UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DE UN MECANISMO DE CUATRO BARRAS PARA SU USO EN LA ENSEÑANZA

T E S I S

QUE PARA OBTENER EL GRADO DE:
INGENIERO MECATRÓNICO

PRESENTA

TORRES REYES VICTOR MANUEL

DIRECTOR DE TESIS: M.I. BENJAMÍN VALERA OROZCO

CIUDAD UNIVERSITARIA, MÉXICO, D.F. NOVIEMBRE 2009

Agradecimientos.

Esta tesis de licenciatura pudo ser realizada gracias al apoyo del CCADET, que me abrigo en sus instalaciones durante la duración del proyecto.

Así mismo agradezco al proyecto **CONACYT-103931 "UNIDAD DIDÁCTICA PARA EL ANÁLISIS DE MECANISMOS"** que me otorgó una beca y financiamiento para realizar el proyecto.

Mi más amplio agradecimiento para el Dr. Gabriel Ascanio Gasca, Secretario académico del CCADET, cuyo invaluable y generoso apoyo e interés hicieron posibles la realización de esta Tesis, así como por haber creído en mí y por haberme hecho sentir en todo momento como en casa.

Finalmente, expreso mi agradecimiento a quienes estuvieron vinculados de alguna manera a este proyecto tanto personal administrativo, mecánicos del taller, académicos del CCADET y en particular a Juan Salvador Pérez Lomeli y Alejandro Gómez Ramírez quienes se encargaron de construir el prototipo mecánico.

A todos mi mayor agradecimiento y gratitud.

Dedicatorias.

A Dios, por permitirme terminar este camino, por darme valor, perseverancia y fuerza para afrontarlo en los momentos difíciles, y capacidad para disfrutarlo en los momentos felices.

A mis padres, por su apoyo porque sin ellos no hubiera sido realidad concluir uno de los objetivos de mi vida. Por darme la mejor herencia, que se le puede dar a un hijo: una educación.

A mi director y tutor de Tesis, M.I. Benjamín Valera, su esfuerzo y dedicación. Sus conocimientos, sus orientaciones, su manera de trabajar, su paciencia y su motivación han sido fundamentales para mi formación. Él ha inculcado en mí un sentido de seriedad, responsabilidad y rigor académico sin los cuales no podría tener una formación completa como ingeniero. A su manera, ha sido capaz de ganarse mi lealtad y admiración, así como sentirme en deuda con él por todo lo recibido durante el periodo de tiempo que ha durado esta Tesis.

A mis hermanas, Bibiana, Valeria, Jessica, Sandra, quienes he compartido mi vida y tantas experiencias, y por que representan lo más importante de mi ser. Soy un hombre afortunado.

A la UNAM, por darme la oportunidad de formarme académicamente y como persona en sus aulas.

Nuestra actitud ante la vida no debe depender necesariamente del exterior, sino sobre todo de nuestro interior. Los seres humanos encerramos en nuestro interior toda posibilidad de respuesta y la capacidad de obrar, transformar y crear. De nosotros depende permitir que los acontecimientos, o el estado de ánimo influyan en nuestra conducta. Las situaciones no son, por sí mismas, ni positivas ni negativas. Somos nosotros quienes decidimos.

Índice temático

Introducción

Objetivo

Desarrollar el sistema mecánico y la interfaz gráfica para la implementación de un equipo didáctico en la enseñanza del mecanismo de 4 barras sobre la base de un sistema electrónico con interface a una computadora personal mediante el puerto USB.

Definición del problema

En la enseñanza de la ingeniería mecánica los mecanismos articulados desempeñan un papel importante y son incluidos como parte de los temarios de diversas materias. En particular, el mecanismo de cuatro barras es útil en la generación de trayectorias con aplicaciones domésticas e industriales. La figura 1 muestra esquemáticamente el mecanismo de cuatro barras.

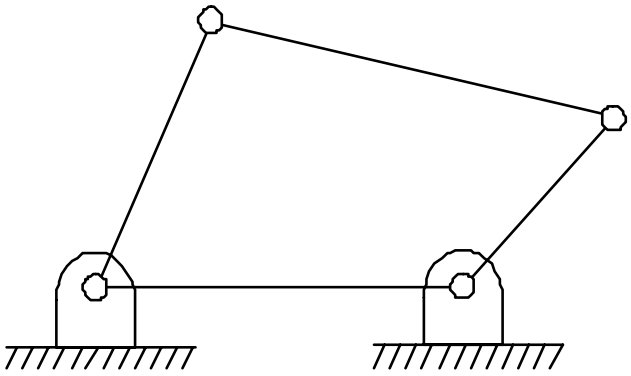


Figura 1. *Mecanismo de cuatro barras*.

Antecedentes

Para el análisis de posición de los mecanismos de barras se han desarrollado diversos simuladores tanto en *software* [1] como en *hardware* [2]. Estos han comprobado su utilidad como parte del material didáctico en diversos cursos de ingeniería mecánica.

Descripción del problema a resolver

La experiencia de los profesores que han utilizado el mecanismo de cuatro barras como parte de sus cursos escolares ha conducido a proponer un prototipo de mecanismo con características particulares que agilicen los métodos de enseñanza. En el presente trabajo de tesis se pretende desarrollar un prototipo electromecánico que cumpla con las siguientes características:

- Operación en tiempo real.
- Combinación del diseño mecánico, electrónico y la interfaz de operación
- Interface electrónica USB a computadoras personales.

Relevancia y justificación

La relevancia del presente proyecto de tesis radica en el apoyo a la creación de infraestructura de desarrollo propio que puede ser utilizada en la prestación de servicios o que puede ser ofrecida como una transferencia tecnológica. En el proyecto participan alumnos y académicos realizando labores de desarrollo tecnológico.

Alcance y limitaciones

El alcance del presente proyecto es la construcción de un prototipo funcional cuyos resultados puedan ser comparados contra el modelo teórico.

Las limitaciones del prototipo están relacionadas con la longitud de las barras ya que solo serán utilizadas las longitudes más representativas para un mecanismo de cuatro barras *manivela-balancín*. La medición angular de la barras *manivela*, *biela* y *balancín* se hace mediante el uso de codificadores ópticos incrementales con características que serán mencionadas más adelante. Sin embargo existe otra variedad de dispositivos para obtener la medición angular.

El análisis cinemático se abordo analíticamente por el método vectorial utilizando números complejos. En la mayoría de las ocasiones el diseño del mecanismo no se completa hasta que se deciden la forma de los eslabones que se requiere para que estos soporten los esfuerzos mecánicos a los que están sujetos debido al movimiento que realizan. Resulta natural entonces, aplicar principios de dinámica para determinar las fuerzas y torques del sistema. Para que nuestro análisis sea más sencillo se desprecia la forma y la masa de las barras.

Relación con otras áreas

Existe una gran relación con las áreas de electrónica digital, programación de computadoras, sistemas electrónicos digitales, mecatrónica, cinemática y medición e instrumentación.

Método

El método a emplear en el presente proyecto de tesis consiste en integrar un sistema electromecánico empleando técnicas de diseño de sistemas digitales sobre la base de una PC y microcontrolador PIC18F4550 con interface USB 2.0. La figura 2 muestra el esquema propuesto.

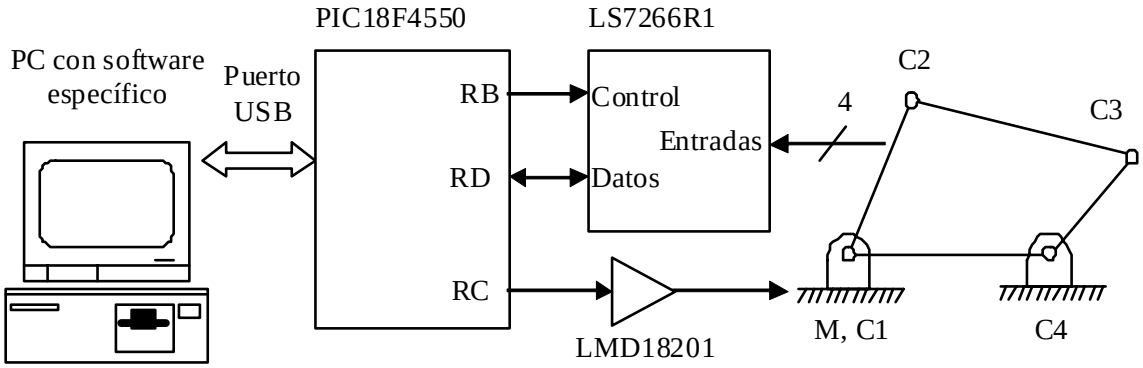


Figura 2. *Esquema del prototipo electromecánico con interface USB*.

En la figura 2 el microcontrolador PIC18F4550 controla el circuito LS7266R1 para la interface entre los transductores ópticos de posición, C1 a C4, y la PC. También, el microcontrolador genera la señal de modulación por ancho de pulso que es adecuada por el amplificador LMD18201 para impulsar el motor M que mueve las barras.

La posición medida por los transductores C1 a C4 es recibida por la interface USB en la computadora y procesada para su despliegue en un programa desarrollado específicamente para el presente proyecto. En el programa para PC también se deberá especificar la velocidad rotacional de cada eslabón del mecanismo.

Resultados

El prototipo debe mostrar su buen desempeño al validar resultados obtenidos experimentalmente contra el modelo teórico.

La comprensión del sentido físico de las cosas, depende de la capacidad de abstracción de un modelo teórico, pero la verificación experimental es importante, es decir se hacen fundamentales para la investigación y la docencia.

Resumen de la tesis

En el primer capítulo se describe el método vectorial de análisis cinemático es un método exacto para el cálculo de las variables de posición, velocidad y aceleración angulares asociadas a las barras de un mecanismo, que conforman los lazos a los

cuales se les plantea la ecuación de cierre vectorial. También se habla de los dispositivos electrónicos utilizados para la adquisición de datos.

En el segundo capítulo se describe la implementación de los sistemas que se componen de *hardware* (mecánica y electrónica) y software para que cumplan sus respectivas funciones. Como por ejemplo la transmisión de datos de los instrumentos electrónicos en tiempo real por medio de un microcontrolador para comunicarse con un *software* para mostrar los datos a través de una interfaz gráfica.

Finalmente se muestra el mecanismo de cuatro barras con los componentes mecánicos; también se muestra el sistema de adquisición de datos tanto del circuito electrónico como de la interfaz de operación. Comparan los datos de la medición cinemática experimental con el modelo teórico correspondiente y las variables adquiridas experimentalmente.

Capítulo 1
Antecedentes

En el presente capitulo se muestran las bases teóricas utilizadas en el desarrollo de este trabajo de tesis. Abarca conocimientos de áreas de mecánica, especialmente la teoría de los mecanismos, además se describen los dispositivos electrónicos utilizados en la construcción del prototipo de mecanismo de cuatro barras con operación en tiempo real.

1.1 Mecanismo de cuatro barras

Uno de los mecanismos más comunes es el mecanismo de cuatro barras. La figura 1.1 muestra una forma de mecanismo de cuatro barras donde el eslabón 1 es el eslabón motriz y el eslabón 2 es el de referencia. Este mecanismo puede tomar otras formas como se verá mas adelante.

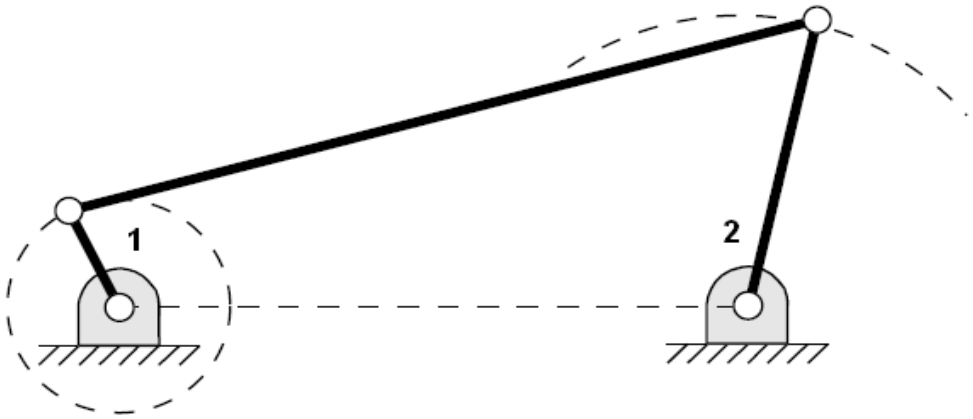


Figura 1.1. *Mecanismo típico de cuatro barras.*

Los elementos de un mecanismo se clasifican según diversos criterios. Atendiendo al comportamiento del material, pueden ser rígidos, elásticos o fluidos. Si se presta atención a sus características inerciales, pueden ser de inercia despreciable o no.

Otra clasificación de los elementos se puede realizar según el número de pares a los cuales se encuentran ligados. Así se dice que un miembro es binario, terciario, etc., cuando está ligado con dos pares, tres pares, etc.

Los miembros también se pueden clasificar según el tipo de movimiento. Así, un miembro con un punto articulado fijo se denomina *manivela* si puede dar vueltas enteras y *balancín* si solamente puede oscilar. Si el miembro no tiene ningún punto articulado fijo, recibe el nombre de *biela o acoplador.* Como se muestra en la figura 1.2.

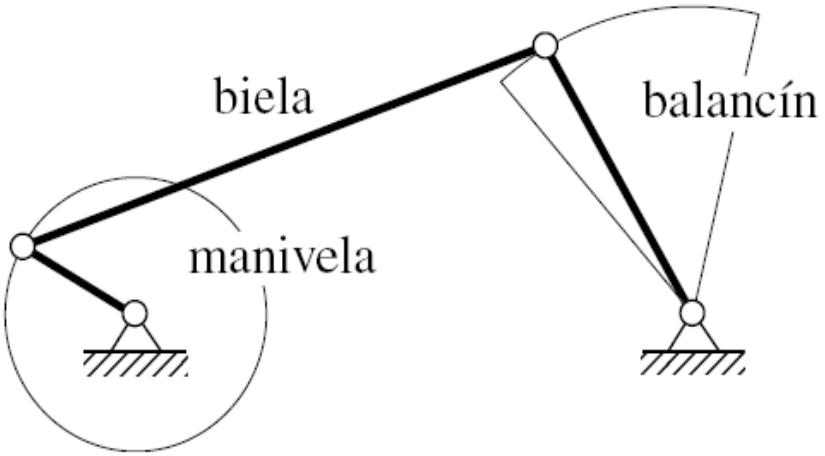


Figura 1.2. *Mecanismo articulado con nomenclatura de sus miembros*

1.1.1 Ley de Grashof

La consideración mas importante cuando se diseña un mecanismo que será impulsado por un motor es asegurarse que la manivela de entrada pueda girar una revolución completa. Para los mecanismos de cuatro barras, la *ley de Grashof* permite averiguar de manera sencilla si se cumple esta condición.

La *ley de Grashof* afirma que la barra mas corta de un mecanismo de cuatro barras da vueltas enteras respecto a todas las otras si se cumple que la suma de la longitud de la barra mas larga l y la de la mas corta s es mas pequeña o igual que la suma de las longitudes de las otras dos p y q.

La *ley de Grashof* especifica que uno de los eslabones, en particular la barra mas corta, girará continuamente solo cuando

$$s+l \leq p+q \qquad (1.1)$$

Donde:

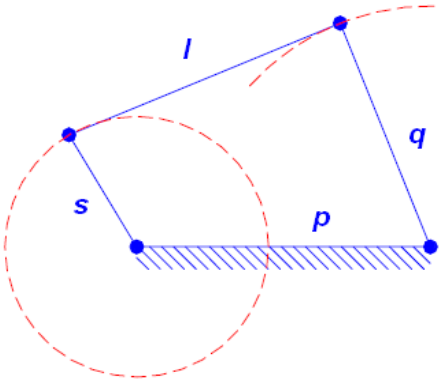
- l es la longitud del eslabón más largo

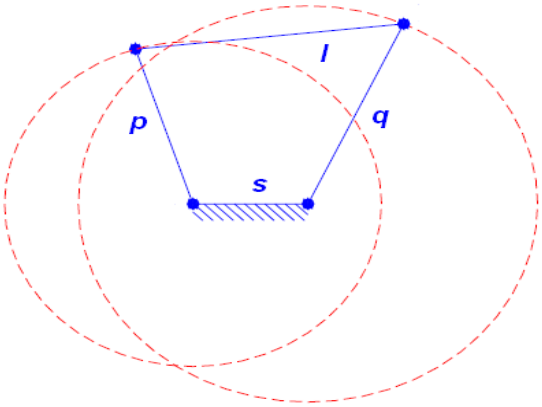
- *s* es la longitud del eslabón más corto
- *p* y *q* son las longitudes de los eslabones restantes

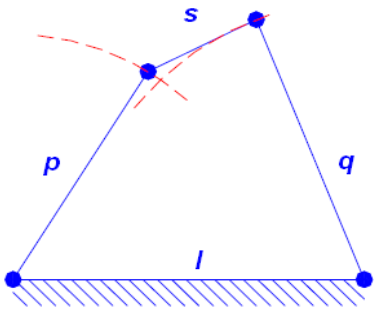
En el enunciado de la ley no interviene el orden en que se conectan las barras, ni cuál es la barra fija. De esta forma, existen varios mecanismos que se pueden formar dependiendo de la forma en que los eslabones se configuran:

1. Si el soporte del mecanismo es una de las barras adyacentes a la menor, la barra menor actúa de manivela y su opuesta de balancín (**mecanismos de manivela-balancín**).
2. Si el soporte del mecanismo es la barra menor, las dos barras adyacentes a él actúan de manivelas (**mecanismos de doble-manivela**).
3. Cuando un mecanismo no cumple una de las condiciones anteriores, las dos barras que giran respecto al soporte, se comportan como balancines (**mecanismos de doble-balancín**).
4. Paralelogramo articulado: Mecanismo donde cada barra es igual a su opuesta (la barra soporte es igual a la biela y la barra conductora es igual a la barra conducida). En este tipo de mecanismos las dos barras adyacentes al soporte son manivelas (**mecanismos de doble-manivela**).

La tabla 1.1 muestra las características de estas configuraciones.

INVERSIONES	**CARACTERÍSTICAS**
MANIVELA-BALANCÍN	Manivela-biela-balancín $s+l \leq p+q$ *s* ⇒ *barra menor (manivela)* *l* ⇒ *barra mayor (biela)* *p* ⇒ *barra fija (tierra)* *q* ⇒ *barra adyacente (balancín)*
	

DOBLE-MANIVELA	
	Manivela-biela-manivela $s+l \leq p+q$ *s⇒ barra menor (manivela)* *l⇒ barra mayor (biela)* *p⇒ barra fija (tierra)* *q⇒ barra adyacente (balancín)*

DOBLE-BALANCÍN	
	Balancín-biela-balancín $s+l \leq p+q$ *s⇒ barra menor (biela)* *l⇒ barra fija (tierra)* *p⇒ barra adyacente (balancín)* *q⇒ barra adyacente (balancín)*

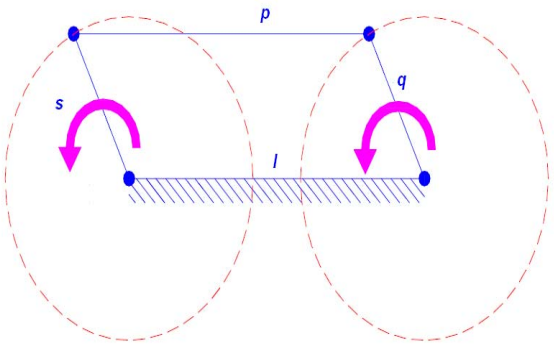
PARALELOGRAMO ARTICULADO	
	$s+l = p+q$ siendo $s=q$ y $l=q$ s y q tienen el mismo sentido de giro

Tabla 1.1. *Tipos de mecanismos.*

1.1.2 Análisis de posición

Considere el mecanismo de cuatro barras que se muestra en la figura 1.3. Si se desea hacer el análisis detallado para determinar las velocidades y aceleraciones a las que esta sujeto el mecanismo se deben conocer las relaciones entre los eslabones que forman dicho mecanismo.

Se requiere hacer el análisis de posición de este mecanismo teniendo como datos de entrada la posición inicial del mecanismo θ_2 y la longitud de las barras r_1, r_2, r_3 y r_4. Se pueden usar vectores para analizar la posición y proporcionar una descripción matemática del mecanismo.

La manera de abordar analíticamente los problemas vectoriales bidimensionales es a través del álgebra compleja. Aunque los números complejos no son vectores, se pueden usar para representar vectores en un plano $R = r^x + j\, r^y$. La utilidad real de los números complejos en el análisis en el plano se debe a la facilidad con la que se pueden pasar a forma polar. Si se usa la notación compleja rectangular para el *vector* R se puede escribir $R = r\cos\theta + j\, r\, sen\theta$.

El *vector* R también puede expresarse en forma polar compleja utilizando la ecuación de Euler:

$$e^{\pm j\theta} = \cos\theta_2 \pm j\,\mathrm{sen}\,\theta_2 \qquad (\mathbf{1.0})$$

De tal manera que el *vector R* queda expresado de la siguiente forma:

$$R = re^{j\theta} \qquad (\mathbf{1.1})$$

Para describir el movimiento se puede utilizar un sistema de referencia *xy* tal como se muestra en la figura 1.3. Una vez fijado el sistema de referencia se puede utilizar el marco de referencia el punto *O* que se encuentra fijo dentro del sistema de coordenadas.

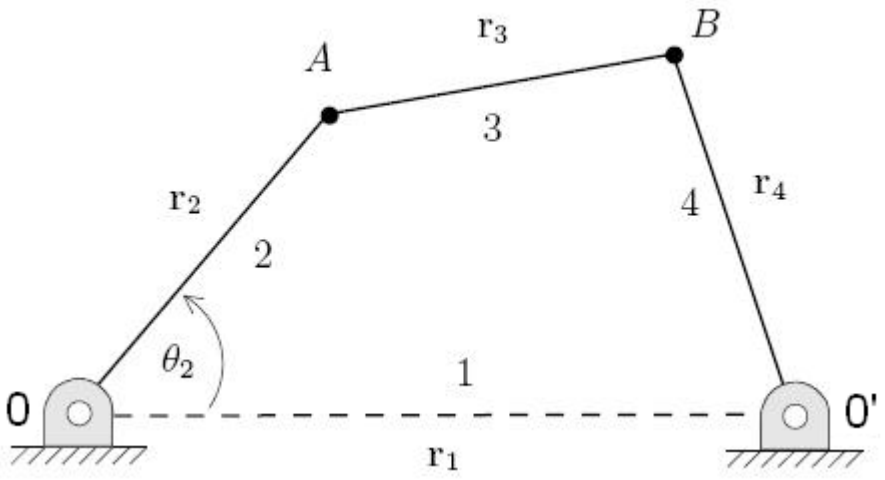


Figura 1.3. *Mecanismo de cuatro barras.*

Del mecanismo se pueden escribir las siguientes relaciones:

$$r_B = r_2 + r_3 = r_2\, e^{j\theta 2} + r_3\, e^{j\theta 3} \qquad (\mathbf{1.2})$$

$$r_B = r_1 + r_4 = r_1\, e^{j0} + r_4\, e^{j\theta 4} \qquad (\mathbf{1.3})$$

$$r_2\, e^{j\theta 2} + r_3\, e^{j\theta 3} = r_1 + r_4\, e^{j\theta 4} \qquad (\mathbf{1.4})$$

La ecuación (1.4) se conoce como *ecuación de lazo cerrado* y expresa el hecho de que el lazo forma una cadena cerrada. Las longitudes constantes de los vectores aseguran que los centros de articulación permanecerán separados a distancias constantes, requisito de los eslabones rígidos.

La ecuación de lazo cerrado representa un modelo del mecanismo puesto que contiene todas sus restricciones básicas. Debe notarse que esta ecuación por si sola no puede proporcionar una descripción completa del movimiento del mecanismo. Para obtener tal descripción será necesario agregar restricciones tales como especificar que los pasadores *O* y *O'* se encuentran en posiciones fijas.

Volviendo a la ecuación (1.4) puede escribirse en componentes como:

$$r_2\,(\cos\theta_2 + j\,\mathrm{sen}\,\theta_2) + r_3\,(\cos\theta_3 + j\,\mathrm{sen}\,\theta_3) = r_1 + r_4\,(\cos\theta_4 + j\,\mathrm{sen}\,\theta_4) \quad (\mathbf{1.5})$$

Recordando que:

$$r_1\,e^{j0} = r_1\,(\cos(0) + j\,\mathrm{sen}(0)) = r1$$

La ecuación anterior puede separarse en parte real y parte imaginaria dando como resultado:

$$r_2\cos\theta_2 + r_3\cos\theta_3 = r_1 + r_4\cos\theta_4$$

$$r_2\,\mathrm{sen}\,\theta_2 + r_3\,\mathrm{sen}\,\theta_3 = r_4\,\mathrm{sen}\,\theta_4 \quad (\mathbf{1.6})$$

A partir de las ecuaciones (1.6) puede resolverse el mecanismo. Sin embargo, las relaciones trigonométricas que se derivan de las mismas son muy complejas. Para simplificar el procedimiento de solución se puede definir un vector auxiliar $r_e = r_1 - r_2$ tal y como se muestra en la figura 1.4.

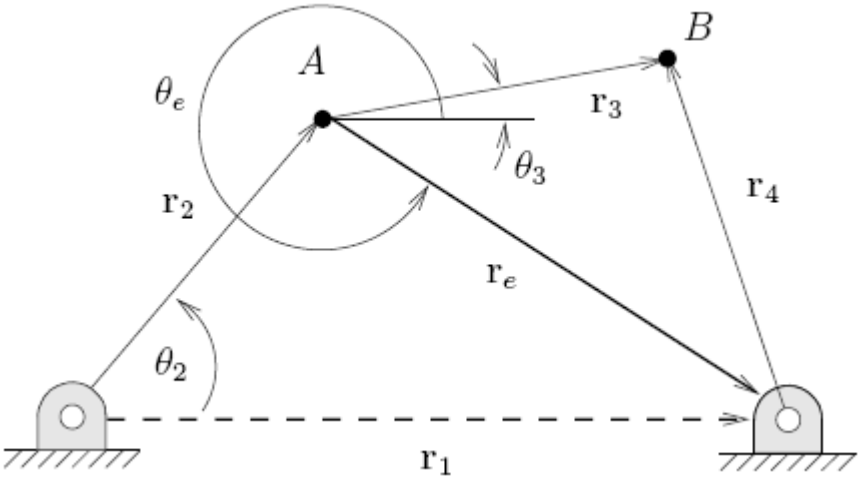


Figura 1.4. *Vector auxiliar r_e en el mecanismo de cuatro barras.*

Definiendo el *vector* r_e se pueden escribir un nuevo grupo de relaciones que se pueden utilizar para encontrar la posición del mecanismo. Por ejemplo,

$$r_e = r_1 - r_2 \quad (\mathbf{1.7})$$

$$r_e\,e^{j\theta e} = r_1 - r_2\,e^{j\theta 2} \quad (\mathbf{1.8})$$

$$r_e\,(\cos\theta_e + j\,\mathrm{sen}\,\theta_e) = r_1 - r_2\,(\cos\theta_2 + j\,\mathrm{sen}\,\theta_2) \quad (\mathbf{1.9})$$

Igualando parte real e imaginaria

$$r_e \cos\theta_e = r_1 - r_2 \cos\theta_2 \qquad \textbf{(1.10)}$$

$$r_e \operatorname{sen}\theta_e = -r_2 \operatorname{sen}\theta_2 \qquad \textbf{(1.11)}$$

Las ecuaciones anteriores pueden elevarse al cuadrado y simplificarse para eliminar el ángulo θ_e:

$$r_e^2 (\cos^2\theta_e + \operatorname{sen}^2\theta_e) = r_1^2 - 2 r_1 r_2 \cos\theta_2 + r_2^2 (\cos^2\theta_2 + \operatorname{sen}^2\theta_2) \qquad \textbf{(1.12)}$$

$$r_e^2 = r_1^2 + r_2^2 - 2 r_1 r_2 \cos\theta_2 \qquad \textbf{(1.13)}$$

Recordando que $\cos^2\theta + \operatorname{sen}^2\theta = 1$.

De la ecuación anterior puede encontrarse la longitud de r_e para cualquier ángulo θ_2:

$$r_e = \sqrt{r_1^2 + r_2^2 - 2 r_1 r_2 \cos\theta_2} \qquad \textbf{(1.14)}$$

Una vez que r_e es conocido, el ángulo θ_e puede encontrarse de la ecuación (1.11)

$$\theta_e = \operatorname{arc\,sen}\left[\frac{-r_2 \operatorname{sen}\theta_2}{r_e} \right] \qquad \textbf{(1.15)}$$

Una vez que r_e puede utilizarse para encontrar una nueva relación geométrica en el mecanismo:

$$r_3 = r_e + r_4 \qquad \textbf{(1.16)}$$

Usando la formula de Euler en la ecuación anterior y separando en partes real e imaginaria se tiene,

$$r_3 \cos\theta_3 = r_e \cos\theta_e + r_4 \cos\theta_4 \qquad \textbf{(1.17)}$$

$$r_3 \operatorname{sen}\theta_3 = r_e \operatorname{sen}\theta_e + r_4 \operatorname{sen}\theta_4 \qquad \textbf{(1.18)}$$

Elevando al cuadrado las dos ecuaciones anteriores y sumándolas se tiene

$$r_3^2 = r_e^2 + r_4^2 + 2 r_e r_4 \cos\theta_e \cos\theta_4 + 2 r_e r_4 \operatorname{sen}\theta_e \operatorname{sen}\theta_4 \qquad \textbf{(1.19)}$$

Utilizando la relación $\cos(\alpha - \beta) = \cos\alpha \cos\beta + \operatorname{sen}\alpha \operatorname{sen}\beta$ puede escribir

$$r_3^2 = r_e^2 + r_4^2 + 2 r_e r_4 \cos(\theta_e - \theta_4) \qquad \textbf{(1.20)}$$

De la ecuación (1.20) puede encontrarse el valor θ_4

$$\Theta_4 = -\arccos\left[\frac{r_3^2 - r_e^2 - r_4^2}{2\, r_e\, r_4}\right] + \Theta_e \qquad (\mathbf{1.21})$$

Finalmente, una vez que θ_4 es conocido, θ_3 puede encontrarse mediante la segunda de las ecuaciones (1.6) con lo que el análisis de posición queda concluido:

$$\Theta_3 = \arcsen\left[\frac{r_4\ \text{sen}\,\Theta_4 - r_2\ \text{sen}\,\Theta_2}{r_3}\right] \qquad (\mathbf{1.22})$$

Ángulo de transmisión

El ángulo de transmisión es una prueba útil para medir la calidad del diseño de un mecanismo. El ángulo de transmisión se define como el *ángulo entre el acoplador y el eslabón de salida.*

El ángulo se toma como el valor absoluto del ángulo agudo formado en la intersección de los dos eslabones (ver figura 1.6) y puede ser fácilmente calculado como la diferencia entre los ángulos θ_4 y θ_3 de tal forma que $\mu=\theta_4 - \theta_3$. Este ángulo es una medida de la calidad de la transmisión de la fuerza y de velocidad en la junta.

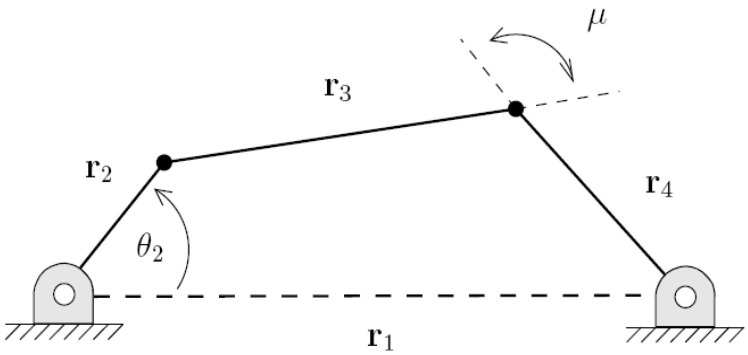


Figura 1.5. *Definición del ángulo de transmisión.*

Como se muestra en la figura 1.7, cuando se aplica un par $\mathbf{T}_2$, se genera una fuerza $\mathbf{F}_{34}$ que es la fuerza que ejerce el eslabón 3 sobre el eslabón 4 en la unión entre ambos eslabones.

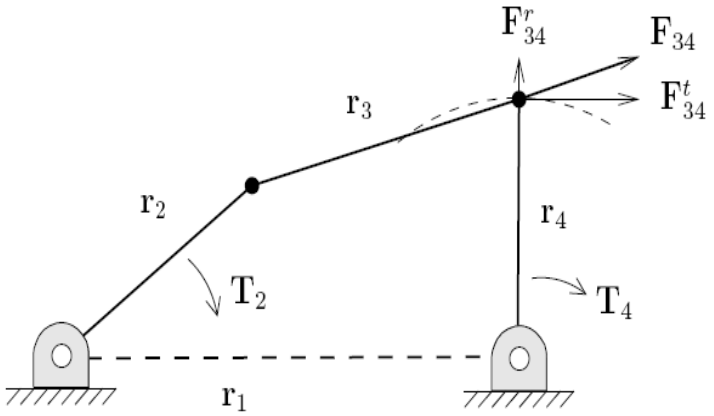


Figura 1.7. *Definición del ángulo de transmisión.*

La fuerza $\mathbf{F}_{34}$ puede descomponerse en dos componentes $\mathbf{F}^r_{34}$ la componente radial, $\mathbf{F}^t_{34}$ la componente tangencial.

Idealmente, toda la fuerza $\mathbf{F}_{34}$ debería convertirse en fuerza tangencial que es la fuerza que genera $\mathbf{T}_4$, lo cual se logra cuando $\mu = \pi/2$. Desafortunadamente esto no es posible debido a que el mecanismo se encuentra en movimiento. Como resultado, se genera una fuerza radial $\mathbf{F}^r_{34}$ que no contribuye al momento de salida pero que aumenta la fricción en el pivote.

Para poder verificar la calidad del diseño, se vuelve necesario conocer los valores máximos y mínimos de μ .Estos valores extremos pueden encontrarse como:

$$\mu_1 = \arccos\left[\frac{r_3^2 + r_4^2 - (r_1 + r_2)^2}{2\, r_3\, r_4}\right] \qquad (\mathbf{1.23})$$

$$\mu_2 = \arccos\left[\frac{r_3^2 + r_4^2 - (r_1 - r_2)^2}{2\, r_3\, r_4}\right] \qquad (\mathbf{1.24})$$

Aunque el ángulo de transmisión ideal es $\mu = \pi/2$, $\mu < \pi/4$ representan ángulos para los cuales la fuerza radial será mayor que la fuerza tangencial. Sustituyendo los valores de los eslabones en la ecuación 1.24 se obtiene el siguiente valor mínimo:

$$\mu_2 = \arccos\left[\frac{(40^2+30^2-(40-15)^2}{2(40)(30)}\right] = 38.625°$$

En general, se considera buena práctica mantener el ángulo de transmisión mínimo por arriba de los 35°- 40°.

1.1.3 Análisis de velocidad

Hasta ahora se ha visto que el análisis de posición de un mecanismo puede llevarse a cabo de una manera más o menos directa utilizando algebra compleja. Este análisis se basa en expresar los vectores que representan a los eslabones en forma compleja como:

$$R = re^{j\theta} \qquad (\mathbf{1.25})$$

Esta forma resulta muy apta para realizar un análisis de velocidad puesto que la velocidad de un punto puede encontrarse directamente como la derivada con respecto al tiempo del vector de posición R:

$$\dot{R} = \frac{dR}{dt} = \frac{d}{dt} Re^{j\theta} \qquad (\mathbf{1.26})$$

La ecuación anterior puede desarrollarse utilizando la fórmula para la derivada de un producto y la regla de la cadena:

$$\frac{dR}{dt} = \dot{R}e^{j\theta} + j\dot{\theta}Re^{j\theta} \qquad (\mathbf{1.27})$$

Donde el punto denota derivada con respecto al tiempo de tal forma que

$$\dot{R} = \frac{dR}{dt} \quad y \quad \dot{\theta} = \frac{d\theta}{dt}$$

Considere una vez más un mecanismo de cuatro barras como el mostrado en la figura 1.8. Una vez que el análisis de posición de este mecanismo se ha realizado, los valores θ_3 y θ_4 son conocidos.

