UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

Sistema Posicionador para Camara Monocular Utilizando Motores de Paso y Microcontroladores para Implementarse en una Plataforma de Celda de Manufactura Experimental

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

INGENIERÍA ELECTRICA - SISTEMAS ELECTRÓNICOS

P R E S E N T A :

PABLO BECERRIL RODRÍGUEZ

TUTOR:

Dr. MARIO PEÑA CABRERA

2012

Jurado Asignado:

Presidente:	Dr. PEÑA CABRERA MARIO
Secretario:	Dra. OROPEZA RAMOS LAURA
Vocal:	Dra. NAVARRETE MONTESINOS MARGARITA
1$^{er.}$ Suplente:	Dr. PRADO MOLINA JORGE
2$^{do.}$ Suplente:	M.I. HARO RUÍZ LUIS ARTURO

Lugar donde se realizó la Tesis:

Instituto de Investigaciones en Matemáticas Aplicadas
y en Sistemas
Universidad Nacional Autónoma de México

TUTOR DE TESIS:

Dr. PEÑA CABRERA MARIO

FIRMA

Dedicatoria:

A mis padres Ricardo y Lupita por su gran afecto y apoyo incondicional a la largo de estos años.

Agradecimientos:

A la Universidad Nacional Autónoma de México por permitir mi superación como individuo y profesional.

Al Consejo de Estudios de Posgrado (CEP) por el apoyo económico brindado durante este posgrado.

Al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) por permitirme hacer uso de sus instalaciones durante el trascurso de mis estudios de maestría.

Al Dr. Mario Peña Cabrera por asesorarme en el desarrollo de este proyecto.

A la Dra. Margarita Navarrete Montesinos, a la Dra. Laura Oropeza Ramoas, al Dr. Jorge Prado Molina y al M.I. Arturo Haro Ruíz por el tiempo dedicado a la revisión de este trabajo.

A mi compañero Javier Martínez por sus valiosos consejos y apoyo en la realización de este proyecto.

A Esther Bernal por permitirme formar parte de su vida y por estar conmigo en todo momento.

Y sobre todo a Dios quien me sostiene y hace todo posible.

Resumen

Se presenta un sistema posicionador para una cámara web con dos grados de libertad lineal asociados a los ejes del plano cartesiano x y y. El posicionamiento se logra con un microcontrolador, una PC y dos tipos de motores eléctricos: a) de paso y b) corriente continua.

El trabajo describe el diseño de la estructura del posicionador, la electrónica requerida para manejar los motores y los elementos mecánicos empleados para desplazar el sensor de visión en el plano xy. Asimismo, se describe el proceso de adquisición de una imagen y se plantea un método de calibración de cámara que es utilizado para corregir el error de posicionamiento.

El posicionador forma parte de un sistema de visión que le permite a un robot manipulador de seis grados de libertad, obtener información del POSE (localización del objeto) para realizar tareas de ensamble en línea. El sistema de visión forma parte de una celda de manufactura inteligente experimental que se está implementado en el IIMAS (Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas).

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Los posicionadores industriales, conocidos como sistemas gantry, presentan una estructura prismática rectangular con una configuración que permite tres movimientos lineales. Cada uno de éstos es perpendicular entre si y se asocia a los ejes del sistema de coordenadas cartesianas (x, y, z). Los ejes x y y están situados en el plano horizontal mientra que el eje z se ubica en el plano vertical. Por lo general, se adapta una herramienta (o efector) al eje z de modo que esta se puede llevar a un punto específico dentro de un espacio designado como volumen de trabajo. El *volumen de trabajo* se define como el área dentro de la cual un robot manipulador coloca y orienta el efector final [1].

Los sistemas gantry fueron construidos por primera vez en los años 50 por la corporacion General Mills para manipular materiales radiactivos, conforme fueron evolucionando, se comenzaron a emplear para realizar tareas de soldadura, carga y descarga, perforación, pintura, y apilamiento de cajas voluminosas.

En la actualidad, se encuentran empreas tales como Schneider Electric Motion, Cimcorp y Parker[1] que se dedican a la fabricación de sistemas gantry, así como a la fabricación de las partes que lo conforman. Los posicionadores que ofrecen dichas empresas se caracterizan por el desplazamiento máximo que se tiene por eje, por la carga que soporta, por la velocidad de desplazamiento y por la precisión de su movimiento. El fabricante específica la precisión de su máquina en términos de repetibilidad y exactitud. La Figura 1.1 muestra un sistema gantry de dos dimensiones que ofrece la empresa Parker. Sus características se presentan en el Cuadro 1.1.

[1] Schneider Electric Parker, www.schneider-electric.com y Parker Hannifin Corp 2012, www.parker.com

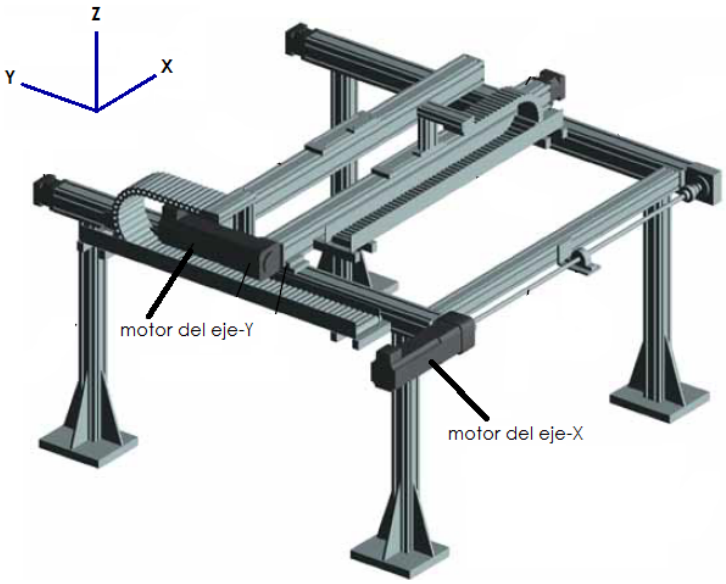


Figura 1.1. Sistema gantry de dos ejes [2].

Carga Máx	**Desplazamiento**		**Velocidad**	
	Eje x	Eje y	Eje x	Eje y
100	7.5	3	2	3

Cuadro 1.1. Características del sistema gantry de dos eje de la empresa Parker.

Aunque las características y dimensiones de un sistema gantry varían de un modelo a otro, existen componentes que se repiten en todas las concepciones que hay de este tipo de sistema. Sus módulos principales son:

- Estructura mecánica,
- actuadores (motores eléctricos),
- transmisión lineal, y
- unidad de control de movimiento.

El módulo de transmisión lineal es un sistema mono-eje diseñado para efectuar movimientos en una dimensión. Estos módulos integran un sistema de guía lineal y el elemento de transmisión de movimiento. El elemento de transmisión puede estar compuesto por un mecanismo tipo polea-correa, husillo-tuerca, o piñon-cremallera. La elección entre uno u otro depende en gran manera de la rapidez, fuerza, precisión así como la velocidad que se requiera.

1.1. Planteamiento del Problema

Se busca desarrollar un sistema posicionador usando una cámara web que se desplace dentro de una celda de manufactura experimental[2]. La celda de manufactura utiliza un sistema de visión para guiar los movimientos de un robot manipulador que reside dentro de la misma. El sistema de vision es el que le proporciona a la celda de manufactura la capacidad necesaria para maquinar los componentes de un producto de una manera autónoma.

La Figura 1.2 muestra la configuración general del sistema de visión para la celda de manufactura. La señal adquirida por la cámara es procesa para obtener una imagen binaria, sobre la cual se aplican algoritmos que permiten transformarla en una colección ordenada de pares numéricos. La misma es analizada y utilizada para obtener información de las coordenadas del centroide y puntos importantes. Éstos permiten el cálculo de la orientación del objeto para obtener el POSE[3]. El análisis de los pares númericos conduce a la obtención de un vector descriptivo que permite el reconocimiento de un objeto. La información del vector descriptivo se envía a una computadora maestra a través de un puerto serial para instruir al controlador los movimientos necesarios para alcanzar una parte que se quiere atrapar y luego realizar la tarea de ensamble requerida.

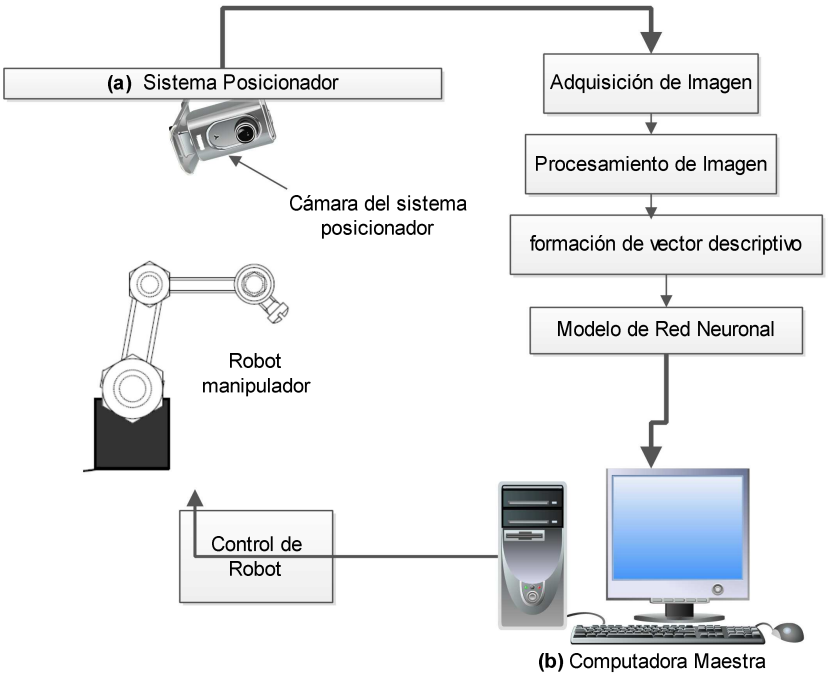


Figura 1.2. Configuración general del sistema de visión que consta de (a) sistema posicionador; (b) computadora maestra.

[2]Una *celda de manufactura* es un colección de equipo que se requiere para fabricar una parte aislada o una familia de partes con caracteristicas similares [3].

[3]El *POSE* de una imagen se refiere a la posición y orientación de un objeto.

Una cámara situada en un lugar fijo podría capturar una imagen que comprenda toda una escena del espacio donde se desenvuelve el robot manipulador. Sin embargo, un mayor campo de visión significa una mayor distancia entre la cámara y el área observada. Esta distancia trae como consecuencia una baja resolución en pixeles para objetos con dimensiones pequeñas.

1.1.1. Solución Propuesta

La solución aquí propuesta consiste en establecer primero una distancia constante entre la cámara y el área observada y darle movilidad para situarse en más de un punto dentro del área de trabajo del robot. Para ello, se propone el uso de un posicionador en configuración de vista área que desplace la cámara en dos direcciones que correspondan a los ejes x y y.

El área de influencia del posicionador debe envolver el volumen de trabajo del robot manipulador. Con base a este volumen, se establece un área de trabajo de 80 $\times$ 80 cm y una elevación de cámara de 1.5 m con respecto al suelo. La Figura 1.3 muestra las cuatro zonas que la cámara debe alcanzar. La división por zonas se formula con base en el tipo de tarea que se realiza dentro de tal espacio.

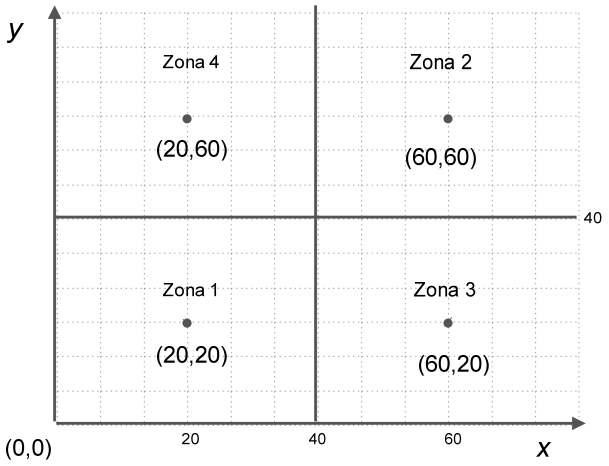


Figura 1.3. Zonas de trabajo.

Zona	**Coordenada** (en cm)
1	20,20
2	60,60
3	60,20
4	20,60

Cuadro 1.2. Puntos específicos por alcanzar.

Se propone un movimiento simultáneo por eje, para alcanzar los puntos específicos descritos en el Cuadro 1.2. Es decir, los dos ejes se movilizan en un instante dado

finalizando cuando el eje de mayor duración llega a una posición. Para un módulo de transmisión lineal basado en un husillo, se sabe que el avance lineal es igual al paso del elemento roscado por la velocidad angular [4]. Por tanto, el desplazamiento por eje en función del tiempo se describe por medio de las ecuaciones 1.1 y 1.2. Donde p representa el paso de rosca y ω representa la velocidad angular del motor eléctrico en términos de sus revoluciones por minuto (rpm).

$$x(t) = p \times t \times \omega_x \tag{1.1}$$

$$y(t) = p \times t \times \omega_y \tag{1.2}$$

La velocidad en cada eje esta en función del paso de la rosca y la velocidad angular de su motor. Por tanto, la velocidad de desplazamiento por eje para un módulo de transmisión lineal basado en un husillo se obtiene mediante las ecuaciones 1.3 y 1.4

$$v_x = p \times \omega_x \tag{1.3}$$

$$v_y = p \times \omega_y \tag{1.4}$$

1.2. Objetivos

1.2.1. Objetivo General

Diseñar y construir un sistema posicionador de dos dimensiones en la parte superior de una celda de manufactura que permita obtener cuadros de imagen en diferentes localidades del área de trabajo.

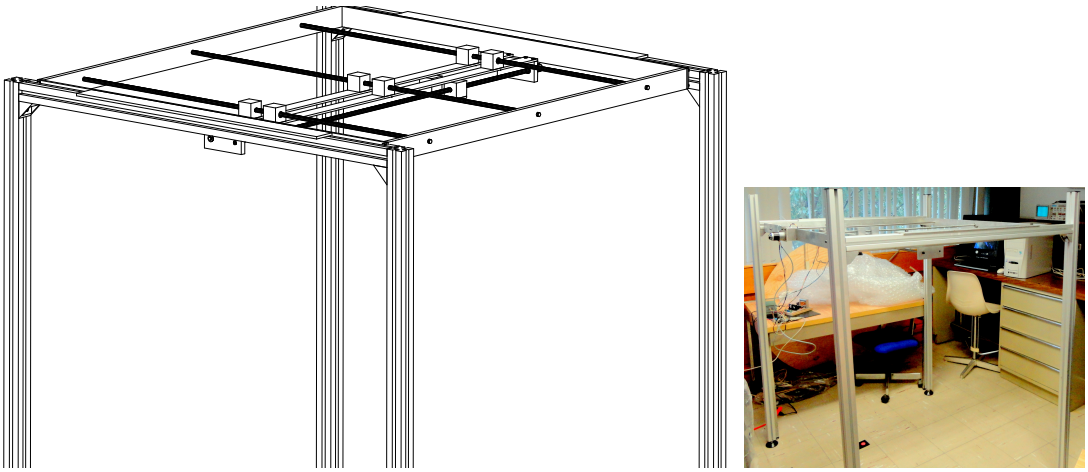
1.2.2. Objetivos Específicos

- Realizar el sistema con base en microcontroladores y dos motores eléctricos.
- Desarrollar una interfaz gráfica de usuario que permita enlazar los circuitos de control con una PC.
- Diseñar y construir un mecanismo que permita un desplazamiento lineal en los ejes x y y.
- Emplear las imágenes adquiridas por la cámara de video para corregir la posición de la misma en el plano xy.

