

**UNIVERSIDAD NACIONAL DE INGENIERÍA**  
**FACULTAD DE INGENIERÍA MECÁNICA**



**IMPLEMENTACION DEL CONTROL DE VELOCIDAD  
Y DISEÑO MECANICO DEL SISTEMA  
POSICIONADOR DE SOLDADURA MIG-MAG**

**TESIS**

**PARA OPTAR EL TÍTULO PROFESIONAL DE  
INGENIERO MECATRÓNICO**

**MARCELO JAIME QUISPE CCACHUCO**

**PROMOCIÓN 2003-II**

**LIMA – PERÚ**

**2011**

**Digitalizado por:**

**Consortio Digital del  
Conocimiento MebLatam,  
Hemisferio y Dalse**

*Dedicado a Mis padres:  
Santos y Teresa,  
y a mis 10 hermanos,  
por haberme permitido llegar  
a mi alma mater donde aprendí más que  
conocimientos  
y encontré mis mejores amigos.*

## CONTENIDO

|   |    |
|---|----|
| PRÓLOGO .....   | 1  |
| <b>CAPITULO I</b>   |    |
| INTRODUCCIÓN.....   | 3  |
| 1.1. Antecedentes .....   | 5  |
| 1.2. Justificación .....  | 6  |
| 1.3. Planteamiento del problema .....                                       | 7  |
| 1.4. Objetivo .....   | 8  |
| 1.5. Alcances y Limitaciones .....  | 9  |
| <b>CAPITULO II</b>  |    |
| DISEÑO MECÁNICO DEL SISTEMA POSICIONADOR .....                              | 10 |
| 2.1. Cálculo de la relación de transmisión del motor a la rueda .....       | 10 |
| 2.2. Diseño del sistema de transmisión y cálculo del torque necesario ..... | 13 |
| 2.2.1. Cálculo de los parámetros de diseño de los engranajes .....          | 16 |
| 2.2.2. Cálculo del torque necesario para mover la máquina .....             | 20 |
| 2.3. Diseño mecánico de la estructura de la máquina .....                   | 22 |
| 2.3.1 Sistema motriz .....  | 22 |
| 2.3.2 Sistema de protección de la parte electrónica .....                   | 23 |
| 2.3.3 Sistema de posicionamiento de la antorcha de soldadura. ....          | 24 |
| <b>CAPITULO III</b>   |    |
| EL MOTOR DC .....   | 26 |
| 3.1 Modelo matemático del motor DC .....                                    | 26 |
| 3.2 Identificación de parámetros del motor DC .....                         | 29 |
| 3.2.1. Esquema general de la adquisición de datos .....                     | 29 |

|   |  |    |
|---|--|----|
| 3.2.2.                                  | Programación del software para la identificación de parámetros ..... | 31 |
| 3.2.2.1                                 | Programación en labview .....  | 31 |
| 3.2.2.2                                 | Programación en Matlab .....   | 32 |
| 3.3                                     | Analizando el controlador continuo .....                             | 42 |
| 3.3.1.                                  | Diseño del controlador Proporcional-Integral (PI) .....              | 42 |
| 3.4                                     | Analizando el control discreto .....                                 | 47 |
| 3.4.1.                                  | Aproximadores digitales .....  | 48 |
| <br><b>CAPITULO IV</b>                  |  |    |
| DESCRIPCIÓN DE HARDWARE Y SOFTWARE..... |  | 51 |
| 4.1                                     | Microcontroladores Freescale: Familia HCS08 .....                    | 53 |
| 4.1.1.                                  | Arquitectura de la familia HCS08 .....                               | 54 |
| 4.1.2.                                  | Modelos disponibles .....  | 62 |
| 4.1.3.                                  | Diagrama de Pines .....  | 63 |
| 4.1.4.                                  | Características del microcontrolador MC9S08QE128 .....               | 63 |
| 4.1.5.                                  | Breve Descripción del módulo DEMOQE128 .....                         | 65 |
| 4.2                                     | Módulo TPM del microcontrolador MC9S08QE128 .....                    | 69 |
| 4.2.1.                                  | Definición de registros del módulo TPM .....                         | 69 |
| 4.2.1.1                                 | Registro de estado y control (TPMxSC) .....                          | 69 |
| 4.2.1.2                                 | Registro contador del TPM (TPMxCNTH:TPMxCNTL) .....                  | 71 |

|         |  |     |
|---------|--|-----|
| 4.2.1.3 | Registro módulo contador del TPM (TPMxMODH:TPMxMODL) ..              | 71  |
| 4.2.1.4 | Registro de estado y control de canal (TPMxCnSC) .....               | 72  |
| 4.2.1.5 | Registro de valor del canal (TPMxCnVH:TPMxCnVL) .....                | 74  |
| 4.3     | Módulo TPM como temporizador de propósito general .....              | 75  |
| 4.4     | Módulo TPM como PWM.....   | 78  |
| 4.4.1   | Funcionamiento del TPM como PWM (Pulse Width Modulation) .....       | 80  |
| 4.4.2   | PWM alineado al flanco .....   | 84  |
| 4.5     | El encoder y el módulo TPM como input capture.....                   | 87  |
| 4.5.1.  | El Encoder .....   | 87  |
| 4.5.1.1 | Sensor Foto – interruptores de barrera .....                         | 88  |
| 4.5.1.2 | Encoders Ópticos .....   | 88  |
| 4.5.2.  | El módulo TPM como INPUT CAPTURE .....                               | 90  |
| 4.6     | El teclado matricial y el microcontrolador MC9S08QE128 .....         | 97  |
| 4.7     | El display LCD 2x16 y el microcontrolador MC9S08QE128 .....          | 111 |
| 4.8     | El driver L293D y el microcontrolador MC9S08QE128 .....              | 129 |
| 4.8.1.  | Diagrama de bloques del L293D .....                                  | 130 |
| 4.8.2.  | Aplicaciones del L293D.....  | 131 |
| 4.9     | Software CODEWARRIOR .....   | 133 |
| 4.9.1.  | Creación de proyectos en C .....                                     | 133 |
| 4.9.2.  | Diagramas de Flujo de la programación del control del motor DC ..... | 139 |

|   |     |
|---|-----|
| 4.9.2.1 El programa principal .....   | 139 |
| 4.9.2.2 Interrupción por teclado KBI2 .....                                 | 141 |
| 4.9.2.3 Interrupción por captura de datos .....                             | 144 |
| 4.9.2.4 Interrupción por PWM.....   | 145 |
| 4.9.2.5 Interrupción por desbordamiento del módulo TPM1 .....               | 146 |
| 4.9.3. Depuración de proyectos en C .....                                   | 148 |
| <br><b>CAPITULO V</b>   |     |
| RESULTADO DE LAS SIMULACIONES .....   | 154 |
| 5.1 Simulación del algoritmo de control del motor .....                     | 154 |
| 5.1.1 I etapa: programación en Microchip en lenguaje assembler .....        | 155 |
| 5.1.2 II etapa: programación en Microchip en lenguaje C .....               | 157 |
| 5.2 Implementación de la tarjeta de control.....                            | 158 |
| 5.2.1 I etapa: Prototipo para uC Microchip PIC16F877A .....                 | 158 |
| 5.2.2 II etapa: Prototipo para uC Freescale MC9S08QE128 .....               | 160 |
| 5.3 Medición de la velocidad del motor DC .....                             | 163 |
| 5.3.1 Determinación de la velocidad máxima del motor empleado .....         | 163 |
| 5.3.2 Verificación de la lectura de velocidad con el uso de un tacómetro .. | 164 |
| 5.4 Prueba del PWM .....  | 166 |
| CONCLUSIONES Y RECOMENDACIONES .....  | 169 |
| BIBLIOGRAFÍA .....  | 171 |
| ANEXOS  |     |

## PRÓLOGO

El presente proyecto desarrolla el control del avance de una máquina semiautomática que sea capaz de realizar corte o soldadura de planchas metálicas de acero de hasta  $\frac{1}{2}$ " de espesor, con el fin de tener una mejor calidad en la soldadura. Para lo cual se realiza el control de un motor DC por PWM, utilizando un microcontrolador Freescale.

Para complementar este proyecto se realiza el diseño mecánico de la máquina, desde el sistema de transmisión formado por los engranajes, ejes y ruedas; hasta el diseño del sistema de posicionamiento de la antorcha de soldadura MIG – MAG.

En este proyecto una de las dificultades fue que se trabajó para las pruebas con un motor reciclado, cuyos parámetros se desconocían. En el Capítulo 3 de este proyecto se realiza la identificación de parámetros del motor DC, se explica el procedimiento empleado y se muestran los programas implementados así como los resultados de las simulaciones obtenidas luego de aplicar un control digital.

Para regular la velocidad de la máquina, se requiere ingresar la consigna de la velocidad previamente, de ese modo podemos operar la velocidad angular del

motor; esto es importante debido a que verificamos la performance de la respuesta en relación a la consigna, es decir obtenemos un error en estado estacionario nulo. En el capítulo 4 se describe el hardware utilizado así como la programación del microcontrolador, para controlar los diferentes dispositivos utilizados, en particular al motor DC.

Finalmente en el Capítulo 5 se muestran las pruebas realizadas en lo que se refiere al control de la velocidad del motor DC por Modulación de Ancho de Pulso (PWM). Finalmente se muestran los resultados de simulación y los software electrónicos, como son: Proteus, Code Warrior en Freescale y las pruebas obtenidas con diferentes tipos de lenguajes de programación, desde Assembler hasta lenguaje C.



## **CAPITULO I**

### **INTRODUCCIÓN**

La automatización del corte o soldadura no sólo mejora la calidad y el tiempo de producción. También reduce costos de muchas formas que no siempre son consideradas. Las máquinas portátiles de corte y soldadura están siendo utilizadas por un amplio rango de clientes, que van desde negocios muy pequeños hasta los grandes astilleros con cientos de soldadores y cortadores <sup>(1)</sup>.

El corte semiautomático usualmente es utilizado en corte de placas, tuberías, cortar figuras y preparar las uniones para soldadura. Un gran rango de beneficios se atribuye al realizar cortes de alta calidad antes del armado o soldaduras de piezas.

En este proyecto se aborda el diseño de una máquina de corte de planchas de acero, principalmente la máquina está formada por unos rieles y la máquina propiamente dicha, la máquina viaja sobre los rieles a una velocidad constante que se fija de acuerdo al espesor de la plancha. La máquina lleva la antorcha de una máquina de soldar MIG-MAG, que está a cierta distancia de la línea de corte, dicha distancia es regulable manualmente.

---

(1) Koike Aronson, Inc. Empresa fabricante de máquinas portátiles de cortes y soldadura

En la parte de diseño mecánico se toma como referencia la máquina que aparece en la Fig. 1. La parte más crítica de este proyecto es el control de velocidad de la máquina, porque el objetivo es que la máquina se desplace a velocidad constante. Para lograrlo se utiliza el control por *modulación de ancho de pulso* (PWM) de un motor DC.

Existen muchos trabajos de control de velocidad, la mayoría usando microcontroladores MICROCHIP, en este proyecto se inició con este microcontrolador; pero al no tener buenos resultados se cambió a un microcontrolador Freescale y se obtuvieron los resultados esperados. Éstos microcontroladores no son tan populares como los MICROCHIP y no hay mucha información acerca de librerías para manejo de teclado matricial y display LCD, así que se tuvo que crear librerías especialmente para este microcontrolador lo cual se detalla en el Capítulo 4 de este trabajo. En la parte del ingreso de velocidad de operación se realiza una innovación ya que para el ingreso de velocidad no se usa un perillero o potenciómetro como se ve en la máquina de referencia (ver Fig. 1), sino un teclado matricial con un display LCD 2x16.

En el display LCD se visualizará la velocidad deseada y la velocidad real, para confirmar que la velocidad ingresada es la velocidad deseada se presiona cualquiera de las teclas \* ó # del teclado matricial, la velocidad máxima de operación de la máquina es de 400 RPM - que equivale a un desplazamiento de 1800mm/min - así cualquier valor mayor a esta velocidad no se tomara en cuenta.

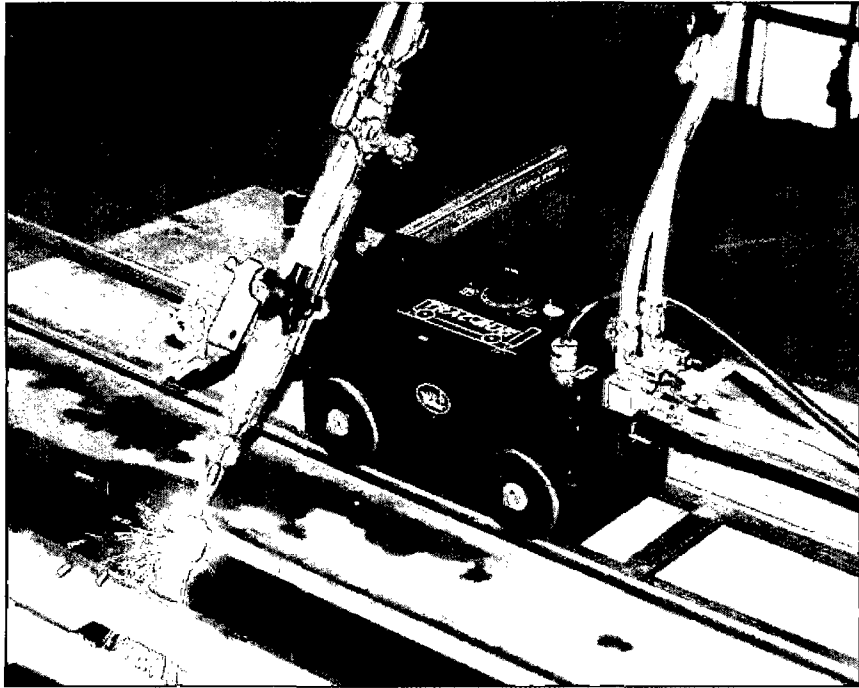


Fig. 1.1: Máquina de soldadura y corte por Oxy-Fuel o Plasma TRAC-BUG de la empresa BUG-O Systems[12].

## 1.1 ANTECEDENTES

En lo que se refiere a soldadura automática se conoce trabajos de robots manipuladores utilizados en los ensambles de autos que usan la tecnología de soldadura por puntos, en la actualidad ya se cuentan con máquinas CNC de corte por plasma [11]; pero se conoce poco de la tecnología de soldadura y corte semiautomático por MIG-MAG. Sin embargo, hay varias empresas que se dedican a la fabricación de estas máquinas, pero son equipos costosos. A pedido de una empresa privada (dedicada a la soldadura) a la Universidad de que se construya equipos similares a menor precio nació este proyecto.

Se han implementado muy pocos trabajos de investigación similares al proyecto de esta Tesis; sin embargo, lo que se ha podido encontrar es bastante información en relación al control de motores DC por PWM. Podemos hacer

referencia al trabajo del Ing. Gabriel Sánchez Suárez de la Universidad Francisco de Paula Santander España, cuyo trabajo está orientado a Microchip y lenguaje ensamblador. El aporte de esta tesis es que se ha usado el microcontrolador Freescale con lenguaje C.

Para este proyecto sólo se tuvo como punto de partida imágenes de folletos de las empresas fabricantes de estas máquinas, disponibles en internet [12], proporcionados por la empresa privada, y no se hizo el proceso de ingeniería inversa, se partió pensando en la forma que debería funcionar una máquina de este tipo. Al buscar información se encontró videos de la forma como operan estas máquinas y fue así que se hizo el diseño mecánico y el control del motor DC que se presentan en este proyecto.

## **1.2 JUSTIFICACIÓN**

El desarrollo de este proyecto se justifica ya que, las empresas dedicadas a la soldadura en nuestro país necesitan de estas máquinas; pero tienen un alto precio por lo cual solo cuentan con una máquina o en el peor caso no existen tales máquinas- Al disminuir el costo podrán tener más de dos máquinas; así mejorar la calidad, ahorro de material y disminuir el tiempo de producción.

La importancia de este proyecto es que en el mercado se encuentran distintos tipos de máquinas semiautomáticas de soldadura y corte, no solo por MIG-MAG sino también por Oxi-Fuel. Y es un campo bastante amplio para investigar. También es importante como proyecto de vinculación de la Universidad con el sector privado, así como el enfoque hacia una nueva línea de investigación y la aplicación práctica de la carrera.

### 1.3 PLANTEAMIENTO DEL PROBLEMA

En lo que se refiere a la construcción de la máquina lo más importante es el **cálculo del torque necesario del motor** así como de la transmisión de velocidad angular para tener una velocidad lineal de salida de 4-72 ipm (100-1800 mm/min)<sup>(2)</sup>, que son valores típicos de soldadura.

En lo que se refiere a diseño mecánico de la estructura se tiene imágenes de la forma; pero no dimensiones de las partes que conforman estas máquinas, disponibles en internet. Por lo tanto parte de este proyecto es **diseñar la estructura de la máquina**.

Tanto el tema de la dirección o giro de la máquina no se aplican en este proyecto, ya que se está abordando el corte recto de las planchas, y no se controla dirección porque la máquina viaja sobre unos rieles los cuales se fijan a la plancha o una superficie fija para realizar el corte.

El otro problema que se tiene es el control de la velocidad, es decir se fija una velocidad de operación según el espesor de la plancha y el motor debe llegar a esa velocidad de forma suave y en un tiempo corto, de tal manera que se tenga un corte perfecto.

Se probará el algoritmo de control en simulación para ver que funciona correctamente luego se implementará en un microcontrolador y se realizaran pruebas con diferentes motores que tengan encoders incrementales de diferente resolución.

---

(2) Velocidad de operación de máquina de corte por plasma Trac-Bug.

Para lograr un buen control de la velocidad de la máquina se estudiarán temas como control por modulación de ancho de pulso PWM. Para realizar la simulación de control del motor DC usando el control proporcional no se conocen los parámetros del motor DC con el que se realizan las pruebas, ya que es un motor reciclado; pero existen formas de **determinar la ecuación de transferencia del motor mediante algoritmos.**

## 1.4 OBJETIVOS

### 1.4.1 Objetivo general

Implementar el control de velocidad usando la modulación por ancho de pulso, para posicionar la antorcha de soldadura MIG-MAG en la operación de corte de láminas de acero.

### 1.4.2 Objetivos específicos

- Implementar un control por PWM para un motor DC cuyos parámetros se desconocen.
- Identificar las principales fuentes de información y comprender los métodos utilizados.
- Documentar las prácticas, métodos, modelo, resultados, análisis y diseño de la máquina.

## **1.5 ALCANCES Y LIMITACIONES**

Este tipos de máquinas tiene varios tipos de variantes como soldadura de grandes tuberías donde la máquina es guiada por una cadena que se sujeta a la tubería, soldadura de grandes tanques de almacenamiento en desplazamiento vertical, la máquina en ese caso se fija magnéticamente al tanque, dentro de las máquinas del tipo de este proyecto existen máquinas donde la antorcha no solo se desplaza en forma recta sino realizando figuras que son conocidas en la soldadura. En este proyecto solo se aborda el desplazamiento recto.

Este proyecto solo se realizaron simulaciones y pruebas del control del motor DC, de lo cual se tuvieron buenos resultados, pero no se han realizado aún pruebas de corte porque la máquina aún no está construida, se espera construir la máquina con el financiamiento de la empresa privada como se acordó en un primer encuentro.

## CAPITULO II

### DISEÑO MECÁNICO DEL SISTEMA POSICIONADOR

#### 2.1 CÁLCULO DE LA RELACIÓN DE TRASMISIÓN DEL MOTOR A LA RUEDA.

Para el diseño mecánico se toma como ejemplo la máquina TRAC-BUG/PLASMA de la empresa BUG-O SYSTEMS (Fig. 2.1). Según el catálogo disponible en internet de esta máquina su velocidad de desplazamiento varia de 4-72 pulgadas por minuto (100-1800 milímetros por minuto). Es decir, la velocidad máxima es de 3cm/seg. Este valor permite iniciar el cálculo de las dimensiones de la máquina.

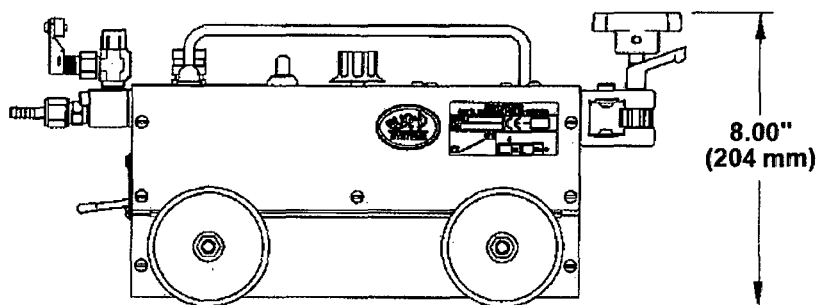


Fig. 2.1: Máquina TRAC BUG/PLASMA de BUG-O SYSTEMS



De acuerdo a la figura anterior, la dimensión que se tiene es el alto total de la máquina, por estimación se tiene que el diámetro de la rueda es  $D= 8.4$  cm, ésta dimensión es la que usaremos en los cálculos de los parámetros de los engranajes del sistema de transmisión.

Se sabe que la velocidad tangencial en una rueda es:

$$v_T = w * D/2 \quad (\text{Ec. 2.1})$$

Dónde:

$v_T$ : velocidad tangencial en cm/seg.

$w$ =velocidad angular de la rueda en rad/seg.

$D$ =diámetro de la rueda en cm.

Luego la velocidad angular máxima de la rueda- teniendo que la velocidad tangencial máxima de la rueda debería ser de  $v=3\text{cm/seg}$ - es:

$$w = 2 * v/D$$

$$w = 2 * 3/8.4$$

$$w = 0.7142 \text{ rad/seg} \quad (\text{Ec. 2.2})$$

En este proyecto la velocidad angular se trabaja en RPM (Revoluciones por minuto); para transformar las unidades de velocidad angular de rad/seg a RPM se usa la siguiente ecuación:

$$W_{RPM} = W_{rad/seg} * \frac{60}{2\pi} \quad (\text{Ec. 2.3})$$

Y finalmente la velocidad angular máxima de la rueda en RPM es:

$$W_{RPM} = 0.7142 * \frac{60}{2\pi}$$

$$W_{RPM} = 6.82 \text{ RPM} \quad (\text{Ec. 2.4})$$

La velocidad máxima del motor con reductor con el que se realizaron las pruebas es de 400 RPM; pero el duty cycle máximo del PWM es del 90%, que nos da como velocidad máxima 350 RPM según las pruebas realizadas. Lo cual requiere una relación de reducción de:

$$r = \frac{W_{MOTOR}}{W_{RUEDA}} = \frac{350}{6.82} = 51.3196 \quad (\text{Ec. 2.5})$$

Es decir una relación de **50 a 1**. Según las pruebas, al motor lo podemos controlar de una forma óptima entre 150 y 350 RPM. Y con la relación de 50 a 1, tendríamos una velocidad angular de salida en la rueda **de 3 a 7 RPM**, lo cual nos da una velocidad lineal de la máquina de

$$v = W_{RPM} * \frac{2\pi}{60} * D/2$$

$$v_{MIN} = 3 * \frac{2\pi}{60} * \frac{8.4}{2} = \frac{1.3194 \text{ cm}}{\text{seg}}$$

$$v_{MIN} = \frac{1.3194 \text{ cm}}{\text{seg}} = 791.68 \text{ mm/min}$$

$$v_{MIN} = 791.68 \text{ mm/min} \quad (\text{Ec. 2.6})$$

$$v_{MAX} = 7 * \frac{2\pi}{60} * \frac{8.4}{2} = \frac{3.0787 \text{ cm}}{\text{seg}}$$

$$v_{MAX} = \frac{3.0787 \text{ cm}}{\text{seg}} = 1847.25 \text{ mm/min}$$

$$v_{MAX} = 1847.25 \text{ mm/min} \quad (\text{Ec. 2.7})$$

Es decir la máquina se podrá desplazar entre 792 - 1847 mm/min. Es justificado este valor ya que el propósito de esta máquina es el de cortar planchas de hasta ½", y no se requiere una velocidad tan lenta.

Hasta este punto se observa que si se quiere que la máquina trabaje en velocidades de operación que son típicas de estas máquinas, se tiene que diseñar un sistema de reducción de velocidad del motor en la proporción de 50 a 1, que se detalla a continuación.

## **2.2 DISEÑO DEL SISTEMA DE TRASMISIÓN Y CÁLCULO DEL TORQUE NECESARIO:**

La potencia puede transmitirse desde un árbol a otro por medio de correas, ruedas de fricción engranajes o cadenas. Cuando la razón entre las velocidades tiene que ser constante se aplica ruedas de engrane. Es evidente que cualquier par de superficies que rueden juntas con un movimiento de rodadura pura, de manera a dar la relación de velocidades deseada, puede servir de base para el diseño de un par de ruedas dentadas. El movimiento transmitido por un par de ruedas dentadas bien diseñadas es idéntico al de las curvas o superficies básicas rodando una sobre otra. Para que un par de curvas puedan moverse una sobre otra con un movimiento

de rodadura pura, el punto de tangencia de las curvas tiene que hallarse siempre sobre la recta que une los centros de rotación de las curvas [3].

Para el diseño de la transmisión se toma el siguiente sistema de transmisión que es comúnmente usado en vehículos pequeños. Las 4 ruedas son engranajes cilíndricos rectos.

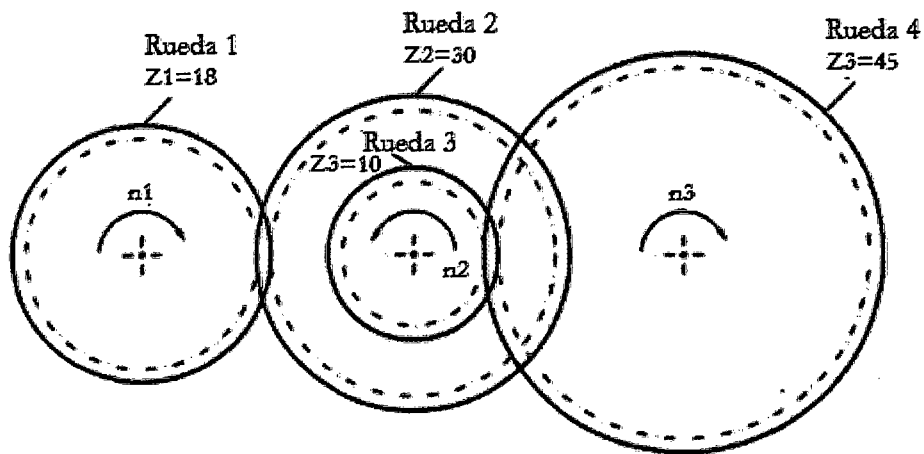


Fig. 2.2: Tren de engranajes del sistema de transmisión para las ruedas posteriores de la máquina.

(Z: número de dientes del engranaje)

En este sistema se tiene que la Rueda 1 es el piñón o engranaje unido al eje del motor, la rueda 4 es el engranaje que está unido al eje de las ruedas traseras y que da movimiento lineal a la máquina.

A continuación resolveremos este sistema de engranajes para determinar los diámetros de los 4 engranajes:

- De la rueda 1 a la rueda 2, ruedas tangentes:

$$v_{t1} = v_{t2}$$

$$w_1 * \frac{D_1}{2} = w_2 * \frac{D_2}{2}$$

$$w_2 = \frac{D_1}{D_2} * w_1 \quad (\text{Ec.: 2.8})$$

- De la rueda 2 a la rueda 3, ruedas concéntricas:

$$w_3 = w_2 \quad (\text{Ec.: 2.9})$$

- De la rueda 3 a la rueda 4, ruedas tangentes:

$$v_{t3} = v_{t4}$$

$$w_3 * \frac{D_3}{2} = w_4 * \frac{D_4}{2}$$

$$w_4 = \frac{D_3}{D_4} * w_3 \quad (\text{Ec.: 2.10})$$

Reemplazando la ecuación 2.8 en la ecuación 2.9 se tiene:

$$w_3 = \frac{D_1}{D_2} * w_1 \quad (\text{Ec.: 2.11})$$

Reemplazando la ecuación 2.11 en la ecuación 2.10 se tiene:

$$w_4 = \frac{D_3}{D_4} * w_3 = \frac{D_1 * D_3}{D_2 * D_4} * w_1$$

$$w_4 = \frac{D_1 * D_3}{D_2 * D_4} * w_1 \quad (\text{Ec.: 2.12})$$

Es decir que la relación de reducción de velocidad depende de los 4 diámetros de la siguiente forma:

$$G = \frac{D_1 * D_3}{D_2 * D_4} = \frac{1}{50} \quad (\text{Ec.: 2.13})$$

Es decir se tiene que  $w_{salida} = \frac{D_1 * D_3}{D_2 * D_4} * w_{entrada}$  (Ec.: 2.14)

Y también la relación de torques  $T_{salida} = \frac{D_2 * D_4}{D_1 * D_3} * T_{entrada}$  (Ec.: 2.15)

Hasta aquí se observa que mientras que la velocidad se reduce el torque aumenta en la misma relación G. Se tiene que calcular los parámetros de diseño de los 4 engranajes de tal manera que se cumpla la relación anterior **50 a 1** del ítem 2.1 de este proyecto.

### **2.2.1 Cálculo de los parámetros de diseño de los engranajes**

Para calcular los parámetros de diseño del engranaje se tiene que conocer dos de los siguientes parámetros, modulo, numero de dientes y diámetro primitivo. A partir de esos dos parámetros se calculan los demás como se muestran en la Tabla 2.1. El ángulo presión es estándar  $B=20^\circ$ .

Se empieza con el engranaje 1 que es el engranaje del motor, se tiene como primer dato el diámetro primitivo del engranaje, de acuerdo al motor con el que se trabaja en las pruebas estimamos el diámetro D1 en 10mm. Se toma el modulo  $m=1$  que es un módulo normalizado (ver **Anexo A**).

Tabla 2.1: Valores de diseño para el Engranaje 1. Dimensiones en mm.

|                          |                |       |
|--------------------------|----------------|-------|
| m (Módulo)               | Dato           | 1     |
| z (número de dientes)    | $D_p/m$        | 10    |
| $D_p$ diámetro primitivo | dato (mz)      | 10    |
| p (paso)                 | $\pi m$        | $\pi$ |
| a, addendum              | M              | 1     |
| b, dedendum              | $b=1.25m$      | 1.25  |
| $D_e$                    | $d_e=m(z+2)$   | 12    |
| $D_i$                    | $d_i=m(z-2.5)$ | 7.5   |
| L anchura de diente      | $L=10m$        | 10    |
| Tipo                     | evolvente      |       |

Obteniendo el siguiente engranaje con el software solidworks:

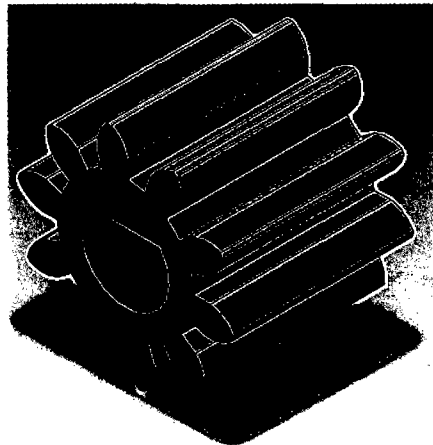


Fig. 2.3: Engranaje 1 piñón.

Seguimos con el engranaje 2, para que dos ruedas dentadas puedan engranar deben tener el mismo paso "p" o lo que es lo mismo, el mismo módulo "m", ya que  $p = m \cdot \pi$ .

El engrane 1 debe engranar con el engranaje 2 que es el engranaje de mayor tamaño del medio, entonces el siguiente paso es calcular los parámetros de diseño del engranaje 2, sabiendo que tendrá un

módulo  $m=1$  igual que el engranaje 1, y el Diámetro primitivo será de 60mm porque se requiere una relación de transmisión alta.

Tabla 2.2: Valores de diseño para el Engranaje 2. Dimensiones en mm.

|                         |                |         |
|-------------------------|----------------|---------|
| m (Módulo)              | Dato           | 1       |
| z (número de dientes)   | $Dp/m$         | 60      |
| $Dp$ diámetro primitivo | dato ( $mz$ )  | 60      |
| p (paso)                | $\pi m$        | $\pi m$ |
| a, addendum             | M              | 1       |
| b, dedendum             | $b=1.25m$      | 1.25    |
| $D_e$                   | $d_e=m(z+2)$   | 62      |
| $D_i$                   | $d_i=m(z-2.5)$ | 57.5    |
| L anchura de diente     | $L=10m$        | 10      |
| Tipo                    | Evolvente      |         |

Obteniendo el siguiente engranaje con el software solidworks:

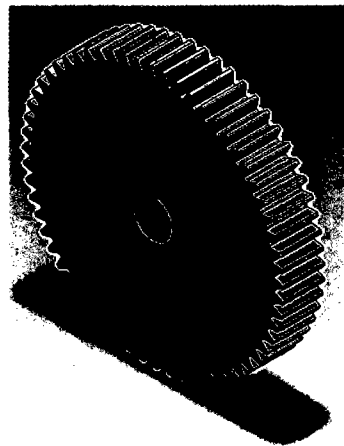


Fig. 2.4: Engranaje 2

Como primer diseño se tomó el engranaje 3 igual al engranaje 1 y el engranaje 4 igual al engranaje 2, obteniendo el siguiente sistema de transmisión:



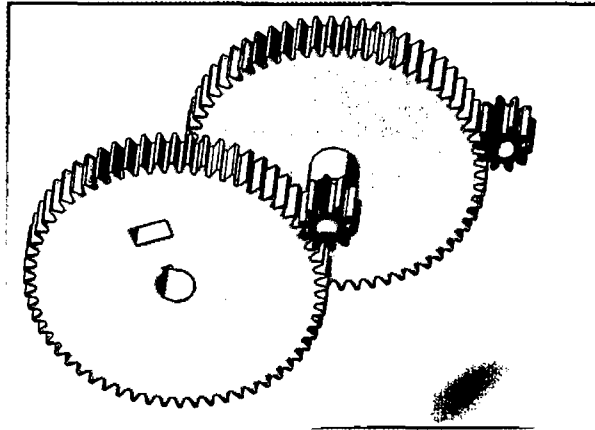


Fig. 2.5: Sistema de transmisión diseñado.

Este sistema de transmisión nos da una relación de transmisión de 1 a 36. Que es cercano a la relación inicial de 1 a 50, el limitante para tener una relación mayor es el ángulo de presión, ya que también esta normalizado en  $20^\circ$ . El sistema motriz incluye un motor y un par de ruedas que se desplazan sobre dos rieles, como se muestra a continuación:

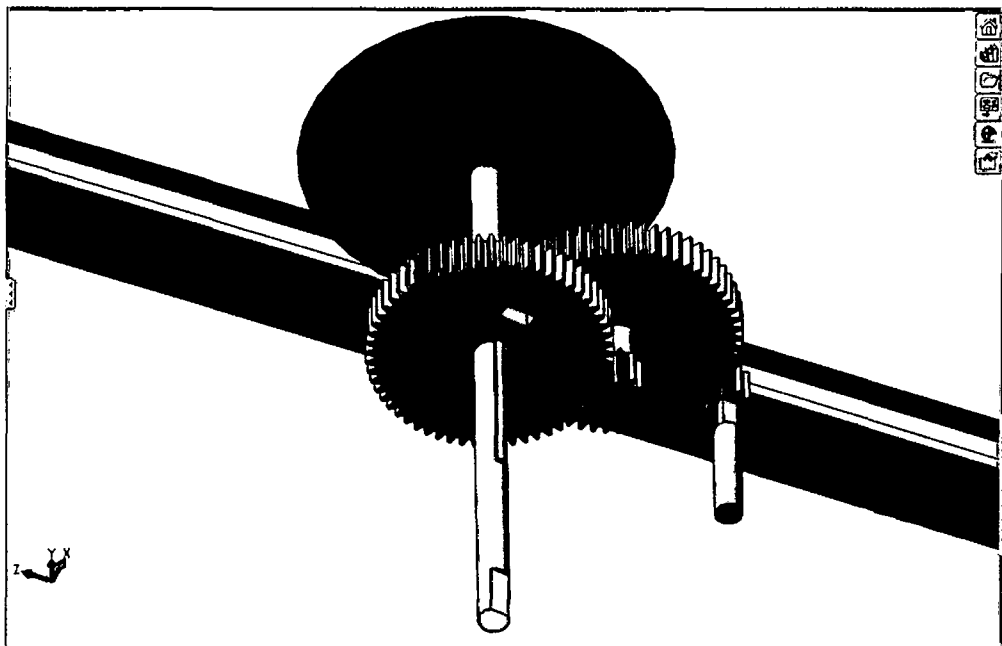


Fig. 2.6: Sistema completo de transmisión diseñado.

### 2.2.2 Cálculo del torque necesario para mover la máquina

Para el diseño de la máquina se sabe que tendrá 4 ruedas, y que el peso se distribuye uniformemente entre ellas. El peso máximo de la máquina que se está tomando como referencia es de 17 Kg según su catálogo disponible en internet. Luego el peso en cada rueda sería de 4.25Kg considerando este peso y el radio de la rueda, utilizando la ecuación fundamental de momento.

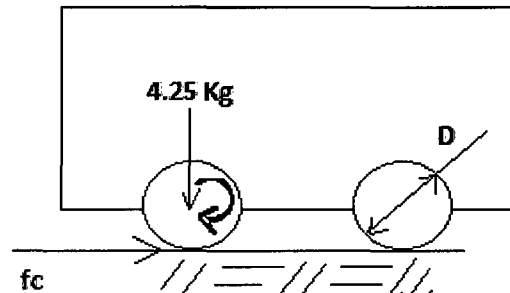


Fig. 2.7: Diagrama de cuerpo libre de una rueda

*Momento = Fuerza \* Distancia*

$$\text{Momento} = 4.25 * \frac{\text{Diametro}}{2}$$

$$\text{Momento} = 4.25 * \frac{8.4}{2} = 17.85 \text{ Kg.cm}$$

Obteniendo 17.85Kgcm de momento, sumando el par producido por el efecto del peso al considerar que el centro de gravedad del carro esta 2.4 cm arriba del eje de las ruedas, se obtuvo un momento debido al efecto del peso: 4.25 kg x 2.4 cm = 10.2 kg.cm

Por lo que el par de torsión requerido por diseño se determinó sumando  $17.85 + 10.2 = 28.05 \text{ kg.cm}$ , en una rueda, es decir el par total necesario es:

$$T_{salida} = 4 * 28.05 \text{ Kg. cm}$$

$$T_{salida} = 112.2 \text{ Kg. cm} \quad (\text{Ec.: 2.16})$$

Evidentemente, el par de torsión mínimo que debe generar el tren de engranaje debe ser mayor al que se necesita para mover la rueda. De acuerdo a la ecuación (2.15) se tiene:

$$T_{salida} = \frac{D_2 * D_4}{D_1 * D_3} * T_{entrada}$$

Según el primer diseño de transmisión realizado se tiene una relación de reducción G de 36 a 1, es decir:

$$T_{salida} = 36 * T_{entrada} \quad (\text{Ec.: 2.17})$$

Reemplazando la ecuación (2.16) se obtiene el par que debe proveer el motor DC al eje de las ruedas posteriores.

$$112.2 = 36 * T_{entrada}$$

$$T_{entrada} = 3.11 \text{ Kg. cm}$$

$$T_{entrada} = 3.11 * \left( \frac{9.81N}{1Kg} * \frac{1m}{100cm} \right) \text{ Kg. cm} = 0.3057N.m$$

Para asegurarnos que el motor suministre el suficiente torque, consideraremos un torque necesario del motor de:

$$T_{entrada} = 0.5 \text{ N.m} \quad (\text{Ec.: 2.18})$$

En resumen el motor DC debe tener las siguientes características:

Tabla 2.3: Parámetros de placa del motor DC a ser empleado.

| Parámetros necesarios       |     |
|-----------------------------|-----|
| Rpm                         | 400 |
| Par(N.m)                    | 0.5 |
| Parámetros opcionales       |     |
| Voltaje de alimentación (V) | 24  |

## 2.3 DISEÑO MECÁNICO DEL SISTEMA POSICIONADOR DE SOLDADURA MIG – MAG

La máquina consta de tres partes principales (para mayor detalles ver **Anexo B**).

### 2.3.1 Sistema motriz

Para el movimiento, la máquina tiene 4 ruedas, la tracción está en las ruedas posteriores, las cuales se desplazan sobre dos rieles, estos rieles se fijan a una superficie fija respecto de la máquina. Las ruedas se desplazan sobre los rieles sin deslizar por las guías de los rieles.

Muchas máquinas de este tipo usan diferentes tipos de rieles, nosotros usaremos el siguiente modelo:

Las ruedas de estas máquinas, las de tracción son de metal, con pequeñas hendiduras para asegurar que no haya deslizamiento.

### 2.3.2 Sistema de protección de la parte electrónica

La máquina cubre el sistema de transmisión, y contiene la tarjeta de control, tarjeta de potencia, transformador y cables de alimentación y buses de datos. Está compuesta por 4 partes

- Tapa inferior.
- Tapa superior.
- Tapa posterior.
- Tapa frontal.

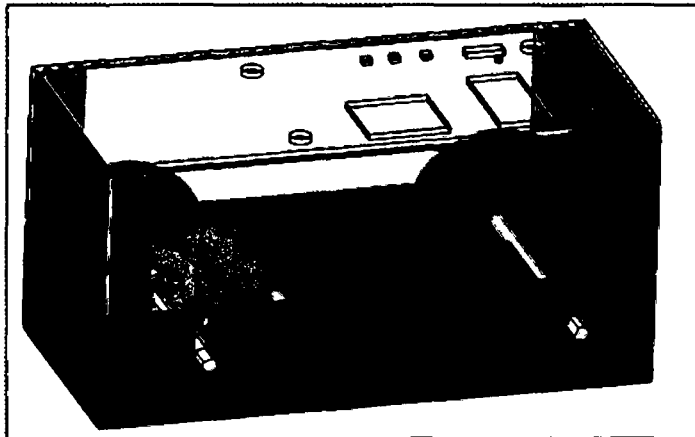


Fig. 2.8: sistema de protección de la parte electrónica.

En la parte frontal ira el teclado matricial, el display LCD 2x16, interruptores y otros componentes necesarios para que el operario realice el proceso de corte. Es decir el encendido, seteo de la velocidad y visualización de las velocidades real y deseada.

Estas 4 partes cubren a los componentes electrónicos de alta temperatura, polvo y humedad.

### 2.3.3 Sistema de posicionamiento de la antorcha de soldadura

Esta parte se une a la parte frontal de la máquina, sirve para unir la antorcha a la máquina, cuenta con un sistema manual que le permite cambiar el ángulo, la distancia de la boquilla hacia la línea de soldadura y también permite sujetar la antorcha de soldadura. Se observa que se tiene 4 grados de libertad, con lo cual se logra posicionar la antorcha a la distancia y ángulo de soldadura o corte deseado.

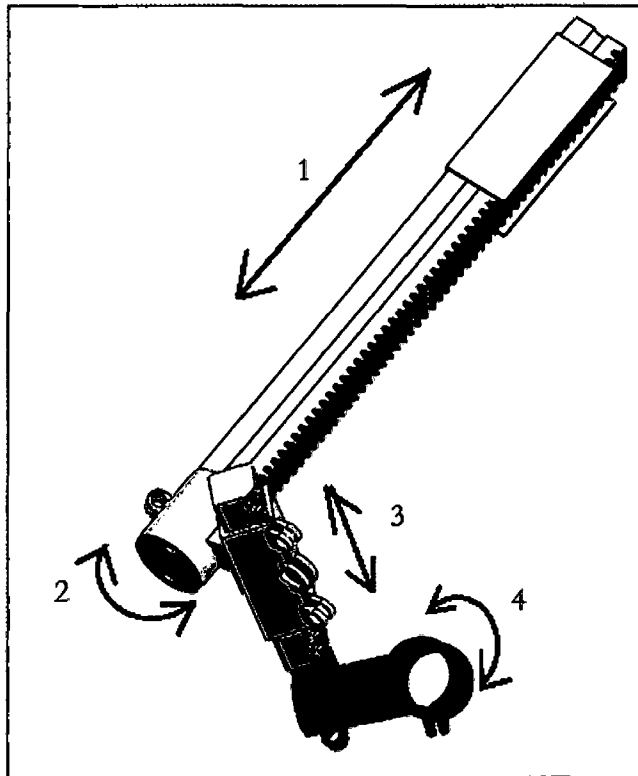


Fig. 2.9: Sistema de posicionamiento de la antorcha.

