

# MANUAL DE USUARIO DE MOWAY



**Copyright (c) 2007 Bizintek Innova, S.L.**

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

## Índice

Índice .....	3
1. Prólogo .....	4
2. ¿Qué es Moway? .....	6
3. Robot Moway .....	7
3.1. Procesador .....	7
3.2. Sistema motriz .....	8
3.3. Grupo de sensores e indicadores .....	10
3.3.1. Sensores de línea .....	11
3.3.2. Sensores detectores de obstáculos .....	13
3.3.3. Sensor de luz.....	14
3.3.4. Conector de expansión.....	14
3.3.5. Leds frontales .....	15
3.3.6. Led superior bicolor.....	15
3.3.7. Pad libre.....	15
3.4. Sistema de alimentación .....	16
4. Base Moway .....	17
4.1. Características.....	17
4.2. Moway Center .....	17
4.2.1. Principal.....	17
4.2.2. Radio frecuencia.....	19
4.3. Firmware.....	21
5. Primeros Pasos.....	22
5.1. Instalación del software .....	22
5.2. Conexión de la Base Moway .....	22
5.3. Primeros pasos de Moway .....	24
6. Programación de Moway empleando MPLAB .....	26
6.1. Creación de un proyecto .....	26
6.2. Primer programa en ensamblador.....	30
6.3. Librerías.....	34
6.3.1. Librería sensores Moway ensamblador .....	34
6.3.2. Librería motores Moway ensamblador.....	40
7. Programación del Moway empleando CCS PIC C Compiler .....	51
7.1. Creación de un Proyecto.....	51
7.2. Primer programa en CCS.....	53
7.3. Librerías.....	57
7.3.1. Librería sensores Moway en C para CCS.....	57
7.3.2. Librería motores Moway en C para CCS .....	63
8. Ejercicios Prácticos.....	72

## 1. Prólogo

Comienza una nueva era, la de los minirobots. Cada vez son más las aplicaciones de la robótica móvil en nuestra vida cotidiana. Actualmente podemos ver robots que nos ayudan en tareas sencillas como limpiar el suelo de casa, segar el césped o mantener limpia la piscina. A medida que avanza la tecnología estos pequeños artilugios, mezcla de mecánica, electrónica y software, van asumiendo tareas más complejas\*. Poco a poco se van abriendo camino hacia nosotros siéndonos cada vez más útiles y descargándonos de los trabajos menos gratificantes.

No es ningún disparate pensar que la revolución que se dio en la informática o en las telecomunicaciones se va a repetir en la robótica durante la próxima década. Actualmente disponemos de la tecnología suficiente para fabricar estos dispositivos y la sociedad está cada vez más preparada para recibirlos en el mercado. Pero para poner en marcha toda esta revolución hace falta un catalizador específico. Es necesario que exista gente capaz de identificar dónde tiene una oportunidad la microbótica y qué nuevas aplicaciones puede ser interesante implementar.

Hasta ahora los procesadores no se movían. Las cosas han cambiado. Uno de los elementos fundamentales en el mundo de la robótica móvil es el software. La principal diferencia a la hora de desarrollar programas para estos robots con respecto a hacerlo para ejecutarlos en un ordenador personal estriba en la interacción con el entorno. En las aplicaciones para pc el entorno no cambia aleatoriamente, con la que la toma de decisiones se simplifica y con ello los programas. Por otro lado, en la ejecución de comandos dentro de una aplicación para un microbot lo habitual es que se desconozca de antemano cuál va a ser el resultado, por lo que los algoritmos deben contemplar situaciones con un abanico mucho más amplio de posibilidades, algunas incluso inesperadas.

Los Moways son herramientas diseñadas específicamente para la docencia e investigación. Su objetivo es acercar el mundo de la robótica autónoma a los centros docentes.

El objetivo principal de Moway es ser una herramienta útil tanto para quienes se introducen por primera vez en el mundo de la microbótica como para quienes ya tienen experiencia y desean realizar aplicaciones complejas de robótica colaborativa.

Moway aspira a despertar inquietud por esta nueva y apasionante rama de la ingeniería a quien le tenga en su mano, a través de las prácticas contenidas en este manual de una forma rápida y entretenida.

- Una forma de aprender fácil y divertida.
- Objetivo de este libro: Manual de Moway, no un libro de microbótica genérico.

Este manual ha sido realizado con el propósito de facilitar el aprendizaje en el manejo de Moway. Se pretende dar unas nociones básicas de una forma rápida y clara del funcionamiento y uso de Moway.

El manual está dividido en dos partes. A lo largo de la primera se dará una descripción de los elementos que componen el robot y de su funcionamiento. La segunda parte del manual contiene una serie de prácticas que podrán realizar con Moway.

## 2. ¿Qué es Moway?

Moway es un pequeño robot autónomo programable diseñado principalmente para realizar aplicaciones prácticas de microbótica.

Con él se ha conseguido una plataforma hardware perfecta tanto para quien quiere dar sus primeros pasos en el mundo de los robots móviles como para quien ya ha trabajado con microbots y desea realizar aplicaciones más complejas.

El robot Moway está dotado de una serie de sensores que le ayudarán a desenvolverse en un entorno real. A su vez cuenta con un grupo motor que le permitirá desplazarse en un terreno liso comandado a través del bus de comunicaciones I2C. Todos estos periféricos están conectados a un microcontrolador que será el encargado de gobernar el robot.

Este pequeño robot cuenta además con opciones de ampliación a través de un bus de expansión por I2C/SPI. En él se puede conectar, por ejemplo, un módulo de comunicaciones inalámbrico, una cámara de video, una tarjeta de prototipos o cualquier otro dispositivo que se considere interesante para el desempeño de una tarea.

El diseño exterior de Moway es muy compacto, diseñado para que pueda moverse con agilidad y elegancia sin opción a quedarse enganchado en ninguna esquina. Tan pequeño como un móvil se ha ganado la denominación de “robot de bolsillo”.

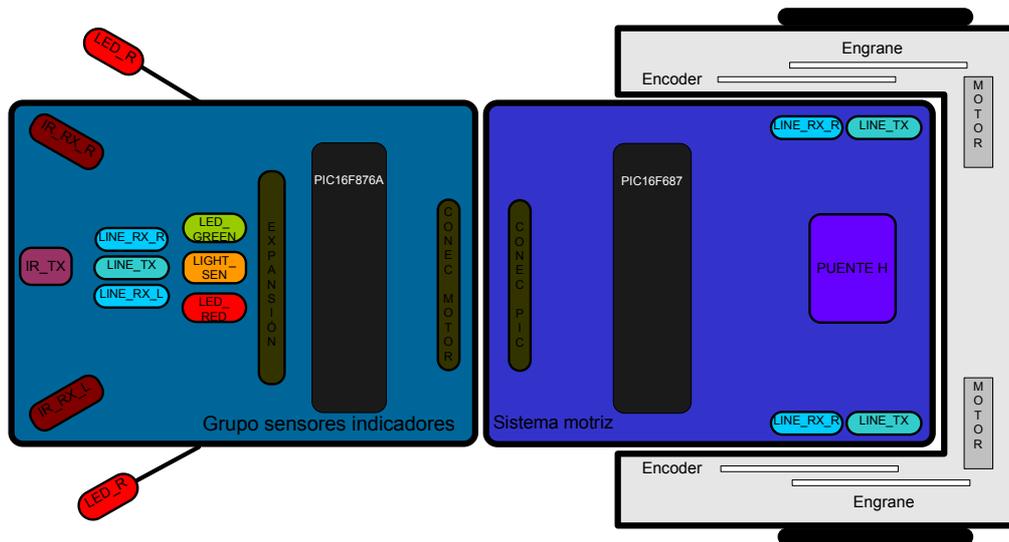
Moway es una herramienta perfecta para quien quiere aprender y para quien quiere enseñar qué es la microbótica. El usuario se verá sorprendido con la rapidez que comienza a cosechar logros incluso si éste es su primer contacto con los robots móviles.

### 3. Robot Moway

A continuación y a lo largo de todo este capítulo se van a ir explicando cada una de las partes que componen un Moway. Cabe destacar que no es necesario conocer el funcionamiento interno del robot para poder realizar la programación del mismo, al menos no al nivel de detalle que se expone a continuación.

En el interior de un Moway tenemos los siguientes elementos:

1. Procesador
2. Sistema motriz
3. Grupo de sensores e indicadores
4. Sistema de alimentación
5. Conector de expansión



**Imagen 1. Representación de las partes de Moway**

#### 3.1. *Procesador*

Los Moways están gobernados 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. Algunos de ellos necesitan de una entrada o salida digital, otros de una analógica y otros en cambio se controlan a través de uno de los buses de comunicación I2C/SPI. A continuación se muestra una tabla con la asignación de pines del microcontrolador.

**Tabla 1. Conexiones PIC-sensores**

Pin PIC	I/O	Sensor
<b>PORTA</b>		
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
<b>PORTB</b>		
RB1	O	Transmisor sensores línea
RB2	O	Transmisor infrarrojo
RB4	O	LED inferior derecho
RB6	O	LED superior verde
<b>PORTC</b>		
RC6	O	LED inferior izquierdo
RC7	I/O	Pad libre

La grabación del microprocesador se realiza mediante la Base Moway.

**Tabla 2. Conexiones PIC-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

### **3.2. Sistema motriz**

Los Moways disponen de un grupo servo-motor doble para poder desplazarse. Consta de una parte electrónica y otra mecánica. La parte electrónica se encarga principalmente de controlar la velocidad de los motores y la parte mecánica permite el desplazamiento con una potencia suficiente para que Moway se mueva sin problemas diferentes entornos.

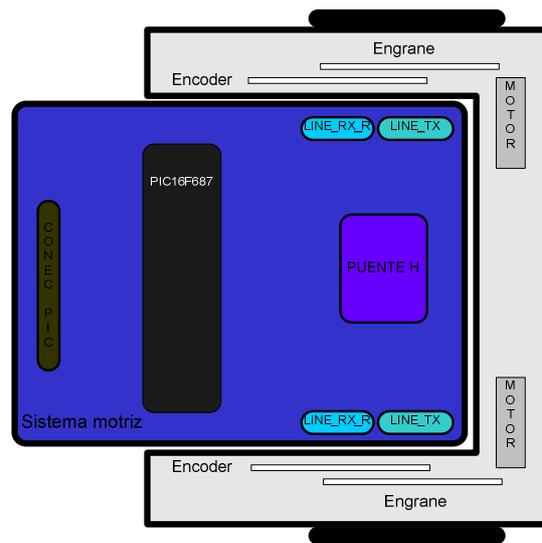


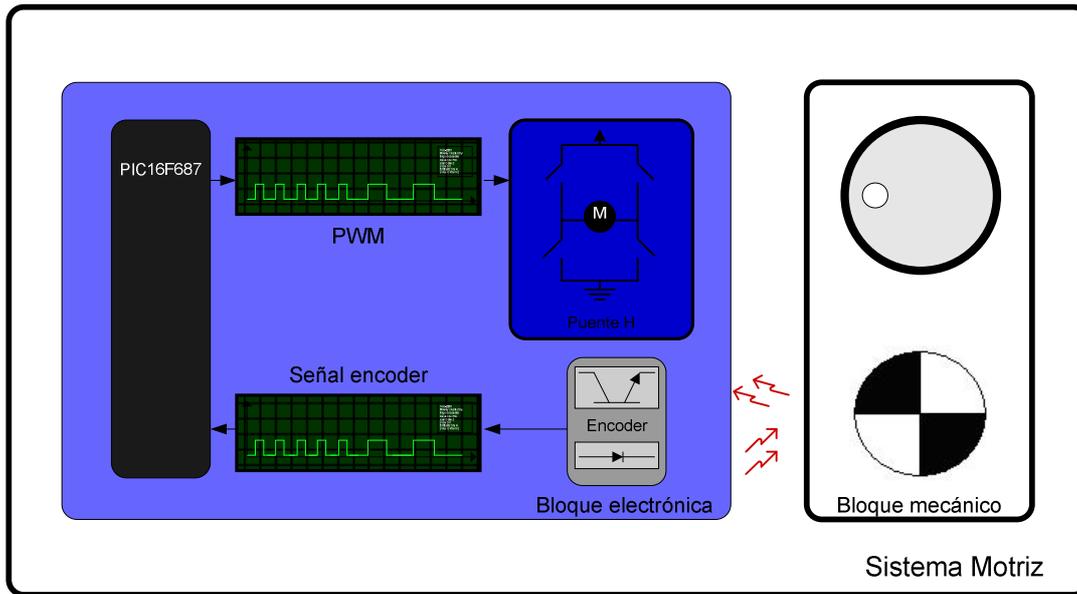
Imagen 2. Sistema motriz: electrónica y mecánica

El grupo servo-motor tiene diversas funcionalidades:

1. **Control de velocidad:** Controla la velocidad de cada motor por separado.
2. **Control de tiempo:** Controla el tiempo en cada comando con una precisión de 100ms.
3. **Control de distancia recorrida:** Controla la distancia recorrida en cada comando con una precisión de 1.7mm.
4. **Cuentakilómetros general:** Cuenta la distancia recorrida desde el comienzo de los comandos.
5. **Control de ángulo:** Control de ángulo cuando se produce la rotación de Moway.

El microcontrolador sólo tiene que mandar 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, pudiendo este último realizar otras tareas.

El control de velocidad se realiza mediante control proporcional con retroalimentación negativa de la señal de los encoders. En la figura podemos observar el sistema de control. El microcontrolador alimenta los motores mediante un puente H controlado por señales PWM (modulación del ancho del pulso). La rotación de la rueda es monitorizada por medio de una pegatina encoder y un sensor infrarrojo. Cuando la pegatina se encuentra en la parte negra, se tendrá un 1 lógico a la salida, mientras que si tenemos un sector blanco la salida será 0. El microcontrolador analiza esta señal (midiendo el ancho del pulso sabe la velocidad exacta de la rueda) y actúa sobre los motores. De esta manera, Moway mantendrá la velocidad constante en cualquier superficie.


**Imagen 3. Control de motores**

Cuando desde el microcontrolador principal se quiere que el robot realice un desplazamiento sólo tenemos que mandar un comando de movimiento con sus parámetros. Para ello se han diseñado unas librerías en ensamblador y en C con las que esta comunicación queda simplificada por unas funciones que se encargan de la comunicación I2C. El formato de las tramas se explica en el apartado de la librería de motores.

En la siguiente tabla podemos ver la conexión entre el PCB principal y el grupo servo-motor.

**Tabla 3. Conexiones Procesador-motores**

<b>Motor</b>	<b>I/O</b>	<b>PIC</b>
Pin1	O	Vcc
Pin2	O	GND
Pin3	O	Vcc_Batt
Pin4	I	RB5
Pin5	I/O	SDA
Pin6	I/O	SCL

### 3.3. 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:

- Dos sensores de línea.
- Dos sensores detectores de obstáculos.
- Sensor de luz.