Capítulo 2

Diseño y Manufactura del Posicionador Experimental

La configuración mecánica del sistema posicionador contempla una estructura tipo gantry de dos grados de libertad asociados a los ejes x y y. Los elementos de transmisión permiten desplazar una cámara web de 300 gr una distancia máxima de 80 cm por eje. La conceptualización de la estructura mecánica se formuló mediante el software CAD (diseño asistido por computadora, *computer-aided design*). Por medio de este programa se logró dimensionar y diseñar el posicionador con base en el área de trabajo requerida (ver Figura 2.1).



(a) Diseño en CAD (b) Implementación

Figura 2.1. Vista general del sistema posicionador.

2.1. Mecanismo de transmisión

El posicionador esta pensado para ser una alternativa más económica frente a la adquisición de módulos de transmisión lineal ya existentes. Para cumplir con esto, se decidió manufacturar el módulo de transmision lineal para reducir el costo del sistema de una manera substancial.

Tras una exploración de diversas opciones (ver Anexo A.1), se optó por emplear el mecanismo de transformación husillo-tuerca. En primera instancia se consideró el adquirir un tornillo de bola o Acme, para conformar el elemento roscado. Esta clase de tornillos tienen una buena eficiencia e incorporan tuercas anti-backlash[1]. Sin embargo, el costo de un tornillo con una longitud de 1 metro cuesta alrededor de 2,200 pesos. Finalmente se propuso el uso de un tornillo tipo espárrago, el cual se puede conseguir en cualquier tlapalería.

El tornillo que se utilizó, mostrado en la Figura 2.2, tiene un diametro de 3/8" y cuenta con 16 filetes por pulgada. De la ecuación A.2 se obtiene que el paso de rosca es:

$$p = \frac{1}{16} = 0{,}625 \text{ in o bien } 1.587 \text{ mm}$$

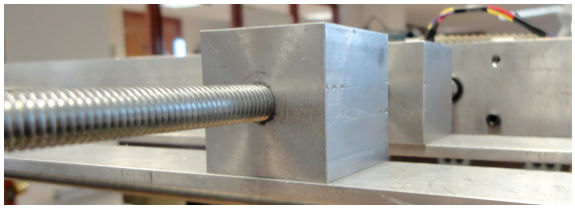


Figura 2.2. Mecanismo de transformación husillo-tuerca.

El mecanismo hace uso de rieles redondos que proporcionan la trayectoria de la guía lineal para los rodamientos lineales, los cuales viajan a lo largo del riel. Los rodamientos lineales junto con los rieles proporcionan el movimiento rectilíneo a lo largo de los ejes.

2.2. Elaboración de la estructura

La estructura del sistema posicionador mostrada en la Figura 2.3, se construyó con barras de aluminio (aleación 6061 y 6063) dado que este material es liviano, fuerte, resistente a la corrosión y fácil de manejar. La estructura consiste de un marco general, un cuadro-x y un cuadro-y. El primero sujeta al mecanismo que permite el movimiento de la cámara a lo largo del eje x mientras que el segundo hace lo propio en el eje y.

[1] *backlash* se refiere a la cantidad de juego o movimiento que presenta la tuerca cuando cambia de dirección a lo largo del tornillo [5].

El formato que se utiliza a lo largo de capítulo para especificar las dimensiones de las barras de aluminio es:

$$\text{largo} \times \text{ancho} \times \text{alto}.$$

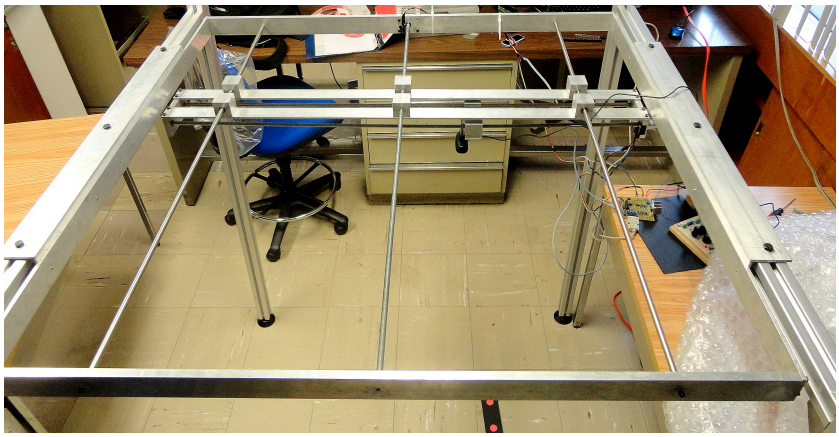


Figura 2.3. Ubicación de la cámara en el sistema posicionador.

2.2.1. Marco general

La estructura general delimita los bordes del sistema, sujeta los rieles redondos y el husillo del mecanismo de transmisión. El cuadro general se construyó con dos barras de aluminio de $1090 \times 6 \times 50$ mm y con dos de $1100 \times 12.7 \times 50$ mm. Las cuatro barras forman un cuadro de 109×111 cm, tal y como se presenta en la Figura 2.4. En los extremos de cada barra se taladraron 2 agujeros de 3 mm para unirlas.

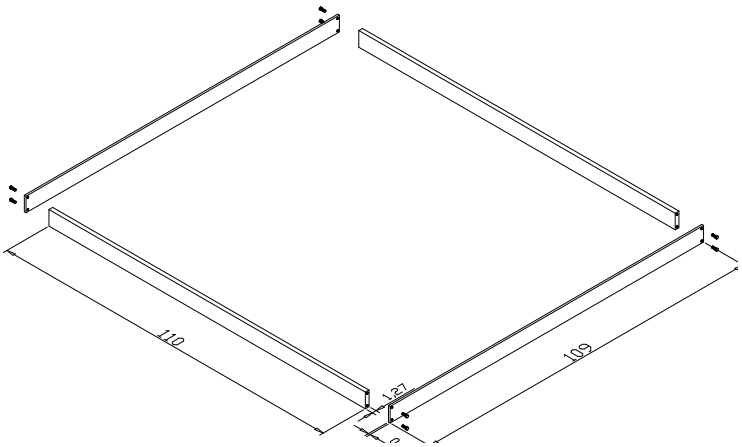


Figura 2.4. Marco general.

2.2.2. Estructura del eje Y

El cuadro-y está formado por dos barras de aluminio de 1090 × 4.8 × 38 mm y por dos barras de aluminio de 115 × 12.7 × 5 mm. En los extremos de las barras de 1090 mm se taladraron 2 agujeros de 3 mm para sujetar las barras de aluminio de 115 mm por medio de tornillos allen. Se colocaron seis bloques de aluminio de 38 × 38 mm a lo largo de la placa de 1090 mm tal y como se muestra en la Figura 2.5. Los dos bloques centrales llevan cuerda de tal forma que estos funcionen como las tuercas del mecanismo de transmisión. Los cuatro bloques laterales, situados a 40 cm de los bloques centrales, funcionan como el carruaje para las guias lineales en el cual se apoya el sistema de transmisión del eje y. Cada uno de estos cuatro bloques lleva incrustado un rodamientos lineal cerrado de 3/8". La distribución de los bloques se puede apreciar en la Figura 2.6.

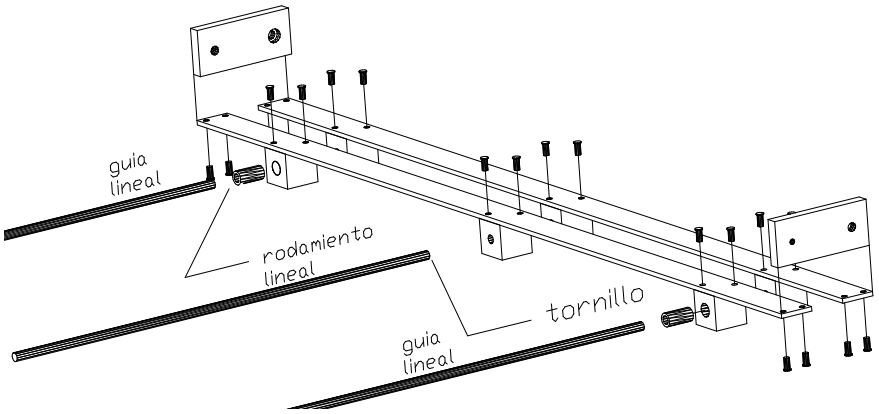


Figura 2.5. Cuadro-y.

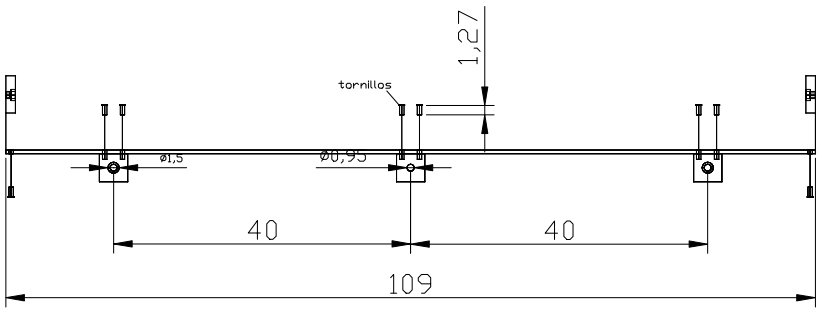


Figura 2.6. Vista lateral del cuadro-y.

2.2.3. Estructura del eje X

El cuadro-x está formado por dos bloques de aluminio de 4 × 4 mm y dos barras de 115 × 4.8 × 38 mm. Los dos bloques se colocan a una distancia 4 cm y se unen por medio de una barra de 115 mm. Esto se puede ver en la Figura 2.7. Un bloque de aluminio funciona como tuerca para el mecanismo de transmisión y el otro bloque funciona como carruaje para el sistema de guía. Tanto el riel redondo como el tornillo se fijan de sus extremos a las barras de 115 mm. Se incrustó un balero de 3/8" de diámetro en la barra de 115 mm para permitir que el tornillo gire de manera suave.

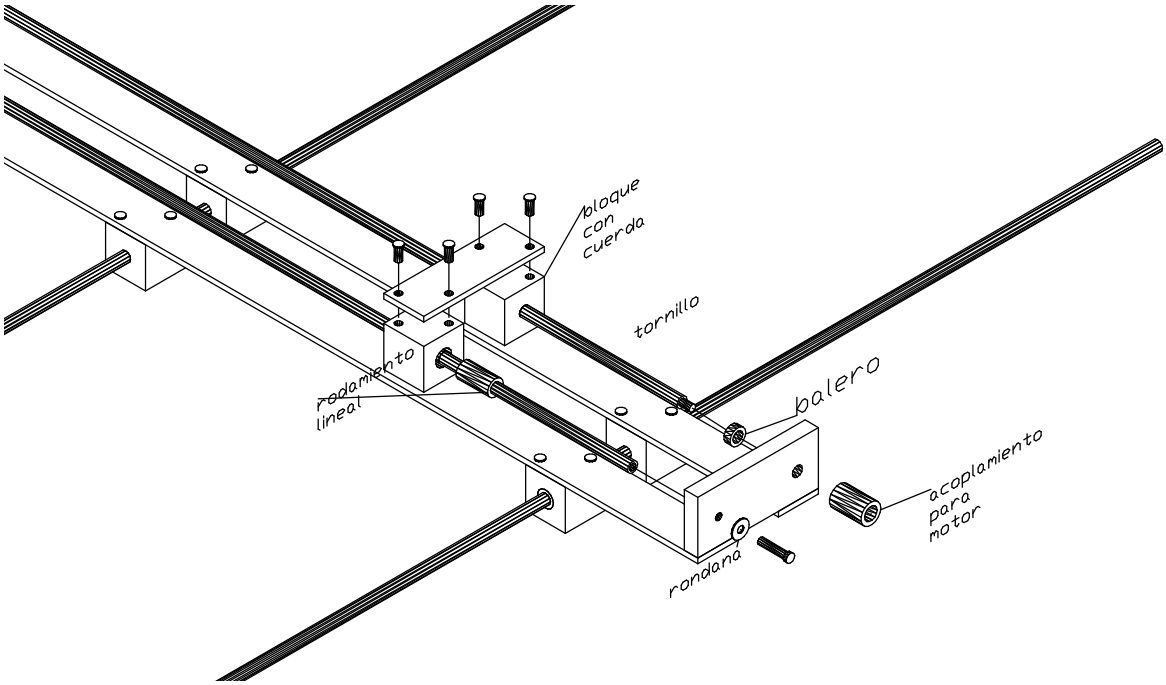


Figura 2.7. Cuadro-x.

Capítulo 3

Desarrollo de Circuitos Electrónicos

3.1. Esquema General

El circuito de control se diseñó para controlar los motores que accionan los mecanismos que desplazan la cámara en los ejes x-y. El circuito de control realiza dos tareas principales: el movimiento de los motores y el procesamiento e intercambio de datos con una computadora personal (PC). Los comandos referentes a las coordenadas del sistema posicionador se envían a través de un puerto serial, el cual se encuentra conectado a un módulo inalámbrico para transmitirse al circuito de control que maneja los motores. La Figura 3.1 muestra el diagrama a bloques de la unidad de control de movimiento.

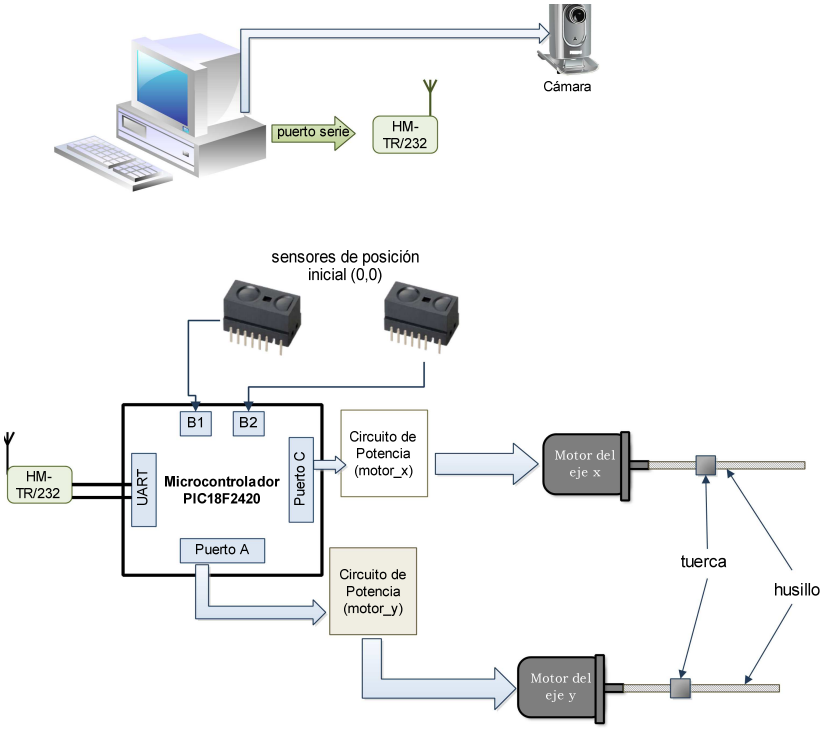


Figura 3.1. Esquema general de control del posicionador.