Para acabar este capítulo a continuación se muestran 2 principales vistas de la máquina completa (para mayor detalles ver Anexo B). Las dimensiones están en milímetros.

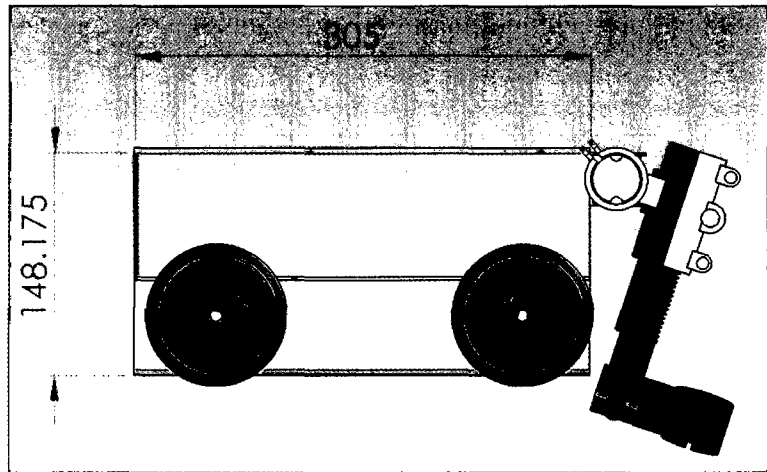


Fig. 2.10: Vista lateral.

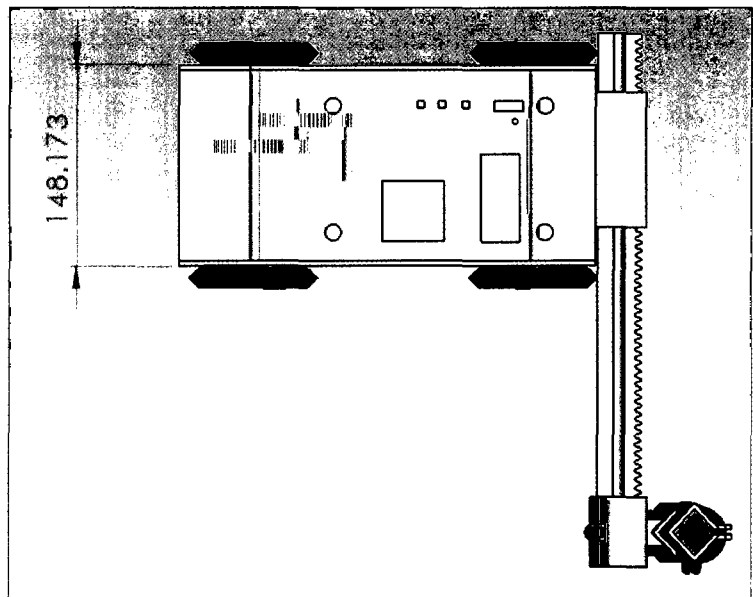


Fig. 2.11: Vista superior.

## CAPITULO III

### EL MOTOR DC

Para hacer que la máquina avance sobre los rieles en línea recta, para poder cortar la plancha, es necesario un actuador eléctrico. En este proyecto se emplea un motor DC que se puede controlar por diferentes métodos.

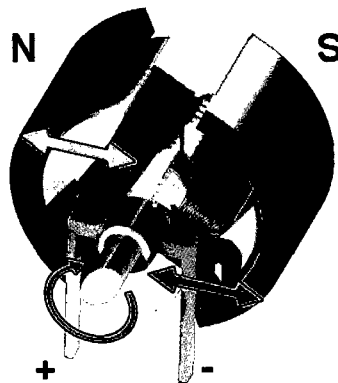


Fig.3.1: Esquema de un motor DC de imán permanente.

#### 3.1 MODELO MATEMÁTICO DEL MOTOR DC.

Para poder diseñar correctamente un controlador para el motor DC, es necesario obtener previamente el modelo matemático de dicho motor. Este modelo será prácticamente igual al dado por un motor simple de corriente continua, aunque el uso de engranajes introducirá ciertas variantes físicas y técnicas.



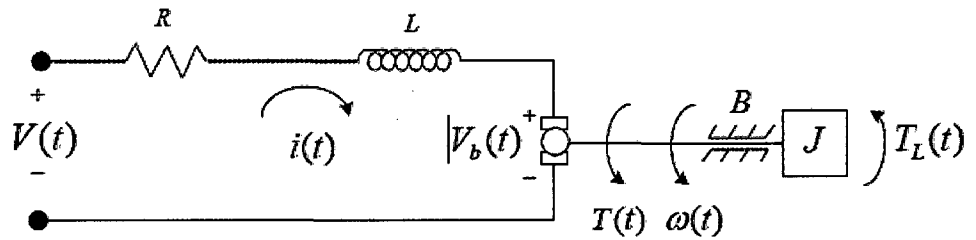


Fig. 3.2: Modelo Eléctrico - Mecánico del motor

| Parámetros                     | Símbolo        | Unidades          |
|--------------------------------|----------------|-------------------|
| Inercia                        | J              | Kg.m <sup>2</sup> |
| Fricción viscosa               | B              | Nm/rad/s          |
| Resistencia de armadura        | R              | Ohmios            |
| Inductancia de armadura        | L              | Henrios           |
| Constante contra electromotriz | K <sub>b</sub> | V/rad/s           |
| Constante de motor             | K <sub>T</sub> | Nm/A              |
| Velocidad angular              | w(t)           | rad/seg           |

Ecuaciones que gobiernan la dinámica del sistema electromecánico:

$$v(t) = Ri(t) + L \frac{d}{dt} i(t) + v_b(t),$$

$$v_b(t) = K_b w(t),$$

$$J \frac{d}{dt} w(t) + Bw(t) = T(t) + T_L(t),$$

$$T(t) = k_T i(t),$$

Tomando la transformada de Laplace.

$$V(s) = RI(s) + LsI(s) + V_b(s), \quad (\text{Ec. 3.1})$$

$$V_b(s) = K_b W(s), \quad (\text{Ec. 3.2})$$

$$JsW(s) + BW(s) = T(s) + T_L(s), \quad (\text{Ec. 3.3})$$

$$T(s) = k_T I(s), \quad (\text{Ec. 3.4})$$

Reemplazando la ecuación (3.2) en (3.1) se tiene:

$$V(s) = RI(s) + LsI(s) + K_b W(s), \quad (\text{Ec. 3.5})$$

Reemplazando la ecuación (3.4) en (3.3) se tiene:

$$JsW(s) + BW(s) = k_T I(s) + T_L(s), \quad (\text{Ec. 3.6})$$

Despejando  $I(s)$  de la ecuación (3.6) y haciendo que  $T_L(s)=0$  para motor sin carga:

$$I(s) = \frac{W(s)(Js + B)}{k_T}$$

Y reemplazando  $I(s)$  en la ecuación (3.5)

$$V(s) = I(s)(R + Ls) + K_b W(s)$$

$$V(s) = \frac{W(s)(Js + B)}{k_T} (R + Ls) + K_b W(s)$$

$$G_w(s) = \frac{W(s)}{V(s)} \Big|_{T_L(s)=0} = \frac{K_T}{(Ls + R)(Js + B) + K_b K_T}$$

Se observa que es una función de transferencia de segundo orden; pero como se trabaja en este proyecto con un motor reciclado por lo cual no se conocen los parámetros de la función de transferencia  $L$ ,  $R$ ,  $J$ ,  $B$ ,  $K_b$  y  $K_T$ . Para poder realizar las pruebas de control del motor DC se necesita conocer la función de transferencia. Para obtener la función de transferencia se usara el algoritmo ACIL[7] que permite identificar los parámetros del motor de acuerdo a la curva Voltaje versus tiempo, el empleo de este algoritmo para este proyecto se detalla a continuación.

### 3.2 IDENTIFICACIÓN DE PARÁMETROS DEL MOTOR DC

#### 3.2.1 Esquema general de la adquisición de datos

Para la adquisición de la señal de motor se desarrolló varias etapas las cuales se describen a continuación en un diagrama de bloques:

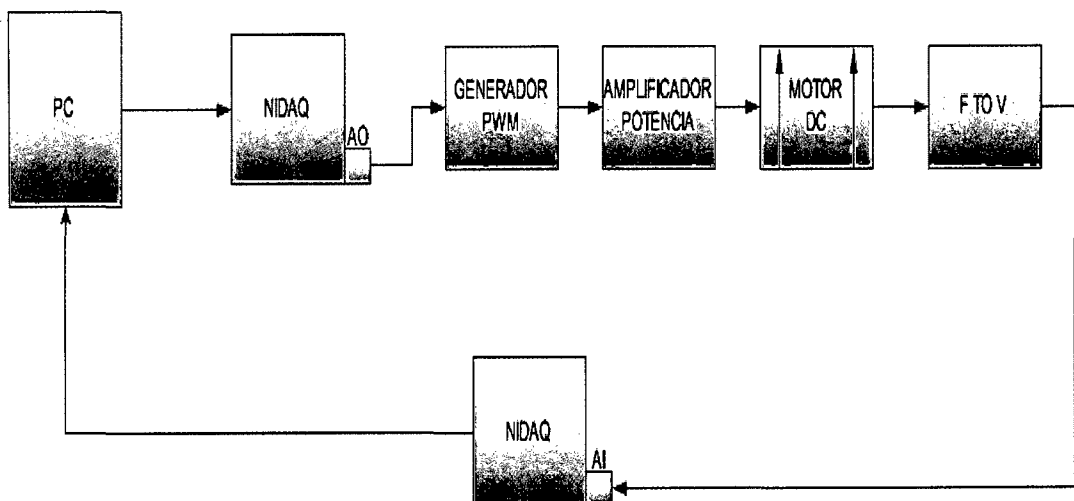


Fig. 3.3: Esquema general de la adquisición de datos.

- GENERADOR PWM

Para excitar al motor se envía un nivel de voltaje hacia el generador de PWM el cual nos generara un tren de pulsos con un ciclo de trabajo máximo a una frecuencia de 20KHz.

- AMPLIFICADOR DE POTENCIA

Está conformado por un puente H a base mosfets los cuales conmutaran a la frecuencia determinada para darle al motor el voltaje a plena carga la cual se tomara como referencia para el modelamiento respectivo.

- MOTOR DC

Es el sistema en estudio el cual recibirá la señal de excitación y partir del cual se tomaran las medidas de velocidad.

- F TO V

Este es un circuito conversor de la frecuencia del encoder del motor a niveles de voltaje los cuales expresan la velocidad que desarrolla el motor en el tiempo.

- NIDAQ AI- AO

La señal proveniente del circuito de frecuencia voltaje es registrada por el puerto AI de la tarjeta USB6008, los cuales se almacenaran en el PC para realizar la identificación respectiva.

Además del puerto análogo de la tarjeta se envía el nivel de voltaje necesario para excitar al generador de PMW.

### 3.2.2 Programación del software para la identificación de parámetros del motor DC

#### 3.2.2.1 Programación en labview

A continuación se muestra el programa que fue necesario para la adquisición de datos del motor, ya que la tarjeta NIDAQ trabaja con este software.

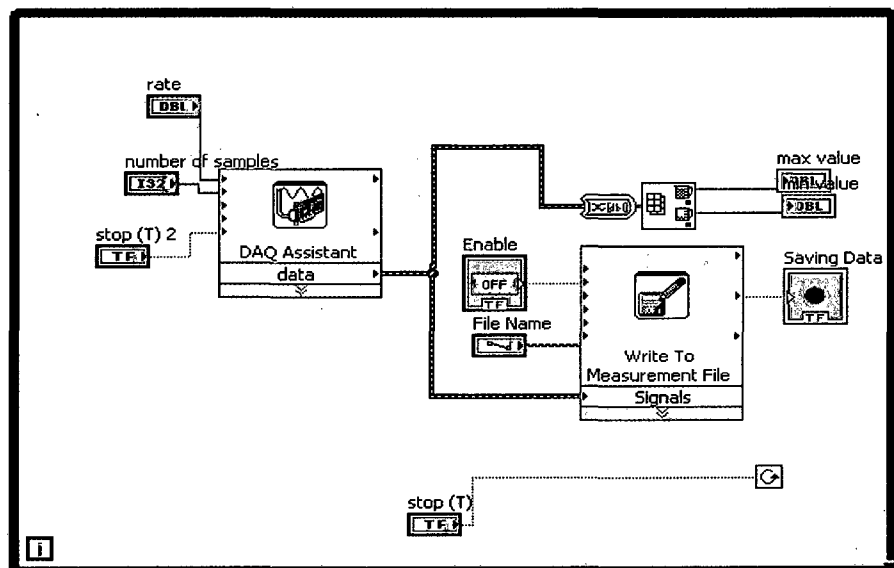


Fig. 3.4: Programa de adquisición de datos.

En la fig. 3.4 se observa que la data capturada por AN0 se almacena en forma simultánea en la PC, además de esto se calcula los valores máximos y mínimos de la data capturada.

Al final del proceso de adquisición ya se tiene una data inicial. A la data capturada se le omite la cabecera, y se guarda con nombre load data\_FV\_motor\_red1.lvm se lleva esta data a MATLAB donde se programa el algoritmo:

| Line | Time     | Value 1   | Value 2 |
|------|----------|-----------|---------|
| 1    | 0.000000 | -0.448484 |         |
| 2    | 0.001000 | -0.458653 |         |
| 3    | 0.002000 | -0.458653 |         |
| 4    | 0.003000 | -0.458653 |         |
| 5    | 0.004000 | -0.448484 |         |
| 6    | 0.005000 | -0.458653 |         |
| 7    | 0.006000 | -0.458653 |         |
| 8    | 0.007000 | -0.448484 |         |
| 9    | 0.008000 | -0.448484 |         |
| 10   | 0.009000 | -0.458653 |         |
| 11   | 0.010000 | -0.448484 |         |
| 12   | 0.011000 | -0.448484 |         |
| 13   | 0.012000 | -0.448484 |         |
| 14   | 0.013000 | -0.458653 |         |
| 15   | 0.014000 | -0.458653 |         |
| 16   | 0.015000 | -0.458653 |         |
| 17   | 0.016000 | -0.458653 |         |
| 18   | 0.017000 | -0.458653 |         |
| 19   | 0.018000 | -0.458653 |         |

Fig. 3.5: parte de la data adquirida

### 3.2.2.2 Programación en MATLAB

- **Cargando data en MATLAB**

```
load data_FV_motor_red1.lvm
```

```
DATA = data_FV_motor_red1;
```

```
t = DATA(:,1);
```

```
y = DATA(:,2);
```

```
ttol = round(max(t));
```

```
N = length(t);
```

```
F = N/ttol
```

```
plot(t,y);
```

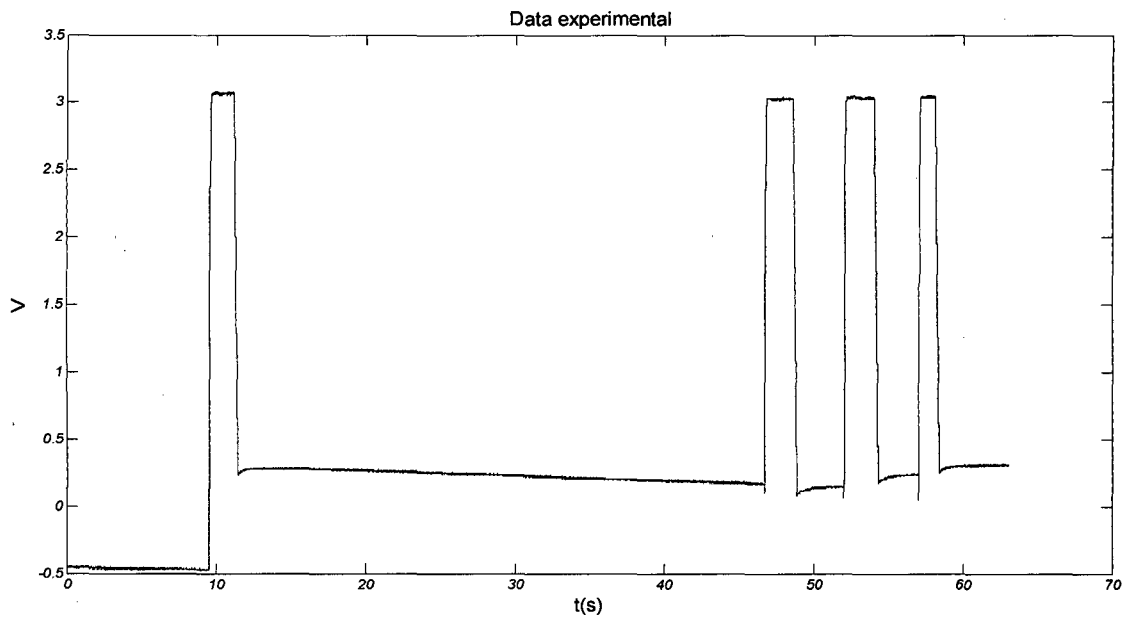


Fig. 3.6: Ploteo de la data capturada

- **Seleccionando la data para la identificación**

```
y =y(46696:47000);
```

```
N = length(y);
```

```
ymin = min(y);
```

```
y1 = y - ymin*ones(N,1);
```

```
N = length(y1);
```

```
y1 = y1(1:2:N);
```

```
N = length(y1);
```

```
Title('DATA EXPERIMENTAL');
```

```
plot(y1),xlabel('t(s)'), ylabel('V(s)'), axis([0 N 0 4]),title('DATA Experimental');
```

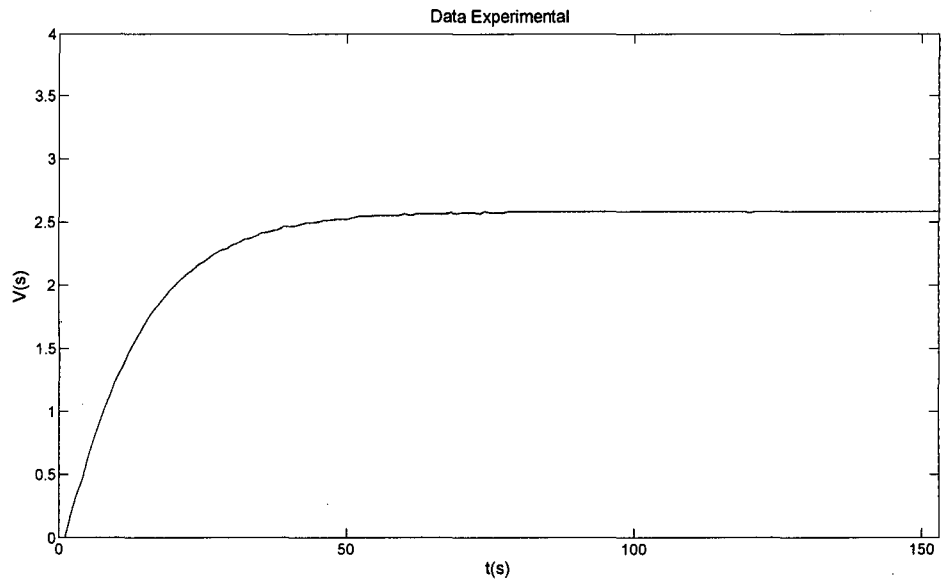


Fig. 3.7: Curva seleccionada.

Una vez que se tiene la data ahora procedemos a identificar los parametros del motor DC, para lo cual empleamos dos metodos :

- ✓ Algoritmo de identificacion ARX (Modelo autoregresivo con variables externas)
- ✓ Algoritmo de identificación por interpolación lineal con mínimos cuadrados

- **Algoritmo de identificación ARX**

Modelo a seguir con matlab:

$$\%y(k) = -a_1y(k-1) - a_2y(k-2) + b_1u(k-1) + b_2u(k-2)$$

%Condiciones iniciales:

$$olv = 0.52;$$

$$theta = [0 \ 0 \ 0 \ 0]';$$

$$P = 15 * eye(4);$$

$$e(3) = \max(y1);$$



```

for i=3:N

    m = [-y1(i-1) -y1(i-2) u(i-1) u(i-2)];

    k = P*m'/(olv + m*P*m');

    e(i) = y1(i) - m*theta;

    theta = theta + k*e(i);

    P = (P - P*k*m)/olv;

    a1(i-2) = theta(1);

    a2(i-2) = theta(2);

    b1(i-2) = theta(3);

    b2(i-2) = theta(4);

end

figure

plot(a1,'r')

hold

plot(a2,'g')

plot(b1,'b')

plot(b2,'y')

```

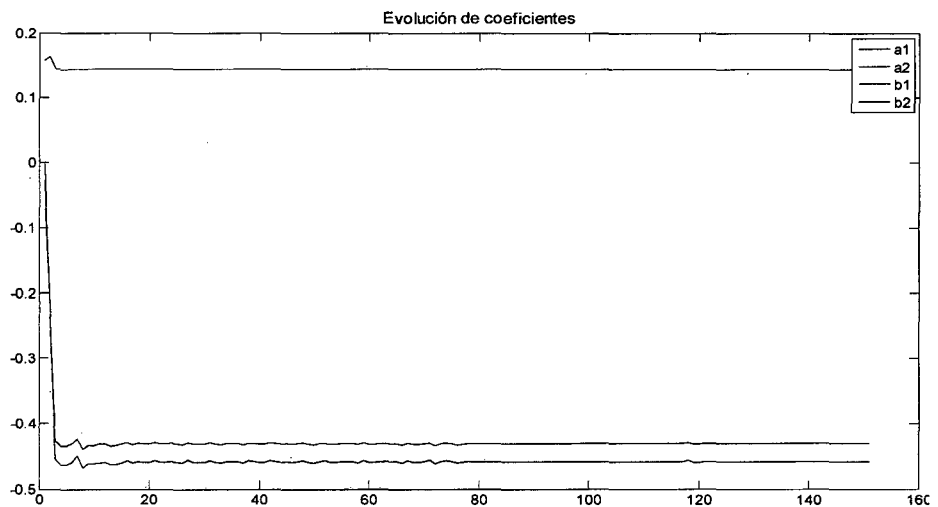


Fig. 3.7: Evolución de coeficientes en el tiempo

```
%reconstruyendo la ecuación en diferencias

Ts = 1;

N2 = length(a1)

a1 = median(a1(100:150));
a2 = median(a2(100:150));
b1 = median(b1(100:150));
b2 = median(b2(100:150));

y(k) = -a1y(k-1) - a2y(k-2) + b1u(k-1) + b2u(k-2)

n = [b1 b2];
d = [1 a1 a2];

G = tf(n,d,Ts)

SYS = d2c(G)

figure

t = 0:1:(N-1);

y = step(SYS,t);

plot(t,y,'b')

zpk(SYS)

hold

plot(t,y1,'r');

axis([0 N 0 4])
```

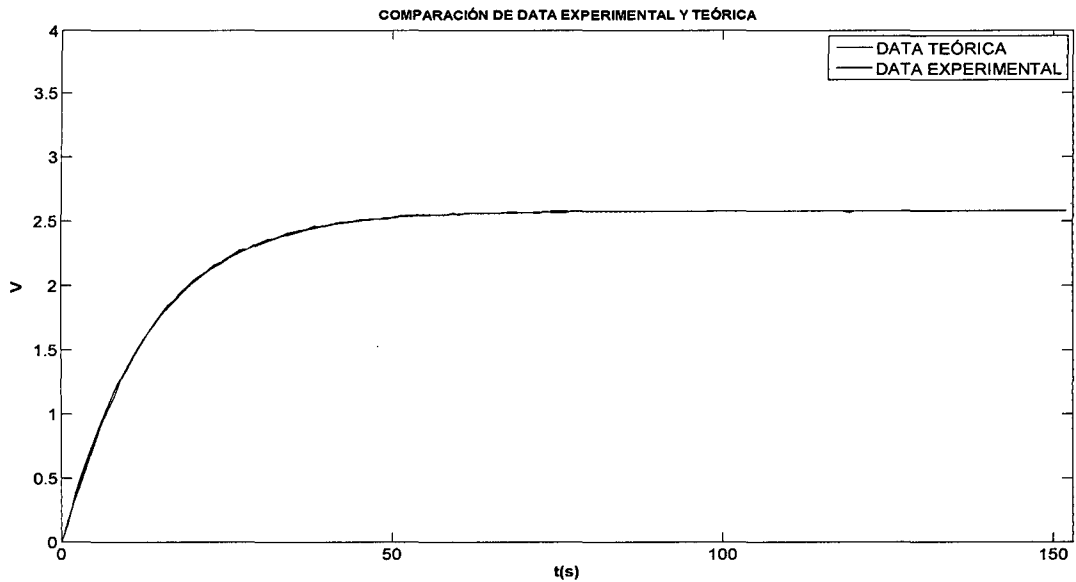


Fig. 3.8: Comparación de curvas real e identificada

**Función transferencia identificada:**

$$0.1785 s^2 - 0.07118 s + 2.07$$

---


$$s^3 + 1.481 s^2 + 10.47 s + 0.8014$$

- **Algoritmo de identificación por interpolación lineal con mínimos cuadrados**

%Se toma como modelo un sistema de orden 2 de la forma:

%  $A / (s + p1)(s + p2)$

%Seleccionando data necesaria para la identificación

```
load data_motor_red1.lvm
```

```
DATA = data_motor_red1;
```

```
t = DATA(:,1);
```

```
y = DATA(:,2);
```

```
%Seleccionando data
y2 = y(11821:1:12050);
t2 = t(11821:1:12050);
N = length(t2);
%eliminando offset
ymin = min(y2);
tmin = min(t2);
y3 = y2 - ymin*ones(N,1);
t3 = t2 - tmin*ones(N,1);

%Tomando el valor medio de estado estacionario para determinar la
ganancia del sistema
A = median(y3(190:230));
x_data = t3(1:60);
y_data = log(A-y3(1:60));
%calculando pendiente
t_min = 0.008; t_max = 0.024;
xi = t_min:0.001:t_max;
yi = interp1(x_data,y_data,xi,'linear');
figure
plot(x_data,y_data,'r'),axis([0 0.1 0 1]), title('Cálculo de mejor pendiente');
xlabel('t(s)'), ylabel('V voltaje');
hold on
plot(xi,yi,'.k');axis([0 0.1 0 1])
```

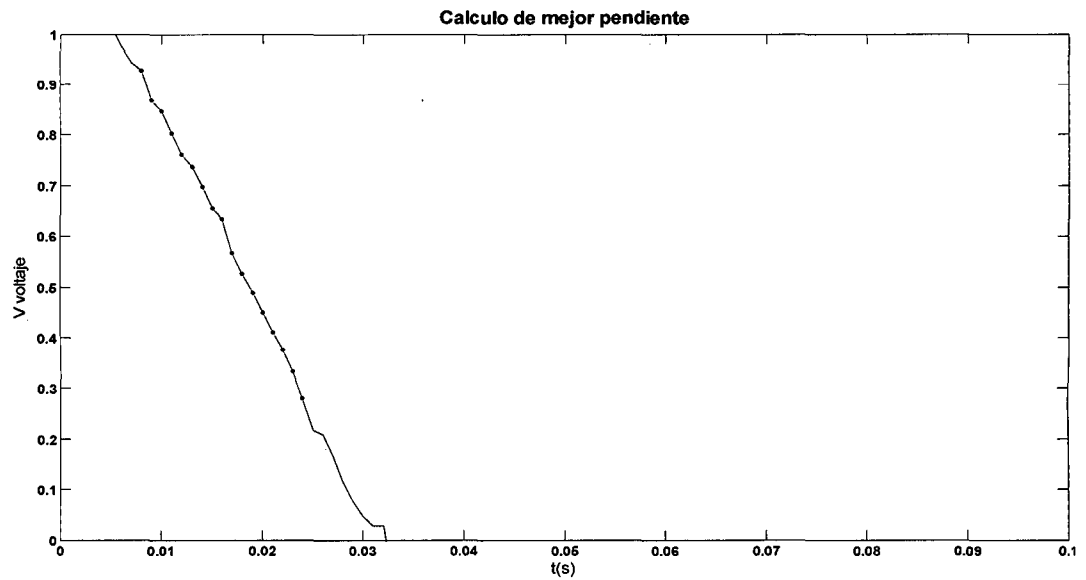


Fig. 3.9: Tomando mejor pendiente

```
%calculando p1=m, primer polo del sistema identificado
```

```
p1 = (yi(1) - yi(length(yi)))/(t_max-t_min)
```

```
%cálculo de B: B(t)= (y(t)- A)/ exp(-p1*t);
```

```
B = (y3 - A)./exp(-p1*t3);
```

```
figure
```

```
plot(t3, B, 'k.');
```

```
title ('Grafica de B(t)= y(t)-A/exp(-p1*t)');
```

```
axis([0 0.1 -30 0]);
```

```
hold on
```

```
plot([0 4], [median(B(6:11)) median(B(6:11))], 'b:')
```

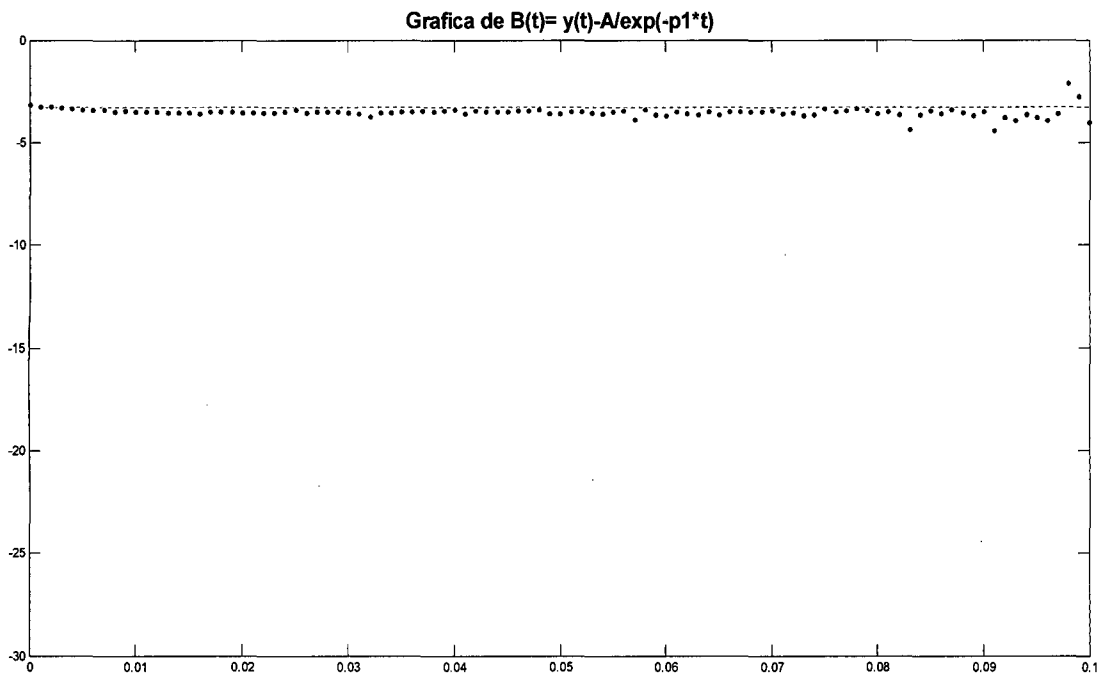


Fig. 3.10: Gráfica de la variación de B

$B = \text{median}(B(60:100))$

%cálculo de  $C = -(A+B)$

$C = -(A+B)$

%cálculo de P2

$p_2 = -(B/C) \cdot p_1$

%cálculo de la ganancia

$k = A \cdot p_1 \cdot p_2$

Resultados obtenidos del algoritmo:

$p_1 = 40.3684$

$p_2 = 460.5445$

$k = 5.8946e+004$

```

%función transferencia experimental

Gexp = tf(k,conv([1 p1], [1 p2]))

Gexp2= zpk(Gexp)

%[a b c d]= tf2ss(k,conv([1 p1], [1 p2]))

t4 = 0:0.0001:0.3;

yexp = step(Gexp,t4);

figure

plot(t4,yexp,'b');title('Gexperimental'), ylabel('Voltaje'), xlabel('t(s)');

hold on

plot(t3,y3,'r');

legend('G_{experimental}', 'G_{data}',4);

%plot([0 5], [1 1],'k:');

```

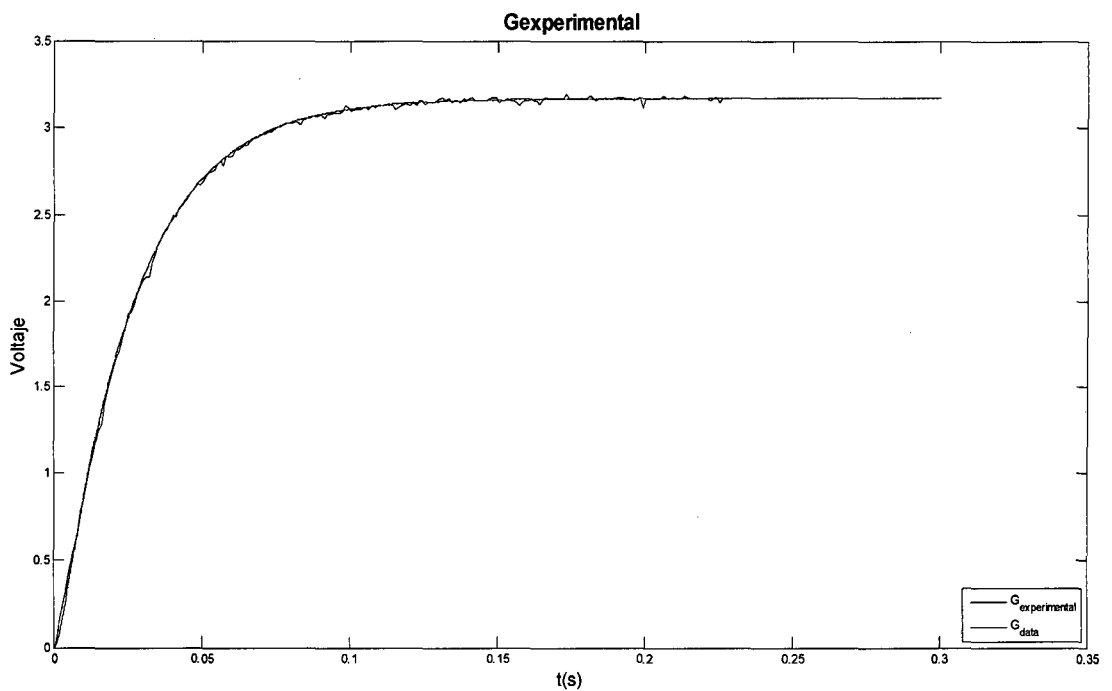


Fig. 3.11: Comparación de data experimental y teórica

**Función transferencia identificada:**

$$\frac{5.895e004}{s^2 + 500.9 s + 1.859e004}$$

En este momento se tienen dos funciones de transferencia que se comportan como la función de transferencia del motor DC, una de segundo y otra de tercer grado, de acuerdo al apartado 3.1 de este proyecto se tiene que de acuerdo al modelo electromecánico la función de transferencia es de segundo grado, por lo cual tomamos la función de transferencia.

$$G(s) = \frac{58950}{s^2 + 500.9s + 18590}$$

Como la función de transferencia de la planta, es decir la función de transferencia del motor DC.

### **3.3. ANALIZANDO EL CONTROLADOR CONTINUO**

#### **3.3.1 Diseño del controlador Proporcional-Integral (PI)**

El objetivo del diseño del controlador Proporcional - Integral (PI), es establecer los parámetros de sintonía deseados para encontrar la mejor performance del controlador. Las características principales del controlador PI son:



- a) Elimina el error estado estable
- b) Reduce el tiempo de subida
- c) Incrementa el tiempo de establecimiento

El controlador PI presenta la siguiente función de transferencia:

$$G_c(s) = \frac{K(s + a)}{s}$$

Donde K y a son la ganancia y cero del controlador que deben ser hallados. Para este fin utilizaremos una técnica muy conocida, que es la ubicación de los polos de la planta y ubicación de los polos en el lugar geométrico de las raíces.

Recordemos que se está trabajando con la función de transferencia de la planta:

$$G(s) = \frac{58950}{s^2 + 500.9s + 18590}$$

Considerando los polos de la planta  $s_{p1} = -460.533, s_{p2} = -40.366$ , y con las consideraciones de diseño como son, el tiempo de establecimiento  $t_s \leq 0.2 \text{ seg}$  y un amortiguamiento de  $\zeta = 0.5$ , podemos realizar nuestro diseño.

$$t_s = 0.2 \text{ seg}$$

$$\zeta = 0.5 \text{ seg}$$

$$M_p = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} = 16.3034\%$$

$$w_n = \frac{5}{\zeta t_s} = 50 \text{ rad/seg}$$

$$\sigma = -\zeta w_n = -25$$

$$w_d = w_n \sqrt{1-\zeta^2} = 25\sqrt{3} = 43.3013 \text{ rad/seg}$$

$$T_d = \frac{2\pi}{w_d} = 0.1451 \text{ seg}$$

$$T_s = \frac{T_d}{10} = 14.51 \text{ mseg}$$

Con los parámetros podemos obtener los polos deseados:

$$s_{1,2} = \sigma \pm jw_d$$

$$s_{1,2} = -25 \pm j25\sqrt{3}$$

Podemos graficar el lugar de las raíces desde el plano de la figura 3.12, en ella se observa el polo deseado, así como los polos de la planta y el cero del controlador PI cuyos parámetros están hallados aplicando el criterio de fase para el caso del cero y el criterio de la magnitud para la ganancia.

Según los parámetros de la figura 3.12 podemos hallar las magnitudes:

$$r_{int} = -25 + j25\sqrt{3}$$

$$r_2 = -15.36 + j25\sqrt{3}$$

$$r_1 = -435.53 + j25\sqrt{3}$$

$$r_3 =$$

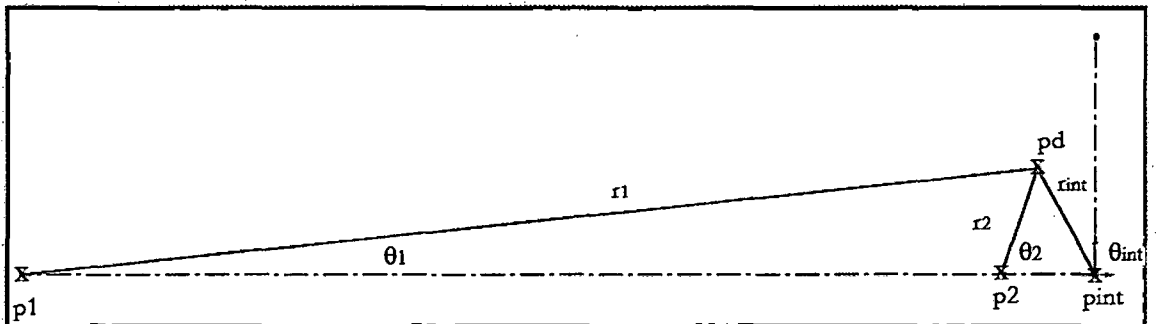


Fig. 3.12: Lugar de raíces del polo deseado

En este momento  $r_3$  aún no ha sido calculado, esta magnitud corresponde al cero del integrador que más adelante será calculado. En este caso lo resolveremos gráficamente por relaciones geométricas, en este caso con la ayuda del solidworks se obtiene los ángulos

$$\theta_{int} = 120^\circ$$

$$\theta_2 = 70.47^\circ$$

$$\theta_1 = 5.68^\circ$$

$$\theta_3 =$$

El valor de  $\theta_3$  corresponde al ángulo del cero del controlador.

Aplicando el criterio de la fase obtenemos:

$$\theta_{int} + \theta_2 + \theta_1 - \theta_3 = -180^\circ$$

Con ello obtenemos  $\theta_3 = 376.15^\circ = 16.15^\circ$ , entonces el valor del cero es  $a=174.531$  como se aprecia en la siguiente figura 3.13.

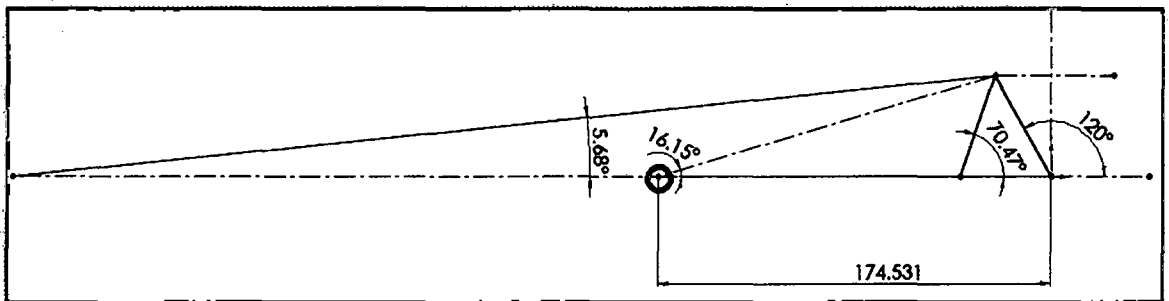


Fig. 3.13: Cálculo del cero del controlador.

Para el cálculo de la ganancia  $K$ , hacemos uso de la ganancia unitaria en el lugar de las raíces: **5.895e004**

$$K = \left| \frac{1}{\frac{58950 * r_3}{r_1 * r_2 * r_{int}}} \right| = \frac{r_1 * r_2 * r_{int}}{58950 * r_3} = \frac{437.677 * 45.945 * 50}{58950 * 155.674} = 0.1096$$

Finalmente el controlador queda de la forma:

$$G_c(s) = \frac{0.1096s + 17.0619}{s}$$

Probando la performance del controlador mediante simulación que se muestra en la figura 3.14.

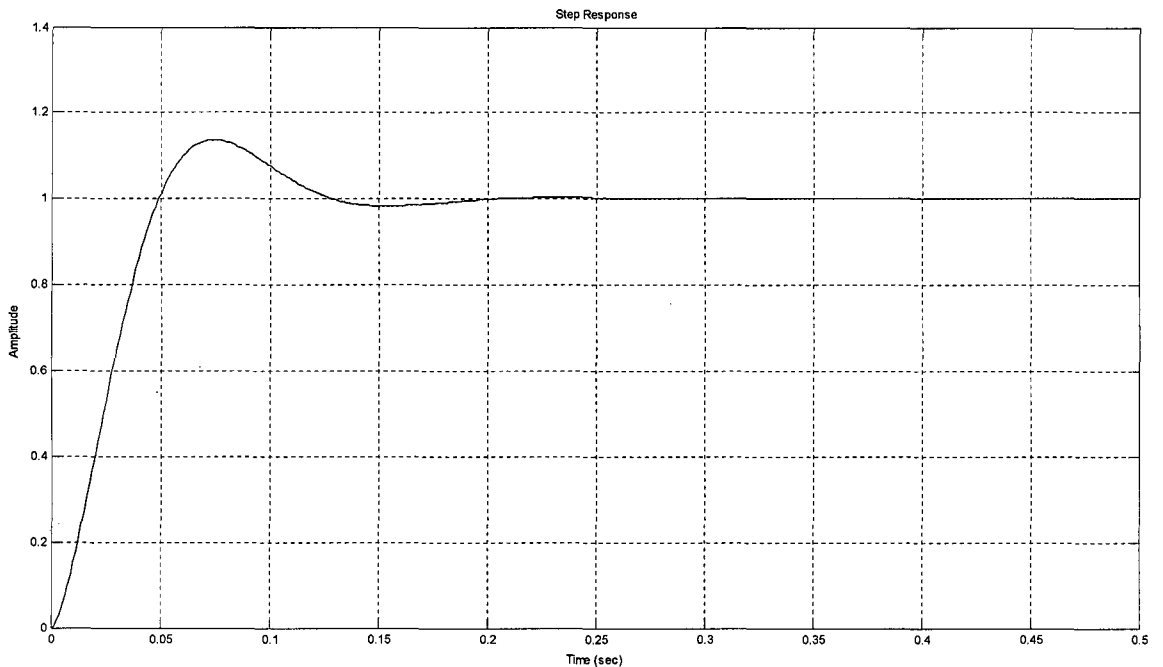


Fig. 3.14: Respuesta del sistema a una entrada escalón usando un controlador PI continuo.

