UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

INGENIERÍA INDUSTRIAL

"PROGRAMACIÓN DE MICROBOTS MOWAY PARA LA REALIZACIÓN DE TAREAS COOPERATIVAS MEDIANTE COMUNICACIÓN POR RF"

PROYECTO FIN DE CARRERA

Septiembre-2009

AUTORA: María del Carmen Vera Sánchez
Directora: María Asunción Vicente Ripoll

VISTO BUENO Y CALIFICACIÓN DEL PROYECTO

Título proyecto:

PROGRAMACIÓN DE MICROBOTS MOWAY PARA LA REALIZACIÓN DE TAREAS COOPERATIVAS MEDIANTE COMUNICACION POR RF

Proyectante: MARÍA DEL CARMEN VERA SÁNCHEZ

Director/es: MARÍA ASUNCIÓN VICENTE RIPOLL

VºBº director/es del proyecto:

Fdo.: Fdo.:

Lugar y fecha:

CALIFICACIÓN NUMÉRICA ☐ MATRÍCULA DE HONOR ☐

Méritos justificativos en el caso de conceder Matrícula de Honor:

Conforme presidente:	Conforme secretario:	Conforme vocal:
Fdo.:	Fdo.:	Fdo.:

Lugar y fecha:

ÍNDICE

INTRODUCCIÓN

Este proyecto está centrado en el desarrollo de aplicaciones de robótica móvil haciendo uso de la comunicación por radiofrecuencia en el robot Moway.

Moway es un pequeño robot diseñado principalmente para realizar aplicaciones prácticas de microbótica. Es una herramienta práctica dentro del mundo de la enseñanza y el aprendizaje, ideal para todos aquellos interesados en el mundo de la robótica y que desean introducirse en él de forma económica. Moway es un robot programable y autónomo de apenas 100 gramos de peso, creado para comunicarse inalámbricamente con los de su especie o con un ordenador, interactuando con su entorno.

Básicamente, los objetivos que se persiguen en este trabajo son los siguientes:

- Estudio y análisis del nivel físico del microbot, especialmente su módulo de comunicación por radiofrecuencia.
- Análisis de los programas ejemplo de RF que se distribuyen junto con el microbot.
- Diseño de tareas de robótica cooperativa.
- Programación de algoritmos para resolver las tareas cooperativas propuestas.

En este documento se detallan las características físicas generales del microbot en el capítulo 1 (*Descripción del robot Moway*). Posteriormente, en el capítulo 2 (C*omunicación por radiofrecuencia en el robot Moway*) se incide en los aspectos más relevantes del módulo de comunicación por radiofrecuencia que se distribuye junto con el Moway. En el capítulo 3 (*Primeros programas*) se presenta una recopilación de los primeros programas desarrollados para el Moway que hacen uso de su módulo de comunicación por radiofrecuencia. Y por último, en el capítulo 4 (*Tareas cooperativas con Moway*) se presentan las tareas de robótica móvil cooperativa diseñadas para los Moways, además de la explicación de los algoritmos empleados para resolverlas. Las tareas cooperativas diseñadas se pueden clasificar en cuatro tipos:

- Compartición de información para la repetición de un trayecto.
- Cooperación para desplazar una carga pesada o voluminosa.
- Cooperación para la limpieza de un área determinada.
- Cooperación para el guiado de objetos por un camino.

Todo el material generado durante la realización de este proyecto fin de carrera se adjunta en un CD al final de este documento.

En este CD se encuentran todos los códigos fuente de los programas descritos en los capítulos 3 y 4, así como las librerías empleadas para la comunicación por RF y el control del sistema motriz y de los sensores. Además hemos realizado una serie de videos que muestran las tareas propuestas. Y de forma complementaria, se adjuntan todos los manuales relacionados con el Moway junto con su software de grabación y control.

CAPÍTULO 1

DESCRIPCIÓN DEL ROBOT MOWAY

1.1.- DESCRIPCION FÍSICA DEL MICROBOT MOWAY

1.1.1.- Introducción

Moway es un pequeño robot diseñado esencialmente para desarrollar aplicaciones prácticas de microbótica. Fue diseñado para emplearse como herramienta práctica dentro del ámbito de la enseñanza y el aprendizaje, es idóneo para todos los interesados en la robótica y que quieran introducirse en este mundo de forma económica.

Debido a su reducido tamaño (similar al de un ratón de ordenador), se le conoce como robot de bolsillo. A pesar de sus pequeñas dimensiones y su peso de apenas 100 gramos, Moway es programable, autónomo y posee la capacidad de comunicarse inalámbricamente con los de su especie o con un ordenador, interactuando con su entorno.

Figura 1: 2 Moways con sus tarjetas de radiofrecuencia y su base

Su diseño compacto le permite moverse con agilidad y elegancia, es capaz de desarrollar acciones como la detección de obstáculos, reclusión en recintos cerrados, seguimiento de línea y reconocimiento de focos luminosos.

A continuación se citan algunas de las habilidades más destacadas de Moway:

- Programable en diferentes lenguajes: Moway puede ser programado en lenguaje ensamblador, en C o a través de un compilador gráfico diseñado específicamente para él, apto para elaborar desde simples aplicaciones hasta complicadas tareas.
- Dotado de múltiples sensores que le permite interactuar y desenvolverse en un entorno real.
- Permite la comunicación con otros microbots, para aplicaciones en las que interactúan varios Moways.
- Posee la facultad de comunicarse con un ordenador para su programación y desarrollo de tareas.
- Dispone, adicionalmente, de módulos de radiofrecuencia: mediante 2 módulos RF (uno para el PC y otro para el robot), podemos ponernos en contacto con el robot desde nuestro PC sin usar cables, o conectar varios robots por RF.
- Batería que le proporciona una autonomía de 2 horas.

- Preparado para robótica colaborativa.

1.1.2.- Componentes del microbot Moway

Se pueden observar todos los componentes de Moway en la figura 2:

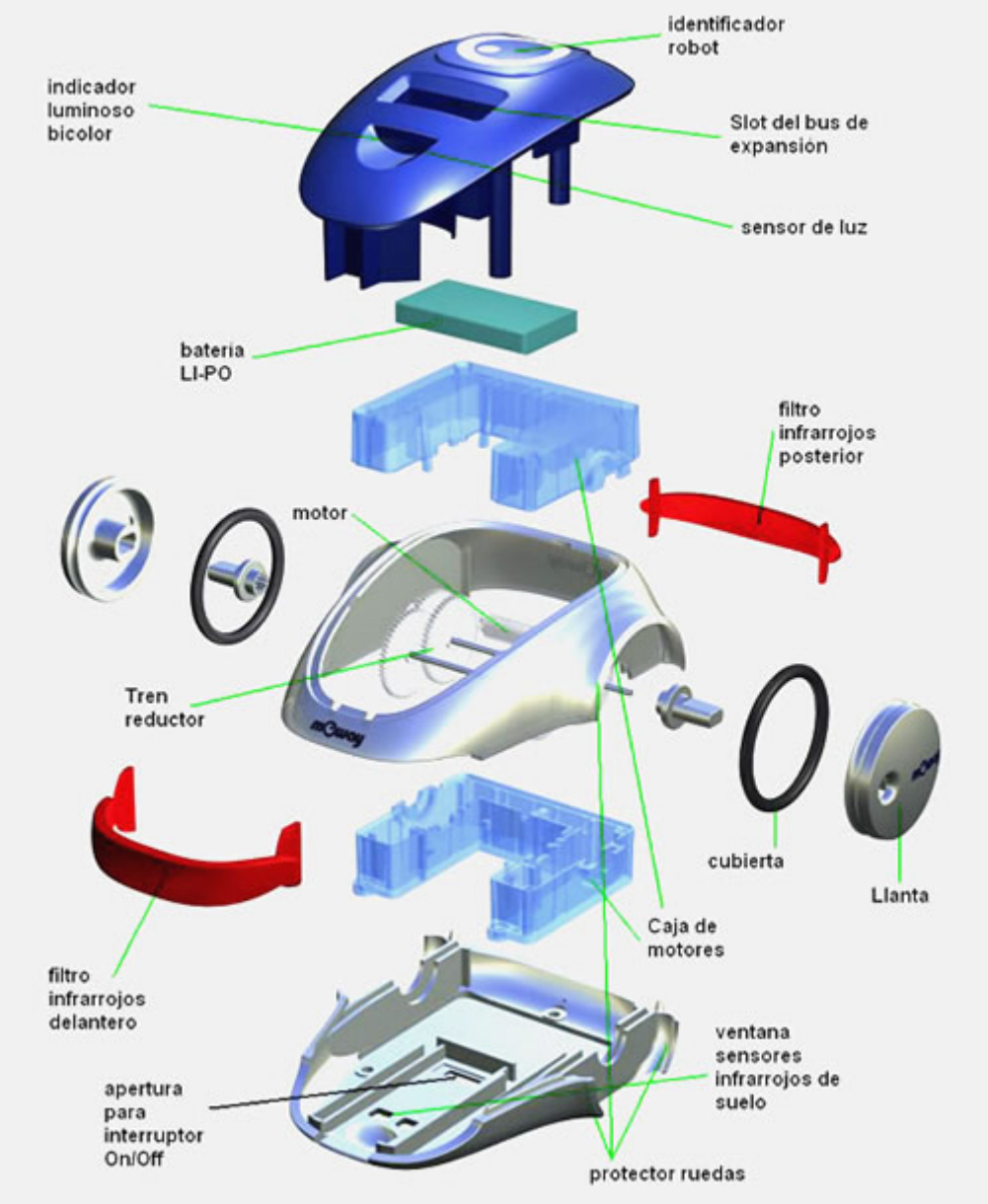

Figura 2: Vista explotada de Moway

En el interior de un Moway se encuentran los elementos que se especifican a continuación:

- Procesador
- Sistema motriz
- Grupo de sensores e indicadores
- Conector de expansión

- Pad libre
- Sistema de alimentación

Procesador o microcontrolador principal

El microbot Moway está gobernado por un microcontrolador PIC16F876A del fabricante Microchip Technology que trabaja a 4Mhz. De sus puertos de entrada/salida cuelgan todos los periféricos distribuidos por el robot.

A continuación se muestra la tabla de asignación de pines:

Pin PIC	I/O	Sensor
PORTA		
RA0	I	Luz
RA1	I	Receptor infrarrojo derecho
RA2	I	Receptor sensor línea derecho
RA3	I	Receptor infrarrojo izquierdo
RA4	O	LED superior rojo
RA5	I	Receptor sensor línea izquierdo
PORTB		
RB1	O	Trasmisor sensores línea
RB2	O	Trasmisor infrarrojo
RB4	O	LED inferior derecho
RB6	O	LED superior verde
PORTC		
RC6	O	LED inferior izquierdo
RC7	I/O	Pad libre

Tabla 1: Entradas y salidas del PIC

La grabación del microprocesador se realiza mediante la Base Moway.

Grabación	I/O	PIC
Pin1	O	GND
Pin2	I	MCLR
Pin3	I	ICSPDAT
Pin4	I	ICSPCLK
Pin5	I	RB3
Pin6	O	Vcc_Batt 4.1v

Tabla 2: Pines de grabación

Sistema motriz

Se trata de un grupo motor con control de trayectoria comandado por I2C.

Los microbots Moway disponen de un servo-motor doble para poder desplazarse. Consta de una parte electrónica y otra mecánica. La parte electrónica se encarga de controlar la velocidad de los motores y la parte mecánica permite el desplazamiento con

una potencia suficiente para que Moway se desplace sin problemas en diferentes entornos.

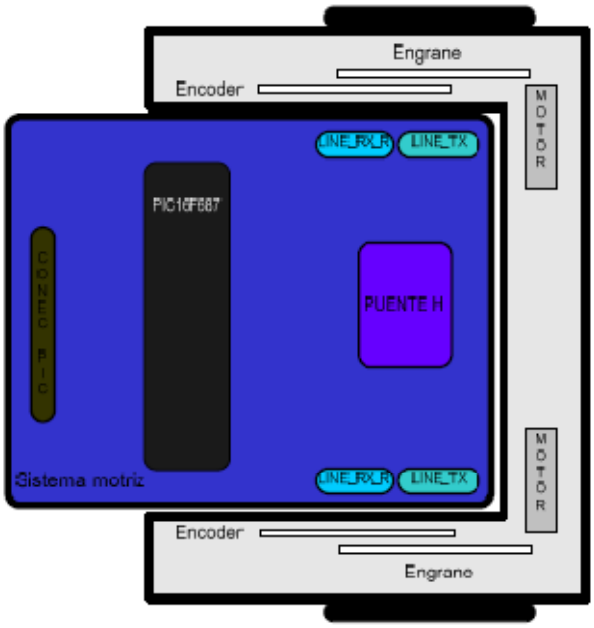

Figura 3: Sistema motriz

Funcionalidades del grupo servo-motor:

- Control de velocidad: controla la velocidad de cada motor por separado mediante control proporcional con retroalimentación negativa de la señal de los encoders (la rotación de la rueda es monitorizada por medio de una pegatina encoder y un sensor infrarrojo).
- Control de tiempo: controla el tiempo en cada comando con una precisión de 100ms.
- Control de distancia recorrida: controla la distancia recorrida en cada comando con una precisión de 1.7mm.
- Cuentakilómetros general: cuenta la distancia recorrida desde el comienzo de los comandos.
- Control de ángulo: control de ángulo cuando se produce la rotación de Moway.

El microcontrolador sólo tiene que mandar el comando por el protocolo I2C al sistema motriz y éste será el encargado de controlar los motores dejando libre de carga de trabajo al microcontrolador principal.

Grupo de sensores e indicadores

Este grupo consta de diferentes sensores e indicadores luminosos conectados al microprocesador de Moway con los que el robot interactúa con el mundo exterior:

- Sensores de línea: dos sensores optorreflectivos infrarrojos para el suelo.
- Detectores de obstáculos: dos sensores infrarrojos anticolisión.
- Sensor de intensidad de luz direccional.

- Dos LEDS frontales.
- Indicador luminoso superior bicolor.
- Un conector de expansión.
- Un pad libre.

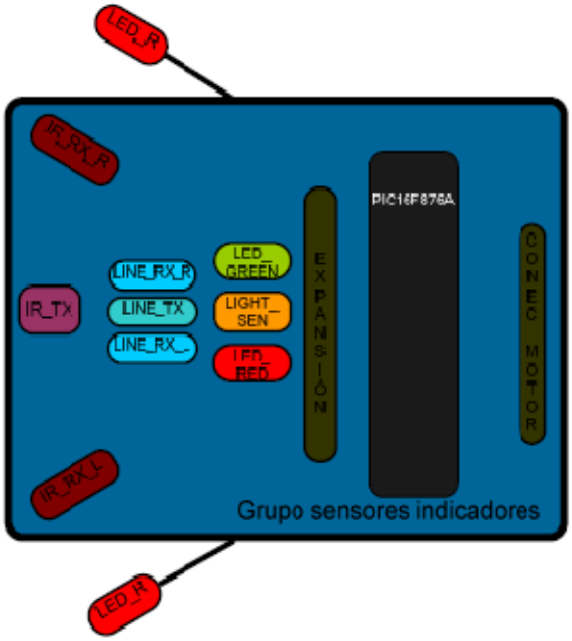

Figura 4: Sensores e indicadores

Sensores de línea

Los sensores de línea son dos optoacopladores de reflexión montados en la parte inferior delantera del robot. Utilizan la reflexión de luz infrarroja para detectar el tono del suelo en el punto en que se encuentra el robot.

Estos dos sensores están conectados a dos de los puertos analógicos del microcontrolador de manera que no sólo se pueden detectar contrastes fuertes en el suelo, como líneas blancas sobre fondo negro, sino que es posible discernir entre diferentes tonos. Cada sensor (KTIR0711S de Kingbright) consta de un diodo LED que emite luz en el espectro infrarrojo y de un receptor de alta sensibilidad capaz de detectar la luz reflejada en el suelo.

Los casos que se tiene son los siguientes:

- Superficie clara: La superficie blanca hace que toda la luz infrarroja se refleje y por lo tanto a la salida del transistor en modo común obtenemos un voltaje bajo.
- Superficie de color: La superficie de color hace que parte de la luz emitida se refleje obteniendo un voltaje intermedio en la entrada del canal analógico del microcontrolador. De esta manera es fácil identificar colores.
- Superficie oscura: La superficie oscura hace que se refleje muy poca luz teniendo un voltaje alto a la salida del sensor.

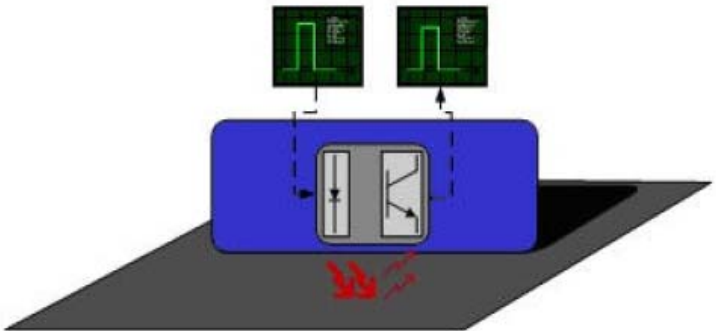

Figura 5: Salida del sensor de línea para superficies oscuras

En la siguiente tabla se muestran las conexiones del PIC con el sensor de línea:

Pin PIC	I/O	Sensor
PORTA		
RA2	I	Receptor sensor línea derecho
RA5	I	Receptor sensor línea izquierdo
PORTB		
RB1	O	Trasmisor sensores de línea izquierdo y derecho

Tabla 3: Pines del sensor de línea

Los dos diodos LED están conectados al mismo pin del microcontrolador.

Sensores detectores de obstáculos

Igual que los sensores de línea, los detectores de obstáculos utilizan también la luz infrarroja para detectar objetos situados en la parte delantera de Moway. El material del frontal de Moway es un filtro infrarrojo que bloquea parte de la componente infrarroja de la luz ambiente. El sensor está compuesto por una fuente de luz infrarroja (KPA3010-F3C de Kingbright) y dos receptores colocados en ambos extremos de Moway.

La salida de los receptores PT100F0MP de Sharp está conectada a las entradas analógicas del microcontrolador por lo que no sólo se detecta la presencia de algún objeto (modo digital) sino que también podemos medir la distancia al mismo (modo analógico).

El funcionamiento del sensor es similar al sensor de línea. El emisor de luz emite un pulso de una duración de 70µs que, si existe un obstáculo, el receptor capta utilizando una etapa de filtrado y amplificación. La distancia de alcance en digital es de unos 3cm. y se recomienda un entorno claro para aumentar la posibilidad de reflexión de la luz infrarroja.

A continuación aparece la tabla de asignación de pines del sensor:

Pin PIC	I/O	Sensor
PORTA		
RA0	I	Luz
RA1	I	Receptor infrarrojo derecho
RA3	I	Receptor infrarrojo izquierdo
PORTB		
RB2	O	Trasmisor infrarrojo

Tabla 4: Pines del sensor de obstáculo

Sensor de luz

Este sensor permite conocer la intensidad de luz que entra por una pequeña apertura con forma de media luna en la parte superior del chasis. Al estar ésta orientada hacia delante permite conocer dónde está situada la fuente de luz.

Figura 6: El sensor de luz se encuentra en la apertura con forma de media luna

La salida del sensor está conectada a un puerto analógico del microcontrolador de manera que con una simple lectura del ADC podemos saber el nivel de intensidad de luz y si éste ha aumentado o disminuido con respecto a la última lectura.

Pin PIC	I/O	Sensor
PORTA		
RA0	I	Luz

Tabla 5: Pin del sensor de luz

LEDS frontales

Los dos LEDS, de color rojo y situados uno en cada lado de la parte frontal del robot, se encuentran detrás del filtro infrarrojo delantero de manera que sólo se ven cuando se activan. Están conectados a dos salidas digitales del microcontrolador.

Estos LEDS deben estar apagados cuando se usan los detectores de obstáculos.

Figura 7: 2 Moways con sus LEDS frontales encendidos

Pin PIC	I/O	Sensor
PORTA		
RB4	O	LED inferior derecho
PORTC		
RC6	O	LED inferior izquierdo

Tabla 6: Pines de los LEDS frontales

LED superior bicolor

Este indicador doble comparte la misma apertura en la parte superior del robot que el sensor de luz. Están conectados a dos salidas digitales del microcontrolador. Deben estar apagados cuando se desee hacer una lectura de la intensidad de luz.

Pin PIC	I/O	Sensor
PORTA		
RA4	O	LED superior rojo
PORTB		
RB6	O	LED superior verde

Tabla 7: Pines del LED superior

Conector de expansión

Este conector permite la conexión de Moway con módulos comerciales o con circuitos electrónicos que se desee. Permite hacer aplicaciones colaborativas complejas sin tener que preocuparse de la complicada comunicación inalámbrica.

Como se observa en la siguiente tabla, es posible la conexión de dispositivos I2C y SPI comerciales. Por otro lado existe en el mercado el módulo de RF BZI-RF2GH4 totalmente compatible con Moway y con librerías específicas que permite la comunicación entre varios Moways y con el PC mediante la Base Moway.

Pin Expa	I/O	PIC
Pin1	O	Vcc 2,8v
Pin2	O	GND
Pin3	I/O/CCP1	RC2
Pin4	I/O/ICSPDAT	RB7
Pin5	I/O/I2C/SPI	RC3
Pin6	I/O/SPI	RC5
Pin7	I/O/I2C/SPI	RC4
Pin8	I/O/INT	RB0

Tabla 8: Pines del conector de expansión

Pad libre

El PCB de Moway tiene un Pad, accesible sólo abriendo el robot, situado al lado de los sensores de línea para que el usuario pueda conectar sus circuitos electrónicos.

Pin PIC	I/O	Sensor
PORTC		
RC7	I/O	Pad libre

Tabla 9: Pin del pad libre

Sistema de alimentación

La batería es una pequeña célula LI-PO recargable por USB. Se carga de forma automática al conectar Moway al PC a través del puerto USB de cualquier ordenador mediante la Base Moway.

Su pequeño tamaño, ligereza y flexibilidad hacen de estas baterías una perfecta fuente de energía para Moway.

La duración de la batería depende en gran medida de los sensores activos y del tiempo de utilización de los motores. De todas formas, la Base Moway indica la cantidad de carga que tiene el robot en cada momento. El tiempo de carga aproximado es de 2h.

Módulo de radiofrecuencia

Para la comunicación inalámbrica se utilizan los módulos BZI-RF2GH4 basados en el transceptor nRF24L01 fabricado por “Nordic Semiconductors”. Se acoplan a Moway gracias al conector que posee en la rendija superior. Del mismo modo, se pueden insertar en la Base Moway. Mediante un juego de 3 tarjetas podemos mantener comunicaciones entre 2 microbots y también enviar datos del PC a Moway y viceversa.

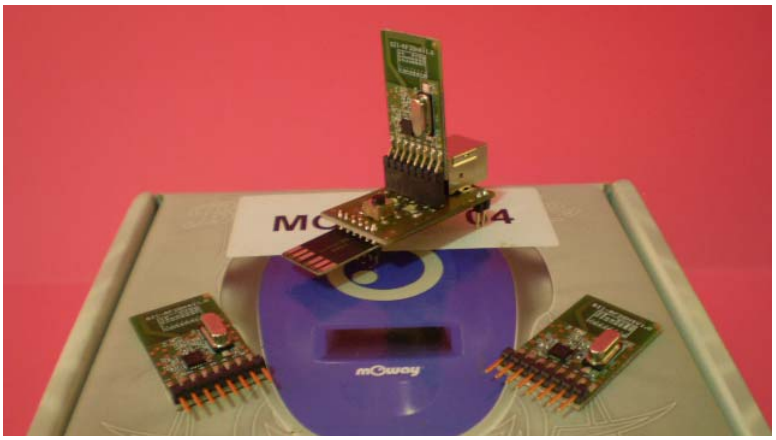

Figura 8: Módulos de radiofrecuencia BZI-RF2GH4

1.2.- SOFTWARE DE GRABACIÓN. MOWAY CENTER

1.2.1.- Base Moway o programador

La Base Moway o programador es una PIC 18F2550 de reducido tamaño y muy ligero. Es una pieza importante de Moway ya que es la encargada de generar las señales de grabación de los programas. Otras tareas que realiza son el control de la carga de la batería de Moway y la gestión de las comunicaciones entre los microbots y el ordenador. El programador se conecta a Moway mediante una pequeña apertura que éste posee debajo del filtro infrarrojo delantero. Para enchufarla al ordenador se emplea el cable USB.

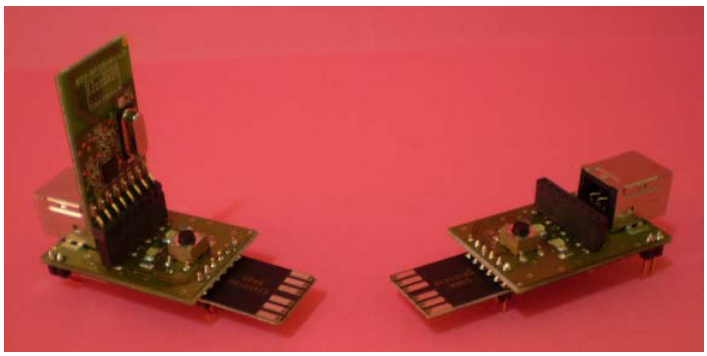

Figura 9: Bases Moway

1.2.2.- Descripción de Moway Center

Moway Center es el software encargado de controlar la Base Moway. Este programa lo podemos obtener en la página web de Moway de forma gratuita. Con él podemos observar el estado del microbot, cargar los programas, transmitir datos desde el PC a Moway y viceversa. Está compuesto por dos ventanas: Principal y Radio Frecuencia.

1.2.2.1.- Principal

Esta ventana permite observar el estado de Moway y realizar acciones sobre él.

Como se puede apreciar en la siguiente figura, la ventana se divide en 5 apartados: Moway Status, Acciones, Batería, Radio-Frecuencia e Info.

Figura 10: Ventana principal de Moway Center

Moway Status

Esta sección indica el estado de Moway. Hay 3 estados:

- Conectado. El robot está conectado a la base y encendido.
- Apagado. El robot está conectado a la base pero apagado, así se puede ir cargando la batería.
- Desconectado. El robot no está conectado a la base.

Figura 11: Estado de Moway

Acciones

Moway Center dispone de 3 botones para realizar acciones sobre Moway cuando éste está conectado:

- Grabar un programa, mediante al archivo ".HEX".

- Leer la memoria de programa de Moway.
- Borrar la memoria de programa de Moway.

Figura 12: Acciones

Batería

En esta sección aparece el porcentaje de carga de la batería. Mientras el robot está conectado al ordenador se produce su carga.

Figura 13: Moway cargando su batería

Como se observa en la figura 13, el robot está conectado a la base Moway, la cual está conectada al PC mediante un cable USB. Mientras la batería se está cargando, el LED verde de la base permanece encendido.

Radio-Frecuencia

Se utiliza para mostrar u ocultar la ventana de Radio Frecuencia.

Info

Muestra diferente información sobre el software y los procesos.

1.2.2.2.- Radio Frecuencia

Con esta ventana se pueden enviar y recibir datos a través de la tarjeta de comunicaciones BZI-RF2GH4. La tarjeta debe estar conectada a la base Moway, y ésta a su vez tiene que estar enchufada al PC.

Observamos que se divide en 4 secciones:

Figura 14: Ventana de Radiofrecuencia

Configuración RF

Para configurar el módulo RF hay que introducir la dirección y el canal de comunicaciones en hexadecimal.

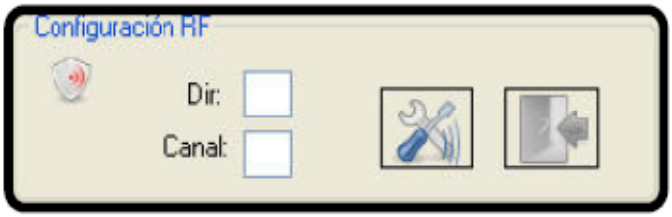

Figura 15: Configuración RF

Dato a enviar

Los datos a enviar se indican en hexadecimal y deben tener los 8 bytes definidos. El byte de la izquierda es el menos significativo.

Figura 16: Dato a enviar

Envío

Al pulsar el botón, los datos son enviados y si se produce algún problema se muestra un mensaje con el error producido.

Recepción

En esta parte se muestran los mensajes recibidos especificando la dirección del emisor y la hora de llegada.

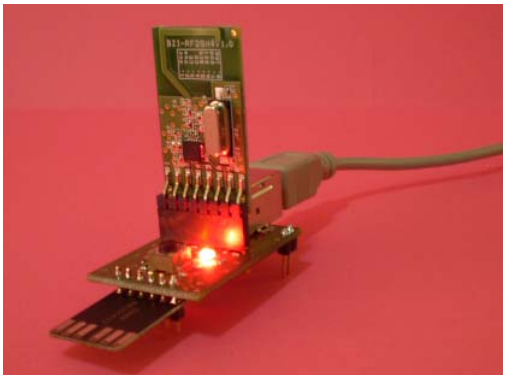

Figura 17: Base Moway con la tarjeta de comunicación preparada para recibir datos

1.2.3.- Grabación de un programa mediante Moway Center

En la página web de Moway podemos encontrar el pack de instalación. Éste contiene el programa Moway Center, el driver de la Base Moway, las librerías de motores y sensores, el manual de Moway, los recursos de radiofrecuencia y algunos proyectos compilados.

Al conectar la base al PC mediante el cable USB por primera vez, se debe indicar la dirección del driver en el asistente para nuevo hardware encontrado. El driver se encuentra en la carpeta “Driver” que está dentro de la carpeta “Moway Pack v1.0.0”, éste última se genera al instalar el software Moway Center.

Para poder grabar un programa, se conecta el robot a la base y ésta el ordenador. Se ejecuta Moway Center, que indica el estado del robot y su carga, y se enciende el robot. Se pulsa el botón de grabar de las acciones y se selecciona el archivo “.HEX”.

Moway comenzará a ejecutar el programa que tiene cargado tras desconectarlo del PC y activarlo.

Se puede programar tanto en ensamblador, utilizando el programa Mplab, como en C, mediante CCS PIC C Compiler o mediante Mplab compilando con CCS. En este trabajo se ha decidido programar en C (bajo CCS) porque así resulta más simple elaborar códigos con algoritmos más complejos. Aunque la programación en ensamblador es más eficiente, también es más costosa a la hora de programar rutinas complejas.

Figura 18: Grabación de un programa

1.2.4.- Creación de un proyecto con CCS PIC C Compiler

Para crear el proyecto se selecciona “Pic Wizard” del menú “Proyect”. Tras indicar el nombre y la dirección del proyecto, se selecciona, en la pestaña “General”, el PIC 16F876A, la frecuencia de reloj de 4000000Hz y Cristal Osc <=4Mhz. Por último, en la sección “Comunications” se deshabilita el RS-232. El programa nos crea un archivo “.C”.

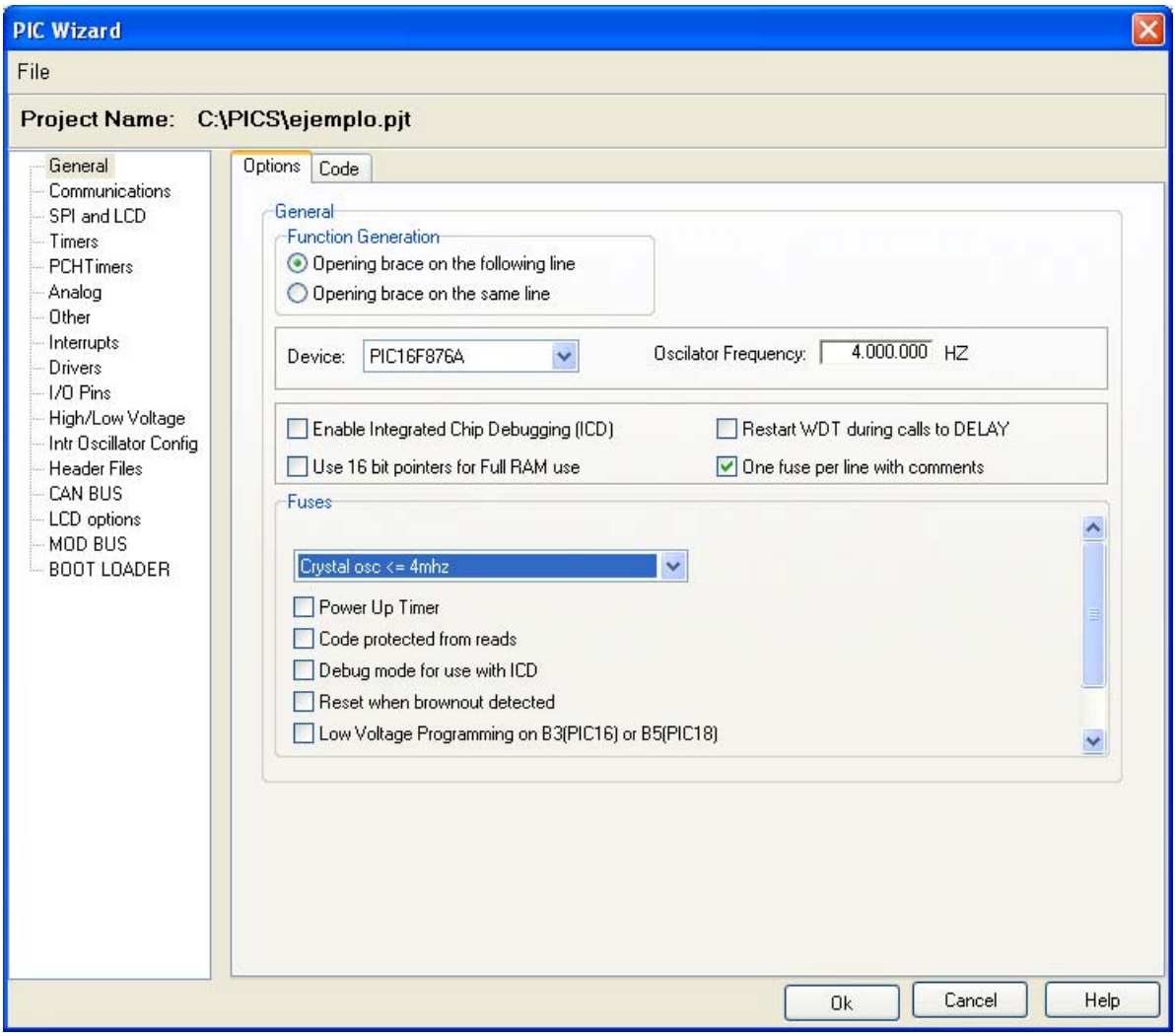

Figura 19: Selección del PIC y la frecuencia en el programa CCS

Se introducen las librerías de motores y sensores en la carpeta del proyecto y también se incluyen en la cabecera el programa.

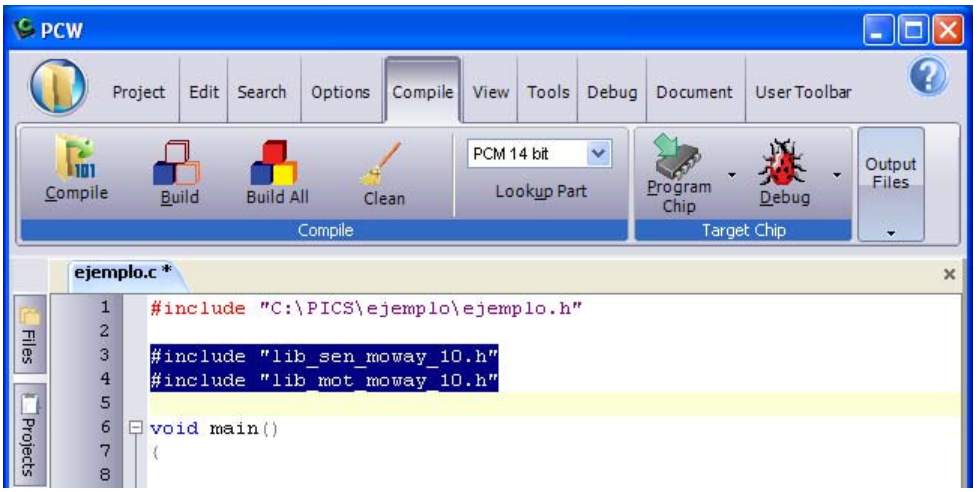

Figura 20: Programa “.c” creado por CCS incluyendo las librerías

Se escribe el programa y se compila mediante la opción “compile” del menú “Compile”. Con la opción “Built All”, el programa crea el archivo .HEX que se grabará al robot.

1.2.5.- Creación de un proyecto con MPLAB

Igual que con el otro programa, primero se crea la carpeta del proyecto y en ella se copian las librerías que se utilizan. Se crea un archivo nuevo que se guarda con extensión “.c” en la carpeta del proyecto. En él se escribe el código del programa. Se crea el proyecto mediante “PROYECT WIZARD” del menú “PROYECT”. Se selecciona el PIC, se elige como compilador CCS, se guarda el proyecto en la carpeta creada y se le añaden las librerías y el programa.