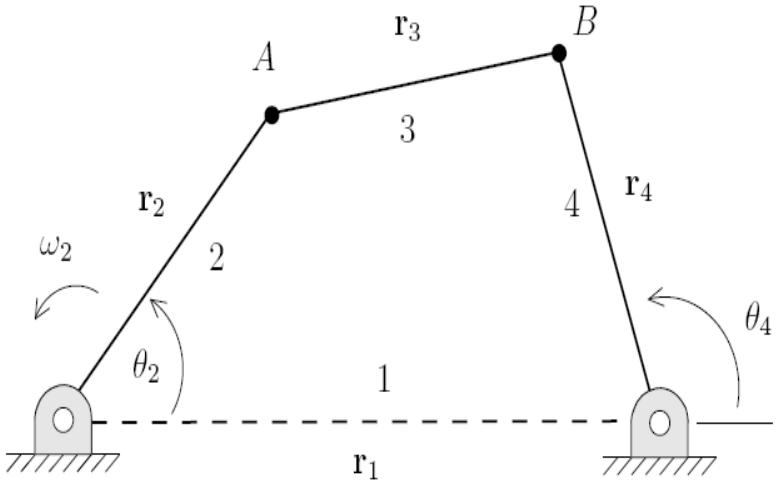


Figura 1.8. *Mecanismo de cuatro barras.*

Si se desea conocer. Por ejemplo, el valor de v_B es necesario realizar ahora el análisis de velocidad para este mecanismo. Si se conecta el eslabón 2 de este mecanismo a un motor, entonces se conocerá no solo la posición del eslabón sino también su velocidad angular.

Para comenzar el análisis de velocidad, se pueden considerar las dos ecuaciones que describen el movimiento del punto B del mecanismo:

$$r_B = r_2 + r_3$$

$$r_B = r_1 + r_4$$

Estas ecuaciones pueden igualarse para obtener

$$r_2 + r_3 = r_1 + r_4$$

Derivando la ecuación anterior con respecto al tiempo

$$\dot{r}_2 + \dot{r}_3 = \dot{r}_1 + \dot{r}_4$$

El siguiente paso es expandir la ecuación anterior usando Euler recordando que

$$\dot{R} = \dot{R}\, e^{j\theta} + j\,\dot{\theta}\, R\, e^{j\theta}$$

$$r_2 e^{j\theta_2} + j\,\dot{\theta}_2\, r_2\, e^{j\theta_2} + r_3 e^{j\theta_3} + j\,\dot{\theta}_3\, r_3\, e^{j\theta_3} = r_1 e^{j\theta_1} + j\,\dot{\theta}_1\, r_1\, e^{j\theta_1} + r_4\, e^{j\theta_4} + j\,\dot{\theta}_4\, r_4 e^{j\theta_4} \qquad (\mathbf{1.28})$$

Del mecanismo se puede apreciar que r_1, r_2, r_3, r_4 y θ_1 no cambian en el tiempo por lo que sus derivadas son iguales a cero. Simplificando la ecuación (1.28):

$$j\,\dot{\theta}_2\, r_2\, e^{j\theta_2} + j\,\dot{\theta}_3\, r_3\, e^{j\theta_3} = j\,\dot{\theta}_4\, r_4 e^{j\theta_4} \qquad (\mathbf{1.29})$$

El siguiente paso para la solución del problema es expandir la ecuación anterior tomando en cuenta que $e^{j\theta} = \cos\theta + j\,\text{sen}\,\theta$:

$$j\,\dot{\theta}_2\, r_2\,(\cos\theta_2 + j\,\text{sen}\,\theta_2) + j\,\dot{\theta}_3\, r_3\,(\cos\theta_3 + j\,\text{sen}\,\theta_3) = j\,\dot{\theta}_4\, r_4\,(\cos\theta_4 + j\,\text{sen}\,\theta_4) \qquad (\mathbf{1.30})$$

La ecuación anterior puede separarse en parte real y parte imaginaria recordando que $\dot{\theta} = \omega$

$$-\omega_2\, r_2\, \text{sen}\,\theta_2 - \omega_3\, r_3\, \text{sen}\,\theta_3 = -\omega_4\, r_4\, \text{sen}\,\theta_4 \qquad (\mathbf{1.31})$$

$$\omega_2\, r_2 \cos\theta_2 + \omega_3\, r_3 \cos\theta_3 = \omega_4\, r_4 \cos\theta_4 \qquad (\mathbf{1.32})$$

Donde la ecuación (1.31) representa la parte real y la ecuación (1.32) representa la parte imaginaria. Pasando los términos conocidos del lado derecho y lo desconocido del izquierdo en ambas ecuaciones.

$$-\omega_3 r_3 \operatorname{sen}\theta_3 - \omega_4 r_4 \operatorname{sen}\theta_4 = \omega_2 r_2 \operatorname{sen}\theta_2 \qquad (\mathbf{1.33})$$

$$\omega_3 r_3 \cos\theta_3 - \omega_3 r_3 \cos\theta_3 = -\omega_2 r_2 \cos\theta_2 \qquad (\mathbf{1.34})$$

Las ecuaciones (1.33) y (1.34) representan dos ecuaciones simultáneas con dos incógnitas, ω_3 y ω_4. Este sistema puede resolver fácilmente utilizando el método de Cramer [3].

$$\omega_3 = \frac{\begin{vmatrix} \omega_2 r_2 \operatorname{sen}\theta_2 & r_4 \operatorname{sen}\theta_4 \\ -\omega_2 r_2 \cos\theta_2 & -r_4 \cos\theta_4 \end{vmatrix}}{\begin{vmatrix} -r_3 \operatorname{sen}\theta_3 & r_4 \operatorname{sen}\theta_4 \\ r_3 \cos\theta_3 & -r_4 \cos\theta_4 \end{vmatrix}} \qquad (\mathbf{1.35})$$

De donde se obtiene que

$$\omega_3 = \frac{-\omega_2 r_2 r_4 \operatorname{sen}\theta_2 \cos\theta_4 + \omega_2 r_2 r_4 \cos\theta_2 \operatorname{sen}\theta_4}{r_3 r_4 \operatorname{sen}\theta_3 \cos\theta_4 - r_3 r_4 \operatorname{sen}\theta_4 \cos\theta_3} \qquad (\mathbf{1.36})$$

En la última ecuación, si se conoce previamente la posición del mecanismo y adicionalmente se conoce la velocidad angular de la manivela, quedan como incógnitas las velocidades angulares de la biela y el balancín; estos valores pueden hallarse de la siguiente forma:

$$\omega_3 = -\frac{r_2}{r_3}\,\omega_2\,\frac{\operatorname{sen}(\theta_4 - \theta_2)}{\operatorname{sen}(\theta_4 - \theta_3)} \qquad (\mathbf{1.37})$$

De forma análoga, se pueden encontrar que el valor de ω_4 está dado por:

$$\omega_4 = -\frac{r_2}{r_4}\,\omega_2\,\frac{\operatorname{sen}(\theta_3 - \theta_2)}{\operatorname{sen}(\theta_4 - \theta_3)} \qquad (\mathbf{1.38})$$

Estas expresiones reflejan claramente que la relación de velocidades entre cada barra de salida y la entrada, depende exclusivamente de las posiciones de los eslabones que se pueden obtener de los codificadores ópticos. Por lo tanto las ecuaciones (1.37) y (1.38) se usan para obtener la velocidad angular de la *biela* y el *balancín* respectivamente.

El comportamiento de la velocidad del mecanismo queda completamente descrito una vez encontrados los valores de ω_3 y ω_4. Si se desea encontrar la velocidad v_B, esta puede encontrarse directamente puesto que del mecanismo la velocidad v_B es igual a la velocidad v_4. Así, $v_B = v_4 = r_4 \times \omega_4$, donde la dirección de v_B es $\theta_4 + \pi/2$.

1.1.4 Análisis de aceleración

Una vez se han realizado los análisis de posición y velocidad en un mecanismo, es conveniente realizar el análisis de aceleración del mismo. La importancia del análisis de aceleración reside en el hecho de que las fuerzas a las que están sujetos los mecanismos son directamente proporcionales a la aceleración que estos experimentan.

Para realizar el análisis de aceleración de un mecanismo, es necesario derivar dos veces con respecto al tiempo su ecuación de cierre, para tal efecto, la ecuación de Euler vuelve a presentarse como una herramienta conveniente.

Partiendo de la expresión de Euler para el análisis de velocidad

$$\dot{R} = \frac{dR}{dt} = \frac{d}{dt} R e^{j\theta}$$

Se puede encontrar la expresión para la aceleración derivando esta última ecuación con respecto al tiempo

$$\frac{d\dot{R}}{dt} = \ddot{R} = \frac{d}{dt}(\dot{R} e^{j\theta}) + \frac{d}{dt}(j\dot{\theta} R e^{j\theta})$$

Donde los dos términos en el lado derecho de la ecuación tienen que derivarse como productos. Para este fin, resulta conveniente separarlos como $(\dot{R})(e^{j\theta})$ y $(j\dot{\theta})(Re^{j\theta})$. De tal forma que,

$$\ddot{R} = \ddot{R} e^{j\theta} + j\dot{\theta}\dot{R} e^{j\theta} + j\ddot{\theta} R e^{j\theta} + j\dot{\theta}(\dot{R} e^{j\theta} + j\dot{\theta} R e^{j\theta})$$

$$\ddot{R} = \ddot{R} e^{j\theta} + j\dot{\theta}\dot{R} e^{j\theta} + j\ddot{\theta} R e^{j\theta} + j\dot{\theta}\dot{R} e^{j\theta} + jj\dot{\theta}^2 R e^{j\theta}$$

$$\ddot{R} = \ddot{R} e^{j\theta} + 2j\dot{\theta}\dot{R} e^{j\theta} + j\ddot{\theta} R e^{j\theta} - \dot{\theta}^2 R e^{j\theta} \qquad (\mathbf{1.39})$$

Donde $\ddot{R} = \frac{d^2R}{dt^2}$ y $\ddot{\theta} = \frac{d^2\theta}{dt^2}$

El término $2j\dot{\theta}\dot{R}e^{j\theta}$ de la ecuación (1.39) presenta una característica especial ya que toma en cuenta el acoplamiento que existe entre los cambios de longitud y ángulo del *vector R* con respecto al tiempo. Este término, conocido como la *aceleración de Coriolis*, ha aparecido de manera natural al realizar la diferenciación y estará presente

siempre que haya una velocidad de deslizamiento asociada a cualquier elemento que también tenga una velocidad angular. En ausencia de uno u otro de estos factores. La componente de *Coriolis* será nula.

Considere una vez más el mecanismo de cuatro barras descrito en la figura 1.9. Ya se ha visto que, en base a la figura, se puede escribir

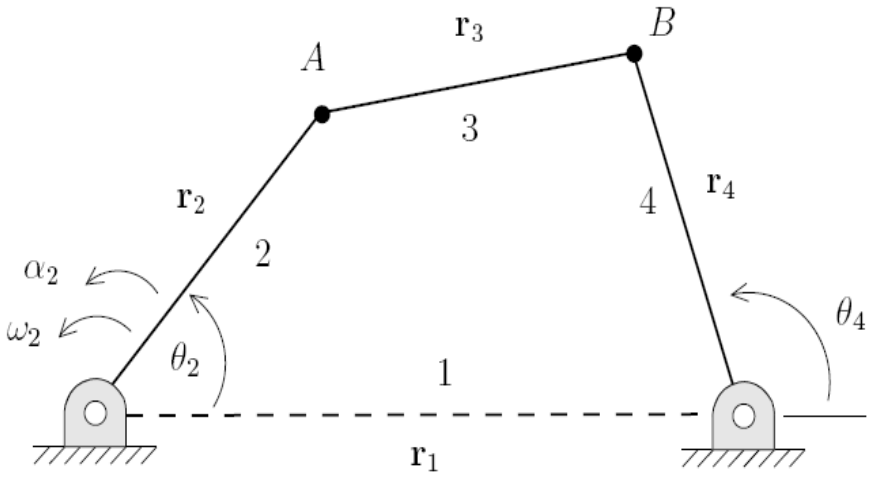


Figura 1.9. *Definición del ángulo de transmisión.*

$$\mathbf{r}_2+\mathbf{r}_3=\mathbf{r}_1+\mathbf{r}_4$$

Para realizar el análisis de aceleración es necesario derivar la expresión anterior dos veces con respecto al tiempo

$$\ddot{\mathbf{r}}_2+\ddot{\mathbf{r}}_3=\ddot{\mathbf{r}}_1+\ddot{\mathbf{r}}_4$$

La ecuación anterior se puede expandir usando la ecuación (1.39)

$$\begin{aligned}&\ddot{r}_2 e^{j\theta_2}+j\,2\,\dot{\theta}_2\,r_2\,e^{j\theta_2}-\dot{\theta}_2^2\,r_2\,e^{j\theta_2}+j\,\ddot{\theta}_2\,r_2\,e^{j\theta_2}+\\&\ddot{r}_3 e^{j\theta_3}+j\,2\,\dot{\theta}_3\,r_3\,e^{j\theta_3}-\dot{\theta}_3^2\,r_3\,e^{j\theta_3}+j\,\ddot{\theta}_3\,r_3\,e^{j\theta_3}=\\&\ddot{r}_1 e^{j\theta_1}+j\,2\,\dot{\theta}_1\,r_1\,e^{j\theta_1}-\dot{\theta}_1^2\,r_1\,e^{j\theta_1}+j\,\ddot{\theta}_1\,r_1\,e^{j\theta_1}+\\&\ddot{r}_4 e^{j\theta_4}+j\,2\,\dot{\theta}_4\,r_4\,e^{j\theta_4}-\dot{\theta}_4^2\,r_4\,e^{j\theta_4}+j\,\ddot{\theta}_4\,r_4\,e^{j\theta_4}\end{aligned}\qquad(\mathbf{1.40})$$

Una vez más, del mecanismo se puede observar que los eslabones r_1, r_2, r_3 y r_4 no cambian su longitud con respecto al tiempo por lo que sus derivadas son iguales a cero. También el ángulo θ_1 permanece constante, ayudando a simplificar aun más la ecuación.

Reescribiendo (1.40) usando $\dot{\theta} = \omega$ y $\ddot{\theta} = \alpha$

$$-\omega_2^2 r_2 e^{j\theta_2} + j\alpha_2 r_2 e^{j\theta_2} - \omega_3^2 r_3 e^{j\theta_3} + j\alpha_3 r_3 e^{j\theta_3} = -\omega_4^2 r_4 e^{j\theta_4} + j\alpha_4 r_4 e^{j\theta_4} \quad (\mathbf{1.41})$$

Expandiendo la ecuación anterior en términos de senos y cosenos

$$\begin{aligned} &-\omega_2^2 r_2 (\cos\theta_2 + j\,\mathrm{sen}\,\theta_2) + j\alpha_2 r_2 (\cos\theta_2 + j\,\mathrm{sen}\,\theta_2) \\ &-\omega_3^2 r_3 (\cos\theta_3 + j\,\mathrm{sen}\,\theta_3) + j\alpha_3 r_3 (\cos\theta_3 + j\,\mathrm{sen}\,\theta_3) \\ &= -\omega_4^2 r_4 (\cos\theta_4 + j\,\mathrm{sen}\,\theta_4) + j\alpha_4 r_4 (\cos\theta_4 + j\,\mathrm{sen}\,\theta_4) \end{aligned} \quad (\mathbf{1.42})$$

Separando la ecuación anterior en la parte real y parte imaginaria se obtienen las siguientes expresiones

$$\begin{aligned} &-\omega_2^2 r_2 \cos\theta_2 + \alpha_2 r_2 \,\mathrm{sen}\,\theta_2 - \omega_3^2 r_3 \cos\theta_3 + \alpha_3 r_3 \,\mathrm{sen}\,\theta_3 \\ &= -\omega_4^2 r_4 \cos\theta_4 + \alpha_4 r_4 \,\mathrm{sen}\,\theta_4 \end{aligned} \quad (\mathbf{1.43})$$

$$\begin{aligned} &-\omega_2^2 r_2 \,\mathrm{sen}\,\theta_2 + \alpha_2 r_2 \cos\theta_2 - \omega_3^2 r_3 \,\mathrm{sen}\,\theta_3 + \alpha_3 r_3 \cos\theta_3 \\ &= -\omega_4^2 r_4 \,\mathrm{sen}\,\theta_4 + \alpha_4 r_4 \cos\theta_4 \end{aligned} \quad (\mathbf{1.44})$$

Dejando los términos conocidos del lado derecho en ambas ecuaciones

$$\begin{aligned} &-\alpha_3 r_3 \,\mathrm{sen}\,\theta_3 + \alpha_4 r_4 \,\mathrm{sen}\,\theta_4 \\ &= -\omega_2^2 r_2 \cos\theta_2 + \alpha_2 r_2 \,\mathrm{sen}\,\theta_2 - \omega_3^2 r_3 \cos\theta_3 + \omega_4^2 r_4 \cos\theta_4 \end{aligned} \quad (\mathbf{1.45})$$

$$\begin{aligned} &-\alpha_3 r_3 \cos\theta_3 + \alpha_4 r_4 \cos\theta_4 \\ &= -\omega_2^2 r_2 \,\mathrm{sen}\,\theta_2 + \alpha_2 r_2 \cos\theta_2 - \omega_3^2 r_3 \,\mathrm{sen}\,\theta_3 + \omega_4^2 r_4 \,\mathrm{sen}\,\theta_4 \end{aligned} \quad (\mathbf{1.46})$$

Escribiendo las ecuaciones anteriores en forma compacta

$$-\alpha_3 r_3 \,\mathrm{sen}\,\theta_3 + \alpha_4 r_4 \,\mathrm{sen}\,\theta_4 = A$$

$$-\alpha_3 r_3 \cos\theta_3 + \alpha_4 r_4 \cos\theta_4 = B \quad (\mathbf{1.47})$$

Usando la fórmula de Cramer para resolver para α_3 y α_4

$$\alpha_3 = \frac{\begin{vmatrix} A & r_4 \,\mathrm{sen}\,\theta_4 \\ B & r_4 \cos\theta_4 \end{vmatrix}}{\begin{vmatrix} -r_3 \,\mathrm{sen}\,\theta_3 & r_4 \,\mathrm{sen}\,\theta_4 \\ -r_3 \cos\theta_3 & r_4 \cos\theta_4 \end{vmatrix}} = -\frac{1}{r_3} \frac{A\cos\theta_4 - B\,\mathrm{sen}\,\theta_4}{\mathrm{sen}\,(\theta_3 - \theta_4)} \quad (\mathbf{1.48})$$

$$\alpha_4 = \frac{\begin{vmatrix} -r_3 \operatorname{sen}\theta_3 & A \\ -r_3 \cos\theta_3 & B \end{vmatrix}}{\begin{vmatrix} -r_3 \operatorname{sen}\theta_3 & r_4 \operatorname{sen}\theta_4 \\ -r_3 \cos\theta_3 & r_4 \cos\theta_4 \end{vmatrix}} = -\frac{1}{r_4} \frac{A\cos\theta_3 - B\operatorname{sen}\theta_3}{\operatorname{sen}(\theta_3 - \theta_4)} \qquad (\mathbf{1.49})$$

Estas expresiones reflejan que dependen exclusivamente de las posiciones de los eslabones que se pueden obtener de los codificadores ópticos. Por lo tanto se usan para obtener la aceleración angular de la biela y el balancín.

1.1.5 Solución del mecanismo por computadora

Como se puede apreciar de las secciones anteriores, el proceso para describir el comportamiento de un mecanismo conlleva si no una gran dificultad, si un gran número de operaciones puesto que las ecuaciones vistas dan información para una sola posición del mecanismo.

Afortunadamente, el uso álgebra compleja para analizar el mecanismo facilita el análisis mediante una computadora. Para tal fin se utilizan una gran variedad de herramientas, desde lenguajes computacionales como: hojas de cálculo hasta paquetes de matemática y *software* especializado, con las cuales es posible mostrar gráficamente el movimiento de un cuerpo rígido o un mecanismo mediante simulaciones.

En el anexo A.1 se muestra el código fuente desarrollado utilizando el paquete "***Mathematica"*** y que se utilizo para comparar el desempeño del prototipo con la teoría anteriormente expuesta.

1.2 Medición de posición

Para la medición de posición se utilizan diversos transductores electrónicos, dependiendo del tipo de aplicación. Para el presente proyecto, se utilizaron codificadores ópticos con el objetivo de determinar las posiciones angulares necesarias para construir el prototipo de mecanismo de cuatro barras.

1.2.1 Codificadores ópticos

Los codificadores ópticos se utilizan en muchos aparatos eléctricos, que requieren gran precisión como ratones de ordenador, robots móviles, maquinaria e impresoras. La pieza clave es un disco con una serie de orificios, conectado a una parte móvil que lo pone en movimiento, en tanto que una fuente luminosa (normalmente un LED) se coloca de manera que filtre la luz a través de las hendiduras. En la cara opuesta del disco se sitúa un fotosensor que produce una señal eléctrica cuando se le somete a un estimulo luminoso. Desde el momento en el que el disco se hace rotar, el sensor genera una señal cada vez que la luz del LED atraviesa un orificio. La electrónica del codificador cuenta, pues, el número de señales luminosas recibidas y, de esta manera, calcula cual es la entidad de la rotación del disco.

Definición

Un codificador óptico es un sensor que permite detectar el movimiento de rotación de un eje. Es en definitiva un transductor que transforma una posición angular a una señal digital (a través de un potencial).

El codificador estará operando en relación al eje del elemento cuya posición se desea determinar y su fundamento viene dado por la obtención de la medida en base a la luz que traspasa una serie de discos superpuestos que codificarán la salida digital.

Principio de operación

La figura 1.10 representa un codificador óptico. Este consiste en dos diodos fotoemisores y dos transistores que realizan la tarea de fotoreceptores. Estos elementos se encargan de detectar la presencia / ausencia de luz para convertir el movimiento en un tren de pulsos eléctricos en función de una codificación en cuadratura para obtener la medida final a través de un disco solidario al eje, con ranuras radiales.

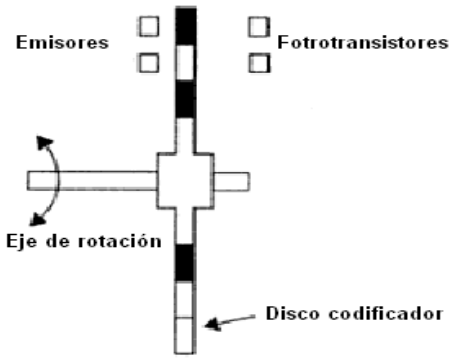


Figura 1.10. *Codificador en cuadratura.*

Clasificación de codificador óptico atendiendo su salida

- **Unidireccionales:** dan una salida y no se puede determinar el sentido de giro. Sólo nos servirá para obtener valores absolutos. Por ejemplo, para obtener velocidades absolutas sin importar el sentido de giro.

- **Bidireccionales:** nos ofrece dos salidas A y B. El sentido se va a distinguir por la diferencia de fase citada anteriormente. Será útil cuando necesitemos saber coordenadas exactas tanto positivas como negativas.

Decodificación

La posición de cada fototransistor es un cuarto de la distancia existente entre dos franjas. Esta disposición provocará dos señales; *Va* y *Vb*, entre ambas señales se provocara un desfasamiento de 90º o de un cuarto de ciclo. La resolución será de 360º/N, donde N es el número de franjas presente en la superficie del cilindro.

Las señales en cuadratura *Va* y *Vb* girando en sentido horario y antihorario se muestran en la figura 1.11 y 1.12, respectivamente.

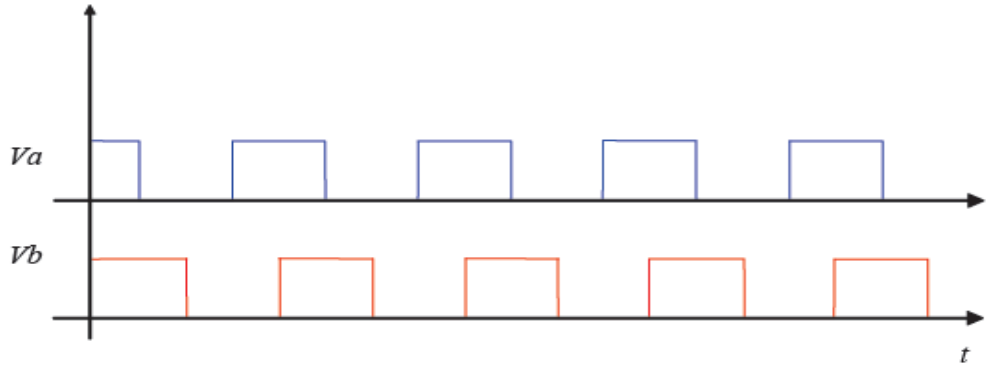


Figura 1.11. *Va y Vb rotando en sentido horario.*

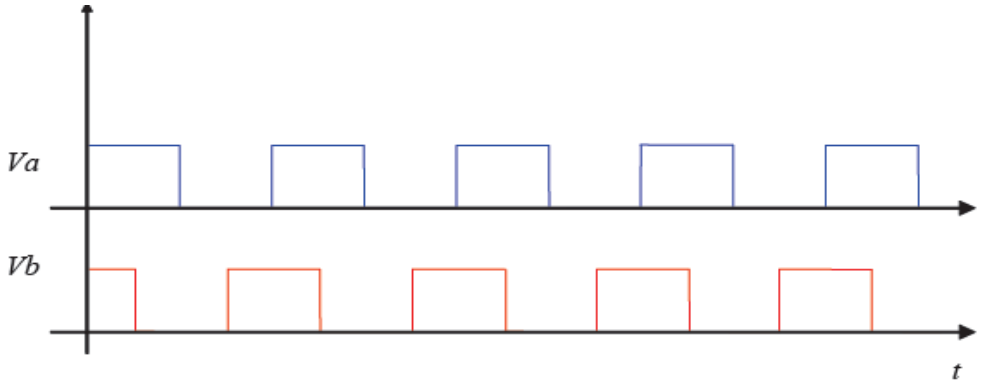


Figura 1.12. *Va y Vb rotando en sentido antihorario.*

Las señales de salida de los codificadores en cuadratura contienen la información y el sentido de giro realizado.

Codificador óptico rotatorio de diámetro hueco (HB5M Hollow Bore Optical Encoder)

El *HB5M* es un codificador óptico rotatorio de diámetro hueco con protección de aluminio anodizado [4]. El codificador incremental óptico HB5M está diseñado para un fácil montaje al eje rotación para proporcionar información digital para cualquier aplicación de control de movimiento.

El codificador *HB5M* tiene un orificio que acepta ejes de 5 mm a 8 mm de diámetro, respectivamente. El codificador se desliza sobre el eje y se puede sujetar con dos tornillos de ajuste 4-48. Hay un aro (1.812”) que esta sujeto por dos tornillos de ajuste 4-40, además contiene dos orificios que proporciona la facilidad de montar el codificador en cualquier superficie y la carcasa viene con una cubierta cerrada para limitar la penetración de partículas. Como se muestra en la figura 1.13.

Figura 1.13. *Vista frontal del HB5M.*

Las características generales de este dispositivo en particular son las siguientes:

- *Resolución*: 1024 ciclos por revolución.
- *Diámetro del agujero*: 5 mm
- *Indexado*: tercer canal (I)
- *Salida*: una sola terminación (con 5 pines) con 2 canales de cuadratura.
- *Carcasa*: sin agujero externo.
- Relación de fase: el disco A se dirige al disco B con sentido a las manecillas del reloj, mientras que B va en sentido contrario al movimiento de A, esto visto desde la parte trasera del codificador.

Las dimensiones del HB5M se muestran en la figura 1.14.

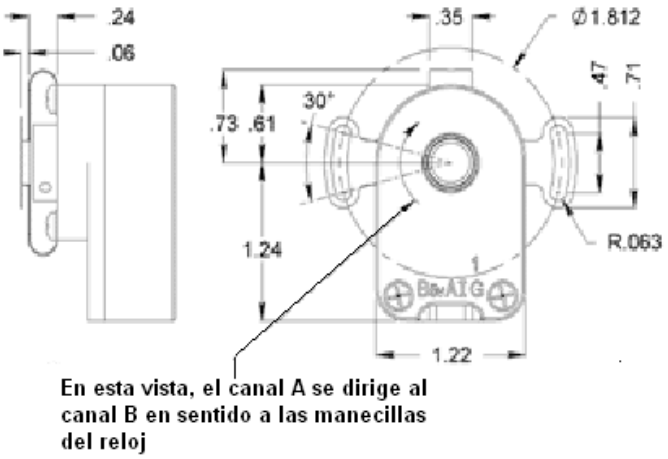


Figura 1.14. *Diseño a detalle del codificador óptico.*

En el desarrollo del prototipo de cuatro barras se utilizaron 4 codificadores HB5M para cada par cinemático con el objetivo de proporcionar la medición de desplazamiento y posteriormente poder implementar los análisis de posición, velocidad y aceleración en tiempo real.

1.2.2 Interface electrónica para codificadores ópticos

Para resolver el problema de medición de posición a partir de las señales en cuadratura que generan los codificadores ópticos, es necesario desarrollar una tarjeta electrónica que transfiera la información de los sensores en cuadratura del mecanismo de cuatro barras a la computadora por puerto USB. Para esto se incorpora un contador en cuadratura y un microcontrolador que cuente con puerto USB de comunicación.

El problema consiste en decodificar la señal de cuadratura proveniente de los codificadores ópticos de los eslabones del mecanismo de cuatro barras. Se emplea el circuito electrónico contador en cuadratura LS7266R1.

El LS7266R1 está construido con tecnología CMOS, cuenta con dos contadores independientes de 24 bits **X** y **Y**, ambos programables por medio de registros internos de 8 bits, así como un control de habilitación o deshabilitación. Cuenta con un bus de comunicación de 8 bits que le permite programar el conteo en tiempo real de señales en cuadratura, el cual funciona como canal de datos y canal de control mediante el control de habilitación del circuito.

Para escribir al canal de datos se cuenta con registros *preset latch* y para la salida están habilitados los registros *output latch*, ambos escriben o leen información de 3 bytes en un ciclo de reloj. Un *byte pointer* se encarga de incrementar automáticamente el direccionamiento del siguiente byte para no crear conflictos de comunicación.

Por medio del canal de control se escriben los comandos internos para controlar la transferencia de datos del *preset latch* hacia el contador y la comunicación de salida del contador hacia los *output latch*.

El canal de control cuando lee regresa 6 bits de estado de los registros de bandera. Cuando se escribe los datos son enviados a uno de los cuatro diferentes registros de control.

- RLD. Registro de decodificación para *preset* y carga.
- CMR Registro del modo de conteo.
- IOR. Registro de control para entrada y salida.
- IDR. Registro de control de índice.

La figura 1.15 muestra un diagrama a bloques de la estructura interna del contador en cuadratura y el acceso a sus registros de control por el bus de datos.

De los ocho bits que se envía al canal de control sólo son configurables D0-D4, D5 Y D6 que determinan el tipo de registro de control y D7 identifica cual de los dos contadores de 24 bits esta habilitado.

Es necesario programar un microcontrolador que administre los registros de control del contador en cuadratura y adquiera los datos del contador de 24 bits, así como la habilitación de cada eje X o Y [5].

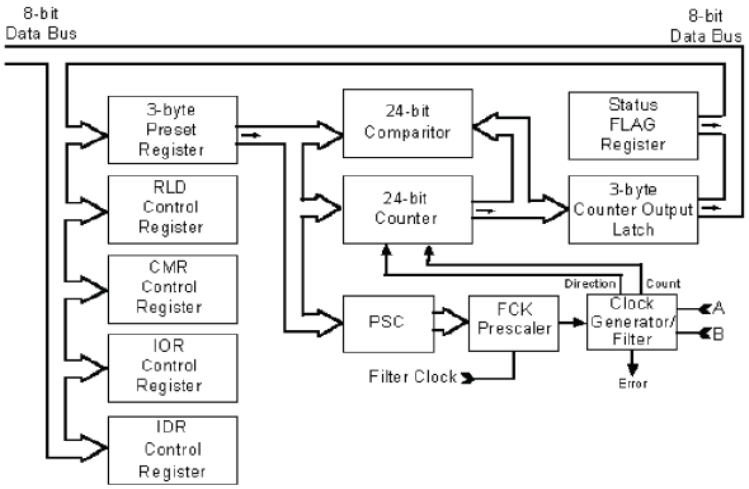


Figura 1.15. *Diagrama a bloques del contador de cuadratura.*

La figura 1.16 muestra un circuito de ejemplo para conectar el circuito LS7266R1 a un microprocesador.

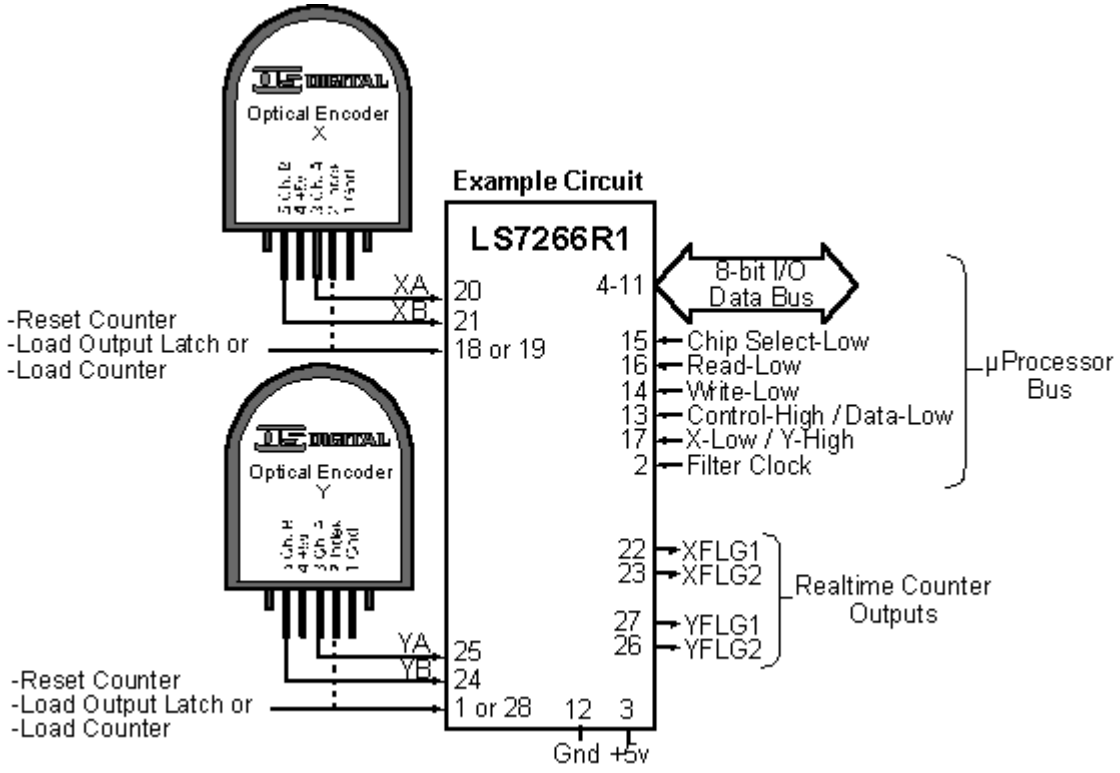


Figura 1.16. *Circuito típico de la interfaz para el LS7266R1.*

1.3 Microcontrolador PIC18F4550

En los últimos años, Microchip ha lanzado varias gamas de PIC con elevadas prestaciones, los PIC18, los PIC24 y los dsPIC. Con la gama alta (PIC18), Microchip mantiene la arquitectura básica que tan buenos resultados ha obtenido con la gama baja y media y, además, reduce las limitaciones de estas dos ultimas. Los PIC18 tienen una arquitectura RISC avanzada Harvard con 16 bits de bus de programa y 8 bits de bus de datos. La figura 1.17 muestra el encapsulado del PIC18F4550.

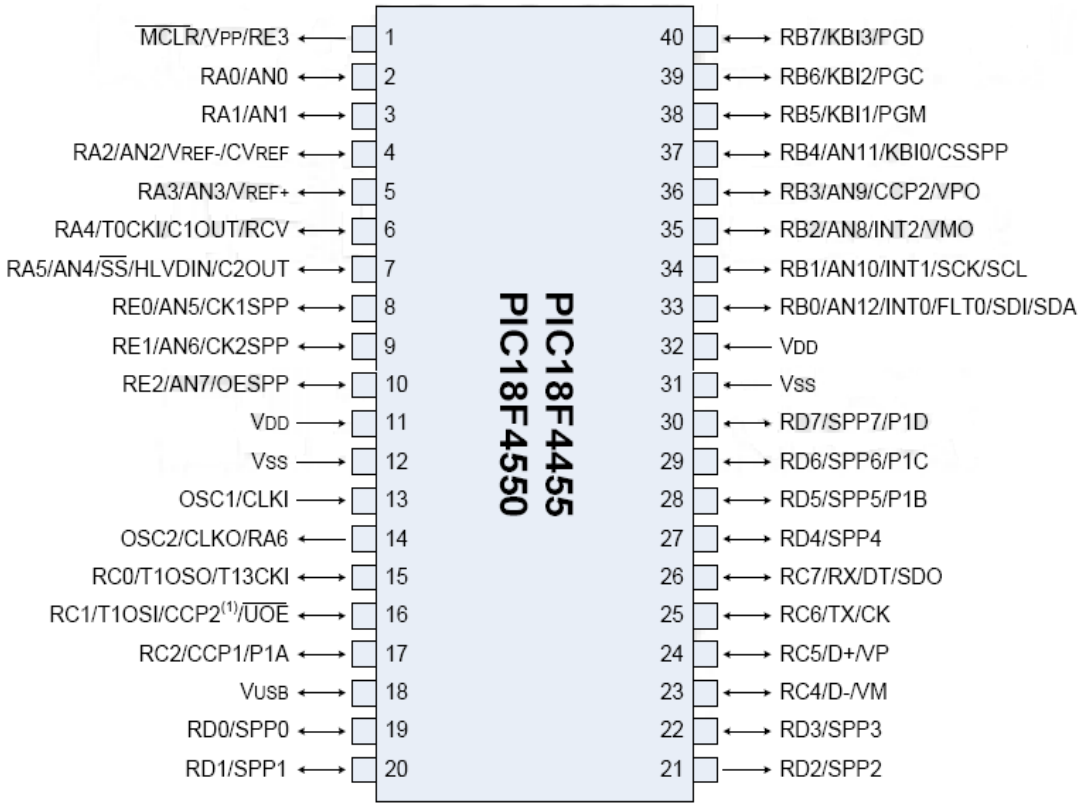


Figura 1.17. *Patigrama del microcontrolador.*

La memoria de programa aumenta hasta 1 MWord (en realidad se manejan hasta 64 Kbytes pero llegan hasta los 2 Mbytes con memoria externa) y desaparece la paginación. La memoria de datos RAM puede llegar hasta 16x256 (4 KBytes) y hasta los 1 Kbytes de EEPROM.

