UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

INGENIERÍA INDUSTRIAL

"ESTUDIO DE LA ODOMETRÍA EN LOS ROBOTS MOWAY Y DESARROLLO DE SUS CAPACIDADES MÓVILES MEDIANTE EL USO DEL SENSOR DE LUZ"

PROYECTO FIN DE CARRERA
Septiembre – 2009

AUTORA: Carmen Coll Cámara
DIRECTORA: María Asunción Vicente Ripoll

VISTO BUENO Y CALIFICACIÓN DEL PROYECTO

Título proyecto:

ESTUDIO DE LA ODOMETRÍA EN LOS ROBOTS MOWAY Y DESARROLLO DE SUS CAPACIDADES MÓVILES MEDIANTE EL USO DEL SENSOR DE LUZ

Proyectante: CARMEN COLL CÁMARA

Directora: MARÍA ASUNCIÓN VICENTE RIPOLL

VºBº directora del proyecto:

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 estudio de la odometría del robot móvil Moway y el desarrollo de algoritmos para mejorar su movilidad, mediante el uso del sensor de luz incorporado en el microbot.

En la última década, la utilización de robots se ha ido extendiendo hacia nuevos sectores como son el educativo, servicios y ocio, gracias a su abaratamiento y a la reducción de tamaño, una tendencia relacionada con la miniaturización de los componentes electrónicos que se utilizan para controlarlos.

Acorde al rumbo que está tomando la robótica en nuestros días, surge Moway: una herramienta práctica en el mundo de la enseñanza y el aprendizaje. De pequeño tamaño, compacto, programable y autónomo, Moway ha sido creado para comunicarse inalámbricamente con los de su especie o con un ordenador, siendo capaz de interactuar con su entorno gracias a los sensores de línea, obstáculos y luz integrados en él.

El trabajo realizado en este proyecto se puede dividir en los siguientes aspectos:

- Estudio detallado del nivel físico del microbot, especialmente sus sistemas de detección: sensores de línea, de obstáculos y sensor de luz.
- Análisis de los programas ejemplo de manejo de sensores que se distribuyen junto con el microbot.
- Estudio de la odometría del microbot a través de la ejecución de varios experimentos con diferentes trayectorias con parametrizaciones diversas de velocidad y ángulo de giro.
- Mejora de las capacidades móviles del microbot haciendo uso de su módulo sensor de luz.
- Desarrollo de tareas o juegos cooperativos que aprovechen la información local proporcionada por los sistemas de detección.

Todos estos apartados se tratan en la memoria en el orden siguiente:

En el primer capítulo se describe tanto el microbot Moway a nivel físico como el funcionamiento del software de grabación, programas ejemplo distribuidos junto al microbot y las librerías de sensores y motores.

En el capítulo 2 se introduce el término de odometría. Se intentan estimar los errores producidos en movimientos rectilíneos, curvos y en los giros, además de los que se producen al intentar subir superficies con pendiente. Al final del capítulo se llegará a una serie de conclusiones.

En el tercer capítulo, se detallan las características de los sensores de luz y de línea y se muestran las diversas pruebas realizadas, para determinar cuales son las características que debe tener el medio para que el microbot interactúe correctamente con él, explotando al máximo las posibilidades que nos ofrecen los sensores integrados en Moway.

El capítulo 4 nos lleva a familiarizarnos con Mowaygui, un programa gráfico que genera automáticamente programas en lenguaje ensamblador a partir de diagramas de flujo, sin necesidad de tener conocimientos de programación.

El capítulo 5 contiene las aplicaciones desarrolladas con el microbot:

- La primera de ellas es la corrección de trayectoria rectilínea mediante el uso de una baliza luminosa.
- La segunda y tercera aplicación se desarrolla en un museo, en el que Moway actúa como guardia de seguridad y guía del museo respectivamente.
- En la cuarta aplicación Moway se logra una interacción máxima con el medio llevando los objetos luminosos que encuentre a su paso, hacia un extremo de la pista y los no luminosos al extremo contrario.
- Y en la última aplicación se simula un juego recreativo, en el que se emplean dos Moway, actuando como contrincantes, para ver cual de ellos es capaz de meter más goles en la portería contraria.

En el capítulo 6 se realiza una recopilación de conclusiones y posibles trabajos futuros.

Adjunto al proyecto, se encuentra un CD con el código completo de cada una de las aplicaciones y los videos que muestran la ejecución de cada uno de los programas. Además se adjuntan las hojas de características del sensor de luz APDS-9002 y del sensor de línea KTIR0711S.

CAPÍTULO 1

PRIMEROS CONCEPTOS

CAPÍTULO 1: PRIMEROS CONCEPTOS

1.1. DESCRIPCIÓN FÍSICA DEL MICROBOT MOWAY

1.1.1. Introducción

Moway es un pequeño robot diseñado principalmente para realizar aplicaciones prácticas de microbótica. Nace con vocación de ser una herramienta práctica dentro del mundo de la enseñanza y el aprendizaje, ideal para todos aquellos interesados en la robótica que deseen introducirse en este mundillo de forma económica.

Figura 1: Moways junto a una placa USB con transceptor BZI-RF2GH4

Moway debido a su pequeño tamaño es denominado robot de bolsillo, 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.

Compacto con un diseño que le permite moverse con agilidad y elegancia, es apto para llevar a cabo tareas como el detección de obstáculos, reclusión en recintos cerrados, seguimiento de líneas y reconocimiento de focos luminosos.

A continuación se nombran algunas de las habilidades más destacables del robot:

- Programable en diferentes lenguajes: Moway puede ser programado en lenguaje ensamblador, en C o a través de un compilador grafico diseñado específicamente para él, desarrollando desde aplicaciones muy sencillas a las mas complejas.
- Dotado de múltiples sensores que le permiten interactuar y desenvolverse en un entorno real.
- Permite la comunicación con otros robots de la misma especie, para aplicaciones en las que se dispone de varios Moway.
- Permite la comunicación con un ordenador para su programación y desarrollo de aplicaciones

- 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 uno o más robots por RF.
- Autonomía de 2 horas
- Preparado para robótica colaborativa

1.1.2. Componentes del microbot moway

Los elementos que conforman el mini-robot Moway son los siguientes:

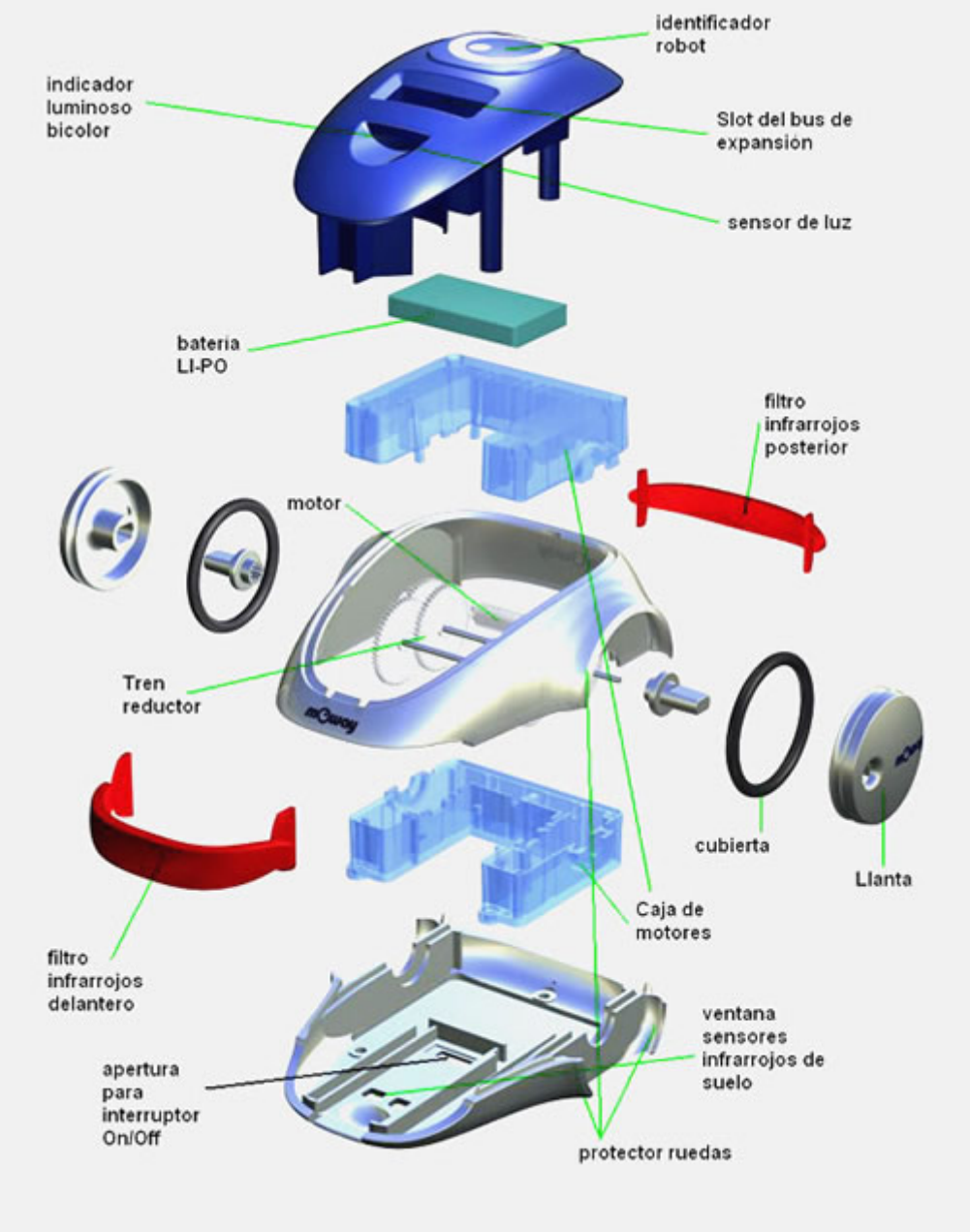

Figura 2: Componentes de Moway

En el interior de un Moway tenemos los siguientes elementos:

- 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: Conexiones PIC-sensores

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: Conexiones PIC-Base Moway

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. Constan 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.

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 (creada a partir de una rueda 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.

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 podemos 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.

A continuación se mostrarán los tres casos que se pueden dar:

- 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.

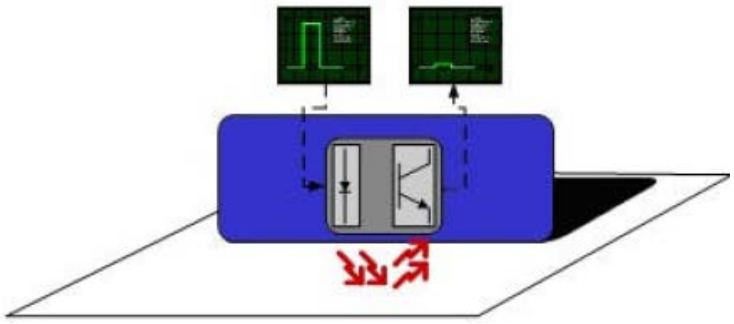

Figura 3: Sensor de línea en superficie clara

- 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.

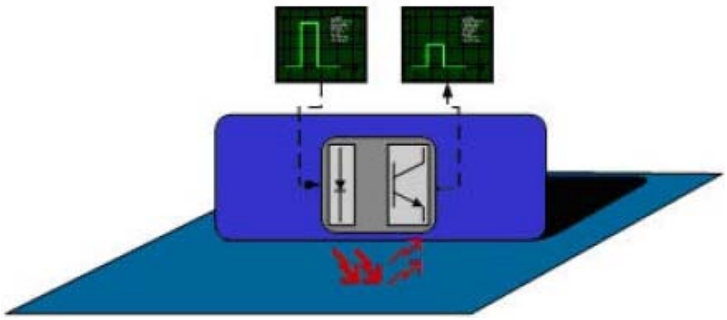

Figura 4: Sensor de línea en superficie de color

- Superficie oscura: La superficie oscura hace que se refleje muy poca luz teniendo un voltaje alto a la salida del sensor.

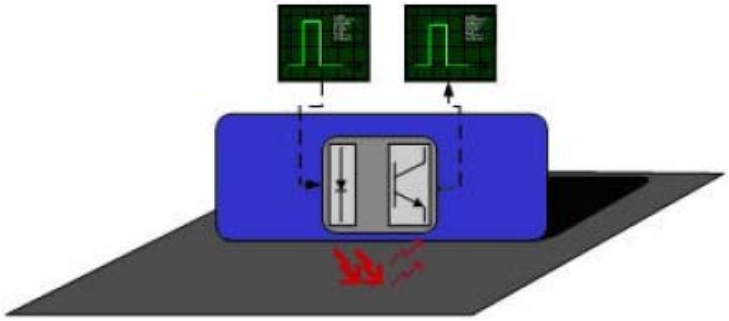

Figura 5: Sensor de línea en superficie oscura

En la siguiente tabla se muestran las conexiones del PIC con el sensor de línea (los dos diodos LED están conectados al mismo pin del microcontrolador).

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: Conexiones Sensores de línea-PIC

Sensores detectores de obstáculos

Al igual que los sensores de línea, los sensores 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 70us que, si existe un obstáculo, el receptor capta utilizando una etapa de filtrado y amplificación. Una vez procesada la señal electrónicamente, el PIC puede medirla mediante el ADC o como entrada digital. 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.

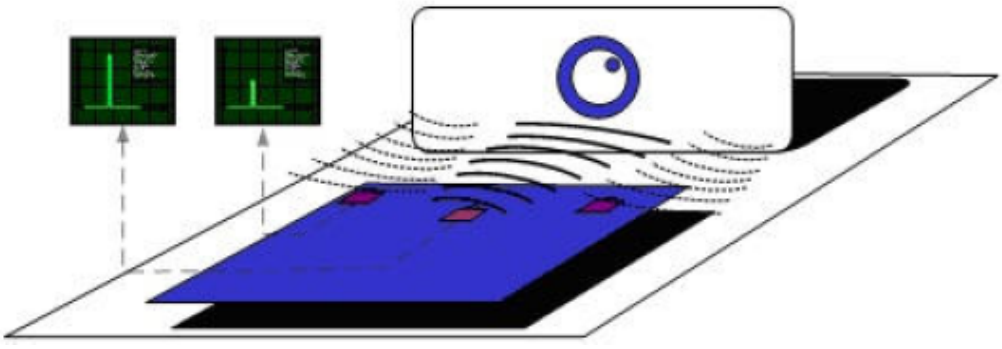

Figura 6: Sensor detector de obstáculos

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: Conexiones sensor antichoque-PIC

Sensor de luz

Este sensor permite a Moway 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 y actuar en consecuencia.

La salida del sensor APDS-9002 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: Conexión PIC-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.

Figura 7: Leds frontales

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

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

Tabla 6: Conexión PIC-Led 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. Cabe destacar que al compartir la misma apertura que el sensor de luz es fundamental apagarlos en el momento que se desee hacer una lectura de la intensidad de luz.

Figura 8: Led superior bicolor

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

Tabla 7: Conexión PIC-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 compleja comunicación inalámbrica.

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: Conexiones PIC-Conector de expansión

Como se observa en la 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 de Moway con otros de su especie y con el PC mediante la Base Moway.

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: Conexión PIC-pad libre

Sistema de alimentación

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

Para finalizar la descripción de componentes del robot Moway mencionaremos el modulo de radiofrecuencia que aparece como opcional en los kits y del puerto USB.

Módulo de radiofrecuencia

El módulo de radiofrecuencia para comunicación inalámbrica, consiste en una pareja de transceptores BZI-RF2GH4 que permiten a Moway comunicarse con otros robots o con el ordenador de forma inalámbrica.

Se trata de unos módulos de muy bajo coste, que se alimentan con 3.3V. , tienen un reducido tamaño, antena integrada y velocidad de comunicación de hasta 2000 kbps. El alcance es de unos 25m en interiores a una frecuencia de 2.4GHz.

Se puede ajustar con 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. 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.

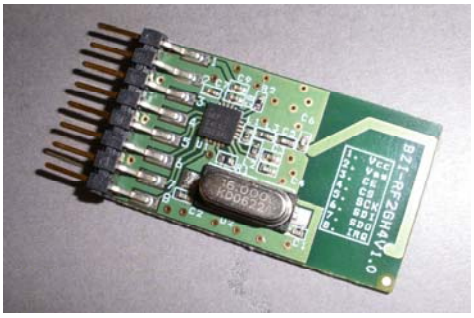

Figura 9: Transceptor BZI-RF2GH4

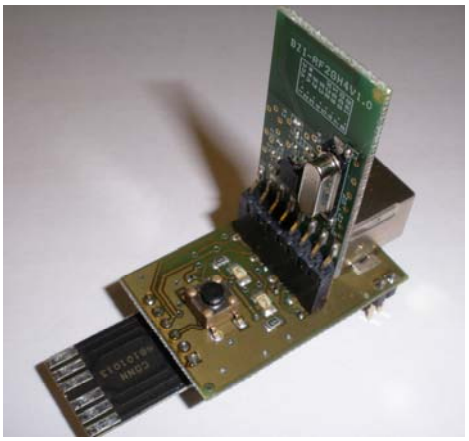

Figura 10: Transceptor insertado en la placa USB

Placa USB

Tarjeta electrónica para la carga de las baterías y conexión USB con el ordenador.

A través de ella conectaremos el robot Moway al ordenador para grabarle el programa pertinente en cada momento.

En el caso de que la batería no este al 100% de su capacidad, de forma automática se encenderá una luz verde en la placa USB que nos indicará que la batería está siendo cargada. Una vez cargada totalmente, esta luz se apagará.

Figura 11: Placa USB

1.2. SOFTWARE DE GRABACIÓN: MOWAY CENTER

1.2.1. Base Moway

En cuanto a la Base Moway, una PIC 18F2550, sus tareas son generar las señales de grabación de Moway, controlar la carga de la batería y gestionar las comunicaciones con otros Moways o con el ordenador.

1.2.2. Descripción de Moway Center

Para controlar la Base Moway se utiliza el software 'Moway Center'. El programa consta de dos partes: la principal y la de radio frecuencia.

1.2.2.1. Principal

Figura 12: Ventana principal de Moway Center

En esta ventana se pueden apreciar los siguientes apartados:

Moway Status

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

- Conectado. El robot está conectado a la base y encendido.

Figura 13: Estado conectado

- Apagado. El robot está conectado a la base pero apagado, así se puede ir cargando la batería.

Figura 14: Estado apagado

- Desconectado. El robot no está conectado a la base.

Figura 15: Estado desconectado

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 16: Acciones de Moway Center

Batería

Aparece el porcentaje de carga de la batería.

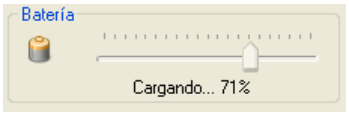

Figura 17: Estado de carga de la batería

Radio-Frecuencia

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

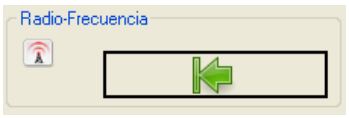

Figura 18: Apartado de Radio Frecuencia

Info

Muestra diferente información sobre el software y los procesos.

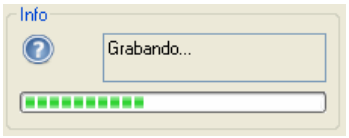

Figura 19: Ventana Info a mitad del proceso de grabación

1.2.2.2. Radio Frecuencia

Figura 20: Ventana de radiofrecuencia de Moway Center

Con esta ventana se pueden enviar y recibir datos a través de la tarjeta de comunicaciones BZI-RF2GH4. Observamos que se divide en 4 secciones:

Configuración RF

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

Figura 21: Configuración RF

Dato a enviar

Los datos a enviar se indican en hexadecimal y tienen que tener los 8 bytes definidos.

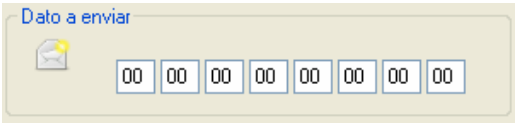

Figura 22: Dato a enviar

Envío

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

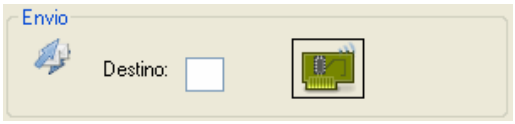

Figura 23: Ventana Envío

Recepción

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

Figura 24: Ventana de recepción de datos

1.2.3. Grabación de un programa mediante Moway Center

Primero se instala Moway Center. El programa se encuentra disponible en la página web de Moway. Al instalarlo obtenemos, a parte del programa, el driver de la base Moway, las librerías de motores y sensores, el manual de moway, los recursos de radio frecuencia y algunos proyectos compilados.

Luego se conecta la base al PC mediante el cable con conexión USB. La primera vez que el ordenador detecta la base se debe indicar la dirección del driver en el asistente para nuevo hardware encontrado.

Figura 25: Moway conectado a la placa USB

Para poder grabar un programa, conectamos el robot a la base y ésta el ordenador. Ejecutamos Moway Center, el cual nos indica el estado del robot y su carga, y encendemos el robot. Pulsamos el botón grabar, en apartado acciones y seleccionamos el archivo .HEX del programa a grabar. Finalmente apagamos el robot, lo desconectamos de la base. Cada vez que lo encendamos ejecutará el programa grabado.

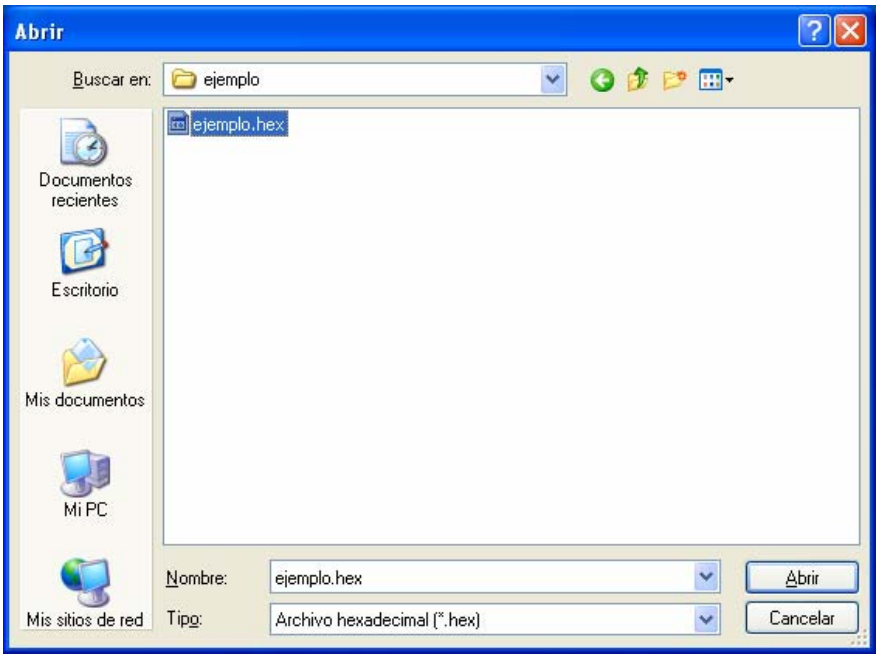

Figura 26: Selección del archivo a grabar

Se puede utilizar MPLAB para programar tanto en ensamblador como en C, ó CCS PIC C Compiler para programar únicamente en C.

En el proyecto presente se programará en C ya que los programas requieren menos instrucciones, menos memoria, y es más intuitivo.

1.2.4. Creación de un proyecto con CCS PIC C Compiler

Para crear el proyecto seleccionamos 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".

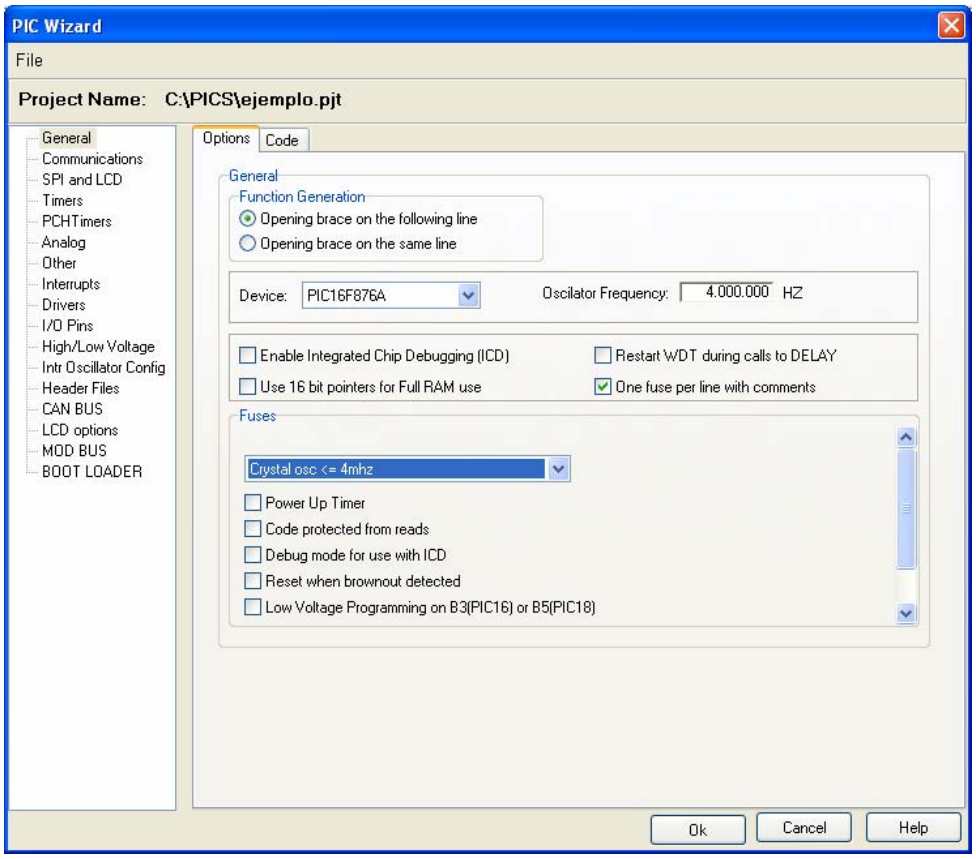

Figura 27: Selección de la frecuencia y tipo de reloj

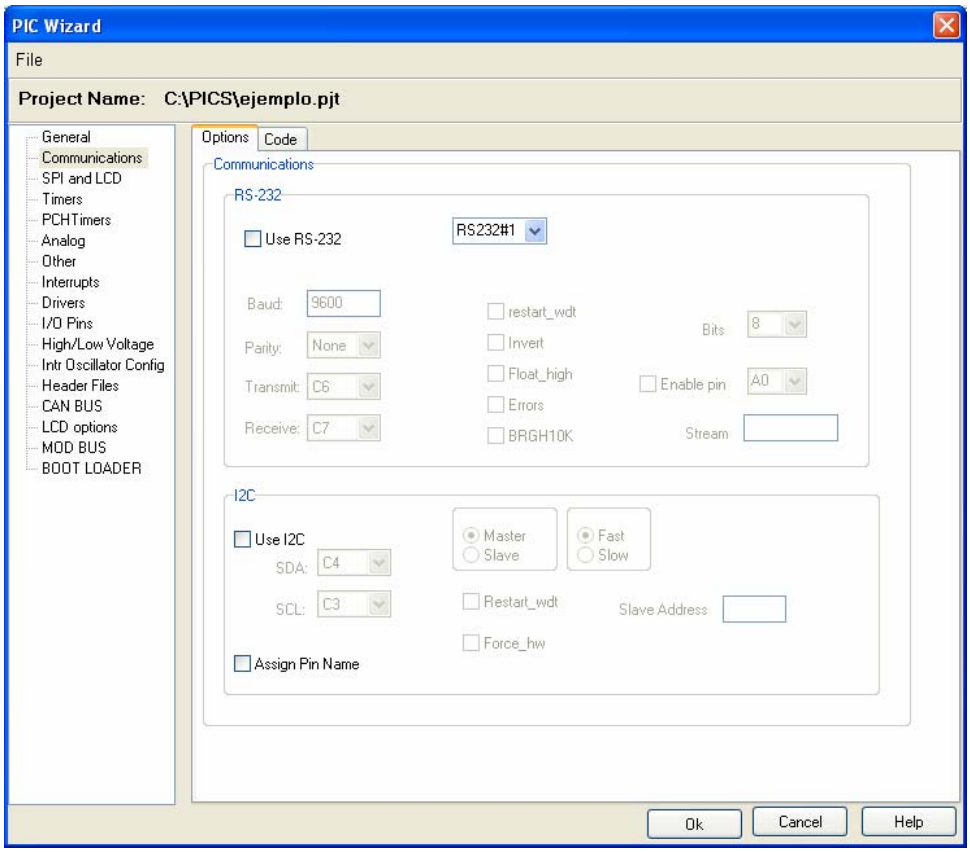

Figura 28: Deshabilitación del RS-232

Introducimos las librerías en la carpeta del proyecto y también las incluimos en la cabecera el programa.

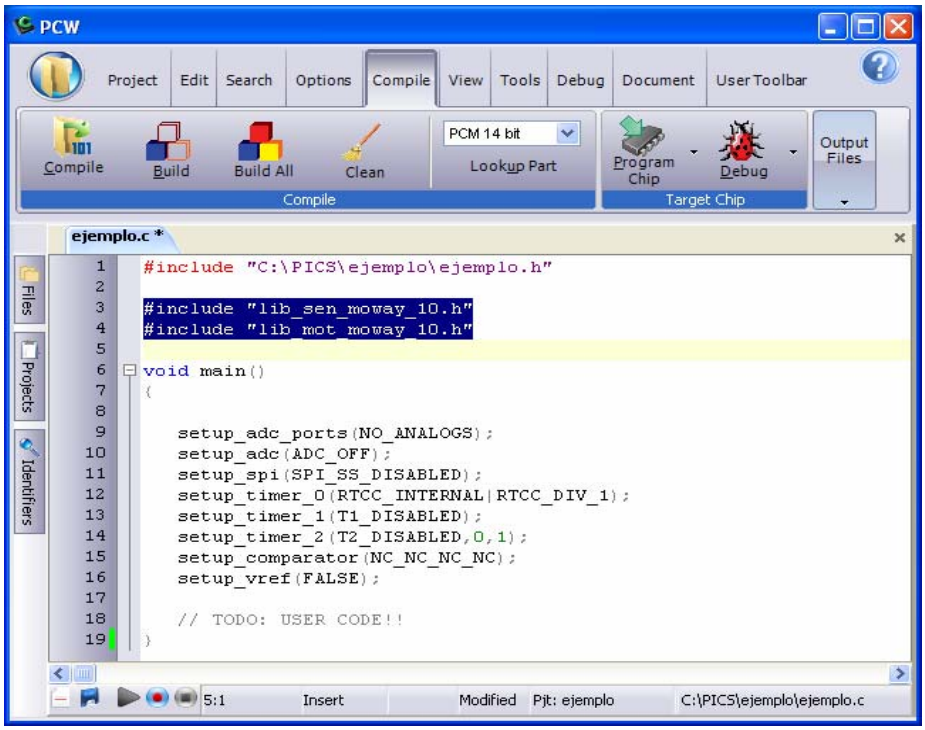

Figura 29: Código generado automáticamente, incluyendo librerías

Escribimos el programa y compilamos mediante compile del menú Compile.

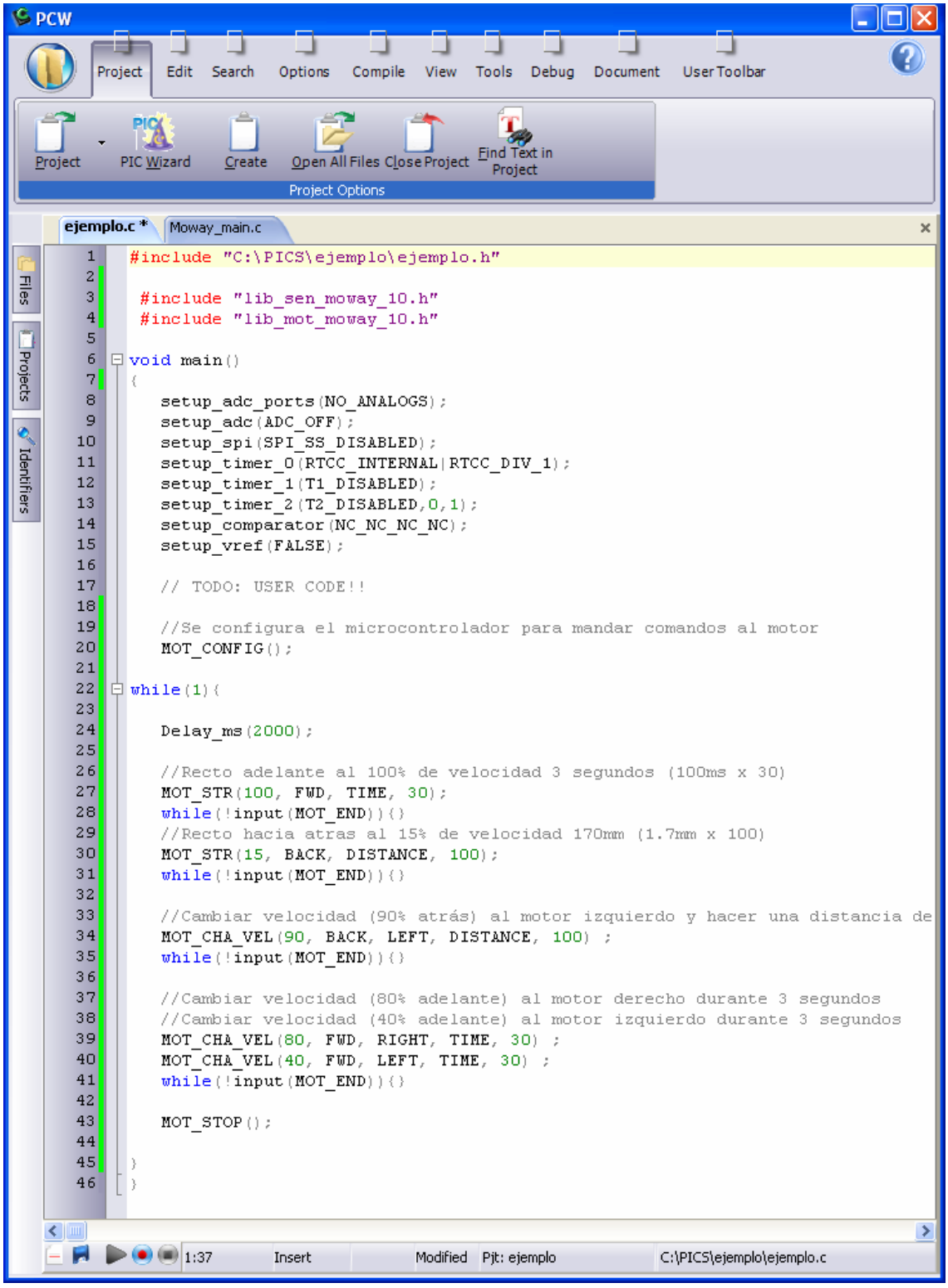

Figura 30: Código ejemplo

Finalmente mediante la opción Built All, el programa nos crea el archivo .HEX que grabaremos al robot.

1.2.5. Creación de un proyecto con MPLAB

Para crear el proyecto se accede a Project Wizard desde la barra Project. Se selecciona el tipo de microcontrolador, en este caso PIC16F876A ya que es el microcontrolador integrado en Moway.

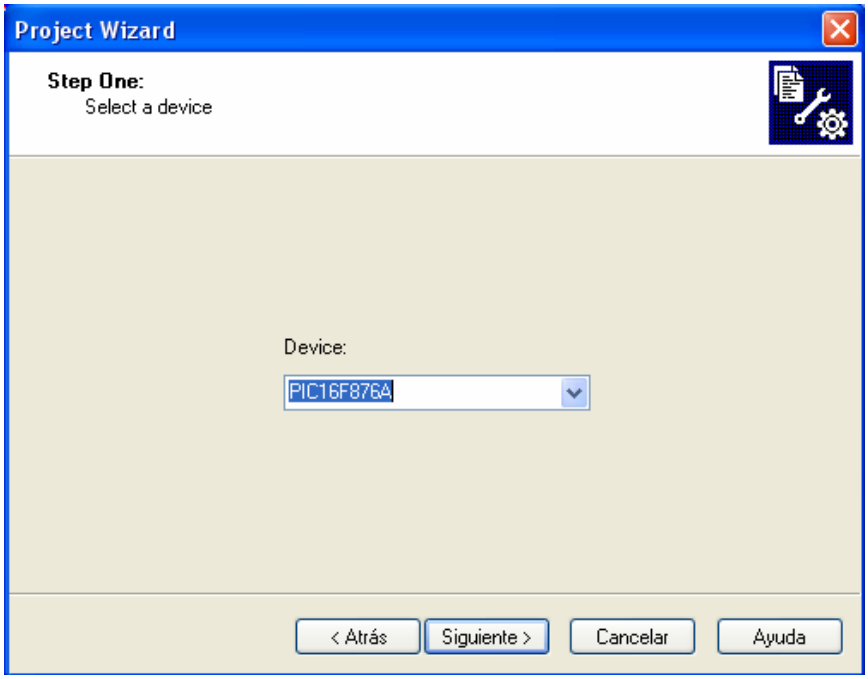

Figura 31: Selección del tipo de microcontrolador

Se indica el compilador CCS C Compiler, ya que programaremos en C.

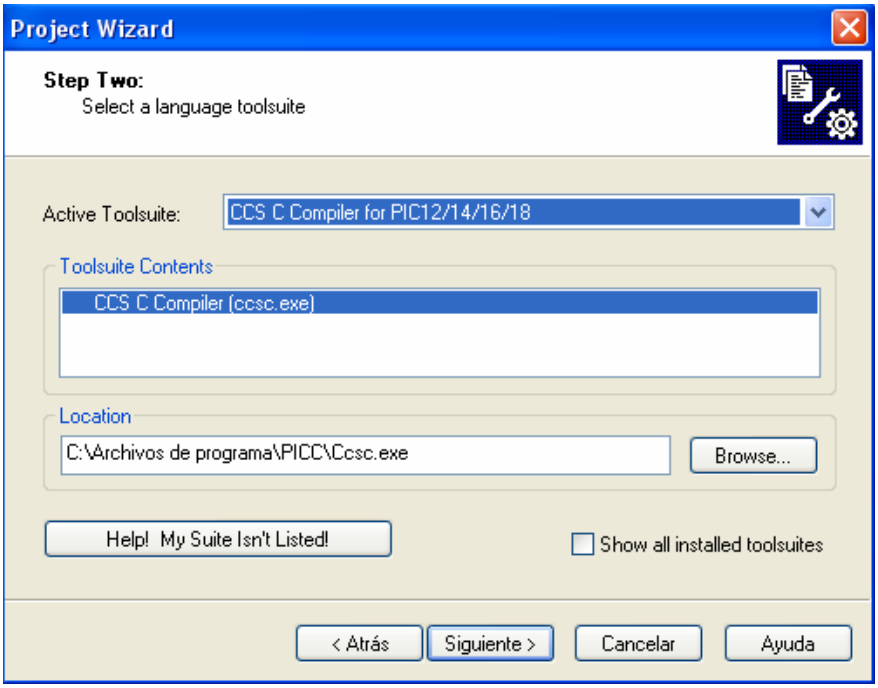

Figura 32: Elección del tipo de compilador

Se le asigna un nombre al proyecto y se indica la ubicación del mismo. Se añaden las librerías, de sensores, motores y de radiofrecuencia si fuese necesario.

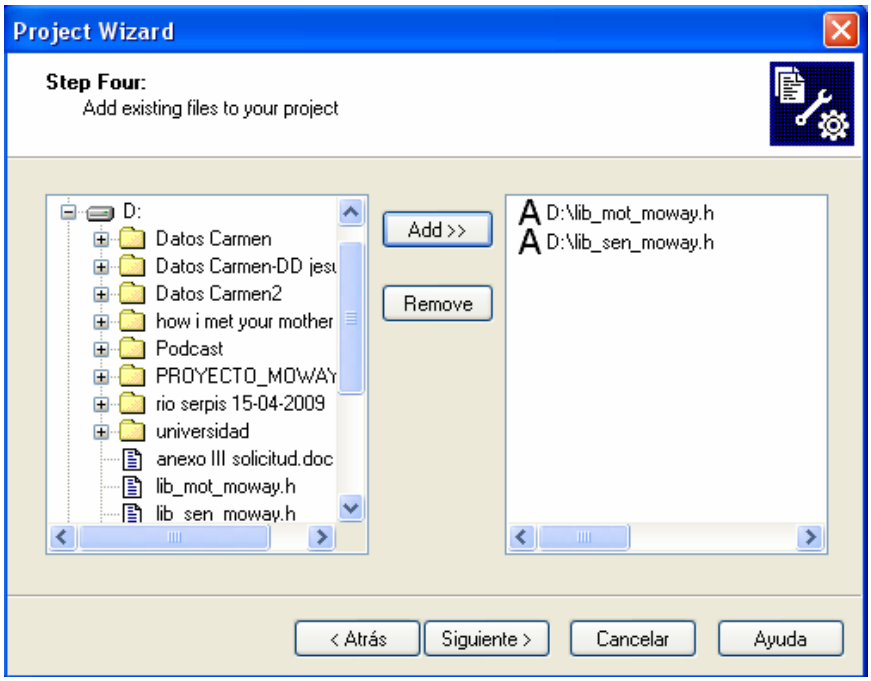

Figura 33: Se añaden las librerías

Tras aceptar, ya tendremos el proyecto creado. En una hoja en blanco generaremos el código (incluyendo librerías, tipo de reloj y modelo de PIC en la cabecera del programa) y lo guardaremos con extensión .c. Lo agregaremos como fuente al proyecto y finalmente lo compilaremos, generándose el archivo .HEX que grabaremos en Moway.

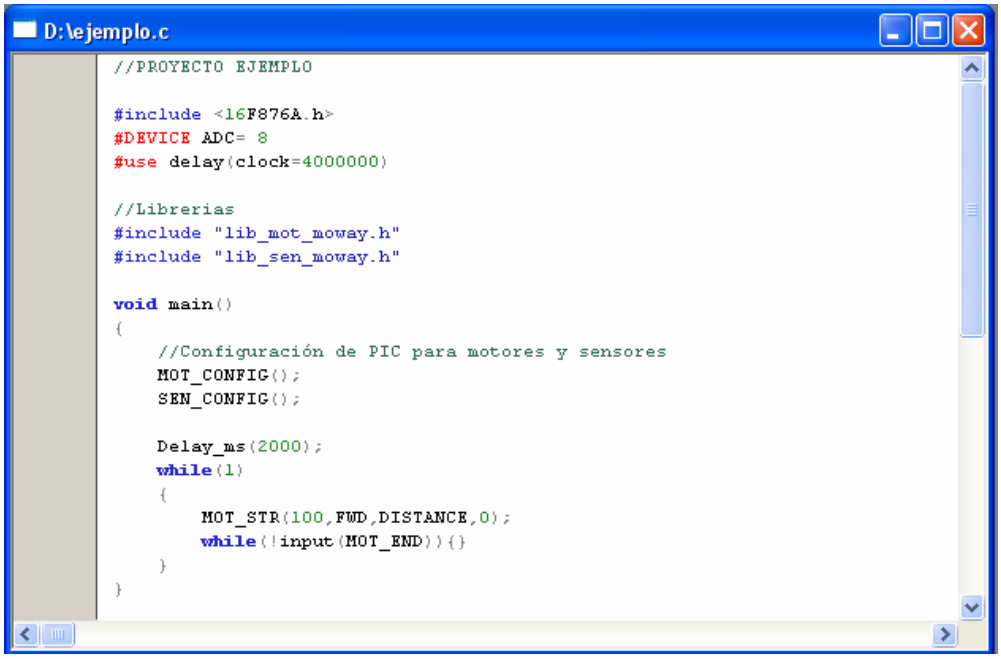

Figura 34: Código del programa ejemplo

1.3. PROGRAMAS EJEMPLO

A continuación, vamos a analizar cada una de las prácticas que están disponibles en la página oficial de Moway. Nuestro objetivo es estudiar su código y realizar algunas modificaciones para mejorar la tarea realizada por el microbot.

1.3.1. Práctica 1: El encierro

En esta primera práctica llamada “El Encierro” Moway debe permanecer dentro de un recinto cerrado descrito mediante líneas en el suelo.

Se van a manejar tanto comandos para detectar línea como comandos de movimiento.

Figura 35: El encierro

El microbot debe mantener un movimiento rectilíneo mientras no detecte línea (la cual delimita el recinto). Al detectar línea, Moway rota a la derecha para evitar salirse del recinto.