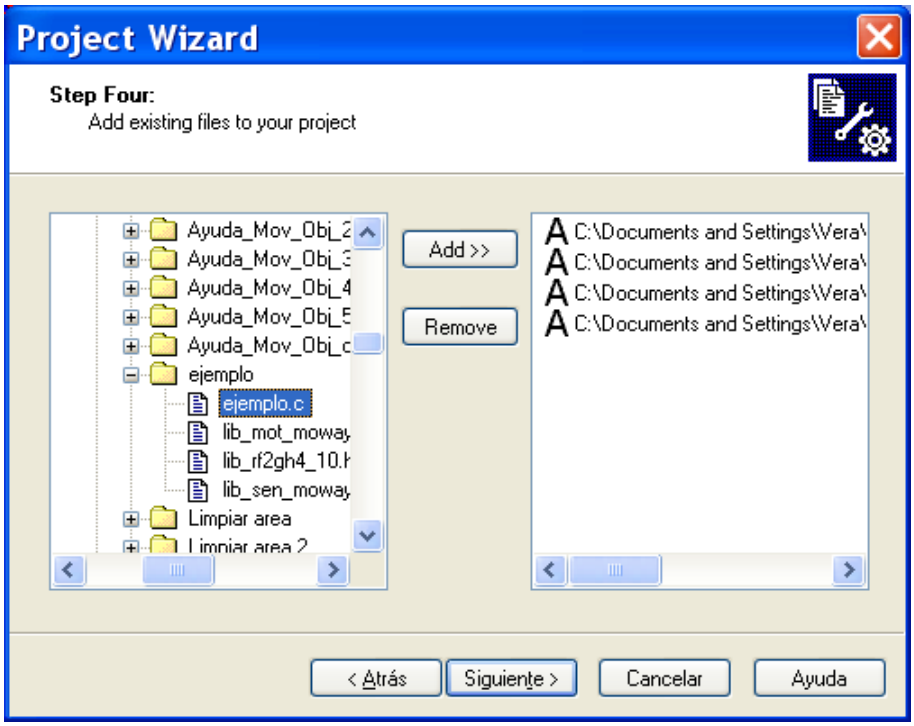

Figura 21: Se añaden las librerías y el archivo “.c” al proyecto

Se incluyen las librerías, el reloj y el modelo del PIC en la cabecera del programa. Se añade el código, se compila y se obtiene el archivo con código máquina que se grabará posteriormente en la memoria de Moway.

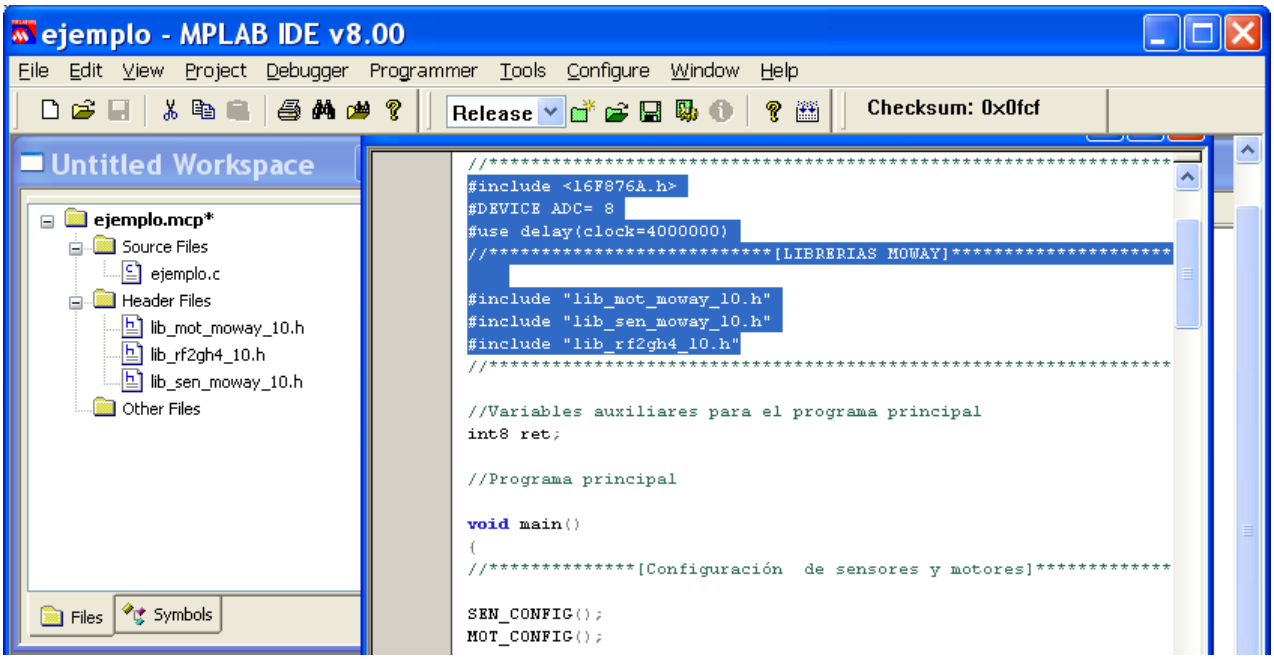

Figura 22: Código del programa con la cabecera

1.3.- ALGUNOS EJEMPLOS DE PROGRAMAS

En esta sección se muestran algunos ejemplos de programas que se distribuyen junto con Moway, y que permiten conocer el manejo de los motores y los sensores. Los códigos están disponibles en el CD adjunto a la memoria del proyecto.

Moway dispone de 2 librerías, una para el manejo de los sensores y otra para controlar los motores. La librería de sensores recopila las funciones para la lectura de los sensores y el gobierno de los LEDS y la de motores recoge las funciones que se encargan de dominar los motores y, por tanto, el movimiento del robot.

Las librerías están disponibles tanto en ensamblador como en C. Para ensamblador existen 2 versiones, una que sitúa las variables en una posición determinada de la memoria de datos (versión 10) y otra que permite situar las variables en cualquier otra zona (versión 11). Las librerías se denominan: “lib_sen_moway_10.inc” (o “lib_sen_moway_11.inc”) y “lib_mot_moway_10.inc” (o “lib_mot_moway_11.inc”), para el caso de ensamblador, y “lib_sen_moway_10.h” y “lib_mot_moway_10.inc”, para el caso de C. El código de estas librerías se ajunta en el CD que acompaña a esta memoria.

1.3.1.- Sensores y motores

El paquete de instalación proporciona 8 proyectos compilados: 3 para probar los sensores, 3 para ver el funcionamiento de los motores y 2 para manejar sensores y motores.

1.3.1.1- Proyectos que prueban los sensores

Los programas “ASM_SEN_01”, “ASM_SEN_02” y “CCS_SEN_01” se encargan de encender los LEDS inferiores si detectan obstáculos.

- “ASM_SEN_01”. Su código está en ensamblador y utiliza las librerías con variables en posición fija.
- “ASM_SEN_02”. Su código también está en ensamblador pero emplea librerías reubicables.
- “CCS_SEN_01”. Emplea el lenguaje C.

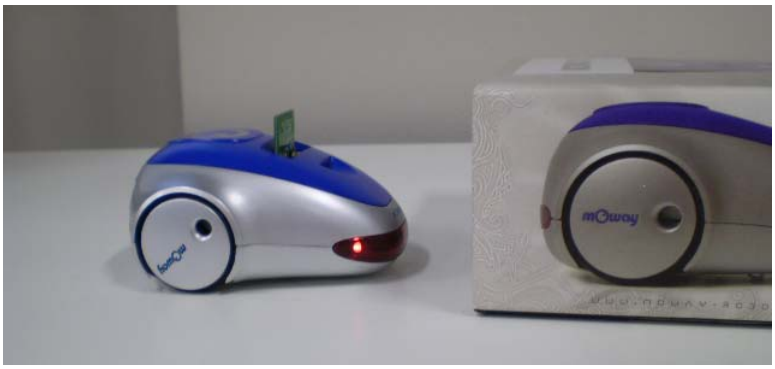

Figura 23: Moway con el programa de prueba de los sensores

Código en ensamblador

```
        list p=16F876A
;*      Definición de los registros internos del PIC
#include “P16F876A.INC”
;*      Vector de reset
        org             0x00
        goto  INIT
;*      Memoria de programa *
        org             0x05
;*************************[LIBRERÍAS DE MOWAY]******************
#include “lib_sen_moway_10.inc”
#include “lib_sen_moway_11.inc”
;*********************************[MAIN]***************************
INIT
        ;Se configuran los sensores
        call   SEN_CONFIG
MAIN
   ;Parpadeo del Led verde
        call   LED_TOP_GREEN_ON_OFF
BUCLE
        ;Se comprueba los sensores de 31ibrerías
   call        SEN_OBS_DIG
        ;Si hay un 31ibrerías en la derecha se enciende el LED derecho
        btfsc  SEN_OBS_R,0
        call   LED_R_ON
        btfss  SEN_OBS_R,0
        call   LED_R_OFF
        ;Si hay un 31ibrerías en la izquierda se enciende el LED izquierdo
        btfsc  SEN_OBS_L,0
        call   LED_L_ON
        btfss  SEN_OBS_L,0
        call   LED_L_OFF
```

```
      goto  BUCLE
END;**************************************************************************
```

Código en C

```
#include <16F876A.h>                    //Microcontrolador de Moway
#DEVICE ADC= 8
#use delay(clock=4000000)
//*****************************[LIBRERÍAS DE MOWAY]*******************
#include "lib_sen_moway_10.h"
//************************************[MAIN]****************************
void main()
{
  //Se configuran los sensores
  SEN_CONFIG();
  //Se espera 1s
  delay_ms(1000);
  //Parpadeo del Led verde
  LED_TOP_GREEN_ON_OFF();
  while(1){
    //Se comprueba los sensores de 32ibrerías
    SEN_OBS_DIG();
    //Si hay un 32ibrerías en la izquierda se enciende el LED izquierdo
    if(SEN_OBS_L==1)
      LED_L_ON();
    else
      LED_L_OFF();
    //Si hay un 32ibrerías en la derecha se enciende el LED derecho
    if(SEN_OBS_R==1)
      LED_R_ON();
    else
      LED_R_OFF();
  }}//*************************************************************************
```

1.3.1.2- Proyectos que prueban los motores

Los ficheros "ASM_MOT_01", "ASM_MOT_02" y "CCS_MOT_01" contienen una serie de movimientos para comprobar el funcionamiento de los motores. Los movimientos son los siguientes:

- Movimiento hacia adelante al máximo de velocidad durante 3 segundos.
- Recorrer hacia atrás 170mm. al 15% de la velocidad.
- Cambio de la velocidad del motor izquierdo al 90% recorriendo 170mm. hacia atrás.
- Cambio de la velocidad del motor derecho al 80% y del izquierdo al 40%, ambos hacia adelante durante 3 segundos.
- Rotación a la derecha al 80% de velocidad respecto al centro y hacia adelante durante 3 segundos.
- Giro 180,72º respecto a la rueda la izquierda al 20% de la velocidad y hacia atrás.
- Curva a derechas al 50% con un radio de 10 hacia adelante durante 3 segundos.
- Curva a izquierdas hacia atrás al 80% con un radio de 15 durante 170mm.
- Recto al 100% hacia atrás indefinidamente.
- Fin del movimiento tras 3 segundos.

- "ASM_SEN_01". Su código está en ensamblador y utiliza las librerías con variables en posición fija.
- "ASM_SEN_02". Su código también está en ensamblador pero emplea librerías reubicables.
- "CCS_SEN_01". Emplea el lenguaje C.

Código en C

```
  #include <16F876A.h>                    //Microcontrolador de Moway
  #DEVICE ADC= 8
  #use delay(clock=4000000)
//*********************[LIBRERÍAS DE MOWAY]*************************
  #include "lib_mot_moway_10.h"            //Se incluye la librería de motores
//************************************[MAIN]*************************
void main()
  {   //Se configura el microcontrolador para mandar comandos al motor
  MOT_CONFIG();
while(1){
  Delay_ms(2000);
  //Recto adelante al 100% de velocidad 3 segundos (100ms x 30)
  MOT_STR(100, FWD, TIME, 30);
  while(!input(MOT_END)){}
  //Recto hacia atrás al 15% de velocidad 170mm (1.7mm x 100)
  MOT_STR(15, BACK, DISTANCE, 100);
  while(!input(MOT_END)){}
  //Cambiar velocidad (90% atrás) al motor izquierdo y hacer una distancia de 170 mm
  //(1.7mm //x 100)
  MOT_CHA_VEL(90, BACK, LEFT, DISTANCE, 100) ;
  while(¡input(MOT_END)){}
  //Cambiar velocidad (80% adelante) al motor derecho durante 3 segundos
  //(100ms x 30)
  //Cambiar velocidad (40% adelante) al motor izquierdo durante 3 segundos
  //(100ms x 30)
  MOT_CHA_VEL(80, FWD, RIGHT, TIME, 30) ;
  MOT_CHA_VEL(40, FWD, LEFT, TIME, 30) ;
  while(¡input(MOT_END)){}
  //Rotación respecto al centro a la derecha al 80% de velocidad durante 3 segundos
  //(100ms x 30)
  MOT_ROT(80, FWD, CENTER, RIGHT, TIME, 30) ;
  while(¡input(MOT_END)){}
  //Rotación respecto la rueda izquierda al 20% de velocidad 180º  (360º/(100/50) = 180
  MOT_ROT(20, BACK, WHEEL, LEFT, ANGLE, 50) ;
  while(¡input(MOT_END)){}
  //Curva a derechas al 50% con un radio de 10 durante 3 segundos
  //(100ms x 30)
  MOT_CUR(50, FWD, 10, RIGHT, TIME, 30) ;
  while(¡input(MOT_END)){}
  // Curva a izquierdas al 80% con un radio de 15 durante 170mm
  //(1.7mm //x 100)
  MOT_CUR(80, BACK, 15, LEFT, DISTANCE, 100) ;
  while(¡input(MOT_END)){}
  //Recto hacia atrasa al 100% indefinidamente
  MOT_STR(100, BACK,TIME,0);
  //Comprobacionde del tiempo transcurrido durante 3 segundos
  MOT_FDBCK(STATUS_T);
  while(MOT_STATUS_DATA_0<30){
  MOT_FDBCK(STATUS_T) ;
  }     //Parada
  MOT_STOP() ;
}}//*********************************************************************
```

1.3.1.3- Proyectos que prueban sensores y motores

Los programas “Moway_fist_project_ASM” y “Moway_firsr_project_CCS” se encargan de que el robot vaya recto al 70% de su velocidad hasta que encuentra un obstáculo. Cuando no puede seguir recto enciende el led inferior del lado en el que encuentra el obstáculo y rota hacia adelante a la derecha y respecto al centro 180.72º, y luego sigue recto. El primero está escrito en ensamblador y el segundo en C.

Código en ensamblador

```
        list p=16F876A
;*      Definición de los registros internos del PIC
#include “P16F876A.INC”
;*      Vector de reset
        org         0x00
        goto  INIT
;*      Memoria de programa *
        org         0x05
;*      Incluir las librerías de Moway
#include “lib_mot_moway_10.inc”
#include “lib_sen_moway_10.inc”
INIT
        ;Se configuran los 34ibrerí
        call   SEN_CONFIG
;*****************************[MAIN PROGRAM]************************
MAIN
        ;Parpadeo del Led verde
        call   LED_TOP_GREEN_ON_OFF
        ;Se mueve hacia adelante
        movlw           .70
        movwf           MOT_VEL
        movlw           .0
        movwf           MOT_T_DIST_ANG
        bsf             MOT_CON,FWDBACK
        bsf             MOT_CON,COMTYPE
        call        MOT_STR
BUCLE
        ;Se comprueba los sensores de 34ibrerías
        call        SEN_OBS_DIG
        ;Si hay un 34ibrerías en la derecha se enciende el LED derecho
        btfsc       SEN_OBS_L,0
        call        OBS_SI
        ;Si hay un 34ibrerías en la izquierda se enciende el LED izquierdo
        btfsc       SEN_OBS_R,0
        call        OBS_SI
        goto  BUCLE
OBS_SI
        call        LED_TOP_RED_ON_OFF
        ;Rotación a la derecha respecto al centro al 70% de velocidad 180.72º
        movlw           .70
        movwf           MOT_VEL
        movlw           .50
        movwf           MOT_T_DIST_ANG
        movlw           0x01
        movwf           MOT_CENWHEEL
        bsf             MOT_CON,FWDBACK
        bcf             MOT_CON,COMTYPE
        bsf             MOT_CON,RL
        call        MOT_ROT
        ;Hasta que el comando no termine no se hace nada
```

```
      btfss       MOT_END
      goto        $-1
      ;Se mueve hacia adelante
      movlw             .70
      movwf             MOT_VEL
      movlw             .0
      movwf             MOT_T_DIST_ANG
      bsf               MOT_CON,FWDBACK
      bsf               MOT_CON,COMTYPE
      call        MOT_STR
      return
END;************************************************************************
```

Figura 24: Primer programa. Moway gira a la derecha al encontrar obstáculo

Código en C

```
#include “Moway_first_project_CCS.h”
#include “lib_sen_moway_10.h”
#include “lib_mot_moway_10.h”
void main()
{
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_spi(FALSE);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);
  // TODO: USER CODE!!
  //Se configuran los sensores
  SEN_CONFIG();
  //Se espera 1s
  delay_ms(1000);
  //Parpadeo del Led verde
  LED_TOP_GREEN_ON_OFF();
  //Se mueve hacia adelante
  MOT_STR(70,FWD,TIME,0);
  while(1){
    //Se comprueba los sensores de 35ibrerías
    SEN_OBS_DIG();
    //Si hay un 35ibrerías en la izquierda se enciende el LED izquierdo y rota
    if(SEN_OBS_L==1){
      LED_L_ON();
      MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,50);
      while(!input(MOT_END)){}
      MOT_STR(70,FWD,TIME,0);
    }
    else
      LED_L_OFF();
    //Si hay un 35ibrerías en la derecha se enciende el LED derecho y rota
    if(SEN_OBS_R==1){
```

```
      LED_R_ON();
      MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,50);
      while(!input(MOT_END)){}
      MOT_STR(70,FWD,TIME,0);
    }
    else
      LED_R_OFF();
} }
```

1.3.2- Prácticas propuestas en la página web de Moway

En la página web de Moway, se pueden encontrar 4 prácticas para probar el robot y familiarizarnos con las funciones de las librerías. Tras comprobar el funcionamiento de estos programas, se realizan algunas modificaciones propuestas.

1.3.2.1- El encierro

Este programa se encarga de que Moway permanezca dentro de un recinto. Al detectar una línea, gira y continúa su movimiento rectilíneo.

Código en C

```
#include <16F876A.h>
  #DEVICE ADC= 8
  #use delay(clock=4000000)
  //*********************[LIBRERÍAS DE MOWAY]************************
  #include “lib_mot_moway_10.h”
  #include “lib_sen_moway_10.h”
///*************************************************************************
  void main()
  {
  //************************[CONFIGURACIÓN DEL PIC]*******************
  //Retardo de 2 segundos
  Delay_ms(2000);
  //Configuración PIC para utilizar sensores y motores
  SEN_CONFIG();
  MOT_CONFIG();
//*************************************************************************
  // Primer movimiento recto hacia adelante al 100% de velocidad
  // tiempo ilimitado
  MOT_STR(100,FWD,TIME,0);
  while(1){
  //*********************[BUCLE PRINCIPAL]*****************************
  //Se comprueba los sensores de línea
  SEN_LINE_DIG();
  //Si encuentra alguna línea rota
  if(SEN_LINE_R==1 || SEN_LINE_L==1)
{
    //Rotación sobre el centro al 100% hacia la derecha 108º
    MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,30);
    //Espera hasta que se termine el movimiento
    while(¡input(MOT_END)){}
    //Sigue el movimiento rectilíneo
    MOT_STR(100,FWD,TIME,0);
  }
}
}//*************************************************************************
```

Figura 25: Encierro

Modificaciones

*** Encendido de los LEDS al encontrar la línea**

En este caso, al encontrar línea se enciende el led rojo correspondiente al lado que ha detectado y, tras girar para evitar la línea, se apaga el led.

Código añadido

```
  while(1){
  //**********************[BUCLE PRINCIPAL]*****************************
  //Se comprueba los sensores de línea
  SEN_LINE_DIG();
  //Si encuentra alguna línea rota
  if(SEN_LINE_R==1)
{
    LED_R_ON();
    //Rotación sobre el centro al 100% hacia la derecha 108º
    MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,30);
    //Espera hasta que se termine el movimiento
    while(!input(MOT_END)){}
    LED_R_OFF();
    //Sigue el movimiento rectilíneo
    MOT_STR(100,FWD,TIME,0);
  }
   if(SEN_LINE_L==1)
{
    LED_L_ON();
    //Rotación sobre el centro al 100% hacia la derecha 108º
    MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,30);
    //Espera hasta que se termine el movimiento
    while(!input(MOT_END)){}
    LED_L_OFF();
    //Sigue el movimiento rectilíneo
    MOT_STR(100,FWD,TIME,0);
  }
}
```

*** Movimiento en zigzag en lugar de en línea recta**

Se crea la variable RL que puede tomar los valores 0 ó 1. Se cambia su valor cada vez que se entra en el bucle principal. Moway va patrullando con movimientos curvos de 2 segundos cuyo sentido se va cambiando mediante la variable RL.

Código añadido

```
  int1 RL;
//***************************************************************************
  // Giramos cada vez en un sentido según la variable RL
   RL=0;
  while(1){
  RL=!RL;
   MOT_CUR(50,FWD,40,RL,TIME,2);
   while(!input(MOT_END)){}
  }
```

*** Detección de obstáculos en lugar de líneas**

En lugar de llamar a la función SEN_LINE_DIG se llama a la función SEN_OBS_DIG y si el robot encuentra obstáculo con algún sensor, gira.

Código añadido

```
  //********************[BUCLE PRINCIPAL]*****************************
  //Se comprueba los sensores de obs
  SEN_OBS_DIG();
  //Si encuentra alguna obs rota
  if(SEN_OBS_R==1 || SEN_OBS_L==1)
{
    //Rotación sobre el centro al 100% hacia la derecha 108º
    MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,30);
    //Espera hasta que se termine el movimiento
    while(!input(MOT_END)){}
    //Sigue el movimiento rectilíneo
    MOT_STR(100,FWD,TIME,0);
  }
```

1.3.2.2- Rastreador

El programa se encarga de encontrar una línea y seguirla.

Código en C

```
  #include <16F876A.h>
  #DEVICE ADC= 8
  #use delay(clock=4000000)
  //********************[LIBRERÍAS DE MOWAY]************************
  #include "lib_mot_moway_10.h"
  #include "lib_sen_moway_10.h"
//***************************************************************************
  void main()
  {
  //********************[CONFIGURACIÓN DE MOWAY]*******************
  int8 lineaDch;
  int8 lineaIzq;
  //Retardo de dos segundos
  Delay_ms(2000);
  //Configuración PIC para utilizar sensores y motores
  SEN_CONFIG();
  MOT_CONFIG();
  //Parpadeo de LED
  LED_TOP_RED_ON_OFF();
  LED_TOP_GREEN_ON_OFF();
```

```
    // Primer movimiento para encontrar la línea
  MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
  while(1){
  //*****************[SEGUIMIENTO DE LINEA]***************************
  //Se comprueba los sensores de línea
  SEN_LINE_DIG();
    // Si está en el borde, sigue recto
if(SEN_LINE_R==1 && SEN_LINE_L==0)
      {
      LED_R_ON();
      LED_L_OFF();
      MOT_STR(80,FWD,TIME,0);
      }
    //Si no ve línea, gira a la derecha
    else if(SEN_LINE_R==0 && SEN_LINE_L==0)
      {
      LED_R_ON();
      LED_L_ON();
      MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
       }
    //Si está encima de la línea, gira a la izquierda
    else if(SEN_LINE_R==1 && SEN_LINE_L==1)
      {
      LED_R_OFF();
      LED_L_OFF();
      MOT_ROT(50,FWD,CENTER,LEFT,TIME,0);
       }
    //Si está en el otro borde de la línea, gira a la derecha
    else if(SEN_LINE_R==0 && SEN_LINE_L==1)
      {
      LED_R_OFF();
      LED_L_ON();
      MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
      }
}
}//*************************************************************************
```

Modificación de la velocidad

Se cambian las velocidades para que, se detecte con el sensor que se detecte, todos los movimienos sean al 10% de la velocidad. En el caso anterior iba al 80% si se detectaba línea con el sensor derecho y al 50% en el resto de casos.

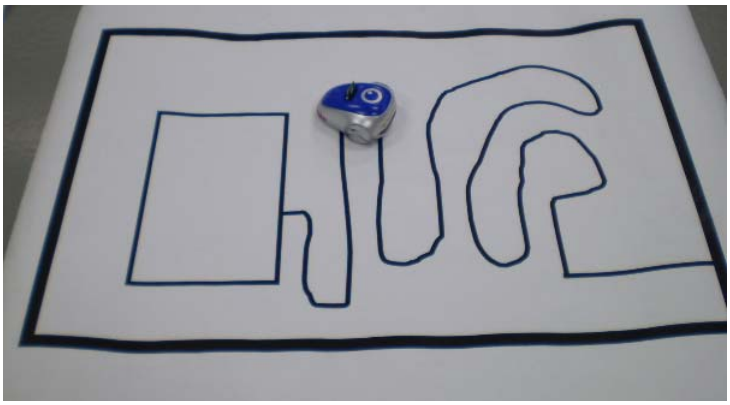

Figura 26: Rastreador

1.3.2.3- Faro

El robot busca un foco luminoso y lo sigue.

Código en C

```
  #include <16F876A.h>
  #DEVICE ADC= 8
  #use delay(clock=4000000)
  //***********************[MOWAY´S LIBRARIES]************************
  #include "lib_mot_moway.h"
  #include "lib_sen_moway.h"
//*********************************************************************
  const  int8    lightValue=90;
  static int8    rotAngle;
  static int1    rotDir;
  void main()
  {
  //***********************[PIC CONFIGURATION]************************
  Delay_ms(2000);
  //Configuración de PIC para motores y sensores
  MOT_CONFIG();
  SEN_CONFIG();
  //***********************[LIGHT TRACKING]***************************
  while(1){
    rotDir=!rotDir;
    //Aumenta el angulo si no se detecta luz para ampliar el campo de busqueda
    rotAngle+=20;
    if(rotAngle>100)
      rotAngle=20;
    //Rotación hacia la izquierda o derecha (depende de rotDir) comprobando si
    //encuetra la fuente de luz
    MOT_ROT(100,FWD,CENTER,rotDir,ANGLE,rotAngle);
    while(!input(MOT_END))
      {
            //Chequeo del estado del sensor de luz
            SEN_LIGHT();
            //Si el valor devuelto por el sensor es mayor que la variable lightValue
            //se mueve recto 100ms
             if(SEN_LIGHT_P>lightValue)
             {
                    MOT_STR(50,FWD,TIME,1);
                    while(!input(MOT_END)){}
                    rotAngle=0;
                    }
             }
      }}//*********************************************************************
```

1.3.2.4- Defender

Moway permanece dentro del recinto delimitado por las líneas del suelo y empuja todo objeto que detecta, expulsándolo fuera del recinto.

Código en C

```
  #include <16F876A.h>
  #DEVICE ADC= 8
  #use delay(clock=4000000)
  //***********************[MOWAY´S LIBRARIES]************************
```

```
  #include "lib_mot_moway.h"
  #include "lib_sen_moway.h"
//*****************************************************************************
  // ATENCIÓN
  // Sensibilidad del sensor de obstáculos. Diferente en cada robot Moway.
  // El usuario deberá ajustarlos según la sensibilidad deseada.
  #define senObsMin 50
//Función que indica si existe o no obstáculo
int1 MOWAY_SEARCH()
{
  //Comprueba el sensor de obstáculos de forma analógica
  SEN_OBS_ANALOG();
  //Si el valor de los sensores sobrepasa el mínimo retorna verdadero.
  if(SEN_OBS_L> senObsMin || SEN_OBS_R> senObsMin)
    return(true);
  else
    return(false);
}
//Función que chequea la existencia de la línea
void MOWAY_LINE()
{
  //Comprueba el valor digital de los sensores.
  SEN_LINE_DIG();
  //Si se encuentra línea rota.
  if(SEN_LINE_L==1 || SEN_LINE_R==1)
  {
    MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,30);
    while(!input(MOT_END)){}
  }
}
// Algoritmo de ataque
void MOWAY_ATTACK()
{
  //Si el obstáculo está en la izquierda
  if(SEN_OBS_L>SEN_OBS_R)
  {
    //Rota 7.2º a la izquierda y mientras rota comprueba la línea
    MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,2);
    while(!input(MOT_END))
    {
    MOWAY_LINE();
    }
  }
  else
  {
    //Rota 7.2º a la derecha y mientras rota comprueba la línea
    MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,2);
    while(!input(MOT_END))
    {
    MOWAY_LINE();
    }
  }
  //Una vez de haber rotado, golpea el objeto 11.9mm mientras comprueba la
  //línea
  MOT_STR(100,FWD,DISTANCE,7);
  while(!input(MOT_END))
  {
  MOWAY_LINE();
  }
}
void main()
  {
```

```
  //***********************[PIC CONFIGURATION]************************
  Delay_ms(2000);
  //Configuración PIC para utilizar sensores y motores
  SEN_CONFIG();
  MOT_CONFIG();
while(1){
  //Comprueba la línea
  MOWAY_LINE();
  //Si existe enemigo Moway lo ataca y sino, sigue recto patrullando.
  if (MOWAY_SEARCH())
    MOWAY_ATTACK();
  else
     MOT_STR(60,FWD,TIME,0);
}}//*****************************************************************************
```

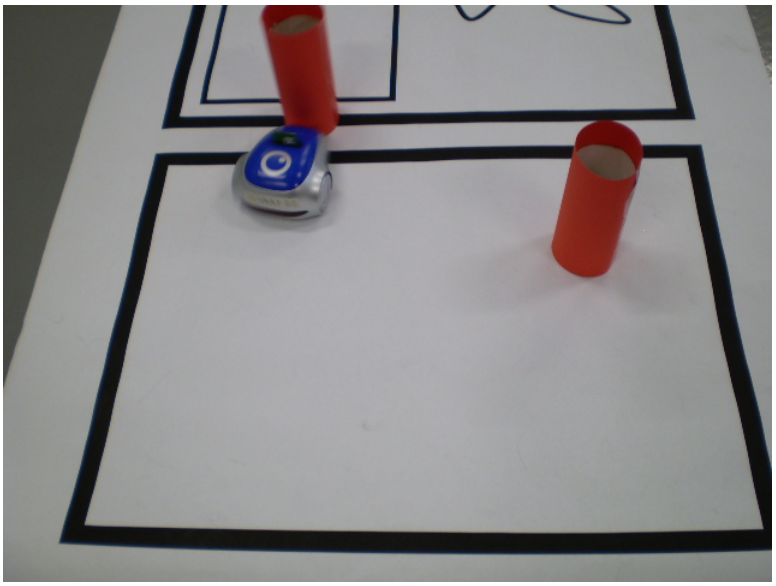

Figura 27: Defender

CAPÍTULO 2

COMUNICACIÓN POR RADIOFRECUENCIA EN EL ROBOT MOWAY

2.1.- TARJETA DE COMUNICACIONES BZI-RF2GH4

El módulo de radiofrecuencia para comunicación inalámbrica, consiste en un grupo de transceptores BZI-RF2GH4 que permiten a Moway comunicarse con otros robots o con el ordenador de forma inalámbrica bidireccional y con acuse de recibo.

Se trata de unos módulos de muy bajo coste y reducido tamaño, con antena integrada y velocidad de comunicación de hasta 2 Mbps. Su alcance es de unos 25m. en interiores a una frecuencia de 2.4GHz.

Figura 28: Robots y Bases Moway con sus tarjetas de comunicación

Se pueden utilizar hasta 128 canales de comunicación distintos lo que permite disponer de varios módulos transceptores en un mismo entorno y sin que se interfieran entre sí. Para evitar interferencias hay que dejar un espacio de valor 2 entre diferentes canales, lo que deja 32 canales útiles. Mediante una dirección de 8 bits se puede seleccionar el destinatario de la transferencia así como conocer el remitente de la misma. Se controla según el protocolo SPI integrado en la mayor parte de los microcontroladores.

El transceptor dispone de cuatro pines accesibles para el bus SPI, dos pines más para el control del módulo y otros dos para la alimentación. En la tabla 10 se muestra la relación de pines del módulo de RF y en la tabla 11 se detallan sus especificaciones técnicas.

Pines	Nº	Descripción
Vcc	1	Tensión de alimentación del módulo
Vss	2	GND
CE	3	Chip Enable
CSN	4	Chip Select del SPI (Negado)
SCK	5	Reloj del bus SPI
SDI	6	Entrada de datos del bus SPI
SDO	7	Salida de datos del bus SPI
IRQ	8	Salida interrupción (negado)

Tabla 10: Pines del módulo BZI-RF2GH4

Parámetro	Valor	Unidad
Tensión mínima de alimentación	1.9	V
Tensión máxima de alimentación	3.6	V
Potencia máxima de salida	0	dBm
Velocidad máxima de transmisión	2000	Kbps
Corriente en modo transmisión @ 0dbm potencia de salida	11.3	mA
Corriente en modo recepción @ 2000kbps	12.3	mA
Corriente en modo *Power Down*	900	nA
Frecuencia máxima del bus SPI	8	Mhz
Rango de temperatura	-40 a +85	ºC

Tabla 11: Especificaciones técnicas del módulo BZI-RF2GH4

2.2.- LIBRERÍA PARA COMUNICACIÓN POR RADIOFRECUENCIA

Junto con Moway también se distribuye la librería que controla el módulo de radiofrecuencia, "lib_rf2gh4_10.h" permite realizar fácilmente una comunicación entre microcontroladores PIC16F87x de "Microchip Technology" y el módulo BZI-F2GH4. Contiene las funciones que permiten recibir y enviar datos en tiempo real y con acuse de recibo. En la tabla 12 se encuentran todas las funciones junto a una pequeña explicación. El conjunto de las funciones de la librería ocupan 1k palabras de memoria de programa, y 33 bytes de memoria de datos. Esta librería se adjunta en el CD que viene con la memoria del proyecto.

Funciones para el módulo BZI-RF2GH4	
int RF_CONFIG(int canal, int dir)	Configura las entradas y salidas del microcontrolador[i] así como parámetros del módulo de radio.
void RF_CONFIG_SPI()	Configura las entradas y salidas del microcontrolador [ii] así como los parámetros necesarios para utilizar el bus SPI.
int RF_ON()	Activa el módulo de radiofrecuencia en modo escucha.
int RF_OFF()	Desactiva el módulo de radiofrecuencia y lo deja en modo de bajo consumo
int RF_SEND()	Envía una trama de datos (8 Bytes) a la dirección indicada.
int RF_RECEIVE()	Comprueba si se ha producido una recepción y de ser así, recoge la trama.
void RF_INT_EN()	Esta rutina habilita en el microcontrolador la interrupción externa para el módulo de radio.

[i] El usuario puede cambiar estos pines modificando la parte de la librería donde se define el patillaje
[ii] Ídem.

Tabla 12: Funciones de la librería "lib_rf2gh4_10.h"

La función **RF_SEND** devuelve tres valores distintos:

- **0** si el **envío** ha sido **correcto y** se ha recibido el **ACK**.
- **1** si **no** se ha recibido el **ACK** pero se ha enviado la trama.
- **2** si se ha producido algún **error** en el **envío**.

La función **RF_RECEIVE** devuelve también tres valores:

- **0** si se ha producido una **única recepción** y no hay ningún otro dato en la pila de recepción.
- **1** si existe una **recepción múltiple**.
- **2** si **no** se han recibido **datos**.

La función **RF_INT_EN** configura como entrada el pin RB0.

Para recibir datos existen 2 opciones:

- **Habilitar la interrupción de recepción**. De este modo se reciben los datos cada vez que se envían, interrumpiendo la acción realizada en ese instante. La información recibida es tratada en la rutina de recepción.
- **Comprobar la recepción de datos en el programa principal**. En este caso no es necesario el uso de interrupciones, a lo largo del código principal se puede llamar a la función de recepción de datos. Puede ser útil cuando el programa sólo se dedica a recibir datos. Tiene el inconveniente de que se producen retardos en la recepción si el programa principal realiza otras tareas.

En la librería se encuentran una serie de variables de utilidad que se explican a continuación:

RF_DATA_OUT

Es una variable de 8 bytes que es utilizada por la función de transmisión. En ella están los datos transmitidos.