3.2. Circuito de Control

Unidad de procesamiento

El uso de una unidad de control de movimiento basado en un PLC (*programmable logic controller*, Controlador Lógico Programable) ha sido una constante en la integración de sistemas gantry [6, 7]. El uso de un PLC en aplicaciones industriales es común debido a la facilidad que el mismo ofrece para realizar cambios en la programación. Sin embargo, los requerimientos del posicionador son específicos y no se requiere de una re-programacion constante por parte del usuario final. En consecuencia, se propone usar un microcontrolador para gobernar los movimientos de los actuadores del sistema posicionador.

El microcontrolador empleado fue el PIC18F2420 [8]. Esta elección se basó principalmente en los recursos que proporciona dicho microcontrolador. El Cuadro 3.1 enlista las características generales del PIC18F2420. Dentro de los recursos empleados destacan, el UART (*Universal Asynchronous Reciever Transmitter*) para la comunicación serial con una PC, y el temporizador para establecer la velocidad de los motores de paso.

Recursos	**Valor nominal**	**Utilizado**
Pines E/S	25	13
Memoria Flash	16K	1.8K
Memoria SRAM/EEPROM	768/256	-/-
Temporizadores de 16 bits	3	1
UART/SPI/ I^2C	1/1/1	1/-/
ADC 10 bits	10 canales	-

Cuadro 3.1. Características generales del PIC18F2420.

Etapa de potencia

Los puertos del microcontrolador no disponen de la potencia necesaria para accionar los motores eléctricos de una forma directa. Además, los motores requieren de un cambio de dirección de flujo de corriente a través de sus bobinas (ver Anexo A). Dado lo anterior, es necesario utilizar un puente-H con la capacidad de manejar las corrientes que requieren los motores.

Existen en el mercado una variedad de integrados que incluyen en su interior los puentes-H necesarios para manejar un motor de paso o un motor de corriente continua (CC). El factor distintivo entre tales integrados es la cantidad de corriente que entregan en sus salidas. Los motores de paso que se integran en el sistema posicionador requieren de una corriente de 1.4 A mientras que los motores de CC requieren de una corriente de 300 mA. Con esto en mente, se optó por el integrado L298 que entrega hasta 2 A por canal.

El integrado L298 es capaz de manejar dos motores de CC o un motor de paso. Su diagrama a bloques se muestra en la Figura 3.2. El integrado tiene cuatro entradas de control (INT1, INT2, INT3, INT4) que se utilizan para manejar los puentes A y B. El circuito integrado también cuenta con dos entradas; Enable A y Enable B, que funcionan para habilitar o deshabilitar los puentes independientemente de las señales de entrada. Adicionalmente, el dispositivo cuenta con salidas de sensado(Sense A y Sense B) que permiten medir la corriente que circula por las terminales OUTn. El monitoreo en las terminales de salida es especialmente útil cuando se desea limitar la corriente en las bobinas del motor. Un resumen de la descripción de los pines se muestra en el Cuadro 3.2.

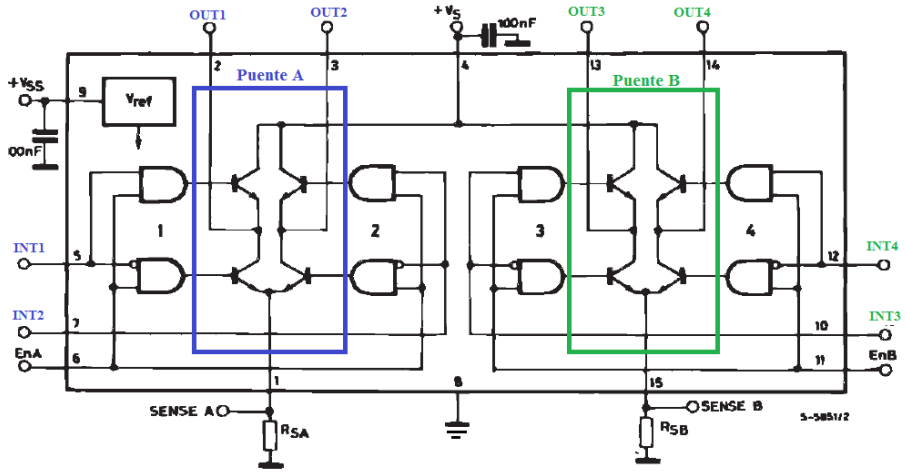


Figura 3.2. Diagrama a bloques de un L298.

Las entradas de control (pines 5,6,7,12,10,11) son compatibles con niveles TTL, lo cual simplifica el control por medio de un microcontrolador. El integrado requiere de una alimentación adicional en VSS para el bloque lógico.

Nombre	**Función**
Sense A:Sense B	Entre este pin y tierra se conecta una resistencia para sensar la corriente que circula por la carga. De lo contrario se debe conectar a tierra
Out 1; Out2	Salidas del Puente A; la corriente que fluye a través de la carga conectada entre estos dos pines es monitoreada en el pin1.
V_S	Fuente de alimentación para las etapas de potencia.
INT1;INT2	Entradas TTL del Puente A
Enable A; Enable B	Entradas TTL; estado bajo deshabilita el puente A(enable A) y/o el puente B (enable B).
VSS	Fuente de voltaje para el bloque lógico
INT3; INT4	Salidas del Puente B. La corriente que fluye a través de la carga conectada entre estos dos pines es monitoreada por el pin 15.

Cuadro 3.2. Descripción de pines del circuito integrado L298.

El circuito de potencia que se empleó para manejar un motor de paso bipolar se muestra en la Figura 3.3. Como señala la hoja de datos, el integrado no cuenta con

diodos de protección por lo que se debe incorporar diodos de recuperación rapida de 2A en las salidas de los puentes. Las entradas INT1 e INT2 manejan la bobina 1 mientras las entradas INT3 e INT4 manejan la bobina 2. Si la entrada INT1 está en alto y INT2 en bajo la corriente fluirá de la termina A hacia la terminal B. Si la entrada INT2 está en alto y INT1 en bajo la corriente fluirá de la terminal B a la terminal. La misma lógica se aplica para el manejo de la bobina 2.

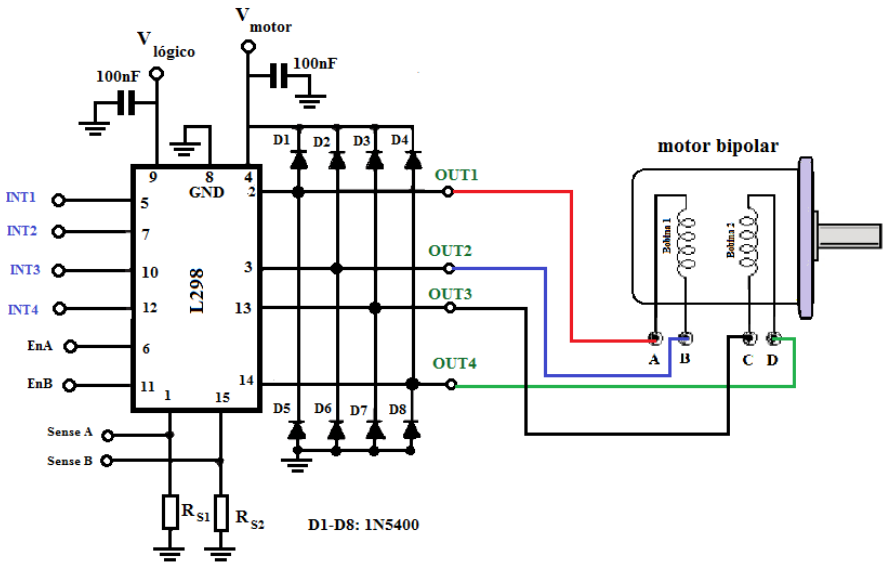


Figura 3.3. Etapa de potencia del circuito de control.

La Figura 3.4 muestra la polaridad que debe tener cada bobina para lograr un avance utilizando el modo paso completo. Para energizar las bobinas de esta manera, se bebe aplicar una secuencia de pulsos en las entradas de los puentes. El cuadro 3.3 indica la secuencia lógica que deben tener las entradas de control (INT1,INT2 INT3 e INT4) para operar en modo paso completo. Para obtener una rotación en sentido horario se debe ejecutar los pasos de forma ascendente. Si se desea un movimiento anti horario se ejecutan los pasos de manera inversa.

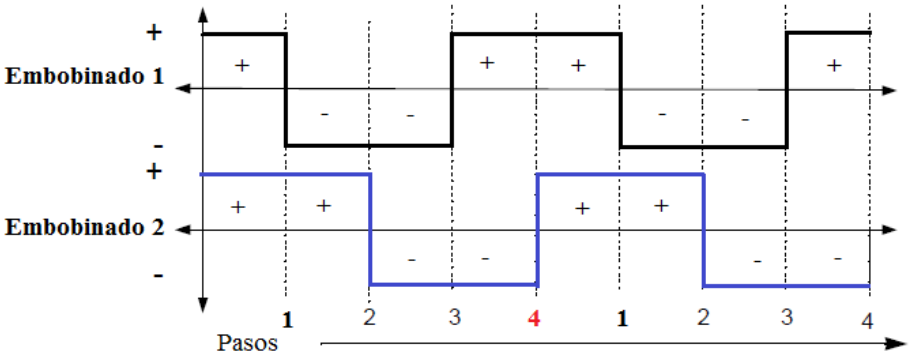


Figura 3.4. Secuencia de encendido de las bobinas.

Paso Completo				
	Bobina 1		Bobina 2	
Paso	INT1	INT2	INT3	INT4
1	0	1	0	1
2	1	0	0	1
3	1	0	1	0
4	0	1	1	0

Cuadro 3.3. Secuencia de pulsos para las entradas de control.

La Figura 3.5 muestra el diseño del circuito de control realizado en ISIS[1] para controlar los motores eléctricos a través de una PC. El circuito de control consiste de un microcontrolador PIC18F2420, dos integrados L298, un módulo inalámbrico, y un convertidor de señales RS232 a señales TTL. El circuito requiere de dos fuentes: una de +5 V @ 500 mA para alimentar etapa lógica y otra de +5 V @ 2A para los motores de paso.

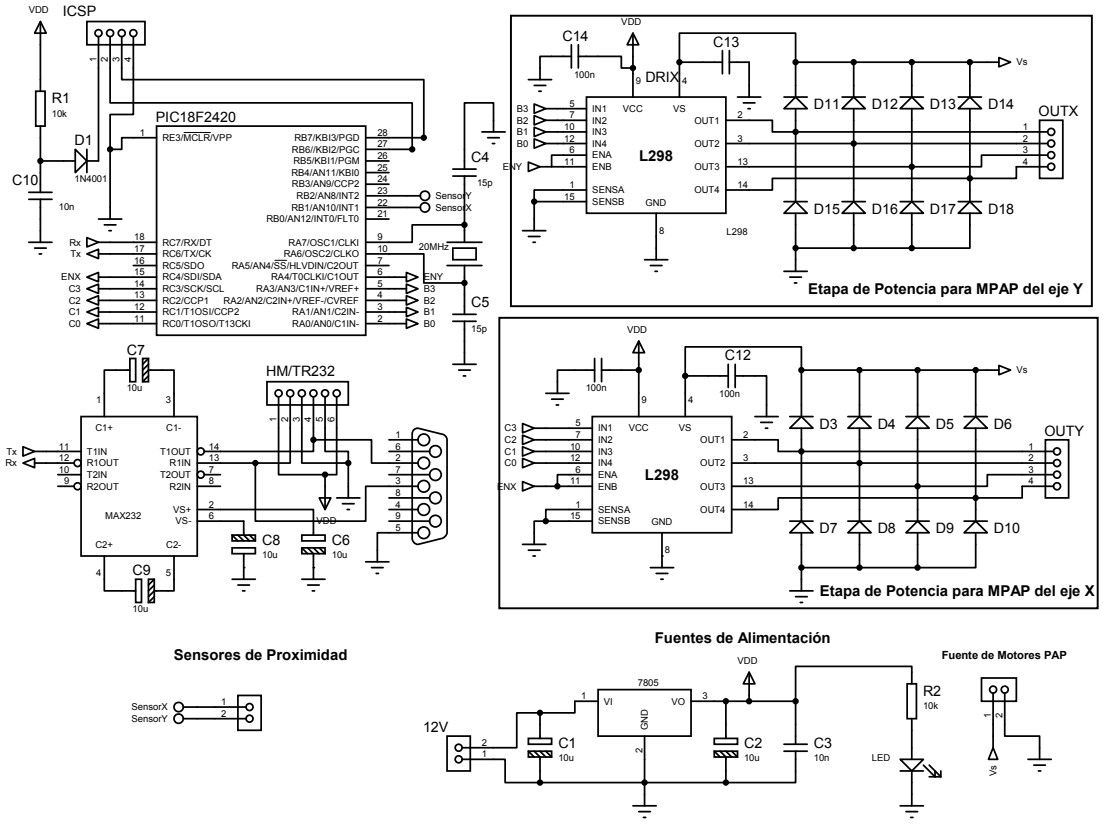


Figura 3.5. Esquemático en ISIS del circuito de control.

[1]ISIS (*Inteligent Schematic Input System*, sistema de enrutado de esquema inteligente) es un programa desarrollado por Labcenter Electronics para diseñar el plano eléctrico del circuito que se desea realizar.

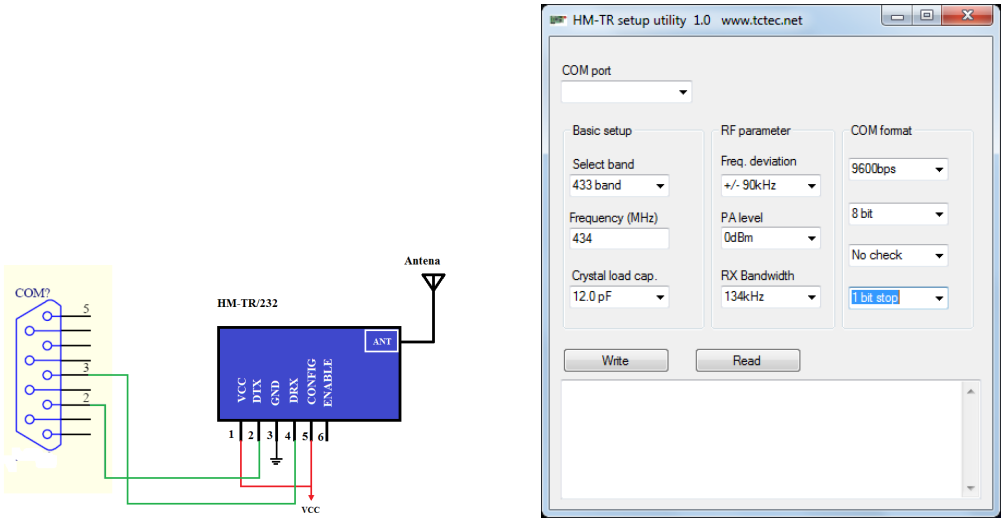
3.3. Transmisión inalámbrica

Se incorporó un par de módulos inalámbricos HM-TR para transmitir datos de una PC al microcontrolador. Los módulo utilizan niveles lógicos RS232 y transmiten a una frecuencia de 433 MHz. Una descripción de los pines se despliega en el Cuadro 3.4.