- Un conector de expansión.
- Cuatro diodos LED.
- Un pad libre

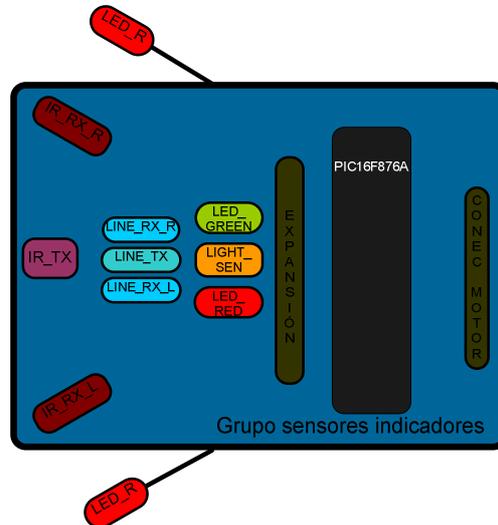


Imagen 4. Grupo sensores e indicadores

### 3.3.1. 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.

El 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.

En las siguientes imágenes podemos ver los tres casos que se pueden dar:

1. **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.

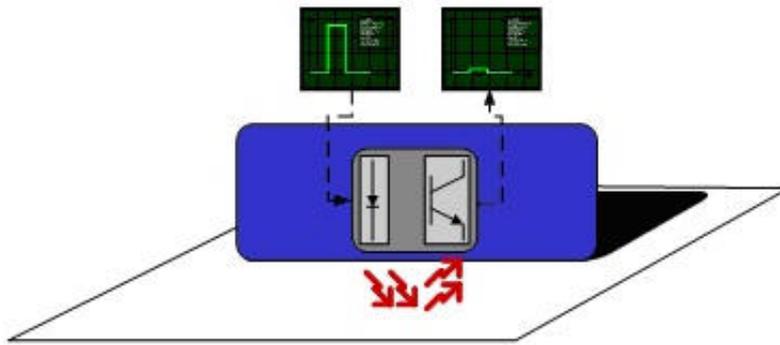


Imagen 5. Sensor de línea en superficie clara.

2. **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.

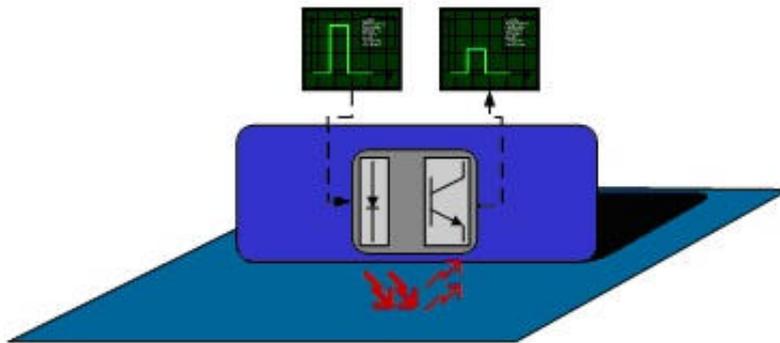


Imagen 6. Sensor de línea en superficie de color.

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

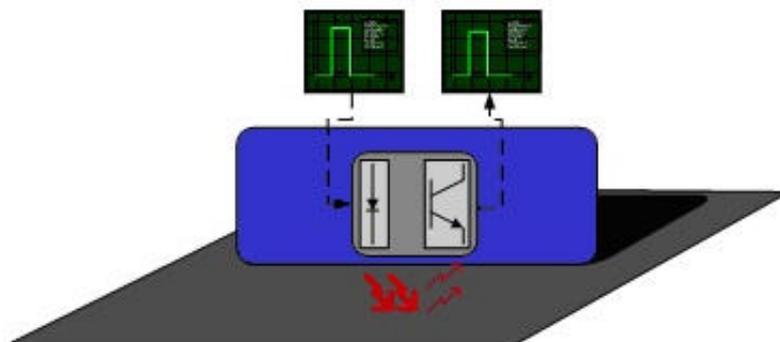


Imagen 7. Sensor de línea en superficie oscura.

En la siguiente tabla se muestra las conexiones del PIC con el sensor de línea. Hay que tener en cuenta que los dos diodos LED están conectados al mismo pin del microcontrolador.

**Tabla 4. Conexiones Sensores de línea-PIC**

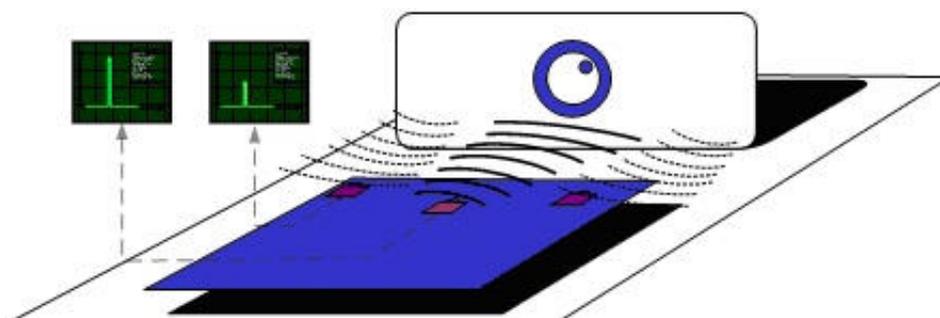
Pin PIC	I/O	Sensor
<b>PORTA</b>		
RA2	I	Receptor sensor línea derecho
RA5	I	Receptor sensor línea izquierdo
<b>PORTB</b>		
RB1	O	Trasmisor sensores de línea izquierdo y derecho

### 3.3.2. 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 infrarrojos 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.



**Imagen 8. Sensor detector de obstáculos**

**Tabla 5. Conexiones sensor antichoque-PIC**

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

### 3.3.3. 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 de Avago Technologies 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.

**Tabla 6. Conexión PIC-sensor de luz**

Pin PIC	I/O	Sensor
<b>PORTA</b>		
RA0	I	Luz

### 3.3.4. Conector de expansión

Este conector permite la conexión de Moway con módulos comerciales o con circuitos electrónicos que el usuario desee.

**Tabla 7. Conexiones PIC-Conector de expansión**

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

Como se observa en la tabla anterior 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.

Este módulo permite hacer aplicaciones colaborativas complejas sin tener que preocuparse de la compleja comunicación inalámbrica.

### 3.3.5. Leds frontales

Los dos leds frontales se encuentran detrás del filtro infrarrojo delantero de manera que sólo se ven cuando se activan. Se han dispuesto uno en cada lateral de la parte frontal y son de color rojo. Están conectados a dos salidas digitales del microcontrolador. Estos Leds deben estar apagados cuando se usan los detectores de obstáculos.

Tabla 8. Conexión PIC- Led frontales

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

### 3.3.6. 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.

Tabla 9. Conexión PIC-Led superior

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

### 3.3.7. 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.

Tabla 10. Conexión PIC-pad libre

Pin PIC	I/O	Sensor
<b>PORTC</b>		
RC7	I/O	Pad libre

### **3.4. Sistema de alimentación**

La batería de Moway se encuentra en el interior y no es accesible a no ser que se desmonte el producto. Se trata de una pequeña célula de LiPo recargable\*.

La recarga de la batería se realiza por el puerto usb de cualquier ordenador mediante la Base Moway. No es necesario esperar a que la batería esté completamente descargada para poder enchufarla, puede hacerse en cualquier momento puesto que este tipo de baterías no tienen efecto memoria. 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.

## 4. Base Moway

Esta base es una tarjeta electrónica que se encarga de gestionar tanto la descarga del programa como la carga de la batería y la comunicación por RF. La Base se conecta con Moway a través de una pequeña apertura debajo del filtro infrarrojo delantero.

### 4.1. Características

Consta de un PIC18F2550 que es el encargado de generar las señales de grabación de Moway, controlar la carga de la batería y gestionar las comunicaciones tanto con Moway como con el PC.

### 4.2. Moway Center

Moway Center es el software para poder manejar la Base Moway que se proporciona mediante la web de Moway.

Moway Center consta de dos partes diferenciadas: Principal y Radio Frecuencia.

#### 4.2.1. Principal

En esta ventana se puede observar el estado de Moway (batería, conexión a la base, etc.) y se permite realizar acciones sobre él.



**Imagen 9. Aspecto de la Ventana Principal**

La Ventana Principal consta de cinco secciones diferenciadas:

## Moway Status

En esta sección se informará de forma automática del estado del robot. Existen 3 estados posibles:

1. *Conectado:* La única forma de realizar acciones sobre Moway.
2. *Apagado:* El robot está conectado a la base y la batería se está cargando, pero no se puede realizar ninguna acción sobre él puesto que está apagado.
3. *Desconectado:* El robot está desconectado de la base.



**Imagen 10. Moway Status**

## Acciones

Consta de tres botones con los que podemos realizar las siguientes acciones si el robot está en estado conectado:

1. *Grabar:* Grabación de Moway. Se debe especificar el archivo .HEX que se quiera grabar. El proceso termina con la verificación de la grabación.
2. *Leer:* Lectura de la memoria de programa de Moway. Al terminar la lectura se muestra, en una ventana diferente, la memoria de programa, la memoria EEPROM y la palabra de configuración.
3. *Borrar:* Se borra la memoria de programa de Moway.



**Imagen 11. Acciones**

## Batería

Se muestra el estado de la batería.



**Imagen 12. Estado de la batería de Moway**

## Radio-Frecuencia

Botón para ocultar o mostrar la ventana RF.

**Imagen 13. Radio-Frecuencia****Info**

Se mostrará diferente información sobre el software y los procesos.

**Imagen 14. Información****4.2.2. Radio frecuencia**

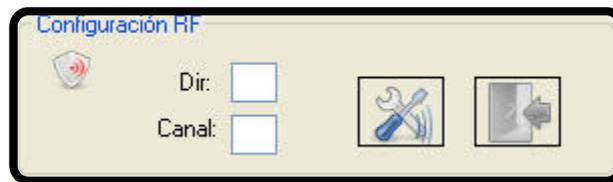
La ventana de Radio Frecuencia posibilita el envío y recepción de datos a través del módulo de comunicaciones BZI-RF2GH4. Para poder utilizar esta funcionalidad es necesario tener conectado a la Base Moway esta tarjeta que nos permite una comunicación sencilla con Moway o con otros Módulos de Grabación.

**Imagen 15. Ventana RF**

En la ventana Principal podemos observar cinco secciones:

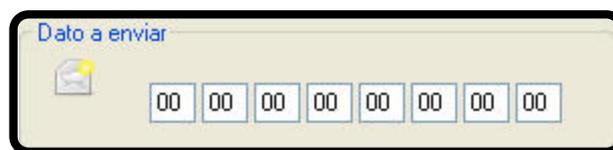
**Configuración RF**

Para configurar el módulo RF sólo es necesario indicar la dirección y el canal en que se enmarcaran las comunicaciones (datos en hexadecimal). Recordar que para comunicarse con los robots es necesario estar en el mismo canal y tener diferente dirección.


**Imagen 16. Configuración RF**

### Dato a enviar

El dato que queremos enviar tiene que estar en formato hexadecimal y tener definidos los 8 bytes.


**Imagen 17. Dato a enviar**

### Envío

El destinatario recibirá los datos configurados anteriormente. Si el receptor no recoge correctamente los datos enviados se mostrará un mensaje indicando el problema.


**Imagen 18. Envío de dato**

### Recepción

Si el integrante que participa en la comunicación manda algún dato a la base, se mostrará automáticamente especificando la dirección del emisor y la hora de llegada.


**Imagen 19. Recepción de datos**

### 4.3. Firmware

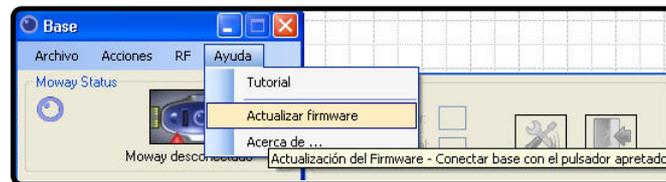
El Firmware de la Base Moway es el programa interno que tiene el microcontrolador para realizar todas las acciones. Éste se puede actualizar mediante el Moway Center. Los pasos para actualizar el Firmware son los siguientes:

1. *Conectar la Base Moway con el pulsador activado:* el software detectará a la base en modo Actualización.



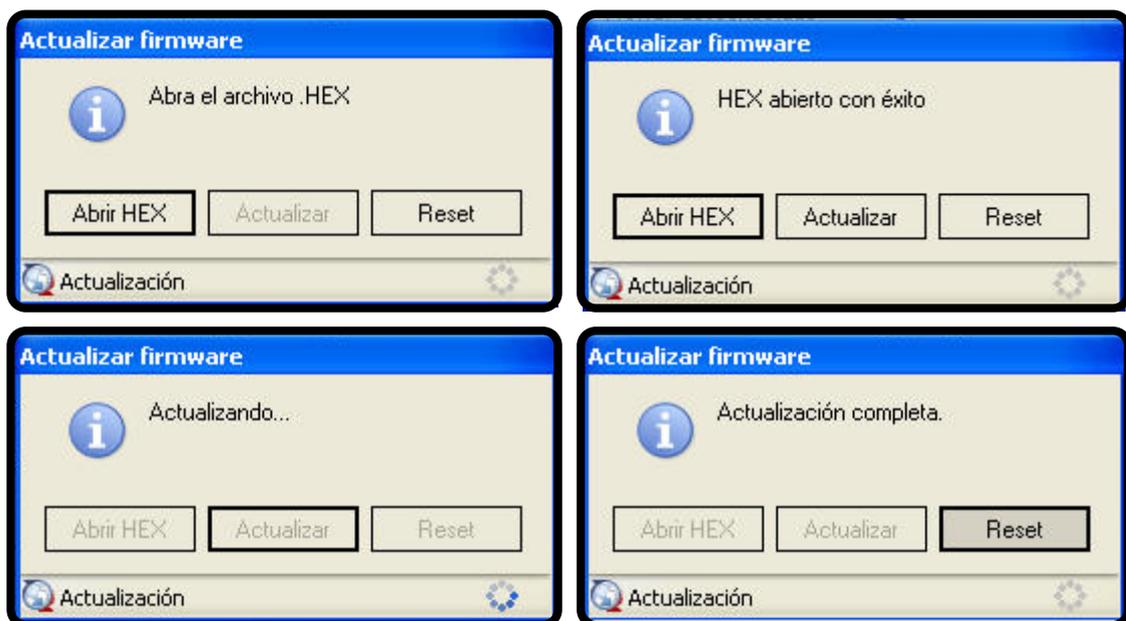
**Imagen 20. Estado de la base: Actualización**

2. *Entrar en la ventana de “Actualización”:* Accederemos a ella a través de Menú → Actualización firmware



**Imagen 21. Entrada a ventana de Actualización**

3. *Proceso de Actualización:* en la ventana de “Actualización” el primer paso es abrir el archivo .HEX con el nuevo Firmware. A continuación se pulsa el botón de actualizar y empezará el proceso de actualización que tardará unos pocos segundos. Si se produce una interrupción en el proceso se deberá empezar otra vez desde el primer paso. Al finalizar, resetee la base.