RF_DATA_IN

Es una variable de 8 bytes que es utilizada por la función de recepción. En ella están los datos recibidos.

RF_DIR_OUT

Esta variable es de un solo byte. Indica el destino del mensaje de 8 bytes de RF_DATA_OUT. En esta variable se escribe la dirección del destinatario a la hora de realizar un envío.

RF_DIR_IN

Esta variable es de un solo byte. Indica el origen del mensaje de 8 bytes de RF_DATA_IN. En esta variable se puede ver la dirección del módulo que está enviando la información.

RF_DIR

Esta variable es de un solo byte. Indica la dirección del robot.

En las siguientes tablas se muestran los pines de configuración del módulo de RF y del SPI:

PIN RF	PIN PIC
IRQ	RB0
CSN	RB7
CE	RC2

PIN RF	PIN PIC
SCK	RC3
SDI	RC4
SDO	RC5

Tabla 13: Pines de configuración del módulo RF a la izquierda y del SPI a la derecha

2.3.- COMUNICACIÓN POR RADIOFRECUENCIA MEDIANTE EL SOFTWARE MOWAY CENTER

La ventana de radiofrecuencia del programa Moway Center posibilita el envío de datos entre Moway y el PC. Esto puede ser muy práctico a la hora de hacer pruebas con los programas creados. Por un lado se pueden mandar tramas a Moway para comprobar que realiza una determinada tarea al recibir una trama concreta. Por otra parte permite detectar en qué parte del programa se encuentra Moway y facilitar así la detección de posibles errores u omisiones en el código.

Primero se selecciona tanto la dirección como el canal que se le asigna al ordenador y se pulsa el botón de conectar RF. Hay que tener en cuenta que los valores que puede tomar el canal se encuentran entre 0x00 y 0x7F y que la dirección varía entre 0x01 y 0xFE. Se selecciona el mismo canal que se ha introducido en el programa del robot para que se pueda producir la comunicación entre ambos.

Para enviar datos del PC se introduce la dirección del robot con el que se quiere establecer la comunicación. Si la dirección es la 0x00, la información le llegará a todos los robots que pertenezcan al mismo canal. Se rellena el vector de datos a enviar, considerando que en cada byte se pueden enviar números hexadecimales que van de 0x00 a 0xFF (de 0 a 255) y que el byte de la izquierda es el menos significativo (el byte de la izquierda corresponde a la posición 0 del vector de datos de salida y el byte de la derecha, a la posición 7). El envío se efectúa al pulsar el botón de envío. A través del envío de argumentos desde el PC se puede comprobar el funcionamiento de los algoritmos antes de la interactuación de varios Moways con programas diferentes. Así se puede verificar cada programa independientemente, lo que facilita la programación.

En la ventana de recepción se muestra la información recibida. En ella aparece la dirección del emisor, el dato transmitido y la hora. En este caso también se recibe un vector de 8 bytes con valores hexadecimales con la posición 0 a la izquierda y la posición 7 a la derecha. Con esta comunicación se logra localizar en un tiempo reducido los errores u omisiones del código. Esto se logra colocando envíos de datos diferentes en las distintas partes del programa.

En la siguiente imagen podemos apreciar un ejemplo de comunicación. Al PC se le ha asignado la dirección 2 del canal 40 y envía datos a la dirección 1. En este caso el robot devuelve la trama enviada por el PC. Se observa que el ordenador ha recibido varios datos y el último es el que acaba de enviar: 11 00 11 00 11 00 00 11.

Figura 29: Ejemplo de comunicación Moway-PC

CAPÍTULO 3

PRIMEROS PROGRAMAS

Este capítulo es una introducción a la programación de Moway en C empleando el módulo BZI-RF2GH4. Para entrar en el mundo de la comunicación por radiofrecuencia, primero se analizan una serie de programas ejemplo que se distribuyen junto a Moway; y, posteriormente, se explican algunos programas que se han creado y algunas funciones de utilidad que se han añadido a las librerías proporcionadas con el software del microbot. Todos los códigos que aparecen en este capítulo se pueden consultar en el CD que acompaña a esta memoria.

3.1.- PROYECTOS QUE COMPRUEBAN EL FUNCIONAMIENTO DEL MÓDULO DE RADIOFRECUENCIA

En este apartado se va a comprobar el funcionamiento del módulo de radiofrecuencia. Para ello se analizan y ejecutan una serie de programas que se adjuntan en el paquete de instalación de Moway Center.

3.1.1.- Módulo RF

A continuación se explican los programas “CCS_RF_01.c” y “CCS_RF_02.c” para conocer mejor las librerías de radiofrecuencia.

“CCS_RF_01.c”

Este programa se encarga de **enviar** la trama **“00 00 00 00 00 00 00 00” cada 5 segundos** a la dirección 2 del canal 0x40. Tras el envío, tiene el código necesario para comprobar si ha habido algún error en el envío y si se ha recibido el ACK.

En este programa se configuran los módulos RF y SPI, se habilita la interrupción de recepción de datos y tiene implementada la rutina de **recepción con interrupción**, en la que se puede añadir el código para procesar los datos recibidos. Para que se pueda producir la comunicación se asigna el canal 0x40 y la dirección 1.

Código

```
//Variables auxiliares para el programa principal
int8 ret;
int8 i;
//--------------[Rutina de recepción con interrupcion]--------------------------
//Interrupción del módulo RF
#int_ext
void int_externo()
{
      while(RF_RECEIVE()!=2)
        {
      //Tratar datos
      }
}
//--------------------------------------------------------------------------------
//Programa principal
void main()
{
//-----------[Rutina de configuración y activación con interrupción]------------
  //Habilitar interrupciones
  RF_INT_EN();
  //Configurar módulos SPI del PIC
```

```
  RF_CONFIG_SPI();
  //Configurar módulo RF (canal y dirección)
  if(RF_CONFIG(0x40,0x01)==0)
{
  //Módulo no configurado
  }
  //Activar el módulo RF
  if(RF_ON()==0)
{
  //Módulo no inicializado
  }
//-------------------------------------------------------------------------
while(1){
     Delay_ms(5000);
//-------------------------[Rutina de envio de datos]--------------------------
     //Preparación de la dirección del receptor
     //y de los datos.
     RF_DIR_OUT=0x02;
     for(i=0;i<8;i++)
     {
     RF_DATA_OUT[i]=0x00;
     }
     //Envio de los datos
     ret=RF_SEND();
     if(ret==0)
        {
        //Envio realizado y ACK recibido
        }
     else if(ret==1)
        {
        //Envio realizado y ACK no recibido
        }
     else
        {
        //Envio no realizado
        }//-----------------------------------------------------------------------
   }
}//-------------------------------------------------------------------------
```

“CCS_RF_02.c”

Este programa **comprueba** todo el tiempo **si se ha recibido** alguna **trama**. En caso de que se haya recibido, **se reenvía** a la dirección 2, pudiendo comprobar si se han producidos fallos en el envío. En este caso **no hay interrupción por recepción**, ya que siempre se está a la escucha. Como en todos los programas en los que se quiere establecer una comunicación, se configuran los módulos SPI y RF adjudicando la dirección 1 del canal 0x40.

Código

```
//Variables auxiliares para el programa principal
int8 ret;
int8 i;
//Programa principal
void main()
{
//-----------[Rutina de configuración y activación sin interrupción]------------
  //Configurar módulos SPI del PIC
  RF_CONFIG_SPI();
```

```
  //Configurar módulo RF (canal y dirección)
  if(RF_CONFIG(0x40,0x01)==0)
{
  //Módulo no configurado
  }
  //Activar el módulo RF
  if(RF_ON()==0){
  //Módulo no inicializado
  }
//-------------------------------------------------------------------------
while(1)
{
   while(RF_RECEIVE()!=2){
         //--------------------[Rutina de envio de datos]----------------------
          RF_DIR_OUT=0x02;
          for(i=0;i<8;i++)
               RF_DATA_OUT[i]=RF_DATA_IN[i];
          //Envio de los datos
          ret=RF_SEND();
          if(ret==0)
        {
          //Envio realizado y ACK recibido
          }
          else if(ret==1)
        {
          //Envio realizado y ACK no recibido
          }
          else
        {
          //Envio no realizado
          }
      }
  }
} //---------------------------------------------------------------------
```

Experimentos con PC

Se comprueba el funcionamiento de ambos programas con la ayuda de la ventana de radiofrecuencia de Moway Center. Se observa que “CCS_RF_01.c” transmite cada 5 segundos un vector de ceros y “CCS_RF_02.c” reenvía el argumento mandado por el ordenador.

3.1.2.-Carrera de relevos

Los programas “Moway_relevos_00.c” y “Moway_relevos_01.c” simulan una carrera de relevos. Para comprobar el funcionamiento de estos programas se necesita una línea cerrada por la que los microbots puedan avanzar. Mientras uno de los robots comienza a seguir la línea, el otro permanece parado. Cuando ambos se encuentran, intercambian sus papeles, el Moway que estaba parado comienza a recorrer la línea y el otro espera a recibir el relevo. En las figuras 30, 31 y 32 se puede ver un esquema de la carrera de relevos.

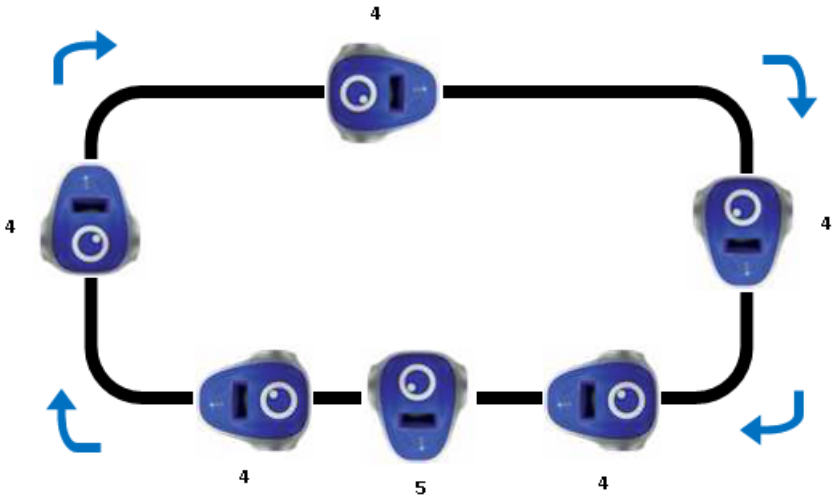

Figura 30: Moway 4 sigue línea mientras Moway 5 espera el relevo

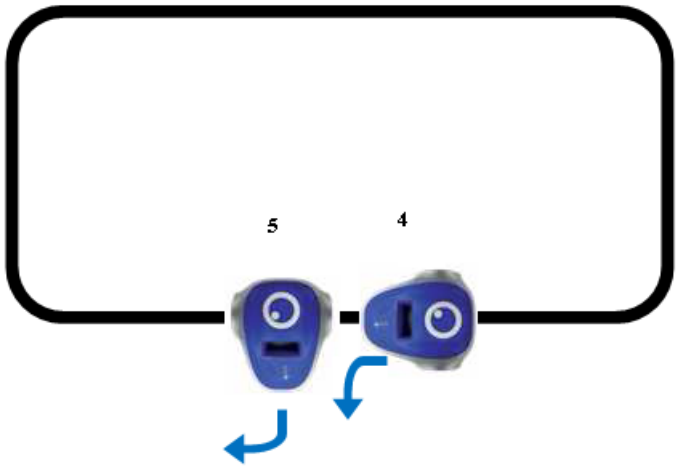

Figura 31: Los Moways se pasan el relevo y giran al encontrarse

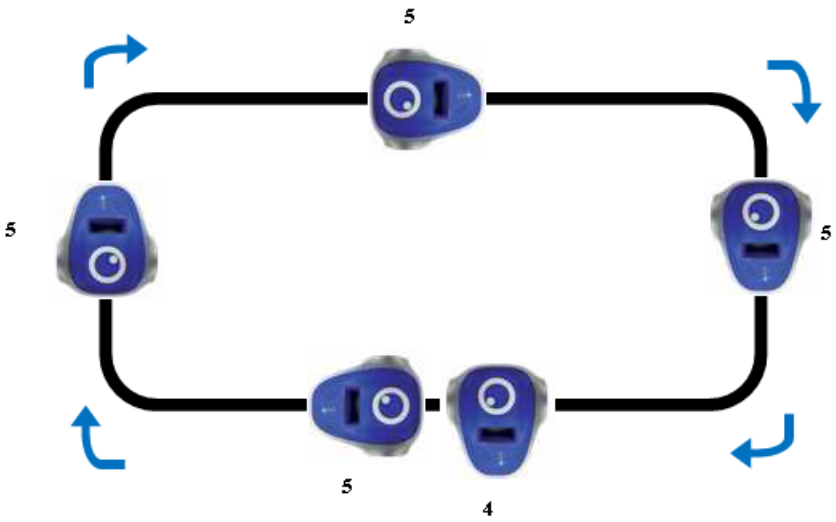

Figura 32: Moway 5 recorre la línea y Moway 4 espera su turno

“Moway_relevos_00.c”

El programa se divide en tres partes bien diferenciadas: la rutina de recepción con interrupción, el relevo y el seguimiento de línea. A continuación se explica en qué consisten cada uno de estos segmentos.

Rutina de recepción con interrupción

Al **detectar** la transmisión del valor **0xFF en la posición 0** del vector de variables de entrada, **se pone a 1** la variable **“star”,** que indica que se ha recibido **el relevo** y debemos comenzar el movimiento.

Relevo

Se comprueba la existencia de obstáculo. Cuando **se detecta** el otro **robot,** se **gira 90º a la izquierda** sobre el centro, **se manda 0xFF en la posición 0** del vector de variables de salida, se enciende y apaga el LED rojo superior, y se mantiene a la **espera** de recibir el relevo con el LED superior vede parpadeando. **Cuando llega el relevo**, se produce un **giro de 90º a la derecha**.

Seguimiento de línea

Primero se comprueban los sensores de línea y según el valor de los sensores se ejecuta un determinado movimiento para rastrear la línea:

- Si detecta con el **sensor derecho** → Movimiento **hacia adelante** al 80%.
- Si **ningún sensor** detecta→ **Giro a la derecha.**
- Si detecta con el **sensor izquierdo** → El robot también **gira a la derecha.**
- Si detecta con **ambos sensores**→ **Giro a la izquierda.**

El programa tiene un contador que se va aumentando cada vez que no se producen cambios en los valores de los sensores. En el caso de que este contador llegue a 75, se producen giros con menor radio, y si asciende hasta 150, los giros se realizan respecto al centro del robot.

Código

```
  static int1 start;
  //Dirección propia
  int8 yo=0x01;
  //Dirección del siguiente robot
  int8 siguiente=0x02;
  //Canal de comunicaciones
  int8 canal=0x40;
  //**************[Rutina de recepción con interrupcion]**********************
  //Interrupción del módulo RF
#int_ext
  void int_externo()
{
     while(RF_RECEIVE()!=2)
      {
      //Iniciar seguimiento de línea
      if(RF_DATA_IN[0]==0xFF)
         start=1;
     }
  } //*****************************************************************************
```

```
  void main()
  {
  //*****************************[PIC CONFIGURATION]*****************************
  int8 lineaDch;
  int8 lineaIzq;
  static int16 count;
  Delay_ms(2000);
  //Configuración PIC para utilizar sensores y motores
  SEN_CONFIG();
  MOT_CONFIG();
  //Configuración RF
  RF_CONFIG_SPI();
  RF_CONFIG(canal,yo);
  RF_INT_EN();
  RF_ON();
  //Parpadeo de LED
  LED_TOP_RED_ON_OFF();
  LED_TOP_GREEN_ON_OFF();
  while(1){
  //*****************************[Relevo]****************************************
  //Se comprueba los sensores de obstáculos
  SEN_OBS_DIG();
  //Si se detecta obstáculo se manda comando de rotar y se pasa relevo
  if(SEN_OBS_R==1 || SEN_OBS_L==1)
{
    start=0;
    MOT_ROT(50,FWD,CENTER,LEFT,ANGLE,25);
    RF_DIR_OUT=siguiente;
    RF_DATA_OUT[0]=0xFF;
    //Se manda relevo
    while(RF_SEND()!=0)
      {
    LED_TOP_RED_ON_OFF();
    }
    //Se espera relevo
    while(!start)
      {
    LED_TOP_GREEN_ON_OFF();
    }
    MOT_ROT(50,FWD,CENTER,RIGHT,ANGLE,25);
    while(!input(MOT_END)){}
  }
 //***********************[Seguimiento linea]***********************************
  //Se comprueba los sensores de línea
  SEN_LINE_DIG();
  //Si cambian de estado se cambia de movimiento
  //Zona movimientos suaves
  if(SEN_LINE_R!=lineaDch || SEN_LINE_L!=lineaIzq)
 {
       //Nuevo estado
        lineaDch=SEN_LINE_R;
        lineaIzq=SEN_LINE_L;
        count=0;
       if(lineaDch==1 && lineaIzq==0)
       {
              LED_L_ON();
               LED_R_OFF();
              MOT_STR(80,FWD,TIME,0);
       }
       else if(lineaDch==0 && lineaIzq==0)
       {
               LED_L_ON();
```

```
            LED_R_ON();
             MOT_CHA_VEL(30,FWD,RIGHT,TIME,0);
            MOT_CHA_VEL(100,FWD,LEFT,TIME,0);
      }
     else if(lineaDch==1 && lineaIzq==1)
      {
            LED_L_OFF();
            LED_R_OFF();
            MOT_CHA_VEL(30,FWD,LEFT,TIME,0);
            MOT_CHA_VEL(100,FWD,RIGHT,TIME,0);
      }
     else if(lineaDch==0 && lineaIzq==1)
      {
            LED_R_ON();
            LED_L_OFF();
            MOT_CHA_VEL(30,FWD,RIGHT,TIME,0);
            MOT_CHA_VEL(100,FWD,LEFT,TIME,0);
      }
  }
  else
{
   //Contador de tiempo en el mismo estado
   count++;
   //Zona movimientos rápidos
   if(count==75)
      {
            if(lineaDch==1 && lineaIzq==0)
            {
                  LED_L_ON();
                  LED_R_OFF();
                  MOT_STR(80,FWD,TIME,0);
            }
            else if(lineaDch==0 && lineaIzq==0)
            {
                  LED_L_ON();
                  LED_R_ON();
                  MOT_CHA_VEL(1,FWD,RIGHT,TIME,0);
                  MOT_CHA_VEL(100,FWD,LEFT,TIME,0);
            }
            else if(lineaDch==1 && lineaIzq==1)
            {
            LED_L_OFF();
            LED_R_OFF();
            MOT_CHA_VEL(1,FWD,LEFT,TIME,0);
            MOT_CHA_VEL(100,FWD,RIGHT,TIME,0);
            }
            else if(lineaDch==0 && lineaIzq==1)
            {
                  LED_R_ON();
                  LED_L_OFF();
                  MOT_CHA_VEL(1,FWD,RIGHT,TIME,0);
                  MOT_CHA_VEL(100,FWD,LEFT,TIME,0);
            }
     }
     else if(count==150)
      {
       //Zona movimientos extremos
       if(lineaDch==1 && lineaIzq==0)
      {
            LED_L_ON();
            LED_R_OFF();
             MOT_STR(80,FWD,TIME,0);
```

```
        }
        else if(lineaDch==0 && lineaIzq==0)
        {
                LED_L_ON();
                 LED_R_ON();
                MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
        }
        else if(lineaDch==1 && lineaIzq==1)
        {
                LED_L_OFF();
                LED_R_OFF();
                MOT_ROT(50,FWD,CENTER,LEFT,TIME,0);
        }
        else if(lineaDch==0 && lineaIzq==1)
        {
                LED_R_ON();
                 LED_L_OFF();
                MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
        }
    }
}}}//*****************************************************************************
```

“Moway_relevos_01.c”

El código es similar al anterior, la única diferencia es que, en este caso, primero se produce la espera del relevo y, tras obtenerlo, comienza el seguimiento de línea.

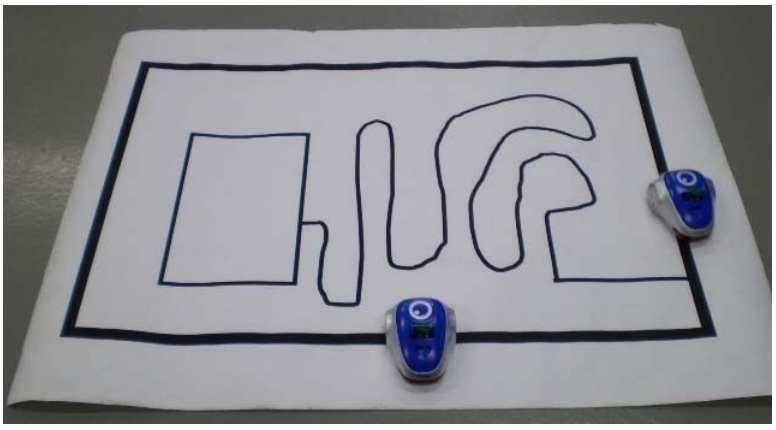

Figura 33: Carrera de relevos

Experimentos con ambos programas

Se observa que para un buen funcionamiento, el robot que espera debe ponerse perpendicular a la línea y su sentido debe ser el que le permita seguir la línea sin encontrarse con el robot que le acaba de pasar el relevo, ya que, si sucede esto, ambos se quedan girando y enviándose el relevo indefinidamente.

3.1.3.- Teleoperación de Moway por RF con Moway_RC_Center_v1.0.0

La aplicación Moway_RC_Center_v1.0.0 es un entorno gráfico que permite teleoperar a Moway. Para ello es necesario cargar el código del programa “Moway_RC_1.0.0.c”. Este algoritmo permite controlar los movimientos y los LEDS

del robot solamente apretando los botones que aparecen en pantalla y, a la vez, muestra el valor de los sensores y el cuentakilómetros.

En la ventana del programa se pueden encontrar 3 zonas importantes como se pueden apreciar en la figura 34:

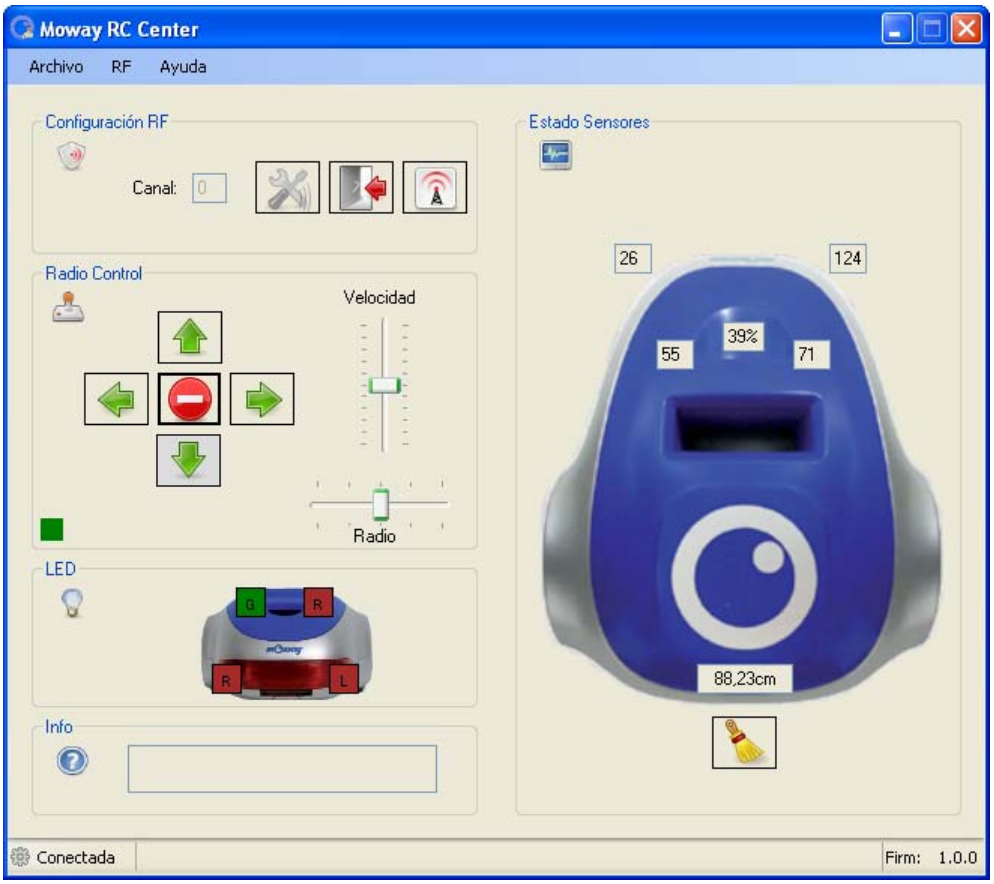

Figura 34: Moway RC Center

Radio control

Esta zona permite **dirigir los movimientos** del robot. Con las flechas verdes se selecciona la dirección de movimiento y mediante las barras de la derecha se pueden variar tanto la velocidad como el radio de giro.

LED

Pinchando sobre los botones de los LEDS se pueden **encender y apagar los LEDS** de Moway.

Estado Sensores

Este apartado muestra la **lectura de los sensores** (los de obstáculos se encuentran delante de la figura del robot, los de línea sobre la parte delantera en los laterales y el de luz en el centro de la parte delantera) **y** la **distancia recorrida** (en la parte trasera de la figura y en centímetros).

“Moway_RC_1.0.0.c”

Este programa contiene el código que permite que Moway ejecute los comandos ordenados por el PC. A continuación se explican un poco sus tres secciones principales:

Definición de comandos

Al principio del programa **se asigna un valor a cada movimiento**, a cada **acción** que se puede realizar **sobre los LEDS y** a los **cambios** que se pueden producir en las diferentes variables (velocidad, radio, canal y reset).

Rutina de interrupción

Si se recibe alguna trama, se comprueba el valor recibido y **se ejecuta el movimiento o la acción correspondiente**.

Monitorización de Moway

En la monitorización **se almacenan los valores** de los sensores y el cuentakilómetros **y se envían** a la dirección 2. En la posición 0 del vector de datos de salida se almacena el valor del sensor de obstáculo derecho; en la posición 1, el valor del sensor de obstáculo izquierdo; en la posición 2, el valor del sensor de línea derecho; en la posición 3, el valor del sensor de línea izquierdo; en la posición 4 el valor del sensor de luz; y en las posiciones 5 y 6 el valor de la variable “status”.

Código

```
  //Comandos movimiento
  #define cmd_go            0xE1
  #define cmd_back          0xE2
  #define cmd_go_left       0xE3
  #define cmd_go_right      0xE4
  #define cmd_back_left     0xE5
  #define cmd_back_right    0xE6
  #define cmd_stop          0xE7
  //Comandos LED
  #define cmd_led_l_on      0xA0
  #define cmd_led_r_on      0xA1
  #define cmd_led_green_on  0xA2
  #define cmd_led_red_on    0xA3
  #define cmd_led_l_off     0xA4
  #define cmd_led_r_off     0xA5
  #define cmd_led_green_off 0xA6
  #define cmd_led_red_off   0xA7
  //Comandos varios
  #define cmd_vel           0xB0
  #define cmd_rad           0xB1
  #define cmd_reset         0xB2
  #define cmd_canal         0xB3
  #define cmd_reset_dist    0xB4
  //Velocidad y radio inicial
  int8 RC_VEL=50;
  int8 RC_RADIO=20;
  //Canal y dirección de mOway
  int8 yo=0x01;
  int8 canal=0x00;
//--------------[Rutina de recepción con interrupcion]--------------------------
//Interrupción del módulo RF
#int_ext
```

```
void int_externo()
{
    while(RF_RECEIVE()!=2)
     {
       //Motor
       if(RF_DATA_IN[0]==cmd_go)              //Adelante
       MOT_STR(RC_VEL,FWD,TIME,0);
       else if(RF_DATA_IN[0]==cmd_back)       //Atras
       MOT_STR(RC_VEL,BACK,TIME,0);
       else if(RF_DATA_IN[0]==cmd_go_left)    //Adelante-izquierda
       MOT_CUR(RC_VEL,FWD,RC_RADIO,LEFT,TIME,0);
       else if(RF_DATA_IN[0]==cmd_go_right)   //Adelante-derecha
       MOT_CUR(RC_VEL,FWD,RC_RADIO,RIGHT,TIME,0);
       else if(RF_DATA_IN[0]==cmd_back_left)  //Atras-izquierda
       MOT_CUR(RC_VEL,BACK,RC_RADIO,LEFT,TIME,0);
       else if(RF_DATA_IN[0]==cmd_back_right) //Atras-derecha
       MOT_CUR(RC_VEL,BACK,RC_RADIO,RIGHT,TIME,0);
       else if(RF_DATA_IN[0]==cmd_stop)       //Stop
       MOT_STOP();
       //LED
       else if(RF_DATA_IN[0]== cmd_led_l_on)  //Cambio Led izquierdo
       LED_L_ON();
       else if(RF_DATA_IN[0]==cmd_led_r_on)   //Cambio Led derecho
       LED_R_ON();
       else if(RF_DATA_IN[0]==cmd_led_green_on)  //Cambio Led verde
       LED_TOP_GREEN_ON();
       else if(RF_DATA_IN[0]==cmd_led_red_on)    //Cambio Led rojo
       LED_TOP_RED_ON();
       else if(RF_DATA_IN[0]== cmd_led_l_off)    //Cambio Led izquierdo
       LED_L_OFF();
       else if(RF_DATA_IN[0]==cmd_led_r_off)     //Cambio Led derecho
       LED_R_OFF();
       else if(RF_DATA_IN[0]==cmd_led_green_off) //Cambio Led verde
       LED_TOP_GREEN_OFF();
       else if(RF_DATA_IN[0]==cmd_led_red_off)   //Cambio Led rojo
       LED_TOP_RED_OFF();
       //Varios
       else if(RF_DATA_IN[0]==cmd_vel)        //Cambio velocidad
       RC_VEL=RF_DATA_IN[1];
       else if(RF_DATA_IN[0]==cmd_rad)        //Cambio Radio
       RC_RADIO=RF_DATA_IN[1];
       else if(RF_DATA_IN[0]==cmd_reset)      //Reset
       {
       RC_VEL=50;
       RC_RADIO=20;
       LED_L_OFF();
       LED_R_OFF();
       LED_TOP_RED_OFF();
       LED_TOP_GREEN_OFF();
       }
       else if(RF_DATA_IN[0]==cmd_canal)      //Cambio de canal
       {
       canal=canal+8;
       if(canal>128)
       canal=0;
       RF_CONFIG(canal,yo);
       }
       else if(RF_DATA_IN[0]==cmd_reset_dist)  //Reset cuentakilomentros
       {
       MOT_RST(RST_KM);
       }
    }
```

```
}//-------------------------------------------------------------------------
void main()
{
//------------------------[Configuración de mOway]-----------------------------
   Delay_ms(500);
   SEN_CONFIG();
   MOT_CONFIG();
   RF_CONFIG_SPI();
   RF_CONFIG(canal,yo);
   RF_INT_EN();
   RF_ON();
//-------------------------------------------------------------------------
   LED_TOP_GREEN_ON_OFF();
   LED_TOP_RED_ON_OFF();
   Delay_ms(2000);
//---------------------[Monitorización de mOway]-------------------------------
  while(1){
   Delay_ms(1000);
   //El valor de los sensores de obstaculo se guardan en las variables de
   //salida RF
   SEN_OBS_ANALOG();
   RF_DATA_OUT[0]=SEN_OBS_R;
   RF_DATA_OUT[1]=SEN_OBS_L;
   //El valor de los sensores de línea se guardan en las variables de salida RF
   SEN_LINE_ANALOG();
   RF_DATA_OUT[2]=SEN_LINE_R;
   RF_DATA_OUT[3]=SEN_LINE_L;
   //El valor del sensor de luz se guarda en una variable de salida RF
   SEN_LIGHT();
   RF_DATA_OUT[4]=SEN_LIGHT_P;
   //El valor del cuentakilometros se guarda en una variable de salida RF
   MOT_FDBCK(STATUS_KM);
   RF_DATA_OUT[5]=MOT_STATUS_DATA_1;
   RF_DATA_OUT[6]=MOT_STATUS_DATA_0;
   //Se indica la dirección de salida
   RF_DIR_OUT=0x02;
   //Se mandan los datos
   RF_SEND();
  }
}//-------------------------------------------------------------------------
```

Experimentos

Para dirigir a Moway por RF se carga el programa, se cierra la ventana de Moway Center y se abre el programa de teleoperación. Una vez activada la comunicación por radiofrecuencia, se puede controlar el robot y leer el valor de los sensores.

3.2.- DESARROLLOS PROPIOS DE COMUNICACIÓN POR RF

Se crean los programas “ej_RF.c” y “ej2_RF.c” como primeros programas que emplean la comunicación por RF. Ambos programas intercambian datos y actúan según la información recibida. Estos programas están compuestos por la rutina de recepción de datos y la de envío de datos que se comentan a continuación.

Rutina de recepción con interrupción

Los dos programas reciben la información mediante la interrupción del módulo RF. Según al **valor de la posición 0 del vector de variables de entrada** realizan un movimiento u otro:

- Si el dato es **0x00→ se desplazan hacia adelante** 1 segundo al 20%.
- Si el dato **no es 0x00→ se desplazan hacia atrás** 1 segundo al 20%.

Rutina de envío de datos

"ej_RF.c" crea un vector de salida **de ceros y lo envía** a la dirección 2 del canal 0x40 **cada 20 segundos**. **"ej2_RF.c" crea un vector** cuyos **valores** son todos **0x01 y lo envía** a la dicción 1 del canal 0x40 **cada 5 segundos**.

En ambos programas, si se manda el vector correctamente el LED superior verde parpadea, y si hay algún problema en la transmisión, parpadea el LED superior rojo.

Código de "ej_RF.c"

```
//                      ej_RF.c
//PFC Mari Carmen Vera Sánchez
//
//Programa para comprobar el uso de la RF.
//*****************************************************************************
#include <16F876A.h>
#DEVICE ADC= 8
#use delay(clock=4000000)
//*****************************[LIBRERIAS MOWAY]*********************************
#include "lib_mot_moway_10.h"
#include "lib_sen_moway_10.h"
#include "lib_rf2gh4_10.h"
//*****************************************************************************
//Variables auxiliares para el programa principal
int8 ret;
int8 i;
//*************[Rutina de recepción con interrupcion]**************************
//Interrupción del módulo RF
#int_ext
void int_externo()
{
      while(RF_RECEIVE()!=2)
      {
            if(RF_DATA_IN[0]==0x00)
            MOT_STR(20,BACK,TIME,10); //Atras al 20% 1 seg si se recibe 0
            else
                  MOT_STR(20,FWD,TIME,10);   //Adelante al 20% 2 seg si no se recibe 0
      }
}
//Programa principal
void main()
{
//***************[Configuración  de sensores y motores]************************
SEN_CONFIG();
MOT_CONFIG();
//*************[Rutina de configuración y activación con interrupción]**********
//Habilitar interrupciones
RF_INT_EN();
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x01)==0)
```

```
{
	//Módulo no configurado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
	//Módulo no inicializado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa led sup rojo
}
while(1){
Delay_ms(5000); //Esperar 5 seg y enviar datos
//*****************************[Rutina de envio de datos]*************************
//Preparación de la dirección del receptor y de los datos.
RF_DIR_OUT=0x02;
for(i=0;i<8;i++)
	RF_DATA_OUT[i]=0x00;
//Envio de los datos
ret=RF_SEND();
//Envio realizado y ACK recibido
if(ret==0)
	LED_TOP_GREEN_ON_OFF(); //Parpadeo del led sup verde si el envio es correcto
//Envio realizado y ACK no recibido
else if(ret==1)
	LED_TOP_RED_ON_OFF();	//Parpadeo del led sup rojo si envio es incorrecto
//Envio no realizado
else
	LED_TOP_RED_ON_OFF();	//Parpadeo del led sup rojo si envio es incorrecto
}}//-------------------------------------------------------------------------------
```

Rutina de envío de datos de “ej2_RF.c”

```
//Preparación de la dirección del receptor y de los datos.
RF_DIR_OUT=0x01;
for(i=0;i<8;i++)
	RF_DATA_OUT[i]=0x01;
//Envio de los datos
ret=RF_SEND();
//Envio realizado y ACK recibido
if(ret==0)
	LED_TOP_GREEN_ON_OFF(); //Parpadeo del led sup verde si el envio es correcto
//Envio realizado y ACK no recibido
else if(ret==1)
	LED_TOP_RED_ON_OFF();	//Parpadeo del led sup rojo si envio es incorrecto
//Envio no realizado
else
	LED_TOP_RED_ON_OFF();	//Parpadeo del led sup rojo si envio es incorrecto
```

Experimentos

Se verifica el funcionamiento de los programas realizando diferentes experimentos. Por un lado interactúan un robot y el PC y, por otro, ambos robots, cada uno con uno de los algoritmos.

