Ingeniería de Microsistemas Programados S.L
Alda. Mazarredo Nº 47 –1º Dpto 2
48009 Bilbao – Vizcaya (Spain)
Tfno/Fax: 94 4230651

MSE Microsystems Engineering
www.microcontroladores.com
info@microcontroladores.com

MD25: Driver doble puente H para motores

1.- DESCRIPCION

Este driver, desarrollado por la firma Devantech, Ltd., se muestra en la figura 1 y está diseñado para controlar dos motores DC modelo EMG30. Sus principales características son:

- Dispone de entradas para codificadores de cuadratura desde los motores.
- Lectura de esos codificadores para proporcionar un contador que determina el desplazamiento producido en los motores y el sentido de giro o dirección de los mismos.
- Actúa sobre dos motores que pueden ser controlados de forma autónoma o combinada (diferencial).
- Permite conocer el consumo de cada motor.
- Se alimenta con una tensión única de +12VDC.
- Dispone de un regulador de +5Vcc a 300mA para alimentar circuitería externa auxiliar.
- Capaz de suministrar una corriente de hasta 3A por cada motor.
- Interfaz serie o I2C con posibilidad de conectar hasta 8 drivers MD25 en el mismo bus.
- Control de potencia y aceleración.
- Reducidas dimensiones de 110 x 52 x 25 mm

Figura 1. *El Driver MD25*

Figura 2. *El kit de tracción RD02*

Con la referencia RD02 se puede adquirir un sistema completo de tracción para cualquier sistema móvil como puede ser un robot. Ver la figura 2.

El kit consta del driver MD25 objeto del presente documento, dos motores EMG30 de 12VDC con codificadores de cuadratura, dos piezas de soporte de los motores sobre la estructura móvil y 2 ruedas de 100mm de diámetro con sistema de fijación para ejes de 5 mm.

Estas ruedas se fijan directamente sobre los ejes de los motores EMG30 apretando simplemente unos tornillos.

Es una solución completa, flexible y potente para todo tipo de aplicaciones en las que se requiera un control total de desplazamientos.

2.- CONEXIONES Y GENERALIDADES

Se muestran en la figura 3. Existen dos conectores que permiten la conexión directa con los motores EMG30. Los dos conectores son idénticos y disponen de 6 señales:

- 2 salidas que se conectan a los bornes del motor.
- +Vcc para alimentación de los sensores Hall de los codificadores.
- GND de alimentación de los sensores Hall.
- Señal del codificador A
- Señal del codificador B

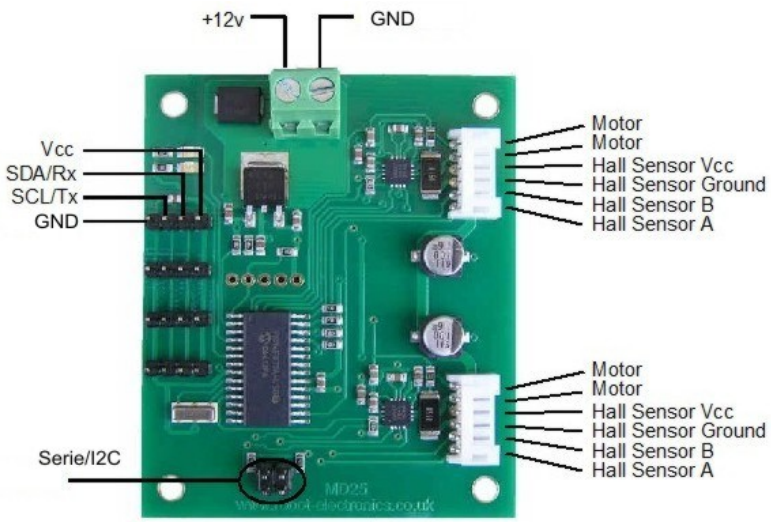


Figura 3. *Conexiones del driver MD25*

El driver dispone de 4 conectores idénticos de 4 vías para el interfaz I2C:

- +5Vcc para alimentar a los dispositivos I2C. Carga máxima admitida de 300mA.
- SDA/Rx línea de datos del bus I2C o recepción de datos en el modo serie.
- SCL/Tx línea de reloj del bus I2C o transmisión de datos en el modo serie.
- GND

¡ Importante ! En el modo I2C es imprescindible que las señales SDA y SCL del bus dispongan de sendas resistencias pull-up de unos 2K2 conectadas a +Vcc

La tarjeta se alimenta con una única tensión de +12VDC que se aplica por las bornas correspondientes. También dispone de dos jumpers para la selección del modo de comunicación con el controlador entre I2C o serie:

JP2 (izda.)	JP1 (dcha.)	Modo
Abierto	Abierto	Modo I2C
Abierto	Cerrado	Modo serie a 9600 baudios
Cerrado	Abierto	Modo serie a 19200 baudios
Cerrado	Cerrado	Modo serie a 38400 baudios

Supresión de ruido de los motores.

Los ruidos que pueden generar los motores eléctricos se deben suprimir añadiendo un condensador de unos 10nF en los bornes del motor. El condensador debe ser capaz de soportar la tensión total aplicada. En el caso de emplear los motores EMG30 dichos condensadores ya están incluidos.

Leds

El led rojo indica que el Driver MD25 está debidamente alimentado. El led verde indica actividad entre el bus I2C/Serie y el módulo. Este led también realiza una serie de intermitencias cada vez que se reasigna una nueva dirección I2C. Por defecto se emplea la dirección 0xB0 y, cada vez que se conecta el módulo MD25, se enciende durante 500mS.

Alimentación de los motores

El MD25 está diseñado para trabajar con una alimentación de 12VDC (p.e. batería). En general puede servir cualquier alimentación de 9 a 14VDC. En *Ingeniería de Microsistemas Programados* hemos probado con éxito la alimentación mediante una batería LIPO de 7.4VDC

Regulación automática de velocidad

El Driver MD25 dispone de un sistema de regulación automática por realimentación. Efectivamente, los codificadores de los motores EMG30 proporcionan información sobre el desplazamiento, giro y velocidad real. Si esta no coincide con la velocidad seleccionada, de forma automática se actúa corrigiendo y aplicando más o menos potencia a los motores hasta conseguir la velocidad deseada.

Esta característica por defecto está activada y puede ser habilitada o no por el usuario mediante el comando apropiado tanto en el modo I2C como en el modo serie.

Timeout automático para los motores

El MD25 puede desconectar automáticamente los motores si, transcurrido 2 segundos, no se ha recibido ninguna transferencia o comando I2C.

La práctica más habitual consiste en que el microcontrolador principal que gobierna el MD25 esté en constante comunicación con éste a través del protocolo I2C, enviando comandos para leer el valor de los codificadores, el consumo de los motores, ajustando aceleración, velocidad, etc.. Si esto no ocurre durante un tiempo de 2 segundos se puede entender que se ha perdido la comunicación, lo que origina un "Timeout" y, por seguridad, la parada de los motores.

Esta característica está activada por defecto, pero el usuario la puede desactivar mediante el comando correspondiente.

Ejemplos

Junto con el driver MD25 se adjunta un CDROM con una serie de librerías y ejemplos didácticos desarrollados por Ingeniería de Microsistemas Programados S.L., que facilitan el empleo del driver. Estos ejemplos están escritos tanto en ensamblador como en lenguaje C para el PIC16F886 y desarrollados y probados sobre el laboratorio USB-PIC'SCHOOL.

3.- CONTROL DEL MD25 EN EL MODO I2C

El módulo driver MD25 está diseñado para trabajar según el estándar I2C con una dirección que se puede seleccionar fácilmente, mediante el jumper o los comandos apropiados, entre la 0xB0 y la 0xBE (por defecto la 0xB0). Las conexiones y el protocolo I2C se puede implementar con cualquiera de los controladores más populares como PIC, BS2p, PICAXE, OIPC, AVR, Atmel, Arduino, etc..