**Imagen 22. Proceso de Actualización**

## 5. Primeros Pasos

### 5.1. Instalación del software

En la página web de [Moway](#) podrá encontrar el pack de instalación que contiene el software para la Base Moway, las librerías para manejar el robot, los programas de prueba y la documentación.

Basta con seguir los pasos del instalador para tener todos los recursos de Moway:

- Driver de la Base Moway
- Librerías de motor y sensores
- El manual de usuario
- El software Moway Center
- Los recursos del módulo de RF **BZI-RF2GH4**
- Proyectos compilados de Moway

### 5.2. Conexión de la Base Moway

Al conectar la Base Moway por primera vez hay que seguir los siguientes pasos si se está utilizando el sistema operativo Windows xp (con otros SO los pasos serían análogos):

1. La primera vez que se conecta la base, el PC lo detectará como un nuevo dispositivo y aparecerá el “Asistente para hardware nuevo encontrado”. Elegiremos la opción *No por el momento*.



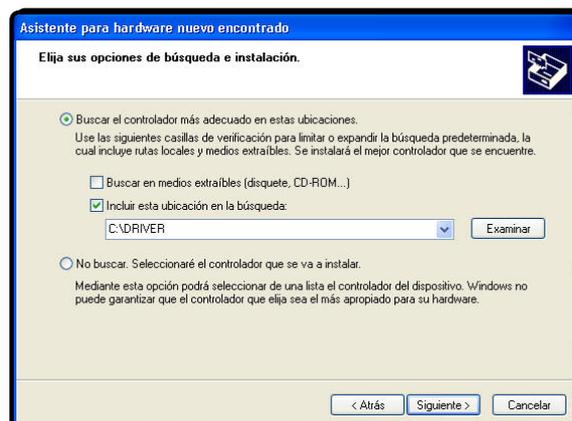
Imagen 23. Asistente para nuevo hardware encontrado

- En la siguiente ventana elegimos la opción avanzada: Instalar desde una lista o ubicación específica.



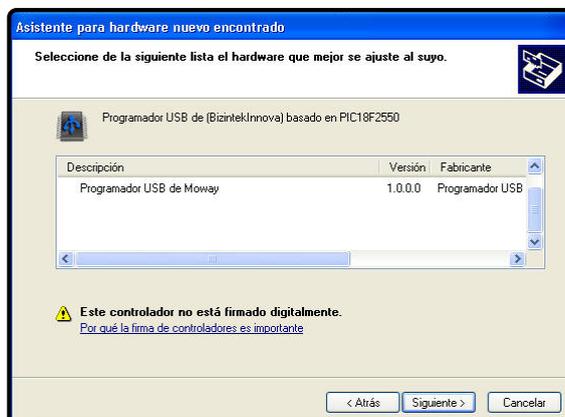
**Imagen 24. Instalar Driver desde una ubicación específica**

- A continuación elegimos la opción de *Busca controlador más adecuado en estas ubicaciones* indicando la ubicación del driver.



**Imagen 25. Especificación de ubicación**

- El SO encontrará el driver de la Base Moway y se elige.



**Imagen 26. Driver Programador USB de Moway**

5. Comenzará la instalación.



**Imagen 27. Comienzo de instalación Driver**

6. Finalmente el asistente indicará que el hardware está instalado y listo para usarse.



**Imagen 28. Finalización de instalación Base Moway**

7. Comprobar que el software de Moway detecta la Base.

### **5.3. Primeros pasos de Moway**

Una vez instalado el software Moway Center y la Base Moway, sólo nos queda grabar uno de los programas de muestra que viene con el pack de instalación (Proyectos Moway).

Al conectar Moway en la Base en la sección *Moway Status* indicará el estado del robot. Una vez encendido se activa la posibilidad de grabarlo. Al pulsar el botón de grabar memoria de programa se abrirá una ventana donde se tiene que indicar el programa que se quiere grabar en Moway.

En el instalador se suministran 8 de proyectos compilados: 3 para probar los sensores, 3 para ver el funcionamiento de los motores y los dos programas que se explican en el apartado Primer Programa.

Proyectos que prueban los sensores:

- **ASM\_SEN\_01**: Software en ensamblador para probar los sensores. Se encienden los led inferiores si se detecta obstáculo. Este proyecto utiliza las librerías con variables en posición fija **lib\_sen\_moway\_10**.
- **ASM\_SEN\_02** : Software en ensamblador para probar los sensores. Se encienden los led inferiores si se detecta obstáculo. Este proyecto utiliza las librerías reubicables **lib\_sen\_moway\_11**.
- **CCS\_SEN\_01** : Software en CCS para probar los sensores. Se encienden los led inferiores si se detecta obstáculo.

Proyectos que prueban los motores:

- **ASM\_MOT\_01**: Software en ensamblador para probar los motores. Se ejecutan diferentes movimientos de Moway. Este proyecto utiliza las librerías con variables en posición fija **lib\_sen\_moway\_10**.
- **ASM\_MOT\_02**: Software en ensamblador para probar los motores. Se ejecutan diferentes movimientos de Moway. Este proyecto utiliza las librerías reubicables **lib\_sen\_moway\_11**.
- **CCS\_MOT\_01**: Software en CCS para probar los motores. Se ejecutan diferentes movimientos de Moway.

Primeros proyectos:

- **Moway\_first\_project\_ASM**: Software en CCS del manual. Moway se mueve recto y evita los obstáculos girando 180°.
- **Moway\_first\_project\_CCS**: Software en ensamblador del manual. Moway se mueve recto y evita los obstáculos girando 180°.

## 6. Programación de Moway empleando MPLAB

El MPLAB IDE de Microchip es el entorno de programación más utilizado para los microcontroladores PIC (ya que Microchip también es el fabricante de dichos microcontroladores). En principio el lenguaje que utiliza es el ensamblador, aunque se le pueden añadir otros lenguajes. Gracias a él se puede compilar el código fuente y generar los ficheros hexadecimales (.HEX). Este compilador se puede descargar gratuitamente desde la página web de Microchip.

En la web de Moway encontrará librerías para el manejo de los sensores, motores y módulo RF escritas para MPLAB.

En resumen:

- Muy interesante para aprender a programar en ensamblador (lenguaje de bajo nivel).
- Recomendable si el programa a realizar va a ser largo (en cuanto a código se refiere).
- Imprescindible si el tiempo de respuesta es crítico.

### 6.1. Creación de un proyecto

Para crear el primer proyecto utilizaremos el Project Wizard de MPLAB IDE que permite crear proyectos rápidamente.

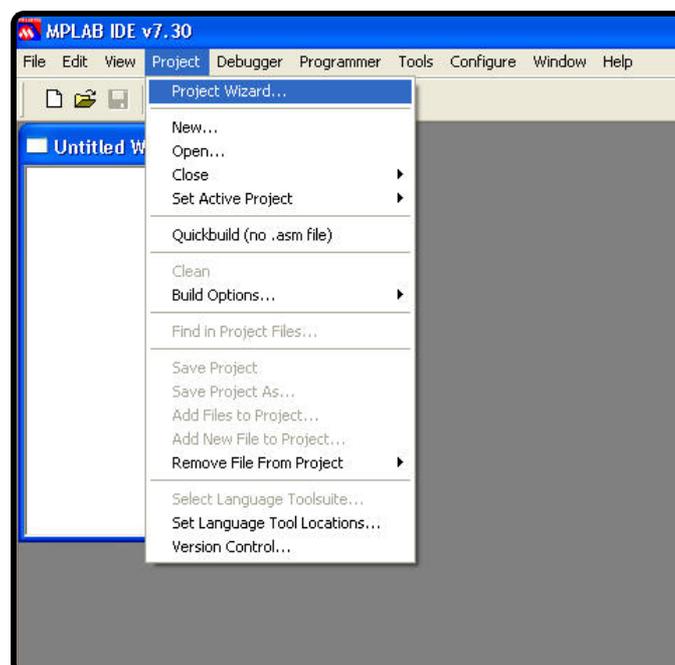
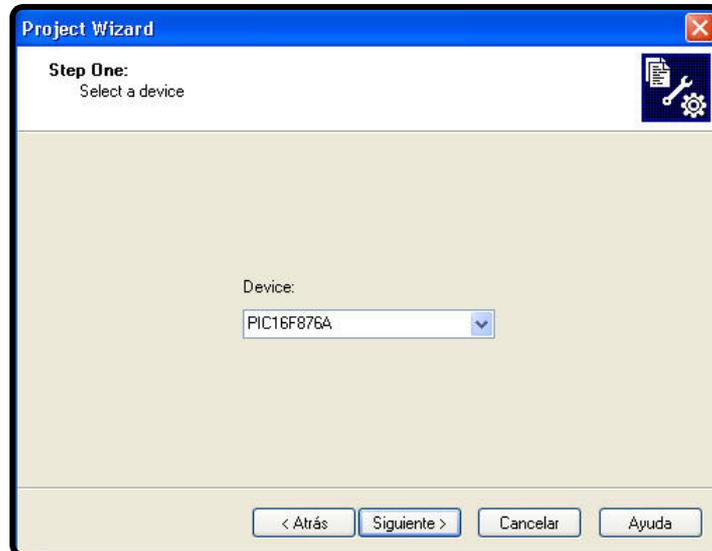


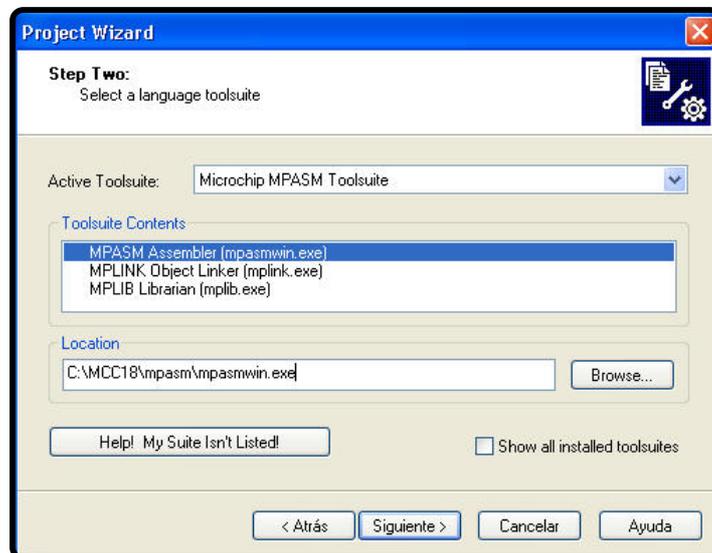
Imagen 29. Project Wizard

1. El primer paso consiste en elegir el PIC instalado en Moway: PIC16F876A.



**Imagen 30. Selección del PIC**

2. A continuación se elige la herramienta de ensamblado: MPASM.



**Imagen 31. Selección de herramienta**

3. En el paso tres se debe indicar el nombre del proyecto y la ubicación del mismo.

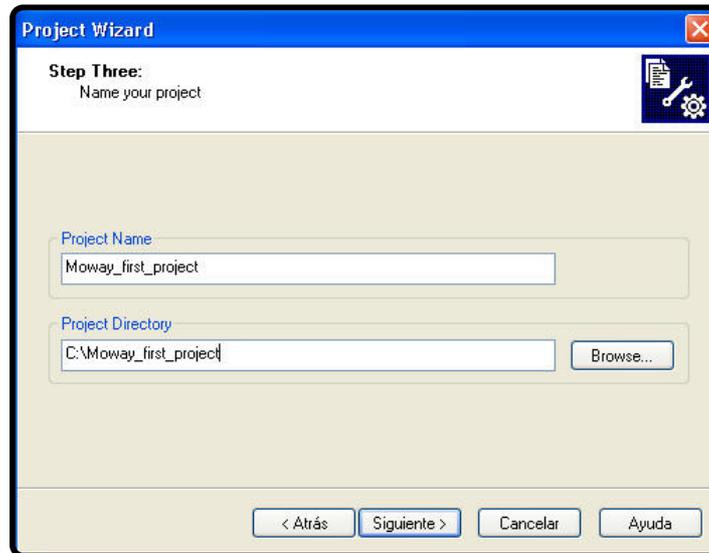


Imagen 32. Elección de nombre y carpeta

4. En el siguiente paso se añaden al proyecto las librerías de Moway que controlan diferentes aspectos del robot.

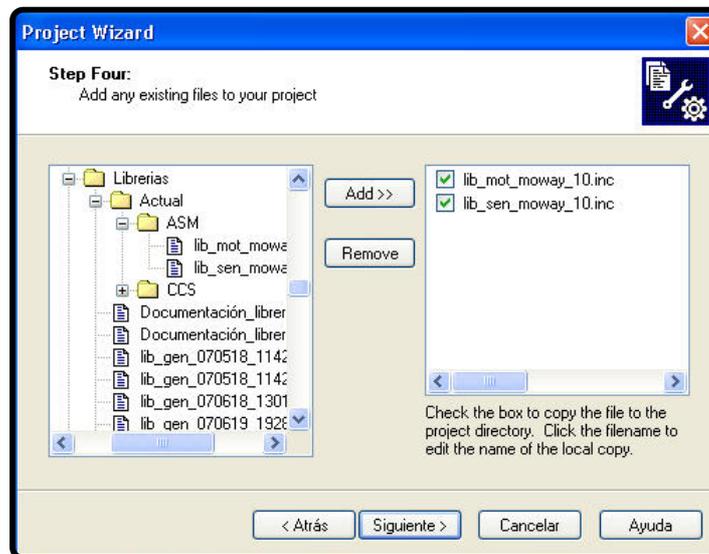


Imagen 33. Elección de librerías Moway

5. Siguiendo estos pasos el proyecto se creará, pero faltará todavía la creación del fichero .ASM donde se inserta el código fuente.

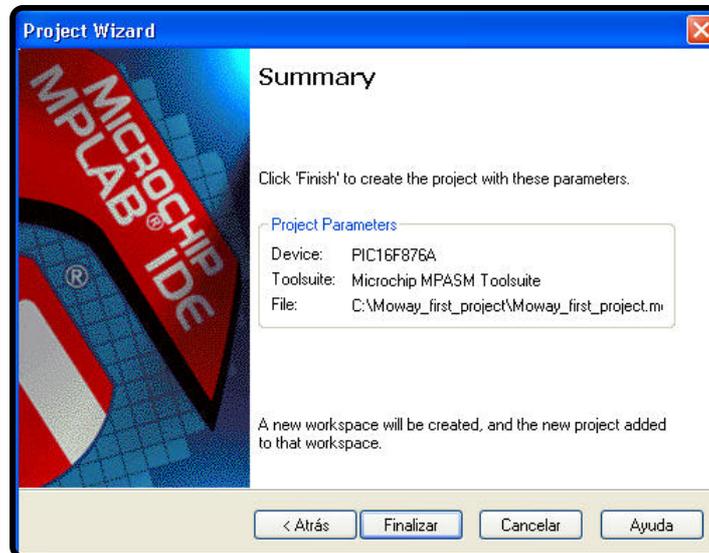


Imagen 34. Finalización Wizard

6. El siguiente paso es abrir el proyecto y crear un nuevo archivo (*New File*) guardándolo en la misma carpeta del proyecto como *Main.asm*. Éste será nuestro fichero fuente.