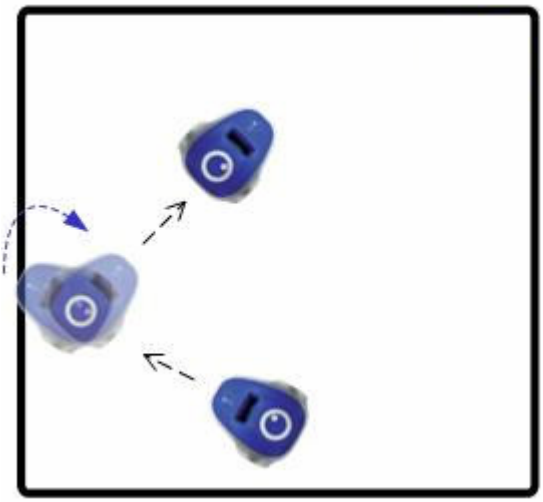

Figura 36: Reciento de reclusión

A continuación se muestra el código del programa en C:

```
//*********************************************************************
//*                                Moway_enclosed                     *
//*********************************************************************
 #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"
//*********************************[MAIN]*****************************
 void main()
 {
        //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)
        {
        //Se comprueba los sensores de línea
         SEN_LINE_DIG();
                //Si encuentra alguna línea bien sea por el sensor derecho o por el izquierdo, 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);
                }
        }
}
```

Seguidamente haremos algunas modificaciones:

- **Encender los indicadores luminosos al encontrar la línea.**

En este apartado se activa el indicador luminoso derecho en caso de detectar línea con el sensor derecho y se activa el izquierdo en caso de detectar línea con el izquierdo. Tras ello, Moway rota para evitar salirse del recinto y continúa con un movimiento rectilíneo.

```
//Si encuentra línea con el sensor derecho
 if(SEN_LINE_R==1)
{
        //Encendemos el diodo LED inferior derecho
```

```
        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)){}
        //Apagamos el diodo LED inferior derecho
        LED_R_OFF();

        //Sigue el movimiento rectilíneo
        MOT_STR(100,FWD,TIME,0);
}
//Si encuentra línea con el sensor izquierdo
if(SEN_LINE_L==1)
{
        //Enciende el diodo LED inferior izquierdo
        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)){}
        //Apaga el diodo LED inferior izquierdo
        LED_L_OFF();

        //Sigue el movimiento rectilíneo
        MOT_STR(100,FWD,TIME,0);
}
```

- **Cambiar el movimiento recto en movimiento en zig-zag.**

En vez de avanzar en línea recta, se emplea el comando MOT_CUR para desplazarse dentro del recinto cerrado con un movimiento de zigzagueante.

```
//Se da un valor inicial a la variable que representa la dirección del movimiento
RL=0;
while(1)
{
        //En cada ciclo de programa se cambia la dirección del movimiento para producir el zig-zag
        RL=!RL;
        //Movimiento curvo en el que los motores giran al 10% y 90% de velocidad durante 0.2
        //segundos
        MOT_CUR(50,FWD,40,RL,TIME,2);
        //El movimiento continúa hasta que se acaba el tiempo
        while(!input(MOT_END)){}

        //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)){}

        }
```

- **Cambiar las líneas por paredes y utilizar los sensores de obstáculos.**

Moway se mantiene realizando un movimiento rectilíneo. Al detectar obstáculo con cualquiera de los sensores de obstáculos, rota a la derecha.

```
// Primer movimiento recto hacia adelante al 100% de velocidad por tiempo ilimitado
MOT_STR(100,FWD,TIME,0);

while(1)
{
        //Se comprueba los sensores de obstáculo
         SEN_OBS_DIG();

        //Si encuentra algún obstáculo 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. Práctica 2: Rastreador

En esta segunda práctica llamada “Rastreador”, Moway sigue fielmente una línea dibujada en el suelo.

Figura 37: Rastreador

El robot tiene que, con la ayuda de la entrada de los sensores de línea, detectar línea y encaminar su movimiento a recorrer dicha línea. La estrategia es la de seguir el borde izquierdo de la línea (mantener la línea a la derecha) encendiendo los LED oportunos en función de que sensores estén detectando línea.

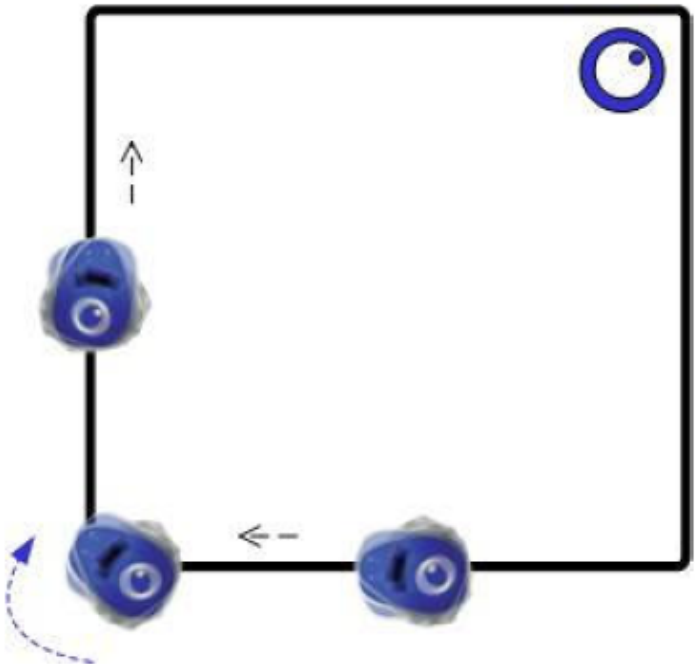

Figura 38: Comportamiento a seguir en el rastreador

A continuación se muestra el código del programa en C:

```
//************************************************************************
//*                        Moway_line_track                              *
//************************************************************************
#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()
{
        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 superior
        LED_TOP_RED_ON_OFF();
        LED_TOP_GREEN_ON_OFF();

        // Primer movimiento para encontrar la línea (movimiento rotatorio a la derecha)
        MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);

        while(1)
        {
```

```
		//Se comprueba los sensores de línea
		SEN_LINE_DIG();

		//Si detecta borde izquierdo de la línea, sigue recto encendiendo el LED frontal derecho
		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_OFF();
			LED_L_OFF();
			MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);
		}

		//Si está encima de la línea, gira a la izquierda encendiendo los dos LEDs frontales
		else if(SEN_LINE_R==1 && SEN_LINE_L==1)
		{
			LED_R_ON();
			LED_L_ON();
			MOT_ROT(50,FWD,CENTER,LEFT,TIME,0);
		}

		//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);
		}
	}
}
```

- **Modificar los movimientos según el recorrido.**

Para una velocidad baja de rotación se producen movimientos suaves en los tramos rectos. Al comando MOT_ROT se le introduce un valor más pequeño en su primer parámetro, correspondiente a la velocidad del movimiento de rotación.

```
	// Primer movimiento para encontrar la línea
	MOT_ROT(50,FWD,CENTER,RIGHT,TIME,0);

	while(1)
	{
		//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(10,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(10,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(10,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(10,FWD,CENTER,LEFT,TIME,0);
                }
        }
}
```

1.3.3. Práctica 3: Faro

En esta tercera práctica, llamada “Faro”, Moway se dirige a una fuente de luz.

Figura 39: Faro

El robot toma decisiones según el valor del sensor de luz. La estrategia se basa en cabecear hasta que el valor del sensor de luz sea mayor que una consigna.

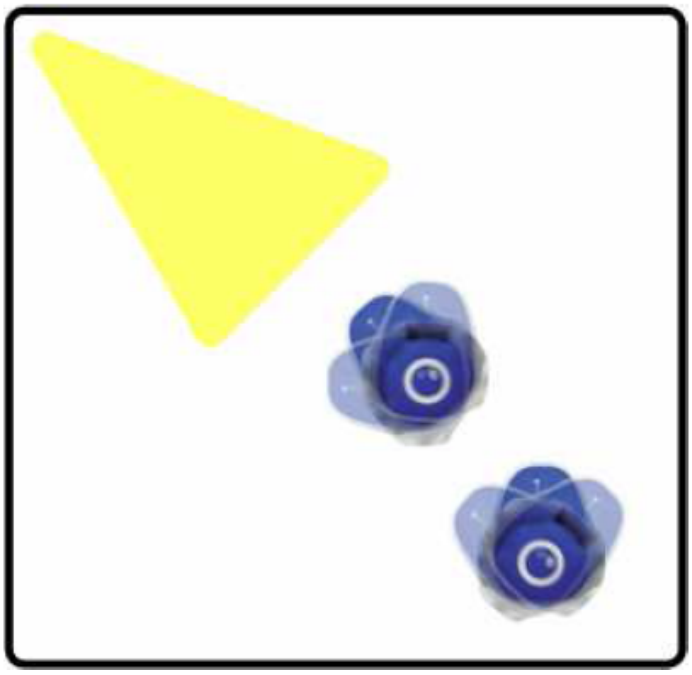

Figura 40: Comportamiento a seguir en el Faro

El programa, que se incluye a continuación, se comporta de forma que Moway realiza un movimiento rotatorio en torno a si mismo, mientras el sensor de luz no detecte un valor superior al establecido mediante la variable "lightValue".

Una vez que el sensor de luz detecta un valor superior al que se desea, Moway se mueve sin salirse de la zona iluminada por el foco (delimitada según el valor del parámetro lightValue). Cuanto mayor sea el valor de lightValue, más reducida será el área por la cual Moway se desplaza una vez encontrado el foco luminoso.

```
//*************************************************************************
//*                        Moway_light_track                              *
//*************************************************************************
#include <16F876A.h>
#DEVICE ADC= 8
#use delay(clock=4000000)

//***************************[LIBRERIAS MOWAY]*****************************
#include "lib_mot_moway.h"
#include "lib_sen_moway.h"

const  int8    lightValue=90;
static int8    rotAngle;
static int1    rotDir;

void main()
{
        Delay_ms(2000);

        //Configuración de PIC para motores y sensores
        MOT_CONFIG();
        SEN_CONFIG();

        while(1)
```

```
        {
                rotDir=!rotDir;
                //Aumenta el ángulo si no se detecta luz para ampliar el campo de búsqueda
                rotAngle+=20;
                if(rotAngle>100)
                rotAngle=20;

                //Rotación hacia la izquierda o derecha comprobando si encuentra 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
                        if(SEN_LIGHT_P>lightValue)
                        {
                                //Movimiento rectilineo durante 0.1 segundos
                                MOT_STR(50,FWD,TIME,1);
                                while(!input(MOT_END)){}
                                rotAngle=0;
                        }
                }
        }
}
```

• Modificar el programa para que Moway recorra un área determinada en busca de la luz.

Se definen una serie de variables necesarias, en la cabecera del programa.

```
//Declaración de variables
const  int8    lightValue=90;
static int8    distanciaRect=20;
static int8    rotAngle;
static int1    rotDir;
static int8    aux=1;
static int1    aux2=1;
static int8    aux3=1;
static int8    i;
```

El área es recorrida en forma de cuadrados hasta que el sensor de luz no detecte un valor superior a la variable lightValue. Una vez que ha entrado en la zona iluminada, actúa exactamente igual que en el programa anterior, evitando salirse de dicha zona.

```
while(1)
{
        //Recorre la zona formando cuadrados mientras que el sensor no detecta un valor superior a
        //lightValue
        while(aux2==1)
        {
                //Chequeo del estado del sensor de luz
                SEN_LIGHT();
                if(aux>3)
                {
                        aux3=aux3+1;
                        aux=1;
```

```
                }
                for(i=0;i<aux3;i++)
                {
                        MOT_STR(50,FWD,0,distanciaRect);
                        while(!input(MOT_END))
                        {
                                //Chequeo del estado del sensor de luz
                                SEN_LIGHT();
                                if(SEN_LIGHT_P>lightValue)
                                        aux2=0;
                        }
                }
                aux=aux+1;
                MOT_ROT(100,FWD,1,RIGHT,0,25);
                while(!input(MOT_END))
                {
                        //Chequeo del estado del sensor de luz
                        SEN_LIGHT();
                        if(SEN_LIGHT_P>lightValue)
                                aux2=0;
                }
        }
        //Una vez que ya se ha introducido en la zona en torno a la fuente luminosa en la que el valor del
        //sensor es superior a lightValue que actúe exactamente igual al ejercicio inicial
        rotDir=!rotDir;
        //Aumenta el ángulo si no se detecta luz para ampliar el campo de búsqueda
        rotAngle+=20;
        if(rotAngle>100)
        rotAngle=20;

        //Rotación hacia la izquierda o derecha comprobando si encuentra 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;
                }
        }
}
```

1.3.4. Práctica 4: Defender

En esta cuarta práctica llamada “Defender”, Moway expulsa a los objetos que se introduzcan en su territorio empleando para ello los sensores de línea y de obstáculos.

Figura 41: Defender

El robot tiene que tomar decisiones según el valor de los sensores de línea y los de obstáculos. Por un lado debe hacer lo mismo que en la práctica del encierro, evitar salirse del recinto. Por otro lado, cuando detecte algún objeto con los sensores de obstáculo tiene que empujarlo y sacarlo del área a defender.

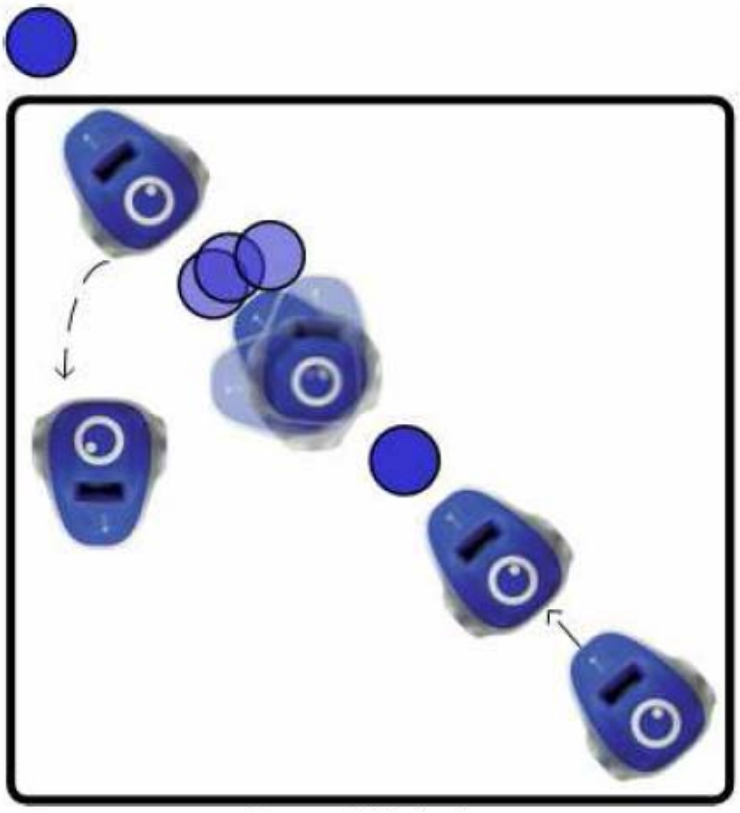

Figura 42: Comportamiento de Moway en Defender

Desde main, se llamará a las 3 funciones creadas: MOWAY_LINE, MOWAY_SEARCH y MOWAY_ATTACK.

- MOWAY_LINE: detecta si se encuentra sobre una línea. Si es así, rota para evitar salirse del recinto.
- MOWAY_SEARCH: detecta la presencia de un "intruso", es decir, un obstáculo.
- MOWAY_ATTACK: esta función se encarga de empujar fuera del recinto al objeto que se haya detectado.

```
//************************************************************************
//*                        Moway_defender                                *
//************************************************************************
#include <16F876A.h>
#DEVICE ADC= 8
#use delay(clock=4000000)

//***************************[LIBRERIAS DE MOWAY]*****************************
#include "lib_mot_moway.h"
#include "lib_sen_moway.h"

// Sensibilidad del sensor de obstáculos. Diferente en cada robot Moway.
// El usuario deberá ajustarlos según la sensibilidad deseada.
#define senObsMin 50

//***********************FUNCION  DETECCION DE OBSTACULOS**********************
//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  DETECCIÓN DE LÍNEA***********************
//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)){}
        }
}
//*****************************FUNCION DE ATAQUE*******************************
// 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();
        }
}
//*****************************FUNCION PRINCIPAL MAIN*****************************
void main()
{
        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();
                //En caso contrario, continúa en línea recta por tiempo indefinido
                else
                        MOT_STR(60,FWD,TIME,0);
        }
}
```

1.4. LIBRERÍAS EMPLEADAS

Existen dos librerías, disponibles en C, que facilitan la gestión de los sensores y el módulo motor del robot Moway, llamadas LIB_SEN_MOWAY.H y LIB_MOT_MOWAY.H respectivamente. Además existe otra librería, de radiofrecuencia (llamada lib_rf2gh4.h), que se encarga de gestionar el envío de información entre el PC y un microbot y entre varios microbots por radiofrecuencia.

En este apartado únicamente comentaremos las dos primeras librerías, de motores y sensores, porque en los programas desarrollados generalmente no emplearemos radiofrecuencia. A continuación se detallan aspectos importantes a destacar en cada una de ellas.

1.4.1. Librería sensores Moway: "LIB_SEN_MOWAY.H"

Esta librería en CCS puede ser incluida en cualquier proyecto de Moway, permitiendo al usuario controlar los sensores de una forma sencilla.

El conjunto de las funciones de la librería ocupan unas 470 palabras de memoria de programa y 5 bytes de memoria de datos.

Es importante saber que cada llamada a cualquier función de la librería utiliza un nivel adicional de la pila de llamadas. Esto es, antes de llamar a una de estas funciones debe de haber al menos dos niveles libres de la pila de llamadas para que no haya errores.

Descripción

La librería contiene una serie de funciones encargadas de leer los datos que proporcionan los sensores del robot. Ellas son las encargadas de configurar los puertos de entrada y salida adecuadamente, el ADC del microcontrolador y los indicadores luminosos.

Variables

SEN_LIGHT_P

En esta variable se guarda el porcentaje de la luz incidente en el sensor de luz. Se actualiza cada vez que se llama a la función SEN_LIGHT().

SEN_LINE_L

En esta variable se guarda el valor del sensor de línea izquierdo, siendo este dato digital o analógico dependiendo de la función que se encarga de actualizarla: SEN_LINE_DIG() y SEN_LINE_ANALOG().

SEN_LINE_R

En esta variable se guarda el valor del sensor de línea derecho, siendo este dato digital o analógico dependiendo de la función que se encarga de actualizarla: SEN_LINE_DIG() y SEN_LINE_ANALOG().

SEN_OBS_L

En esta variable se guarda el valor del sensor de obstáculo izquierdo, siendo este dato digital o analógico dependiendo de la función que se encarga de actualizarla: SEN_OBS_DIG() y SEN_OBS_ANALOG().

SEN_OBS_R

En esta variable se guarda el valor del sensor de obstáculo derecho, siendo este dato digital o analógico dependiendo de la función que se encarga de actualizarla: SEN_OBS_DIG() y SEN_OBS_ANALOG().

Funciones

En la librería lib_sen_moway existen una serie de funciones que están orientadas al control de los sensores y de los diodos LED de Moway.

A continuación se realiza una breve descripción de cada una de ellas.

Nombre	Variable entrada	Variable salida	Descripción
void SEN_CONFIGURAR()	-	-	Configura para utilizar los sensores
void SEN_LIGHT()	-	SEN_LIGHT_P	Lee el valor del sensor de luz
int1 SEN_OBS_DIG()	-	SEN_OBS_R SEN_OBS_L	Detecta la presencia de obstáculos
int1 SEN_OBS_ANALOG()	-	SEN_OBS_R SEN_OBS_L	Detecta distancia a obstáculos
void SEN_LINE_DIG()	-	SEN_LINE_R SEN_LINE_L	Detecta zona oscura (línea negra)
void SEN_LINE_ANALOG()	-	SEN_LINE_R SEN_LINE_L	Detecta tonalidades en la superficie
void LED_R_ON()	-	-	Encendido del diodo LED inferior derecho
void LED_L_ON()	-	-	Encendido del diodo LED inferior izquierdo
void LED_TOP_RED_ON()	-	-	Encendido del diodo LED superior rojo
void LED_TOP_GREEN_ON()	-	-	Encendido del diodo LED superior verde
void LED_R_OFF()	-	-	Apagado del diodo LED inferior derecho
void LED_L_OFF()	-	-	Apagado del diodo LED inferior izquierdo
void LED_TOP_RED_OFF()	-	-	Apagado del diodo LED superior rojo
void LED_TOP_GREEN_OFF()	-	-	Apagado del diodo LED superior verde
void LED_R_ON_OFF()	-	-	Parpadeo del diodo LED inferior derecho
void LED_L_ON_OFF()	-	-	Parpadeo del diodo LED inferior izquierdo
void LED_TOP_RED_ON_OFF()	-	-	Parpadeo del diodo LED superior rojo
void LED_TOP_GREEN_ON_OFF()	-	-	Parpadeo del diodo LED superior verde

Tabla 10: Funciones librería de sensores

***void* SEN_CONFIG()**

Esta función configura las entradas y salidas para poder manejar los sensores e inicializa las variables.

***void* SEN_LIGHT()**

Captura el valor analógico dependiente de la luz incidente en el fototransistor.

***int1* SEN_OBS_DIG()**

Indica si un obstáculo se encuentra en la parte delantera derecha o en la parte delantera izquierda.

***int1* SEN_OBS_ANALOG()**

Esta función indica si un obstáculo se encuentra en la parte delantera derecha o en la parte delantera izquierda y la distancia al robot.

***void* SEN_LINE_DIG()**

Indica si los sensores están sobre una superficie oscura o no

***void* SEN_LINE_ANALOG()**

Indica la cantidad de luz que se ha reflejado en los optoacopladores.

***void* LED_R_ON()**

Enciende el diodo LED inferior derecho.

***void* LED_L_ON()**

Enciende el diodo LED inferior izquierdo.

***void* LED_TOP_RED_ON()**

Enciende el diodo LED superior rojo.

***void* LED_TOP_GREEN_ON()**

Enciende el diodo LED superior verde.

***void* LED_R_OFF()**

Apaga el diodo LED inferior derecho.

***void* LED_L_OFF()**

Apaga el diodo LED inferior izquierdo.

***void* LED_TOP_RED_OFF()**

Apaga el diodo LED superior rojo.

***void* LED_TOP_GREEN_OFF()**

Apaga el diodo LED superior verde.

***void* LED_R_ON_OFF()**

Parpadeo del diodo LED inferior derecho.

***void* LED_L_ON_OFF()**

Parpadeo del diodo LED inferior izquierdo.

***void* LED_TOP_RED_ON_OFF()**

Parpadeo del diodo LED superior rojo.

***void* LED_TOP_GREEN_ON_OFF()**

Parpadeo del diodo LED superior verde.

1.4.2. Librería motores Moway: "LIB_MOT_MOWAY.H"

Esta librería en CCS puede ser incluida en cualquier proyecto de Moway y permite al usuario controlar el sistema motriz.

El conjunto de las funciones de la librería ocupan unas 1070 palabras de memoria de programa y 5 bytes de memoria de datos.

Es importante saber que cada llamada a cualquier función de la librería utiliza dos niveles adicionales de la pila de llamadas. Esto es, antes de llamar a una de estas funciones debe de haber al menos tres niveles libres de la pila de llamadas para que no haya errores.

Descripción

La librería contiene una serie de funciones encargadas de mandar comandos por I2C al Sistema Motriz, que será el encargado de controlar los motores dejando libre de carga de trabajo al microcontrolador principal, pudiendo éste realizar otras tareas.

La comunicación con Sistema Motriz se realiza mediante el protocolo I2C. Cualquier microcontrolador con este tipo de comunicación puede controlar los motores,

sólo basta con tomar como referencia las librerías en ensamblador. Cada una de estas tramas I2C tiene una duración de 350us.

Variables

MOT_STATUS_DATA_0-1

En estas dos variables se almacena el valor del dato requerido por la función MOT_FDBCK.

Funciones

En la librería **lib_mot_moway** existen una serie de funciones que están orientadas al control del sistema de motores de Moway.

A continuación se realiza una breve descripción de cada una de ellas.

Nombre	Entrada	Retorno	Descripción
void **MOT_CONFIG**()	-	-	Configuración para la comunicación con los motores
int8 **MOT_STR** (*int* MOT_VEL, *int1* FWDBACK, *int1* COMTYPE, *int* MOT_T_DIST)	MOT_VEL FWDBACK COMTYPE MOT_T_DIST	0: Envío correcto 1: Envío incorrecto 2: Datos incorrectos	Comando para movimiento en línea recta
int8 **MOT_CHA_VEL** (*int* MOT_VEL, *int1* FWDBACK, *int1* RL, *int1* COMTYPE, *int* MOT_T_DIST)	MOT_VEL FWDBACK RL COMTYPE MOT_T_DIST	0: Envío correcto 1: Envío incorrecto 2: Datos incorrectos	Comando para cambiar la velocidad de un motor
int8 **MOT_ROT** (*int* MOT_VEL, *int1* FWDBACK, *int* MOT_CENWHEEL, *int1* RL, *int1* COMTYPE, *int* MOT_T_ANG)	MOT_VEL FWDBACK MOT_CENWHEEL RL COMTYPE MOT_T_ANG	0: Envío correcto 1: Envío incorrecto 2: Datos incorrectos	Comando para realizar la rotación del robot
int8 **MOT_CUR** (*int* MOT_VEL, *int1* FWDBACK, *int* MOT_RAD, *int1* RL, *int1* COMTYPE, *int* MOT_T_DIST)	MOT_VEL FWDBACK MOT_RAD RL COMTYPE MOT_T_DIST	0: Envío correcto 1: Envío incorrecto 2: Datos incorrectos	Comando para realizar una curva
int1 **MOT_STOP**()	-	0: Envío correcto 1: Envío incorrecto	Comando para detener el robot
int1 **MOT_RST** *(int8* RST_COM*)*	RST_COM	0: Envío correcto 1: Envío incorrecto	Comando para resetear las variables temporales de tiempo y distancia
int1 **MOT_FDBCK** (*int8* STATUS_COM)	STATUS_COM	0: Envío correcto 1: Envío incorrecto MOT_STATUS_DATA_0 MOT_STATUS_DATA_1	Comando para ver el estado de los motores

Tabla 11: Funciones librería de motores

void **MOT_CONFIG()**

Esta función configura las entradas y salidas para que el microcontrolador pueda comunicarse con el sistema motriz.

El puerto RB5, que en la librería aparece con la etiqueta MOT_END, nos indica la finalización de un comando.

int8 **MOT_STR(***int* **MOT_VEL,** *int1* **FWDBACK,** *int1* **COMTYPE,** *int* **MOT_T_DIST)**

Comando para desplazamiento en línea recta. Es necesario especificar velocidad, sentido, tipo de comando y el tiempo o la distancia a recorrer. El tiempo tiene una resolución de 100ms y la distancia 1.7mm, y con un valor de 0 en MOT_T_DIST el comando se mantendrá hasta que no se especifique otra orden.

int8 **MOT_CHA_VEL (***int* **MOT_VEL,** *int1* **FWDBACK,** *int1* **RL,** *int1* **COMTYPE,** *int* **MOT_T_DIST)**

Comando para cambiar la velocidad a uno de los dos motores. Es necesario especificar velocidad, sentido, el motor, tipo de comando y el tiempo o la distancia a recorrer. El tiempo tiene una resolución de 100ms y la distancia 1.7mm, y con un valor de 0 en MOT_T_DIST el comando se mantendrá hasta que no se especifique otra orden.

int8 **MOT_ROT (***int* **MOT_VEL,** *int1* **FWDBACK,** *int* **MOT_CENWHEEL,** *int1* **RL,** *int1* **COMTYPE,** *int* **MOT_T_ANG)**

Comando para hacer rotar a Moway. Es necesario especificar velocidad, sentido, tipo de rotación, el motor, tipo de comando y el tiempo o el ángulo a rotar. El tiempo tiene una resolución de 100ms, y con un valor de 0 en MOT_T_ANG el comando se mantendrá hasta que no se especifique otra orden.

En cuanto al ángulo, las siguientes ecuaciones muestran cómo calcular el valor de MOT_T_ANG teniendo en cuenta el ángulo de rotación deseado.

$$MOT\_T\_ANG = round\left(\frac{Ángulo° \times 3.33}{12°}\right)$$

Ecuación 1: Rotación sobre el centro

$$MOT\_T\_ANG = round\left(\frac{Ángulo° \times 1.66}{6°}\right)$$

Ecuación 2: Rotación sobre una rueda

int8 **MOT_CUR (***int* **MOT_VEL,** i*nt1* **FWDBACK,** *int* **MOT_RAD,** *int1* **RL,** *int1* **COMTYPE,** *int* **MOT_T_DIST)**

Comando para dar una curva. Es necesario especificar velocidad, sentido, radio, dirección, tipo de comando y el tiempo o la distancia a recorrer. El radio es la velocidad que se resta o se suma a la velocidad global del robot.

Por lo tanto el radio tiene que cumplir las siguientes restricciones:

$$0 \leq MOT\_VEL - MOT\_\bar{RAD} \leq 100$$

Ecuación 3: Condición 1 MOT_RAD

$$0 \leq MOT\_VEL + MOT\_\bar{RAD} \leq 100$$

Ecuación 4: Condición 2 MOT_RAD

El tiempo tiene una resolución de 100ms y la distancia 1.7mm, y con un valor de 0 en MOT_T_ANG el comando se mantiene hasta que no se especifique otra orden. El motor cuenta la distancia recorrida por el motor que está en el exterior de la curva.

int1 **MOT_STOP()**

Comando para parar el robot.

int1 **MOT_RST (***int8* **RST_COM)**

Resetea las variables temporales internas de tiempo, distancia y cuentakilómetros del motor.

int1 **MOT_FDBCK (***int8* **STATUS_COM)**

Comando para conocer diversos parámetros del sistema motriz. Podemos consultar el tiempo transcurrido, el ángulo (sólo en el comando MOT_ROT), velocidad de los dos motores, distancia recorrida por cada motor y el cuentakilómetros desde el último comando.

Esta función actualiza dos variables dónde se guarda la información requerida. Todas las peticiones menos STATUS_KM devuelven un byte (MOT_STATUS_DATA_0) manteniendo MOT_STATUS_DATA_1 al valor 0xFF. Estas dos variables se actualizan cada vez que se manda un comando nuevo. Cuando se use STATUS_KM hay que tener en cuenta los dos bytes.

Para más información, consultar el manual de usuario de Moway y las librerías incluidas en el CD anexo.

CAPÍTULO 2

ERRORES EN LA ODOMETRÍA

CAPÍTULO 2: ERRORES EN LA ODOMETRÍA

2.1. DEFINICIÓN

Odometría proviene de la unión de las palabras griegas hodos (“viaje”) y metron (“medida”).

Un aspecto clave en el desarrollo de sistemas autónomos móviles es el conocimiento en tiempo real de la posición precisa del sistema. En ambientes naturales o aquellos que puedan interactuar de forma imprevisible con nuestro sistema, se nos plantea el problema de incapacidad para determinar la posición exacta del vehículo en cada instante.

En aplicaciones con microbots, utilizaremos la odometría para estimar tanto la distancia recorrida como la trayectoria y la posición de nuestro robot en cada momento.

En estos casos es imprescindible la utilización de sensores tanto internos como externos. Los sensores internos proporcionan de forma continua e inmediata la localización del robot sin conocimiento alguno del entorno, calculando el número de vueltas dadas por las ruedas en cada ciclo de control de movimiento. Estos sensores (internos) no se utilizan como único sensor en entornos que carecen de referencias fijas, pues las condiciones abruptas del terreno y la extensión de los recorridos a realizar hacen que sus estimaciones de posición lleven asociado un error elevado. Estos errores tienen su origen en deslizamientos de las ruedas, desajustes en el alineamiento, existencia de más de un punto de contacto entre la rueda y el suelo, baja resolución de un sensor o baja frecuencia de muestreo. De todos los errores, sólo aquellos debidos a imperfecciones en el diseño mecánico y sensorial del vehículo se mantienen constantes, pudiendo ser eliminados en el proceso de calibrado.

En lo que a los sensores externos se refiere, las técnicas de posicionamiento absoluto basan sus medidas en las relaciones del vehículo con elementos externos como son: sistemas de balizas, marcas activas o pasivas, o satélites. Cada una de estas técnicas proporciona la posición del vehículo respecto a un sistema de referencia global, y puede ser implantada utilizando una gran variedad de métodos y sensores, como cámara de vídeo, sensores de ultrasonido, láser o sistemas de posicionamiento global, como el GPS.

Con la introducción de sensores y una programación adaptada, se busca la eliminación de los errores, los cuales se dividen en sistemáticos y no sistemáticos.

Entre los errores sistemáticos destacan: diámetros de las ruedas diferentes, mal alineamiento de las ruedas, resolución discreta de encoder…entre otros.

Entre los errores no sistemáticos se encuentran: desplazamiento en suelos desnivelados, desplazamiento sobre objetos inesperados que se encuentren en el suelo, patinaje de las ruedas debido a suelos resbaladizos, sobre-aceleración, derrapes y fuerzas externas.

2.2. ESTIMACIÓN DEL ERROR

Al trabajar con Moway, notamos que no sigue fielmente la trayectoria programada, sino que se producen desviaciones y errores que se van acumulando. La cuestión que nos planteamos inmediatamente es si esa desviación es siempre la misma, si para un entorno determinado nos va a ser posible rectificar esa desviación a la hora de programar y conseguir la trayectoria exacta que se desea.

Para ello simularemos en un entorno adecuado (superficie sin desniveles, no resbaladiza, sin agentes externos que modifiquen los resultados) una serie de trayectorias muy simples, variando la velocidad y distancia del recorrido, para finalmente llegar a la conclusión de si el error que se produce es de naturaleza sistemática o no.

Es posible que al analizar el comportamiento de un microbot, descubramos ciertos defectos puntuales que son sólo característicos de ese microbot en cuestión. Para poder extrapolar resultados, trabajaremos con dos microbots, a los que hemos llamado Moway 01 y Moway 02. Inicialmente trabajaremos con Moway 02 y a continuación, repetiremos los mismos experimentos para Moway 01.

2.3. EXPERIMENTOS TRAYECTORIA RECTILINEA

Con este experimento se mide el error entre la trayectoria ideal y la real.

Consiste en realizar un movimiento rectilíneo hacia delante regresando a continuación al punto inicial. Este experimento se realiza con distintas longitudes y distintas velocidades para comprobar el comportamiento del robot al variar estos dos parámetros.

Al realizar el experimento se observa que lo que idealmente tenía que ocurrir, no ocurre, puesto que el robot se desvía y no regresa a la misma posición en la que se encontraba inicialmente. Para calcular esa desviación, se realizan 10 pruebas con idénticos parámetros, calculando así la desviación media. En el caso de que los puntos a los que se llega en esas pruebas idénticas estén lo suficientemente próximos, es decir no exista una dispersión elevada, se podrá corregir la trayectoria desde la programación, introduciendo factores correctivos en los parámetros que lo requieran. En caso contrario se deberá introducir algún tipo de elemento externo que permita corregir ese error no sistemático.

Experimento 1: Se recorren 50mm en línea recta hacia delante, para regresar al punto inicial a distintas velocidades.

RESULTADOS EXPERIMENTO 1						
	A = (0 , 50 , 0)			B = (0 , 0 , 0)		
Velocidad	X(mm)	Y(mm)	Angulo(º)	X(mm)	Y(mm)	Angulo(º)
100%	-6,0	51,0	3,7	-1,0	-0,9	2,4
60%	-6,1	47,6	4,4	-2,3	-3,2	4,7
20%	-5,2	46,8	3,7	-1,9	-2,2	4,0
Media Aritm.	-5,8	48,5	3,9	-1,7	-2,1	3,7

Tabla 12: Resultados experimento 1 para Moway 02

Se genera el siguiente código en matlab para ver más claramente los resultados.

```
% EXPERIMENTO 1 AL 100% DE VELOCIDAD
clear all

%Trayectoria IDEAL
punto_inicio_ideal=[0 0]; %(B)
punto_final_ideal=[0 50]; %(A)
% esto es la trayectoria
plotdespacio(punto_inicio_ideal, punto_final_ideal,'b');
% esto son los extremos
plot(punto_inicio_ideal(1),punto_inicio_ideal(2),'b*');
plot(punto_final_ideal(1),punto_final_ideal(2),'b*');

%Trayectoria REAL MEDIA   (A')
punto_inicio_real_media=[0 0];
punto_final_real_media=[-6 51];
% esto es la trayectoria
plotdespacio(punto_inicio_real_media, punto_final_real_media,'r');
% esto son los extremos
plot(punto_inicio_real_media(1),punto_inicio_real_media(2),'r*');
plot(punto_final_real_media(1),punto_final_real_media(2),'r*');

%Trayectoria REAL MEDIA   (A'  --->  B')
punto_inicio_real_media=[-6 51];
punto_final_real_media=[-1 -0.9];
% esto es la trayectoria
plotdespacio(punto_inicio_real_media, punto_final_real_media,'r');
% esto son los extremos
plot(punto_inicio_real_media(1),punto_inicio_real_media(2),'r*');
plot(punto_final_real_media(1),punto_final_real_media(2),'r*');

title('EXPERIMENTO 1 AL 100% DE VELOCIDAD');
xlabel('X ( mm )');
ylabel('Y ( mm )');

%Todos los extremos de las trayectorias REALES (A')
dA=[[-6  1];[-6 2]; [-7  0];[-7 1];[-4 2];[-6 1];[-5 0];[-7 1];[-5 1]; [-7 1]];
A(:,1)=dA(:,1)+punto_final_ideal(1);
A(:,2)=dA(:,2)+punto_final_ideal(2);
plot(A(:,1),A(:,2),'co');