Pin	Nombre	Descripción
1	VDD	Fuente de alimentación +5V.
2	DTX	Salida de datos del módulo.
3	GND	Tierra
4	DRX	Entrada de datos al módulo.
5	CONFIG	En alto(H) el módulo entra en modo de configuración. En bajo(L) el módulo entra en modo de comunicación.
6	ENABLE	En bajo el módulo entra en modo de hibernación. En alto el módulo esta activo.

Cuadro 3.4. Descripción de pines del HM-TR/232.

Antes de establecer la comunicación entre los dos módulos, se debe configurar cada uno de ellos con los mismo parámetros de trasmisión que los del microcontrolador. Para esto, se conecta el pin 5 en alto y se conecta el puerto serial tal y como lo indica la Figura 3.6a. Una vez conectada a una PC, se configura el módulo con el programa HM-TR Setup.



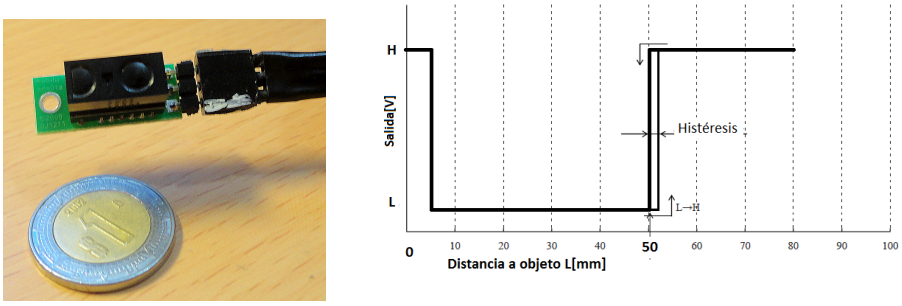
(a) conexión (b) programa

Figura 3.6. Configuración del módulo inalámbrico.

El módulo inalámbrico entra en modo de comunicación cuando el pin 5 se pone en bajo.

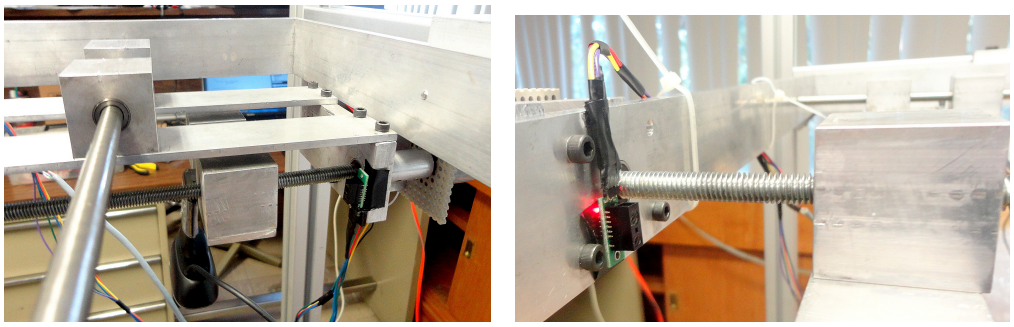
3.4. Sensores de posición inicial

Se colocaron dos sensores GP2Y0D805Z0F de proximidad (Figura 3.7) en dos extremos de la estructura para proporcionarle al posicionador un punto inicial, o bien, un punto de partida (0,0). La Figura 3.8 muestra la cámara en su posición inicial así como la ubicación de los sensores de proximidad dentro del sistema posicionador. El sensor utilizado, compuesto por un diodo infrarrojo emisor y un circuito de procesamiento de señal, detecta objetos a 50mm de distancia. Estos sensores cuentan con salidas digitales las cuales están normalmente en alto. Al detectar un objeto a una distancia de 50 mm su salida cambia de nivel, tal y como se ilustra en la Figura 3.7b



(a) Encapsulado (b) Salida digital

Figura 3.7. Sensor de distancia GP2Y0D805Z0F.



(a) Sensor del eje x. (b) Sensor del eje y.

Figura 3.8. Cámara en posición incial y ubicación de los sensores.

La información proveniente de los sensores es utilizada por el circuito de control para detener los motores al llegar a la posición inicial. Para realizar esta función, las salidas de los sensores se conectan a los pines RB1 y RB2 del PIC dado que el cambio de estado en estos pines genera una interrupción. De esta forma, el PIC responde de forma inmediata ante un cambio de estado en la salida de los sensores.

3.5. Programación del Microcontrolador

La programación del PIC se llevó a cabo a través del compilador C de CCS. El compilador permite realizar un programa en C estándar y dispone de una amplia libreria de funciones y comandos de procesamiento [9].

La Figura 3.9 muestra la secuencia lógica que sigue el PIC para movilizar los motores de paso según las ordenes recibidas por una PC. El PIC comienza configurando los puertos A, donde los pines A0 a A5 se emplean como líneas de control para las entradas del circuito de potencia que maneja el motor del eje x. Así mismo, el puerto C se configura como puerto de salida, donde los pines C0 a C5 se configuran como líneas de control para las entradas del circuito de potencia que maneja el motor del eje y. Posteriormente, espera los pasos por ejecutar en x y y asi como el sentido de giro de los motores. El microcontrolador lleva la cuenta del número de pasos que se han ejecutado y realiza una comparación (al final de cada pulso) de este valor con el número de pasos solicitados. Al finalizar la ejecución de los pasos, el circuito de control detiene los motores y queda en espera de nuevas instrucciones.

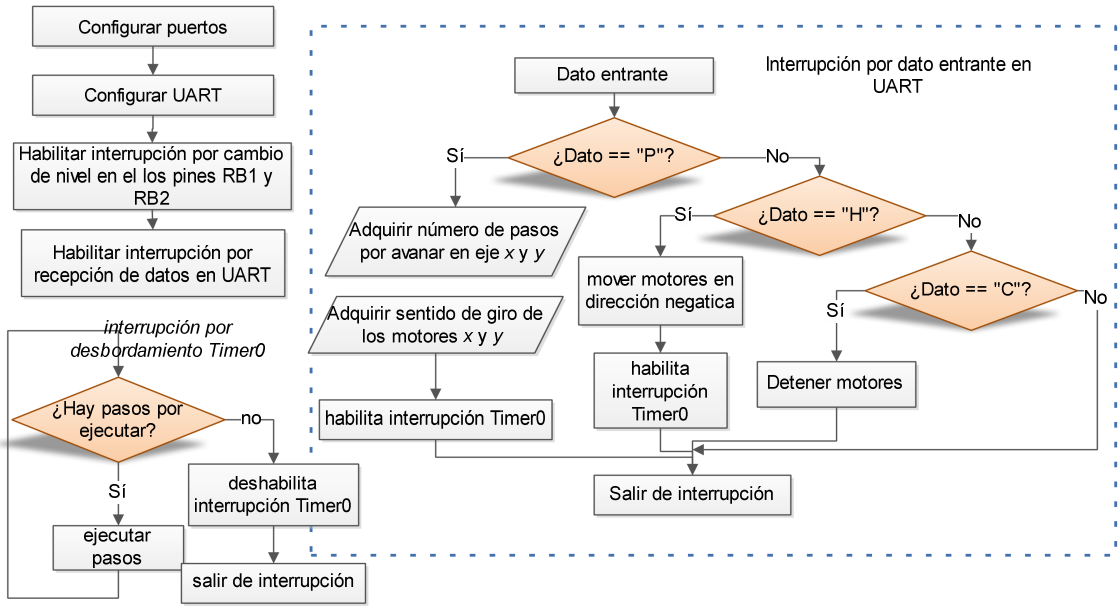


Figura 3.9. Diagrama de flujo del programa principal.

3.5.1. Control de motores de paso

El programa principal del microcontrolador PIC lleva a cabo la tarea de generar la secuencia lógica descrita en el Cuadro 3.3. El tiempo que transcurre entre un paso y otro determina la velocidad del motor. El retardo(δt) se obtiene mediante las ecuaciones 3.1 y 3.2 donde NP y θ_S es el número de pasos por revolución y el ángulo de

paso respectivamente [13].

$$\delta t = \frac{1}{rpm/60 \times NP} \tag{3.1}$$

$$NP = \frac{360}{\theta_S} \tag{3.2}$$

Es posible establecer el tiempo entre cada paso por medio de una rutina de retardo como *delay_ ms*. No obstante, el uso de tales rutinas no garantiza un lapso de tiempo constante entre un paso y otro. Por lo tanto, se decide establecer la velocidad de los motores en base a una interrupción por desbordamiento del temporizador cero (Timer0). Es decir, se ejecuta un paso cada vez que se genera una interrupción ocasionada por el desbordamiento del Timer0. Este lapso de tiempo, se calcula por medio de la ecuación 3.3.

$$\delta t = T_{CM} \times \text{Prescalador} \times (65536 - \text{TMR0}) \tag{3.3}$$

Donde T_{CM} es el ciclo de máquina y es igual a

$$T_{CM} = \frac{4}{F_{OSC}} \tag{3.4}$$

TMR0 representa el valor, de dos bytes, que alcanza el contador para provocar una interrupción por desbordamiento. El contador se incrementa cada ciclo de máquina, es decir, cada 0,2μS (dado un cristal de 20 Mhz). El valor se asocia con una unidad de tiempo despejando TMR0 de la ecuación 3.3.

$$\text{TMR0} = 65556 - \frac{\delta t}{T_{CM} \times \text{Prescaldador}}$$

Dado un ciclo de máquina (T_{CM}) de 0,2μS y un prescalador de 16 se tiene:

$$\text{TMR0} = 65556 - \frac{\delta t}{3{,}2\mu s} \tag{3.5}$$

El código en C que se desarrolló para movilizar los motores de paso se despliega a continuación. Las variables $sPasos\_x$ y $sPasos\_y$ corresponden a las coordenadas enviadas desde una PC través del puerto serial. Los motores avanzan de forma simultanea y el movimiento finaliza cuando se alcanza el número de pasos solicitados. En tal punto, las interrupciones por desbordamiento se deshabilitan para evitar que el microcontrolador ejecute pasos adicionales.

```
//***variables globales**
 unsigned int Paso[4]={0x05,0x09,0x0A,0x06};// secuencia de pasos
 unsigned int i=0,j=0,singo_x,signo_y;
 unsigned int16 nPasos_x,nPasos_y;//contador de pasos
 unsigned int16 sPasos_x,sPasos_y;// pasos a ejecutar

//***interrupción por desbordamiento de Timer0****
#int_TIMER0
void TIMER0_isr(void)
{
  if(nPasos_x>=sPasos_x)
        output_low(PIN_A4); //deshabilita puente A
```

```
      else  output_high(PIN_A4); //habilita puente A
    if(nPasos_y>=sPasos_y)
        output_lowPIN_B4);//deshabilita puente B
      else output_high(PIN_B4); //habilita puente B

  if(signo_x=='+'){ sentido de giro motorX

   if(i==4) // se ejecutó el último paso
     i=0; // regresar a primer paso
   else
      i++;
  }
 else
    {
       if(i==0)
          i=4;
        else
       i--;
  }
if(signo_y=='+'){ sentido de giro de motorY

   if(j==0) //se ejecutó el primer paso
     j=4; // ir a último paso
   else
      j++;
  }
 else
    {
       if(j==0)
          j=4;
        else
       j--;
  }
  output_A(Paso[i]); // Avanzar o retroceder un paso en X
  output_B(Paso[ij); // Avanzar o retroceder un paso en Y
  nPasos_x++;
  nPasos_y++;
 if(nPasos_x>=sPasos_x && nPasos_y>=sPasos_y)
   disable_interrupts(INT_TIMER0);// deshabilita interrupciónes de Timer0

}

void main() {
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_16);// prescalador de 16 oscilador interno
set_timer0(58600); // se indica el valor de TMR0
}
```

3.5.2. Comunicación serial

Se empleó el módulo de comunicación serie UART para comunicar el PIC con una PC. La función del módulo UART es transmitir o recibir un conjunto de bits en serie. La transmisión serie consiste en enviar los datos bit a bit a través de una línea común en periodos de tiempo fijos. El tiempo transcurrido entre cada bit establece la velocidad de transmisión, o bien, el número de bits enviados por segundo (baudios). El registro SPBRG (*UART Baud Rate Generator Register*) determina la tasa de baudios y está dada por la ecuación 3.6, donde f_{OSC} es la frecuencia de reloj.

$$\text{BAUD} = \frac{f_{OSC}}{16(UBRR+1)} \tag{3.6}$$

El cristal que se utilizó para establecer la frecuencia de reloj es de 20 MHz y la

velocidad deseada es de 9600 baudios, por lo tanto el UBRR es de:

$$\text{UBRR} = \frac{f_{OSC}}{16 \times UBRR} - 1 = \frac{20Mhz}{9600 \times 16} - 1 = 130{,}208$$

El UART se configuró de la siguiente manera:

- Bits por segundo(baudios) : **9600**
- Paridad: **Ninguna**
- Bits de datos: **1**
- bits de parada: **1**

Se habilitaron interrupciones por dato recibido para que el programa responda de manera inmediata a los datos entrantes. Se empleó el siguiente código utilizando el compilador CCS:

```
#include <18F2420.h> // microcontrolador que se usa
#use delay(clock=20000000) // se usa un cristal de 20 MHz
#FUSES HS,NOWDT // cristal de Alta velocidad ,  Sin perro guardian
#use standard_io(A)
// Configuración del usart
#use rs232(baud=9600, xmit=pin_c6, rcv=pin_c7, bits=8, parity=N)

#int_rda // interrupción por dato recibido
void serial_isr() {
int dato;
 dato=getc();
   if(dato=='P') // ¿Los datos a recibir son de posición?
    {
  // obtener coordenadas
 enable_interrupts(INT_TIMER0);// habilita interrupción por desbordamiento (timer0)
    }
    else if(dato=='D') // ¿Cancelar instrucción?
       {
   //detener motores
       }
  else if( dato=='H') ¿Ir a Posición (0,0)?
       {
  // Girar motores en reversa hasta llegar a la posición inicial
       }
void main() {

enable_interrupts(int_rda); // Habilita interrupciones por dato entrante
enable_interrupts(global);     //Habilita interrupción global

  while(1){

  }
}
```

Capítulo 4

Programación

El enlace entre el circuito de control y la PC, se lleva a cabo por medio de una interfaz gráfica de usuario (GUI, *graphical user interface*) desarrollado en Visual Studio 2010. El GUI se diseñó utilizando los formularios Windows Form que asemejan el aspecto y comportamiento de Microsoft Windows [10, 11]. La conectividad entre la PC y el circuito de control es realizada bajo la interfaz serial RS-232. El programa desarrollado para el GUI se encarga de establecer las coordenadas del sistema, monitorear el circuito de control y realizar la adquisición y calibración de la cámara.

4.1. Ventana principal del GUI

La integración de la ventana principal del GUI se realizó mediante las herramientas que proporciona Visual Studio. El programa que rigue el comportamiento de dicha ventana se escribió utilizando el lenguaje C sharp (abreviado C# y pronunciado "Se Sharp"). El Cuadro 4.1 describe la función de los elementos que conforman el GUI.