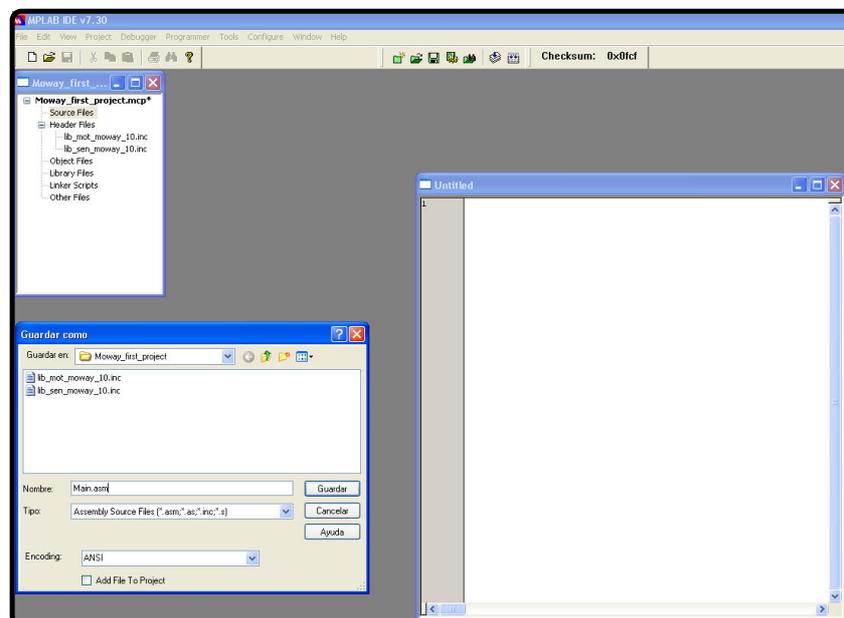


Imagen 35. Creación de .ASM

7. Por último se añade el fichero fuente al proyecto accediendo a *Project/Add Files to Project...*

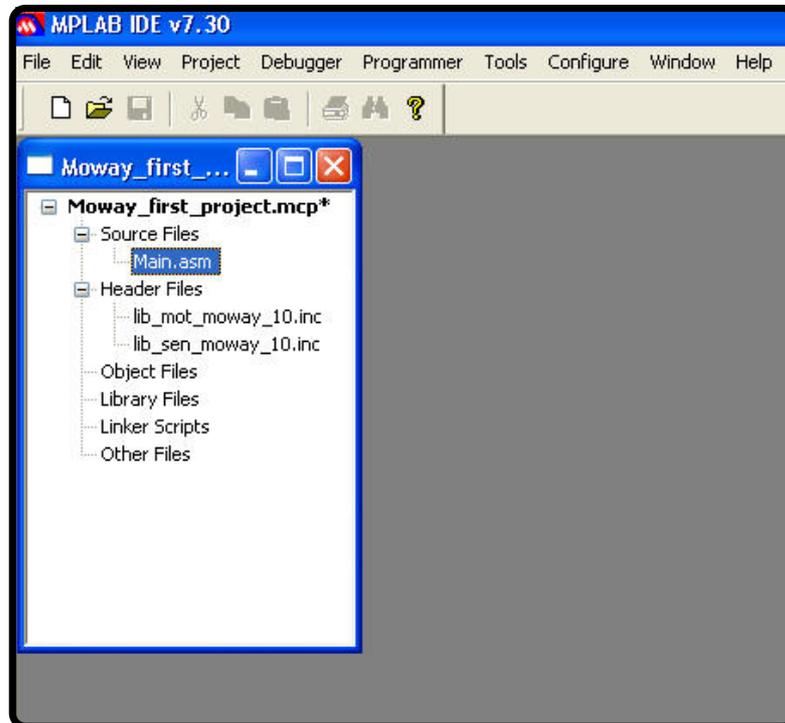


Imagen 36. Proyecto con .ASM

## 6.2. Primer programa en ensamblador

Para hacer el primer programa es necesario haber creado un proyecto (capítulo anterior). Este primer programa básico hará que Moway evite los obstáculos.

1. En primer lugar hay que añadir en el archivo Main.ASM el pic que tiene Moway instalado: list p=16F876A.
2. También es necesario añadir la librería de este microcontrolador en la carpeta del proyecto que encontrará en el directorio de instalación de MPLAB o en los programas de prueba del pack de Moway. Una vez copiada la librería en la carpeta se debe incluir en el Main: #include "P16F876A.INC".
3. El próximo paso es añadir los vectores de inicio y reset, e incluir las librerías de Moway.
4. A continuación se llama a la función SEN\_CONFIG encargada de configurar las entradas y salidas del microcontrolador.
5. Se añade el parpadeo de uno de los leds.
6. Probar programa en Moway y comprobar que el led verde parpadea.

```

1  list p=16F876A
2  ;* Definición de los registros internos del PIC
3  #include "P16F876A.INC"
4  ;* Vector de reset
5  org 0x00
6  goto INIT
7  ;* Memoria de programa *
8  org 0x05
9  ;*****
10 ;* Incluir las librerías de Moway
11 #include "lib_mot_moway_10.inc"
12 #include "lib_sen_moway_10.inc"
13
14
15 INIT
16
17     call    SEN_CONFIG
18
19 ;***** [MAIN PROGRAM] *****
20 MAIN
21
22
23     call    LED_TOP_GREEN_ON_OFF
24
25
26
27 BUCLE
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47     goto    BUCLE
48 ;*****
49
50 END

```

**Imagen 37. Primer programa: configuración y led**

7. Para detectar obstáculos se llama a la función SEN\_OBS\_DIG en el bucle infinito que devolverá en las variables SEN\_OBS\_R y SEN\_OBS\_L si tenemos obstáculo o no.
8. Si encuentra obstáculo enciende los led delanteros.
9. Probar el programa en Moway y comprobar que los led se encienden cuando se acerca un objeto a la parte delantera.

```

1      list p=16F876A
2      ;* Definición de los registros internos del PIC
3      #include "P16F876A.INC"
4      ;* Vector de reset
5      org    0x00
6      goto  INIT
7      ;* Memoria de programa *
8      org    0x05
9      ;*****
10     ;* Incluir las librerías de Moway
11     #include "lib_mot_moway_10.inc"
12     #include "lib_sen_moway_10.inc"
13
14
15     INIT
16
17     call   SEN_CONFIG
18
19     ;*****[MAIN PROGRAM]*****
20     MAIN
21
22
23     call   LED_TOP_GREEN_ON_OFF
24
25
26
27     BUCLE
28
29     call   SEN_OBS_DIG
30
31     btfsc  SEN_OBS_R,0
32     call   LED_R_ON
33     btfss  SEN_OBS_R,0
34     call   LED_R_OFF
35
36     btfsc  SEN_OBS_L,0
37     call   LED_L_ON
38     btfss  SEN_OBS_L,0
39     call   LED_L_OFF
40
41
42
43     goto  BUCLE
44
45     ;*****
46
47
48
49
50     END

```

**Imagen 38. Primer programa: detección de obstáculo**

10. Añadimos movimiento al robot: comando recto indefinidamente hasta que encuentra un obstáculo.
11. Cuando encuentra obstáculo se manda un comando para que realice una rotación de 180° y enciende el led rojo superior (los led delanteros no funcionarán). El robot esperará hasta que el comando termine y continuará en un movimiento recto.

```

;*****[MAIN PROGRAM]*****
MAIN

;Parpadeo del Led verde
call    LED_TOP_GREEN_ON_OFF

;Se mueve hacia adelante
movlw   .70
movwf   MOT_VEL
movlw   .0
movwf   MOT_T_DIST_ANG
bsf     MOT_CON,FWDBACK
bsf     MOT_CON,COMTYPE
call    MOT_STR

BUCLE

;Se comprueba los sensores de obstaculo
call    SEN_OBS_DIC

;Si hay un obstaculo en la derecha se enciende el LED derecho
btfsc   SEN_OBS_L,0
call    OBS_SI

;Si hay un obstaculo en la izquierda se enciende el LED izquierdo
btfsc   SEN_OBS_R,0
call    OBS_SI

goto    BUCLE

;*****
OBS_SI
call    LED_TOP_RED_ON_OFF
;Rotación a la derecha respecto al centro al 70% de velocidad 180.72°
movlw   .70
movwf   MOT_VEL
movlw   .50
movwf   MOT_T_DIST_ANG
movlw   0x01
movwf   MOT_CENWHEEL
bsf     MOT_CON,FWDBACK
bcf     MOT_CON,COMTYPE
bsf     MOT_CON,RL
call    MOT_ROT
;Hasta que el comando no termine no se hace nada
btfs    MOT_END
goto    $-1
;Se mueve hacia adelante
movlw   .70
movwf   MOT_VEL
movlw   .0
movwf   MOT_T_DIST_ANG
bsf     MOT_CON,FWDBACK
bsf     MOT_CON,COMTYPE
call    MOT_STR
return

END

```

**Imagen 39. Primer programa: movimiento con detección**

Este proyecto se suministra en el pack de Moway.

## 6.3. Librerías

### 6.3.1. Librería sensores Moway ensamblador

Existen dos librerías en ensamblador que pueden ser incluidas en cualquier proyecto de Moway que permiten al usuario controlar los sensores con gran facilidad. Ambas son idénticas salvo el hecho de que una de ellas (01) tiene la posibilidad de reubicar tanto el código como las variables (utilizando proyectos del MPLAB IDE).

El conjunto de las funciones de la librería ocupan unas 400 palabras de memoria de programa y 9 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\_STATUS

Esta variable de sólo lectura informa de la validez de los datos obtenidos por la lectura de los sensores.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Sin uso	SENOK						
-	-	-	-	-	-	-	

Bit 7-1: **Sin uso**

Bit 0: **SENOK:** Muestra si el sensor se ha leído correctamente.  
**1** = Lectura correcta. Datos de salida válidos.  
**0** = Lectura incorrecta. Datos de salida inválidos.

## **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 las librerías lib\_sen\_moway\_10 y lib\_sen\_moway\_11 existen una serie de funciones que están orientadas al control de los sensores y de los diodos LED de Moway.

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

**Tabla 11. Resumen de funciones en ensamblador**

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

## SEN\_CONFIG

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

**Tabla 12. Conexiones PIC-sensores**

Pin PIC	I/O	Sensor
<b>PORTA</b>		
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
<b>PORTB</b>		
RB1	O	Trasmisor sensor línea
RB2	O	Trasmisor infrarrojo
RB4	O	LED inferior derecho
RB6	O	LED superior verde
<b>PORTC</b>		
RC6	O	LED inferior izquierdo

## SEN\_LIGHT

<i>Variables de salida</i>	
SEN_LIGHT_P	Porcentaje de la luz incidente.

La función SEN\_LIGHT captura el valor analógico dependiente de la luz incidente en el fototransistor. Para ello se deben seguir los siguientes pasos:

- Activar el ADC.
- Esperar el tiempo de adquisición de datos (100us).
- Leer el valor analógico.
- Con el voltaje analógico medido se calcula el porcentaje de la luz incidente.
- Se copia el dato en la variable SEN\_LIGHT\_P.

## SEN\_OBS\_DIG

<i>Variables de salida</i>	
SEN_OBS_R	Indica si existe o no objeto en la parte derecha
SEN_OBS_L	Indica si existe o no objeto en la parte izquierda
<i>Salida</i>	
SEN_STATUS: SENOK	

Esta función indica si un obstáculo se encuentra en la parte delantera derecha o en la parte delantera izquierda. Para ello se han de seguir los siguientes pasos:

- Antes de mandar el pulso de luz infrarroja asegurarse de que no exista ninguna fuente de ruido que interfiera.
- Se manda pulso de luz infrarroja para la detección del obstáculo. Si hay algún obstáculo la luz rebotará y esta señal será captada por el receptor infrarrojo.
- Se comprueba si se presenta alguna señal en los dos receptores IR.
- Se copia el valor del receptor digital en las variables de salida.
- Se desactiva diodo infrarrojo.
- Se comprueba que no haya ninguna señal interferente.
- Si no se presenta ninguna señal interferente y el proceso se ejecuta sin problemas el flag SENOK es activado.
- 

## SEN\_OBS\_ANALOG

<i>Variables de salida</i>	
SEN_OBS_R	Indica la distancia a la que se encuentra un obstáculo en la parte derecha del robot
SEN_OBS_L	Indica la distancia a la que se encuentra un obstáculo en la parte izquierda del robot
<i>Salida</i>	
SEN_STATUS: SENOK	

Esta función indica si un obstáculo se encuentra en la parte delantera derecha o en la parte delantera izquierda y la distancia al robot. Para ello se han de seguir los siguientes pasos:

- Antes de mandar el pulso de luz infrarroja asegurarse de que no exista ninguna fuente de ruido que interfiera.
- Se manda pulso de luz infrarroja para la detección del obstáculo.
- Se comprueba si se presenta alguna señal en los dos receptores IR.
- Se copia el valor del receptor analógico en las variables de salida. Cuanto mayor sea este valor más cerca se encontrará el obstáculo.
- Se desactiva diodo infrarrojo.
- Se comprueba que no haya ninguna señal interferente.
- Si no se presenta señal interferente y el proceso se ejecuta sin problemas el flag SENOK es activado.

## SEN\_LINE\_DIG

<i>Variables de salida</i>	
SEN_LINE_R	Indica si el sensor derecho se encuentra sobre una superficie de tonalidad oscura.
SEN_LINE_L	Indica si el sensor izquierdo se encuentra sobre una superficie de tonalidad oscura.

La función SEN\_LINE\_DIG indica si los sensores están sobre una superficie oscura o no. Para ello se deben seguir los siguientes pasos:

- Activar diodo izquierdo y derecho.
- Esperar el tiempo de adquisición de datos (900us).
- Se lee el sensor derecho.
- Se mueve el dato a la variable SEN\_LINE\_R. Si la superficie es oscura (la luz no se refleja) obtendremos un '1' en la variable.
- Se lee el sensor izquierdo.
- Se mueve el dato a la variable SEN\_LINE\_L. Si la superficie es oscura (la luz no se refleja) obtendremos un '1' en la variable.

## SEN\_LINE\_ANALOG

<i>Variables de salida</i>	
SEN_LINE_R	Indica la tonalidad detectada por el sensor de línea derecho. Esta luz reflejada depende del color y del material de la superficie.
SEN_LINE_L	Indica la tonalidad detectada por el sensor de línea izquierdo. Esta luz reflejada depende del color y del material de la superficie.