%Todos los extremos de las trayectorias REALES (B')
dB=[[-2  0];[-1 0]; [0  0];[-2 -1];[0 0];[0 -1];[0 -2];[-2 -1];[-1 -2];[-2 -2]];
B(:,1)=dB(:,1)+punto_inicio_ideal(1);
B(:,2)=dB(:,2)+punto_inicio_ideal(2);
plot(B(:,1),B(:,2),'co');
```

Se representan de este modo la trayectoria ideal en azul, la trayectoria media real en rojo y los diferentes puntos a los que se llega y a partir de los cuales se calcula la media, en forma de círculos color cian.

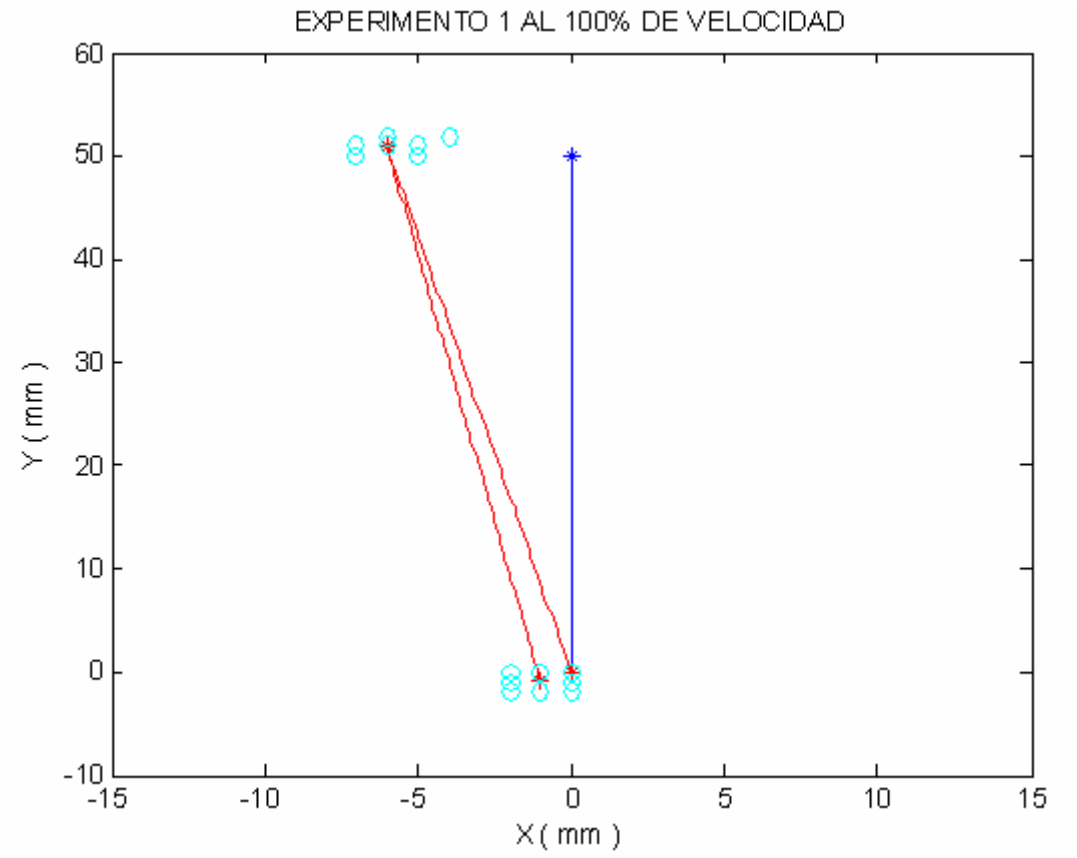

Figura 43: Experimento 1 al 100% de la velocidad para Moway 02

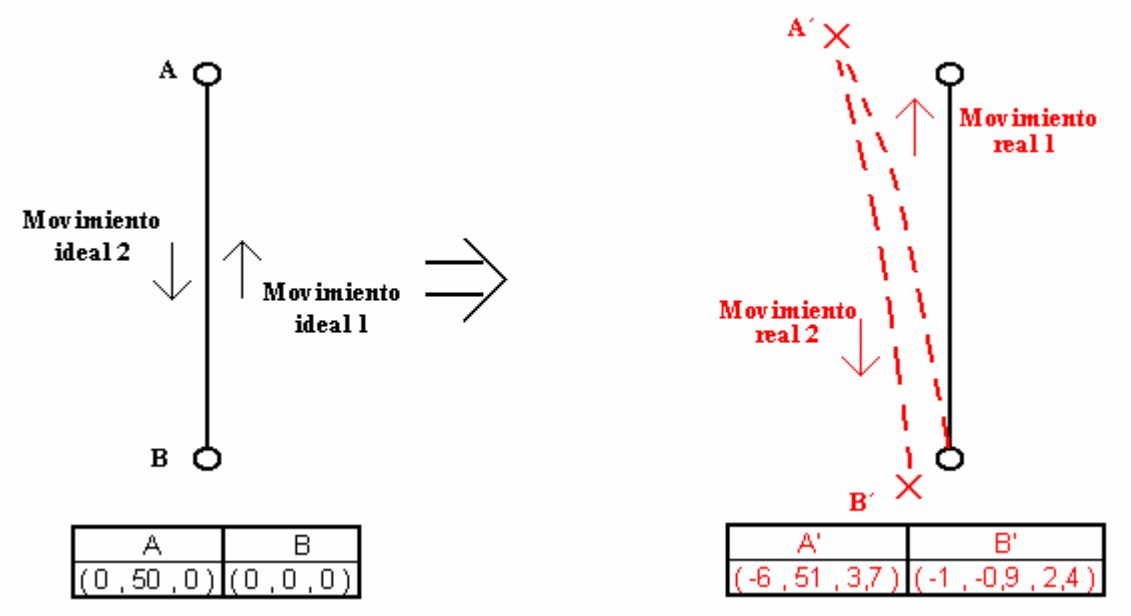

Figura 44: Experimento 1 al 100% de la velocidad para Moway 02

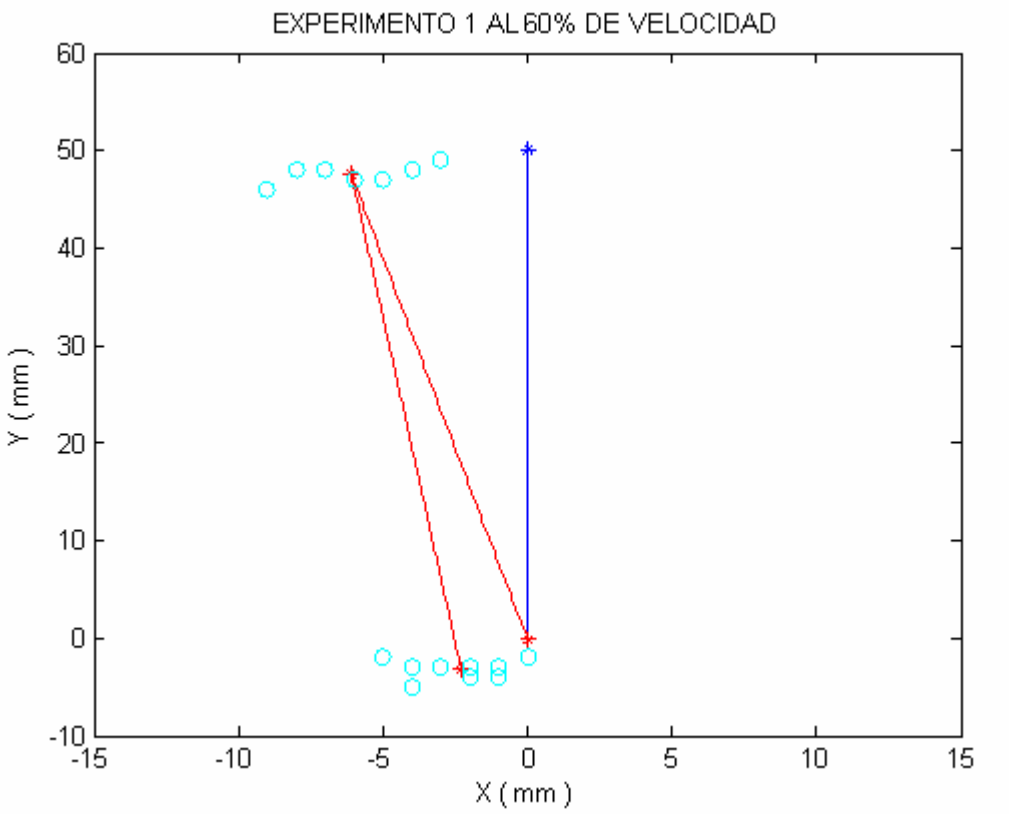

Figura 45: Experimento 1 al 60% de la velocidad para Moway 02

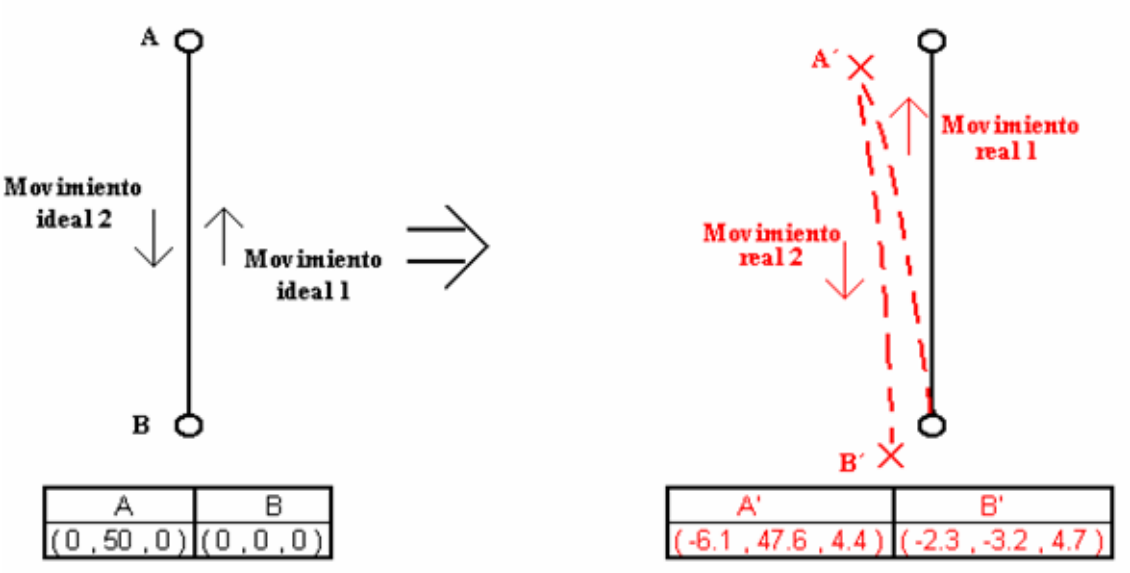

Figura 46: Experimento 1 al 60% de la velocidad para Moway 02

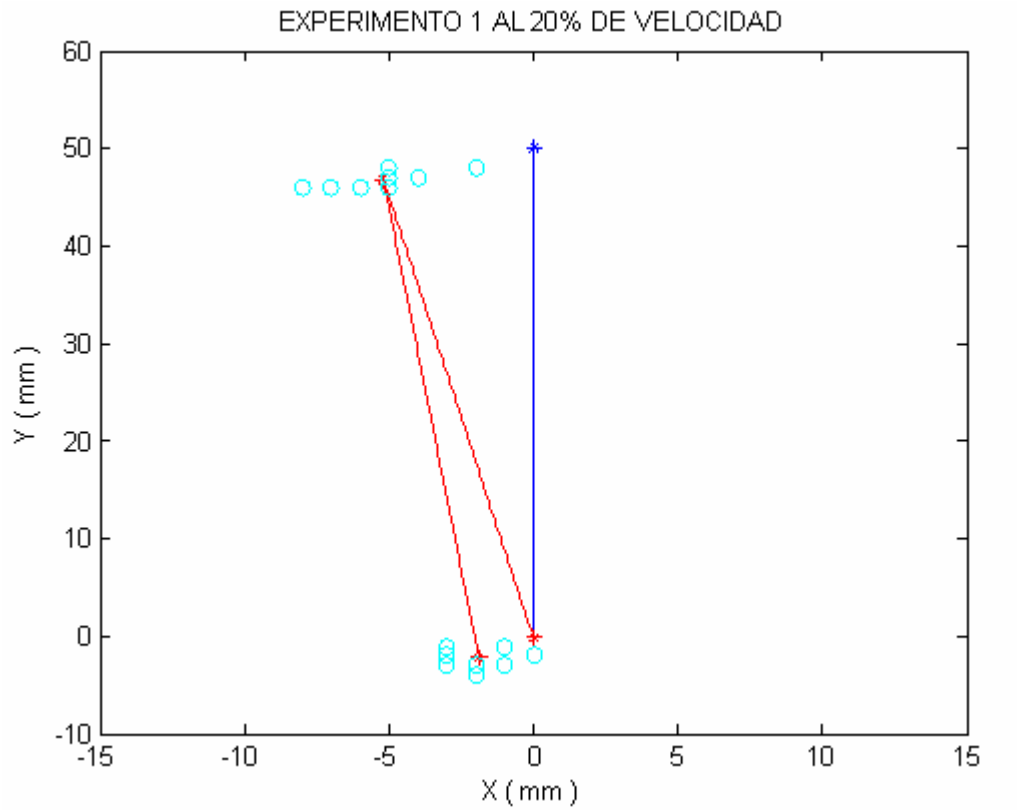

Figura 47: Experimento 1 al 20% de la velocidad para Moway 02

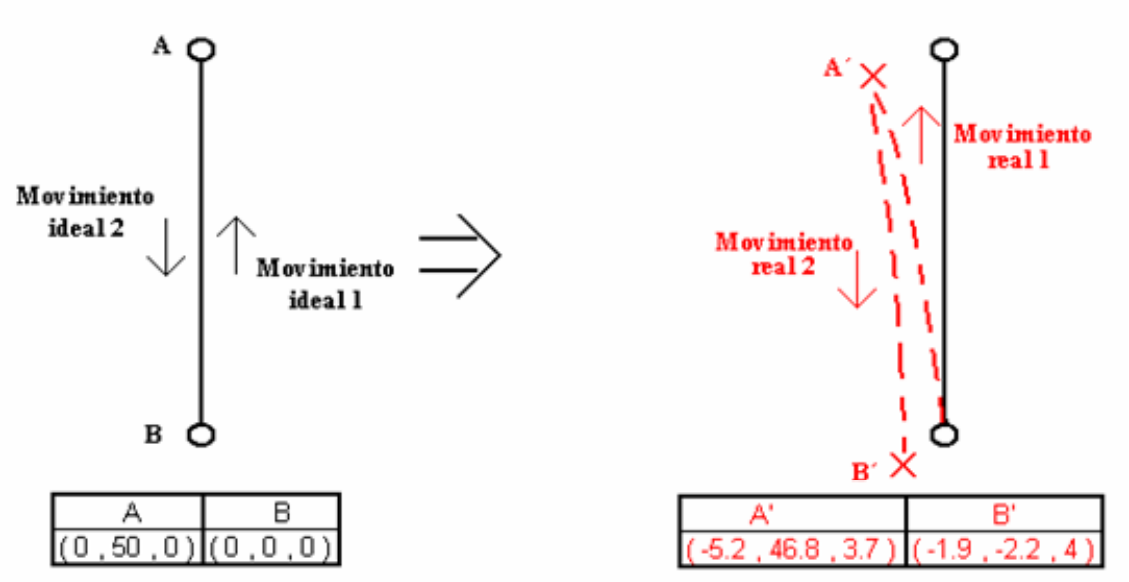

Figura 48: Experimento 1 al 20% de la velocidad para Moway 02

Para las velocidades estudiadas obtenemos errores considerables y con una alta dispersión, por lo que la media calculada en los tres casos no recoge el comportamiento del robot. No podemos obtener una relación directa entre error cometido y la velocidad a la cual se realizan los movimientos.

Experimento 2: Se recorren 100mm en línea recta hacia delante, para regresar al punto inicial a distintas velocidades.

RESULTADOS EXPERIMENTO 2						
	A = (0 , 100 , 0)			B = (0 , 0 , 0)		
Velocidad	X(mm)	Y(mm)	Angulo(º)	X(mm)	Y(mm)	Angulo(º)
100%	-13,7	96,8	7,5	-1,5	-1,7	3,6
60%	-9,7	96,1	4,7	-0,7	-3,1	4,0
20%	-8,7	95,8	4,1	0,0	-2,6	4,2
Media Aritm.	-10,7	96,2	5,4	-0,7	-2,5	3,9

Tabla 13: Resultados experimento 2 para Moway 02

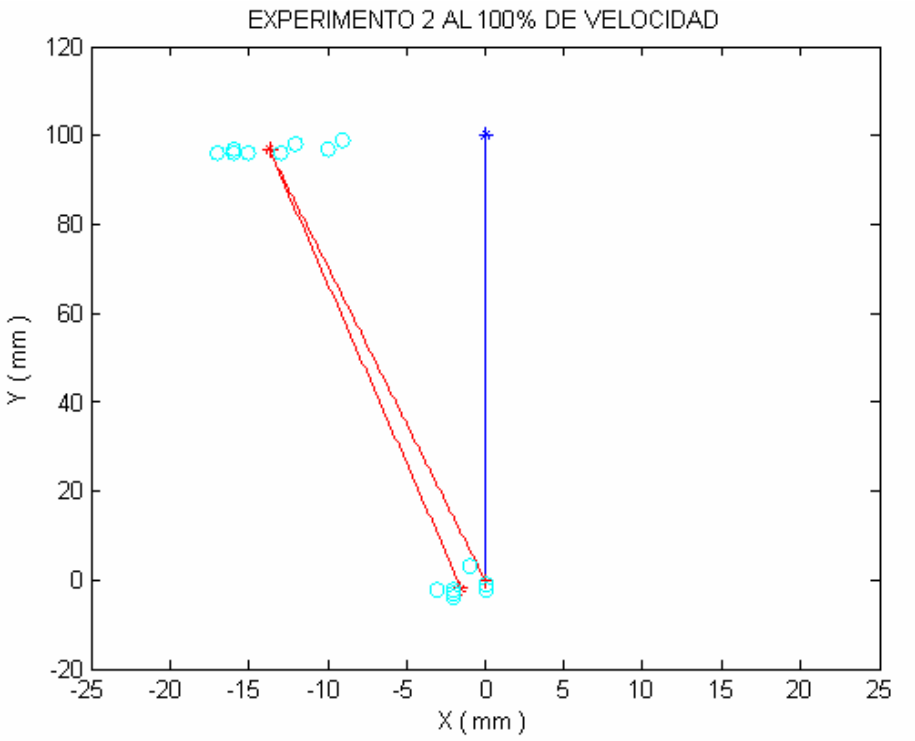

Figura 49: Experimento 2 al 100% de la velocidad para Moway 02

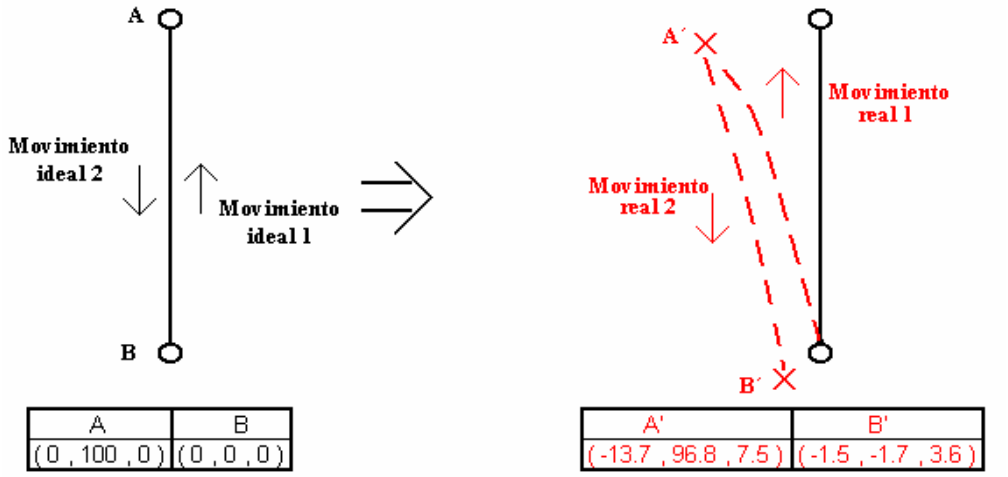

Figura 50: Experimento 2 al 100% de la velocidad para Moway 02

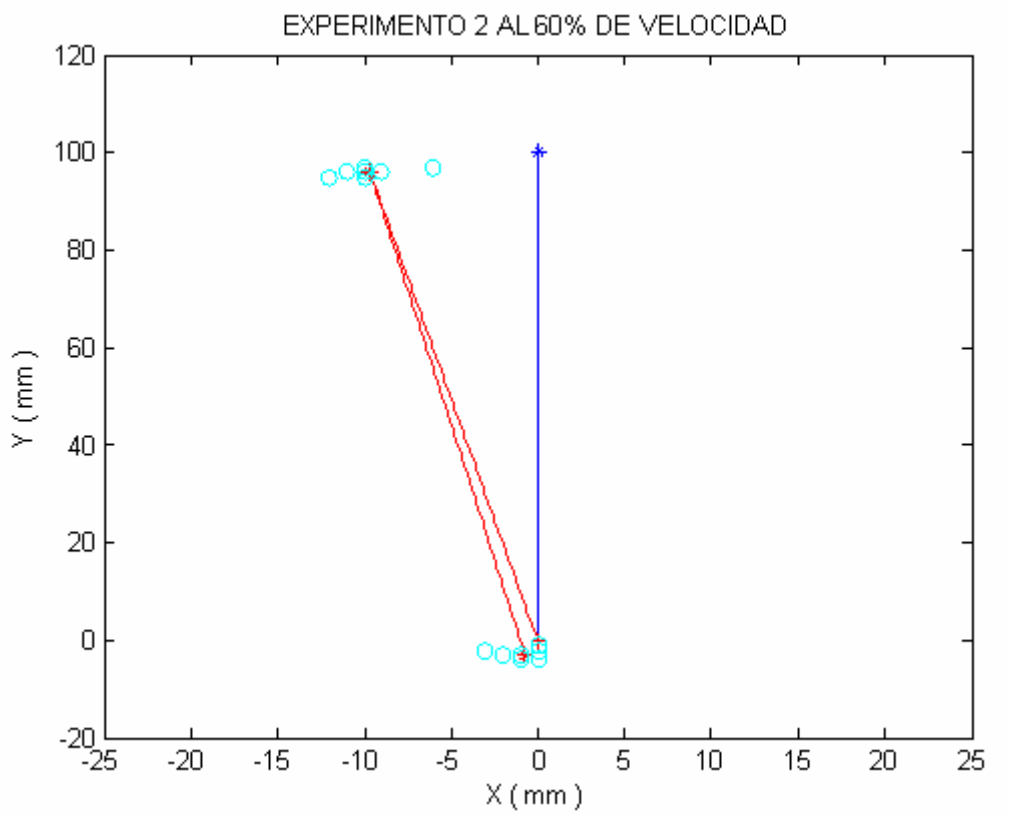

Figura 51: Experimento 2 al 60% de la velocidad para Moway 02

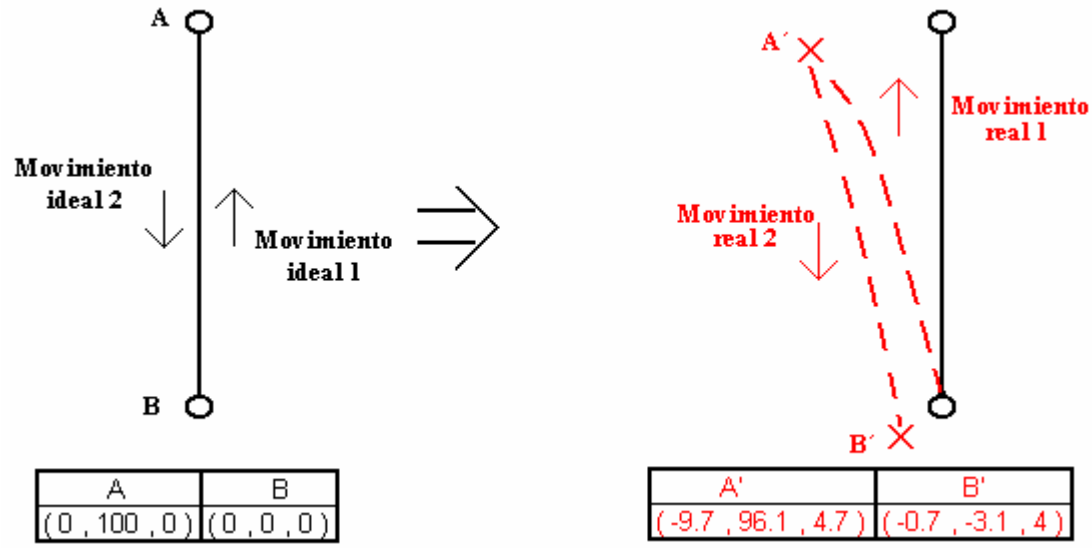

Figura 52: Experimento 2 al 60% de la velocidad para Moway 02

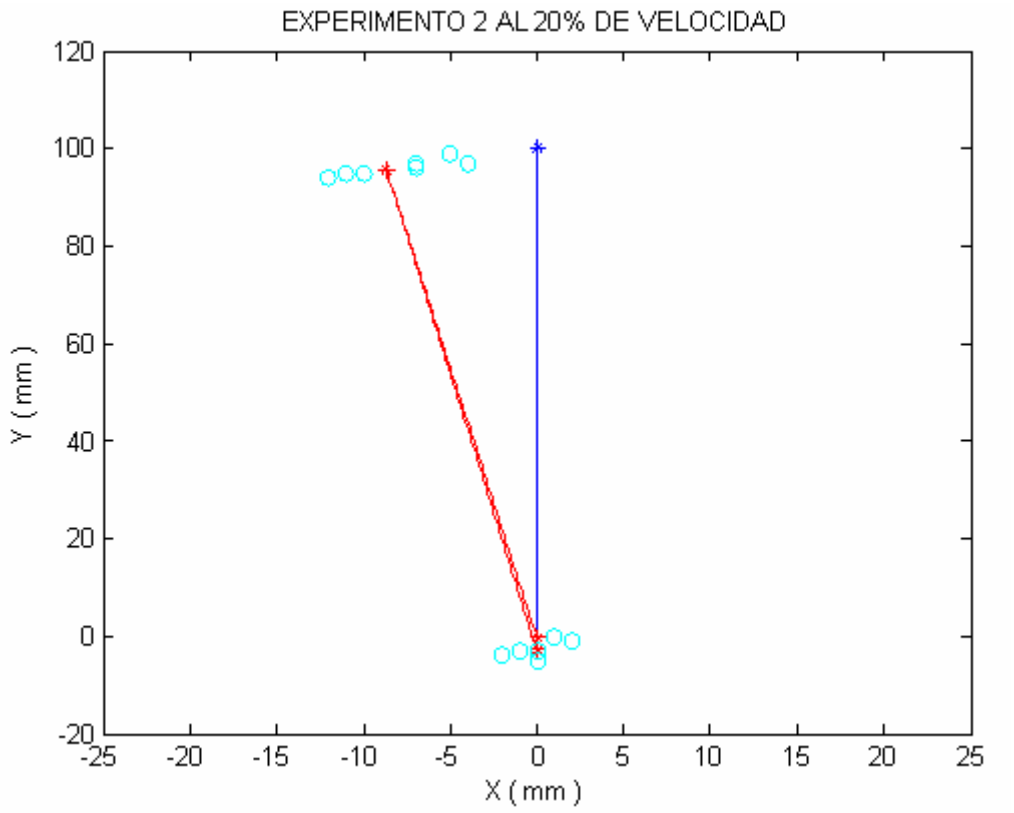

Figura 53: Experimento 2 al 20% de la velocidad para Moway 02

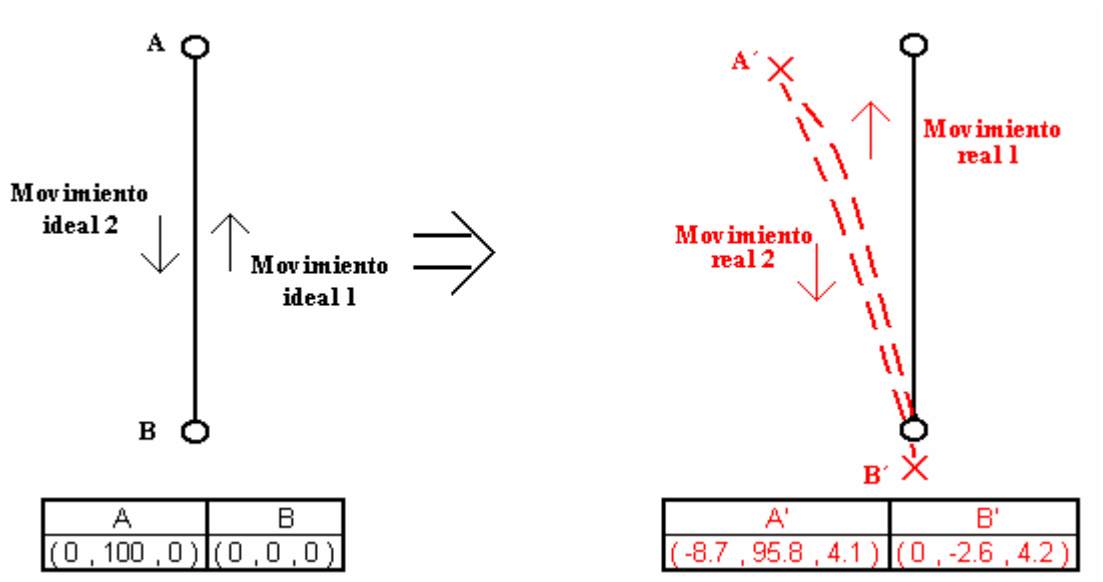

Figura 54: Experimento 2 al 20% de la velocidad para Moway 02

Para las velocidades estudiadas obtenemos, igual que en el caso anterior, errores considerables y con una alta dispersión. No podemos obtener una relación directa entre el error cometido y la velocidad a la cual se efectúan los movimientos.

Experimento 3: Se recorren 150mm en línea recta hacia delante, para regresar al punto inicial a distintas velocidades.

RESULTADOS EXPERIMENTO 3						
	A = (0 , 150 , 0)			B = (0 , 0 , 0)		
Velocidad	X(mm)	Y(mm)	Angulo(º)	X(mm)	Y(mm)	Angulo(º)
100%	-22,5	146,6	9,0	-0,6	-3,1	3,7
60%	-15,7	146,3	5,5	-0,3	-2,7	2,2
20%	-14,0	146,9	4,6	1,1	-3,0	3,8
Media Aritm.	-17,4	146,6	6,4	0,1	-2,9	3,2

Tabla 14: Resultados experimento 3 para Moway 02

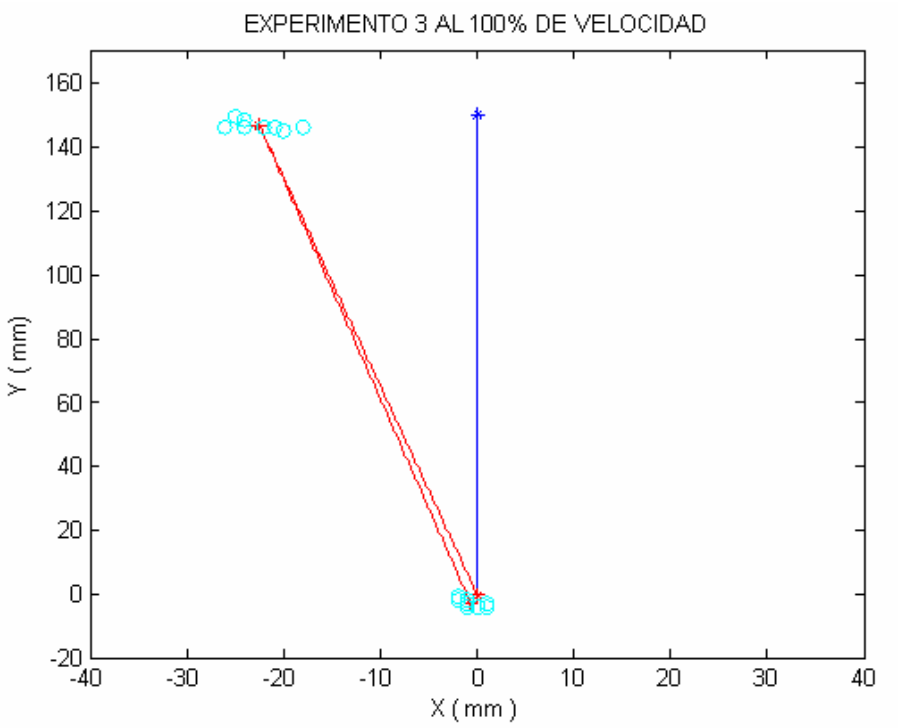

Figura 55: Experimento 3 al 100% de la velocidad para Moway 02

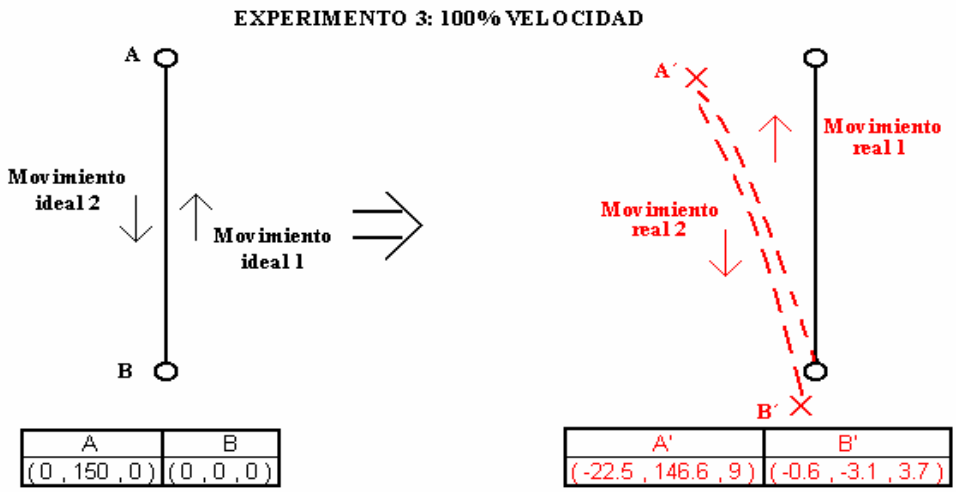

Figura 56: Experimento 3 al 100% de la velocidad para Moway 02

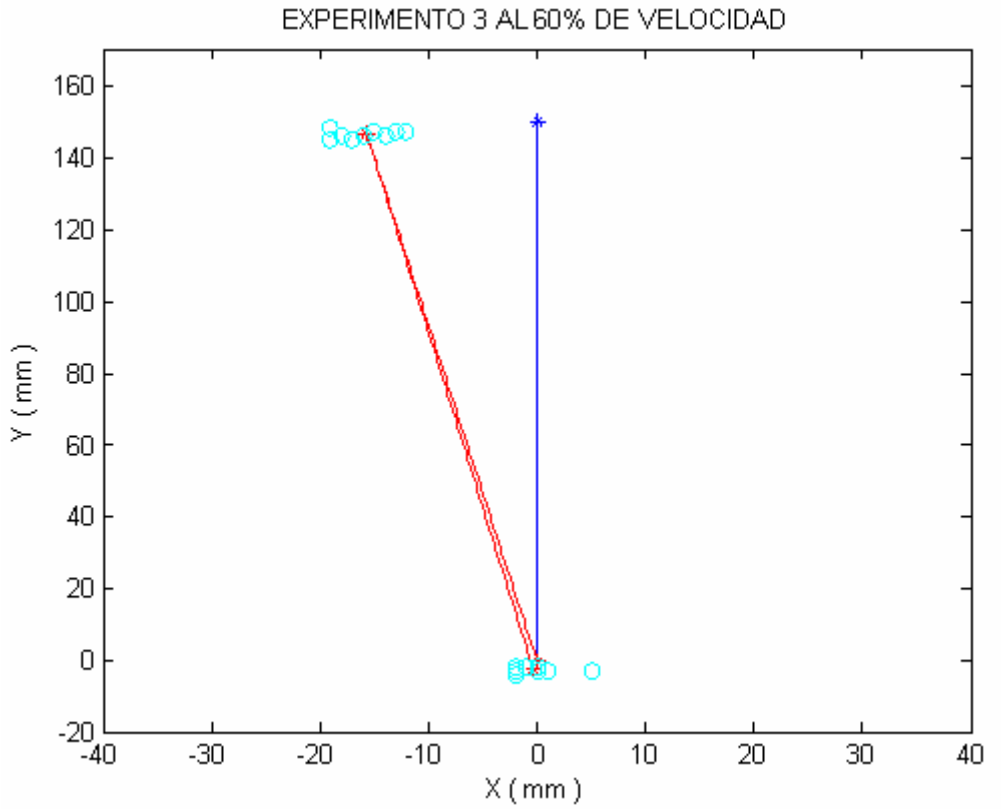

Figura 57: Experimento 3 al 60% de la velocidad para Moway 02

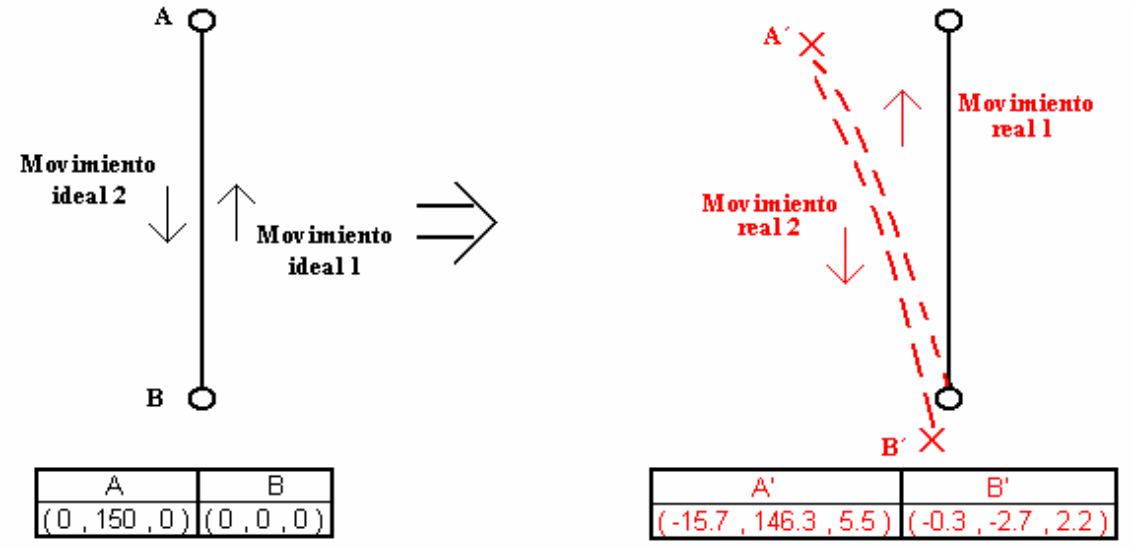

Figura 58: Experimento 3 al 60% de la velocidad para Moway 02

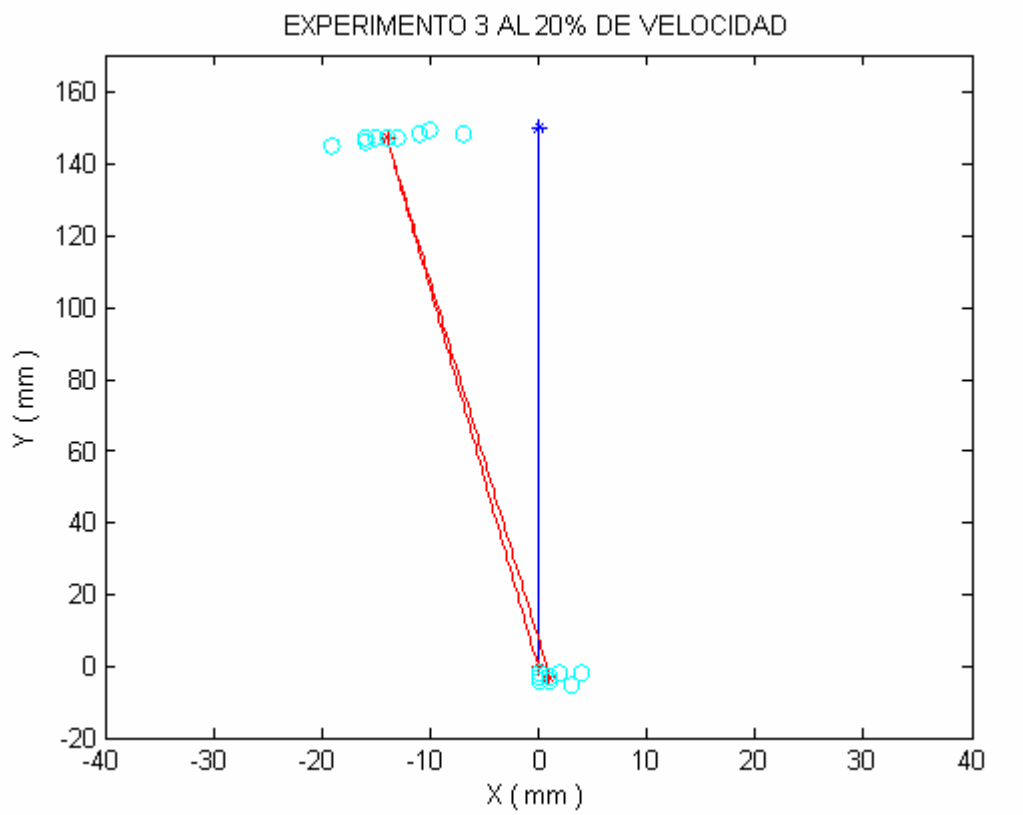

Figura 59: Experimento 3 al 20% de la velocidad para Moway 02

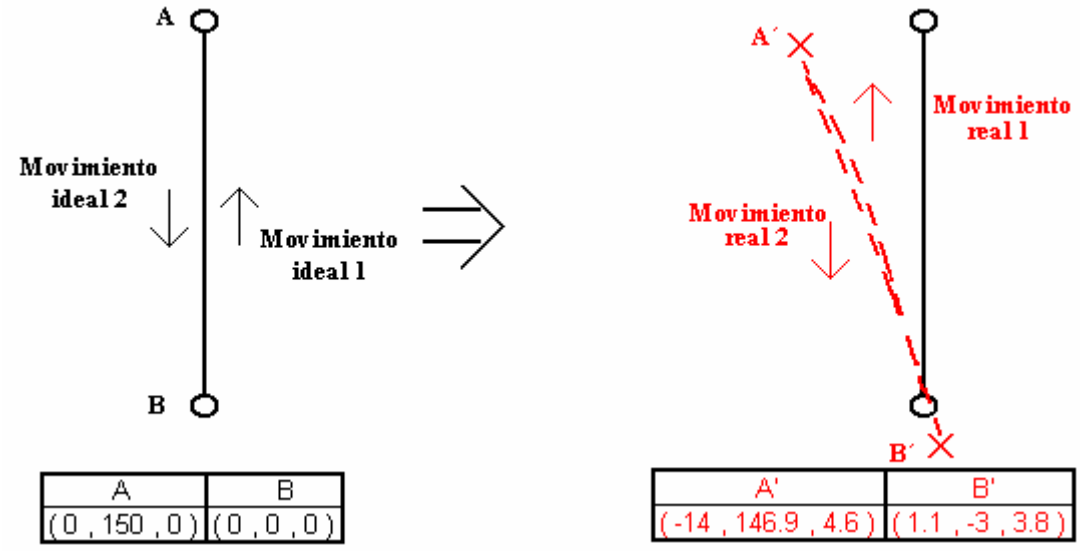

Figura 60: Experimento 3 al 20% de la velocidad para Moway 02

En este caso tenemos dispersiones menores. Tenemos valores en el eje Y que se asemejan a lo ideal, pero con grandes variaciones en el eje X. Seguimos sin obtener una relación clara entre velocidad y error.

Quizá se pueda llegar a pensar que tal vez para una velocidad determinada si exista una relación clara entre distancia recorrida y error cometido. Para ello se van a expresar los errores obtenidos por unidad de longitud para cada una de las velocidades.

ERROR POR UNIDAD DE LONGITUD PARA 100% DE VELOCIDAD						
	A = (0 , long , 0)			B = (0 , 0 , 0)		
Longitud(mm)	X(mm)/long	Y(mm)/long	Angulo(º)/long	X(mm)/long	Y(mm)/long	Angulo(º)/long
50	-0,120	1,020	0,074	-0,020	-0,018	0,048
100	-0,137	0,968	0,075	-0,015	-0,017	0,036
150	-0,150	0,977	0,060	-0,004	-0,021	0,025
Media Aritm.	-0,136	0,988	0,070	-0,013	-0,019	0,036

Tabla 15: Error por unidad de longitud al 100% de la velocidad para Moway 02

ERROR POR UNIDAD DE LONGITUD PARA 60% DE VELOCIDAD						
	A = (0 , long , 0)			B = (0 , 0 , 0)		
Longitud(mm)	X(mm)/long	Y(mm)/long	Angulo(º)/long	X(mm)/long	Y(mm)/long	Angulo(º)/long
50	-0,122	0,952	0,088	-0,046	-0,064	0,094
100	-0,097	0,961	0,047	-0,007	-0,031	0,040
150	-0,105	0,975	0,037	-0,002	-0,018	0,015
Media Aritm.	-0,108	0,963	0,057	-0,018	-0,038	0,050

Tabla 16: Error por unidad de longitud al 60% de la velocidad para Moway 02

ERROR POR UNIDAD DE LONGITUD PARA 20% DE VELOCIDAD						
	A = (0 , long , 0)			B = (0 , 0 , 0)		
Longitud(mm)	X(mm)/long	Y(mm)/long	Angulo(º)/long	X(mm)/long	Y(mm)/long	Angulo(º)/long
50	-0,104	0,936	0,074	-0,038	-0,044	0,080