La idea básica consiste en poder acceder a cualquiera de los registros internos del MD25 para transferirle los diferentes comandos y modos de trabajo así como leer sus variables de trabajo y su estado interno. El MD25 dispone de 17 registros internos que se resumen en la siguiente tabla:

Reg.Nº	Nombre	Tipo	Descripción
0	Velocidad 1	Lectura/Escritura	Ajusta la velocidad del motor 1 en los modos 0 y 1 o la velocidad de ambos en los modos 2 y 3
1	Velocidad 2 / Giro	Lectura/Escritura	Ajusta la velocidad del motor 2 en los modos 0 y 1 o el giro en los modos 2 y 3
2	Codificador 1 a	Sólo lectura	Posición del codificador 1. 1er. byte (mas peso)
3	Codificador 1 b	Sólo lectura	Posición del codificador 1, 2º byte
4	Codificador 1 c	Sólo lectura	Posición del codificador 1, 3er. Byte
5	Codificador 1 d	Sólo lectura	Posición del codificador 1, 4º byte (menos peso)
6	Codificador 2 a	Sólo lectura	Posición del codificador 2. 1er. byte (mas peso)
7	Codificador 2 b	Sólo lectura	Posición del codificador 2, 2º byte
8	Codificador 2 c	Sólo lectura	Posición del codificador 2, 3er. Byte
9	Codificador 2 d	Sólo lectura	Posición del codificador 2, 4º byte (menos peso)
10	Batería	Sólo lectura	Refleja la tensión actual de la batería de alimentación
11	Consumo Motor 1	Sólo lectura	Refleja el consumo actual del motor 1
12	Consumo Motor 2	Sólo lectura	Refleja el consumo actual del motor 2
13	Versión	Sólo lectura	Indica la versión del firmware interno
14	Aceleración	Lectura/Escritura	Registro opcional para la aceleración
15	Modo	Lectura/Escritura	Establece los modos de trabajo del MD25
16	Comando	Lectura/Escritura	Se introduce el comando a ejecutar por el MD25

3.1 El registro de Velocidad 1

Dependiendo del modo de trabajo seleccionado, este registro permite establecer la velocidad de uno o de ambos motores. Si se selecciona el modo 0 o 1, se ajusta la velocidad y sentido de giro del motor 1. Cuanto mayor sea el valor introducido en este registro, mayor será la potencia aplicada al motor.

Si se selecciona el modo 2 o 3 de trabajo, este registro determina la velocidad y dirección de ambos motores, pero esta sujeta al contenido del registro de giro.

3.2 El registro de Velocidad 2 / Giro

En los modos de trabajo 0 y 1 el contenido de este registro determina la velocidad y sentido de giro del motor 2. Cuando se seleccionan los modos 2 o 3 este registro se emplea como registro de giro. Su contenido se combina con el contenido del registro Velocidad 1 para controlar el movimiento de ambos motores de forma diferencial:

Si el movimiento es de avance:

Velocidad del motor 1 = Registro de Velocidad 1 – Registro de giro
Velocidad del motor 2 = Registro de Velocidad 1 + Registro de giro

Si el movimiento es de retroceso:

Velocidad del motor 1 = Registro de Velocidad 1 + Registro de giro
Velocidad del motor 2 = Registro de Velocidad 1 – Registro de giro

3.3 Registros codificadores

El módulo driver MD25 dispone de entradas para detectar las señales de hasta dos codificadores de cuadratura como los que disponen los motores EMG30. El resultado es un contador de 4 bytes por cada motor. Se obtiene así un número de 32 bits con signo que refleja el desplazamiento producido, el sentido de giro y la velocidad. Los registros 2, 3, 4 y 5 reflejan el contador del codificador 1 y los 6, 7, 8 y 9 el contador del codificador 2. Cada vez que se lee el byte de más peso de cualquiera de esos contadores (registros 2 y 6) se capturan y almacenan los siguientes bytes con el valor actual de cada codificador.

Mediante el comando apropiado estos contadores se puede poner a 0 para iniciar una nueva cuenta.

3.3 Batería

La lectura de este registro proporciona información del estado de la batería. La tensión leída representa décimas de voltio, por lo que una lectura de 121 corresponde a una tensión de 12,1V.

3.4 Consumo

Hay dos registros que permiten conocer el consumo de cada motor. El valor leído en cualquiera de estos registros representa décimas de amperio, por lo que una lectura de 25 corresponde a un consumo de 2,5A.

3.5 Versión

La lectura de este registro permite conocer la versión del firmware interno que controla el módulo driver MD25.

3.6 Aceleración

Este registro permite establecer el periodo de aceleración o tiempo que tardan los motores en pasar de una velocidad a otra.

La regulación de velocidad se realiza mediante una serie de pasos o intervalos de 25mS cada uno. Cuando pasamos de una velocidad a otra, se puede determinar cuantos pasos se han de dar. Conociendo que estos se producen cada 25mS, se puede calcular el tiempo que tardan los motores en alcanzar la nueva velocidad. El contenido de este registro permite dividir el nº de pasos entre un factor de 1 a 10, con lo que es consigue variar ese tiempo.

Si por ejemplo un motor se encuentra girando a la máxima velocidad en un sentido (255) y se desea pasar a la máxima pero en sentido opuesto (0), el número de pasos es de 255 si el registro de aceleración vale 1. Ahora bien si el registro valiera 2, el número de pasos se reduce a 128 con lo que se aumenta la aceleración. Todo esto se puede resumir con la siguiente ecuación:

- Si la nueva velocidad es mayor que la velocidad en curso:

$$Pasos = \frac{(NuevaVelocidad - VelocidadActual)}{\mathrm{Re}\,g.Aceleración}$$

- Si la nueva velocidad es menos que la velocidad en curso:

$$Pasos = \frac{(VelocidadActual - NuevaVelocidad)}{\mathrm{Re}\,g.Aceleración}$$

Unos ejemplos nos permitirán apreciar la misión de este registro:

Reg. Aceleración	Intervalo/ paso	Velocidad Actual	Nueva Velocidad	Pasos	Tiempo de aceleración
1	25mS	0	255	255	6,375 s
2	25mS	127	255	64	1,600 s
3	25mS	80	0	27	0,675 s
5 (por defecto)	25mS	0	255	255	1,275 s
10	25mS	255	0	26	0,65 s

3.7 Registro de modo

Mediante este registro se puede seleccionar uno de los diferentes modos de trabajo del MD25. Los valores y modos admitidos son:

Modo 0 (0x00): Es el modo por defecto. Cuando se escribe el valor 0x00 en el registro modo, la velocidad para ambos motores se establece literalmente en los registros de Velocidad 1 y Velocidad 2 para cada uno de los dos motores. El rango para estos registros es de 0 (máxima en retroceso), 128 (stop) y 255 (máxima en avance).

Modo 1 (0x01): Es similar al modo 0 excepto que en los registros de Velocidad 1 y 2 se deben cargar valores con signo para el ajuste de velocidad: -128 (máxima en retroceso), 0 (stop) y 127 (máxima en avance)

Modo 2 (0x02): En este modo los motores actúan de forma combinada o diferencial. Con el registro Velocidad 1 se establece la velocidad para ambos motores. El contenido del registro Velocidad 2/Giro se combina con el anterior para controlar el movimiento de ambos motores de forma diferencial. El rango de Velocidad 1 es de 0 (máxima en retroceso), 128 (stop) y 255 (máxima en avance).

Modo 3 (0x03): Es similar al modo 2 excepto que el valor de la velocidad se expresa con signo: -128 (máxima en retroceso), 0 (stop) y 127 (máxima en avance).