La pila aumenta hasta 31 niveles. Incluyen tres punteros FSR que permiten direccionar la memoria de datos de forma indirecta y sin bancos. El juego de instrucciones aumenta hasta las 75 instrucciones. Introduce un multiplicador hardware 8x8. La frecuencia máxima de reloj es de 48 MHz y la velocidad de procesador llega a los 10 MIPS con oscilador de 10 MHz. Incluye periféricos de comunicación avanzados (CAN y USB). Con la filosofía tradicional de Microchip, los PIC18 son compatibles con los PIC16CXX y PIC17CXX. Además ha desarrollado un compilador C especifico para esta gama alta, el C18 [6].

1.3.1 Especificaciones generales

Las especificaciones generales del microcontrolador se muestran en la tabla 1.2.

CARACTERISTICAS	PIC18F2455	PIC18F2450	PIC18F4455	PIC18F4450
Frecuencia de Operación	Hasta 48MHz	Hasta 48MHz	Hasta 48MHz	Hasta 48MHz
Memoria de Programa (bytes)	24.576	32.768	24.576	32.768
Memoria RAM de Datos (bytes)	2.048	2.048	2.048	2.048
Memoria EEPROM Datos (bytes)	256	256	256	256
Interrupciones	19	19	20	20
Líneas de E/S	24	24	35	35
Temporizadores	4	4	4	4
Módulos de Comparación/Captura/PWM (CCP)	2	2	1	1
Módulos de Comparación/Captura/PWM mejorado (ECCP)	0	0	1	1
Canales de Comunicación Serie	MSSP,EUSART	MSSP,EUSART	MSSP,EUSART	MSSP,EUSART
Canal USB	1	1	1	1
Puerto Paralelo de Transmisión de Datos (SPP)	0	0	1	1
Canales de Conversión A/D de 10 bits	10 Canales	10 Canales	13 Canales	13 Canales
Comparadores analógicos	2	2	2	2
Juego de instrucciones	75 (83 ext.)	75 (83 ext.)	75 (83 ext.)	75 (83 ext.)
Encapsulados	PDIP 28 pines SOIC 28 pines	PDIP 28 pines SOIC 28 pines	PDIP 40 pines QFN 40 pines TQFP 40 pines	PDIP 40 pines QFN 40 pines TQFP 40 pines

Tabla 1.2. *Características generales de los microcontroladores PIC18F2455/2550/4455/4550.*

1.3.2 Organización de la memoria

EL PIC18F4550 dispone de 5 memorias:

- **Memoria de programa**: memoria de *FLASH* interna de 32.768 bytes:
    - Almacena instrucciones y constantes/datos.
    - Puede ser escrita/leída mediante un programador externo o durante la ejecución del programa mediante unos punteros.

- **Memoria RAM de datos**: memoria SRAM interna de 2048 bytes en la que están incluidos los registros de función especial:
    - Almacena datos de forma temporal durante la ejecución del programa.
    - Puede ser escrita/leída en tiempo de ejecución mediante diversas instrucciones.
- **Memoria EEPROM de datos**: memoria no volátil de 256 bytes.
    - Almacena datos que se deben conservar aun en ausencia de tensión de alimentación.

- Puede ser escrita/leída en tiempo ejecución a través de registros.

- **Pila**: bloque de 31 palabras de 21 bits.

  - Almacena la dirección de la instrucción que debe ser ejecutada después de una interrupción o subrutina.

- **Memoria de configuración**: memoria en la que se incluyen los bits de configuración (12 bytes de memoria flash) y los registros de identificación (2 bytes de memoria de solo lectura).

El PIC18F4550 dispone de buses diferentes para el acceso a la memoria de programa y a la memoria de datos (arquitectura Harvard):

- Bus de la memoria de programa:
  - 21 líneas de dirección.
  - 16/8 líneas de datos (16 líneas para instrucciones/8 líneas para datos).

- Bus de la memoria de datos:
  - 12 líneas de dirección
  - 8 líneas de datos.

Esto permite acceder simultáneamente a la memoria de programa y a la memoria de datos. Es decir, se puede ejecutar una instrucción (por lo que generalmente requiere acceso a la memoria de datos) mientras se lee de la memoria de programa la siguiente instrucción (proceso *pipeline*).

La figura 1.18 muestra la distribución de la memoria de programa en donde es importante destacar los siguientes vectores:

- Vectorización del Reset es 0000H.
- Vectorización de las interrupciones de alta prioridad es la 0008H.
- Vectorización de las interrupciones de baja prioridad es la 0018H.

Todas la interrupciones pueden ser programadas con cualquiera de las dos prioridades, salvo la interrupción externa 0, que siempre tiene alta prioridad.

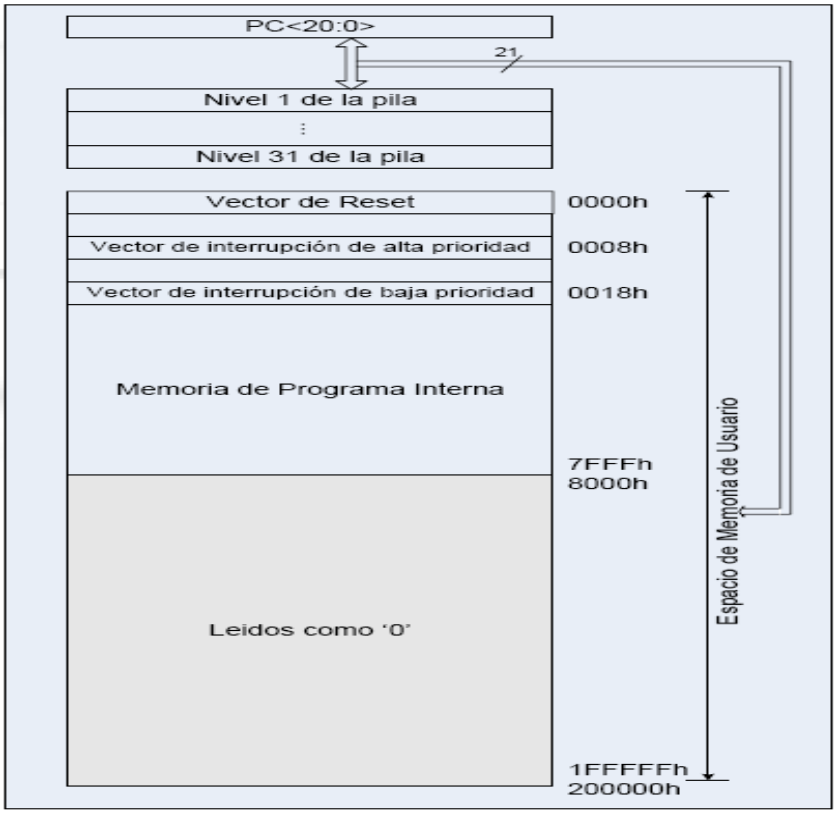


Figura 1.18. *Distribución de la memoria de programa.*

1.3.3 Diagrama a bloques

El diagrama a bloques del microcontrolador se muestra en la figura 1.19.

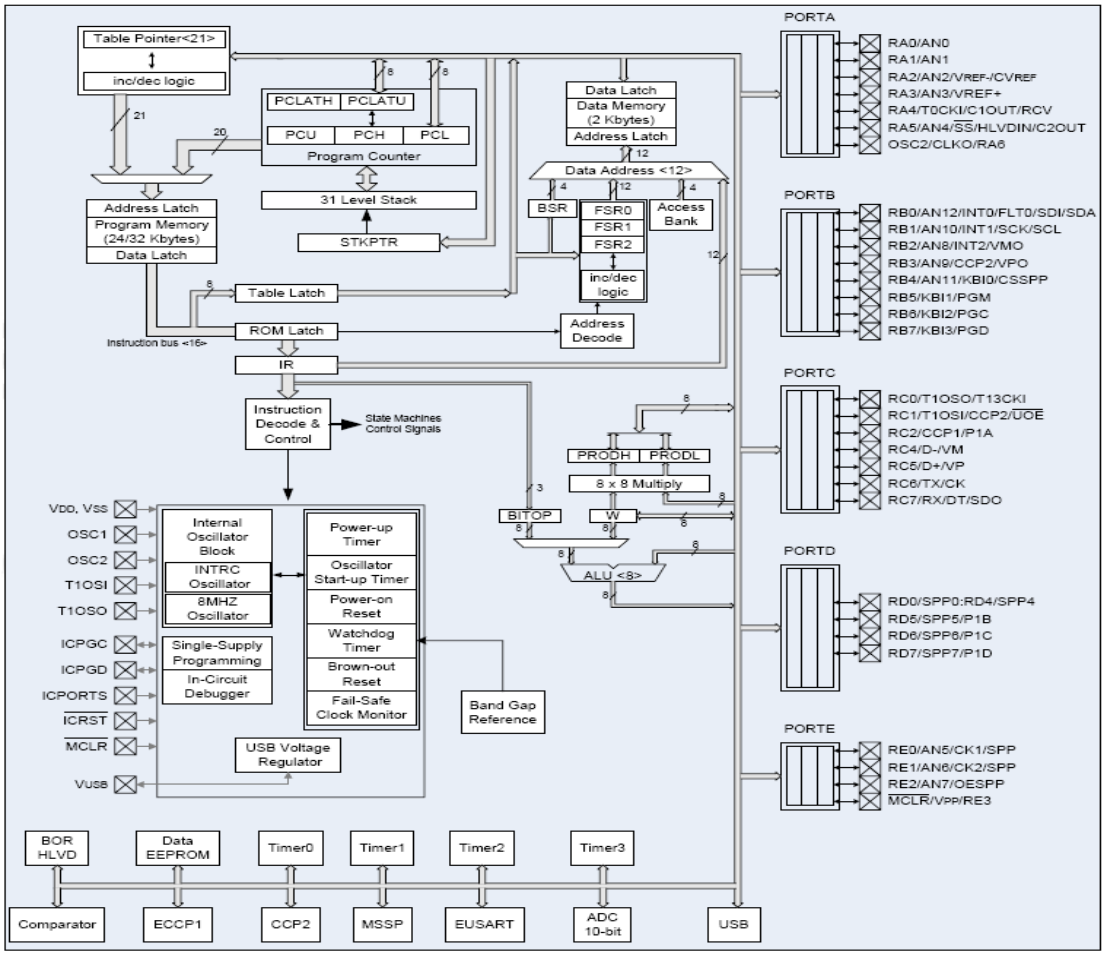


Figura 1.19. *Diagrama a bloques del PIC18F4550.*

1.3.4 Oscilador

El PIC18F4550 permite las configuraciones múltiples que se muestran en la tabla 1.3:

XT	*Cristal-Resonador (max. 4MHz)*
XTPLL	*Cristal-Resonador con habilitación de PLL*
HS	*Cristal-Resonador Alta-Velocidad*
HSPLL	*Cristal-Resonador Alta-Velocidad con habilitación de PLL*
EC	*Reloj externo con la salida FOSC/4*
ECIO	*Reloj externo con la entrada/salida en RA6*
ECPLL	*Reloj externo con habilitación de PLL y FOSC/4 con salida en RA6*
ECPIO	*Reloj externo con habilitación de PLL, entrada/salida en RA6*
INTHS	*Oscilador interno usado como fuente del reloj, HS usado como fuente del reloj.*
INTXT	*Oscilador interno usado como fuente del reloj, XT usado como fuente del reloj del USB.*

INT IO	*Oscilador interno usado como fuente del reloj, EC usado como fuente del reloj del USB, entrada-salida digital en RA6.*
INTCKO	*Oscilador interno usado como la fuente del reloj, EC usado como fuente del reloj del USB, FOSC/4 salida en RA6.*

Tabla 1.3. *Configuraciones del reloj.*

Para configurar el oscilador se requieren activar los bits de los registros necesarios. Para un cristal de 20 MHz, podemos configurar el CPU del PIC18F4550 a 48 MHz, como se muestra en la tabla 1.4.

Input Oscillator Frequency	PLL Division (PLLDIV2:PLLDIV0)	Clock Mode (FOSC3:FOSC0)	MCU Clock Division (CPUDIV1:CPUDIV0)	Microcontroller Clock Frequency
20 MHz	÷5 (100)	HS, EC, ECIO	None (00)	20 MHz
			÷2 (01)	10 MHz
			÷3 (10)	6.67 MHz
			÷4 (11)	5 MHz
		HSPLL, ECPLL, ECPIO	÷2 (00)	48 MHz
			÷3 (01)	32 MHz
			÷4 (10)	**24 MHz**
			÷6 (11)	16 MHz

Tabla 1.4. *Configuración del oscilador de entrada 20 MHz.*

Posteriormente se deben activar los fusibles necesarios para poder trabajar a la máxima velocidad de transmisión de datos, como se muestra en la figura 1.20.

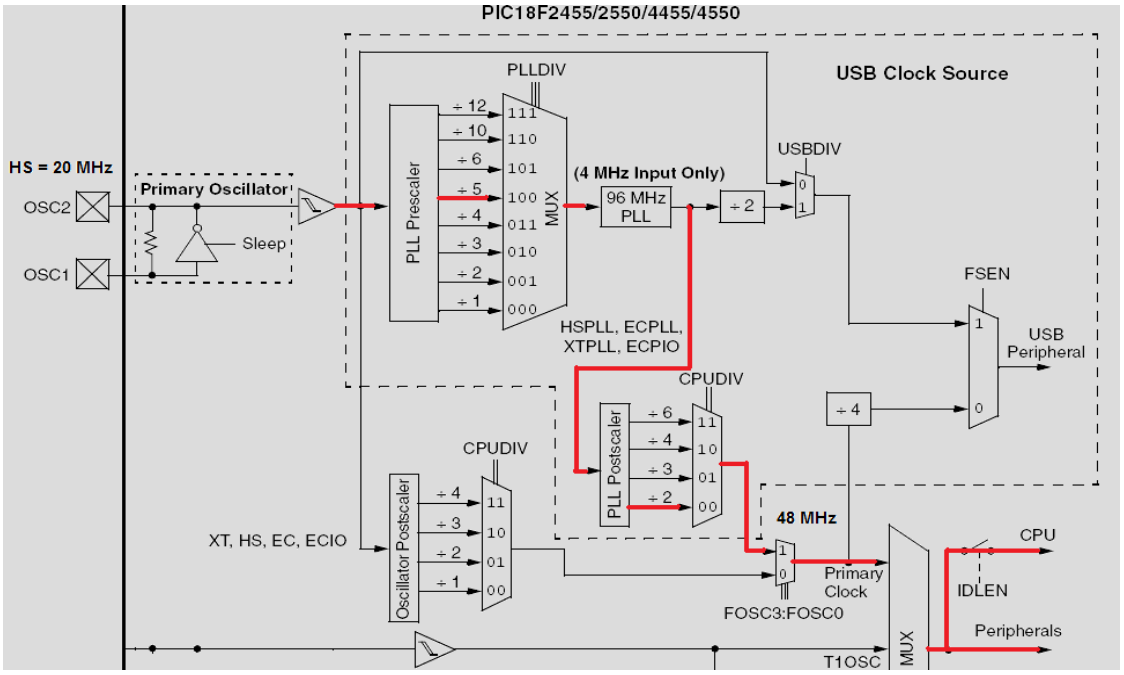


Figura 1.20. *Configuración del oscilador interno a 48 MHz.*

1.3.5 Unidades funcionales

El PIC18F4550 dispone de una serie de unidades funcionales que le permiten:

- Realizar tareas específicas especializadas (conversión A/D, transmisión/recepción de datos, generación de señales digitales con temporizaciones programables, etc.).

- Optimizar el rendimiento del PIC, ya que estas unidades trabajan en paralelo a la CPU permitiendo que esta se centre en otras tareas como el procesado de datos, cálculos, movimiento de datos, etc.

Las unidades funcionales más importantes del PIC18F4550 son las que se muestran en la tabla 1.5:

Puerto de E/S	*Unidad de comparación/Captura/PWM mejorada (ECCP)*
Temporizador 0	*Canal de comunicación serie EUSART*
Temporizador 1	*Canal de comunicación serie MSSP*
Temporizador 2	*Modulo analógico de comparación*
Temporizador 3	*Canal de transmisión de datos en paralelo (SSP)*
Convertidor A/D	*Acceso de memoria externa (EMA)*
	Unidad de Comparación/Captura/PWM (CCP)

Tabla 1.5. *Unidades funcionales.*

A continuación se describen las unidades funcionales del microcontrolador.

Puertos de entrada/salida

El PIC18F4550 dispone de 5 puertos de E/S que incluyen un total de 35 líneas digitales de E/S, como se muestra en la tabla 1.6.

PUERTO	LINEAS DE ENTRADA/SALIDA
PORTA	*7 LINEAS DE ENTRADA/SALIDA*
PORTB	*8 LINEAS DE ENTRADA/SALIDA*
PORTC	*6 LINEAS DE ENTRADA/SALIDA + 2 LINEAS DE ENTRADA*
PORTD	*8 LINEAS DE ENTRADA/SALIDA*
PORTE	*3 LINEAS DE ENTRADA/SALIDA + 1 LINEA DE ENTRADA*

Tabla 1.6. *Puertos de entrada/salida.*

Todas las líneas digitales de E/S disponen, como mínimo, de una función alternativa asociada algún circuito especifico del PIC. Cuando una línea trabaja en el modo alternativo no puede ser utilizada como línea digital de E/S estándar.

Temporizadores

Los temporizadores del PIC18F4550 se utilizan para generar eventos en tiempos específicos. Se cuenta con los siguientes cuatro temporizadores:

- Temporizador 0
    - Configurable como temporizador/contador de 8 bits/16 bits.
    - Pre-escalar de 8 bits programable.
    - Interrupción por desbordamiento.
- Temporizador 1
    - Configurable como temporizador/contador de 16 bits.
    - Dispone de un oscilador propi que puede funcionar como:
        - Señal de reloj del temporizador 1.
        - Señal de reloj del PIC en modos de bajo consumo.
    - Pre-escalar de 3 bits programable.
    - Interrupción por desbordamiento.
- Temporizador 2
    - Temporizador de 8 bits (registro TMR2).
    - Registro de periodo PR2.
    - Pre-escalar de 2 bits programable (1:1,1:4,1:16)
    - Post-escalar d 4 bits (1:1…1:16).
    - Interrupción por igualdad entre TMR2 y PR2.
    - Se puede utilizar junto con los módulos CCP y ECCP.
    - Se puede utilizar como señal de reloj del modulo MSSP en modo SPI.
- Temporizador 3
    - Configurable como temporizador/contador de 16 bits.
    - Dispone de varias opciones de señal de reloj en el modo temporizador.
        - Oscilador principal con o sin pre-escalar.
        - Oscilador del temporizador 1 con o sin pre-escalar.
    - Pre-escalar de 3 bits programable.
    - Interrupción por desbordamiento.

Convertidor Analógico-Digital

El modulo de conversión analógica a digital tiene las siguientes características:

- 10 bits de resolución.
- 13 canales multiplexados.

- Señal de reloj de conversión configurable.
- Tiempo de adquisición programable (0 a 20 TAD).
- Posibilidad de establecer el rango de tensiones de conversión mediante tensiones de referencia externas.

Canal de Comunicación Serie (EUSART)

El modulo EUSART es uno de los dos módulos de entrada/salida serie del microcontrolador. Sus características son las siguientes:

- Modos de trabajo:
  - Modo asíncrono de 8 bits
  - Modo asíncrono de 9 bits
  - Modo síncrono Maestro.
  - Modo síncrono Esclavo.
- Auto-activación por detección de datos percibido.
- Detección automática de velocidad de comunicación (baudrate).
- Transmisión y detección de carácter de BREAK (bus LIN).

Modulo Master SSP (MSSP)

El modulo MSSP es la segunda vía de comunicación serie del microcontrolador. Este puede operarse en uno de los siguientes modos:

- Serial Peripheral Interface (SPI).
- Inter-Integrated Circuit (I2C).

La interfaz I2C admite los siguientes modos:

- Master mode.
- Multi-Master mode.
- Slave mode.

Módulo de Comparación / Captura/ PWM (CCP)

Dispone de tres modos de funcionamiento:

- **Modo de captura**: se utiliza para medir eventos externos como la duración de pulsos digitales.

- **Modo de comparación**: se utiliza para generar señales digitales con temporizaciones programables. Este tipo de señales son muy útiles para el control de etapas de potencia (convertidores DC/DC, DC/AC, AC/DC).

- **Modo PWM**: se utiliza para generar señales de modulación de ancho de pulso (PWM).

Módulo de comparación/captura/PWM mejorado (ECCP)

Dispone de cuatro modos de funcionamiento:

- **Modo de captura**: se utiliza para medir eventos externos como la duración de pulsos digitales.

- **Modo de comparación**: se utiliza para generar señales digitales con temporizaciones programables. Este tipo de señales son muy útiles para el control de etapas de potencia (convertidores DC/DC, DC/AC, AC/AC).

- **Modo PWM**: se utiliza para generar señales de modulación de ancho de pulso (PWM).

- **Modo PWM mejorado:** se utiliza para generar señales PWM complementarias para el control de semipuentes de transistores.

1.4 Sistema básico de desarrollo

Para el desarrollo del presente proyecto fue necesario elaborar un sistema básico de desarrollo con base en el PIC18F4550 [7]. Lo anterior es indispensable ya que el proceso de desarrollo y depuración de aplicaciones con microcontrolador así lo requiere.

El sistema de desarrollo es una versión personalizada del *PICDEM-FS-USB Demonstration Board* que Microchip ofrece, de tamaño compacto, versátil, autónomo, que ha sido diseñado pensando en la necesidad de contar con un modulo básico, fácilmente de usar con cualquier aplicación. El sistema ofrece un potente microcontrolador de la gama serie 18 equipado con puerto de comunicaciones USB, serie, I2C, paralelo, convertidores A/D, múltiples puerto I/O, los cuales están disponibles para ser conectados a diferentes circuitos y configuraciones sin necesidad de retirar el PIC de la tarjeta para programarlo cada vez que sea necesario modificar el programa grabado en el PIC. Lo anterior facilita la depuración del programa en el proceso de desarrollo.

El diseño tiene como base lo sugerido por el fabricante, con las modificaciones necesarias para trabajar con el *Bootloader* y adaptarlo con el proyecto final [8]. El diagrama electrónico del sistema de desarrollo se muestra en la figura 1.21.

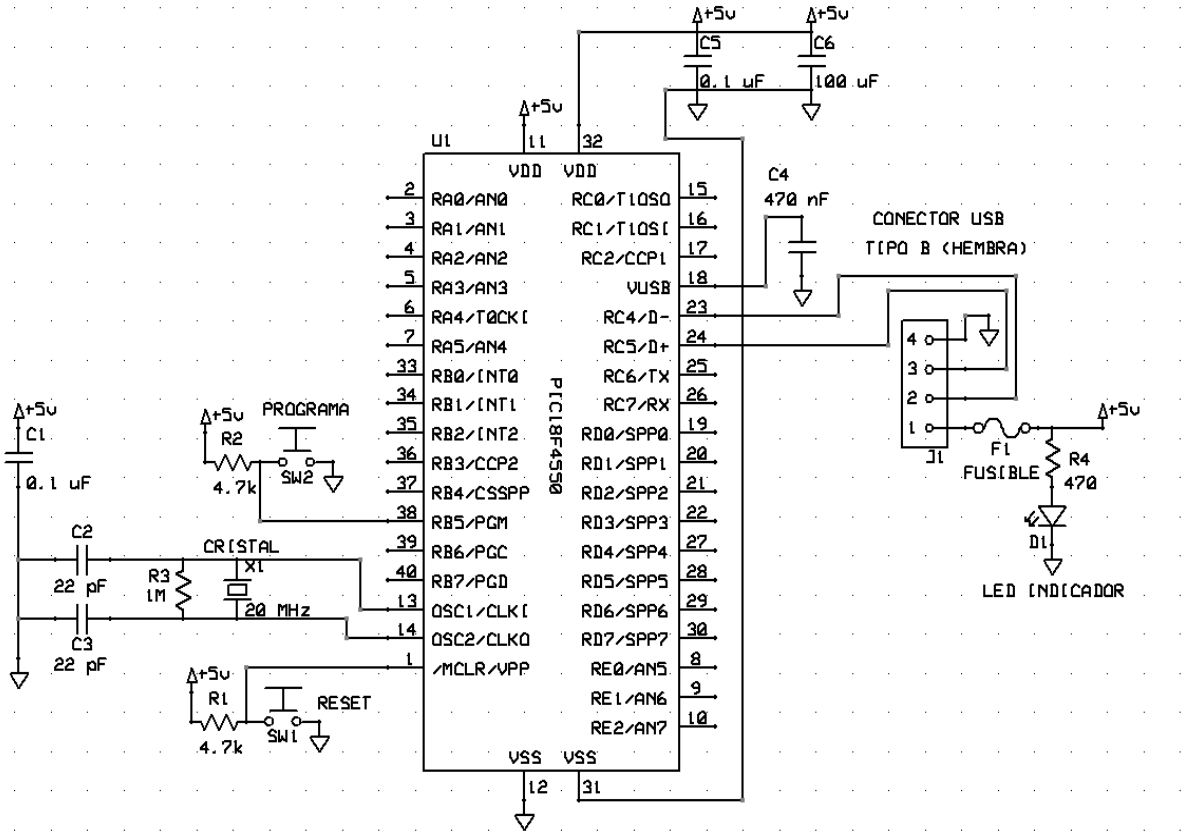


Figura 1.21. *Sistema de desarrollo.*

1.4.1 Descripción de los componentes

Conector USB: El conector del USB es un conector estándar del “tipo B”. Hay cuatro conexiones en un cable del USB, dos alimentan al circuito, mientras que los otros dos son las líneas de comunicaciones D+ y D-. La información se transfiere entre el ordenador y el PIC cuando se está programando, y mientras que el *firmware* envía o recibe datos a la computadora.

LED INDICADOR: Cuando la conexión USB es conectada, se enciende el led indicador. Hay que observar que los circuitos del dispositivo, pueden demandar más corriente (cantidad máxima 100mA que se permite ser extraída de un solo puerto USB). La alternativa es utilizar un regulador de voltaje estándar tal como el LM7805, que entonces proporciona +5V en 1000mA (1A).

PUSH BUTTONS: Los dos botones en el circuito se utilizan durante el proceso de programar la aplicación. Se etiquetan "RESET" y “PROGRAMA”, al oprimir el botón de reset es el equivalente de desconectar el cable del USB y de volver a conectarlo (lo cual la computadora debe reconocer el circuito e inicializar el *driver* correspondiente). Si se mantiene oprimido el botón de reset mientras que mantiene el botón de

programa, el circuito entrará en el modo *Bootloader*, lo que permitirá que una nueva aplicación sea cargada en el PIC (programa de PDFSUSB.EXE).

PIC18F4550: Es un microcontrolador popular que se comunica por USB 2.0, programable con 32Kbytes de la memoria flash y 2kbytes de SRAM. Tiene 13 entradas del A/D y 18 puertos de fines generales de la entrada-salida.

CRISTAL DE 20 MHz: En particular el oscilador tiene el trabajo de proporcionar una señal de reloj al PIC, con los capacitores correspondientes de 15pF y la resistencia de 1M se requieren para que oscile correctamente. Los cristales con otras frecuencias están disponibles, pero 20 MHz fue elegido para trabajar a *Full-Speed*

PORTAFUSIBLE: Esta parte evita el daño a la computadora y al circuito que contiene al microcontrolador en caso de condición del cortocircuito o de la sobrepaso de corriente.

Todos los componentes para el sistema de desarrollo están incluidos en la tabla 1.7.

Parte	Cantidad	Componente
C2, C3	2	Capacitor cerámico 22pF
C1, C5	2	Capacitor cerámico monolítico 0.1uF
C6	1	Capacitor electrolítico 100uF
C4	1	Capacitor cerámico monolítico 470nF
J1	1	Conector USB tipo B DIP
X1	1	Cristal 20 MHz
F1	1	Portafusible tipo europeo
U1	1	PIC18F4550
D1	1	Led
F1	1	Fusible 0.5 A
R1, R2	2	Resistencia 1/4W 4k7
R3	1	Resistencia 1/4W 1M
R4	1	Resistencia 1/4W 470
SW1, SW2	2	Push botón
U1	1	Base para circuito integrado 40 pos

Tabla 1.7. *Lista de componentes*.

1.4.2 Comunicación USB

Este circuito puede ser conectado al puerto USB de cualquier PC ahora bien, en caso de que el microcontrolador no contuviera ninguna información y se conectara a una computador personal, se generaría un mensaje del sistema operativo, tal como "*Dispositivo USB no reconocido*", por lo que es necesario crear el *firmware* que ira almacenado en la memoria de programa del microcontrolador, que contendrá todas las capacidades del dispositivo que se está creando para que sean anunciadas al sistema *Host* cuando éste sea conectado, así como el programa que permita crear

cualquier funcionalidad que utilice los periféricos del microcontrolador, una vez que éste sea enumerado.

Como se comentó, el *firmware* es el programa que se encuentra almacenado en el dispositivo, es decir, es el programa insertado en cualquier aparato electrónico por el fabricante, para proveer funcionalidades en un producto específico. Con el *firmware* el dispositivo informará al equipo *Host*, los números de identificación del fabricante y del dispositivo, así como, la clase, numero y tamaño de Endpoints, cantidad de energía que se necesita (máximo 500 mA), texto de identificación, etcétera. El desarrollo del firmware se encuentra en el anexo A.2.

Un aspecto importante a notar es que tanto el identificador del fabricante como el identificador de producto han sido seleccionados aleatoriamente, debido a que estos códigos deben ser proveídos por el *USB-IF* (*USB implementers Forum***)**, si el producto va a ser comercializado, por lo que para el objetivo perseguido, esto no es necesario, estos valores denominados *VID* y *PID*, serán utilizados mas adelante en el desarrollo del *software de operación*, ya que el sistema operativo identifica a los dispositivos USB conectados al sistema leyendo su *VID* y *PID*, que deberán ser únicos para cada dispositivo.

Para ello el *Host* periódicamente revisa su *Hub raíz* para averiguar si un dispositivo se ha conectado o desconectado. El *Host* lee los descriptores que poseen la información de las características de dicho dispositivo.

El siguiente código muestra como inicializar el dispositivo USB para poder transmitir por los endpoints declarados en los descriptores, cómo esperar la correcta enumeración del dispositivo y cómo recibir y enviar un paquete por los *endpoints* de entrada y salida, respectivamente.

```
usb_init();                                 //inicializamos el USB
usb_task();                                 //habilita periférico USB e
                                            interrupciones
usb_wait_for_enumeration();                 //esperamos hasta que el PicUSB sea
                                            configurado por el host

   while (TRUE)
   {
      if(usb_enumerated())                  //si el PicUSB está configurado
      {
         if (usb_kbhit(1))                  //si el endpoint de salida contiene
                                            datos del host
         {
            usb_get_packet(1,recibe,3);     //recibimos el paquete de tamaño
                                            3bytes del EP1 y almacenamos
                                            en recibe
            usb_put_packet(1,envia,3,USB_DTS_TOGGLE); //enviamos el paquete
                                                  de tamaño 3byte del EP1 al PC
```

En este apartado sólo se mostró una porción del código fuente, la versión completa se mostrará en el anexo A.5.

Después de obtener la información, el Host asigna y carga un *driver* al dispositivo; después busca un *archivo.INF* el cual informa el nombre y ubicación del *driver* ya existente. En caso de no ser encontrado, el software de la PC pedirá al usuario que cargue un driver propio del fabricante de dicho dispositivo. Los pasos de instalación del *drive*r y el *archivo. INF* se muestran en el anexo A.3.

1.4.3 Programación USB en CCS C

Para programar en lenguaje C, CCS suministra librerías para comunicar PIC con el PC utilizando el bus USB, mediante periféricos internos (familia PIC18F) o mediante dispositivos externos al PIC se deben conocer algunas funciones ya definidas en el programa CCS C, que se describirán a continuación:

Las librerías suministradas son:

- **pic_usb.h:** *driver* de capa hardware de la familia PIC16C765
- **pic_18usb.h:** *driver* de capa hardware de la familia PIC18F4550.
- **usb.h:** Definiciones y prototipos utilizados en el *driver* USB.
- **usb.c:** El USB *stack*, que maneja las interrupciones USB y el USB *setup Requests* en *Endpoint 0.*

Las funciones más importantes, entre otras muchas, son:

- **usb_init():** Inicializa la interface USB. Espera en un bucle infinito hasta que el periférico USB es conectado al bus (aunque eso no significa que ha sido enumerado por el PC). Habilita y utiliza la interrupción USB.

- **usb_task():** Si se utiliza una detección de conexión para la inicialización, entonces se debe llamar periódicamente a esta función para controlar el pin de detección de conexión. Cuando el PIC es conectado o desconectado del bus, esta función inicializa el periférico USB o resetea el USB *stack* y el periférico.

- **usb_wait_enumerated():** Espera hasta que el dispositivo sea configurado por Windows.

- **usb_enumerated():** Devuelve un *TRUE* si el dispositivo ha sido enumerado por el PC y, en este caso, el dispositivo entra en modo de operación normal y puede enviar y recibir paquetes de datos.

- **usb_kbhit():** Envía el paquete de tamaño un byte del *endpoint 1* a la PC, coloca el byte en el buffer de transmisión; en el caso de que esté lleno esperara hasta que pueda enviarlo.

Existen funciones específicas entre ellas:

- **usb_get_packet():** lee el paquete de tamaño byte(s) del *endpoint1*. Recibe un byte(s) del buffer de transmisión; en el caso de estar vacio esperará hasta que se reciba.

- **usb_put_packet():** envía el paquete de tamaño byte(s) del *endpoint1* a la PC, coloca el byte en el buffer de transmisión; en el caso de que esté lleno esperara hasta que pueda enviarlo.

En la figura 1.22 se muestra el área designada para el *firmware* del *Bootloader*.

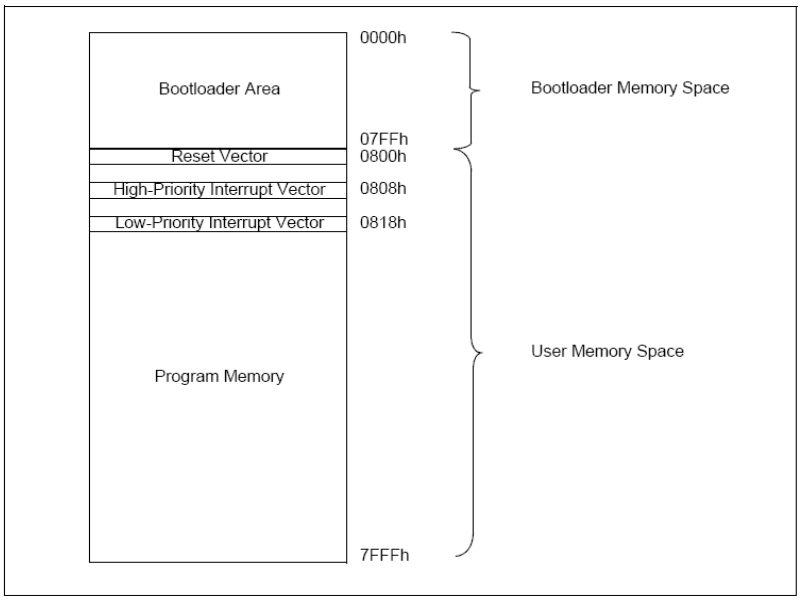


Figura 1.22. *Mapa de memoria del PIC18F4550.*

Es muy importante que se adicionen las líneas que se describen a continuación en el programa del microcontrolador. Sin estas líneas, el programa sobrescribirá el *Bootloader*, perdiendo así la funcionalidad.

```
#build(reset=0x800,interrupt=0x808)
#org 0x000, 0x7ff { }
```

Recordemos que el programa *Bootloader* reserva 2k de memoria. Siendo así, que el programa no puede exceder de 14k. Sin embargo, al tener nuestro programa funcionando podemos quitar las líneas de código, para no ocupar el espacio de memoria que ocupa el *Bootloader*.