“ej_RF.c” con PC

Se asigna la dirección 2 y el canal 40 al módulo RF de Moway Center. La ventana de radiofrecuencia recibe cada 5 segundos la trama, y se produce el parpadeo del LED

superior verde del robot. Se comprueba que si el primer valor del dato a enviar es 00, Moway va hacia atrás, y si es distinto, va hacia adelante.

Figura 35: Comprobación del programa “ej_RF.c”

“ej2_RF.c” con PC

Como con el programa anterior, Moway se mueve hacia atrás 1 segundo cuando recibe 00 en la primera posición del vector de entradas y se mueve hacia adelante si recibe cualquier otro valor. Parpadea el LED superior derecho cada 20 segundos y se envía el dato al PC.

Figura 36: Comprobación del programa “ej2_RF.c”

“ej_RF.c” y “ej2_RF.c” en los microbots

Moway 4 se desplaza hacia atrás cada 5 segundos y Moway 5 va hacia adelante cada 20 segundos. En ambos robots el LED superior verde parpadea cuando han enviado los datos.

3.3.- MODIFICACIONES EN LAS LIBRERÍAS ORIGINALES DE MOWAY

A lo largo del diseño de los diferentes programas de este proyecto se ha necesitado mejorar las librerías proporcionadas para el manejo del microbot. Se han modificado algunas funciones existentes para poder realizar tareas complejas fácilmente y se han creado algunas funciones nuevas para reducir código y no repetirlo sucesivas veces en un mismo programa.

3.3.1.- Librería de motores

Se realizan una serie de cambios en las funciones de esta librería para que los robots puedan compartir los movimientos que realizan. A esta nueva librería la denominamos “lib_mot_new_10.h” y está disponible en el CD adjunto a la memoria.

Estudiando la librería de motores proporcionada por el fabricante, se observa que el tipo de movimiento que realiza el robot se almacena en la variable “MOT_COMAND”. A demás del tipo de movimiento, en esta variable también se pueden consultar la dirección, derecha o izquierda (RL), y el tipo de unidades de movimiento (CT), es decir, si se especifica que se mueva un tiempo, una distancia o un ángulo.

Byte MOT_COMAND y parámetros

```
//Byte MOT_COMAND              nºbit   7   6   5   4   3   2   1   0
//                                    ---------------------------------
//Comandos movimientos
#define   COM_STR       0x00   //     "0   0   0   0   0   0   CT  RL"
#define   COM_CHA_VEL    0x01   //     "0   0   0   0   0   1   CT  RL"
#define   COM_ROT       0x02   //     "0   0   0   0   1   0   CT  RL"
#define   COM_CUR       0x03   //     "0   0   0   0   1   1   CT  RL"
#define   COM_STOP      0x04   //     "0   0   0   1   0   0   CT  RL"
//Comandos información
#define   STATUS_T      0x00   //     "1   0   0   0   0   0   0   0 "
#define   STATUS_A      0x01   //     "1   0   0   0   0   0   0   1 "
#define   STATUS_V_R     0x02   //     "1   0   0   0   0   0   1   0 "
#define   STATUS_V_L     0x03   //     "1   0   0   0   0   0   1   1 "
#define   STATUS_D_R     0x04   //     "1   0   0   0   0   1   0   0 "
#define   STATUS_D_L     0x05   //     "1   0   0   0   0   1   0   1 "
#define   STATUS_KM      0x06   //     "1   0   0   0   0   1   1   0 "
//Comandos reset
#define   RST_T         0x00   //     "1   1   1   1   0   0   0   0 "
#define   RST_D         0x01   //     "1   1   1   1   0   0   0   1 "
#define   RST_KM         0x02   //     "1   1   1   1   0   0   1   0 "
//Parámetros
#define   RIGHT          1
#define   LEFT          0
#define   DISTANCE       0
#define   TIME          1
#define   ANGLE          0
#define   FWD           1
```

```
#define   BACK          0
#define   CENTER         1
#define   WHEEL          0
```

En las funciones de movimiento, primero se le asigna a “MOT_COMAND” el tipo de movimiento y luego se le añaden por la derecha, y desplazando a la izquierda sus bits, los valores de las varibles “COMTYPE” y “RL”. Estas variables son entradas de la función de movimiento y toman los valores 0 ó 1 (“COMTYPE” es 0 si se le pasa a la función una distancia o un ángulo y es 1 si se le pasa el tiempo de ejecución; “RL” es 1 si el movimiento es a la derecha y 0 si es a la izquierda). A continuación se muestra como ejempo unas líneas de la función MOT_CHA_VEL (cambio de velocidad).

```
int8 MOT_CHA_VEL(int MOT_VEL,int1 FWDBACK,int1 RL,int1 COMTYPE,int MOT_T_DIST)
{
      //El comando a mandar es COM_CHA_VEL (0x01)
      MOT_COMAND=COM_CHA_VEL;
      //Se mandan COMTYPE y RL en el mismo byte de MOT_COMAND
      shift_left (&MOT_COMAND, 1, COMTYPE);
      shift_left (&MOT_COMAND, 1, RL);
}
```

El intercambio de la variable “MOT_COMAND” entre los microbots puede ser muy interesante para que un robot conozca el movimiento que realiza el otro. Para saber con exactitud el movimiento, también debemos conocer la velocidad, que se puede consultar mediante la función MOT_FDBCK; y los valores del centro de giro y del sentido del movimiento (hacia adelante o hacia atrás). En la librería viene definido el movimiento hacia adelante como 1 y el movimiento hacia atrás como 0; mientras que el giro respecto al centro toma el valor 1 y respecto a la rueda, 0.

Para conocer estos últimos datos, se crea la variable “CW_FB”. En la posición cero de este byte se almacena el sentido y en la posición 1, el cento de giro. Se añade esta nueva variable a todas las funciones de la librería en las que se necesite saber el sentido y/o el centro de giro. En estas funciones se establece el valor del sentido y del centro según los parámetro de entrada de la función, el centro sólo se emplea en la función de rotación. “CW_FB” puede ser consultada en cualquier momento, permitiendo una definición completa de cualquier movimiento.

Modificaciones de las funciones

```
int8 MOT_STR(int MOT_VEL, FWDBACK,int1 COMTYPE, int MOT_T_DIST)
{
      //Se almacena el sentido
      CW_FB=FWDBACK;
}
 int8 MOT_CHA_VEL(int MOT_VEL,int1 FWDBACK,int1 RL,int1 COMTYPE,int MOT_T_DIST)
{
      //Se almacena el sentido
      CW_FB=FWDBACK;
}
int8 MOT_ROT(int MOT_VEL,int1 FWDBACK,int MOT_CENWHEEL,int1 RL,int1 COMTYPE,int
MOT_T_ANG)
{
      //Se almacena el sentido y el centro de giro
      CW_FB=FWDBACK;
      if(MOT_CENWHEEL==1)
```

```
            bit_set(CW_FB,1);
}
int8 MOT_CUR(int MOT_VEL,int1 FWDBACK,int MOT_RAD,int1 RL,int1 COMTYPE, int MOT_T_DIST)
{
      //Se almacena el sentido
      CW_FB=FWDBACK;
}
```

3.3.2.- Librería de radiofrecuencia

Se añaden 2 funciones a la librería que viene con el módulo de radiofrecuencia para evitar la repetición innecesaria de código. Las funciones son GUARDA_MOV y ENVIO_DATOS. A la librería de radiofrecuencia con estas dos nuevas funciones se le asigna el nombre de "lib_rf_new_10.h", esta librería se puede consultar también en el CD adjunto a la memoria.

GUARDA_MOV

Se ha visto que para definir completamente el movimiento realizado por un robot, se necesitan saber una serie de variables: las velocidades, la variable "MOT_COMAND" y la variable "CW_FB". Para poder enviar estos datos primero se deben almacenar en diferentes posiciones del vector de datos de salida. Mediante la función GUARDA_MOV se leen los valores de estas variables y se almacenan en el vector de salidas, así no hay que repetir todo el código cada vez que se quiera enviar un movimiento, basta con llamar a esta función.

Código

```
void GUARDA_MOV()
{
      //Velocidad del motor derecho
      MOT_FDBCK(STATUS_V_R);
      //Se almacena en la pos 0 del vector de salidas
      RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
      //Velocidad motor izquierdo
      MOT_FDBCK(STATUS_V_L);
      //Se almacena en la pos 1 del vector de salidas
      RF_DATA_OUT[1]=MOT_STATUS_DATA_0;
      //Se guardan las variables MOT_COMAND y CW_FB en las pos 2 y 3 del vector de salidas
      RF_DATA_OUT[2]=MOT_COMAND;
      RF_DATA_OUT[3]=CW_FB;
}//**************************************************
```

ENVIO_DATOS

Cada vez que se envían datos, es aconsejable comprobar que se han enviado correctamente y que se ha recibido el acuse de recibo. Por ello se ha añadido una función en la que, a la vez que se envían datos, se comprueba si se ha producido algún error y, si es así, se vuelven a enviar los datos de nuevo.

Código

```
void ENVIO_DATOS()
{
      int8 ret;
      //Envío de los datos
```

```
        ret=RF_SEND();
        if(ret==0)
        {
                //Envio realizado y ACK recibido, no se realiza ninguna acción
        }
        //Envio realizado y ACK no recibido
        else if(ret==1)
                ret=RF_SEND();          //Se repite otra vez el envío
        //Envio no realizado
        else
                ret=RF_SEND();          //Se repite otra vez el envío
}//*******************************************************
```

3.4.- PROGRAMAS PARA COMPROBAR EL RANGO DE VALORES DE LOS SENSORES

Se crean una serie de programas que permiten **visualizar el valor de los sensores** en la ventana de radiofrecuencia de Moway Center. Los algoritmos "sensor_linea_analog.c", "sensor_luz.c" y "sensor_obs.c" se encargan de comprobar el valor de cada sensor y mandarlo al PC.

"sensor_linea_analog.c"

Se realizan diferentes experimentos sobre varias superficies. Las diferentes zonas estudiadas comprenden desde folios y cartulinas de diferentes colores hasta cinta aislante y folios con un rectángulo impreso de diversas luminancias, este último experimento se puede apreciar en la figura 37. Estas pruebas consisten en que Moway vaya avanzando sobre la superficie (realizando varios barridos), leyendo ambos sensores y enviando estas magnitudes al ordenador. La información es almacenada en hexadecimal en la ventana de recepción de la pantalla de radiofrecuencia. De todas las pasadas efectuadas se toma el rango de valores más amplio, se eligen los valores menor y mayor de todos.

Figura 37: Prueba sensores de línea, luminancia 175

En el anexo I de esta memoria se muestra una tabla con los datos obtenidos de forma ordenada. Para cada elemento se recogen los valores de los dos sensores de Moway 3 y Moway 4, tanto en hexadecimal como en decimal. También se añade el

rango de valores de todos los sensores de ambos robots para cada material, y un rango incrementado para tener la seguridad de que todos los valores de esa zona se encuentran en ese rango. En la tabla se observa que las superficies están ordenadas de mayor a menor, los valores mayores de los sensores se pueden encontrar en la parte superior y los menores en la parte inferior.

De estos experimentos se extraen las siguientes conclusiones:

- Los sensores de línea toman valores **entre 0 y 255**.
- El valor de los sensores **aumenta cuando disminuye** el valor de **luminancia**.
- Los **robots tienen diferentes sensibilidades**, incluso sus propios sensores tienen sensibilidades diferentes.
- **Moway puede detectar cuatro superficies diferentes** si se toma el rango de valores genérico, pero este número podría ampliarse teniendo en cuenta el valor que toma cada sensor de cada robot en las diferentes zonas. Esto puede ser muy útil para la realización de tareas cooperativas.

Código

```
//Sensor_linea_analog.c
//
//PFC Mari Carmen Vera Sánchez
//
//Programa para comprobar los valores de los sensores de obstáculos enviándolos al PC
//*****************************************************************************
#include <16F876A.h>
#DEVICE ADC= 8
#use delay(clock=4000000)
//*****************************[LIBRERIAS MOWAY]*********************************
#include "lib_mot_new_10.h"
#include "lib_sen_moway_10.h"
#include "lib_rf_new_10.h"
//*****************************************************************************
//Programa principal
void main()
{
//**************[Configuración  de sensores y motores]************************
SEN_CONFIG();
MOT_CONFIG();
//************[Configuración de FR]**********
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x01)==0)
{
      //Módulo no configurado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
      //Módulo no inicializado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa el led sup rojo
}
//Direccion de envio
RF_DIR_OUT=0x02;
Delay_ms(1000); //Esperar 1 seg
```

```
MOT_STR(20,FWD,TIME,0);
while(1)
{
//***************[COMPROBAMOS SENSORES Y ENVIAMOS]*************
      SEN_LINE_ANALOG();
   RF_DATA_OUT[7]=SEN_LINE_R;
      RF_DATA_OUT[6]=SEN_LINE_L;
      ENVIO_DATOS();
      Delay_ms(50);
}}//*****************************************************************************
```

“sensor_obs.c”

En este caso los experimentos realizados consisten en colocar objetos delante del microbot para ver a qué distancia empiezan a detectar, a qué distancia detectan el máximo y los valores obtenidos a dos centímetros de distancia del objeto. Los datos obtenidos se recogen en la tabla 14.

Moway 3	Empieza a detectar		Sensor Izq.	5.5cm.
			Sensor De.	7cm.
Moway 4			Sensor Izq.	5cm.
			Sensor De.	4.5cm.
Moway 3	Detecta Máximo: FF(255)		Sensor Izq.	1cm.
			Sensor De.	1cm.
Moway 4			Sensor Izq.	1cm.
			Sensor De.	1cm.
Moway 3	2cm.	Sensor Izq.	79-7F	121-127
		Sensor De.	8B-90	139-144
Moway 4		Sensor Izq.	6B-6F	107-111
		Sensor De.	77-81	119-129

Tabla 14: Sensor de obstáculos

Se aprecia que, como en el caso del sensor de línea, los **valores** obtenidos por **cada sensor** son **diferentes**, aunque todos detectan el **máximo valor a** una distancia aproximada de **1 centímetro**. Las cifras varían **entre 0 y 255**, **aumentando** el valor **conforme Moway se acerca** al objeto.

Código

```
//***************[COMPROBACIÓN SENSORES Y ENVÍO]**********************
      RF_DATA_OUT[7]=SEN_OBS_R;
      RF_DATA_OUT[6]=SEN_OBS_L;
      ENVIO_DATOS();
      Delay_ms(500);
}//*****************************************************************************
```

“sensor_luz.c”

Se coloca a los Moway a diferentes separaciones de un foco luminoso y se van leyendo los valores de la variable SEN_LIGHT_P. Se Obtiene 0 cuando están lejos de la luz y 94 cuando están muy cerca. Así que parece que el sensor varía **entre 0 y 100**.

Código

```
//***************[COMPROBACIÓN SENSOR Y ENVÍO]**************************
        SEN_LIGHT();
        RF_DATA_OUT[7]=SEN_LIGHT_P;
        ENVIO_DATOS();
        Delay_ms(500);
//**********************************************************************************
```

CAPÍTULO 4

TAREAS COOPERATIVAS CON MOWAY

CAPÍTULO 4. TAREAS COOPERATIVAS CON MOWAY

Actualmente, en el terreno de la robótica, se está produciendo un gran desarrollo en el campo de los robots cooperativos. Moway, gracias a sus sensores y a su tarjeta de radiofrecuencia, puede realizar tareas en grupo. Algunos problemas son demasiado difíciles de resolver para un único Moway (empujar una caja, limpiar un recinto…). Por ello, se han diseñado sistemas compuestos de varios microbots capaces de resolver problemas conjuntamente para realizar las tareas con mayor eficiencia. Estos programas pueden servir de base para implementar tareas cooperativas en robots más complejos.

Este capítulo se divide en 4 apartados en los que se presentan las tareas cooperativas desarrolladas:

4.1.- Compartición de información para la repetición de un trayecto.

En este apartado se estudia la posibilidad de que un robot realice los mismos movimientos que otro a partir de la información recibida. Por un lado se envían los valores de los sensores de líneas cuando uno de los robots realiza una trayectoria dibujada en el suelo; y por otro lado se envían directamente los movimientos que está ejecutando el robot en ese momento.

4.2.- Cooperación para desplazar una carga pesada o voluminosa.

Esta tarea consiste en que 2 robots se ayuden a la hora de desplazar un objeto grande o pesado desde un punto a otro en línea recta. Mediante el uso de 2 robots se consigue que el objeto llegue a su destino de forma más recta ya que tiene 2 puntos de apoyo y además se pueden desplazar objetos más pesados. A la hora de mover el objeto, los microbots tienen que sincronizar sus movimientos.

4.3.- Cooperación para la limpieza de un área determinada.

En este apartado se comprueba le eficiencia de un sistema de limpieza formado por 2 robots que se van comunicando. La zona a limpiar tiene una serie de objetos y los robots cooperan para sacarlos todos. A la hora de realizar esta tarea se han tenido que solventar numerosos inconvenientes y por ello se han realizado una gran cantidad de experimentos y modificaciones en los códigos iniciales. Se han solucionado problemas como que un robot pueda expulsar a otro, que 2 robots saquen fuera del recinto a un mismo objeto, que se extraiga 2 veces el mismo objeto por estar muy cerca de la zona interior, que un robot se salga por error de la zona interior (se ha conseguido que los robots tengan la capacidad de regresar al recinto)…

4.4.- Cooperación para el guiado de objetos por un camino.

Esta tarea consiste en llevar un objeto de un punto a otro siguiendo un camino marcado en el suelo. En este caso se emplean 3 robots, uno se encarga de desplazar el objeto por el camino y los otros (colocados de forma perpendicular a la dirección de avance uno a cada lado del objeto) lo mantienen centrado en la zona a recorrer. Los robots realizan sus movimientos de forma sincronizada.

Los códigos explicados en este capítulo se pueden consultar en el CD adjunto a esta memoria.

4.1.- COMPARTICIÓN DE INFORMACIÓN PARA LA REPETICIÓN DE UN TRAYECTO

En este apartado se va a estudiar la capacidad de Moway de repetir los mismos movimientos de otro Moway. Se pretende realizar esta tarea en tiempo real partiendo únicamente de los datos que los robots se transmiten por radiofrecuencia. En un primer momento se utiliza la información de los sensores de línea, un robot sigue la línea y el otro lo imita. Luego se consigue elaborar un algoritmo más complejo en el que se transmiten los distintos movimientos que en cada instante ejecuta el robot.

4.1.1.- Compartición de información sensorial

En este apartado se van a presentar los algoritmos implementados en los códigos fuente “RF1.c” y “RF2.c”. El primer programa, “RF1.c”, tiene el código básico del seguimiento de una línea al que se le ha añadido la comunicación por RF. El segundo programa, “RF2”, se encarga de recibir datos por RF. “RF1.c” envía un dato según el valor de sus sensores de línea, y “RF2.c” hace el movimiento correspondiente. En las figuras 38 y 39 se puede ver un esquema de ambos programas en ejecución.

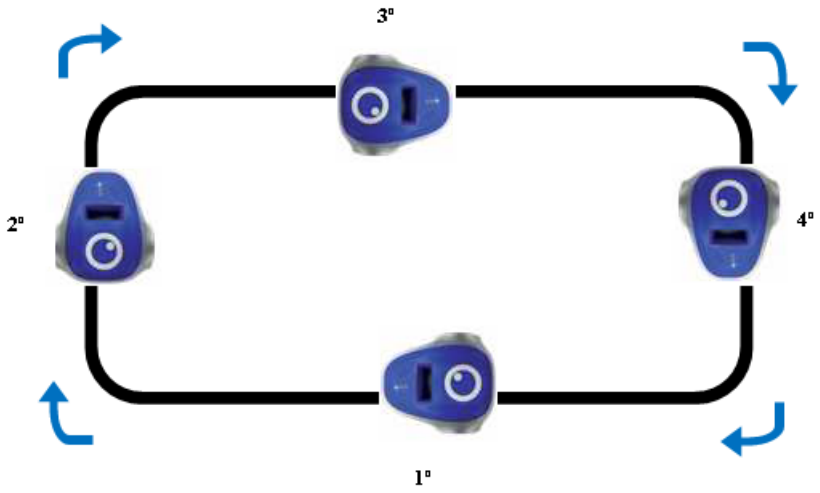

Figura 38: Moway 3 sigue línea

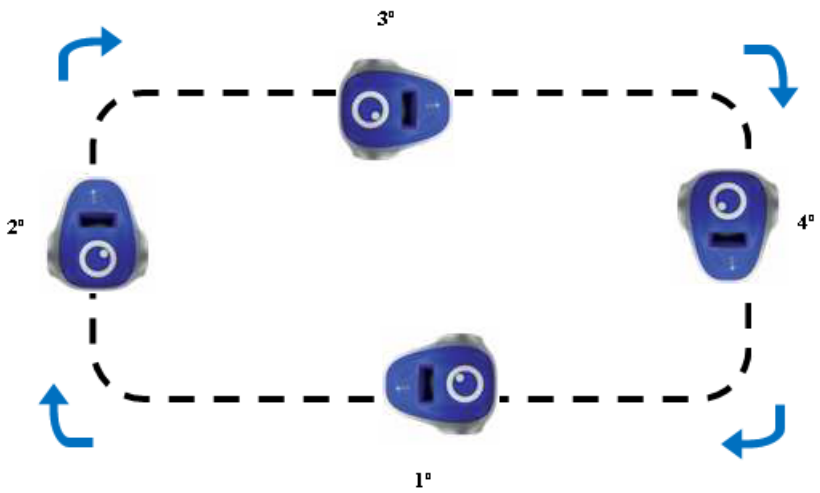

Figura 39: Moway 4 emula la trayectoria de Moway 3

Estos programas pueden resultar útiles para diversas tareas que requieran que ambos robots realicen el mismo movimiento. A continuación se explican 2 posibles ejemplos de utilización, el primero es un ejemplo industrial y el segundo es un ejemplo lúdico.

Como primer ejemplo, vamos a hablar de la utilidad de estos programas en una empresa que posee varios pisos idénticos con diferentes despachos. En todas las empresas se reciben paquetes en las diferentes oficinas (ya sea correo o material útil para el trabajo). Los robots pueden encargarse de hacer la ronda de cada piso cargando con el material y deteniéndose un tiempo prudencial en la puerta de cada despacho. Para ello sólo sería necesario que uno de los pisos tuviera la línea dibujada en el suelo y, en los otros, lo demás robots imitarían el movimiento del primero.

Como segundo ejemplo de utilización cabe la posibilidad de emplear estos algoritmos como un juego infantil. El juego consiste en colocar un robot en una zona que no puedan ver los niños. El robot sigue una línea con una determinada forma: letras, polígonos, dibujos sencillos… El otro robot imita su recorrido y la finalidad del juego es adivinar de qué dibujo se trata.

A continuación se comentan ambos programas.

“RF1.c”

Seguimiento de una línea

El programa está diseñado para que el robot vaya siguiendo el borde izquierdo de la línea como se puede comprobar en el esquema de la figura 40. Mediante los sensores de línea se comprueba durante todo el programa con qué sensor se detecta línea. Existen 4 casos:

1) La línea es detectada mediante el **sensor derecho** → El movimiento que realiza es **hacia adelante** al 80% de la velocidad.
2) La línea no es detectada por **ningún sensor** → El robot **gira a la derecha** al 50% de la velocidad respecto a su centro.
3) La línea es detectada por **ambos sensores** → Se realiza un **giro a la izquierda** al 50% de la velocidad respecto al centro.
4) La línea es detectada por el **sensor izquierdo** → El robot también **gira a la izquierda**.

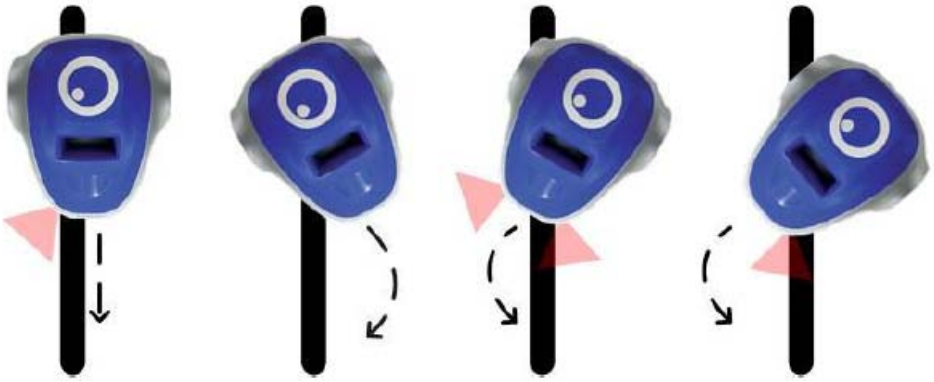

Figura 40: Seguimiento de línea

Envío de datos por RF

Se configura el módulo de RF para que se le asigne la dirección 1 del canal 40. Se le destina como dirección de salida la 2. Según la detección de los sensores, se atribuye un valor a la posición 0 del vector de datos a enviar.

1) La línea es detectada mediante el **sensor derecho** → El movimiento que realiza es **hacia adelante** al 80% de la velocidad y se envía **0x00**.
2) La línea no es detectada por **ningún sensor** → El robot **gira a la derecha** al 50% de la velocidad respecto a su centro y se envía **0x01**.
3) La línea es detectada por **ambos sensores** → Se realiza un **giro a la izquierda** al 50% de la velocidad respecto al centro y se envía **0x02**.
4) La línea es detectada por el **sensor izquierdo** → El robot también **gira a la izquierda** y se envía **0x02**.

Otros aspectos del programa

Si se produce algún error en la configuración del módulo de RF, se enciende el LED superior rojo.

Tras las configuraciones y las inicializaciones de variables, se espera un segundo y el robot comienza girando a la derecha al 50% de la velocidad respecto al centro.

Cuando se comprueba el sensor de línea también se encienden los LEDS rojos inferiores del lado que está detectando línea en ese momento.

Código

```
void main()
{
//**************[Configuración  de sensores y motores]**************************
SEN_CONFIG();
MOT_CONFIG();
//************[Configuración de RF]**********
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x01)==0)
{
      //Módulo no configurado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF

if(RF_ON()==0)
{
      //Módulo no inicializado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa led sup rojo
}
//Inicialización de variables
RF_DIR_OUT=0x02;
Delay_ms(1000);              //Esperar un segundo y empezar a buscar la línea
// Primer movimiento para encontrar la línea
MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
while(1){
//**************[SEGUIMIENTO DE LINEA Y ENVIO DE DATOS]**********
//Se comprueban los sensores de línea
SEN_LINE_DIG();
// Si está en el borde, sigue recto
if(SEN_LINE_R==1 && SEN_LINE_L==0)
{
   LED_R_ON();
```

```
   LED_L_OFF();
   MOT_STR(80,FWD,TIME,0);
       RF_DATA_OUT[0]=0x00;
       ENVIO_DATOS();
}
//Si no ve línea, gira a la derecha
else if(SEN_LINE_R==0 && SEN_LINE_L==0)
{
   LED_R_ON();
   LED_L_ON();
   MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
       RF_DATA_OUT[0]=0x01;
       ENVIO_DATOS();
}
//Si está encima de la línea, gira a la izquierda
else if(SEN_LINE_R==1 && SEN_LINE_L==1)
{
       LED_R_OFF();
       LED_L_OFF();
       MOT_ROT(50,FWD,CENTER,LEFT,TIME,0);
       RF_DATA_OUT[0]=0x02;
       ENVIO_DATOS();
}
//Si está en el otro borde de la línea, gira a la izquierda
else if(SEN_LINE_R==0 && SEN_LINE_L==1)
{
       LED_R_OFF();
       LED_L_ON();
   MOT_ROT(50,FWD,CENTER,LEFT,TIME,0);
       RF_DATA_OUT[0]=0x02;
       ENVIO_DATOS();
}}}//*****************************************************************************
```

“RF2.c”

En el programa principal se espera a recibir un dato. Según el valor de la posición 0 del vector de datos de entrada, se produce un movimiento u otro:

- **0x00** → Movimiento **recto** hacia adelante al 80% de la velocidad.
- **0x01** → **Giro a la derecha** al 50% de la velocidad respecto al centro.
- **0x02** → **Giro a la izquierda** al 50% de la velocidad respecto al centro.
- Otro valor → El robot se para.

En este programa al robot se le asigna la dirección 2 del canal 40 y también se comprueba si hay algún error en esta configuración (encendiéndose el LED superior rojo).

Como el programa sólo se encarga de recibir datos, se realiza la recepción en el programa principal, sin emplear interrupciones.

Código

```
void main()
{
//*********[Configuración  de sensores, motores y RF]*********************
SEN_CONFIG();
MOT_CONFIG();
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
```

```
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x02)==0)
{
	//Módulo no configurado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
	//Módulo no inicializado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
while(1)
{
	while(RF_RECEIVE()!=2)
	{
		if(RF_DATA_IN[0]==0x00)
		MOT_STR(80,FWD,TIME,0);
		else if(RF_DATA_IN[0]==0x01)
			MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
		else if(RF_DATA_IN[0]==0x02)
			MOT_ROT(50,FWD,CENTER,LEFT,TIME,0);
		else
			MOT_STOP();
	}
}}//*****************************************************************************
```

Experimentos

Se comprueba que tanto “RF1.c” como “RF2.c” realizan la tarea prevista. “RF1.c” sigue la línea, da igual si el circuito es un rectángulo, un pentágono, una elipse o cualquier otra figura, Moway recorre la línea siempre sobre el borde izquierdo. Se observa que con el programa “RF2.c” el robot ejecuta el movimiento programado según el valor recibido, para ello se van enviando los datos mediante el PC.

Finalmente se comprueba el funcionamiento de ambos programas juntos y se observa que se produce la copia de los movimientos, aunque el robot que sigue la trayectoria del otro se va desviando poco a poco debido a los errores en la odometría.

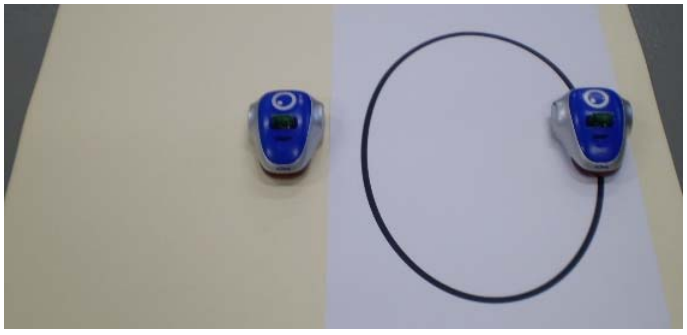

Figura 41: Prueba de “RF1.c” y “RF2.c”

4.1.2.- Compartición del movimiento ejecutado

En este caso se pretende que Moway copie los movimientos del otro robot a partir de otras variables como la velocidad, la variable "MOT_COMAND" y la nueva variable creada "CW_FB".

Moway imita los movimientos rectos hacia adelante

Se toma como punto de partida el envío por RF de la velocidad. Pare ello se crean los programas "RF3.c" y "RF4.c". "RF3.c" realiza movimientos rectos y hacia adelante a diferentes velocidades, mientras que "RF4.c" se dedica a leer el valor recibido y modificar su velocidad.

"RF3.c"

Para poder conocer el dato de la velocidad, se tiene la función MOT_FDBCK. Esta función asigna a la variable "STATUS" el valor del dato solicitado como parámetro, se puede leer la velocidad de cada sensor, el valor del cuentakilómetros...

Este algoritmo contiene cuatro movimientos rectos hacia adelante: al **80%** de la velocidad **2 segundos**, al **10% 5 segundos**, al **100% 2 segundos** y al **30% 4 segundos**. Tras cada desplazamiento **manda la velocidad** del motor derecho en la posición 0 del vector de datos de salida. Este Moway tiene la dirección 1 del canal 0x40 y envía los datos a la dirección 2.

Código

```
void main()
{
//**************[Configuración  de sensores y motores]************************
SEN_CONFIG();
MOT_CONFIG();
//************[Configuración de RF]**********
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x01)==0)
{
      //Módulo no configurado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
      //Módulo no inicializado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa led sup rojo
}
//Direccion de envio
RF_DIR_OUT=0x02;
Delay_ms(1000); //Esperar 1 seg
while(1){
//**********************[Movimientos y envio de datos]**************************
MOT_STR(80,FWD,TIME,0);            //mov al 80%
MOT_FDBCK(STATUS_V_R);         //Se obtiene el valor de la velocidad del mot derecho
RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
ENVIO_DATOS();
```

```
Delay_ms(2000);
MOT_STR(10,FWD,TIME,0);          //mov al 10%
MOT_FDBCK(STATUS_V_R);           //Se obtiene el valor de la velocidad del mot derecho
RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
ENVIO_DATOS();
Delay_ms(5000);
MOT_STR(100,FWD,TIME,0);         //mov al 100%
MOT_FDBCK(STATUS_V_R);           //Se obtiene el valor de la velocidad del mot derecho
RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
ENVIO_DATOS();
Delay_ms(2000);
MOT_STR(30,FWD,TIME,0);          //mov al 30%
MOT_FDBCK(STATUS_V_R);           //Se obtiene el valor de la velocidad del mot derecho
RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
ENVIO_DATOS();
Delay_ms(4000);
}}//*****************************************************************
```

Experimentos

Se prueba el programa enviando las velocidades al PC como se observa en la figura 42. Los valores de las velocidades coinciden exactamente con las programadas, por lo que se deduce que el valor de la velocidad que lee el robot es el que se le ha cargado en lugar del real.

Velocidades: 1E=30 64=100 A=10 50=80

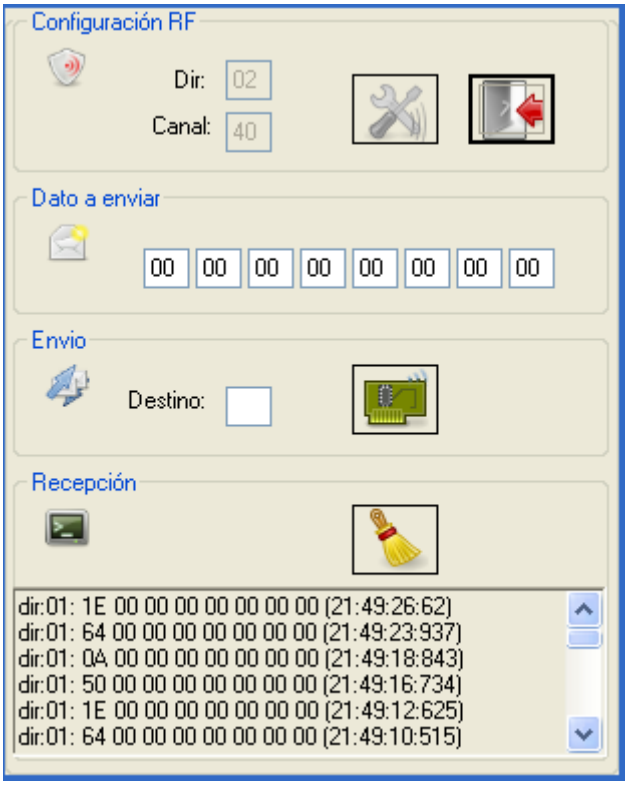

Figura 42: Velocidades enviadas por el programa “RF3.c”

“RF4.c”

Este programa **espera a recibir datos**. Cuando los recibe, **actualiza** el valor de **la velocidad** y **avanza** recto hacia adelante **a esa velocidad**. Se configura con la dirección 2 del canal 0x40.

Código

```
//Variables auxiliares
int8 vel;
//Programa principal
void main()
{
//*********[Configuración  de sensores, motores y RF]*********************
SEN_CONFIG();
MOT_CONFIG();
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x02)==0)
{
	//Módulo no configurado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
	//Módulo no inicializado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Inicialización de la velocidad a 0
vel=0;
while(1)
{
	while(RF_RECEIVE()!=2)
	{
		if(vel!=RF_DATA_IN[0])
			vel=RF_DATA_IN[0];
	MOT_STR(vel,FWD,TIME,0);
	}
}}//*****************************************************************************
```

Experimentos con ambos programas

Se observa que Moway 4 consigue ir a la misma velocidad que Moway 3 en tiempo real.

<u>Moway imita una trayectoria hacia adelante</u>

En este apartado se van a modificar los programas anteriores para poder imitar más tipos de movimientos. El programa “RF3_1.c” contiene 5 movimientos diferentes y el programa “RF4_1.c” utiliza la función cambio de velocidad para copiar los movimientos. En este caso se modifican las velocidades de ambos motores.

En un primer momento el programa “RF3_1.c” contiene los cinco movimientos que se especifican a continuación:

1) Movimiento recto al 80% de la velocidad durante 2 segundos.
2) Cambio de velocidad del motor derecho al 10% durante 5 segundos.
3) Rotación de 360º al 100% respecto al centro y a la izquierda.
4) Giro a la derecha con un motor al 30% y el otro al 90% durante 4 segundos.
5) Parada de 3 segundos.