La función SEN\_LINE\_ANALOG indica la luz que se ha reflejado en los optoacopladores. Para ello se deben seguir los siguientes pasos:

- Activar diodo izquierdo y derecho.
- Esperar el tiempo de adquisición de datos (900us).
- Se lee el sensor derecho.
- Mover ese dato a la variable SEN\_LINE\_R. Cuanto más alto sea este valor más oscura será la superficie.
- Se lee el sensor izquierdo.
- Lee el valor analógico.
- Mover ese dato a la variable SEN\_LINE\_L. Cuanto más alto sea este valor más oscura será la superficie.

#### **LED\_R\_ON**

Enciende el diodo LED inferior derecho.

#### **LED\_L\_ON**

Enciende el diodo LED inferior izquierdo.

#### **LED\_TOP\_RED\_ON**

Enciende el diodo LED superior rojo.

#### **LED\_TOP\_GREEN\_ON**

Enciende el diodo LED superior verde.

#### **LED\_R\_OFF**

Apaga el diodo LED inferior derecho.

#### **LED\_L\_OFF**

Apaga el diodo LED inferior izquierdo.

#### **LED\_TOP\_RED\_OFF**

Apaga el diodo LED superior rojo.

#### **LED\_TOP\_GREEN\_OFF**

Apaga el diodo LED superior verde.

#### **LED\_R\_ON\_OFF**

Parpadeo del diodo LED inferior derecho.

#### **LED\_L\_ON\_OFF**

Parpadeo del diodo LED inferior izquierdo.

#### **LED\_TOP\_RED\_ON\_OFF**

Parpadeo del diodo LED superior rojo.

#### **LED\_TOP\_GREEN\_ON\_OFF**

Parpadeo del diodo LED superior verde.

### **6.3.2. Librería motores Moway ensamblador**

Existen dos librerías en ensamblador que pueden ser incluidas en cualquier proyecto de Moway que permiten al usuario controlar el sistema motriz con gran facilidad. Ambas son idénticas salvo el hecho de que una de ellas (01) tiene la posibilidad de reubicar tanto el código como las variables (utilizando proyectos del MPLAB IDE).

El conjunto de las funciones de la librería ocupan unas 510 palabras de memoria de programa y 21 bytes de memoria de datos.

Es importante saber que cada llamada a cualquier función de la librería utiliza tres niveles adicionales de la pila de llamadas. Esto es, antes de llamar a una de estas funciones debe de haber al menos cuatro 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 el módulo motor 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. El formato de las tramas I2C del Sistema Motriz se puede observar en las siguientes figuras. Cada una de estas tramas tiene una duración de 350us.

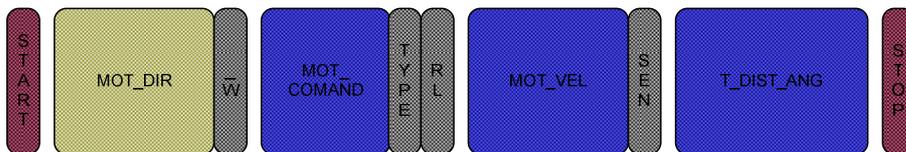


Imagen 40. Formato de comandos: MOT\_STR, MOT\_CHA\_VEL

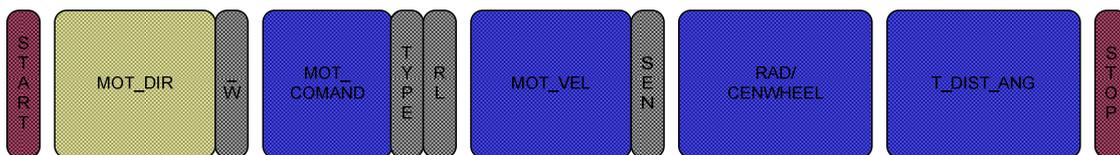


Imagen 41. Formato de comandos: MOT\_CUR, MOT\_ROT

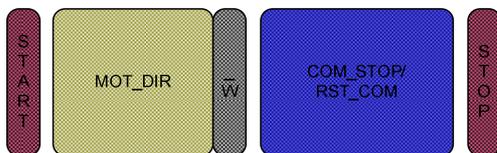


Imagen 42. Formato de comandos: MOT\_STOP, MOT\_RST

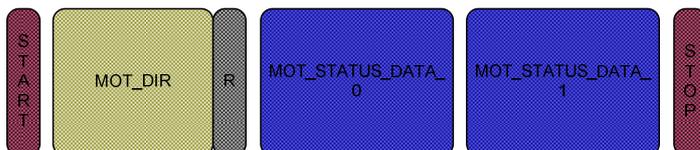


Imagen 43. Formato de comando: MOT\_FDBCK

## Variables

### MOT\_STATUS

Registro que indica el estado del comando.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Sin uso	DWRONG	COMOK					
-	-	-	-	-	-		

Bit 7-2: **Sin uso**

- Bit 1:       **DWRONG:** Muestra si los datos son incorrectos.  
          **1** = Datos incorrectos.  
          **0** = Datos correctos.
- Bit 0:       **COMOK:** Muestra si el comando ha sido enviado correctamente por I2C.  
          **1** = Envío correcto.  
          **0** = Envío incorrecto.

## MOT\_CON

Registro de control. En este registro se definen parámetros de los comandos.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Sin uso	COMTYPE	RL	FWDBACK				
-	-	-	-	-			

- Bit 7-3:       **Sin uso**
- Bit 2:       **COMTYPE:** Tipo de comando.  
          **1** = Tiempo.  
          **0** = Distancia o ángulo (en MOT\_ROT).
- Bit 1:       **RL:** Derecha o Izquierda  
          **1** = Derecha.  
          **0** = Izquierda.
- Bit 0:       **FWDBACK:** Adelante o hacia atrás.  
          **1** = Adelante.  
          **0** = Atrás.

## MOT\_VEL

Velocidad deseada en el comando.

## MOT\_T\_DIST\_ANG

Según el valor de COMTYPE y del comando, esta variable será el tiempo, distancia o el ángulo.

## MOT\_CENWHEEL

Rotación sobre el centro o sobre una de las ruedas del robot.

## MOT\_RAD

Radio para el comando MOT\_CUR.

## MOT\_RST\_COM

Tipo de reset que se desea.

## MOT\_STATUS\_COM

Tipo de dato que se quiere leer del motor.

## 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\_10 y lib\_mot\_moway\_11 existen una serie de funciones que están orientadas al control del sistema de motores de Moway.

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

**Tabla 13. Resumen de funciones en ensamblador de lib\_mot\_moway\_10**

Nombre	Entrada	Retorno	Descripción
MOT_CONFIG	-	-	Configuración para la comunicación con los motores
MOT_STR	MOT_VEL MOT_T_DIST  <i>MOT_CON</i> FWDBACK COMTYPE	<i>MOT_STATUS</i> COMOK DWRONG	Comando para movimiento en línea recta
MOT_CHA_VEL	MOT_VEL MOT_T_DIT  <i>MOT_CON</i> FWDBACK COMTYPE RL	<i>MOT_STATUS</i> COMOK DWRONG	Comando para cambiar la velocidad de un motor
MOT_ROT	MOT_VEL MOT_CENWHEEL MOT_T_ANG  <i>MOT_CON</i> FWDBACK COMTYPE RL	<i>MOT_STATUS</i> COMOK DWRONG	Comando para realizar la rotación del robot
MOT_CUR	MOT_VEL MOT_RAD MOT_T_DIST  <i>MOT_CON</i> FWDBACK COMTYPE RL	<i>MOT_STATUS</i> COMOK DWRONG	Comando para realizar una curva
MOT_STOP	-	<i>MOT_STATUS</i> COMOK DWRONG	Comando para detener el robot
MOT_RST	RST_COM	<i>MOT_STATUS</i> COMOK DWRONG	Comando para resetear las variables temporales de tiempo y distancia

<b>MOT_FDBCK</b>	STATUS_COM	MOT_STATUS_ DATA_0 MOT_STATUS_ DATA_1  <i>MOT_STATUS</i> COMOK DWRONG	Comando para ver el estado de los motores
------------------	------------	--	---

## MOT\_CONFIG

Esta función configura las entradas y salidas para que el microcontrolador pueda comunicarse con el Sistema Motriz.

**Tabla 14. Conexiones PIC-motores**

Pin PIC	I/O	Sensor
<b>PORTB</b>		
RB5	I	Indica cuándo el motor termina el comando
<b>PORTC</b>		
RC0	O	SCL del protocolo I2C
RC1	O	SDA del protocolo I2C

El puerto RB5 nos indica la finalización de un comando. Este puerto tiene la etiqueta MOT\_END en la librería.

Ejemplo:

*;Recto adelante al 100% de velocidad 10 segundos (100ms x 100)*

```
movlw    .100
movwf    MOT_VEL
movlw    .100
movwf    MOT_T_DIST_ANG
bsf      MOT_CON,FWDBACK
bsf      MOT_CON,COMTYPE
call     MOT_STR
```

*;Hasta que el comando no termine no se hace nada*

```
btfs    MOT_END
goto     $-1
```

## MOT\_STR

<i>Entrada</i>			
MOT_VEL	Velocidad deseada	0	100
MOT_CON, FWDBACK	Sentido de la marcha	1-FWD	0-BACK
MOT_CON, COMTYPE	Tipo de comando	1-TIME	0-DIST
MOT_T_DIST	El valor de Tiempo	0	255
	Distancia	0	255
<i>Variables de salida</i>			
FLAGS MOT_STATUS: COMOK y DWRONG			

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.

Ejemplo:

**;Recto adelante al 100% de velocidad 10 segundos (100ms x 100)**

```
movlw    .100
movwf    MOT_VEL
movlw    .100
movwf    MOT_T_DIST_ANG
bsf      MOT_CON,FWDBACK
bsf      MOT_CON,COMTYPE
call     MOT_STR
```

**;Recto hacia atras al 15% de velocidad 170mm (1.7mm x 100)**

```
movlw    .15
movwf    MOT_VEL
movlw    .100
movwf    MOT_T_DIST_ANG
bcf      MOT_CON,FWDBACK
bcf      MOT_CON,COMTYPE
call     MOT_STR
```

## MOT\_CHA\_VEL

<b>Entrada</b>			
MOT_VEL	Velocidad deseada	0	100
MOT_CON, FWDBACK	Sentido de la marcha	1-FWD	0-BACK
MOT_CON, RL	Izquierda o derecha	1-RIGHT	0-LEFT
MOT_CON, COMTYPE	Tipo de comando	1-TIME	0-DIST
MOT_T_DIST	El valor de Tiempo	0	255
	Distancia	0	255
<b>Variables de salida</b>			
FLAGS MOT_STATUS: COMOK y DWRONG			

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.

Ejemplo:

**;Cambiar velocidad (80% adelante) al motor derecho durante 10 segundos  
;(100ms x 100)**

```
movlw    .80
movwf    MOT_VEL
movlw    .100
movwf    MOT_T_DIST_ANG
bsf      MOT_CON,FWDBACK
bsf      MOT_CON,COMTYPE
bsf      MOT_CON,RL
call     MOT_CHA_VEL
```

**;Cambiar velocidad (20% atrás) al motor izquierdo y hacer una distancia de 170 mm  
;(1.7mm //x 100)**

```
movlw    .20
movwf    MOT_VEL
movlw    .100
movwf    MOT_T_DIST_ANG
bcf      MOT_CON,FWDBACK
bcf      MOT_CON,COMTYPE
bcf      MOT_CON,RL
call     MOT_CHA_VEL
```

## MOT\_ROT

<b>Entrada</b>			
MOT_VEL	Velocidad deseada	0	100
MOT_CON, FWDBACK	Sentido de la marcha	1-FWD	0-BACK
MOT_CENWHEEL	Sobre centro o rueda	0x01-CE	0x00-WH
MOT_CON, RL	Derecha o izquierda	1-RIGHT	0-LEFT
MOT_CON, COMTYPE	Tipo de comando	1-TIME	0-ANG
MOT_T_ANG	El valor de Tiempo Ángulo	0	255
		0	100
<b>Variables de salida</b>			
FLAGS MOT_STATUS: COMOK y DWRONG			

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 como calcular el valor de MOT\_T\_ANG teniendo en cuenta el ángulo de rotación deseado. Si la rotación se produce sobre una de las ruedas se obtiene más resolución. Por otro lado, hay que tener en cuenta la inercia mecánica por lo que se aconseja reducir la velocidad para conseguir una mayor precisión.

**Ecuación 1. MOT\_T\_ANG en rotación sobre el centro**

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

**Ecuación 2. MOT\_T\_ANG en rotación sobre una rueda**

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

Ejemplo:

;Rotación respecto al centro a la derecha al 80% de velocidad durante 10 segundos  
;(100ms x 100)

```
movlw    .80
movwf    MOT_VEL
movlw    .100
movwf    MOT_T_DIST_ANG
movlw    0x01
movwf    MOT_CENWHEEL
bsf      MOT_CON,FWDBACK
bsf      MOT_CON,COMTYPE
bsf      MOT_CON,RL
call     MOT_ROT
```

;Rotación respecto la rueda izquierda adelante al 20% de velocidad 180.72°

```
movlw    .20
movwf    MOT_VEL
movlw    .50
movwf    MOT_T_DIST_ANG
movlw    0x00
movwf    MOT_CENWHEEL
bsf      MOT_CON,FWDBACK
bcf      MOT_CON,COMTYPE
bcf      MOT_CON,RL
call     MOT_ROT
```

**MOT\_CUR**

<b>Entrada</b>			
MOT_VEL	Velocidad deseada	0	100
MOT_CON, FWDBACK	Sentido de la marcha	1-FWD	0-BACK
MOT_RAD	Radio	0	100
MOT_CON, RL	Derecha o izquierda	1-RIGHT	0-LEFT
MOT_CON, COMTYPE	Tipo de comando	1-TIME	0-DIST
MOT_T_DIST	El valor de Tiempo	0	255
	Distancia	0	255
<b>Variables de salida</b>			
FLAGS MOT_STATUS: COMOK y DWRONG			

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 restará o se sumará a la velocidad global del robot. Esto es, si la velocidad especificada es 50 y el radio 10, uno de los motores tendrá 60 de velocidad y el otro 40. Por lo tanto el radio tiene que cumplir la siguiente restricción:

**Ecuación 3. Condición 1 MOT\_RAD**  

$$0 \leq MOT\_VEL - MOT\_RAD \leq 100$$

**Ecuación 4. Condición 2 MOT\_RAD**  

$$0 \leq MOT\_VEL + MOT\_RAD \leq 100$$

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 mantendrá hasta que no se especifique otra orden. El motor cuenta la distancia recorrida por el motor que está en el exterior de la curva.