	Control	**Nombre**	**Función**
1	button	btnEnviar	Envía las coordenadas por el puerto serial
2	button	btnDetener	Detiene la instrucción que se esta ejecutando
3	button	btnHome	Posicionador regresa a (0,0)
4	button	btnIniciar	Incia \ Detiene la cámara
5	button	btnCapturar	Permite capturar una imagen
6	ComboBox	cmbDispositivos	Enlistan los dispositivos de video
7	ComboBox	cmbZona	Cuadro que enlista las zonas de trabajo
8	textbox	txbCoordenadas	Espacio donde se ingresan las coordenadas
9	textbox	txbPosicion_actual	Indica la posición actual del sistema
10	ProgressBar	ProgressBar1	Indica el progreso de una instruccón
11	toolstrip	toolstripMenu	La barra de menú del programa
12	statusStrip1	lblEstado	Muestra información referente al estado general del sistema
13	label	lblPuntosReferencia	Muestra las coordenadas en pixeles de los puntos de referencia

Cuadro 4.1. Descripción de componentes que se utilizaron en el formulario principal.

La Figura 4.1 muestra la ventana (o formulario) principal del sistema posicionador donde el usuario ingresa las coordenadas del sistema y visualiza las imágenes adquiridas por la cámara web. La ventana cuenta con una barra de menú donde se modifican las propiedades de la cámara y configura el puerto serial.

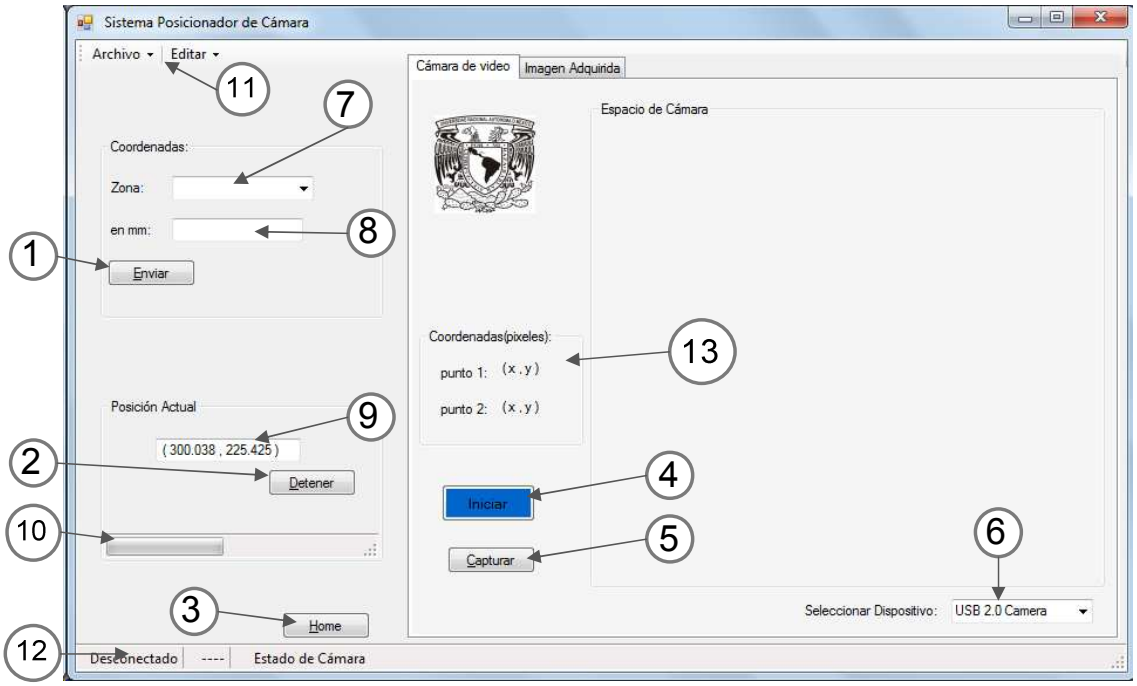


Figura 4.1. Ventada principal del GUI.

4.2. Configuración del puerto serial

El usuario tiene la opción de configurar el puerto serial seleccionando "Configurar Puerto" del menú Editar. La ventana que aparece se muestra en la Figura 4.2. El programa busca los puertos disponibles y los despliega en el primer comboBox. El usuario debe seleccionar el puerto y los bits por segundo. El código en C# para configurar el puerto serial es el siguiente:
formulario1 (ventana principal):

```
   //************Se selecionó Configurar puerto del menú archivo************
   //       ********  formulario principal***********
 private void configurarPuertoToolStripMenuItem_Click(object sender, EventArgs e)
  {

DialogResult D = new DialogResult(); //D es el nombre del Resultado del diálogo
Form4 config_puertoSerial = new Form4(); // Formulario4 :config_puertoSerial
D = config_puertoSerial.ShowDialog(); //
 if (D == DialogResult.OK)  // ¿Se oprimió el botón "OK" de la venta Configurar Puerto?
    {
          //**SI**
       COMX = Form4.COMx;  // Toma el nombre del puerto seleccionado en la ventana Configurar Puerto
       Int32 Baudio = Convert.ToInt32(Form4.Baud); // convierte string en int
       serialPort1.PortName = COMX;
       serialPort1.BaudRate = Baudio; // Introduce los bits por segundo establecidos en el formulario
       serialPort1.StopBits = StopBits.One; // bits de parada 1
```

```
        serialPort1.Parity = Parity.None; // paridad: Ninguna
        serialPort1.Open();  // Abre el puerto
        lblEstadoGlobal.Text = COMX + " esta abierto"; // Barra de estado indica que esta abierto el puerto
        conectarToolStripMenuItem.Text = "Desconectar";
      }
  }
```

ventana para configurar puerto (formulario 4):

```
namespace Sistema_Posicionador
{
 public partial class Form4 : Form
   {
     public static string COMx; // variable global
     public static string Baud; // variable global
     public Form4()
      {
         InitializeComponent();
         foreach (string s in SerialPort.GetPortNames()) // Buscar los puertos disponibles
           {
                comboBox1.Items.Add(s); // anexar los puertos en comboBox1
           }
        }
  //*************** Se oprimió el botón Aceptar*******************
        private void btnAceptar_Click(object sender, EventArgs e)
        {
            Baud = cmbBaud.Text;
            COMx = comboBox1.Text;
        }
    }
}
```

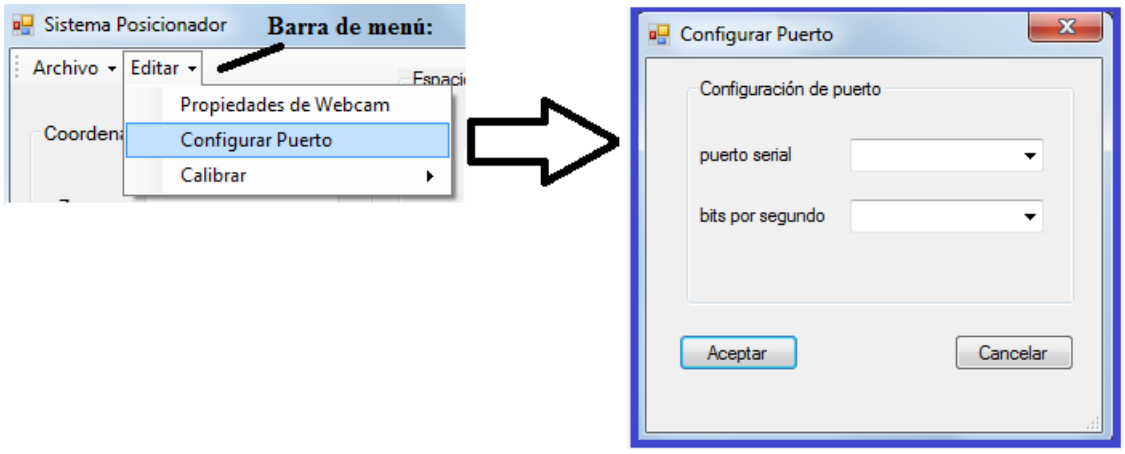


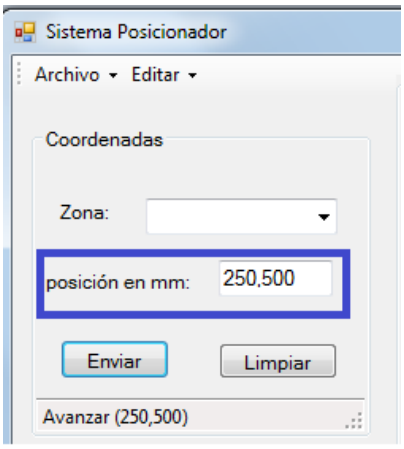
Figura 4.2. Ventana para configurar el puerto serial.

Al oprimir el botón “Aceptar”, el puerto elejido queda configurado y se podrán transmitir o recibir datos. Si el usuario oprime un botón antes de haber establecido el puerto, el programa envía una advertencia en forma de cuadro de diálogo.

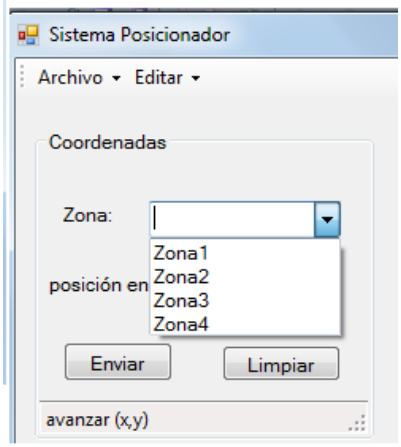
4.3. Modo de introducir coordenadas

Para posicionar la cámara de video dentro del área de trabajo se ingresan las coordenadas directamente en el espacio en blanco o se elige una de las cuatro zonas enlistadas en el submenú (ver Figura 4.3a). Al ingresar las coordenadas manualmente,

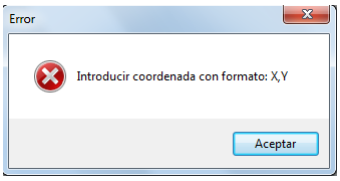
se utiliza una coma entre la coordenada en x y la coordenada en y (por ejemplo x,y). Para enviar las coordenadas solicitadas a través del puerto serial, se debe oprimir el botón "Enviar". Si el usuario ingresa una letra en el espacio de coordenadas o no proporciona el formato requerido, el programa desplegará el cuadro de diálogo mostrado en la Figuraa 4.3c.



(a) por coordenadas



(b) por zona



(c) error en el formato

Figura 4.3. Modos de posicionar la cámara.

Tras oprimir el botón enviar, el programa realiza una comparación entre la posición actual y la solicitada para determinar el avance en x y y. Si la posición solicitada en x es mayor que la posición actual en x, la cámara tendrá un avance positivo. Si la posición solicitad es menor que la actual, la cámara tendrá un avance negativo. La misma lógica aplica para determinar el avance en el eje y. La Figura 4.4 muestra el proceso que sigue el programa para determinar el avance en los ejes x y y.

La ventana principal cuenta con un botón "HOME", el cual lleva a la cámara web a la posición (0,0). Al oprimir este botón, se envía el caracter en código ASCII "H" por el puerto serial indicando al circuito de control que gire los motores en sentido negativo hasta llegar a dicha posición. Un cuadro de diálogo aparecerá en pantalla cuando el circuito de control haya confirmando la llegada del posicionador.

Si por alguna razón se desea cancelar una instrucción, se debe oprimir el botón "Detener". En ese instante el circuito de control recibirá la orden de parar los motores. Un cuadro de diálogo aparecerá en pantalla confirmado esta instrucción.

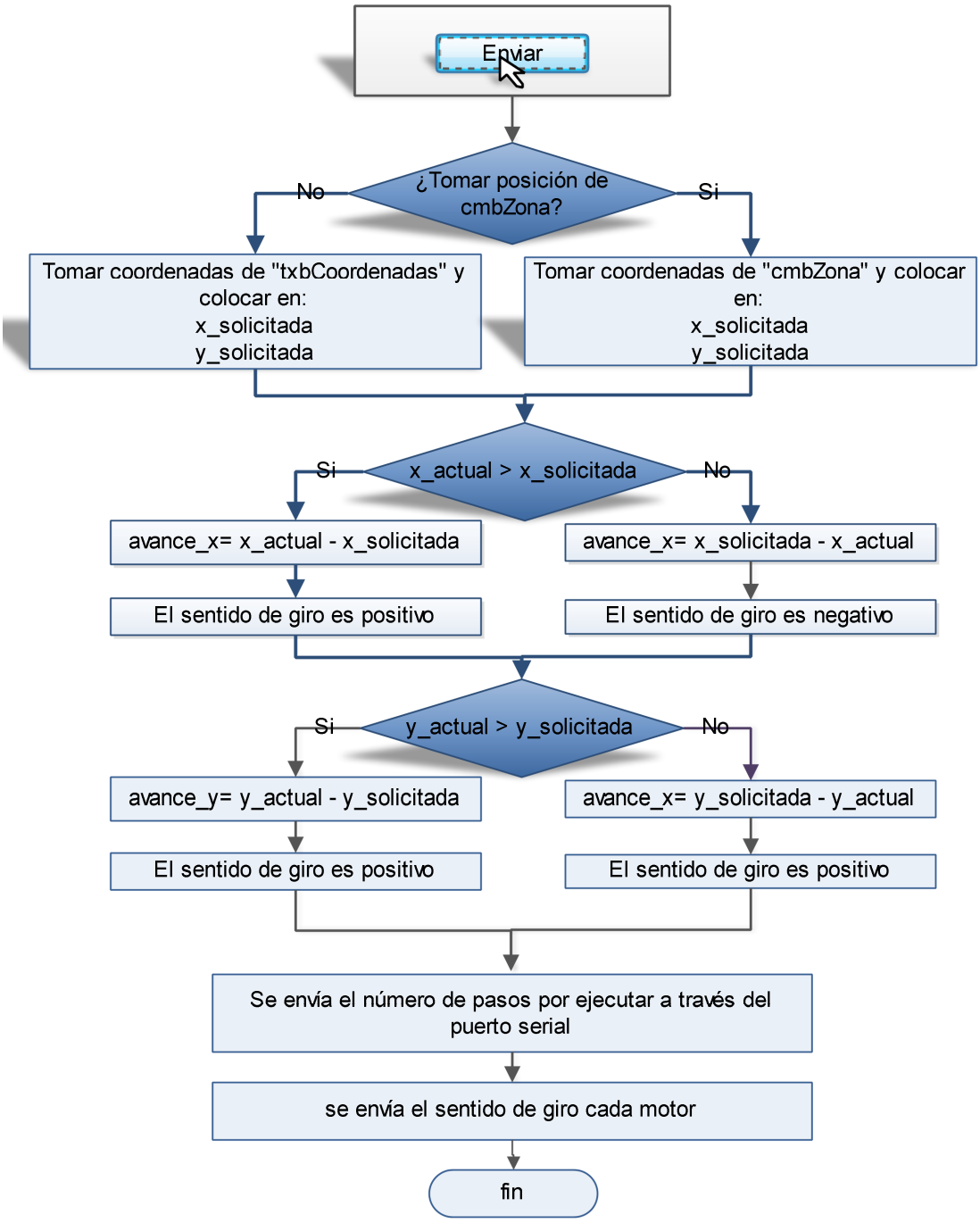


Figura 4.4. Diagrama de flujo del proceso de envío de coordenadas.