El programa “RF4_1.c” se encarga de comprobar si ha recibido información y si es así, actualiza el valor de las velocidades y modifica la velocidad de cada motor.

Probando estos 2 programas se observa que en el giro sobre el centro, el robot sigue recto y, en la parada, continúa con el movimiento anterior. Se envían las velocidades por RF al PC como se ve en la siguiente figura y se comprueba que en el giro sobre el centro ambos motores tienen la misma velocidad, pero al utilizar la función cambio de velocidad no se tiene en cuenta que los motores giran en sentido contrario. También se observa que cuando Moway se detiene la velocidad es la misma que en el movimiento anterior.

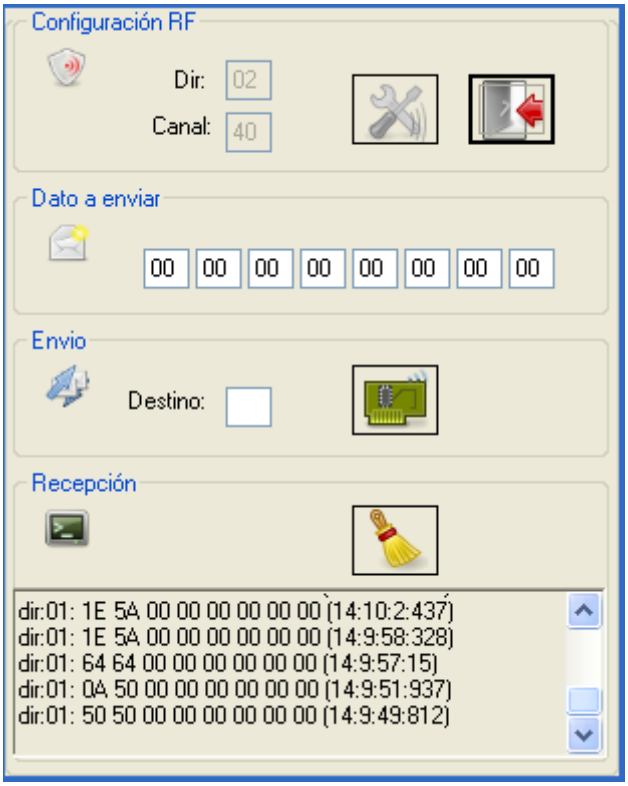

Figura 43: Velocidad del motor izquierdo en la posición 0 y del motor derecho en la 1

Se comprueba en la librería de motores que, efectivamente, la función MOT_STOP no actualiza el valor de la velocidad. También se descubre la variable “MOT_COMAND” en la que se almacena el tipo de movimiento, el sentido y el tipo de unidad. Esta variable puede ser muy útil.

Se le añade a “RF3_1.c” el envío de la variable “MOT_COMAND” en la posición 2 del vector de datos de salida y se prueba a enviar el valor del ángulo de giro en la posición 3. A continuación aparece el código añadido para almacenar el ángulo.

```
MOT_FDBCK(STATUS_A);
RF_DATA_OUT[3]=MOT_STATUS_DATA_0;
```

En la figura 44 se encuentran los datos recibidos por el ordenador. Se aprecia que cuando el robot para, manda en "MOT_COMAND" el valor 0x10. También se observa que la función MOT_FDBCK no actualiza el valor del ángulo, ya que es 0 en todos los casos y en algunos se ha especificado un ángulo distinto de 0 en el programa.

Figura 44: envío de las velocidades, MOT_COMAND y el ángulo

Finamente se modifica el código de los programas teniendo en cuenta las pruebas realizadas.

"RF3_1.c"

En el programa se ejecutan todo el tiempo **5 movimientos**, en este caso se sustituye el giro respecto al centro por un giro respecto a la rueda izquierda. Por lo que los movimientos que realiza el robot son los siguientes:

1) Movimiento **recto al 80%** de la velocidad durante **2 segundos**.
2) **Cambio** de la **velocidad** del **motor derecho al 10%** durante **5 segundos**.
3) **Rotación** de **360º al 100% respecto** a la **rueda izquierda**.
4) **Giro a la derecha** con un motor al 30% y el otro al 90% durante **4 segundos**.
5) **Parada de 3 segundos**.

Tras cada movimiento se envían las velocidades de ambos motores **y "MOT_COMAND"**. En la posición 0 se envía el valor de la velocidad del motor

derecho, en la posición 1 el valor de la velocidad del motor izquierdo y en la posición 2 “MOT_COMAND”.

Código

```
void main()
{
//**************[Configuración  de sensores y motores]************************
SEN_CONFIG();
MOT_CONFIG();
//************[Configuración de RF]**********
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x01)==0)
{
	//Módulo no configurado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
	//Módulo no inicializado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Direccion de envio
RF_DIR_OUT=0x02;
Delay_ms(1000); //Esperar 1 seg
while(1)
{
//**********************[Movimientos y envio de datos]**************************
//En todos los mov se envía en la pos 0 la vel del motor derecho y en la pos 1 la del motor izquierdo
//Mov1
MOT_STR(80,FWD,TIME,0);		//mov recto al 80%
MOT_FDBCK(STATUS_V_R);
RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
MOT_FDBCK(STATUS_V_L);
RF_DATA_OUT[1]=MOT_STATUS_DATA_0;
RF_DATA_OUT[2]=MOT_COMAND;
ENVIO_DATOS();
Delay_ms(2000);
//Mov2
MOT_CHA_VEL(10,FWD,RIGHT,TIME,0);	//cambio de velocidad del motor derecho al 10%
MOT_FDBCK(STATUS_V_R);
RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
MOT_FDBCK(STATUS_V_L);
RF_DATA_OUT[1]=MOT_STATUS_DATA_0;
RF_DATA_OUT[2]=MOT_COMAND;
ENVIO_DATOS();
Delay_ms(5000);
//Mov3
MOT_ROT(100,FWD,WHEEL,LEFT,ANGLE,100);		//rotar 360º al 100% respecto a la rueda izquierda
MOT_FDBCK(STATUS_V_R);
RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
MOT_FDBCK(STATUS_V_L);
RF_DATA_OUT[1]=MOT_STATUS_DATA_0;
RF_DATA_OUT[2]=MOT_COMAND;
ENVIO_DATOS();
while(!input(MOT_END)){}			//No se realiza otro movimiento hata que se acabe este
//Mov4
```

```
MOT_CUR(60,FWD,30,RIGHT,TIME,0);          //giro a la derecha con un motor al 30% y otro al 90%
MOT_FDBCK(STATUS_V_R);
RF_DATA_OUT[0]=MOT_STATUS_DATA_0;
MOT_FDBCK(STATUS_V_L);
RF_DATA_OUT[1]=MOT_STATUS_DATA_0;
RF_DATA_OUT[2]=MOT_COMAND;
ENVIO_DATOS();
Delay_ms(4000);
//Mov5
MOT_STOP();          //parar 3 seg
RF_DATA_OUT[2]=MOT_COMAND;
ENVIO_DATOS();
Delay_ms(3000);
}}//*****************************************************************************
```

"RF4_1.c"

El programa se dedica a **comprobar si ha recibido algún dato**. **Si** recibe un valor de la variable "**MOT_COMAND" distinto de 0x10 (parada)**, **actualiza** el valor de **las velocidades y cambia la velocidad de cada motor** mediante la función MOT_CHA_VEL. Estos nuevos valores de velocidades se mantienen de forma indefinida debido a que los últimos parámetro que le pasamos a la función son TIME y 0. **Si**, por el contrario, **recibe 0x10** en "MOT_COMAND", entonces el robot **se para**.

Código

```
//Variables auxiliares
int8 VEL_R;
int8 VEL_L;
//Programa principal
void main()
{
//*********[Configuración  de sensores, motores y RF]*********************
SEN_CONFIG();
MOT_CONFIG();
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x02)==0)
{
      //Módulo no configurado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
      //Módulo no inicializado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa el led sup rojo
}
//inicialización de velocidades a cero
VEL_R=0;
VEL_L=0;
//Direccion de envio
RF_DIR_OUT=0x01;
while(1)
{
      while(RF_RECEIVE()!=2)
      {
            //Se comprueba si el comando es stop, si no lo es, se cambia la velocidad de los motores
```

```
            if(RF_DATA_IN[2]!=0x10)
            {
                VEL_R=RF_DATA_IN[0];
                VEL_L=RF_DATA_IN[1];
                MOT_CHA_VEL(VEL_R,FWD,RIGHT,TIME,0);
                MOT_CHA_VEL(VEL_L,FWD,LEFT,TIME,0);
            }
            //Si se ha enviado parar, se para
            else if(RF_DATA_IN[2]==0x10)
                MOT_STOP();
    }
}}//*****************************************************************************
```

Experimentos

Se observa que un robot imita los movimientos del otro.

Moway copia cualquier movimiento

En este apartado se va a hablar de algunas mejoras que se han hecho en los programas anteriores. Los nuevos algoritmos son “RF3_2.c” y “RF4_2.c” y con ellos se consigue que un robot imite cualquier movimiento que el otro realice.

Como se ha visto, la variable MOT_COMAND contiene el tipo de movimiento que realiza el robot, así como la dirección y el tipo de dato introducido (tiempo, ángulo o distancia).

Como ya se ha comentado, para definir completamente un movimiento se necesita saber también, a parte de las velocidades, el sentido y el centro de giro. Para ello, como se vio en el apartado de modificaciones de las librerías, se crea la variable “CW_FB”. En su bit 0 se almacena el sentido (1 hacia adelante y 0 hacia atrás); y en el bit 1, el centro de giro (1 si el centro es el centro del robot y 0 si el centro es una de las ruedas).

Mediante la función GUARDA_MOV se almacenan en el vector de datos de salida las velocidades de los motores y las variables “MOT_COMAND” y “CW_FB”. En la posición 0 se almacena el valor de la velocidad de motor derecho; en la 1, la velocidad del motor izquierdo; en la 2, “MOT_COMAND”; y en la 3, “CW_FB”.

“RF3_2.c”

Este programa ejecuta **8** tipos distintos de **movimientos**. A la hora de elegir los movimientos se debe tener en cuenta que después de parar no se puede colocar un cambio de velocidad, ya que el robot almacena como velocidades actuales las del movimiento anterior a la parada. A continuación aparecen los diferentes movimientos:

1) Movimiento **recto hacia atrás al 80%** de la velocidad durante **2 segundos**.
2) **Cambio de velocidad del motor derecho al 10%** durante **5 segundos.**
3) **Rotación de 360º al 100%** de la velocidad **respecto al centro y a la izquierda**.
4) **Giro a la derecha** con un motor al 30% de la velocidad y el otro al 90% durante **4 segundos**.
5) **Parada de 3 segundos**.

6) Movimiento **hacia adelante al 30%** de la velocidad durante **3 segundos**.
7) **Giro hacia atrás al 10%** de la velocidad **respecto a la rueda izquierda** durante **5 segundos**.
8) **Parada de 3 segundos**.

Tras cada movimiento, éste se almacena y se envía.

Código

```
void main()
{
//**************[Configuración  de sensores, motores y RF]*************************
SEN_CONFIG();
MOT_CONFIG();
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x01)==0)
{
      //Módulo no configurado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
      //Módulo no inicializado
      LED_TOP_RED_ON();            //Si hay problemas con RF se activa led sup rojo
}
//Direccion de envio
RF_DIR_OUT=0x02;
Delay_ms(1000); //Esperar 1 seg
while(1)
{
//**********************[Movimientos y envio de datos]**************************
//Mov1          recto al 80%
MOT_STR(80,BACK,TIME,0);
GUARDA_MOV();
ENVIO_DATOS();
Delay_ms(2000);
//Mov2          Cambio de velocidad del motor derecho al 10%
MOT_CHA_VEL(10,BACK,RIGHT,TIME,0);
GUARDA_MOV();
ENVIO_DATOS();
Delay_ms(5000);
//Mov3          rotar 360º al 100% respecto al centro
MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,100);
GUARDA_MOV();
ENVIO_DATOS();
while(!input(MOT_END)){}
//Mov4          giro a la derecha con un motor al 30% y el otro al 90%
MOT_CUR(60,FWD,30,RIGHT,TIME,0);
GUARDA_MOV();
ENVIO_DATOS();
Delay_ms(4000);
//Mov5          parada 3 seg
MOT_STOP();
GUARDA_MOV();
ENVIO_DATOS();
Delay_ms(3000);
//Mov6          Movimiento hacia adelante al 30%
```

```
MOT_STR(30,FWD,TIME,0);
GUARDA_MOV();
ENVIO_DATOS();
Delay_ms(3000);
//Mov7          Giro hacia atrás al 10% de la velocidad respecto a la rueda izquierda
MOT_ROT(10,BACK,WHEEL,LEFT,ANGLE,100);
GUARDA_MOV();
ENVIO_DATOS();
Delay_ms(5000);
//Mov8          parada 3 seg
MOT_STOP();
GUARDA_MOV();
ENVIO_DATOS();
Delay_ms(3000);
}}//**************************************************************************
```

"RF4_2.c"

El programa se encarga de **recibir datos todo el tiempo**. Tras recibir, **actualiza los valores** de "**RL"** (dirección), de los **bits 2 y 3 de "MOT_COMAND"**, del centro de giro ("**CW"**) **y** del sentido ("**FW"**). Para ello se utiliza la función "bit_test".

Según la librería de motores, si los bits 2 y 3 de "MOT_COMAND" son 0 y 1 respectivamente, la función de movimiento utilizada es MOT_ROT. En esta función es en la única que se utiliza el valor del centro de giro. Por otro lado, si el valor de MOT_COMAND es 0x10, entonces la función utilizada es MOT_STOP. Estos 2 tipos de movimientos son casos especiales que se deben programar de forma diferente a los otros. Todos los demás movimientos se pueden implementar como un cambio de velocidades. Por lo tanto existen **3 caso diferentes según los datos recibidos**:

- **Si se recibe** que el comando es **MOT_ROT**, es decir si el bit 3 de la variable "MOT_COMAND" es 1 y el bit 2 es 0. En tal caso, se tiene que **actualizar** el valor de la **velocidad** (en este caso la de los dos motores se almacena con el mismo valor). Y se **llama a la función MOT_ROT** a la que **se le pasa** como parámetros la **nueva velocidad** ("VEL_R"), la dirección ("**FB"**), el centro de giro ("**CW"**), el sentido ("**RL"**) y se le especifica que realice este movimiento indefinidamente.

- **Si el comando no es MOT_ROT ni MOT_STOP**. En este caso **se comprueba si la nueva velocidad es distinta a la anterior**. **Si es distinta, se realiza un cambio de velocidad en el motor que corresponda**. Para ello se utiliza la función MOT_CHA_VEL a la que se le pasan como parámetros la nueva velocidad ("VEL_R" o "VEL_L"), la dirección y el motor.

- **Si el comando es MOT_STOP**. En este caso **se llama a la función MOT_STOP** y se actualizan los valores de la velocidades a 0.

Código

```
//Variables auxiliares
int8 VEL_R;
int8 VEL_L;
int1 RL;
```

```
int1 bit2;
int1 bit3;
int1 CW;
int1 FB;
//Programa principal
void main()
{
//*********[Configuración  de sensores, motores y RF]*********************
SEN_CONFIG();
MOT_CONFIG();
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x02)==0)
{
	//Módulo no configurado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
	//Módulo no inicializado
	LED_TOP_RED_ON();		//Si hay problemas con RF activa el led sup rojo
}
//Se inicializan las velocidades a cero
VEL_R=0;
VEL_L=0;
//Direccion de envio
RF_DIR_OUT=0x01;
while(1)
{
	while(RF_RECEIVE()!=2)
	{
		//Se actualizan los valores de los bits de MOT_COMAND
		RL=bit_test(RF_DATA_IN[2],0);
		bit2=bit_test(RF_DATA_IN[2],2);
		bit3=bit_test(RF_DATA_IN[2],3);
		//Se actualizan los valores de los bits de CW_FB
		CW=bit_test(RF_DATA_IN[3],1);
		FB=bit_test(RF_DATA_IN[3],0);
		//Se comprueba si el comando es MOT_ROT
		if(bit3==1 && bit2==0)
		{
			//Se utiliza el comando MOT_ROT con la velocidad, la dirreción y sentido actializados
			VEL_R=RF_DATA_IN[0];
			MOT_ROT(VEL_R,FB,CW,RL,TIME,0);
		}
		//Si el comando no es MOT_ROT, ni es MOT_STOP se comprueba si ha variado la velocidad
		//y se modifica el mov del motor que haya cambiado teniendo en cuenta el sentido
		else if(RF_DATA_IN[2]!=0x10)
		{
			if(	VEL_R!=RF_DATA_IN[0])
			{
				VEL_R=RF_DATA_IN[0];
				MOT_CHA_VEL(VEL_R,FB,RIGHT,TIME,0);
			}
			if(VEL_L!=RF_DATA_IN[1])
			{
				VEL_L=RF_DATA_IN[1];
				MOT_CHA_VEL(VEL_L,FB,LEFT,TIME,0);
			}
		}
```

```
            //Si se ha enviado parar, se para
            else if(RF_DATA_IN[2]==0x10)
            {
                  MOT_STOP();
                  //Se ponen las velocidades a 0 para que sean diferentes a las anteriores
            VEL_R=0;
            VEL_L=0;
            }
      }
}}//*****************************************************************************
```

Experimentos con ambos programas

Se comprueba que ambos Moways realizan el mismo movimiento.

4.2.- COOPERACIÓN PARA DESPLAZAR UNA CARGA PESADA O VOLUMINOSA

El desplazamiento de una carga pesada o voluminosa entre dos robots es un ejemplo tradicional de robótica cooperativa. Los robots se deben coordinar para comenzar a empujar a la vez y tener un ritmo constante en cada movimiento. Mediante el uso de 2 robots se consigue que el desplazamiento del objeto sea más homogéneo, ya que al empujarlo por 2 puntos la carga se desplaza más recta; y además se consiguen mover objetos de mayor peso que con un único robot. En este apartado se va a comenzar comentando los programas "**Mov_Obj_1.c"** y **"Mov_Obj_2.c"; para finalmente explicar sus versiones mejoradas: "Mov_Obj1_2.c"** y **"Mov_Obj2_2.c".**

"Mov_Obj_1.c" y "Mov_Obj_2.c"

En un primer momento se desarrollan los programas "**Mov_Obj_1.c"** y **"Mov_Obj_2.c".** Estos programas se encargan de acercar los robots hasta que detecten el máximo con su sensor de obstáculo, esperar a que ambos se encuentren cerca de la caja y avanzar desplazando el objeto. A continuación se explica cada apartado.

Acercamiento del robot hasta detectar el máximo

El robot **avanza 5mm.** y **comprueba sus sensores de obstáculos**. Este **bucle continúa hasta** que se **detectan** los valores de **255 en ambos sensores**. Entonces la variable **"MAX" pasa a valer 1 y se envía** en la posición 7 del vector de salidas.

Esperar a recibir máximo del otro Moway

Tras alcanzar el obstáculo, se pasa a **modo recepción** de datos **hasta que** se recibe que **el otro Moway** también **ha detectado** el objeto.

Turnos para avanzar

Cuando ambos Moways han detectado el objeto, comienzan a **avanzar 5mm. al 30%** de la velocidad. **Primero** avanza el robot que contiene el programa **"Mov_Obj_1.c"**, tras avanzar, la variable **"MOV" pasa a valer 1, la envía, y espera a recibir** esta variable con valor 0, cuando el otro Moway se ha desplazado.

Código de "Mov_Obj_1.c"

```
//**************[Acercamiento del robot hasta detectar máximo]*****
while(SEN_OBS_R!=255 || SEN_OBS_L!=255)
```

```
{
	//Avanzar 0.5cm y comprobar los sensores
	MOT_STR(50,FWD,DISTANCE,3);
	while(!input(MOT_END)){}
	MOT_STOP();
	SEN_OBS_ANALOG();
}
//**************************[Envío de máximo]*********************
MAX=1;
RF_DATA_OUT[7]=MAX;
ENVIO_DATOS();
//**************[Esperar a recibir máximo del otro Moway]***********
while(MAX_IN!=1)
{
	MOT_STOP();
	if(RF_RECEIVE()!=2)
		MAX_IN=RF_DATA_IN[7];
}
while(1)
{
//********************[Turnos para avanzar]*********************
//Este moway comienza el movimiento
if(MOV==0)
{
	MOT_STR(30,FWD,DISTANCE,3);
	while(!input(MOT_END)){}
	MOT_STOP();
	MOV=1;
	RF_DATA_OUT[6]=MOV;
	ENVIO_DATOS();
}
//Espera movimiento del otro
while(MOV==1)
{
	if(RF_RECEIVE()!=2)
		MOV=RF_DATA_IN[6];
}}}//*****************************************************************************
```

Experimentos

Se comprueba que los microbots consiguen mover la caja.

Mejoras

Se crean los programas **"Mov_Obj1_2.c"** y **"Mov_Obj2_2.c"** mejorando los programas presentados anteriormente. Los cambios más significativos son los siguientes:

- El acercamiento a la caja se produce de forma más continua.
- La recepción de datos se realiza mediante interrupción.
- Los robots vuelven a avanzar en busca del objeto si dejan de detectarlo; si se coloca el objeto lejos de los robots cuando han empezado a empujarlo por turnos, los robots avanzan hasta detectar de nuevo el objeto.
- Moway se detiene al detectar una línea en el suelo.

A continuación se explican las partes de los programas:

“Mov_Obj1_2.c” y “Mov_Obj2_2.c”

Rutina de recepción

Se actualizan los valores de las variables **“MOV” y “MAX_IN”**. Ambas variables toman valor 0 ó 1. El valor de la primera indica qué robot debe seguir el movimiento de los turnos. El valor de la segunda señala si el otro robot ha dejado de detectar el objeto (0), en este caso se pone a cero “MOV”, o sigue detectándolo (1).

Acercamiento del robot hasta detectar máximo

Se comprueban los sensores al principio del bucle y si se obtiene el máximo valor con alguno de los dos sensores, “MAX” cambia su valor a 1. Si los sensores no detectan máximo, el **robot avanza al 50% de la velocidad comprobando los sensores**. **Cuando algún sensor alcanza el valor máximo, “MAX” se pone a 1 y el robot se detiene.**

Envío de máximo

Se **envía el valor de “MAX”** en la posición 7 del vector de variables de salida.

Esperar a recibir máximo

Mientras el robot haya detectado el máximo y no lo haya recibido del otro Moway, permanece **parado**. Va **comprobando** los **sensores** para ver si el objeto se ha alejado y se vuelve a enviar el valor del máximo por si ha variado. Antes de enviarlo se coloca un contador, si éste llega a 10, se ponen “MAX” y “MOV” a 0, para que no se quede un robot en este bucle enviando y el otro recibiendo continuamente.

Turnos para avanzar

Primero se **comprueba el sensor de línea**, y **se para si se detecta línea**. Luego, **si es el turno del robot** (depende del valor de “MOV”), se comprueba el valor de obstáculo; **si se detecta el obstáculo se avanza 5mm. al 30%** de la velocidad **y se modifica** el valor de **“MOV”**, **si no se detecta obstáculo**, **se ponen a cero “MOV” y “MAX”**. Finalmente **se envían los datos**.

Código de “Mov_Obj1_2.c”

```
//Variables auxiliares
static int1 MAX;
static int1 MAX_IN;
static int1 MOV;
static int8 i;
//*************[Rutina de recepción con interrupcion]*************************
//Interrupción del módulo RF
#int_ext
void int_externo()
{
	while(RF_RECEIVE()!=2)
	{
		MOV=RF_DATA_IN[6];
		MAX_IN=RF_DATA_IN[7];
		if(MAX_IN==0)
			MOV=0;
	}
}
//Programa principal
void main()
```

```
{
//**************[Configuración  de sensores, motores y RF]*************************
SEN_CONFIG();
MOT_CONFIG();
//Habilitar interrupciones
RF_INT_EN();
//Configurar módulos SPI del PIC
RF_CONFIG_SPI();
//Configurar módulo RF (canal y dirección)
if(RF_CONFIG(0x40,0x01)==0)
{
	//Módulo no configurado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Activar el módulo RF
if(RF_ON()==0)
{
	//Módulo no inicializado
	LED_TOP_RED_ON();		//Si hay problemas con RF se activa el led sup rojo
}
//Direccion de envio
RF_DIR_OUT=0x02;
//inicio de variables variables
MOV=0;
SEN_OBS_R=0;
SEN_OBS_L=0;
MAX=0;
MAX_IN=0;
i=0;
Delay_ms(10000); //Esperar 10 seg
while(1)
//********************[Acercar el robot hasta que detecte el máximo]*******************
{
if(MAX==0)
{
	//Se comprueban los sensores
	SEN_OBS_ANALOG();
	if(SEN_OBS_R==255 || SEN_OBS_L==255)
		MAX=1;
	else
	{
		//Avance al 50% mientras no se detecte obstaculo
		MOT_STR(50,FWD,TIME,0);
		while(!input(MOT_END))
		{
			SEN_OBS_ANALOG();
			if(SEN_OBS_R==255 || SEN_OBS_L==255)
			{
				MAX=1;
				MOT_STOP();
			}
		}
	}
}
//*****************************[Envío de máximo]**********************
RF_DATA_OUT[6]=MOV;
RF_DATA_OUT[7]=MAX;
ENVIO_DATOS();
//**************[Moway espera a recibir máximo]***********
while(MAX==1 && MAX_IN==0)
{
	MOT_STOP();
```

```
	//Se comproba que se sigue recibiendo el máximo
	SEN_OBS_ANALOG();
	if(SEN_OBS_R==255 || SEN_OBS_L==255)
		MAX=1;
	else
	{
		MAX=0;
		MOV=0;
	}
	//Se utiliza un contador para que el programa no se quede en este bucle
	//mientras el otro robot se queda sólo recibiendo y ambos se queden parados
	i=i+1;
	if(i==10)
	{
		MAX=0;
		MOV=0;
		i=0;
	}
	//Envío
		RF_DATA_OUT[6]=MOV;
		RF_DATA_OUT[7]=MAX;
		ENVIO_DATOS();
}
//*******************[Turnos para avanzar]*********************
//Este moway comienza el movimiento
while(MAX==1 && MAX_IN==1)
{
	//Comprobación sensores de línea
	SEN_LINE_DIG();
	if(SEN_LINE_R==1 && SEN_LINE_L==1)
	{
		MOT_STOP();
		break;
	}
	if(MOV==0)
	{
		//Se comprueba que se sigue recibiendo el máximo
		SEN_OBS_ANALOG();
		if(SEN_OBS_R==255 || SEN_OBS_L==255)
		{
			//Si se detecta el obstáculo se avanza al 30% 0.5 cm.
			MAX=1;
			MOT_STR(30,FWD,DISTANCE,3);
			while(!input(MOT_END)){}
			MOT_STOP();
			MOV=1;
		}
		else
		{
			MAX=0;
			MOV=0;
		}
		RF_DATA_OUT[6]=MOV;
		RF_DATA_OUT[7]=MAX;
		ENVIO_DATOS();
	}
}}}//*****************************************************************************
```

Figura 45: Moways se ayudan a mover una caja

Experimentos

Se comprueba que los programas funcionan, aunque debido a los errores en la odometría los robots no se desplazan en línea recta exactamente.

4.3.- COOPERACIÓN PARA LA LIMPIEZA DE UN ÁREA DETERMINADA

En este caso la tarea a realizar es la limpieza de un área con un número determinado de objetos. Los microbots cooperan para saber el número de objetos restantes en la zona a limpiar. Buscan los objetos y al encontrarlos los expulsan y restan 1 al contador. Cuando el recinto está vacío, los robots se ponen a seguir la línea y al encontrarse, se detienen.

En este apartado se van a explicar los diferentes algoritmos creados y los diferentes experimentos realizados, hasta que finalmente se obtienen unos programas cuyo funcionamiento es idóneo. En primer lugar se trabaja con una zona delimitada por una línea. Pero luego se estima la conveniencia de dividir el área en una zona interior y otra exterior, y también se aumenta el tamaño del recinto.

4.3.1.- Los microbots limpian un área delimitada por una línea

En primer lugar se crean los programas “Limpiar_Area_1.c” y “Limpiar_Area_2.c” para realizar este trabajo cooperativo. En la figura 46 se muestra el recinto utilizado para los experimentos.

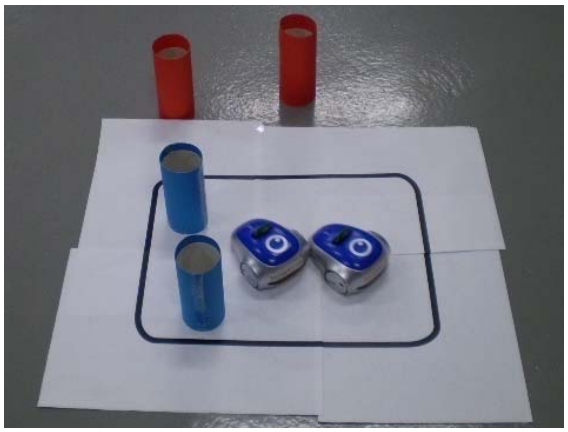

Figura 46: Moways expulsan objetos de un recinto

A continuación se va a explicar el programa **"Limpiar_Area_1.c"**. Este algoritmo contiene 5 funciones: MOWAY_SEARCH, MOWAY_LINE, MOWAY_ATTACK, SACAR_FUERA y Sigue_Lin; más el Main y la rutina de recepción de datos.

"Limpiar_Area_1.c"

MOWAY_SEARCH

Esta función se encarga de comprobar si hay algún objeto delante del robot mediante la función de comprobación de los sensores de obstáculo analógico. Si el valor obtenido en alguno de los sensores es mayor que el valor definido como mínimo para el sensor, la función devuelve 1; en caso contrario, la función devuelve 0. Este algoritmo es el empleado en la práctica Defender que se vio en el capítulo 1. En este caso se ha optado por tomar como valor mínimo para detectar el obstáculo, 100, ya que, como se ha comprobado en la tabla 14, todos los sensores toman un valor superior a 107 cuando el robot se encuentra a unos 2cm. del objeto. Así Moway no detecta al objeto cuando éste se encuentra fuera del recinto (cuando Moway está buscando al resto de objetos o cuando está siguiendo la línea).

MOWAY_LINE

Se encarga de comprobar si el robot se encuentra ante una línea. Para ello se comprueba el sensor de línea de forma analógica. Si se detecta línea con algún sensor Moway gira 108º hacia la derecha. El valor mínimo del sensor de línea se toma como 50, ya que, según la tabla del anexo I, la superficie de un folio tiene un valor máximo de 30 y una línea de luminancia 0 tiene un valor mínimo de 175.

Al principio se utilizó la función de detección de línea digital, pero se producían fallos, y a veces el robot se salía del recinto.

MOWAY_ATTACK

La función se encarga de ir empujando al objeto hasta encontrar la línea. Comprueba cuál es el mayor valor de los sensores de obstáculo y gira 7.2º hacia ese lado. Mientras Moway gira se comprueban los sensores de línea y si se detecta línea con alguno de ellos se le asigna 1 a la variable "sacar" (esta variable indica que se han detectado línea y obstáculo). Si después del giro no se ha detectado la línea (sacar es 0), se empuja al objeto 11.9 mm mientras se comprueba la línea. Del mismo modo que en el giro, si al avanzar se detecta línea, se pone a 1 "sacar".

Se elige una velocidad de avance del 40%, inferior a la de avance mientras se buscan los objetos para que un robot no pueda sacar a otro.

A continuación se muestra el código:

```
void MOWAY_ATTACK()
{
  //Si el obstáculo está en la izquierda
  if(SEN_OBS_L>SEN_OBS_R)
  {
    //Rotar 7.2º a la izquierda y mientras se comprueba la línea
    MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,2);
    while(!input(MOT_END))
    {
      SEN_LINE_ANALOG();
            //Si se encuentra línea llama a función para expulsar a Moway unos cms
            if(SEN_LINE_L>=senLinMin|| SEN_LINE_R>=senLinMin)
                  sacar=1;
```

```
    }
  }
  else if(SEN_OBS_R>SEN_OBS_L && sacar==0)
  {
    //Rota 7.2º a la derecha y mientras comprueba la línea
    MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,2);
    while(!input(MOT_END))
    {
      SEN_LINE_ANALOG();
            //Si se encuentra línea se llama a función para expulsar unos cms
            if(SEN_LINE_L>=senLinMin|| SEN_LINE_R>=senLinMin)
                  sacar=1;
    }
  }
 if(sacar==0)
 {
  //Golpea el objeto 11.9mm mientras comprueba la línea
  MOT_STR(40,FWD,DISTANCE,7);
  while(!input(MOT_END))
  {
            SEN_LINE_ANALOG();
            //Si se encuentra línea llamar a función para expulsar unos cms
            if(SEN_LINE_L>=senLinMin|| SEN_LINE_R>=senLinMin)
                  sacar=1;
  }   }}
```

SACAR_FUERA

Esta función se encarga de sacar al objeto fuera del recinto unos centímetros. Se necesita que el objeto salga una cierta distancia del área porque de lo contrario pueden producirse problemas:

- El robot puede detectar un objeto al que ha sacado y volver a empujarlo y contarlo, por lo que no se expulsarían todos los objetos del recinto y los programas no funcionarían correctamente.
- Moway puede detectar como obstáculo a los objetos expulsados cuando está siguiendo la línea y pararse antes de que ambos robots se encuentren.

Por eso se ha creado esta función que aleja a los objetos de la línea que delimita el recinto. A continuación se explica detalladamente este algoritmo.

El primer bucle “while” se encarga de que el robot avance, girando hacia donde se encuentra el objeto, hasta que deje de detectarse la línea. Se van comprobando los sensores de obstáculo y el robot gira 7.2º hacia el lado del sensor cuyo valor es mayor. Tras girar se comprueba la línea: si se detecta con algún sensor, se avanza 1.7mm.; si no se detecta línea con ninguno, se pone a la variable “sacar” a 0, ya que se ha sacado al objeto de la línea y se acaba el bucle.

Mediante las variables “izq” y “der” se tiene en cuenta el sentido del último movimiento del robot antes de salir de la línea: si el último movimiento del robot es a la derecha, “der” toma el valor 1; si el último movimiento es la izquierda, “izq” pasa a valer 1; y si el último movimiento es recto, las 2 variables son 0.

Con el siguiente bucle “for” se expulsa al objeto unos 8 centímetros del recinto. Como en el bucle anterior, el robots va girando hacia el objeto 7.2º y luego avanza 0.5cm. Esto se repite 16 veces (8cm.). En los giros se van aumentado las variables “izq” y “der” para saber el ángulo exacto con el que el robot sale del recinto.

Después se gira 180º teniendo en cuenta los ángulos que se han ido girando, se reinician las variables y se avanza unos 9cm. El robot avanza unas 27 veces 3.4mm, o hasta detectar línea. Tras cada movimiento, se comprueba el sensor de línea. En el

momento en el que se detecta línea con alguno de los sensores se avanza 15.3 mm. (suficiente para traspasar la línea) y se termina el bucle.

Finalmente se le resta 1 al contador y se envía en la posición 7 del vector de variables de salida.

Se ha seleccionado una velocidad elevada para expulsar al objeto de la línea para que éste se quede lo más alejado posible del recinto.

A continuación se muestra el código:

```
void SACAR_FUERA()
{
	//Mientras detecte línea avanza 1.7mm. girando hacia el objeto y comprobando línea
	while(sacar==1)
	{
		//Gira para empujar mejor al objeto
		SEN_OBS_ANALOG();
		//Si el obstáculo está en la izquierda rota 7.2º a la izquierda
		if(SEN_OBS_L>SEN_OBS_R)
		{
			MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,2);
			while(!input(MOT_END)){}
			izq=1;
		 }
		else if(SEN_OBS_R>SEN_OBS_L)
		{
			//Rota 7.2º a la derecha
			MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,2);
			while(!input(MOT_END)){}
			der=1;
		}
		//Comprueba línea
		SEN_LINE_ANALOG();
		if(SEN_LINE_L>=senLinMin|| SEN_LINE_R>=senLinMin)
		{
			izq=0;
			der=0;
			MOT_STR(80,FWD,DISTANCE,1);
			while(!input(MOT_END)){}
		}
		else if(SEN_LINE_L<senLinMin && SEN_LINE_R<senLinMin)
			sacar=0;
	}
	//Sacar al objeto unos 8cm del recinto
	for(i=0; i<16; i++)
	{
		//Girar para empujar mejor al objeto
		SEN_OBS_ANALOG();
		//Si el obstáculo está en la izquierda rota 7.2º a la izquierda
		if(SEN_OBS_L>SEN_OBS_R)
		{
			MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,2);
			while(!input(MOT_END)){}
			izq=izq+1;
		}
		 else if(SEN_OBS_R>SEN_OBS_L)
		{
		//Rota 7.2º a la derecha
		MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,2);
			while(!input(MOT_END)){}
			der=der+1;
		}
```

```
            //Avanzar 0.5cm
            MOT_STR(80,FWD,DISTANCE,3);
            while(!input(MOT_END)){}
      }
      //Volver al recinto y decrementar el contador, girar 180ºteniendo en cuenta giros
      angulo=angulo-2*der+2*izq;
      MOT_ROT(50,FWD,CENTER,RIGHT,ANGLE,angulo);
      while(!input(MOT_END)){}
      //Las variables vuelven a su valor inicial
      der=0;
      izq=0;
      angulo=50;
      //Avanzar unos 9cm
      for(i=0; i<27; i++)
      {
            //Avanzar 3.4mm
            MOT_STR(40,FWD,DISTANCE,2);
            while(!input(MOT_END)){}
            SEN_LINE_ANALOG();
            //Si encuentra línea avanza 15.3mm
            if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
            {
                  MOT_STR(40,FWD,DISTANCE,9);
                  while(!input(MOT_END)){}
                  i=27;
            }
      }
      //Decrementar el contador y enviarlo
      contador=contador-1;
      RF_DATA_OUT[7]=contador;
      ENVIO_DATOS();
}
```

Sigue_Lin

Esta función se encarga de seguir la línea por el borde izquierdo. Para ello comprueba los sensores de línea de forma digital y según su valor realiza un tipo de movimiento como se ha visto en el programa “RC1.c”.