Capítulo 2
Sistema propuesto

En el presente capítulo se describe el sistema propuesto para el estudio cinemático de un mecanismo de cuatro barras. El sistema consta de un prototipo mecánico y la interfaz gráfica desarrollados a partir de herramientas básicas. El sistema incluye un sistema electrónico controlado por el microcontrolador PIC18F4550, interfaz de operación desarrollado en Visual C++ para una computadora personal e interface USB entre el sistema electrónico y la PC.

2.1 Esquema general

Se propone en el presente trabajo el desarrollo de un mecanismo de cuatro barras que permita complementar los métodos de enseñanza. El esquema adoptado en el presente proyecto consiste básicamente de los siguientes componentes:

- Un motor CD, *M*, marca JAMECO, rango de operación 12-24 VDC a 2 RPM
- Tres codificadores angulares incrementales HB5M, *C*1 a *C*3, de 1024x4 = 4096 cpr (cuentas por revolución).
- Un sistema electrónico con base en el microcontrolador PIC18F4550 para el control del motor en lazo abierto, registro de los codificadores e interface USB 2.0.
- Un sistema mecánico de cuatro barras y cuatro articulaciones cada una con acoplamiento a los codificadores HB5M.
- Una computadora personal con la interfaz gráfica para el análisis de posición, velocidad y aceleración desarrollado específicamente para esta aplicación.

El esquema general se muestra en la figura 2.1.

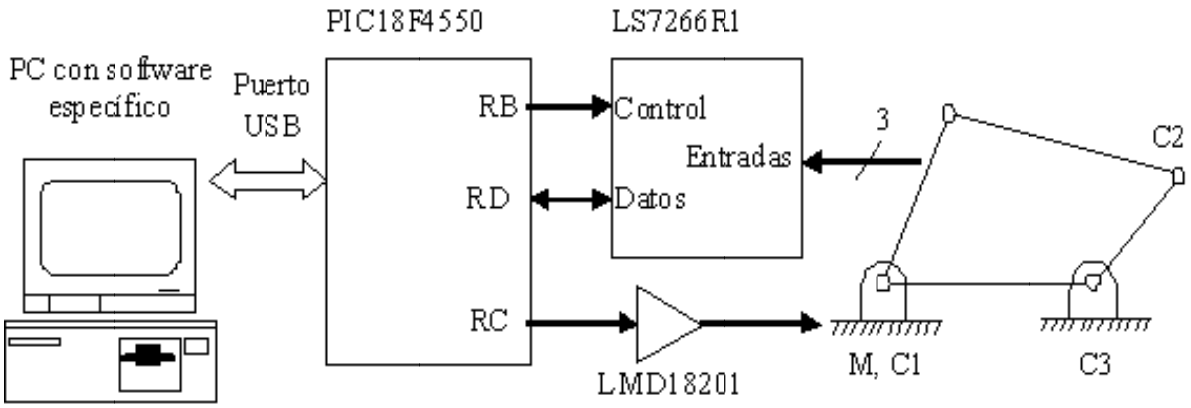


Figura 2.1. *Esquema general del prototipo electromecánico.*

En el esquema adoptado el sistema electrónico carece completamente de elementos para interactuar con el operador humano, pero el *software* en la computadora personal implementa la interface con el operador.

2.1.1 Sistema electromecánico

En forma general el mecanismo de cuatro barras es un dispositivo electromecánico con un grado de libertad, es decir, el mecanismo está formado por tres eslabones de aluminio conectados entre sí mediante pernos embalados, figura 2.2. En los extremos de dichos eslabones se colocan tres codificadores ópticos HB5M (Hollow Bore Optical Encoder) para medir los ángulos de cada eslabón, como se muestra en la figura 2.3.

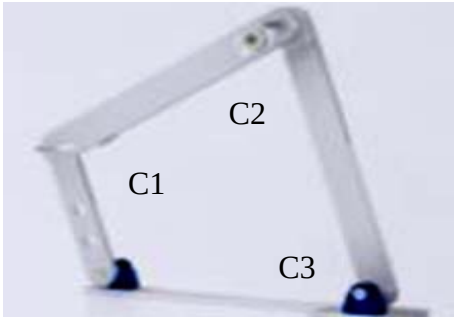


Figura 2.2. *Prototipo mecánico.*

Figura 2.3. *Hollow Bore Optical Encoder.*

Adicionalmente, un motor impulsor de CD se conecta al eslabón impulsor (eslabón de entrada). Dichos elementos se conectan mediante una tarjeta electrónica para la interface entre los dispositivos antes mencionados a una computadora personal que permite controlar y registrar la velocidad del motor y la posición de cada codificador. La información de los codificadores se registra y almacena en la computadora y mediante las ecuaciones dinámicas se obtienen los parámetros requeridos para el análisis cinemático. Cabe mencionar, que se cuenta con una interfaz gráfica para computadora.

El anexo A.7 muestra los planos de construcción del prototipo mecánico.

2.1.2 Sistema electrónico

El tarjeta desarrollada para esta aplicación esta sobre la base de un microcontrolador PIC18F4550 con interface USB. En esta aplicación, el microcontrolador maneja dispositivos periféricos para el control de movimiento de lazo abierto de un motor CD y tres codificadores angulares ópticos, como se muestra en la figura 2.4.

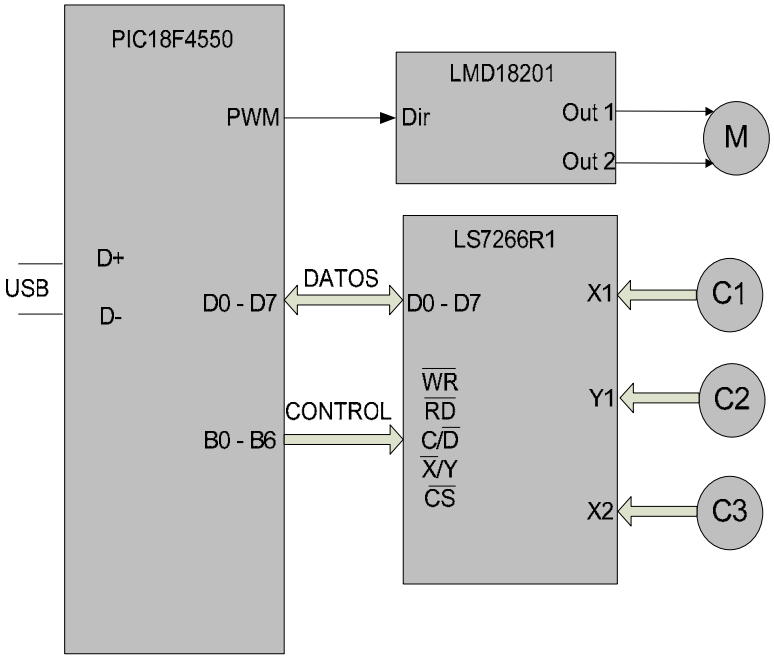


Figura 2.4. *Sistema electrónico.*

El microcontrolador genera una señal modulada por ancho de pulso para (PWM1) para controlar el movimiento del motor impulsor. La señal es amplificada por una etapa de potencia (Puente H LMD1820 IT) para cumplir con los requerimientos de corriente y voltaje necesarios en el motor. La señal de PWM esta configurada a 50% del ciclo de trabajo que equivale a 0% en movimiento en posición angular; cuando se inicia el movimiento del motor en sentido contrario a las manecillas del reloj (sentido positivo), se tiene un ciclo de trabajo de 0%; cuando se desea volver a iniciar el proceso, dicha señal de PWM esta configurada al 100% de ciclo de trabajo lo cual se inicia el movimiento del motor en sentido a las manecillas del reloj (sentido negativo).

Los codificadores ópticos son usados para medir la posición angular de cada eslabón del mecanismo de cuatro barras. En este prototipo el codificador óptico es de 1024 cuentas por revolución. Las señales de cuadratura A y B son procesadas por un circuito integrado LS7266R1 que es configurado como un contador de cuadratura y

realiza un conteo de los flancos ascendentes o descendentes de los pulsos recibidos. Este conteo puede determinar la posición angular del mecanismo.

En el esquema propuesto, la interface USB es utilizada para enviar y recibir datos del PIC18F4550, ya que controla al circuito LS7266R1 es un circuito de interface que sirve para la comunicación entre los tres codificadores al microcontrolador. También, genera la señal de PWM ya mencionada anteriormente. En este sentido, la PC lleva a cabo procesos de alto nivel constituyéndose como la interfaz con el usuario. En el *software* se tiene una manera de enviarle datos al microcontrolador, mediante los botones de control, desplegando y graficando respectivamente los valores obtenidos por los codificadores.

2.1.3 Interfaz gráfica

El objetivo general del presente proyecto es desarrollar una unidad didáctica que sirva de apoyo en el proceso de enseñanza-aprendizaje en los laboratorios de análisis y síntesis de mecanismos para estudiantes de licenciatura y el posgrado en ingeniería mecánica.

Como parte de los objetivos particulares se tienes los siguientes:

- Una interfaz grafica que permita controlar y registrar la información proveniente de los sensores y actuadores. La información de los sensores se almacenará en la computadora y mediante las ecuaciones dinámicas se tendrá acceso a los parámetros requeridos para el análisis cinemático

- Graficar en tiempo real la posición, velocidad, aceleración angular respectivamente de cada uno de los elementos mediante una *interfaz gráfica*.

Se desarrolló la interfaz de operación en Visual C++ 2008 que permite la interacción con el prototipo. Desde éste se controla el encendido, la velocidad del motor y se realiza la adquisición de datos. Además permite la visualización, por medio de graficas en tiempo real, de las variables de interés. Las gráficas se implementaron con un Control ActiveX de libre distribución [9].

La interfaz de operación para el presente proyecto es una aplicación MFC basada en cuadros de diálogo. Para el manejo de las directivas de USB 2.0 se necesita un archivo DLL que ofrece Microchip con funciones para comunicarse con su dispositivo. Además de una nueva clase USB para manipular las funciones que ayudan a la comunicación USB. Con esto se tiene enlazado el proyecto MFC y la librería USB de Microchip [10].

2.2 Sistema electrónico

El sistema electrónico esta constituido por tres bloques: el subsistema de control, de medición e impulsor respectivamente que serán descritos a continuación. Su elemento central es el microcontrolador PIC18F4550 gobernando una serie de dispositivos electrónicos que adecuan las señales a los requerimientos eléctricos del prototipo. El anexo A.4 muestra el diagrama electrónico completo. A continuación, una descripción detallada del sistema electrónico para este proyecto.

2.2.1 Subsistema de control

El diagrama se basó en el sugerido por el fabricante, con las modificaciones necesarias para adaptarlo al proyecto. El diagrama particular para conexión del microcontrolador se muestra en la figura 2.5.

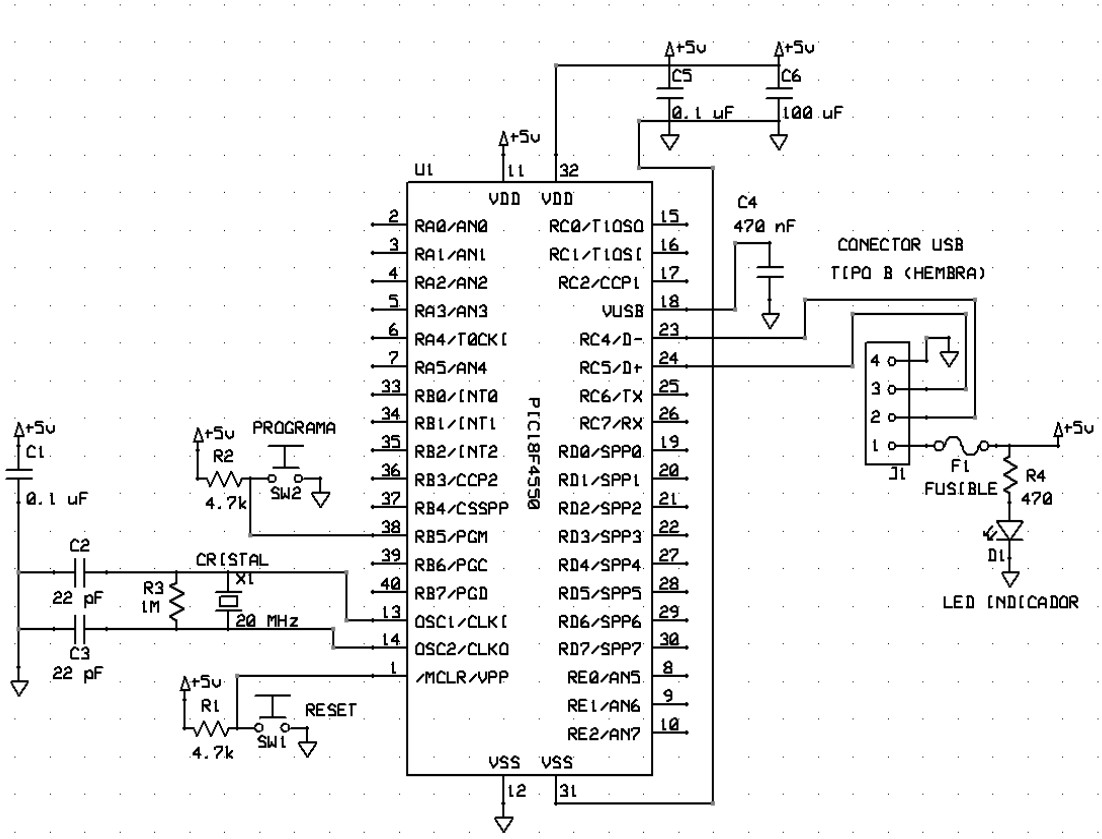


Figura 2.5. *Arquitectura del microcontrolador PIC18F4550.*

Este es el circuito básico para poder iniciar el trabajo con el dispositivo USB, al que pueden agregarse elementos para otras funcionalidades contenidas en el dispositivo. El microcontrolador cuenta con soporte para la interface USB. Opera con un oscilador de 20 MHz, para que la comunicación sea de alta velocidad y capacitores de 22 pF

para mayor estabilidad del oscilador. El conector USB es de cuatro pines (+5, GND, D-, D+) que proporcionan la alimentación de 5 V que requiere la tarjeta. Las entradas D- y D+ son para entrada y salida de datos. Además se necesita una fuente de alimentación de 12 V 1A regulado para alimentar al puente H. El circuito electrónico que proporciona este nivel de voltajes se muestra en la figura 2.6.

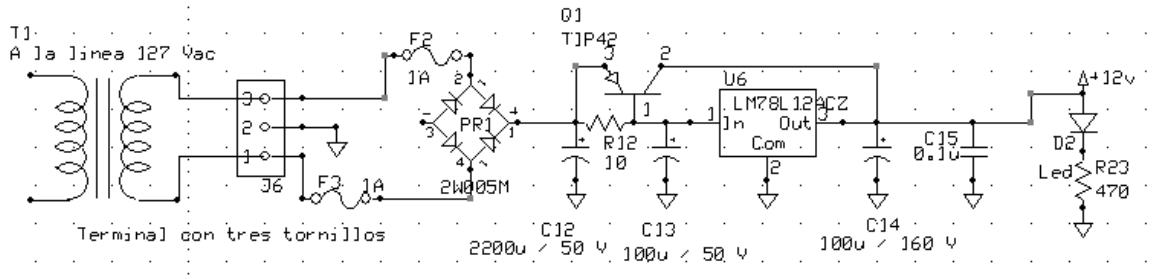
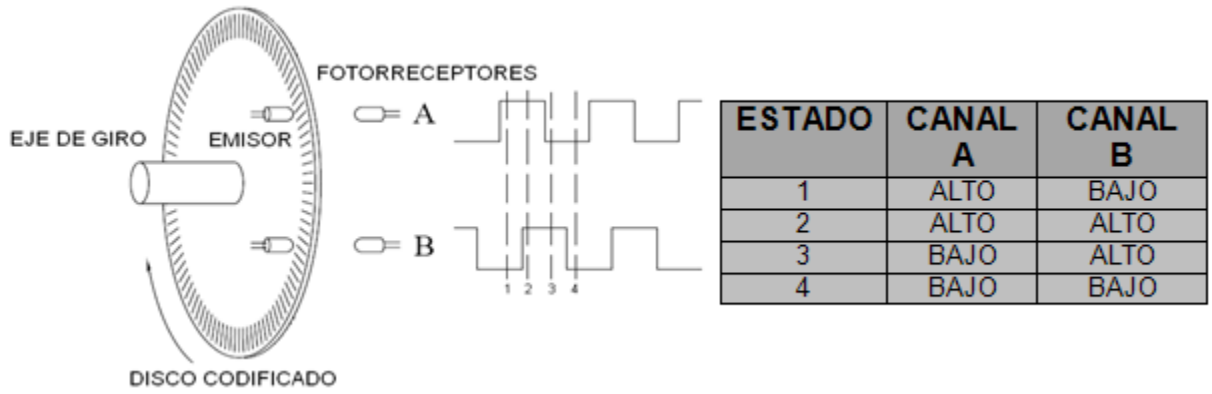


Figura 2.6. *Fuente de alimentación.*

La fuente es el arreglo clásico para el regulador positivo de voltaje de la serie 78XX con un amplificador de corriente con el transistor TIP42, lo que proporciona hasta 3 A de salida.

2.2.2 Subsistema de medición

El codificador en cuadratura se localiza en cada uno de los ejes del mecanismo de cuatro barras. La figura 2.7 muestra la composición de un codificador en cuadratura, generalmente consta de un disco unido a un eje giratorio, con sectores opacos y transparentes; de pulsos defasadas un cuarto de ciclo. El conteo de los pulsos determina la posición de los tres eslabones, mientras que la diferencia de fase determina la dirección de rotación.



ESTADO	CANAL A	CANAL B
1	ALTO	BAJO
2	ALTO	ALTO
3	BAJO	ALTO
4	BAJO	BAJO

Figura 2.7. *Esquemático de un codificador incremental.*

Un decodificador de cuadratura toma las señales de los codificadores y realiza un conteo de los flancos ascendentes o descendentes de los pulsos recibidos. Este conteo puede determinar la posición.

La solución consiste en desarrollar un sistema que realice la decodificación de las señales en cuadratura y que sea capaz de transmitirla por puerto USB a una computadora personal.

Funcionamiento del circuito interface

Para la decodificación se eligió un circuito integrado de fin específico, se trata del decodificador *LS7266R1*, con capacidad para decodificar dos ejes, dos contadores de 24 bits independientes y un bus de datos de 8 bits. Las instrucciones de control son de 5 bits y el modo de conteo puede ser configurado en X1, X2 y X4 según se requiera la resolución. Opera a una frecuencia máxima de 17 MHz para conteo en cuadratura.

En lo que respecta a la interface de comunicación se eligió el microcontrolador PIC18F4550 con puerto USB, además de que implementa fácilmente elementos de *hardware* y *firmware*.

El microcontrolador realizara básicamente las siguientes funciones:

- Envío de comandos de control para el codificador.
- Recepción de información proveniente del codificador.
- Envío de información por puerto USB hacia la PC.

El control sobre el decodificador en cuadratura LS7266R1 se elaboro con base en la configuración de sus registros de control.

La figura 2.8 muestra las opciones disponibles para los registros de bandera, indica la información del estado actual de los contadores “CNTR’s” para leer la salida del bus de datos. Destaca el bit 4 que en estado activo indica una salida excesiva en ruido.

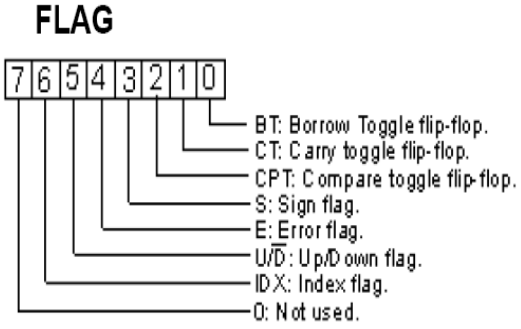


Figura 2.8. *Registros de bandera.*

Los registros de reinicio y carga ó “RLD” se muestran en la figura 2.9 ; controlan la transferencia de información de los registros “PRESET” a los contadores “CNTR”, y de estos a los registros de salida “output latch” u “OL”, además pueden configurar el reinicio del contador, bandera y el byte indicador de secuencia BP.

La configuración del conteo se realizará en modo de cuadratura X4 para obtener la resolución presente en la unión de los eslabones.

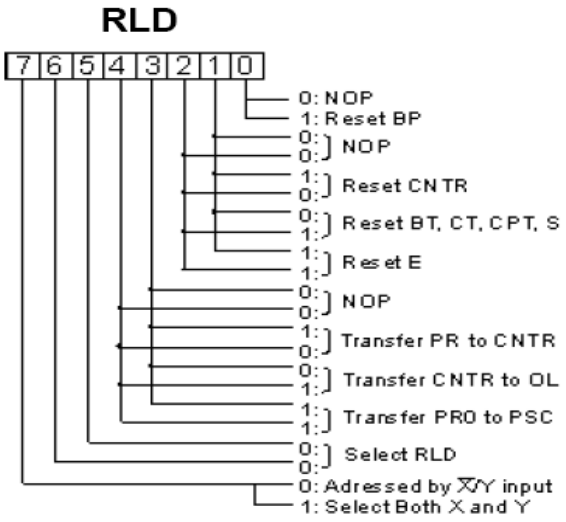


Figura 2.9. *Registros de inicio y carga.*

De esta forma en el registro de configuración de conteo ó CMR, figura 2.10, se determina configurar en cuadratura X4, en modo BCD y modo normal de conteo.

El microcontrolador será programado para realizar las siguientes tareas sobre el decodificador:

- Escribir los registros *preset*, de cada eje X o Y en el decodificador.
- El flujo de entrada o salida de información habilitando el eje correspondiente.
- Configuración del modo de conteo.
- Leer la información en los registros de salida *output latch*.
- Lectura de las banderas de control.

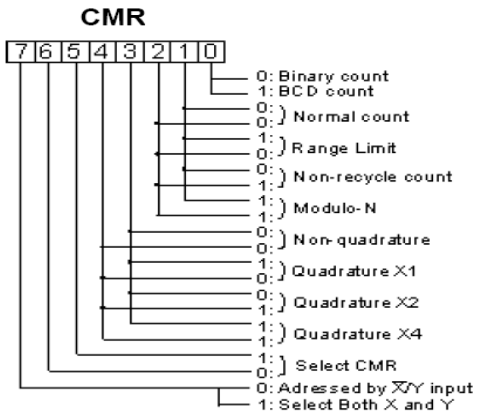


Figura 2.10. *Registros de configuración de conteo.*

El modo de direccionamiento para algunas instrucciones que se implementaron en el programa, se ilustran en la figura 2.11.

REGISTER ADRESSING MODES

D7	D6	D5	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	X/$\overline{Y}$	$\overline{CS}$	Function
X	X	X	X	X	X	X	1	Disable Chip
0	0	0	1	1	⊓⊔	0	0	Write to XRLD
0	0	0	1	1	⊓⊔	1	0	Write to YRLD
1	0	0	1	1	⊓⊔	X	0	Write to both XRLD and YRLD
0	0	1	1	1	⊓⊔	0	0	Write to XCMR
0	0	1	1	1	⊓⊔	1	0	Write to YCMR
1	0	1	1	1	⊓⊔	X	0	Write to both XCMR and YCMR
0	1	0	1	1	⊓⊔	0	0	Write to XIOR
0	1	0	1	1	⊓⊔	1	0	Write to YIOR
1	1	0	1	1	⊓⊔	X	0	Write to both XIOR and YIOR
0	1	1	1	1	⊓⊔	0	0	Write to XIDR
0	1	1	1	1	⊓⊔	1	0	Write to YIDR
1	1	1	1	1	⊓⊔	X	0	Write to both XIDR and YIDR
X	X	X	0	1	⊓⊔	0	0	Write to X Preset Register, increment Address Counter
X	X	X	0	1	⊓⊔	1	0	Write to Y Preset Register, increment Address Counter
X	X	X	0	⊓⊔	1	0	0	Read X Output Latch, increment Address Counter
X	X	X	0	⊓⊔	1	1	0	Read Y Output Latch, increment Address Counter
X	X	X	1	⊓⊔	1	0	0	Read X FLAG Register
X	X	X	1	⊓⊔	1	1	0	Read Y FLAG Register

Figura 2.11. *Modos de direccionamiento para el codificador LS7266R1.*

Básicamente la programación se desarrolló tomando en cuenta los ciclos de lectura y escritura de datos o registros de control, así como habilitación y deshabilitación del circuito integrado. La figura 2.12 muestra el diagrama del circuito integrado LS7266R1, en esta podemos ver el *bus* de datos, la entrada de las señales de cuadratura y el bus de control; las entradas para configurar ciertos parámetros de interés, como lo son las terminales 13-17.

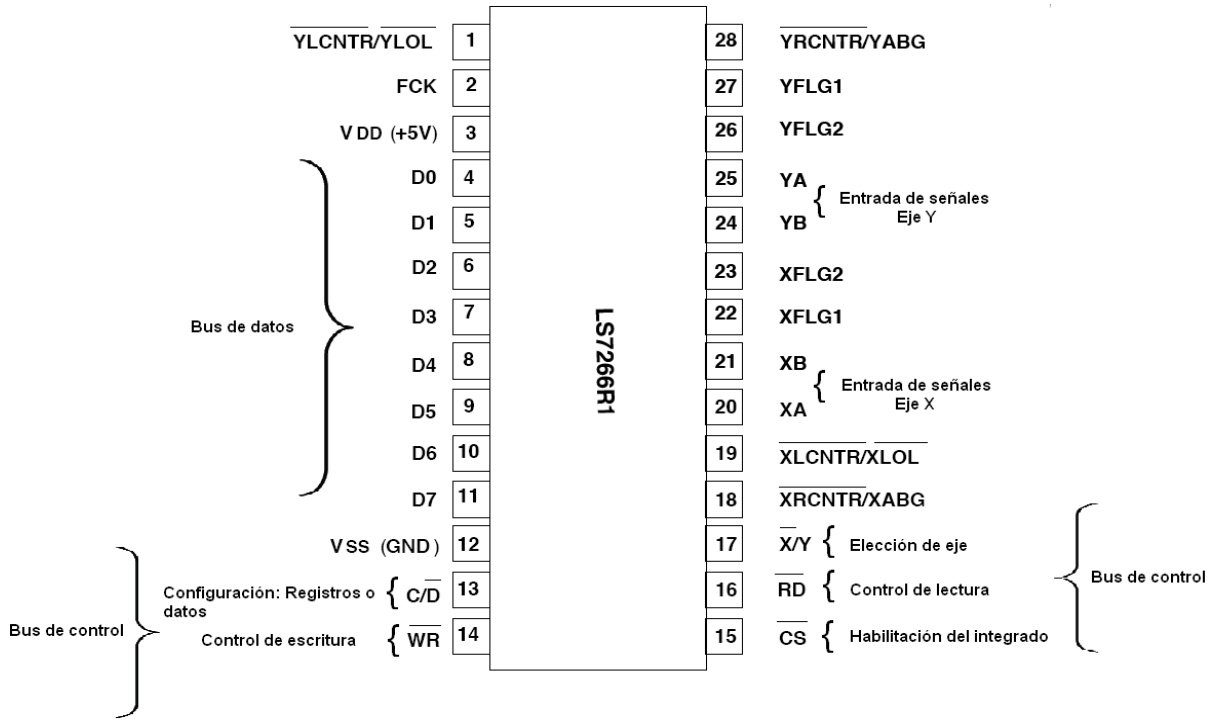


Figura 2.12. *Patigrama del decodificador LS7266R1.*

La secuencia de lectura se ilustra en la figura 2.13 en la cual RD (patilla 16) en nivel bajo configura al integrado para leer información de los “OL’s” proveniente de los registros PRESET.

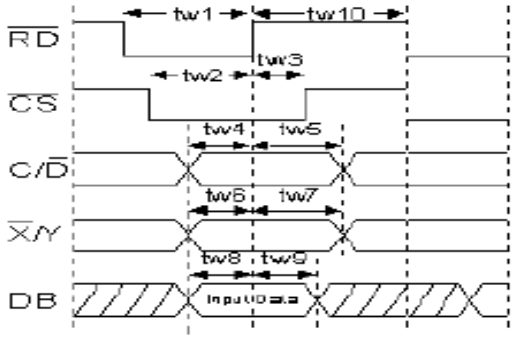


Figura 2.13. *Ciclo de lectura.*

La señal C/D en estado alto identifica si la lectura es de un registro de control o en nivel bajo si la lectura es de un registro de datos. La señal X/Y es la selección entre cualquiera de los dos ejes X bajo o Y alto.

Para el ciclo de escritura, mostrado en la figura 2.14, la señal RD pasa a nivel bajo y configura al integrado para escribir, espera el paso a nivel bajo de CS para habilitar, el tiempo mínimo de operación tw1 y tw2 es de 45 ns para ambas señales. Entonces empieza el envío de información, y esta se deshabilita cuando RD cambia a estado

alto, lo que provoca que sólo se tenga un tiempo mínimo de transmisión correspondiente a tw9 de 10 ns antes de quedar deshabilitada la función escribir.

El tiempo de espera mínimo para iniciar un nuevo ciclo de escritura es tw10 de 90 ns. De igual forma C/D identifica entre datos o registros de control y X/Y el eje en uso.

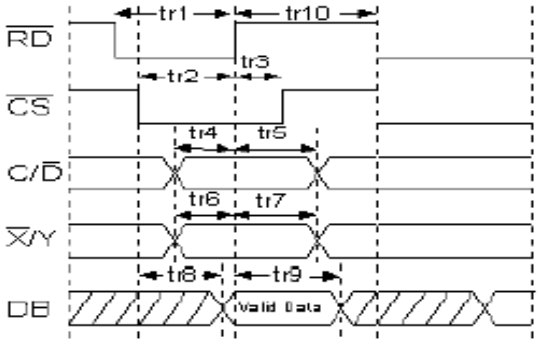


Figura 2.14. *Ciclo de escritura*

Configuración del circuito interface

Una vez descrito el funcionamiento del LS7266R1, se dispone a configurar los dos circuitos integrados. En este proyecto se utilizan tres codificadores angulares, puesto que por cada circuito integrado se pueden utilizar dos codificadores.

En la tabla 2.1 se describen los registros y funciones utilizadas, para configurar al circuito de interface LS7266R1 conectado su puerto de control al puerto B del microcontrolador. La configuración se lleva a cabo en *modo de escritura*.

REGISTRO	HABILITACIÓN	FUNCIÓN	HEXADECIMAL	BINARIO
RLD	X e Y	Reset E	0X86	b10000110
RLD	X e Y	Reset BP	0x81	b10000001
RLD	X e Y	PRO to PSC	0x98	b10011000
IOR	X e Y	Inputs A and B	0xC1	b11000001
CMR	X e Y	X4,BCD count, Normal count	0xB8	b10111000
RLD	X e Y	Reset CNTR	0x82	b10000010

Tabla 2.1. *Registros de configuración.*

En la tabla 2.2 se describen los registros y funciones utilizadas, para configurar al circuito de interface LS7266R1, ahora en forma "*modo lectura*"

REGISTRO	HABILITACIÓN	FUNCIÓN	HEXADECIMAL	BINARIO
RLD	X e Y	CNTR to OL	0X90	b10010000
RLD	X e Y	Reset BP	0x81	b10000001

Tabla 2.2. *Registros de lectura.*

La figura 2.15 muestra el diagrama de los decodificadores de señales de cuadratura. Como se ha dicho el puerto B del microcontrolador es utilizado para habilitar al decodificador, así como para la elección del eje, modo de operación a ejecutar; leer o escribir. El puerto D se acondiciona para el intercambio de datos y de registros de control. Obsérvese que los pines 1, 18, 19 y 28 están en estado alto. Lo cual configura al decodificador en carga directa, es decir, se pasa la información de los CNTRS hacia los registros de salida sin verificar el índice de las señales en cuadratura.

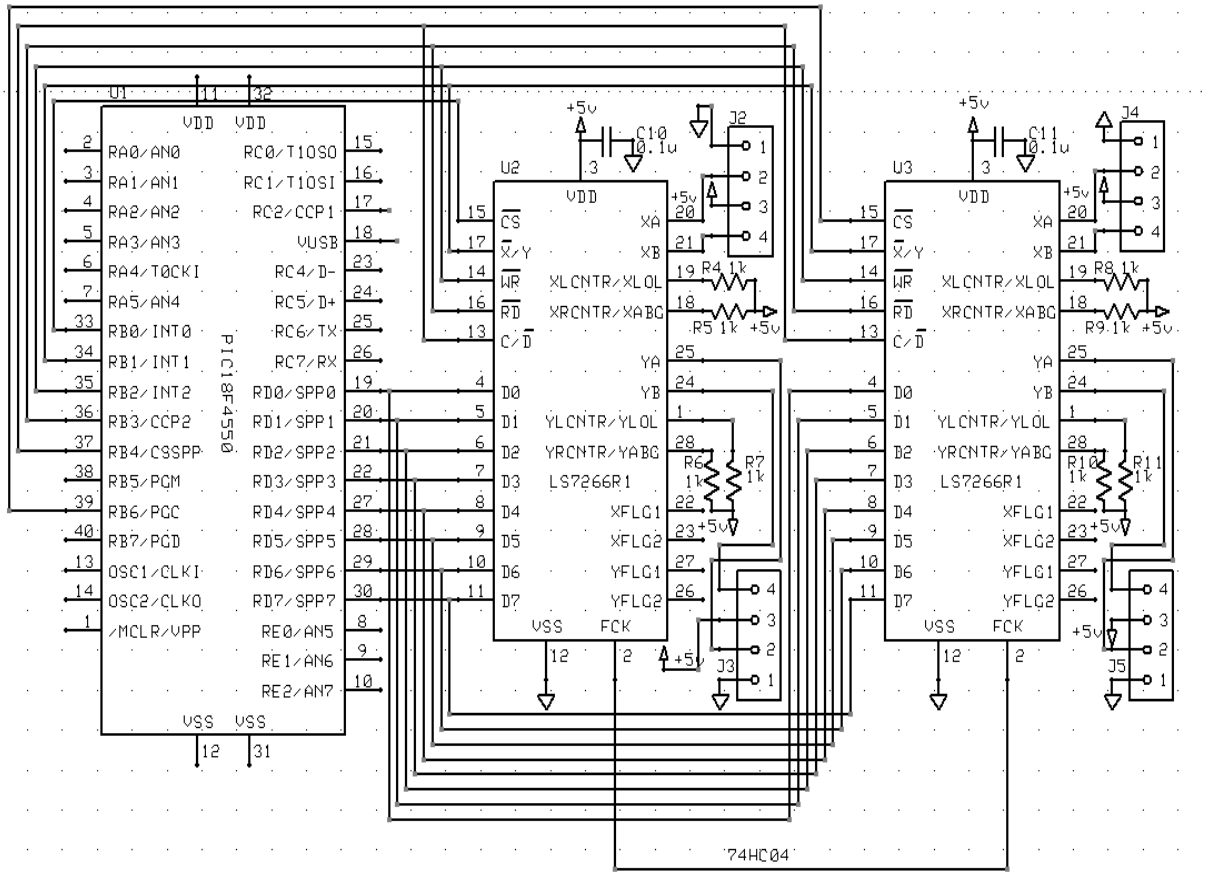


Figura 2.15. *Diagrama del subsistema de medición.*

Se utiliza un circuito de reloj con el inversor 74HC04 y un cristal de cuarzo de 6 MHz, para generar la señal de reloj para los circuitos decodificadores LS7266R1. Por especificación del fabricante, la señal FCK en el decodificador para conteo en cuadratura debe cumplir con la siguiente condición: $f_{FCK} \geq 8f_{QA}$ ó $8f_{QA,}$ en donde f_{QA} es la frecuencia de las transiciones de las señales en cuadratura. El diagrama se muestra en la figura 2.16.