### 3.4 ANALIZANDO EL CONTROL DISCRETO

Una vez obtenido tanto la ganancia como el cero del controlador PI, se procede a rediseñar el controlador, mediante los métodos de Control Digital, entre ellos el ZOH, Tustin y Tustin con predesvío. Para tal propósito es conveniente como primer paso, determinar el período de muestreo, ya que nos será de vital importancia para las conversiones y los futuros diseños que se realice al controlador mediante los diferentes métodos antes mencionados.

Entonces solo basta observar los cálculos realizados en la primera parte de este capítulo, en los cuales nos muestra un período de muestreo de 14.51 mseg, que por cierto, es muy pequeño pero como se sabe cuánto más pequeño sea el período de muestreo mejor será el diseño (Esto me permite probar un período de muestreo de  $T=0.005\text{seg}$ , ya que con este tiempo se hicieron las primeras pruebas

en el microcontrolador) y de esta manera el sistema se aproximará mejor al sistema original continuo.

### 3.4.1 Aproximadores digitales

Para el rediseño respectivo usando el método de *aproximación por ZOH*, lo que nos interesa es el período de muestreo, que en el paso anterior ha sido calculado, y así con este valor determinar la equivalencia, para posteriormente realizar el cambio respectivo. Para obtener la función de transferencia del controlador discreto, realizamos la discretización con el elemento ZOH que resulta como sigue:

$$G_{ZOH} = (1 - z^{-1})Z \left\{ \frac{G_c(s)}{s} \right\}$$

$$G_{ZOH} = \frac{0.1096(z + 1.2573)}{z - 1}$$

Para el *controlador de Tustin*, al igual que el caso anterior, se tiene en cuenta el tiempo de muestreo que es fundamental en el rediseño del controlador, ya que con este dato, podemos calcular la equivalencia y hacer el cambio respectivo. Para encontrar la función de transferencia, hacemos uso del cambio del operador por equivalente  $\frac{2z-1}{Tz+1}$  resultando lo siguiente:

$$G_{Tustin} = G_c(s) \Big|_{s=\frac{2z-1}{Tz+1}}$$

$$G_{Tustin} = \frac{0.2333(z + 0.0604)}{z - 1}$$

En la figura 3.15 se muestra las respuestas de los aproximadores digitales debido a una entrada escalón unitario para un tiempo de muestreo de 14.51 mseg.

En la figura 3.16 observamos una mejora en la respuesta cuando ponemos un tiempo de muestreo de  $T=0.005$  seg, que es el tiempo de muestreo que se utilizó en el microcontrolador desde un principio.

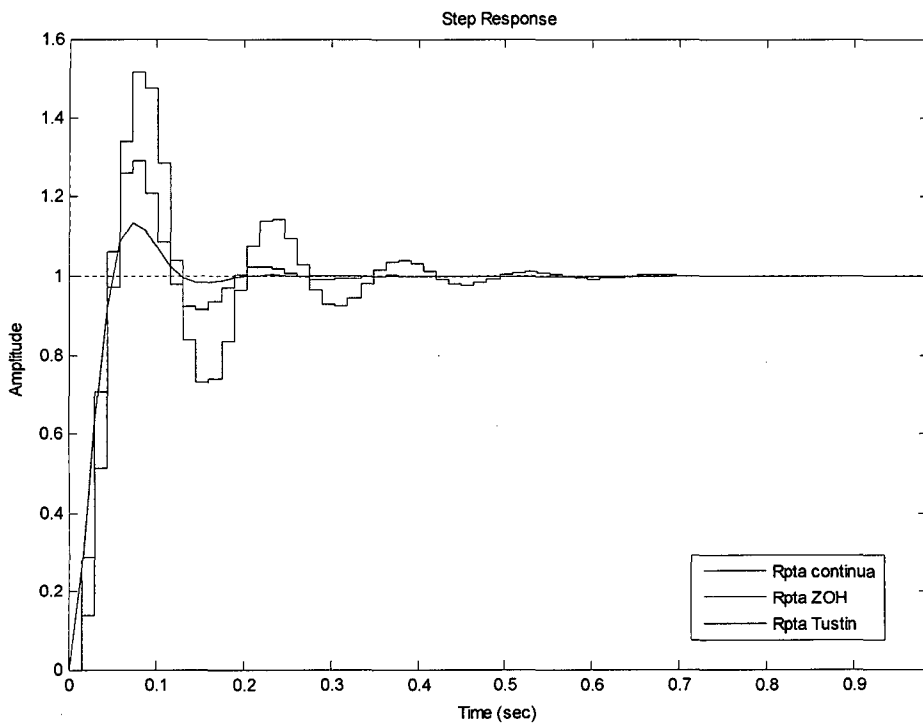


Fig. 3.15: Respuesta del sistema al escalón usando aproximadores digitales con  $T=0.0145$ s.

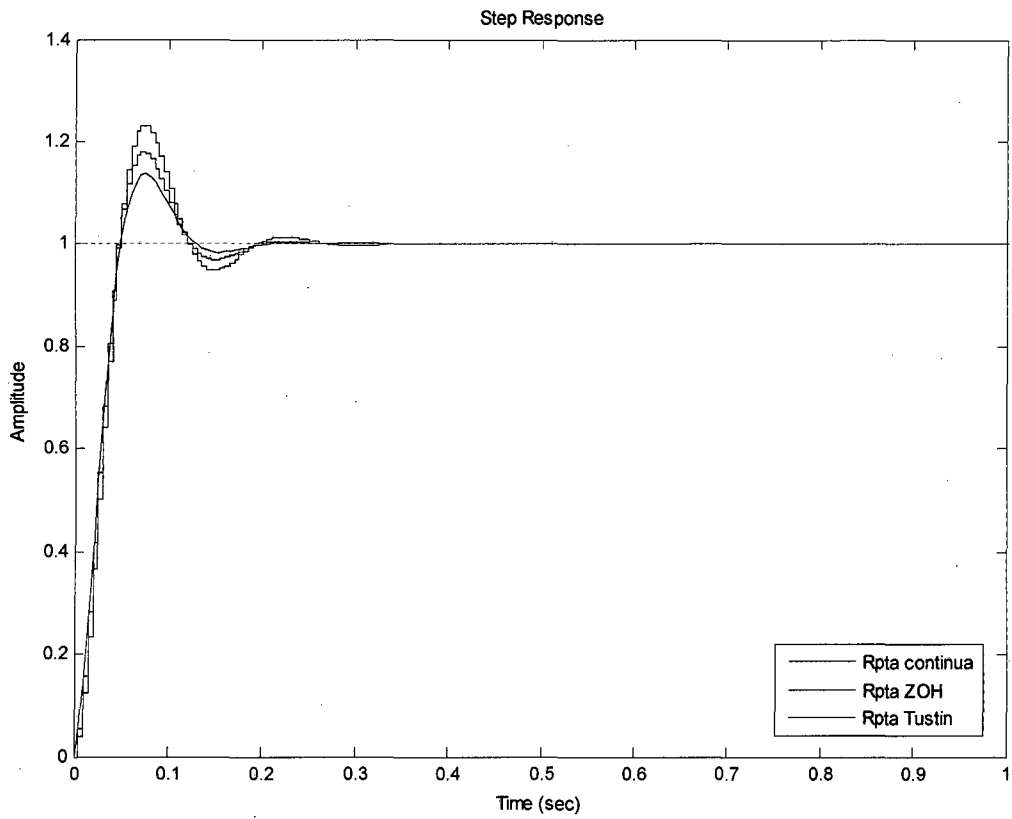


Fig. 3.16: Respuesta del sistema al escalón usando aproximadores digitales con  $T=0.005s$ .



## CAPITULO 4

### DESCRIPCION DE HARDWARE Y SOFTWARE

Para realizar el control por PWM se realiza un control en lazo cerrado:

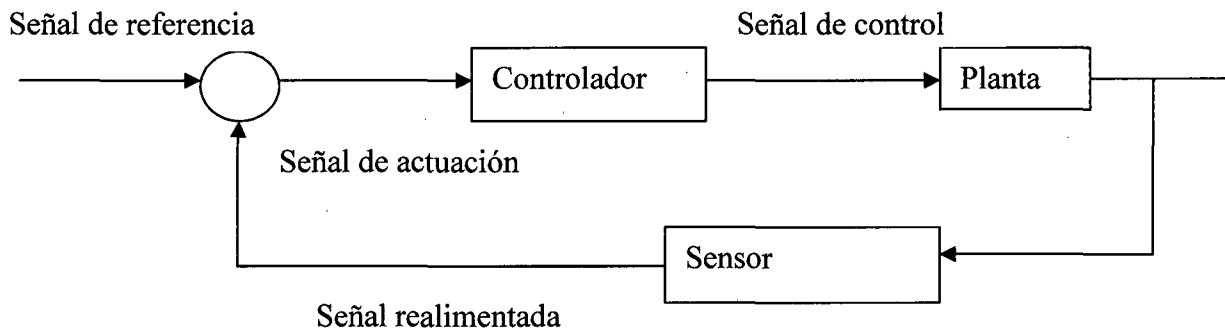


Fig. 4.1: Diagrama de bloques del control empleado.

Para lograr dicho control a continuación se muestra un esquema: los dispositivos que conforman este sistema, la relación entre cada uno y las funciones que cumplen en el control de la velocidad del motor DC por PWM.

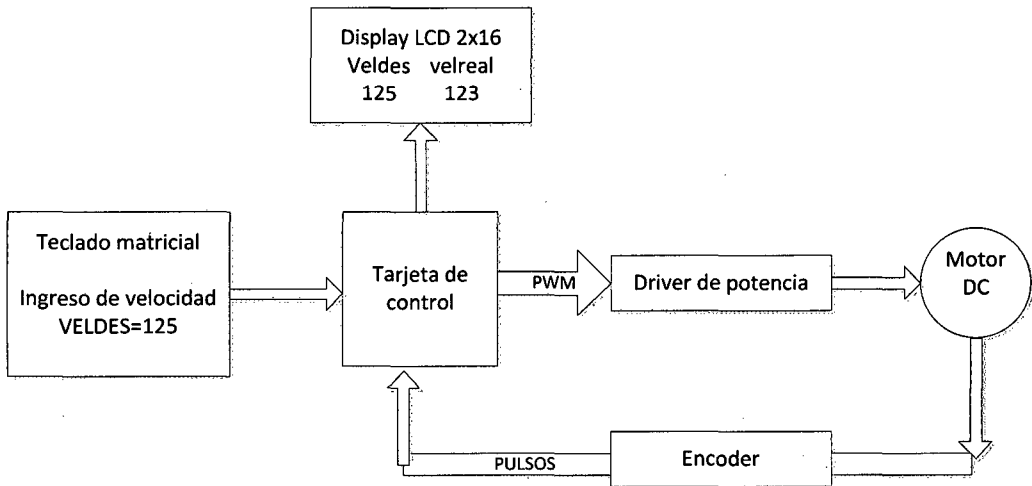


Fig. 4.2: Diagrama de bloques del sistema de control de velocidad de la máquina.

Los elementos que conforman este sistema son:

- Teclado matricial 4x4: Para el ingreso de la velocidad deseada.
- Tarjeta de control que incluye un microcontrolador de la familia Freescale: Realiza el algoritmo de control.
- Display LCD 2x16: Monitorea la velocidad deseada y la real del motor.
- Driver de potencia: Da el voltaje DC necesario para girar el motor.
- Motor DC: Da el movimiento giratorio a velocidad angular constante.
- Encoder: Esta unido al eje del motor y nos da la velocidad del motor DC.
- Tren de engranajes: Reducen la velocidad angular y le dan a la máquina una velocidad lineal baja.

A continuación se describe cada uno de ellos, así como el algoritmo, pseudocódigo y las configuraciones necesarias en los registros del microcontrolador.

#### 4.1 MICROCONTROLADORES FREESCALE: FAMILIA HCS08 [4]

La historia de esta familia se remonta a 1979, cuando Motorola desarrolla los primeros microcontroladores de 8 bits (M6805, M146805 y MC6800).

Las mejoras en la tecnología del silicio posibilitó la creación de la familia M68HC05 (tecnología CMOS), años después otras mejoras en los M68HC05 dieron como resultado la creación de la familia HC08; estos chips se caracterizaron por un buen soporte para su programación en C gracias a la inclusión de un registro extra (H, nuevos modos de direccionamiento e instrucciones).

En los últimos años la aparición de la familia HCS08 ha traído mejoras en tres frentes: La inclusión de un hardware de depuración, aumento de la velocidad del reloj y reducción del consumo general de potencia.

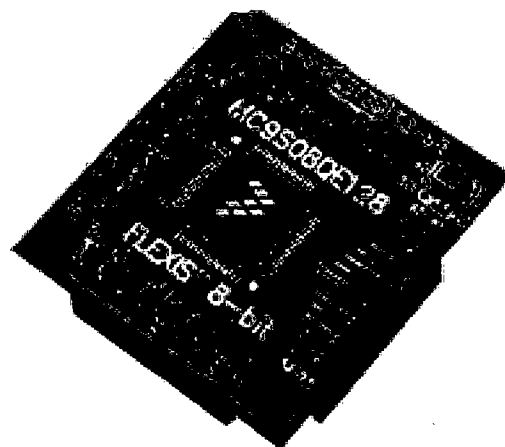


Fig. 4.3: Microcontrolador Freescale MC9S08QE128-LQFP64

#### **4.1.1 Arquitectura de la familia HCS08**

La CPU de los HCS08 está basada en la arquitectura Von Newman (también conocida como Princeton) con un CISC (Conjunto complejo de instrucciones de computadora). Esta arquitectura se caracteriza por el uso de un bus de datos común compartido entre la memoria y los periféricos, un bus de direcciones (responsable de la selección e la dirección de memoria o del registro del periférico a ser ejecutado) y un bus de control (Para controlar el tipo de operación que será ejecutada). Esto significa que la CPU solo ve un espacio de direcciones, compartido entre el código de la aplicación y los datos del usuario. En la CPU de los HCS08 el bus de datos está dividido en dos buses separados: uno para las operaciones de escritura y otro para las operaciones de lectura.

El bus de direcciones es de 16 bits, permitiendo el direccionamiento de hasta 216 bytes o 65536 bytes. El bus de datos es de 8 bits, permitiendo el movimiento desde o hasta la CPU de bloques de 8 bits.

El uso de un conjunto de instrucciones CISC supone que existen muchas operaciones complejas codificadas en instrucciones simples. Por un lado esto puede hacer que el conjunto de instrucciones en lenguaje de ensamblado sea difícil de aprender; por otro lado, los procesadores de arquitectura CISC generalmente son altamente compatibles con el lenguaje C y esto ayuda a que los compiladores de C traduzcan un código más eficiente.

Otro concepto importante de la arquitectura de los HCS08 es la asignación de la memoria a los periféricos: Algunos espacios de dirección son usados para acceder a los registros de periféricos. De esta manera, las mismas instrucciones usadas para manipular datos pueden ser usadas para leer o escribir los registros de periféricos.

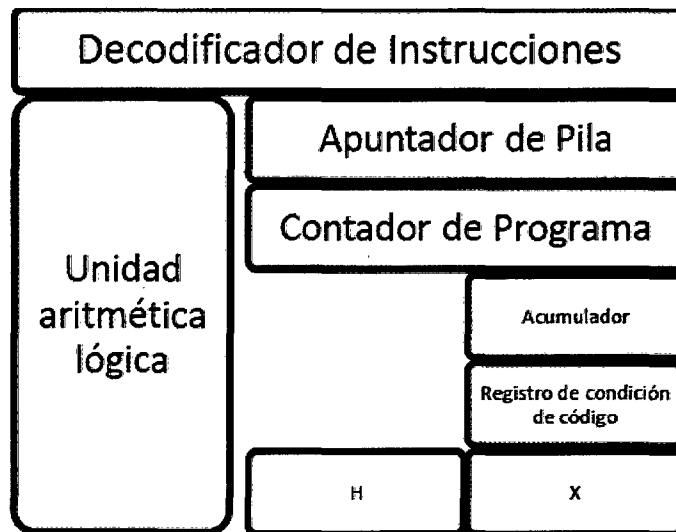


Fig. 4.4: Diagrama de bloques simplificado de la CPU de los HCS08

La CPU de los HCS08 también usa cinco registros especiales (dentro del núcleo).

- Un acumulador de 8 bits (A)
- Un contador de programa de 16 bits (PC)
- Un registro de direccionamiento de 16 bits (H:X)
- Un apuntador de pila de 16 bits (SP)
- Un registro de condición del código de 8 bits (CCR)

Estos registros internos no aparecen en el mapa de memoria del procesador y son guardados automáticamente cuando una interrupción es llamada.

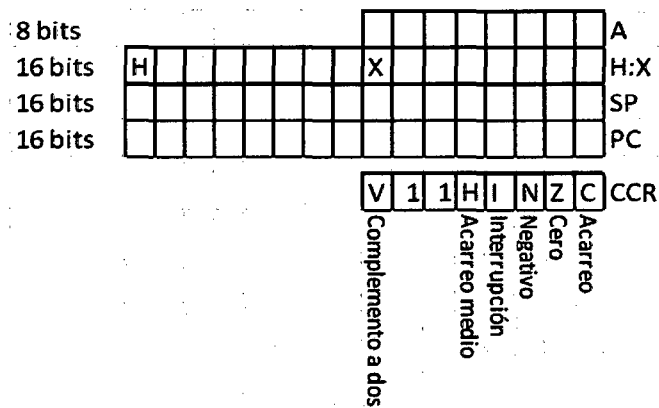


Fig. 4.5: Modelo de Programación Freescale HCS08

### i. Acumulador "A"

Este registro de 8 bits es usado para el almacenamiento temporal de los resultados de las operaciones de la CPU y se localiza a la salida de la ALU.

Usualmente, el acumulador es usado en las operaciones aritméticas como un operando o como el destino del resultado (algunas operaciones pueden usar el registro X).

### ii. Contador de programa "PC"

El PC (Program Counter) es un contador binario que apunta a la dirección de memoria de la siguiente instrucción u operando a ser buscado. Después de cada búsqueda, el contenido del PC es incrementado en uno, de modo que siempre apunte a la siguiente instrucción u operando.

El contenido del PC no puede ser modificado directamente mediante código, a no ser que se use una instrucción de salto.

El contenido del PC también puede ser modificado mediante un hardware de depuración.

Luego del reset de la CPU el PC es inicializado con el valor de 16 bits almacenado en las direcciones 0xFFFFE y 0xFFFF.

### **iii. Registro de direccionamiento “H:X”**

La arquitectura de los HCS08 implementa un registro de direccionamiento de 16 bits llamado “H:X”, este registro está formado por la unión de dos registros de 8 bits H (parte alta) y X (parte baja).

El registro concatenado de 16 bits H:X permite el direccionamiento indexado, el cual accesa todo el mapa de memoria de 64K.

El compilador suele usar el registro H para manipular datos enteros (2 bytes), entregar argumentos a funciones y manejar tablas, arreglos y estructuras (1).

El registro X también puede trabajar como acumulador auxiliar de la ALU.

### **iv. Apuntador de pila “SP”**

La función principal de este registro de 16 bits es la de apuntar a la cima de la pila (stack) del programa, siendo automáticamente incrementado o decrementado por las instrucciones de manejo de la pila.

La pila es una estructura de datos tipo LIFO (last in, first out) y es usada para almacenar las direcciones de retorno en las subrutinas de llamada

**v. Registro de condición de código “CCR”**

El CCR (condition code register) es un registro de 8 bits que almacena las banderas del procesador y el bit de máscara de interrupciones.

|           | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|
| Lectura   | V     | 1     | 1     | H     | I     | N     | Z     | C     |
| Escritura |       |       |       |       |       |       |       |       |
| Reset     | X     | 1     | 1     | x     | 1     | x     | x     | x     |

**a. Bandera de desbordamiento “V”.**

Esta bandera se pone a 1 cuando ocurre un desbordamiento de complemento a 2, lo cual facilita el control de operaciones con signo (1). Luego también se suele decir que indica la ocurrencia de un desbordamiento después de una operación aritmética entre operadores con signo (2).

Esta bandera es puesta a 1 cuando la última operación matemática resulta un un valor mayor a +127 o menor a -128 (2).

**b. Bandera de acarreo medio “H”.**

Bandera de acarreo medio, Indica el evento de acarreo ocurrido del tercer al cuarto bit del resultado de una operación aritmética.

**c. Máscara de interrupción global “I”.**

Este bit controla si una solicitud de interrupción es procesada por la CPU (I = 0) o no (I = 1). Una solicitud de interrupción ocurrida mientras



I = 1, permanece pendiente hasta que I sea puesto a cero (habilitando así el procesamiento de esta interrupción).

d. Bandera de valor negativo "N".

Esta bandera se pone a uno luego de alguna operación aritmética, lógica o movimiento de dato que resulta en un valor negativo (esto implica que el bit 7 del resultado sea 1).

e. Bandera de cero "Z".

Esta bandera se pone a 1 luego de alguna operación aritmética, lógica o movimiento de dato que resulta en un valor nulo (todos los bits del resultado son cero). Si el resultado de es diferente de cero, la bandera Z es puesta a 0.

f. Bandera de acarreo "C".

Esta bandera tiene 4 diferentes usos:

- En operaciones de adición, C es puesto a 1 cuando el resultado es mayor a 255
- En operaciones de sustracción, C es puesto a 1 cuando el resultado es negativo (menor que 0)
- En operaciones de cambio y rotación de bits, C actúa como el noveno bit, recibiendo el valor del octavo bit (bit 7) en las rotaciones por la izquierda o el valor del primer bit (bit 0) en las rotaciones por la derecha.
- En operaciones de verificación de bits, C recibe el valor del bit verificado.

**vi. Modos de direccionamiento y conjunto de instrucciones.**

Existen 16 modos de direccionamiento, pero solo algunos están disponibles para cada instrucción. Por "modos de direccionamiento" entendemos la manera de representar la forma en que la CPU puede llevar los operandos necesarios para lograr ejecutar la orden dada por alguna instrucción (algunas instrucciones requieren solo un parámetro, otras requieren dos).

**vii. Interrupciones.**

Una interrupción es un evento externo que puede causar un cambio en el flujo del programa. El tratamiento de una interrupción se puede pensar como la implementación de una subrutina de la aplicación que se está programando, sin embargo esto no es del todo exacto, la principal diferencia radica en que las interrupciones son llamadas por el hardware mientras el control de las subrutinas comunes se da en la misma aplicación.

Los procesadores HCS08 poseen 32 fuentes de interrupción, cada una con su propio vector de dirección. El vector de dirección es la dirección de la rutina de servicio de interrupción (ISR por interrupt service routine) escrita por el programador. La ISR es automáticamente llamada por el hardware tan pronto como el evento de la interrupción es detectado. Para esto, se debe cumplir:

- La máscara global de interrupciones (I) debe estar en cero
- El bit de habilitación de la interrupción particular debe estar en 1

Una tabla con la lista completa de interrupciones, sus características y número de vector pueden ser consultadas en la página 95 del manual de referencia del MC9S08QE128.

### viii. Latencia de la interrupción.

Se denomina latencia de la interrupción al tiempo transcurrido entre la detección del enveto de la interrupción y la ejecución de la primera instrucción de la ISR.

Este es el tiempo necesario para completar la instrucción que se esté ejecutando en el momento de la detección de la interrupción, reconocer la interrupción, colocar los registros PC, X, A y CCR en el stack (la pila), decodificar la interrupción, buscar y ejecutar la primera instrucción de la ISR.

En los dispositivos HCS08 el tiempo mínimo de latencia está entre 11 y 22 veces el tiempo del ciclo de reloj (BUSCLK).

### ix. Mapas de memoria

La Fig. 4.6 muestra el mapa genérico de memoria de los microcontroladores HCS08.

|                 |                                    |
|-----------------|------------------------------------|
| 0x0000 a 0x007F | Página directa (registros)         |
| 0x0080 a 0x00FF | Página directa (RAM)               |
| 0x0100 a 0x07FF | RAM/FLASH/No implementado          |
| 0x0800 a 0x086F | Página alta (registros)            |
| 0x0870 a 0xFFAF | FLASH/No implementado              |
| 0xFFB0 a 0xFFBF | Registros no volátiles             |
| 0xFFC0 a 0xFFFF | Vectores de reset e interrupciones |

Fig. 4.6: Mapa de memoria genérico para los HCS08 de Freescale.

## x. Sistema de depuración

Los microcontroladores (MCUs) HCS08 también incluyen un sistema de depuración integrado, se puede programar y depurar el MCU conectado al circuito de la aplicación, usando un solo pin.

Este sistema se compone de dos módulos:

### a. Background Debug controller (BDC)

Responsable del acceso interno a los recursos del CPU y el control de la comunicación con la computadora anfitriona.

### b. On-Chip debug system (DBG)

Responsables de característica avanzadas de depuración.

La interface entre el chip y la computadora se realiza a través de un hardware especial llamado BDM que es solamente un conjunto de pines.

Para usar este módulo se debe poner a 0 el bit BKGDPPE del registro SOPT1.

## 4.1.2 Modelos disponibles

Existen una gran cantidad de modelos disponibles en el mercado, en esta ocasión abordaremos exclusivamente al modelo MC9S08QE128, ya que es el microcontrolador usado en este proyecto.

### 4.1.3 Diagrama de Pines

A continuación se muestra el diagrama de pines del MC9S08QE128.

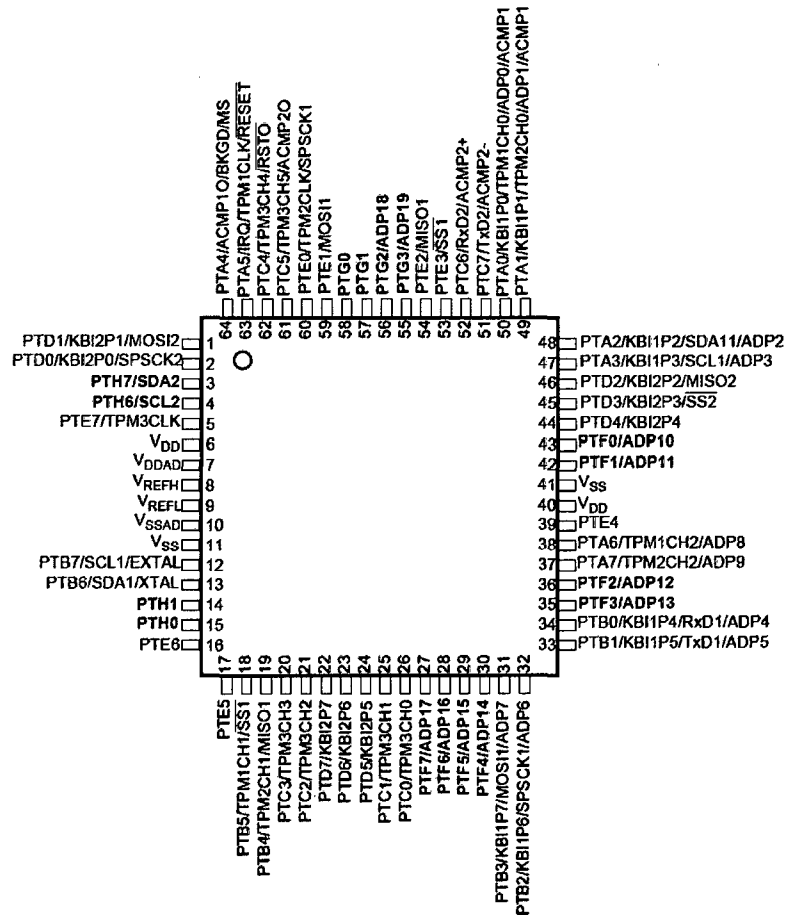


Fig. 4.7: Diagrama de pines del MC9S08QE128 – LQFP

### 4.1.4 Características del microcontrolador MC9S08QE128

#### i. Memoria interna

- Incluye memoria leible y programable.
- Memoria de Acceso aleatorio (RAM)
- Circuitos de seguridad para prevenir el acceso no autorizado a la memoria RAM y al contenido de la memoria flash.

## ii. Registros de entrada y salida

| Registro | Función o descripción  |
|----------|--|
| PTxD     | Para lectura y escritura de los pines del puerto "x"                           |
| PTxDD    | Para controlar la dirección del flujo de datos en el puerto "x"                |
| PTxPE    | Para habilitar (o deshabilitar si se pone a 0) los pull up resistores internos |
| PTxSE    | Para habilitar el control de slew-rate   |
| PTxDS    | Para habilitar el control de drive strength                                    |
| PTxSET   | Para poner los pines a 1   |
| PTxCLR   | Para poner los pines a 0   |
| PTxTOG   | Para invertir el estado del pin (de 0 a 1 ó de 1 a 0)                          |

Nota: x es el puerto que se quiere usar, por ejemplo A, B, C, etc.

Para programar este microcontrolador se usan módulos de desarrollo del mismo fabricante que permite programar y depurar programas que el usuario desee implementar, a continuación se describe este módulo y más adelante el software usado.

#### 4.1.5 Breve Descripción del módulo DEMOQE128 [5]

Este sistema de desarrollo soporta los microcontroladores de Freescale: MC9S08QE128 y MCF51QE128 en empaques 64LQFP, que pueden ser intercambiables. El módulo DEMOQE128 permite programar el microcontrolador vía USB de la PC, así como depurar el programa. Además, el modulo se alimenta usando el bus USB

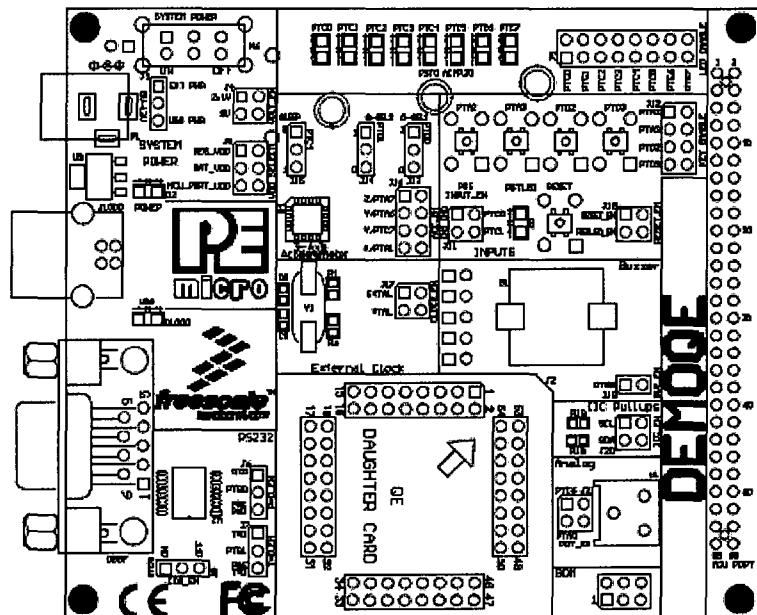


Fig. 4.8: Sistema de desarrollo DEMOQE.

El DEMOQE128 es un sistema de demostración y desarrollo para microcontroladores MC9S08QE128 y MCF51QE128 de la familia Freescale. Los contenidos más importantes del sistema DEMOQE128 son:

- Un analizador lógico de dos canales, que puede ser utilizado para la visualización de datos en tiempo real sobre un PC.
- Un puerto virtual USB conectado a un puerto SCI del MCU QE.

- Un conector asimétrico para la inserción de los MCU's (MC9S08QE128 y MCF51QE128).
- Una interfase embebida P&E MULTILINK, para la programación y depuración de los programas.
- Conector DB9-F /RS-232 Serial Port
- Puerto SCI conectado, vía puentes, a la interfase embebida P&E MULTILINK.
- Interruptor ON/OFF con indicador a LED.
- Conector de fuente externa entre 5Vcd y 8Vcd.
- Selección, vía puentes, del voltaje de alimentación entre las siguientes fuentes:
  - Desde el MULTILINK embebido.
  - Desde fuente externa (ver punto anterior).
  - Desde el conector Mini AB.
  - Desde el conector a puertos I/O.
  - Desde las baterías.
  - También se puede alimentar a través del conector del microcontrolador
- Pulsador de RESET e indicador a LED.
- Accesorios para usuario
  - Acelerómetro de tres ejes con puentes para habilitar cada uno.
  - Ocho LED's de usuario con puentes para habilitar cada uno.
  - Cuatro pulsadores de usuario con puentes para habilitar cada 1.
  - Un parlante piezo-eléctrico con puente para habilitar.



- Un potenciómetro de 10K con puente para habilitar.
  - Puerto IIC con pullups.
- Algunas especificaciones del circuito son:
- Dimensiones: 8.9cms x 10cms.
  - Alimentación:
    - Cable USB: 5VCD @ 500mA máximo.
    - Fuente externa: 5VDC a 8VDC con conector Jack: 2.5/5.5mm con positivo al centro.
- Optional External Crystal Circuitry Layout (not populated)
- Option Jumpers:
- COM\_EN for 1.8V to 4.25V RS232 Transceiver
  - TXD\_EN for SCI\_TXD to Embedded Multilink
  - RXD\_EN for SCI\_RXD to Embedded Multilink
  - INPUT\_EN for two input channels to Embedded Multilink
  - Two AAA Battery Cells

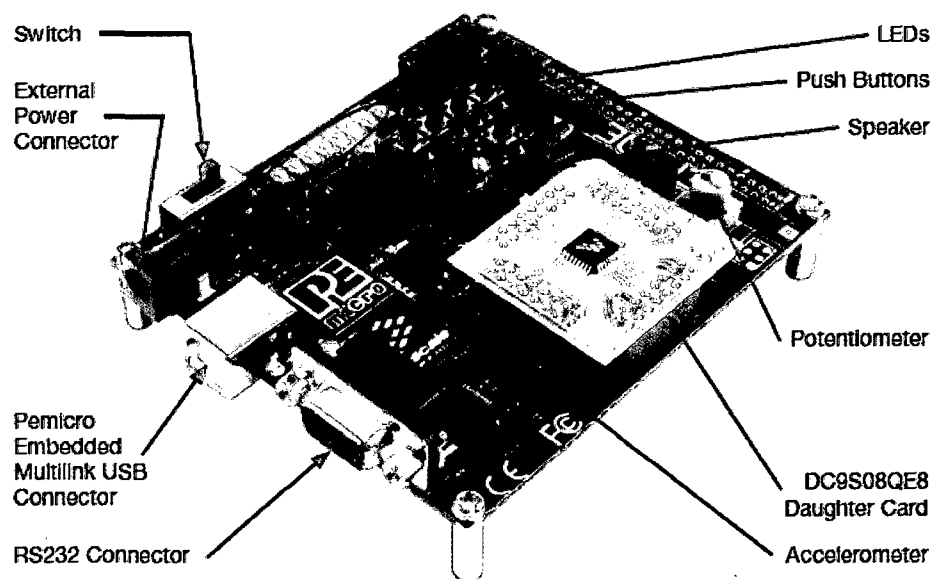


Fig. 4.9: Modulo DEMOQE128 y sus componentes más importantes

Para habilitar o deshabilitar los accesorios para usuario del módulo DEMOQE se tienen puentes (jumpers). En el circuito esquemático de la tarjeta del módulo DEMOQE128 (DEMOQEBSCH.pdf) disponible en [www.freescale.com](http://www.freescale.com) se detallan las funciones de los jumpers. La Tabla 4.1 muestra la ubicación, por defecto, de los puentes:

Tabla 4.1. Establecimiento de puentes por defecto DEMOQE.

| jumper | Installed settings                                |
|--------|---|
| J3     | 2&3   |
| J4     | 3&4   |
| J5     | 1&2   |
| J6     | 2&3   |
| J7     | 2&3   |
| J8     | 2&3   |
| J9     | 1&2, 3&4, 5&6, 7&8, 9&10,<br>11&12, 13&14, 15&16, |
| J11    | 1&2, 3&4  |
| J12    | 1&2, 3&4, 5&6, 7&8                                |
| J13    | 2&3   |
| J14    | 2&3   |
| J15    | 2&3   |
| J16    | 1&2, 3&4, 7&8                                     |
| J18    | 1&2, 3&4  |
| J19    | 1&2   |
| J20    | 1&2, 3&4  |
| J21    | 1&2, 3&4  |

La Fig. 4.10: muestra la distribución de pines (PINOUT) del conector MCU PORT en el circuito impreso del DEMOQE.

| MCU_PORT_VDD | J1 |   |                                 |
|--------------|----|---|---------------------------------|
|              | 1  | VDD   | PTA5/IRQ/TPMICLK/RESET          |
|              | 3  | VSS   | PTA5/IRQ/TPMICLK/RESET          |
| PTB1         | 5  | PTB1/KBI1P5/TxD1/ADP5                                     | PTA4/ACMP10/BKGD/MS             |
| PTB0         | 7  | PTB0/KBI1P4/RxD1/ADP4                                     | PTE7/TMP3CLK (n/c for 32 LQFP)  |
| PTA2         | 9  | PTA2/KBI1P2/SDA1/ADP2                                     | VREFH                           |
| PTA3         | 11 | PTA3/KBI1P3/SCL1/ADP3                                     | VREFL                           |
| PTC0         | 13 | PTC0/TPM3CH0  | PTA0/KBI1P0/TPM1CH0/ADP0/ACMP1+ |
| PTC1         | 15 | PTC1/TPM3CH1  | PTA1/KBI1P1/TPM2CH0/ADP1/ACMP1- |
| PTB3         | 17 | PTB3/KBI1P7/MOSI1/ADP7                                    | PTF0/ADP10 (n/c for 32 LQFP)    |
| PTB4         | 19 | PTB4/TPM2CH1/MISO1  | PTF1/ADP11 (n/c for 32 LQFP)    |
| PTB2         | 21 | PTB2/KBI1P6/SPSCK1/ADP6                                   | PTA6/TPM1CH2/ADP8               |
| PTB5         | 23 | PTB5/TPM1CH1/SS1  | PTA7/TPM2CH2/ADP9               |
| PTD1         | 25 | PTD1/KBI2P1/MOSI2/PTH6/SCL2 (PTB7/SCL1/EXTAL for 32 LQFP) | SDA                             |
| PTD2         | 27 | PTD2/KBI2P2/MISO2 PTH7/SDA2 (PTB6/SDA1/XTAL for 32 LQFP)  | PTD4                            |
| PTD0         | 29 | PTD0/KBI2P0/SPSCK2  | PTD5/KBI2P5 (n/c for 32 LQFP)   |
| PTD3         | 31 | PTD3/KBI2P3/SS2   | PTD6/KBI2P6 (n/c for 32 LQFP)   |
| PTC2         | 33 | PTC2/TPM3CH2  | PTD7/KBI2P7 (n/c for 32 LQFP)   |
| PTC3         | 35 | PTC3/TPM3CH3  | PTC7/TxD2/ACMP2-                |
| PTC4         | 37 | PTC4/TPM3CH4/RSTO   | PTC6/RxD2/ACMP2+                |
| PTC5         | 39 | PTC5/TPM3CH5/ACMP2O                                       | PTB7/SCL1/EXTAL                 |
| PTF2         | 41 | PTF2/ADP12  | PTB6/SDA1/XTAL                  |
| PTF3         | 43 | PTF3/ADP13  | PTG0                            |
| PTF4         | 45 | PTF4/ADP14  | PTG1                            |
| PTF5         | 47 | PTF5/ADP15  | PTH0                            |
| PTF6         | 49 | PTF6/ADP16  | PTH1                            |
| PTF7         | 51 | PTF7/ADP17  | PTE6                            |
| PTG2         | 53 | PTG2/ADP18  | NC                              |
| PTG3         | 55 | PTG3/ADP19  |                                 |

Fig. 4.10: Conector MCU PORT

## 4.2 MÓDULO TPM DEL MICROCONTROLADOR MC9S08QE128

Este módulo es muy importante para este proyecto ya que se usa el modulo en tres de las 4 funciones que tiene y que se describen más adelante.

### 4.2.1 Definición de registros del módulo TPM

Esta sección contiene la definición de registros en orden de dirección. La forma para hacer que el módulo TPM (Timer), trabaje en la forma que uno desee es configurando estos registros relacionados al módulo seleccionado.

#### 4.2.1.1 Registro de estado y control (TPMxSC):

Este registro contiene el flag de estado de overflow y bits de control usados para configurar la habitación de la interrupción, TPM

configuración, fuente de reloj, y factor de prescaler. Esos controles se refieren a todos los canales dentro de este módulo.

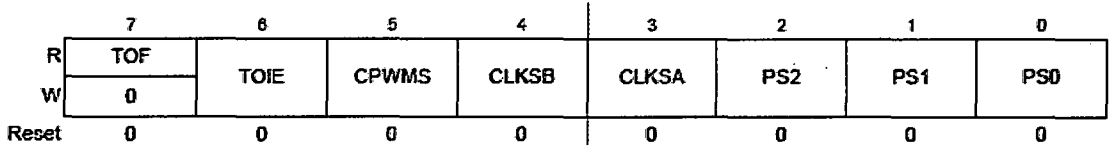


Fig. 4.11: Registro de Estado y Control (TPMxSC)

Tabla 4.2: Descripción de bits del Registro de Estado y Control (TPMxSC)

| bit                  | Descripción   |                    |                     |                    |                     |                    |                     |                    |                      |
|----------------------|---|--------------------|---------------------|--------------------|---------------------|--------------------|---------------------|--------------------|----------------------|
| 7<br>TOF             | <p><b>Timer overflow flag</b> (Bandera de sobreflujo del contador de 16 bits). Esta bandera se pone a "1" cuando se ha alcanzado el valor de 0x0000 una vez ha sido superado el valor programado en el registro de módulo del contador.</p> <p>Para aclarar el bit TOF es necesario leer el registro TPMSC y luego escribir un "0" en el bit TOF.</p> <p>0: El contador del TPM no ha alcanzado el sobreflujo<br/>1: El contador ha alcanzado un sobreflujo</p> |                    |                     |                    |                     |                    |                     |                    |                      |
| 6<br>TOIE            | <p><b>Timer overflow interrupt enable</b>. TOIE="1", el sistema genera un evento de interrupción por sobreflujo del contador del TPM.</p> <p>0: Para detectar un evento de sobreflujo es necesario hacer polling sobre TOF<br/>1: Habilita un evento de interrupción cuando TOF = "1"</p>   |                    |                     |                    |                     |                    |                     |                    |                      |
| 5<br>CPWMS           | <p><b>Center-aligned PWM select</b>. Habilita que todos los canales del TPM actúen como PWM alineado en el centro del período (center align). El objetivo es disminuir ruido en las conmutaciones del pin de salida y el contador trabaja en modo up/down.</p> <p>0: No está activa la opción de alineado al centrado<br/>1: Activa opción de PWM alineado al centro</p>  |                    |                     |                    |                     |                    |                     |                    |                      |
| 4-3<br>CLKS<br>[B:A] | <p><b>Clock source selects</b>. Selecciona la fuente de reloj del contador del TPM.</p> <p>00: Módulo inactivo<br/>01: Reloj del bus interno<br/>10: Reloj fijo del sistema (sólo para opción con circuito PLL)<br/>11: Reloj externo</p>   |                    |                     |                    |                     |                    |                     |                    |                      |
| 2-0<br>PS[2:0]       | <p><b>Prescalerfactor select</b>. Selección del divisor de la fuente de reloj.</p> <table> <tr> <td>000: Divisor por 1</td> <td>100: Divisor por 16</td> </tr> <tr> <td>001: Divisor por 2</td> <td>101: Divisor por 32</td> </tr> <tr> <td>010: Divisor por 4</td> <td>110: Divisor por 64</td> </tr> <tr> <td>011: Divisor por 8</td> <td>111: Divisor por 128</td> </tr> </table>  | 000: Divisor por 1 | 100: Divisor por 16 | 001: Divisor por 2 | 101: Divisor por 32 | 010: Divisor por 4 | 110: Divisor por 64 | 011: Divisor por 8 | 111: Divisor por 128 |
| 000: Divisor por 1   | 100: Divisor por 16   |                    |                     |                    |                     |                    |                     |                    |                      |
| 001: Divisor por 2   | 101: Divisor por 32   |                    |                     |                    |                     |                    |                     |                    |                      |
| 010: Divisor por 4   | 110: Divisor por 64   |                    |                     |                    |                     |                    |                     |                    |                      |
| 011: Divisor por 8   | 111: Divisor por 128  |                    |                     |                    |                     |                    |                     |                    |                      |

#### 4.2.1.2 Registro contador del TPM (TPMxCNTH:TPMxCNTL)

Los dos registros contador de solo lectura contienen la parte alta y la parte baja del valor en el TPM contador. Las dos siguientes figuras muestran los registros que conforman el contador de 16 bits del TPM. La acción de escribir en cualquiera de los dos registros, hace que se aclare el contador de 16 bits.