Ejemplo:

**;Curva hacia delante a la derecha al 50% con un radio de 10 durante 10 segundos**

**;(100ms x 100)**

**;VEL\_I=60**

**;VEL\_D=40**

movlw .50

movwf MOT\_VEL

movlw .100

movwf MOT\_T\_DIST\_ANG

movlw .10

movwf MOT\_RAD

bsf MOT\_CON,FWDBACK

bsf MOT\_CON,COMTYPE

bsf MOT\_CON,RL

call MOT\_CUR

**;Curva hacia atrás a la izquierda al 80% con un radio de 15 durante 170mm**

**;(1.7mm //x 100)**

**;VEL\_I=95**

**;VEL\_D=65**

movlw .80

movwf MOT\_VEL

movlw .100

movwf MOT\_T\_DIST\_ANG

movlw .15

movwf MOT\_RAD

bcf MOT\_CON,FWDBACK

bcf MOT\_CON,COMTYPE

bcf MOT\_CON,RL

call MOT\_CUR

## MOT\_STOP

### *Variables de salida*

FLAGS MOT_STATUS: COMOK
-------------------------

Comando para parar el robot.

Ejemplo:

```
;Parar Moway  
call MOT_STOP
```

## MOT\_RST

### *Entrada*

RST_COM	El parámetro que se desea resetear	RST_T RST_DIST RST_KM
---------	------------------------------------	-----------------------------

### *Variables de salida*

FLAGS MOT_STATUS: COMOK
-------------------------

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

Ejemplo:

```
;Reseteo del tiempo transcurrido  
movlw    RST_T  
movwf    MOT_RST_COM  
call     MOT_RST
```

```
;Reseteo de la distancia recorrida  
movlw    RST_D  
movwf    MOT_RST_COM  
call     MOT_RST
```

**MOT\_FDBCK**

<i>Entrada</i>		
STATUS_COM	El parámetro que se desea consultar	STATUS_T STATUS_A STATUS_V_R STATUS_V_L STATUS_D_R STATUS_D_L STATUS_KM
<i>Variables de salida</i>		
MOT_STATUS_DATA_0	Primer byte de respuesta (tiempo, ángulo, velocidad, distancia y primer byte del cuentakilómetros)	
MOT_STATUS_DATA_1	Segundo byte de respuesta (segundo byte del cuentakilómetros)	
FLAGS MOT_STATUS: COMOK y DWRONG		

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.

Esta función actualiza dos variables donde se guardará 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 (ej. Se puede pedir el tiempo transcurrido desde el último comando). Cuando se use STATUS\_KM hay que tener en cuenta los dos bytes. Este comando resulta muy útil para calcular la longitud de una línea mientras el robot la sigue.

Ejemplo:

*;Petición de tiempo transcurrido desde el último comando*

```
movlw    STATUS_T
movwf    MOT_STATUS_COM
call     MOT_FDBCK
```

*;Ej. Salida:*

```
;MOT_STATUS_DATA_0=0x7F => Han transcurrido 12.7 segundos desde el
;último comando
;MOT_STATUS_DATA_1=0xFF; => Dato no válido
```

*;Petición de distancia recorrida por el motor derecho desde el último comando*

```
movlw    STATUS_KM
movwf    MOT_STATUS_COM
call     MOT_FDBCK
```

*;Ej. Salida:*

```
;MOT_STATUS_DATA_0=0x08
;MOT_STATUS_DATA_1=0x01;
```

byte 1	byte 0
0x01	0x08
0000 0001	0000 0100
264	
Distancia: 264*1.7mm	
448.8mm	

## 7. Programación del Moway empleando CCS PIC C Compiler

PICC, de CCS, es un compilador de pago que soporta el microcontrolador PIC16F876A. En la web de Moway encontrará librerías para el manejo de los sensores, motores y módulo RF escritas para el compilador.

La gran ventaja que tiene es que el lenguaje que compila es C. El manejo de variables numéricas (int8, int16, etc.) y de estructuras de control de flujo (if, for, etc.) es muy sencillo y contienen gran cantidad de funciones pre-compiladas que facilitan enormemente la tarea de programación (I2C, SPI). Sin embargo, los programas generados son más grandes en cuanto a tamaño que en ensamblador.

En resumen:

- Muy interesante para comenzar a trabajar con Moway rápidamente.
- Muy interesante para realizar tareas sencillas o de complejidad media.
- No recomendable si el programa a realizar va a ser largo (en cuanto a código se refiere).
- No recomendable si el tiempo de respuesta es crítico.

### 7.1. Creación de un Proyecto

1. El primer paso es acceder al *Project Wizard* en la pestaña de *Project* del entorno gráfico y una vez indicado dónde queremos guardar el proyecto se elige el PIC16F876A, una frecuencia de reloj de 4000000Hz y en la sección *Fuses* se selecciona la opción *Crystal Osc <=4Mhz*. Para finalizar con la configuración se deshabilita el RS-232 de la sección de *Communications*.

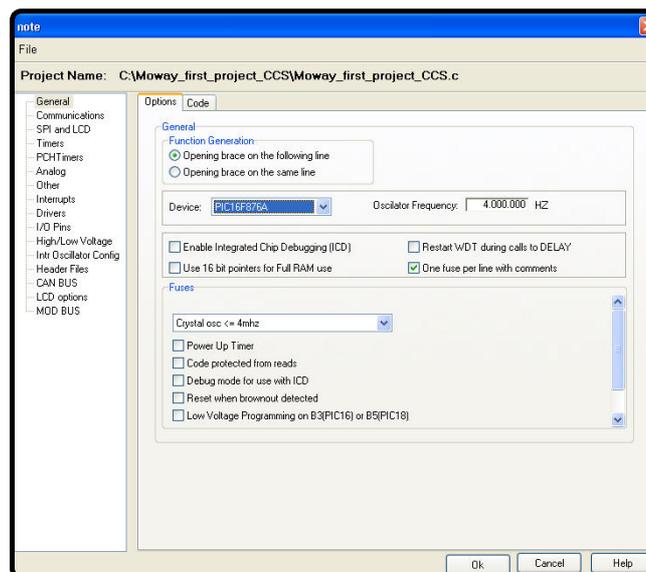


Imagen 44. Project Wizard General

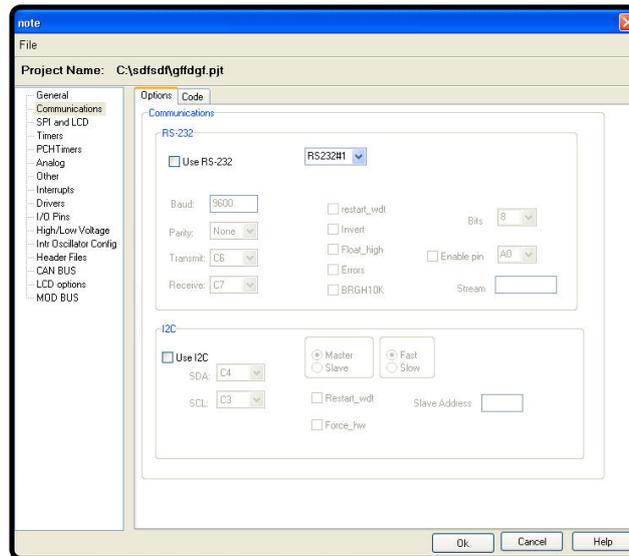


Imagen 45. Project Wizard Communications

2. El Project Wizard creará un fichero .C donde introducir el código fuente deseado y un archivo .h donde se definen diferentes aspectos de la configuración del microcontrolador.

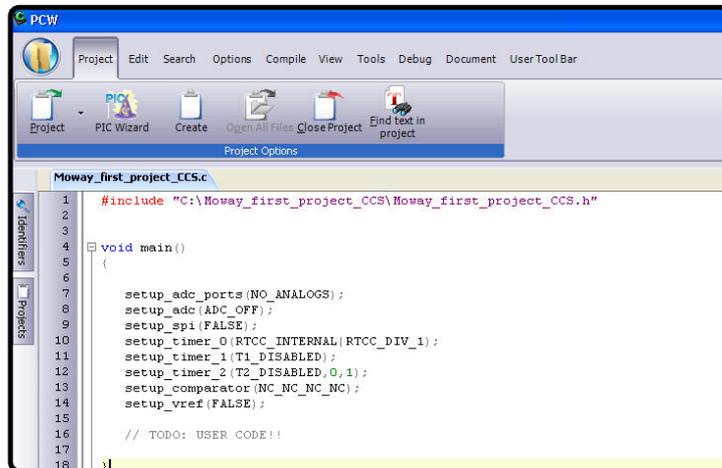


Imagen 46. Main del proyecto

3. Por último se copian las librerías dentro de la carpeta del nuevo proyecto y se incluyen en el fichero .C. (`#include`).

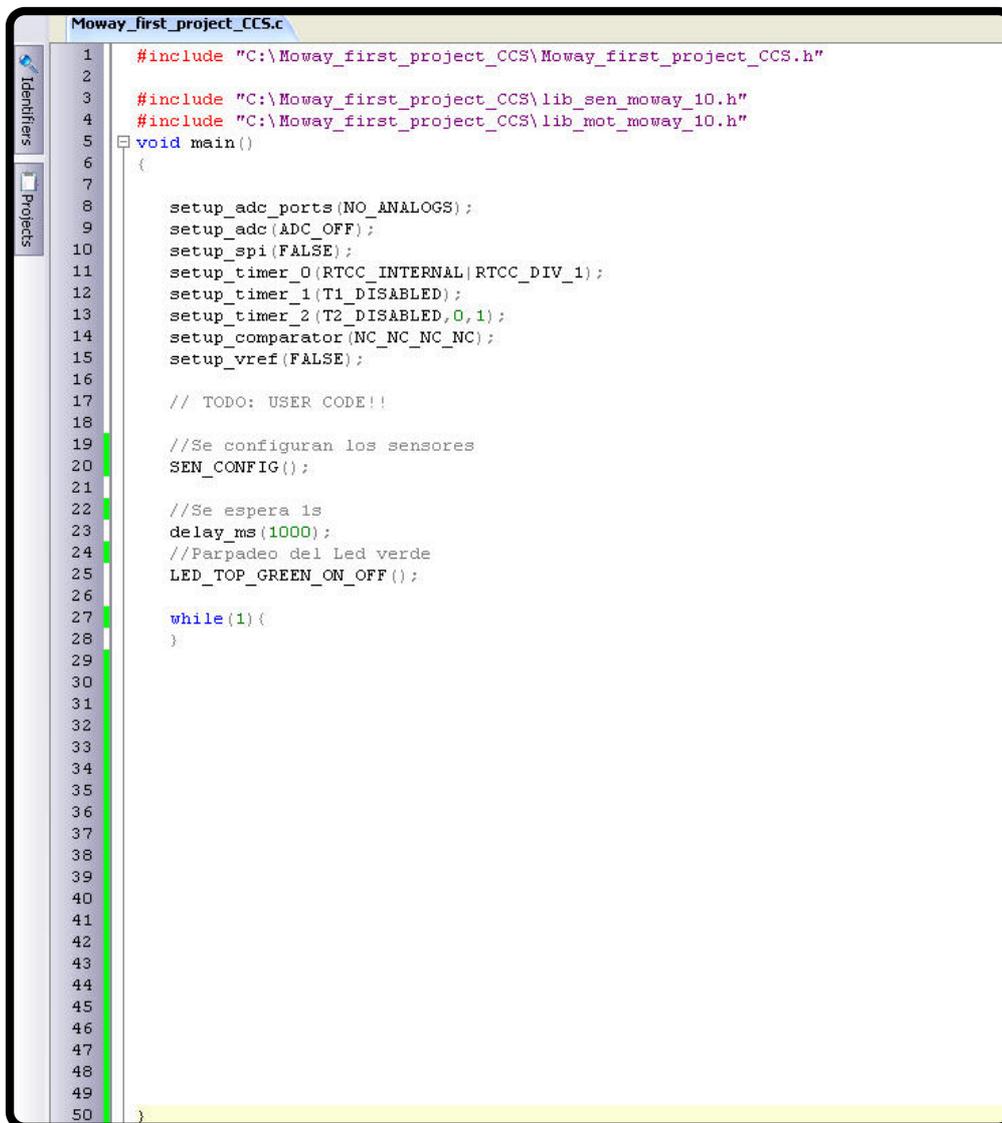
```
1 #include "C:\Moway_first_project_CCS\Moway_first_project_CCS.h"
2
3 #include "lib_sen_moway_10.h"
4 #include "lib_mot_moway_10.h"
5
6 void main()
7 {
8     setup_adc_ports(NO_ANALOGS);
9     setup_adc(ADC_OFF);
10    setup_spi(FALSE);
11    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
12    setup_timer_1(T1_DISABLED);
13    setup_timer_2(T2_DISABLED,0,1);
14    setup_comparator(NC_NC_NC_NC);
15    setup_vref(FALSE);
16
17    // TODO: USER CODE!!
18
19 }
```

**Imagen 47. Añadir librerías de Moway**

## **7.2. Primer programa en CCS**

Para hacer el primer programa es necesario haber creado un proyecto (capítulo anterior). Este primer programa básico hará que Moway evite los obstáculos.