4.4. Monitoreo de la posición de cámara

El usuario monitorea la posición actual del sistema en la parte inferior de la ventana principal. El circuito de control envía un caracter por el puerto serial cada vez que el eje de uno de los dos motores complete una revolución. Dado que una revolución completa del eje del motor equivale a un avance lineal de 1.58 mm, se determina la posición de la cámara contando el número de revoluciones que da cada motor. El diagrama de la Figura 4.5 muestra la forma en la cual el programa interpreta los datos que recibe del puerto serial.

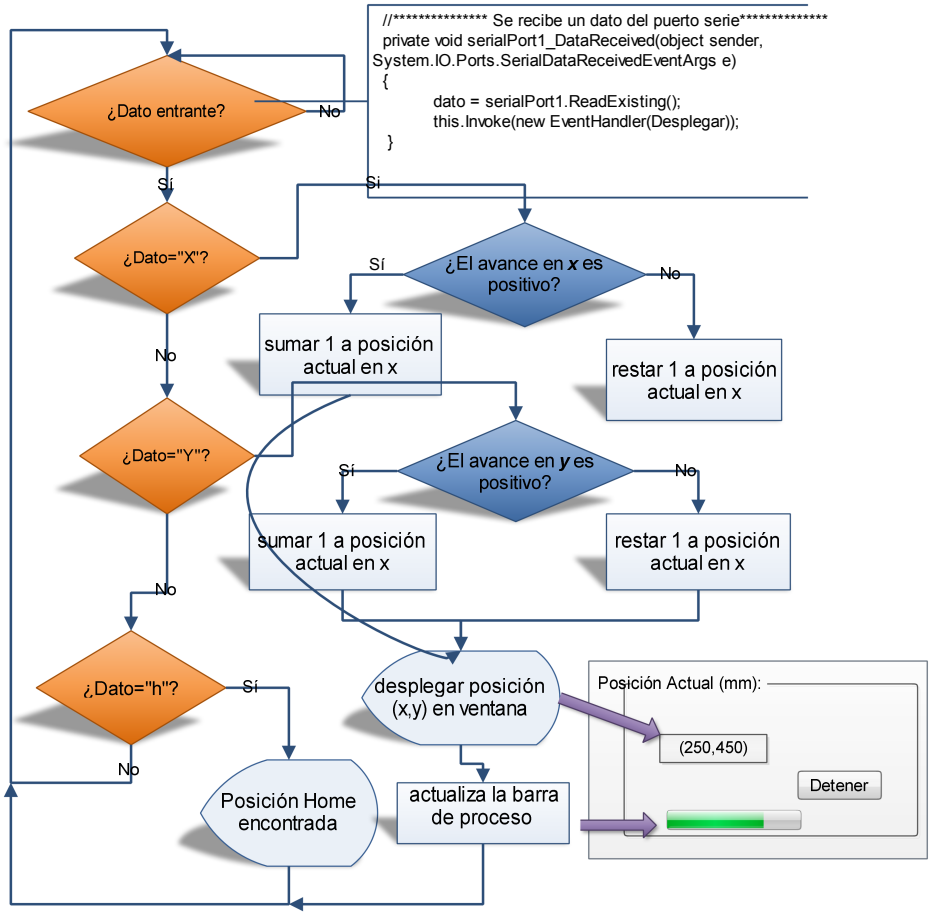


Figura 4.5. Proceso de recepción de datos.

Es posible guardar la posición actual del sistema al cerrar la aplicacion. Para ello, el asuario debe seleccionar guardar-posición del menú archivo. Al abrir el GUI nuevamente, el programa carga las variables de posición y las despliega en la ventana principal.

4.5. Configuración de la cámara

El programa accede a una cámara conectada a un puerto USB por medio de unas librerias encontradas en AForge.Net. AForge.Net es un marco de trabajo de código abierto para C# diseñado para desarrollo e investigación en los campos de visión artificial, inteligencia artificial, procesamiento digital de imágenes, redes neuronales, algoritmos genéticos y otras áreas afines. AForge.Net esta disponible en la siguiente página:

`http://code.google.com/p/aforge/downloads/list`

El programa comienza buscando los dispositivos de video conectados al equipo y los enlista en submenú 6 (ver Figura 4.1). El código en C# se muestra a continuación:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.IO.Ports;
using System.Windows.Forms;
//*****Librerias AForge.net**********************
using AForge.Video;
using AForge.Video.DirectShow;
using AForge;
using AForge.Imaging;
using AForge.Imaging.Filters;
using AForge.Vision;
using AForge.Math;


namespace Sistema_Posicionador
{
    public partial class frmPrincipal : Form
    {

     private bool ExistenDispositivos = false;
   // Almacena todos los dispositivos disponibles
    private FilterInfoCollection DispositivosDeVideo;
    private VideoCaptureDevice FuenteDeVideo = null; // cámara que se va usar
    private bool capturar = false; // Variable tipo boolena. Le indica al
   // programam cuando se solicita una captura
    private string nombreArchivo; // Nombre del la imagen dado por el usuario
    private string direccion;// Lugar donde se guarda la imagen
    int cont_imagen=0; // Contador de Imagenes

    int posicion_actual_x; /* posicion del sistema en revoluciones
    int posicion_actual_y;   del motor         */
    decimal Posicion_real_x; // Posicion del sistema en milimetros
    decimal Posicion_real_y;
    string dato; // almacena el dato recibido por el puerto serial
    string sentido_x;
    string sentido_y;
    const Decimal pasoDeRrosca = 1.5875m;//variable constante
   ProcesarImagen procesar = new ProcesarImagen(); // delcara un objeto de la clase ProcesarImagen

 public frmPrincipal()
        {
            InitializeComponent();
            BuscarDispositivos();    // ir a función para busca dispositivos
                                        conectados al equipo
        }
// Esta funcion se encarga de enlistar todos los dispositivos conectados
 public void CargarDispositivos(FilterInfoCollection Dispositivos)
        {
            for (int i = 0; i < Dispositivos.Count; i++)
                cmbDispositivos.Items.Add(Dispositivos[i].Name.ToString());
```

```
        // carga el Disposito a comboBox (cmbDispositivos)
            cmbDispositivos.Text = cmbDispositivos.Items[0].ToString();
        }
// Funcion que permite buscar Dispositivos de Video
        public void BuscarDispositivos()
        {
            DispositivosDeVideo = new FilterInfoCollection(FilterCategory.VideoInputDevice); // Filtra para que solo apar
            if (DispositivosDeVideo.Count == 0) // ¿Existen Dispositivos?
                ExistenDispositivos = false;  //no
            else
            {
                ExistenDispositivos = true;
                CargarDispositivos(DispositivosDeVideo);
            }
        }

        public void TerminarFuenteDeVideo()// detener la cámara
        {
                    FuenteDeVideo.SignalToStop();
                    FuenteDeVideo = null;
                }
        }
        // *****nuevo cuadro(New Frame) entrante************
  private void video_NuevoFrame(object sender, NewFrameEventArgs eventArgs)
      {
            if (capturar == false)
          {
   // ********plasmar la imagen en el espacio de cámara***************
              Bitmap Imagen = (Bitmap)eventArgs.Frame.Clone();
              picEspacioCam.Image = Imagen;
          }
          else
        {
 //se solicita capturar imagen. Plasmar el siguiente cuadro en "picImagenAdquirida".
          Bitmap Imagen = (Bitmap)eventArgs.Frame.Clone();
          picImagenAdquirida.Image = Imagen;
          direccion = nombreArchivo + cont_imagen + ".Jpeg";
          picImagenAdquirida.Image.Save(direccion); // Guardar imagne
          cont_imagen++;
          capturar = false;
          PuntosDeReferencia();//  calcular puntos de ref
             }