100	-0,087	0,958	0,041	0,000	-0,026	0,042
150	-0,093	0,979	0,031	0,007	-0,020	0,025
Media Aritm.	-0,095	0,958	0,049	-0,010	-0,030	0,049

Tabla 17: Error por unidad de longitud al 20% de la velocidad para Moway 02

A partir de estos datos se puede determinar que no existe ninguna relación clara entre distancia recorrida y el error cometido para ninguna de las velocidades. Por lo tanto, el error que se produce en un movimiento rectilíneo se puede considerar aleatorio (no sistemático).

A través de los sucesivos experimentos, se observa la tendencia del robot a desviar su trayectoria a la izquierda en su movimiento rectilíneo hacia delante. Además esta tendencia se incrementa a medida que aumenta la velocidad de su movimiento, generando una mayor diferencia respecto al eje x (no respecto al eje y) de la trayectoria real respecto a la esperada. Se conoce cual es la tendencia, pero no es posible cuantificar ese error, debido en parte a esa elevada dispersión. Tal vez se debería comprobar, si eligiendo otro robot Moway y realizando con él las mismas pruebas en el mismo entorno, se puede cuantificar ese error (error sistemático).

A continuación se van a realizar exactamente las mismas pruebas con Moway 01.

RESULTADOS EXPERIMENTO 1						
	A = (0 , 50 , 0)			B = (0 , 0 , 0)		
Velocidad	Desv x(mm)	Desv y(mm)	Angulo(º)	Desv x(mm)	Desv y(mm)	Angulo(º)
100%	0,1	2,8	0,3	0,0	0,6	0,9
60%	-1,9	-0,3	1,1	-1,0	-0,2	2,1
20%	0,1	-3,3	0,1	0,7	-1,9	-1,2
Media Aritm.	-0,6	-0,3	0,5	-0,1	-0,5	0,6

Tabla 18: Resultados experimento 1 para Moway 01

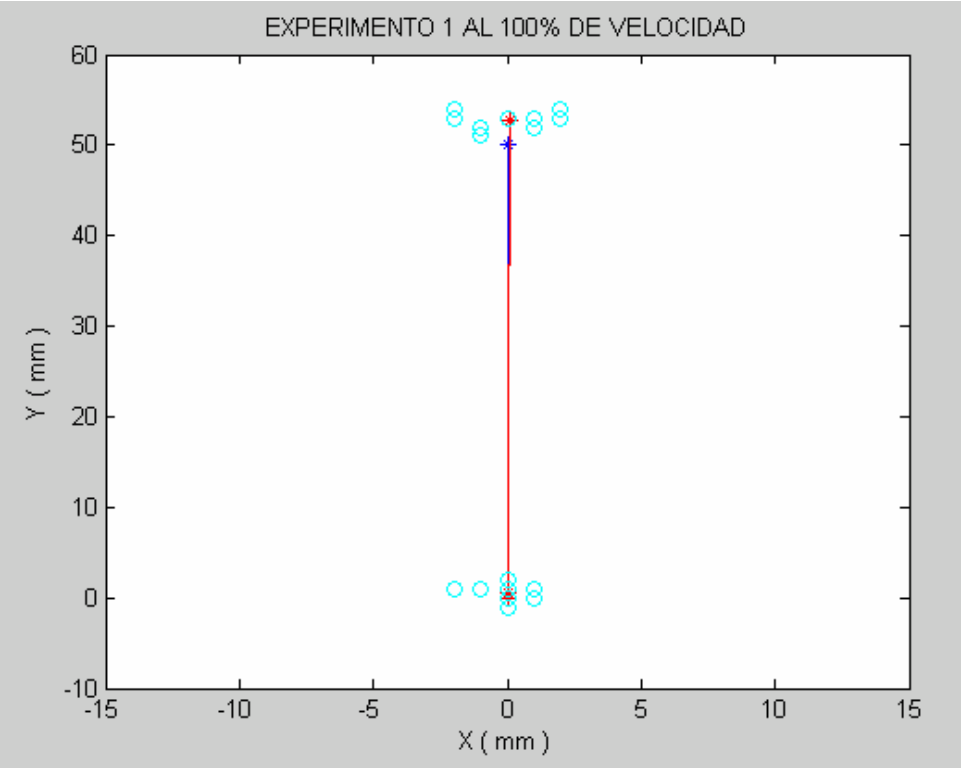

Figura 61: Experimento 1 al 100% de la velocidad para Moway 01

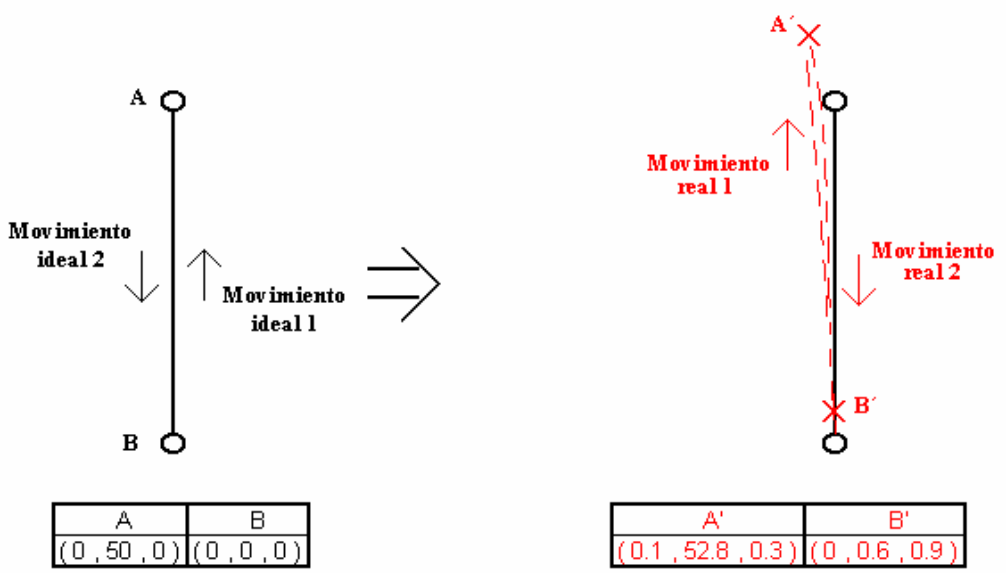

Figura 62: Experimento 1 al 100% de la velocidad para Moway 01

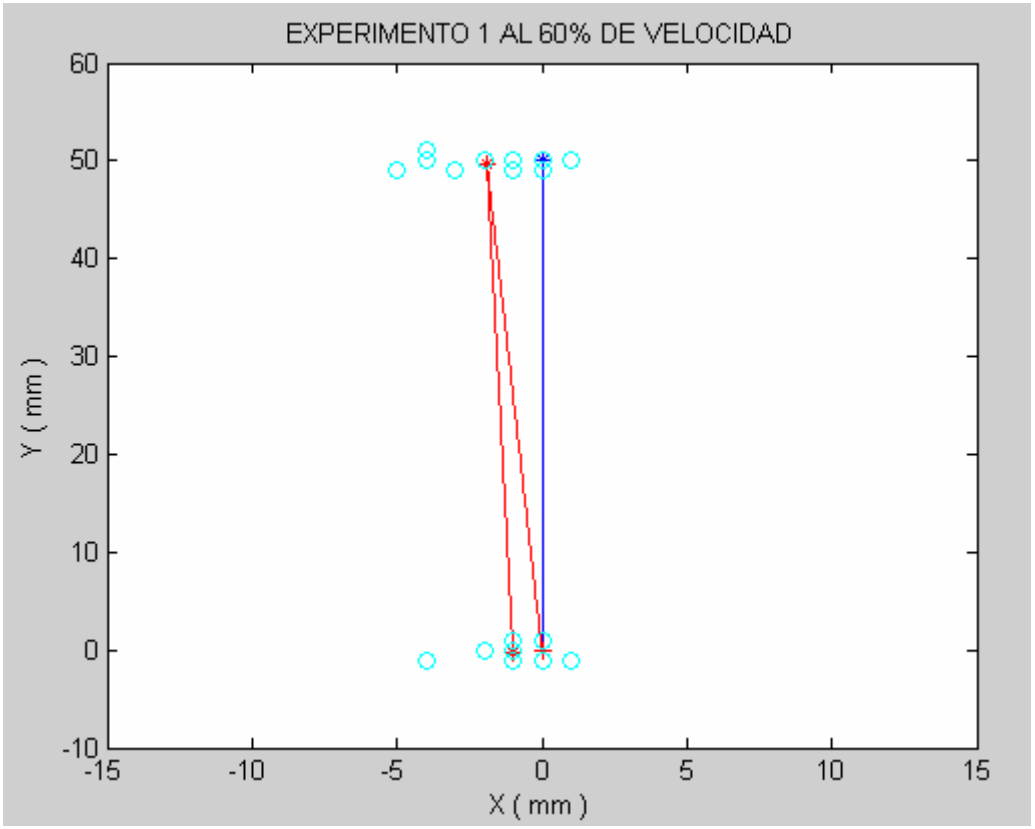

Figura 63: Experimento 1 al 60% de la velocidad para Moway 01

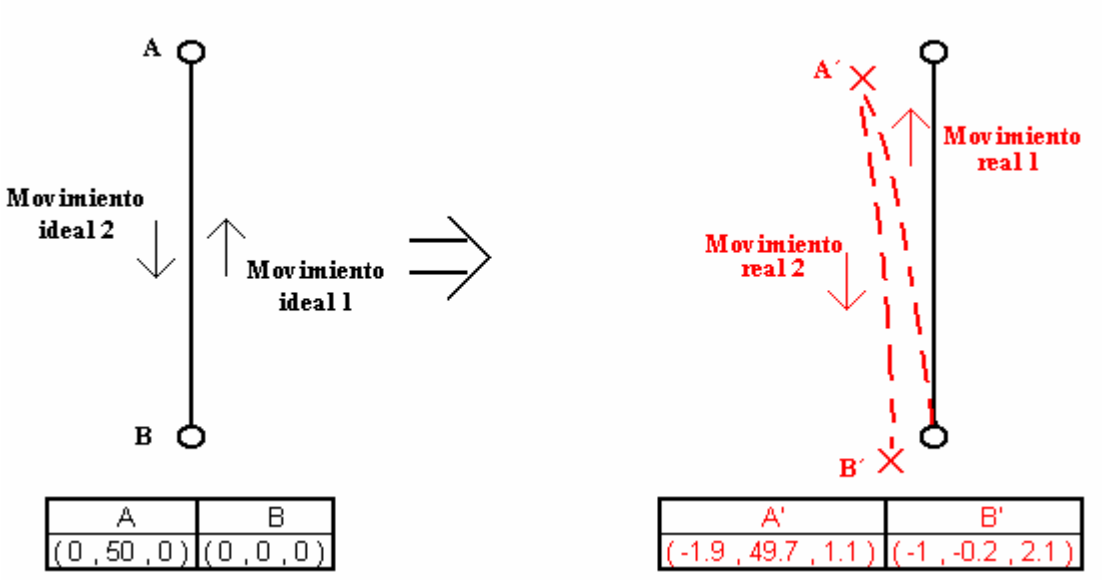

Figura 64: Experimento 1 al 60% de la velocidad para Moway 01

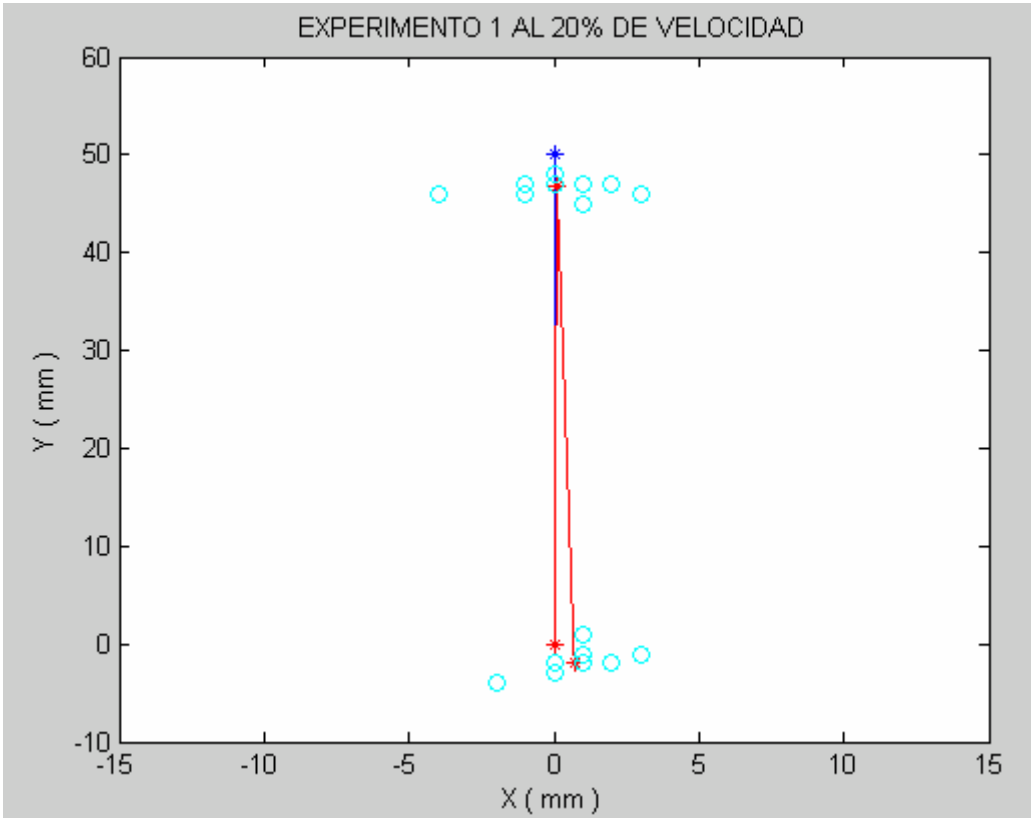

Figura 65: Experimento 1 al 20% de la velocidad para Moway 01

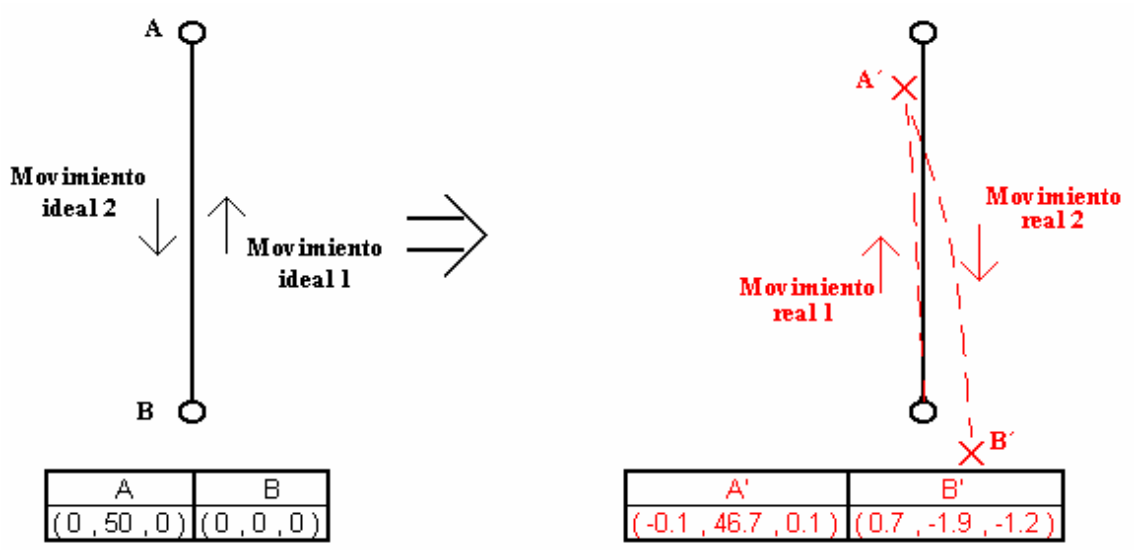

Figura 66: Experimento 1 al 20% de la velocidad para Moway 01

RESULTADOS EXPERIMENTO 2						
	A = (0 , 100 , 0)			B = (0 , 0 , 0)		
Velocidad	Desv x(mm)	Desv y(mm)	Angulo(º)	Desv x(mm)	Desv y(mm)	Angulo(º)
100%	-3,3	1,2	1,8	1,0	-0,3	2,8
60%	-3,2	-2,0	1,9	1,8	-2,4	-0,5
20%	-3,6	-2,7	2,0	1,3	-1,4	1,1
Media Aritm.	-3,4	-1,2	1,9	1,4	-1,4	1,1

Tabla 19: Resultados experimento 2 para Moway 01

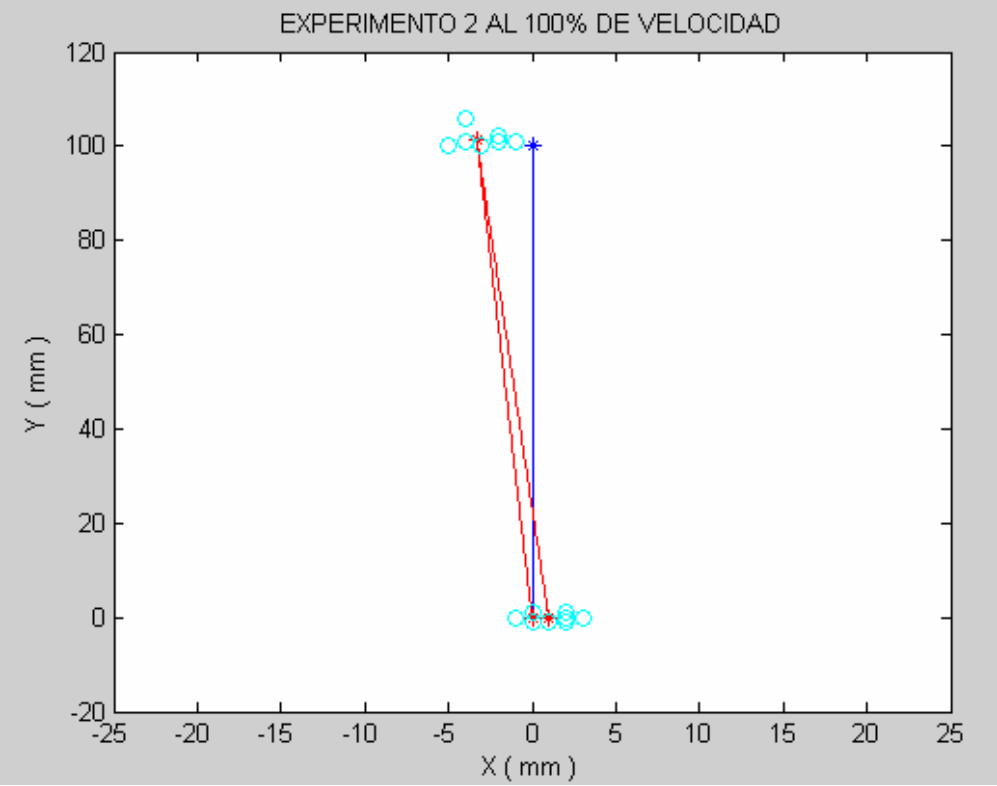

Figura 67: Experimento 2 al 100% de la velocidad para Moway 01

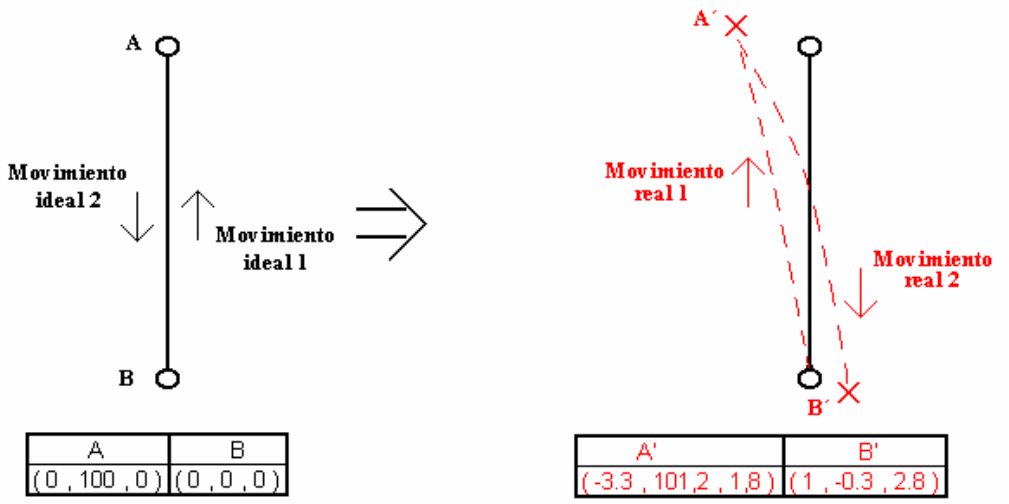

Figura 68: Experimento 2 al 100% de la velocidad para Moway 01

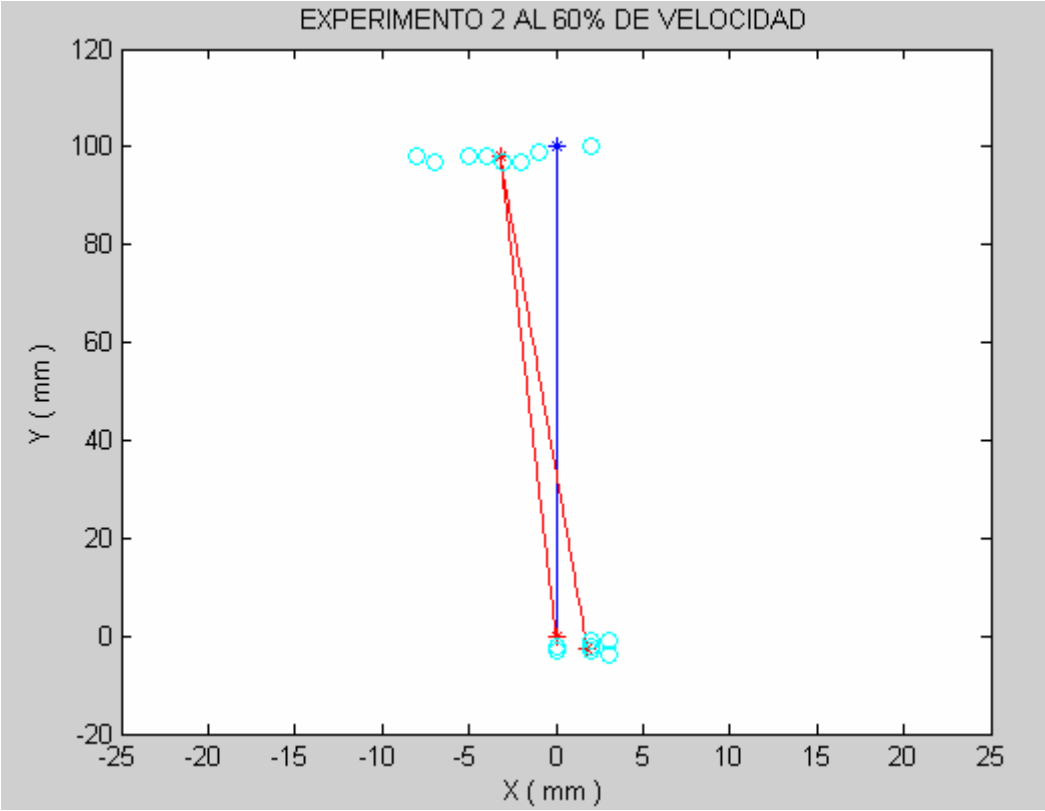

Figura 69: Experimento 2 al 60% de la velocidad para Moway 01

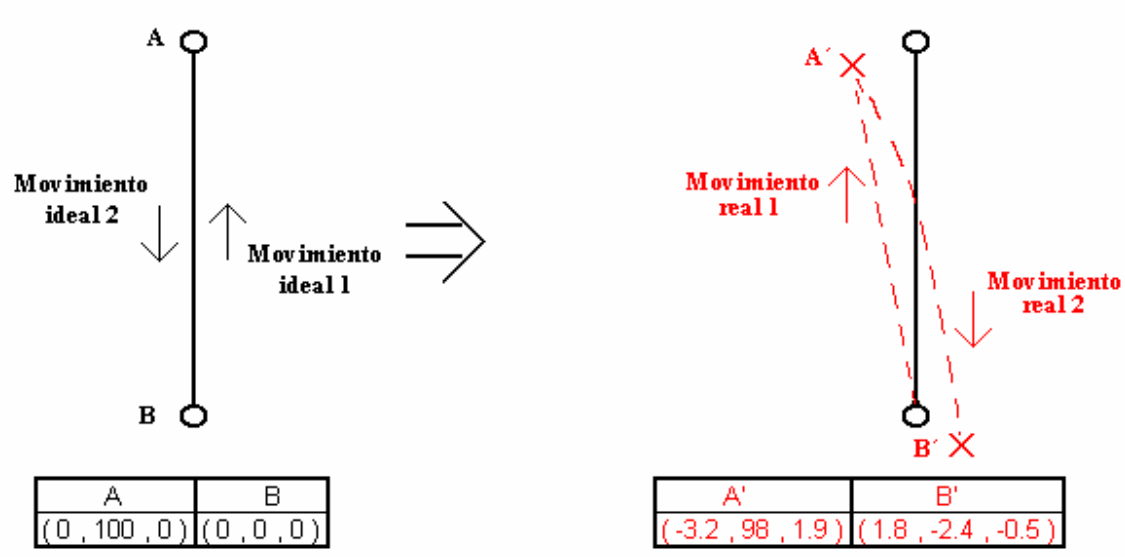

Figura 70: Experimento 2 al 60% de la velocidad para Moway 01

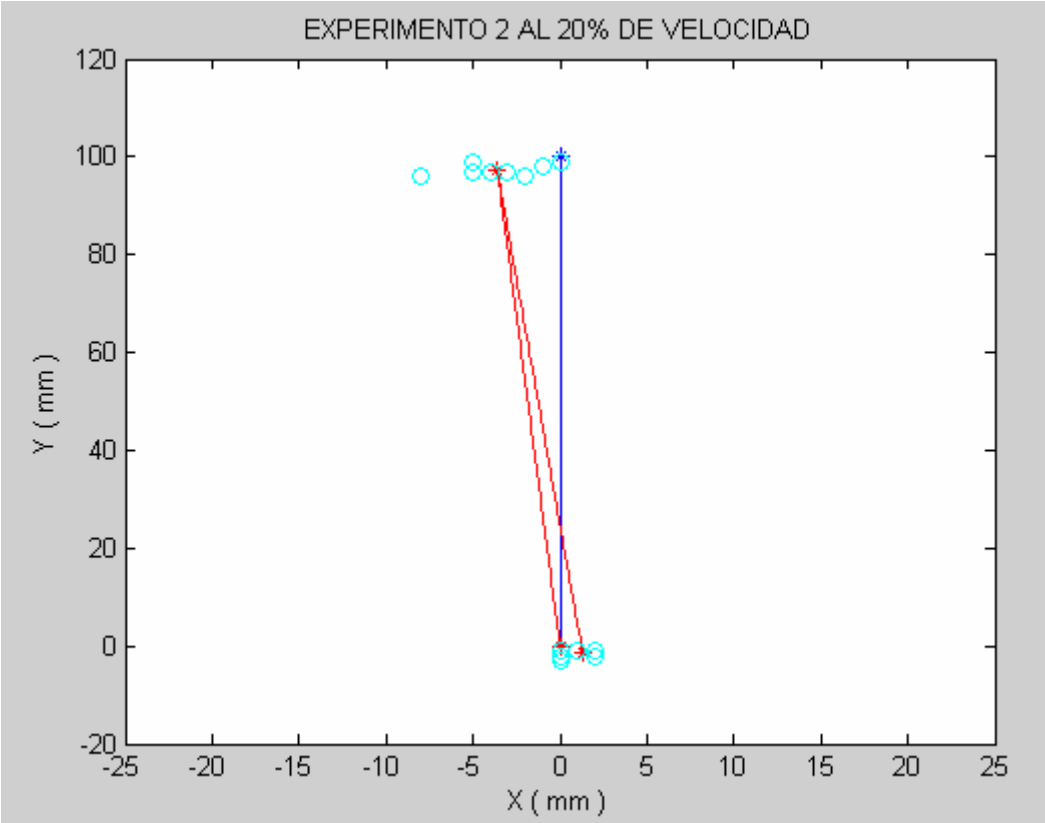

Figura 71: Experimento 2 al 20% de la velocidad para Moway 01

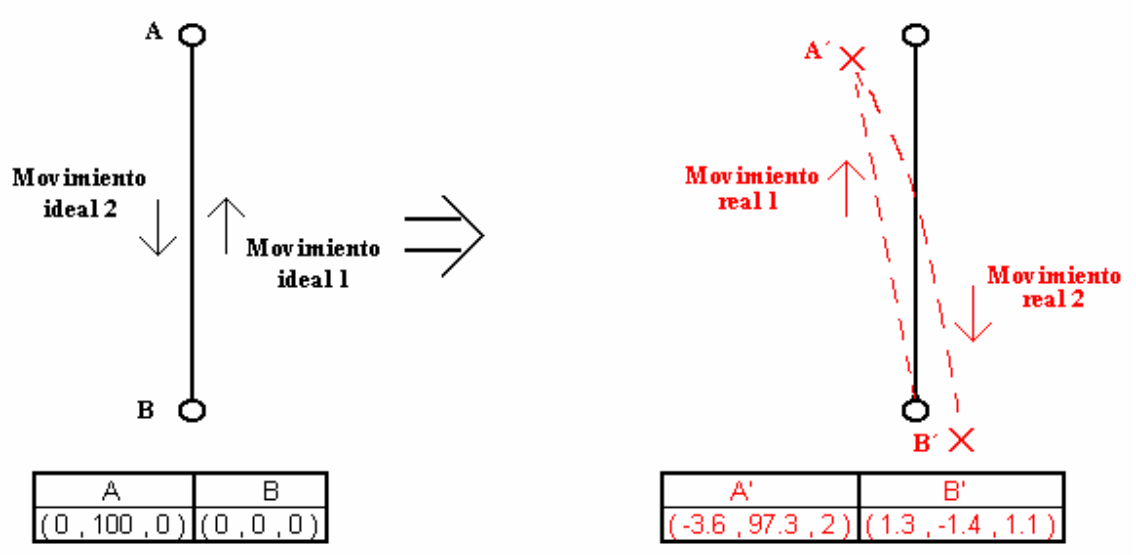

Figura 72: Experimento 2 al 20% de la velocidad para Moway 01

RESULTADOS EXPERIMENTO 3						
	A = (0 , 150 , 0)			B = (0 , 0 , 0)		
Velocidad	Desv x(mm)	Desv y(mm)	Angulo(º)	Desv x(mm)	Desv y(mm)	Angulo(º)
100%	-8,5	1,1	3,3	2,3	-1,1	1,0
60%	-10,6	-1,8	4,1	3,0	-2,9	2,3
20%	-8,3	-2,8	3,6	2,8	-2,1	1,8
Media Aritm.	-9,1	-1,2	3,7	2,7	-2,0	1,7

Tabla 20: Resultados experimento 3 para Moway 01

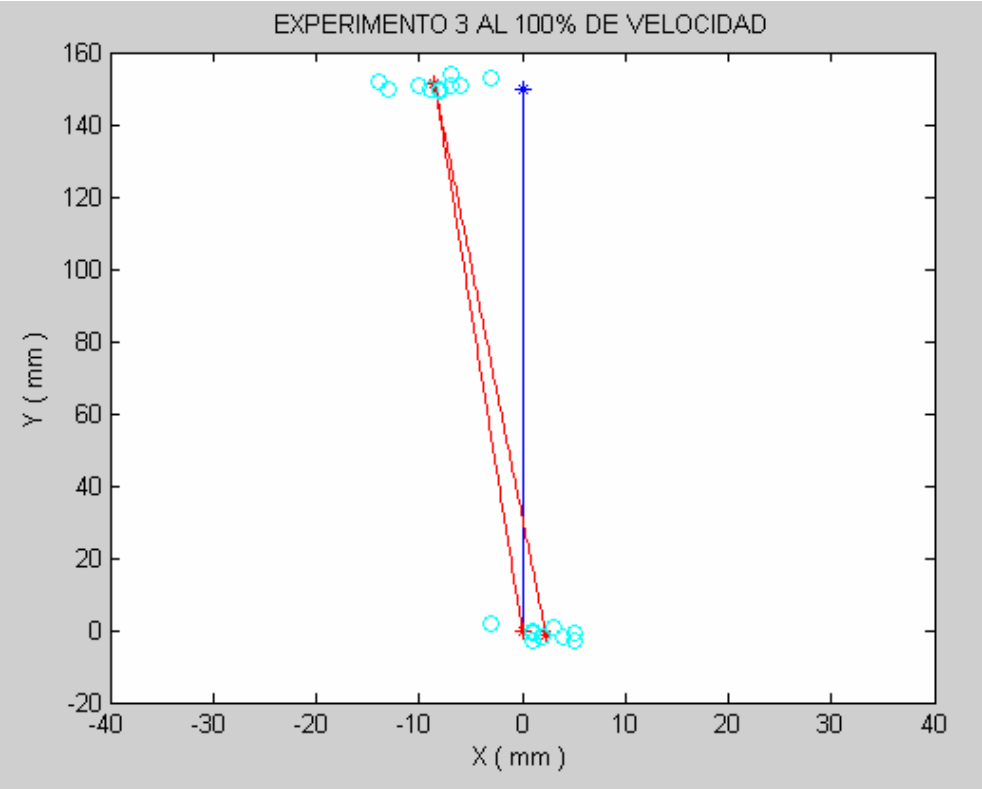

Figura 73: Experimento 3 al 100% de la velocidad para Moway 01

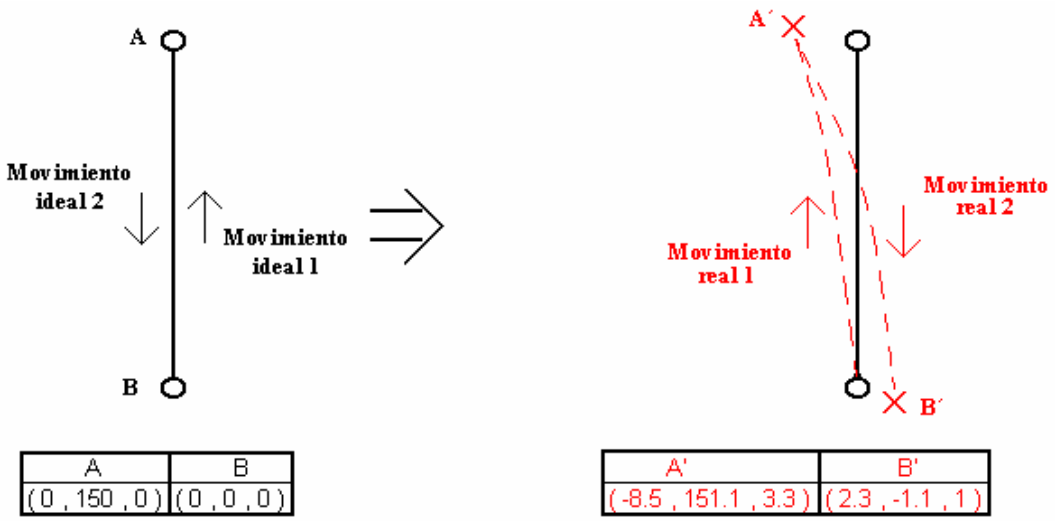

Figura 74: Experimento 3 al 100% de la velocidad para Moway 01

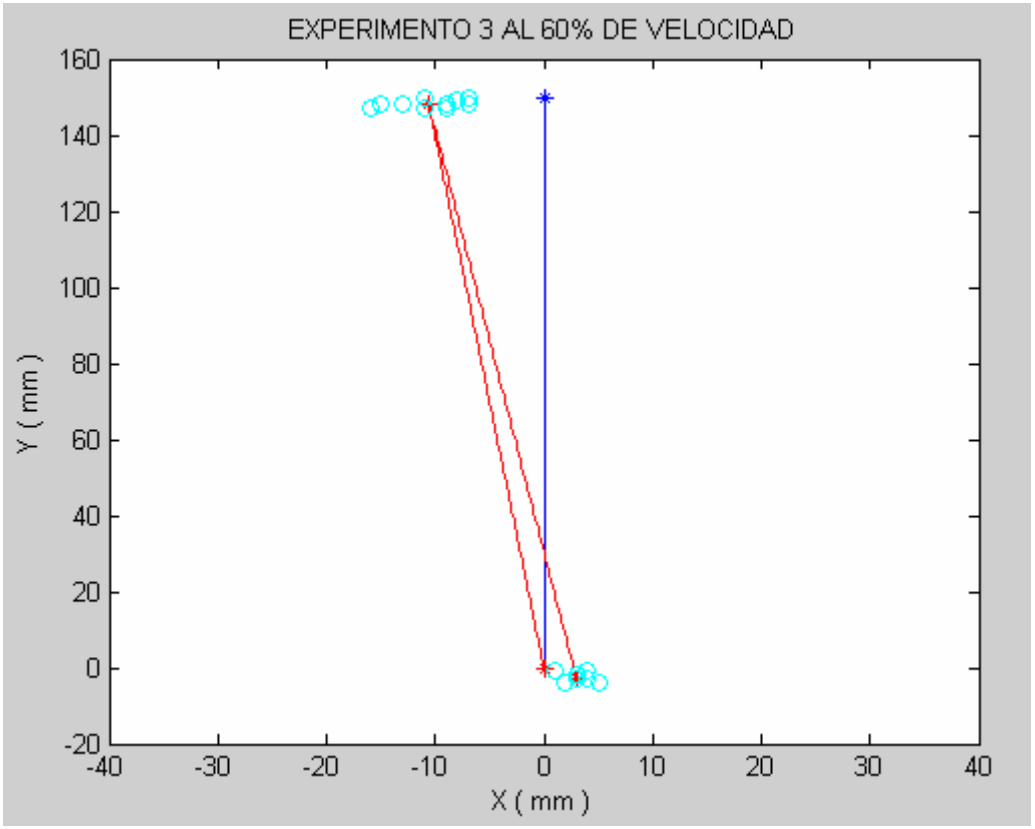

Figura 75: Experimento 3 al 60% de la velocidad para Moway 01

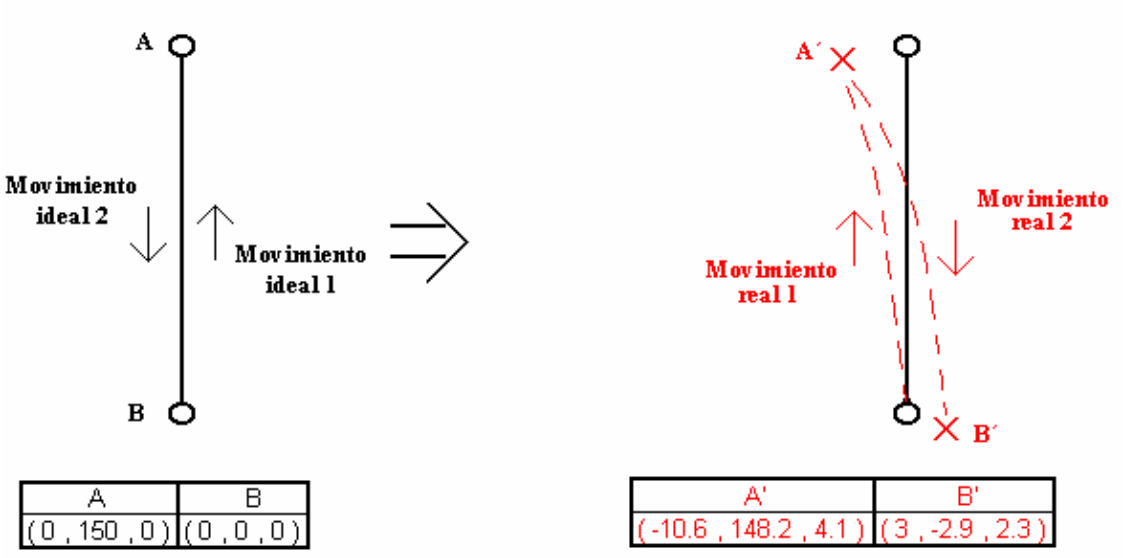

Figura 76: Experimento 3 al 60% de la velocidad para Moway 01

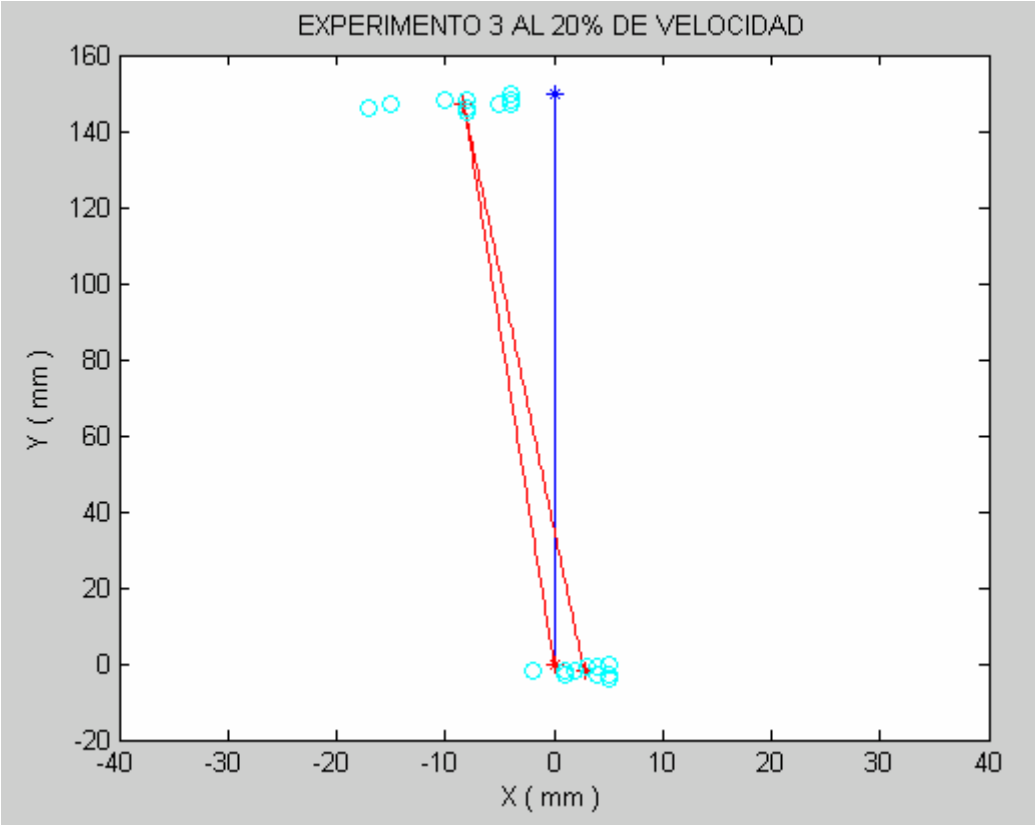

Figura 77: Experimento 3 al 20% de la velocidad para Moway 01

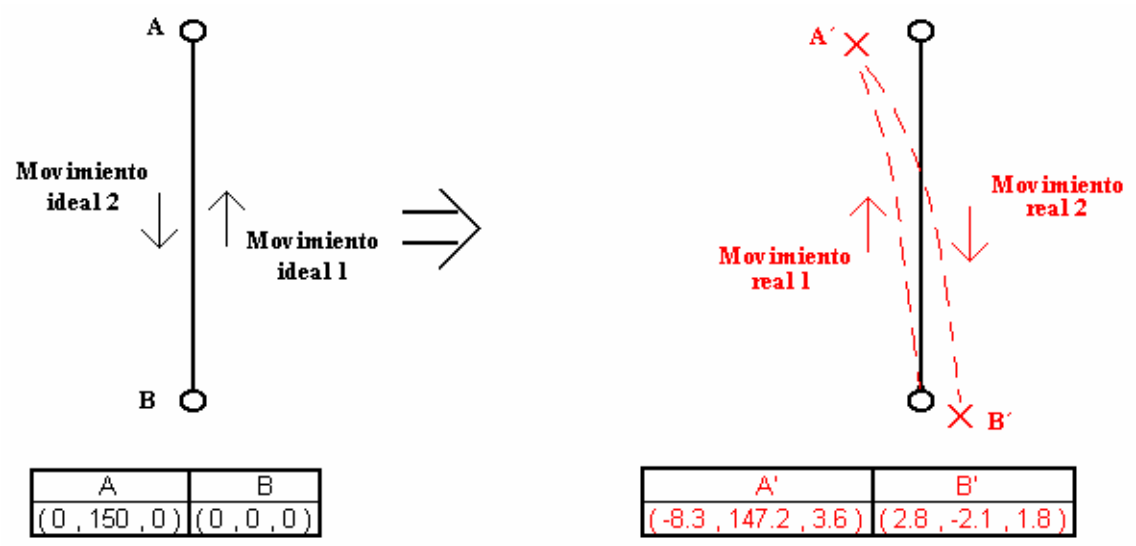

Figura 78: Experimento 3 al 20% de la velocidad para Moway 01

Con Moway 01, se obtienen resultados totalmente aleatorios. Es imposible extraer alguna conclusión acerca de su comportamiento o de la tendencia de sus movimientos, mucho menos de sacar alguna relación entre velocidad, desviación y distancia recorrida.

2.4. ACUMULACIÓN DE ERRORES

Se ha verificado que se producen pequeños errores cada vez que Moway ejecuta un comando. En principio, se puede llegar a pensar que estas desviaciones son de escasa importancia, pero nos daremos cuenta a través de este experimento que deben ser una de las mayores preocupaciones del programador, ya que a medida que la aplicación es mas complicada y requiere más comandos a ejecutar, los errores acumulados son tan grandes que trayectoria ideal y real son totalmente diferentes.

A continuación se realizan una serie de experimentos para determinar ese error acumulado. El recorrido consiste en avanzar en línea recta y regresar posteriormente al punto inicial 50 veces. Una vez finalizado el número de repeticiones, se mide la distancia entre el punto en el que queda el robot y el punto de inicio. Se varía tanto la velocidad como la distancia recorrida, para intentar sacar alguna relación entre estas y el error cometido.