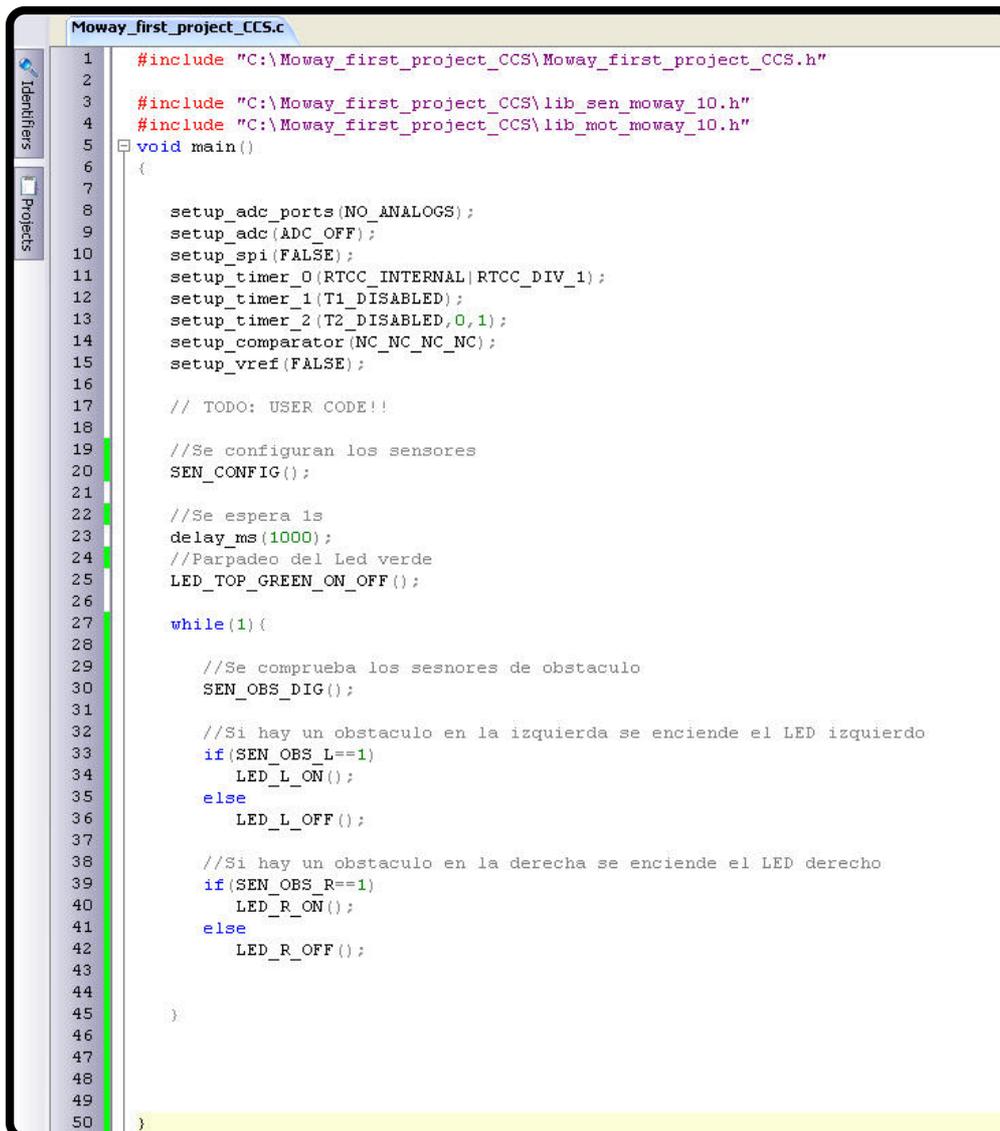
1. El primer paso es configurar los sensores. Se llama a la función SEN\_CONFIG() encargada de configurar las entradas y salidas del microcontrolador.
2. El siguiente paso es añadir un retardo de un segundo: delay\_ms(1000).
3. Se añade el parpadeo de uno de los leds.
4. Probar el programa y comprobar que después de esperar 1 segundo se enciende el led verde.



```
Moway_first_project_CCS.c
1  #include "C:\Moway_first_project_CCS\Moway_first_project_CCS.h"
2
3  #include "C:\Moway_first_project_CCS\lib_sen_moway_10.h"
4  #include "C:\Moway_first_project_CCS\lib_mot_moway_10.h"
5  void main()
6  {
7
8      setup_adc_ports(NO_ANALOGS);
9      setup_adc(ADC_OFF);
10     setup_spi(FALSE);
11     setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
12     setup_timer_1(T1_DISABLED);
13     setup_timer_2(T2_DISABLED,0,1);
14     setup_comparator(NC_NC_NC_NC);
15     setup_vref(FALSE);
16
17     // TODO: USER CODE!!
18
19     //Se configuran los sensores
20     SEN_CONFIG();
21
22     //Se espera 1s
23     delay_ms(1000);
24     //Parpadeo del Led verde
25     LED_TOP_GREEN_ON_OFF();
26
27     while(1){
28     }
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50 }
```

Imagen 48. Primer programa: configuración y parpadeo de led

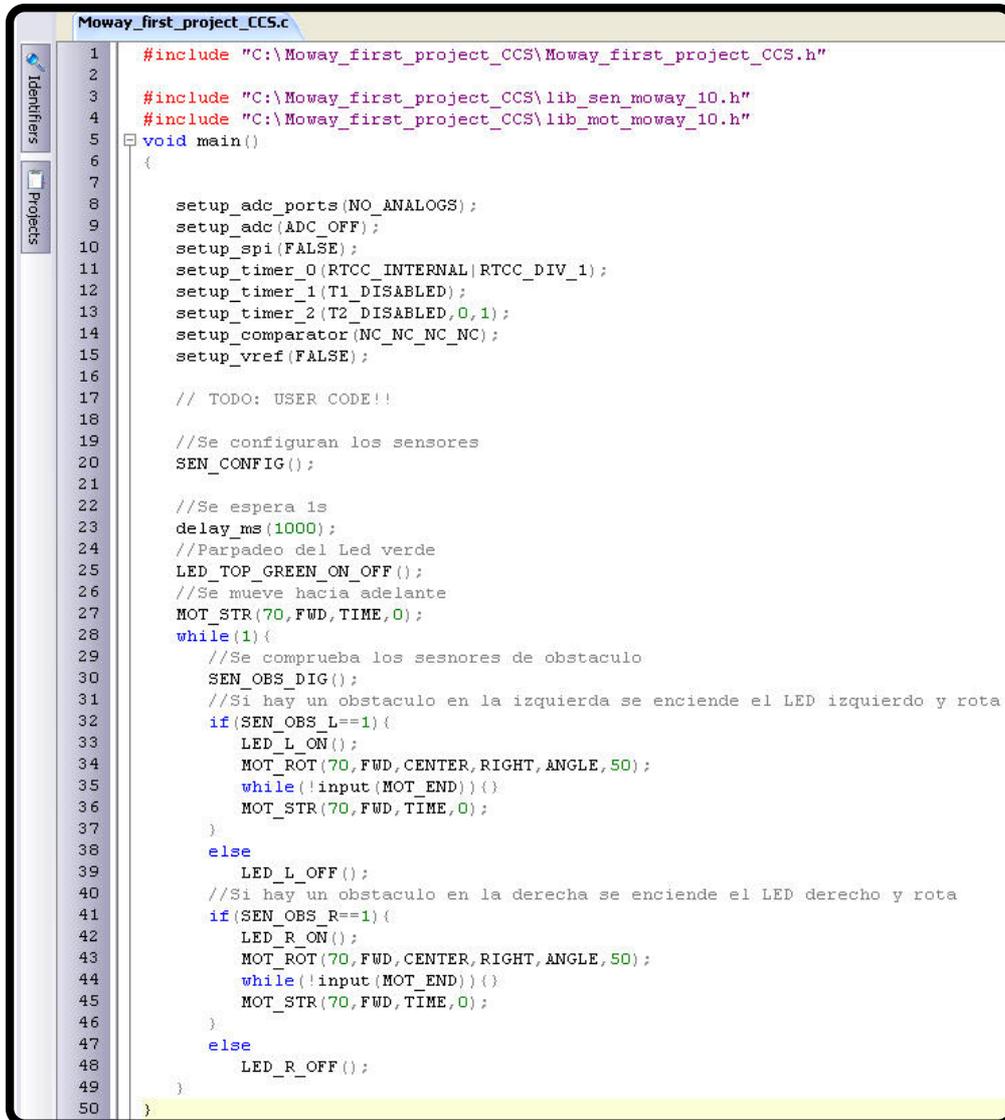
5. Para detectar obstáculos se llama a la función SEN\_OBS\_DIG() que devolverá en las variables SEN\_OBS\_R y SEN\_OBS\_L si tenemos obstáculo o no.
6. Si encuentra obstáculo enciende los led delanteros.
7. Probar el programa y comprobar que los led delanteros se encienden cuando detectan un obstáculo.



```
Moway_first_project_CCS.c
1  #include "C:\Moway_first_project_CCS\Moway_first_project_CCS.h"
2
3  #include "C:\Moway_first_project_CCS\lib_sen_moway_10.h"
4  #include "C:\Moway_first_project_CCS\lib_mot_moway_10.h"
5  void main()
6  {
7
8      setup_adc_ports(NO_ANALOGS);
9      setup_adc(ADC_OFF);
10     setup_spi(FALSE);
11     setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
12     setup_timer_1(T1_DISABLED);
13     setup_timer_2(T2_DISABLED,0,1);
14     setup_comparator(NC_NC_NC_NC);
15     setup_vref(FALSE);
16
17     // TODO: USER CODE!!
18
19     //Se configuran los sensores
20     SEN_CONFIG();
21
22     //Se espera 1s
23     delay_ms(1000);
24     //Parpadeo del Led verde
25     LED_TOP_GREEN_ON_OFF();
26
27     while(1){
28
29         //Se comprueba los sensores de obstaculo
30         SEN_OBS_DIG();
31
32         //Si hay un obstaculo en la izquierda se enciende el LED izquierdo
33         if(SEN_OBS_L==1)
34             LED_L_ON();
35         else
36             LED_L_OFF();
37
38         //Si hay un obstaculo en la derecha se enciende el LED derecho
39         if(SEN_OBS_R==1)
40             LED_R_ON();
41         else
42             LED_R_OFF();
43
44
45     }
46
47
48
49
50 }
```

Imagen 49. Primer programa: detección de obstáculo

8. Añadimos movimiento al robot: comando recto indefinidamente hasta que encuentra un obstáculo.
9. Cuando encuentra obstáculo se manda un comando para que realice una rotación de 180°. El robot esperará hasta que el comando termine y continuará en un movimiento recto.



```

1  #include "C:\Moway_first_project_CCS\Moway_first_project_CCS.h"
2
3  #include "C:\Moway_first_project_CCS\lib_sen_moway_10.h"
4  #include "C:\Moway_first_project_CCS\lib_mot_moway_10.h"
5  void main()
6  {
7
8      setup_adc_ports(NO_ANALOGS);
9      setup_adc(ADC_OFF);
10     setup_spi(FALSE);
11     setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
12     setup_timer_1(T1_DISABLED);
13     setup_timer_2(T2_DISABLED,0,1);
14     setup_comparator(NC_NC_NC_NC);
15     setup_vref(FALSE);
16
17     // TODO: USER CODE!!
18
19     //Se configuran los sensores
20     SEN_CONFIG();
21
22     //Se espera 1s
23     delay_ms(1000);
24     //Parpadeo del Led verde
25     LED_TOP_GREEN_ON_OFF();
26     //Se mueve hacia adelante
27     MOT_STR(70,FWD,TIME,0);
28     while(1){
29         //Se comprueba los sensores de obstaculo
30         SEN_OBS_DIG();
31         //Si hay un obstaculo en la izquierda se enciende el LED izquierdo y rota
32         if(SEN_OBS_L==1){
33             LED_L_ON();
34             MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,50);
35             while(!input(MOT_END)){}
36             MOT_STR(70,FWD,TIME,0);
37         }
38         else
39             LED_L_OFF();
40         //Si hay un obstaculo en la derecha se enciende el LED derecho y rota
41         if(SEN_OBS_R==1){
42             LED_R_ON();
43             MOT_ROT(70,FWD,CENTER,RIGHT,ANGLE,50);
44             while(!input(MOT_END)){}
45             MOT_STR(70,FWD,TIME,0);
46         }
47         else
48             LED_R_OFF();
49     }
50 }

```

**Imagen 50. Primer programa: movimiento con detección**

Este proyecto se suministra en el pack de Moway.

## **7.3. Librerías**

### **7.3.1. Librería sensores Moway en C para CCS**

Esta librería en CCS puede ser incluida en cualquier proyecto de Moway que permite al usuario controlar los sensores.

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\_10 existen una serie de funciones que están orientadas al control de los sensores y de los diodos LED de Moway.

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

**Tabla 15. Resumen de funciones en C**

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

**void SEN\_CONFIG()**

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

**Tabla 16. Conexiones PIC-sensores**

Pin PIC	I/O	Sensor
<b>PORTA</b>		
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
<b>PORTB</b>		
RB1	O	Trasmisor sensor línea
RB2	O	Trasmisor infrarrojo
RB4	O	LED inferior derecho
RB6	O	LED superior verde
<b>PORTC</b>		
RC6	O	LED inferior izquierdo

**void SEN\_LIGHT()**
**Variables de salida**

SEN_LIGHT_P	Porcentaje de la luz incidente.
-------------	---------------------------------

La función SEN\_LIGHT captura el valor analógico dependiente de la luz incidente en el fototransistor. Para ello debemos de seguir los siguientes pasos:

- Activar el ADC.
- Esperar el tiempo de adquisición de datos (100us).
- Leer el valor analógico.
- Con el voltaje analógico medido se calcula el porcentaje de la luz incidente.
- Se copia el dato en la variable SEN\_LIGHT\_P.

**int1 SEN\_OBS\_DIG()**
**Variables de salida**

SEN_OBS_R	Indica si existe o no objeto en la parte derecha
SEN_OBS_L	Indica si existe o no objeto en la parte izquierda

**Retorno de la Función**

1: Lectura correcta	Dato correcto
0: Lectura incorrecta	Dato incorrecto. Interferencia detectada.

Esta función indica si un obstáculo se encuentra en la parte delantera derecha o en la parte delantera izquierda. Para ello se han de seguir los siguientes pasos:

- Antes de mandar el pulso de luz infrarroja asegurarse de que no exista ninguna fuente de ruido que interfiera.
- Se manda pulso de luz infrarroja para la detección del obstáculo. Si hay algún obstáculo la luz rebotará y esta señal será captada por el receptor infrarrojo.
- Se comprueba si se presenta alguna señal en los dos receptores IR.
- Se copia el valor del receptor digital en las variables de salida.
- Se desactiva diodo infrarrojo.
- Se comprueba que no haya ninguna señal interferente.
- Si no se presenta ninguna señal interferente y el proceso se ejecuta sin problemas la función retorna un '1'.

#### ***int1 SEN\_OBS\_ANALOG()***

<b><i>Variables de salida</i></b>	
SEN_OBS_R	Indica la distancia a la que se encuentra un obstáculo en la parte derecha del robot
SEN_OBS_L	Indica la distancia a la que se encuentra un obstáculo en la parte izquierda del robot
<b><i>Retorno de la Función</i></b>	
1: Lectura correcta	Dato correcto
0: Lectura incorrecta	Dato incorrecto. Interferencia detectada.

Esta función indica si un obstáculo se encuentra en la parte delantera derecha o en la parte delantera izquierda y la distancia al robot. Para ello se han de seguir los siguientes pasos:

- Antes de mandar el pulso de luz infrarroja asegurarse de que no exista ninguna fuente de ruido que interfiera.
- Se manda pulso de luz infrarroja para la detección del obstáculo.
- Se comprueba si se presenta alguna señal en los dos receptores IR.
- Se copia el valor del receptor analógico en las variables de salida. Cuanto mayor sea este valor más cerca se encontrará el obstáculo.
- Se desactiva diodo infrarrojo.
- Se comprueba que no haya ninguna señal interferente.
- Si no se presenta ninguna señal interferente y el proceso se ejecuta sin problemas la función retorna un '1'.

**void SEN\_LINE\_DIG()**

<i>Variables de salida</i>	
SEN_LINE_R	Indica si el sensor derecho se encuentra sobre una superficie de tonalidad oscura.
SEN_LINE_L	Indica si el sensor izquierdo se encuentra sobre una superficie de tonalidad oscura.