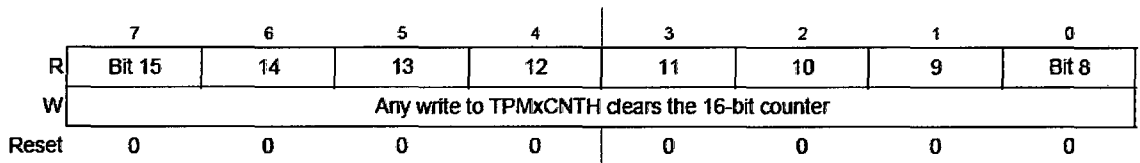


Fig. 4.12: Registro contador alto del TPM (TPMxCNTH).

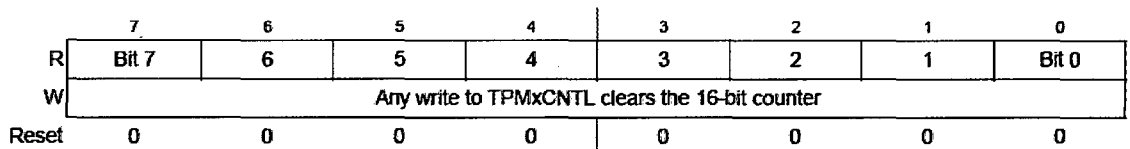


Fig. 4.13: Registro contador bajo del TPM (TPMxCNTL).

#### 4.2.1.3 Registro módulo contador del TPM (TPMxMODH:TPMxMODL)

Registro módulo del TPM (TPMxMODH:TPMxMODL): Está configurado por dos registros de 8 bits. Las Figuras 4.14 y 4.15 muestran los registros que conforman el módulo de 16 bits del TPM. El módulo es el valor hasta donde debe contar el contador del TPM. Una vez que el contador haya alcanzado el valor programado en el módulo, este se fija en ceros (0x0000) y la bandera TOF se pone a "1".

La escritura de un nuevo valor en el registro módulo obedece al valor de los bits CLKS del registro TPMSC y su comportamiento es:

CLKS = "00": El módulo es actualizado cuando el segundo byte es escrito.

CLKS  $\neq$  "00": El módulo cambia cuando ambos registros hayan sido escritos y el contador pasa del valor TPMxMODH:TPMxMODL - 1, al valor TPMxMODH:TPMxMODL. Si el contador está en modo freerunning, el módulo cambia cuando el contador pasa de 0xFFFE a 0xFFFF.

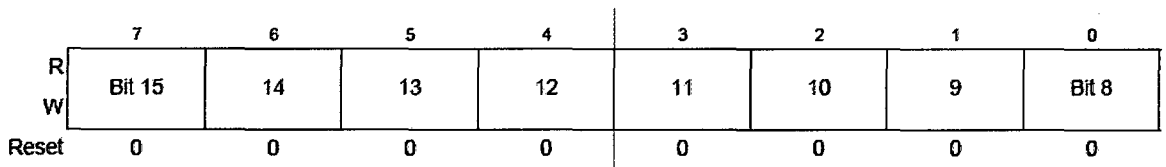


Fig. 4.14: TPM Counter Modulo Register High (TPMxMODH)

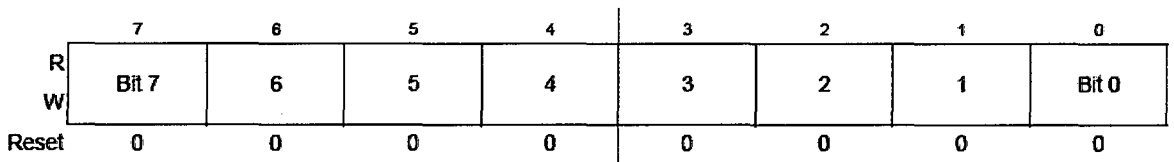


Fig. 4.15: TPM Counter Modulo Register Low (TPMxMODL)

#### 4.2.1.4 Registro de estado y control de canal (TPMxCnSC)

Este registro contiene los flags de estados de interrupción de los diferentes canales del módulo y los bits de control usados para configurar la habilitación de interrupción, la configuración de canal y las funciones de los pines.

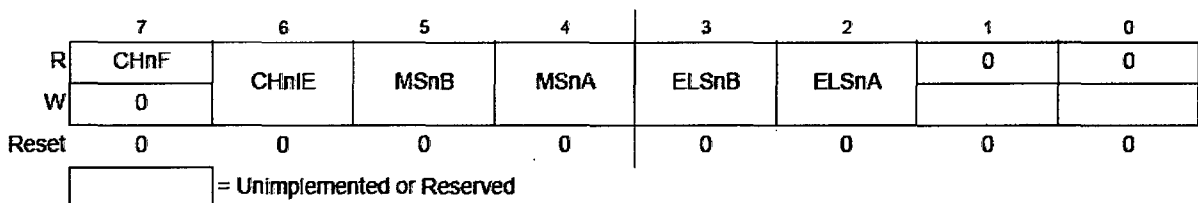


Fig. 4.16: Registro de estado y control de canal (TPMxCnSC)

Tabla 4.3: Descripción de bits del Registro de estado y control de canal (TPMxCnSC)

| bit                   | Descripción   |
|-----------------------|---|
| 7<br>CHnF             | <p>En el modo de INPUT CAPTURE, esta bandera se pone a "1" cuando el evento de flanco se presenta. En los modos OUTPUT COMPARE o PWM, esta bandera se pone a "1" cuando el valor del contador iguala al valor programado en el registro del canal. Para aclarar esta bandera durante un evento de interrupción, es necesario leer primero el estado del registro TPMxCnSC y luego escribir un cero en la bandera CHnF.</p> <p>0: No ha ocurrido de INPUT CAPTURE, PWM u OUTPUT COMPARE.<br/>1: Ha ocurrido un evento de INPUT CAPTURE, PWM u OUTPUT COMPARE en el canal</p> |
| 6<br>CHnIE            | <p>- CHnIE: Bit para habilitar un evento de interrupción. Cuando la bandera CHnF es "1" y el bit CHnIE="1", el sistema genera un evento de interrupción por canal.</p> <p>0: Para detectar un evento de canal es necesario hacer polling sobre CHnF<br/>1: Habilita un evento de interrupción cuando CHnF = "1"</p>   |
| 5<br>MSnB             | <p>Mode select B for TPM channel n. cuando CPWMS=0, MSnB=1 configura TPM canal para modo PWM alineado al flanco. Como se muestra en la tabla 16-6.</p>  |
| 4<br>MSnA             | <p>Mode select A for TPM channel n. Cuando CPWMS=0 y MSnB=0, MSnA configura TPM canal para modo INPUT-CAPTURE. Como se muestra en la tabla 16-6.</p>  |
| 3-2<br>ELSnB<br>ELSnA | <p>Flanco/nivel selección de bits. Dependiendo del modo de operación para el timer del canal como el seteo de los bits CPWMS:MSnB:MSnA y mostrado en la tabla 4.4 , esos bits seleccionan la polaridad del flanco de entrada que activa un evento de input capture , o selecciona la polaridad de la salida PWM,</p> <p>Seteando ELSnB:ELSnA a 0:0 configura el referido pin del timer como un pin de propósito general I/O no relacionado a ninguna función del timer.</p>   |

Tabla 4.4: Modo, Flanco, y selección de nivel del TPM

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Modo                                      | Configuración                             |
|-------|-----------|-------------|---|---|
| X     | XX        | 00          | Modulo TPM deshabilitado                  |   |
| 0     | 00        | 01          | INPUT CAPTURE                             | Capture en el flanco de subida            |
|       |           | 10          |   | Capture en el flanco de bajada            |
|       |           | 11          |   | Capture en el flanco de subida o bajada   |
|       | 01        | 01          | OUTPUT CAPTURE                            | Cambie estado pin en OUTPUT COMPARE       |
|       |           | 10          |   | Ponga pin en cero en OUTPUT COMPARE       |
|       |           | 11          |   | Ponga pin en uno en OUTPUT COMPARE        |
|       | 1X        | 10          | PWM ALINEADO AL FLANCO                    | Comienza en alto y cae en OUTPUT COMPARE  |
|       |           | X1          |   | Comienza en bajo y sube en OUTPUT COMPARE |
|       | 1         | XX          | 10  | PWM ALINEADO AL CENTRO                    |
| X1    |           |             | Comienza en bajo y sube en OUTPUT COMPARE |   |

4.2.1.5 Registro de valor del canal (TPMxCnVH:TPMxCnVL)

Registro de valor del canal (TPMxCnVH:TPMxCnVL): Está configurado por dos registros de 8 bits.

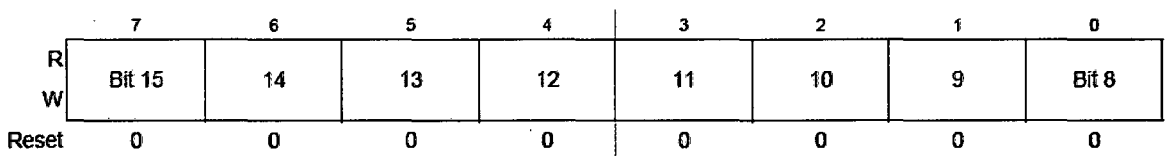


Figure 4.17: TPM Channel Value Register High (TPMxCnVH)

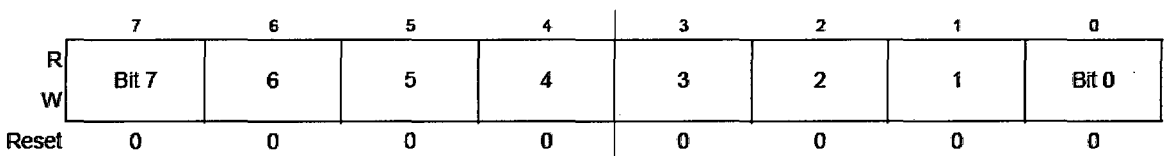


Figure 4.18: TPM Channel Value Register Low (TPMxCnVL)



### 4.3 MÓDULO TPM COMO TEMPORIZADOR DE PROPÓSITO GENERAL

Como se había explicado en aparte anterior, la temporización general es un modo de funcionamiento del TPM y no involucra a los canales ni a los pines del sistema.

Basta con habilitar el TPM en alguno de sus cuatro modos y utilizar la bandera de sobreflujo (TOF), para dar cuenta de una temporización lograda. La Figura 4.19 ilustra sobre las partes involucradas en una temporización de propósito general.

En el primer paso (A), es necesario elegir alguno de los modos de operación del TPM, ubicando un valor diferente a "00" en los bits ELS0A y ELS0B. A continuación se realiza la selección de la fuente de reloj del TPM (B), teniendo en cuenta que la opción de RELOJ FIJO depende del circuito PLL y sus modos.

Seguido (C), es necesario seleccionar un divisor del reloj del TPM, de acuerdo al fenómeno de temporización que se requiera. Lo anterior establece la base de tiempo del contador de 16 bits. Luego (D), se establece el módulo de conteo, que se calcula como:  $M = \text{Tiempo} / \text{Base de tiempo}$ . A continuación, se habilita el evento de interrupción (E) o si se prefiere se hace un polling sobre la bandera TOF. Finalmente, aclarar el contador escribiendo cualquier valor en el registro del contador (F).

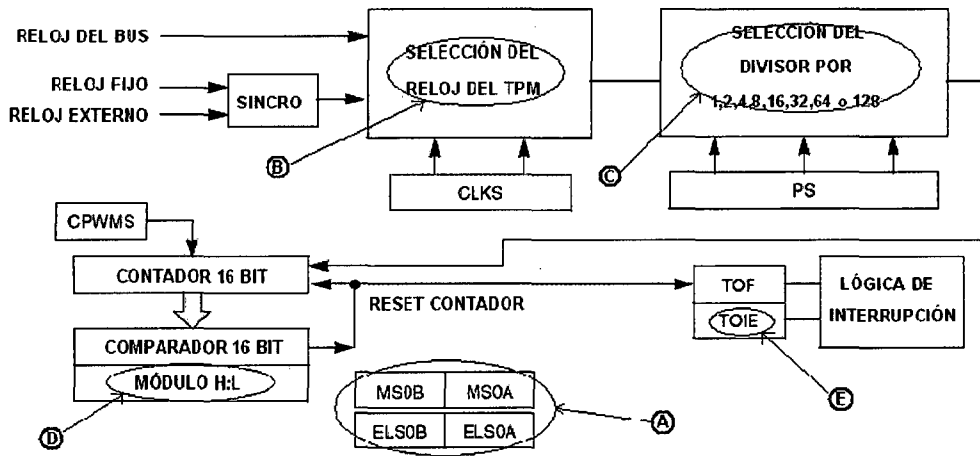


Fig. 4.19: Temporización de propósito general.

A continuación se configura el módulo TPM1 como temporizador de propósito general para que mida un período de muestreo de la velocidad deseada así como el tiempo en el cual se actualiza el duty cycle del PWM. De acuerdo a las simulaciones y pruebas se tiene que un tiempo de 50 mseg es apropiado para el tipo de motor Dc con el que se está trabajando.

$$T_{\text{periodo}} = \frac{\text{Prescaler} * (\text{MODULO} + 1)}{\text{frecuencia de reloj}} \quad (\text{Ec. 4.1})$$

Luego se tiene: 
$$\text{MODULO} = \frac{T_{\text{periodo}} * \text{Frecuencia de reloj}}{\text{prescaler}} - 1 \quad (\text{Ec. 4.2})$$

Al reemplazar:

$$\text{Prescaler} = 128$$

$$\text{Frecuencia de reloj} = 4 \text{ Mhz}$$

$$T_{\text{periodo}} = 50 \text{ mseg}$$

$$\text{MODULO} = \frac{50 * 10^{-3} * 4 * 10^6}{128} - 1 = 1561$$

Configuración de los registros TPM3SC, involucrados en la interface del módulo de teclado con el microcontrolador.

|            |           |            |             |             |             |           |           |           |
|------------|-----------|------------|-------------|-------------|-------------|-----------|-----------|-----------|
| TPM1S<br>C | TOF=<br>1 | TOIE=<br>0 | CPWMS=<br>0 | CLKSB=<br>0 | CLKSA=<br>1 | PS2=<br>1 | PS1=<br>1 | PS0=<br>1 |
|------------|-----------|------------|-------------|-------------|-------------|-----------|-----------|-----------|

**TOIE[6]** = 1 Se habilita la interrupción por overflow

**CPWMS[5]** = 0 Porque el PWM no estará centrado, sino al alineado al flanco.

**CLKSB:CLKSA[4:3]** = 01 Se selecciona reloj de bus interno.

**PS2:PS1[2:0]** = 111 para tener un prescaler de 128 .

Inicialización del módulo TPM1 para módulo PWM

```
/**/
```

```
/* TPM1_Init
```

```
/**/
```

```
void TPM_Init(void){
```

```
// Configuración de la interrupción por TIMER 1
```

```
/*
```

```
temporizado de 50ms
```

```
PRESCALER * (MODULO + 1)
```

```
TPM_T = -----
```

```
CLOCK
```

```
128 * ( MOD + 1 )
```

```
50 ms = -----
```

```
4 MHz
```

```
*/
```

```
TPM1MOD = 1561; //0x0000; //calcular para conseguir
```

```
TPM1SC = TPM_DIV128 | TPM_BUSCLK | bTOIE;
```

```
TPM1SC_TOF = 0; // borrar el flag de interrupción
```

```
}
```

Rutina de atención a la interrupción por overflow del módulo TPM1.

```
//*****
//***** atencion al vector de Interrupciont *****
//*****
void interrupt VectorNumber_Vtpm1ovf Tpm1_Ovf_Isr(void){
    TPM1SC_TOF = 0;
}
}
```

#### 4.4 MÓDULO TPM COMO PWM

El módulo de generación de Modulación por Ancho de pulso (PWM) es un recurso muy utilizado para controlar motores de corriente continua. A pesar de la salida que es una señal digital que toma valores o niveles lógicos alto (uno) o bajo (cero) se puede generar una señal analógica. La señal generada que es una onda cuadrada, con frecuencia constante y ancho de pulso variable. Estos conceptos están directamente relacionados con período fijo y ciclo activo (duty cycle) respectivamente.

El duty cycle está definida como el porcentaje de tiempo que la señal esta

en nivel lógico alto en un período fijo: 
$$duty\ cycle = \frac{DUTY}{PERIODO} \times 100\%$$

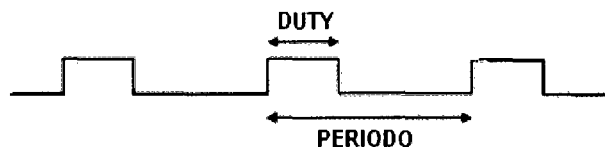


Fig. 4.20: Duty cycle en una onda cuadrada.

Es decir, cuando se tiene un duty cycle de 100%, tenemos un nivel lógico alto por todo el período. Un duty cycle de 50% define la mitad del período en un nivel lógico alto y la otra mitad en nivel lógico bajo. Si la salida es utilizada como una señal TTL, la tensión media de salida en un duty cycle de 50% será 2,5V. Estos conceptos son mostrados en la figura de abajo.

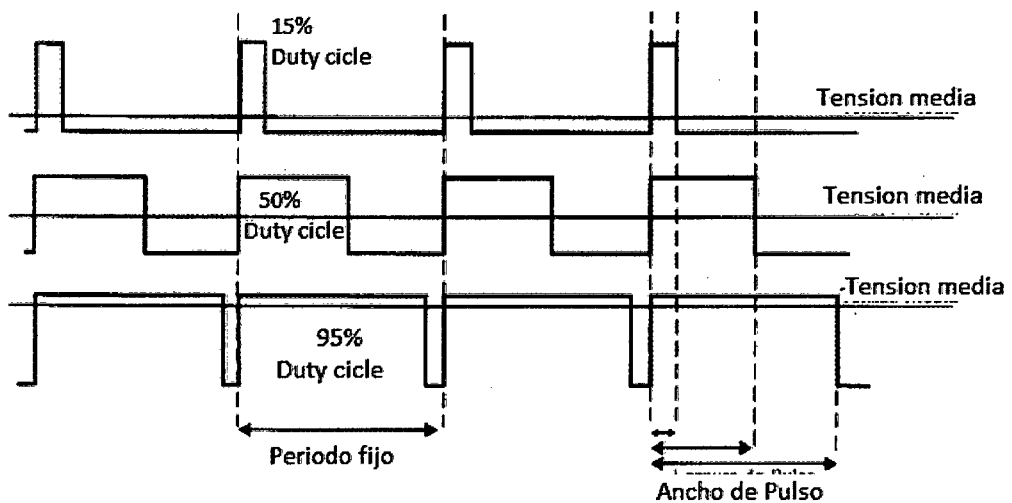


Fig. 4.21: Señales moduladas por ancho de pulso

Se debe mencionar que el PWM no siempre posee un estado inicial positivo, pudiendo iniciar el período con nivel lógico bajo.

La base de tiempo de los módulos PWM normalmente es implementada de dos formas. Una de estas formas es utilizando el propio temporizador como base de tiempo de PWM, es decir, si el temporizador está configurado para un período de 1 mseg, la frecuencia del PWM será 1 KHz. La otra forma es utilizando un temporizador específico para PWM, que debe ser configurado para la frecuencia deseada. Además un temporizador puede ser utilizado como base de tiempo de varias salidas PWM, es decir, varias PWM con una misma frecuencia, pero los anchos de pulsos diferentes.

La Figura 4.22 ilustra sobre la generación de una señal PWM, aplicada a un pin de la máquina

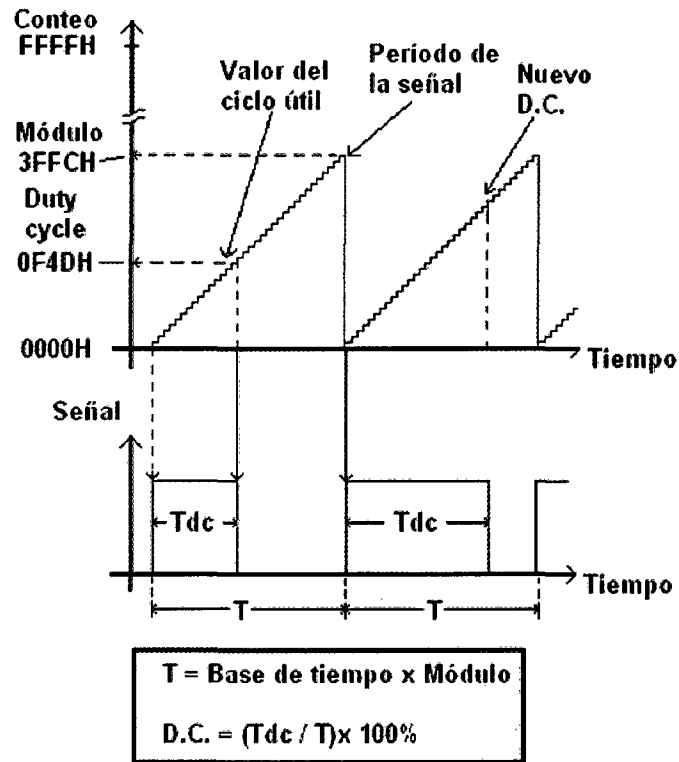


Fig. 4.22: Generación de una señal PWM en un pin

A continuación se presentan las configuraciones de los registros para la implementación del módulo PWM alineado al flanco en el microcontrolador MC9S08QE128, así como los registros necesarios para la configuración de los canales de PWM, de manera que se pueda implementar el ejemplo mostrado anteriormente.

#### 4.4.1 Funcionamiento del TPM como PWM (Pulse Width Modulation)

La Figura 4.23 muestra las partes involucradas en un evento de salida de PWM.

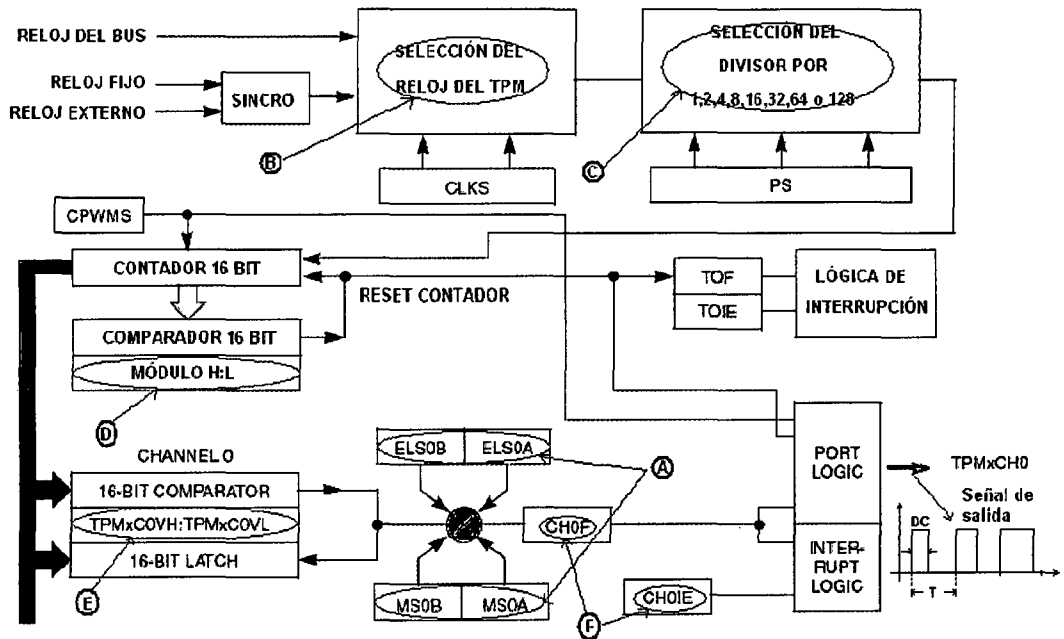


Fig. 4.23: Generación de señal PWM.

Para la generación de una señal periódica tipo PWM presentada en algún pin  $TPMxCHn$ , el primer paso es definir el modo de funcionamiento como PWM en los bits marcados como (A) (ver Tabla 4.4). El modo PWM tiene la opción de trabajar iniciando con un flanco de subida o terminando con un flanco de bajada, desde el punto de vista del ciclo de trabajo (DC: Duty Cycle); o como alineado al centro del período T. Esta última característica es muy popular para el control de servo motores de 3 fases en CA y sin escobillas (brushless) en CD, en donde son necesarios varios canales de PWM. A continuación se elige el reloj apropiado según el rango de frecuencias de la señal a generar (B). Seguido (C), es necesario especificar un divisor para el reloj elegido. Luego se programa el período (T) de la señal en el módulo del contador de 16 bits (D) y el ciclo de trabajo (DC) de la señal en el registro del canal (E).

Cada que sea necesario actualizar el valor del ciclo de trabajo, el usuario deberá hacer un polling o generar un evento de interrupción, utilizando la bandera CHxIF y/o habilitando el mecanismo de interrupción con el bit CHxIE (F).

Si la opción elegida es la de atender un evento de interrupción por canal (OUTPUT COMPARE), se recomienda actualizar los valores del nuevo DC en la atención a dicha interrupción.

Para la opción de PWM alineado al centro y pulso a alto, el contador de 16 bits trabaja en modo up/down. En la Figura 4.24 se ilustra de manera temporal la generación de una señal PWM alineada al centro y pulso a alto, con una programación de 20 conteos en el módulo (T) y un ciclo de trabajo (DC) de 14 conteos. El contador inicia en un conteo descendiente y el pin de salida se mantiene en bajo, cuando el contador llega al valor programado en el canal el pin sube (A). El contador se decrementa hasta adquirir un conteo ascendente y cuando llega al valor de conteo programado en el canal, el pin de salida vuelve a caer (B). Este esquema se repite hasta tanto no se detenga el PWM o se cambien los valores de su período y ciclo de trabajo.

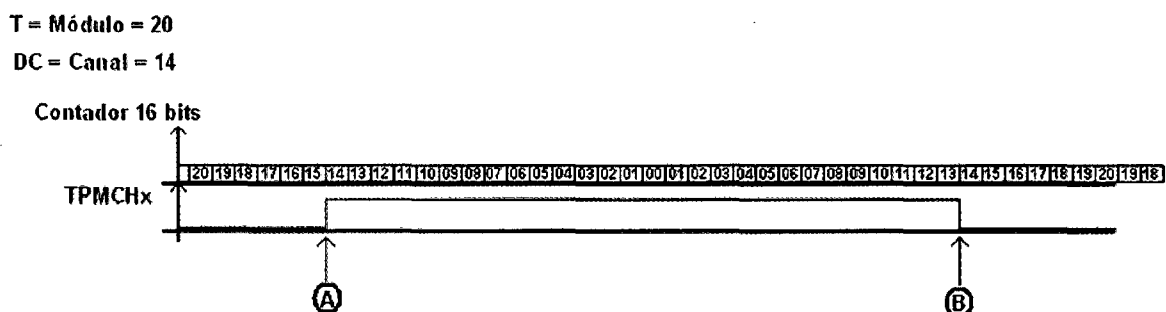


Fig. 4.24: PWM alineado al centro y pulso a alto.



El período (T) de la señal de salida del PWM se calcula así:

Base de tiempo =  $1 / (\text{Frecuencia Reloj Elegido} / \text{Divisor})$

$T = \text{Base de tiempo} \times \text{TPMxMODH} : \text{TPMxMODL}$

$DC = \text{Base de Tiempo} \times \text{TPMxCnVH} : \text{TPMxCnVL}$

El circuito de la Figura 4.25 muestra el sistema empleado para este proyecto, así como el pin de salida de la señal PWM en modo alineado con el flanco. En la parte externa se usa un driver de motor L293D que se describe más adelante.

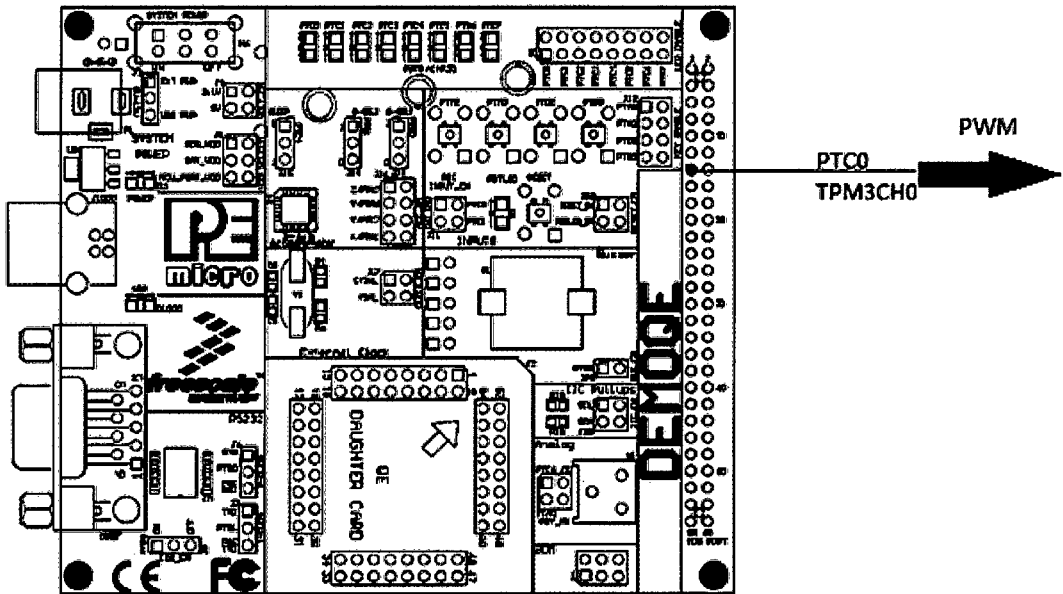


Fig. 4.25: Circuito PWM para control de motor.

A continuación se describe las configuraciones implementadas así como el código necesario en lenguaje C.

#### 4.4.2 PWM alineado al flanco

Inicialización de los registros correspondientes al módulo TPM3 canal 0.

Se requiere un período de PWM=2.5mseg, por lo cual realizamos los cálculos siguientes:

Base de tiempo =  $1 / (\text{Frecuencia Reloj Elegido} / \text{Divisor})$

$T = \text{Base de tiempo} \times \text{TPMxMODH} : \text{TPMxMODL}$

$DC = \text{Base de Tiempo} \times \text{TPMxCnVH} : \text{TPMxCnVL}$

$$\text{Base de tiempo} = \frac{1}{\text{Frecuencia Reloj} / \text{Prescaler}}$$

$$T_{\text{período PWM}} = \text{Base de tiempo} * \text{TPMxMODH} : \text{TPMxMODL}$$

Luego se tiene:  $TPMOD = \frac{T * \text{Frecuencia de reloj}}{\text{prescaler}}$

Al reemplazar:

Prescaler =16

Frecuencia de reloj=4Mhz

$$T_{\text{período PWM}} = 2.5\text{mseg}$$

$$TPMOD = \frac{2.5 * 10^{-3} * 4 * 10^6}{16} = \frac{2500 * 4}{16} = \frac{10000}{16} = 625$$

Configuración de los registros TPM3SC, involucrados en la interface del módulo de teclado con el microcontrolador.

|        |       |        |         |         |         |       |       |       |
|--------|-------|--------|---------|---------|---------|-------|-------|-------|
| TPM3SC | TOF=0 | TOIE=0 | CPWMS=0 | CLKSB=0 | CLKSA=1 | PS2=1 | PS1=0 | PS0=0 |
|--------|-------|--------|---------|---------|---------|-------|-------|-------|

**TOIE[6]** =0 No se habilita la interrupción por overflow

**CPWMS[5]** =0 Porque el PWM no estará centrado, sino al alineado al flanco.

**CLKSB:CLKSA[4:3]** =01 Se selecciona reloj de bus interno.

**PS2:PS1[2:0]** =100 para tener un prescaler de 16 .

|          |        |         |        |        |         |         |   |   |
|----------|--------|---------|--------|--------|---------|---------|---|---|
| TPM3C0SC | CHnF=0 | CHnIE=1 | MSnB=1 | MSnA=1 | ELSnB=1 | ELSnA=0 | 0 | 0 |
|----------|--------|---------|--------|--------|---------|---------|---|---|

**CHnIE[6]** Se pone a 1 para habilitar la interrupción por canal 0 del módulo TPM3, para actualizar el valor del duty cycle.

**MSnB:MSnA[5:4]** =11 PWM alineado al flanco.

**ELSnB:ELSnA[3:2]** =10 El pulso de salida inicia en nivel alto.

Inicialización del módulo TPM3 para módulo PWM

```

//*****
//* *****TPM3_Init*****
void TPM3_Init(void){
    // TOF TOIE CPWMS CLKSB CLKSA PS2 PS1 PS0
    //TOF:No se puede escribir
    //TOIE:0 PARA INHABILITAR INTERRUPCION
    //CPWMS :0 PARA que no se alinie al centro el PWM
    //CLKSB CLKSA: 01 Reloj bus interno
    //PS2 PS1 PS0:_100_PARA Q CUENTE CON ESCALA DE 16
    TPM3SC=0b00001100; //original

```

```

// CHnF CHnIE MSnB MSnA ELSnB ELSnA 0 0
//CHnF=0 DEPORQ NO SE PUEDE ESCRIBIR
//CHnIE:1 Para habilitar la Interrupcion
//MSnB MSnA :00 PARA INPUT CAPTURE
//CLKSB CLKSA: 11 PARA Q SEA EXTERNO
//ELSnB ELSnA:_01_PARA CAMBIOS DE 1 A 0
//VOY A TRABAJR CON EL CHANEL 0 DEL TPM3
TPM3C0SC= 0b01111000; //PWM edge align era 0x78 originalmente
/* temporizado de 2.5 mseg

          PRESCALER * MODULO
TPM_T = -----
          CLOCK

          16 * MODULO
2.5 ms = -----
          4 MHz          */
TPM3MOD=625; // Valor del período del PWM
TPM3C0V=60;//equivale al 10% como valor por defecto
}

Rutina de atención a la interrupción del canal 0 del módulo TPM3
//*****
/* *****VectorNumber_Vtpm3ch0 Tpm3_ch0_isr()*****
void interrupt VectorNumber_Vtpm3ch0 Tpm3_ch0_isr(void){
    TPM3C0SC;
    TPM3C0SC_CH0F=0; //borrar el flag de interrupcion;
    TPM3C0V=Tpm3_Get_new_DC();//Actualiza el duty cycle
}

```

## 4.5 EL ENCODER Y EL MÓDULO TPM COMO INPUT CAPTURE

### 4.5.1 El Encoder

Generalmente se basan en colocar en el eje del motor un disco con ranuras estrechas, este disco además se sitúa entre un emisor y un receptor de infrarrojos. De tal forma que al girar el motor las ranuras del disco dejan pasar o interrumpen alternativamente el haz infrarrojo. A la salida del receptor de infrarrojos se tiene una señal cuadrada, contando las crestas se puede saber cuántas ranuras han pasado y en consecuencia que ángulo se ha girado. En nuestro caso lo que se hace es contar cuantas ranuras conto en determinado tiempo y eso nos da la velocidad angular del motor DC.

En la siguiente figura se observa una vista del encoder utilizado en el proyecto que tiene una resolución de 100 pulsos por vuelta.

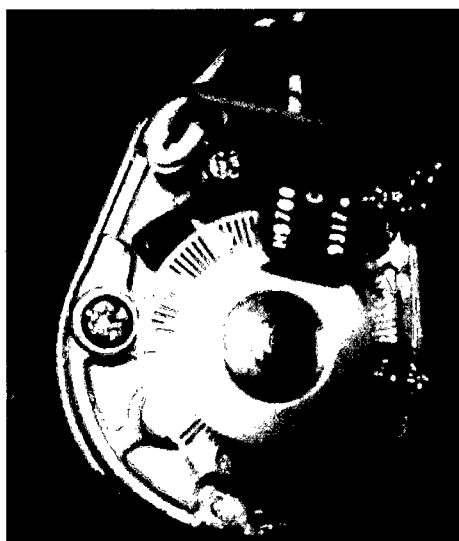


Fig.4.26: Vista del encoder acoplado al eje del motor DC

#### 4.5.1.1 Sensor Foto-interruptores de barrera

Están formados por un emisor de infrarrojos y un fototransistor separados por una abertura donde se insertará el disco que producirá un corte del haz. La salida será 0 o 1.

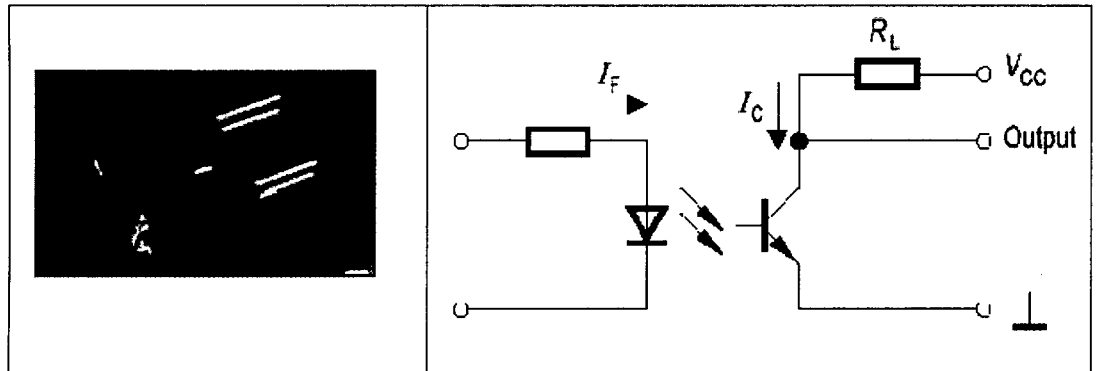


Fig. 4.27 Circuito eléctrico del sensor Foto – interruptor de barrera.

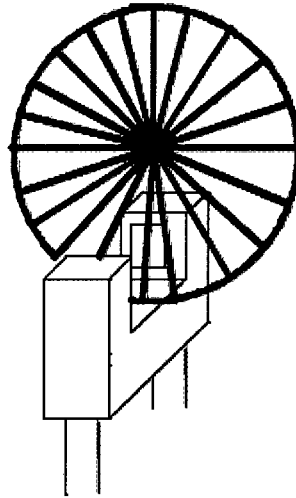


Fig. 4.28 Acoplamiento del encoder con el sensor.

#### 4.5.1.2 Encoders ópticos

Con los foto-interruptores se pueden montar los encoders ópticos, formados por un interruptores se pueden montar los encoders ópticos, formados por un disco que tiene dibujados segmentos para ser detectados

por los sensores. Existen dos tipos de encoders, los encoders Incrementales y encoders Absolutos.

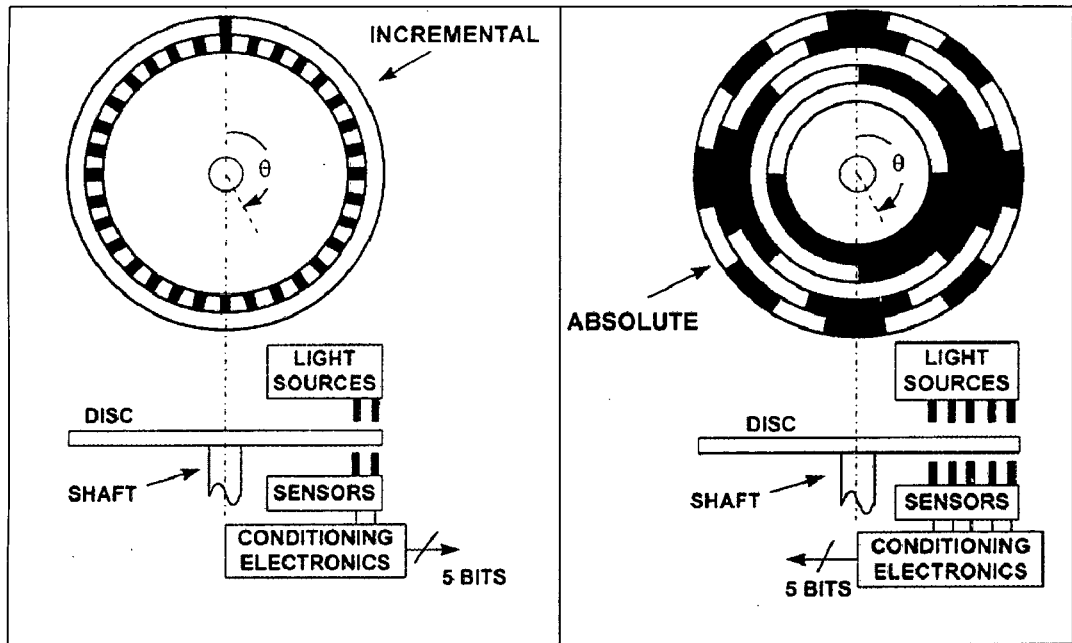


Fig. 4.29 Tipos de encoders.

**Encoder Incrementales:** permiten que un sensor óptico detecte el número de segmentos que dispone el disco y otro sensor detecte la posición cero de dicho disco. Una característica fundamental de los encoders es la cantidad de pulsos cuadrados que genera en una vuelta del eje del motor.

**Encoders Absolutos:** permiten conocer la posición exacta en cada momento sin tener que dar una vuelta entera para detectar el punto cero del disco. La diferencia es que se necesitan varios sensores ópticos y el disco debe de tener una codificación tipo Manchester como se muestra en la figura 4.29.

En este proyecto el objetivo era controlar velocidad y se trabajó con diferentes motores que tenían acoplados encoders incrementales, en algunos casos de la misma resolución la mayoría de 100 pulsos por vuelta,

en otros casos de mayor resolución; pero igual se los pudo controlar cambiando parte del programa del microcontrolador.

#### 4.5.2 El módulo TPM como INPUT CAPTURE

Modo de captura de eventos externos (INPUT CAPTURE): En este modo la máquina medirá eventos temporales externos, aplicados a pines de puertos. Estos eventos pueden ser: la medida del ancho de un pulso o la frecuencia de una señal.

La Figura 4.30 ilustra sobre la medida de una señal cuadrada, aplicada a un pin de la máquina.

