giro: Tipo_Motor_Giro;
           Motor_Fondo: Tipo_Motor_Fondo;
           Motor_Altura: Tipo_Motor_Altura;
           Motor_Pinzas: Tipo_Motor_Pinzas;
       end record;

   function Leer_Entradas return Tipo_Entradas_Robot;
   pragma Inline (Leer_Entradas);

   procedure Escribir_Salidas (Output : in Tipo_Salidas_Robot);
   pragma Inline (Escribir_Salidas);

private

   for Tipo_Pulsador use
           (Pulsado => 0, No_Pulsado =>1);
   for Tipo_Pulsador'size use 1;

   for Tipo_Motor_Giro use
            (Parado => 0, Giro_Antihorario => 1, Giro_Horario => 2);
   for Tipo_Motor_Giro'size use 2;

   for Tipo_Motor_Fondo use
            (Parado => 0, Atras => 1, Adelante => 2);
```

```
   for Tipo_Motor_Fondo'size use 2;

   for Tipo_Motor_Altura use
            (Parado => 0, Arriba => 1, Abajo => 2);
   for Tipo_Motor_Altura'size use 2;

   for Tipo_Motor_Pinzas use
            (Parado => 0, Abrir => 1, Cerrar => 2);
   for Tipo_Motor_Pinzas'size use 2;

   for Tipo_Entradas_Robot use
       record
           Final_Carrera_Giro at 0 range 0..0;
           Final_Carrera_Fondo at 0 range 1..1;
           Final_Carrera_Altura at 0 range 2..2;
           Final_Carrera_Pinzas at 0 range 3..3;
           Pasos_Giro at 0 range 4..4;
           Pasos_Fondo at 0 range 5..5;
           Pasos_Altura at 0 range 6..6;
           Pasos_Pinzas at 0 range 7..7;
       end record;

   pragma Pack (Tipo_Entradas_Robot);

   pragma Atomic (Tipo_Entradas_Robot);

   pragma Suppress_Initialization (Tipo_Entradas_Robot);

   for Tipo_Salidas_Robot use
       record
           Motor_Giro at 0 range 0..1;
           Motor_Fondo at 0 range 2..3;
           Motor_Altura at 0 range 4..5;
           Motor_Pinzas at 0 range 6..7;
       end record;

   pragma Pack (Tipo_Salidas_Robot);

   pragma Atomic (Tipo_Salidas_Robot);

   pragma Suppress_Initialization (Tipo_Salidas_Robot);

end Robot_IO;
```

robot_io.adb

```
with System.Machine_Code;
use System.Machine_Code;

package body Robot_IO is

   type Tipo_Puerto_IO is range 0 .. 16#FFFF#;
   for  Tipo_Puerto_IO'Size use 16;

   Entradas_Digitales_Robot : constant Tipo_Puerto_IO := 16#202#;
   Salidas_Digitales_Robot : constant Tipo_Puerto_IO := 16#202#;

   function Leer_Entradas return Tipo_Entradas_Robot is
      Tmp : Tipo_Entradas_Robot;
   begin
```

```
      Asm ("inb %%dx, %0",
           Tipo_Entradas_Robot'Asm_Output ("=a", Tmp),
           Tipo_Puerto_IO'Asm_Input  ("d",  Entradas_Digitales_Robot),
           "edx",
           True);
      return Tmp;
   end Leer_Entradas;

   procedure Escribir_Salidas (Output : in Tipo_Salidas_Robot) is
   begin
      Asm ("outb %0, %%dx",
           No_Output_Operands,
           (Tipo_Salidas_Robot'Asm_Input ("a", Output),
            Tipo_Puerto_IO'Asm_Input ("d", Salidas_Digitales_Robot)),
           "edx", True);
   end Escribir_Salidas;