La función SEN\_LINE\_DIG indica si los sensores están sobre una superficie oscura o no. Para ello se deben seguir los siguientes pasos:

- Activar diodo izquierdo y derecho.
- Esperar el tiempo de adquisición de datos (900us).
- Se lee el sensor derecho.
- Se mueve el dato a la variable SEN\_LINE\_R. Si la superficie es oscura (la luz no se refleja) obtendremos un '1' en la variable.
- Se lee el sensor izquierdo.
- Se mueve el dato a la variable SEN\_LINE\_L. Si la superficie es oscura (la luz no se refleja) obtendremos un '1' en la variable.

**void SEN\_LINE\_ANALOG()**

<i>Variables de salida</i>	
SEN_LINE_R	Indica la tonalidad detectada por el sensor de línea derecho. Esta luz reflejada depende del color y del material de la superficie.
SEN_LINE_L	Indica la tonalidad detectada por el sensor de línea izquierdo. Esta luz reflejada depende del color y del material de la superficie.

La función SEN\_LINE\_ANALOG indica la luz que se ha reflejado en los optoacopladores. Para ello se han de seguir los siguientes pasos:

- Activar diodo izquierdo y derecho.
- Esperar el tiempo de adquisición de datos (900us).
- Se lee el sensor derecho.
- Mover ese dato a la variable SEN\_LINE\_R. Cuanto más alto sea este valor más oscura será la superficie.
- Se lee el sensor izquierdo.
- Leer el valor analógico.
- Mover ese dato a la variable SEN\_LINE\_L. Cuanto más alto sea este valor más oscura será la superficie.

***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.

### **7.3.2. Librería motores Moway en C para CCS**

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

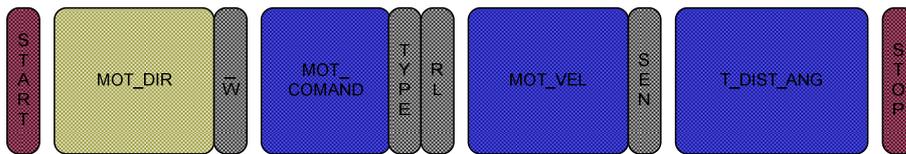
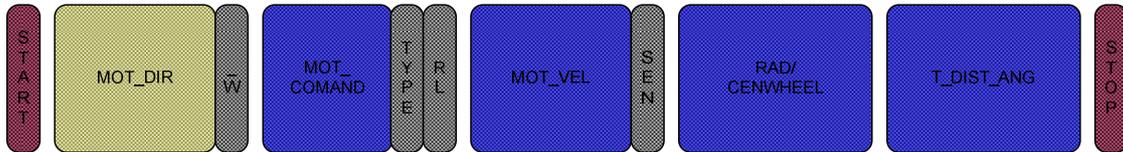
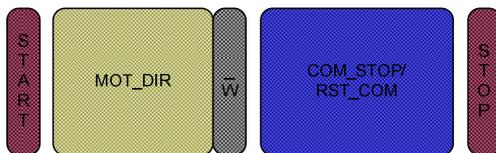
El conjunto de las funciones de la librería ocupan unas 1070 palabras de memoria de programa y 5 bytes de memoria de datos.

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 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. El formato de las tramas I2C del Sistema Motriz se puede observar en las siguientes figuras. Cada una de estas tramas tiene una duración de 350us.


**Imagen 51. Formato de comandos: MOT\_STR, MOT\_CHA\_VEL**

**Imagen 52. Formato de comandos: MOT\_CUR, MOT\_ROT**

**Imagen 53. Formato de comandos: MOT\_STOP, MOT\_RST**

**Imagen 54. Formato de comando: MOT\_FDBCK**

## ***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\_10** existen una serie de funciones que están orientadas al control del sistema de motores de Moway.

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

**Tabla 17. Resumen de funciones en C de lib\_mot\_moway\_10**

Nombre	Entrada	Retorno	Descripción
<i>void</i> MOT_CONFIG()	-	-	Configuración para la comunicación con los motores
<i>int8</i> MOT_STR ( <i>int</i> MOT_VEL, <i>int1</i> FWDBACK, <i>int1</i> COMTYPE, <i>int</i> 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
<i>int8</i> MOT_CHA_VEL ( <i>int</i> MOT_VEL, <i>int1</i> FWDBACK, <i>int1</i> RL, <i>int1</i> COMTYPE, <i>int</i> 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
<i>int8</i> MOT_ROT ( <i>int</i> MOT_VEL, <i>int1</i> FWDBACK, <i>int</i> MOT_CENWHEEL, <i>int1</i> RL, <i>int1</i> COMTYPE, <i>int</i> 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
<i>int8</i> MOT_CUR ( <i>int</i> MOT_VEL, <i>int1</i> FWDBACK, <i>int</i> MOT_RAD, <i>int1</i> RL, <i>int1</i> COMTYPE, <i>int</i> 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
<i>int1</i> MOT_STOP()	-	0: Envío correcto 1: Envío incorrecto	Comando para detener el robot
<i>int1</i> MOT_RST ( <i>int8</i> RST_COM)	RST_COM	0: Envío correcto 1: Envío incorrecto	Comando para resetear las variables temporales de tiempo y distancia
<i>int1</i> MOT_FDBCK ( <i>int8</i> 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

***void* MOT\_CONFIG()**

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

**Tabla 18. Conexiones PIC-motores**

Pin PIC	I/O	Sensor
<b>PORTB</b>		
RB5	I	Indica cuándo el motor termina el comando
<b>PORTC</b>		
RC0	O	SCL del protocolo I2C
RC1	O	SDA del protocolo I2C

El puerto RB5 nos indica la finalización de un comando. Este puerto tiene la etiqueta MOT\_END en la librería.

Ejemplo:

```
//Recto adelante al 100% de velocidad 10 segundos (100ms x 100)
MOT_STR(100, FWD, TIME, 100);
//Hasta que el comando no termine no se hace nada
while(!input(MOT_END)){ }
```

*int8* MOT\_STR(*int* MOT\_VEL, *int1* FWDBACK, *int1* COMTYPE,  
*int* MOT\_T\_DIST)

<b>Entrada</b>			
MOT_VEL	Velocidad deseada	0	100
FWDBACK	Sentido de la marcha	FWD	BACK
COMTYPE	Tipo de comando	TIME	DISTANCE
MOT_T_DIST	El valor de Tiempo	0	255
	Distancia	0	255
<b>Retorno de la Función</b>			
0: Envío correcto	El comando se ha enviado correctamente		
1: Envío incorrecto	El comando no se ha enviado. Problema de conexión		
2: Datos incorrectos	Los datos son incorrectos		

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.

Ejemplo:

```
//Recto adelante al 100% de velocidad 10 segundos (100ms x 100)
MOT_STR(100, FWD, TIME, 100);

//Recto hacia atras al 15% de velocidad 170mm (1.7mm x 100)
MOT_STR(15, BACK, DISTANCE, 100);
```

*int8* MOT\_CHA\_VEL (*int* MOT\_VEL, *int1* FWDBACK, *int1* RL, *int1* COMTYPE, *int* MOT\_T\_DIST)

<b>Entrada</b>			
MOT_VEL	Velocidad deseada	0	100
FWDBACK	Sentido de la marcha	FWD	BACK
RL	Derecha o izquierda	RIGHT	LEFT
COMTYPE	Tipo de comando	TIME	DISTANCE
MOT_T_DIST	El valor de Tiempo	0	255
	Distancia	0	255
<b>Retorno de la Función</b>			
0: Envío correcto	El comando se ha enviado correctamente		
1: Envío incorrecto	El comando no se ha enviado. Problema de conexión		
2: Datos incorrectos	Los datos son incorrectos		

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.

Ejemplo:

//Cambiar velocidad (80% adelante) al motor derecho durante 10 segundos

//(100ms x 100)

MOT\_CHA\_VEL(80, FWD, RIGHT, TIME, 100) ;

//Cambiar velocidad (20% atrás) al motor izquierdo y hacer una distancia de 170 mm

//(1.7mm //x 100)

MOT\_CHA\_VEL(20, BACK, LEFT, DISTANCE, 100) ;

*int8* MOT\_ROT (*int* MOT\_VEL, *int1* FWDBACK, *int* MOT\_CENWHEEL, *int1* RL, *int1* COMTYPE, *int* MOT\_T\_ANG)

<b>Entrada</b>			
MOT_VEL	Velocidad deseada	0	100
FWDBACK	Sentido de la marcha	FWD	BACK
MOT_CENWHEEL	Sobre centro o rueda	CENTER	WHEEL
RL	Derecha o izquierda	RIGHT	LEFT
COMTYPE	Tipo de comando	TIME	DISTANCE
MOT_T_ANG	El valor de Tiempo	0	255
	Ángulo	0	100
<b>Retorno de la Función</b>			
0: Envío correcto	El comando se ha enviado correctamente		
1: Envío incorrecto	El comando no se ha enviado. Problema de conexión		
2: Datos incorrectos	Los datos son incorrectos		

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. Si la rotación se produce sobre una de las ruedas se obtiene más resolución. Por otro lado, hay que tener en cuenta la inercia mecánica por lo que se aconseja reducir la velocidad para conseguir una mayor precisión.

**Ecuación 5. MOT\_T\_ANG en rotación sobre el centro**

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

**Ecuación 6. MOT\_T\_ANG en rotación sobre una rueda**

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

Ejemplo:

//Rotación respecto al centro a la derecha al 80% de velocidad durante 10 segundos  
 //(100ms x 100)

MOT\_ROT(80, FWD, CENTER, RIGHT, TIME, 100) ;

//Rotación respecto la rueda izquierda al 20% de velocidad 180.72°

MOT\_ROT(20, BACK, WHEEL, LEFT, ANGLE, 50) ;

*int8* MOT\_CUR (*int* MOT\_VEL, *int1* FWDBACK, *int* MOT\_RAD, *int1* RL, *int1* COMTYPE, *int* MOT\_T\_DIST)

<b>Entrada</b>			
MOT_VEL	Velocidad deseada	0	100
FWDBACK	Sentido de la marcha	FWD	BACK
MOT_RAD	Radio	0	100
RL	Derecha o izquierda	RIGHT	LEFT
COMTYPE	Tipo de comando	TIME	DISTANCE
MOT_T_DIST	El valor de Tiempo	0	255
	Distancia	0	255
<b>Retorno de la Función</b>			
0: Envío correcto	El comando se ha enviado correctamente		
1: Envío incorrecto	El comando no se ha enviado. Problema de conexión		
2: Datos incorrectos	Los datos son incorrectos		

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 restará o se sumará a la velocidad global del robot. Esto es, si la velocidad especificada es 50 y el radio 10, uno de los motores tendrá 60 de velocidad y el otro 40. Por lo tanto el radio tiene que cumplir la siguiente restricción:

**Ecuación 7. Condición 1 MOT\_RAD**  
 $0 \leq MOT\_VEL - MOT\_RAD \leq 100$

**Ecuación 8. Condición 2 MOT\_RAD**  
 $0 \leq MOT\_VEL + MOT\_RAD \leq 100$

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 mantendrá hasta que no se especifique otra orden. El motor cuenta la distancia recorrida por el motor que está en el exterior de la curva.

Ejemplo:

```
//Curva a derechas al 50% con un radio de 10 durante 10 segundos
//(100ms x 100)
//VEL_I=60
//VEL_D=40
MOT_CUR(50, FWD, 10, RIGHT, TIME, 100) ;
```

```
//Curva a izquierdas al 80% con un radio de 15 durante 170mm
//(1.7mm //x 100)
//VEL_I=95
//VEL_D=65
MOT_CUR(80, BACK, 15, LEFT, DISTANCE, 100) ;
```

*int1* MOT\_STOP()

<b>Retorno de la Función</b>	
0: Envío correcto	El comando se ha enviado correctamente
1: Envío incorrecto	El comando no se ha enviado. Problema de conexión

Comando para parar el robot.

Ejemplo:

```
//Parar Moway
MOT_STOP() ;
```

**int1 MOT\_RST (int8 RST\_COM)**

<b>Entrada</b>		
RST_COM	El parámetro que se desea resetear	RST_T RST_DIST RST_KM
<b>Retorno de la Función</b>		
0: Envío correcto	El comando se ha enviado correctamente	
1: Envío incorrecto	El comando no se ha enviado. Problema de conexión	

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

Ejemplo:

```
//Reseteo del tiempo transcurrido
MOT_RST(RST_T);
```

```
//Reseteo de la distancia recorrida
MOT_RST(RST_D);
```

**int1 MOT\_FDBCK (int8 STATUS\_COM)**

<b>Entrada</b>		
STATUS_COM	El parámetro que se desea consultar	STATUS_T STATUS_A STATUS_V_R STATUS_V_L STATUS_D_R STATUS_D_L STATUS_KM
<b>Variables de salida</b>		
MOT_STATUS_DATA_0	Primer byte de respuesta (tiempo, ángulo, velocidad, distancia y primer byte del cuentakilómetros)	
MOT_STATUS_DATA_1	Segundo byte de respuesta (segundo byte del cuentakilómetros)	
<b>Retorno de la Función</b>		
0: Envío correcto	El comando se ha enviado correctamente	
1: Envío incorrecto	El comando no se ha enviado. Problema de conexión	

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.

Esta función actualiza dos variables dónde se guardará 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 (ej. Se puede pedir el tiempo transcurrido desde el último comando). Cuando se use STATUS\_KM hay que tener en cuenta los dos bytes. Este comando resulta muy útil para calcular la longitud de una línea mientras el robot la sigue.

Ejemplo:

//Petición de tiempo transcurrido desde el último comando

MOT\_FDBCK(STATUS\_T);

//Ej. Salida:

//MOT\_STATUS\_DATA\_0=0x7F => Han transcurrido 12.7 segundos desde el

//último comando

//MOT\_STATUS\_DATA\_1=0xFF; => Dato no válido

//Petición de distancia recorrida por el motor derecho desde el último comando

MOT\_FDBCK(STATUS\_KM);

//Ej. Salida:

//MOT\_STATUS\_DATA\_0=0x08

//MOT\_STATUS\_DATA\_1=0x01;

byte 1	byte 0
0x01	0x08
0000 0001	0000 0100
264	
Distancia: 264*1.7mm	
448.8mm	

## 8. Ejercicios Prácticos

Los sensores instalados y los módulos expansibles nos dan flexibilidad a la hora de realizar aplicaciones. Se propondrán ejercicios que se pueden llevar a cabo fácilmente con Moway.

### *Seguimiento de línea*

Moway tiene que seguir una línea marcada en el suelo. Para ello utiliza los sensores de línea para saber la radiación reflejada del suelo, y según estos datos, toma decisiones.

### *Seguimiento luz*

En una habitación a oscuras, el robot debe seguir la luz emitida por una fuente lumínica utilizando para ello el sensor de luz.

### *Laberinto*

Moway debe salir de un laberinto utilizando para ello los sensores de proximidad.

### *Sumo*

Moway debe sacar a su contrincante de un ring.

### *Slalom*

Moway debe esquivar los objetos que se le pongan por delante.

### *Control remoto*

Moway es controlado mediante un PC utilizando para ello el módulo RF.