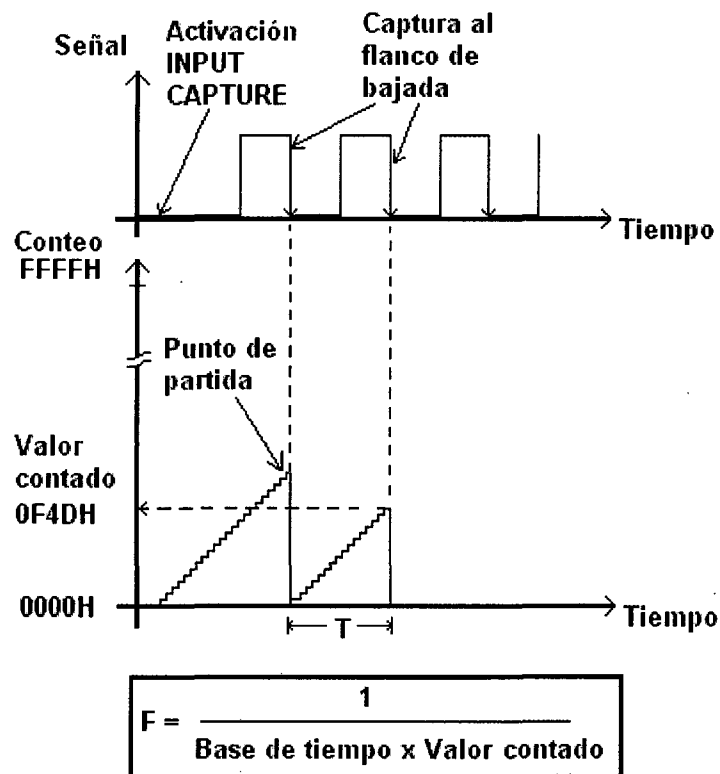


Fig. 4.30: Modo INPUT CAPTURE.



Una vez se habilita el modo de captura de la señal, por flanco de bajada (Activación INPUT CAPTURE), el sistema queda a la espera del evento externo y el contador del módulo inicia el conteo. Al presentarse el primer flanco de bajada de la señal externa, se toma este evento como el punto de partida (contador en ceros) para la medida del período de la señal externa. El valor de frecuencia de la señal será el inverso del valor contado multiplicado por la base de tiempo del sistema, al presentarse el segundo evento de flanco.

Funcionamiento del TPM como captura de evento de entrada (INPUT CAPTURE: La Figura 4.31 muestra las partes involucradas en un evento de captura por TPM.

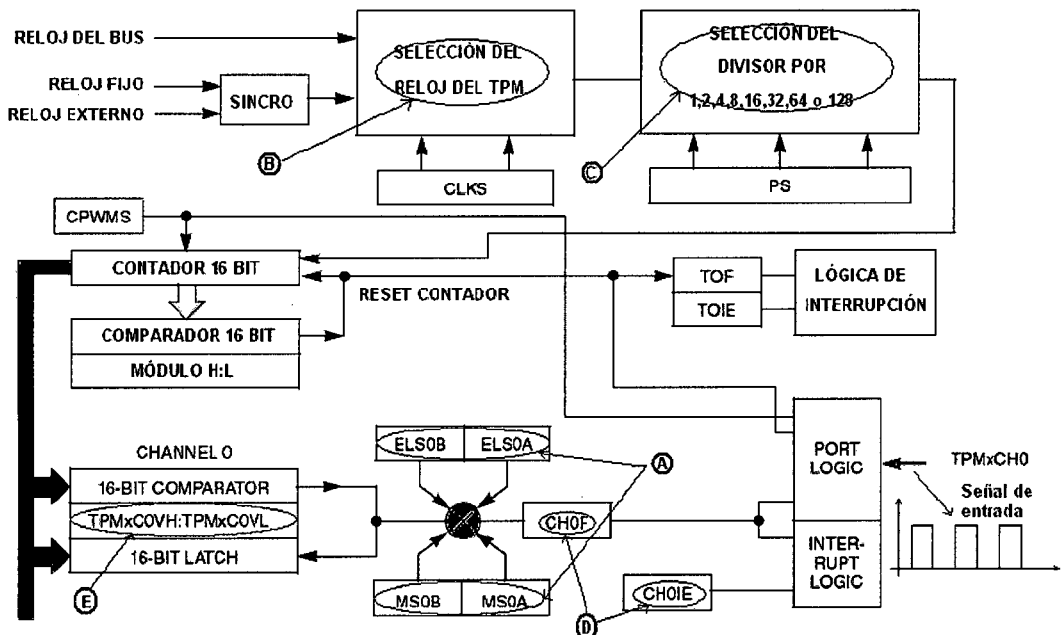


Fig. 4.31: Captura de un evento externo.

Para la medida de una señal periódica aplicada al pin TPM1CH0, el primer paso es definir el modo de funcionamiento como captura de evento en el flanco de subida (A) (ver Tabla 4.4). A continuación se elige el reloj apropiado según el rango de frecuencias de la señal a medir (B). Seguido (C), es necesario especificar un divisor para el reloj elegido. En este momento se podría hacer un polling sobre la bandera CH0F (D) y de esta manera detectar un evento de flanco de subida en el pin TPM1CH0, pero si el propósito es generar un evento de interrupción se deberá poner a "1" el bit CH0IE (D). Finalmente, es necesario leer el valor capturado desde el contador de 16 bits y almacenado en el registro del canal TPM1C0VH:TPM1C0L (E). Este registro contiene el valor que alcanzó el contador de 16 bits, desde la habilitación del TPM hasta el evento de flanco de subida de la señal de entrada.

En términos de la medida de la frecuencia de la señal de entrada, el valor anterior carece de importancia y podría asignarse como el punto de partida, para medir el siguiente valor del contador debido al siguiente flanco. El usuario deberá aclarar el contador antes de iniciar la nueva medida, para que de esta manera se obtenga el valor del período (TPM1C0VH:TPM1C0L) de la señal de entrada.

La manera para calcular la frecuencia de la señal de entrada es:

Frecuencia Base = Frecuencia Reloj Elegido / Divisor Elegido

Frecuencia Señal = Frecuencia Base / TPM1C0VH:TPM1C0L

El circuito de la Figura 4.32 muestra el sistema empleado para este proyecto, así como el pin de entrada del encoder que se describe más adelante.

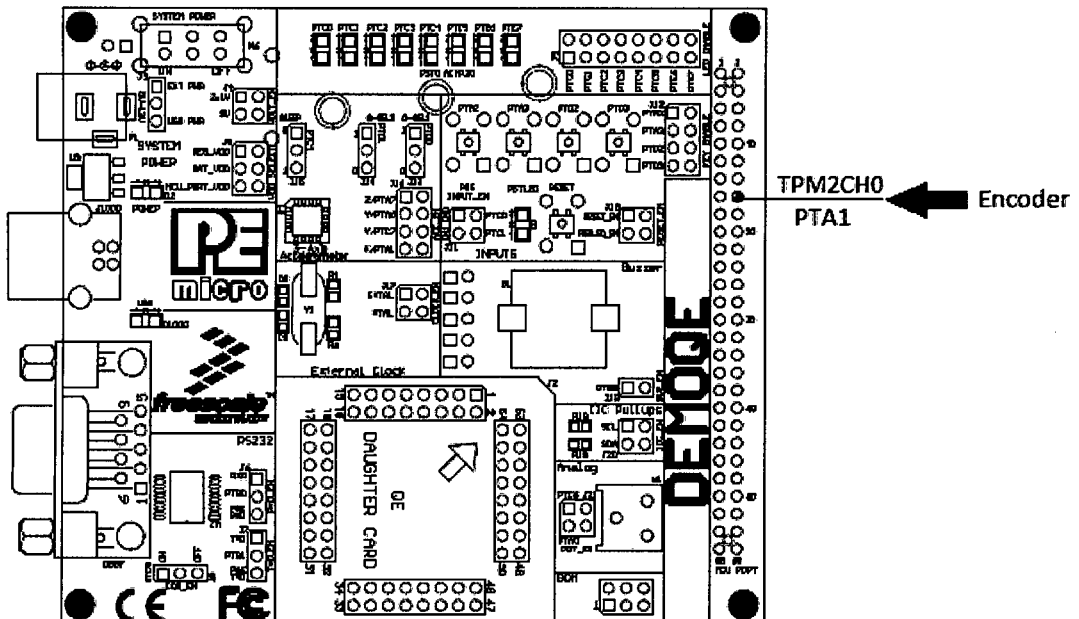


Fig. 4.32: Circuito PWM para control de motor.

A continuación se describe las configuraciones implementadas así como el código necesario en lenguaje C.

Para el proceso de medición de velocidad se realiza un algoritmo más simple, cada vez que se detecta un flanco descendente, se produce una interrupción entonces la cantidad de pulsos aumenta en uno, y la velocidad angular del motor se detecta con la siguiente fórmula:

$$\text{RPM}_{\text{med}} = \frac{\text{Cant\_Pulsos} * 20 * 60[\text{s}]}{100[\text{Pulsos/vuelta}]}$$

Configuración de los registros TPM2SC, TPM2C0SC involucrados en la interface del módulo de teclado con el microcontrolador.



|            |           |            |             |             |             |           |           |           |
|------------|-----------|------------|-------------|-------------|-------------|-----------|-----------|-----------|
| TPM2S<br>C | TOF=<br>0 | TOIE=<br>0 | CPWMS=<br>0 | CLKSB=<br>0 | CLKSA=<br>1 | PS2=<br>0 | PS1=<br>0 | PS0=<br>0 |
|------------|-----------|------------|-------------|-------------|-------------|-----------|-----------|-----------|

**TOIE[6]** =0 No se habilita la interrupción por overflow

**CPWMS[5]** =0 Porque el PWM no estará centrado, sino al alineado al flanco.

**CLKSB:CLKSA[4:3]** =01 Se selecciona reloj de bus interno.

**PS2:PS1[2:0]** =000 para tener un prescaler de 1 .

|          |        |         |        |        |         |         |   |   |
|----------|--------|---------|--------|--------|---------|---------|---|---|
| TPM2C0SC | CHnF=0 | CHnIE=1 | MSnB=0 | MSnA=0 | ELSnB=0 | ELSnA=1 |  |  |
|----------|--------|---------|--------|--------|---------|---------|---|---|

**CHnIE[6]** Se pone a 1 para habilitar la interrupción por canal 0 del módulo TPM2, para actualizar el número de pulsos.

**MSnB:MSnA[5:4]** =00 para habilitar modo INPUT CAPTURE.

**ELSnB:ELSnA[3:2]** =01 para cambios de 1 a 0 en la entrada del encoder.

Rutina de atención a la interrupción del canal 0 del módulo TPM2

```

/*****
/** VectorNumber_Vtpm2ch0 Tpm3_ch0_isr()
/*****

void interrupt VectorNumber_Vtpm2ch0 Tpm2_ch0_isr(void){

    PTETQG=BIT_7; //hasta el momento solo se ve la interrupcion

    TPM2C0SC;

    TPM2C0SC_CH0F=0; //borrar el flag de interrupcion;

    cantidad_de_pulsos++;

}

```

Inicialización del módulo TPM2 para modo INPUT CAPTURE

```

/*****
/** TPM2_Init
/*****

void TPM2_Init(void){

    /*******Configuracion de los registros *****/

    // TOF TOIE CPWMS CLKSB CLKSA PS2 PS1 PS0

    //TOF:No se puede escribir

    //TOIE:0 PARA INHABILITAR INTERRUPCION

    //CPWMS :0 PARA Q SEA MODO CAPTURA

    //CLKSB CLKSA: 01 PARA Q SEA EXTERNO

    //PS2 PS1 PS0:_000_PARA Q CUENTE CON ESCALA DE 1

    TPM2SC=0b00001000;

    //TPM2SC_CLKSA=1;//BUS CLOCK

    //TPM2SC_CLKSB=0;

```

```
//*****Configuracion de los registros *****  
  
// CHnF CHnIE MSnB MSnA ELSnB ELSnA 0 0  
  
//CHnF=0 DEPORQ NO SE PUEDE ESCRIBIR  
//CHnIE:0 PAR DESHABILITAR INTERRUPCION  
//MSnB MSnA :00 PARA INPUT CAPTURE  
//CLKSB CLKSA: 11 PARA Q SEA EXTERNO  
//ELSnB ELSnA:_01_PARA CAMBIOS DE 1 A 0  
//VOY A TRABAJR CON EL CHANEL 0 DEL TPM2  
  
TPM2C0SC= 0b01000100;  
TPM2CNTH=0;  
}
```

#### 4.6 EL TECLADO MATRICIAL Y EL MICROCONTROLADOR MC9S08QE128

En este capítulo iremos a presentar la forma más tradicional de conexión de pulsadores como entradas de microcontroladores. Para que podamos conectar un pulsador a un pin del microcontrolador debemos configura este pin para operar como entrada. Normalmente los pulsadores se conectan a los microcontroladores como se muestra en la figura de abajo.

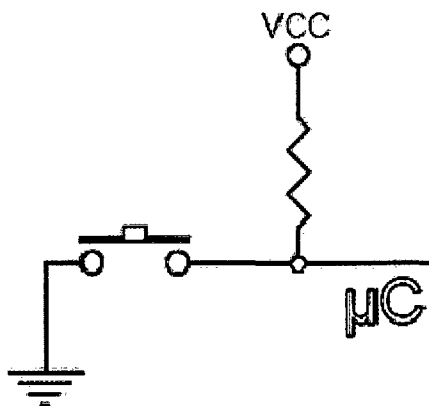


Fig. 4.33: Esquema de conexión de un pulsador

Podemos notar que el lado del pulsador que va hacia el microcontrolador utiliza un resistor pull-up. Este resistor hará que el nivel lógico del pin del microcontrolador configurado como entrada sea normalmente alto. En caso de no utilizar este resistor el pin del microcontrolador estará sujeto a ruidos.

En algunos microcontroladores los pines de entrada pueden ser conectados internamente a un resistor pull-up, evitando que sea necesario utilizar este resistor externamente.

Los microcontroladores generalmente poseen una interrupción de hardware asociado a pines específicos para teclado. Esta interrupción tiene como objetivo evitar que sea necesario el monitoreo por software de pines conectados a pulsadores. El servicio de interrupción de estas interrupciones puede ser activado por eventos de detección de flancos de subida o bajada, o activadas por detección de nivel de tensión.

Los pulsadores poseen micro ranuras generadas en su proceso de fabricación. Debido a estas micro ranuras se generan ruidos (causados por la vibración generadas por las micro ranuras) que podrán ser detectados por software como múltiples ciclos durante el proceso de lectura. Estas vibraciones que causan ruidos, también conocidas como *rebotes*, deben ser eliminadas por hardware (capacitores, flip-flop tipo D) o por software, usando retardos en las lecturas de los pulsadores.

En la figura de abajo se muestra el nivel de tensión de entrada de un pin de un microcontrolador conectado a un pulsador recién presionado.

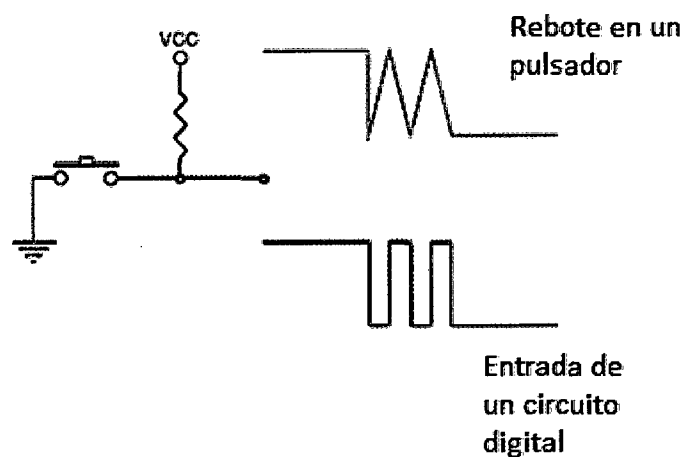


Fig. 4.34: Esquema del funcionamiento eléctrico de un pulsador



Para conectar los pulsadores a los pines del microcontrolador MC9S08QE128 debemos configurar los siguientes registros asociado a uno de los puertos entrada/salida del microcontrolador (PTxDDn,PTxPEn, KBIPEn y KBI2SC)

*Registro de Estado de control de Modulo de Teclado KBI (KBI1SC)*

|        |   |   |   |   |       |         |        |          |
|--------|---|---|---|---|-------|---------|--------|----------|
| KBI2SC | 0 | 0 | 0 | 0 | KBF=1 | KBACK=0 | KBIE=1 | KBIMOD=0 |
|--------|---|---|---|---|-------|---------|--------|----------|

**KBEDG[7:4]** (*Keyboard Edge Select for KBI Port Bits*) – Cada una de estos bits de lectura/escritura selecciona el flanco y/o nivel que será reconocido como evento de interrupción del pin del microcontrolador configurado como entrada de interrupción de teclado (KBIPEn = 1). A su vez selecciona el pullup/pulldown asociado al pin.

0 – Flanco de bajada / Nivel lógico 0, para la detección de evento de KBI y a su vez habilita pullup al pin asociado.

1 – flanco de subida / Nivel lógico 1, para la detección de evento de KBI y a su vez habilita pulldown pin asociado.

**KBF** (*Keyboard Interrupt Flag*) – Este bit de *flag* de estado se pone a nivel lógico 1 cuando un evento asociado al puerto seleccionado, es detectado en cualquiera de los pines configurados como entrada de interrupción de teclado. Este *flag* se limpia escribiendo un 1 lógico en el bit de control KBACK. El *flag* estará permanentemente en nivel lógico 1 si KBIMOD = 1 , o el pin de entrada de teclado permanece en el nivel de tensión configurado como evento de interrupción.

Este *flag* puede ser utilizado para *polling* por software que podrá generar un pedido de interrupción de hardware de módulo de teclado (KBIE = 1).

0 – No existe interrupción de teclado presente.

1 – Interrupción de teclado presente.

**KBACK** (*Keyboard Interrupt Acknowledge*) – Este bit solamente de escritura es utilizado para limpiar el *flag* de estado KBF, por la escritura de nivel lógico 1. El *flag* KBF permanecerá en nivel lógico 1 si KBIMOD=1 para seleccionar el modo de operación y Cualquiera de los pines habilitados para entrada permanecen en nivel de tensión configurado como evento de interrupción. En este caso escribir un 1 lógico en el bit KBACK no tendrá efecto sobre el *flag* KBF.

0 – No tiene efecto.

1 – Limpia el *flag* KBF.

**KBIE** (*Keyboard Interrupt Enable*) – Este bit de control de lectura/escritura determina si la interrupción de hardware será generada cuando el flag de estado KBF este en un nivel lógico igual a 1

Cuando KBIE = 0, ninguna interrupción de hardware será generada por módulo de teclado, pero KBF continuara siendo utilizada para polling por software.

0 – El flag KBF no genera interrupción por hardware (utilizar *polling*)

1 – Sera generada una interrupción de hardware de teclado cuando KBF = 1

**KBIMOD** (*Keyboard Detection Mode*) – Este bit de control de lectura/escritura selecciona entre detección solamente de flanco o flanco y nivel.

–Solamente flanco de subida o flanco de subida y nivel lógico alto (KBEDGn = 1)

– Solamente flanco de bajada o flanco de bajada y nivel lógico bajo (KBEDGn = 0)

0 – Detección solamente por flanco

1 – Detección por flanco y nivel

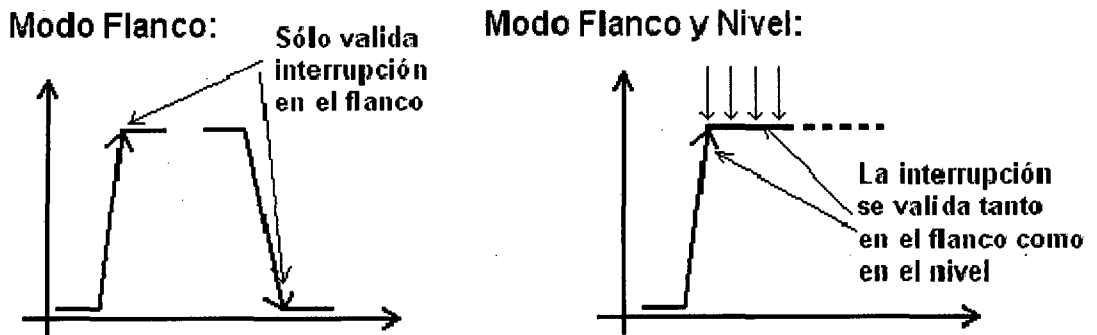


Fig. 4.35: Modos de validación de la interrupción por KBI.

#### Registro para habilitación de modo teclado (**KBI1PE**)

|        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| KBIPE7 | KBIPE6 | KBIPE5 | KBIPE4 | KBIPE3 | KBIPE2 | KBIPE1 | KBIPE0 |
|--------|--------|--------|--------|--------|--------|--------|--------|

Este registro es utilizado para configurar si los pines asociados al módulo de teclado serán habilitados o no.

**KBIPE[7:0]** (*Keyboard Pin Enable for KBI Port Bits*) – Cada uno de estos bits de lectura/escritura selecciona si un pin del puerto del teclado asociado será habilitado como entrada de interrupción de teclado o como pin de entrada/salida para propósito general.

0 – El bit n del puerto de teclado es un pin de entrada/salida de propósito general.

1 – El bit n del puerto de teclado es habilitado como entrada para interrupción por teclado

La tabla de abajo presenta las posibles configuraciones para pull-up o pull-down en los pines asociados al módulo del teclado.

Tabla 4.5: Configuraciones posibles de pull-up o pull-down.

| PTxPE <sub>n</sub><br>(Pull Enable) | PTxDD <sub>n</sub><br>(Dirección de los datos) | KBIPE <sub>n</sub><br>(Habilita los pines Para teclado) | KBEDG <sub>n</sub><br>(Selección de flanco) | Pull -Up      | Pull-Down     |
|-------------------------------------|--|---|---|---------------|---------------|
| 0                                   | 0  | 0   | X   | deshabilitado | deshabilitado |
| 1                                   | 0  | 0   | X   | habilitado    | deshabilitado |
| X                                   | 1  | 0   | X   | deshabilitado | deshabilitado |
| 1                                   | X  | 1   | 0   | habilitado    | deshabilitado |
| 1                                   | X  | 1   | 1   | deshabilitado | habilitado    |
| 0                                   | X  | 1   | X   | deshabilitado | deshabilitado |

A partir de las configuraciones de los registros del microcontrolador MC9S08QE128 presentadas arriba se muestra más adelante la inicialización para el módulo de teclado

Una forma bastante común de implementación de teclado en sistemas embebidos es el teclado matricial. Un teclado matricial de 4 líneas por 4 columnas permite una generación de 16 valores independientes utilizando apenas 8 pines de un puerto del microcontrolador.

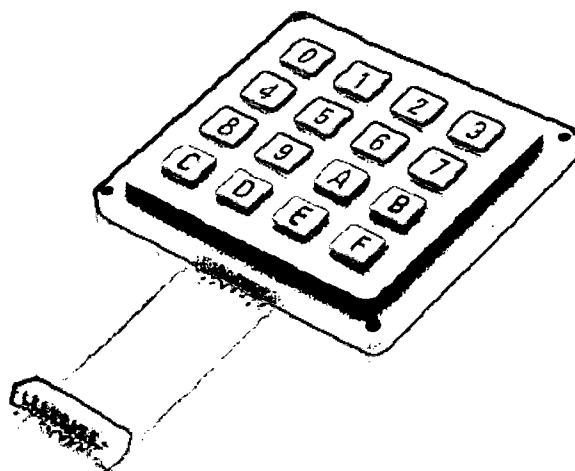


Fig. 4.36: Aspecto físico de un teclado matricial.

La conexión interna de un teclado matricial de 16 valores usada en este proyecto se puede ver en la figura de abajo. Note que para realizar la interface con el microcontrolador las columnas del teclado matricial son conectadas como pines de salida y las filas como entradas. Además a los pines configurados como entradas se le habilitarán los pull-up.

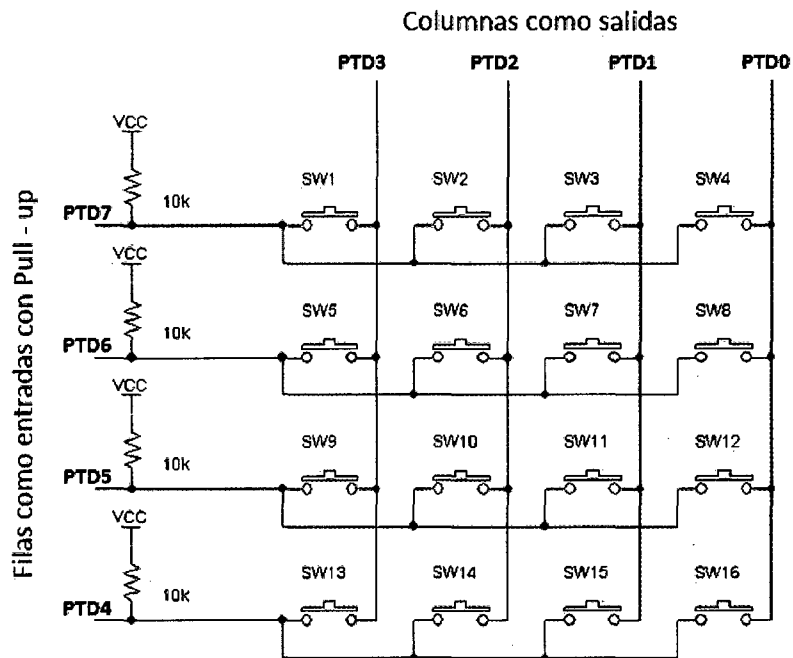


Fig. 4.37 Esquema de conexión interna de un teclado matricial

Los pines configurados como salidas son utilizados para realizar un barrido en el teclado matricial. Este barrido tiene como tarea identificar cuando una tecla ha sido presionada. El barrido lo que haces es alternar los valores lógicos en los pines configurados como salidas. Estos pines siempre estarán configurados para que solamente una de ellas contenga el nivel lógico 0 en un determinado momento. De esta forma, cuando una tecla es presionada genera un código bien definido, donde solamente dos pines del puerto utilizado estarán en un nivel lógico cero (0) (un pin de salida y un pin de entrada).

Una tabla abajo presenta un el códigos de identificación de teclas usado en este proyecto en una interface con un teclado matricial (El puerto D del microcontrolador será utilizado para realizar la interface del teclado como se muestra más abajo).

Tabla 4.6 tabla de identificación de tecla pulsada

| Pines como entradas (filas) |      |      |      | Pines como salidas (columnas) |      |      |      | Tecla |
|-----------------------------|------|------|------|-------------------------------|------|------|------|-------|
| PTD7                        | PTD6 | PTD5 | PTD4 | PTD3                          | PTD2 | PTD1 | PTD0 |       |
| 0                           | 1    | 1    | 1    | 0                             | 1    | 1    | 1    | 1     |
| 0                           | 1    | 1    | 1    | 1                             | 0    | 1    | 1    | 2     |
| 0                           | 1    | 1    | 1    | 1                             | 1    | 0    | 1    | 3     |
| 1                           | 0    | 1    | 1    | 0                             | 1    | 1    | 1    | 4     |
| 1                           | 0    | 1    | 1    | 1                             | 0    | 1    | 1    | 5     |
| 1                           | 0    | 1    | 1    | 1                             | 1    | 0    | 1    | 6     |
| 1                           | 1    | 0    | 1    | 0                             | 1    | 1    | 1    | 7     |
| 1                           | 1    | 0    | 1    | 1                             | 0    | 1    | 1    | 8     |
| 1                           | 1    | 0    | 1    | 1                             | 1    | 0    | 1    | 9     |
| 1                           | 1    | 1    | 0    | 0                             | 1    | 1    | 0    | *     |
| 1                           | 1    | 1    | 0    | 1                             | 0    | 1    | 1    | 0     |
| 1                           | 1    | 1    | 0    | 1                             | 1    | 0    | 1    | #     |

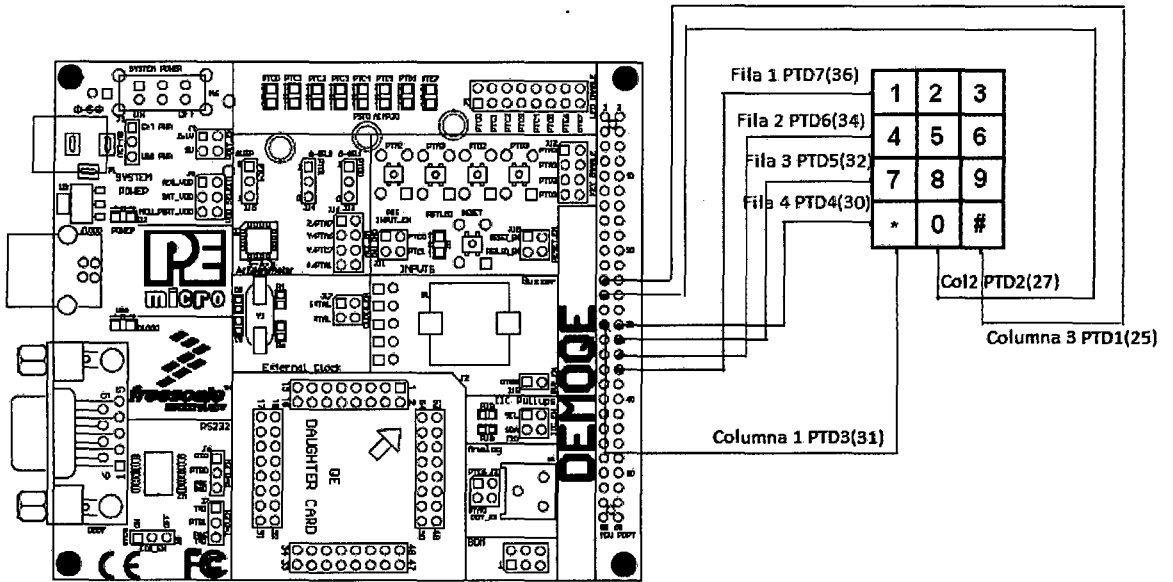


Fig. 4.38: Conexión del teclado al módulo DEMOQ128.

Configuración de los registros PTDDD, PTDPE involucrados en la interface del módulo de teclado con el microcontrolador.

|       |   |   |   |   |      |      |      |      |
|-------|---|---|---|---|------|------|------|------|
| PTDDD | 0 | 0 | 0 | 0 | D3=1 | D2=1 | D1=1 | D0=1 |
|-------|---|---|---|---|------|------|------|------|

**PTDDD[7:0] (PortD Data Direction) –**

0 – El bit n del puerto de teclado es un pin de entrada de propósito general.

1 – El bit n del puerto de teclado es un pin de salida de propósito general

|       |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|
| PTDPE | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|-------|---|---|---|---|---|---|---|---|

**PTDPE[7:0] (PortD Pull Up Enable) –**

0 – Deshabilita el pull-up/pull-down del bit n del puerto D.

1 – Habilita el pull-up/pull-down del bit n del puerto D.

```

/* Configuración de los registros PTDDD Y PTDPE para modulo teclado
*****

/* La parte mas baja son las columnas

/* la mas alta las filas

*****

void PTD_Init(){

    //luego deben tomar estos valores

    PTDD=0xFF;// Inicialmente PTDD=0b11111111

    PTDDD=0x0F;//la parte + baja como salidas Y la + alta como entradas

    PTDPE=0xF0;//se debe habilitar los pullup de las filas

}

```

Configuración de los registros KBI2SC, KBI2PE, KBI2ES para modulo teclado

|        |   |   |   |   |       |         |        |          |
|--------|---|---|---|---|-------|---------|--------|----------|
| KBI2SC | 0 | 0 | 0 | 0 | KBF=1 | KBACK=0 | KBIE=1 | KBIMOD=0 |
|--------|---|---|---|---|-------|---------|--------|----------|

**KBF** (*Keyboard Interrupt Flag*) =1: Interrupción de teclado presente.

**KBACK** (*Keyboard Interrupt Acknowledge*) =0: No tiene efecto.

**KBIE** (*Keyboard Interrupt Enable*) =1: Sera generada una interrupción de hardware de teclado cuando KBF = 1

**KBIMOD** (*Keyboard Detection Mode*) =0: Detección solamente por flanco

–En este caso solamente flanco de bajada (ya que los 4 bits más significativos del registro KBI2SC son ceros: KBEDGn = 0)



|        |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|
| KBI2PE | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|--------|---|---|---|---|---|---|---|---|

**KBIPE[7:0]** (*Keyboard Pin Enable for KBI Port Bits*) –

0 – El bit n del puerto de teclado es un pin de entrada/salida de propósito general.

1 – El bit n del puerto de teclado es habilitado como entrada para interrupción por teclado

//1 – El bit n del puerto de teclado es habilitado como entrada para interrupción por teclado(las filas).

|        |          |          |          |          |   |   |   |   |
|--------|----------|----------|----------|----------|---|---|---|---|
| KBI2ES | KBEDG7=0 | KBEDG6=0 | KBEDG5=0 | KBEDG4=0 | 0 | 0 | 0 | 0 |
|--------|----------|----------|----------|----------|---|---|---|---|

**KBEDG[7:4]** (*Keyboard Edge Select for KBI Port Bits*) –

0 – Flanco de bajada / Nivel lógico 0, para la detección de evento de KBI y a su vez habilita pullup al pin asociado.

/\* Configuración de los registros KBI2SC KBI2PE para modulo teclado

//\*\*\*\*\*

```
void KBI2_Init(char KBI2SC_Config,char KBI2PE_Config, char
KBI2ES_Config ){
```

```
    KBI2PE=0xF0;//0b11110000
```

```
    KBI2SC_KBACK=1;
```

```
    KBI2ES=0x00;
```

```
    KBI2SC=0x06;//0b00001010
```

```
}
```

Proceso de reconocimiento de tecla implementado en microcontrolador MC9S08QE128.

Proceso de reconocimiento de fila activa, atención a la interrupción.

```
/* Servicio de atencion a Interrupcion por teclado
void interrupt VectorNumber_Vkeyboard Kbi_Isr(void){
    char temp;
    column=PTDD&0x0F;
    if(PTDD_PTDD7==0){
        fila=1; }
    if(PTDD_PTDD6==0){
        fila=2;
    }
    if(PTDD_PTDD5==0){
        fila=3;
    }
    if(PTDD_PTDD4==0){
        fila=4;
    }
    temp=PTDD;
    PTDD=0x0F;
    Delay_ms(200);
    KBI2SC_KBACK=1; //para borrar el flag de interrupcion
    PTDD=temp;
}
```

Algoritmo de Barrido.

```

void Rote_Col(void){ //Se alterna el 0 en cada columna

    PTDD=0x07;//cero a la columna 1 0b 0000 0111

    Delay_ms(2);

    PTDD=0x0B;//cero a la columna 2 0b 0000 1011

    Delay_ms(2);

    PTDD=0x0D;//cero a la columna 3 0b 0000 1101

    Delay_ms(2);

}

```

Proceso de reconocimiento de tecla pulsada, conociendo la fila y la columna activa.

```

unsigned int Tecla_Pulsada(){

    unsigned int temp1;

    if(columna==0x07&fila==1){

        temp1=1; }

    if(columna==0x0B&fila==1){

        temp1=2; }

    if(columna==0x0D&fila==1){

        temp1=3; }

    if(columna==0x07&fila==2){

        temp1=4; }

    if(columna==0x0B&fila==2){

        temp1=5; }

    if(columna==0x0D&fila==2){

        temp1=6; }

    if(columna==0x07&fila==3){

        temp1=7; }

```

```
if(columna==0x0B&fila==3){
    temp1=8; }
if(columna==0x0D&fila==3){
    temp1=9; }
if(columna==0x0B&fila==4){
    temp1=0; }
if(columna==0x07&fila==4){ //michi
    temp1=14; }
if(columna==0x0D&fila==4){ //asterisco
temp1=13; }
return(temp1);
}
```

```
//*****
```

#### 4.7 EL DISPLAY LCD 2X16 Y EL MICROCONTROLADOR MC9S08QE128

En este capítulo se describirá el proceso de realización de la interfaz con el display LCD con el microcontrolador MC9S08QE128.

Las pantallas de cristal líquido LCD o display LCD para mensajes (Liquid Cristal Display) tienen la capacidad de mostrar cualquier carácter alfanumérico, permitiendo representar la información que genera cualquier equipo electrónico de una forma fácil y económica.

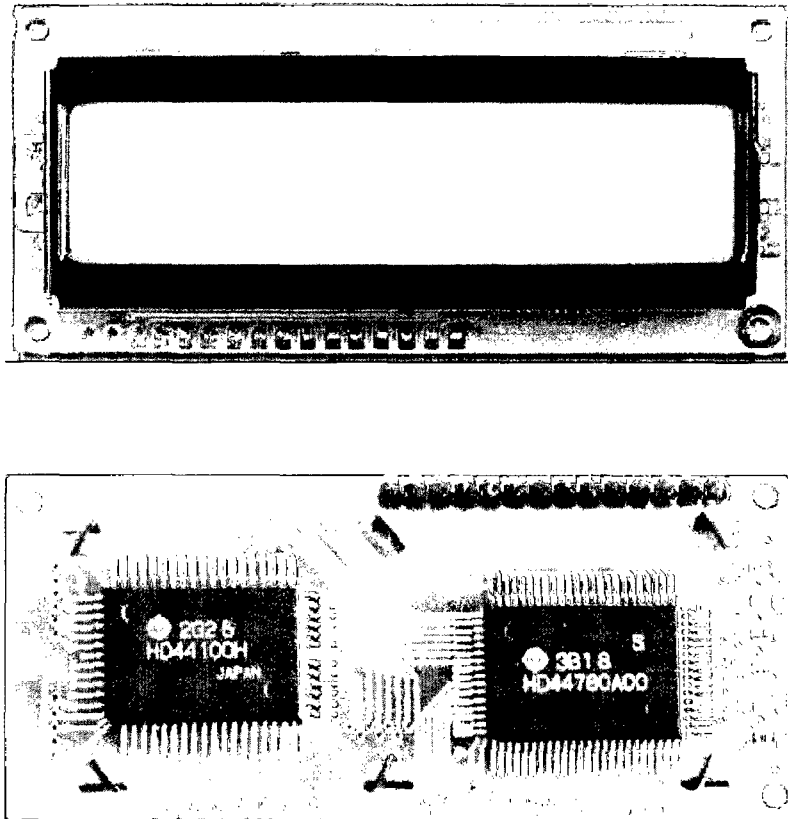


Fig. 4.39: Aspecto físico de un Display LCD 2x16.

La pantalla consta de una matriz de caracteres (normalmente de 5x7 o 5x8 puntos) distribuidos en una, dos, tres o cuatro líneas de 16 hasta 40 caracteres cada línea.

El proceso de visualización es gobernado por un microcontrolador incorporado a la pantalla, siendo el Hitachi 44780 el modelo de controlador más utilizado. Estos controladores poseen 80 bytes de memoria RAM para los datos de display (DDRAM), 64 bytes de RAM para generador de caracteres de usuario y 1240 bytes para ROM de generador de caracteres.

Los módulos LCD normalmente presentan distribución de pines estándar, con un conector de 16 pines, cuyas funciones se detallan a continuación.

Tabla 4.7: distribución de pines y funciones del display LCD

| PIN | SÍMBOLO  | DESCRIPCIÓN  |
|-----|----------|--|
| 1   | $V_{SS}$ | Patilla de tierra de alimentación  |
| 2   | $V_{DD}$ | Patilla de alimentación de 5 V   |
| 3   | $V_O$    | Patilla de contraste del cristal líquido. Normalmente se conecta a un potenciómetro a través del cual se aplica una tensión variable entre 0 y +5V que permite regular el contraste del cristal líquido. |
| 4   | RS       | Selección del registro de control/registro de datos:<br>RS=0 Selección del registro de control<br>RS=1 Selección del registro de datos   |
| 5   | R/W      | Señal de lectura/escritura<br>R/W=0 El módulo LCD es escrito<br>R/W=1 El módulo LCD es leído   |
| 6   | E        | Señal de activación del módulo LCD:<br>E=0 Módulo desconectado<br>E=1 Módulo conectado   |
| 7   | D0       | Bit 0 de dato/comando. No utilizado en modo de 4 bits  |
| 8   | DB1      | Bit 1 de dato/comando. No utilizado en modo de 4 bits  |
| 9   | DB2      | Bit 2 de dato/comando. No utilizado en modo de 4 bits  |
| 10  | DB3      | Bit 3 de dato/comando. No utilizado en modo de 4 bits  |
| 11  | DB4      | Bit 4 de dato/comando  |
| 12  | DB5      | Bit 5 de dato/comando  |
| 13  | DB6      | Bit 6 de dato/comando  |
| 14  | DB7      | Bit 7 de dato/comando  |
| 15  | A        | Ánodo de LED de backlight  |
| 16  | K        | Cátodo de LED de backlight   |

Del D0 – D7: Bus de datos bi-direccional. A través de estas líneas se realiza la transferencia de información entre el módulo LCD y el sistema informático que lo gestiona

La DDRAM es el área de memoria RAM interna del LCD donde se envían los caracteres (en código ASCII de 8 bits) que se requieren visualizar en la pantalla. Se organiza de forma que los registros desde el 0x00 a 0x27 almacenan los datos de la primera línea y desde el 0x40 al 0x67 los datos de la segunda línea. Es claro que de esos 40 bytes de cada línea, apenas 16 en cada línea es visible (en un display de 16 caracteres). La figura de abajo representa la organización de la DDRAM

| Linha | Endereço |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1     | 00       | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 2     | 40       | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 |

En modo normal el primer carácter de la primera fila estará en la dirección 0x00, el segundo en 0x01 y así sucesivamente.

La CGRAM es un área de memoria dedicada a creación de caracteres de usuario. En los 64 bytes de CGRAM es posible la creación de 8 caracteres de 8x5 puntos o 4 de 10x5. Esos caracteres de usuario pueden ser seleccionados por los códigos de la primera columna de la tabla de abajo. La CGROM es un área de memoria ROM interna del LCD donde está definido todo el juego de caracteres que el display puede mostrar (números, caracteres latinos, griegos, caracteres japoneses "Kanji" y otros). Tiene almacenados hasta 160 caracteres de 5x7 puntos (para números, caracteres latinos y Kanji) y de 32 de 5x10 puntos (para caracteres griegos y otros), conforme puede ser visto en la tabla de abajo (la tabla siguiente presenta un conjunto de caracteres de la ROM)

| Lower 4 Bits \ Upper 4 Bits | 0000       | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|-----------------------------|------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| xxxx0000                    | CG-RAM (1) |      | 0    | @    | P    | `    | P    |      |      |      | -    | 夕    | ミ    | α    | ρ    |      |
| xxxx0001                    | (2)        |      | !    | 1    | A    | Q    | a    | q    |      |      | 。    | ア    | チ    | △    | ä    | q    |
| xxxx0010                    | (3)        |      | "    | 2    | B    | R    | b    | r    |      |      | 「    | イ    | ツ    | ×    | ρ    | θ    |
| xxxx0011                    | (4)        |      | #    | 3    | C    | S    | c    | s    |      |      | 」    | ウ    | テ    | モ    | ε    | ε    |
| xxxx0100                    | (5)        |      | \$   | 4    | D    | T    | d    | t    |      |      | 、    | エ    | ト    | フ    | μ    | Ω    |
| xxxx0101                    | (6)        |      | %    | 5    | E    | U    | e    | u    |      |      | ・    | オ    | ナ    | 1    | ε    | ü    |
| xxxx0110                    | (7)        |      | &    | 6    | F    | V    | f    | v    |      |      | ヲ    | カ    | ニ    | ヨ    | ρ    | Σ    |
| xxxx0111                    | (8)        |      | '    | 7    | G    | W    | g    | w    |      |      | ア    | キ    | ヌ    | ラ    | g    | π    |
| xxxx1000                    | (1)        |      | <    | 8    | H    | X    | h    | x    |      |      | ィ    | ク    | ネ    | リ    | ρ    | ×    |
| xxxx1001                    | (2)        |      | >    | 9    | I    | Y    | i    | y    |      |      | ウ    | ケ    | ル    | ル    | ρ    | γ    |
| xxxx1010                    | (3)        |      | *    | :    | J    | Z    | j    | z    |      |      | エ    | コ    | ン    | レ    | j    | κ    |
| xxxx1011                    | (4)        |      | +    | ;    | K    | L    | k    | l    |      |      | オ    | サ    | ヒ    | ロ    | *    | π    |
| xxxx1100                    | (5)        |      | ,    | <    | L    | ¥    | l    | l    |      |      | カ    | シ    | フ    | ワ    | φ    | 円    |
| xxxx1101                    | (6)        |      | -    | =    | M    | ]    | m    | }    |      |      | ユ    | ス    | ハ    | ン    | ε    | ÷    |
| xxxx1110                    | (7)        |      | .    | >    | N    | ^    | n    | ‡    |      |      | ヨ    | セ    | ホ    | ”    | π    |      |
| xxxx1111                    | (8)        |      | /    | ?    | O    | _    | o    | †    |      |      | ウ    | ソ    | マ    | ”    | ö    | ■    |

Fig. 4.40: Tabla de caracteres de un display.