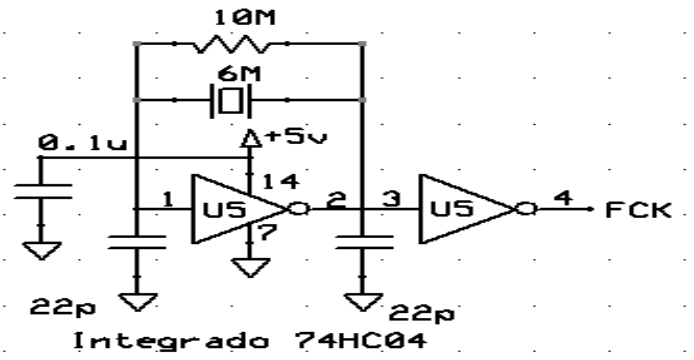


Figura 2.16. *Arquitectura interna del circuito integrado 74HC04.*

2.2.3 Subsistema impulsor

La señal de PWM que genera el microcontrolador es de 10 bits de resolución y la salida del PIC18F4550 es el pin 17 (CCP1) hacia el circuito LMD18201. El LMD18201 es un puente H de 3 A para operar hasta 55 V, figura 2.17 [11]. Tiene aplicaciones tales como en mecanismos de posición y velocidad, motores a pasos, Plotters, impresoras, etcétera. Es necesario usar una fuente de 12 V para aumentar la potencia de la señal de PWM y así controlar el motor.

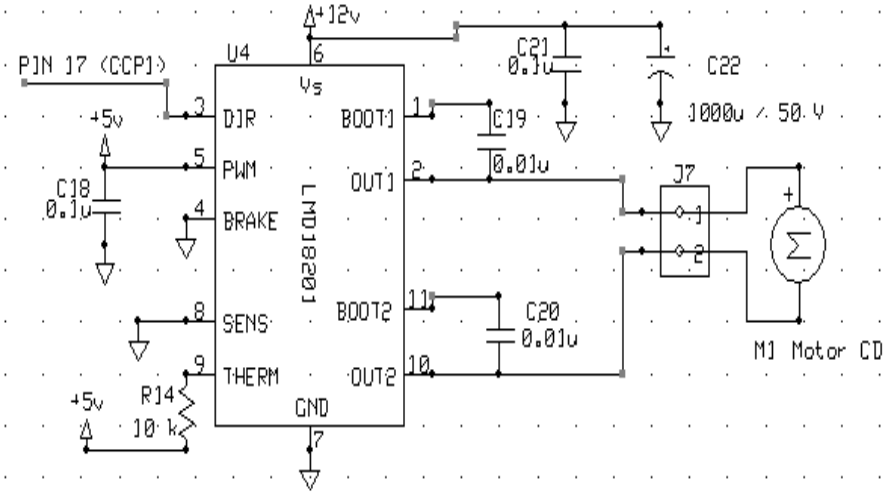


Figura 2.17 *Diagrama del subsistema impulsor.*

2.2.4 Algoritmo de operación

El *software* en el sistema permite el control del movimiento del motor impulsor y los algoritmos de medición de los codificadores. El código fuente esta escrito en lenguaje C para el PIC18F4550 y se omite en el presente trabajo debido a su gran longitud, pero se describe en el anexo A.5.

Como herramienta de programación se utilizo lenguaje C para PIC, con el compilador PIC C. La idea de programación consiste en configurar los puertos del microcontrolador para intercambiar información: iniciarlo y configurarlo. Los pasos de programación son las siguientes:

- Definir el puerto B como bus de control, el puerto D como bus de datos.
- Se envían datos para inicialización a los decodificadores.
- Entonces se inicia el protocolo de comunicación de registros de control y datos entre el microcontrolador y los decodificadores.
- Se envían comandos de control para el modo “lectura” a los decodificadores.
    - Se habilita el primer decodificador con ayuda del CS (*chip select*); después se leen los valores de cada codificador con ayuda X/Y alternadamente y se deshabilita el decodificador.
    - Se habilita el segundo decodificador y se deshabilita al primer decodificador, después se lee el valor del tercer codificador.
- Se espera a recibir un comando USB de la computqdora para ejecutar la operación:
    - Se envían los datos de los tres codificadores con intervalos de tiempo en microsegundos por la interface USB.
    - Se actualiza el valor de la señal de PWM que impulsa el motor.

El funcionamiento se encuentra representado por el diagrama de flujo de la figura 2.18.

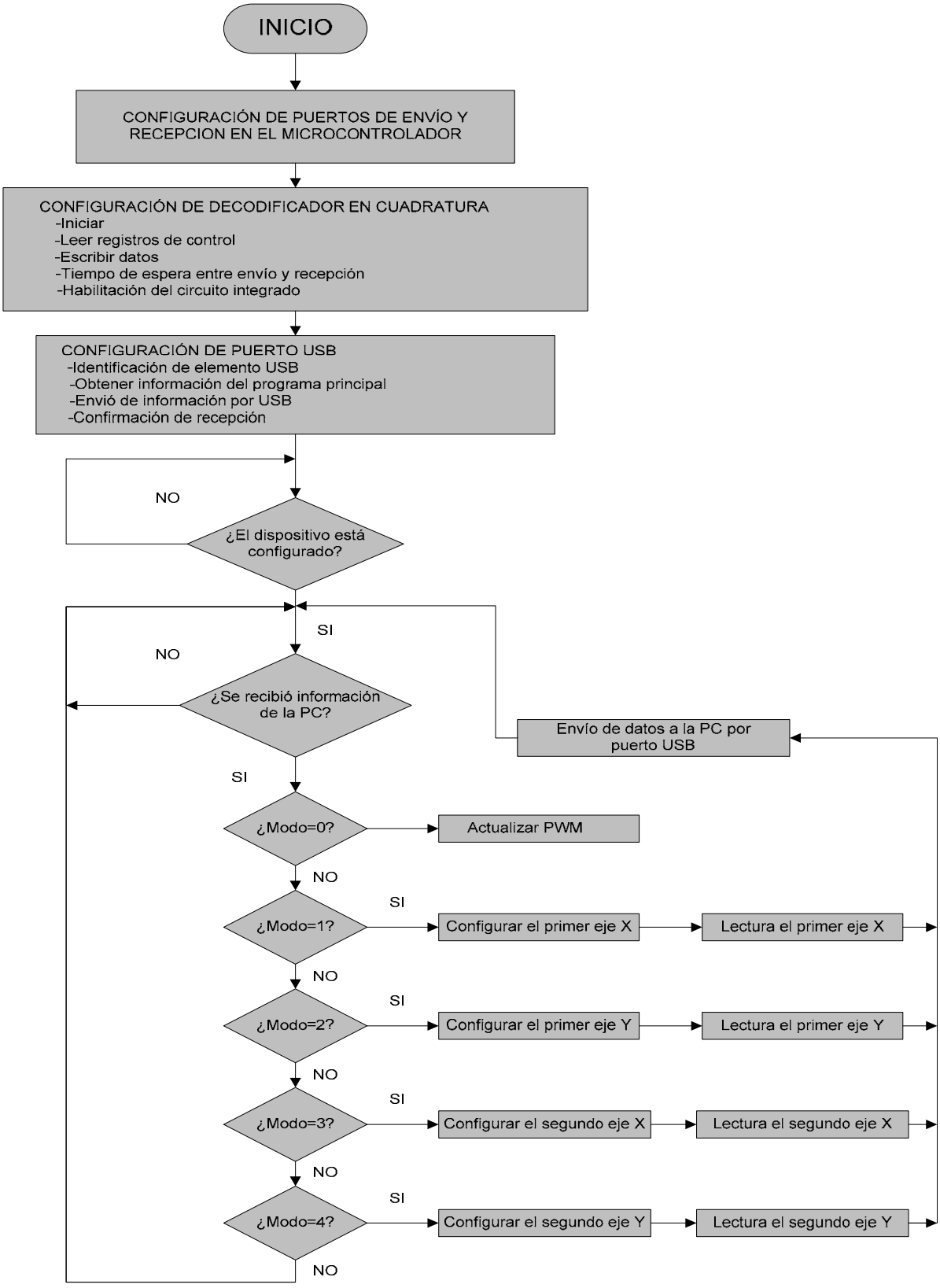


Figura 2.18. *Diagrama de flujo del microcontrolador*.

2.3 Implementación de la interfaz gráfica

El sistema electrónico, es controlado por el PIC18F4550 que implementa el algoritmo de operación descrito en 2.2.4. El prototipo de mecanismo de cuatro barras tiene como software de operación un programa para la computadora personal que es utilizado como ventana de despliegue de resultados y envió de comandos al sistema electrónico. La interfaz de operación fue desarrollado en Visual C++ 2008 y es ejecutable desde Windows XP.

2.3.1 Descripción en el ámbito del usuario

El *software* que implementa la interfaz para el control del motor de CD, y la adquisición de datos de los dispositivos a través de la interfaz USB, es el mostrado en la figura 2.19. El usuario tiene las siguientes posibilidades en la interfaz grafica:

1. Una área de despliegue grafico x-y para la presentación de los datos obtenidos de los codificadores ópticos y del análisis de posición, velocidad y aceleración.
2. Una área de botones en donde el operador ingresa los datos necesarios para realizar la prueba, para ello se cuenta con los siguientes elementos:

    - Botón 1 I*NICIO*. Inicia el movimiento del motor impulsor con una velocidad inicial de 1 RPM.
    - Botón 2 *RESET.* Regresa el mecanismo a su posición de inicial
    - Botón 3 *STOP*. Para el proceso
    - Botón 4 *VELOCIDAD.* Actualiza el valor de la velocidad del motor impulsor
        - Caja de texto 1. Sirve para establecer el nuevo valor de la velocidad del motor impulsor.
    - Cajas de texto 2-10. Muestra posición, velocidad y aceleración angular respectivamente de los tres eslabones en estudio.

Los requisitos mínimos para operar el sistema son:

- Computadora Pentium IV, puerto USB, 512 MB RAM, SVGA, Windows 2000/XP, VISUAL C++ 2008.
- Sistema electrónico incluyendo motor CD y tres codificadores ensamblados en el prototipo mecánico.

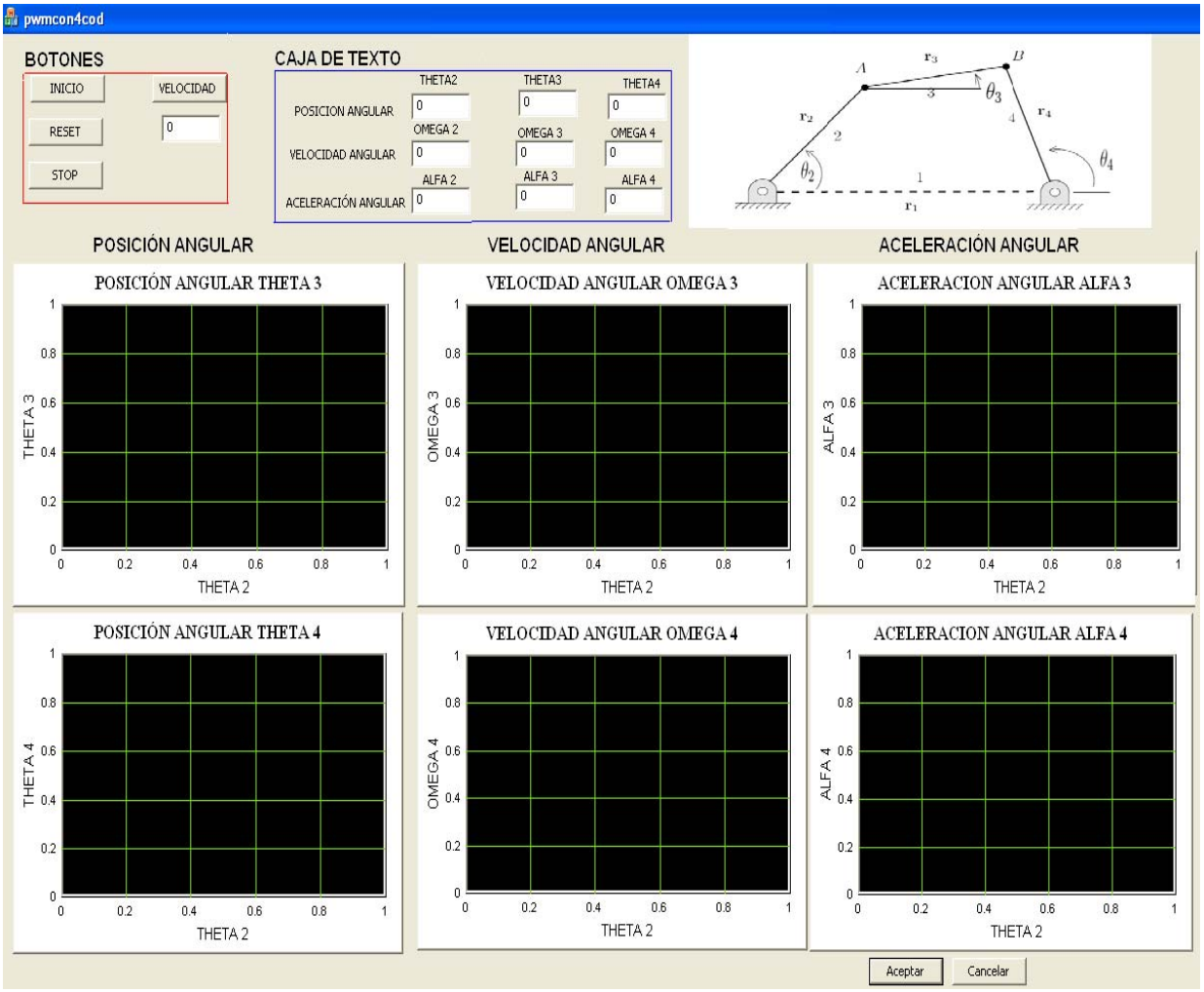


Figura 2.19. *Interfaz gráfica.*

2.3.2. Descripción en el ámbito del programador

La presente sección describe la programación utilizada para la implementación en tiempo real de la aplicación “pwmcon4cod” que se utiliza como software de operación para el mecanismo de cuatro barras.

El programa se desarrolló usando la aplicación MFC como una aplicación “basada en cuadro de dialogo”, sin documento ni vista.

En primer lugar se copian los siguientes archivos al directorio raíz del proyecto “pwmcon4cod”. Las librerías y los archivos de cabeceras son proporcionados por Microchip. La clase USB.h contiene funciones para enviar y recibir datos del microcontrolador, será descrita más detalladamente a continuación:

- *usb2550.dll*:
- *usb2550.lib*:

- *usb2550.h*:
- *USB.cpp*:
- *USB.h*:

Para graficar en dos dimensiones se requirió de un control ActiveX, que nos permite trazar datos en dos dimensiones. A pesar de la amplia serie de controles que vienen en Visual C++, no existe un dispositivo en la caja de control que proporcione una sencilla y clara visualización de datos en 2D [12].

La figura 2.20 muestra las clases en la implementación del programa para el mecanismo de cuatro barras para propósito didáctico. Debido a la gran extensión del código fuente, en la presente sección sólo se describen las porciones más significativas del programa y se omite el listado completo.

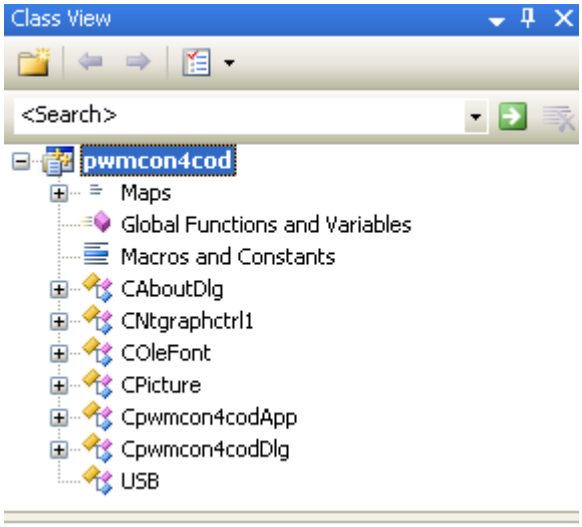


Figura 2.20. *Clases del proyecto.*

Clase CAboutDlg

La clase CAboutDlg se utiliza para desplegar la caja de dialogo “Acerca de ….”, clase insertada por omisión.

```
// Cuadro de diálogo CAboutDlg utilizado para el comando Acerca de
class CAboutDlg : public CDialog
{
public:
	CAboutDlg();
// Datos del cuadro de diálogo
	enum { IDD = IDD_ABOUTBOX };
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // Compatibilidad
con DDX/DDV

// Implementación
protected:
```

```
      DECLARE_MESSAGE_MAP()
};
```

Clase CNtgraphctrl1

La clase CNtgraphctrl1 es un control ActiveX capaz de trazar un gran número de puntos XY y la actualización de uno o más lotes en la gráfica con los nuevos datos. El control posee múltiples propiedades, como el titulo de la grafica, el tipo de línea de trazado, el estilo de los puntos graficados y el ancho puede ser personalizado en tiempo de ejecución.

Clases COleFont, CPicture y Cpwmcon4codApp

Las clases *COleFont* y *CPicture* se insertan por omisión por tal motivo no se incluirán para el análisis descrito. La clase *Cpwmcon4codApp* implementa la instancia de aplicación para su ejecución como caja de diálogo por tal motivo no se incluirá para el análisis descrito.

Clase Cpwmcon4codDlg

La clase *Cpwmcon4codDlg* declara el tipo de datos de los elementos que contiene el código principal.

```
// Cuadro de diálogo de Cpwmcon4codDlg
class Cpwmcon4codDlg : public CDialog
{
// Construcción
public:
      Cpwmcon4codDlg(CWnd* pParent = NULL);     // Constructor estándar

// Datos del cuadro de diálogo
      enum { IDD = IDD_PWMCON4COD_DIALOG };

      protected:
      virtual void DoDataExchange(CDataExchange* pDX);     // Compatibilidad
con DDX/DDV
// Implementación
protected:
      HICON m_hIcon;

      // Funciones de asignación de mensajes generadas
      virtual BOOL OnInitDialog();//INICIA EL CUADRO DE DIALOGO
      afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
      afx_msg void OnPaint();
      afx_msg HCURSOR OnQueryDragIcon();
      DECLARE_MESSAGE_MAP()

public:
      USB myUSB;        //CONTIENE LAS FUNCIONES PARA COMUNICARSE CON EL PIC.
      afx_msg void OnBnClickedvelocidad();     //BOTON VELOCIDAD

      int m_pwm;        //CONTIENE EL VALOR DEL PWM, MODIFICAR VELOCIDAD.
```

```
	double m_leer;    //CONTIENE EL VALOR DEL PRIMER CODIFICADOR (THETA2)
	double m_leer2;   //CONTIENE EL VALOR DEL SEGUNDO CODIFICADOR
	double m_leer3;   //CONTIENE EL VALOR DEL TERCER CODIFICADOR
	double m_leer4;   //CONTIENE EL VALOR DEL CUARTO CODIFICADOR (THETA4)
	double m_theta3;  //CONTIENE EL VALOR DE THETA 3 INDIRECTAMENTE.

	double m_omega22; //CONTIENE EL VALOR OMEGA2
	double m_omega33; //CONTIENE EL VALOR OMEGA3
	double m_omega44; //CONTIENE EL VALOR OMEGA4
	double m_alfa22;  //CONTIENE EL VALOR ALFA2
	double m_alfa33;  //CONTIENE EL VALOR ALFA3
	double m_alfa44;  //CONTIENE EL VALOR ALFA4


	afx_msg void OnTimer(UINT_PTR nIDEvent);//SE ADQUIEREN LOS VALORES EN

                                                // TIEMPO REAL
private:
	int m_t1;         //CONTIENE EL VALOR DEL TIMER

public:

	CNtgraphctrl1 m_Graph; //MUESTRA EN PANTALLA THETA2 V.S THETA3
	CNtgraphctrl1 m_Graph2;//MUESTRA EN PANTALLA THETA2 V.S THETA4
	CNtgraphctrl1 m_Graph3;//MUESTRA EN PANTALLA THETA2 V.S OMEGA3
	CNtgraphctrl1 m_Graph4;//MUESTRA EN PANTALLA THETA2 V.S OMEGA4
	CNtgraphctrl1 m_Graph5;//MUESTRA EN PANTALLA THETA2 V.S ALFA3
	CNtgraphctrl1 m_Graph6;//MUESTRA EN PANTALLA THETA2 V.S ALFA4


	FILE *out,*out2;  //SE ALMACENAN VALORES EN ARCHIVO DE TEXTO.

};
```

El código fuente de la implementación de la clase *Cpwmcon4codDlg* se omite en el presente trabajo debido a su gran longitud, pero se describe en el anexo A.6.

Clase USB

La clase USB contiene las funciones para la comunicación del microcontrolador PIC18F4550 con la PC.

```
class USB
{
public:
	//FUNCIONES QUE SE COMUNICAN CON LA DLL DE MICROCHIP
	void ClosePipes();  //Cierra las tuberias,si están abiertas
	void Initialice(); //Inicializa la conexión USB
	void OpenPipes(); //Abre las tuberias de comunicación

	//FUNCIONES PARA ENVIAR Y RECIBIR DATOS A TRAVÉS DEL USB
	void ReceivePacket(/*PVOID*/BYTE *ReceiveData,PDWORD ReceiveLength);
	//Recibe datos del PIC
	void SendPacket(BYTE *SendData, DWORD SendLength);//Envía datos al PIC
```

```
      //FUNCION QUE ACTUALIZA EL PWM.
      void prueba(unsigned int bajo, unsigned int alto);//Envía valor a PWM

      //FUNCIONES QUE INDICAN A QUE CONDICIÓN SE ENTRA.
      void modo1(void);//Actualiza PWM
      void modo2(void);//Entra al primer eje X
      void modo3(void);//Entra al primer eje Y
      void modo4(void);//Entra el segundo eje X
      void modo5(void);//Entra al segundo eje Y

      //FUNCIONES QUE ACTUALIZAN VALORES DE LOS CODIFICADORES
      long RESULTADO_R11(void);//RECIBIMOS VALOR DEL PRIMER CODIFICADOR
      long RESULTADO_R22(void);//RECIBIMOS VALOR DEL SEGUNDO CODIFICADOR
      long RESULTADO_R33(void);//RECIBIMOS VALOR DEL TERCER CODIFICADOR
      long RESULTADO_R44(void);//RECIBIMOS VALOR DEL CUARTO CODIFICADOR
};
```

El código fuente de la implementación de la clase *USB* se omite en el presente trabajo debido a su gran longitud y los derechos de propiedad intelectual.

Capítulo 3
Resultados y conclusiones

El principal resultado es la implementación en tiempo real de un prototipo electromecánico y la interfaz gráfica para el mecanismo de cuatro barras. Adicionalmente se adquirió experiencia en el microcontrolador PIC18F4550 así como en el manejo de herramientas de programación que pueden ser incluidos en cualquier aplicación a futuro.

3.1 Resultados

El prototipo desarrollado consta de tres componentes: mecánica, electrónica y la interfaz de operación. En la figura 3.1 se muestra el mecanismo de cuatro barras con los codificadores ópticos.

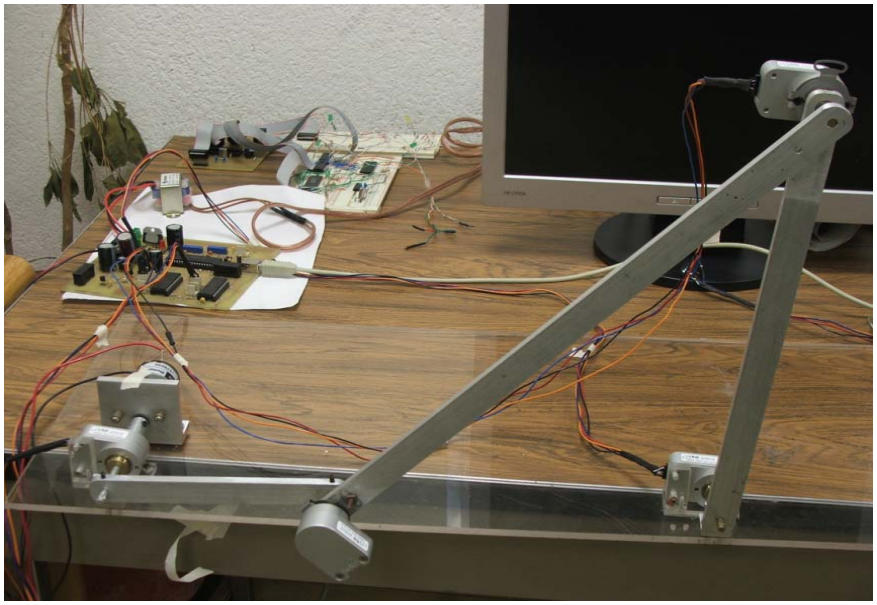


Figura 3.1. *Mecanismo de cuatro barras.*

En la figura 3.2 se muestra el sistema de control con operación en tiempo real.

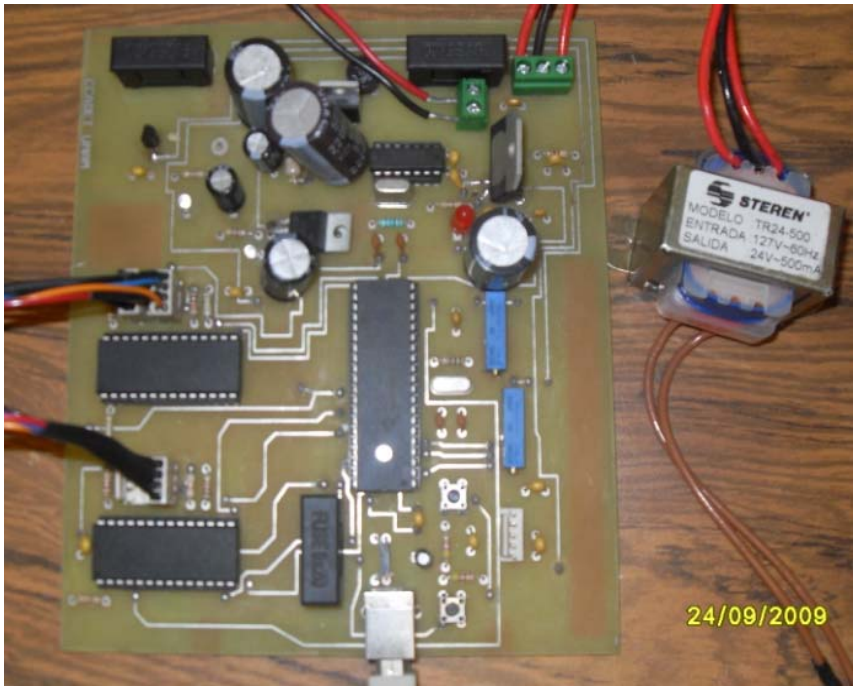


Figura 3.2. *Electrónica desarrollada.*

Los valores experimentales $\theta 2$, $\theta 4$ se obtienen de manera directa al hacer una recepción de las señales de los codificadores ópticos, cabe mencionar que la posición angular $\theta 3$ se determina con una relación geométrica $\theta 3=\pi-(\tau+\varphi)$, donde τ se obtiene directamente de un codificador óptico.

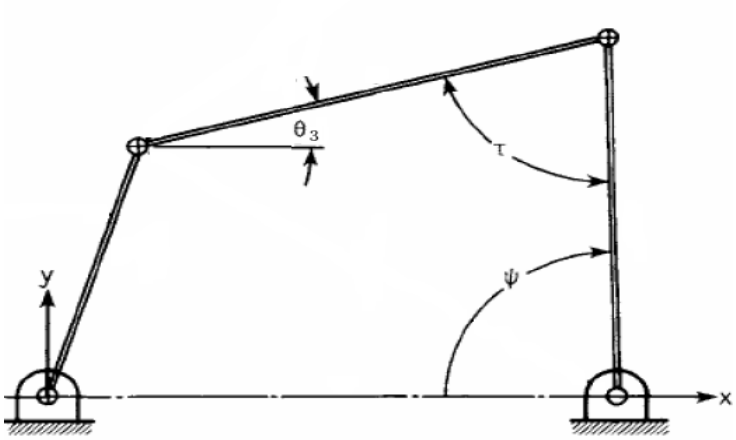


Figura 3.3. *Obtención de la posición angular* $\theta 3$ *.*

La velocidad y aceleración angular respectivamente, se obtuvieron a través de las ecuaciones de lazo vectorial (1.37, 1.38, 1.48 y 1.49) mencionadas en el capitulo 1. Estas expresiones dependen exclusivamente de las posiciones experimentales.

En la figura 3.4 se muestra la interfaz gráfica en operación con un dibujo mostrando los ángulos a medir.

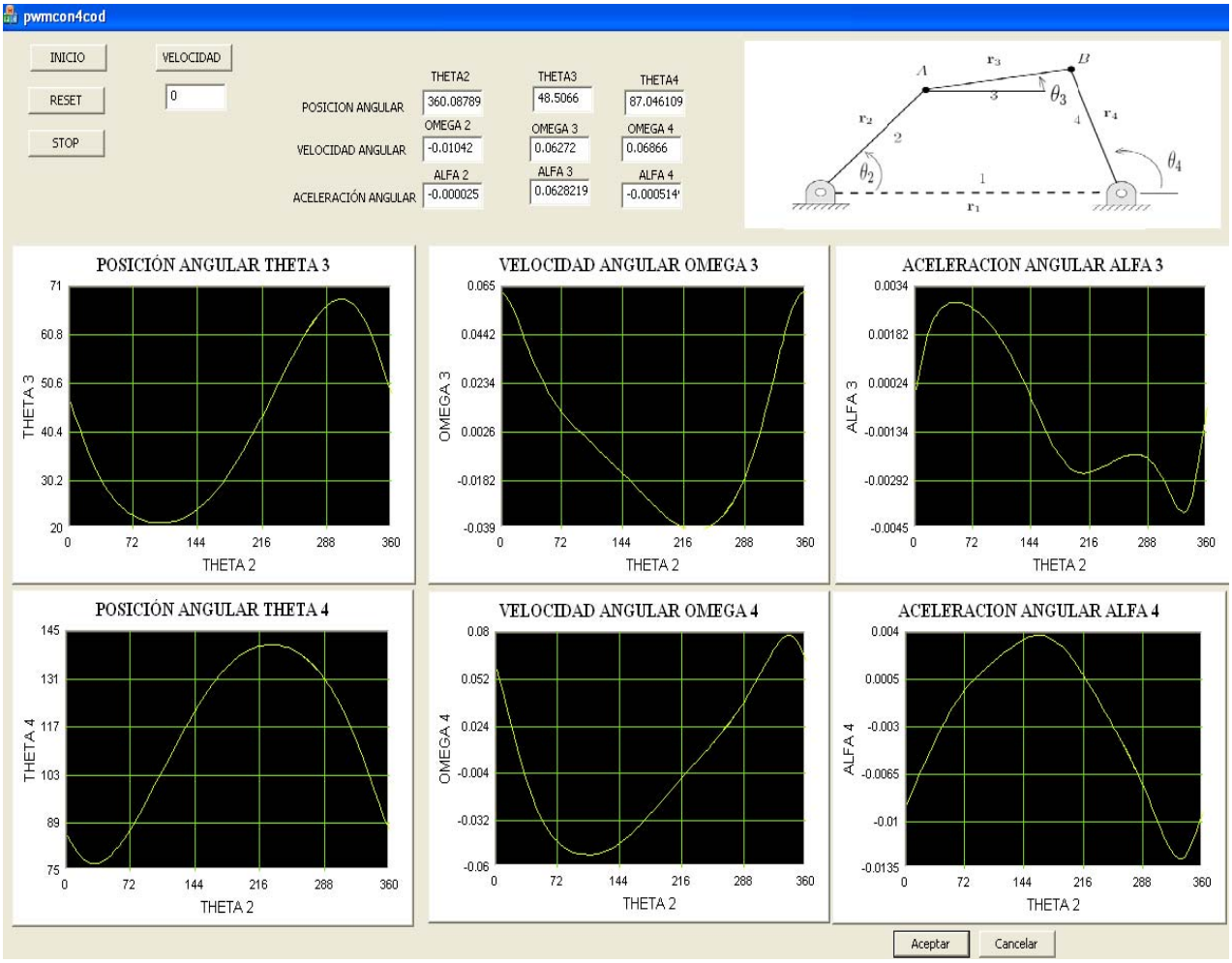


Figura 3.4. *Interfaz gráfica en operación.*

El desarrollo del *software* se convierte en un enlace entre los conceptos teóricos y prácticos del análisis de mecanismos, mostrando de manera virtual su comportamiento y permitiendo la visualización de los fenómenos de una manera versátil. El estudiante podrá comparar y analizar la información que proviene del sistema real contra la obtenida con el sistema teórico, con un ahorro de tiempo en comparación con los métodos tradicionales.

En la figura 3.5 se muestra la obtención de las curvas de posición angular entre datos experimentales contra datos teóricos, estos últimos obtenidos en un paquete llamado *"Mathematica"*

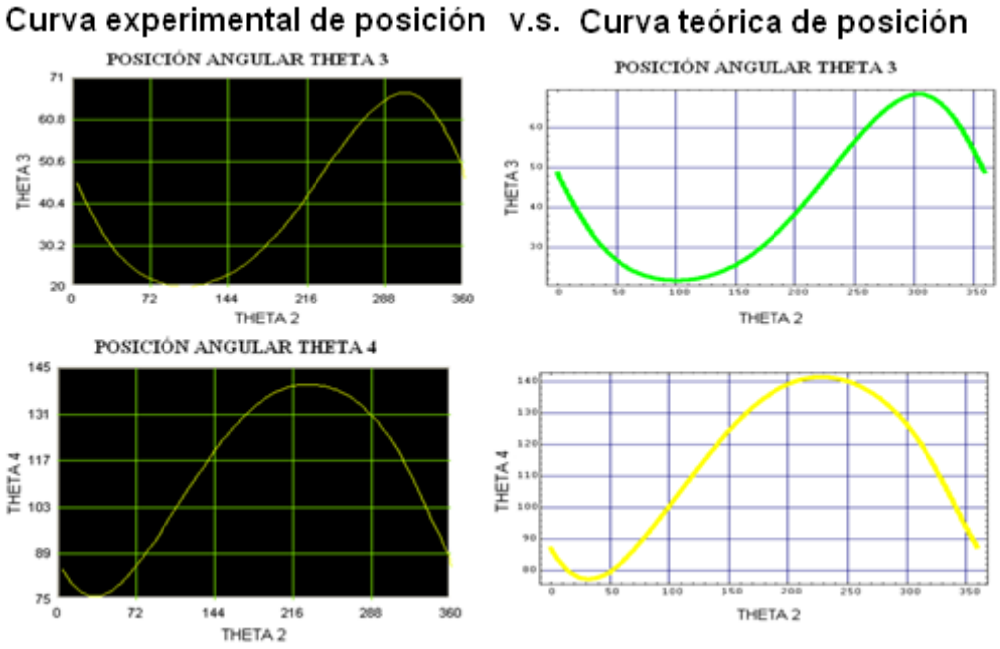


Figura 3.5. *Análisis de posición.*

En las figuras 3.6 y 3.7 se muestran la comparación de las curvas de posición angular entre los valores experimentales y teóricos respectivamente.

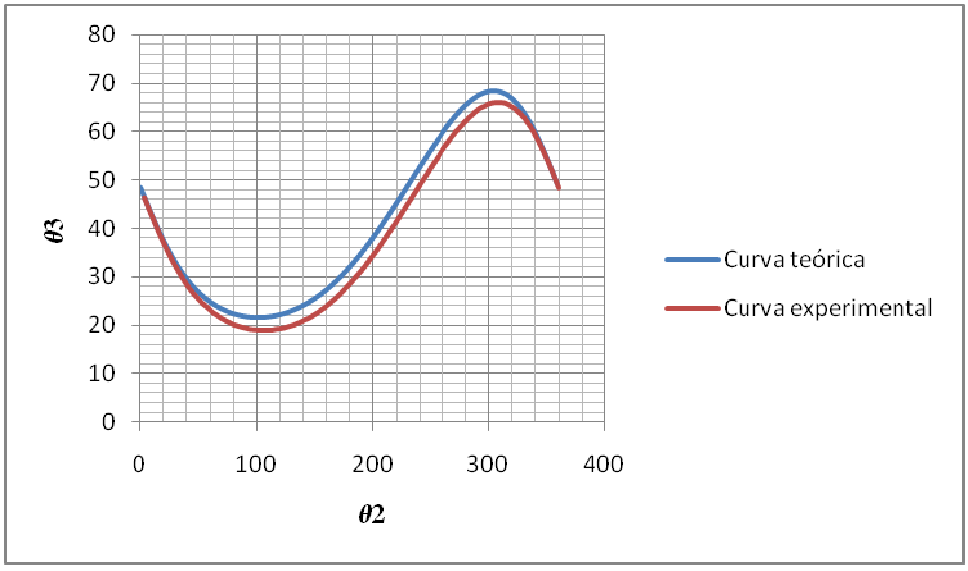


Figura 3.6. *Comparación* $\theta 3_{experimental}$ *.v.s.* $\theta 3_{teórica}$ *.*

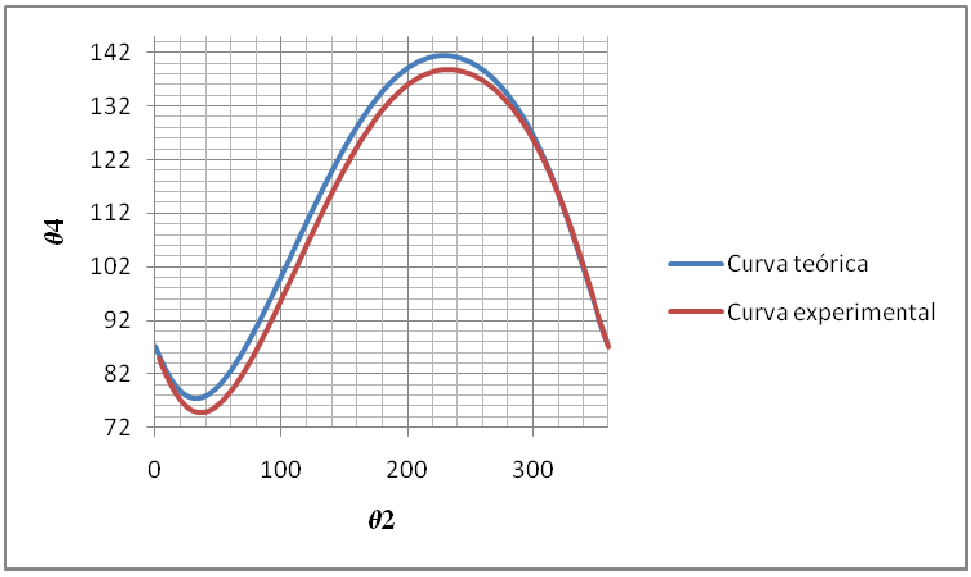


Figura 3.7. *Comparación $\theta 4_{experimental}$ v.s. $\theta 4_{teórica}$.*

En la figura 3.8 se muestra la obtención de las curvas de velocidad angular entre datos experimentales contra datos teóricos.