3.8 Registro de comandos

Escribiendo ciertos valores o comandos sobre este registro se consigue que el driver MD25 ejecute ciertas tareas que se resumen en la siguiente tabla:

Comando		Descripción
Dec.	**Hex.**	
32	**20**	Resetea y pone a 0 los registros contadores de los codificadores
48	**30**	Desconecta la característica de regulación automática de velocidad
49	**31**	Activa la característica de regulación automática de velocidad (por defecto)
50	**32**	Desconecta la característica de detener los motores automáticamente si, durante 2 segundos, no se realiza ninguna transferencia I2C
51	**33**	Activa la característica de detener los motores automáticamente si, durante 2 segundos, no se realiza ninguna transferencia I2C (por defecto)
160	**A0**	1º Byte de la secuencia para cambiar la dirección I2C del driver MD25
170	**AA**	2º Byte de la secuencia para cambiar la dirección I2C del driver MD25
165	**A5**	3º Byte de la secuencia para cambiar la dirección I2C del driver MD25

3.8.1 Cambiando la dirección I2C

La dirección del MD25 se puede cambiar escribiendo una nueva. Para escribir una nueva dirección es imprescindible que sólo haya un módulo MD25 conectado al bus. A continuación se envía una secuencia de tres comandos en el orden apropiado, seguido de la nueva dirección. A partir de ese momento queda grabada en el módulo y será utilizada por defecto. Por ejemplo, si la dirección actual es la 0xB0 y se desea cambiar por

la 0xB4, basta con escribir en el registro de comandos la secuencia siguiente: 0xA0, 0xAA, 0xA5, 0xB4. Es imprescindible que estos 4 bytes se escriban sobre el registro de comandos de forma secuencial e ininterrumpida. Entre el ciclo de escritura de un byte y el siguiente debe dejarse un tiempo mínimo de unos 5mS para que el MD25 realice sus operaciones internas.

En el instante de arranque del módulo MD25, cada vez que se conecta la alimentación, el led verde realiza una secuencia de intermitencias que reflejan la dirección I2C actual. Mientras se está realizando dicha secuencia es recomienda no realizar ninguna transferencia. Esta se resume en la siguiente tabla adjunta.

Dirección		Encendido Largo	Encendidos Cortos
DEC.	HEX.		
176	B0	1	0
178	B2	1	1
180	B4	1	2
182	B6	1	3
184	B8	1	4
186	BA	1	5
188	BC	1	6
190	BE	1	7

4. CONTROL DEL MD25 EN EL MODO SERIE

Otra forma de interface del driver MD25 con nuestro controlador Host, es mediante la comunicación serie con niveles TTL tipo USART, como la que integran la mayor parte de controladores. Mediante los jumpers se selecciona la velocidad de transferencia entre 9600, 19200 y 38400 baudios y siempre se realiza con 8 bits de datos, 1 de stop y sin paridad.

Toda comunicación la inicia siempre nuestro controlador host. Empieza siempre con un byte de sincronismo (0x00) a continuación se envía el byte del comando a ejecutar y finalmente el byte de parámetro que necesitan algunos comandos.

Una vez enviado un comando, el controlador Host se debe quedar a la espera de recibir la respuesta por parte del driver MD25. Esta puede constar desde 0 bytes (sin respuesta) hasta 8 bytes de respuesta. La siguiente tabla resume todos los comandos disponibles en el modo serie. La mayor parte de ellos son equivalentes a los empleados y vistos en el modo I2C de funcionamiento.

Comando	Nombre	Nº de Bytes enviados	Nº de bytes devueltos	Descripción
0x21	GET SPEED 1	2	1	Devuelve la velocidad actual del motor 1
0x22	GET SPEED 2	2	1	Devuelve la velocidad actual del motor 1
0x23	GET ENCODER 1	2	4	Devuelve 4 bytes con signo (1º el de más peso) que expresan el contaje del encoder 1.
0x24	GET ENCODER 2	2	4	Devuelve 4 bytes con signo (1º el de más peso) que expresan el contaje del encoder 2.
0x25	GET ENCODERS	2	8	Devuelve 8 bytes consecutivos. Los 4 primeros corresponden con el encoder 1 y los otros 4 con el encoder 2
0x26	GET VOLTS	2	1	Devuelve, en décimas de voltio, el valor de la tensión de alimentación del driver MD25
0x27	GET CURRENT 1	2	1	Devuelve la intensidad que consume el motor 1
0x28	GET CURRENT 2	2	1	Devuelve la intensidad que consume el motor 2
0x29	GET VERSION	2	1	Devuelve un byte con la versión del firmware interno del MD25
0x2A	GET ACELERACION	2	1	Devuelve el valor actual de la aceleración

0x2B	GET MODE	2	1	Devuelve el modo actual de trabajo (1, 2, 3 o 4)
0x2C	GET VI	2	3	Devuelve 3 bytes que representan la tensión de alimentación, el consumo del motor 1 y el consumo del motor 2 respectivamente
0x31	SET SPEED1	3	0	El tercer byte enviado permite ajustar la nueva velocidad del motor 1
0x32	SET SPEED 2 /TURN	3	0	El tercer byte enviado permite ajustar la nueva velocidad del motor 2 o bien el giro de ambos
0x33	SET ACCELERATION	3	0	El tercer byte expresa entre 1 y 10 el nuevo valor de aceleración
0x34	SET MODE	3	0	El tercer byte expresa el nuevo modo de trabajo (1, 2, 3 o 4)
0x35	RESET ENCODERS	2	0	Pone a cero el contador de ambos encoders
0x36	DISABLE REGULATOR	2	0	Desconecta el sistema de regulación automática de velocidad
0x37	ENABLE REGULATOR	2	0	Conecta el sistema de regulación automática de velocidad (por defecto)
0x38	DISABLE TIMEOUT	2	0	Desconecta el sistema de parada por TimerOut. Los motores siguen activados aunque no haya comunicación con el MD25
0x39	ENABLE TIMEROUT	2	0	Activa el sistema de parada por TimerOut. Los motores se paran al de 2 seg, si se pierde la comunicación (por defecto)

5. EL MOTOR EMG30

Independientemente de que otros motores puedan ser compatibles con el driver MD25, éste está diseñado expresamente para el motor EMG30 mostrado en la figura 4. Se trata de un motor de 12VDC totalmente equipado con codificadores de cuadratura, caja reductora de 30:1 y condensador supresor de ruido. Es ideal para aplicaciones de robótica con un coste asequible y la posibilidad de un control total por parte del usuario.

Viene provisto de un conector hembra paso 2.54 con 6 cables que transportan las señales que lo controlan.

Figura 4. *El motor EMG30*

En la siguiente figura 5 se muestra las dimensiones mecánicas del mismo y la distribución de señales en el conector.

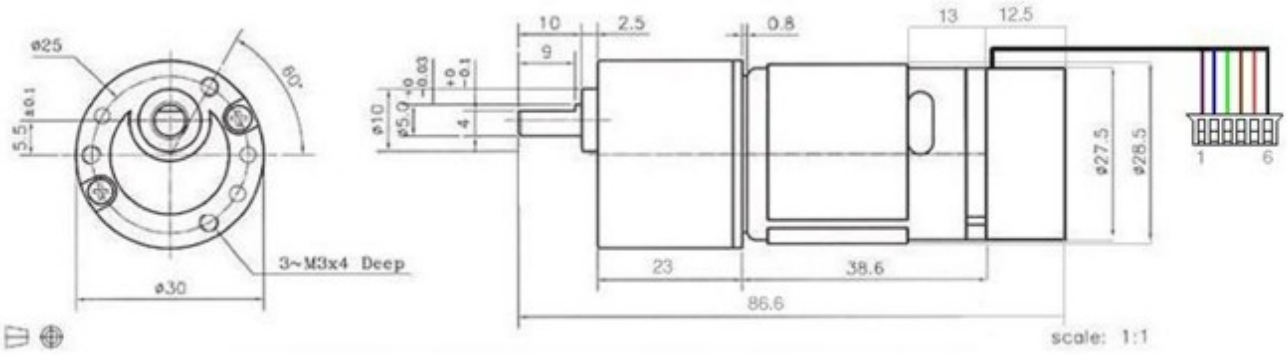


Figura 5. *Dimensiones mecánicas del motor MG30*

Cable	Descripción
Púrpura (1)	Señal de salida del sensor Hall del codificador B
Azul (2)	Señal de salida del sensor Hall del codificador A
Verde (3)	Alimentación GND para los sensores Hall
Marrón (4)	Alimentación +Vcc para los sensores Hall
Rojo (5)	+VDC para alimentación de los motores
Negro (6)	GND para alimentación de los motores

Las salidas de los sensores Hall necesitan unas resistencias pull-up de unos 4K7Ω, que están incluidas si se emplea el driver MD25.