Para usar un módulo LCD es necesario conocer su hardware. En este caso se utilizó un LCD de 2 líneas por 16 caracteres con controlador KS0065. Estos módulos trabajan en dos modos distintos de comunicación: modo 4 u 8 bits.

En el modo de 8 bits los datos de comandos son leídos o escritos por el barrido de 8 bits del módulo (pines DB0-DB7). En el modo de 4 bits se utiliza solo los pines DB4-DB7 y la información es enviada en dos etapas. Para utilizar el módulo en modo 4 u 8 bits es necesario utilizar una secuencia de inicialización determinada por el fabricante, conforme será visto más tarde. Por lo tanto antes de presentar los modos de inicialización es necesario conocer los tipos de comandos disponibles en el módulo y su forma de uso.



Es importante notar que el envío de un comando debe obedecer los siguientes pasos.

1. El dato/comando es colocado en un barrido de datos;
2. La línea RS es configurada a 1 para modo dato y a "0" modo comando.
3. La línea E genera un pulso de por lo menos 450ns.

La secuencia necesaria para inicializar el módulo de comunicación en modo 8 bits es:

1. Se envía el comando de configuración del display
2. Se envía el comando de configuración de líneas y

La secuencia necesaria para inicialización del módulo LCD en modo de 4 bits que es usado en este proyecto es el siguiente:

1. Se envía un comando de 4 bits conteniendo el valor 0x03 (0011 en binario) por las líneas DB4 A DB7.
2. Se espera aproximadamente 5 mseg.
3. Se envía nuevamente conteniendo el valor 0x03 (0011 en binario) por las líneas DB4 A DB7.
4. Se espera aproximadamente 100useg.
5. Se envía nuevamente conteniendo el valor 0x03 (0011 en binario) por las líneas DB4 A DB7.
6. Se envía el comando valor 0x02 (0010 en binario) por las líneas DB4 A DB7. A partir de este instante el display ya está configurado en modo de 4 bits y debemos enviar cada byte en dos mitades de 4 bits (llamadas nibble), iniciando por los bits más significativos.
7. Se envía comando de configuración de display.
8. Se envía comando de configuración de modo de operación.

9. Se envía comando para apagar cualquier mensaje en el display y posicionar el cursor en el inicio de la primera línea.

Configurado el modulo podemos escribir los datos en cualquier momento basta enviar los datos con la línea RS en nivel lógico alto.

Tabla 4.9: Principales comandos interpretados por los módulos LCD con controladores KS0065 o HD44780 en formato hexadecimal.

| <b>Configuración del display</b>                         |      |
|--|------|
| 1 línea con caracteres de 5x7 (8 bits)                   | 0x30 |
| 2 líneas con caracteres de 5x7 (8 bits)                  | 0x38 |
| 1 línea con caracteres de 5x10 (8 bits)                  | 0x34 |
| 1 líneas con caracteres de 5x7 (4 bits)                  | 0x20 |
| 2 líneas con caracteres de 5x7 (4 bits)                  | 0x28 |
| 1 línea con caracteres de 5x10 (4 bits)                  | 0x24 |
| <b>Control Activo/Inactivo del display</b>               |      |
| Display ON, Cursor en ON                                 | 0x0E |
| Display ON, Cursor intermitente                          | 0x0F |
| Display ON sin cursor                                    | 0x0C |
| Display apagado  | 0x08 |
| <b>Configuración de modo de operación</b>                |      |
| Escribe de izquierda a derecha                           | 0x07 |
| Escribe de derecha a izquierda                           | 0x05 |
| Escribe desplazando el cursor a la derecha               | 0x06 |
| Escribe desplazando el cursor a la izquierda             | 0x04 |
| <b>Comandos útiles</b>                                   |      |
| Limpia el display y cursor a la posición 1               | 0x01 |
| Retorna el cursor (sin alterar la DDRAM)                 | 0x02 |
| Desplaza solamente el cursor a derecha                   | 0x14 |
| Desplaza solamente el cursor a izquierda                 | 0x10 |
| Desplaza cursor +mensaje a derecha                       | 0x1C |
| Desplaza cursor + mensaje a izquierda                    | 0x18 |
| Desplaza el cursor a la primera línea y primera posición | 0x80 |
| Desplaza el cursor a la segunda línea , primera posición | 0xC0 |

Para realizar la interfaz con el microcontrolador MC9S08QE128 en modo 4 bits utilizaremos el Puerto B, como se muestra en la siguiente figura:

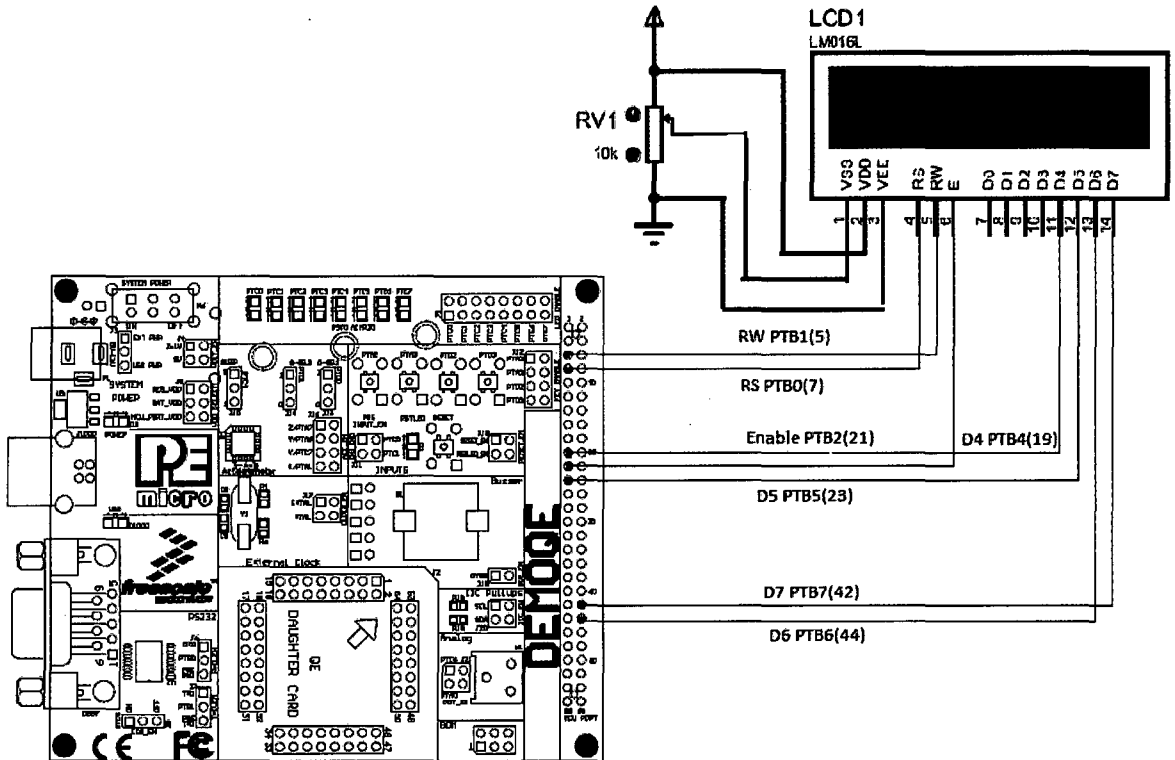


Fig. 4.41: Esquema de conexión del LCD al módulo DEMOQE128.

En base a lo mencionado a lo largo de este capítulo procedemos a elaborar la inicialización del display LCD en el microcontrolador MC9S08QE128 en modo 4 bits.

```

//*****lcd.h*****
//*****ASIGNACION DE LOS PINES DEL MICROCONTROLADOR*****
#ifndef LCD_ENABLE // if lcd_enable is not defined
#define LCD_ENABLE PTBD_PTBD2 // LCD enable pin on PTB1
#define LCD_ENABLE_DIR PTBDD_PTBD2 // LCD enable pin direction
#define LCD_RW PTBD_PTBD1 // LCD r/s pin on PTB0
#define LCD_RW_DIR PTBDD_PTBD1 // LCD r/s pin direction
#define LCD_RS PTBD_PTBD0 // LCD r/s pin on PTB0
#define LCD_RS_DIR PTBDD_PTBD0 // LCD r/s pin direction
#define LCD_D4 PTBD_PTBD4 // LCD data D4 pin
#define LCD_D4_DIR PTBDD_PTBD4 // LCD data D4 pin direction
#define LCD_D5 PTBD_PTBD5 // LCD data D5 pin
#define LCD_D5_DIR PTBDD_PTBD5 // LCD data D5 pin direction
#define LCD_D6 PTBD_PTBD6 // LCD data D6 pin
#define LCD_D6_DIR PTBDD_PTBD6 // LCD data D6 pin direction
#define LCD_D7 PTBD_PTBD7 // LCD data D7 pin
#define LCD_D7_DIR PTBDD_PTBD7 // LCD data D7 pin direction
#endif

#define LCD_SEC_LINE 0x40 // Address of the second line of the LCD

#ifndef lcd_type
#define lcd_type 2 // 0=5x7, 1=5x10,2=2 lines
#endif

```

Inicialización del display LCD en modo 4 bits en el microcontrolador MC9S08QE128.

```

//*****
/* LCD initialization
//*****
void LCD_init()

{   PTDDD=0b11111111;// // Configure the pins as outputs

    LCD_RS = 0;

    LCD_RW = 0;

    LCD_ENABLE = 0;

    Delay_ms(15);

    // LCD 4-bit mode initialization sequence send three times 0x03 and //then
    0x02 to finish configuring the LCD

    LCD_send_nibble(0x30); //podrian ser 2 veces
    Delay_ms(5); //queda pq lo minimo es 4.2 mseg
    LCD_send_nibble(0x30); //podrian ser 2 veces
    Delay_ms(5); // //queda pq lo minimo es 4.2 mseg
    LCD_send_nibble(0x30); //podrian ser 2 veces
    Delay_x10us(20); // //queda pq lo minimo es 4.2 mseg
    LCD_send_nibble(0x20);

    // Now send the LCD configuration data LCD_2lineas4Bits 5x7
    LCD_send_byte(0,0x28); //definitivamnete tiene q ser 28, porq el //display
    es de 2 lineas y los caractres de 5x7 puntos.

    LCD_send_byte(0,0xe); //Borra el display y LCD_Cursor_on
    LCD_send_byte(0,1);//LCD_Borra

    LCD_send_byte(0,6); //LCD_cursor en modo incrementar
    lcd_mode = 0x0e;}

```

Envió de comandos por las líneas DB4 a DB7 del display LCD en modo 4 bits. Enviamos cada byte en dos mitades de 4 bits (llamadas nibble) iniciando por los bits más

```

//*****

/** Send a nibble to the LCD

//*****

/** Calling arguments

/** char data : data to be sent to the display

//*****

void LCD_send_nibble(Byte data)
{
    byte temp1, temp2;

    temp1=data; //se queda con el nibble alto

    temp1&=0xF0;//y se guarda

    temp2=PTDD;

    temp2&=0x0F;

    PTDD=temp1|temp2;//se envia la parte alta del dato(+significativos)

    asm{

    nop;

    }

    LCD_ENABLE=1; //minimo 450useg

    Delay_x10us(1);

    LCD_ENABLE=0;

}

```



Escritura de un carácter en el display LCD en modo 4 bits.

```

//*****
/* Write a character on the display
//*****
/* Calling arguments:
/* char c : character to be written
//*****
/* Notes :
/* \f clear the display
/* \n and \r return the cursor to line 1 column 0
//*****
void LCD_write_char(char c) {
    switch (c)    {
        case '\f' :    LCD_send_byte(0,1);
                        Delay_ms(2);
                        break;
        case '\n' : LCD_pos_xy(1,2); break;
        case '\b' :    LCD_pos_xy(0,0x10); break;
        default  : LCD_send_byte(1,c);
    }
}

```

Este proceso de escritura llama a una rutina LCD\_send\_byte( ).

```

//*****
/* Write a byte into the LCD
//*****
/* Calling arguments:
/* char address : 0 for instructions, 1 for data
/* char data : command or data to be written
//*****
void LCD_send_byte(char address, char data)

{

    byte a;

    LCD_RS=0;

    a=lcd_read_byte()&&0b10000000; //monitoreo al bit 7

    while(a==0b10000000){ //espera a q se desocupe o libere

    }

    Delay_ms(10);

    LCD_RS = address;

    asm{

    nop; }

    LCD_RW=0;

    asm{

    nop; }

        LCD_ENABLE = 0;          // set LCD enable line to 0

        LCD_send_nibble(data & 0xf0); // send the higher nibble

        LCD_send_nibble(data << 4); // send the lower nibble

        if (LCD_RS==1){

            LCD_RS=0;

        }

}

```

Escritura de un número en el display LCD en modo 4 bits.

```

//*****
/* Write a numero on the display
//*****
/* Calling arguments:
/* unsigned int c : numero to be written
//*****
/* Notes : al numero q se quiere mostra se debe sumar 48
//*****
void LCD_write_num(unsigned int c) {

    LCD_send_num(1,c);

}

```

Este proceso de escritura llama a una rutina LCD\_send\_num().

```

//*****
/* Calling arguments:
/* char address : 0 for instructions, 1 for data
/* unsigned int data : numero a ser escrito
//*****
void LCD_send_num(char address,unsigned int data)

{ byte a;

    LCD_RS=0;

    a=lcd_read_byte()&&0b10000000; //monitoreo al bit 7

    while(a==0b10000000){ //espera a q se desocupe o libere

    }

    Delay_ms(10);

    LCD_RS = address;

    asm{

    nop; }

    LCD_RW=0;

    asm{

    nop; }

```

```

LCD_ENABLE = 0;          // set LCD enable line to 0
LCD_send_nibble(data & 0xf0); // send the higher nibble
LCD_send_nibble(data << 4); // send the lower nibble
if (LCD_RS==1){
LCD_RS=0;
}
}

```

Escritura de una cadena de caracteres LCD\_write\_string().

```

//*****
/* Write a string on the display
//*****
/* Calling arguments:
/* char *c : pointer to the string
//*****
void LCD_write_string (char *c)
{
    while (*c)
    {
        LCD_write_char(*c);
        c++;
    }
}

```

Otras funciones importantes:

```

//*****

/* Turn the display on

//*****

void LCD_display_on(void)
{
    lcd_mode |= 4;
    LCD_send_byte (0,lcd_mode);
}

//*****

/* Turn the display off

//*****

void LCD_display_off(void)
{
    lcd_mode &= 0b11111011;
    LCD_send_byte (0,lcd_mode);
}

//*****

/* Turn the cursor on

//*****

void LCD_cursor_on(void)
{ lcd_mode |= 2;
    LCD_send_byte (0,lcd_mode);
}

```

```
/*******  
  
/* Turn the cursor off  
  
/*******  
void LCD_cursor_off(void)  
{  
    lcd_mode &= 0b11111101;  
    LCD_send_byte (0,lcd_mode);  
}  
  
/*******  
  
/* Turn on the cursor blink function  
  
/*******  
void LCD_cursor_blink_on(void)  
{  
    lcd_mode |= 1;  
    LCD_send_byte (0,lcd_mode);  
}  
  
/*******  
  
/* Turn off the cursor blink function  
  
/*******  
void LCD_cursor_blink_off(void)  
{  
    lcd_mode &= 0b11111110;  
    LCD_send_byte (0,lcd_mode);  
}
```

#### 4.8 EL DRIVER L293D Y EL MICROCONTROLADOR MC9S08QE128 [10]

El L293D es un drive de 4 canales capaz de proporcionar una corriente de salida de hasta 1 A por canal. Cada canal es controlado por señales de entrada compatibles TTL y cada pareja de canales dispone de una señal de habilitación que desconecta las salidas de los mismos.

Dispone de una patilla para alimentación de las cargas que están controlando, de forma que dicha alimentación es independiente de la lógica de control.

En este proyecto se empleó este integrado porque las pruebas se realizaron con el motor sin carga y la corriente era pequeña. La figura 4.42 muestra el encapsulado de 16 pines, la distribución de patillas y la descripción de las mismas.

| Pin | Nombre        | Descripción                       | Patillaje |
|-----|---------------|-----------------------------------|-----------|
| 1   | Chip Enable 1 | Habilitación de los canales 1 y 2 |           |
| 2   | Input 1       | Entrada del Canal 1               |           |
| 3   | Output 1      | Salida del Canal 1                |           |
| 4   | GND           | Tierra de Alimentación            |           |
| 5   | GND           | Tierra de Alimentación            |           |
| 6   | Output 2      | Salida del Canal 2                |           |
| 7   | Input 2       | Entrada del Canal 1               |           |
| 8   | Vs            | Alimentación de las cargas        |           |
| 9   | Chip Enable 2 | Habilitación de los canales 3 y 4 |           |
| 10  | Input 3       | Entrada del Canal 3               |           |
| 11  | Output 3      | Salida del Canal 3                |           |
| 12  | GND           | Tierra de Alimentación            |           |
| 13  | GND           | Tierra de Alimentación            |           |
| 14  | Output 4      | Salida del Canal 4                |           |
| 15  | Input 4       | Entrada del Canal 4               |           |

Fig. 4.42: Diagrama de pines del encapsulado

#### 4.8.1 Diagrama de bloques del L293D

En la figura 4.43 se muestra el diagrama de bloques del L293D. La señal de control **EN1** activa la pareja formada por los drivers 1 y 2. La señal **EN2** activa la pareja de drivers 3 y 4. Las salidas **OUTn** se asocian con las correspondientes **OUTn**. Las señales de salida son amplificadas respecto a las de entrada tanto en tensión (hasta +V<sub>ss</sub>) como en corriente (máx. 1 A).

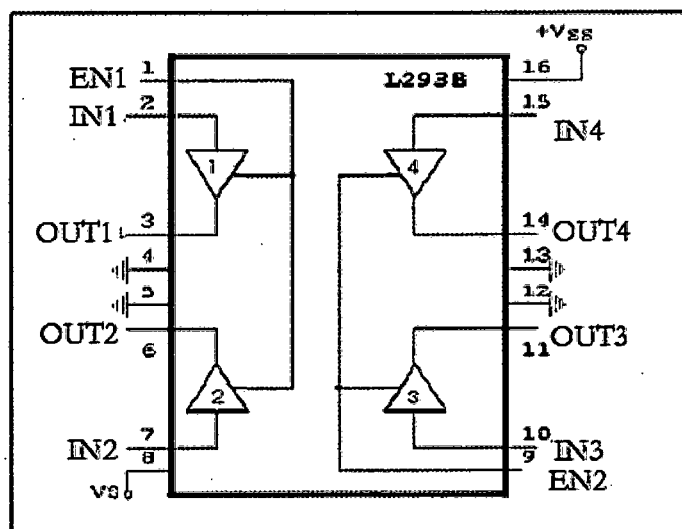


Fig. 4.43: Diagrama de bloques del L293D

La tabla de funcionamiento para cada uno de los drivers es la siguiente:

Tabla 4.10 Funcionamiento para cada uno de los driver.

| $V_{IN_n}$ | $V_{OUT_n}$ | $V_{EN_n}$ | Dónde:<br>H: nivel alto "1".<br>L: nivel bajo "0"<br>Z: alta impedancia |
|------------|-------------|------------|---|
| H          | H           | H          |   |
| L          | L           | H          |   |
| H          | Z           | L          |   |
| L          | Z           | L          |   |



#### 4.8.2 Aplicaciones del L293D

En este apartado se muestran distintas configuraciones de conexión de motores al L293D.

##### a) Giro de 2 motores en único sentido

En la figura 4.44 se muestra el modo de funcionamiento de motores de corriente continua que giran en un único sentido

El motor M1 se activa al poner a nivel bajo la entrada de control A

El motor M2 se activa al poner a nivel bajo la entrada de control B

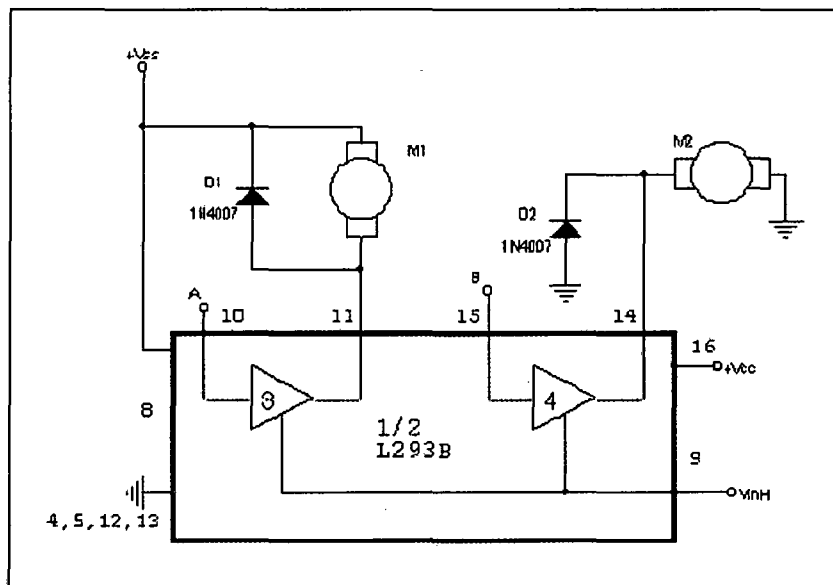


Fig. 4.44: Conexión de 2 motores de continua

Tabla 4.11: Tabla de verdad de la conexión de 2 motores de continua.

| $V_{IN4}$ | A | M1                             | B | M2                             |
|-----------|---|--------------------------------|---|--------------------------------|
| H         | H | Parada rápida del motor        | H | Giro                           |
| H         | L | Giro                           | L | Parada rápida del motor        |
| L         | X | Motor desconectado, giro libre | X | Motor desconectado, giro libre |

Los diodos D1 y D2, están conectados para proteger el circuito cuando se generan los picos de arranque de los motores. Si no se trabaja a máxima potencia de trabajo pueden eliminarse del circuito.

### b) Control del giro de un motor en los dos sentidos

El circuito de la figura 4.45 permite controlar el doble sentido de giro del motor.

Cuando la entrada C está a nivel alto, el motor gira hacia la izquierda. Cuando la entrada C a nivel alto y la entrada D a nivel bajo, se cambia al sentido de giro del motor de la derecha. Dicho de otra manera con un pulsador se puede cambiar de sentido el giro del motor DC, también se podría usar una compuerta lógica negadora y un pulsador.

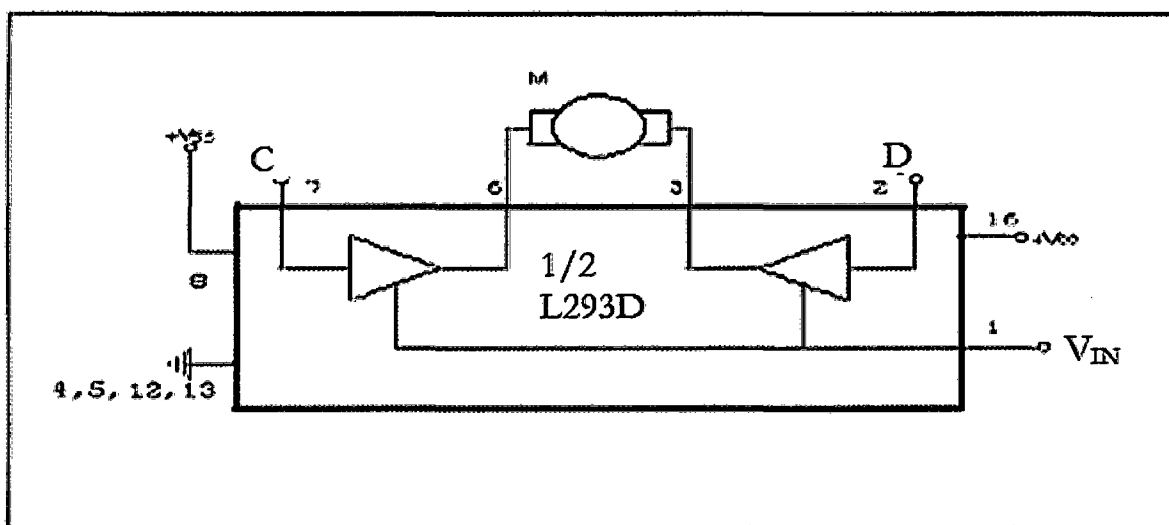


Fig. 4.45: Circuito de control para el doble giro del motor DC.

Tabla 4.11: Tabla de verdad para la conexión doble giro del motor DC.

| $V_{IN_A}$ | A | M1                             | B | M2                             |
|------------|---|--------------------------------|---|--------------------------------|
| H          | H | Parada rápida del motor        | H | Giro                           |
| H          | L | Giro                           | L | Parada rápida del motor        |
| L          | X | Motor desconectado, giro libre | X | Motor desconectado, giro libre |

Esta configuración fue la que se usó en este proyecto, para el cambio de sentido se trabajó con un pulsador doble.

#### 4.9 SOFTWARE CODEWARRIOR. [4]

Para programar los microcontroladores Freescale se usa el software CODE WARRIOR. Este programa es un ambiente de Programación y Herramienta de Desarrollo, así como de depuración de los microcontroladores Freescale.

##### 4.9.1 Creación de proyectos en C

A continuación, son listados los pasos mínimos para crear un proyecto en el software CodeWarrior® 6.2.

Paso 1. Ejecución del CodeWarrior® 6.2: Para iniciar un programa es necesario seguir la ruta del Windows®: Start > Programs > Freescale CodeWarrior > CodeWarrior IDE. El programa comenzará a ejecutarse y la ventana de la Figura 4.46 aparecerá.

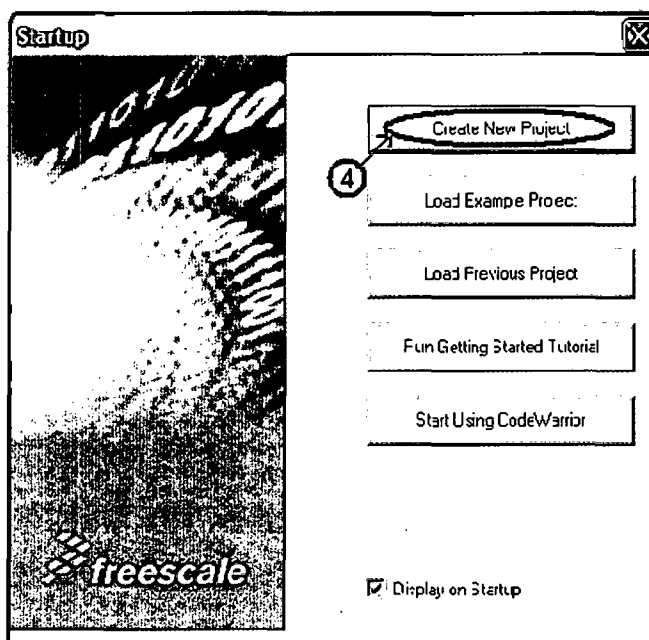


Fig. 4.46: Ventana de inicio.

Paso 2. Creación de un nuevo proyecto: Pulsar sobre el botón Create New Project.

Paso 3. Selección del MCU y la conexión: La nueva ventana que aparece es la mostrada en la Figura 4.47. En esta ventana seleccionar el microcontrolador a trabajar y la opción de conexión disponible.

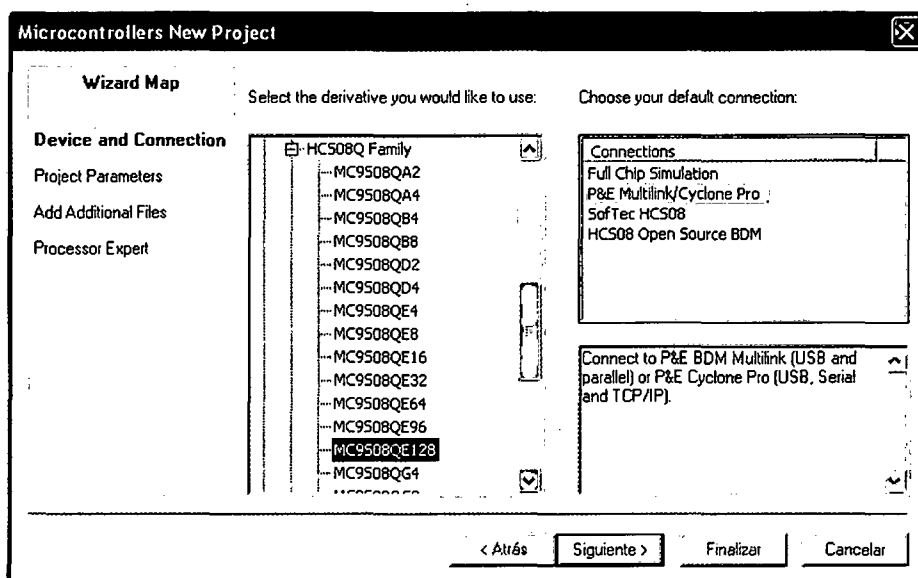


Fig. 4.47: Selección del MCU y la conexión.

El microcontrolador a trabajar será el MC9S08QE128 con la opción de conexión P&E Multilink/Cyclone Pro. Presionar el botón de Next, para pasar a la siguiente ventana.

**Paso 4. Selección de parámetros del proyecto:** En la ventana de la Figura 4.48 se detallan: la selección de los tipos de lenguaje a desarrollar, el nombre del proyecto y la ruta de almacenamiento del mismo. Si es el caso seleccionar como lenguaje el C. Finalmente presionar el botón de Next.

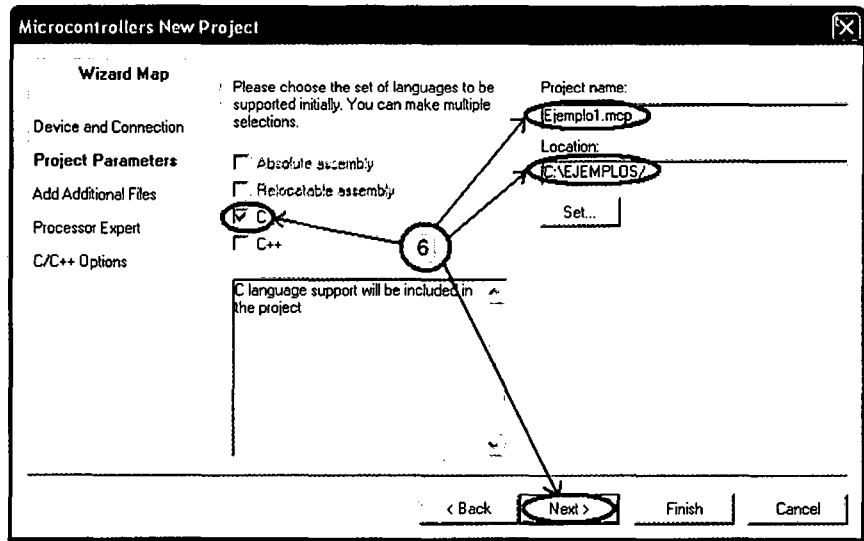


Fig. 4.48: Selección del lenguaje y otros.

Paso 5. Opción de adición de archivos: Si se tiene librerías que agregar se indica la ubicación y se presiona Add, sino se presiona el botón de Next.

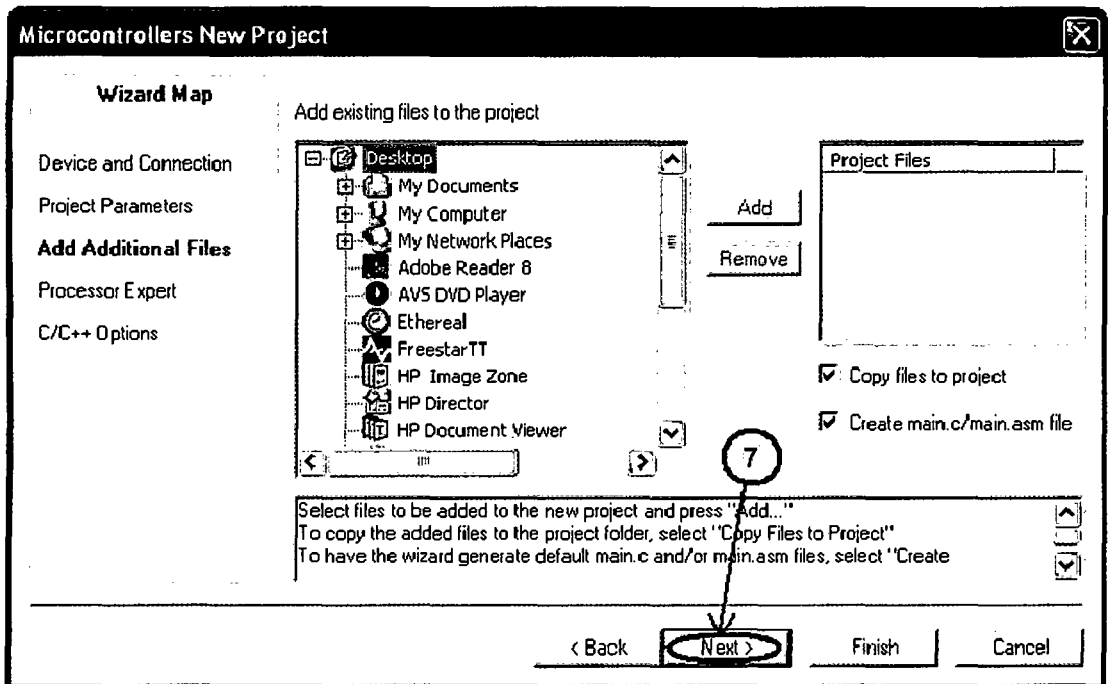


Fig. 4.49: Ventana para agregar librerías y otros.

Paso 6. Opción de Processor Expert: Esta opción de ventana no es de interés para el proyecto, por este motivo se sobrepasa. Seleccionar la opción none y presionar el botón de Next.

Paso 7. Opciones de configuración para el C/C++: Esta opción de ventana no es de interés para el proyecto, por este motivo se sobrepasa. Se presiona el botón Finish.

Finalmente, aparece la ventana del menú principal del CodeWarrior® 6.2 Development Studio y la ventana de navegación del proyecto (ver Figura 4.50).

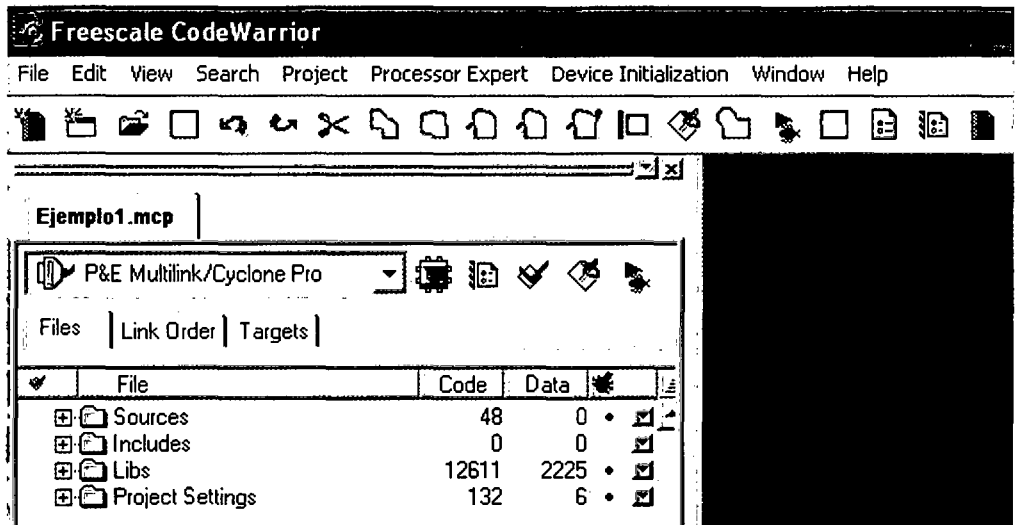


Fig. 4.50: Ventana principal del CodeWarrior® 6.2 Development Studio

Dentro de la carpeta sources se encuentra el programa principal main, haciendo doble click sobre el programa main.c aparece el cuerpo inicial de cualquier proyecto, que se inicie desde cero y cuyo código es mostrado en la Figura 4.51.

Este cuerpo sólo incluye:

- **hidef.h**: Macro para el proceso de interrupciones.
- **derivative.h**: Declaración de los periféricos de MCU.
- **Función main**: Programa principal del cuerpo en C.

Aquí es ejecutado el macro `enable interrupts`, que consiste en poner la máscara I en ceros, con el propósito de habilitar de manera global las interrupciones del sistema. Dentro de la función `main`, se espera que el programador incluya su código. También es ejecutada una instrucción de `for` infinito, que puede ser utilizada como iteración principal del proceso y que es necesaria desde el punto de vista de la acción cíclica que ejecuta un proceso desarrollado con un MCU. Dentro del `for` se ejecuta un macro llamado `_RESET_WATCHDOG`, con el objetivo de eliminar la acción del COP y aislar la MCU de un evento de `RESET` por sobreflujo del temporizador del COP.



```

main.c
Path: C:\Ejemplo1\Sources\main.c

#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */

void main(void) {
    EnableInterrupts; /* enable interrupts */
    /* include your code here */

    for(;;) {
        __RESET_WATCHDOG(); /* feeds the dog */
    } /* loop forever */
    /* please make sure that you never leave main */
}
Line 1 Col 1

```

Fig. 4.51: Programa inicial por defecto

## 4.9.2 Diagramas de Flujo de la programación del control del motor DC

### 4.9.2.1 El programa principal

En resumen el programa principal habilita todas las interrupciones, inicializa el display LCD, va explorando de la columna 1 a la columna 3 para identificar que tecla se pulsa del teclado matricial.

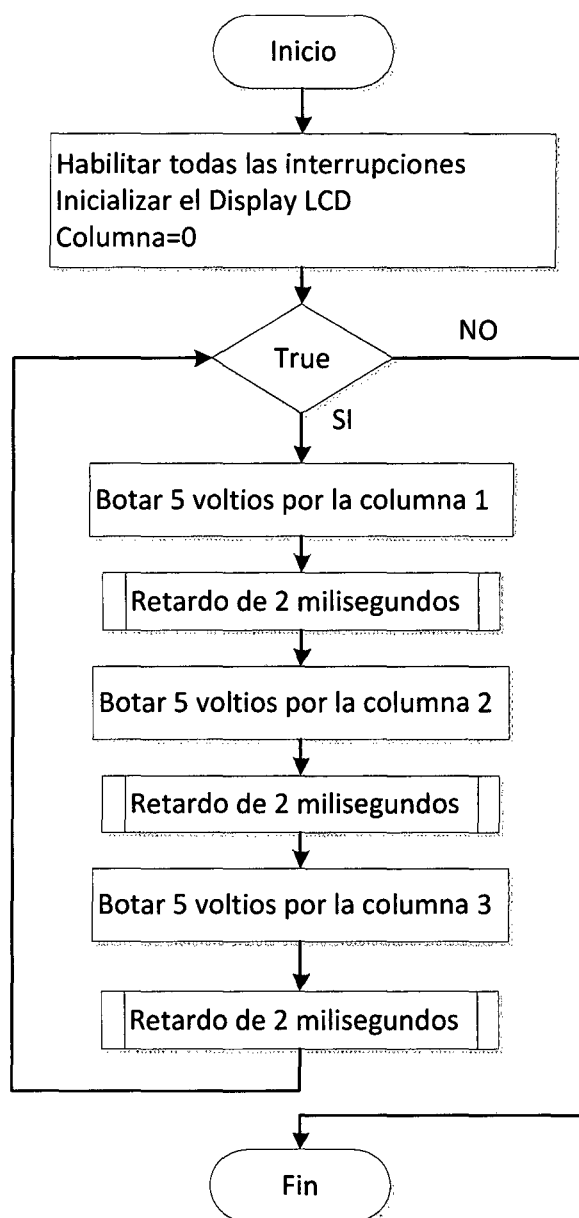


Fig. 4.52: Diagrama de flujo del Programa principal.

Las demás operaciones son interrupciones:

- Si se presiona alguna tecla del teclado.
- Para medir la frecuencia de los pulsos del encoder.

- Para actualizar el PWM.
- Si se cumple los 50mseg del tiempo de muestreo.

#### 4.9.2.2 Interrupción por teclado KBI2.

Cuando se presiona una tecla del teclado hexadecimal, se produce la interrupción, en ese momento el microcontrolador atiende a esa interrupción según el diagrama de flujo de la figura 4.55. El objetivo es saber entre coordenada (fila, columna) se produjo la interrupción, para identificar el número correspondiente de acuerdo al teclado matricial con el que se trabaje.

A los números del 0 al 9 se les asigna el mismo valor, pero a las teclas #(numeral) o \*(asterisco) se le asigna el valor de 15, ya que esas teclas sirven para confirmar la velocidad deseada.

**Proceso de validación y displayado:** Si el número ingresado es de 0 a 9, entonces se muestra en el display y ese valor se convierte en la cifra de las unidades, la siguiente vez que se ingrese otro valor, el número anterior pasará a ser la cifra de las decenas y así sucesivamente las tres cifras, mientras que no se pulse # o \* ningún número visto en el display será válido.

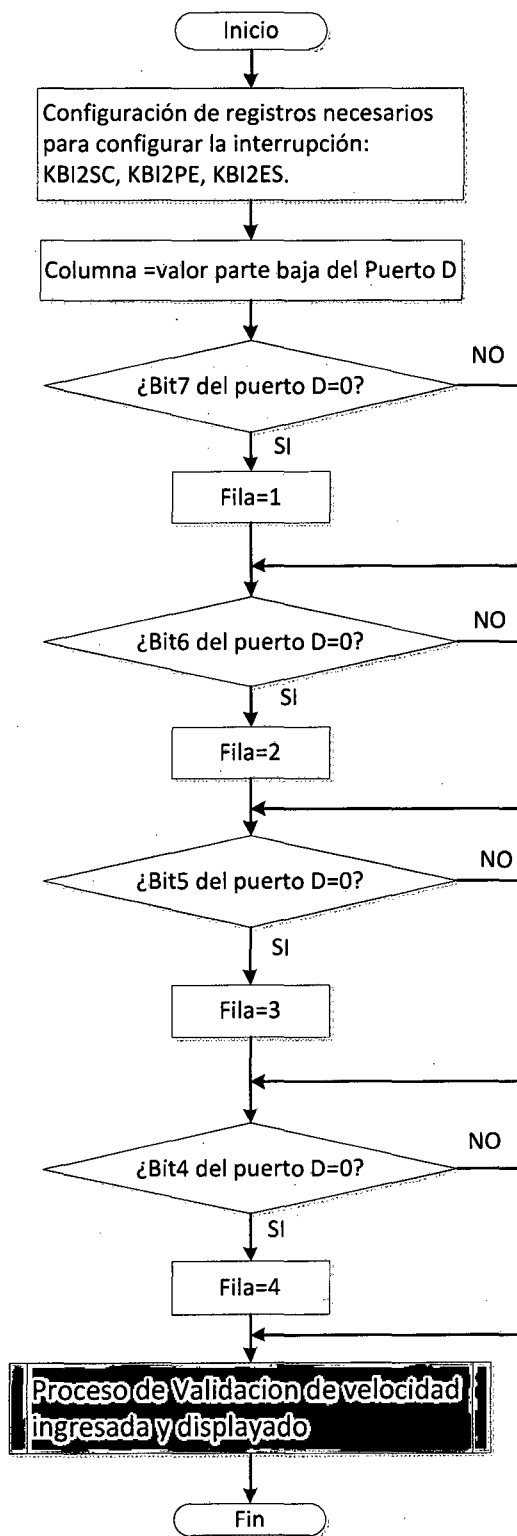


Fig. 4.53: Diagrama de flujo de la atención a la interrupción por teclado.