```
//Con este programa se intenta cuantificar el error acumulado en sucesivas repeticiones

#include <16F876A.h>
#DEVICE ADC= 8
#use delay(clock=4000000)

//Librerías
#include "lib_sen_moway.h"
#include "lib_mot_moway.h"

//Declaración de variables
static int8    i;

void main()
{
          //Configuración PIC para utilizar sensores y motores
          SEN_CONFIG();
          MOT_CONFIG();

          Delay_ms(5000);

          for(i=0;i<50;i++)
          {
                    //recorre 5cm hacia delante y hacia atrás al 100% de velocidad 50 veces
                    MOT_STR(100,FWD,DISTANCE,500/17);
                    while(!input(MOT_END)){}
                    Delay_ms(200);
                    MOT_STR(100,BACK,DISTANCE,500/17);
                    while(!input(MOT_END)){}
                    Delay_ms(200);
          }
}
```

La línea azul corresponde a la trayectoria ideal que debería de realizar el robot. Los círculos azul claro, simbolizan los puntos finales a los que llega Moway después de las 50 repeticiones. Y el punto en estrella de color rojo corresponde a la media de todos los puntos finales.

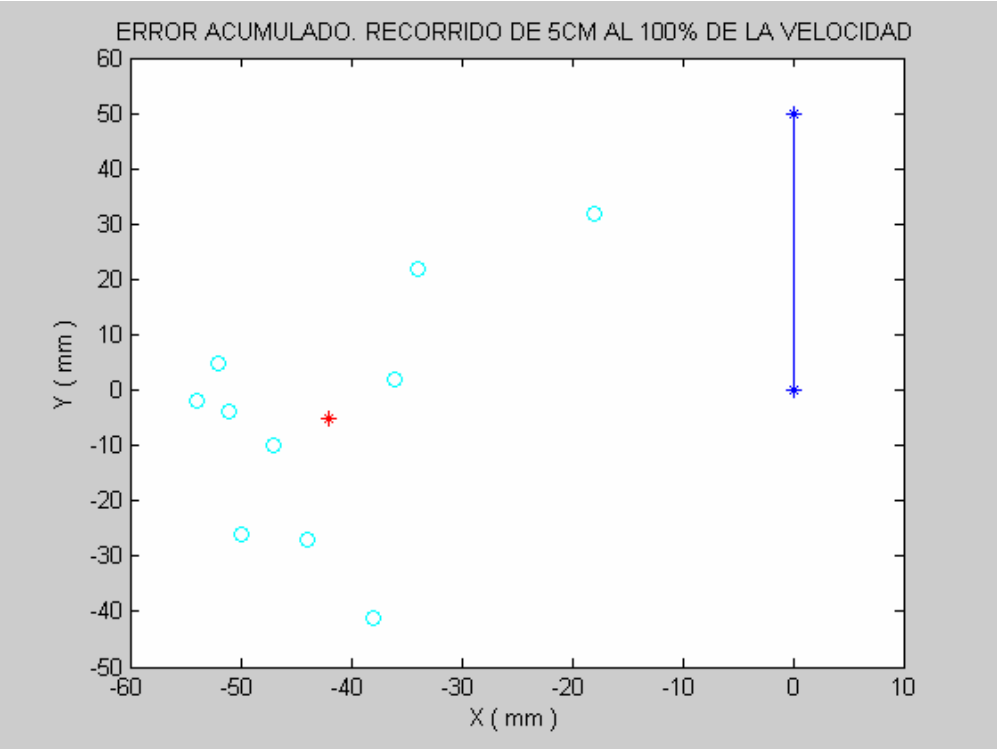

Figura 79: Error acumulado, con recorrido 5cm al 100% de la velocidad, con Moway 02

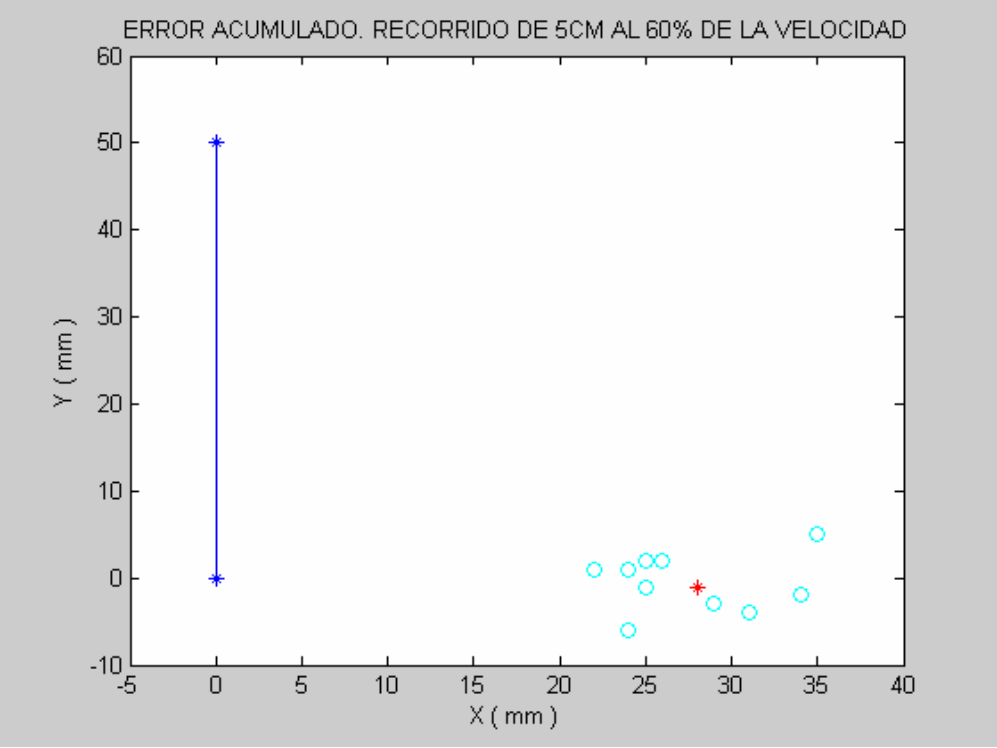

Figura 80: Error acumulado, con recorrido 5cm al 60% de la velocidad, con Moway 02

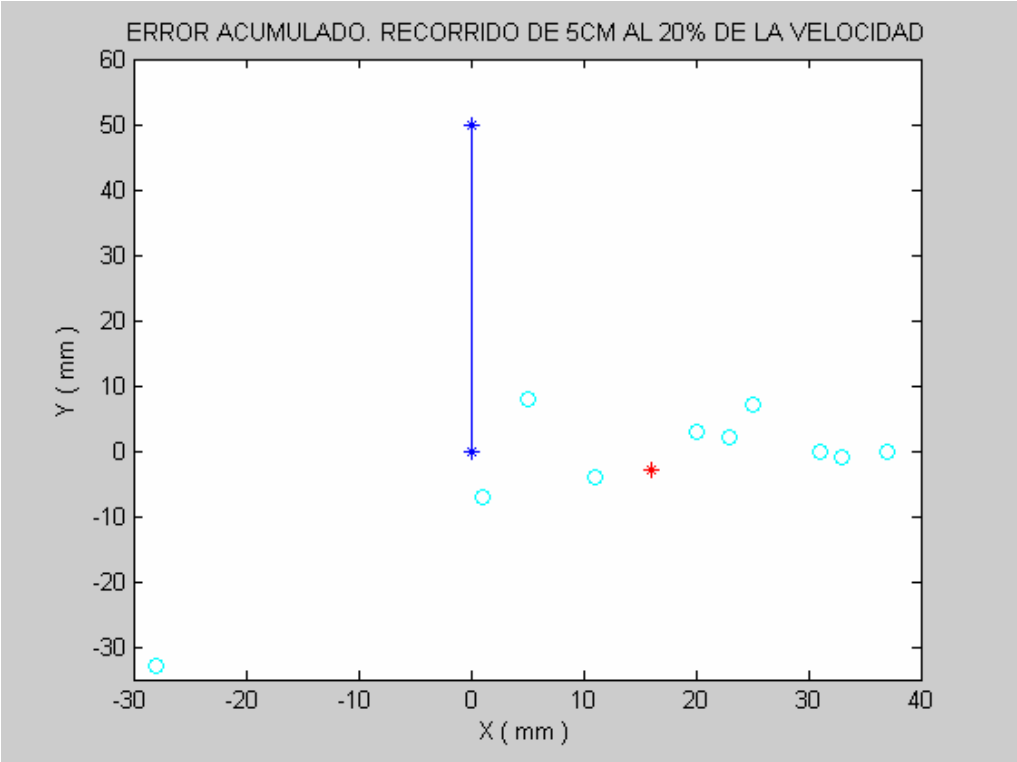

Figura 81: Error acumulado, con recorrido 5cm al 20% de la velocidad, con Moway 02

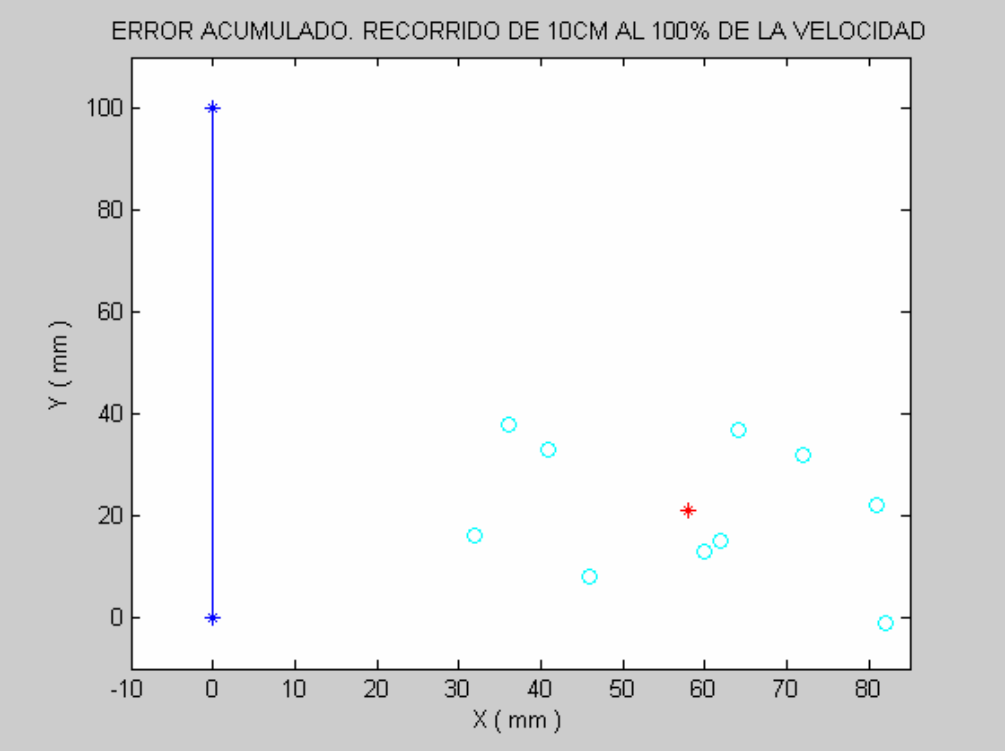

Figura 82: Error acumulado, con recorrido 10cm al 100% de la velocidad, con Moway 02

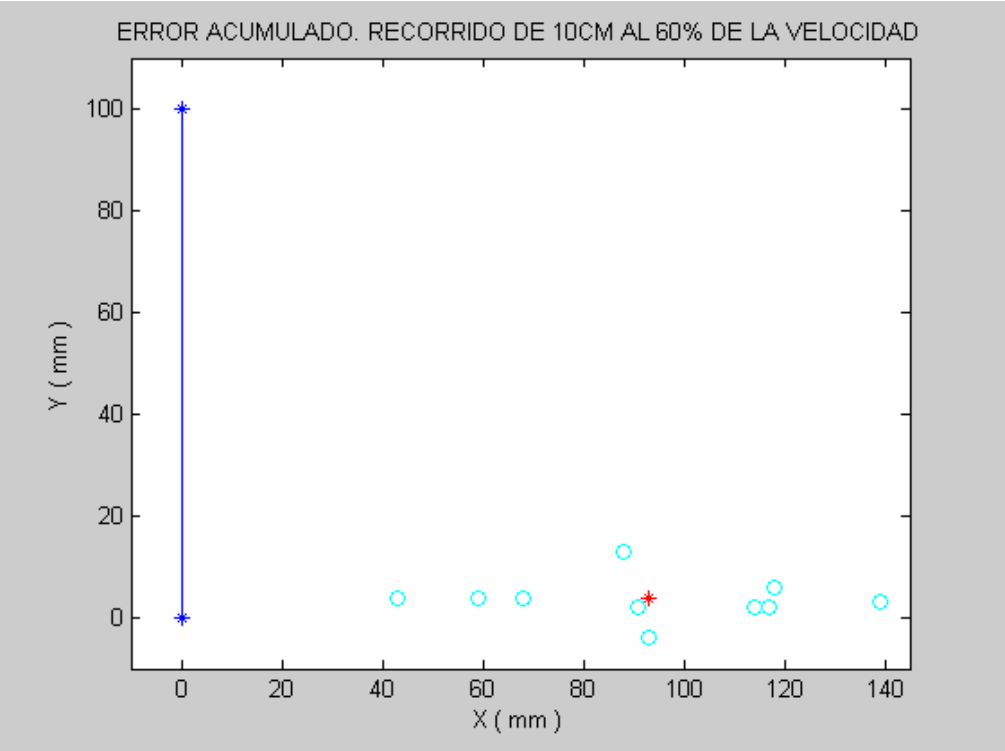

Figura 83: Error acumulado, con recorrido 10cm al 60% de la velocidad, con Moway 02

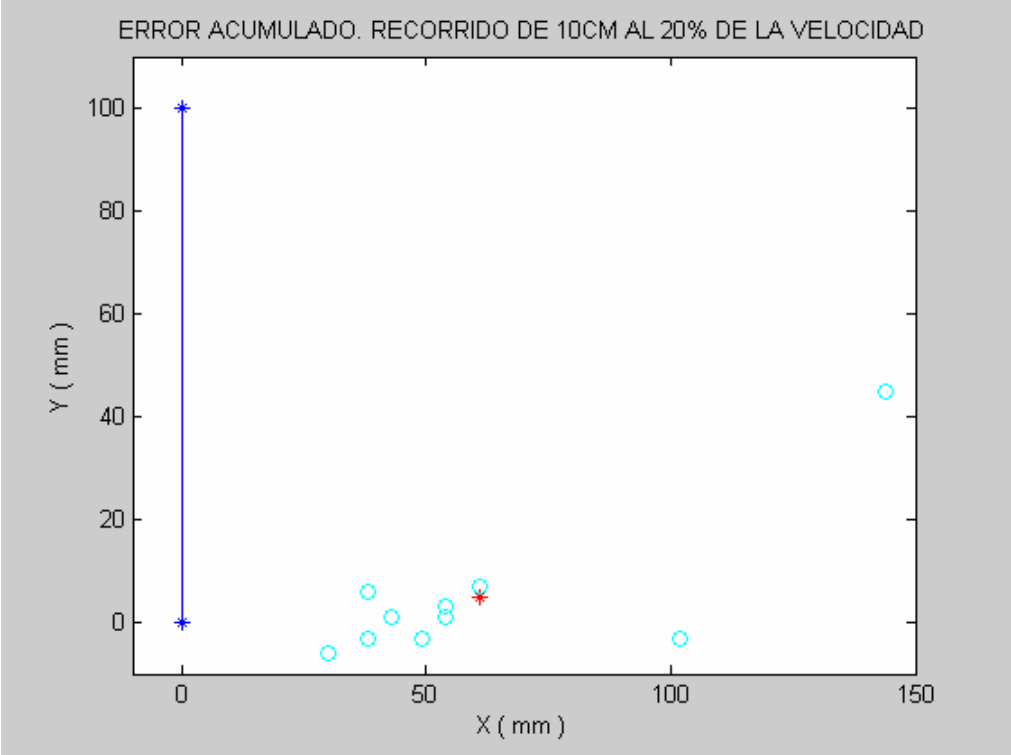

Figura 84: Error acumulado, con recorrido 10cm al 20% de la velocidad, con Moway 02

PRUEBAS CON MOWAY 01

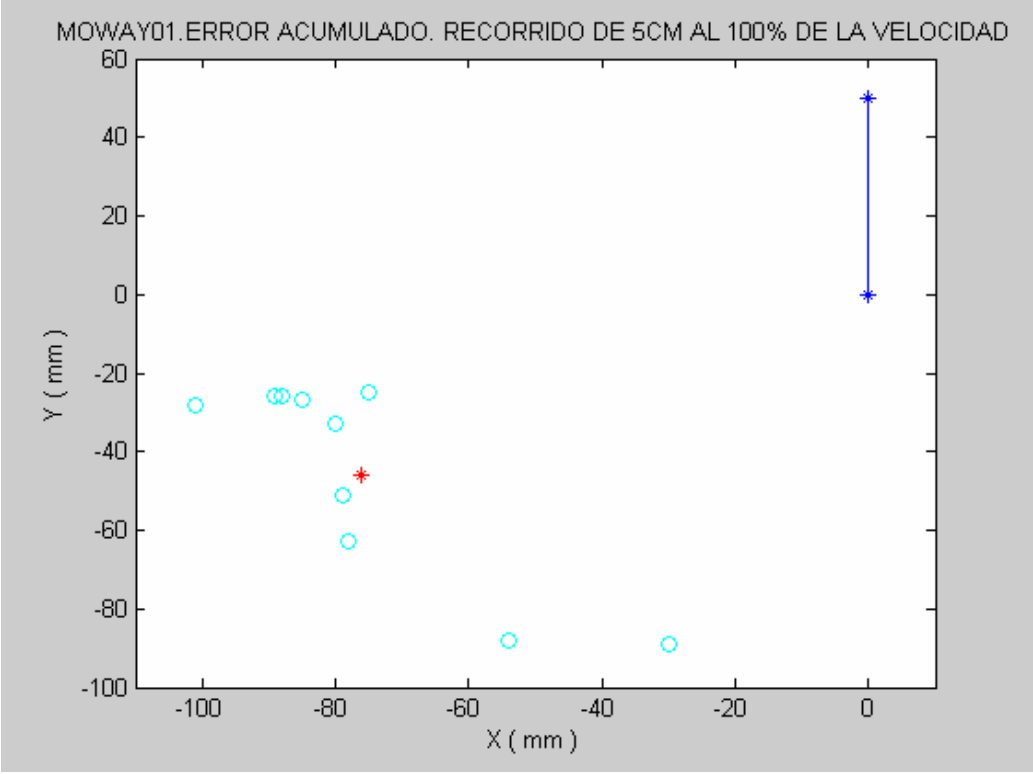

Figura 85: Error acumulado, con recorrido 5cm al 100% de la velocidad, con Moway 01

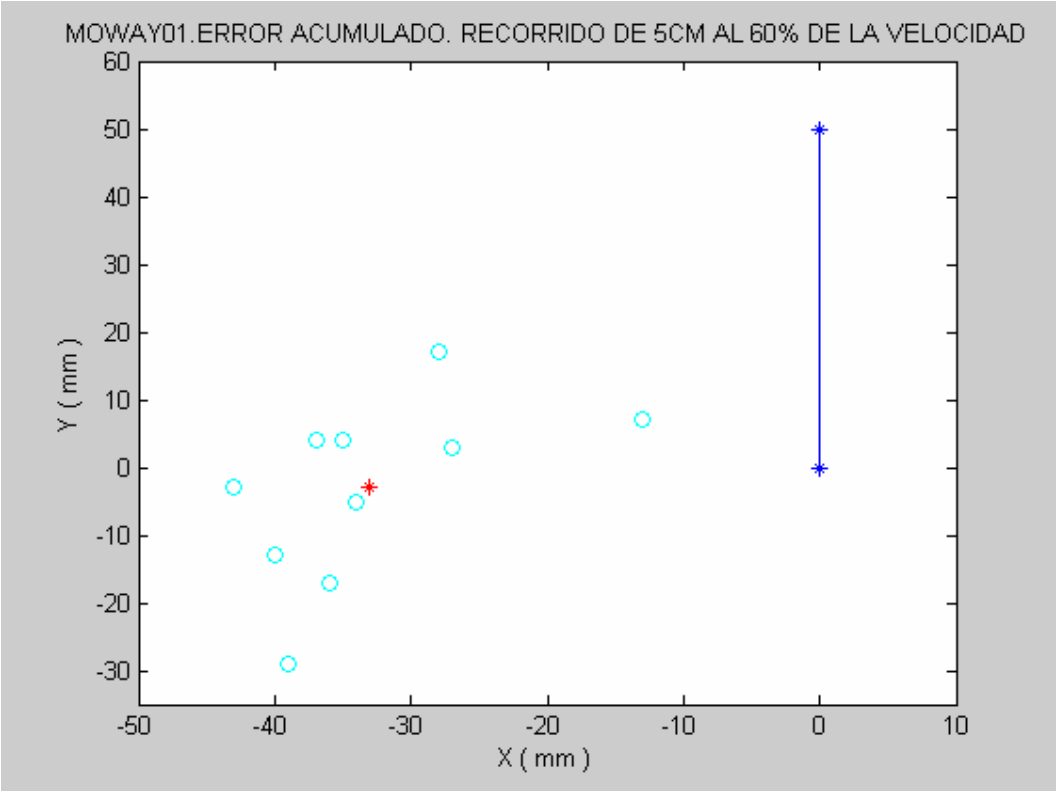

Figura 86: Error acumulado, con recorrido 5cm al 60% de la velocidad, con Moway 01

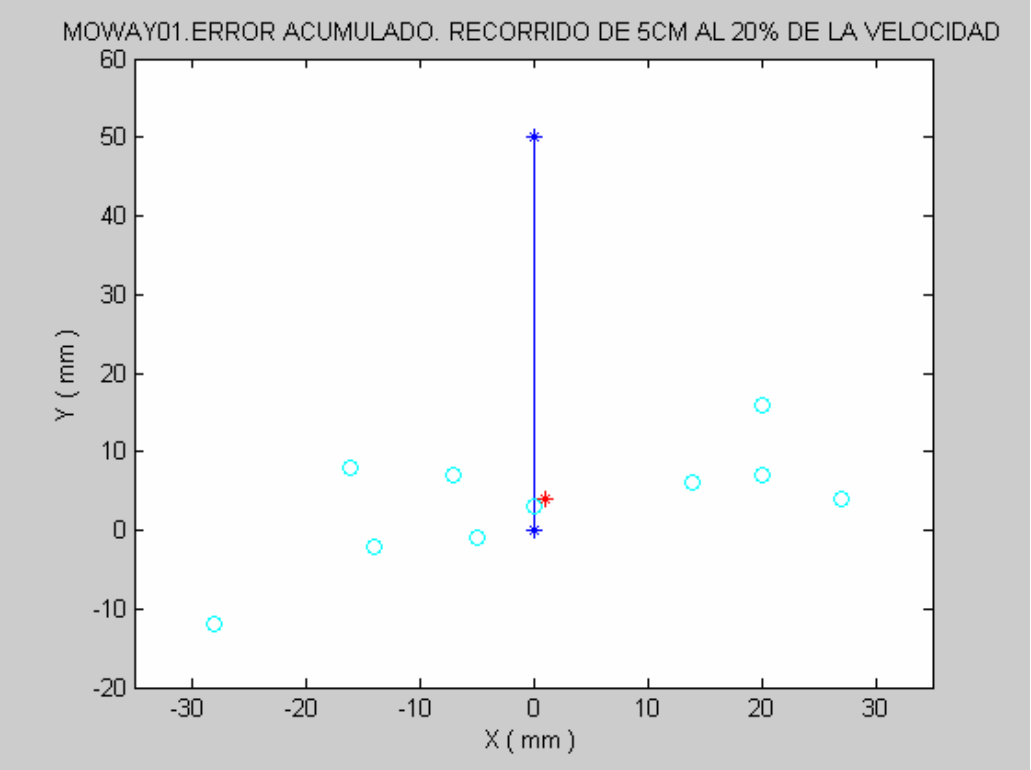

Figura 87: Error acumulado, con recorrido 5cm al 20% de la velocidad, con Moway 01

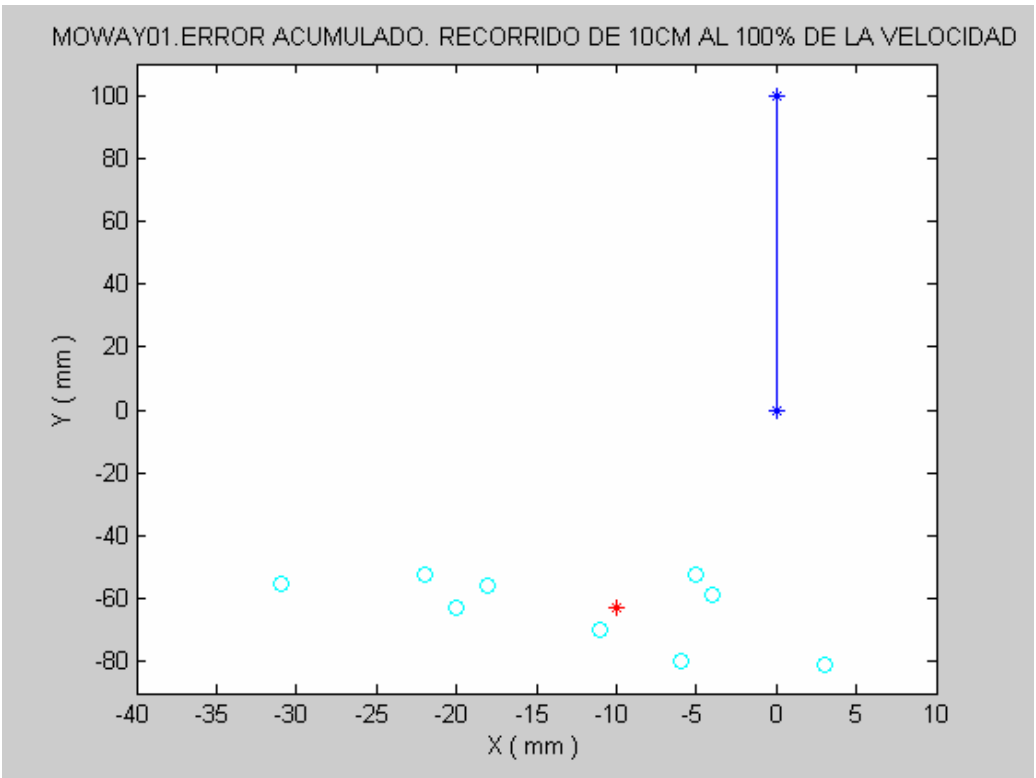

Figura 88: Error acumulado, con recorrido 10cm al 100% de la velocidad, con Moway 01

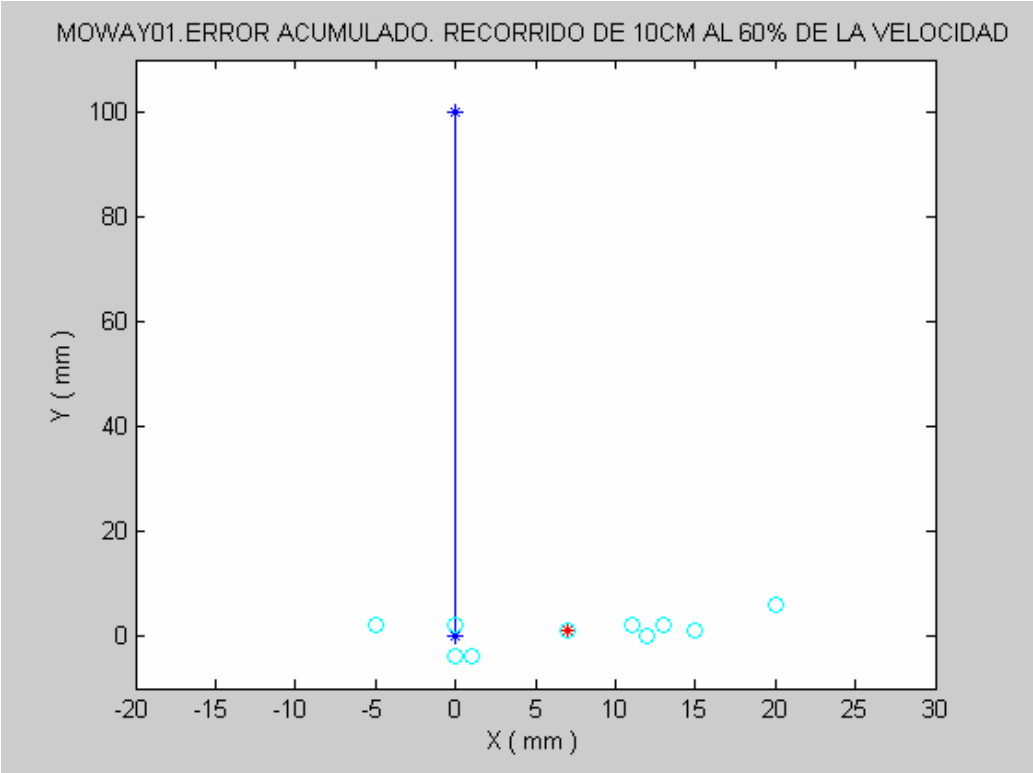

Figura 89: Error acumulado, con recorrido 10cm al 60% de la velocidad, con Moway 01

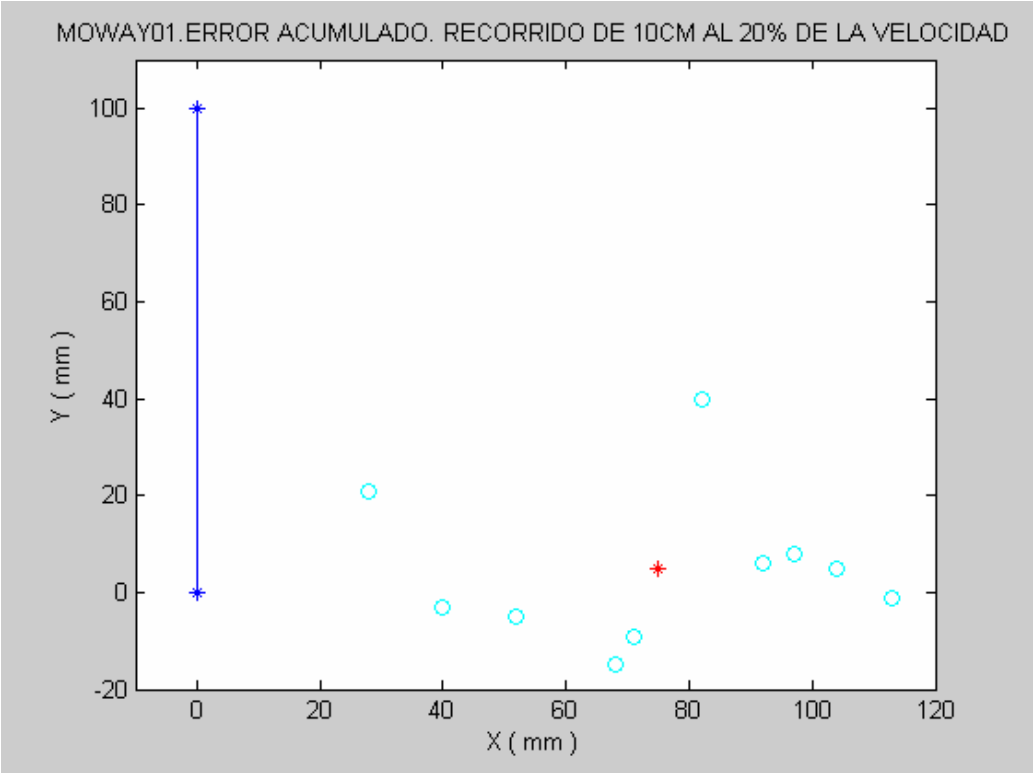

Figura 90: Error acumulado, con recorrido 10cm al 20% de la velocidad, con Moway 01

Este experimento es ideal para llegar a conclusiones determinantes en el comportamiento de Moway.

Primero se ha trabajado con Moway 02. En este caso, Moway 02 siempre se desvía hacia la derecha en el eje horizontal salvo en el primer caso que se corresponde a una longitud de 5 centímetros y 100% de la velocidad máxima. Con esto se puede asegurar que la tendencia para este robot es la de desviarse a la derecha, pero existen excepciones.

Por otro lado, debemos señalar la dispersión de los puntos a los que se llega en los diez casos en los que se repite el experimento. Esta dispersión suele ser mas acentuada en el eje horizontal que en el vertical, pero no se puede asegurar con rotundidad puesto que también en este aspecto surgen excepciones.

Con Moway 01 ocurre tanto de lo mismo. Aunque parece ser que la tendencia es a desviarse a la derecha cuanto mayor es la longitud recorrida y menor la velocidad, es tanta la dispersión que nos es imposible prever el comportamiento de Moway y tomar la media de la desviación no tendría ningún sentido ya que no reflejaría la realidad.

Después de este apartado, se tiene más claro que al ir ejecutando comando tras comando, se van acumulando errores aleatorios que llevarían al robot a una posición totalmente diferente, desconocida y alejada de la que se supone idealmente.

2.5. ERRORES EN EL GIRO

Al hacer rotar a Moway un cierto ángulo se produce un error, por lo que el robot rota un ángulo diferente al programado. Del mismo modo, igual que para el movimiento rectilíneo, se realizan una serie de experimentos para cuantificar, si fuera posible, ese exceso o defecto de ángulo girado.

Se comienza trabajando con Moway 02. El programa que se genera es el siguiente:

```
Delay_ms(5000);

//avanza 1.7mm
MOT_STR(100,FWD,DISTANCE,1);
while(!input(MOT_END)){}

//Rota a la derecha un ángulo que varía para adoptar valores comprendidos entre 45 y 360º
MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,100);
while(!input(MOT_END)){}

//avanza 1.7mm
MOT_STR(100,FWD,DISTANCE,1);
while(!input(MOT_END)){}
```

Se comprueba el comportamiento de Moway para distintos ángulos:

ÁNGULO	D.A. MEDIA
45	4,10
60	3,40
90	3,80
120	3,40
125	1,10
130	0,40
135	-0,20
150	-2,80
180	-3,00
210	-5,10
270	-8,60
360	-10,70

Tabla 21: Correspondencia entre ángulo girado y desviación para Moway 02

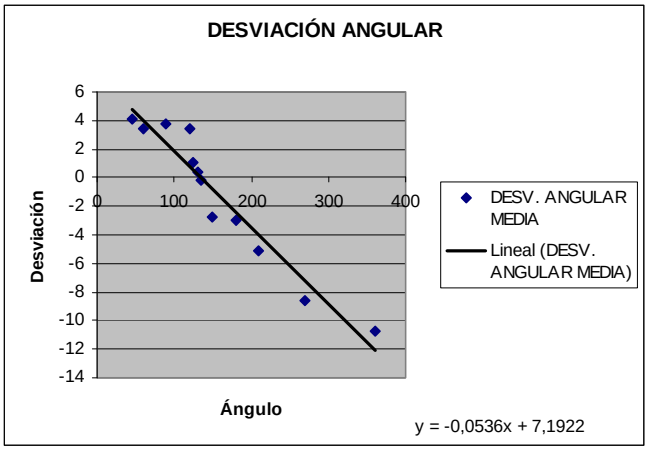

Figura 91: Desviación angular frente a ángulo girado para Moway 02

Al parecer se ha obtenido una relación entre ángulo girado y error cometido. Teniendo en cuenta esta relación se conseguiría, cuando las condiciones de entorno fuesen las mismas que en el experimento, rotar más o menos el ángulo real deseado, ya que los puntos experimentales obtenidos no se ajustan totalmente a la recta.

Al obtener estos resultados, se puede pensar que con todos los Moway se obtiene una recta similar. Se van a realizar los mismos experimentos con Moway 01 para comprobar si es cierto:

ÁNGULO	D.A. MEDIA
45	7,60
60	8,10
90	5,50
120	4,60
135	3,30
150	2,50
180	1,20
200	-1,20
210	-0,10
270	0,80
360	0,40

Tabla 22: Correspondencia entre ángulo girado y desviación para Moway 01

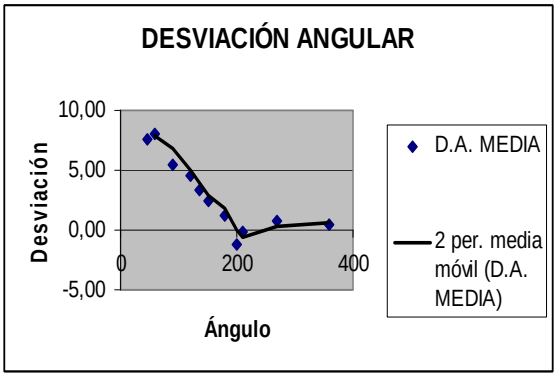

Figura 92: Desviación angular frente a ángulo girado para Moway 01

En este caso, ya no se puede obtener una relación lineal entre la desviación angular y el ángulo girado. Por lo que se debe desestimar la idea de poder corregir directamente en la programación, el error que se produce al emplear el comando MOT_ROT( ).

2.6. ERRORES EN MOVIMIENTO CURVO

Uno de los comandos de movimiento implementados en las librerías disponibles es MOT_CUR(). Con este comando se puede ejecutar un movimiento curvo, cuyo radio viene dado por una diferencia (que se suma o resta a la velocidad global del robot), entre las velocidades de cada una de las ruedas, especificada en el parámetro denominado MOT_RAD.

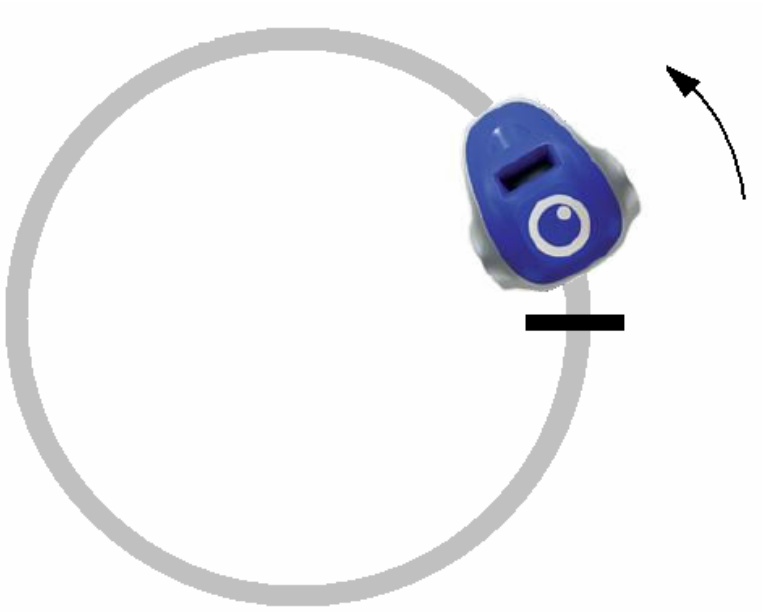

Figura 93: Recorrido curvo

En este experimento Moway realiza un recorrido cerrado circular. Como es imposible indicarle que realice una circunferencia completa (360º) ya que el radio de curvatura se indica a través de una diferencia de velocidades de las ruedas, lo que se hará es indicarle el final mediante una línea. De este modo, el robot parte de la línea y al llegar otra vez a ésta (cuando el sensor de línea la detecte) Moway para. Mediremos a continuación la distancia entre el punto inicial del que parte Moway y el punto al que retorna, en el eje radial de la circunferencia realizada.

El programa se detalla seguidamente:

```
//Comienza a realizar la curva para salir de la línea
MOT_CUR(50,FWD,45,LEFT,DISTANCE,30);
while(!input(MOT_END)){}
//Sigue realizando la curva, hasta que uno de los sensores detecte la línea
MOT_CUR(50,FWD,45,LEFT,DISTANCE,0);
while(!input(MOT_END))
{
        SEN_LINE_DIG();
        //en el momento que uno de los sensores detecte línea, para
        if(SEN_LINE_R==1 || SEN_LINE_L==1)
                MOT_STOP();
}
```

Se ha comprobado la desviación que se producía con distintos radios de curvatura variando el parámetro MOT_RAD y manteniendo la velocidad global del robot (MOT_VEL), tanto para Moway 01 como para Moway 02. La desviación positiva se corresponde a cuando se aleja del centro de curvatura y la negativa a cuando se acerca.

MOWAY 01	
MOT_RAD	DESVIACIÓN EJE RADIAL (mm)
10	-8
15	3
20	5
25	4
30	-1
35	12
40	10
45	8

Tabla 23: Desviación en el eje radial en función de MOT_RAD para Moway 01

MOWAY 02	
MOT_RAD	DESVIACIÓN EJE RADIAL (mm)
10	12
15	4
20	3
25	4
30	7
35	-2
40	9
45	6

Tabla 24: Desviación en el eje radial en función de MOT_RAD para Moway 02

Como se puede observar en las tablas, los resultados son totalmente aleatorios. En ninguno de los dos casos se puede hablar de ningún tipo de linealidad o dependencia, entre la desviación producida y el radio de curvatura.

Generalmente el microbot termina más alejado del centro de curvatura de lo que debería, pero esto (como en apartados anteriores, en el caso de movimiento rectilíneo o giro) no ocurre en todos los casos, por lo que no se pueden extrapolar resultados.

2.7. ERROR DEBIDO AL DESNIVEL DEL TERRENO

Para obtener un comportamiento óptimo de Moway, se debe garantizar un entorno con unas condiciones ideales. Entre ellas se encuentra la de tener una superficie de trabajo plana (sin ningún tipo de relieve) y horizontal (ángulo de inclinación con la horizontal nulo). Pero, ¿qué ocurre cuando la superficie no es la ideal?

Pues bien, en este apartado se realiza un experimento muy sencillo que consiste en hacer subir a Moway con movimiento rectilíneo por una superficie inclinada, desde 5 hasta 50 grados de inclinación, aumentando de 5 en 5 grados. La superficie escogida es de madera, aunque podrían emplearse otros materiales si para alguna aplicación se necesitase aumentar la adherencia del microbot a la superficie.

Figura 94: Moway sobre una superficie con pendiente

Los resultados obtenidos nos llevan a la conclusión de que para una superficie con inclinación inferior a 15 grados, el desnivel superficial no nos impide que el microbot se siga desplazando por la superficie de forma prácticamente rectilínea.

En los casos de 20 y 25 grados de inclinación, se observa un aumento del error por deslizamiento lateral de las ruedas respecto a la superficie.

A partir de 30 grados, el deslizamiento es tal que resulta imposible que Moway ascienda por la superficie.

Por este motivo, debemos de reducir el campo de aplicación a superficies con un grado de inclinación inferior a 15 grados. En el caso de que la aplicación se tenga que desarrollar por superficies con una inclinación mayor, se debería trabajar en la búsqueda de materiales para conseguir una adherencia máxima y reducir en lo posible la inclinación. En cualquier caso, recordar que la inclinación de la superficie es un factor muy importante que introduce errores importantes.

2.8. CONCLUSIONES

A lo largo del apartado número 2, se ha buscado relacionar los parámetros principales que definen los movimientos principales (avance en línea recta, giro, curva…) con el error que se produce.

En algunos casos se observa que en cierta manera, ese error que se genera va ligado a la velocidad del movimiento o a la longitud recorrida. Sin embargo, al repetir una y otra vez el mismo experimento manteniendo los mismos parámetros, se obtienen unos resultados totalmente dispersos. Somos capaces en algunos casos, de observar una tendencia del comportamiento, pero en ningún caso sería posible obtener la relación en forma de ecuación o relación numérica entre movimiento y error generado, debido a la aleatoriedad de los resultados.

"Es imposible conocer la posición exacta del microbot a partir de los comandos ejecutados". Este aspecto es muy importante tenerlo en cuenta, para no emplear una programación errada.

A partir de aquí, se debe indagar en la utilización de los sensores de línea, luz y obstáculos incorporados en Moway, para conocer la posición y situación real del microbot en cada instante. Además, el uso de balizas luminosas aportará puntos externos que servirán de referencia.

CAPÍTULO 3

DESARROLLO DE LAS CAPACIDADES DE MOWAY

Como se ha demostrado en el apartado anterior, es imposible desarrollar aplicaciones basándose en el conocimiento exacto del punto del plano en el que se encuentra Moway, debido a los errores de trayectoria que van unidos a cualquier movimiento. Es por esto, que nos debemos centrar en tres aspectos clave de Moway: su sensibilidad luminosa, capacidad de detección de obstáculos y de línea.

A continuación se desarrollan a fondo cada uno de estos aspectos para tener muy claro cuales deberían de ser las condiciones de superficie, fuente de luz, color de los obstáculos…para obtener un comportamiento óptimo.

3.1. ELECCIÓN DE LA MEJOR FUENTE DE LUZ

En el robot Moway, la luz penetra en el sensor APDS-9002 a través de una pequeña apertura en forma de media luna situada en la parte superior del chasis.

3.1.1. Sensor APDS-9002

Figura 95: Sensor APDS-9002

Características

- Excelente sensibilidad, la cual presenta picos en la curva de luminosidad del ojo humano.
- Es un sensor sin plomo, de pequeño tamaño protegido por una especie de carcasa. Medidas: alto de 0.80mm, ancho de 2.00mm y profundidad de 1.25mm.
- Salida lineal a lo largo del rango de iluminación.
- Baja variación de sensibilidad al emplear varias fuentes de luz.
- Prestaciones garantizadas a temperatura entre -40ºC y 85ºC.
- Tensión de alimentación entre 2.4 y 5.5V.

Aplicaciones

- Detectar la luz ambiente para adecuar la iluminación del display o pantalla a dicha iluminación ambiente. Se emplea en teléfonos móviles, PDAs, cámaras de video, TVs entre otros.
- Automatizar el control de luces en residenciales y centros comerciales.
- Señales electrónicas.
- Mecanismos expuestos a luz natural y artificial.

Descripción

El APDS-9002 es un fotosensor de pequeño tamaño y bajo coste, con salida analógica y que trabaja adecuadamente con luz ambiente. Es un fototransistor con un espectro adecuado, cuyo pico coincide más o menos con la curva de luminosidad del ojo humano. Proporciona una excelente respuesta cercana a la respuesta del ojo humano (la parte del espectro electromagnético que es visible para el ojo humano, son las radiaciones cuyas longitudes de onda están comprendidas entre los 400nm y 770nm), lo cual se muestra en el siguiente gráfico.

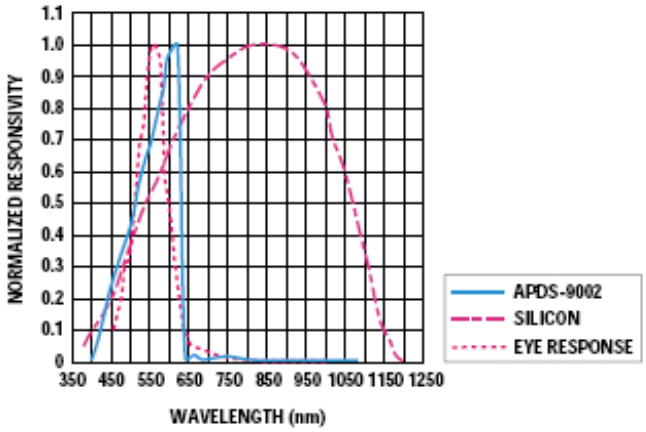

Figura 96: Curva de sensibilidad en el espectro electromagnético

El sensor APDS-9002 es ideal para aplicaciones en las cuales la medida de la luz ambientes se utiliza para control de luminosidad de un display. Con la introducción de este sensor en diseños como el de teléfonos móviles o PDAs, se consigue reducir significativamente su consumo de energía.

Condiciones de funcionamiento

Absolute Maximum Ratings
For implementations where case to ambient thermal resistance is ≤ 50°C/W

Parameter	Symbol	Min.	Max.	Units
Storage Temperature	T_S	-40	85	°C
Operating Temperature	T_A	-40	85	°C
Supply Voltage	V_{CC}	2.4	5.5	V

Tabla 25: Condiciones de funcionamiento de APDS-9002

Recommended Operating Conditions

Parameter	Symbol	Min.	Max.	Units	Conditions
Operating Temperature	T_A	-40	85	°C	
Supply Voltage	V_{CC}	2.4	5.5	V	

Tabla 26: Condiciones de funcionamiento recomendadas para APDS-9002

3.1.2. Algunos conceptos físicos relativos a la intensidad luminosa

Inicialmente se definen los conceptos de intensidad luminosa e iluminancia, para indicar la relación entre éstas y la distancia al foco luminoso.

Intensidad luminosa

La intensidad luminosa es la cantidad de luz emitida por una fuente puntiforme que se propaga en una determinada dirección. Tal intensidad se define como el cociente del flujo emitido en una cierta dirección en un cono de ángulo sólido, por lo tanto:

$$I = \frac{\Phi}{\omega}$$

Ecuación 5

donde I: intensidad luminosa en candelas (cd)

Φ: flujo que incide sobre la superficie en lúmenes (lm)

ω: ángulo sólido en estereorradianes (sr)

Luminancia

Si la fuente luminosa no es puntiforme se debe considerar en el ámbito de una determinada dimensión, por consiguiente la definición que había sido dada de intensidad luminosa no se puede seguir aplicando. De esta forma, será necesario introducir un nuevo concepto que evalúe la cantidad de energía luminosa emitida por estas superficies, ya sea una fuente de luz propia o una luz reflejada.

El valor fotométrico introducido de esta manera es la luminancia que se define como la relación entre la intensidad luminosa de la fuente en la dirección de un observador y la superficie emisora así como la ve el mismo observador (o superficie aparente). Las unidades de medida utilizadas son la cd/m^2 y el $lm/sr \times m^2$ (nit), siendo la relación fundamental dada por:

$$L = \frac{I_\alpha}{A \cdot \cos\alpha}$$

Ecuación 6

donde L: luminancia en candelas por metro cuadrado (cd/m^2)
I_α: intensidad luminosa en candelas en la dirección α
A: área de la fuente en metros cuadrados (m^2)
α: ángulo comprendido entre el ojo del observador y la recta normal a la fuente.

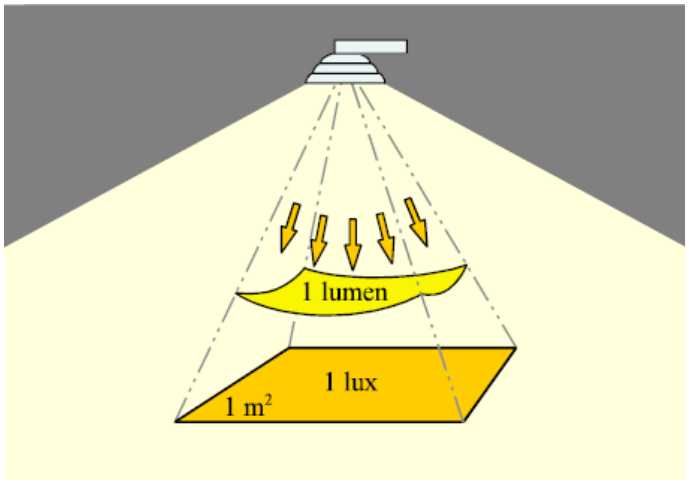

Figura 97: Relación superficie emisora y aparente

Ley inversa de los cuadrados

Expresa matemáticamente la relación entre la intensidad luminosa y la iluminancia. Establece que la iluminancia en un punto de una superficie es directamente proporcional a la intensidad luminosa de la luz incidente sobre el punto, e inversamente proporcional al cuadrado de la distancia desde la fuente. Cuando el punto se encuentra sobre una superficie normal a la luz incidente, la fórmula a aplicar es:

$$E = \frac{I}{D^2}$$

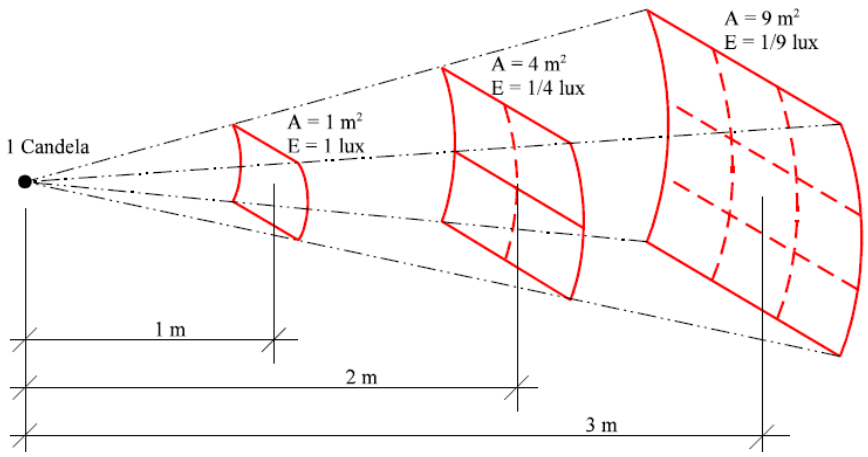

Figura 98: Ley inversa de los cuadrados

Esta Ley se basa en el concepto de fuente puntual, que produce radiación constante en todas direcciones. Bajo tales condiciones, el flujo contenido en un ángulo sólido unitario se distribuye sobre una superficie cada vez mayor a medida que aumenta la distancia a la fuente. Por tanto, la iluminancia decrece inversamente con el cuadrado de la distancia.

Esta última relación nos introduce un concepto que es intuitivo y conocido. Y es que el valor de iluminación en un punto es inversamente proporcional al cuadrado de la distancia al foco luminoso.

Esto tan intuitivo se debe tener en cuenta en el aspecto de que Moway va a tener que estar cerca del foco luminoso, como se observa experimentalmente, para reconocerlo como tal.

3.1.3. Iluminación ambiente

Otro aspecto clave es la iluminación ambiente. Esta debe de ser muy tenue para evitar la contaminación lumínica y que el microbot sea incapaz de reconocer la fuente de luz empleada como baliza.

A la hora de simular programas se deben buscar lugares con poca o nula iluminación cuando se haga uso del sensor de luz.

3.1.4. Direccionalidad del foco

Para obtener un buen comportamiento del microbot, se deben emplear focos luminosos que no sean direccionales (que emitan luz en todas direcciones).

Para comprobarlo, lo que se hizo fue colocar un foco direccional orientado horizontalmente a la altura de la ventana en forma de media luna tras la que se encuentra el fotosensor APDS-9002 que Moway lleva incorporado. Se grabó en el microbot el programa ejemplo 3 llamado “El faro”.

Cuando se sitúa la fuente luminosa direccional a unos 7 centímetros del robot, apuntando directamente a la ventana del fotosensor, Moway reconoce sin problemas la fuente de luz y se acerca a ella. El problema surge cuando el foco no esta apuntando directamente a la ventana del fotosensor, que sería lo más habitual. En este caso el microbot se muestra incapaz de dirigirse al foco luminoso.

Siempre se ha de intentar utilizar fuentes que emitan luz en todas direcciones.

3.1.5. Tipo de luz que utilizaremos

La fuente de luz que se emplee, debe de reunir dos requisitos importantes: emitir luz en una longitud de onda óptima para que sea captada por el fotosensor y que la emita en todas direcciones.

Según la hoja de características del fotosensor, de la cual se han extraído los datos más relevantes en el apartado 3.1.1., la parte del espectro electromagnético que detecta el sensor APDS-9002, va de los 400nm a los 650nm, situándose el pico de máxima sensibilidad en los 620nm (ver figura 96).

La fuente de luz ideal a emplear emitirá luz con una longitud de onda entorno a los 620nm, que se corresponde al color naranja en el espectro visible, según se aprecia en la figura 99.

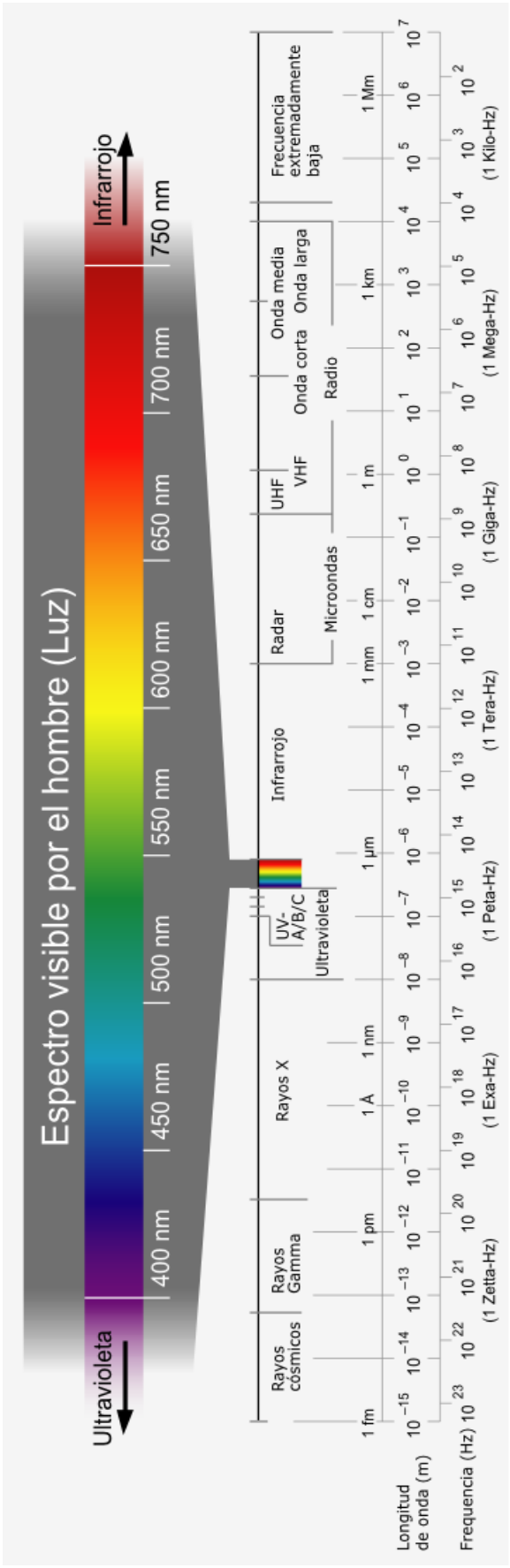

Figura 99: Espectro visible

3.2. CAPACIDAD DE DETECCIÓN DE LÍNEA

Los dos sensores de línea KTIR0711S van conectados dos puertos analógicos de manera que no solo es posible detectar el contraste de líneas negras sobre fondo blanco y viceversa, sino que también es posible discernir entre distintos tonos de color.

Esta nueva cualidad que se descubre en Moway abre un amplio abanico de posibilidades. Se podría asignar un color distinto a cada zona del plano de trabajo. De esta forma el sensor de línea daría la posición del robot en el plano (una especie de pixelado).

Es aquí cuando debemos empezar a preguntarnos ciertas cuestiones:

- ¿Cuántos colores diferentes es capaz de reconocer Moway?
- ¿Reconoce cualquier tipo de color o solo es posible distinguir tonalidades de gris?
- ¿Todos los sensores de línea KTIR0711S integrados en diferentes microbots dan el mismo valor ante el mismo color, es decir, todos los sensores están calibrados correcta e idénticamente?
- ¿La textura de la superficie influye en el valor que nos muestra el sensor?

Es necesario resolver todas estas cuestiones antes de pensar en todas las posibilidades que surgen. A partir de la hoja de características del sensor de línea, no es imposible solucionar las cuestiones planteadas. Por lo que será necesario resolverlas experimentalmente.

3.2.1. Sensor KTIR0711S

Características

- Compacto y delgado.
- Valores independientes de la iluminación.
- Alta sensibilidad.

Aplicaciones

- Grabadoras VCRs.
- Disk drive flexible
- Equipamiento de control microcomputerizado.

3.2.2. Sensibilidad espectral obtenida experimentalmente

Intentaremos resolver experimentalmente las cuestiones planteadas anteriormente: saber si reconoce como distintos varios colores RGB o solo tonalidades de gris, que número de colores es capaz de distinguir correctamente, si los dos sensores integrados en cada microbot están idénticamente calibrados, y si lo están distintos robots entre sí, entre otras.

Gama de colores

En este apartado se imprimen diez colores básicos (blanco, amarillo, naranja, rojo, azul cielo, verde, verde oscuro, violeta, azul oscuro y negro) y se sitúan uno junto al otro. El microbot recorre la superficie, leyendo constantemente el valor de cada uno de los sensores de línea.

Figura 100: Franjas de colores

El programa a ejecutar es el siguiente:

```
//valorLinea: realiza un movimiento rectilíneo de un metro midiendo cada 400ms el valor de cada uno de
//los sensores de línea, reflejados en la pantalla de recepción de radio-frecuencia de Moway Center.