end Robot_IO;
```

5.2 Apéndice B

5.2.1 Diagrama de motores y actuadores del robot

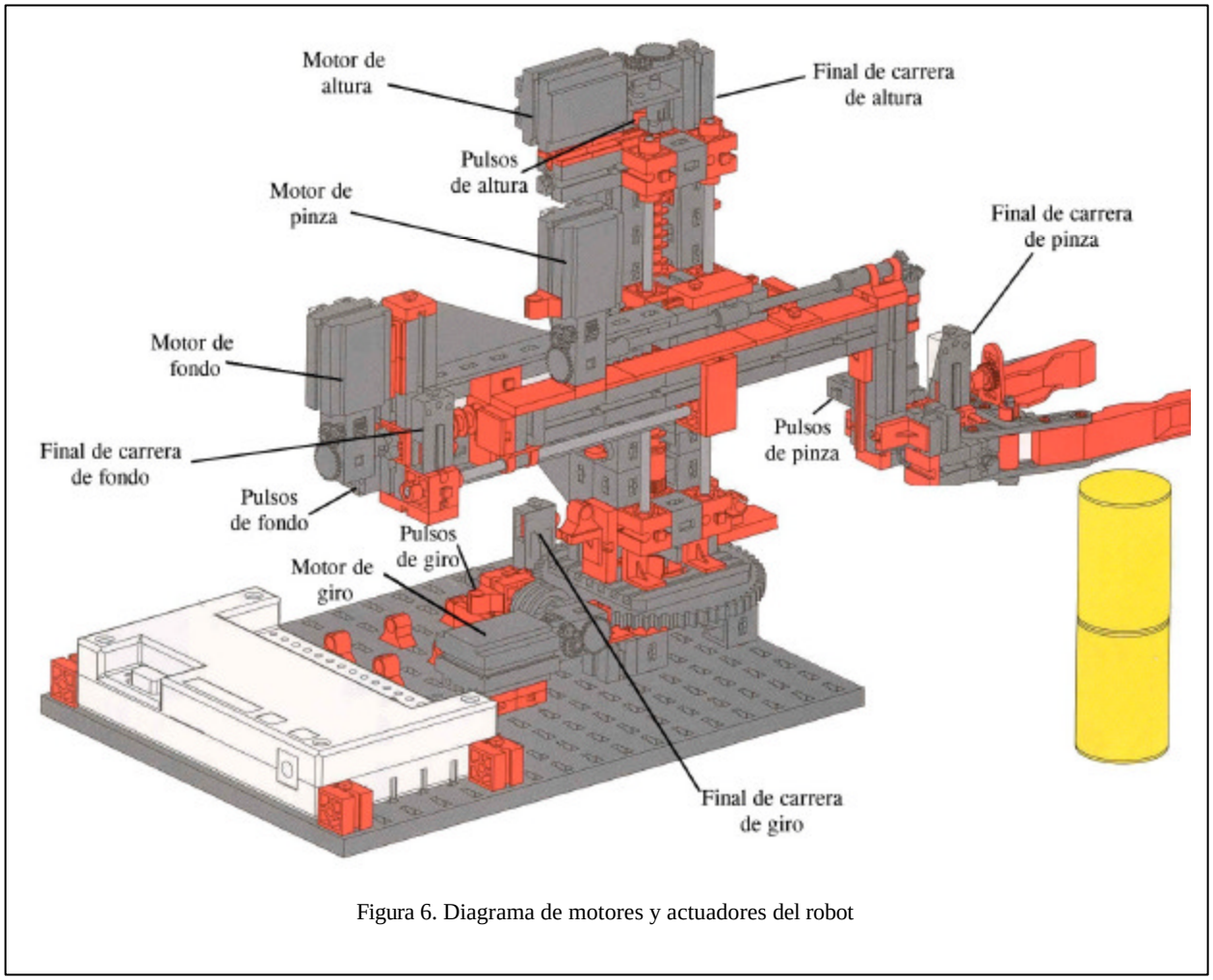
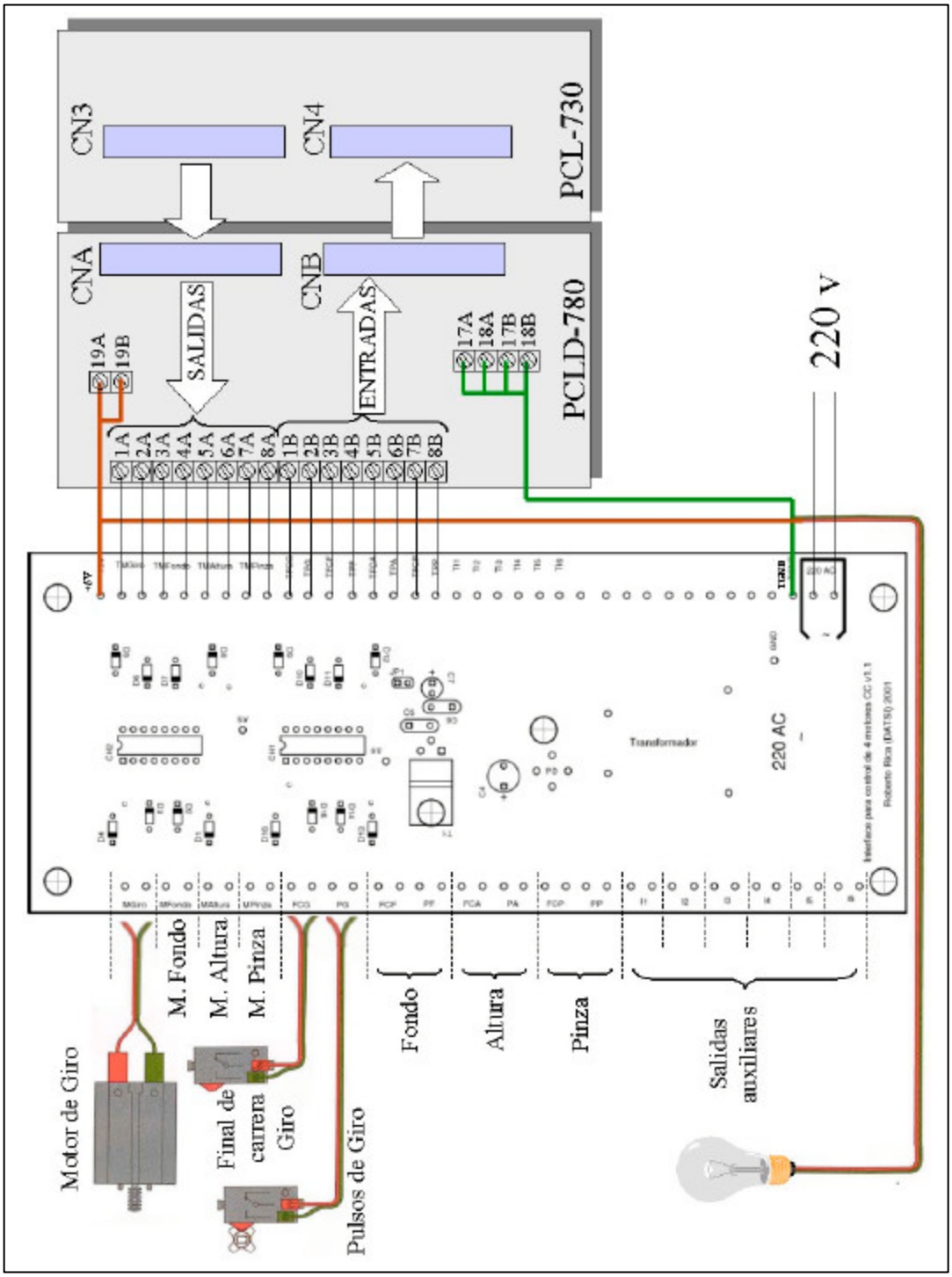


Figura 6. Diagrama de motores y actuadores del robot

5.3 Apéndice C

5.3.1 Diagrama de conexiones



6 Bibliografía

[1] RTEMS. Sistema operativo en tiempo real de libre distribución (http://www.rtems.com)
[2] Sala canalobre (Centro de cálculo) http://www.fi.upm.es/ccfi/recursos/aulas/canalobre.html
[3] Linux (Red Hat 6.2) http://www.linux.org
[4] Robot Industrial Fischer http://www.fischerwerke.de/test/ft/englisch/ft-30408.html
[5] Advantech PCL-730 http://www.advantech.com/products/PCL-730.asp
[6] Roberto RICA GUTIÉRREZ, *Construcción de un Brazo Robótico e Interfaz de Control para un Sistema Empotrado Basado en el Sitema Operativo RTEMS.* Proyecto fin de carrera. Universidad Politécnica de Madrid, Facultad de Informática (Diciembre de 2001)
[7]Antonio BARRIENTOS, L. Felipe PEÑÍN, Carlos BALAGUER y Rafael ARACIL. *Fundamentos de Robótica*. Editorial Mc Graw Hill. ISBN: 84-481-0815-9. Pp 327 (1999)