        }
```

El usuario debe seleccionar la cámara de video a utilizar (si más de dos se detectan en el equipo) y posteriormente edita la resolución y cuadros por minuto de la misma. Para esto, el usuario debe seleccionar "propiedades de cámara" en el menú editar donde aparecerá la ventana mostrada en la Figura 4.6. Una vez realizado los cambios requeridos, se enciende la cámara oprimiendo el botón Iniciar.

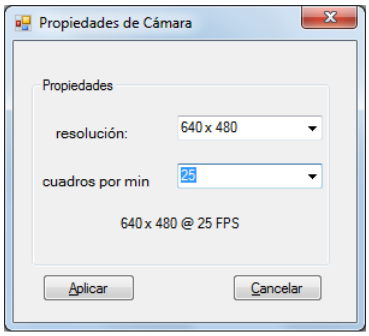


Figura 4.6. Editar propiedades de cámara

4.5.1. Adquisición de imágenes

El posicionador adquiere imágenes en formato JPG con una resolución de 640×480 (o la establecida en la ventana "Propiedades de cámara"). El modo de adquisición de imágenes se realiza en dos formas: de manera automática y de manera manual. Se realiza una captura automática cuando la cámara se haya situado por primera vez en una de las cuatro zonas de trabajo. El usuario puede capturar una imagen en cualquier momento orpimiendo el botón "capturar "de la ventana principal del GUI. Para ello, se debe especificar la carpeta en la cual se guardarán las imágenes seleccionando "Guardar Imágenes" del menú Archivo. El cuadro de diálogo que aparece se muestra en la Figura 4.7.

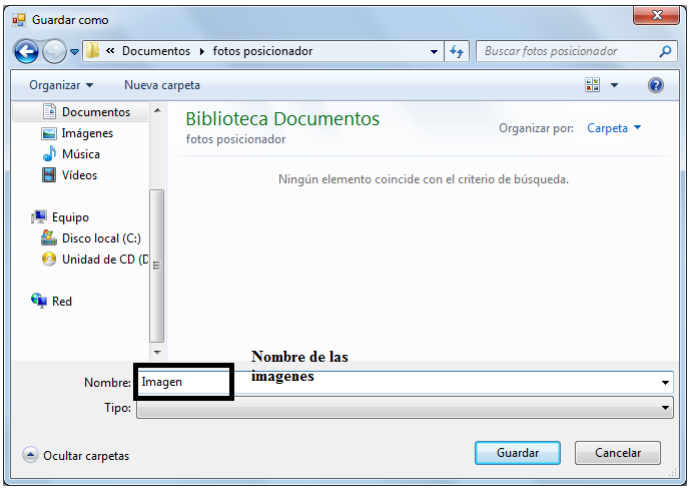


Figura 4.7. Guardar las imágenes adquiridas.

Lo único que debe proporcionar el usuario es el nombre del archivo, ya que se auto numeran las imágenes. Por ejemplo: si se teclea *Imagen* en el espacio "Nombre:", y se toman cuatro fotos, la carpeta seleccionada tendrá cuatro imágenes como muestra la Figura 4.8.

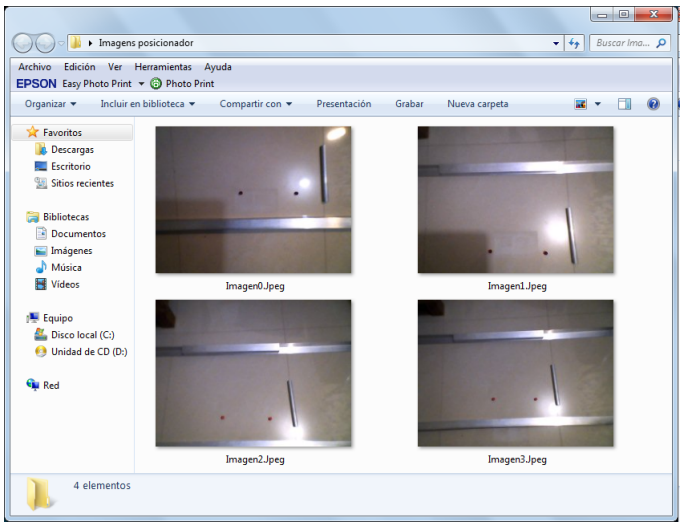


Figura 4.8. Numeración de imágenes adquiridas.

4.6. Corrección de error mediante imágenes

El programa utiliza las imágenes adquiridas por la cámara para calcular el error que obtiene el posicionador al situarse en un lugar específico dentro de una de las cuatro zonas de trabajo. El cálculo de error y la corrección del mismo se basa en la localización de dos puntos de referencia preestablecidos en cada zona de trabajo. La imagen adquirida por la cámara se manipula por medio de técnicas de procesamiento para obtener las coordenadas en pixeles de tales puntos. Para realizar esta tarea, se hace uso de las rutinas disponibles en las librerias de Aforge.net. El proceso y las técnicas empleadas para ubicar los puntos de referencia se muestra en la Figura 4.9.

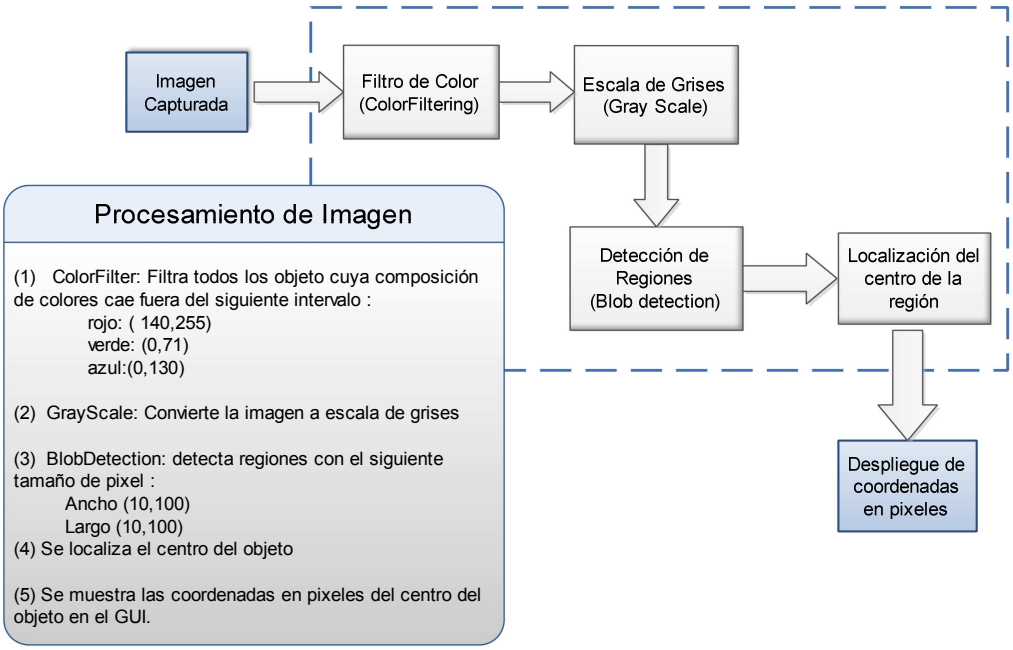


Figura 4.9. Acondicionamiento de imagen y localización de puntos de referencia.

Lo primero que realiza el programa después de haber adquirido una imagen es aplicar un filtro de color para "eliminar" todos los colores fuera de un intervalo de colores RGB establecido. Dado que el color de los puntos de referencia es rojo y tomando en cuenta que la codificación de los colores primarios es de un byte, se utiliza el intervalo de (170, 250) para el color rojo. Los intervalos para los colores verde y azul se fija realizando un barrido del valor del límite superior de cada intervalo. La Figura 4.10 presenta la ventana desarrollada en visual studio para calcular el intervalo de colores RGB.

Posteriormente, la imagen se convierte a escala de grises y se aplica una técnica de procesamiento conocida como *Blob detection* para detectar puntos o regiones más claras o más oscuras en la imagen. La rutina denominada *BlobCounter*, cuenta los objetos en la imagen que están separados por un fondo negro. Existe la posibilidad de que el programa detecte más de dos objetos en la imagen si es que en la rutina se hallan

Figura 4.10. Ajuste de intervalo de colores RGB.

otros elementos que posean la misma tonalidad que los puntos de referencia. Para evitar este suceso, se filtran los objetos cuya dimensión sea menor que 35×35 pixeles. Una vez que la rutina *BlobCounter* haya ubicado todos los elementos separados por un fondo negro, se seleccionan los dos objetos con mayor área. Posteriormente se obtienen las coordenadas en pixeles del centroide de los dos puntos de referencia (Figura 4.11).

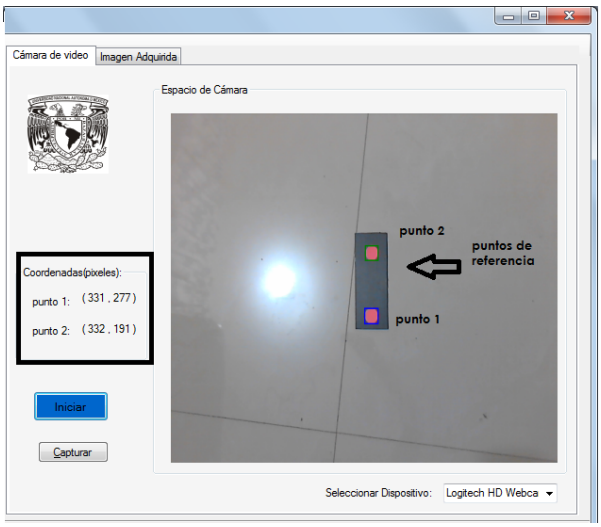


Figura 4.11. Localización de los puntos de referencia.

El programa descrito en esta sección se integró dentro del código principal en forma de una función que se define fuera del mismo. Esta función recibe una imagen y devuelve las coordenadas en pixeles de los puntos de referencia (ver Anexo B).

4.6.1. Método de calibración

El método de calibración utilizado para obtener las coordenadas de la imagen (en pixeles) con las coordenadas de mundo real (en escena real de trabajo), se resume en el diagrama que se presenta en la Figura 4.12. Los puntos de referencia son círculos de papel rojo de 6 mm de diámetro y tienen una separación constante de 10 cm. Teniendo las coordenadas en pixeles de los puntos P1(x_1, y_1) y P2(x_2, y_2) se calcula la distancia entre estos dos mediante la ecuación 4.1.

$$d(P1, P2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.1}$$

Conociendo la distancia $d(P1, P2)$ se establece una relación entre pixeles y unidades métricas. Por ejemplo, si la distancia en pixeles es de 100, se dice que 1 cm equivale a 10 pixeles.

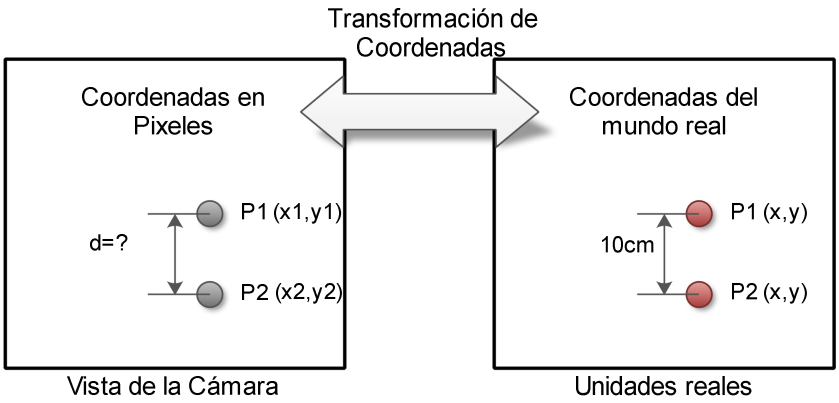


Figura 4.12. Método de calibración de cámara.

El método que se emplea para la calibración estática en el sistema es el siguiente:

- Se establece un sistema de coordenadas en 2D con dos puntos proyectados en el área de trabajo, los puntos de referencia son "puntos rojos" posicionados en las diferentes zonas de la celda,
- Se establece la localización de los 2 puntos dentro de la imagen,
- Se asocian las posiciones de los 2 puntos en la imagen con medidas de las coordenadas del mundo real,
- Se obtienen las transformaciones de medidas en pixeles a unidades reales,
- Cada vez que se cambie de localidad en la cámara, se recalibra el sistema si se cambia la distancia cámara-objeto.

4.6.2. Retroalimentación de error

Los puntos de referencia preestablecidos en cada zona de la celda tienen una coordenada en pixeles inicial. Esta coordenada indica el punto exacto que debe encontrar el posicionador, por medio de las imágenes adquiridas, para que el error sea nulo.

Cuando el sistema posicionador se ubica en una zona, el programa realiza la calibración de la cámara e identifica los puntos de referencia de esa zona. Si las coordenadas de los puntos obtenidos no coinciden con las coordenadas iniciales, se sabe que el posicionador tuvo un error de posicionamiento. La retroalimentación del error se efectúa restando las coordenadas en pixeles del punto 1 inicial ($P1_{inicial}$) con las coordenadas en pixeles del punto 1 obtenido ($P1_{real}$) a través de la imagen adquirida. La diferencia entre estos dos puntos representa el tamaño del error. Sin embargo, este valor debe convertirse a una dimensión métrica a través del método descrito en la sección 4.6.1.

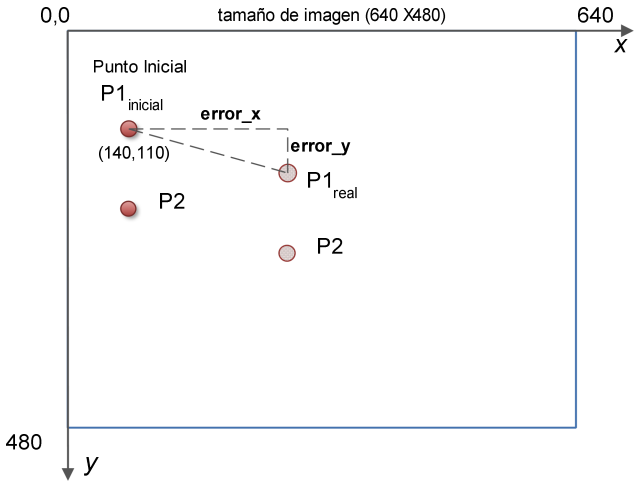


Figura 4.13. Error de posicionamiento.

La Figura 4.13 muestra un ejemplo del error que puede tener la cámara al situarse en la zona 1. El punto inicial de la zona 1 corresponde a la coordenada en pixeles ($P1_{inicial}$)(140,110). Si el punto real obtenido es de ($P1_{real}$)(144,120), entonces el error en pixeles es

$$Error\_x = 144 - 140 = 4$$
$$Error\_y = 115 - 110 = 5$$

Si se estableció previamente que 100 mm = 200 pixeles, entonces el error en milímetros que el sistema posicionador debe corregir en cada eje es

$$Error\_x = \frac{4 \times 100}{200} = 2mm$$
$$Error\_y = \frac{5 \times 100}{200} = 2{,}5mm$$

Este error se convierte en un número de pasos a ejecutar y se envía al circuito de control por medio del puerto serial.

Capítulo 5

Pruebas y Resultados

5.1. Prueba con motores de paso

Para demostrar la funcionalidad del sistema posicionador de cámara, se realizaron pruebas de movimiento con motores de paso bipolares de 5.1V@1.4A. En la Figura 5.2 se muestra el acoplamiento del motor de paso al tornillo del mecanismo de transformación. Las mediciones se llevaron a cabo utilizando el vernier digital mostrado en la Figura 5.1a. Sus características se presentan en el Cuadro 5.1b.

(a) Aspecto

Intervalo de medición	Resolución	Exactitud
0-155mm/0-6"	0.01mm/0.0005"	0.03mm/0.0015"

(b) Características

Figura 5.1. Vernier Digital.

Como el ángulo de paso de este motor es de 1.8 °se sabe, por la ecuación 3.2, que el número de pasos por revolución es de

$$\frac{360}{1{,}8^\circ} = 200$$

El tornillo que se intregró en el mecanismo de transformación tiene un paso de rosca de 1.587 mm. De la ecuación 1.1 se sabe que el avance en la tuerca producida por una vuelta completa del tornillo deberá ser de 1.587 mm. Por lo tanto, 200 pasos del motor de paso corresponde a un avance de 1.587 mm.

5.1.1. Pruebas de desplazamiento

Con la finalidad de comprobar el desplazamiento anteriormente planteado, se hizo que los motores de paso avanzarán una serie de pasos a la vez. Las mediciones se

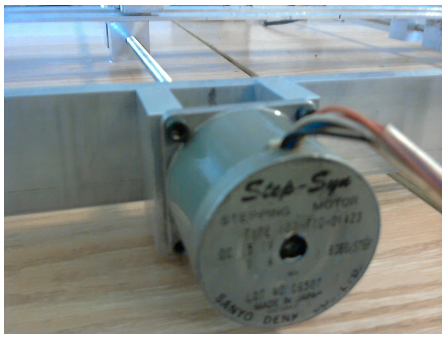


Figura 5.2. Acoplamiento del motor de paso bipolar.

llevaron a cabo midiendo la distancia de un punto del borde del posicionador a un punto del bloque con rosca. El Cuadro 5.1 muestra un resumen del desplazamiento obtenido en el eje x dado un determinado número de revoluciones. La columna designada "teórico"representa la posición que se espera obtener, mientras que la columna con nombre "práctico"representa el valor de la medición obtenida para un determinado número de revoluciones. Los dos valores se comparan para obtener el porcentaje de error de posicionamiento. De forma similar, se midió el desplazamiento de la cámara en el eje y para un determinado número de revoluciones del motor (ver Cuadro 5.2). La Figura A.1 muestra la representación gráfica de los movimientos obtenidos en función del número de revoluciones del eje del motor.

	Avance en el eje x		**Error**	
Revoluciones	Práctico	Teórico	mm	porcentaje
1	1.54	1.5875	0.0475	2.992
2	2.951	3.175	0.224	7.055
3	4.467	4.7625	0.2955	6.205
4	5.883	6.35	0.467	6.845
5	7.65	7.9375	0.2875	3.622
6	8.873	9.525	0.652	6.845
7	10.84	11.1125	0.2715	2.452
8	12.0406	12.7	0.294	2.315
9	13.46	14.2875	0.8275	5.792
10	15.21	15.875	0.665	4.189
11	16.607	17.4625	0.8555	4.899
12	18.3167	19.05	0.7333	3.849
13	20.233	20.6375	0.4045	1.96
14	21.663	22.225	0.562	2.529
15	22.9167	23.8125	0.8958	3.762
16	24.446	25.4	0.954	3.337
17	26.087	26.9875	0.9005	3.337
18	27.76	28.575	0.815	2.852
19	29.277	30.1625	0.8855	2.936

Cuadro 5.1. Desplazamiento en el eje x con un motor de paso de $\theta = 1{,}8°$.

	Avance en el eje x		**Error**	
Revoluciones	Práctico	Teórico	mm	porcentaje
1	1.54	1.5875	0.0475	2.992
2	3.063	3.175	0.112	3.528
3	4.69	4.7625	0.0725	1.522
4	5.883	6.35	0.467	7.354
5	7.763	7.9375	0.1745	2.198
6	9.303	9.525	0.222	2.331
7	10.89	11.1125	0.2225	2.002
8	12.407	12.7	0.293	2.307
9	13.647	14.2875	0.6405	4.483
10	15.213	15.875	0.662	4.17
11	17.5133	17.4625	0.0508	0.291
12	18.3167	19.05	0.7333	3.849
13	20.677	20.6375	0.0395	0.191
14	21.632	22.225	0.592	2.664
15	22.9167	23.8125	0.8958	3.762
16	24.8733	25.4	0.5267	2.074
17	26.46	26.9875	0.5275	1.955
18	28.411	28.575	0.164	0.574
19	30.173	30.1625	0.0105	0.035

Cuadro 5.2. Desplazamiento en el eje y con un motor de paso de $\theta = 1{,}8°$.

(a) eje x

(b) eje y

Figura 5.3. Gráfica de desplazamiento en los ejes.

5.1.2. Prueba de velocidad

Para identificar la velocidad máxima de cada motor, se realizó un barrido en la cual se varió el retardo del pulso. Mediante la interfaz descrita en el capítulo cuatro, se envió una serie de pulsos al microcontrolador. Sabiendo que los motores trabajan a una frecuencia máxima alrededor de los 60Hz, se buscó la velocidad máxima permitida por el sistema introduciendo valores de retardo entre 22ms y 15ms. Al probar cada uno de los valores de retardos de la tabla 5.3 se encontró que el motor del eje x y del eje y se volvía inestable a partir de un retardo de 19ms y 17ms respectivamente. Dicho de otra manera, las rmp (revoluciones por minuto) máximas permitido por el sistema son de 15.8 y 17.648 para el motor del eje x y el motor del eje y, respectivamente.

Retardo(ms)	rmp	frecuencia(Hz)
23	13.04	43.48
22	13.64	47.62
21	14.29	47.62
20	15	50
19	15.79	52.63
18	16.67	55.56
17	17.648	58.82
16	18.75	62.5
15	20	66.67

Cuadro 5.3. Prueba para ubicar la velocidad máxima del motor de paso.

5.2. Pruebas con motores de CC

Se repitieron las pruebas anteriormente realizadas con motoreductores de CC. El motorreductor Pololu[1] que se utilizó para realizar estas pruebas se muestra en la Figura 5.4.

El motorreductor empleado cuenta con un enconder de tipo incremental, que proporciona información acerca de la posición del eje del motor. El encoder tiene un sensor de efecto Hall y cuenta con dos canales que detectan la rotación del disco magnético que viene sujeto al eje del motor. La salida de los canales presentan 8400 pulsos por revolución del eje del motor. Si se cuentan únicamente los flancos de subido de un solo canal, el conteo total que se obtiene por revolución es de 2096. Lo último mencionado es la técnica que se empleó para conocer la posición del eje del motor.

Los movimientos obtenidos con este motor se presentan en el Cuadro 5.5.