Un robot sigue la línea con movimientos a velocidades elevadas (no se pueden poner velocidades muy altas porque entonces el seguimiento de línea falla) y el otro, a velocidades bajas. Se toma una diferencia de velocidades grande entre ambos robots para que se encuentren en poco tiempo cuando siguen línea.

Main

El programa principal se divide en 2 secciones importantes: **Sacar los objetos del recinto** y **seguir la línea hasta encontrarse.** A continuación se explican ambas.

Sacar los objetos del recinto

Mientras no se hayan sacado todos los objetos (contador sea mayor que 0) y la variable “sacar” sea 0 (esta variable vale 1 cuando se detectan objeto y línea a la vez), se comprueba si hay línea, girando en su caso un ángulo de 108º. Si no se encuentra la línea, se comprueba si hay obstáculo, si hay se le ataca, y si no, se sigue recto.

Cuando “sacar” es 1 y el contador aún no es 0, se llama a la función SACAR_FUERA para que empuje al objeto unos centímetros fuera de la línea.

La función se muestra en las siguientes líneas:

```
//***********[Sacar todos los objetos del recinto]***************
while(contador>0)
{
	//Se detecta objeto pero no linea o no se detecta objeto
	if(sacar==0)
	{
		//Comprueba la línea y la evita
		MOWAY_LINE();
		//Si existe enemigo Moway lo ataca y si no, sigue recto patrullando.
		if (MOWAY_SEARCH())
			MOWAY_ATTACK();
		else
			MOT_STR(60,FWD,TIME,0);
	}
	//Si se ha detectado objeto y se ha llegado el borde del recinto se saca el objeto unos cms
	else
		SACAR_FUERA();
}
```

Seguir la línea hasta encontrarse

Tras eliminar a todos los objetos (el contador ha llegado a cero) los robots se ponen a seguir la línea. Para ello, primero deben buscarla, ya que uno de los dos puede que no se encuentre sobre la línea. Por eso, mientras no se encuentra línea, van avanzando 5mm. hacia adelante y se comprueba la línea. Cuando se detecta, la variables “línea” pasa a ser 1.

Antes de comenzar a seguir la línea se vuelve a enviar que el contador es cero, por si se ha producido algún error en el envío.

Cuando se encuentra la línea, pero aún no se ha detectado obstáculo, se comprueba si hay obstáculo. En caso de encontrarse con el otro robot, Moway se detiene, “parada” pasa a valer 1 y se envía en la dirección 6 del vector de variables de salida. Mientras no se detecte obstáculo Moway sigue la línea con la función Sigue_Lin.

Finalmente, si se recibe que el otro robot ha parado (“parada” es 1), Moway se detiene.

```
//**********[Seguir línea hasta encontrarse]**********************
while(contador==0)
{
	//Enviar el valor del contador
	RF_DATA_OUT[7]=contador;
	ENVIO_DATOS();
	//Primero se busca la línea
	while(linea==0)
	{
		//Avanzar 5 mm. y comprobar linea
		MOT_STR(60,FWD,DISTANCE,3);
		while(!input(MOT_END))
		{
			 SEN_LINE_ANALOG();
			if(SEN_LINE_L>=senLinMin|| SEN_LINE_R>=senLinMin)
				linea=1;
		}
	}
	//Cuando se encuentra línea, Moway la sigue hasta encontrar obstaculo
	while(parada==0 && linea==1)
	{
		SEN_OBS_DIG();
		//Parar al detectar obstáculo y se le comunica al otro robot que pare
```

```
            if (MOWAY_SEARCH())
            {
                MOT_STOP();
                parada=1;
                RF_DATA_OUT[6]=parada;
                ENVIO_DATOS();
            }
            //Seguir la línea si no se encuentra obstaculo
            else
                Sigue_Lin();
        }
        //Si el otro robot ha parado, Moway se detiene
        if(parada==1)
            MOT_STOP();
}
```

Rutina de recepción de datos

Se actualizan los valores del contador y de la variable parada.

El contenido de "Limpiar_Area_2.c" es similar, cambiando únicamente la dirección de los robots y la función de seguimiento de línea, que en éste toma velocidades menores.

Experimentos

Los programas funcionan tal como se esperaba aunque a veces se producen algunos errores que a continuación se detallan:

- Ambos robots pueden sacar el mismo objeto a la vez y contarlo 2 veces.
- Los robots pueden sacar a los objetos al empujarlos lateralmente, sin detectarlos, y no contarlos.
- A veces un robot expulsa a otro.
- Se pueden producir errores al contar los objetos.
- A veces el robot no encuentra la línea tras expulsar el objeto.
- Los robots pueden salir del recinto al encontrarse con la línea, pero sin detectar un objeto.

Para solventar estos problemas se modifican los programas anteriores y se les asigna los nombres **"Limpiar_Area2_1.c" y "Limpiar_Area2_2.c"**. En este caso se realizan los experimentos con 4 objetos y sobre una cartulina verde cuya línea es una cinta aislante azul. Se toma 60 como valor del sensor de línea mínimo, que es el valor mínimos que toma el rango ampliado de los sensores de línea.

A continuación se explican las modificaciones que se han realizado.

"Limpiar_Area2_1.c"

MOWAY_LINE

Se observa que hay veces que el robot al encontrar línea y girar a la derecha se sale del recinto. Por eso, se cambia esta función para que gire a la derecha en el caso de que detecte con el sensor izquierdo (sólo sensor izquierdo o sensor izquierdo y derecho); y gire a la derecha si detecta sólo con el sensor derecho.

```
void MOWAY_LINE()
{
  SEN_LINE_ANALOG();
  //Si se encuentra línea con sensor izquierdo, rota 108º a la derecha.
  if(SEN_LINE_L>=senLinMin)
  {
    MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,30);
    while(!input(MOT_END)){}
  }
      //Si no, si se encuentra línea con sensor derecho rota 108º a la izquierda.
      else if(SEN_LINE_R>=senLinMin)
  {
    MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,30);
    while(!input(MOT_END)){}
  }
}
```

MOWAY_ATTACK

Se le añade que el robot pare si detecta línea y automáticamente se salga de la función con el comando "return". También se disminuye la velocidad a la que se empuja al objeto, para evitar expulsar al otro robot.

SACAR_FUERA

Como ahora las anchuras de la línea y de la zona exterior son mayores, se debe modificar esta función.

Al disminuir el contador se tiene en cuenta que se puede haber producido algún error al contar los objetos. Si el contador era 0, tras expulsar el objeto sigue siendo 0.

Se toma como ángulo de giro para seguir al objeto 3.6º y se va comprobando la existencia de línea durante el giro. Se va sumando 1 a las variables "izq" y "der" en todo momento, aunque no se haya dejado de detectar línea. El avance para salir de la línea se realiza a velocidad muy baja para poder detectar que se ha salido del recinto.

Tras dejar de detectar línea, el robot avanza unos 6cm. siempre girando hacia el objeto. Mientras gira, se comprueba la existencia de línea, por si al girar Moway vuelve a entrar al recinto en lugar de salir. Si ocurre esto, Moway se detiene, la variable "salir" vuelve a valer 1 y se sale del bucle y de la función, a la que se volverá a llamar en el siguiente ciclo de programa.

Se comprueba si el ángulo resultante de sumar a 180º los giros realizamos es mayor de 360º, en cuyo caso se toma el mismo ángulo comprendido entre 0 y 360º.

Para entrar, primero Moway rota el ángulo calculado, comprobando si encuentra línea, si se detecta línea, la variable "linea2" pasa a valer 1. Se añade esto porque, en ocasiones, Moway entra al recinto sin detectar línea y luego cuando la detecta se sale de él. Así Moway ya no se sale del recinto.

Si tras girar no se ha detectado línea, entonces se avanza recto hasta detectarla. Después se avanza recto hasta dejar de detectar línea y luego se avanza 0.5cm. comprobando línea por si el robot no ha entrado del todo al recinto. En este último caso, el robot vuelve otra vez a avanzar hasta que no detecte la línea.

Con estos cambios se ha conseguido disminuir los errores. Pero sigue ocurriendo que un robot puede expulsar a otro. Para mejorar este problema se considera que se deben diferenciar la zona exterior de la interior y el recinto debe tener un tamaño mayor.

4.3.2.- Los microbots limpian una zona mayor con zonas interior y exterior diferenciadas

Se decide dividir la zona de trabajo de los robots en una zona interior y otra exterior, para así saber en todo momento en que zona se encuentra el robot. Con esto se pretende eliminar los errores: por un lado que un robot no pueda expulsar a otro, por otro lado que los 2 robots no puedan expulsar el mismo objeto.

Se opta por tomar objetos cilíndricos, ya que así es casi imposible que un robot expulse 2 objetos a la vez y cuente que sólo ha sacado uno.

Por lo tanto para los sucesivos experimentos, se emplean cartulinas con una zona interior de diferente luminancia que la exterior y para delimitarlas se emplea una línea de cinta aislante. Como objetos se utilizan cilindros de cartulina.

Se crean 2 proyectos, cada uno es una mejora del anterior y a continuación se van a explicar brevemente las mejoras del primero, y finalmente se explicará el último proyecto más en profundidad.

“Limpiar_Area3_1.c” y “Limpiar_Area3_2.c”

Los experimentos de estos programas se realizan en una cartulina con una luminancia interior de 50. Los cambios más significativos de este programa con respecto al anterior son los siguientes:

- Los sensores de línea tienen en cuenta 3 zonas diferentes. La zona exterior toma un valor máximo de 25, la línea toma valores entre 60 y 160, y la zona interior toma un valor mínimo de 175 (en la siguiente figura se aprecian las diferentes zonas junto a sus respectivos valores del sensor de línea). Se modifican todas las funciones que emplean el sensor de línea para poder diferenciar las zonas.

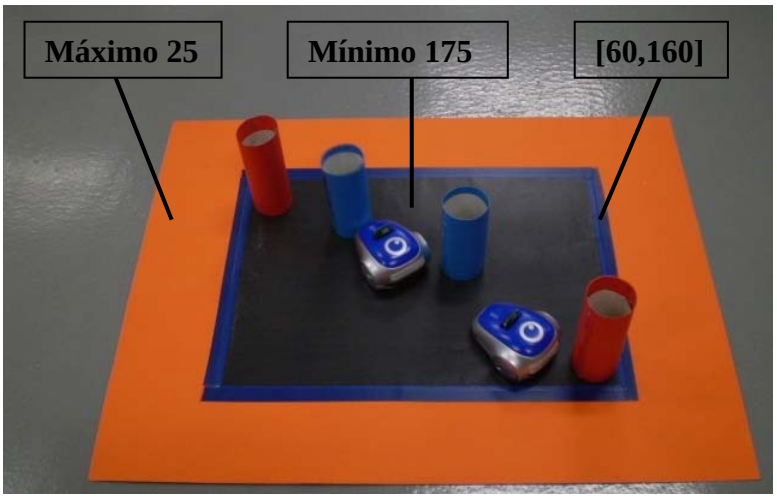

Figura 47: Robots expulsando objetos, valores de las diferentes zonas

- Se crea una función para volver a la zona interior en caso de que el robot se encuentre en la zona exterior cuando no le corresponde. “Entrar” se encarga de

que el robot vuelva al recinto si ha sido empujado por el otro robot o si tras expulsar un objeto no consigue localizar la línea.

- Como a veces los robots expulsan al mismo objeto o un robot expulsa al otro y se producen errores en el contador, se modifican los programas para que sólo uno de los 2 microbots pueda estar en la zona exterior. Cuando un robot se encuentra en la zona de la cartulina, el otro se encarga de patrullar evitando los objetos y la línea.

Finalmente se crean los programas **"Limpiar_Area4_1.c"** y **"Limpiar_Area4_2.c"**. En este caso las pruebas se hacen en un área mayor, la zona está compuesta por 4 cartulinas cuyo interior lo forman cuatro A3 de luminancia 100, divididas por una línea de cinta aislante. A continuación se van a exponer las características principales de estos programas y después se explicará más detenidamente su contenido.

"Limpiar_Area4_1.c" y "Limpiar_Area4_2.c"

Los robots avanzan buscando los objetos y evitando la línea. Cuando un robot encuentra un objeto lo expulsa unos centímetros de la línea y vuelve a entrar. Al expulsar un objeto, un contador se actualiza en ambos robots. Si un robot está fuera, el otro sigue buscando un objeto, y si lo encuentra, se detiene hasta que el otro Moway consigue entrar. Si en algún momento un robot se desorienta y está en la zona exterior, vuelve dentro del recinto. Tras expulsar todos los objetos los robots buscan la línea, la siguen y se detienen al encontrarse.

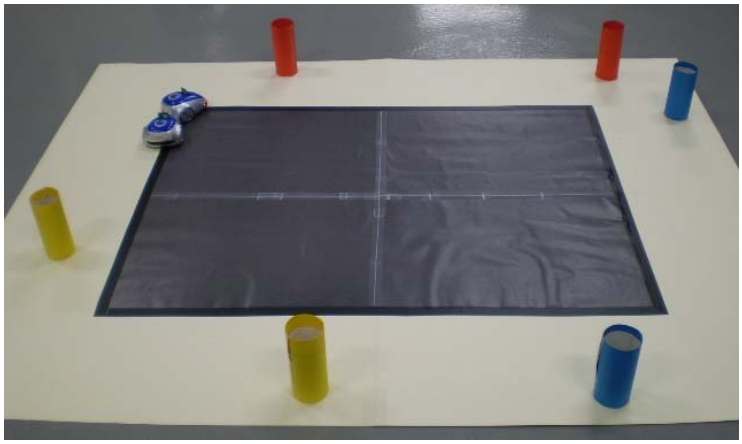

Figura 48: Moways siguen la línea tras sacar los 6 objetos del recinto

Como se observa en la figura anterior, la zona de trabajo se divide en tres partes: zona exterior de cartulina beige, el sensor de línea la detecta con un valor máximo de 25; zona interior en color de luminancia 100, cuyos valores varían entre 130 y 195; y línea formada por cinta aislante negra, con valores mayores de 210. Por lo tanto se definen las siguientes variables para poder trabajar con los sensores: senFueraMax=25, senAreaMin=130, senAreaMax=195 y senLinMin=210.

"Limpiar_Area4_1.c"

El programa contiene seis funciones (MOWAY_SEARCH, MOWAY_LINE, MOWAY_ATTACK, SACAR_FUERA, Entrar y Sigue_Lin), la rutina de recepción de datos y el programa principal. A continuación se explica cada una de estas secciones.

MOWAY_SEARCH

Esta función se encarga de comprobar **si** se **detecta obstáculo de forma analógica**. Si el valor de alguno de los sensores es mayor que el valor de la variable "senObsMin", se **devuelve 1**; en caso contrario se devuelve 0. Se ha tomado como valor mínimo de obstáculo 180 para asegurar que los robots sólo detectan objetos del interior.

Antes de comprobar el sensor de obstáculo, se comprueba el sensor de línea y en caso de estar en el exterior se llama a la función "Entrar". Así Moway vuelve al recinto si se ha salido.

```
int1 MOWAY_SEARCH()
{
	SEN_LINE_ANALOG();
	//Si se detecta el exterior con ambos sensores se vuelve a la zona interior
	if(SEN_LINE_L<=senFueraMax && SEN_LINE_R<=senFueraMax)
	{
		Entrar();
		return(false);
	}
	SEN_OBS_ANALOG();
	if(SEN_OBS_L>=senObsMin || SEN_OBS_R>=senObsMin)
		return(true);
	else
		return(false);
}
```

MOWAY_LINE

Esta función se encarga de **girar en caso de encontrar línea** mientras se está buscando objetos. Se comprueban los sensores de línea de forma analógica, y según su valor se realiza una acción u otra. Se pueden dar 4 casos:

1) **El robot se encuentra fuera del recinto** (se detecta el exterior con algún sensor) → **Se llama a la función Entrar**.
2) El robot **detecta** la **línea con ambos sensores → Se rota hasta detectar el interior con ambos sensores**.
3) **Se detecta línea con el sensor izquierdo → Se gira** 108º **a la derecha**.
4) **Se detecta línea con el sensor derecho → Se gira** 108º **a la izquierda**.

```
void MOWAY_LINE()
{
    SEN_LINE_ANALOG();
      //Si se detecta el exteriror se llama a Entrar
      if(SEN_LINE_L<=senFueraMax || SEN_LINE_R<=senFueraMax)
            Entrar();
      //Si se detecta línea con ambos sensores, se gira hasta detectar interior
      else if(SEN_LINE_L>=senLinMin && SEN_LINE_R>=senLinMin)
      {
            //Girar 360º o hasta entrar
            MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,100);
            while(!input(MOT_END))
            {
```

```
                    SEN_LINE_ANALOG();
                    if((SEN_LINE_L>=senAreaMin &&
SEN_LINE_L<=senAreaMax)&&(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                            MOT_STOP();
                }
        }
        //Si se encuentra línea con sensor izquierdo, rotar 108º a la derecha.
        else if(SEN_LINE_L>=senLinMin)
          {
                MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,30);
                while(!input(MOT_END)){}
        }
        //Si no, si se encuentra línea con sensor derecho, rotar 108º a la izquierda.
        else if(SEN_LINE_R>=senLinMin)
        {
                MOT_ROT(70,FWD,CENTER,LEFT,ANGLE,30);
               while(!input(MOT_END)){}
         }
}
```

MOWAY_ATTACK

La función se encarga de **ir empujando al objeto que ha detectado**. Primero se comprueba hacia qué lado se encuentra el objeto y **se gira hacia él**. **Y** luego **avanza 5mm.** empujándolo.

Mientras se realizan los giros y el avance, se va comprobando el sensor de línea. **Si se detecta línea**, el robot **se detiene**, **se modifica "sacar" a 1**, **se envía** esta variable al otro robot **y se sale de la función**. Con este envío se conoce en todo momento si el otro robot ha salido del área.

Se toma una velocidad de avance muy baja (20%) para evitar que un robot expulse a otro.

```
void MOWAY_ATTACK()
{
        SEN_OBS_ANALOG();
        //Si el obstáculo está en la izquierda rotar 7.2º a la izquierda y mientras se comprueba la línea
        if(SEN_OBS_L>SEN_OBS_R)
        {
             MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,2);
              while(!input(MOT_END))
              {
                    SEN_LINE_ANALOG();
                    //Si se encuentra línea se detiene y pone sacar a 1
                    //Entonces se llama a la función SACAR_FUERA para sacar al objeto unos cms del área
                    if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                    {
                          MOT_STOP();
                          sacar=1;
                          RF_DATA_OUT[5]=sacar;
                          ENVIO_DATOS();
                          //Salir de la función
                          return;
                    }
              }
        }
         else if(SEN_OBS_R>SEN_OBS_L)
        {
                //Rotar 7.2º a la derecha y mientras se comprueba la línea
               MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,2);
```

```
            while(!input(MOT_END))
            {
                  SEN_LINE_ANALOG();
                  //Si se encuentra línea se detiene y pone sacar a 1
                  if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                  {
                        MOT_STOP();
                        sacar=1;
                        RF_DATA_OUT[5]=sacar;
                        ENVIO_DATOS();
                        //Salir de la función
                        return;
                  }
            }
      }
      //Se golpea el objeto 5mm mientras se comprueba línea
       MOT_STR(20,FWD,DISTANCE,3);
       while(!input(MOT_END))
      {
            SEN_LINE_ANALOG();
            //Si se encuentra línea se detiene y pone sacar a 1
            if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
            {
                  MOT_STOP();
                  sacar=1;
                  RF_DATA_OUT[5]=sacar;
                  ENVIO_DATOS();
                  //Salir de la función
                  return;
            }
      }
}
```

SACAR_FUERA

Esta función se encarga de **sacar fuera del recinto unos 10cm. el objeto**, de **disminuir el contador y** de que **Moway regrese al recinto**.

Primero **se avanza hasta dejar de detectar la línea** (sacar=1). **Si** el robot **ha dejado de detectar el objeto**, se pone "sacar" a 0, se **vuelve al recinto** mediante la llamada a la función **Entrar** y se vuelve al programa principal; en este caso se comprueba que ambos sensores de obstáculos sean menores que "senObsMin2" que se define como 80, se toma este valor menor por si el objeto está un poco alejado. En caso de que **se siga detectando el objeto**, el robot **va girando hacia él** (según el valor de los sensores de obstáculo) **3.2º** si el objeto no está centrado delante de él. En el giro se comprueba el sensor de línea y si se detecta el exterior se para y se pone "sacar" a 0. Tras girar se aumenta en uno la variable "izq" o "der" según el giro realizado. Después de girar si ha sido necesario, se **avanza recto** a muy baja velocidad **hasta detectar el exterior**, se selecciona una baja velocidad para que el robot no pierda de su ángulo de visión al obstáculo.

Luego **se avanza 20 veces unos 0.5cm. para expulsar al objeto del recinto** unos 10cm. y no detectarlo en la búsqueda de otro objeto o cuando se está siguiendo línea. Como en el caso anterior, se comprueba si se ha perdido el objeto de delante y **se va girando 3.2º hacia el objeto**. Si el objeto deja de detectarse cuando se ha avanzado 1cm. o más, entonces se cuenta como expulsado, se envía el contador y que el robot está fuera, se llama a la función Entrar y se regresa al programa principal.

Luego **se resta 1 al contador y se envía, el robot gira y entra al recinto**. Tras contar el objeto y enviar el contador, se **gira 180º teniendo en cuenta los giros realizados**. Mientras se gira se va comprobando el sensor de obstáculo por si se detecta la línea, en ese caso "linea2" se pone a 1. Luego se **avanza 13.5cm. o hasta encontrar la línea con algún sensor**; si tras avanzar no se encuentra la línea, se llama a la función Entrar y se vuelve al programa principal.

Tras detectar línea se avanza 4.25cm. o hasta detectar el interior con ambos sensores. **Si** tras recorrer los 4.25cm. se **detecta el interior sólo con un sensor**, entonces el robot **gira hacia ese lado hasta encontrar el interior con ambos**. Por último se avanzan 5mm. para conseguir entrar del todo y si no se consigue, se llama a la función Entrar. **Cuando ha entrado** al recinto se **vuelva a avanzar 5mm. comprobando los sensores por si no se ha entrado realmente** al recinto o si el robot se encuentra en una esquina y puede volver a salirse.

```
void SACAR_FUERA()
{
	//Se avanza comprobando línea y obtáculo hasta salir por completo de la línea
	while(sacar==1)
	{
		//Girar hacia el objeto
		SEN_OBS_ANALOG();
		//Si ha perdido al obstáculo entra
		if(SEN_OBS_L<senObsMin2 && SEN_OBS_R<senObsMin2)
		{
			Entrar();
			sacar=0;
			return;
		}
		//Si el obstáculo está en la izquierda rotar 3.6º a la izquierda
		else if(SEN_OBS_L>SEN_OBS_R)
		{
			MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,1);
			while(!input(MOT_END))
			{
				SEN_LINE_ANALOG();
				if(SEN_LINE_L<senFueraMax && SEN_LINE_R<senFueraMax)
				{
					MOT_STOP();
					//El objeto ha salido del recinto
					sacar=0;
				}
			}
			izq=izq+1;
		}
		else if(SEN_OBS_R>SEN_OBS_L)
		{
			//Rotar 3.6º a la derecha
			MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,1);
			while(!input(MOT_END))
			{
				SEN_LINE_ANALOG();
				if(SEN_LINE_L<senFueraMax && SEN_LINE_R<senFueraMax)
				{
					MOT_STOP();
					//El objeto ha salido del recinto
					sacar=0;
				}
			}
```

```
                der=der+1;
        }
        //Avanzar despacio hasta dejar de detectar línea
        MOT_STR(10,FWD,TIME,0);
        while(!input(MOT_END))
        {
                SEN_LINE_ANALOG();
                if(SEN_LINE_L<senFueraMax && SEN_LINE_R<senFueraMax)
                {
                        MOT_STOP();
                        //El objeto ha salido del recinto
                        sacar=0;
                }
        }
}
//Se saca al objeto unos 10.2cm del recinto, se comprueba linea para ver si se ha entrado al girar
for(i=0; i<20; i++)
{
        //Se gira para empujar mejor al objeto
        SEN_OBS_ANALOG();
        //Si se ha perdido al obstáculo se vuelve al recinto
        if(SEN_OBS_L<senObsMin2 && SEN_OBS_R<senObsMin2)
        {
                //Si ya se ha sacado al objeto un buen trozo se contabiliza
                if(i>2)
                {
                        //Decrementar el contador y  enviarlo
                        if(contador!=0)
                                contador=contador-1;
                        //Enviar que aún se está fuera
                        RF_DATA_OUT[5]=1;
                        RF_DATA_OUT[4]=1;
                        RF_DATA_OUT[7]=contador;
                        ENVIO_DATOS();
                        RF_DATA_OUT[4]=0;
                }
                Entrar();
                return;
        }
        //Si el obstáculo está en la izquierda rotar 3.6º a la izquierda
        else if(SEN_OBS_L>SEN_OBS_R)
        {
                MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,1);
                while(!input(MOT_END))
                {
                        //Si detecta línea es que aún no se ha salido del todo y se vuelve a sacar al objeto
                        SEN_LINE_ANALOG();
                        if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                        {
                                MOT_STOP();
                                sacar=1;
                                i=20;
                        }
                }
                izq=izq+1;
        }
         else if(SEN_OBS_R>SEN_OBS_L)
        {
                //Rotar 3.6º a la derecha
                MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,1);
                while(!input(MOT_END))
                {
```

```
                    SEN_LINE_ANALOG();
                    if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                    {
                        MOT_STOP();
                        sacar=1;
                        i=20;
                    }
                }
                der=der+1;
            }
            if(sacar==0)
            {
                //Avanzar 0.5cm
                MOT_STR(100,FWD,DISTANCE,3);
                while(!input(MOT_END))
                {
                    SEN_LINE_ANALOG();
                    if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                    {
                        MOT_STOP();
                        sacar=1;
                        i=20;
                    }
                }
            }
        }
        if(sacar==0)
        {
            //Decrementar el contador y  enviarlo, enviar tb que está fuera
            if(contador!=0)
                contador=contador-1;
            RF_DATA_OUT[5]=1;
            RF_DATA_OUT[4]=1;
            RF_DATA_OUT[7]=contador;
            ENVIO_DATOS();
            RF_DATA_OUT[4]=0;
            //Volver al recinto
            //Giro de 180º teniendo en cuenta los giros que se han realizado al orientarse hacia el objeto
            angulo=angulo-der+izq;
            if(angulo>100)
            {
                //Se obtiene el número entero de vueltas que se han dado
                i=angulo/100;
                angulo=angulo-100*i;
            }
            MOT_ROT(20,FWD,CENTER,RIGHT,ANGLE,angulo);
            while(!input(MOT_END))
            {
                SEN_LINE_ANALOG();
                //Si se encuentra línea se cambia linea2 a 1
                if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                    linea2=1;
            }
            //Las variables vuelven a su valor inicial
            der=0;
            izq=0;
            angulo=50;
            if(linea2==0)
            {
                //Avanzar unos 13.5cm o hasta encontrar la línea
                MOT_STR(40,FWD,DISTANCE,80);
                while(!input(MOT_END))
```

```
                {
                        SEN_LINE_ANALOG();
                        //Si se encuentra línea se cambia linea2 a 1
                        if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                        {
                                linea2=1;
                                MOT_STOP();
                        }
                }
                if(linea2==0)
                {
                        Entrar();
                        return;
                }
        }
        //Avanzar hasta detectar interior
        while(linea2==1)
        {
                //Avanzar 4.25cm o hasta detectar con algún sensor zona interior
                MOT_STR(40,FWD,DISTANCE,25);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax) &&
(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                        {
                                //Se ha entrado al recinto
                                linea2=0;
                                MOT_STOP();
                        }
                }
                if(linea2==1)
                {
                        SEN_LINE_ANALOG();
                        //Si se detecta interior sólo con un sensor, se gira hasta detectarlo con ambos
                        if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax) ||
(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                        {
                                MOT_STOP();
                                if(SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax)
                                        RL=LEFT;
                                else if(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax)
                                        RL=RIGHT;
                                //Girar 360º o hasta encontrar interior con ambos
                                MOT_ROT(20,FWD,CENTER,Rl,ANGLE,100);
                                while(!input(MOT_END))
                                {
                                        SEN_LINE_ANALOG();
                                        //Si se detecta interior linea2=0 y se para
                                        if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax)
&& (SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                                        {
                                                linea2=0;
                                                MOT_STOP();
                                        }
                                }
                        }
                        if(linea2==1)
                        {
                                //Si tras girar no se ha encontrado el interior se avanzan 5mm
                                MOT_STR(40,FWD,DISTANCE,3);
                                while(!input(MOT_END))
```

```
                        {
                            SEN_LINE_ANALOG();
                            if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax)
&& (SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                            {
                                //Se ha entrado al recinto
                                linea2=0;
                                MOT_STOP();
                            }
                        }
                    }
                }
                //Si tras avanzar 4.25cm no se ha detectado interior, se llama a Entrar
                if(linea2==1)
                {
                    linea2=0;
                    Entrar();
                    return;
                }
                //Antes de salir del bucle se avanza para ver que ralmente se ha entrado
                if(linea2==0)
                {
                    //Avanzar 0.5cm comprobando línea
                    MOT_STR(10,FWD,DISTANCE,3);
                    while(!input(MOT_END))
                    {
                        SEN_LINE_ANALOG();
                        if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                        {
                            //Aún no se ha entrado del todo en el recinto
                            linea2=1;
                            MOT_STOP();
                        }
                        //Se comprueba si se ha salido del recinto
                        else if(SEN_LINE_L<=senFueraMax && SEN_LINE_R<=senFueraMax)
                        {
                            Entrar();
                            return;
                        }
                    }
                }
            }
            //Se envía que se ha entrado
            fuera=0;
            RF_DATA_OUT[5]=fuera;
            ENVIO_DATOS();
        }
}
```

Entrar

La función se encarga de que el **robot consiga entrar al recinto** si se ha salido por cualquier motivo. **Primero** se realizan **2 giros para ver si se detecta el interior del recinto o la línea**. **Después** se entra en un bucle en el que se realizan un **ciclo de movimientos que van aumentando su distancia de avance cada vez** para localizar la línea. **Tras detectar la línea se avanza hasta estar dentro** del recinto y, finalmente, se envía que se ha entrado.

Primero se comprueba el sensor de línea para ver si realmente alguno de los sensores se encuentra fuera. En caso de estar fuera se pone la variable “fuera” a 1 (que indica que el robot está en el exterior) y se envía. Por si el robot está muy cerca de la

línea se realiza un giro de 360º comprobando si el robot detecta el interior con ambos sensores, en cuyo caso se detiene y se envía que ha entrado, saliendo de la función. Si tras el giro no se detecta el interior, se da otra vuelta en busca de la línea, si se detecta línea se pone "fuera" a 0 y "linea3" a 1 (esta variable sirve para después avanzar sobre la línea hasta detectar el interior).

Mientras no encuentra la línea el robot realiza el siguiente **ciclo de movimientos**: **Avanzar hacia adelante "dist"** (esta variable comienza valiendo 2cm. y va aumentando 2cm. cuando el robot realiza 2 veces esta serie de movimientos), **girar 360º**, **retroceder 2 veces "dist"**, **girar 360º**, **avanzar "dist" y girar 90º**. En todos estos movimientos se comprueba si se detecta línea con algún sensor y si se detecta, Moway se detiene, "fuera" pasa a ser 0 y "linea3" pasa a ser 1, para entrar en el siguiente bucle. Se realizan estos movimientos porque así se va avanzando en cada cuadrante y Moway puede encontrar en pocos movimientos la línea.

Cuando se ha detectado la línea se realizan una serie de movimientos para entrar del todo al recinto. **Se avanza 3.5cm. hacia adelante**. Mientras se realiza este movimiento, se van comprobando los sensores de línea: **si se detecta el interior con ambos, el robot se detiene** y "línea3" se pone a 0; **si se detecta el interior con un sensor, se realiza un giro hacia ese lado hasta que se detecta el interior con ambos** sensores; **si** tras girar **no se detecta el interior con ambos sensores, avanza 5mm.** comprobando si se ha detectado el interior con ambos sensores. **Si** después de estos movimientos **sigue sin encontrarse el interior**, se hace lo siguiente: **se gira 360º o hasta detectar el interior con ambos sensores y se gira 360º o hasta detectar el interior con algún sensor**. Finalmente **se avanza 5mm. comprobando los sensores para asegurar que se ha entrado al recinto**; si se detecta línea con algún sensor se pone "linea3" a 1 para que vuelva a empezar el bucle y avance de nuevo para encontrar el interior, si se detecta el exterior con algún sensor se pone "fuera" a 1 para que tras salir de la función se la vuelva a llamar en el programa principal e intente entrar otra vez.