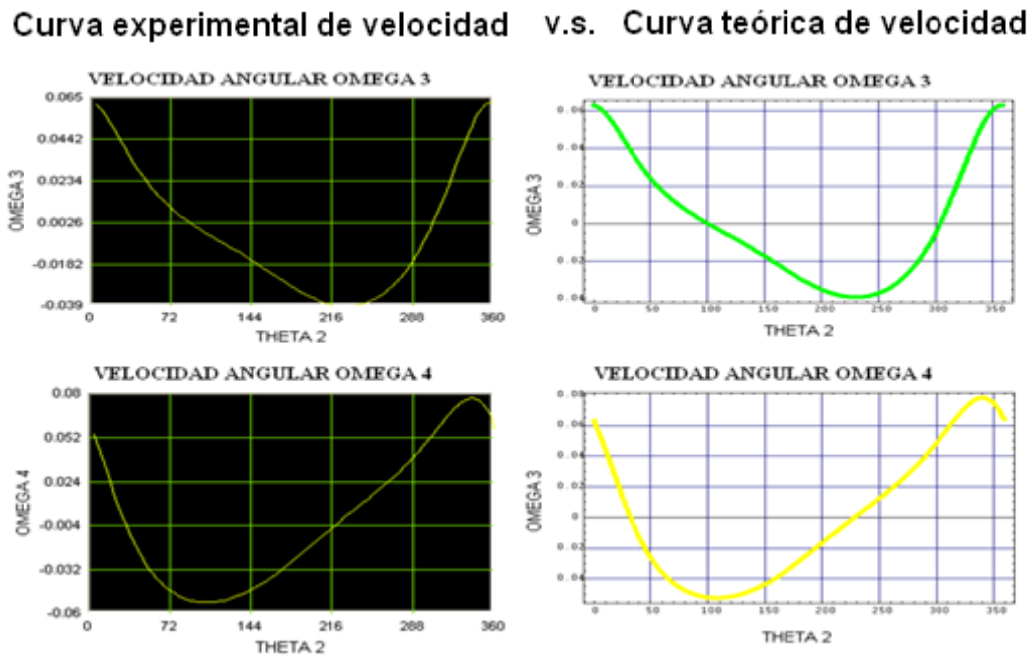


Figura 3.8. *Análisis de velocidad.*

En las figuras 3.9 y 3.10 se muestran la comparación de las curvas de velocidad angular entre los valores experimentales y teóricos respectivamente.

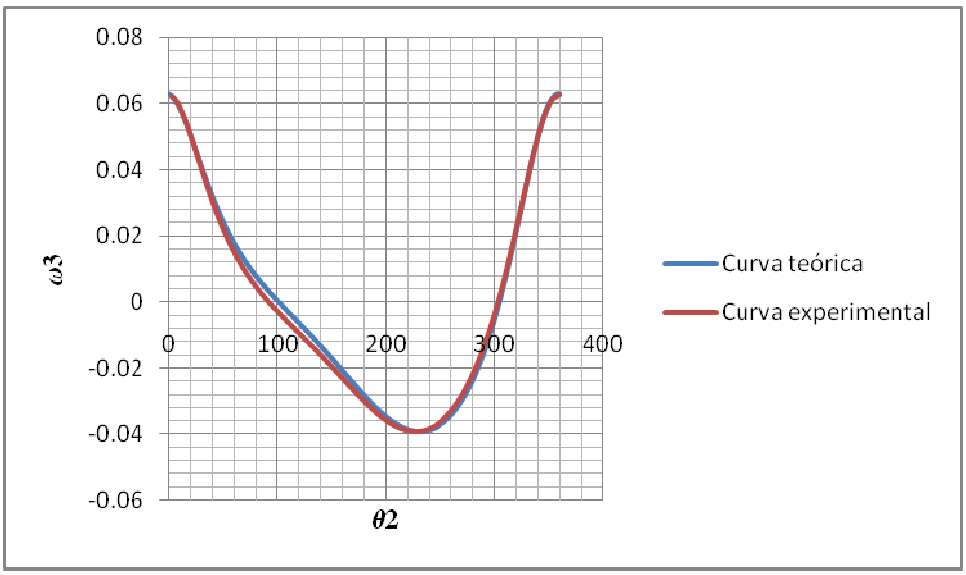


Figura 3.9. *Comparación* $\omega 3_{experimental}$ *.v.s.* $\omega 3_{teórica}$ *.*

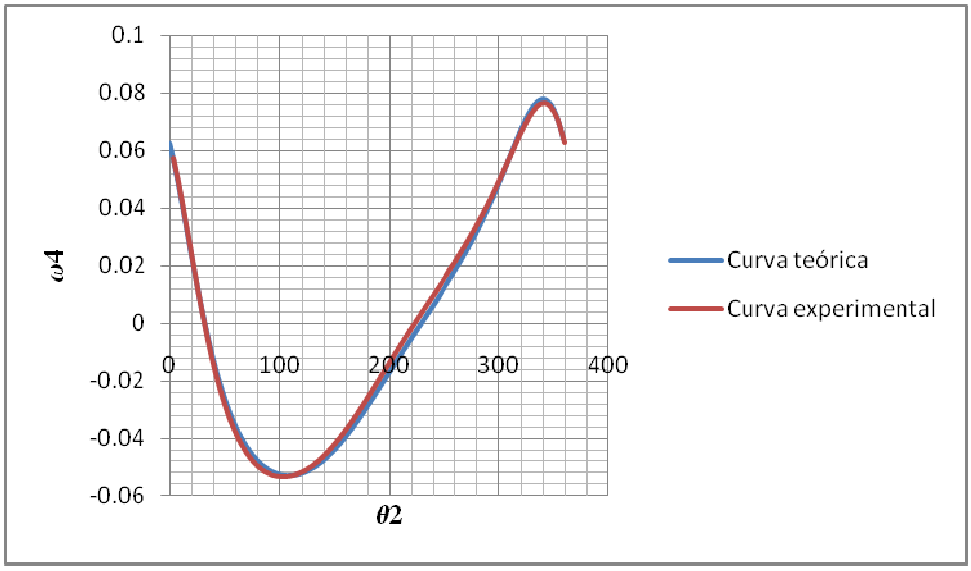


Figura 3.10. *Comparación* $\omega 4_{experimental}$ *.v.s.* $\omega 4_{teórica}$ *.*

En la figura 3.11 se muestra la obtención de las curvas de aceleración angular entre datos experimentales contra datos teóricos.

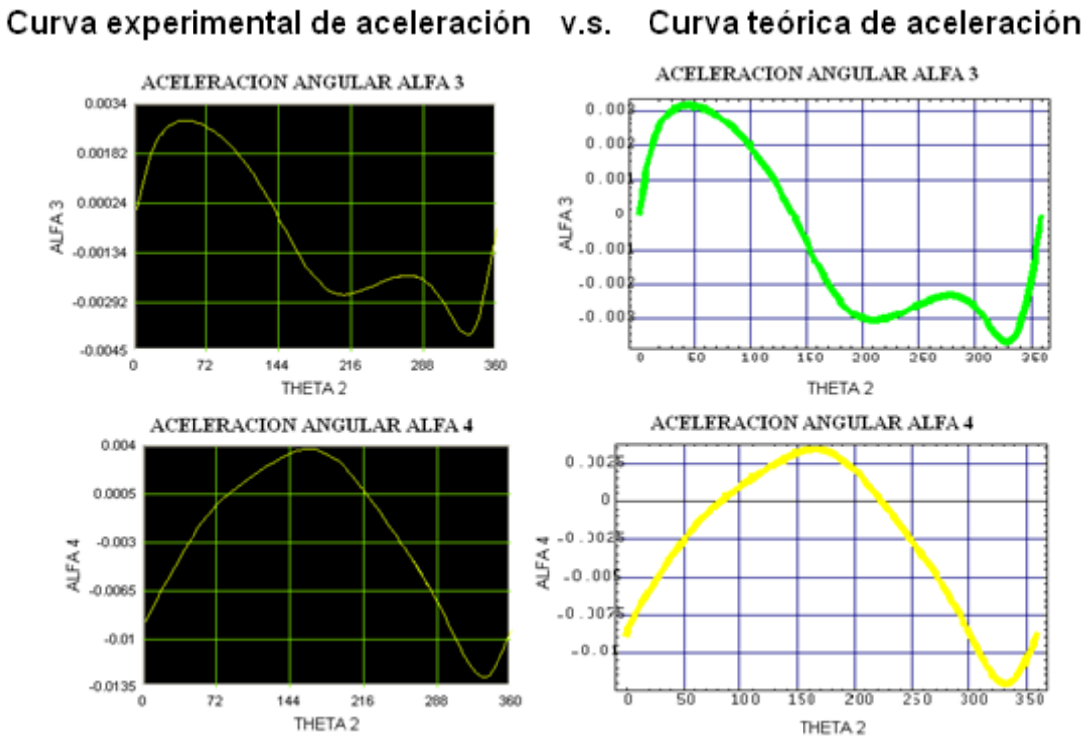


Figura 3.11. *Análisis de aceleración.*

En las figuras 3.12 y 3.13 se muestran la comparación de las curvas de velocidad angular entre los valores experimentales y teóricos respectivamente.

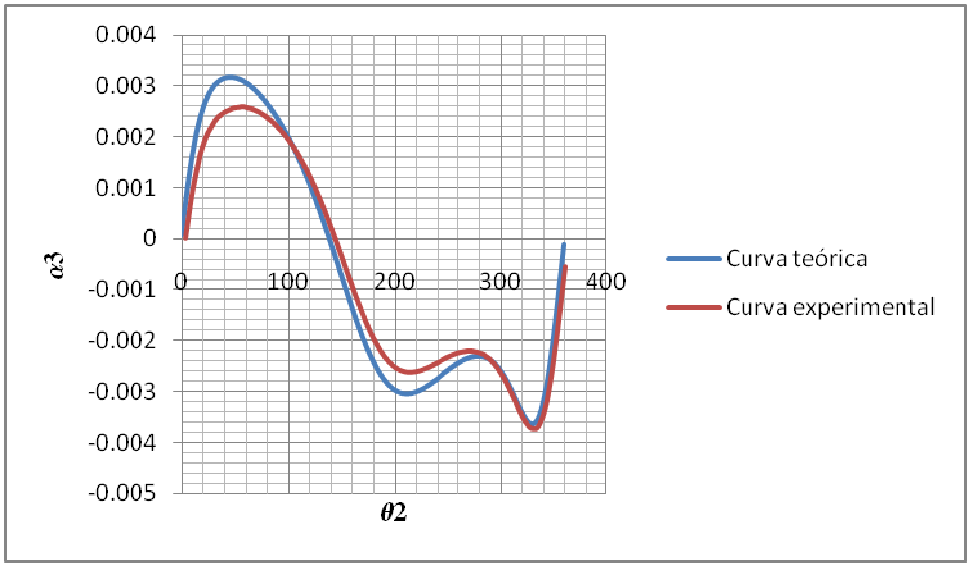


Figura 3.12. *Comparación* $\alpha 3_{experimental}$ *.v.s.* $\alpha 3_{teórica}$ *.*

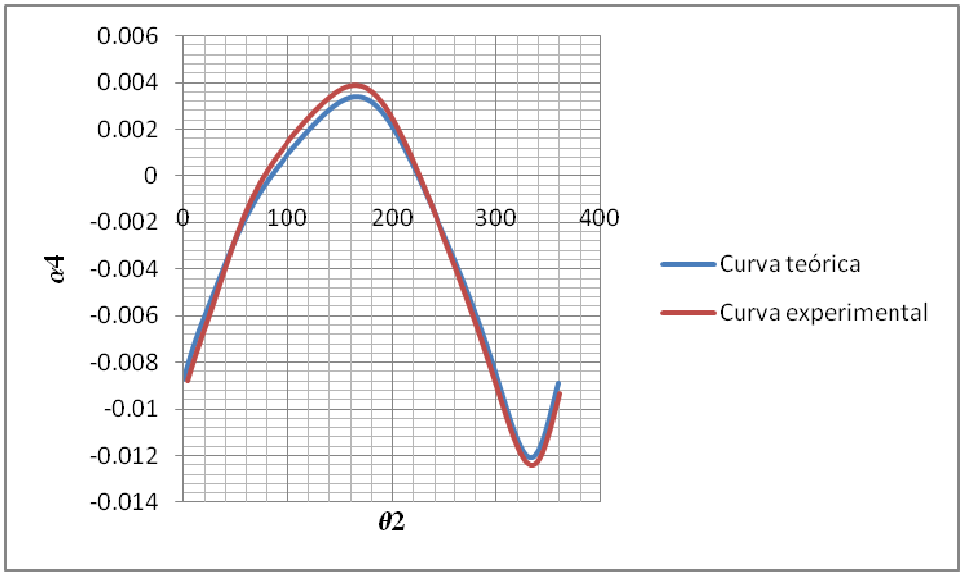


Figura 3.13. *Comparación* $\alpha 4_{experimental}$ *.v.s.* $\alpha 4_{teórica}$ *.*

3.2 Conclusiones

La creación de una interfaz grafica, que permite visualizar la cinemática de mecanismos, complementa el proceso de aprendizaje y sirve de base para la validación de modelos teóricos y un avance en el estado del arte de la simulación de sistemas dinámicos, en el manejo de *software* basado en objetos gráficos y en el manejo dinámico de información con controles ActiveX.

La teoría cinemática de mecanismos planos se ha desarrollado desde hace muchas décadas atrás. Sin embargo, en términos prácticos, ésta no tenía mucha aplicación ya que la solución de las ecuaciones del análisis cinemático, es compleja con los métodos tradicionales. Con la llegada de herramientas computacionales, la solución de ecuaciones de complejidad deja de ser un obstáculo y permite que mediante métodos numéricos se llegue a soluciones adecuadas de manera efectiva.

En las gráficas obtenidas el comportamiento del error lleva a considerar una única causa probable, la deformación elástica transitoria en los elementos flexibles de la transmisión de los codificadores. Dado que la medición se hizo en condiciones dinámicas y no estáticas, se originan esfuerzos variables y de diferente magnitud instantánea para ambas barras, que a su vez producen deformaciones angulares transitorias en las transmisiones a los codificadores ópticos. Para la medición de los ángulos se necesitaron tres codificadores ópticos incrementales con una resolución bastante buena, podemos concluir que con dos codificadores ópticos era suficiente para obtener los tres valores experimentales. Además, la velocidad y aceleración angular de cada barra se obtuvieron de la información de los codificadores ópticos, con ayuda de las ecuaciones cinemáticas descritas anteriormente.

3.3 Trabajo a futuro

Se pretende hacer una unidad didáctica para el análisis cinemático y dinámico de mecanismos. Adicionalmente, se desarrollará una unidad motriz en general para tres mecanismos a desarrollar que se conectarán al eslabón impulsor (eslabón de entrada). Dicha unidad motriz constará básicamente del motor de CD impulsor, un dinamómetro de la marca *Lorenz Messtechnik*. Dichos elementos se conectarán a una computadora personal que permitirá controlar y registrar la velocidad del motor, el par transmitido al eslabón de entrada y la potencia consumida. La información de los sensores se registrará y almacenará en la computadora y mediante las ecuaciones dinámicas se tendrá acceso a los parámetros requeridos para el análisis. Cabe mencionar, que se complementará el desarrollo de la interfaz gráfica.

Bibliografía

[1]. Mecanismo de 4 barras, Francisco T. Sánchez Marín, Departamento de Ingeniería Mecánica y Construcción, Universitat Jaume, http://www.emc.uji.es/d/IngMecDoc/Mecanismos/index.html

[2]. Modelo de mecanismo de cuatro barras, G.U.N.T Hamburg, Equipos para la educación en ingeniería, http://www.gunt.de:80/static/s3604_3.php?p1=&p2=&pN

[3]. Baldor, Aurelio. ALGEBRA. Publicaciones CULTURAL. 2002, pp. 346

[4]. US Digital, “HB5M Hollow Bore Optical Encoder”, US Digital, 2008, 6pp.

[5]. US Digital “LS7266R1 Encoder to Microprocessor Interface Chip”, US Digital, 2006, 6pp.

[6]. García,Eduardo. *Compilador C CCS y simulador PROTEUS para Microcontroladores PIC*. Alfaomega Grupo Editor. 2008

[7]. Microchip, “PIC18F4550 Datasheet”, Microchip Technology, 2007, pp. 2-95

[8]. Microchip, “PICDEM™ FS USB DEMONSTRATION BOARD USER’S GUIDE”, Microchip Technology, 2008.

[9]. Nikolai Teofilov “2D Graph ActiveXControl", http://www.codeproject.com/KB/miscctrl/ntgraph_activex.aspx/, 2003

[10]. Muchotrasto, http://www.muchotrasto.com/

[11]. Rashid, Muhammad H. ELECTRONICA DE POTENCIA: CIRCUITOS, DISPOSITIVOS Y APLICACIONES. Prentice Hall. 2004.

[12]. Ceballos, Francisco Javier. Visual C++. Ra-Ma, Librería y Editorial Microinformática. 2001

Anexos

A.1 Código del programa en *Mathematica*

Se muestra la simulación del mecanismo de cuatro barras, utilizando el código de *Mathematica* descrito a continuación, puede encontrarse el comportamiento en posición, velocidad y aceleración. Se deja al lector la tarea de reproducir estos resultados utilizando cualquier herramienta de su elección. Todas las ecuaciones utilizadas para el análisis cinemático se encuentran descritas en el capítulo 1.

MECANISMO DE CUATRO BARRAS

Funciones

```
R[θ_] := {{Cos[θ], -Sin[θ]}, {Sin[θ], Cos[θ]}};
Ω[ω_] := {{0, -ω}, {ω, 0}};
Alf[ω_, α_] := {{-ω^2, α}, {α, -ω^2}}
```

Ecuaciones Cinemáticas

Posición

```
r1 = {cx1, 0};
r2 = {-cy2, 0};
r3 = {-cx3, 0};
r4 = {cy4, 0};


R1 = r1;
R2 = R[θ2].r2;
R3 = R[θ3].r3;
R4 = R[θ4].r4;


LVP1 = R1 + R4 + R3 + R2;
```

Velocidad

```
V2 = Ω[ω2].R2;
V3 = Ω[ω3].R3;
V4 = Ω[ω4].R4;

LVV1 = V4 + V3 + V2;
```

Aceleración

```
A2 = Alf[ω2, α2].R2;
A3 = Alf[ω3, α3].R3;
A4 = Alf[ω4, α4].R4;

LVA1 = A4 + A3 + A2;
```

Datos

```
cx1 = 40;
cy2 = 15;
cx3 = 40;
cy4 = 30;
```

Solución Cinemática

Solución posición inicial

```
Clear[θ3, θ4];
θ2 = 60 * Degree;

SolIni = FindRoot[
   {LVP1[[1]] == 0,
    LVP1[[2]] == 0},
   {θ3, 40 * Degree},
   {θ4, 120 * Degree},
   MaxIterations → 15];
?? SolIni
```

```
Global`SolIni

SolIni = {θ3 → 0.432504, θ4 → 1.44323}
```

```
Clear[θ3, θ4];
θ2 = 60 * Degree;
θ3i = (θ3 /. SolIni);
θ4i = (θ4 /. SolIni);
For[i = 0, i ≤ 360, i += 1,
 θ2 = i * Degree;
 SolPos[i] = FindRoot[
   {LVP1[[1]] == 0,
    LVP1[[2]] == 0},
   {θ3, θ3i},
   {θ4, θ4i},
   MaxIterations → 15];
 θ3i = (θ3 /. SolPos[i]);
 θ4i = (θ4 /. SolPos[i]);]

?? SolPos
```

Solución Velocidad

```
Clear[ω3, ω4];

For[i = 0, i ≤ 360, i += 1,
  ω2 = -(0.033333333 * Pi);
  θ2 = i * Degree;
  SolVel[i] = Solve[{
       LVV1[[1]] == 0,
       LVV1[[2]] == 0} /. SolPos[i],
     {ω3, ω4}] // Flatten];

?? SolVel
```

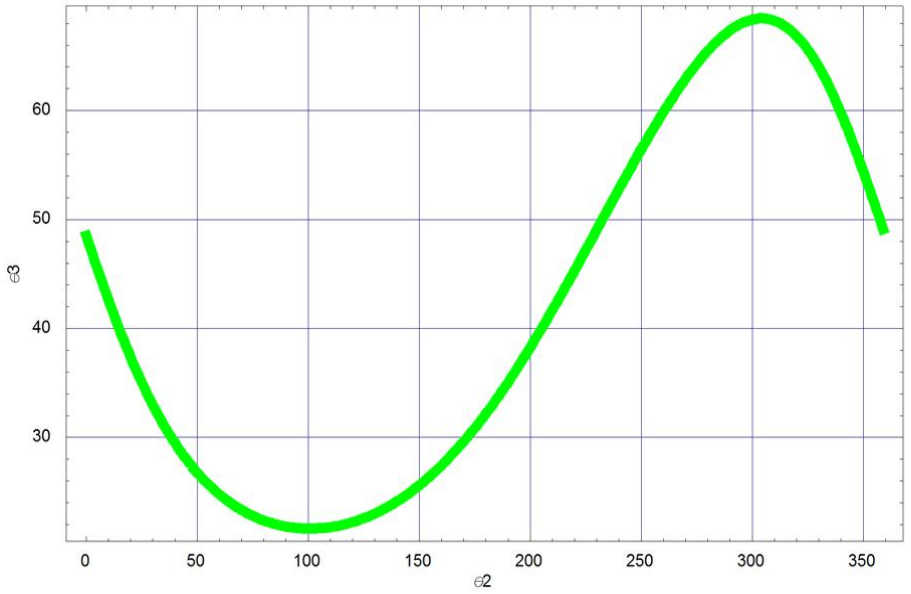
Solución Aceleración

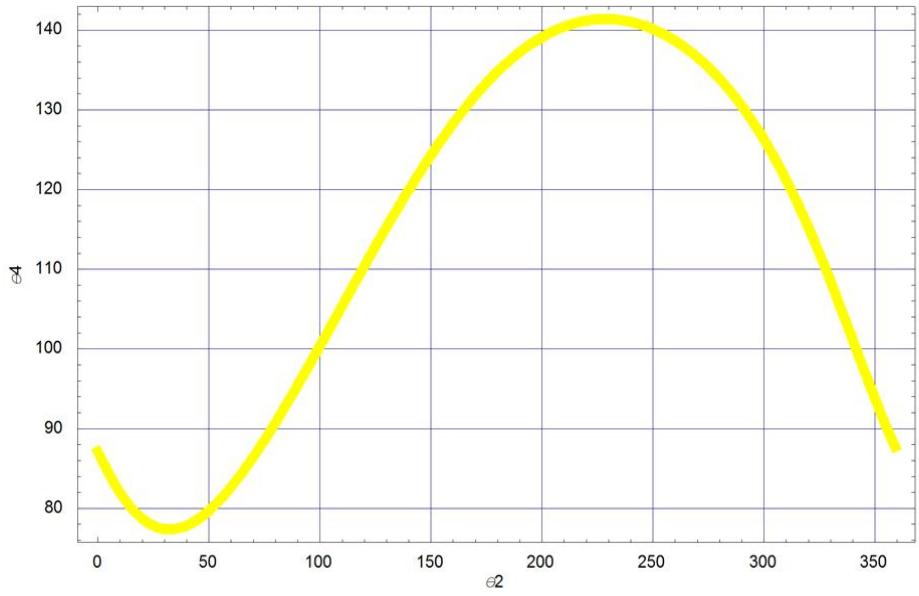
```
Clear[α3, α4];

For[i = 0, i ≤ 359, i += 1,
  α2 = -0.001;
  θ2 = i * Degree;
  SolAcel[i] = Solve[{
        LVA1[[1]] == 0,
        LVA1[[2]] == 0} /. SolPos[i] /. SolVel[i],
     {α3, α4}] // Flatten];

?? SolAcel
```

Gráficas

Gráficas Posición

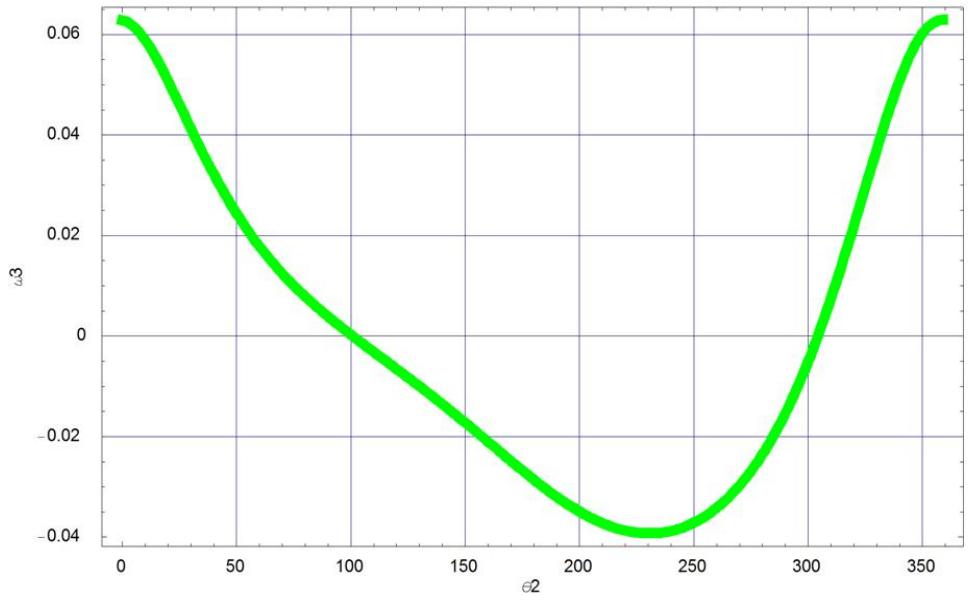
```
Teta3 = Table[{i, ((θ3 /. SolPos[i]) / Degree)}, {i, 0, 359, 1}];
Teta4 = Table[{i, (θ4 /. SolPos[i]) / Degree}, {i, 0, 359, 1}];

Export["out.dat", Teta3];
Tetapro1 = Table[{(θ3 /. SolPos[i]) / Degree, (θ4 /. SolPos[i]) / Degree}, {i, 0, 359, 1}];
Tetapro2 = Table[{(θ4 /. SolPos[i]) / Degree, (θ3 /. SolPos[i]) / Degree}, {i, 0, 359, 1}];

Figura1 = ListPlot[Teta3,
   PlotJoined → True,
   Frame → True,
   FrameLabel → {"θ2", "θ3"},
   DefaultFont → {"courier", 10},
   GridLines → Automatic,
   PlotStyle → AbsoluteThickness[6],
   Prolog → RGBColor[0, 1, 0],
   Background → RGBColor[1, 1, 1]];

Figura2 = ListPlot[Teta4,
   PlotJoined → True,
   Frame → True,
   FrameLabel → {"θ2", "θ4"},
   DefaultFont → {"courier", 10},
   GridLines → Automatic,
   PlotStyle → AbsoluteThickness[6],
   Prolog → RGBColor[1, 1, 0],
   Background → RGBColor[1, 1, 1]];
```

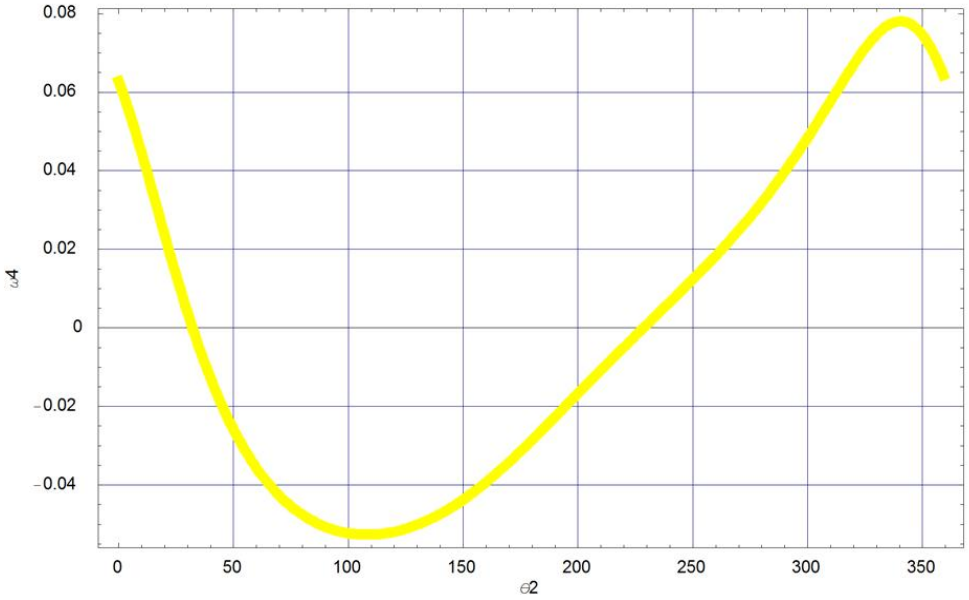

60
50
40
30
θ3
0
50
100
150
200
250
300
350
θ2


140
130
120
110
100
90
80
θ4
0
50
100
150
200
250
300
350
θ2

Gráficas Velocidad

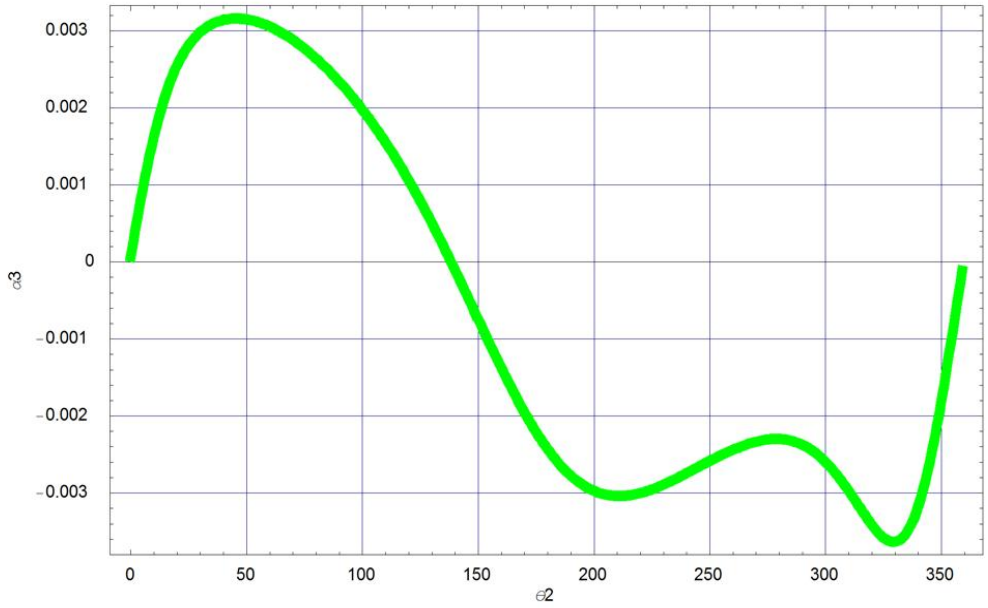
```
Omega3 = Table[{i, (ω3 /. SolVel[i])}, {i, 0, 359, 1}];
Omega33 = Table[{(ω3 /. SolVel[i])}, {i, 0, 360, 1}];
Omega4 = Table[{i, (ω4 /. SolVel[i])}, {i, 0, 359, 1}];

Export["out2.dat", Omega33];
Figura5 = ListPlot[Omega3,
   PlotJoined → True,
   Frame → True,
   FrameLabel → {"θ2", "ω3"},
   DefaultFont → {"courier", 10},
   GridLines → Automatic,
   PlotStyle → AbsoluteThickness[6],
   Prolog → RGBColor[0, 1, 0],
   Background → RGBColor[1, 1, 1]];
Figura6 = ListPlot[Omega4,
   PlotJoined → True,
   Frame → True,
   FrameLabel → {"θ2", "ω4"},
   DefaultFont → {"courier", 10},
   GridLines → Automatic,
   PlotStyle → AbsoluteThickness[6],
   Prolog → RGBColor[1, 1, 0],
   Background → RGBColor[1, 1, 1]];
```

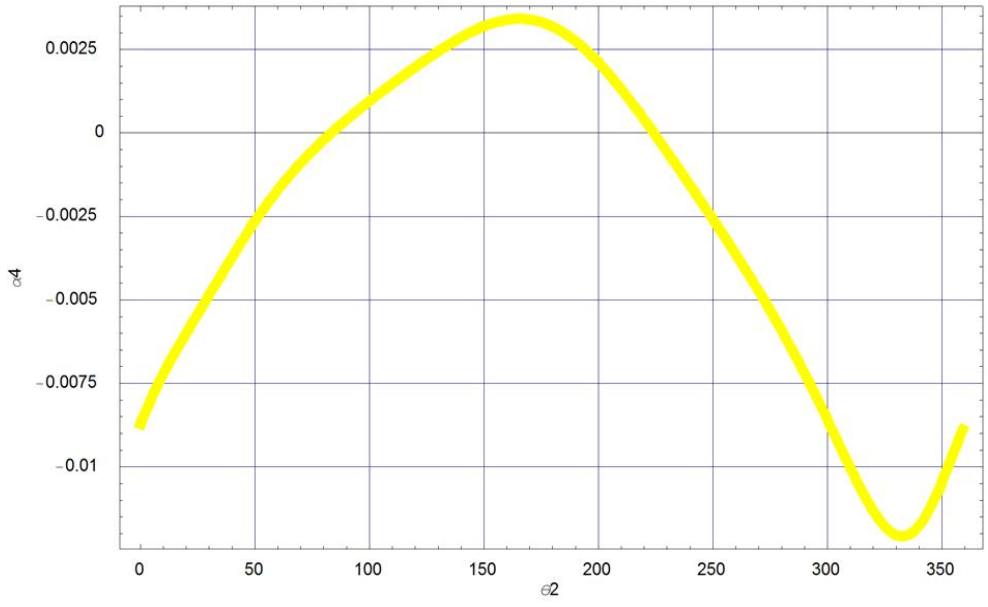




Gráficas Aceleración

```
Alfa3 = Table[{i, (α3 /. SolAcel[i])}, {i, 0, 359, 1}];
Alfa4 = Table[{i, (α4 /. SolAcel[i])}, {i, 0, 359, 1}];

Figura9 = ListPlot[Alfa3,
   PlotJoined → True,
   Frame → True,
   FrameLabel → {"θ2", "α3"},
   DefaultFont → {"courier", 10},
   GridLines → Automatic,
   PlotStyle → AbsoluteThickness[6],
   Prolog → RGBColor[0, 1, 0],
   Background → RGBColor[1, 1, 1]];
Figura10 = ListPlot[Alfa4,
   PlotJoined → True,
   Frame → True,
   FrameLabel → {"θ2", "α4"},
   DefaultFont → {"courier", 10},
   GridLines → Automatic,
   PlotStyle → AbsoluteThickness[6],
   Prolog → RGBColor[1, 1, 0],
   Background → RGBColor[1, 1, 1]];
```