5.1 Especificaciones del MG30

- Tensión nominal 12VDC
- Fuerza 1.5Kg / cm
- Velocidad nominal 170 rpm
- Consumo nominal 530 mA
- Velocidad sin carga 216 rpm
- Consumo sin carga 150mA
- Codificador de 90 pulsos por cada vuelta del eje de salida (4º de giro por cada pulso)

5.2 Soporte de montaje

Permite fijar el motor a cualquier tipo de estructura. Está fabricada en aluminio de 2 mm esmaltado en azul. Las siguientes figuras 6 y 7 muestran el anclaje con el motor y las dimensiones mecánicas.

Figura 6. *Anclaje del motor EMG30*

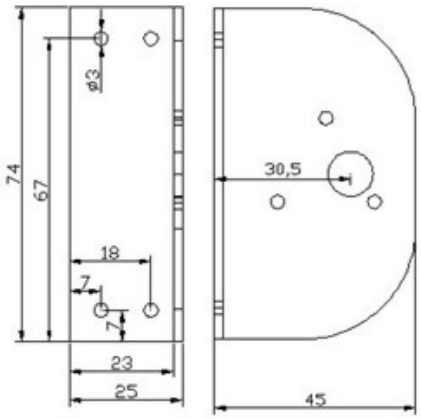


Figura 7. *Dimensiones del anclaje*

6. EJEMPLOS

Para realizar los siguientes ejemplos, en Ingeniería de Microsistemas Programados hemos empleado el laboratorio USB-PIC'SCHOOL como plataforma para la experimentación, depuración y puesta a punto de los mismos. También hemos usado el kit de tracción (ref.RD02). Sobre una superficie de madera hemos mecanizado y sujetado los soportes de los motores, los motores y la tarjeta driver MD25 tal y como se muestra en la figura 8.

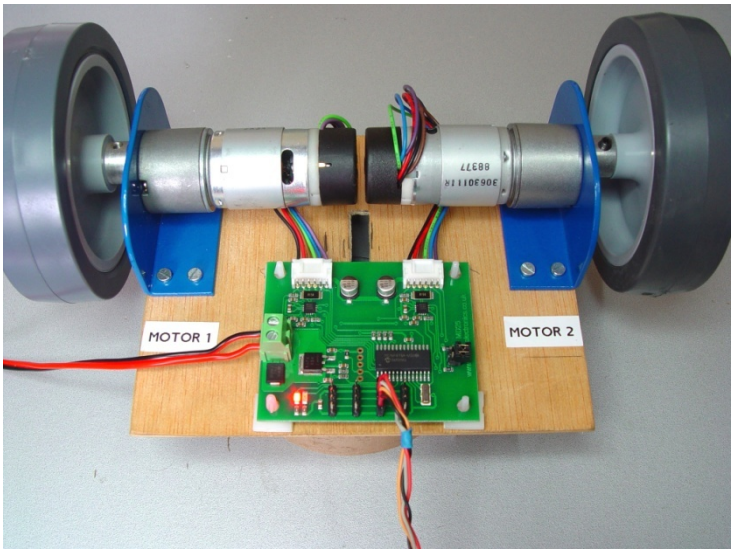


Figura 8. *Fijación de los motores y tarjeta*

Junto con el driver MD25se adjunta un CDROM que contiene librerías y ejemplos sencillos realizados por *Ingeniería de Microsistemas Programados S.L* , que muestran el funcionamiento de este dispositivo. Se proporcionan los programas fuente escritos en ensamblador y en C para el PIC16F886. Se trata de ejemplos didácticos cuya única pretensión es proporcionar los elementos básicos para el manejo este controlador de motores de cara a su empleo en las múltiples aplicaciones en que puede ser utilizado. Se incluyen también una serie de librerías que facilitan el diseño de los programas:

Librería: *lcd4bitsPIC16.inc*

Contiene rutinas para el control de una pantalla LCD mediante interface de 4 bits y que se usará para visualizar los resultados de los distintos ejemplos.

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
UP_LCD	Ninguno	Ninguno	Configura las líneas de E/S para adaptarlas a la pantalla LCD
LCD_INI	Ninguno	Ninguno	Rutina para la inicialización de la pantalla LCD según especificaciones del fabricante
LCD_DATO	*W*=Contiene el carácter a visualizar	Ninguno	Envía a la pantalla el dato a visualizar en la posición actual del cursor
LCD_REG	*W*=Contiene el comando a ejecutar por parte de la pantalla LCD	Ninguno	Envía a la pantalla el comando que debe ejecutar

Librería: *MSE_Mat_PIC16.inc*

Esta librería contiene una serie de funciones que resuelven las operaciones matemáticas más elementales y que son empleadas por algunos de los ejemplos escritos en ensamblador. Los ejemplos escritos en C no necesitan de esta librería ya que la mayor parte de las funciones matemáticas están integradas en el propio compilador.

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
Sum_BCD	Mat_Dato_AL+Mat_Dato_BL	Mat_Dato_BL Mat_Dato_AH	Suma dos números BCD (00-99): Mat_Dato_AL + Mat_Dato_BL. El resultado se almacena en Mat_Dato_BL. En Mat_Dato_AH queda, si lo hubiera, el overflow (resultado mayor de 99)
Sub_BCD	Mat_Dato_AL- Mat_Dato_BL		Resta dos números BCD (00-99): Mat_Dato_AL - Mat_Dato_BL. El resultado se almacena en Mat_Dato_AL. En Mat_Dato_BH queda, si lo hubiera, el overflow (A<B).
Sum16	Mat_Dato_BH:Mat_Dato_BL+ Mat_Dato_AH:Mat_Dato_AL	Mat_Dato_BH Mat_Dato_BL C	Suma dos números de 16 bits. Mat_Dato_BH:Mat_Dato_BL+ Mat_Dato_AH:Mat_Dato_AL El resultado se deposita en Mat_Dato_BH:Mat_Dato_BL. Si hay llevada en el bit 16º, C=1
Sub16	Mat_Dato_AH:Mat_Dato_AL- Mat_Dato_BH:Mat_Dato_BL.	Mat_Dato_BH Mat_Dato_BL C	Resta dos números de 16 bits. Mat_Dato_AH:Mat_Dato_AL- Mat_Dato_BH:Mat_Dato_BL. El resultado se almacena en Mat_Dato_BH:Mat_Dato_BL. Si hay llevada en el bit 16º, C=0
Comp16	Mat_Dato_AH:Mat_Dato_AL(A) Mat_Dato_BH:Mat_Dato_BL(B)	C, Z	Compara dos nº de 16 bits contenidos en Mat_Dato_AH:Mat_Dato_AL(A) con Mat_Dato_BH:Mat_Dato_BL(B) Si A>B --> STATUS<C>=1 y <Z>=0; Si A<B --> STATUS<C>=0 y <Z>=0; Si A=B --> STATUS<C>=1 y <Z>=1
Mul8x8	Mat_Dato_AL * Mat_Dato_BL.	Mat_RES2:Mat_RES3	Multiplica dos nºs de 8 bits contenidos en Mat_Dato_AL y Mat_Dato_BL. El resultado de 16 bits se almacena en Mat_RES2 y Mat_RES3 (LSB).
Mul16x16	Mat_Dato_BH:Mat_Dato_BL* Mat_Dato_AH:Mat_Dato_AL	Mat_RES0:Mat_RES3	Multiplica dos números de 16 bits: Mat_Dato_BH:Mat_Dato_BL* Mat_Dato_AH:Mat_Dato_AL El resultado de 32 bits se almacena en Mat_RES0:Mat_RES3 (LSB)
Div16x16	Mat_Dato_AH:Mat_Dato_AL/ Mat_Dato_BH:Mat_Dato_BL	Mar_Dato_AH:Mat_Dato_ AL Mat_RES3:Mat_RES2	Esta rutina divide dos números de 16 bits. El dividendo se almacena en Mat_Dato_AH:Mat_Dato_AL y el divisor en Mat_Dato_BH:Mat_Dato_BL. El cociente se almacena en Mar_Dato_AH:Mat_Dato_AL y el resto en Mat_RES3:Mat_RES2.
Bits8_BCD	W	Mat_RES0	Convierte un número binario de 8 bits en el registro W, en 2 dígitos BCD (de 00 a 99). El resultado se almacena en Mat_RES0. P.e. W=0x2A, Mat_RES0=42 (0x2A = 42).
Bits16_BCD	Mat_Dato_AH:Mat_Dato_AL	Mat_RES0 Mat_RES1 Mat_RES2 (LSB)	Esta rutina convierte un número binario de 16 bits situado en Mat_Dato_AH y Mat_Dato_AL y, lo convierte en 5 dígitos BCD que se depositan en las variables Mat_RES0, Mat_RES1 y Mat_RES2, siendo esta última la de menos peso.
BCD_Bits16	Mat_RES0:Mat_RES2 (LSB)	Mat_Dato_AH:Mat_Dato_ AL	Convierte un número de 5 dígitos en BCD, en un número de 16 bits. En Mat_RES0:Mat_RES2 (LSB) se encuentra en nº de 5 dígitos. El resultado se almacena en Mat_Dato_AH:Mat_Dato_AL.
BCD_ASCII	Mat_RES0:Mat_RES2 (LSB) Mat_Point	Mat_RES0(MSB):Mat_RE S6(LSB).	Convierte un número BCD de hasta 6 dígitos almacenados en Mat_RES0:Mat_RES2 (LSB) En una cadena ASCII que se almacena a partir de Mat_RES0(MSB) hasta Mat_RES6(LSB). La variable Mat_Point indica el lugar donde colocar un punto decimal (0 =ninguno). P.e. Mat_Point=1, el resultado sería .xxxxxx; Mat_Point=3, el resultado sería xx.xxxx