Finalmente, si se ha entrado al recinto, el otro robot es avisado.

```
void Entrar()
{
	SEN_LINE_ANALOG();
	//Si está fuera gira 360º comprobando línea
	if(SEN_LINE_L<=senFueraMax || SEN_LINE_R<=senFueraMax)
	{
		fuera=1;
		RF_DATA_OUT[5]=fuera;
		ENVIO_DATOS();
		//Girar 360º o hasta entrar
		MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,100);
		while(!input(MOT_END))
		{
			SEN_LINE_ANALOG();
			if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax) &&
(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
			{
				//Se para al detectar el interior con ambos sensores
				MOT_STOP();
				linea3=0;
				fuera=0;
				RF_DATA_OUT[5]=fuera;
				ENVIO_DATOS();
				return;
```

```
                }
        }
        //Girar 360ºo hasta encontrar linea
        MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,100);
        while(!input(MOT_END))
        {
                SEN_LINE_ANALOG();
                if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                {
                        //Parar al detectar línea, se entra en el código para pasar la línea: linea3=1
                        MOT_STOP();
                        linea3=1;
                        fuera=0;
                }
        }
}
while(linea3==0 && fuera==1)
{
        if(j==2)
        {
                dist=dist+12;
                j=0;
        }
        //Avanzar
        MOT_STR(40,FWD,DISTANCE,dist);
        while(!input(MOT_END))
        {
                SEN_LINE_ANALOG();
                if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                {
                        //Parar al detectar línea, se entra en el código para pasar la línea: linea3=1
                        MOT_STOP();
                        linea3=1;
                }
        }
        if(linea3==0)
        {
                //Girar 360ºo hasta encontrar linea
                MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,100);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                        {
                                //Parar al detectar línea, se entra en el código para pasar la línea: linea3=1
                                MOT_STOP();
                                linea3=1;
                        }
                }
        }
        //Se retrocede el doble
        if(linea3==0)
        {
                //Si no se ha encontrado linea se retroce lo avanzado
                MOT_STR(40,BACK,DISTANCE,2*dist);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                        {
                                //Parar al detectar línea, se entra en el código para pasar la línea: linea3=1
                                MOT_STOP();
```

```
                        linea3=1;
                    }
                }
            }
            if(linea3==0)
            {
                //Girar 360ºo hasta encontrar linea
                MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,100);
                while(!input(MOT_END))
                {
                    SEN_LINE_ANALOG();
                    if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                    {
                        //Parar al detectar línea, se entra en el código para pasar la línea: linea3=1
                        MOT_STOP();
                        linea3=1;
                    }
                }
            }
            if(linea3==0)
            {
                //Avanzar
                MOT_STR(40,FWD,DISTANCE,dist);
                while(!input(MOT_END))
                {
                    SEN_LINE_ANALOG();
                    if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                    {
                        //Parar al detectar línea, se entra en el código para pasar la línea: linea3=1
                        MOT_STOP();
                        linea3=1;
                    }
                }
            }
            if(linea3==0)
            {
                //Girar 90ºo hasta encontrar linea
                MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,25);
                while(!input(MOT_END))
                {
                    SEN_LINE_ANALOG();
                    if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                    {
                        //Parar al detectar línea, se entra en el código para pasar la línea: linea3=1
                        MOT_STOP();
                        linea3=1;
                    }
                }
            }
            j=j+1;
        }
        fuera=0;
        j=0;
        dist=12;
        //Avanzar hasta detectar interior
        while(linea3==1)
        {
            //Avanzar unos 3.5cm o hasta detectar con algún sensor zona interior
            MOT_STR(40,FWD,DISTANCE,21);
            while(!input(MOT_END))
            {
                SEN_LINE_ANALOG();
```

```
                if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax) &&
(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                {
                        //Se ha entrado al recinto
                        linea3=0;
                        MOT_STOP();
                }
                //Si se detecta interior sólo con un sensor, se gira hasta detectar con los 2
                else if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax) ||
(SEN_LINE_R>=senAreaMin && SEN_LINE_R>=senAreaMax))
                {
                        if(SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax)
                                RL=LEFT;
                        else if(SEN_LINE_R>=senAreaMin && SEN_LINE_R>=senAreaMax)
                                RL=RIGHT;
                        //Girar 360º o hasta encontrar interior con ambos
                        MOT_ROT(20,FWD,CENTER,Rl,ANGLE,100);
                        while(!input(MOT_END))
                        {
                                SEN_LINE_ANALOG();
                                //Si se detecta interior linea3=0 y se detiene
                                if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax) &&
(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                                {
                                        linea3=0;
                                        MOT_STOP();
                                }
                        }
                        //Si tras girar no se detectainterior con ambos se avanza 5 mm.
                        if(linea3==1)
                        {
                                MOT_STR(40,FWD,DISTANCE,3);
                                while(!input(MOT_END))
                                {
                                        SEN_LINE_ANALOG();
                                        if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax)
&& (SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                                        {
                                                //Se ha entrado al recinto
                                                linea3=0;
                                                MOT_STOP();
                                        }
                                }
                        }
                }
        }
        //Si tras avanzar 3.5 cm no se ha detectado interiror
        if(linea3==1)
        {
                //Se gira 360º o hasta detectar interior con ambos sensores
                MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,100);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax) &&
(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                        {
                                //Se detiene al detectar el interior con ambos sensores
                                MOT_STOP();
                                linea3=0;
                                fuera=0;
                                RF_DATA_OUT[5]=fuera;
```

```
                        ENVIO_DATOS();
                        return;
                    }
                }
                //Girar 360º o hasta detectar interior con algún sensor
                MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,100);
                while(!input(MOT_END))
                {
                    SEN_LINE_ANALOG();
                    if((SEN_LINE_L>=senAreaMin && SEN_LINE_L<=senAreaMax) ||
(SEN_LINE_R>=senAreaMin && SEN_LINE_R<=senAreaMax))
                    {
                        //linea3=1 para volver a avanzar y entrar completamente al área
                        MOT_STOP();
                        linea3=1;
                        fuera=0;
                    }
                }
            }
            //Antes de salir del bucle se avanza para ver que realmente se ha entrado
            if(linea3==0)
            {
                //Avanzar 0.5cm comprobando línea
                MOT_STR(10,FWD,DISTANCE,3);
                while(!input(MOT_END))
                {
                    SEN_LINE_ANALOG();
                    if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                    {
                        //Aún no se ha entrado del todo en el recinto
                        linea3=1;
                        MOT_STOP();
                    }
                    //Si se ha salido del recinto, fuera=1 y se volverá a llamar a Entrar
                    else if(SEN_LINE_L<=senFueraMax && SEN_LINE_R<=senFueraMax)
                    {
                        linea3=0;
                        fuera=1;
                        MOT_STOP();
                    }
                }
            }
        }
        //Si se ha entrado se manda
        if(linea3==0 && fuera==0)
        {
            RF_DATA_OUT[5]=fuera;
            ENVIO_DATOS();
        }
}
```

Sigue_Lin

La función es la **función de seguimiento de línea** utilizada en otros programas, el robot avanza recto mientras detecta sólo con su sensor derecho (va siguiendo el borde izquierdo de la línea). En este caso se utiliza el valor analógico del sensor de línea ya que existen diferentes zonas.

Se le ha añadido un contador ("j") que se va aumentando cada vez que el robot no detecta línea con ningún sensor y se inicializa cada vez que se detecta la línea con algún sensor. En caso de que el contador llegue a 30, se llama a la función Entrar. Con esto se

soluciona el problema de que un robot, al buscar la línea empuje al otro que ya la está siguiendo y lo expulse; así si se da este caso, el robot puede volver a buscar la línea.

El programa "Limpiar_Area4_1.c" recorre la línea a alta velocidad (70% cuando avanza recto y 40% en los giros) mientras que "Limpiar_Area4_2.c" avanza a baja velocidad (20% en todos los movimientos). Con esta diferencia de velocidades se consigue que un robot alcance al otro rápidamente.

```
void Sigue_Lin()
{
	SEN_LINE_ANALOG();
	// Si detecta con el sensor derecho, sigue recto
	if(SEN_LINE_R>=senLinMin && SEN_LINE_L<senLinMin)
	{
		j=0;
		LED_R_ON();
		LED_L_OFF();
		MOT_STR(70,FWD,TIME,0);
	}
	//Si no ve línea, gira a la derecha
	else if(SEN_LINE_R<senLinMin && SEN_LINE_L<senLinMin)
	{
		j=j++;
		LED_R_OFF();
		LED_L_OFF();
		MOT_ROT(40,FWD,CENTER,RIGHT,TIME,0);
		if(j==30)
		{
			Entrar();
			linea=0;
			j=0;
		}
	}
	//Si está encima de la línea, gira a la izquierda
	else if(SEN_LINE_R>=senLinMin && SEN_LINE_L>=senLinMin)
	{
		j=0;
		LED_R_ON();
		LED_L_ON();
		MOT_ROT(40,FWD,CENTER,LEFT,TIME,0);
	}
	//Si detecta con el sensor izquierdo, gira a la izquierda
	else if(SEN_LINE_R<senLinMin && SEN_LINE_L>=senLinMin)
	{
		j=0;
		LED_R_OFF();
		LED_L_ON();
		MOT_ROT(40,FWD,CENTER,LEFT,TIME,0);
	}
}
```

Rutina de recepción de datos

Esta parte del programa se encarga de **actualizar las variables** que se envían los robots: **"contador", "parada", "fuera_otro", "objeto" y "línea_otro".** "**contador**" indica el **número de objetos restantes**, se actualiza en el caso de que en la posición 4 del vector de entradas haya un 1, así sólo se tiene en cuenta el valor del contador cuando éste es mandado, ya que en los demás casos toma el valor 0. "**parada**" señala que **se ha acabado completamente la tarea** si es 1. "**fuera_otro**" muestra que **el otro robot está fuera** si vale 1. "**objeto**" es 1 cuando **el otro robot está fuera y se detecta un objeto**,

por eso cuando el otro robot entra se pone a 0. "**linea_otro**" es 1 cuando **el otro robot ha detectado la línea** al terminar de expulsar los objetos.

```
//Interrupción del módulo RF
#int_ext
void int_externo()
{
	while(RF_RECEIVE()!=2)
	{
		if(RF_DATA_IN[4]==1)
			contador=RF_DATA_IN[7];
		parada=RF_DATA_IN[6];
		fuera_otro=RF_DATA_IN[5];
		if(fuera_otro==0)
			objeto=0;
		linea_otro=RF_DATA_IN[3];
	}
}
```

Programa principal

El programa principal se encarga de sacar todos los objetos del recinto y seguir la línea hasta que los robots se encuentran.

Sacar todos los objetos del recinto

En este bucle se contemplan los casos que se pueden dar mientras quedan objetos en la zona interior:

1) Si "fuera" es 1 quiere decir que **se sigue fuera tras llamar a la función Entrar → Se vuelve a llamar** a la función Entrar.
2) Si **el otro robot no ha salido y no se ha detectado el obstáculo o si se ha detectado obstáculo pero aún no se ha llegado a la línea → Si hay obstáculo se** le **ataca** y **si no**, **se comprueba línea evitándola y se sigue recto**.
3) Si **el otro robot está fuera y no se ha detectado objeto →** Se comprueba **si hay objeto delante**: si lo hay, **se retrocede 2cm, se para** y se pone "objeto" a 1 para permanecer quieto mientras el otro robot está fuera; **si no se encuentra obstáculo el robot sigue buscándolo** (comprueba línea, la evita y sigue recto).
4) Si **se ha detectado objeto y línea a la vez y el otro robot está dentro → Se llama a Sacar_Fuera para expulsar al objeto** unos centímetros del recinto.

```
while(contador>0)
{
	//Entrar si está fuera
	if(fuera==1)
		Entrar();
	//Si se detecta objeto pero no linea o no se detecta objeto y el otro robot no ha salido
	else if(sacar==0 && fuera_otro==0)
	{
		//Si se detecta objeto se ataca
		if (MOWAY_SEARCH())
			MOWAY_ATTACK();
		//Si no, se esquiva línea y se recorre el área
		else
		{
			MOWAY_LINE();
			MOT_STR(60,FWD,TIME,0);
		}
	}
```

```
	//Si el otro robot está fuera
	else if(fuera_otro==1 && objeto==0)
	{
		//Si se encuentra objeto, se retrocede 2cm y se para
		if (MOWAY_SEARCH())
		{
			MOT_STR(60,BACK,DISTANCE,12);
			while(!input(MOT_END)){}
			MOT_STOP();
			objeto=1;
		}
		else
		{
			//Se comprueba la línea y se evita
			MOWAY_LINE();
			MOT_STR(60,FWD,TIME,0);
		}
	}
	//Si se ha detectado objeto y se ha llegado el borde del recinto
	//se saca al objeto del recinto unos centímetros si el otro está dentro
	else if(sacar==1 && fuera_otro==0)
		SACAR_FUERA();
}
```

Seguir línea hasta encontrarse

Cuando el contador llega a 0, mientras "parada" sea 0 **se busca la línea y se la sigue hasta que ambos robots se encuentren**.

Primero se entra al recinto si el robot está fuera, gira 360º comprobando la línea y si no la encuentra avanza recto hasta encontrarla. En ambos movimientos, si se detecta línea con algún sensor el robot se detiene, se pone "línea" a 1 y se envía.

Tras encontrar la línea, el robot la sigue mientras el otro no haya detectado la línea también. Con esto se soluciona el problema de que un robot que sigue la línea detecte a otro que aún no la ha detectado.

Cuando ambos robots siguen la línea, se va comprobando el sensor de obstáculo. Si se detecta obstáculo, el robot se detiene pone "parada" a 1 y se lo envía al otro robot. Mientras no detecta obstáculo, Moway sigue la línea.

Finalmente si "parada" es 1 el robot se detiene.

```
if(contador==0)
{
	if(parada==0)
	{
		//Se envía el valor del contador
		RF_DATA_OUT[7]=contador;
		ENVIO_DATOS();
	}
	while(parada==0)
	{
		//Si no se detecta línea, se busca
		if(linea==0)
		{
			//Entrar si está fuera
			Entrar();
			//Girar 360º comprobando línea
			MOT_ROT(30,FWD,CENTER,LEFT,DISTANCE, 100);
			while(!input(MOT_END))
			{
				SEN_LINE_ANALOG();
```

```
                        if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                        {
                                MOT_STOP();
                                linea=1;
                                RF_DATA_OUT[3]=linea;
                                ENVIO_DATOS();
                        }
                }
                //Avanzar recto hasta encontrar línea
                MOT_STR(40,FWD,TIME,0);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        if(SEN_LINE_L>=senLinMin || SEN_LINE_R>=senLinMin)
                        {
                                MOT_STOP();
                                linea=1;
                                RF_DATA_OUT[3]=linea;
                                ENVIO_DATOS();
                        }
                }
        }
        //Cuando se encontra linea, se la sigue hasta que ambos encuentren línea
        while(linea==1 && linea_otro==0)
                Sigue_Lin();
        //Cuando ambos detectan línea la siguen hasta que uno detecta obstáculo
        while(parada==0 && linea==1 && linea_otro==1)
        {
                SEN_OBS_ANALOG();
                //Se detiene al detectar obstáculo y se le comunica al otro robot que pare
                if (SEN_OBS_L>=senObsMin || SEN_OBS_R>=senObsMin)
                {
                        MOT_STOP();
                        if(parada==0)
                        {
                                parada=1;
                                RF_DATA_OUT[6]=parada;
                                ENVIO_DATOS();
                        }
                }
                //Se sigue la línea si no se ha encontrado obstaculo
                else
                        Sigue_Lin();
        }
    }
    //Si el otro robot ha parado, se para también
    if(parada==1)
        MOT_STOP();
}
```

Experimentos

Se realizan experimentos para expulsar 6 objetos y también para expulsar 10 objetos. Se comprueba que los programas funcionan correctamente.

Para comprobar si esta tarea es eficiente, se realiza también con un solo Moway. Se modifica el programa y se crea el programa **“1_Limpia_Area.c”**. El programa realiza las mismas tareas que los anteriores, pero en este caso sólo hay un robot y por lo tanto

se eliminan los envíos y recepciones. Tras eliminar a todos los objetos el robot sigue la línea y se detiene a los 10 segundos.

Para contar 10 segundos se utiliza un temporizador. Se estudió la posibilidad del uso del TIMER1, pero esta opción se descartó ya que éste necesita los pines RC0 y RC1 libres, y éstos son empleados por los motores como SCL y SDA respectivamente. Por lo tanto se ha empleado el Timer2.

TIMER2

Es un temporizador ascendente de 8 bits con un preescaler y un postescaler. La señal de reloj interna (FOSC/4), antes de pasar por el timer, pasa por un preescaler que puede tomar los valores 1, 4 y 16, según el valor de los bits 0 y 1 del byte T2CON. El postescaler pone a 1 la bandera (bit 1 de PIR1) de interrupción del timer cuando se hayan igualado PR2 y TMR2 un número de veces del postescaler, que se configura con los bits del 3 al 6 de T2CON.

Con el temporizador se puede medir como máximo 0.065536s:

$$256*16*16=65536\mu s.=0.065536s$$

Como interesa contar 10 segundos, se puede temporizar 200 veces 0.05s. Para obtener este valor, el periodo que se debe utilizar es 195:

$$0.05s=50000=16*16*periodo \rightarrow periodo=195$$

Para poder utilizar el temporizador en el programa se ha de activar su interrupción y programar la tarea que se quiere realizar en la rutina de interrupción:

```
//**********************[Rutina de interrupción del TIMER2]*****************
//Interrupción del timer
#int_timer2
void interrupcion_timer2()
{
	cont=1;
}
//Programa principal
void main()
{
//**************[Configuración  de sensores y motores]************************
SEN_CONFIG();
MOT_CONFIG();
//Habilitar la interrupción del TIMER
enable_interrupts(INT_TIMER2);
```

Cuando se encuentra la línea se activa el temporizador con la función setup_timer_2 a la que se le pasan el preescaler, el periodo y el postescaler, por este orden. Tras los 10 segundos el robot se detiene.

```
//Se activa el TIMER2 para 50ms
setup_timer_2 (T2_DIV_BY_16, 195, 16);
//Cuando se encuentra línea, se sigue hasta que pasan 10seg
while(linea==1)
{
	Sigue_Lin();
	if(cont==1)
	{
		//Temporizar 200 veces 50ms, 10 seg
		if(k<200)
		{
```

```
                    //50ms
                    setup_timer_2 (T2_DIV_BY_16, 195, 16);
                    k=k+1;
                    cont=0;
            }
            else
            {
                    //Tras 10seg se para
                    MOT_STOP();
                    linea=0;
            }
    }
}
```

Se realizan 10 experimentos con 2 robots y éstos se repiten para uno solo, con los objetos en la misma disposición. En 5 de los experimento se utilizan 6 objetos y en los otros 5, 10 objetos. En la siguiente tabla se recoge el tiempo que ha durado cada experimento.

Los videos de los experimentos están incluidos dentro del CD que se adjunta con esta memoria.

6 Objetos				
Experimento 1	**Experimento 2**	**Experimento 3**	**Experimento 4**	**Experimento 5**
1 Moway				
4min 37s	7min 53s	3min 5s	**9min 44s**	6min 45s
2 Moways				
2min 2s	2min 54s	3min 24s	**1min 39s**	1min 59s
10 Objetos				
Experimento 1	**Experimento 2**	**Experimento 3**	**Experimento 4**	**Experimento 5**
1 Moway				
22min 10s	5min 27s	8min10s	3min 8s	6min 25s
2 Moways				
4min 32s	2min 37s	2min 34s	3min 34	4min 1s

Tabla 15: Experimentos cronometrados para la limpieza de un área

Como se puede observar, en casi todos los casos 2 robots tardan menos de la mitad de tiempo que uno solo. Incluso en el experimento 4 para 6 objetos 2 Moways tardan casi 6 veces menos que 1 sólo. Por lo que queda demostrada la eficiencia de emplear 2 robots para realizar esta tarea.

4.4.- COOPERACIÓN PARA GUIADO DE OBJETOS POR UN CAMINO

Esta tarea consiste en desplazar un objeto por un camino con la ayuda de 3 microbots. Un robot se encarga de empujar el objeto hacia adelante y los otros 2 los van guiando por los lados y lo van centrando, como se muestra en la siguiente figura.

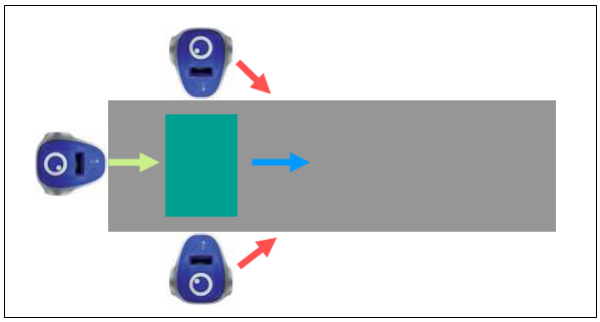

Figura 49: Esquema del guiado de un objeto por un camino

Como camino se utiliza una franja gris de luminancia 150 cuyo ancho es el suficiente para contener el objeto a transportar y permite que los robots laterales detecten la zona gris. El ancho es algo mayor que el ancho de la caja más 2 veces la longitud entre la parte delantera del robot y el final del sus sensores, así se permite que haya holgura y la caja no vaya aprisionada entre los robots laterales y pueda avanzar mejor.

Uno de los robots debe ir empujando la caja dentro del área y ser capaz de encontrar la caja si la pierde de delante. Los otros 2 robots deben mantener la caja centrada, avanzando lateralmente. Como primera solución para esta tarea se decide que los robots realicen los siguientes movimientos:

- El robot que debe desplazar al objeto se pone perpendicular a él y avanza 3cm.
- Los robots laterales se ponen perpendiculares al camino y avanzan 5mm.

A partir de estas ideas se crean los programas “Empujar.c”, “LateralDer.c” y “LateralIzq.c”. El primero es el que va desplazando la caja hacia adelante y los otros dos son los que se encargan de centrar la caja, estos 2 programas son muy similares.

“Empujar.c”, “LateralDer.c” y “LateralIzq.c”

En estos programas se produce un avance por turnos. Cuando “Empujar.c” detecta la caja, avanza 3.5cm, se detiene y manda el turno. Después los otros 2 retroceden, giran, avanzan, se ponen perpendiculares a la línea y avanzan 5mm. Todos los robots se detienen cuando el que va empujando la caja encuentra una línea negra.

Tras una serie de experimentos se observa que los robots Moway 2 y Moway 4 son los mejores para realizar la tarea de los laterales, ya que sus sensores de línea son más parecidos. Por lo tanto, Moway 2 y Moway 4 trabajan en los laterales y Moway 3 es el que va empujando.

A continuación se comenta el código y los resultados obtenidos:

“Empujar.c”

Este programa se encarga de **llegar hasta el objeto, ponerse perpendicular a él y empujarlo**. Luego **envía el turno** a los otros robots **y espera a recibir el suyo**. Mientras avanza **comprueba línea y si la detecta, se detiene y envía a los otros que se detengan.**

A continuación se va a comentar la parte del código del avance:

Si el robot está muy cerca del objeto (alguno de los sensores de obstáculo detecta un valor superior o igual a 150), retrocede 6cm. y avanza hasta detectar el máximo con alguno de los sensores.

Luego está el bucle para ponerse perpendicular al objeto (a él se entra si el valor de los sensores es inferior a 150 o si se ha detectado ya el máximo con algún sensor). En esta parte se consideran 3 casos:

1) Si el objeto se encuentra más cerca de uno de los sensores → Se gira hacia él 3.6º, comprobando si se detecta el máximo con ambos sensores de obstáculo. Si tras el giro no se ha detectado el máximo con algún sensor, se retrocede 4cm. y se avanza recto hasta detectar el máximo con algún sensor y se comprueba si se detecta el máximo con los 2. Cuando se detecta el máximo con ambos sensores se pone "max" a 1 para salir del bucle.
2) Si se detecta el máximo con ambos sensores → Se pone "max" a 1.
3) Si se detecta con ambos sensores el mismo valor pero no es el máximo → Se avanza 2cm. comprobando los sensores de obstáculo.

Después se avanza 3.5cm. comprobando el sensor de línea. Si se encuentra la línea, se detiene y se comunica. Y finalmente el robot envía el turno.

```
//**********************************[Avance]*****************************//
//Este robot avanza primero
while(turno1==0 && linea==0)
{
	turno2=0;
	turno3=0;
	SEN_OBS_ANALOG();
	//Si está cerca del objeto
	if(SEN_OBS_R>=150 || SEN_OBS_L>=150)
	{
		//Retrocer 6cm
		MOT_STR(40,BACK,DISTANCE,35);
		while(!input(MOT_END)){}
		//Avanzar hasta detectar el máximo con algún sensor o detectar línea
		MOT_STR(40,FWD,TIME,0);
		while(!input(MOT_END))
		{
			SEN_OBS_ANALOG();
			if(SEN_OBS_R==255 || SEN_OBS_L==255)
				MOT_STOP();
		}
	}
	//Si el obstáculo no está cerca o ya se ha detectado el máximo con algún sensor
	while(max==0)
	{
		SEN_OBS_ANALOG();
		//Si el objeto está a la izquierda
		if(SEN_OBS_L>SEN_OBS_R)
		{
			//Se gira 3.6ºa la izq comprobando si se detectamos máximo con los 2
			MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,1);
			while(!input(MOT_END))
			{
				SEN_OBS_ANALOG();
				if(SEN_OBS_R==255 && SEN_OBS_L==255)
				{
					MOT_STOP();
					max=1;
```

```
                }
            }
            if(max==0)
            {
                //Retroceder 4cm
                MOT_STR(40,BACK,DISTANCE,24);
                while(!input(MOT_END)){}
                //Avanzar hasta detectar el máximo con algún sensor
                MOT_STR(40,FWD,TIME,0);
                while(!input(MOT_END))
                {
                    SEN_OBS_ANALOG();
                    if(SEN_OBS_R==255 || SEN_OBS_L==255)
                        MOT_STOP();
                    if(SEN_OBS_R==255 && SEN_OBS_L==255)
                        max=1;
                }
            }
        }
        //Si el obstáculo está a la derecha
        else if(SEN_OBS_R>SEN_OBS_L)
        {
            //Girar 3.6º a la der comprobando si se detecta máximo con los 2
            MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,1);
            while(!input(MOT_END))
            {
                SEN_OBS_ANALOG();
                if(SEN_OBS_R==255 && SEN_OBS_L==255)
                {
                    MOT_STOP();
                    max=1;
                }
            }
            if(max==0)
            {
                //Retroceder 4cm
                MOT_STR(40,BACK,DISTANCE,24);
                while(!input(MOT_END)){}
                //Avanzar hasta detectar el máximo con algún sensor
                MOT_STR(40,FWD,TIME,0);
                while(!input(MOT_END))
                {
                    SEN_OBS_ANALOG();
                    if(SEN_OBS_R==255 || SEN_OBS_L==255)
                        MOT_STOP();
                    if(SEN_OBS_R==255 && SEN_OBS_L==255)
                        max=1;
                }
            }
        }
        //Si se detecta el máximo con ambos
        else if(SEN_OBS_R==255 && SEN_OBS_L==255)
        {
            max=1;
        }
        //Si se detecta lo mismo con ambos pero no el máximo
        //Se avanza recto 2cm o hasta detectar el máximo con alguno
        else
        {
            MOT_STR(40,FWD,DISTANCE,12);
            while(!input(MOT_END))
            {
```

```
                        SEN_OBS_ANALOG();
                        if(SEN_OBS_R==255 || SEN_OBS_L==255)
                            MOT_STOP();
                        if(SEN_OBS_R==255 && SEN_OBS_L==255)
                            max=1;
                    }
                }
            }
            max=0;
            //Empujar al objeto 3.5cm comprobando línea
            MOT_STR(40,FWD,DISTANCE,21);
            while(!input(MOT_END))
            {
                SEN_LINE_ANALOG();
                //Si se detecta la línea se detiene y lo envía
                if(SEN_LINE_R>=175 || SEN_LINE_L>=175)
                {
                    MOT_STOP();
                    linea=1;
                    RF_DATA_OUT[4]=linea;
                    ENVIO_DATOS();
                }
            }
            //Se envía el turno
            turno1=1;
            RF_DATA_OUT[0]=1;
            RF_DATA_OUT[1]=turno1;
            ENVIO_DATOS();
            RF_DATA_OUT[0]=0;
}
```

“LateralDer.c”

Este programa se encarga de **girar en el sentido de avance del objeto** y luego **ponerse perpendicular al camino y centrar el objeto**.

Para girar en la dirección de avance, el robot realiza los siguientes movimientos:

- Retrocede 6cm.
- Gira 10.8º a la derecha.
- Avanza hasta detectar la línea con algún sensor.
- Gira 10.8º a la izquierda para enderezarse.
- Retrocede 6cm.
- Avanza hasta detectar la línea con algún sensor.

Tras estos movimientos el robot se coloca perpendicular a la línea si no lo está ya. Va girando 3.6º hacia el lado que detecta la línea, retrocede 6cm. y avanza hasta detectar línea con ambos sensores. Estos movimientos se repiten hasta que se detecta el camino con los 2 sensores a la vez (el robot se encuentra perpendicular al camino).

Finalmente, el robot avanza 1.5cm. para colocar la caja en el centro del camino y envía el turno.

El programa “LateralIzq.c” es igual que éste, sólo cambian las direcciones y los sentidos de giro.

Se ha seleccionado un ángulo de giro de 10.8º porque con este ángulo se consigue que los robots avancen a la par. Este ángulo podría variarse si se modificara la distancia recorrida por el robot que va desplazando la caja.

Experimentos

Se observa que los robots laterales no consiguen que la caja quede muy centrada en el camino, la caja avanza por el camino, pero no va recta.

También se producen fallos con el robot que va empujando ya que no consigue ponerse siempre perpendicular a la caja, esto se debe a que los sensores de los robots no son idénticos.

Se decide colocar un parachoques a los robots laterales, para ello se crea un acople de cartón como se observa en la siguiente figura.

En este caso se disminuye el avance de los robots para centrar la caja a 1 cm. Con esto se consigue que los robots dejen la caja recta dentro del camino, pero en algunos casos, el robot que va empujando, en lugar de avanzar recto, se va torciendo, por los errores en la odometría, y a veces se choca con los robots laterales.

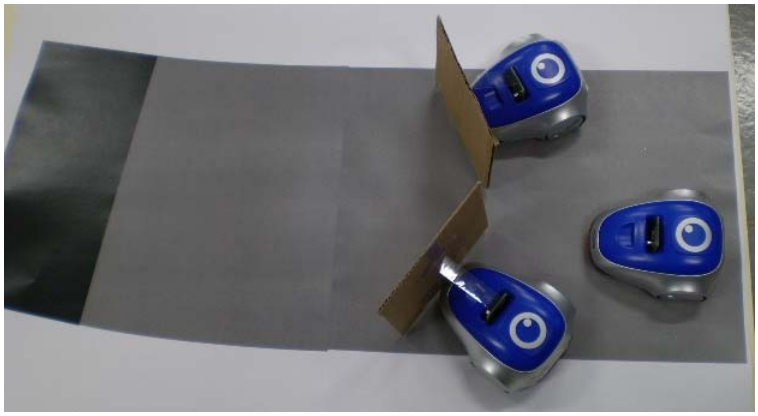

Figura 50: Guiado de un objeto por un camino, los robots laterales aparecen con lo acoples empleados

Para solucionar este último problema se decide modificar el camino de forma que sean 2 franjas laterales y una línea central y el Moway que va empujando avance en todo momento sobre la línea. Por lo tanto se modifican los programas anteriores y se obtienen los siguientes: “Empujar_2.c”, “LateralDer_2.c” y “LateralIzq_2.c”.

Guiado de un objeto por un camino formado por 2 franjas laterales y una línea central

Como en este caso el robot que desplaza al objeto se encuentra centrado en el camino, ya que va siguiendo la línea, no es necesario que se ponga perpendicular a la caja. Se comprueba que los robots laterales dejan la caja recta cuando la empujan, por lo que se opta por no usar los acople y así poder emplear su sensor de obstáculo.

El robot central se encarga de avanzar siguiendo la línea hasta detectar el objeto. Cuando lo detecta, vuelve a seguir la línea durante un tiempo y a una velocidad que se comentarán más adelante. Luego se detiene y pasa el turno a los otros robots. Éstos se encargan de avanzar lateralmente un ángulo y a una velocidad que también se

comentarán más adelante. Después se ponen perpendiculares a la caja, avanzan 1cm. y envían el turno.

Antes de moverse y después de avanzar, los robots laterales comprueban si se han adelantado o se han retraso y entonces retroceden o avanzan, según corresponda. Ya que los robots laterales no avanzan igual y a veces se atrasan o se adelantan.

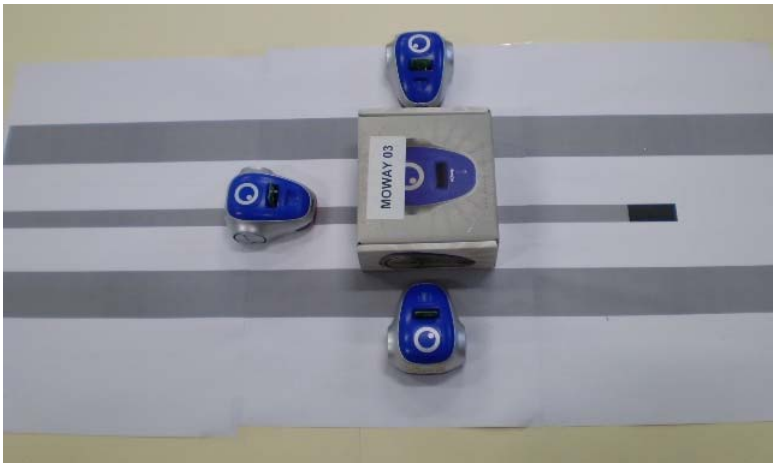

Figura 51: Guiado de un objeto por un camino utilizando líneas

A continuación se explica cada uno de los programas:

“Empujar_2.c”

Este programa contiene la función de seguimiento de línea, la función de búsqueda de obstáculo, la rutina de recepción de datos, la rutina de interrupción del TIMER2 y el programa principal. Éste último se divide a su vez en las siguientes partes: avance, espera y parada al detectar línea. A continuación se comenta cado una de los apartados.

“Sigue_Lin”

La función se encarga de **seguir la línea mientras no se detecte la zona negra**. El robots **va siguiendo la línea si los sensores toman valores entre 50 y 155** (rango de valores ampliado para una línea de luminancia 150). En caso de que algún sensor detecte valores superiores a 160, el robot se detiene y “línea” se pone a 1.

Los movimientos se realizan a la velocidad “vel” definida al comienzo del programa, así puede ser modificada con facilidad en los diferentes experimentos realizados.

```
void Sigue_Lin()
{
	SEN_LINE_ANALOG();
	if(SEN_LINE_R>senLinMax || SEN_LINE_L>senLinMax)
	{
		MOT_STOP();
		linea=1;
	}
	//Si está detecta con el sensor derecho, sigue recto
	else	if((SEN_LINE_R>=senLinMin	&&	SEN_LINE_R<=senLinMax)	&&
SEN_LINE_L<=senLinMin)
	{
		LED_R_ON();
```

```
            LED_L_OFF();
      MOT_STR(vel,FWD,TIME,0);
      }
      //Si no ve línea, gira a la derecha
      else if(SEN_LINE_R<senLinMin && SEN_LINE_L<senLinMin)
      {
            LED_R_OFF();
            LED_L_OFF();
            MOT_ROT(vel,FWD,CENTER,RIGHT,TIME,0);
      }
      //Si está encima de la línea, gira a la izquierda
      else if((SEN_LINE_R>=senLinMin && SEN_LINE_R<=senLinMax) &&
(SEN_LINE_L>=senLinMin && SEN_LINE_L<=senLinMax))
      {
            LED_R_ON();
            LED_L_ON();
            MOT_ROT(vel,FWD,CENTER,LEFT,TIME,0);
      }
      //Si detecta con el sensor izquierdo, gira a la izquierda
      else if(SEN_LINE_R<senLinMin && (SEN_LINE_L>=senLinMin &&
SEN_LINE_L<=senLinMax))
      {
            LED_R_OFF();
            LED_L_ON();
            MOT_ROT(vel,FWD,CENTER,LEFT,TIME,0);
      }
}
```

MOWAY_SEARCH

Esta función se encarga de **enviar 1 en caso de que el robot se encuentre pegado al objeto**. Se comprueban los sensores de obstáculo y si toman el valor máximo (255), se devuelve 1, en caso contrario la función devuelve 0.

Rutina de recepción de datos

Cada vez que se reciben datos, **se comprueba** desde qué **dirección** son enviados **y se actualiza el turno correspondiente**. **Cuando los dos turnos son 1, el turno de este robot pasa a 0** (“turno1”). Así este robot sólo avanza cuando los otros 2 han colocado la caja en su sitio.

```
#int_ext
void int_externo()
{
      while(RF_RECEIVE()!=2)
      {
            //Si recibe de la dirección 2 el turno se actualiza
            if(RF_DIR_IN==0x02)
            {
                  if(RF_DATA_IN[1]==1)
                        turno2=1;
            }
            //Si se recibe el turno de la dirección 3 se actualiza
            if(RF_DIR_IN==0x03)
            {
                  if(RF_DATA_IN[1]==1)
                        turno3=1;
            }
            //Se modifica el propio turno si se han recibo los turnos delos otros 2 Moways
            if(turno2==1 && turno3==1)
```

```
                turno1=0;
        }
}
```

Rutina de interrupción del TIMER2

Esta rutina se encarga de **cambiar a 1 el valor de "cont"**.

Main

Avance siguiendo la línea

Primero **se avanza hasta que se detecta el obstáculo** muy muy cerca.

Luego **se activa el temporizador** para j*50ms **y se va siguiendo la línea**. **Cuando transcurre el tiempo**, el robot se detiene y comprueba el sensor de línea; **si se detecta la zona negra**, se pone **"línea"** a **1**; y **si no** se ha llegado al final **se envía el turno**.

```
//Este robot avanza primero y llega hasta el objeto
while(obs==0)
{
        if(!MOWAY_SEARCH())
                Sigue_Lin();
        else
                obs=1;
}
//Se activa el TIMER2 para 50ms
setup_timer_2 (T2_DIV_BY_16, 195, 16);
while(turno1==0 && linea==0)
{
        turno2=0;
        turno3=0;
        //Cuando se encontra línea se sigue hasta que pasa el tiempo
        Sigue_Lin();
        if(cont==1)
        {
                //Se temporizan j veces 50ms
                if(k<j)
                {
                        //50ms
                        setup_timer_2 (T2_DIV_BY_16, 195, 16);
                        k=k+1;
                        cont=0;
                }
                else
                {
                        //Tras el tiempo se para y se comproba si se ha llegado al final
                        MOT_STOP();
                        SEN_LINE_ANALOG();
                        //Se detiene al detectar la línea negra
                        if(SEN_LINE_R>senLinMax || SEN_LINE_L>senLinMax)
                        {
                                MOT_STOP();
                                linea=1;
                        }
                        else
                        {
                                //Se envía el turno si no se ha llegado al final
                                turno1=1;
                                RF_DATA_OUT[1]=turno1;
                                ENVIO_DATOS();
```

```
                        cont=0;
                        k=0;
                    }
                }
            }
}
```

Espera y parada al detecta línea

Mientras sea el turno de los otros robots, éste se queda parado. Y si se ha detectado línea ocurre lo mismo y, a demás, se les ordena a los otros robots que se detengan.

```
//*****************************[Espera]*********************************//
while(turno1==1 && linea==0)
{
    MOT_STOP();
}
//*********************************[Parada al detectar línea]**************//
if(linea==1)
{
    MOT_STOP();
    //Se envía que se ha detectado línea
    RF_DATA_OUT[4]=linea;
    ENVIO_DATOS();
}
```

"LateralDer_2.c"

Este programa contiene cuatro funciones (para buscar un obstáculo, para avanzar lateralmente en un sentido, para avanzar lateralmente en sentido contrario y para ponerse perpendicular al camino), la rutina de recepción de datos y el Main. Este último se divide en la espera, el avance y la parada. A continuación se va a explicar cada apartado.