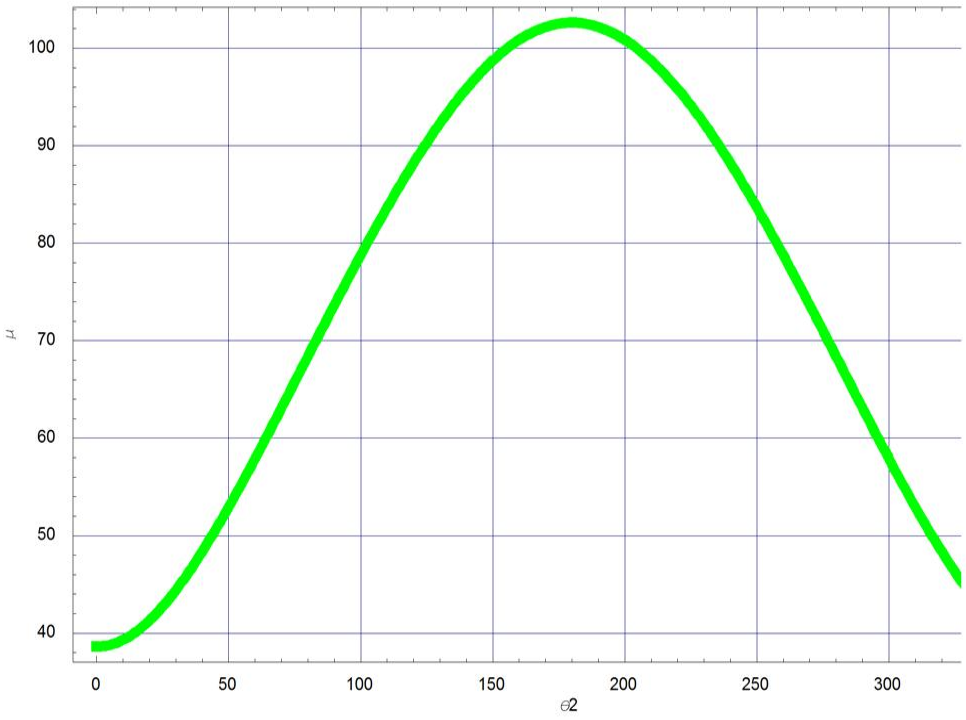

0.003
0.002
0.001
0
-0.001
-0.002
-0.003
θ3
0
50
100
150
200
250
300
350
θ2


0.0025
0
-0.0025
-0.005
-0.0075
-0.01
θ4
0
50
100
150
200
250
300
350
θ2

Grafica ángulo de transmisión

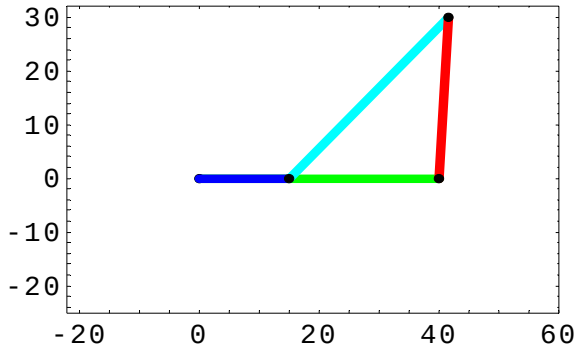
```
mu = Table[{i, (θ4 /. SolPos[i]) / Degree - (((θ3 /. SolPos[i]) / Degree))}, {i, 0, 359, 1}];
```

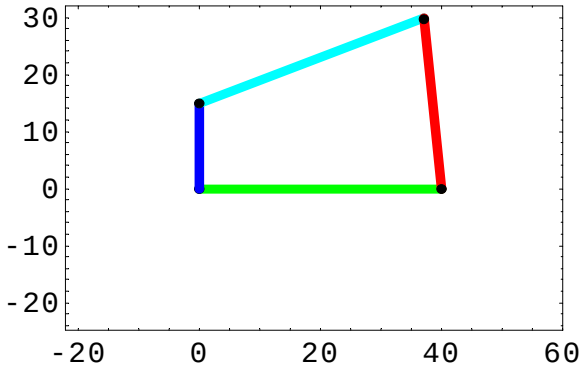
```
Figura1 = ListPlot[mu,
   PlotJoined → True,
   Frame → True,
   FrameLabel → {"θ2", "μ"},
   DefaultFont → {"courier", 10},
   GridLines → Automatic,
   PlotStyle → AbsoluteThickness[6],
   Prolog → RGBColor[0, 1, 0],
   Background → RGBColor[1, 1, 1]];
```

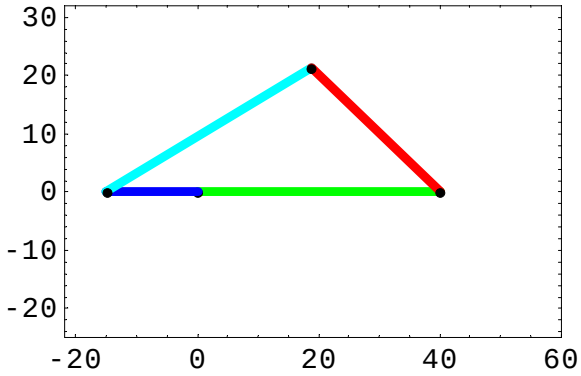


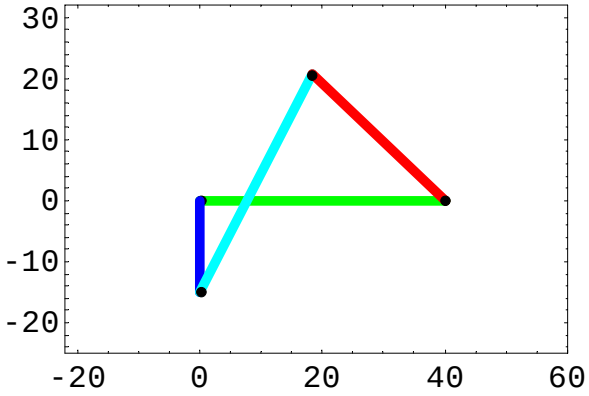
Simulación

```
For[i = 0, i ≤ 360, i += 10,
  θ2 = i * Degree;
  cero = {0, 0};
  linea1 = Line[{cero, R1}];
  linea4 = Line[{R1, R1 + R4}] /. SolPos[i];
  linea3 = Line[{R1 + R4, R1 + R4 + R3}] /. SolPos[i];
  linea2 = Line[{R1 + R4 + R3, R1 + R4 + R3 + R2}] /. SolPos[i];


  punto1 = Point[cero];
  punto2 = Point[R1];
  punto3 = Point[R1 + R4] /. SolPos[i];
  punto4 = Point[R1 + R4 + R3] /. SolPos[i];


  barra1 = Graphics[{AbsoluteThickness[6], RGBColor[0, 1, 0], linea1}];
  barra2 = Graphics[{AbsoluteThickness[6], RGBColor[0, 0, 1], linea2}];
  barra3 = Graphics[{AbsoluteThickness[6], RGBColor[0, 1, 1], linea3}];
  barra4 = Graphics[{AbsoluteThickness[6], RGBColor[1, 0, 0], linea4}];


  perno1 = Graphics[{PointSize[0.02], punto1}];
  perno2 = Graphics[{PointSize[0.02], punto2}];
  perno3 = Graphics[{PointSize[0.02], punto3}];
  perno4 = Graphics[{PointSize[0.02], punto4}];

  Show[barra1, perno1, barra2, barra3, barra4, perno2, perno3, perno4,
   Frame → True,
   DefaultFont → {"Courier", 20},
   AspectRatio → Automatic,
   PlotRange → {{-22, 60}, {-25, 32}}];];
```


30
20
10
0
-10
-20
-20
0
20
40
60


30
20
10
0
-10
-20
-20
0
20
40
60


30
20
10
0
-10
-20
-20
0
20
40
60


30
20
10
0
-10
-20
-20
0
20
40
60

A.2 Descriptores

Este ejemplo muestra como configurar el dispositivo USB y sus descriptores. Los únicos cambios con respecto a su versión original (usb_desc_scope.h) han sido realizados en el apartado start device descriptors, concretamente el vendor y product id y en apartado de start string descriptors, para definir el nombre del dispositivo y la compañia.

```
////////////////////////////////////////////////////////////////////////
////                        usb_desc_scope.h                        ////
////                                                                ////
//// An example set of device / configuration descriptors for use with ////
//// the USB Bulk demo (see ex_usb_scope.c)                         ////
////                                                                ////
////////////////////////////////////////////////////////////////////////
////                                                                ////
//// Version History:                                               ////
////                                                                ////
//// July 13th, 2005:                                               ////
////   Endpoint descriptor works if USB_EP1_TX_SIZE is 16bits       ////
////   Endpoint descriptor works if USB_EP1_RX_SIZE is 16bits       ////
////                                                                ////
//// June 20th, 2005:                                               ////
////   18fxx5x Initial release.                                     ////
////                                                                ////
//// March 21st, 2005:                                              ////
////   EP 0x01 and EP 0x81 now use USB_EP1_TX_SIZE and USB_EP1_RX_SIZE ////
////      to define max packet size, to make it easier for dynamically ////
////      changed code.                                             ////
////                                                                ////
//// June 24th, 2002: Cleanup                                       ////
////                                                                ////
//// May 6th, 2003: Fixed non-HID descriptors pointing to faulty    ////
////                strings                                         ////
////                                                                ////
//// August 2nd, 2002: Initial Public Release                       ////
////                                                                ////
////                                                                ////
////////////////////////////////////////////////////////////////////////
///        (C) Copyright 1996,2005 Custom Computer Services        ////
//// This source code may only be used by licensed users of the CCS ////
//// C compiler.  This source code may only be distributed to other ////
//// licensed users of the CCS C compiler.  No other use,           ////
//// reproduction or distribution is permitted without written      ////
//// permission.  Derivative programs created using this software   ////
//// in object code form are not restricted in any way.             ////
////////////////////////////////////////////////////////////////////////

#IFNDEF __USB_DESCRIPTORS__
#DEFINE __USB_DESCRIPTORS__
```

```
#include <usb.h>

//////////////////////////////////////////////////////////////////
///
///   start config descriptor
///   right now we only support one configuration descriptor.
///   the config, interface, class, and endpoint goes into this array.
///
//////////////////////////////////////////////////////////////////

   #DEFINE USB_TOTAL_CONFIG_LEN      32 //config+interface+class+endpoint

   //configuration descriptor
   char const USB_CONFIG_DESC[] = {
   //config_descriptor for config index 1
         USB_DESC_CONFIG_LEN,     //length of descriptor size
         USB_DESC_CONFIG_TYPE,         //constant CONFIGURATION (0x02)
         USB_TOTAL_CONFIG_LEN,0,  //size of all data returned for this
config
         1,      //number of interfaces this device supports
         0x01,                //identifier for this configuration.  (IF we
had more than one configurations)
         0x00,                //index of string descriptor for this
configuration
         0xC0,                //bit 6=1 if self powered, bit 5=1 if supports
remote wakeup (we don't), bits 0-4 reserved and bit7=1
         0x32,                //maximum bus power required (maximum
milliamperes/2)  (0x32 = 100mA)

   //interface descriptor 0 alt 0
         USB_DESC_INTERFACE_LEN,  //length of descriptor
         USB_DESC_INTERFACE_TYPE,      //constant INTERFACE (0x04)
         0x00,                //number defining this interface (IF we had
more than one interface)
         0x00,                //alternate setting
         2,       //number of endpoints, not counting endpoint 0.
         0xFF,                //class code, FF = vendor defined
         0xFF,                //subclass code, FF = vendor
         0xFF,                //protocol code, FF = vendor
         0x00,                //index of string descriptor for interface

   //endpoint descriptor
         USB_DESC_ENDPOINT_LEN, //length of descriptor
         USB_DESC_ENDPOINT_TYPE,     //constant ENDPOINT (0x05)
         0x81,              //endpoint number and direction (0x81 = EP1 IN)
         0x02,              //transfer type supported (0 is control, 1 is
iso, 2 is bulk, 3 is interrupt)
         USB_EP1_TX_SIZE & 0xFF,USB_EP1_TX_SIZE >> 8,         //maximum
packet size supported
         0x01,              //polling interval in ms. (for interrupt
transfers ONLY)

   //endpoint descriptor
         USB_DESC_ENDPOINT_LEN, //length of descriptor
         USB_DESC_ENDPOINT_TYPE,     //constant ENDPOINT (0x05)
         0x01,              //endpoint number and direction (0x01 = EP1 OUT)
```

```
         0x02,              //transfer type supported (0 is control, 1 is
iso, 2 is bulk, 3 is interrupt)
         USB_EP1_RX_SIZE & 0xFF,USB_EP1_RX_SIZE >> 8,         //maximum
packet size supported
         0x01,              //polling interval in ms. (for interrupt
transfers ONLY)

  };

   //****** BEGIN CONFIG DESCRIPTOR LOOKUP TABLES ********
   //since we can't make pointers to constants in certain pic16s, this is an
offset table to find
   //  a specific descriptor in the above table.

   //NOTE: DO TO A LIMITATION OF THE CCS CODE, ALL HID INTERFACES MUST START
AT 0 AND BE SEQUENTIAL
   //      FOR EXAMPLE, IF YOU HAVE 2 HID INTERFACES THEY MUST BE INTERFACE
0 AND INTERFACE 1
   #define USB_NUM_HID_INTERFACES   0

   //the maximum number of interfaces seen on any config
   //for example, if config 1 has 1 interface and config 2 has 2 interfaces
you must define this as 2
   #define USB_MAX_NUM_INTERFACES   1

   //define how many interfaces there are per config.  [0] is the first
config, etc.
   const char USB_NUM_INTERFACES[USB_NUM_CONFIGURATIONS]={1};

   #if (sizeof(USB_CONFIG_DESC) != USB_TOTAL_CONFIG_LEN)
      #error USB_TOTAL_CONFIG_LEN not defined correctly
   #endif


////////////////////////////////////////////////////////////////
///
///   start device descriptors
///
////////////////////////////////////////////////////////////////

   //device descriptor
   char const USB_DEVICE_DESC[] ={
         USB_DESC_DEVICE_LEN,          //the length of this report
         0x01,                 //constant DEVICE (0x01)
         0x10,0x01,            //usb version in bcd
         0x00,                 //class code (if 0, interface defines class.
FF is vendor defined)
         0x00,                 //subclass code
         0x00,                 //protocol code
         USB_MAX_EP0_PACKET_LENGTH,   //max packet size for endpoint 0.
(SLOW SPEED SPECIFIES 8)
         0xD8,0x04,            //vendor id (0x04D8 is Microchip)
         0x11,0x00,            //product id
         0x01,0x00,            //device release number
         0x01,                 //index of string description of manufacturer.
therefore we point to string_1 array (see below)
         0x02,                 //index of string descriptor of the product
```

```
        0x00,                //index of string descriptor of serial number
        USB_NUM_CONFIGURATIONS   //number of possible configurations
   };

//////////////////////////////////////////////////////////////////
///
///   start string descriptors
///   String 0 is a special language string, and must be defined.  People in
U.S.A. can leave this alone.
///
///   You must define the length else get_next_string_character() will not
see the string
///   Current code only supports 10 strings (0 thru 9)
///
//////////////////////////////////////////////////////////////////

//the offset of the starting location of each string.
//offset[0] is the start of string 0, offset[1] is the start of string 1,
etc.
const char USB_STRING_DESC_OFFSET[]={0,4,12};

#define USB_STRING_DESC_COUNT sizeof(USB_STRING_DESC_OFFSET)

char const USB_STRING_DESC[]={
   //string 0
        4, //length of string index
        USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING)
        0x09,0x04,   //Microsoft Defined for US-English
   //string 1
        8, //length of string index
        USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING)
        'C',0,
        'C',0,
        'S',0,
   //string 2
        36, //length of string index
        USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING)
        'V',0,
        'i',0,
        'c',0,
        't',0,
        'o',0,
        'r',0,
        ' ',0,
        'U',0,
        'S',0,
        'B',0,
        ' ',0,
        'D',0,
        'e',0,
        'v',0,
        'i',0,
        'c',0,
        'e',0,

};
#ENDIF
```

A.3 Instalando el controlador

Al conectar el microcontrolador PIC18F4550 a la PC se iniciará el proceso de detección de nuevo hardware, como se muestra en la figura A.1.

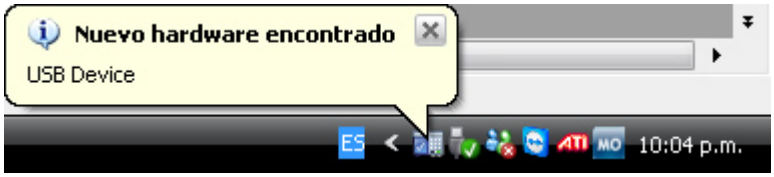


Figura A.1. *Nuevo hardware.*

A continuación aparecerá una ventana como se muestra en la figura A.2.

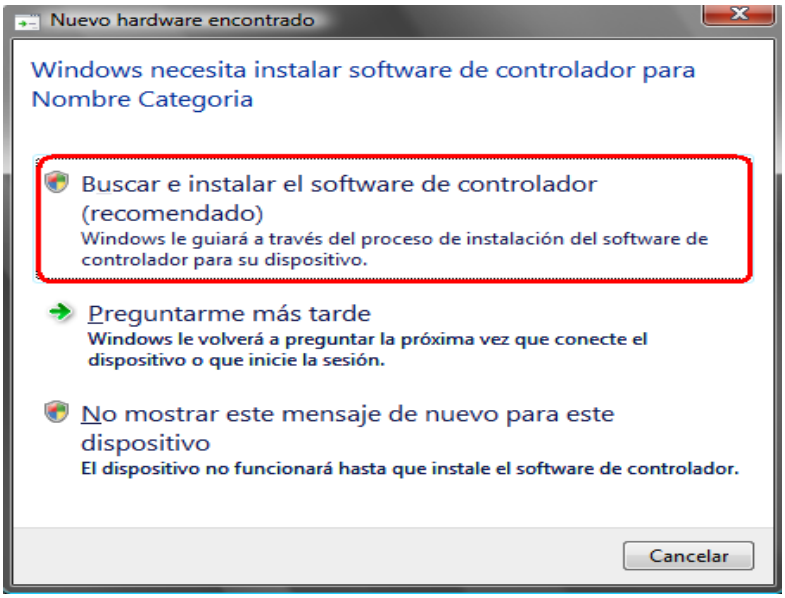


Figura A.2. *Nuevo hardware encontrado.*

Después, se localizará el controlador como se muestra en la figura A.3 y A.4.

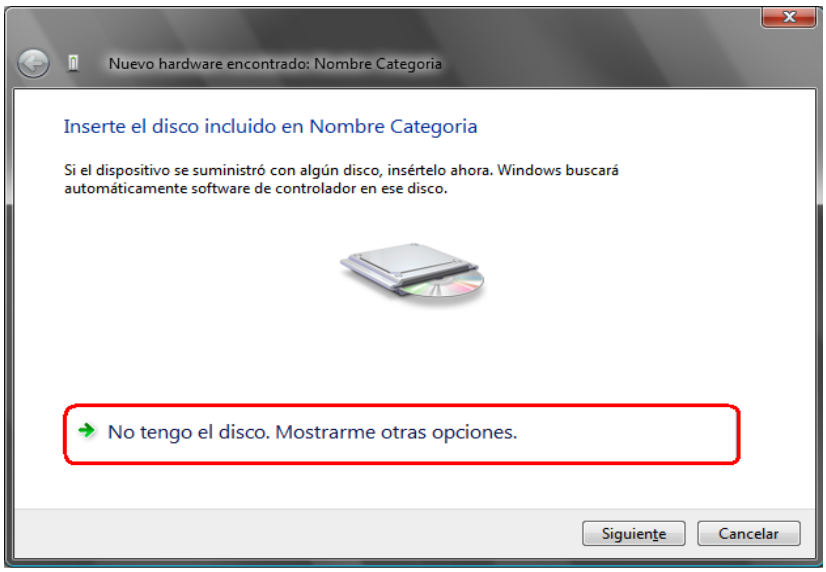


Figura A.3. *Insertar disco.*

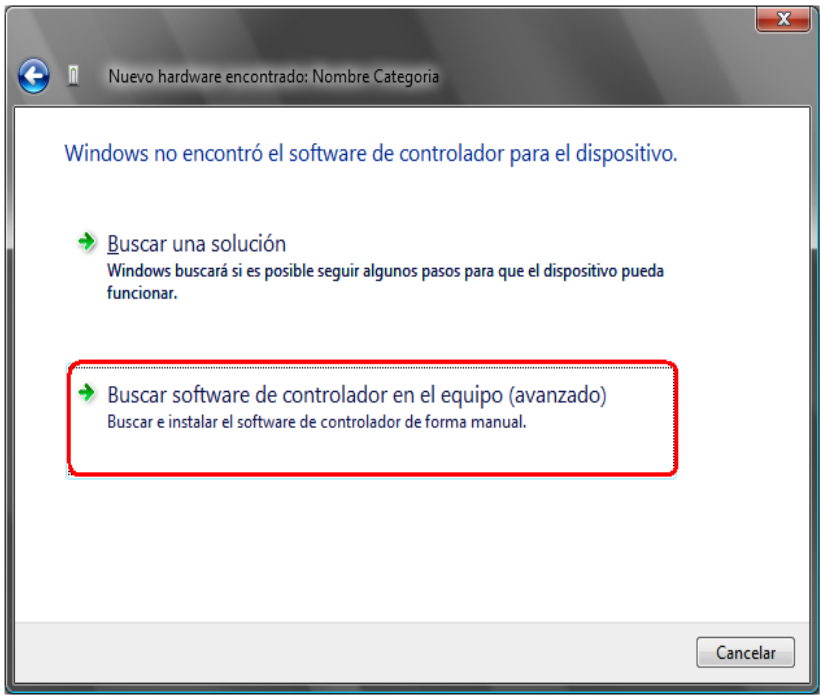


Figura A.4. *Buscar software.*

Aparecerá otra ventana preguntando por el origen del controlador, se buscara la ruta donde se encuentra el archivo *mchpusb.inf*, como se muestra en la figura A.5 a A.8.

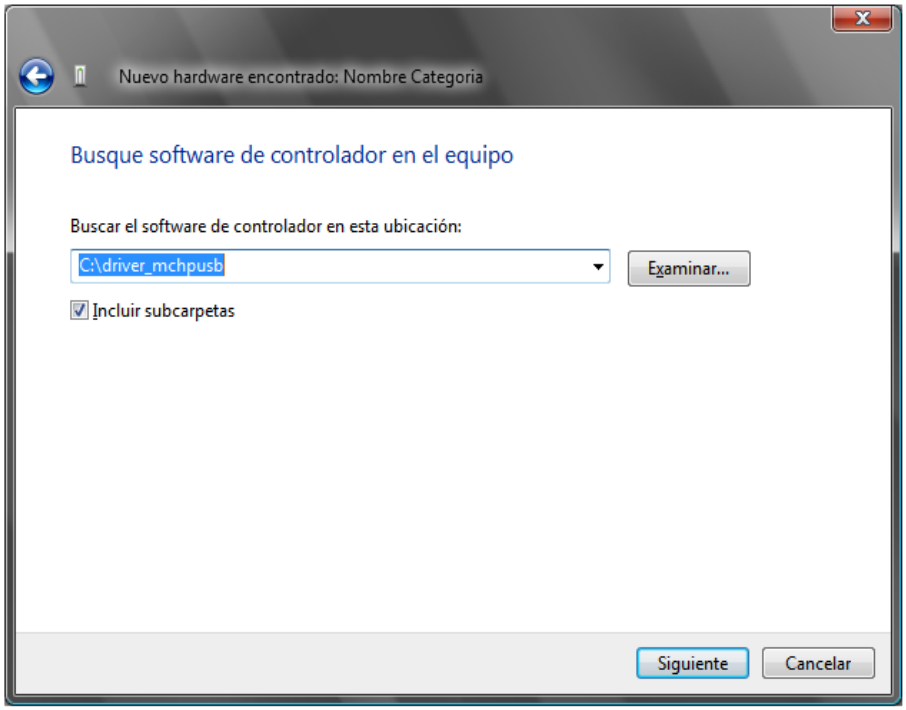


Figura A.5. *Ruta de archivo.*

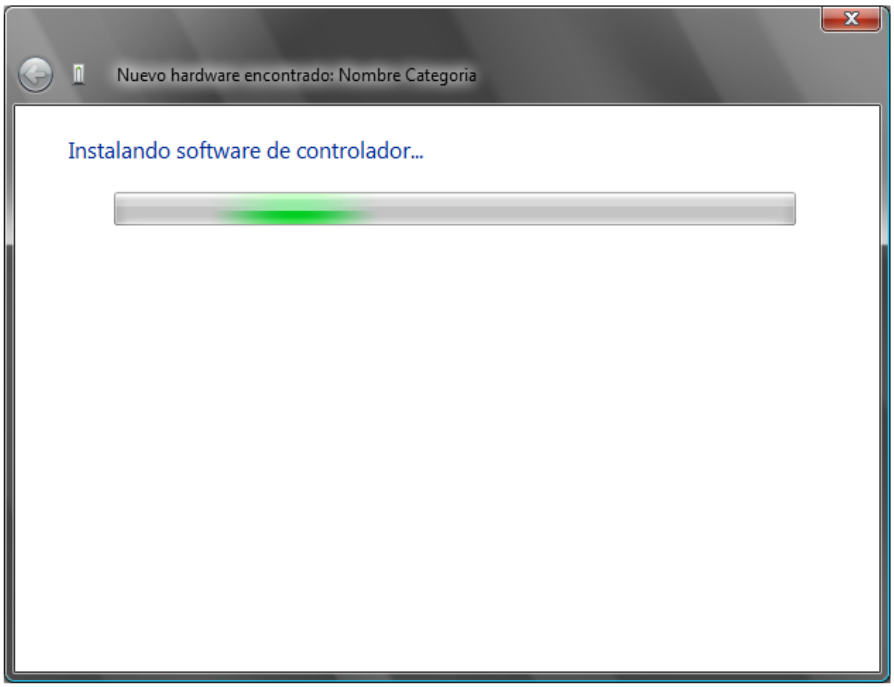


Figura A.6. *Instalando software.*

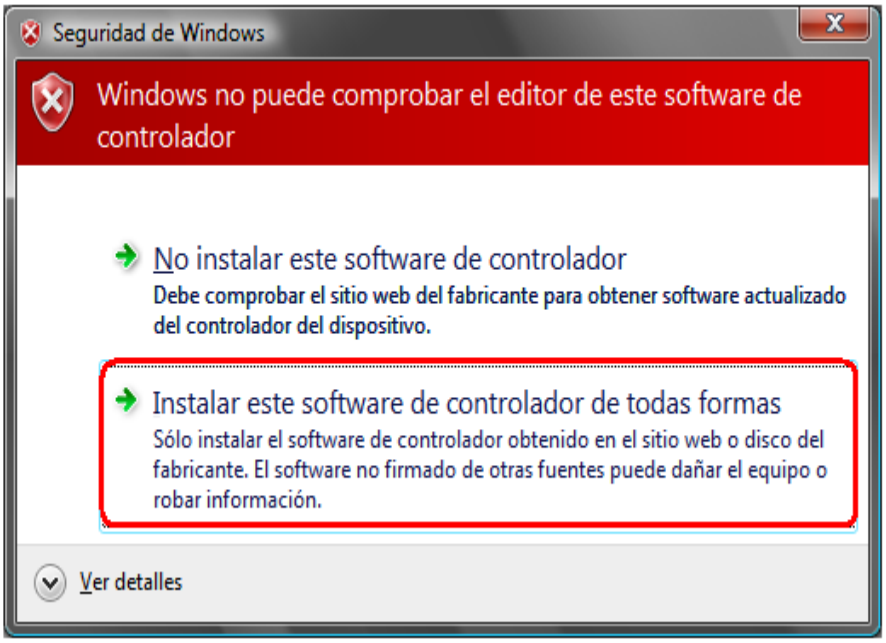


Figura A.7. *Seguridad de Windows.*

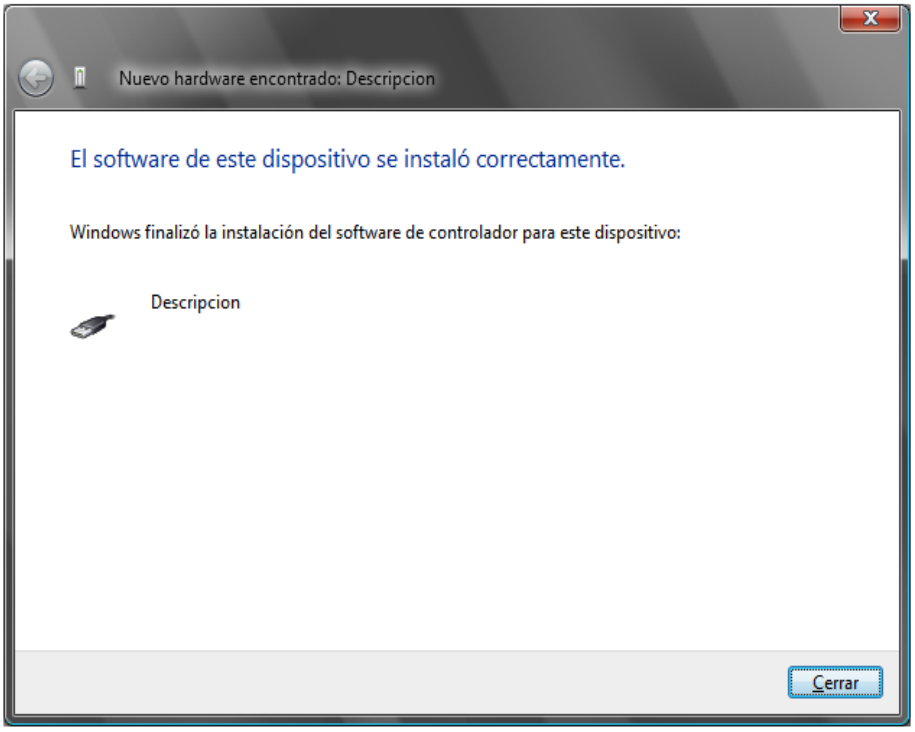


Figura A.8. *Finalizando.*

Finalmente tenemos los pasos completado exitosamente como se muestra en la figura A.9.

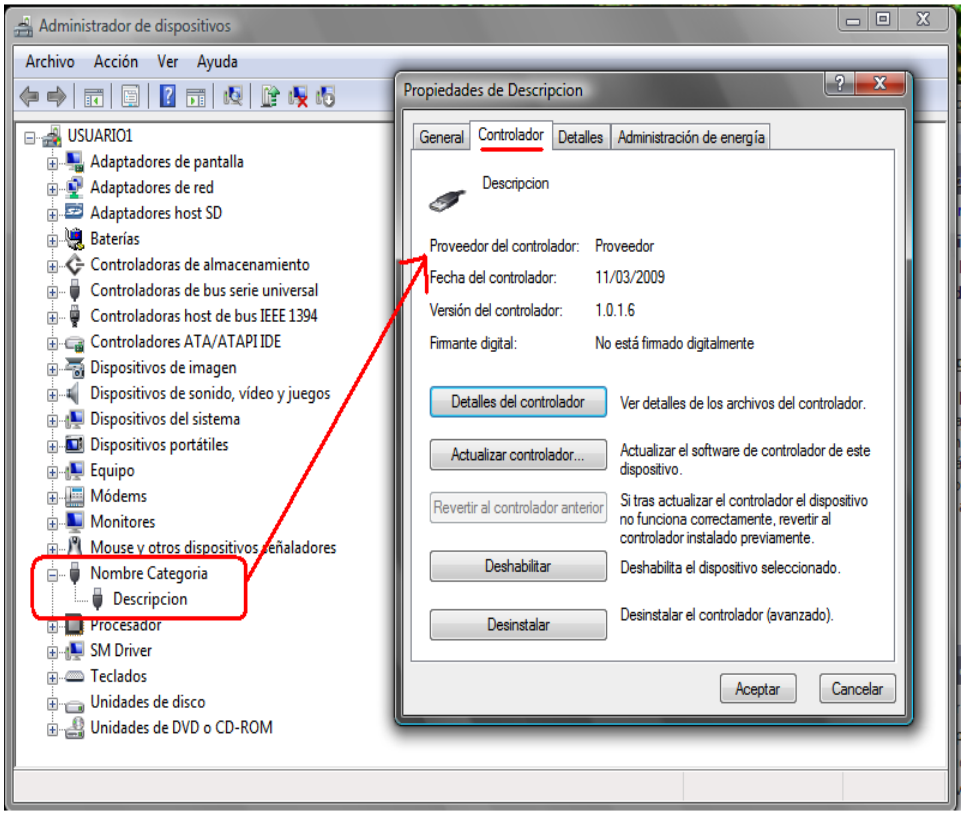


Figura A.9. *Verificación en el administrador de dispositivos.*

Código del archivo " mchpusb.inf "

```
; Installation file for Microchip's Custom USB Driver
; Copyright (C) 2007 by Microchip Technology, Inc.
; All rights reserved

[Version]
Signature=$Windows NT$
Class=CustomUSBDevice
ClassGuid={A503E2D3-A031-49DC-B684-C99085DBFE92}

Provider=%MFGNAME%
CatalogFile=%MFGFILENAME%.cat
DriverVer=03/11/2009,1.0.1.6

[Manufacturer]
%MFGNAME%=DeviceList,ntamd64

[DestinationDirs]
DefaultDestDir=12
icono_device=11

[SourceDisksNames]
1=%INSTDISK%,,,

[ClassInstall32]
```

```
AddReg=ClassInstall_AddReg

[ClassInstall_AddReg]
HKR,,,,%DEVICEMANAGERCATEGORY%
HKR,,EnumPropPages32,,"VICTOR.ico,0"

;------------------------------------------------------------------------------
;  Windows 2000/XP/Vista 32 Section
;------------------------------------------------------------------------------

[DriverInstall]
CopyFiles = DriverCopyFiles,icono_device

[DriverCopyFiles]
%MFGFILENAME%.sys,,,2

[icono_device]
VICTOR.ico,,,2

[DriverInstall.Services]
AddService=MCHPUSB,2,DriverService

[DriverService]
ServiceType=1
StartType=3
ErrorControl=1
ServiceBinary=%12%\%MFGFILENAME%.sys
;------------------------------------------------------------------------------
;  Windows XP/Vista 64 Section
;------------------------------------------------------------------------------

[DriverInstall64]
CopyFiles=DriverCopyFiles64

[DriverCopyFiles64]
%MFGFILENAME%64.sys,,,2

[DriverInstall64.Services]
AddService=MCHPUSB,2,DriverService64

[DriverService64]
ServiceType=1
StartType=3
ErrorControl=1
ServiceBinary=%12%\%MFGFILENAME%64.sys

;------------------------------------------------------------------------------
;  Vendor and Product ID Definitions
;------------------------------------------------------------------------------
; When developing your custom USB device, the VID and PID used in the PC
side
```

```
; application program and the firmware on the microcontroller must match.
; Modify the below line to use your VID and PID.  Use the format as shown
below.
; Note: One INF file can be used for multiple devices with different VID and
PIDs.
; For each supported device, append ",USB\VID_xxxx&PID_yyyy"to the end of
the line.
;------------------------------------------------------------------------------
[DeviceList]
%DESCRIPTION%=DriverInstall, USB\VID_04D8&PID_0011

[DeviceList.ntamd64]
%DESCRIPTION%=DriverInstall64, USB\VID_04D8&PID_0011

;------------------------------------------------------------------------------
;  String Definitions
;------------------------------------------------------------------------------
;Modify these strings to customize your device
;------------------------------------------------------------------------------

[Strings]
DEVICEMANAGERCATEGORY="Victor USB Device"
MFGFILENAME= "mchpusb"
MFGNAME="Microchip Technology, Inc."
INSTDISK="Proveedor - Disco Instalacion "
DESCRIPTION="UNAM"

;------------------------------------------------------------------------------
;  Source Files
;------------------------------------------------------------------------------
;The source file name prefixes need to be the same name as the string
MFGFILENAME
;above
;------------------------------------------------------------------------------

[SourceDisksFiles]
mchpusb.sys=1
mchpusb64.sys=1
VICTOR.ico=1
```

A.4 Diagramas electrónicos

La presente sección contiene información útil para analizar o reproducir el sistema electrónico desarrollado.

A.4.1 Dibujo esquemático

El dibujo esquemático consta de microcontrolador, puente H y el circuito de medición, figura A.10.