#include <16F876A.h>
#DEVICE ADC= 8
#use delay(clock=4000000)

#include "lib_mot_moway.h"
#include "lib_sen_moway.h"
#include "lib_rf2gh4.h"

//Canal y dirección de mOway
int8 yo=0x01;
int8 canal=0x00;

//variable
static int8 i;

void main()
{
```

```
        Delay_ms(2000);

         //Configuración de PIC para motores, sensores y radio-frecuencia
        MOT_CONFIG();
        SEN_CONFIG();
        RF_CONFIG_SPI();
        RF_CONFIG(canal,yo);
        RF_INT_EN();
        RF_ON();

        //Realiza un movimiento rectilíneo de un metro al 10% de velocidad
        //En cada bucle se recorrerá 250mm.
        for(i=0;i<4;i++)
        {
                MOT_STR(10,FWD,DISTANCE,2500/17);
                while(!input(MOT_END))
                {
                        //Se comprueba los sensores de línea
                        SEN_LINE_ANALOG();
                        //se muestran en la posición 0 y 1 en la pantalla de recepción
                        RF_DATA_OUT[0]=SEN_LINE_L;
                        RF_DATA_OUT[1]=SEN_LINE_R;
                        //Se indica la dirección de salida
                        RF_DIR_OUT=0x02;
                        //Se mandan los datos
                        RF_SEND();
                        //se esperan 400ms para evitar tener una cantidad enorme de datos
                        Delay_ms(400);
                }
        }
}
```

En éste programa, se incluye la librería de radiofrecuencia para poder acceder a los datos que captan los sensores de línea. Moway es capaz de enviar 8 bytes de información por radiofrecuencia, que pueden ser leídos en la pantalla de recepción de Moway Center, ajustando el canal de comunicación. En éste caso, sólo se envian 2 bytes de información, uno por cada sensor de línea.

Los sensores devuelven un valor en analógico, comprendido entre 0 y 255. Cuando la superficie sobre la que se encuentra es clara el valor será cercano a 0 y cuando ésta sea muy oscura, el valor rondará el 255. Hay que tener en cuenta, que la información que se muestra en la pantalla de recepción se indica en hexadecimal, por lo que los valores estan comprendidos entre 0 y FF.

En la pantalla aparece una línea por cada recepción de datos. Cada línea consta de lo siguiente:

- La dirección desde donde ha sido enviada la información. En este caso se le ha asignado la dirección 01 a Moway (ver código del programa).
- El byte 0 corresponde al valor que devuelve el sensor de línea izquierdo.
- El byte 1 corresponde al valor que devuelve el sensor de línea derecho.
- Los seis bytes siguientes no se han utilizado en este caso para transmitir ningún tipo de información.
- Y finalmente se muestra el instante en el que se ha recibido la información.

A continuación se muestran la información transmitida por radiofrecuencia y almacenada en la pantalla de recepción de Moway Center al ejecutar el programa anterior con Moway 02:

```
//NEGRO
dir:01: E7 E8 00 00 00 00 00 00 (0:43:52:515)
dir:01: E7 E8 00 00 00 00 00 00 (0:43:52:109)
dir:01: E5 E7 00 00 00 00 00 00 (0:43:51:718)
dir:01: E4 E5 00 00 00 00 00 00 (0:43:51:312)
//AZUL OSCURO
dir:01: CF D4 00 00 00 00 00 00 (0:43:50:906)
dir:01: C3 C6 00 00 00 00 00 00 (0:43:50:500)
dir:01: C1 C3 00 00 00 00 00 00 (0:43:50:109)
dir:01: C1 C2 00 00 00 00 00 00 (0:43:49:703)
//VIOLETA
dir:01: 12 12 00 00 00 00 00 00 (0:43:49:296)
dir:01: 12 13 00 00 00 00 00 00 (0:43:48:890)
dir:01: 11 12 00 00 00 00 00 00 (0:43:48:484)
dir:01: 12 12 00 00 00 00 00 00 (0:43:48:93)
//VERDE OSCURO
dir:01: 15 16 00 00 00 00 00 00 (0:43:47:687)
dir:01: 13 14 00 00 00 00 00 00 (0:43:47:281)
dir:01: 12 13 00 00 00 00 00 00 (0:43:46:875)
dir:01: 13 13 00 00 00 00 00 00 (0:43:46:468)
//VERDE
dir:01: 11 11 00 00 00 00 00 00 (0:43:46:78)
dir:01: 10 10 00 00 00 00 00 00 (0:43:45:671)
dir:01: 10 10 00 00 00 00 00 00 (0:43:45:265)
dir:01: 0F 10 00 00 00 00 00 00 (0:43:44:859)
//AZUL CIELO
dir:01: 14 15 00 00 00 00 00 00 (0:43:44:468)
dir:01: 11 10 00 00 00 00 00 00 (0:43:44:62)
dir:01: 10 10 00 00 00 00 00 00 (0:43:43:656)
dir:01: 10 10 00 00 00 00 00 00 (0:43:43:250)
//ROJO
dir:01: 13 14 00 00 00 00 00 00 (0:43:42:843)
dir:01: 10 10 00 00 00 00 00 00 (0:43:42:453)
dir:01: 10 0F 00 00 00 00 00 00 (0:43:42:46)
dir:01: 0F 10 00 00 00 00 00 00 (0:43:41:640)
//NARANJA
dir:01: 10 11 00 00 00 00 00 00 (0:43:41:234)
dir:01: 10 10 00 00 00 00 00 00 (0:43:40:828)
dir:01: 10 10 00 00 00 00 00 00 (0:43:40:437)
dir:01: 0F 10 00 00 00 00 00 00 (0:43:40:31)
//AMARILLO
dir:01: 10 11 00 00 00 00 00 00 (0:43:39:625)
dir:01: 11 11 00 00 00 00 00 00 (0:43:39:218)
dir:01: 10 10 00 00 00 00 00 00 (0:43:38:828)
dir:01: 10 10 00 00 00 00 00 00 (0:43:38:421)
//BLANCO
dir:01: 0F 10 00 00 00 00 00 00 (0:43:38:15)
dir:01: 10 11 00 00 00 00 00 00 (0:43:37:609)
dir:01: 11 11 00 00 00 00 00 00 (0:43:37:203)
dir:01: 0F 10 00 00 00 00 00 00 (0:43:36:812)
```

De estos resultados se pueden sacar muchas conclusiones. En un principio pueden resultar descorazonadores, porque solo hay dos colores perfectamente distinguibles del resto (el azul oscuro y el negro) y para los demás colores, los sensores devuelven unos valores comprendidos entre 0F-14 el derecho y 10-16 el izquierdo. Por lo que de entre los diez colores empleados solo se podrían generar tres zonas perfectamente distinguibles para el robot.

A raíz de aquí, nos preguntamos que es lo que realmente mide el sensor. Un color está definido generalmente por tres parámetros, distintos según el modelo de color que se emplee para definirlo. Así que buscamos un parámetro que nos de la explicación de porque el microbot no distingue esos 7 colores escogidos al azar.

Al estudiar las propiedades del color, hay que fijarse en un parámetro, la luminancia del modelo de color HSL, el cual parece dar la explicación.

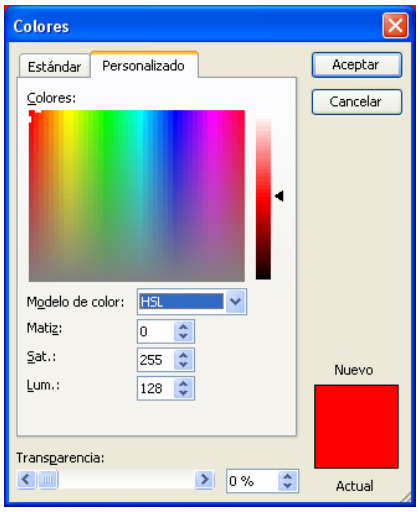

Figura 101: Selección luminancia

COLOR	LUMINANCIA
Blanco	255
Amarillo	128
Naranja	128
Rojo	128
Azul cielo	120
Verde	144
Verde Oscuro	88
Violeta	104
Azul Oscuro	48
Negro	0

Tabla 27: Luminancia de cada franja escogida

La mayoría de los colores escogidos tienen una luminancia similar.

Para una luminancia entre 255 y 88 cd/m^2, el sensor de línea devuelve unos valores similares. Si bien es cierto que para el verde oscuro con luminancia de 88 cd/m^2, el sensor comienza a dar valores ligeramente superiores al resto.

El azul oscuro de 48 cd/m^2 y el negro de 0 cd/m^2, los distingue perfectamente el sensor de línea.

Se puede por tanto asegurar, que Moway diferencia mejor los colores oscuros entre sí que los claros, ya que es más sensible a luminancias entre 0 y 88 cd/m^2 que comprende un rango de 88 valores, que a luminancias entre 88 y 255 cd/m^2 que comprende un rango de 167 valores y en el que no es capaz de distinguir colores.

Tonalidades de gris

Las distintas tonalidades de gris se obtienen al ir variando la luminancia a partir del blanco o negro. Sabemos que Moway es capaz de distinguir unas tonalidades de otras. La cuestión es saber cuál es el número de tonalidades que sabe distinguir perfectamente y cuáles son esas tonalidades. Aspectos que debemos de ir aclarando de cara al diseño de la superficie de trabajo.

Se imprimen una serie de tonalidades de gris. Se han escogido al azar tonos con la siguiente luminancia: 30, 40, 50, 60, 70, 80, 90, 100, 150, 200 y 220. Se colocan uno a continuación del otro y con el programa creado para la escala de color, hacemos que Moway 02 se desplace a lo largo del conjunto de tonalidades. Se obtienen los siguientes resultados:

LUMINANCIA 220
dir:01: 39 45 00 00 00 00 00 00 (15:52:12:203)
dir:01: 3C 49 00 00 00 00 00 00 (15:52:11:890)
dir:01: 3D 49 00 00 00 00 00 00 (15:52:11:593)
dir:01: 3D 4A 00 00 00 00 00 00 (15:52:11:281)
dir:01: **3F 4B** 00 00 00 00 00 00 (15:52:10:984)
dir:01: 3A 45 00 00 00 00 00 00 (15:52:10:687)
dir:01: **31 3E** 00 00 00 00 00 00 (15:52:10:375)
dir:01: 36 43 00 00 00 00 00 00 (15:52:10:78)
dir:01: 36 42 00 00 00 00 00 00 (15:52:9:781)

LUMINANCIA 200
dir:01: 5E 69 00 00 00 00 00 00 (15:52:7:953)
dir:01: **5E 69** 00 00 00 00 00 00 (15:52:7:656)
dir:01: 5A 67 00 00 00 00 00 00 (15:52:7:343)
dir:01: **50 5D** 00 00 00 00 00 00 (15:52:7:46)
dir:01: 56 62 00 00 00 00 00 00 (15:52:6:750)
dir:01: 57 63 00 00 00 00 00 00 (15:52:6:437)
dir:01: 53 5E 00 00 00 00 00 00 (15:52:6:140)

LUMINANCIA 150
dir:01: 94 9C 00 00 00 00 00 00 (15:52:5:234)
dir:01: 92 9B 00 00 00 00 00 00 (15:52:4:921)
dir:01: **94 9C** 00 00 00 00 00 00 (15:52:4:625)
dir:01: 8E 97 00 00 00 00 00 00 (15:52:4:328)
dir:01: 89 92 00 00 00 00 00 00 (15:52:4:15)
dir:01: 81 8B 00 00 00 00 00 00 (15:52:3:718)
dir:01: 83 8B 00 00 00 00 00 00 (15:52:3:406)
dir:01: **80 89** 00 00 00 00 00 00 (15:52:3:109)

LUMINANCIA 100
dir:01: **BE C4** 00 00 00 00 00 00 (15:52:2:203)
dir:01: BA C1 00 00 00 00 00 00 (15:52:1:906)
dir:01: BC C1 00 00 00 00 00 00 (15:52:1:593)
dir:01: BA C0 00 00 00 00 00 00 (15:52:1:296)
dir:01: B3 B8 00 00 00 00 00 00 (15:52:0:984)
dir:01: B0 B5 00 00 00 00 00 00 (15:52:0:687)
dir:01: AE B4 00 00 00 00 00 00 (15:52:0:390)
dir:01: A9 AF 00 00 00 00 00 00 (15:52:0:78)
dir:01: **A6 AD** 00 00 00 00 00 00 (15:51:59:781)

LUMINANCIA 90
dir:01: C5 CB 00 00 00 00 00 00 (15:51:58:875)
dir:01: **C6 CB** 00 00 00 00 00 00 (15:51:58:562)
dir:01: C2 C7 00 00 00 00 00 00 (15:51:58:265)
dir:01: BD C2 00 00 00 00 00 00 (15:51:57:968)
dir:01: BA BF 00 00 00 00 00 00 (15:51:57:656)
dir:01: B8 BE 00 00 00 00 00 00 (15:51:57:359)
dir:01: B6 BC 00 00 00 00 00 00 (15:51:57:46)
dir:01: **B1 B6** 00 00 00 00 00 00 (15:51:56:750)

LUMINANCIA 80
dir:01: **C7 CB** 00 00 00 00 00 00 (15:51:55:843)
dir:01: C6 CA 00 00 00 00 00 00 (15:51:55:531)
dir:01: C5 CA 00 00 00 00 00 00 (15:51:55:234)
dir:01: BE C4 00 00 00 00 00 00 (15:51:54:937)
dir:01: B9 BF 00 00 00 00 00 00 (15:51:54:625)
dir:01: BA C0 00 00 00 00 00 00 (15:51:54:328)
dir:01: B7 BD 00 00 00 00 00 00 (15:51:54:15)
dir:01: **B1 B8** 00 00 00 00 00 00 (15:51:53:718)

LUMINANCIA 70
dir:01: CF D3 00 00 00 00 00 00 (15:51:52:812)
dir:01: **CF D3** 00 00 00 00 00 00 (15:51:52:515)
dir:01: CE D2 00 00 00 00 00 00 (15:51:52:203)
dir:01: C9 CE 00 00 00 00 00 00 (15:51:51:906)
dir:01: C5 C9 00 00 00 00 00 00 (15:51:51:593)
dir:01: C0 C6 00 00 00 00 00 00 (15:51:51:296)
dir:01: BE C4 00 00 00 00 00 00 (15:51:51:0)
dir:01: BD C2 00 00 00 00 00 00 (15:51:50:687)
dir:01: **B8 BD** 00 00 00 00 00 00 (15:51:50:390)

LUMINANCIA 60
dir:01: D5 D7 00 00 00 00 00 00 (15:51:49:484)
dir:01: **D6 D8** 00 00 00 00 00 00 (15:51:49:171)
dir:01: D5 D8 00 00 00 00 00 00 (15:51:48:875)
dir:01: D1 D3 00 00 00 00 00 00 (15:51:48:578)
dir:01: D1 D3 00 00 00 00 00 00 (15:51:48:265)
dir:01: CD D1 00 00 00 00 00 00 (15:51:47:968)
dir:01: C9 CB 00 00 00 00 00 00 (15:51:47:656)
dir:01: C4 C8 00 00 00 00 00 00 (15:51:47:359)
dir:01: C3 C7 00 00 00 00 00 00 (15:51:47:62)
dir:01: **C0 C3** 00 00 00 00 00 00 (15:51:46:750)

LUMINANCIA 50
dir:01: **D8 DA** 00 00 00 00 00 00 (15:51:45:843)
dir:01: D6 D8 00 00 00 00 00 00 (15:51:45:546)
dir:01: D5 D8 00 00 00 00 00 00 (15:51:45:234)
dir:01: D2 D6 00 00 00 00 00 00 (15:51:44:937)
dir:01: CD D1 00 00 00 00 00 00 (15:51:44:640)
dir:01: C9 CF 00 00 00 00 00 00 (15:51:44:328)
dir:01: C7 CC 00 00 00 00 00 00 (15:51:44:31)
dir:01: C4 C9 00 00 00 00 00 00 (15:51:43:718)
dir:01: **C1 C6** 00 00 00 00 00 00 (15:51:43:421)

LUMINANCIA 40
dir:01: **E0 E2** 00 00 00 00 00 00 (15:51:42:515)
dir:01: DF E1 00 00 00 00 00 00 (15:51:42:203)
dir:01: DF E2 00 00 00 00 00 00 (15:51:41:906)
dir:01: DF E1 00 00 00 00 00 00 (15:51:41:609)
dir:01: DC DE 00 00 00 00 00 00 (15:51:41:296)
dir:01: D8 DB 00 00 00 00 00 00 (15:51:41:0)
dir:01: D5 D9 00 00 00 00 00 00 (15:51:40:703)
dir:01: D1 D5 00 00 00 00 00 00 (15:51:40:390)
dir:01: D0 D4 00 00 00 00 00 00 (15:51:40:93)
dir:01: **CD D1** 00 00 00 00 00 00 (15:51:39:781)

LUMINANCIA 30
dir:01: **E2 E5** 00 00 00 00 00 00 (15:51:38:875)
dir:01: E2 E4 00 00 00 00 00 00 (15:51:38:578)
dir:01: E2 E4 00 00 00 00 00 00 (15:51:38:265)
dir:01: E0 E4 00 00 00 00 00 00 (15:51:37:968)
dir:01: DB DE 00 00 00 00 00 00 (15:51:37:671)
dir:01: D7 DA 00 00 00 00 00 00 (15:51:37:359)
dir:01: **D5 DA** 00 00 00 00 00 00 (15:51:37:62)

En el primer byte se muestra el valor analógico del sensor de línea izquierdo y en el segundo, el sensor derecho. De cada tono hemos marcado en negrita el máximo y mínimo valor que nos da el sensor. De esta forma, existen 6 tonalidades de gris de entre las 11 que se han escogido al azar, que son perfectamente distinguibles para el microbot.

TONO GRIS	VALOR SENSOR IZQUIERDO		VALOR SENSOR DERECHO	
LUMINANCIA	MÍNIMO	MÁXIMO	MÍNIMO	MÁXIMO
220	31	3F	3E	4B
200	50	5E	5D	69
150	80	94	89	9C
100	A6	BE	AD	C4
60	C0	D6	C3	D8
30	D5	E2	DA	E5

Tabla 28: Respuesta de los sensores de línea de Moway 02 para cada tono escogido

Las franjas de valores que devuelve el sensor para cada color no se solapan, por lo que es cierto que son distinguibles en este caso. Lo que ocurre es que al realizar otra vez el mismo proceso, no se obtienen exactamente los mismos máximos y mínimos. Por lo que antes de hacer la elección de tonos debemos repetir el proceso varias veces y con distintos microbots, si finalmente vamos a trabajar con varios de ellos.

Tanto el rango de valores en el que oscila el valor devuelto por un sensor para un tono de gris (en algunos casos es superior a 25 unidades) como la diferente calibración entre distintos microbots y entre los sensores derecho e izquierdo del mismo microbot, reducen el número de tonos perfectamente distinguibles que podemos utilizar.

Experimentalmente llegamos a la conclusión de que el número máximo de tonos perfectamente distinguibles tanto por Moway 01 como Moway 02, es de 6. Lo que supone una barrera clara a las intenciones de pixelado del área de trabajo.

3.2.3. Modelo HSL

Existe un modelo llamado **HSL** (del inglés *Hue*, *Saturation*, *Lightness* – Tonalidad, Saturación, Luminancia), también llamado **HSI** (del inglés *Hue*, *Saturation*, *Intensity* – Tonalidad, Saturación, Intensidad), el cual define un modelo de color en términos de sus componentes constituyentes. Es un modelo de representaciones considerado "natural", ya que se acerca bastante a la percepción fisiológica del color que tiene el ojo humano.

El modelo HSL se representa gráficamente como un cono doble o un doble hexágono. Los dos vértices en el modelo HSL se corresponden con el blanco y el negro, el ángulo se corresponde con la tonalidad, la distancia al eje con la saturación y la distancia al eje blanco-negro corresponde a la luminancia, también llamada brillo.

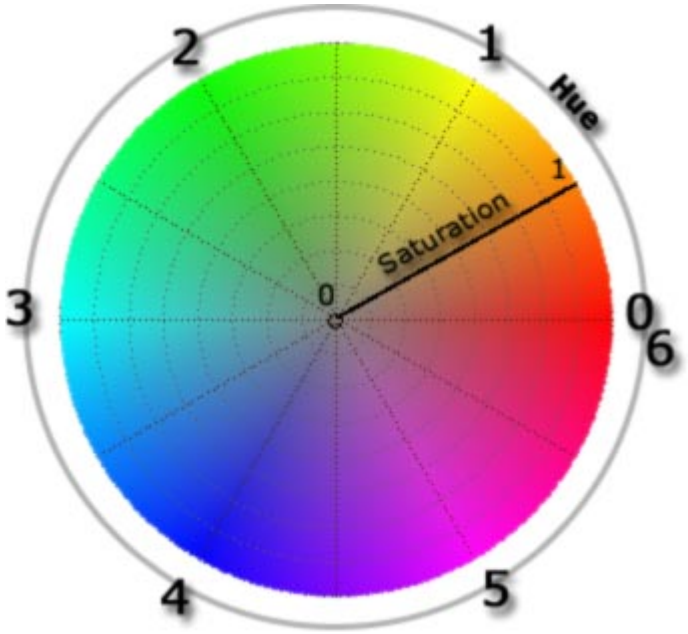

Figura 102: Cono de representación modelo HSL

Hue	Color
0	red
1	yellow
2	green
3	cyan
4	blue
5	magenta
6	red

Tabla 29

Otra forma quizá más clara de representación del modelo HSL se muestra a continuación, donde el color se representa mediante un círculo cromático, y la luminosidad y la saturación se representan mediante dos ejes:

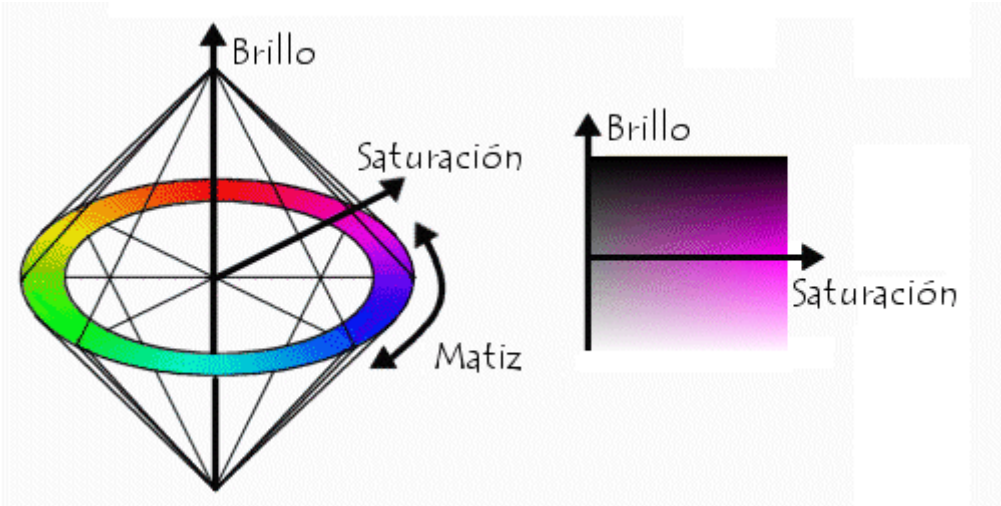

Figura 103: Representación modelo HSL

La luminancia indica la cantidad de luz del color, es decir, el grado de claridad u oscuridad de un color (una camisa en el sol o en la sombra).

La luminancia permite definir lo vivo o iluminado del color: 0 significa que su iluminación es nula, es decir el color está tan poco iluminado que es negro. Conforme aumentamos el valor de la iluminancia conseguimos un color mas vivo hasta obtener el blanco cuando la iluminancia es 1.

Las siguientes seis escalas muestran que sucede con los colores cuando se cambia la luminancia:

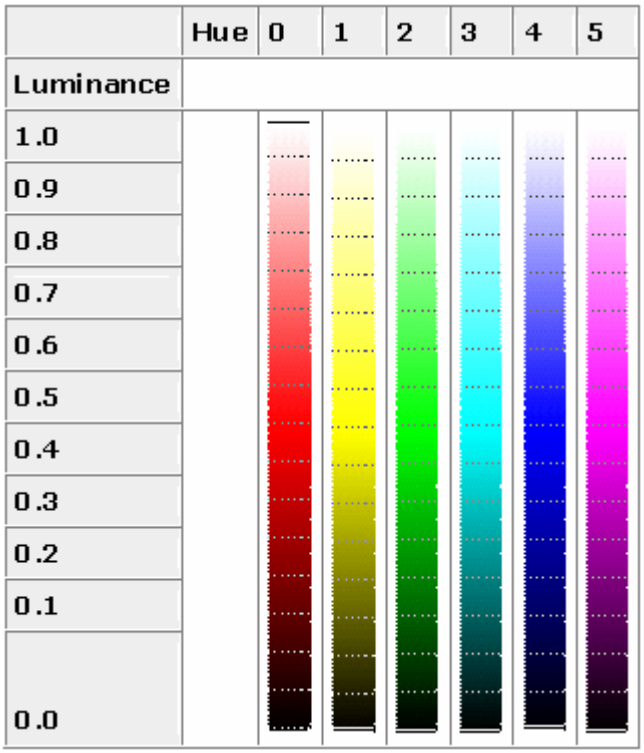

Figura 104: Escalas de colores básicos variando la luminancia

Hacemos hincapié en este modelo de color y sobretodo en el parámetro de luminancia o brillo, porque los sensores de línea KTIR0711S acoplados en Moway, nos dan una medida de la luminancia que presenta la superficie sobre la cual se encuentra.

3.2.4. Gama de colores a utilizar

La elección de los colores a utilizar ha de realizarse minuciosamente. Se puede optar por una gama de grises o gama de color para hacer más vistosa la aplicación. Hay que elegir el color basándonos en la luminancia de éste y comprobar experimentalmente que el sensor distingue los colores incluyendo un rango de seguridad.

Esta gama de colores no será superior a 6 tonos generalmente.

3.2.5. Degradados

La situación ideal a la que se aspira al introducirle un sensor de línea analógico a un microbot, sería la de poder situar la posición exacta del microbot en un plano, a partir del valor que nos devuelve el sensor de línea. El aspecto de la superficie recorrida sería lo más parecido a un pixelado.

Para poder lograr esa exactitud, tendríamos que tener un sensor muy exacto que devolviera un único valor ante un tono de gris y además tener una impresión de muy alta calidad.

Como ya hemos observado en los apartados anteriores, el sensor KTIR0711S incorporado a Moway, no es capaz de aportarnos la exactitud suficiente para pensar en el pixelado.

Otra opción serían los degradados. Un degradado permitiría a Moway conocer la dirección en la que se desplaza. Pero otra vez nos encontramos con los mismos problemas anteriores. Dejando a un lado el hecho de que las impresiones generalmente dejan mucho que desear aunque sean de alta calidad, tenemos el problema de la falta de precisión del sensor.

Figura 105: Degradado

Imaginemos que a lo largo de un A4 tenemos el degradado máximo que se puede obtener, es decir, de una luminancia de 0 hasta 255. La longitud aproximada del folio es de unos 30 centímetros y el número máximo de tonos de gris perfectamente distinguible para Moway de aproximadamente 6, por lo que sólo cada 5 centímetros, el robot sería capaz de saber exactamente la dirección con la que recorre el folio. En cualquier otro punto en el que realizase la medición es posible que le llevase a error.

Por lo que de momento descartaremos los degradados, apostando más por los tonos uniformes distinguibles entre sí.

3.3. ASPECTO SUPERFICIAL DE LOS OBSTÁCULOS

El sensor de detección de obstáculos está compuesto por una fuente de luz infrarroja llamada KPA3010-F3C, y dos receptores PT100F0MP colocados en ambos extremos de Moway y conectados a entradas analógicas.

Tal y como se indica en el manual de Moway, se recomienda un entorno claro para aumentar la posibilidad de reflexión de la luz infrarroja.

Experimentalmente se ha comprobado que los obstáculos transparentes o muy oscuros no los detecta. Por lo que se emplearán obstáculos con superficies claras y opacas.

3.4. COMPARATIVA Y USO DE FUENTES DE ALIMENTACIÓN Y LUZ

El hecho de que Moway lleve incorporado un sensor de luz, abre las puertas a una serie de aplicaciones. En cuatro de las cinco aplicaciones que se desarrollan en el capítulo 5, se emplean focos de luz. En cualquiera de los casos la elección de la fuente de alimentación y de luz ha de ser conjunta y adecuada.

Fuentes de alimentación

La elección de la fuente de alimentación a emplear suele ser fácil aunque de no hacerse correctamente puede provocar que la aplicación no se desarrolle como debería.

Generalmente una fuente de alimentación suele definirse por su voltaje e intensidad o intensidad dada en una hora.

Figura 106: Tipos de fuentes de alimentación

Fuentes de luz

Lo que se intenta buscar para que Moway sea capaz de detectar el foco de luz, es una fuente que emita luz en todas direcciones, es decir que no sea direccional y que lo haga con la intensidad suficiente.

Figura 107. Tipos de fuentes de luz

El primer elemento de la figura 107, es un LED de bajo consumo, que emite luz blanca muy direccional. Al ser direccional, se nos presentaba el problema de que Moway no era capaz de detectarlo, a menos que diera la casualidad de que el haz estuviera impactando directamente sobre el sensor de luz, lo cual ocurría en raras ocasiones.

El segundo elemento es una pequeña bombilla, que funciona con un voltaje entre 3 y 4.5V. Variando la intensidad de luz que proporciona en función de la intensidad que le esté aportando la pila.

El tercer elemento es un LED que emite luz azul con una amplitud de haz de 120º. El problema es que necesita una fuente de alimentación potente. De las vistas en la figura 106, sólo se podría emplear la pila de petaca y no conseguiría que brillase con toda la intensidad posible.

En las aplicaciones que se desarrollan en el capítulo 5 se emplean los focos de luz a modo de balizas luminosas y objetos luminosos que se desplacen al ser empujados por el microbot.

Baliza luminosa

En las aplicaciones en las que se emplee una baliza luminosa, el tamaño de la pila no es relevante puesto que la baliza permanece en un sitio fijo sin moverse. En este caso hemos optado por la bombilla alimentada por la pila de petaca. La baliza creada se describe en el apartado 5.1.1.

Objeto luminoso

En el caso de dotar de luz a objetos a los que Moway debe empujar, un elemento que nos limita en gran medida es el peso de la fuente de alimentación.

Ante la imposibilidad de emplear la pila de petaca por su elevado peso, empleamos la pila de litio de 3V, que aparece en tercer lugar en la figura 107. Esta pila, la de más alto coste, consigue encender la bombilla sin hacer que el objeto a empujar por el microbot sea excesivamente pesado. Este objeto se describe en el apartado 5.4.2.

CAPÍTULO 4

PROGRAMACIÓN GRÁFICA

CAPÍTULO 4: PROGRAMACIÓN GRÁFICA

4.1. MOWAYGUI

MowayGUI (MowayGraphicUserInterface) es un compilador gráfico basado en diagramas de flujo con el que se puede programar aplicaciones en el robot Moway intuitivamente. Diferentes bloques representan los sensores y actuadores del robot que son conectados mediante flechas creando así el programa deseado.

Con esta herramienta se pueden desarrollar aplicaciones para el robot Moway sin necesidad de conocimientos de programación en C o en ensamblador.

El programa se encuentra en fase Beta por lo que aún no tiene todas las funciones implementadas ni tampoco está disponible el manual completo. Algunos de los defectos o deficiencias encontrados son los siguientes:

- A la hora de operar con variables, lo único que se permite es sumar o restar un valor o variable a otra variable.
- El número máximo de subrutinas que se pueden emplear es 7, debido al tipo de microcontrolador incorporado a Moway.
- El comando de movimiento rectilíneo, sólo permite introducir unidades enteras, por lo que la distancia recorrida será en distancia múltiplo de 1.7mm (distancia=1.7*N mm) y en tiempo múltiplo de 100ms (tiempo=100*N ms)
- En el módulo de rotación, en el campo dirección, sólo se puede indicar derecha o izquierda, nunca una variable, por lo en muchos casos es necesario duplicar código.
- En el código generado automáticamente por MowayGUI aparecen errores de programación, que en el caso de no tener conocimientos de programación, no podrán ser subsanados por el usuario. Por ejemplo, al introducir un módulo “Pause” en el programa, se genera un error en el código ensamblador creado automáticamente, que nos indica que la variable “AUX_00” no está definida.

Estas son algunas de las razones que hacen que al programar con MowayGUI, veamos reducidas muchas de las posibilidades que nos aportan los lenguajes de programación C o ensamblador y a las que un programador está acostumbrado, por lo que se hace más difícil la programación actualmente, aunque con toda seguridad pronto se dispondrá de una versión mejorada.

4.1.1. Tipos de bloques

Al abrir MowayGUI, nos aparece la ventana de la figura:

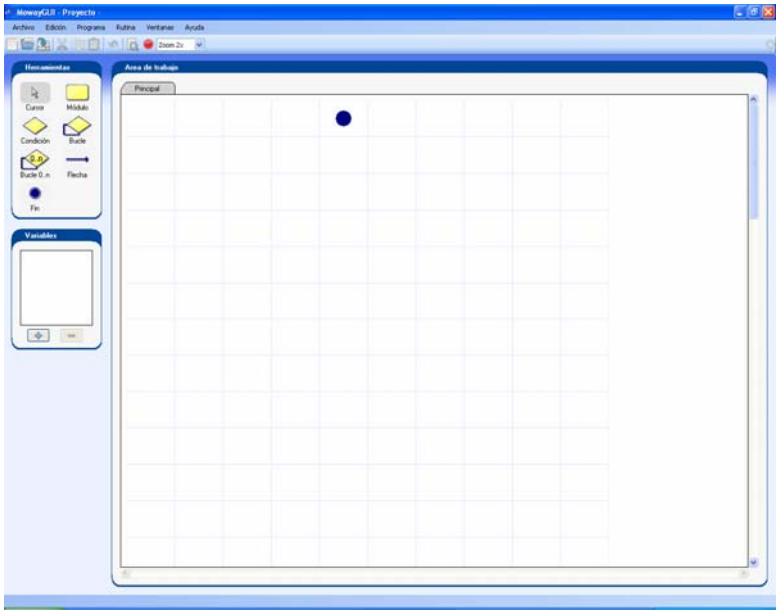

Figura 108: Ventana MowayGUI

Como se puede observar en la parte izquierda del programa, existen una serie de elementos que al combinarlos adecuadamente en distintas subrutinas, generarán el programa gráfico. Estos elementos se dividen en: Módulos, Condicionales, Flechas y elementos de Inicio y Final.

Módulos

Los módulos corresponden a acciones en las que la salida no es crítica. Los módulos existentes son los siguientes: Asignación, Captura sensores, Consulta movimiento, Movimiento (rectilíneo), Curva, Rotación, Reset datos movimiento, Control RF, Matemático, Diodos LED, Pausa y Llamada a subrutina, configurados según funcionalidad.

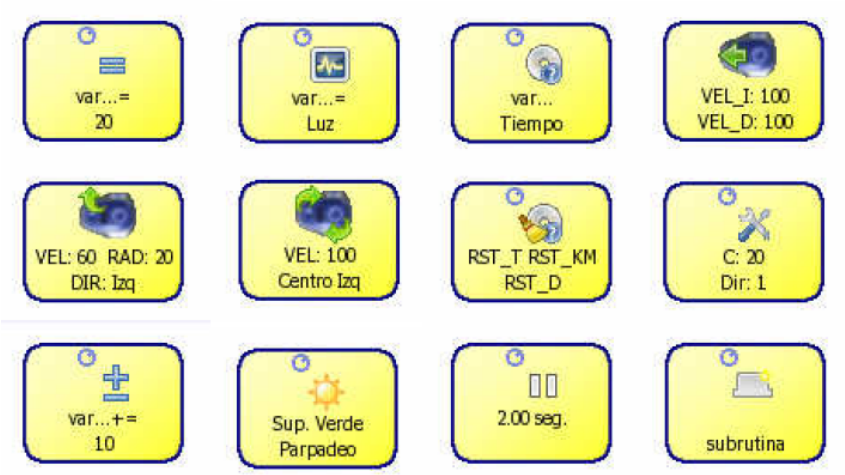

Figura 109: Diferentes Módulos

Figura 110: Elección tipo de Módulo

Condicionales

Los condicionales son acciones en las que la línea de código a ejecutar a continuación vendrá determinada por la salida de dicho bloque. Con estos bloques se pueden realizar las siguientes acciones: Comparativa, Comparativa sensores, Comparativa movimiento, Transmitir RF, Recepción RF, Chequear obstáculo y Chequear línea.

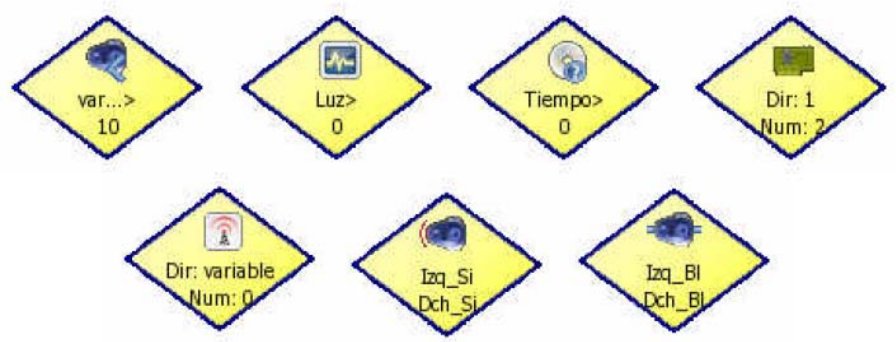

Figura 111: Condicionales

Figura 112: Elección tipo de Condicional

Flechas

Las flechas unen bloques entre sí (Módulos, Condicionales y elementos de Inicio y Fin) para crear así el diagrama de flujo del programa.

Inicio y Final

Todos los programas han de tener un elemento de Inicio, pero no es obligatorio que tengan uno de final (se puede crear un bucle infinito). El elemento inicial puede inicializar las variables.

Subrutinas

Para simplificar los diagramas y optimizar la utilización de la memoria de programa se pueden generar subrutinas reutilizables. Esto es, si en el programa se tiene una parte que se repite con mucha frecuencia, se puede crear una subrutina con esta tarea y sustituirla en el diagrama principal por un Módulo subrutina.

El número máximo de subrutinas es 7, limitado por el tipo de microcontrolador.

4.1.2. Obtención código fuente

Una vez creado el diagrama de flujo correctamente, nos vamos a "Ver fuente" en la pestaña Programa.

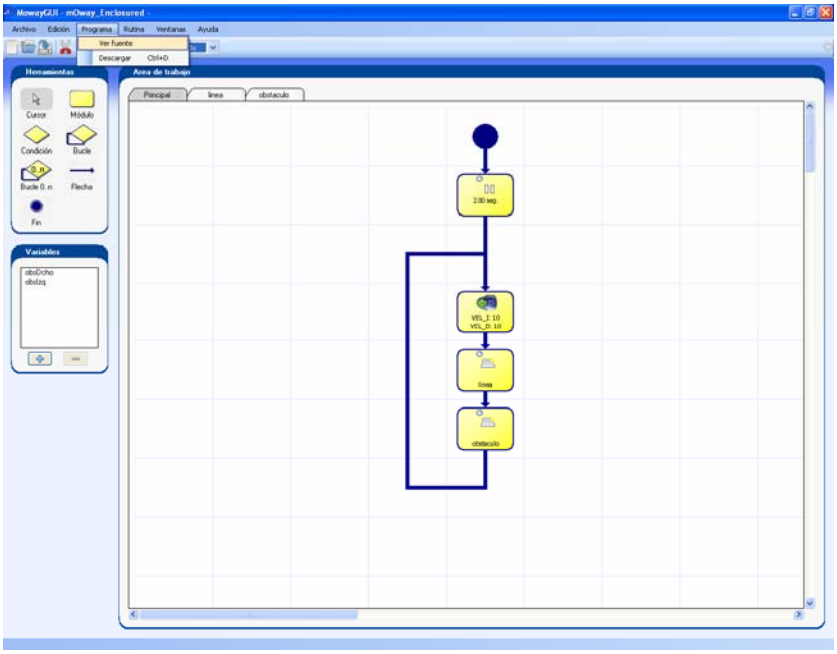

Figura 113: Diagrama de flujo

Se genera automáticamente el código en ensamblador a partir del diagrama de flujo. Dicho código se muestra en la figura 114.

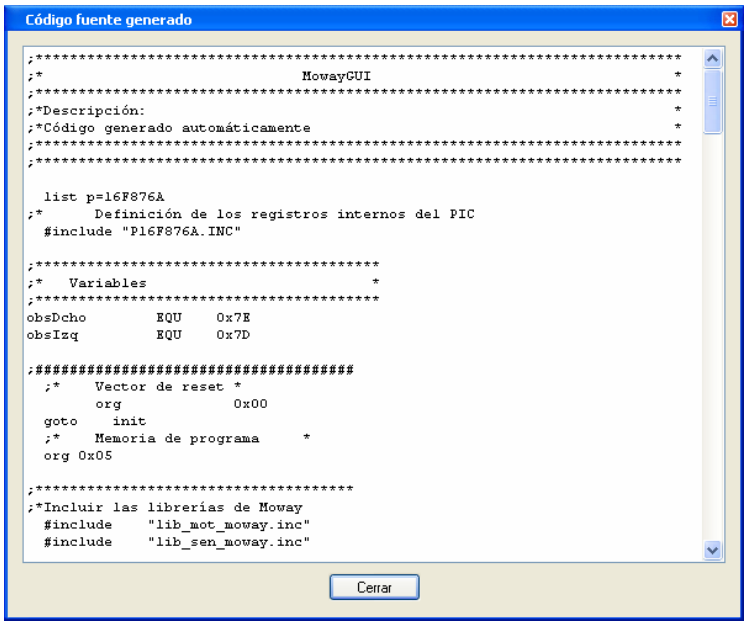

Figura 114: Código en ensamblador generado

Con este código fuente, nos iremos a MPLAB y generaremos el archivo .hex que grabaremos en el microcontrolador.

4.2. PROGRAMA EJEMPLO

Desde la página oficial de Moway se puede acceder a ejemplos muy simples de programación con MowayGUI. En este apartado se va a describir uno de ellos, que se corresponde con la práctica 1, llamada "El encierro" anteriormente descrita en el apartado 1.3.1.

En esta práctica, Moway permanece en el interior de un recinto cerrado. Se desplaza rectilíneamente por el recinto hasta detectar línea con alguno de los dos sensores de línea, derecho o izquierdo. En ese momento, rota a la izquierda para evitar salirse del recinto.

El diagrama de flujo es muy sencillo. Dicho diagrama se muestra a continuación en la figura 115.

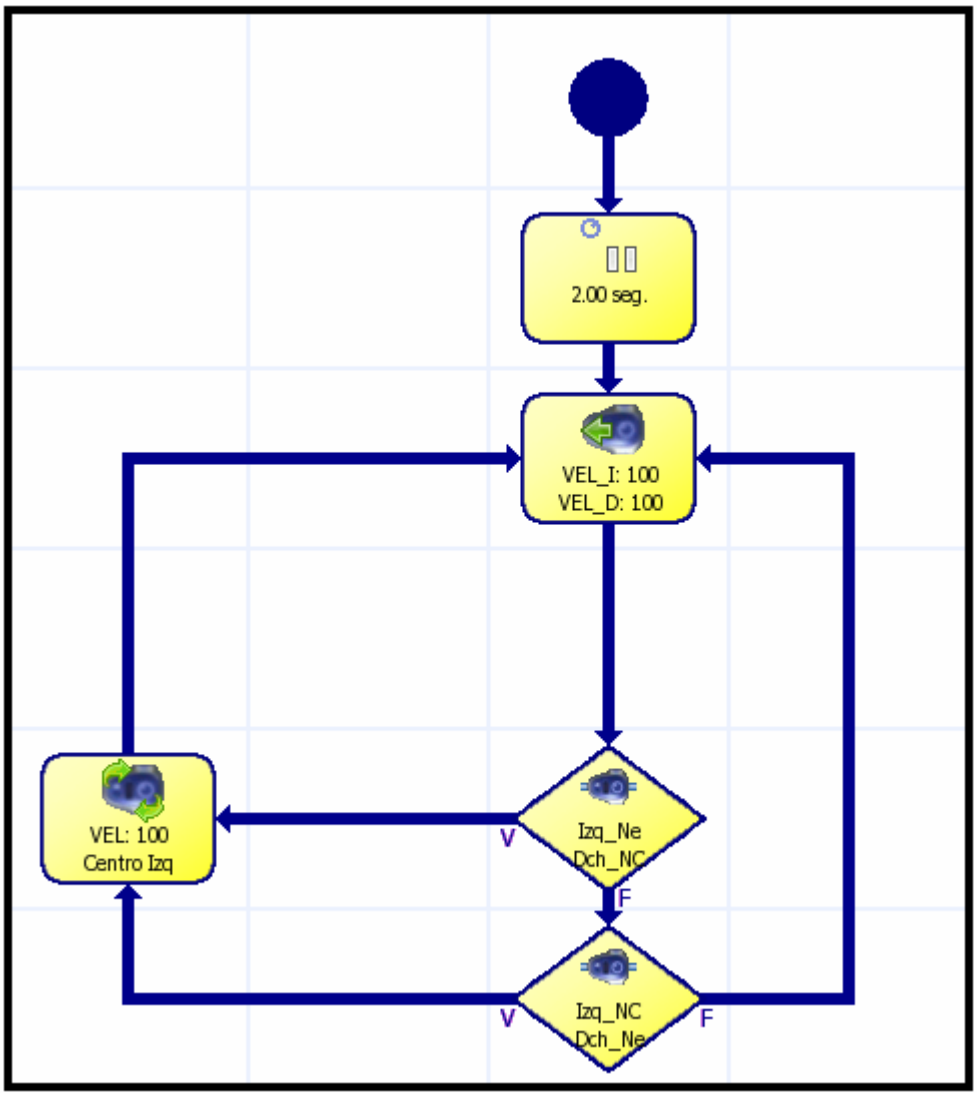

Figura 115: Subrutina Principal

Este programa sólo consta de una subrutina llamada Principal.

El primer bloque es un módulo llamado Pause, que retarda 2 segundos la ejecución del resto de programa.

A continuación, hay un bloque de Movimiento, el cual realiza indefinidamente un movimiento rectilíneo al 100% de la velocidad.

Un bloque Condicional chequea el sensor de línea izquierdo. En caso de detectar línea, se pasa a un Módulo de rotación a la izquierda. Si no se detecta línea se pasa a otro bloque Condicional para chequear el sensor de línea derecho. Y aquí como en el bloque anterior, si se detecta línea se rota y si no, se vuelve al Módulo inicial de movimiento rectilíneo.

4.3. PROGRAMA COMPLEJO

El programa elaborado en este apartado es bastante más complejo que el descrito en el apartado anterior.

Tal y como se vio en el apartado 2, cualquier tipo de movimiento tiene un error asociado. Al realizar sucesivos movimientos, ese error se va acumulando y el punto al que se llega tras varios comandos de movimiento, difiere en gran medida del punto al que se esperaba llegar.

En este programa, Moway realiza un recorrido cerrado. Al finalizar dicho recorrido, busca el punto inicial-final caracterizado por una baliza luminosa, al cual debería haber llegado en caso de no existir errores de trayectoria.

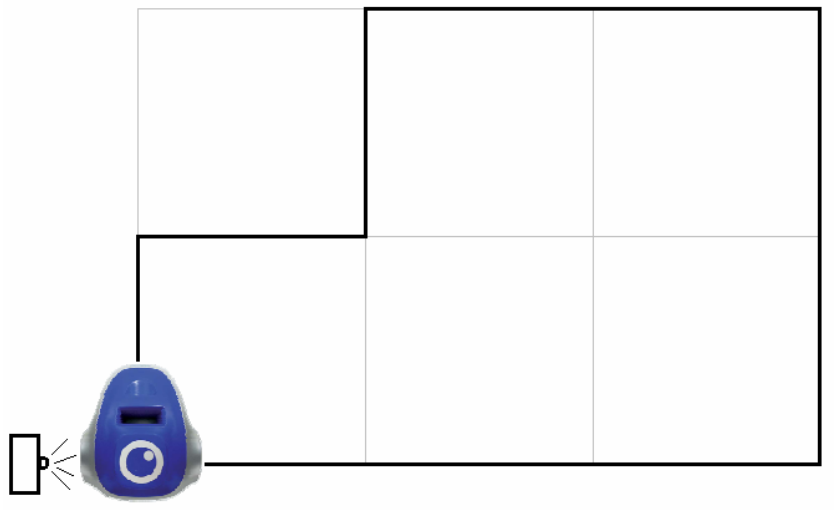

Figura 116: Recorrido a realizar. Cada unidad de cuadricula equivale a 103 mm.

El programa gráfico consta de 6 subrutinas.

Subrutina Principal

En la primera subrutina, llamada Principal, inicialmente se llama a un módulo de pausa de 4 segundos. Al finalizar este se llaman sucesivamente a las subrutinas Recorrido, Busqueda y Posicionar.

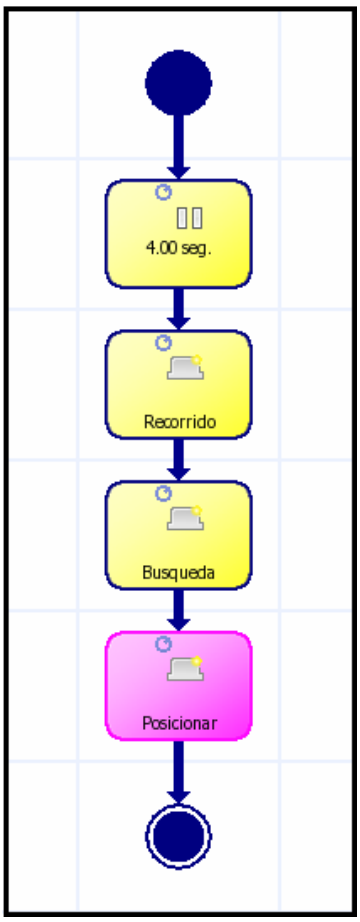

Figura 117: Subrutina Principal

Subrutina Recorrido

La subrutina Recorrido está formada por una serie de módulos de movimiento rectilíneo y rotatorio, que describen el recorrido que se muestra en la figura 116.

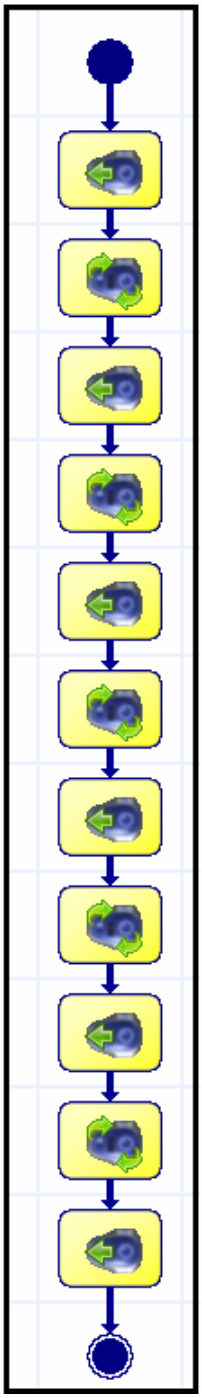

Figura 118: Subrutina Recorrido

Subrutina Busqueda

Una vez finalizado el recorrido, la subrutina Busqueda permite que el microbot se vaya acercando al foco luminoso. Finaliza su búsqueda, cuando Moway esté prácticamente perpendicular (con un margen de 5 entre los valores de los sensores de obstáculo) a la baliza luminosa.

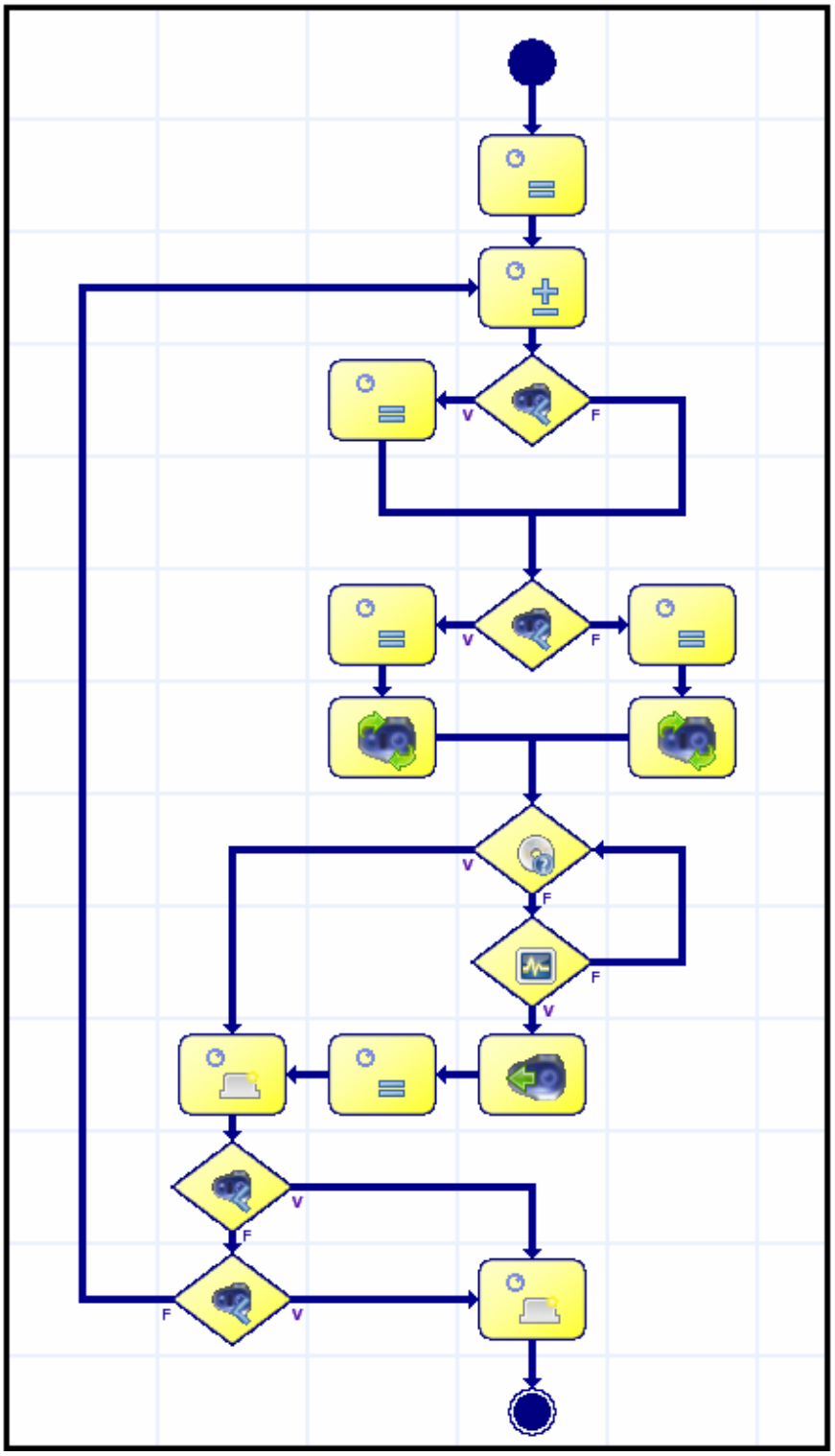

Figura 119: Subrutina Busqueda

Subrutina Obstaculo

La subrutina Obstaculo de lo único que se encarga es de asignar a las variables obsIzqdo y obsDcho los valores de los sensores de obstáculos izquierdo y derecho respectivamente.

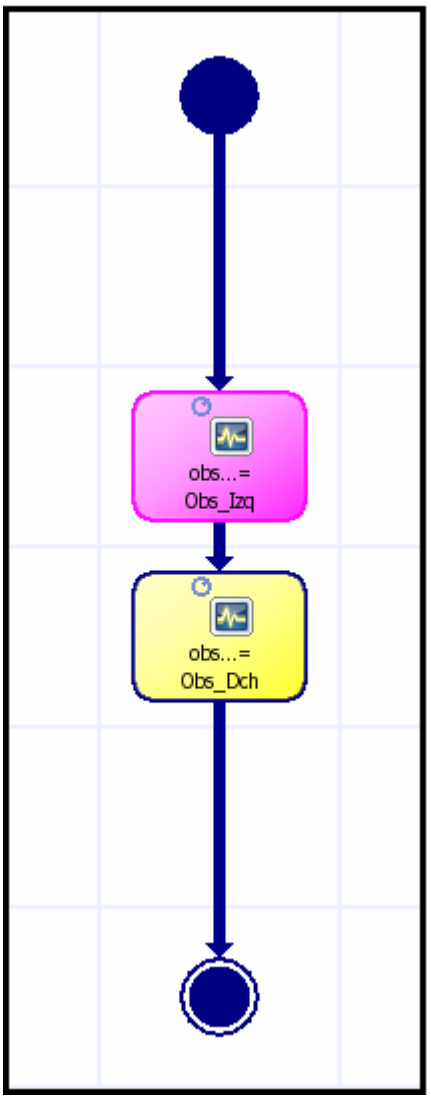

Figura 120: Subrutina Obstaculo

Subrutina CalculoAux

CalculoAux determina la diferencia entre los valores de los sensores de obstáculo en valor absoluto, y guarda ese valor en auxVar. Cuanto menor sea esta variable, significa que el microbot se encuentra más perpendicular a la baliza.

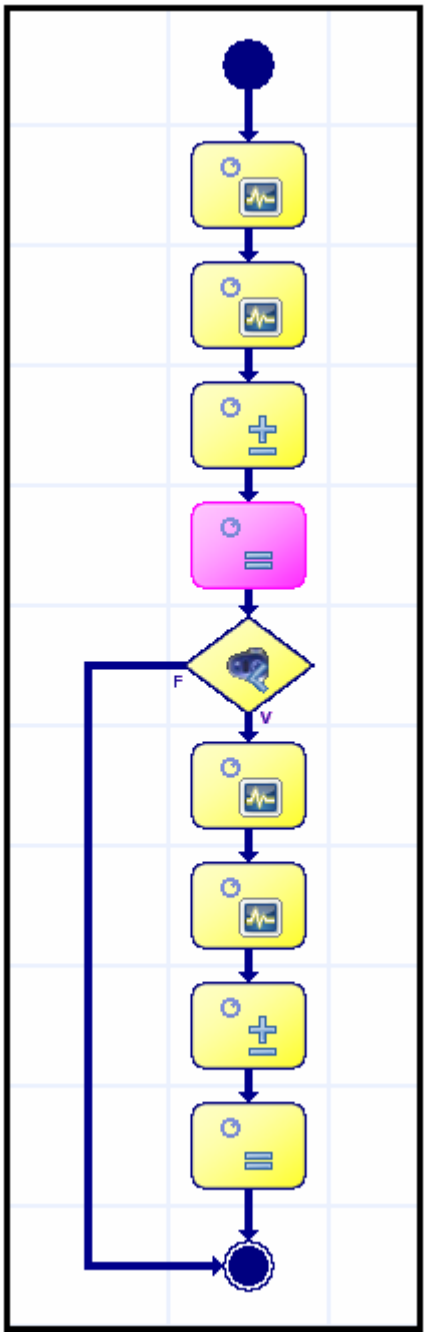

Figura 121: Subrutina CalculoAux

Subrutina Posicionar

Posicionar ejecuta comandos de movimiento según el caso, para conseguir que el microbot quede perpendicular a la baliza. No termina hasta que la diferencia entre los sensores de obstáculo no sea menor a margen, comprobando que ambos sensores indiquen valores mayores que 0 y menores que 255.

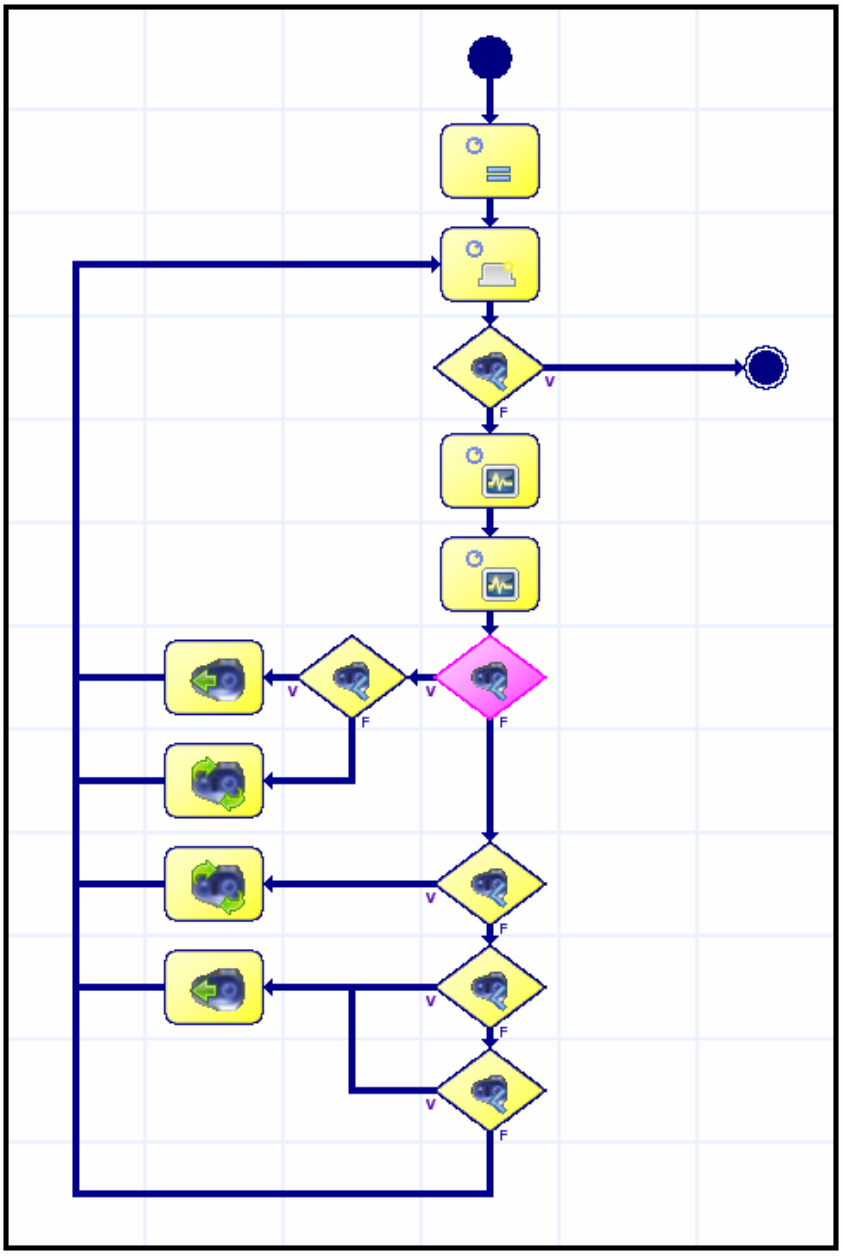

Figura 122: Subrutina Posicionar

Código en ensamblador generado

Al acceder a "Ver fuente", obtendremos el código en ensamblador de considerable tamaño (debido al hecho de que se genera automáticamente, el número de líneas de código es enorme). Se muestra a continuación (desde la página 116 a la 124).