Librería: *I2C_16FXXX.inc*

Contiene rutinas para la gestión de transferencia de datos mediante el protocolo I2C.

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
I2C_INI			Activa e inicia el módulo MSSP de algunos dispositivos PIC16FXXX para trabajar en el modo I2C Master a 100KHz
I2C_Send_Start			Genera en el bus I2C la condición de inicio
I2C_Sentd_Stop			Genera en el bus I2C la condición de Stop
I2C_Send_Byte	W		Transmite el byte contenido en W por el bus I2C. La rutina finaliza cuando se recibe el /ACK generado por el slave.
I2C_Read_Byte	ACK	W	Lee un byte procedente del dispositivo I2C slave seleccionado y lo devuelve en el registro W. Seguidamente se genera y transmite el bit /ACK si procede (bit 0 de variable ACK=0) o el bit NACK (bit 0 de variable ACK=1)
Leer_I2C:	Dir_I2C I2C_Dir_Ini I2C_N_Bytes	I2C_Buffer	Lee un nº de bytes del dispositivo I2C. La variable Dir_I2C contiene la dirección I2C del dispositivo. La variable I2C_Dir_Ini contiene la dirección interna inicial. La variable I2C_N_Bytes contiene el nº de bytes a leer. Los bytes recibidos se depositan en RAM a partir de la posición indicada por I2C_Buffer
Escr_I2C	Dir_I2C Dir_Ini I2C_N_Bytes I2C_Buffer		Escribe sobre el dispositivo I2C un nº de bytes. La variable Dir_I2C contiene la dirección del dispositivo. La variable Dir_Ini contiene la dirección interna inicial. La variable I2C_N_Bytes contiene el nº de bytes a escribir. Se supone que los bytes a escribir están previamente depositados en un buffer de memoria cuyo inicio está representado por I2C_Buffer.

Librería: *MD25_16FXXX.inc*

Esta librería contiene una serie de funciones propias para el control del Driver MD25

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
MD25_Modo	W		El MD25 selecciona un determinado modo de trabajo cuyo código está presente en W: 0x00 modo 0, velocidad sin signo para cada motor (0,128,255). 0x01 modo 1, velocidad con signo para cada motor (-128,0,127). 0x02 Modo 2, velocidad diferencial sin signo para ambos motores (0,128,255). 0x03 Modo 3, Velocidad diferencia con signo para ambos motores (-128,0 127)
MD25_Firm		W	Lee la versión del firmware interno del driver MD25. El resultado se devuelve en el registro W
MD25_Bat		W	Mide la tensión actual de la batería de alimentación en décimas de voltio. El resultado hexadecimal se devuelve en W
MD25_Consumo		MD25_Byte_H MD25_Byte_L	Mide el consumo actual en ambos motores. MD25_Byte_H representa el del motor 1 y MD25_Byte_L el del motor 2. El consumo representado en hexadecimal expresa décimas de amperio.

MD25_Enc_Rst			Pone a 0 los registros de los codificadores
MD25_Enc1_Rd		MD25_Byte_SH (msb) MD25_Byte_SL MD25_Byte_H MD25_Byte_L (lsb)	Lee el valor actual del codificador 1. El resultado hexadecimal de 4 bytes se almacena en MD25_Byte_SH (msb), MD25_Byte_SL, MD25_Byte_H y MD25_Byte_L (lsb). La resolución es de 360 pulsos por cada vuelta del eje (360º)
MD25_Enc2_Rd		MD25_Byte_SH (msb) MD25_Byte_SL MD25_Byte_H MD25_Byte_L (lsb)	Lee el valor actual del codificador 2. El resultado hexadecimal de 4 bytes se almacena en MD25_Byte_SH (msb), MD25_Byte_SL, MD25_Byte_H y MD25_Byte_L (lsb). La resolución es de 360 pulsos por cada vuelta del eje (360º)
MD25_Auto_Stop	C=0 C=1		Esta función permite desconectar o no los motores si, transcurridos 2 seg., no se recibe ninguna transferencia I2C. Si el bit C=0 desactiva esta opción y si C=1 la activa (por defecto)
MD25_Auto_Vel	C=0 C=1		Esta función permite habilitar o no la característica de regulación automática de velocidad. Si el bit C=0 desactiva esta característica y si C=1 la activa (por defecto)
MD25_Acel	W		Permite ajustar la aceleración de los motores. El valor de la misma se indica en W y está comprendida entre 1 y 10 (por defecto 5). Determina el tiempo que los motores tardan en pasar de una velocidad a otra
MD25_M1	W		Activa el motor 1 en el modo 0 a una velocidad determinada por reg. W=0 --> M1 a la máxima en un sentido; W=128 --> M1 stop; W=255 --> M1 a la máxima en sentido contrario
MD25_M2	W		Activa el motor 2 en el modo 0 a una velocidad determinada por reg. W=0 --> M2 a la máxima en un sentido; W=128 --> M2 stop; W=255 --> M2 a la máxima en sentido contrario
MD25_Adelante	W		Combinando M1 y M2 en modo 0 se realiza un movimiento de avance. La velocidad se indica en W (entre 0 y 127)
MD25_Atras	W		Combinando M1 y M2 en modo 0 se realiza un movimiento de retroceso. La velocidad se indica en W (entre 0 y 127)
MD25_Rot_Dcha	W		Combinando M1 y M2 en modo 0 se realiza un movimiento de rotación a la derecha. La velocidad se indica en W (entre 0 y 127)
MD25_Rot_Izda	W		Combinando M1 y M2 en modo 0 se realiza un movimiento de rotación a la izquierda. La velocidad se indica en W (entre 0 y 127)
MD25_Dcha	W		Combinando M1 y M2 en modo 0 se realiza un movimiento de giro a la derecha. La velocidad se indica en W (entre 0 y 127)
MD25_Izda	W		Combinando M1 y M2 en modo 0 se realiza un movimiento de giro a la izquierda. La velocidad se indica en W (entre 0 y 127)
MD25_Stop			Los motores M1 y M2 quedan detenidos
MD25_I2C_Dir	W		Cambia la dirección I2C del MD25. W debe contener la nueva dirección I2C
MD25_Dif2	MD25_Byte_H MD25_Byte_L		Realiza, en el modo 2, un movimiento diferencial en ambos motores. En MD25_Byte_H se establece la velocidad y dirección de ambos motores y en MD25_Byte_L el valor de giro o diferencial
MD25_Dif3	MD25_Byte_H MD25_Byte_L		Realiza, en el modo 3, un movimiento diferencial en ambos motores. En MD25_Byte_H se establece la velocidad y dirección de ambos motores y en MD25_Byte_L el valor de giro o diferencial

6.1 Ejemplo 1: Lectura de parámetros del driver MD25

Objetivos

Tener una primera toma de contacto con el driver MD25 y las funciones de las librerías que lo controlan. En este caso se trata de leer y visualizar sobre la pantalla LCD la versión del firmware interno del driver así como la tensión con la que se le está alimentado.