MOWAY_SEARCH

Es la misma función que en el programa anterior, pero en este caso el valor mínimo de obstáculo es 50. Se emplea para saber si un robot se ha quedado detrás del objeto o se ha adelantado.

AvanceLatDer

Esta función se encarga de **avanzar en el caso de que el robot se encuentre en el lado derecho** del robot que sigue la línea, **y** se encarga de **retroceder si** el robot **se encuentra en el otro lado**.

Los movimientos que se realizan para avanzar lateralmente son los siguientes:

- Retroceder 7cm.
- Girar a la derecha un ángulo "ang".
- Avanzar hasta detectar la franja con algún sensor.
- Girar en sentido contrario para enderezar el robot.
- Retroceder 7cm.
- Avanzar hasta detectar la franja con algún sensor.

Todos los movimientos se realizan a la misma velocidad: "vel".

```
void AvanceLatDer()
{
	//Retrocer
	MOT_STR(vel,BACK,DISTANCE,dist);
	while(!input(MOT_END)){}
	//Girar a la derecha
	MOT_ROT(vel,FWD,CENTER,RIGHT,ANGLE,ang);
	while(!input(MOT_END)){}
	//Avanzar hasta que se detecta línea con algun sensor
	MOT_STR(vel,FWD,TIME,0);
	while(!input(MOT_END))
	{
		SEN_LINE_ANALOG();
		//Si detecta línea se detiene
		if(SEN_LINE_R>=Senlin || SEN_LINE_L>=Senlin)
			MOT_STOP();
	}
	//Girar en sentido contrario para ponerse recto
	MOT_ROT(vel,FWD,CENTER,LEFT,ANGLE,ang);
	while(!input(MOT_END)){}
	//Retrocer
	MOT_STR(vel,BACK,DISTANCE,dist);
	while(!input(MOT_END)){}
	//Avanzar hasta que se detecte línea con algun sensor
	MOT_STR(vel,FWD,TIME,0);
	while(!input(MOT_END))
	{
		//Se comprueban los sensores
		SEN_LINE_ANALOG();
		//Si detecta linea se detiene
		if(SEN_LINE_R>=Senlin || SEN_LINE_L>=Senlin)
			MOT_STOP();
	}
}
```

La función **“AvanceLatIz”** es igual a la anterior cambiando el sentido de los giros.

“PerpLinea”

Mediante este algoritmo **Moway se coloca de forma perpendicular al camino**. Para ello el robot va girando 3.6º hacia el lado que detecta la línea, retrocede “dist” y avanza hasta que detecta la franja con algún sensor. Estos movimientos se repiten hasta que se detecta el interior con ambos sensores.

```
void PerpLinea()
{
	while(max==0)
	{
		//Comprobar sensor de línea para ver si ya se está perpendicular a la línea
		SEN_LINE_ANALOG();
		//Si se detecta con el sensor derecho
		if(SEN_LINE_R>=Senlin && SEN_LINE_L<Senlin)
		{
			//Girar a la derecha 3.6º
			MOT_ROT(vel,FWD,CENTER,RIGHT,ANGLE,1);
			while(!input(MOT_END)){}
			//Retrocer
			MOT_STR(vel,BACK,DISTANCE,dist);
			while(!input(MOT_END)){}
			//Avanzar hasta detectar línea con algún sensor
```

```
                MOT_STR(vel,FWD,TIME,0);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        //Si detecta línea se deteniene
                        if(SEN_LINE_R>=Senlin || SEN_LINE_L>=Senlin)
                                MOT_STOP();
                        //Si detecta con ambos el robot ya está perpendicular a la línea
                        if(SEN_LINE_R>=Senlin && SEN_LINE_L>=Senlin)
                                max=1;
                }
        }
        //Si se detecta el objeto con el sensor izquierdo
        else if(SEN_LINE_R<Senlin && SEN_LINE_L>=Senlin)
        {
                //Girar a la izquierda 3.6º
                MOT_ROT(vel,FWD,CENTER,LEFT,ANGLE,1);
                while(!input(MOT_END)){}
                //Retroceder
                MOT_STR(vel,BACK,DISTANCE,dist);
                while(!input(MOT_END)){}
                //Avanzar hasta detectar línea con algún sensor
                MOT_STR(vel,FWD,TIME,0);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        //Si detecta línea se deteniene
                        if(SEN_LINE_R>=Senlin || SEN_LINE_L>=Senlin)
                                MOT_STOP();
                        //Si detecta con ambos el robot ya está perpendicular a la línea
                        if(SEN_LINE_R>=Senlin && SEN_LINE_L>=Senlin)
                                max=1;
                }
        }
        //Si detecta con ambos se detiene, ya está perpendicular a la línea
        else if(SEN_LINE_R>=Senlin && SEN_LINE_L>=Senlin)
        {
                MOT_STOP();
                max=1;
        }
    }   max=0;
}
```

Rutina de recepción de datos

Si se recibe el turno del robot que empuja la caja, el **turno propio toma el valor contrario del recibido**. También **se actualiza** el valor de la variable **"línea".**

```
#int_ext
void int_externo()
{
    while(RF_RECEIVE()!=2)
    {
        if(RF_DIR_IN==0x01)
        {
            turno1=RF_DATA_IN[1];
            turno2=!turno1;
        }
        linea=RF_DATA_IN[4];
    }
}
```

Main

Espera

El robot **permanece parado** mientras el objeto avanza. Tras recibir el turno **se comprueba si el objeto ha desaparecido de delante**, lo que quiere decir que el robot se ha quedado retrasado, y **se pone** la variable **"detras" a 1**.

```
//Primero se espera a que el otro avance
while(turno2==1 && linea==0)
{
	MOT_STOP();
}
//Se comprueba si se sigue detectando objeto después de que éste haya avanzado
if(turno2==0 && linea==0)
{
	if(!MOWAY_SEARCH())
		detras=1;
}
```

Avance

Esta parte del código se encarga de que el robot avance lateralmente teniendo en cuenta si se ha retrasado o se adelanta tras el movimiento.

Primero se comprueba **si el robot está detrás del objeto** y **se avanza 2 veces** mientras no se detecte el objeto. Para avanzar se utiliza la función "AvanceLatDer", tras cada avance el robot se coloca perpendicular a la zona gris mediante la función "PerpLinea".

Luego el robot **avanza y se pone perpendicular a la franja**. **En caso de que** tras moverse **no detecte el objeto** (el robot se ha adelantado), se **retrocede 2 veces** mientras no se detecte el objeto.

Por último el **Moway se pone perpendicular al camino, avanza un 1cm. y envía el turno.**

```
while(turno2==0 && linea==0)
{
	//Si se ha quedado atrás, llama 2 veces a la funcón de avance hasta que se detecta con 2 sensores
	while(detras==1)
	{
		AvanceLatDer();
		//Moway se pone perpendicular  a la línea
		PerpLinea();
		AvanceLatDer();
		//Moway se pone perpendicular  a la línea
		PerpLinea();
		if(MOWAY_SEARCH())
			detras=0;
	}
	//Avanzar
	AvanceLatDer();
	//Moway se pone perpendicular  a la línea
	PerpLinea();
	//Se comprueba si se ha adelantado al objeto tras el avance
	if(!MOWAY_SEARCH())
		delante=1;
	//Si se ha adelantado, mientras no detecte la caja retroce 2 veces
	while(delante==1)
	{
```

```
            AvanceLatIz();
            AvanceLatIz();
            if(MOWAY_SEARCH())
                  delante=0;
      }
      //Moway se pone perpendicular  a la línea
      PerpLinea();
      //Avanzar 1cm para centrar la caja
      MOT_STR(vel,FWD,DISTANCE,6);
      while(!input(MOT_END)){}
      //Envía que se ha movido
      turno2=1;
      RF_DATA_OUT[1]=turno2;
      ENVIO_DATOS();
}
```

Parada

Si el otro robot ha detectado la zona negra **todos se detienen**.

```
if(linea==1)
{
      MOT_STOP();
}
```

El código del programa "LateralIzq_2.c" es similar a este, a excepción de las direcciones y los sentidos de avance y retroceso.

Experimentos

Las pruebas muestran que la tarea se realiza mejor si se coloca a Moway 2 a la izquierda y a Moway 4 a la derecha, así se compensan sus errores y se adelantan y se atrasan menos. Los experimentos se han hecho en estas condiciones, con un camino recto y con una línea de unos 2cm. de ancho para que el robot que la sigue no gire 180º.

Se han realizado una serie de experimentos que se recogen en la tabla siguiente. En estas pruebas se han ido modificando las velocidades, los ángulos de giro y el tiempo de avance del robot que va empujando. En negrita aparecen las pruebas cuyo funcionamiento es idóneo.

Experimento 1			
Empujar	**40%**	**0.4s**	Funciona bien. Los Robots van más o menos al compás del objeto.
Lateral	**60%**	**21.6º**	
Experimento 2			
Empujar	40%	0.4s	Moway 2 al intentar avanzar y ponerse recto, se queda detrás de la caja y al no detectarla, retrocede indefinidamente. Se toman ángulos menores.
Lateral	60%	25.2º	
Experimento 3			
Empujar	40%	0.4s	Funciona bien, pero los robots se quedan algo retrasados, esto se va solucionando con los avances al no detectar la caja.
Lateral	60%	18º	
Experimento 4			
Empujar	40%	0.35s	Funciona bien, pero los robots se quedan algo retrasados, esto se va solucionando con los avances al no detectar la

Lateral	60%	18º	caja. Quizá haya que poner más avances porque Moway 4 se va quedando atrás.
Experimento 5			
Empujar	40%	0.3s	Funciona bien. Moway 2 se adelanta a veces, pero con la llamada a la función para retroceder vuelve a la caja. Quizá haya que añadir otra llamada más para que el robot se quede más centrado.
Lateral	60%	18º	
Experimento 6			
Empujar	**40%**	**0.4s**	Funciona bien. A veces los robots se adelantan, pero al retroceder 2 veces se quedan centrados en la caja.
Lateral	**80%**	**21.6º**	
Experimento 7			
Empujar	40%	0.35s	Funciona bien, pero a veces cuando Moway 4 se queda retrasado y avanza 2 veces, en lugar de avanzar retrocede al ponerse perpendicular; otras veces al avanzar 2 veces no llega al centro del objeto → avanzar más de 2 veces. Ambos robots se van quedando un poco retrasados.
Lateral	80%	18º	
Experimento 8			
Empujar	**40%**	**0.3s**	Funciona bien. Moway 4 se atrasa un poco y Moway 2 se adelanta.
Lateral	**80%**	**18º**	
Experimento 9			
Empujar	40%	0.3s	Demasiada velocidad, los robots laterales van a trompicones.
Lateral	100%	18º	
Experimento 10			
Empujar	40%	0.3s	Demasiada velocidad, los robots laterales van a trompicones. Se toma 80 para los robots laterales.
Lateral	90%	18º	
Experimento 11			
Empujar	60%	0.2s	Funciona bien. Los robots se van adelantando un poco, pero al retroceder, los laterales cumplen su cometido.
Lateral	80%	18º	
Experimento 12			
Empujar	**60%**	**0.25s**	Funciona muy bien. Mejor que el caso anterior. Ahora los robots laterales van más al compás del objeto.
Lateral	**80%**	**18º**	
Experimento 13			
Empujar	60%	0.25s	Funciona bien. Los laterales se adelantan un poco, pero al retroceder, siguen centrando la caja.
Lateral	80%	21.6º	
Experimento 14			
Empujar	60%	0.3s	Funciona bien. Los laterales se siguen adelantando. Pero al retroceder funciona bien.
Lateral	80%	21.6º	
Experimento 15			
Empujar	**60%**	**0.35s**	Funciona bien. Moway 2 se retrasa un poco y Moway 4 se adelanta un poco.
Lateral	**80%**	**21.6º**	
Experimento 16			
Empujar	60%	0.4s	Funciona bien. Ambos Moways se retrasan un poco.
Lateral	80%	21.6º	
Experimento 17			
Empujar	**80%**	**0.3s**	Funciona bien. Moway 2 se atrasa y Moway 4 se adelanta muy poco.
Lateral	**80%**	**21.6º**	
Experimento 18			
Empujar	80%	0.2s	Funciona bien. Los Moways laterales se adelantan.
Lateral	80%	18º	
Experimento 19			
Empujar	**80%**	**0.25s**	Funciona bien. Moway 4 se adelanta y Moway 2 se atrasa.

Lateral	**80%**	**18º**	
Experimento 20			
Empujar	100%	0.25s	Funciona bien. Ambos Moways se atrasan.
Lateral	80%	18º	
Experimento 21			
Empujar	100%	0.2s	Funciona bien. Los Moways se atrasan un poco, Moway 4 casi nada.
Lateral	80%	18º	
Experimento 22			
Empujar	100%	0.15s	Funciona bien. Ambos Moways se adelantan.
Lateral	80%	18º	
Experimento 23			
Empujar	100%	0.2s	Funciona bien. Ambos Moways se adelantan
Lateral	80%	21.6º	
Experimento 24			
Empujar	**100%**	**0.25s**	Funciona bien. Moway 2 se atrasa y Moway 4 se adelanta un poco.
Lateral	**80%**	**21.6º**	

Tabla 16: Experimentos para obtener los valores óptimos para el guiado de un objeto por un camino

Como se observa en la tabla anterior, se han tomado como parámetros óptimos aquellos que hacen que los robots laterales avancen más o menos al compás de la caja y la vayan centrando correctamente. A continuación aparecen juntos los datos de los experimentos que han tenido éxito:

Experimento 1		
Empujar	40%	0.4s
Lateral	60%	21.6º
Experimento 6		
Empujar	40%	0.4s
Lateral	80%	21.6º
Experimento 8		
Empujar	40%	0.3s
Lateral	80%	18º
Experimento 12		
Empujar	60%	0.25s
Lateral	80%	18º
Experimento 15		
Empujar	60%	0.35s
Lateral	80%	21.6º
Experimento 17		
Empujar	80%	0.3s
Lateral	80%	21.6º
Experimento 19		
Empujar	40%	0.3s
Lateral	100%	18º
Experimento 24		
Empujar	100%	0.25s
Lateral	80%	21.6º

Tabla 17: Parámetros idóneos para guiar un objeto por un camino

Se han encontrado 8 combinaciones diferentes en las que estos programas tienen un buen funcionamiento.

Se repiten los experimentos cronometrándolos. Se realizan 3 experimentos para cada caso: colocando los Moways laterales al principio de la caja, colocando los Moways centrados y colocándolos al final de la caja. A continuación aparece una tabla en la que se recogen los resultados, resaltando en negrita los que se realizan en un tiempo reducido.

Experimento 1		
Centrado	**Adelantado**	**Retrasado**
2min 6.1s	3min 3.6s	1min 43s
Experimento 6		
Centrado	**Adelantado**	**Retrasado**
1min 50.7s	1min 37.3s	1min 17.7s
Experimento 8		
Centrado	**Adelantado**	**Retrasado**
1min 53.8s	1min 22s	1min 40.0s
Experimento 12		
Centrado	**Adelantado**	**Retrasado**
1min 38.4s	1min 31.5s	1min 29.1s
Experimento 15		
Centrado	**Adelantado**	**Retrasado**
1min 9.8s	**1min 9.6s**	**1min 5s**
Experimento 17		
Centrado	**Adelantado**	**Retrasado**
56.8s	**1min 10s**	**1min 0.8s**
Experimento 19		
Centrado	**Adelantado**	**Retrasado**
1min 20.6s	1min 6.9s	1min 40.9s
Experimento 24		
Centrado	**Adelantado**	**Retrasado**
1min 2.3s	**1min 0.9s**	**1min 16.5s**

Tabla 18: Tiempo de los experimentos

A la vista de los tiempos obtenidos, los mejores resultados se obtienen con las pruebas 15, 17 y 24 en las que se obtienen tiempos de alrededor de 1 minuto. En concreto la prueba en la que se obtienen mejores resultados es la 17, cuyos parámetros son los siguientes: el robot que sigue la línea va a una velocidad del 80% durante 0.35s y los otros giran 21.6º y avanzan también a una velocidad del 80%.

Circuito curvo

Se utilizan estos algoritmos para un circuito curvo. Se prueban los 8 casos anteriores y se añaden otros experimentos, obteniéndose los resultados que se muestran en la tabla siguiente.

Experimento 1: Experimento 17			
Empujar	80%	0.3s	No funciona bien con estos parámetros. Al final la caja consigue centrarse, pero le cuesta, ya que Moway 2 se retrasa (por encontrarse en el lado exterior de la curva) y Moway 4 va adelantado. Moway 2 es el que debe ir empujando más la caja. La caja se sale bastante de las franjas exteriores. Moway 2, a veces, cuando va hacia adelante acaba yendo hacia atrás en los giros. En ocasiones, al ir hacia adelante, deja de detectar la caja porque ha retrocedido y entonces retrocede indefinidamente.
Lateral	80%	21.6º	
Experimento 2: Experimento 1			
Empujar	40%	0.4s	A veces funciona bien. Funciona mejor que el caso anterior ya que la caja sigue mejor el camino y no se sale de las franjas exteriores. Pero Moway 2 sigue avanzando a veces hacia atrás cuando intenta avanzar. Algunas veces retrocede indefinidamente. Moway 2 va algo retrasado y Moway 4 va adelantado.
Lateral	60%	21.6º	
Experimento 3: Experimento 6			
Empujar	40%	0.4s	Funciona más o menos bien. Igual que el caso anterior sólo que parece que ahora Moway 2 no retrocede y la caja va algo menos centrada a veces.
Lateral	80%	21.6º	
Experimento 4: Experimento 8			
Empujar	40%	0.3s	No funciona bien. La caja no consigue tener la dirección de la línea, ya que Moway 2 se atrasa bastante. La caja se sale bastante de las franjas exteriores. Sólo funciona bien si ambos robots empiezan adelantados, en este caso se consigue que la caja vaya centrada. Moway 2 sigue avanzando a veces hacia atrás cuando intenta ir hacia adelante. A veces Moway 4 va a trompicones.
Lateral	80%	18º	
Experimento 5: Experimento 12			
Empujar	60%	0.25s	Funciona bien a veces, cuando Moway 2 no se atrasa. La caja se sale de las franjas cuando Moway 2 se atrasa. A veces Moway 4 va a trompicones.
Lateral	80%	18º	
Experimento 6: Experimento 15			
Empujar	60%	0.35s	Funciona bien a veces. Cuando Moway 2 no se pone a retroceder indefinidamente, porque en lugar de avanzar se retrasa y pierde la caja de delante. La caja va más o menos centrada, a veces se sale un poco de la franja exterior. Moway 2 se atrasa y Moway 4 se adelanta.
Lateral	80%	21.6º	
Experimento 7: Experimento 19			
Empujar	80%	0.25s	A veces funciona. Cuando Moway 2 no se atrasa mucho. Al final la caja se queda centrada, más o menos siempre. Moway 2 se atrasa y Moway 4 se adelanta, por lo que el objeto a veces se sale mucho de las franjas.
Lateral	80%	18º	
Experimento 8: Experimento 24			
Empujar	100%	0.25s	Sucede como en los casos anteriores. Moway 2 se retrasa y avanza indefinidamente hacia atrás. La caja puede llegan a salirse bastante de las franjas.
Lateral	80%	21.6º	
Experimento 9			
Empujar	60%	0.2s	Moway 2 sigue yéndose hacia atrás

Lateral	60%	14.4º	
Experimento 10			
Empujar	60%	0.2s	Ambos Moways van adelantándose a la caja, pero a veces Moway 2 sigue yéndose hacia atrás. Ahora la caja se queda más centrada.
Lateral	60%	18	
Experimento 11			
Empujar	60%	0.15s	Sucede lo mismo que en el caso anterior.
Lateral	60%	18	
Experimento 12			
Empujar	40%	0.15s	Funciona bien pero va algo lento.
Lateral	60%	18	

Tabla 19: Experimentos con diferentes parámetros para un camino curvo

Tras observar que los parámetros de los programas anteriores no funcionan cuando el camino es curvo, se prueban otros parámetros:

- **Se baja la velocidad de los laterales** porque a veces van a trompicones.
- **Se disminuye el avance del objeto** para que hayan más puntos en los que los robots laterales lo centren y así vaya mejor por el camino.
- **Se hace que los laterales avancen por la parte delantera del objeto** para poder orientarlo mejor.

“Empujar_3.c”, “LateralDer_3.c” y “LateralIzq_3.c”

Se decide crear los programas “Empujar_3.c”, “LateralDer_3.c” y “LateralIzq_3.c”, modificando el código de los programas anteriores para adaptarse a los requerimientos del nuevo camino:

- En este caso **los Moways laterales sólo avanzan 5mm.** tras ponerse perpendiculares a la línea.
- **Los robots laterales sólo retroceden 1 vez al adelantarse**, ya que ahora conviene que los robots vayan en la parte delantera del objeto.
- **Se disminuye la distancia de retroceso a 6cm**. para que Moway 2 no retroceda cuando intenta avanzar.

En la tabla 20 aparecen los experimentos realizados con los cambios que se han especificado:

Experimento 12			
Empujar	40%	0.15s	Funciona bien, pero sigue siendo algo lento.
Lateral	60%	18	
Experimento 13			
Empujar	60%	0.15s	Funciona bien, pero ambos Moways se adelantan demasiado.
Lateral	70%	21.6	
Experimento 14			
Empujar	**60%**	**0.2s**	Funciona bien.
Lateral	**70%**	**21.6**	

Tabla 20: Experimentos con diferentes parámetros para guiar un objeto por un camino curvo

Se observa que se consigue que los robots realicen la tarea correctamente, la caja permanece centrada y los robots laterales siempre van al principio de ésta para orientarla. Para ello se han tomado los siguientes parámetros:

- Robot que empuja: 60% de la velocidad y avance de 0.2segundos.
- Robots laterales: 70% de la velocidad y ángulo de 21.6º.

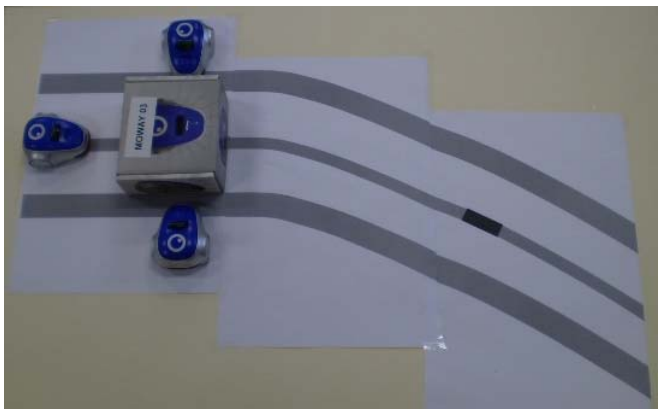

Figura 52: Guiado de un objeto por un camino curvo

Se estudia la posibilidad de que el robot del centro vaya mandando la información de sus sensores a los otros robots. Con esto se pretende que los robots de los lados avancen o esperen al siguiente turno según la forma del camino. Para ello se realizan 3 experimentos:

Experimento 1

El primer experimento se puede ver en los programas "Empujar_4.c", "LateralDer_4.c" y "LateralIzq_4.c". Consiste en que **el robot central siga la línea por el lado izquierdo y mande** por RF **la dirección del camino**. La dirección se obtiene comparando el número de veces que el robot ha girado hacia un lado y hacia el otro mientras avanza. Para ello se tienen las variables "der" (derecha) e "izq" (izquierda) que se incrementan cuando el robots central gira en ese sentido. A continuación se comenta la forma de calcular estas variables y cómo deben actuar los robots laterales:

- **Se le suma 1 a "der" si no se detecta línea** con ningún sensor, la curva es en sentido horario → el robot que está a la izquierda debe avanzar más que el que está a la derecha.
- **Se le suma 1 a "izq" si se detecta** línea **con ambos sensores**, la curva es en sentido antihorario → el robot que está a la derecha debe avanzar más que el que está a la izquierda.
- **Se le suma 2 a "izq" si sólo se detecta con el sensor izquierdo**, ya que se produce una curva muy pronuncia a la izquierda.

Según los valores de las variables "izq" y "der", **el robot central envía un valor** en la posición 2 del vector de variables de salida. Y según este valor los robots **laterales realizan un movimiento** u otro.

- Si **"izq"** es **mayor que "der"** → el valor es **1** → el **robot derecho avanza 2 veces** y el izquierdo ninguna.
- Si **"der"** es **mayor que "izq"** → el valor es **2** → el **robot izquierdo avanza 2 veces** y el derecho ninguna.

- Si **“izq” y “der” son iguales** → el valor es **0** → ambos **avanzan 1 vez**.

Cuando uno de los laterales avanza dos veces, se comprueba si tras el primer avance el robot se ha adelantado, si se ha adelantado, retrocede y ya no vuelve a avanzar.

Se añade que no se tenga en cuenta la detección de los sensores mientras el robot no detecta el objeto, que se empiecen a tener en cuenta cuando el sensor de obstáculo toma valores superiores a 100: “senObsMin2”. Se modifica la función de seguimiento de línea para que se modifiquen las variables “izq” y “der” si los sensores de obstáculo superan este valor. Con esto se consigue que el robot se enderece al principio.

Resultados

Se observa que cuando la línea es recta, a veces Moway toma que la curva es a la derecha, que es lo que pasa al principio, da igual que se aumente el valor del sensor de obstáculo, sigue pasando lo mismo, se detecta como giro a la derecha la línea recta.

Se prueba el programa hacia el otro lado, y falla, cuando debería avanzar el lateral derecho 2 veces, avanza 2 veces el izquierdo. Por lo que estos cambios solo sirven si la curva es en sentido horario.

Experimento 2

Como segundo experimento se cambia la forma de calcular el valor del movimiento realizado. En este caso **se envía el valor del último movimiento del robot que sigue la línea**. Pero sigue sin funcionar bien, las líneas rectas se siguen detectando como curvas a la derecha.

Experimento 3

Este caso es similar al primero, pero en lugar de que **el robot central** siga la línea por el borde izquierdo, la **sigue por el centro**.

- Si detecta con ambos sensores→ avanza recto y se pone “dir” a 0.
- Si detecta sólo con el sensor izquierdo → gira a la izquierda, se suma 1 a “izq” y se pone “dir” a 1.
- Si detecta sólo con el sensor derecho → gira a la derecha, se suma 1 a “der” y se pone “dir” a 2.
- Si no detecta con ningún sensor → el movimiento es contrario al giro que se ha realizado en el movimiento anterior. Para ello en los anteriores casos se ha modificado la variable "dir". Si "dir" es 1 gira a la derecha y si es 2 gira a la izquierda y se suma 1 a la dirección de giro.

Resultados

Se observa que el sistema funciona peor. En este caso es el robot lateral derecho el que avanza 2 veces cuando el camino es recto, y lo hace más a menudo que en los casos anteriores. Al hacer el recorrido al revés funciona muy mal, porque el lateral izquierdo va avanzando 2 veces casi siempre y el otro se queda atrás.

La función de seguimiento de línea por el centro se encuentra en el archivo “Sigue_Lin_Centro.c” y se puede consultar en el CD adjunto a esta memoria.

A la vista de los resultados de estos 3 experimentos, se opta por tomar como mejor solución a la hora de guiar un objeto por un camino curvo, el sistema formado por los programas “Empujar_3.c”, “LateralDer_3.c” y “LateralIzq_3.c”, ya que con ellos se consigue que el objeto vaya centrado y dentro de las franjas en todo momento.

En el CD que se adjunta a esta memoria se pueden encontrar 2 videos para el guiado de un objeto por un camino, uno para un camino recto y otro para un camino curvo.

CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se han estudiado las posibilidades que ofrece el microbot Moway para realizar tareas de robótica cooperativa con robots móviles.

Las principales limitaciones que hemos encontrado en este robot son su escasa potencia de procesamiento dado que está equipado con un microcontrolador de bajas prestaciones, y las limitaciones en la sensorización dado que no es posible incorporar cámaras de video. Adicionalmente, nuestros experimentos han puesto de manifiesto que la fiabilidad de sus desplazamientos es muy reducida. Sin embargo, la disponibilidad de un módulo de radiofrecuencia que permite a dos Moways comunicarse entre sí permite realizar desarrollos en robótica cooperativa.

La parte central de nuestro trabajo ha sido el diseño y resolución de una serie de tareas cooperativas (las detalladas en el capítulo 4) haciendo uso del módulo de comunicación por radiofrecuencia.

Como se ha explicado a lo largo de este documento, las tareas cooperativas que hemos podido realizar con los robots Moway incluyen la compartición de datos sensoriales entre robots; la cooperación para desplazar objetos pesados o voluminosos; y la cooperación para realizar tareas más eficientemente, como la limpieza de una superficie o el guiado de objetos por un camino. Finalmente, tras la programación de diversos algoritmos, en todos los casos propuestos hemos obtenido resultados satisfactorios, tal como se puede apreciar en los videos demostrativos que se adjuntan dentro del material complementario en el CD.

Como trabajos futuros se podría estudiar la posibilidad de adquirir un módulo de expansión (como el que se observa en la figura 53) para poder añadir circuitos electrónicos a los robots. Una de las posibilidades de este módulo podría ser el dotar a Moway de luz mediante una serie de LEDS o que emita sonidos. Así los robots podrían detectarse entre sí. Este módulo puede dar mucho juego ya que se pueden diseñar numerosos circuitos electrónicos que doten a Moway con nuevas capacidades.

Figura 53: Módulo de expansión

Otra de las tareas a realizar en un futuro podría ser añadirle al robot más sensores. Con unos sensores de obstáculos laterales y unos traseros, se detectarían todos los objetos que se encuentren alrededor del microbot con mayor facilidad. A la hora de añadir sensores, los de un mismo robot deberían tener una sensibilidad similar (tanto si son de línea como si son de obstáculo).

En la web del robot se plantea la posibilidad de crear una versión mejorada del microbot a la que se le incorporaría una cámara VGA de color y un sistema de posicionamiento por infrarrojo. Estos 2 elementos aumentarían en gran medida la capacidad del robot, ya que diferenciaría los objetos que ve mediante la cámara (distinguiendo a otros robots de los demás objetos) y sabría en todo momento la posición exacta en la que se encuentra, por lo que no se necesitaría dividir el terreno en varias zonas. Todo esto reduciría el código de los programas y permitiría realizar numerosas tareas de forma más sencilla y eficiente.

ANEXO I. VALORES DEL SENSOR DE LÍNEA

A continuación se presenta una tabla en la que se recogen ordenadamente los datos obtenidos por los sensores de los robots en diferentes pruebas realizadas para diferentes materiales. Se ha tomado el rango de los valores detectados por ambos sensores de ambos robots, para tener un rango genérico. Y se ha establecido un rango ampliado, dando un mayor margen al rango, así se pretende asegurar que todos los sensores se encuentran dentro de esos rangos.

Cinta aislante negra							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
DC-E4	220-228	E4-E7	228-231	E5-EB	229-235	E7-E8	231-232
Rango valores		220-232		**Rango ampliado**		210-240	
Luminancia 10							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
B7-DB	183-219	CB-E4	203-228	CE-E6	206-230	D1-E8	209-232
Rango valores		183-232		**Rango ampliado**		175-240	
Luminancia 15							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
B6-DB	182-219	C6-E4	198-228	CA-E5	202-229	CD-E7	205-231
Rango valores		182-231		**Rango ampliado**		175-240	
Luminancia 30							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
B6-D7	182-215	C2-DA	194-218	CD-E2	205-226	CF-E4	206-228
Rango valores		182-228		**Rango ampliado**		175-235	
Luminancia 40							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
B1-CE	177-206	C2-DA	194-218	CB-DC	203-220	CD-DF	205-223
Rango valores		177-223		**Rango ampliado**		170-230	
Luminancia 50							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
A2-C6	162-198	B8-D4	184-212	BD-D6	189-214	C1-D8	193-216
Rango valores		162-216		**Rango ampliado**		155-225	
Luminancia 50 color							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
B7-C2	183-194	C8-D2	200-210	CD-D4	205-212	D0-D6	208-214
Rango valores		183-214		**Rango ampliado**		175-220	
Luminancia 60							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
9F-C3	158-195	B4-D1	180-208	B6-D7	182-215	BB-D7	187-215
Rango valores		158-215		**Rango ampliado**		150-225	

Luminancia 70							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
92-BA	146-186	AB-CA	171-202	B3-CE	179-206	B6-D1	182-209
Rango valores		146-209		**Rango ampliado**		140-215	
Luminancia 80							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
8B-B1	139-177	A5-C3	165-195	AE-C7	174-199	B1-C9	177-201
Rango valores		139-201		**Rango ampliado**		130-210	
Luminancia 90							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
8B-A7	139-167	A6-BC	166-188	A3-C3	163-195	B0-C5	176-197
Rango valores		139-197		**Rango ampliado**		130-205	
Luminancia 100							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
78-9F	120-159	98-B6	152-182	A1-BB	161-187	A5-BF	165-191
Rango valores		120-191		**Rango ampliado**		110-200	
Luminancia 100 color							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
8A-98	138-152	A2-B4	162-180	AA-B7	170-183	AF-B9	174-185
Rango valores		138-185		**Rango ampliado**		130-195	
Cinta gris							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
5D-88	93-136	7D-A2	125-162	8A-A4	138-164	89-A7	137-167
Rango valores		93-167		**Rango ampliado**		85-175	
Cinta aislante azul							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
44-5F	68-79	70-8A	112-138	6E-91	110-145	75-97	117-151
Rango valores		68-151		**Rango ampliado**		60-160	
Luminancia 150							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
3C-62	60-98	64-86	100-134	68-8C	104-140	79-94	121-148
Rango valores		60-148		**Rango ampliado**		50-155	
Luminancia 175							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
21-35	33-53	4F-60	79-83	5B-6B	91-107	64-72	100-114
Rango valores		33-114		**Rango ampliado**		25-120	
Luminancia 200							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
12-15	18-21	33-41	51-65	43-4E	67-78	4A-56	74-86
Rango valores		18-86		**Rango ampliado**		10-95	

Luminancia 220							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
F-10	15-16	16-1F	22-31	23-2E	35-46	2C-37	44-55
Rango valores		15-55		**Rango ampliado**		5-65	
Cartulina verde							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
E	14	E-F	14-15	F-11	15-17	10-11	16-17
Rango valores		14-17		**Rango ampliado**		5-25	
A4							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
D-E	13-14	D-10	13-16	10-12	16-18	11-13	17-19
Rango valores		13-19		**Rango ampliado**		5-30	
Cinta aislante blanca							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
E	14	E-10	14-16	10	16	F-11	14-17
Rango valores		14-17		**Rango ampliado**		5-25	
Cartulina beige							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
D-E	13-14	D-F	13-15	10-12	16-18	F-11	15-17
Rango valores		13-18		**Rango ampliado**		5-25	
Cinta aislante blanca							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
E	14	E-10	14-16	10	16	F-11	14-17
Rango valores		14-17		**Rango ampliado**		5-25	
Cartulina naranja							
Moway 3				**Moway 4**			
Sensor izquierdo		**Sensor derecho**		**Sensor izquierdo**		**Sensor derecho**	
D-E	13-14	9-10	9-16	10-11	16-17	B-11	11-17
Rango valores		9-17		**Rango ampliado**		0-25	

Tabla 21: Valores del sensor de línea

BIBLIOGRAFÍA

[1] http://www.moway-robot.com/, Microbot Moway.

[2] http://www.robotic-lab.com/, Robotic Lab.

[3] "Microcontrolador PIC16F84, Desarrollo de proyectos", E. Palacios, F. Remiro, L.J. López. Ra-Ma, 2004.

[4] "Microcontroladores PIC diseño práctico de aplicaciones", J.M. Angulo Usategui, McGraw-Hill D.L.1999

[5] "Microcontroladores PIC la solución en un chip", E. Martín Cuenca, Paraninfo D.L. 1999

[6] "Microcontroladores PIC la clave del diseno ", E. Martin Cuenca, Thomson [2003]

[7] http://www.scribd.com/doc/3023860/CURSO-PIC16F87X-6, Curso PIC16F87X-6.

[8] http://www.ccsinfo.com, CCS Inc.

[9] http://www.ro-botica.com/, Ro-botica.