[8] R. Graham, D. Knuth y O. Patashnik. *Concrete Mathematics*
[9] Katsuhiko OGATA. Ingeniería moderna de control 3º Edición. Editorial Prentice Hall. ISBN 979-17-0048-1 (1998)
[10] Antonio BARRIENTOS, Ricardo SANZ, Fernando MATIA y Ernesto GAMBAO. Control de sistemas Continuos. Editorial Mc Graw Hill. ISBN: 84-481-0605-9 (1996)
[11] Alan Burns & Andy Wellings. Real-Time Systems and Programming Languages. 3rd ed. Addison-Wesley, 2001. ISBN 0-201-72988-1
[12] John Barnes. Programming in Ada 95, 2nd. ed. Addison-Wesley, 1998. ISBN 0-201-34293-6
[13] Brian Kernigan & Dennis Richtie. The C Programming Language. 2nd. ed (ANSI-C) Prentice-Hall, 1989. ISBN 0-13-110362-8.

DATSI-UPM

FACULTAD DE INFORMÁTICA

Departamento de Arquitectura y Tecnología de Sistemas Informáticos

Universidad Politécnica de Madrid

www..fi.upm.es

Campus de Montegancedo
Boadilla del Monte
28660 Madrid
ESPAÑA

Telf: +34 91 336 7399
Fax: +34 91 336 7412
http://www.datsi.fi.upm.es

www.upm.es