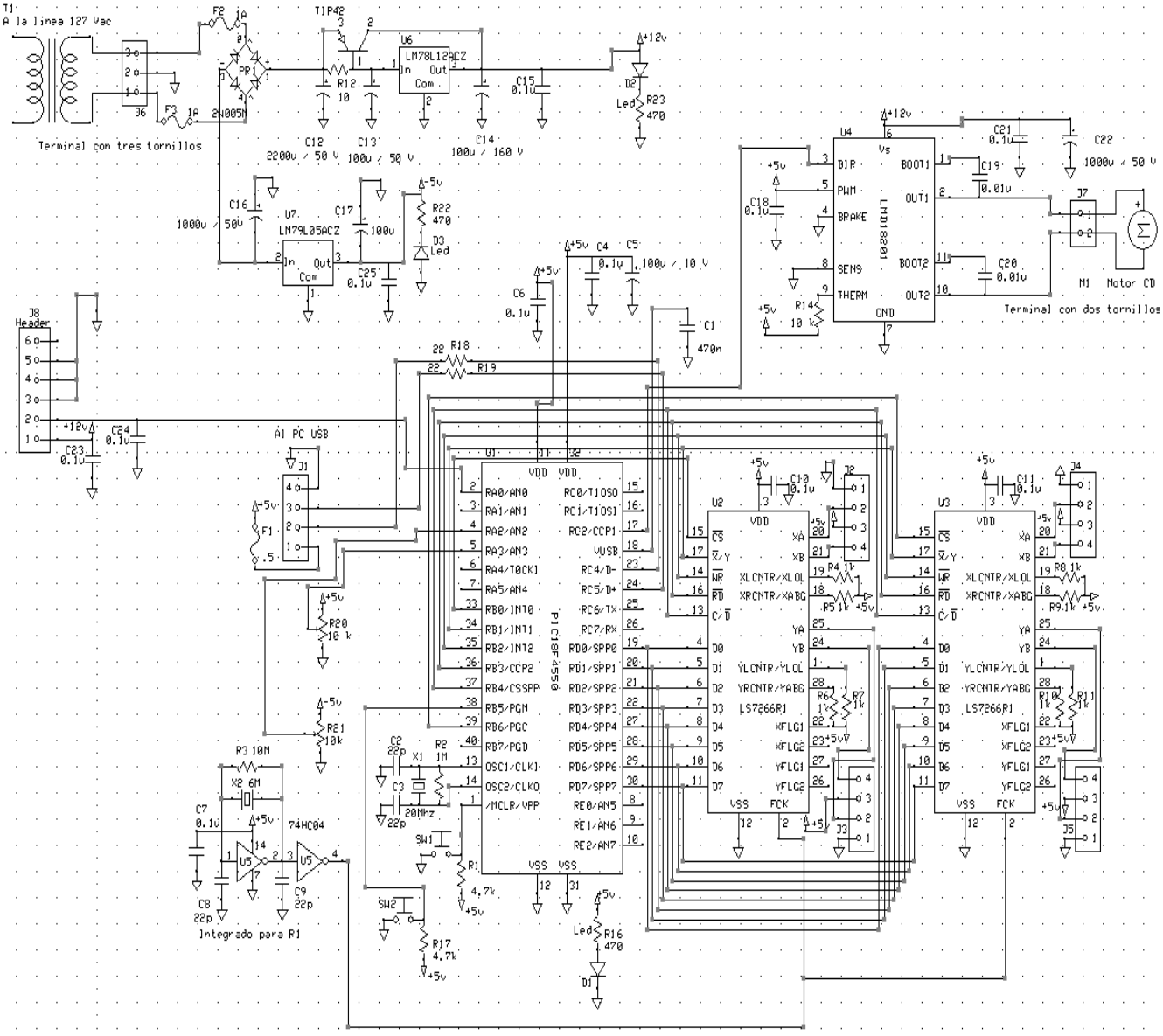


Figura A.10. *Esquemático del sistema electrónico.*

A.4.2 Circuito impreso

El circuito impreso del sistema electrónico es de doble cara, figuras A.11 y A.12.

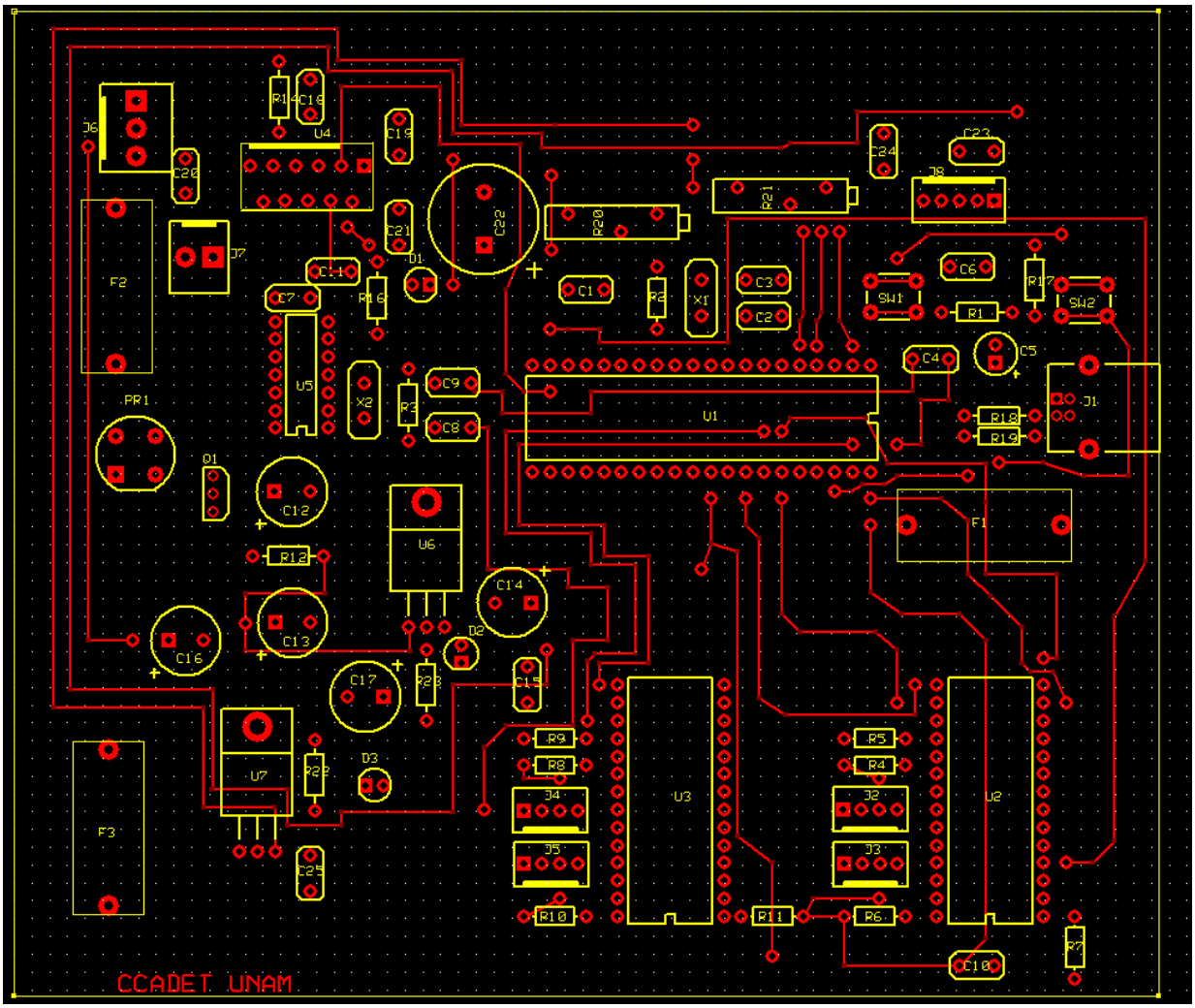


Figura A.11. *Circuito impreso capa superior.*

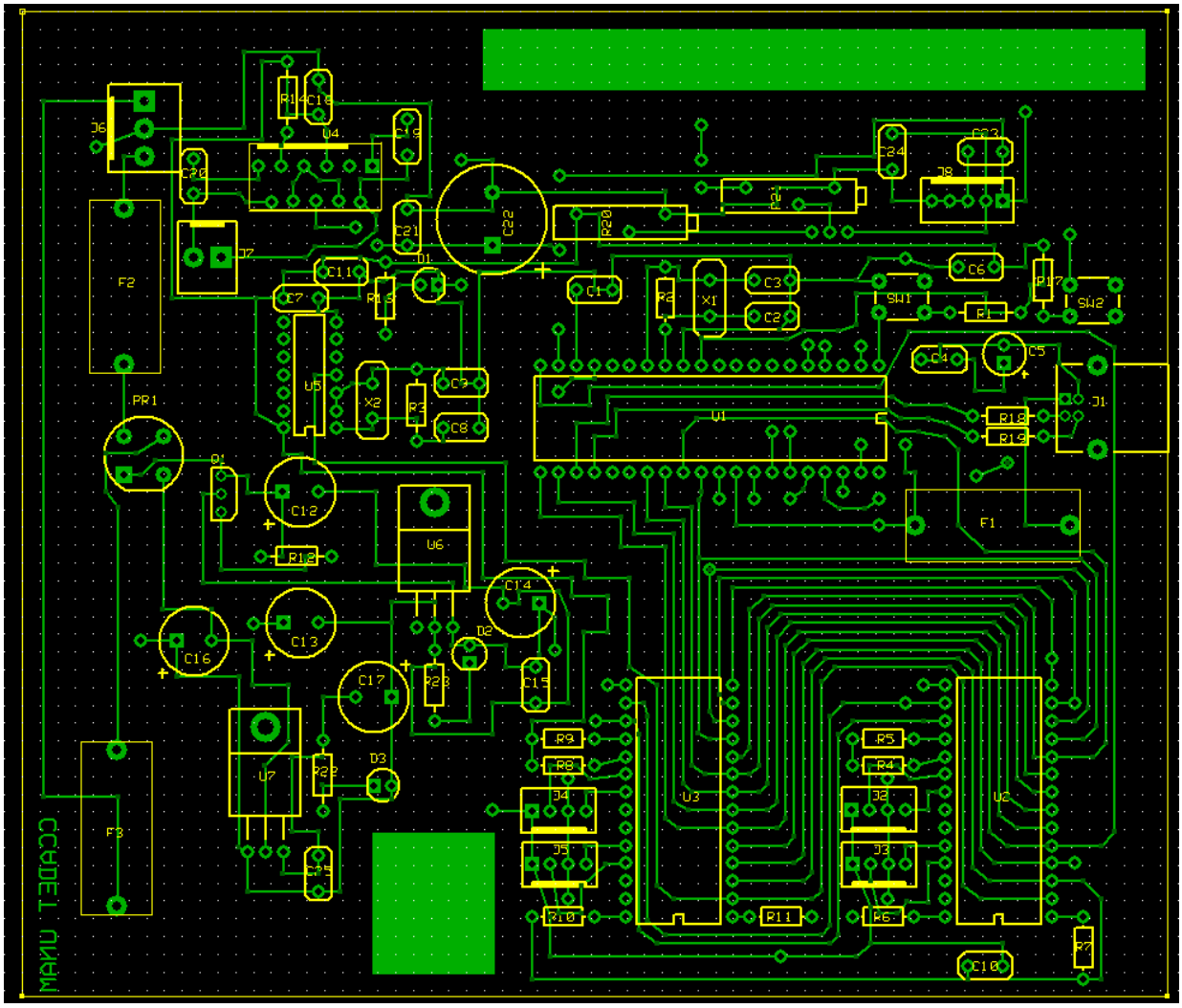


Figura A.12. *Circuito impreso capa inferior.*

A.4.3 Lista de material

La lista del material se resume en la tabla A.1.

Parte	Cantidad	Componente
C1	1	Capacitor cerámico monolítico 470n
C2,C3,C8,C9	4	Capacitor cerámico 22p
C4,C6,C7,C10,C11,C15,C18,C21,C23,C24,C25	11	Capacitor cerámico monolítico 0.1u
C5	1	Capacitor electrolítico 100u/10V
C12	1	Capacitor electrolítico 2200u/50V
C13,C17	2	Capacitor electrolítico 100u/50V
C14	1	Capacitor electrolítico 100u/160V
C16,C22	2	Capacitor electrolítico 1000u/50V

C19,C20	2	Capacitor cerámico monolítico 0.01u
D1,D2,D3	3	Led
F1	1	Fusible 0.5 A
F2.F3	2	Fusible 1 A
J1	1	Conector USB tipo B DIP
J2-J5	4	Header 4 posiciones
J6	1	Terminal con tres tornillos
J7	1	Terminal con dos tornillos
J8	1	Header 5 posiciones
PR1	1	Puente de diodos 2W005M
Q1	1	TIP 42
R1,R17	2	Resistencia 1/4W 4k7
R2	1	Resistencia 1/4W 1M
R3	1	Resistencia 1/4W 10M
R4-R11	8	Resistencia 1/4W 1k
R12	1	Resistencia 1/2W 10
R14,R20,R21	3	Resistencia 1/4W 10k
R16,R22,R23	3	Resistencia 1/4W 470
SW1, SW2	2	Push boton
T1	1	Transformador 24V
U1	1	PIC18F4550
U2,U3	2	LS7266R1
U4	1	LMD18201D
U5	1	74HC04
U6	1	LM7812ACZ
U7	1	LM79L05ACZ
X1	1	Cristal 20 MHZ
X2	1	Cristal 6 MHZ

Tabla A.1. *Lista de material.*

A.5 Programación del PIC18F4550

A continuación se presenta la programación en C para PIC, desarrollado para el presente proyecto de tesis.

```
#include <18F4550.h>
#device adc=10
//#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL3,CPUDIV1,VREGEN
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=48000000)


//******************PROGRAMA PARA UTILIZAR EL BOOTLOADER EN CCS COMPILER********//
#define LOADER_END 0x7FF
#define LOADER_SIZE 0x6FF
#build(reset=LOADER_END+1, interrupt=LOADER_END+9)
```

```
#org 0, LOADER_END {} //nada substituirá la memoria del bootloader
// *****************END OF bootloader ************************//
////////////////////////////////////////////////////////////////////////
//
// CCS Library dynamic defines.  For dynamic configuration of the CCS Library
// for your application several defines need to be made.  See the comments
// at usb.h for more information
//
////////////////////////////////////////////////////////////////////////
#define USB_HID_DEVICE     FALSE             //deshabilitamos el uso de las directivas HID
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK   //turn on EP1(EndPoint1) for IN bulk/interrupt
transfers
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK   //turn on EP1(EndPoint1) for OUT bulk/interrupt
transfers
#define USB_EP1_TX_SIZE    5            //size to allocate for the tx endpoint 1 buffer
#define USB_EP1_RX_SIZE    3           // size to allocate for the rx endpoint 1 buffer

////////////////////////////////////////////////////////////////////////
//
// Include the CCS USB Libraries.  See the comments at the top of these
// files for more information
//
////////////////////////////////////////////////////////////////////////
#include <pic18_usb.h>     //Microchip PIC18Fxx5x Hardware layer for CCS's PIC USB driver
#include <PicUSB.h>         //Configuración del USB y los descriptores para este dispositivo
#include <usb.c>           //handles usb setup tokens and get descriptor reports

////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//    Se definen los bytes de recepcion y de transmisión                //
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
#define modo         recibe[0]
#define pwmLB        recibe[1]
#define pwmHB        recibe[2]
#define VALOR1       envia[0]
#define VALOR2       envia[1]
#define VALOR3       envia[2]
#define adcLB        envia[3]
#define adcHB        envia[4]


//*********************************************************************//
//****************FUNCIONES PARA CONFIGURACIÓN INICIAL ***************//
//***************************PRIMER LS7266R1************************//
//*********************************************************************//

void Reset_E(void)
{
output_d(0x86);
output_b(0b01011000);
delay_cycles(2);
output_high(PIN_B0);
}

void Reset_BP(void)
{
output_d(0x81);
output_b(0b01011000);
delay_cycles(2);
output_high(PIN_B0);
}

void PR0_TO_PSC(void)
{
output_d(0x98);
output_b(0b01011000);
delay_cycles(2);
output_high(PIN_B0);
}
void A_B_INT(void)
```

```
{
output_d(0xC1);
output_b(0b01011000);
delay_cycles(2);
output_high(PIN_B0);
}

void BCD_X4_NORMAL(void)
{
output_d(0xB8);
output_b(0b01011000);
delay_cycles(2);
output_high(PIN_B0);
}

void Reset_CNTR(void)
{
output_d(0x82);
output_b(0b01011000);
delay_cycles(2);
output_high(PIN_B0);
}

//*********************************************************************//
//***************FUNCIONES PARA CONFIGURACIÓN INICIAL **************//
//*************************SEGUNDO LS7266R1************************//
//*********************************************************************//
void Reset2_E(void)
{
output_d(0x86);
output_b(0b00011001);
delay_cycles(2);
output_high(PIN_B6);
}

void Reset2_BP(void)
{
output_d(0x81);
output_b(0b00011001);
delay_cycles(2);
output_high(PIN_B6);
}

void PR0_TO2_PSC(void)
{
output_d(0x98);
output_b(0b00011001);
delay_cycles(2);
output_high(PIN_B6);
}
void A_B_INT2(void)
{
output_d(0xC1);
output_b(0b00011001);
delay_cycles(2);
output_high(PIN_B6);
}

void BCD2_X4_NORMAL(void)
{
output_d(0xB8);
output_b(0b00011001);
delay_cycles(2);
output_high(PIN_B6);
}

void Reset2_CNTR(void)
{
output_d(0x82);
output_b(0b00011001);
delay_cycles(2);
```

```
output_high(PIN_B6);
}


//*********************************************************************//
//***************FUNCIÓN PARA CONFIGURACIÓN PWM INICIAL**************//
//*********************************************************************//
void pwm_inicial(void)
{
setup_ccp1(CCP_PWM);                    //modo PWM CCP1CON<3:0>
setup_timer_2(T2_DIV_BY_1,255,1);       //modo= T2CON<2:0>(prescaler),periodo=255 (PR2),
postscaler= T2CON<6:3>
set_pwm1_duty(511);                     //valor de 10 bits
}

//*********************************************************************//
//***************FUNCIÓN PARA CONFIGURACIÓN CONVERSIÓN AD*************//
//*********************************************************************//

void adc_inicial(void)
{
setup_adc_ports(AN0|VREF_VREF);
setup_adc(ADC_CLOCK_INTERNAL);

}

//*************************************************************************//
//***************FUNCIONES PARA LECTURA DEL PRIMER LS7266R1***************//
//*************************************************************************//
void CNTR_TO_OL(void)
{
output_d(0x90);
output_b(0b01011000);
delay_cycles(2);
output_high(PIN_B0);
}
void PORTB(void)
{
output_b(0b01000100);
delay_cycles(1);
}

void PORTB2(void)
{
output_b(0b01000110);//nuevo
delay_cycles(1);
}

void CS(void)
{
delay_cycles(1);
output_high(PIN_B0);

}

//*************************************************************************//
//***************FUNCIONES PARA LECTURA DEL SEGUNDO LS7266R1***************//
//*************************************************************************//

void CNTR_TO2_OL(void)
{
output_d(0x90);
output_b(0b00011001);
delay_cycles(2);
output_high(PIN_B6);
}

void PORTB3(void)
{
output_b(0b00000101);
delay_cycles(1);
```

```
}

void PORTB4(void)
{
output_b(0b00000111);//nuevo
delay_cycles(1);
}

void CS2(void)
{
delay_cycles(1);
output_high(PIN_B6);

}

//////////////////////////////////////////////////////////////////////////
//********************PROGRAMA PRINCIPAL*********************************//
//////////////////////////////////////////////////////////////////////////
void main(void) {

   int8 recibe[3];                  //declaramos variables
   int8 envia[5];
   int16 resultado;
   int16 value;


   //////////////CONFIGURA PORTB (CONTROL LS7266R1)//////////////////////////
   set_tris_b(0x00);                //PORTB salida
   output_b(0x00);                  //limpia PORTB
   output_high(PIN_B0);             //CS=1
   output_high(PIN_B6);


   //////////////CONFIGURA PORTD (DATOS LS7266R1)///////////////////////////
   set_tris_d(0x00);                //PORTD salida
   output_d(0x00);                  //limpia PORTD

   //////////////////////////////////////////////////////////////////////////
   //********************CONFIGURACIÒN DEL PRIMER LS7266R1*****************//
   //////////////////////////////////////////////////////////////////////////
   Reset_E();                       //RESET E
   Reset_BP();                      //RESET BP
   PR0_TO_PSC();                    //PR0 TO PSC
   A_B_INT();                       //A Y B ENTRADA
   BCD_X4_NORMAL();                 //BCD-COUNT-NORMAL X 4
   Reset_CNTR();                    //RESET CNTR

   //////////////////////////////////////////////////////////////////////////
   //********************CONFIGURACIÒN DEL SEGUNDO LS7266R1****************//
   //////////////////////////////////////////////////////////////////////////

   Reset2_E();                       //RESET E
   Reset2_BP();                      //RESET BP
   PR0_TO2_PSC();                    //PR0 TO PSC
   A_B_INT2();                       //A Y B ENTRADA
   BCD2_X4_NORMAL();                 //BCD-COUNT-NORMAL X 4
   Reset2_CNTR();                    //RESET CNTR


   //*****************CONFIGURACIÓN PWM*************************************//

   pwm_inicial();                   //configurar PWM inicial, 50% ciclo util y Fpwm=46.875 Khz

    //*****************CONFIGURACIÓN AD*************************************//
    adc_inicial();


   //********************INICIALIZANDO EL USB*******************************//

   usb_init();                      //inicializamos el USB
```

```
   usb_task();                       //habilita periferico usb e interrupciones
   usb_wait_for_enumeration();       //esperamos hasta que el PicUSB sea configurado por el host

   LED_OFF(LEDR);
   LED_ON(LEDV);                     //encendemos led verde

   while (TRUE)
   {
      if(usb_enumerated())                  //si el dispositivo está configurado
      {
         if (usb_kbhit(1))                  //si el endpoint de salida contiene datos del host
         {
            usb_get_packet(1, recibe, 3);   //cojemos el paquete de tamaño 3bytes del EP1 y
                                              almacenamos en recibe

            if (modo == 0) // ACTUALIZA PWM
            {
               resultado = pwmLB + (pwmHB*256);        //hacemos la suma 10 bits
               setup_ccp1(CCP_PWM);                    //modo PWM CCP1CON<3:0>
               setup_timer_2(T2_DIV_BY_1,255,1);       //modo= T2CON<2:0>(prescaler),periodo=255
                                                       (PR2), postscaler= T2CON<6:3>
               set_pwm1_duty(resultado);              // ciclo util

            }

            if (modo == 1) // LECTURA DEL  PRIMER "X"  LS7266R1
            {

            //////////////CONFIGURA PORTD SALIDA//////////////////////////
             set_tris_d(0x00);                 //PORTD salida
             output_d(0x00);                   //limpia PORTD

             CNTR_TO_OL();                    //CNTR to OL
             Reset_BP();                      //Reset BP
             //////////////CONFIGURA PORTD ENTRADA//////////////////////////
             output_d(0x00);                   //limpia PORTD
             set_tris_d(0xFF);                 //PORTD entrada

             ////////////ADQUIRIENDO LOS 24 BITS DEL LR7266R1/////////////////
             PORTB();
             VALOR1 = input_d();               //BYTE menos significativo
             CS();

             PORTB();
             VALOR2 = input_d();               //BYTE medio
             CS();

             PORTB();                         //BYTE más significativo
             VALOR3 = input_d();
             CS();

               usb_put_packet(1, envia, 3, USB_DTS_TOGGLE); //enviamos el paquete de tamaño
                                                              3byte del EP1 al PC
            }
            if(modo==2)//LECTURA DEL PRIMERO "Y" LS7266R1
           {
           //////////////CONFIGURA PORTD SALIDA//////////////////////////
             set_tris_d(0x00);                 //PORTD salida
             output_d(0x00);                   //limpia PORTD

             CNTR_TO_OL();                    //CNTR to OL
             Reset_BP();                      //Reset BP
             //////////////CONFIGURA PORTD ENTRADA//////////////////////////
             output_d(0x00);                   //limpia PORTD
             set_tris_d(0xFF);                 //PORTD entrada


             ////////////ADQUIRIENDO LOS 24 BITS DEL LR7266R1/////////////////
             PORTB2();
             VALOR1 = input_d();               //BYTE menos significativo
             CS();
```

```
  PORTB2();
  VALOR2 = input_d();              //BYTE medio
  CS();

  PORTB2();                        //BYTE mas significativo
  VALOR3 = input_d();
  CS();

    usb_put_packet(1, envia, 3, USB_DTS_TOGGLE); //enviamos el paquete de tamaño
                                                   3byte del EP1 al PC
}

if(modo==3)//LECTURA DEL SEGUNDO "X" LS7266R1
{
//////////////CONFIGURA PORTD SALIDA//////////////////////////
  set_tris_d(0x00);                //PORTD salida
  output_d(0x00);                  //limpia PORTD

  CNTR_TO2_OL();                   //CNTR to OL
  Reset2_BP();                     //Reset BP
  //////////////CONFIGURA PORTD ENTRADA//////////////////////////
  output_d(0x00);                  //limpia PORTD
  set_tris_d(0xFF);                //PORTD entrada

  ////////////ADQUIRIENDO LOS 24 BITS DEL LR7266R1/////////////////
  PORTB3();
  VALOR1 = input_d();              //BYTE menos significativo
  CS2();

  PORTB3();
  VALOR2 = input_d();              //BYTE medio
  CS2();

  PORTB3();                        //BYTE mas significativo
  VALOR3 = input_d();
  CS2();

    usb_put_packet(1, envia, 3, USB_DTS_TOGGLE); //enviamos el paquete de tamaño
                                                   3byte del EP1 al PC
}

 if(modo==4)//LECTURA DEL SEGUNDO "Y" LS7266R1
{
//////////////CONFIGURA PORTD SALIDA//////////////////////////
  set_tris_d(0x00);                //PORTD salida
  output_d(0x00);                  //limpia PORTD

  CNTR_TO2_OL();                   //CNTR to OL
  Reset2_BP();                     //Reset BP
  //////////////CONFIGURA PORTD ENTRADA//////////////////////////
  output_d(0x00);                  //limpia PORTD
  set_tris_d(0xFF);                //PORTD entrada

  ////////////ADQUIRIENDO LOS 24 BITS DEL LR7266R1/////////////////
  PORTB4();
  VALOR1 = input_d();              //BYTE menos significativo
  CS2();

  PORTB4();
  VALOR2 = input_d();              //BYTE medio
  CS2();

  PORTB4();                        //BYTE mas significativo
  VALOR3 = input_d();
  CS2();

    usb_put_packet(1, envia, 3, USB_DTS_TOGGLE); //enviamos el paquete de tamaño
                                                   3byte del EP1 al PC
}
```

```
          if (modo == 5) // ACTUALIZA PWM
           {
           set_adc_channel(0);
           delay_us(10);
           value=read_adc();
          VALOR1=value&0xff;
           VALOR2=value>>8;

            //VALOR1=255;
           //VALOR2=01;
           usb_put_packet(1, envia, 2, USB_DTS_TOGGLE); //enviamos el paquete de tamaño 3byte
                                                          del EP1 al PC
           }

        }
     }
  }
}
```

A.6 Implementación del archivo “pwmcon4Dlg.cpp”

A continuación se presentan los segmentos de programación más representativos desarrollados para el presente proyecto de tesis. Se omiten los programas completos debido a lo extenso del código y los derechos de propiedad intelectual.

Se muestra el código en tiempo de ejecución.

```
// ARCHIVO DE IMPLEMENTACIÓN

      // INICIALIZACIÓN DE LAS GRÁFICAS

      m_Graph.SetPlotAreaColor(RGB(0,0,0));
      m_Graph.SetGridColor(RGB(120,225,0));
      m_Graph.SetShowGrid(TRUE);
      m_Graph.SetFrameStyle(1);
      m_Graph.SetCaption("POSICIÓN ANGULAR THETA 3");
      m_Graph.SetXLabel("THETA 2");
      m_Graph.SetYLabel("THETA 3");

      m_Graph2.SetPlotAreaColor(RGB(0,0,0));
      m_Graph2.SetGridColor(RGB(120,225,0));
      m_Graph2.SetShowGrid(TRUE);
      m_Graph2.SetFrameStyle(1);
      m_Graph2.SetCaption("POSICIÓN ANGULAR THETA 4");
      m_Graph2.SetXLabel("THETA 2");
      m_Graph2.SetYLabel("THETA 4");

      m_Graph3.SetPlotAreaColor(RGB(0,0,0));
      m_Graph3.SetGridColor(RGB(120,225,0));
      m_Graph3.SetShowGrid(TRUE);
      m_Graph3.SetFrameStyle(1);
      m_Graph3.SetCaption("VELOCIDAD ANGULAR OMEGA 3");
      m_Graph3.SetXLabel("THETA 2");
      m_Graph3.SetYLabel("OMEGA 3");

      m_Graph4.SetPlotAreaColor(RGB(0,0,0));
```

```
m_Graph4.SetGridColor(RGB(120,225,0));
m_Graph4.SetShowGrid(TRUE);
m_Graph4.SetFrameStyle(1);
m_Graph4.SetCaption("VELOCIDAD ANGULAR OMEGA 4");
m_Graph4.SetXLabel("THETA 2");
m_Graph4.SetYLabel("OMEGA 4");

m_Graph5.SetPlotAreaColor(RGB(0,0,0));
m_Graph5.SetGridColor(RGB(120,225,0));
m_Graph5.SetShowGrid(TRUE);
m_Graph5.SetFrameStyle(1);
m_Graph5.SetCaption("ACELERACION ANGULAR ALFA 3");
m_Graph5.SetXLabel("THETA 2");
m_Graph5.SetYLabel("ALFA 3");

m_Graph6.SetPlotAreaColor(RGB(0,0,0));
m_Graph6.SetGridColor(RGB(120,225,0));
m_Graph6.SetShowGrid(TRUE);
m_Graph6.SetFrameStyle(1);
m_Graph6.SetCaption("ACELERACION ANGULAR ALFA 4");
m_Graph6.SetXLabel("THETA 2");
m_Graph6.SetYLabel("ALFA 4");


// PROGRAMA EN TIEMPO DE EJECUCIÓN

myUSB.modo1();//modo lectura, VALOR DE THETA 2
m_leer=(double)(((360.0*myUSB.RESULTADO_R11())/(4096)));
              //recibiendo valor del primer eje X

////nota no se usa el segundo codificador.

myUSB.modo3();//modo lectura, COMPLEMENTO DE THETA 3
m_leer3=(double) (360*myUSB.RESULTADO_R33())/(4096);
              //recibiendo valor del segundo eje X

myUSB.modo4();//modo lectura, VALOR DE THETA 4
m_leer4=(double)(87.134+((360.0*myUSB.RESULTADO_R44())/(4096.0)));
              //recibiendo valor del segundo eje Y

//se mandan a llamar a las gráficas
m_Graph.SetElementLineColor(RGB(225,255,0));
m_Graph.SetRange(0,360,20,71);

m_Graph2.SetElementLineColor(RGB(225,255,0));
m_Graph2.SetRange(0,360,75,145);

m_Graph3.SetElementLineColor(RGB(225,255,0));
m_Graph3.SetRange(0,360,-0.039,0.065);

m_Graph4.SetElementLineColor(RGB(225,255,0));
m_Graph4.SetRange(0,360,-0.06,0.08);

m_Graph5.SetElementLineColor(RGB(225,255,0));
m_Graph5.SetRange(0,360,-0.0045,0.0034);

m_Graph6.SetElementLineColor(RGB(225,255,0));
```

```
m_Graph6.SetRange(0,360,-0.0135,0.004);

//theta 3 no se puede medir directamente del tercer codificador, se
  utiliza una relación geométrica para determinar el valor de THETA 3.
m_theta3=(double) (180.0-((-m_leer3+38.6248)+(180.0-m_leer4)));


//Se definen constantes y variables a utilizar
doublepi=3.14159265359,omega2,senos,senos2,relacion,relacion2,
relacion3,relacion4,a,b,c,d,e,alfa2,A1,B1,C1,D1,E1,C2,D2,E2;

//Se utliza la ecuación 1.37 y se muestra OMEGA3
omega2= (-1.0)*((2.0*pi*1.0)/(60.0));
relacion=(15.0/40.0);
a=((m_leer4-m_leer));
b=((m_leer4-m_theta3));
senos=((sin(a*(pi/180.0)))/(sin(b*(pi/180.0))));
m_omega33=-relacion*omega2*senos;

//Se utliza la ecuación 1.38 y se muestra OMEGA4
relacion2=(15.0/30.0);
c=((m_theta3-m_leer));
d=((m_leer4-m_theta3));
e=((m_theta3-m_leer4));
senos2=((sin(c*(pi/180.0)))/(sin(d*(pi/180.0))));
m_omega44=-relacion2*omega2*senos2;


//Se utliza la ecuación 1.48 y se muestra ALFA 3
m_alfa33=-(relacion3)*((C1-D1)/(E1));

//Se utliza la ecuación 1.49 y se muestra omega3
m_alfa44=-(relacion4)*((C2-D2)/(E2));

//Se asignan variables para graficar
      dataY1=m_leer;
      dataY2=m_theta3;
      dataY3=m_leer4;
      dataY4=m_omega33;
      dataY5=m_omega44;
      dataY6=m_alfa33;
      dataY7=m_alfa44;

//Se muestran en pantalla con valores reales.

      m_Graph.PlotXY(dataY1,dataY2,0); //THETA2 V.S THETA3
      m_Graph2.PlotXY(dataY1,dataY3,0);//THETA2 V.S THETA4
      m_Graph3.PlotXY(dataY1,dataY4,0);//THETA2 V.S OMEGA3
      m_Graph4.PlotXY(dataY1,dataY5,0);//THETA2 V.S OMEGA4
      m_Graph5.PlotXY(dataY1,dataY6,0);//THETA2 V.S ALFA3
      m_Graph6.PlotXY(dataY1,dataY7,0);//THETA2 V.S ALFA4
```

A.7 Plano de ensamble del prototipo mecánico.

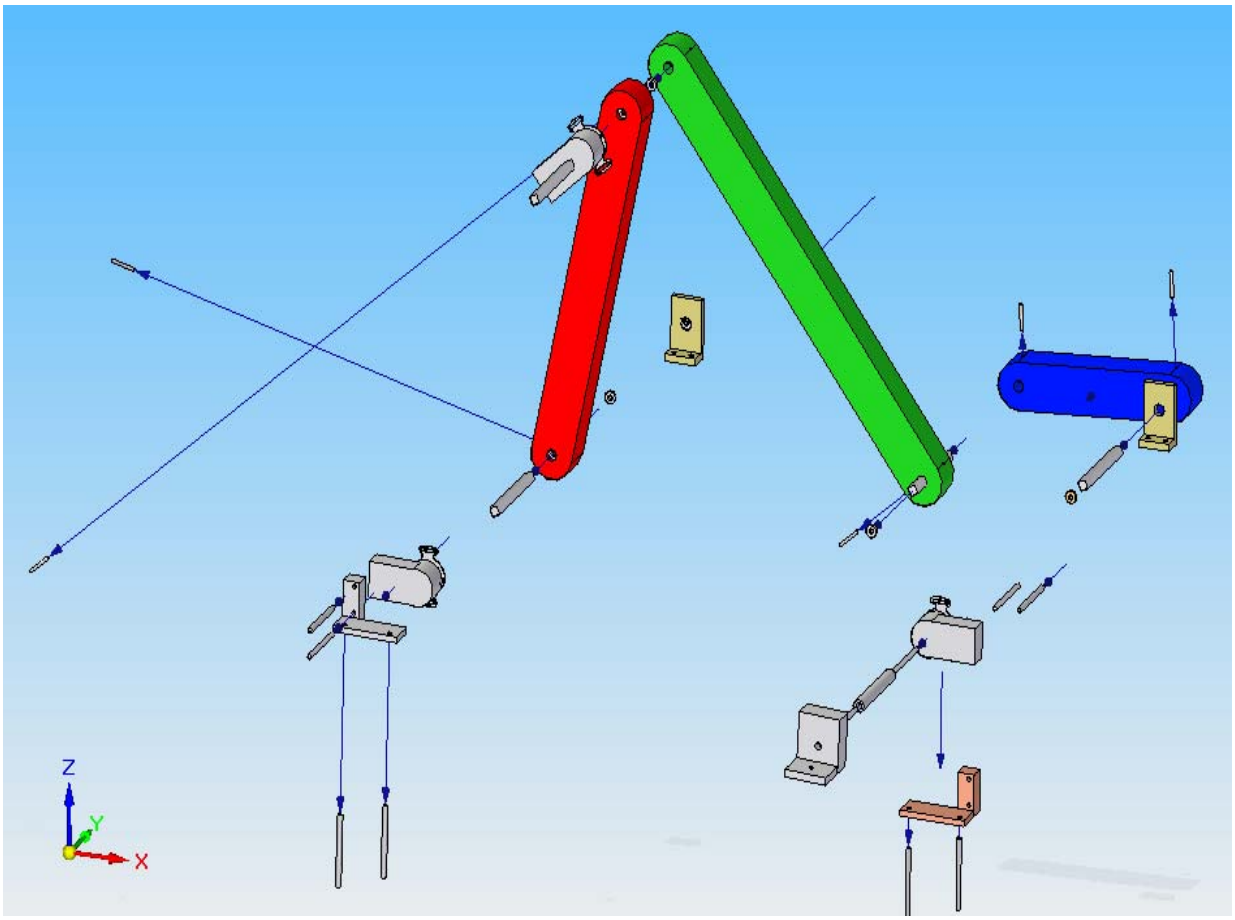


Figura A.13. *Plano de ensamble del prototipo mecánico.*