Si se pulsa las teclas de validación, se toma en cuenta que la velocidad deseada no debe ser mayor a la velocidad nominal del motor en este caso 500 RPM.

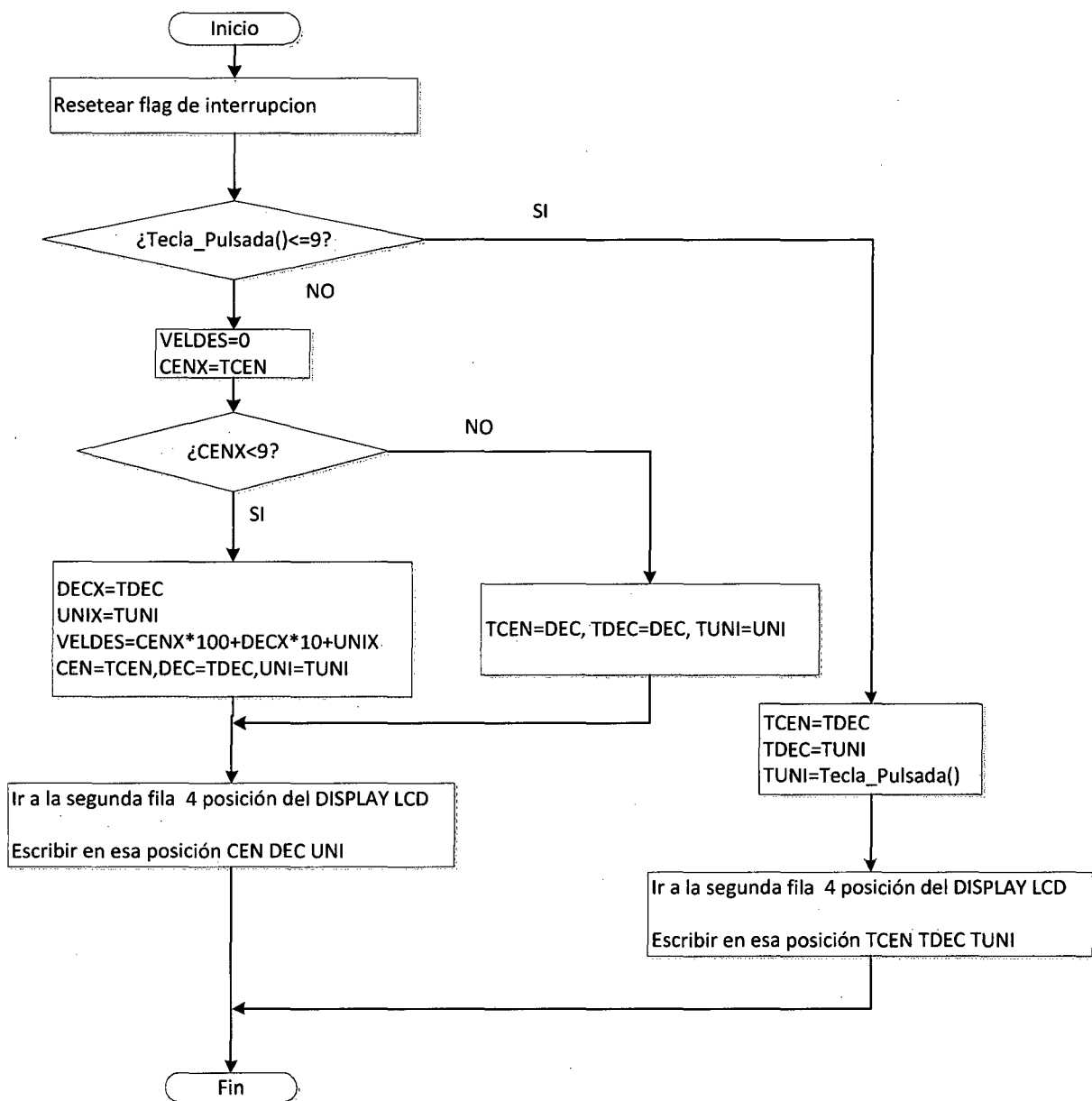


Fig. 4.54: Diagrama de flujo del Proceso de Validación de velocidad ingresada.

#### 4.9.2.3 Interrupción por captura de datos (Conteo de pulsos del encoder).

Para la medición de la velocidad se usa el canal 0 del módulo de temporización TPM2 del microcontrolador, se configura el modulo en modo Input Capture y cada vez que el encoder bote un pulso ocurre una interrupción. El procedimiento a esta interrupción es resetear el flag y la cantidad de pulsos aumenta en 1.

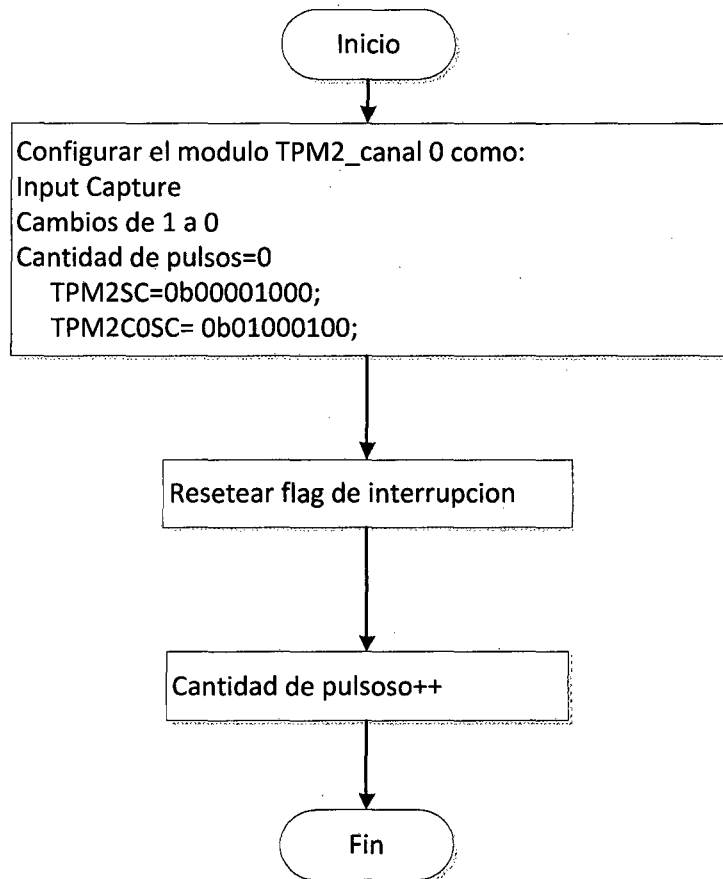


Fig. 4.55: Inicialización y atención a la interrupción por Input Capture.

#### 4.9.2.4 Interrupción por PWM.

En esta parte se configura el canal 0 del módulo TPM3 del microcontrolador como PWM alineado al flanco, se configura un período de PWM de 2.5mseg, al final de este tiempo ocurre la interrupción. El procedimiento para la atención a esta interrupción es actualizar el duty cycle de acuerdo al Algoritmo de control.

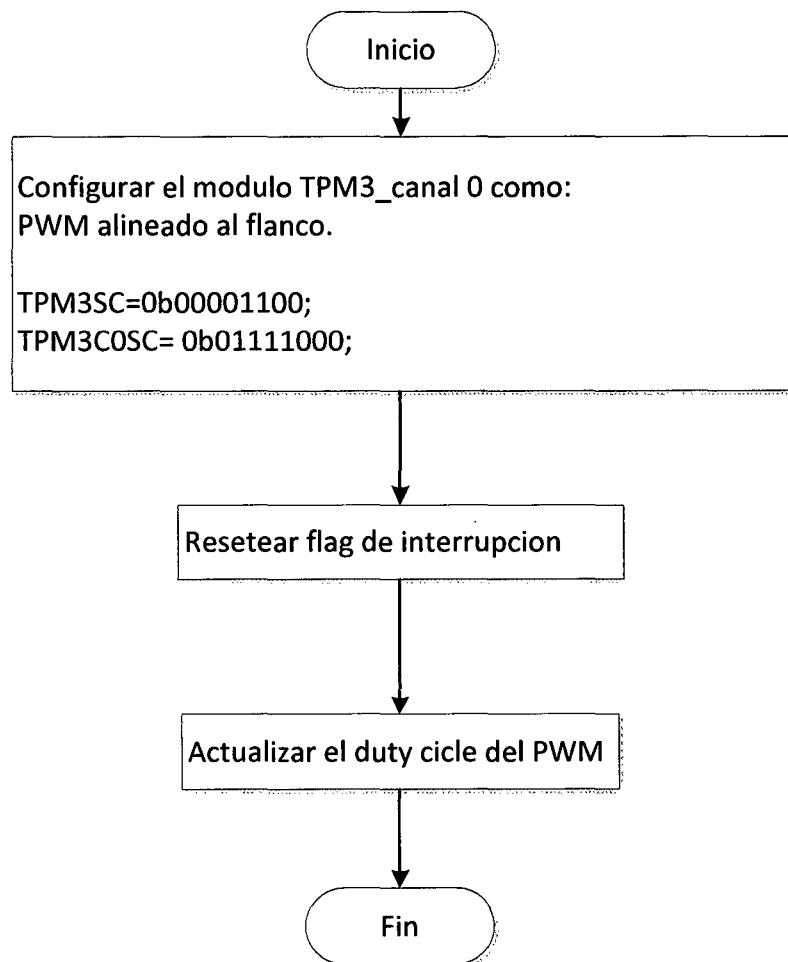


Fig. 4.56: Inicialización y atención a la interrupción por PWM.

#### 4.9.2.5 Interrupción por desbordamiento del módulo TPM1

Este módulo está configurado a un tiempo de 50 mseg. Pasado este tiempo se produce una interrupción. El procedimiento a esta interrupción es:

- ✓ Se calcula la velocidad real del motor en RPM: El factor 12 es debido a la resolución y al tiempo de muestreo de 50 mseg.
- ✓ Se resetea el contador de pulsos: El contador de pulsos sirve para medir la velocidad del motor en RPM.
- ✓ Se calcula el nuevo duty cycle del PWM: De acuerdo al algoritmo de control se calcula el nuevo duty cycle del PWM.
- ✓ Se muestra en el display la velocidad real.



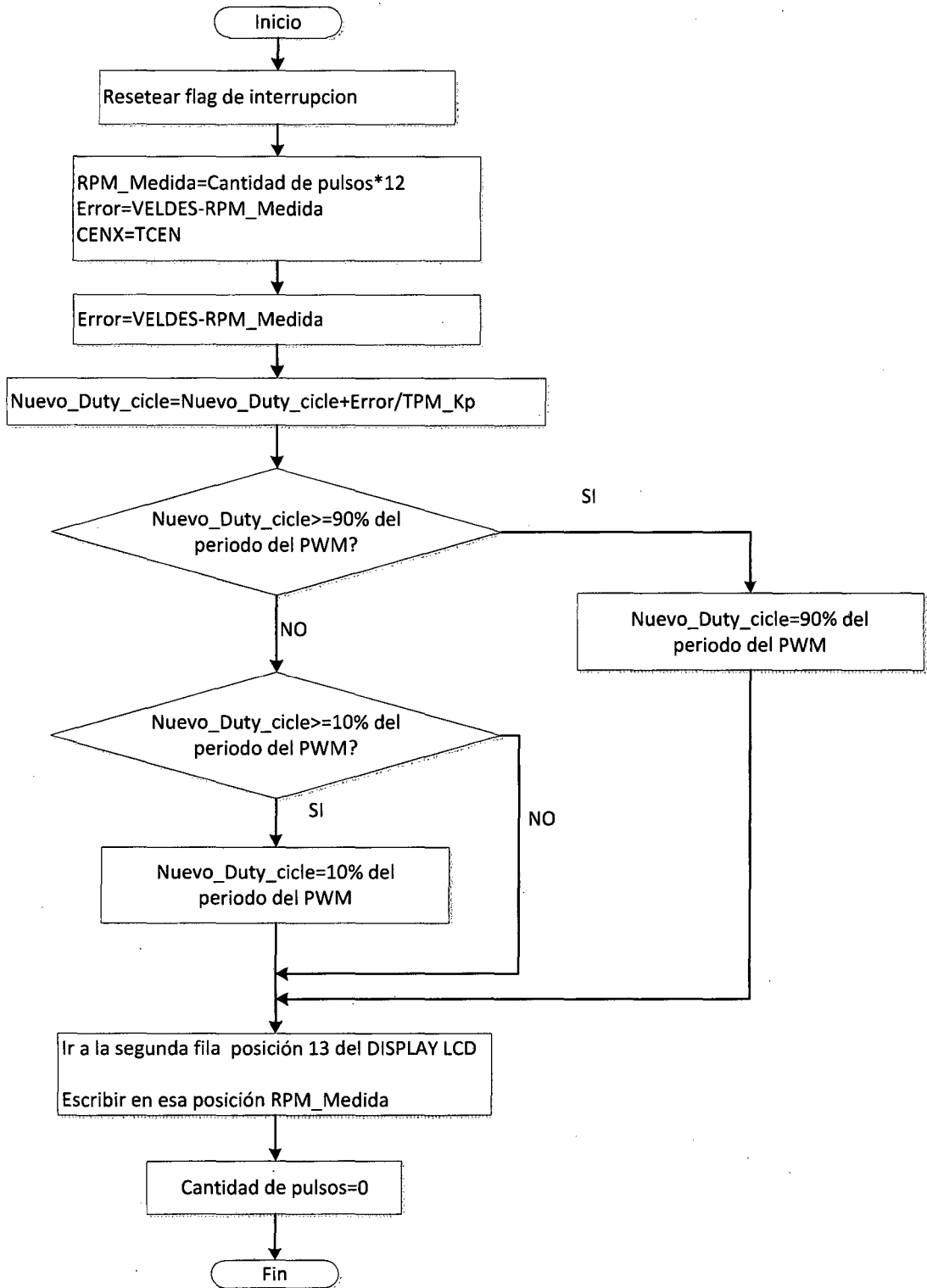


Fig. 4.57: Atención a la interrupción por desbordamiento del TPM1.

### 4.9.3 Depuración de proyectos en C

En este capítulo realizaremos una descripción del trabajo con un programa en C, donde se incluye:


- Trabajo sobre el editor IDE.
- Proceso de compilación.
- Simulación del proyecto.
- Bajar el programa al DEMOQE.

• **Compilación:** Para compilar el proyecto se debe presionar el botón .

. El proyecto no debe generar errores, pero en caso de haberlos es necesario eliminarlos verificando la sintaxis del programa.

• **La Depuración:** Una vez que el proyecto se ha compilado satisfactoriamente, se procede a realizar su depuración. El código fuente es enviado al microcontrolador y comienza su ejecución. Para supervisar el comportamiento del programa es posible ejecutarlo paso a paso o de manera normal, estableciendo puntos de ruptura (Break Points).

Para iniciar la depuración es necesario ir al menú Project\_Debug,

o bien de forma rápida mediante el botón . El usuario puede seleccionar entre:

➤ **Full Chip Simulation:** En la cual no se requiere de una conexión al microcontrolador (simulación en frío) y toda la depuración se realiza en el mismo PC, sin embargo solo es recomendable para secciones de código en las cuales solo interfieren datos y no hardware externo.

➤ P&E Multilink/Cyclone Pro: El código máquina es programado en el microcontrolador y su ejecución se realizara directamente en la máquina final.

Esta es la opción recomendada, porque permite el 100% de interacción con el hardware. La simulación es completamente real y la lectura y escritura sobre los pines del microcontrolador se realizan de la misma forma como se ejecutará una vez el sistema de evaluación sea desconectado del PC.

El menú de botones para la depuración es como sigue:



**Run:** Realiza la ejecución desde la función main(), en el caso de una depuración con conexión (in circuit) la ejecución en el microcontrolador se hará en tiempo real y de forma continua. El procesamiento puede ser parado en cualquier momento mediante el botón (halt) .



**Single Step:** Realiza la ejecución de una línea simple de C. En el caso del llamado a una función, lo realiza y se posiciona en la primera línea de la función.

Este botón es útil para evaluar el comportamiento de algún procedimiento de forma detallada.



**Step-Over:** Realiza la ejecución de una línea de C, sin embargo si la línea corresponde a un llamado de función, esta será

invocada y su ejecución no será intervenida. La función se ejecutará en tiempo real y el PC detiene su ejecución al retornar de la misma.



**Step-Out:** Realiza ejecución hasta que el procesador abandone el procedimiento o función actual. Es útil cuando se ingresa a ejecución detallada



**Assembly-Step:** Ejecuta la instrucción en assembler que está siendo apuntada por el contador de programa (PC). Este botón es útil cuando se requiere conocer el comportamiento del software a nivel de assembler y sus efectos sobre las variables, la memoria y los puertos de entrada y salida.



**Halt:** Obliga al microcontrolador a detenerse en la posición en la que se encuentra en contador de programa (PC) y actualizar las sub-ventanas del depurador.



**Reset:** Genera un reset al microcontrolador, obligando al contador de programa (PC) a posicionarse en la función de inicio Startup () o bien a la apuntada por su vector de RESET (0x0000\_0004).

Para proceder a la depuración es necesario establecer la conexión de la Figura 4.58. El PC deberá instalar automáticamente el hardware del DEMOQE y una vez reconocido seguir los pasos que se detallan a continuación:

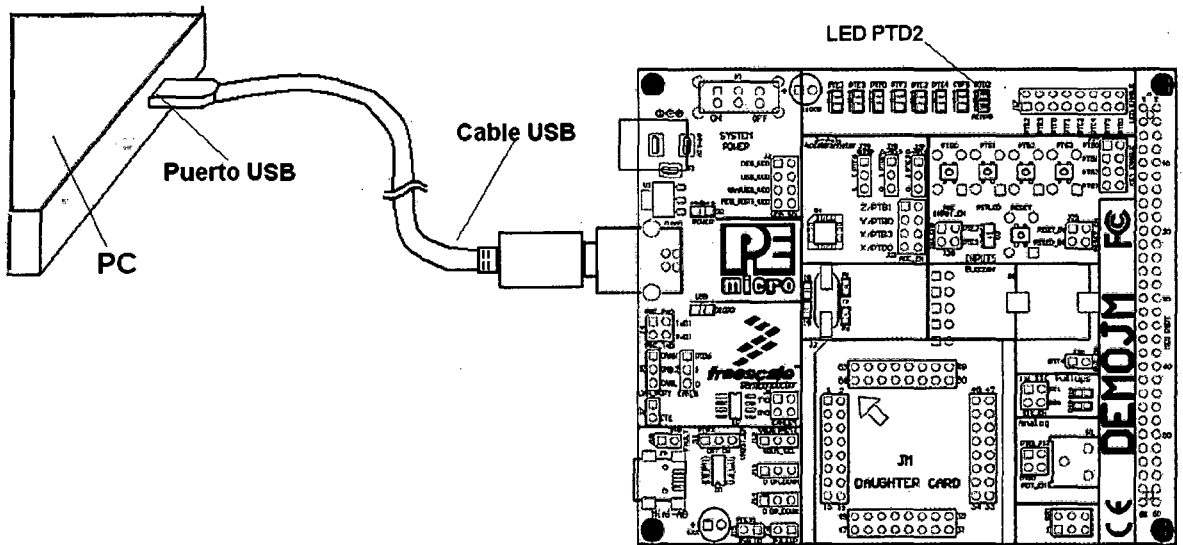


Fig. 4.58: Conexión DEMOQE – PC

**Paso 1:** En el menú principal del CodeWarrior® 6.2 hacer la selección de la opción P&E Multilink/Cyclone Pro, que se encuentra en la ventana de navegación del proyecto Ejemplo1.mcp.

**Paso 2:** Presionar el botón, de esta manera el programa entrará al depurador y la ventana de la Figura 4.59 mostrará la configuración de la conexión establecida.

**Paso 3:** Presionar el botón Connect (Reset) para establecer la conexión con el depurador. La Figura 4.60 muestra una segunda ventana como ingreso al depurador. Esta ventana indica que se va a borrar la FLASH del microcontrolador y ante esto se debe elegir la opción Yes.

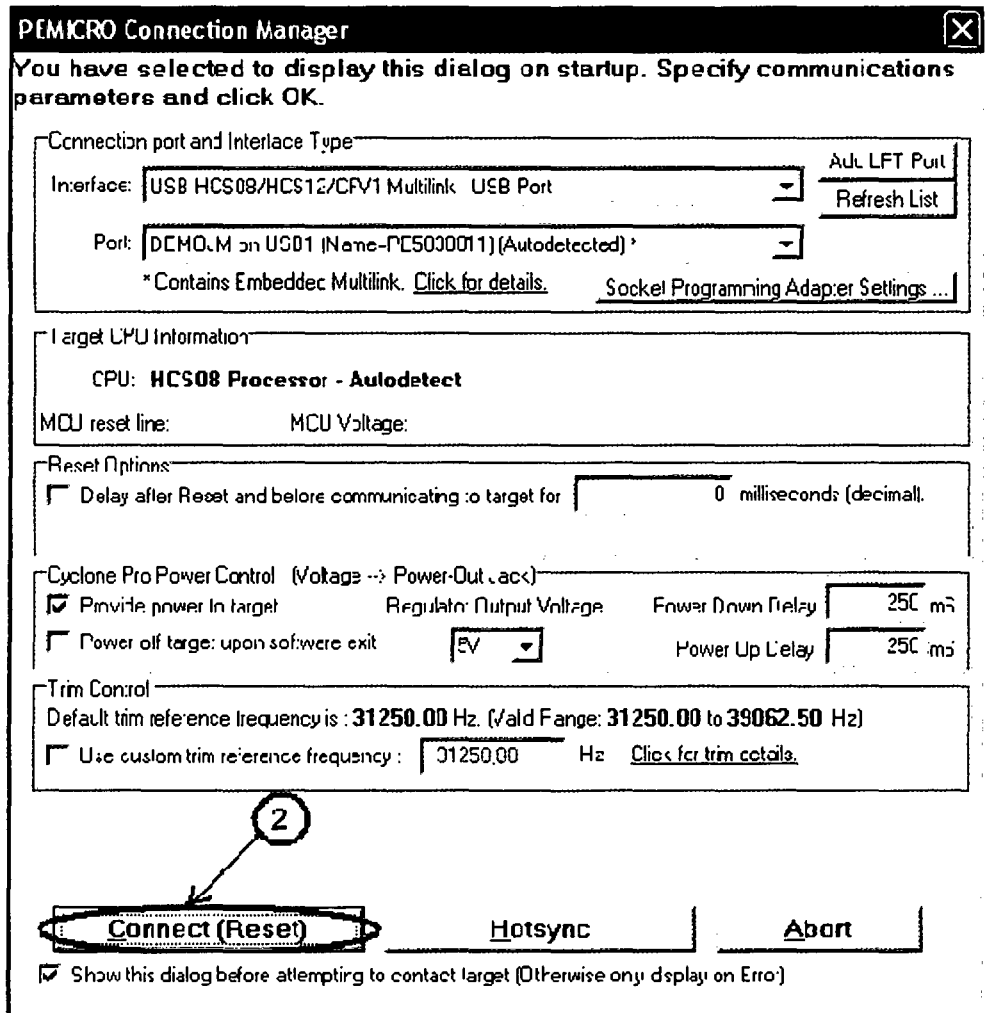


Fig. 4.59: Ventana conexión con el DEMOJM

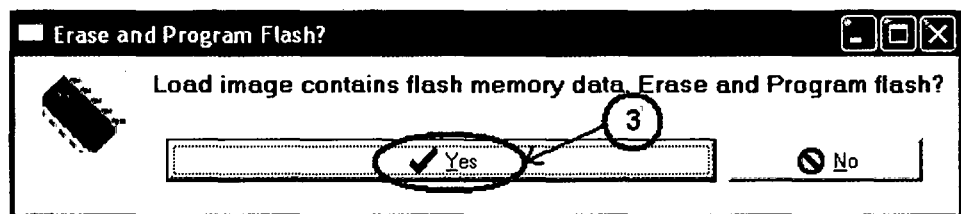


Fig. 4.60 Ventana para borrado de la FLASH

**Paso 4:** Poner en RUN el sistema, pulsando el ícono y verificar la activación y desactivación intermitente del LED, a una frecuencia aproximada de 12Hz, reportando un reloj de bus interno de

aproximadamente 16 MHz. Si el usuario desea, podría salir de la ventana del depurador y verificar que el sistema quedó programado y listo para funcionar desde RESET.

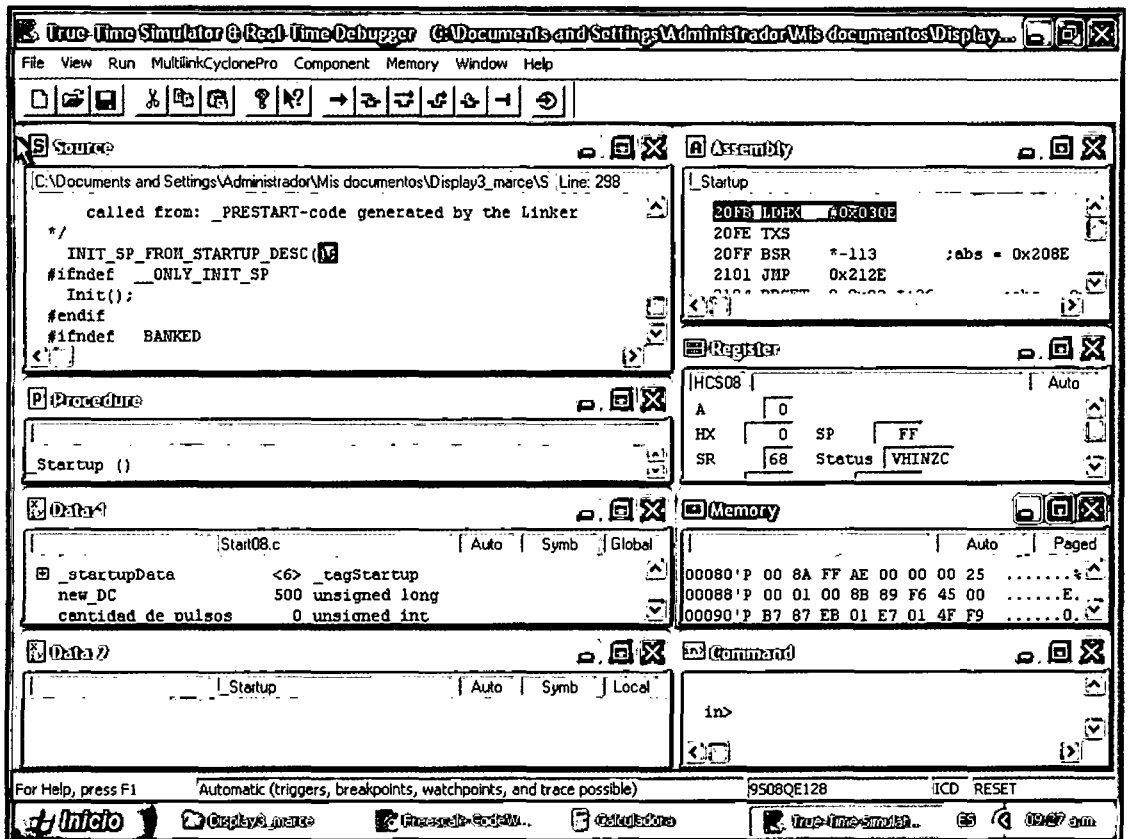


Fig. 4.61: Ventana para depurado del programa

## **CAPITULO 5**

### **RESULTADO DE LAS SIMULACIONES**

#### **5.1 SIMULACIÓN DEL ALGORITMO DE CONTROL DEL MOTOR**

El algoritmo de control detallado en el Capítulo 2 se implementó en un inicio en un microcontrolador MICROCHIP PIC16F877A, para poder simularlo en el software PROTEUS, y esa es una de las pocas desventajas de los microcontroladores Freescale, que los programas aún no se pueden simular en PROTEUS. Las siguientes figuras muestran imágenes de las simulaciones desarrolladas.



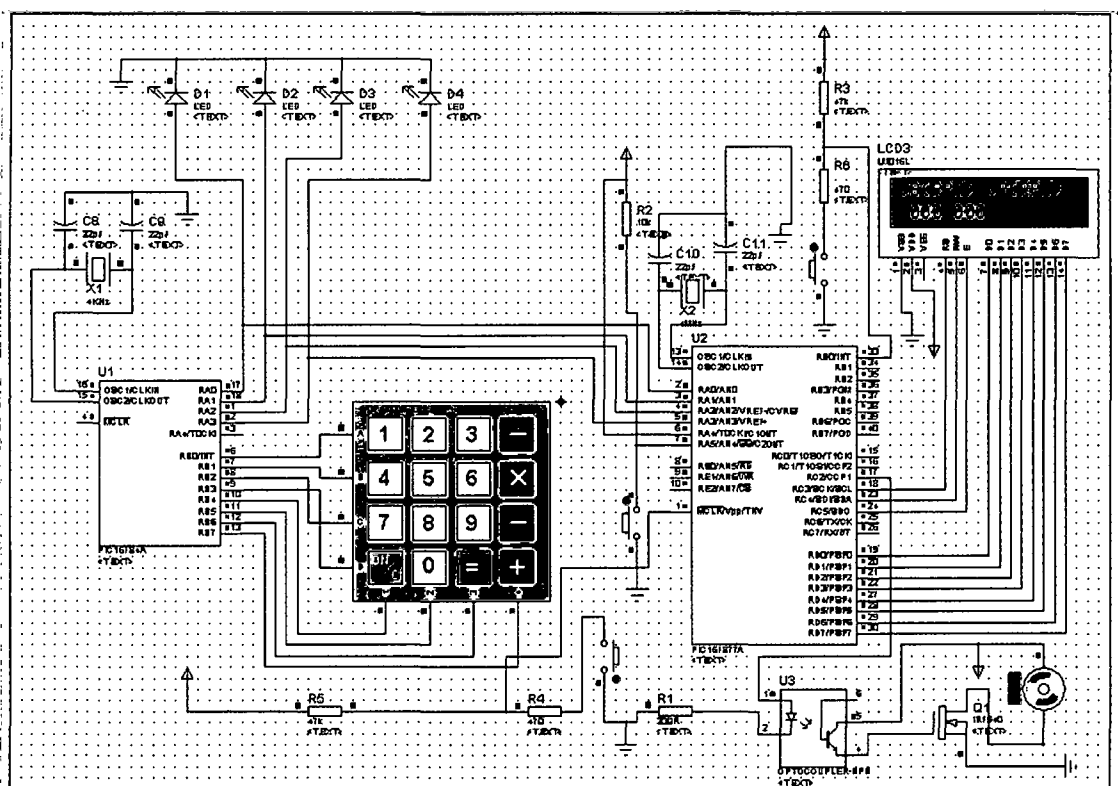


Fig. 5.1: Circuito de control del motor DC con microcontrolador **Microchip**.

La figura anterior muestra el circuito de control, que contiene todos los dispositivos necesarios para controlar el motor DC. El algoritmo de control está en el microcontrolador Microchip PIC 16F877A.

### 5.1.1 I Etapa: programación en Microchip en lenguaje assembler

Los resultados no fueron satisfactorios, sobretodo en el manejo del display LCD, como se muestra a continuación. El principal problema fue el manejo del display, y no se logró controlar el motor ya que la velocidad del motor era muy variable y no se estabilizaba.

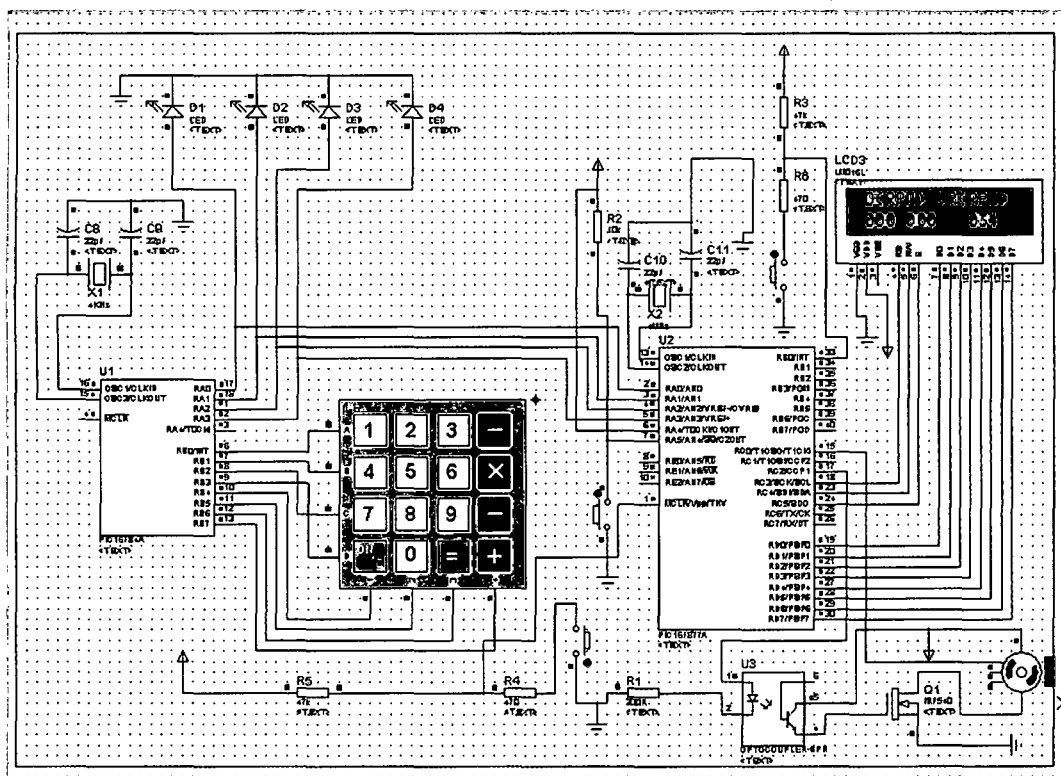


Fig. 5.2: Simulación del algoritmo de control con lenguaje Assembler para PIC16F877A

En esta etapa se usó un microcontrolador PIC16F84A para el control de teclado, la función que cumple este microcontrolador es identificar la tecla pulsada, convertirla de decimal a binario y mandarlo por el Puerto A del PIC16f877A. Para validar el ingreso de la velocidad deseada se utilizaron 2 interruptores. Toda la programación se realizó con el software MPLAB.

## 5.1.2 II Etapa: Programación en Microchip en lenguaje C

En esta etapa se programó el microcontrolador PIC16F877A de Microchip, en lenguaje C, con ayuda del software PICC. Se obtuvieron buenos resultados en la simulación, como se muestra en la figura siguiente.

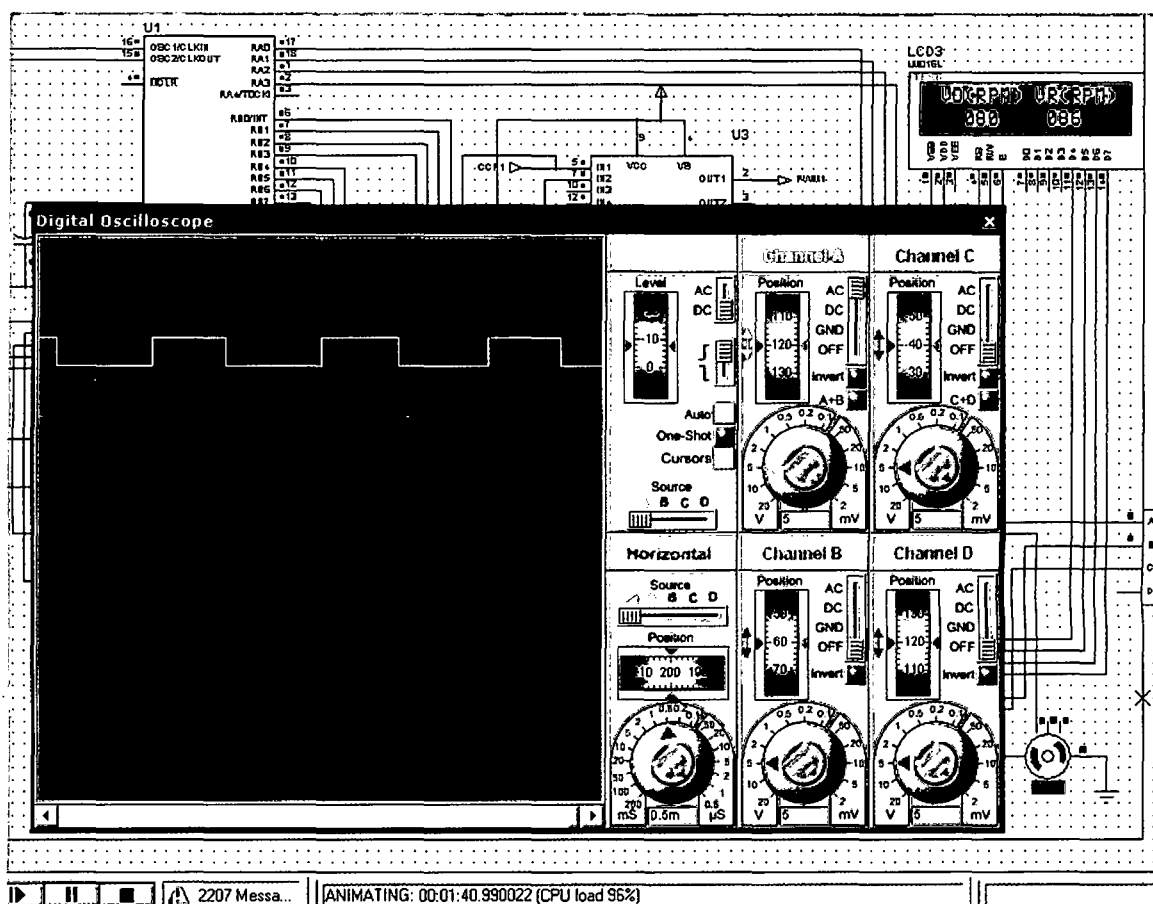


Fig. 5.3: Simulación de algoritmo de control con lenguaje C para PIC16F877A.

Como se puede apreciar en la Figura 5.3 la velocidad deseada es de 80 RPM y la velocidad real es de 86 RPM, y el PWM está en un 40% lo cual es correcto ya que la máxima velocidad es 300RPM.

Y las pruebas que se hicieron fue aumentar la velocidad deseada de 80RPM a 160RPM y también disminuyendo la velocidad deseada, por lo cual se demostró que el algoritmo funciona al 100%.

## **5.2 IMPLEMENTACIÓN DE LA TARJETA DE CONTROL**

### **5.2.1 I Etapa: Prototipo para uC Microchip PIC16F877A**

En esta etapa se realiza la implementación física del control del motor DC, del apartado anterior se tiene que el algoritmo de control programado en lenguaje C, funciona mejor que el programado en Assembler, entonces se procede al grabado del microcontrolador PIC16F877A con el software ICPROG. El circuito de la Figura 5.2 se lleva a una tarjeta electrónica, siendo esta el primer prototipo.

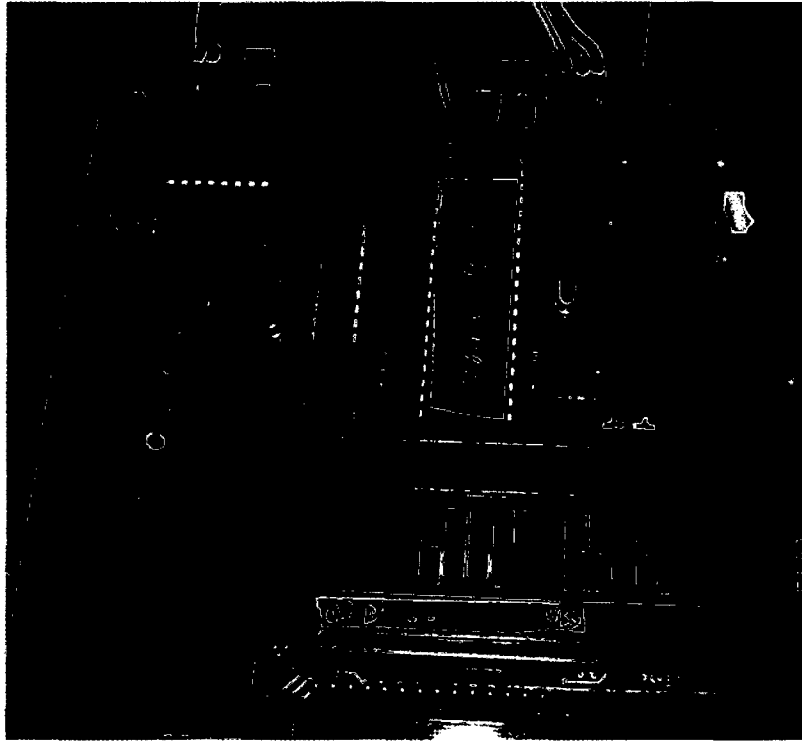


Fig. 5.4: Tarjeta electrónica para microcontrolador Microchip PIC16F877A

Como se aprecia en la figura anterior el resultado no fue el esperado, ya que el motor se estabilizaba a una velocidad, pero en el display mostraba que la velocidad deseada comparada con la real era muy diferente.

Además se pudo comprobar con ayuda de un osciloscopio que el PWM no funcionaba correctamente, ya que cuando se disminuía la velocidad deseada y el motor se estabilizaba, el duty cycle en vez de disminuir, aumentaba en algunos casos.

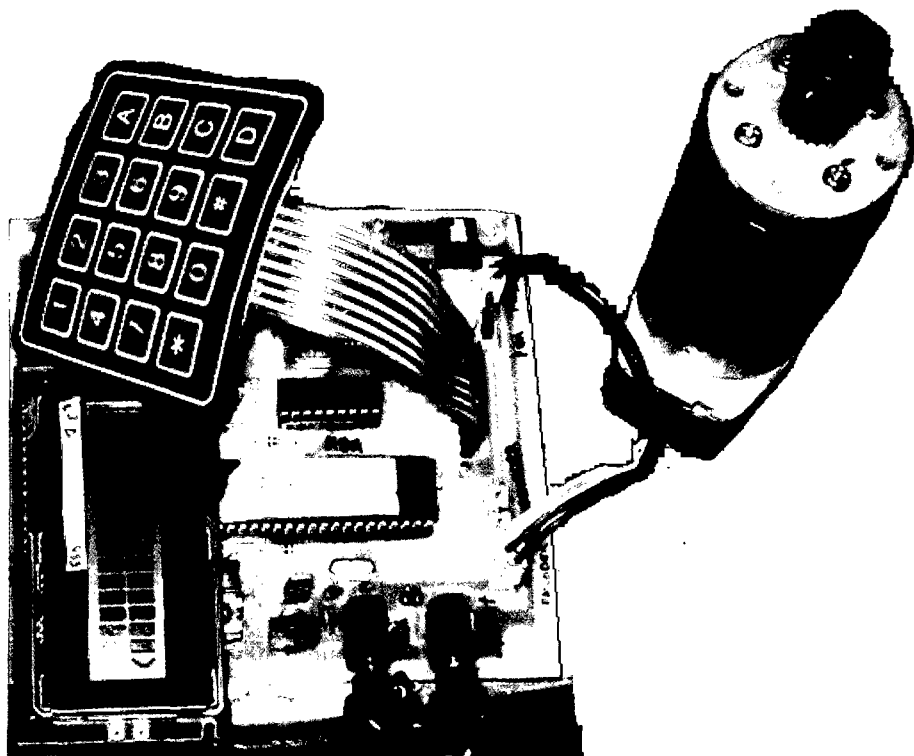


Fig. 5.5: Tarjeta prototipo para microcontrolador Microchip PIC16F877A

### **5.2.2 II Etapa: Prototipo para uC Freescale MC9S08QE128**

Al ver que los resultados no fueron los deseados y no se podía dar con el problema, se elige el microcontrolador Freescale por la capacidad que tiene para depurar online, y de esta manera se implementó el algoritmo que ya funciona en simulación y se eligió el lenguaje C. Finalmente se obtuvieron buenos resultados como se muestra a continuación,

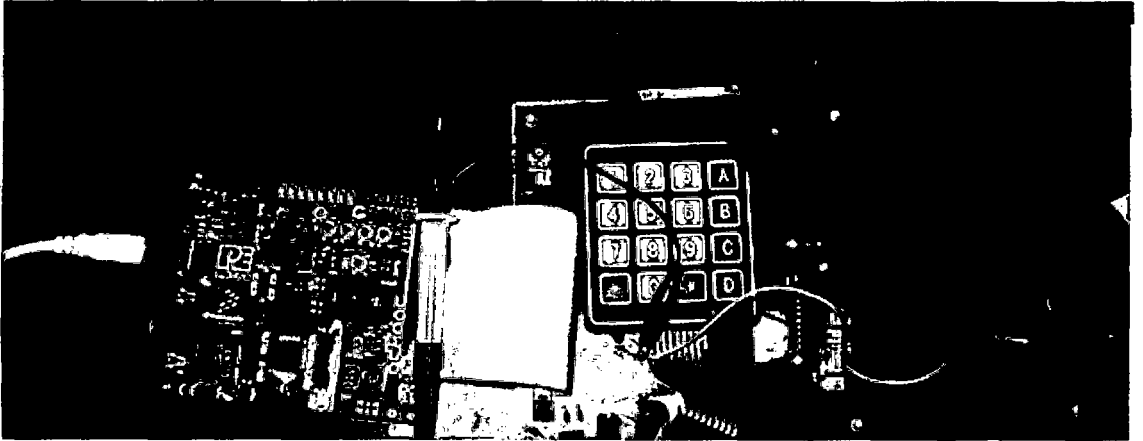


Fig. 5.6: Tarjeta prototipo para microcontrolador Freescale



Fig. 5.7: Circuito de control de motor DC con microcontrolador Freescale.