Esquema

El esquema empleado se muestra en la figura 9. La pantalla LCD del laboratorio USB-PIC'School se gestiona mediante un interface de 4 bits de datos con las líneas RB0:RB3 del controlador y mediante las señales de control E, R/W y RS que se conectan con RA1:RA3 respectivamente.

Las líneas SDA y SCL del driver MD25 se conectan con las correspondientes RC4/SDA y RC3/SCL del controlador y también a las resistencias pull-up de 2K2.

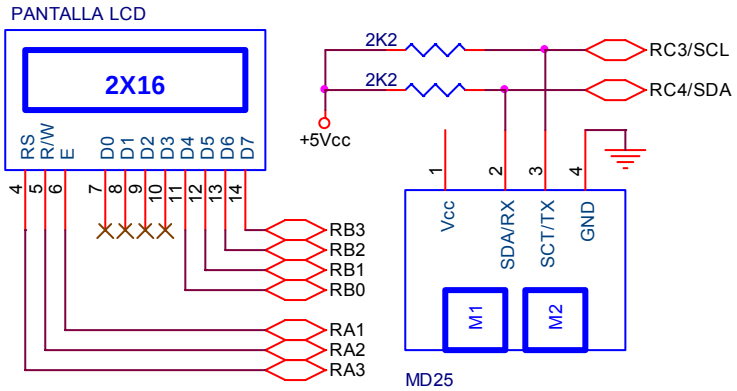
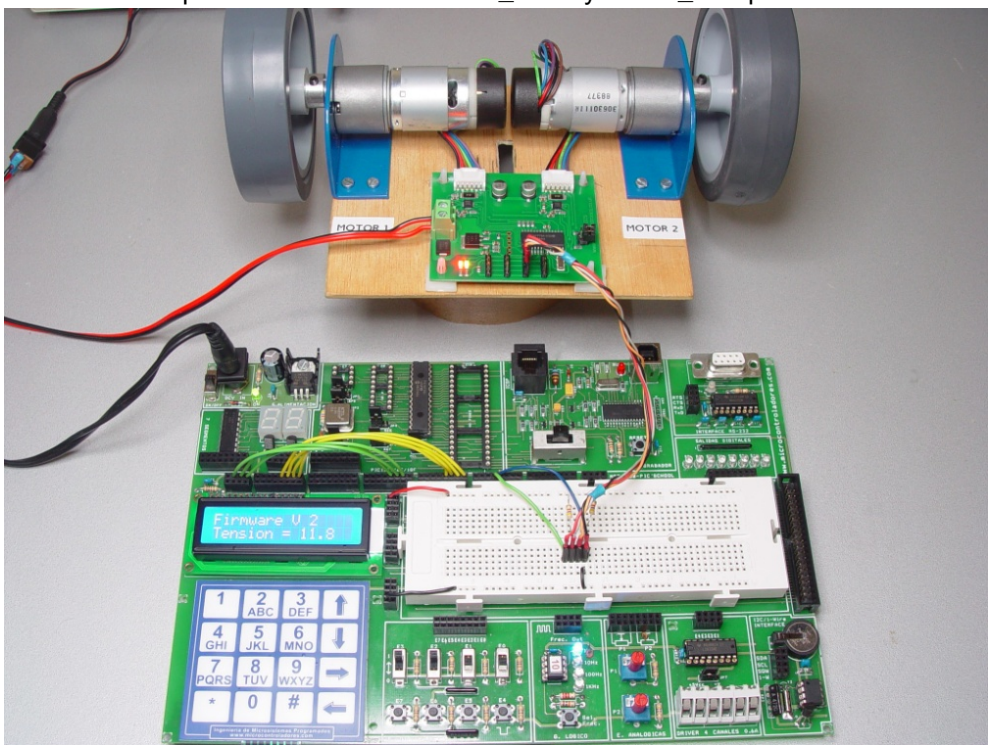


Figura 9. *Esquema de montaje del ejemplo 1*

Comentarios

Se emplean las funciones ***MD25_Firm*** y ***MD25_Bat*** para obtener la versión del firmware interno del driver MD25 y la tensión de alimentación. Esta tensión se expresa en décimas de voltio y se corresponde con la tensión actual que alimenta al driver MD25 proveniente de alimentadores, pilas y o baterías.



En la figura 10 se muestra la conexión del laboratorio USB-PIC'SCHOOL con la plataforma móvil que hemos construido para alojar a los componenetes del kit de tracción RD02 (motores, sus soportes y el propio Driver MD25). También se aprecia el resultado de la ejecución de éste ejemplo.

Figura 10. *Ejecución del ejemplo 1*

6.2 Ejemplo 2: Lectura de parámetros del driver MD25

Objetivos

Empleando algunas de las funciones de la librería MD25_16FXXX.INC, el ejemplo trata de leer el valor actual de los encoders asociados a cada uno de los dos motores que es capaz de controlar el driver MD25

Esquema

El mismo que para el ejemplo 1 anterior.

Comentarios

Inicialmente los registros que mantienen la cuenta de ambos encoders se ponen a 0. Moviendo manualmente los ejes de cualquiera de los motores se varía el valor del encoder correspondiente. Se puede comprobar que cada incremento supone un giro de 1º en el eje. Así, un giro completo de 360º se corresponde con una lectura del encoder de 360 pasos. Cada uno de los encoder proporciona un valor de 32 bits (4 bytes). En el ejemplo sólo se visualizan, sobre el LCD, los 16 bits (2 bytes) de menos peso, tal y como se muestra en la figura 11.

La importancia de disponer de encoders asociados a cada motor, radica en que nos permite realizar aplicaciones en las que es necesario conocer y/o controlar la posición actual del motor o el desplazamiento que éste produce. Sabemos que cada pulso o paso del encoder se corresponde con 1º de giro del eje del motor correspondiente, 360 pasos se corresponden por tanto con un giro completo de 360º del eje.

De la misma manera si conocemos el diámetro o radio de la rueda que fijamos al eje del motor, también conoceremos su circunferencia y por tanto cuanto se desplaza en función de los pasos que indica el encoder.