```
;*********************************************************************
;*                      MowayGUI                                     *
;*********************************************************************
;*Descripción:                                                       *
;*Código generado automáticamente                                    *
;*********************************************************************
 list p=16F876A
;*        Definición de los registros internos del PIC
 #include "P16F876A.INC"
;**************************************
;*  Variables                         *
;**************************************
rotAngle     EQU   0x7E
rotDir       EQU   0x7D
auxVar       EQU   0x7C
obsIzqdo     EQU   0x7B
obsDcho      EQU   0x7A
margen       EQU   0x79
lightValue   EQU   0x78

 ;*       Vector de reset *
          org                0x00
 goto   init
 ;*       Memoria de programa      *
 org 0x05

;***********************************
;*Incluir las librerías de Moway
 #include   "lib_mot_moway.inc"
 #include   "lib_sen_moway.inc"
;***********************************

init:

 ;************************[CONFIGURACIÓN DE MOWAY]**********************
 ;Se configuran los sensores
 call   SEN_CONFIG
 call   MOT_CONFIG

;************Module Pause*********************************************
 movlw  .80
 movwf  AUX_00
 decfsz AUX_00,F
 goto   $+2
 goto   $+3
 call   Delay_50ms
 goto   $-4
;*********************************************************************
;  LLamada a subrutina
   call   Recorrido

;  LLamada a subrutina
```

```
   call    Busqueda

;   LLamada a subrutina
   call    Posicionar

loopEnd:
  goto    loopEnd
```

Recorrido:

```
;************Module Move*************************************************

  movlw   .50
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .59
  movwf   MOT_T_DIST_ANG
  bsf     MOT_CON,FWDBACK
  call    MOT_STR
  btfss   MOT_END
  goto    $-1

  movlw   .10
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .25
  movwf   MOT_T_DIST_ANG
  movlw   0x01
  movwf   MOT_CENWHEEL
  bsf     MOT_CON,FWDBACK
  bsf     MOT_CON,RL
  call    MOT_ROT
  btfss   MOT_END
  goto    $-1
;************Module Move*************************************************

  movlw   .50
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .59
  movwf   MOT_T_DIST_ANG
  bsf     MOT_CON,FWDBACK
  call    MOT_STR
  btfss   MOT_END
  goto    $-1

  movlw   .10
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .25
  movwf   MOT_T_DIST_ANG
  movlw   0x01
  movwf   MOT_CENWHEEL
  bsf     MOT_CON,FWDBACK
  bcf     MOT_CON,RL
  call    MOT_ROT
  btfss   MOT_END
  goto    $-1
;************Module Move*************************************************

  movlw   .50
```

```
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .59
  movwf   MOT_T_DIST_ANG
  bsf     MOT_CON,FWDBACK
  call    MOT_STR
  btfss   MOT_END
  goto    $-1

  movlw   .10
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .25
  movwf   MOT_T_DIST_ANG
  movlw   0x01
  movwf   MOT_CENWHEEL
  bsf     MOT_CON,FWDBACK
  bsf     MOT_CON,RL
  call    MOT_ROT
  btfss   MOT_END
  goto    $-1
;************Module Move************************************************

  movlw   .50
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .118
  movwf   MOT_T_DIST_ANG
  bsf     MOT_CON,FWDBACK
  call    MOT_STR
  btfss   MOT_END
  goto    $-1

  movlw   .10
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .25
  movwf   MOT_T_DIST_ANG
  movlw   0x01
  movwf   MOT_CENWHEEL
  bsf     MOT_CON,FWDBACK
  bsf     MOT_CON,RL
  call    MOT_ROT
  btfss   MOT_END
  goto    $-1
;************Module Move************************************************

  movlw   .50
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .118
  movwf   MOT_T_DIST_ANG
  bsf     MOT_CON,FWDBACK
  call    MOT_STR
  btfss   MOT_END
  goto    $-1

  movlw   .10
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
```

```
 movlw   .25
 movwf   MOT_T_DIST_ANG
 movlw   0x01
 movwf   MOT_CENWHEEL
 bsf     MOT_CON,FWDBACK
 bsf     MOT_CON,RL
 call    MOT_ROT
 btfss   MOT_END
 goto    $-1
;************Module Move************************************************

 movlw   .50
 movwf   MOT_VEL
 bcf     MOT_CON,COMTYPE
 movlw   .177
 movwf   MOT_T_DIST_ANG
 bsf     MOT_CON,FWDBACK
 call    MOT_STR
 btfss   MOT_END
 goto    $-1

 return
```

Busqueda:

```
;************Module Assigned*****************************************

 movlw   .90
 movwf   lightValue

label0:
;************Module Maths*******************************************

 movlw   .20
 addwf   rotAngle,F

;************Module Comparative*************************************

 movfw   rotAngle
 addlw   .135
 btfss   STATUS,C
 goto    label1

;************Module Assigned*****************************************

 movlw   .20
 movwf   rotAngle

label1:
;************Module Comparative*************************************

 movfw   rotDir
 sublw   .0
 btfss   STATUS,Z
 goto    label2

;************Module Assigned*****************************************

 movlw   .1
 movwf   rotDir
```

```
 movlw   .50
 movwf   MOT_VEL
 bsf     MOT_CON,COMTYPE
 movlw   .0
 movwf   MOT_T_DIST_ANG
 movlw   0x01
 movwf   MOT_CENWHEEL
 bsf     MOT_CON,FWDBACK
 bcf     MOT_CON,RL
 call    MOT_ROT

label3:
;************Module MoveStatus*****************************************

 movlw   STATUS_A
 movwf   MOT_STATUS_COM
 call    MOT_FDBCK
 movfw   MOT_STATUS_DATA_0
 subwf   rotAngle,W
 btfsc   STATUS,C
 goto    label4

label5:
;   LLamada a subutina
    call    Obstaculo

;************Module Comparative*****************************************

 movfw   obsIzqdo
 addlw   .255
 btfss   STATUS,C
 goto    label6

label7:
;   LLamada a subutina
    call    Posicionar

 return

 goto    label1

label2:
;************Module Assigned*****************************************

 movlw   .0
 movwf   rotDir

 movlw   .50
 movwf   MOT_VEL
 bsf     MOT_CON,COMTYPE
 movlw   .0
 movwf   MOT_T_DIST_ANG
 movlw   0x01
 movwf   MOT_CENWHEEL
 bsf     MOT_CON,FWDBACK
 bsf     MOT_CON,RL
 call    MOT_ROT

 goto    label3
```

```
label4:
;************Module CompSensor*****************************************

 call   SEN_LIGHT
 movfw  SEN_LIGHT_P
 subwf  lightValue,W
 btfsc  STATUS,C
 goto   label3
;************Module Move***********************************************
 movlw  .50
 movwf  MOT_VEL
 bsf    MOT_CON,COMTYPE
 movlw  .1
 movwf  MOT_T_DIST_ANG
 bsf    MOT_CON,FWDBACK
 call   MOT_STR
 btfss  MOT_END
 goto   $-1
;************Module Assigned*******************************************

 movlw  .0
 movwf  rotAngle

 goto   label5

label6:
;************Module Comparative****************************************

 movfw  obsDcho
 addlw  .255
 btfss  STATUS,C
 goto   label0

 goto   label7

Obstaculo:
;************Module Capture********************************************

 call   SEN_OBS_ANALOG
 movfw  SEN_OBS_L
 movwf  obsIzqdo
;************Module Capture********************************************

 call   SEN_OBS_ANALOG
 movfw  SEN_OBS_R
 movwf  obsDcho

 return

CalculoAux:
;************Module Capture********************************************

 call   SEN_OBS_ANALOG
 movfw  SEN_OBS_L
 movwf  obsIzqdo
;************Module Capture********************************************

 call   SEN_OBS_ANALOG
 movfw  SEN_OBS_R
```

```
 movwf   obsDcho
;************Module Maths*********************************************

 movf    obsIzqdo,W
 subwf   obsDcho,F

;************Module Assigned******************************************

 movfw    obsDcho
 movwf   auxVar

;************Module Comparative***************************************

 movfw   auxVar
 addlw   .255
 btfsc   STATUS,C
 return

;************Module Capture*******************************************

 call    SEN_OBS_ANALOG
 movfw   SEN_OBS_L
 movwf   obsIzqdo
;************Module Capture*******************************************

 call    SEN_OBS_ANALOG
 movfw   SEN_OBS_R
 movwf   obsDcho

;************Module Maths*********************************************

 movf    obsDcho,W
 subwf   obsIzqdo,F

;************Module Assigned******************************************

 movfw    obsIzqdo
 movwf   auxVar

 return
```

Posicionar:

```
;************Module Assigned******************************************

 movlw   .5
 movwf   margen

label8:
;   LLamada a subutina
    call    CalculoAux

;************Module Comparative***************************************

 movfw   auxVar
 subwf   margen,W
 btfss   STATUS,C
 goto    label9

 return
```

```
label9:
;************Module Capture*******************************************

 call   SEN_OBS_ANALOG
 movfw  SEN_OBS_L
 movwf  obsIzqdo
;************Module Capture*******************************************

 call   SEN_OBS_ANALOG
 movfw  SEN_OBS_R
 movwf  obsDcho

;************Module Comparative***************************************

 movfw  obsIzqdo
 sublw  .0
 btfss  STATUS,Z
 goto   label10

;************Module Comparative***************************************

 movfw  obsDcho
 sublw  .0
 btfss  STATUS,Z
 goto   label11

;************Module Move**********************************************

 movlw  .10
 movwf  MOT_VEL
 bcf    MOT_CON,COMTYPE
 movlw  .1
 movwf  MOT_T_DIST_ANG
 bsf    MOT_CON,FWDBACK
 call   MOT_STR
 btfss  MOT_END
 goto   $-1

 goto   label8


label10:
;************Module Comparative***************************************

 movfw  obsDcho
 sublw  .0
 btfss  STATUS,Z
 goto   label12

 movlw  .10
 movwf  MOT_VEL
 bcf    MOT_CON,COMTYPE
 movlw  .1
 movwf  MOT_T_DIST_ANG
 movlw  0x01
 movwf  MOT_CENWHEEL
 bsf    MOT_CON,FWDBACK
 bcf    MOT_CON,RL
 call   MOT_ROT
 btfss  MOT_END
```

```
  goto    $-1

  goto    label8


label11:
  movlw   .10
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .1
  movwf   MOT_T_DIST_ANG
  movlw   0x01
  movwf   MOT_CENWHEEL
  bsf     MOT_CON,FWDBACK
  bsf     MOT_CON,RL
  call    MOT_ROT
  btfss   MOT_END
  goto    $-1

  goto    label8


label12:
;************Module Comparative*****************************************

  movfw   obsIzqdo
  addlw   .0
  btfss   STATUS,C
  goto    label13

label14:
;************Module Move************************************************

  movlw   .10
  movwf   MOT_VEL
  bcf     MOT_CON,COMTYPE
  movlw   .1
  movwf   MOT_T_DIST_ANG
  bcf     MOT_CON,FWDBACK
  call    MOT_STR
  btfss   MOT_END
  goto    $-1

  goto    label8


label13:
;************Module Comparative*****************************************

  movfw   obsDcho
  addlw   .0
  btfss   STATUS,C
  goto    label8

  goto    label14


END
```

CAPÍTULO 5

APLICACIONES

CAPÍTULO 5: APLICACIONES

5.1. MOVIMIENTO RECTILÍNEO CON CORRECCIÓN DE TRAYECTORIA

Según las conclusiones a las que se pudo llegar en el capítulo 2 referente a la odometría, el tipo de errores de trayectoria que se dan en Moway no son constantes, por lo que son necesarias técnicas de posicionamiento absoluto a través de elementos externos. En este caso emplearemos una baliza luminosa para corregir la desviación producida al ejecutar una trayectoria rectilínea.

El microbot realiza un movimiento rectilíneo de 1 metro. Al finalizar el recorrido, Moway se encuentra en un punto distinto al que debería estar en el caso de que el recorrido fuese ideal. Para corregir esa desviación, situamos una baliza luminosa al final del recorrido.

5.1.1. Tipo de baliza

El tipo de baliza a utilizar es luminosa. Consiste en una caja de cartón, de la que sobresale de uno de los laterales una pequeña bombilla, la cual se enciende y se apaga a través de un interruptor situado en la parte superior. La altura de la bombilla es la adecuada para que Moway sea capaz de detectar su presencia. La fuente de alimentación que se emplea, es una pila de petaca situada en el interior de la caja.

Figura 123: Baliza luminosa

5.1.2. Descripción del programa de corrección de trayectoria

El programa que se muestra a continuación, sitúa al microbot en el punto exacto en el que debería de haber finalizado el recorrido, empleando para ello los sensores de luz y de obstáculo.

El programa se divide en dos partes totalmente distintas.

La primera de ellas consiste en recorrer rectilíneamente los primeros 75 centímetros del metro de longitud que se recorre en total, sin preocuparse de ningún otro aspecto.

```
//Se recorre 75 centímetros en línea recta. Este recorrido se realiza mediante tres comandos de
//movimiento de 25 centímetros cada uno
for(i=0;i<3;i++)
{
        MOT_STR(50,FWD,DISTANCE,2500/17);
        while(!input(MOT_END)){}
}
```

La segunda parte del programa consiste en buscar el foco luminoso y colocarse perpendicularmente a él. Para ello previamente se ha colocado la baliza en la posición adecuada. Se recorren los 25 centímetros que faltan para el metro ejecutando primero la función EncontrarFoco() y a continuación Perpendicularidad().

```
MOT_STR(50,FWD,DISTANCE,2500/17);
while(!input(MOT_END))
{
        //se busca el foco luminoso
        EncontrarFoco();
        //una vez situado cerca del foco luminoso se busca la perpendicularidad con la baliza.
        Perpendicularidad();
}
```

Inicialmente se llama a la función EncontrarFoco(), que consta de un algoritmo similar al que podíamos ver en el programa ejemplo Faro, para buscar el foco luminoso. La función entiende que ha encontrado la baliza, cuando detecta obstáculo y además el valor devuelto por el sensor de luz es superior a 85 (valor hallado experimentalmente).

```
//*********************************ENCONTRAR FOCO*********************************
// Se produce un acercamiento al foco. Se rotará a un lado y a otro, aumentando el ángulo de búsqueda,
//hasta que el valor del sensor luminoso de un valor mayor que el almacenado en lightValue. Se avanzará
//en esa dirección actualizando el valor de lightValue. Finalizará al detectar un valor de luminosidad
//mayor de 85 y detecta obstáculo.
void EncontrarFoco()
{
        lightValue=8;
        fin=false;
        while(fin==false)
        {
                rotDir=!rotDir;

                //aumenta el ángulo si no se detecta luz, para ampliar el campo de búsqueda
                rotAngle+=20;
                if(rotAngle>100)
                {
                        rotAngle=20;
                        //cuando se ha rotado sobre el mismo punto hasta rotAngle de 100, se rota para
                        //comenzar a buscar el foco en otra dirección
                        MOT_ROT(10,FWD,CENTER,RIGHT,ANGLE,25);
                        while(!input(MOT_END))
                        {
                                SEN_OBS_ANALOG();
                                //si esta muy cerca de la baliza, retrocede 3.4 mm
                                if(((SEN_OBS_R==255) || (SEN_OBS_L==255)) ||
((SEN_OBS_R==255) & (SEN_OBS_L==255)))
                                {
                                        MOT_STR(10,BACK,DISTANCE,2);
                                        while(!input(MOT_END)){}
```

```
                                        }
                                }

                        }

                        //Rotación hacia la izquierda o derecha (depende de rotDir) comprobando si encuentra
                        //la fuente de luz
                        MOT_ROT(10,FWD,CENTER,rotDir,ANGLE,rotAngle);
                        while(!input(MOT_END))
                        {
                                //se comprueba que no se esté chocando contra la baliza. En ese caso retrocede
                                SEN_OBS_ANALOG();
                                if(((SEN_OBS_R==255) || (SEN_OBS_L==255)) || ((SEN_OBS_R==255) &
(SEN_OBS_L==255)))
                                {
                                        MOT_STR(10,BACK,DISTANCE,2);
                                        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 100 ms comprobando el sensor de obstáculos
                                if(SEN_LIGHT_P>lightValue)
                                {
                                         MOT_STR(10,FWD,TIME,1);
                                         while(!input(MOT_END))
                                         {
                                                 SEN_OBS_ANALOG();
                                                 if(((SEN_OBS_R==255) || (SEN_OBS_L==255)) ||
((SEN_OBS_R==255) & (SEN_OBS_L==255)))
                                                 {
                                                         MOT_STR(10,BACK,DISTANCE,2);
                                                         while(!input(MOT_END)){}
                                                 }
                                         }
                                         //se actualiza el valor de lightValue con el nuevo máximo encontrado
                                         lightValue=SEN_LIGHT_P;
                                          rotAngle=0;
                                         SEN_LIGHT();
                                         //en el momento que el sensor de luz detecte un valor superior a 85 y
                                         //que se detecte obstáculo, considera que ya ha encontrado foco
                                         if((DeteccionObstaculo()==true)&&(SEN_LIGHT_P>85))
                                                 fin=true;
                                 }
                        }
                }
}
```

Una vez detectada la baliza, se busca la perpendicularidad con ésta, utilizando para ello la función Perpendicularidad(), la cual llama a su vez a las funciones Auxiliar(), Comprobación() y Posicionar().

Perpendicularidad()

Esta función considera que el microbot está perpendicular a la baliza en el momento de que la variable aux, calculada como la diferencia entre el valor que devuelven los sensores de obstáculos en valor absoluto, sea inferior a un margen fijado, devolviendo los sensores de obstáculos valores distintos a 0 y 255.

```
//***************BUSCAR LA PERPENDICULARIDAD CON EL OBSTACULO***************
//Realiza las maniobras adecuada para conseguir situar a Moway perpendicular a la baliza. En el
//momento en que la diferencia entre el valor los sensores de obstáculos sea inferior a la variable margen
//y ninguno de ellos está dando un valor nulo ni máximo, se considerará en ese instante que se encuentra
//perpendicular al objeto en cuestión.
void perpendicularidad(void)
  {
          //se calcula aux
           Comprobacion();
          //mientras que la diferencia sea mayor que un margen fijado o alguno de los sensores sea 0 ó 255
          while((aux>margen)||(SEN_OBS_R==0)||(SEN_OBS_L==0)||(SEN_OBS_R==255)||(SEN_OBS
_L==255))
          {
                    //vuelve a comprobar del sensor de obstáculos
                   SEN_OBS_ANALOG();
                   if(((SEN_OBS_R==255) || (SEN_OBS_L==255)) || ((SEN_OBS_R==255) &
(SEN_OBS_L==255)))
                   {
                             MOT_STR(10,BACK,DISTANCE,2);
                             while(!input(MOT_END)){}
                   }
                   SEN_OBS_ANALOG();
                   Auxiliar();
                   //en función del sensor que este más cerca del obstáculo girará a derecha o izquierda
                   if(SEN_OBS_R > SEN_OBS_L)
                             direccion=RIGHT;
                   else
                             direccion=LEFT;

                    MOT_ROT(10,FWD,CENTER,direccion,TIME,1);
                    while(!input(MOT_END))
                   {
                             //mientras esta rotando comprueba los sensores y para cuando la diferencia
                             //entre ellos sea menor que el margen fijado
                             Comprobacion();
                              if(aux<margen)
                              {
                                       SEN_OBS_ANALOG();
                                       auxiliar();
                              }
                   }
                   //en el caso de que algunos de los sensores estén indicando el máximo o que los dos lo
                   //estén haciendo, retrocede 1.7mm
                    if(((SEN_OBS_R==255) || (SEN_OBS_L==255)) || ((SEN_OBS_R==255) &
(SEN_OBS_L==255)))
                   {
                             MOT_STR(20,BACK,DISTANCE,1);
                             while(!input(MOT_END)){}
                   }
                   Comprobacion();
          }
  }
```

Auxiliar()

Realiza las maniobras pertinentes empleando la función Posicionar(), en el caso de que alguno de los sensores de obstáculos deje de detectar la baliza.

```
//*******************************FUNCION AUXILIAR()*********************************
//En el caso de que uno de los sensores no este detectando obstáculo, se llevan a cabo unas maniobras
//para que Moway se desplace hacia el lado conveniente
void Auxiliar(void)
{
        //en el caso de que sólo uno de los sensores no detecte obstáculo supondremos
        //perpendicularidad al obstáculo y Moway se desplaza hacia el lado del sensor que si lo detecta
        if(((SEN_OBS_R==0) & (SEN_OBS_L!=0)) || ((SEN_OBS_R!=0) & (SEN_OBS_L==0)))
        {
                if(SEN_OBS_R==0)
                        giro=RIGHT;
                else
                        giro=LEFT;
                longitud=500/17;
                Posicionar();

                SEN_OBS_ANALOG();
                //si al desplazarse, ninguno de los sensores detecta ahora obstáculo, significa que no
                //estaba perpendicularmente al obstáculo y se debe desplazar en el sentido contrario al
                //que lo acabamos de hacer
                if((SEN_OBS_R==0) & (SEN_OBS_L==0))
                {
                        longitud=1000/17;
                        Posicionar();
                }
                SEN_LIGHT();
                //si no se encuentra justo enfrente de la baliza luminosa
                if(SEN_LIGHT_P<85)
                {
                        //se desplaza a un punto en el que se toma el valor del sensor de luz
                        MOT_STR(10,BACK,DISTANCE,5);
                        while(!input(MOT_END)){}
                        MOT_ROT(10,FWD,CENTER,LEFT,ANGLE,25);
                        while(!input(MOT_END)){}
                        MOT_STR(10,FWD,DISTANCE,2);
                        while(!input(MOT_END)){}
                        SEN_LIGHT();
                        lightLeft=SEN_LIGHT_P;
                        //se desplaza a otro punto en el que se toma el valor del sensor de luz
                        MOT_STR(10,BACK,DISTANCE,2);
                        while(!input(MOT_END)){}
                        MOT_ROT(10,FWD,CENTER,RIGHT,ANGLE,50);
                         while(!input(MOT_END)){}
                        MOT_STR(10,FWD,DISTANCE,2);
                        while(!input(MOT_END)){}
                        SEN_LIGHT();
                        lightRight=SEN_LIGHT_P;
                        //se comparan ambos valores
                        //si el valor mayor nos daba en el punto anterior, se regresa a él
                        if(lightRight<lightLeft)
                        {
                                MOT_STR(10,BACK,DISTANCE,2);
                                while(!input(MOT_END)){}
                                MOT_ROT(10,FWD,CENTER,LEFT,ANGLE,50);
```

```
                                        while(!input(MOT_END)){}
                                        MOT_STR(10,FWD,DISTANCE,2);
                                        while(!input(MOT_END)){}
                                }
                                //se llama a la función EncontrarFoco() para encontrar la baliza luminosa
                                EncontrarFoco();
                        }
                }
                //en el momento en que la diferencia entre los sensores de obstáculo sea menor que margen,
                //se supone que ya se encuentra totalmente perpendicular a la baliza
                Comprobacion();
                if(aux<margen)
                        MOT_STOP();
}
```

Comprobación()

Obtiene el valor analógico de los sensores de obstáculos y calcula aux como la diferencia en valor absoluto entre ambos valores.