En la figura anterior se observa que la velocidad deseada es 500 y la velocidad real es 492, un error muy pequeño, lo importante es que el motor se mantiene estable una vez que llega a la velocidad deseada. Se comprobó con un osciloscopio que cuando se aumenta la velocidad el *duty cycle* también aumenta.

A diferencia del prototipo 1, en este prototipo la velocidad puede ser ingresada en cualquier momento, lo que es importante mencionar es que para validar la velocidad que aparece en el display se debe pulsar la tecla (#) o (\*). La velocidad será válida si está dentro del rango permitido, que lo da la velocidad máxima del motor.

Aquí se debe mencionar que en la máquina que se está diseñando se ingresa la velocidad lineal de avance y no las RPM del motor, esta velocidad lineal depende del espesor y del material que se desea cortar o soldar. Esto implica que en el código de programa se deben hacer operaciones de escalamiento, por eso en el capítulo 2 de este proyecto se hizo los cálculos de reducción de velocidad, así como el diseño de transmisión y reducción.

En el siguiente apartado se muestran las pruebas de medición de velocidad con un osciloscopio y también con un tacómetro. Las pruebas de PWM con ayuda de un osciloscopio, también se indican las velocidades deseadas y el *duty cycle* correspondiente. De esta manera se comprueba que finalmente se logró controlar la velocidad del motor DC.



### 5.3 MEDICIÓN DE LA VELOCIDAD DEL MOTOR

#### 5.3.1 Determinación de la velocidad máxima del motor empleado

Para medir las RPM del motor DC se utilizó un osciloscopio, se conecta el motor al 80% de su voltaje nominal y la salida de uno de los canales del encoder se conectó a un osciloscopio y se obtuvo la siguiente forma de onda.

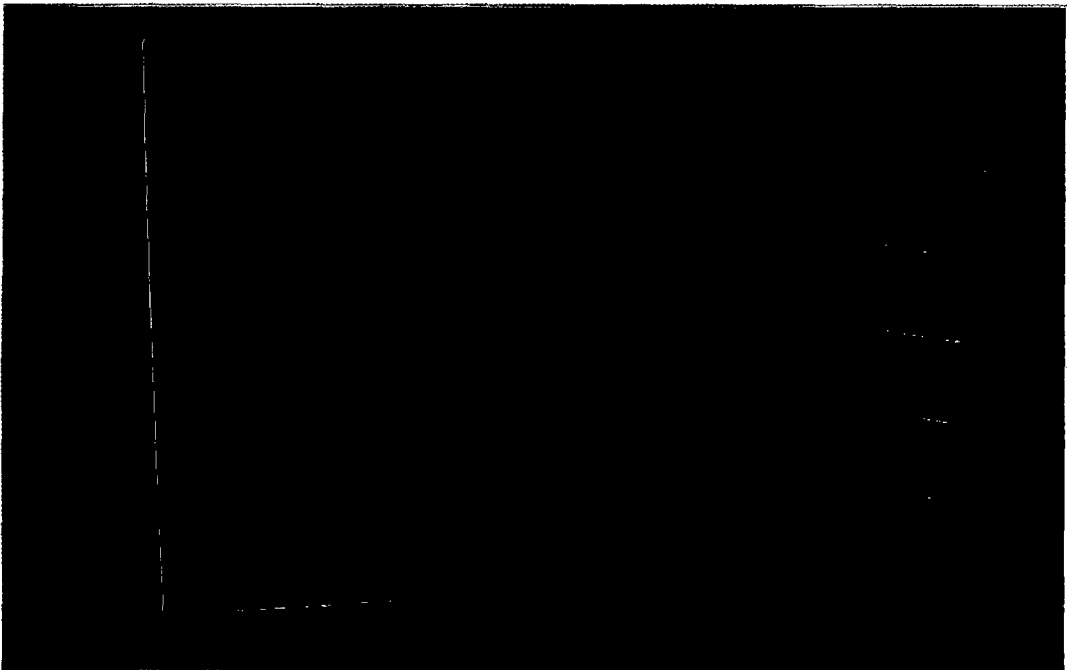


Fig. 5.8: Señal del canal A del encoder para una alimentación del 80% del motor DC.

De la figura 5.8, de la onda cuadrada se obtiene el período  $T=400\mu\text{seg}$ . Sabemos que la resolución del encoder es de  $N=100$  pulsos por minuto, el tiempo en dar una vuelta es  $t=N*T=100*T$ ;

Luego:

$$w = \frac{1}{100*T} \text{ rev/seg}$$

Donde  $w$ =velocidad angular.

Transformando a Revoluciones por minuto se tiene:

$$w = \frac{60}{100*T} \text{ RPM}$$

Reemplazando  $T=400\mu\text{seg}$  finalmente se tiene:

$$w = \frac{60*10^6}{100*400} \text{ RPM} = 1500\text{RPM}$$

Esta velocidad aparentemente es muy alta, para corroborar se utilizó un tacómetro, el procedimiento se muestra continuación:

### **5.3.2 Verificación de la lectura de velocidad con el uso de un tacómetro**

Usando un tacómetro también se determinó la máxima velocidad del motor como se muestran en las siguientes figuras:

En la siguiente figura se alimenta al motor con una tensión de 20.1 voltios y se obtiene 2019 RPM de salida.

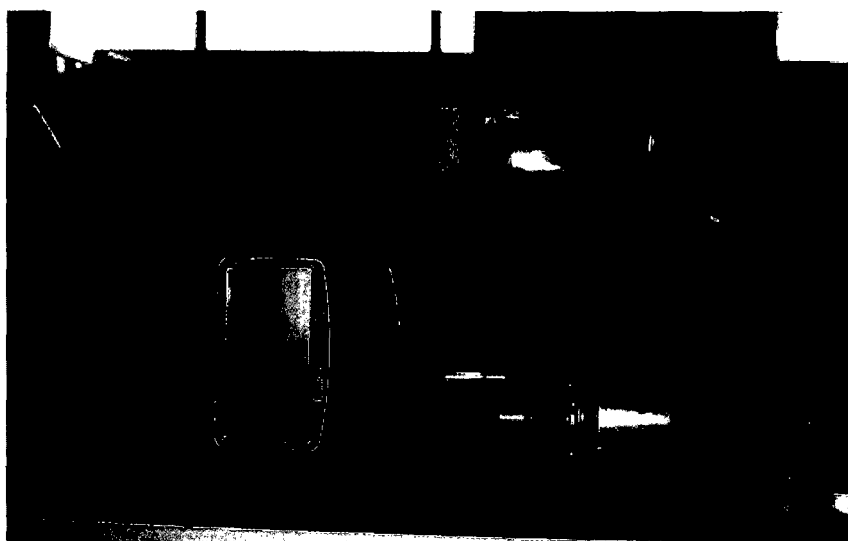


Fig. 5.9: Medición de la velocidad del motor a 20 voltios, con un tacómetro.

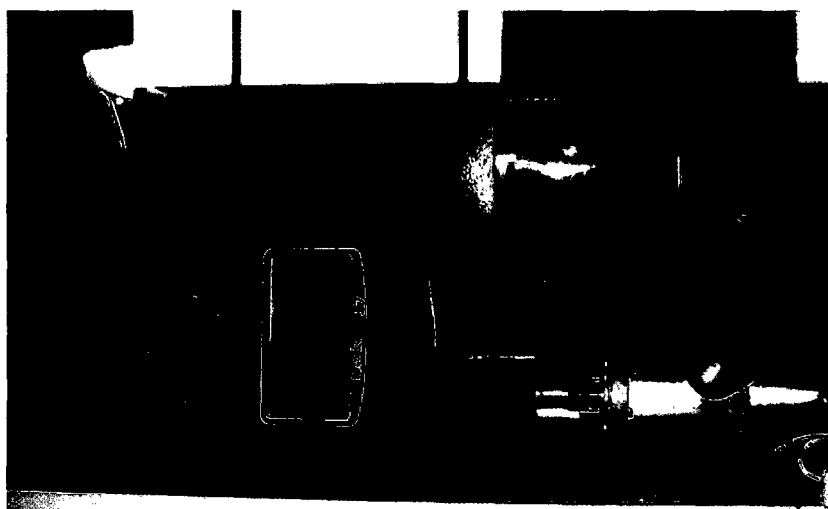


Fig. 5.10: Medición de la velocidad del motor a 24 voltios, con un tacómetro.

Con el procedimiento anterior se comprobó que la velocidad máxima del motor sin reducción era alta, alrededor de 2400RPM, por ser tan alta era difícil de controlar, la mayor dificultad estuvo en la lectura de velocidad.

#### 5.4 PRUEBA DEL PWM

El programa se probó para diferentes motores, es decir con diferentes velocidades máximas y diferentes resoluciones de encoder, en todos los casos funcionó bien. La velocidad de desplazamiento del motor es pequeña, por lo cual se requiere un motor que no tenga mucha velocidad.

A continuación se muestran las pruebas de PWM para diferentes velocidades deseadas a un motor cuya velocidad máxima era cercana a 800 RPM. Se observa que a medida que aumenta la velocidad deseada el duty cycle aumenta (como debe ser) y se estabiliza, con lo cual queda demostrado el funcionamiento del algoritmo de control.

Es necesario para afinar el programa, multiplicar a la velocidad real leída a través del encoder por un factor, para que la velocidad real que se muestra el display sea exactamente igual a la real.

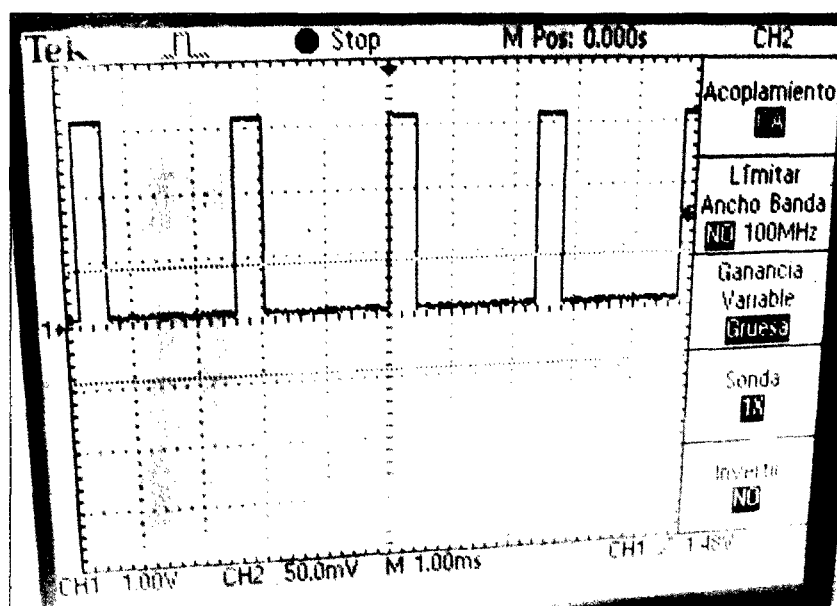


Fig. 5.11: Señal de PWM para una velocidad deseada de 150RPM.

En la figura anterior se tiene un *duty cycle* de 400/2500, es decir de 16%.

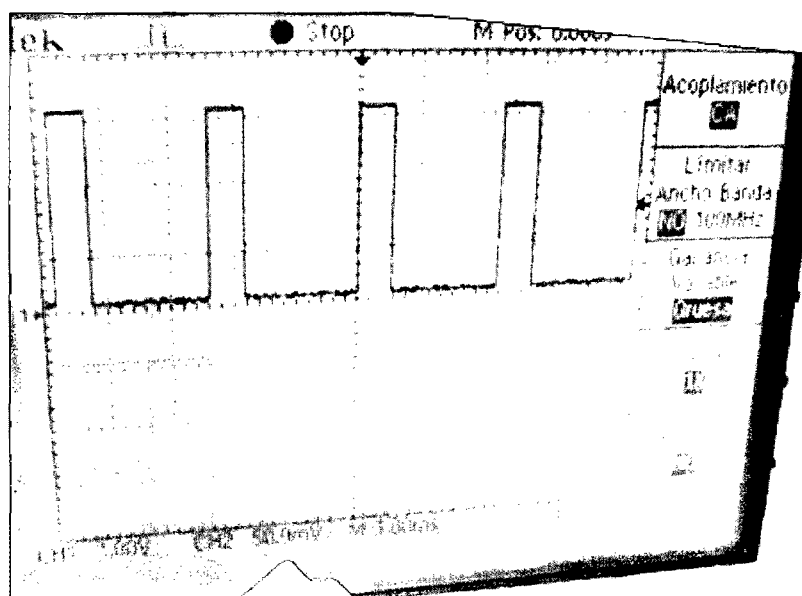


Fig. 5.12: Señal de PWM para una velocidad deseada de 250RPM.

En la figura anterior se tiene un *duty cycle* de 600/2500, es decir de 24%.

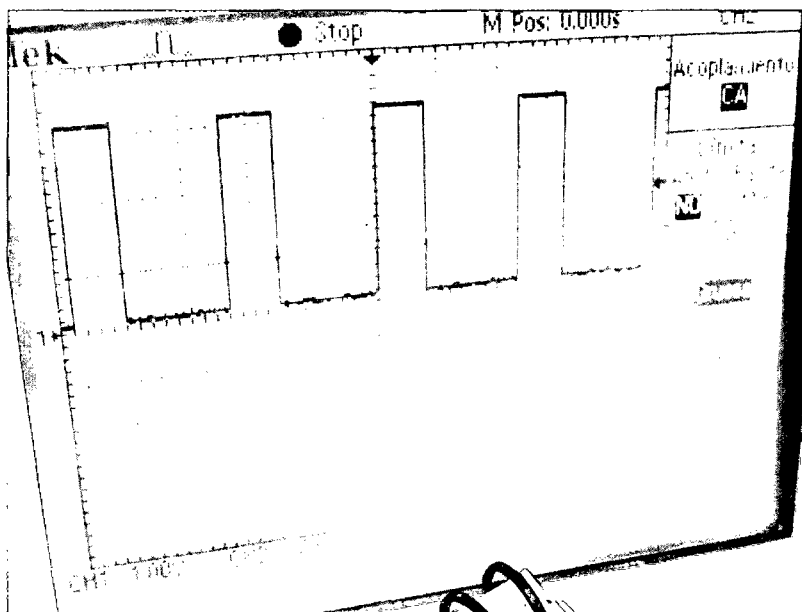


Fig. 5.13: Señal de PWM para una velocidad deseada de 350RPM.

En la figura anterior se tiene un *duty cycle* de 800/2500, es decir de 32%.

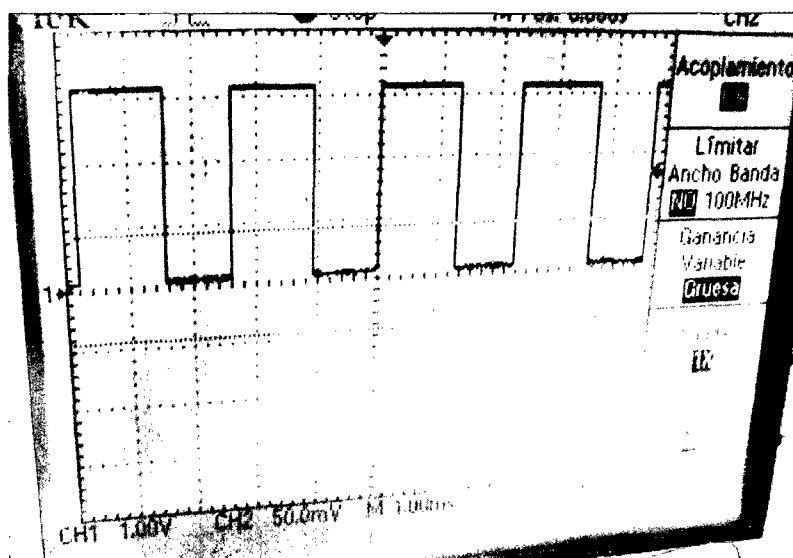


Fig. 5.14: Señal de PWM para una velocidad deseada de 450RPM.

En la figura anterior se tiene un *duty cycle* de 1200/2500, es decir de 48%.

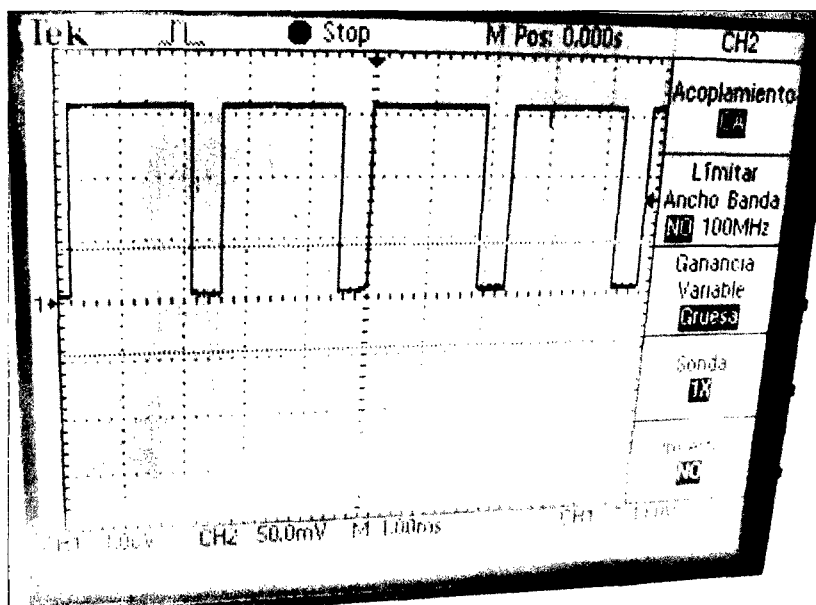


Fig. 5.15: Señal de PWM para una velocidad deseada de 500RPM.

En la figura anterior se tiene un *duty cycle* de 1800/2500, es decir de 72%. Al momento de programar el microcontrolador se programó un *duty cycle* máximo de 90% y un mínimo de 10%. Porque según las pruebas en simulación y con prototipo para microcontrolador Microchip ocurren problemas para un *duty cycle* mayor al 90%.

## CONCLUSIONES Y RECOMENDACIONES

Luego de las pruebas del control del motor DC, se puede concluir que se logró controlar la velocidad del motor con el microcontrolador Freescale. Gracias a la herramienta de depurador del Freescale se pudo determinar que el problema al trabajar con el microcontrolador Microchip fue la alta velocidad del motor con el que se hizo las pruebas, ya que en el trabajo que se tomó como referencia la velocidad máxima del motor de 300RPM; sin embargo el motor con el que se trabajó llegaba hasta 2400 rpm.

El presente trabajo contribuye a una mejora de la máquina, es decir, también se puede utilizar para la soldadura Oxi – fuel., aumentando a la máquina un soporte para las dos mangueras (la de oxígeno y la de acetileno).

Para trabajos con microcontroladores y varios dispositivos, se recomienda trabajar el microcontrolador con cada dispositivo, después de dos en dos y finalmente unir todo en un solo programa. También se recomienda que al momento de programar se creen funciones para controlar cada dispositivo, para tener mejores resultados.

El extremo de la máquina que se diseñó posee 4 grados de libertad, se recomienda que para el siguiente proyecto en base a este se considere usar un servomotor en el eslabón extremo para lograr un movimiento oscilante en el extremo de la antorcha. Este movimiento es necesario para realizar una mejor soldadura; pero no para el proceso de corte.

En la etapa de pruebas de la máquina, se recomienda hacer una tabla de manera que de acuerdo al tipo de material y el espesor se tenga una velocidad recomendada, según el proceso que se quiera; es decir si se quiere cortar o soldar. También se deberá elaborar un manual de operación y una relación de fallas comunes y como solucionar, esto se logrará al realizar las pruebas cuando la máquina ya este construida.



**BIBLIOGRAFÍA****[1] DISEÑO EN INGENIERÍA MECÁNICA DE SHIGLEY.****Richard G. Buddynast y J. Keith Nisbett**

Octava edición, editorial McGraw Hill, México.

**[2] PROGRAMACION DE SISTEMAS EMBEBIDOS EN C.****Gustavo Galeano**

Editorial Alfaomega Colombiana S.A., Bogotá 2009.

**[3] INGENIERIA DE CONTROL MODERNA.****Katsuhiko Ogata**

3ra edición, Prentice Hall.

**[4] MICROCONTROLADORES DE 32 BITS COLDFIRE V1/ FAMILIA JM.****Diego Alejandro Múnera Hoyos**

Revisión técnica: 0.1

**[5] DEMOQE128 User Manual.**

Freescale

**[6] MC9S08QE128RM Manual de referencia del microcontrolador.**

Freescale

Rev. 2 6/2007

**[7] TOOLBOX DE IDENTIFICACIÓN.**

M.Sc. Ricardo Rodríguez. Lima - Perú 2009. [www.gmail.com](mailto:www.gmail.com)

**Páginas Web**

**[8]** [www.freescale.com](http://www.freescale.com)

**[9]** <http://forums.freescale.com/>

**[10]** <http://www.x-robotics.com>

**[11]** <http://www.steeltailor.com>

**[12]** <http://www.bugo.com>

# ANEXOS

## ANEXO A

Módulos normalizados según las Normas DIN 780, ISO 54-1977 y BS 978.

| MODULOS (m) |          |           | DIAMETRAL PICHES (P) |          |
|-------------|----------|-----------|----------------------|----------|
| SERIE I     | SERIE II | SERIE III | SERIE I              | SERIE II |
| 0.12        |          |           | 200                  |          |
|             | 0.14     |           | 180                  |          |
| 0.16        |          |           | 160                  |          |
|             | 0.18     |           | 140                  |          |
| 0.2         |          |           | 120                  |          |
|             | 0.22     |           |                      |          |
| 0.25        |          |           | 100                  |          |
|             | 0.28     |           |                      |          |
| 0.3         |          |           | 80                   |          |
|             | 0.35     |           |                      |          |
| 0.4         |          |           | 64                   |          |
|             | 0.45     |           |                      |          |
| 0.5         |          |           |                      |          |
|             | 0.55     |           | 48                   |          |
| 0.6         |          |           |                      |          |
|             | 0.65     |           | 40                   |          |
| 0.7         |          |           | 36                   |          |
|             | 0.75     |           | 32                   |          |
| 0.8         |          |           |                      |          |
|             | 0.85     |           | 28                   |          |
| 0.9         |          |           |                      |          |
|             | 0.95     |           | 24                   |          |
| 1           |          |           | 20                   |          |
|             | 1.125    |           |                      | 18       |
| 1.25        |          |           | 16                   |          |
|             | 1.375    |           |                      | 14       |
| 1.5         |          |           | 12                   |          |
|             | 1.75     |           |                      | 11       |
| 2           |          |           | 10                   |          |
|             | 2.25     |           |                      | 9        |
| 2.5         |          |           | 8                    |          |
|             | 2.75     |           |                      | 7        |
| 3           |          | 3.25      | 6                    |          |
|             | 3.5      |           |                      | 5.5      |
| 4           |          | 3.75      | 5                    |          |
|             | 4.5      | 4.25      |                      | 4.5      |
| 5           |          | 4.75      | 4                    |          |
|             | 5.5      | 5.25      |                      | 3.5      |
| 6           |          | 5.75      | 3                    |          |
|             | 7        | 6.5       |                      | 2.75     |
| 8           |          |           | 2.5                  |          |
|             | 9        |           |                      | 2.25     |
| 10          |          |           | 2                    |          |
|             | 11       |           |                      | 1.75     |
| 12          |          |           | 1.5                  |          |
|             | 14       |           |                      |          |
| 16          |          |           | 1.25                 |          |
|             | 18       |           |                      |          |
| 20          |          |           | 1                    |          |
|             | 22       |           |                      | 0.875    |
| 25          |          | 27        | 0.75                 |          |
|             | 28       |           |                      |          |
| 32          |          | 30        | 0.625                |          |
|             | 36       |           |                      |          |
| 40          |          | 39        | 0.5                  |          |
|             | 45       | 42        |                      |          |
| 50          |          |           |                      |          |

Los módulos y Diametral Pitches destacados en **negrita** son los que figuran en la Norma ISO 54-1977, a los cuales se les debe dar preferencia, y dentro de ellos se procurará utilizar únicamente la **SERIE I**, estando orientada la **SERIE II** para casos especiales, y la **SERIE III** debe tratar de evitarse.

La norma internacional comenta que los Diametral Pitches se dan de forma provisional y que son a extinguir después del período necesario para permitir su conversión al sistema métrico.

El campo de aplicación de esta norma ISO son los engranajes rectos y helicoidales para ingeniería general e ingeniería pesada, excluyendo el campo de la automoción.

*ISO 54 fue desarrollado por el Comité Técnico ISO/TC 60, Engranajes.*

*DIN (Deutsches Institut für Normung e.V.)*

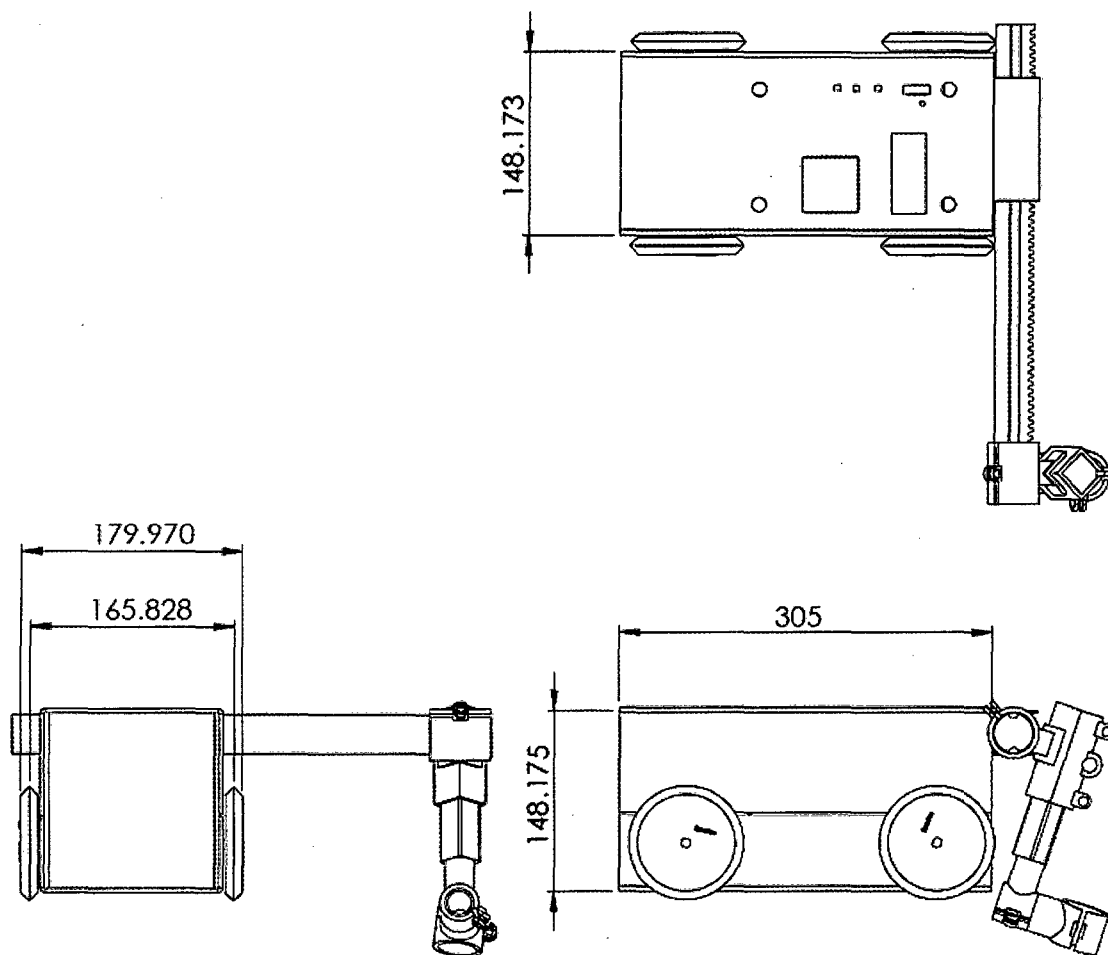
*ISO (the International Organization for Standardization)*

*BS (British Standards)*

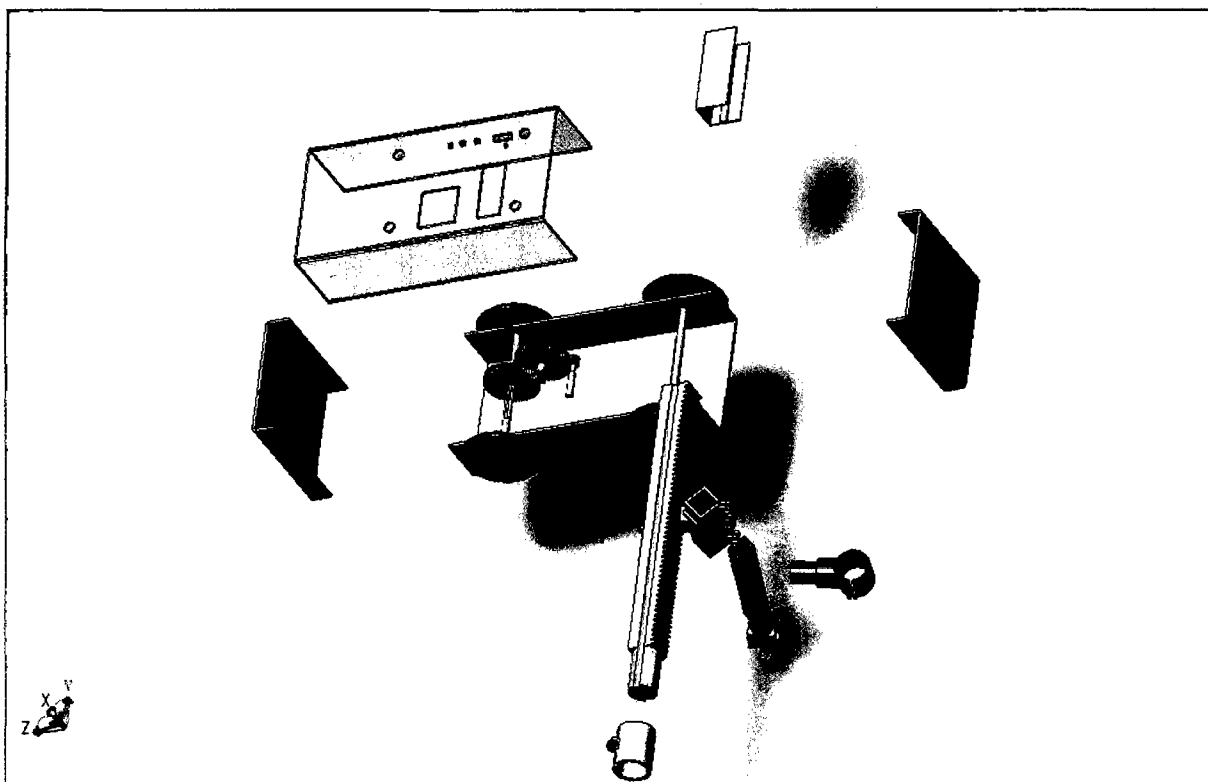
*TC (Technical Committee)*

## ANEXO B

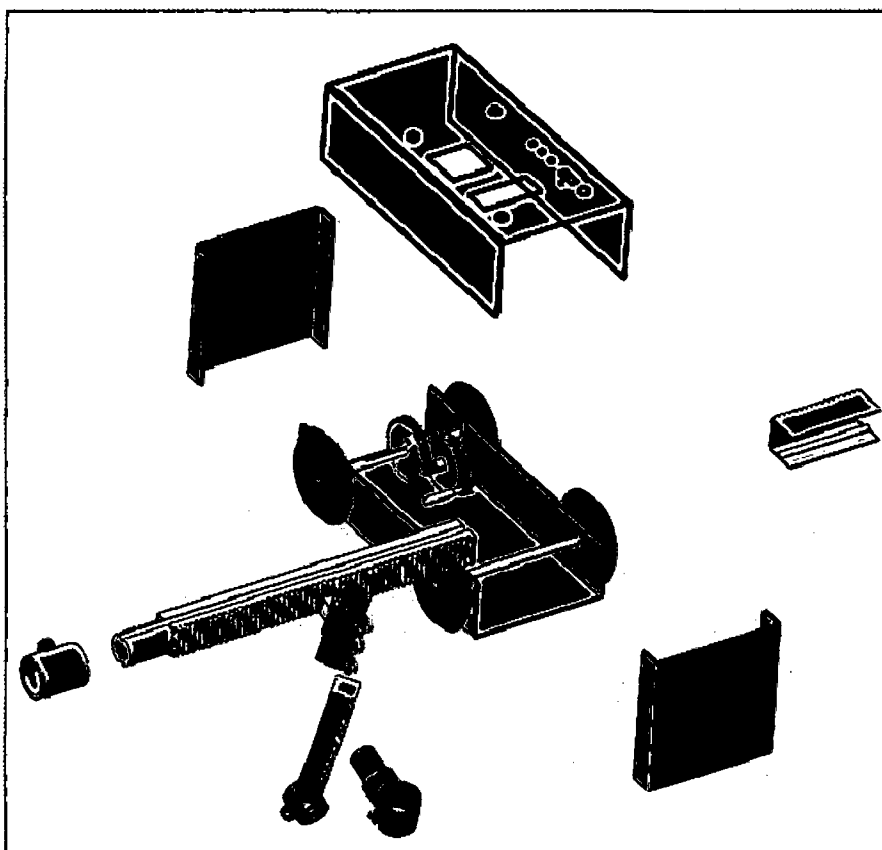
Vistas del diseño Mecánico de la máquina.



B.1: Vistas y dimensiones principales de la máquina en mm.

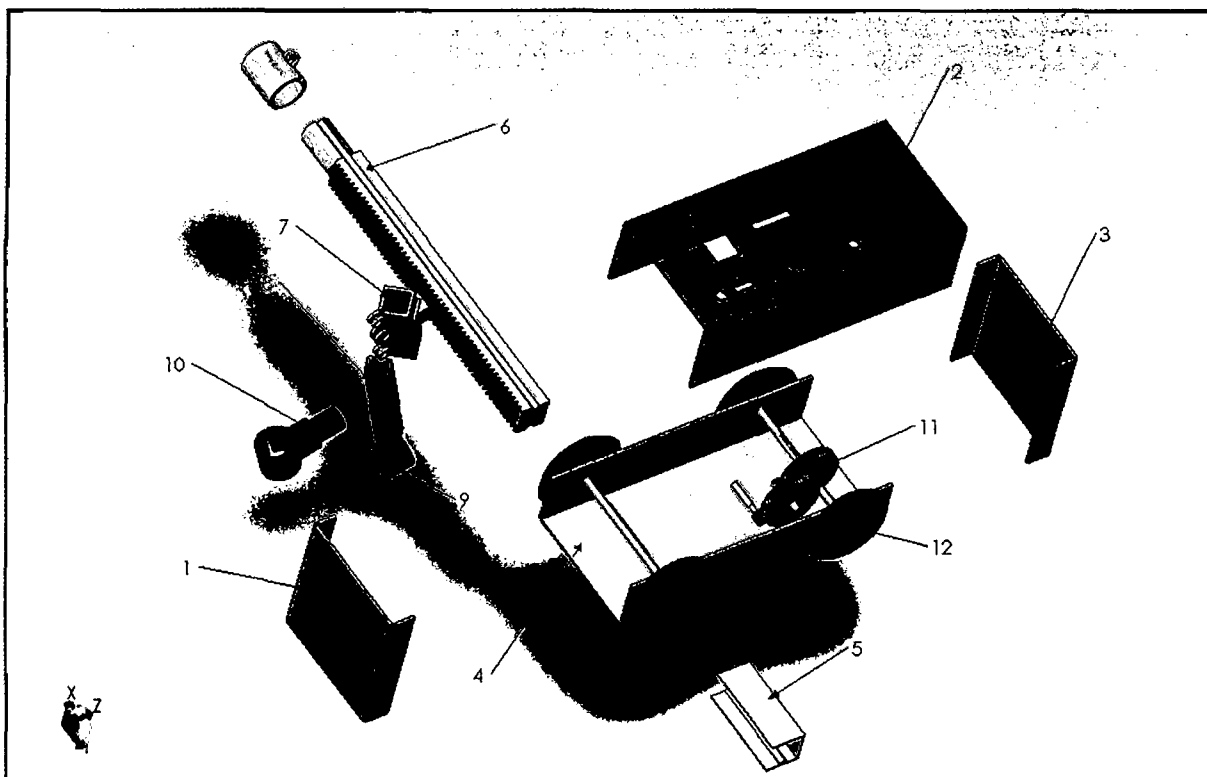


B.2: Vista explosionada 1



B.3: Vista explosionada 2



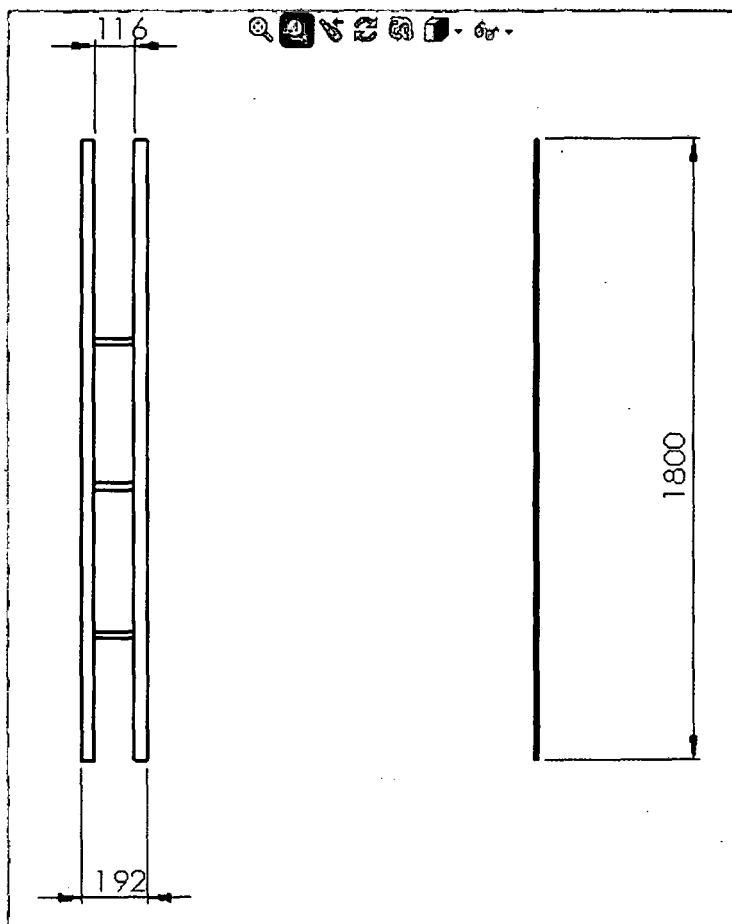


B.4: Vista explosionada 3

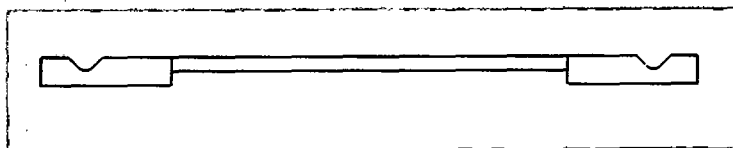
## B.5: Lista de componentes de la máquina.

| N° | Componentes   |
|----|---|
| 1  | Tapa frontal  |
| 2  | Tapa superior (contiene el display LCD, teclado y pulsadores) |
| 3  | Tapa posterior  |
| 4  | Tapa inferior   |
| 5  | Soporte de Cremallera mayor                                   |
| 6  | Cremallera mayor  |
| 7  | Soporte de cremallera inferior                                |
| 8  | Cremallera  |
| 9  | Guía de inclinación   |
| 10 | Soporte de antorcha   |
| 11 | Sistema de transmisión  |
| 12 | Rueda   |

A continuación se muestra las dimensiones del riel, dimensiones que son típicos para este tipo de máquinas, si la distancia cortar es más larga que el riel, entonces se usa más de un riel colocados uno a continuación de otro.

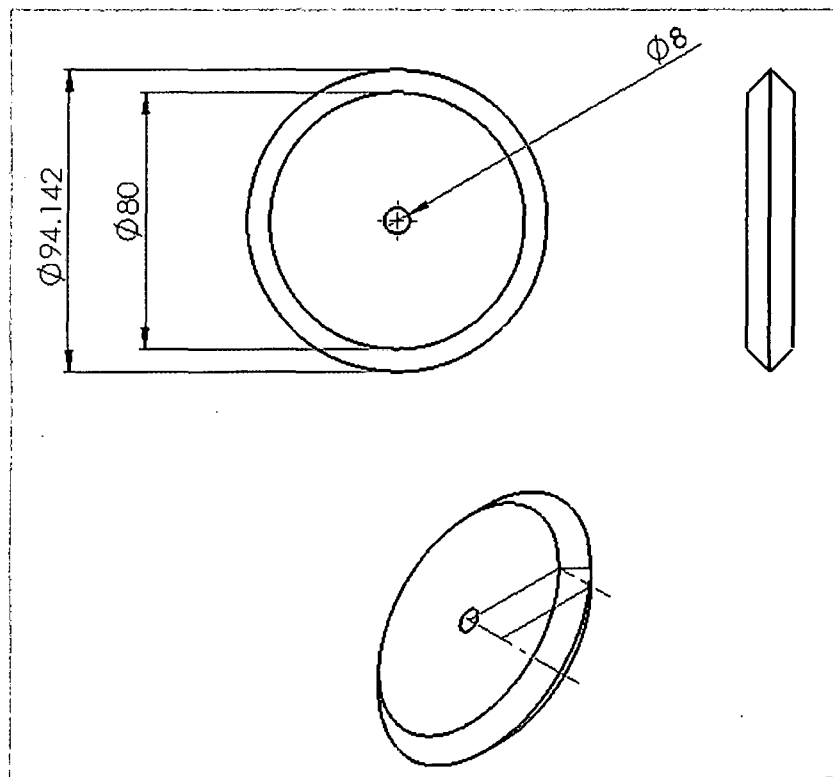


B.6: Dimensiones del riel en mm.



B.7: Detalle del riel.

En esta parte se muestra la rueda que tiene una forma de V para encajar en el riel, de tal manera que el corte sea lo más recto posible. De las cuatro ruedas de la máquina las posteriores son las que transmiten la tracción y normalmente son de metal.



B.8: Detalle de la rueda (dimensiones en mm).