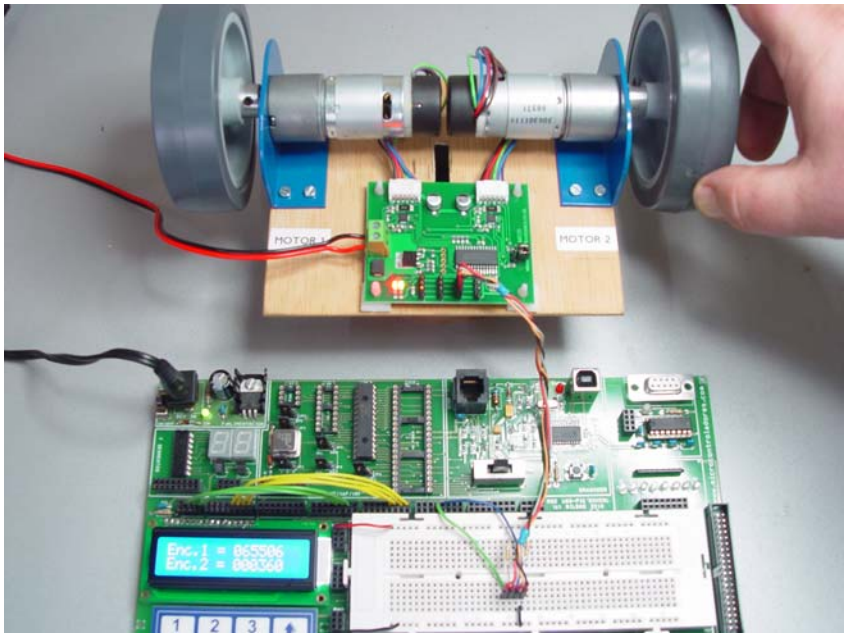


Figura 11. *Visualización del estado actual de los encoders*

6.3 Ejemplo 3: Activación de los motores y medida de consumo

Objetivos

En este ejemplo vamos a activar, por primera vez, ambos motores al tiempo que leemos y visualizamos sobre la pantalla LCD el consumo de los mismos.

Esquema

Se muestra en la figura 12 y es muy similar al empleado en los ejemplos anteriores. Se ha añadido el potenciómetro P1 del laboratorio USB-PIC'SCHOOL que, conectado a la entrada analógica RA0/AN0, genera una tensión variable entre 0 y 5V.

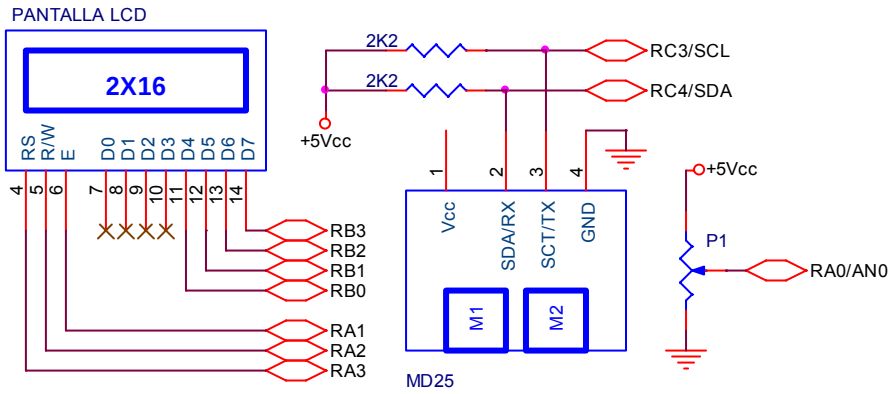


Figura 12. *Esquema de montaje del ejemplo 3*

Comentarios

El ejemplo emplea algunas funciones de la librería MD25_16FXXX para activar ambos motores en el modo 0 así como leer y visualizar en el LCD el valor de la corriente que consume cada uno de ellos. Mediante la entrada analógica RA0/AN0 conectada a un potenciómetro, se ajusta la velocidad de giro de ambos motores hacia adelante. El concepto de "adelante" o "atrás" es relativo a la colocación física de los motores.

Se puede observar que cuando los motores giran libremente apenas se produce un consumo. Podemos producir un aumento de éste simplemente frenando el giro de los ejes o ruedas (como se muestra en la figura 13). Esto equivale a la fuerza de tracción que tendrían que ejercer los motores con el consiguiente aumento del consumo (máximo 2.8 A)

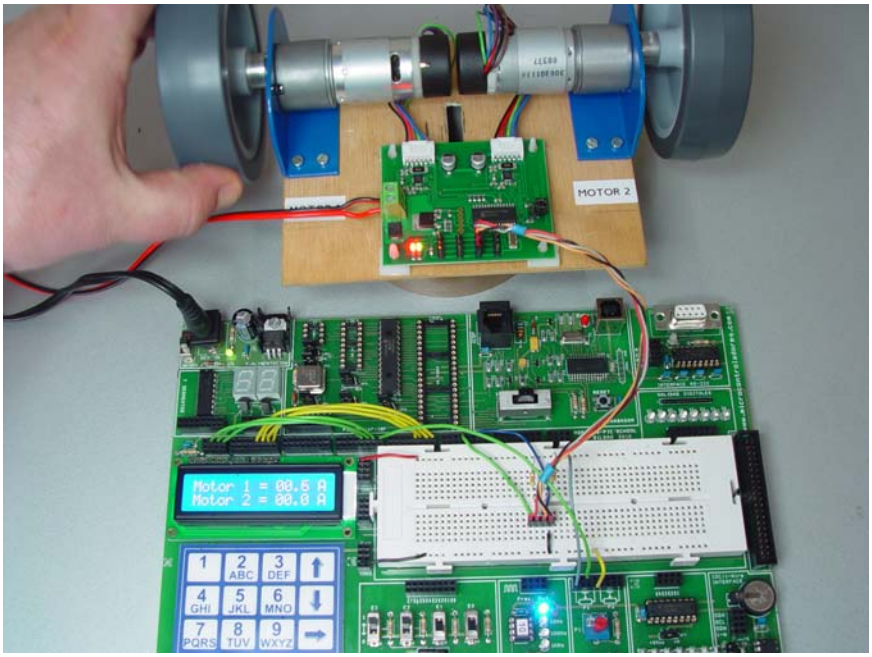


Figura 13. *Ejecución del ejemplo 3*

6.4 Ejemplo 4: La función de auto stop

Objetivos

Mostrar la función auto stop que tiene el driver MD25 para evitar el movimiento descontrolado de los motores.

Esquema

El esquema para realizar este ejemplo se muestra en la figura 14. Con el interruptor E0 conectado a la entrada RC0 se activa o no el modo auto stop. Mediante el pulsador E4 conectado con la entrada RC1 se inicia una sencilla secuencia de movimiento.

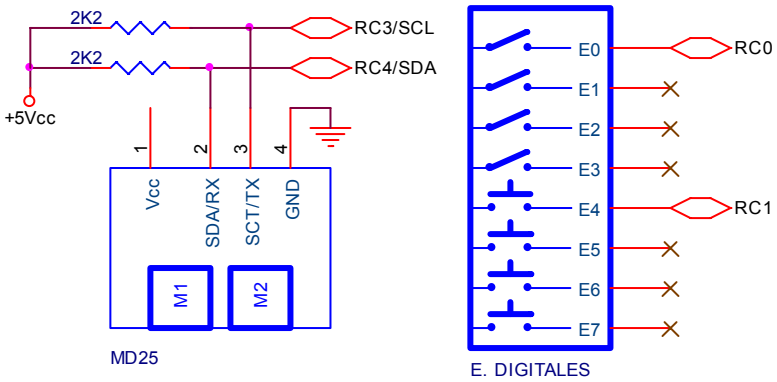


Figura 14. *Esquema del ejemplo 4*

Comentarios

Cuando esta función está activada (RC0=1) los motores paran automáticamente si, transcurridos unos 2 segundos, no se recibe ninguna transferencia I2C. Se evita así que, en el supuesto de que el controlador principal pierda la comunicación con el driver MD25, los motores sigan funcionando sin control.

En el ejemplo el ciclo comienza con un pulso 1-0-1 en la entrada RC1 conectada a un pulsador del laboratorio. Los motores inician un movimiento hacia adelante a la máxima velocidad. Una nuevo pulso 1-0-1 en RC1 hará que se paren.

Con la entrada RC0 = "1" se activa la función de auto stop (por defecto). Se puede apreciar que los motores paran automáticamente transcurridos 2 seg. Si RC0 = "0" el avance se mantiene indefinidamente hasta el 2º pulso en RC1.

El concepto de "adelante" o "atrás" es relativo a la colocación física de los motores.