```
//************************FUNCIÓN COMPROBACION**********************************
//Con esta función se obtienen los valores analógicos del sensor de obstáculos y se calcula el valor de la
//variable aux como la diferencia entre los valores de ambos sensores
static int8 Comprobacion(void)
{
        //se comprueba el sensor de obstáculos en modo analógico
        SEN_OBS_ANALOG();

        if(SEN_OBS_R>SEN_OBS_L)
                aux=SEN_OBS_R-SEN_OBS_L;
         else
                aux=SEN_OBS_L-SEN_OBS_R;
         return(aux);
}
```

Posicionar()

Se suceden una serie de maniobras para posicionar al microbot a lo largo de la paralela al obstáculo.

```
//*****************************FUNCIÓN POSICIONAR**********************************
//Se suceden una serie de maniobras para desplazar a Moway a lo largo de la paralela al obstáculo
void Posicionar(void)
{
        //rota un ángulo de 20º hacia el lado que indique giro
        MOT_ROT(10,FWD,CENTER,giro,ANGLE,50/9);
        while(!input(MOT_END))
        {
                SEN_OBS_ANALOG();
                //en el caso de que algunos de los sensores estén indicando el máximo o que los dos lo
                //estén haciendo, retrocede 1.7mm
                 if(((SEN_OBS_R==255) || (SEN_OBS_L==255)) || ((SEN_OBS_R==255) &
(SEN_OBS_L==255)))
                        MOT_STOP();
        }
        //movimiento rectilíneo hacia atrás
        MOT_STR(10,BACK,DISTANCE,longitud);
```

```
        while(!input(MOT_END)){}
        //se invierte el sentido de giro
        giro=!giro;
        //rota un ángulo de 20º hacia el lado que indique giro
        MOT_ROT(10,FWD,CENTER,giro,ANGLE,50/9);
        while(!input(MOT_END)){}
        //movimiento rectilíneo hacia delante
        MOT_STR(10,FWD,DISTANCE,longitud);
        while(!input(MOT_END))
        {
                SEN_OBS_ANALOG();
                //en el caso de que algunos de los sensores estén indicando el máximo o que los dos lo
                //estén haciendo, retrocede 1.7mm
                if(((SEN_OBS_R==255) || (SEN_OBS_L==255)) || ((SEN_OBS_R==255) &
(SEN_OBS_L==255)))
                {
                        MOT_STR(20,BACK,DISTANCE,1);
                        while(!input(MOT_END)){}
                }
        }
}
```

5.1.3. Modificación de la función BusquedaFoco()

En el programa que acabamos de comentar, se ha fijado un valor lightValue=8, el cual se va actualizando según el microbot se vaya acercando al foco. Cuando, éste valor sea superior a 85 significa que el foco ha sido encontrado.

Este valor, 85, es un valor hallado experimentalmente para este tipo de baliza. En el caso de que quisiéramos un programa válido para cualquier tipo de baliza, el procedimiento sería realizar una vuelta de 360º y obtener el máximo valor que nos da el sensor de luz. En este caso se tarda más tiempo en encontrar el foco luminoso. A continuación se muestra como quedaría la función BusquedaFoco() con el cambio:

```
//*************** BUSQUEDA DEL PUNTO INICIAL (FOCO DE LUZ) ******************
//Inicialmente da una vuelta entera, guardando el máximo de luz. Luego irá actualizando ese valor
//máximo, avanzando en dicha dirección hasta detectar obstáculo.
void BusquedaFoco(void)
{
        valorMax=0;
        luz=0;
        while(luz==0)
        {
                Delay_ms(2000);
                aux2=0;
                //rota 360º a la derecha guardando el máximo de luz
                MOT_ROT(10,FWD,CENTER,RIGHT,DISTANCE,100);
                while(!input(MOT_END))
                {
                        //Chequeo del estado del sensor de luz
                        SEN_LIGHT();
                        //se guarda el valor máximo que detecta el sensor de luz
                        if(SEN_LIGHT_P>valorMax)
                        {
                                valorMax=SEN_LIGHT_P;
                        }
                }
```

```
                Delay_ms(2000);
                //para evitar que rote indefinidamente al no encontrar un valor mayor que valorMax
                //introducimos la variable aux2
                while(aux2<9)
                {
                        aux2=aux2+1;
                        //Aumenta el ángulo si no se detecta luz para ampliar el campo de búsqueda
                        //alternando la dirección de giro
                        rotDir=!rotDir;
                        rotAngle+=20;
                        if(rotAngle>100)
                                rotAngle=20;
                        //Rotación a izquierda o derecha (depende de rotDir) comprobando si encuentra
                        //un valor superior o inferior a valorMax
                        MOT_ROT(5,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 valorMax, se mueve
                                //recto 100ms en esa dirección
                                if(SEN_LIGHT_P>=(valorMax))
                                {
                                        valorMax=SEN_LIGHT_P;
                                        MOT_STR(50,FWD,TIME,1);
                                        while(!input(MOT_END))
                                        {
                                                aux2=0;
                                                //Se comprueba los sensores de obstáculo
                                                SEN_OBS_DIG();
                                                //Si detecta obstáculo se detiene y sale de la función
                                                if(SEN_OBS_R==1 || SEN_OBS_L==1)
                                                {
                                                        MOT_STOP();
                                                        luz=1;
                                                        aux2=10;
                                                        valorMax=255;
                                                }
                                        }
                                        rotAngle=0;
                                }
                        }
                }
        }
}
```

5.2. SEGURIDAD MUSEO

Esta aplicación simula que el microbot fuese un guardia de seguridad de un museo. Cada cierto tiempo Moway se coloca perpendicularmente a la obra de arte que vigila, con la finalidad de comprobar que está en perfecto estado, en el caso de que se le acoplase algún tipo de cámara o dispositivo.

Moway se desplaza evitando la línea que delimita la sala y el pedestal sobre el que se sitúa la obra de arte, en este caso la Dama de Elche. Cada 10 segundos, busca el foco luminoso situado sobre el pedestal y se coloca perpendicularmente a él. Espera unos segundos y continúa su marcha.

Figura 124: Ambiente de trabajo de Moway en Seguridad museo

Éste, es un ejercicio completo en el que hay que trabajar con todos los sensores integrados en Moway: de línea, de obstáculos y de luz. A la hora de elaborar el código, se reutilizan varias de las funciones creadas en el apartado anterior (5.1).

El programa consta de 7 funciones auxiliares, las cuales son llamadas desde el main. Desde main se asigna inicialmente un valor de 100 a la variable tiempo y se llama a la función Avanza(), que permite recorrer la pista durante 10 segundos evitando los posibles obstáculos y sin salirse de la sala. Posteriormente al finalizar esos 10 segundos, las funciones EncontrarFoco() y Perpendicularidad() se encargan de acercarse al foco y situarse perpendicularmente a él. Todo este proceso se repite indefinidamente.

Función DeteccionObstaculo()

La función detecta si existe obstáculo. Devuelve true en caso de que alguno de los sensores dé un valor mayor que senObsMin, y false en caso contrario.

```
//**************************DETECCION OBSTACULO*********************************
//Función que indica si existe o no obstáculo
int1 DeteccionObstaculo()
{
	//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 Avanza()

Realiza un recorrido aleatorio compuesto por tramos rectilíneos durante cierto tiempo, según el valor asignado a la variable tiempo. Cuando detecte línea, rota hacia la izquierda si es el sensor derecho el que detecta línea y hacia la derecha si es el sensor izquierdo. Al detectar obstáculo (a través de la función DeteccionObstaculo()), lo esquiva rotando a la izquierda en el caso de que el sensor derecho esté más cerca del obstáculo y hacia la derecha en el caso de que sea el izquierdo el que esté más cerca.

```
//**********************************AVANZA*****************************************
//Realiza un recorrido aleatorio, compuesto por tramos rectilíneos. Al detectar línea u obstáculo en este
//tiempo, rota para evitarlos.
void Avanza()
{
	//movimiento rectilíneo
	MOT_STR(50,FWD,TIME,tiempo);
	while(!input(MOT_END))
	{
		//si se detecta obstáculo se rota
		if(DeteccionObstaculo()==true)
		{
			//se guarda el tiempo que ha estado ejecutando el comando de movimiento
			MOT_FDBCK(STATUS_T);
			//si el sensor derecho es el que está mas cerca del obstáculo se gira a la
			//izquierda para esquivarlo
			if(SEN_OBS_R>SEN_OBS_L)
			{
				MOT_ROT(10,FWD,CENTER,LEFT,DISTANCE,20);
				 while(!input(MOT_END)){}
			  }
			//si el sensor izquierdo es el que está mas cerca del obstáculo se gira a la
			//derecha para esquivarlo
			 if(SEN_OBS_R<SEN_OBS_L)
			 {

				MOT_ROT(10,FWD,CENTER,RIGHT,DISTANCE,20);
				while(!input(MOT_END)){}
			}
```

```
                        //se continúa con el movimiento rectilíneo refrescando la variable tiempo
                        tiempo=tiempo-MOT_STATUS_DATA_0;
                        MOT_STR(50,FWD,TIME,tiempo);
                }

                SEN_LINE_DIG();
                //si se detecta línea
                if((SEN_LINE_R==1)||(SEN_LINE_L==1))
                {
                        //se guarda el tiempo que ha estado ejecutando el comando de movimiento
                        MOT_FDBCK(STATUS_T);
                        linea=true;
                }
                //si se detecta línea por la derecha se rota hasta dejar de detectarla
                while(SEN_LINE_R==1)
                {
                        i=1;
                        MOT_ROT(10,FWD,CENTER,LEFT,DISTANCE,10*i);
                        while(!input(MOT_END)){}
                        SEN_LINE_DIG();
                        i++;
                 }
                //si se detecta línea por la izquierda se rota hasta dejar de detectarla
                while(SEN_LINE_L==1)
                {
                        i=1;
                        MOT_ROT(10,FWD,CENTER,RIGHT,DISTANCE,10*i);
                        while(!input(MOT_END)){}
                        SEN_LINE_DIG();
                        i++;
                }
                //se determina el tiempo que se tiene que seguir ejecutando el movimiento rectilíneo
                if(linea==true)
                  {
                        tiempo=tiempo-MOT_STATUS_DATA_0;
                        MOT_STR(50,FWD,TIME,tiempo);
                 }
                 linea=false;
        }
}
```

El código de las funciones que se describen a continuación no aparece por ser el mismo de las funciones del apartado 5.1.2 que llevan el mismo nombre. No obstante el código completo del programa se puede encontrar en el CD anexo.

Función EncontrarFoco()

Se rota a derecha e izquierda ampliando el ángulo de búsqueda del foco luminoso.

Al detectar un valor lumínico mayor que el guardado en lightValue, avanza en esa dirección, actualizando lightValue.

Finaliza la búsqueda al obtener un valor luminoso mayor o igual al que experimentalmente da el sensor cuando Moway se sitúa frente al foco (para el tipo de baliza empleado tomamos un valor de 85) y se detecte obstáculo (alguno de los sensores de obstáculo detecten la baliza).

Función Perpendicularidad()

Se ejecuta la función comprobación(), la cual obtiene el valor analógico de los sensores de obstáculos y calcula aux, como la diferencia entre los valores de dichos sensores.

Empleando la función auxiliar y los movimientos de rotación y movimiento rectilíneo, Moway se coloca prácticamente perpendicular a la baliza (ese prácticamente perpendicular implica que la diferencia entre el valor analógico del sensor de obstáculos derecho y el izquierdo no es superior a la variable margen fijada).

Función Comprobacion()

Con esta función se accede a los valores analógicos de los sensores de obstáculos y se calcula el valor de la variable aux como la diferencia entre los valores de ambos sensores.

Función Auxiliar()

Se llevan a cabo una serie de maniobras para que Moway se desplace hacia el lado conveniente en el caso de que uno de los sensores de obstáculos no esté detectando la baliza. Se emplea para ello la función Posicionar().

Función Posicionar()

Se suceden una serie de maniobras para desplazar a Moway a lo largo de la paralela al obstáculo en función de las variables giro y longitud.

5.3. GUÍA DE MUSEO

Un área importante en la que poder incorporar la ayuda de un microbot de forma satisfactoria es cultura y turismo. Existen muchas actividades que resultan monótonas y repetitivas a lo largo de una larga jornada de trabajo, como por ejemplo ir informando sobre obras de arte o edificios de interés cultural a un público que va llegando de forma discontinua.

A partir de esta idea, surge la aplicación que vamos a desarrollar en este apartado.

5.3.1. Desarrollo de la aplicación

Moway se convierte en guía de un museo. En la sala hay 4 obras de arte. A tres de ellas se dirige mediante seguimiento de una serie de líneas dibujadas en el suelo. Para acercarse a la cuarta obra de arte, emplea el sensor de luz, ya que en el pedestal de dicha obra se ha colocado un foco de luz a modo de baliza luminosa.

Al llegar a cada obra, espera unos segundos. Tiempo suficiente para explicar brevemente las características más importantes, en caso de poder incorporarle un sintetizador de voz.

Figura 125: Disposición del museo en el que Moway actuará como guía

5.3.2. Elaboración programa guía

Las dos funciones principales del código generado en esta aplicación, son las funciones de seguimiento de línea y la de búsqueda del foco luminoso, comentadas en los programas ejemplo “Rastreador” y “Faro” en los apartados 1.3.2 y 1.3.3 respectivamente, con alguna modificación.

En el caso de la función de seguimiento de línea, se mantiene siempre a Moway a la derecha de la línea, al contrario de lo que ocurría en el programa ejemplo en el que se realizaba un seguimiento por la izquierda de la línea. Ha sido modificada aunque se podría haber mantenido la función original.

```
//*************************SEGUIMIENTO DE LINEA*********************************
//se sigue la línea por la derecha de la línea
void SeguimientoLinea()
{
  //Se comprueba los sensores de línea
  SEN_LINE_DIG();

        // Si está en el borde derecho de la línea, sigue recto
        if(SEN_LINE_L==1 && SEN_LINE_R==0)
        {
                LED_L_ON();
                LED_R_OFF();
                MOT_STR(10,FWD,TIME,0);
                Delay_ms(10);
        }

        //Si no ve línea, gira a la izquierda
        else if(SEN_LINE_L==0 && SEN_LINE_R==0)
        {
                LED_L_OFF();
                LED_R_OFF();
                MOT_ROT(10,FWD,CENTER,LEFT,TIME,0);
                Delay_ms(10);
        }

        //Si está encima de la línea, gira a la derecha
        else if(SEN_LINE_L==1 && SEN_LINE_R==1)
        {
                LED_L_ON();
                LED_R_ON();
                MOT_ROT(10,FWD,CENTER,RIGHT,TIME,0);
                Delay_ms(10);
        }

        //Si está en el otro borde de la línea, gira a la derecha
        else if(SEN_LINE_L==0 && SEN_LINE_R==1)
        {
                LED_L_OFF();
                LED_R_ON();
                MOT_ROT(10,FWD,CENTER,RIGHT,TIME,0);
                Delay_ms(10);
        }
}
```

El valor de lightValue es elegido según el tipo de baliza luminosa escogida.

```
//********************ENCONTRAR FOCO()************************
//Esta función dirige al microbot hacia el foco luminoso
void EncontrarFoco()
{
	rotDir=!rotDir;

	//Aumenta el angulo si no se detecta luz para ampliar el campo de búsqueda
	rotAngle+=20;
	if(rotAngle>100)
		rotAngle=20;

	//Rotación hacia la izquierda o derecha (depende de rotDir) comprobando si
	//encuentra la fuente de luz
	MOT_ROT(10,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(10,FWD,TIME,1);
			while(!input(MOT_END)){}
			rotAngle=0;
		}
	}
}
```

Cada vez que se llama a alguna de estas funciones (tres veces a la función de seguimiento de línea y a continuación una a la de búsqueda del foco luminoso), se comprueban los sensores de obstáculos constantemente, y en el momento que se detecte obstáculo, se detiene y espera el tiempo suficiente para que el sintetizador de voz haga su trabajo.

```
void main()
{
	//Configuración de PIC para motores y sensores
	MOT_CONFIG();
	SEN_CONFIG();

	Delay_ms(7000);

	MOT_ROT(10,FWD,CENTER,RIGHT,ANGLE,25);
	while(!input(MOT_END)){}
	//tres repeticiones
	for(i=0;i<3;i++)
	{
		//avanza hasta que detecte línea
		MOT_STR(10,FWD,TIME,0);
		while(!input(MOT_END))
		{
			if(DeteccionLinea()==true)
				MOT_STOP();
		}
		//avanza 1.5 centímetros para rebasar la línea
```

```
                MOT_STR(10,FWD,DISTANCE,150/17);
                while(!input(MOT_END)){}
                Delay_ms(2000);
                SEN_OBS_ANALOG();
                //mientras no se detecte obstáculo (sensibilidad indicada en obsValue), seguir la línea
                while((SEN_OBS_R<obsValue)&&(SEN_OBS_L<obsValue))
                {
                        SeguimientoLinea();
                        SEN_OBS_ANALOG();
                }

                MOT_STOP();
                LED_L_OFF();
                LED_R_OFF();
                //espera el tiempo suficiente para explicar aquello que crea conveniente
                Delay_ms(3000);
                //rota 90º a la izquierda para ir a por la siguiente línea
                MOT_ROT(10,FWD,CENTER,LEFT,ANGLE,25);
                while(!input(MOT_END)){}
        }
        //se recorre la última trama de línea para acercarse un poco a la baliza luminosa
        MOT_STR(10,FWD,TIME,0);
        while(!input(MOT_END))
        {
                if(DeteccionLinea()==true)
                        MOT_STOP();
        }
        MOT_STR(10,FWD,DISTANCE,150/17);
        while(!input(MOT_END)){}
        Delay_ms(2000);
        for(i=0;i<250;i++)
        {
                SeguimientoLinea();
        }
        MOT_ROT(10,FWD,CENTER,LEFT,ANGLE,20);
        while(!input(MOT_END)){}
        MOT_STR(10,FWD,DISTANCE,1000/17);
        while(!input(MOT_END)){}
        MOT_ROT(10,FWD,CENTER,LEFT,ANGLE,20);
        while(!input(MOT_END)){}
        //se busca el foco luminoso hasta encontrar el pedestal (obstáculo)
        while(DeteccionObstaculo()==false)
        {
                EncontrarFoco();
        }
        MOT_STOP();
        LED_L_OFF();
        LED_R_OFF();
        Delay_ms(4000);

}
```

5.4. SEPARADOR DE OBJETOS

En esta aplicación, Moway recorre aleatoriamente la pista, sin salirse de ella. Al detectar un obstáculo, lo empuja hasta una u otra portería, en función del tipo de obstáculo de que se trate: si es luminoso hacia la zona más clara y si no lo es, hacia la zona más oscura.

Esta aplicación es bastante completa. Se centra en el uso de los sensores de línea en modo analógico para distinguir la franja (tono de gris) en la que se encuentra y del sensor de luz, para distinguir qué tipo de objeto es el que ha detectado Moway: luminoso o no luminoso.

5.4.1. Campo de trabajo

Tal y como vimos en el apartado 3.2.2 relativo a la sensibilidad espectral del sensor de línea para tonalidades de gris, Moway 02 es capaz de distinguir perfectamente 6 tonalidades distintas de gris.

El procedimiento que seguimos para obtener esas 6 tonalidades distinguibles, fue obtener por pantalla a través de radiofrecuencia, los valores que adopta cada sensor de línea (derecho e izquierdo) para un amplio abanico de tonalidades, y elegir de entre ellas 6 tonalidades.

En la tabla 28, se hizo la elección de 6 tonos de gris caracterizados por su luminancia. Para el campo que utilizamos en esta aplicación, no tomaremos justo esos valores sino otros que sean válidos tanto para Moway 02 como para Moway 01. Los valores de luminancia de cada franja son: 255, 200, 155, 110 y 60 de la más clara a la más oscura y de 0 para la línea que rodea el campo.

El campo está insertado en una caja, con una abertura en cada extremo a modo de porterías.

En las esquinas se han colocado trozos de poliestireno expandido, para evitar que alguno de los objetos se quede en una esquina y sea más difícil dirigirlo a la portería correspondiente.

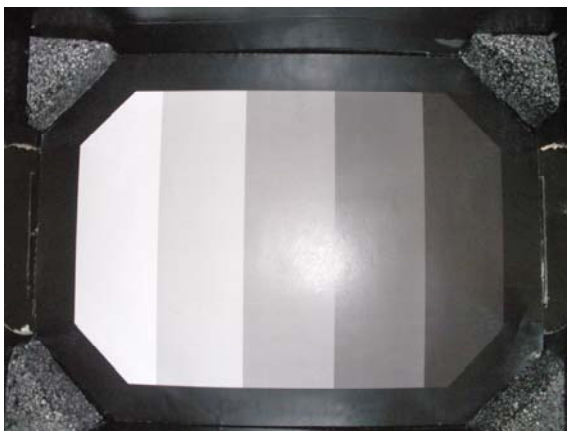

Figura 126

5.4.2. Obstáculos

Se va a trabajar con dos tipos de obstáculos: luminosos y no luminosos. Ambos son del mismo tamaño (unos 5cm. de diámetro) y cilíndricos.

Figura 127: Objeto luminoso y no luminoso

Los obstáculos van rodeados por una cinta blanca, ya que como se indicó en el apartado 3.3, cuando el objeto es claro lo detectan mejor los sensores de obstáculos.

El obstáculo luminoso, consta de una pequeña bombilla, un interruptor para encender y apagar el mecanismo, una pila de litio de 3V y cables para el conexionado de los distintos elementos.

5.4.3. Elaboración programa separador

El programa consta de 8 funciones que son llamadas desde el main o desde otras funciones (realmente son 9 funciones, ya que una de ellas, CalculoDireccion(), ha sido desdoblada para su mejor comprensión).

Los tres aspectos fundamentales que debemos destacar de este programa son: la detección y elección del tipo de objeto del que se trata (luminoso o no luminoso), el cálculo de la dirección en la que se está desplazando Moway y la implementación de la función de ataque.

Detección y elección del tipo de objeto

Inicialmente se lee el sensor de luz. Si nos da un valor elevado (superior a valorFoco) y no se detecta obstáculo, busca el objeto con la función BusquedaFoco().

La función DeteccionObstaculo(), indica si existe o no obstáculo (detecta obstáculo cuando alguno de los sensores de obstáculo esté dando un valor analógico superior a la constante fijada senObsMin).

En el momento que se detecta obstáculo, se lee el valor del sensor luminoso y la dirección que lleva el robot y se ataca o se esquiva el objeto en función de los datos que se obtengan.

```
if(((SEN_LIGHT_P>valorFoco)&(direccion==false))||((SEN_LIGHT_P<valorFoco)&(direccion==true)))
{
        Ataque();
}
//si no se dan ninguno de los dos casos anteriores se esquiva el objeto
else
{
        if(SEN_OBS_L>SEN_OBS_R)
        {
                //si el sensor de obstáculos izquierdo está más cerca del obstáculo, gira a la derecha
                MOT_ROT(10,FWD,CENTER,RIGHT,ANGLE,20);
                while(!input(MOT_END)){}

        }
        else
        {
                //si el sensor de obstáculos derecho está  más cerca del obstáculo, gira a la izquierda
                MOT_ROT(10,FWD,CENTER,LEFT,ANGLE,20);
                while(!input(MOT_END)){}

        }
}
```

Dirección de desplazamiento

Obtenemos la dirección de desplazamiento a través de la función CalculoDireccion(), desde donde se llaman a las funciones oportunas.

Las funciones ZonaSensorL() y ZonaSensorR() asignan a sensorL y sensorR respectivamente un valor comprendido entre 1 y 5 en función de la zona que detecta cada sensor.

Con esta información se asigna el valor a zonaActual. Dicho valor está muy claro cuando sensorL y sensorR son iguales y no tan claro cuando son diferentes. En este último caso, en función de la dirección anteriormente calculada y los valores sensorL y sensorR, el microbot rota un máximo de 36º para intentar que ambos sensores estén en la misma zona y asignar el valor correcto a la variable zonaActual.

En función de zonaActual y zonaAnterior se calcula la dirección (zonaActual se convierte en zonaAnterior al final de la función CalculoDireccion()).

```
//******************CALCULO DE LA DIRECCION**********************************
//Esta función nos da la dirección que lleva el robot en cada momento. Considera como TRUE cuando va
//de MAS CLARO A MAS OSCURO y como FALSE cuando va de MAS OSCURO A MAS CLARO.
void CalculoDireccion()
{
        //calcula el valor de zonaActual
        Auxiliar();
        //si va de más claro a más oscuro
        if(zonaAnterior < zonaActual )
                direccion=true;
        //si va de más oscuro a más claro
        if(zonaAnterior > zonaActual)
                direccion=false;
        zonaAnterior=zonaActual;
}
```

Función de ataque

Mientras que se sigan cumpliendo las condiciones en las cuales se llamo a la función Ataque() (condiciones de tipo de objeto, dirección y que se siga detectando obstáculo) se empuja al obstáculo girando a izquierda, luego a la derecha y finalmente avanzando 3.4mm hacia delante.

El ángulo de giro es de 7.2º o de 10.8º en función de la inclinación con la que se está recorriendo la pista. Dicha inclinación se puede saber en los casos en los que los dos sensores no entran a la misma franja al mismo tiempo.

```
//Si el obstáculo está en la izquierda
if(SEN_OBS_L>SEN_OBS_R)
{
	//Rota anguloL a la izquierda
	MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,anguloL);
	while(!input(MOT_END)){}
	if(DeteccionObstaculo()==false)
	{
		MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,2);
		while(!input(MOT_END)){}
	}
}
else
{
	//Rota anguloR a la derecha
	MOT_ROT(100,FWD,CENTER,RIGHT,ANGLE,anguloR);
	while(!input(MOT_END)){}
	//para evitar que el robot deje de empujar debido a que ha dejado de ver el obstáculo al girar
	if(DeteccionObstaculo()==false)
	{
		MOT_ROT(100,FWD,CENTER,LEFT,ANGLE,2);
		while(!input(MOT_END)){}
	}
}
MOT_STR(100,FWD,DISTANCE,2);
while(!input(MOT_END)){}
CalculoDireccion();
```

5.5. JUEGO MESA DE AIRE

Se ha denominado “Mesa de aire” a esta aplicación porque su principio de funcionamiento es idéntico a las mesas de aire recreativas.

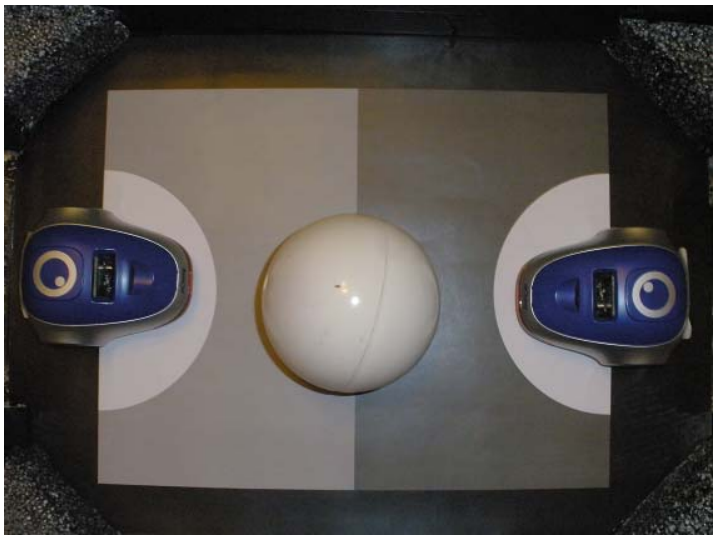

Figura 128: Aspecto de la aplicación Mesa de Aire

El terreno de juego se compone de dos campos, cada uno con una portería. En cada portería se sitúa un microbot, cuyo cometido es impedir que le metan gol y llevar el balón lo más lejos posible de su portería para marcar gol en la portería contraria.

5.5.1. Campo de juego

El terreno de juego se divide en dos campos de tono de gris diferente, rodeados por una franja negra que delimita el conjunto. Un semicírculo blanco situado en ambos extremos, señaliza la zona entorno a la que Moway debe mantenerse para proteger correctamente su portería.

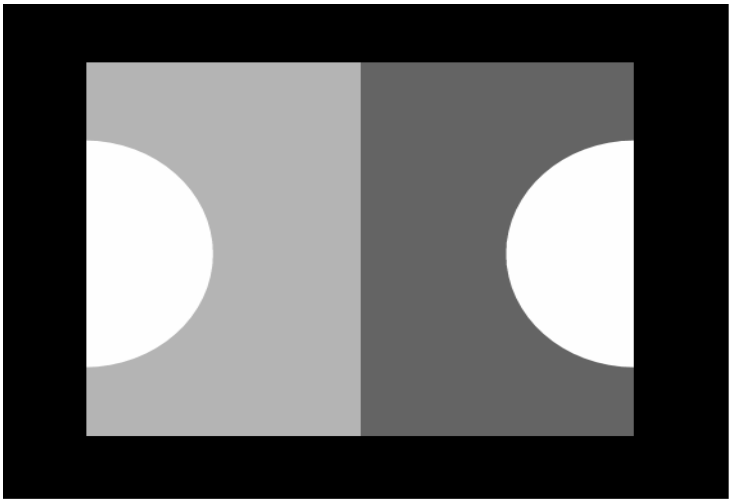

Figura 129: Terreno de juego

El terreno de juego con el cual se va a trabajar, se compone de cuatro tonos de gris perfectamente distinguibles para los microbots, de luminancias 100 y 180 para los campos, 255 para el área cercana a la portería y 0 para la línea que rodea el campo. De esta forma Moway es capaz de reconocer y diferenciar su campo del campo contrario, la zona cercana a la portería y los extremos que delimitan el campo.

Al igual que en aplicaciones anteriores, situamos el campo en el interior de una caja de cartón que impide que el balón se salga por los laterales y permite que si lo haga por los extremos en los que se sitúan las porterías.

5.5.2. Balón de juego

Inicialmente se consideró la posibilidad de emplear como balón el juguete de plástico que se muestra en las figuras 130 y 131.

Figura 130: Aspecto exterior del posible balón

Figura 131: Sistema interior del posible balón

Lleva incorporado un sistema alimentado por dos pilas de 1.5V, el cual aspira aire por la parte superior gracias a una rueda aleteada y lo expulsa a través de una serie de rendijas situadas en la parte inferior en contacto con la superficie. Este sistema permite que el balón se deslice mejor sobre la superficie necesitando un menor esfuerzo para desplazarlo.

El problema que surge con este tipo de obstáculo, es que a pesar del tipo de sistema que lleva integrado, no consigue recorrer una distancia suficiente para introducirse en la portería contraria sin invadir considerablemente el campo contrario, a partir del golpe de embestida del microbot. Es decir, que gracias al sistema incorporado, con un pequeño

golpe se mueve, pero recorre una distancia muy pequeña antes de pararse, por lo que finalmente no se emplea como balón de juego.

En este caso se opta por una simple esfera de plástico hueca de unos 9 centímetros de diámetro, la cual ha sido pintada de blanco, ya que inicialmente era transparente y no era detectada por los sensores de obstáculos.

Figura 132: Balón escogido

5.5.3. Elaboración programa de juego

Antes de comenzar a programar, se debe tener muy claro los valores que devuelven los sensores de línea para cada tono de gris presente en el terreno de juego. Para ello el microbot recorre la pista, enviando por radiofrecuencia los valores que devuelve cada uno de los sensores en distintos instantes. A partir de ellos se definen los valores máximo y mínimo para cada campo. Una vez definidos, el robot es capaz de distinguir en que campo se encuentra.

A continuación se muestran el máximo y mínimo definido para el campo de la figura 129.

```
//Declaración de variables
static    int8 minCampo1=32;//20 en hexadecimal
static    int8 maxCampo1=112;//70
static    int8 minCampo2=113;//71
static    int8 maxCampo2=183;//B7
```

Asignar campo

Inicialmente, el microbot es situado en su campo, fuera del área de portería. Gracias a la función AsignarCampo(), determina cuál es su campo (siendo el campo 1 el más claro y el campo 2 el más oscuro) y posteriormente se sitúa en el área de portería para dar comienzo al juego.

```
//*****************************ASIGNAR CAMPO************************************
//Se obtiene el valor de los sensores de línea para determinar que campo se le ha asignado al microbot
void AsignarCampo()
{
        //mientras que no se realice correctamente la asignación de campo
        while(asignacion==false)
        {
                SEN_LINE_ANALOG();
                //si ambos sensores están comprendidos entre los valores asignados al campo 1,
                //campo toma valor 1
        if((SEN_LINE_R>minCampo1)&&(SEN_LINE_R<maxCampo1)||(SEN_LINE_L>minCampo1
)&&(SEN_LINE_L<maxCampo1))
                {
                        campo=1;
                        asignacion=true;
                        return;
                }
                //si ambos sensores están comprendidos entre los valores asignados al campo 2,
                //campo toma valor 2
        if((SEN_LINE_R>minCampo2)&&(SEN_LINE_R<maxCampo2)||(SEN_LINE_L>minCampo2
)&&(SEN_LINE_L<maxCampo2))
                {
                        campo=2;
                        asignacion=true;
                        return;
                }
                //movimiento rectilíneo durante 100ms
                MOT_STR(10,FWD,DISTANCE,1);
                while(!input(MOT_END)){}
                Delay_ms(1000);
        }

}
```

Descripción del juego

El juego propiamente dicho se divide esencialmente en tres funciones. Mientras que no se detecte obstáculo, el microbot se dedica a defender la portería girando incesantemente a derecha e izquierda, sin salirse del área de portería. Al detectar obstáculo, lo empuja hasta el otro campo a través de una función de ataque y posteriormente regresa al área de portería para retomar la posición de defensa.

Defender portería

Mientras que no se detecte obstáculo, el microbot se mantiene girando a derecha e izquierda sin salirse de la zona que simboliza el área de portería. La función Actuación() se encarga de ello.

Se supone que si Moway se encuentra justo en el centro de la portería, gira aproximadamente 180º desde que detecta línea negra de campo hasta que vuelva a encontrar otra vez línea negra. A continuación se muestra el código que se encarga de mantener al microbot más o menos cerca del centro de la portería, actuando cuando el ángulo girado es menor que 144º o mayor que 216º.

```
MOT_ROT(50,FWD,CENTER,direccionGiro,ANGLE,0);
while(!input(MOT_END))
{
	if(DeteccionObstaculo()==true)
		Ataque();
	SEN_LINE_ANALOG();
	if((SEN_LINE_R>maxCampo2)&&(SEN_LINE_L>maxCampo2))
	{
		MOT_FDBCK(STATUS_A);
		MOT_STOP();
		//si el ángulo girado es menor que 144º, tendrá que alejarse un poco de su portería
		if(MOT_STATUS_DATA_0<40)
		{
			direccionGiro=!direccionGiro;
			//gira 90 grados para poder adentrarse un poco en el campo
			MOT_ROT(10,FWD,CENTER,direccionGiro,ANGLE,25);
			while(!input(MOT_END))
			{
				if(DeteccionObstaculo()==true)
					Ataque();
			}
			//se adentra 1 centímetro en el campo
			MOT_STR(10,FWD,DISTANCE,100/17);
			while(!input(MOT_END))
			{
				if(DeteccionObstaculo()==true)
					Ataque();
			}
			//sigue girando hasta detectar línea de campo
			MOT_ROT(50,FWD,CENTER,direccionGiro,ANGLE,0);
			while(!input(MOT_END))
			{
				if(DeteccionObstaculo()==true)
					Ataque();
				SEN_LINE_ANALOG();
				if((SEN_LINE_R>maxCampo2)||(SEN_LINE_L>maxCampo2))
					MOT_STOP();
			}
		}
		//si el ángulo girado es mayor que 216º, tendrá que retroceder  hacia su propia portería
		if(MOT_STATUS_DATA_0>60)
		{
			direccionGiro=!direccionGiro;
			//gira 90º
			MOT_ROT(10,FWD,CENTER,direccionGiro,ANGLE,25);
			while(!input(MOT_END))
			{
				if(DeteccionObstaculo()==true)
					Ataque();
			}
			//retrocede 1 centímetro para acercarse a la portería
			MOT_STR(10,BACK,DISTANCE,100/17);
			while(!input(MOT_END))
			{
				if(DeteccionObstaculo()==true)
					Ataque();
			}
			//sigue rotando hasta detectar línea de campo
			MOT_ROT(50,FWD,CENTER,direccionGiro,ANGLE,0);
			while(!input(MOT_END))
```

```
                        {
                                if(DeteccionObstaculo()==true)
                                        Ataque();
                                SEN_LINE_ANALOG();
                                if((SEN_LINE_R>maxCampo2)||(SEN_LINE_L>maxCampo2))
                                        MOT_STOP();
                        }
                }
        }
}
```

Ataque

Cuando se detecta obstáculo a través de la función DeteccionObstaculo(), se llama a la función Ataque que consiste básicamente en empujar el obstáculo al 100% de la velocidad máxima, hasta llegar al campo contrario.

```
//sigue avanzando como máximo 8cm a la velocidad máxima, hasta que encuentre el campo contrario o
//línea de campo
MOT_STR(100,FWD,DISTANCE,800/17);
while(!input(MOT_END))
{
        SEN_LINE_ANALOG();
        //si el sensor derecho detecta campo contrario o línea, para
        if(((campo==1)&&(SEN_LINE_R>maxCampo1))||((campo==2)&&((SEN_LINE_R>maxCamp
o2)||(SEN_LINE_R<minCampo2))))
        {
                MOT_STOP();
        }
        //si el sensor izquierdo detecta campo contrario o línea, para
        if(((campo==1)&&(SEN_LINE_L>maxCampo1))||((campo==2)&&((SEN_LINE_L>maxCamp
o2)||(SEN_LINE_L<minCampo2))))
        {
                MOT_STOP();
        }
}
```

Regresar a posición de defensa

Una vez que ha llevado el balón al campo del contrincante intentando marcar gol, regresa a su campo para volver a adoptar la posición de defensa de la portería empleando la función BusquedaPorteria() para ello.

En el caso más sencillo y habitual, al no desviarse ni girar al empujar el balón, por ser éste muy ligero, al retroceder encuentra fácilmente el área de portería.

```
//retrocede 5cm para dejar de detectar o campo contrario o línea
MOT_STR(30,BACK,DISTANCE,500/17);
while(!input(MOT_END)){}

//retrocede un máximo de 17cm para detectar línea que rodea el campo
MOT_STR(30,BACK,DISTANCE,1700/17);
while(!input(MOT_END))
{
        SEN_LINE_ANALOG();
        //si se detecta línea de campo, para
```

```
        if((SEN_LINE_R>maxCampo2)&&(SEN_LINE_L>maxCampo2))
        {
                MOT_STOP();
                MOT_STR(30,FWD,DISTANCE,300/17);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        if((SEN_LINE_R<minCampo1)&&(SEN_LINE_L<minCampo1))
                        {
                                posicionado=true;
                                return;
                        }
                }
                if(posicionado==false)
                {
                        MOT_STR(30,BACK,DISTANCE,300/17);
                        while(!input(MOT_END)){}
                }
        }
}
```

En el peor de los casos, al retroceder detecta la línea de campo lateral o se queda atrapado en alguna de las dos esquinas. En estos casos, gira a derecha e izquierda retrocediendo en ambas direcciones alternativamente hasta conseguir llegar al área de portería.

```
//mientras no encuentre el área de portería
while(posicionado==false)
{
        while((auxL==false)&&(i<5))
        {
                //rota 30º aprox. a la izquierda para detectar área de portería
                MOT_ROT(10,FWD,CENTER,LEFT,ANGLE,8);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        //si detecta área de portería , para
                        if((SEN_LINE_R<minCampo1)||(SEN_LINE_L<minCampo1))
                        {
                                posicionado=true;
                                return;
                        }
                        //si detecta línea de campo, para
                        if((SEN_LINE_R>maxCampo2)&&(SEN_LINE_L>maxCampo2))
                        {
                                MOT_STOP();
                                auxL=true;
                        }
                }
                //retrocede un máximo de 5 centímetros mientras que comprueba si detecta línea o área
                //de portería
                MOT_STR(30,BACK,DISTANCE,500/17);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        if((SEN_LINE_R>maxCampo2)||(SEN_LINE_L>maxCampo2)||(auxL==true))
                        {
                                MOT_STOP();
                                auxL=true;
```

```
                                MOT_STR(30,FWD,DISTANCE,300/17);
                                while(!input(MOT_END))
                                {
                                        SEN_LINE_ANALOG();
                                    if((SEN_LINE_R<minCampo1)||(SEN_LINE_L<minCampo1))
                                        {
                                                posicionado=true;
                                                return;
                                        }
                                }
                        }
                }
                i=i+1;
        }
        i=0;
        while((auxR==false)&&(i<5))
        {
                //rota 30º aprox. a la derecha para detectar área de portería
                MOT_ROT(10,FWD,CENTER,RIGHT,ANGLE,8);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        //si detecta área de portería, para
                        if((SEN_LINE_R<minCampo1)||(SEN_LINE_L<minCampo1))
                        {
                                posicionado=true;
                                return;
                        }
                        //si detecta línea de campo, para
                        if((SEN_LINE_R>maxCampo2)&&(SEN_LINE_L>maxCampo2))
                        {
                                MOT_STOP();
                                auxR=true;
                        }
                }
                //retrocede un máximo de 5cm mientras comprueba si detecta línea o área de portería
                MOT_STR(30,BACK,DISTANCE,500/17);
                while(!input(MOT_END))
                {
                        SEN_LINE_ANALOG();
                        if((SEN_LINE_R>maxCampo2)||(SEN_LINE_L>maxCampo2)||(auxR==true))
                        {
                                MOT_STOP();
                                auxL=true;
                                MOT_STR(30,FWD,DISTANCE,300/17);
                                while(!input(MOT_END))
                                {
                                        SEN_LINE_ANALOG();
                                    if((SEN_LINE_R<minCampo1)||(SEN_LINE_L<minCampo1))
                                        {
                                                posicionado=true;
                                                return;
                                        }
                                }
                        }
                }
                i=i+1;
        }
        i=0;
}
```

CAPÍTULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

CAPÍTULO 6: CONCLUSIONES Y TRABAJOS FUTUROS

A lo largo de este proyecto hemos trabajado con Moway, un robot de pequeño tamaño, bajo coste y de gran acogida en el mundo de la enseñanza y el aprendizaje. Es programable, autónomo y compacto, con un diseño que le permite moverse con agilidad y elegancia, interactuando con el entorno gracias a los sensores de línea, obstáculos y de luz que lleva incorporados.

Al estudiar la odometría del microbot, se observa la gran cantidad de errores no sistemáticos que oculta su movimiento en cualquiera de sus formas y la imposibilidad de eliminarlo. Esto nos llevo a centrarnos en los sensores de línea, obstáculos y con especial interés en el de luz. Un abanico enorme de posibilidades se abría al contar con estos tres tipos de sensores digital-analógicos. Pero dichas posibilidades se iban reduciendo a medida que se conocía la imprecisión y la necesidad de propiciar un ambiente ideal, no compatible con las aplicaciones a desarrollar en algunos casos.

El capítulo 5 muestra la aplicación de todo el trabajo e investigación realizado en capítulos anteriores. Empleando el sensor de luz se consigue corregir trayectorias mediante el uso de balizas, dirigirse a puntos concretos y de distinguir objetos luminosos de los que no lo son. El sensor de línea nos permite el seguimiento de línea, la reclusión en recintos cerrados y en su modo analógico la posibilidad de distinguir perfectamente hasta 6 tonos de gris, permitiendo conocer en que zona del campo se encuentra o la dirección en la cual se desplaza.

Puesto que en este proyecto sólo hemos desarrollado una aplicación en la que se corregía una trayectoria rectilínea situando una baliza luminosa en su extremo final, en el futuro se podría trabajar más a fondo en la corrección de trayectorias empleando balizas en puntos intermedios del recorrido.

En lo relacionado al sensor de línea, se podría investigar sobre texturas para intentar aumentar la precisión y también desarrollar campos de terreno más complejos con los que sea posible situar al microbot en una zona más concreta del campo. Y con esto elaborar juegos de mesa en los que varios Moways interactúen.

Desde hace unos meses se ha comenzado a comercializar una tarjeta de expansión que permitiría dotar a Moway de luz, añadirle más sensores de obstáculos, acoplarle una cámara VGA, realizar la carga de batería a distancia o incorporar algún tipo de sistema de posicionamiento por infrarrojo según la página oficial de Moway.

Figura 133: Tarjeta de expansión

En ocasiones, contando sólo con dos sensores de obstáculos en su parte frontal, ocurre que el microbot está chocando lateralmente o por su parte trasera con algún obstáculo y es incapaz de detectarlo. Ese es un aspecto mejorable con esta nueva tarjeta de expansión.

También es posible hacer aplicaciones más vistosas, introduciendo contadores o dotando de luz a los microbots.

BIBLIOGRAFÍA

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

[2] http://www.bizintekinnova.com/, Bizintek Innova

[3] "Microcontrolador PIC 16F84 desarrollo de proyectos ", E. Palacios Municio, Ra-Ma 2003

[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.samwha.cz/kingbright/ktir0711s.pdf, Sensor KTIR0711S de Kingbright

[8] http://www.avagotech.com/, Sensor APDS-9002 de Avago Technologies

[9] http://geology.heroy.smu.edu/~dpa-www/robo/Encoder/imu_odo/index_IE.htm, Información sobre odometría

[10] http://www.cs.brown.edu/courses/cs092/VA10/HTML/ColorModels.html, Información sobre el modelo de color HSL

[11] http://ro-botica.com/, Robot Moway

[12] http://www.todomicrostamp.com/, Foro de gupos de investigación con Moways

[13] http://www.robotic-lab.com/, Información acerca de Moway

[14] http://www.minirobots.es/, Nuevos productos de Moway