6.5 Ejemplo 5: El efecto de la aceleración

Objetivos

Comprobar el efecto de la aceleración que produce el driver MD25 cuando los motores pasan de una velocidad a otra.

Esquema

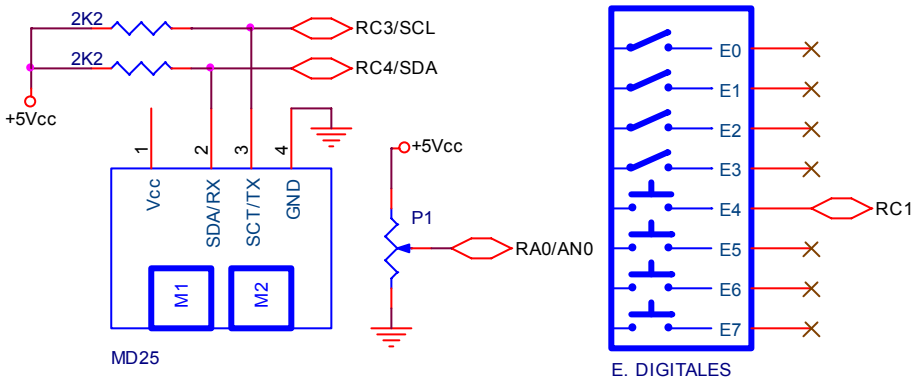


Figura 15. *Esquema del ejemplo 5*

Es el mostrado en la figura 15 y muy similar al empleado en el ejemplo anterior. En este caso sólo se emplea el pulsador E4 conectado a la entrada RC1 para iniciar el ciclo de movimiento, y el potenciómetro P1 del laboratorio para generar una tensión analógica por RA0/AN0, con la que se selecciona un factor de aceleración entre 1 y 10.

Comentarios

El ejemplo trata de mostrar el efecto de la aceleración que se produce al variar la velocidad de los motores desde un valor a otro. La secuencia de movimiento se inicia con un pulso en RC1 conectado a un pulsador y consiste en pasar de velocidad 0 , a la máxima en un movimiento hacia adelante de ambos motores. Otro pulso en RC1 pasa de la velocidad máxima a velocidad 0.

Mediante la entrada analógica RA0/AN0 conectada a un potenciómetro, se ajusta el valor del factor de de aceleración entre 1 y 10. Se puede apreciar claramente el tiempo que transcurre para pasar de una velocidad a otra (aceleración). Por defecto este factor es de 5

6.6 Ejemplo 6: Realización de varios movimientos

Objetivos

Como ejemplo final se pretende mostrar la realización controlada de diferentes movimientos al tiempo que en la pantalla LCD se visualizan distintos parámteros.

Esquema

Se muestra en la figura 16. La pantalla LCD y el driver MD25 se conecta como en los ejemplos anteriores. Mediante el potenciómetro P1 del laboratorio, conectado a RA0/AN0, se regulará la velocidad a la que se realizará el movimiento y, mediante los interruptores E1 y E0 conectados con RC1 y RC0, se selecciona el tipo de movimiento a realizar: Adelante (00), Atrás (01), Derecha (10) e Izquierda (11).

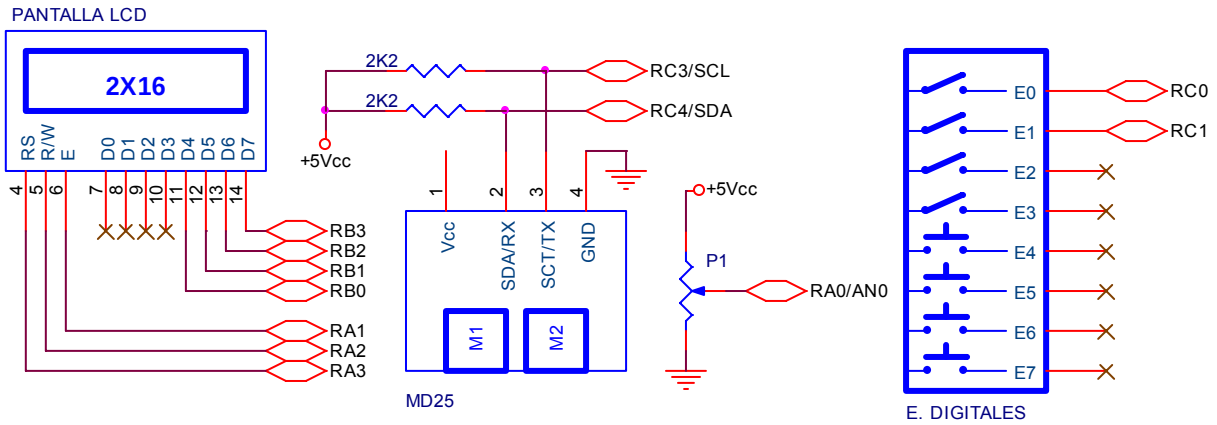


Figura 16. *Esquema de montaje para el ejemplo 6*

Comentarios

Mediante un potenciómetro conectado al canal A/D RA0, se regula la velocidad del movimiento seleccionado. Si dicha velocidad es de 0, (parada) el codificador B del motor 2 se pone a 0. Sobre la pantalla LCD, se visualiza los 16 bits de menos peso del encoder b (M2), tensión de trabajo (estado de la batería) y los consumos de cada motor.

La fotografía de la figura 17 muestra la ejecución de este ejemplo destacando la presentación de los datos de salida sobre la pantalla LCD.

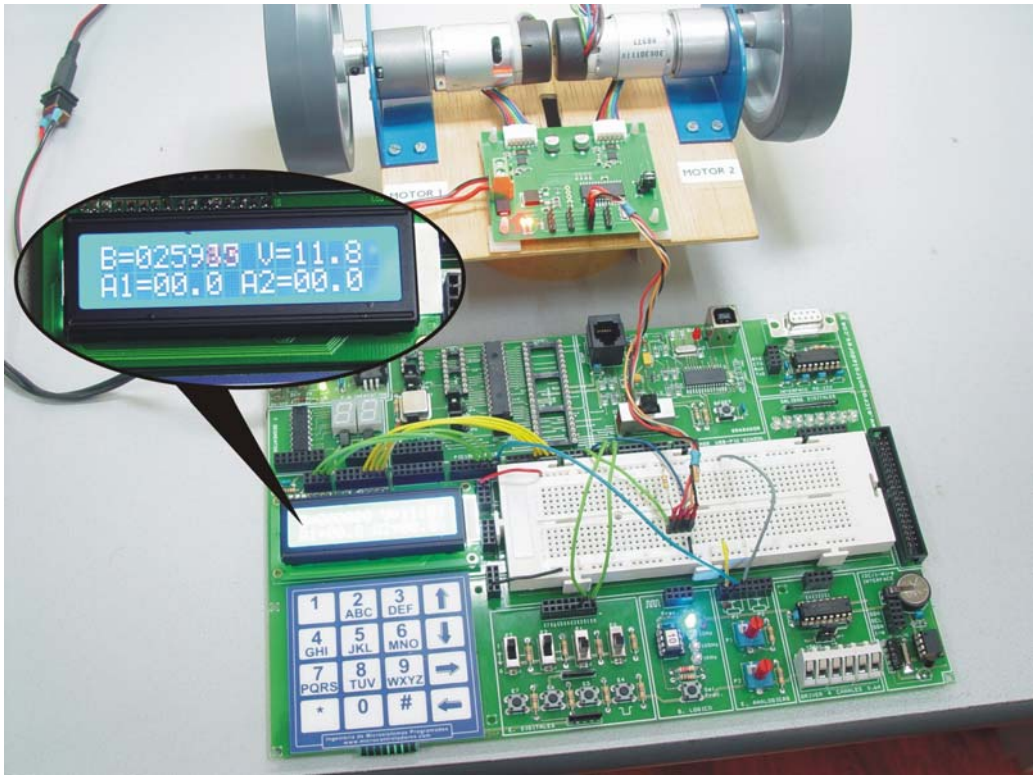


Figura 17. *Ejecución del ejemplo 6*