[1]Pololu es el nombre de una empresa de robótica y electrónica estadounidense, http://www.pololu.com/

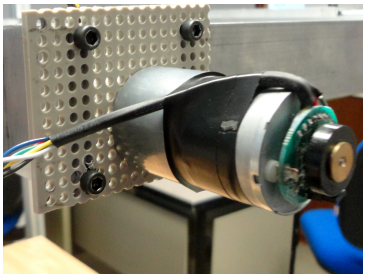


Figura 5.4. Motorreductor con encoder en cuadratura.

Torque	**velocidad @12V**	**relación de engranes**	**resolución de encoder**
18kg-cm	80rmp	131:1	8400 pulsos

Cuadro 5.4. Características del motorreductor Pololu.

Revoluciones	**Desplazamiento** en mm			**Error porcentual**	
	Teórico	Práctico			
		eje x	eje y	eje x	eje y
1	1.5875	1.52	1.53	4.252	3.622
2	3.175	3.061	2.941	3.591	7.37
3	4.7625	4.59	4.421	3.622	7.171
4	6.35	5.781	5.812	8.961	8.472
5	7.9375	7.687	7.721	3.156	2.728
6	9.525	9.412	8.812	1.186	7.486
7	11.1125	10.941	10.854	1.543	2.236
8	12.7	12.521	12.429	1.409	2.134
9	14.2879	13.691	13.56	4.172	5.092
10	15.875	15.269	15.45	3.817	2.677
11	17.4625	17.876	16.513	2.368	5.437
12	19.05	18.7125	18.633	1.772	2.189
13	20.6375	20.892	20.126	1.233	2.478
14	22.225	21.331	21.663	4.022	2.529
15	23.8125	22.457	22.9167	5.692	3.762
16	25.4	24.864	24.446	2.11	3.756
17	26.9875	26.234	26.452	2.792	1.984
18	28.575	28.143	27.766	1.512	2.831
19	30.1625	30.987	29.35	2.734	2.694

Cuadro 5.5. Desplazamiento obtenido con motores de CC.

5.3. Resultados y conclusiones

Los resultados obtenidos en los experimentos realizados demuestran que el posicionador es operativo; una revolución completa del eje del motor equivale a un avance de 1.5875 mm de la pieza que sostiene la cámara. Para el caso particular donde los actuadores del sistema son motores de paso, el posicionador require de 50,000 pasos para transportar la cámara de una zona de la celda a otra. Las pruebas de velocidad indican que el posicionador, operando a su máxima velocidad, logra un avance total de 26.46 mm por minuto en el eje x y un avance de 28.016 mm por minuto en el eje y.

Para un sistema basado en motores de CC, se obtuvó una mayor velocidad en el desplazamiento de los ejes x-y, con la consecutiva disminución de tiempo para alcanzar un par de coordenadas dentro de la zona de trabajo establecida. Las pruebas demuestran que la velocidad máxima en el eje x es de 127 mm por minuto mientras que la velociad máxima en el eje y es de 126.8 mm por minuto.

Las mediciones indican que se situa la cámara en un punto específico con un margen de error del 8 % para un sistema basado en motores de paso. Si el sistema opera con motores de CC el error se vuelve ligeramente mayor, alcanzando 9 %.

Anexo A

Definiciones

A.1. Mecanismos

Un mecanismo es un elemento que transforma el movimiento producido por un elemento motriz en un movimiento deseado de salida llamado elemento conducido. Existen dos grupos de mecanismos:

- Mecanismos de transmisión del movimiento: son aquellos en los cuales el elemento motriz y el elemento conducido tiene el mismo tipo de movimiento.
- Mecanismos de transformación del movimiento: son aquellos que cambian el tipo de movimiento de circular a lineal, o a la inversa.

Existen diversas formas en las cuales se puede aplicar la potencia de un motor a una máquina para producir diferentes tipos de movimiento. Algunos de estos métodos consisten en el uso de bandas, cadenas, cables, piñón, cremallera y tornillos [4].

A.1.1. Mecanismo husillo-tuerca

El mecanismo husillo-tuerca, también conocido como tornillo-tuerca, es un dispositivo de transformación de circular a lineal compuesto por una tuerca alojada en un eje roscado (tornillo). Mediante este sistema se consigue convertir el movimiento circular del tornillo en movimiento rectilíneo de la tuerca.

El husillo se caracteriza por el número de entradas (e) y por el paso de la rosca (p). El *número de entradas* se refiere al número de filetes de una rosca, mientras que el *paso de rosca* se refiere a la distancia entre dos filetes consecutivos.

$$A = p \times e \tag{A.1}$$

El paso de rosca es un parámetro importante para calcular el desplazamiento que se da en la tuerca por cada revolución del eje del motor.Por ejemplo, si se gira una vuelta completa un tornillo con una entrada y cuyo paso es de rosca es de 1mm, el avance producido será de 1mm. La ecuación A.1 determina el avance de la tuerca en función del paso de rosca y el número de entradas. Se puede medir el paso de rosca contando el número de filetes por centímetro o pulgada. El paso de rosca será el

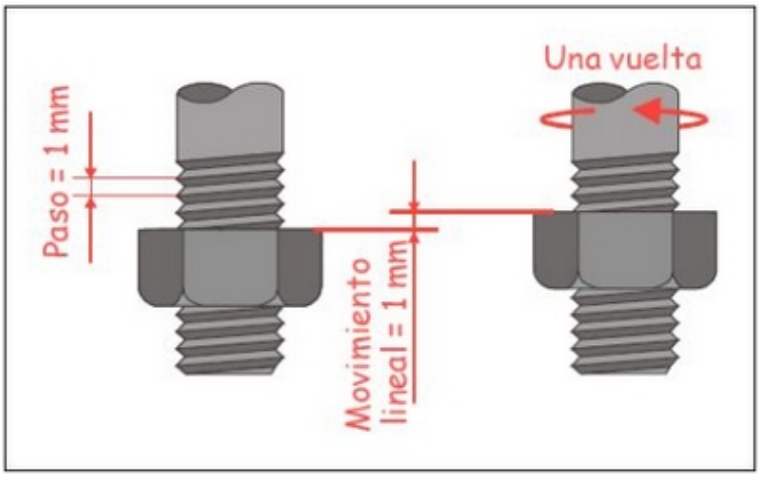


Figura A.1. Mecanismo husillo tuerca

inverso del valor obtenido. Si n es igual al número de filetes por centímetro o pulgada, el paso de rosca se puede expresar como [4]:

$$p = \frac{1}{n} \tag{A.2}$$

El tiempo que tarda la tuerca en recorrer la distancia (l) es

$$t = l/V_a \tag{A.3}$$

donde V_a es

$$V_a = A \times n = p \times e \times n \tag{A.4}$$

Existen diferentes tipos de rosca en función de la forma del perfil del filete, del número de filetes que tenga, del paso de la propia rosca, y del sentido de giro de avance del tornillo. Generalmente el perfil de una rosca suele ser de forma triangular, sin embargo también existen roscas de perfil cuadrado, trapecial, y en diente de sierra o redondo.

En cuanto al sentido de giro de la rosca, normalmente el avance se produce girando la rosca a la derecha, mientras que el retorno se produce girando la rosca a la izquierda. También existen tornillos y tuercas que funcionan en sentido inverso.

A.2. Motores de paso

Un motor de paso es un dispositivo electromecánico capaz de convertir una serie de pulsos de corriente en desplazamientos angulares discretos. Esto quiere decir que es capaz de avanzar una serie de grados a la vez dependiendo del estado de sus entradas de control. Este paso puede variar desde 90 grados hasta pequeños movimientos de tan solo 0.9 grados [12, 13].

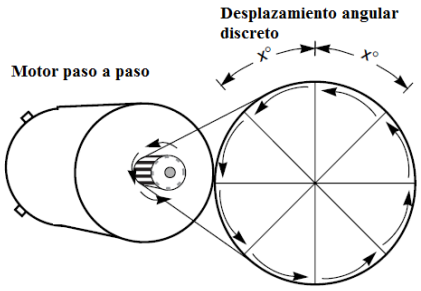


Figura A.2. Motor de paso [12]

La dirección de rotación está relacionada con la secuencia de pulsos aplicados. La duración de la rotación está directamente relacionada con el número de pulsos de entrada que se aplican. La velocidad de rotación del eje del motor se puede controlar variando la frecuencia de pulsos.

Los motores de paso son ideales para la construcción de mecanismos en donde se requiere de movimientos muy precisos. Poseen la habilidad de poder quedar enclavados en una posición o bien totalmente libres. Una de sus principales ventajas es que pueden ser controladas de una manera precisa en lazo abierto.

A.2.1. Tipo de motores de paso

Los motores de paso están constituidos esencialmente por un estator y un rotor. El estator es la parte fija y el rotor es la parte móvil. Los motores de paso se pueden clasificar en tres tipos según su construcción interna [14, 15] :

1. Reluctancia variable
2. Imán permanente
3. Híbridos

Los motores de reluctancia variable están formados por un rotor dentado de hierro suave y por un estator bobinado. La Figura A.3, muestra un ejemplo de la estructura física de un motor de paso de este tipo. Las bobinas que se encuentran físicamente opuestas una de la otra, se energizan para obtener campos magnéticos opuestos. Es decir, se energizaría la bobina A y $\overline{A}$ de tal forma que A fuera polo magnético negativo y $\overline{A}$ un polo magnético positivo. La rotación surge cuando los dientes del rotor son

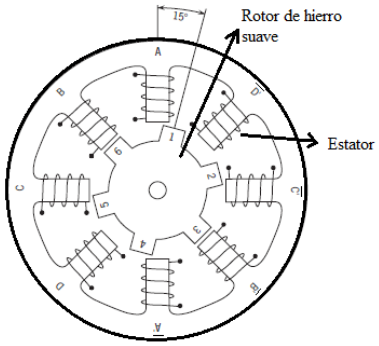


Figura A.3. Motor de reluctancia variable [13]

atraídos por los polos del estator energizado.

Los motores de imán permanente son motores de bajo costo y de baja resolución. Su ángulo de paso se encuentra entre 7.5°y 15°(48 a 24 pasos por revolución). A diferencia del motor de reluctancia variable, su rotor está compuesto por un imán permanente. La Figura A.4 muestra la estructura interna de un motor de imán permanente con un rotor cilíndrico de 6 polos y 8 estatores. En este caso existen únicamente dos bobinas: una conecta los polos A, C, E y G y la otra a los polos B, D, F y H. La rotación del rotor surge cuando se intercambian las polaridades de las bobinas. Estos motores exhiben un mejor par de torsión que los motores de reluctancia variable.

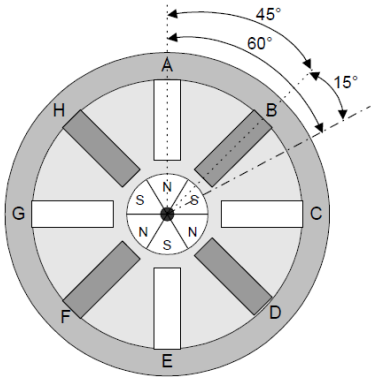


Figura A.4. Motor de imán permanente [15]

Los motores híbridos reúnen las mejores características de los motores de reluctancia variable y de imán permanente. Aunque son más costosos, ofrecen un mejor rendimiento en cuanto a resolución de paso, par de torsión y velocidad. El ángulo de paso típico para un motor híbrido se encuentra entre 3.6°a 0.9°(100-400 pasos por revolución).

Los motores paso a paso también se pueden clasificar según la forma de conexión de las bobinas del estator. Existen dos tipos de motores de paso:

- Bipolares
- Unipolares:

Un motor de paso unipolar está formado por dos bobinas con tomas intermedias. La toma intermedia puede salir del motor como dos cables separados, o bien, se pueden encontrar conectadas internamente y salen del motor como un solo cable. Esto se ilustra en la Figura A.5. Ya sea que el motor presente 5 o 6 hilos, el modo de operación es el mismo. Las tomas internas se conectan a una fuente de alimentación y los extremos de las bobinas se conectan a tierra de manera alternada.

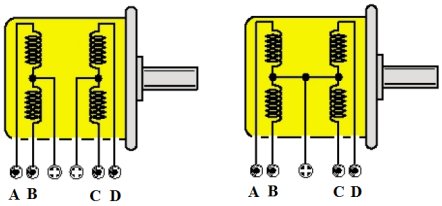


Figura A.5. Esquema de bobinas de un motor de paso unipolar

Un motor de paso bipolar está formado por dos bobinas. Se pueden reconocer externamente porque tiene cuatro cables como muestra la Figura A.6. A diferencia de los motores unipolares, los motores bipolares no tienen tomas intermedias. Esto resulta ventajoso porque la corriente puede fluir a través de una bobina en un instante de tiempo en vez de por solo la mitad de una bobina. Por esta razón, los motores bipolares producen mayor par torsión que los motores bipolares de un mismo tamaño. El flujo de corriente en un motor bipolar es bidireccional, esto quiere decir que requiere de un cambio de polaridad en los extremos de cada bobina [16, 17].

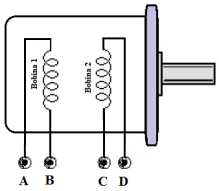


Figura A.6. Esquema de bobinas de un motor de paso bipolar

La elección entre unipolar o bipolar se basa en la simplicidad de control y la relación entre potencia y el peso. Los motores bipolares tienen aproximadamente 30 % más torque que su equivalente unipolar dado cierto volumen, sin embargo su control es más complicado [15].
Además de poder clasificar los motores de paso por su estructura interna y por su forma de conexión, los motores a paso se pueden clasificar por sus dimensiones. Por ejemplo, un motor de paso con tamaño de 11 tiene un diámetro de aproximadamente 1.1 pulgadas. Sin embargo, la longitud de cuerpo para motores del mismo tamaño puede variar según el motor. En general, el motor de paso con mayor longitud de cuerpo que otro con un mismo tamaño de cuadro, tendrá mayor par torsión.

A.2.2. Modos de control

Una rotación del eje del motor se logra aplicando una secuencia de pulsos de corriente en a cada una de las bobinas que compone el estator. Cada vez que alguna de estas bobinas se le aplica un pulso, el motor se desplaza un paso, y queda fija en esa posición. La posición del motor se puede conocer por el número de pulsos que se aplican. Las secuencia más comunes para controlar un motor de paso son:

- paso básico (*wave drive*)
- paso completo (*full step*)
- medio paso(*half step*)

La secuencia paso básico consiste en activar cada una de las bobinas de forma independiente, lo que provoca que el eje del motor se oriente hacia la bobina activa. La desventaja de esta secuencia es que no se obtiene el torque máximo en la salida del motor.

En el modo de funcionamiento paso completo, se activan dos bobinas en un instante dado. El movimiento angular en este modo de control resulta igual al modo paso básico. La desventaja de este modo de operación es que puede haber una pérdida de pasos a velocidades bajas.

El modo medio paso es una combinación del modo paso básico y el paso completo. Cada segundo paso se energizan dos fases y durante los demás pasos se energiza una fase. Su principal ventaja es que presenta una mayor resolución de paso, ya que disminuye el avance angular por la mitad que en modo de paso completo [13, 16].

A.2.3. Circuitos básicos de control

Para controlar un motor bipolar es necesario contar con un circuito capaz de invertir el flujo de corriente que circula a lo largo de cada bobina. Esto se puede lograr por medio de un mecanismo conocido como puente-H que alterna la polaridad en una terminal.El circuito de control básico para controlar un motor bipolar se muestra en la Figura A.7. Por cada embobinado se emplean cuatro elementos de conmutación (Q1,Q2,Q3 y Q4 para embobinado 1). Los diodos que están en paralelo con los MOSFETs protegen a los MOSFETs contra disparos de voltaje causados por los interruptores. Los cuatro MOSFETs deberán ser activados en pares, ya sea Q1 y Q4, o Q2 y Q3, pero nunca se deberá activar dos interruptores que esten en el mismo lado. Si ambos interruptores en un mismo lado (por ejemplo Q1 y Q2) se activan a la vez se creara un corto circuito que dañará al dispositivo.La corriente fluirá de derecha a izquierda cuando los MOSFETs Q1 y Q4 están activados mientras Q2 y Q3 están apagados. La corriente fluirá de izquierda a derecha si los MOSFETs Q2 y Q3 se encuentran encendidos mientras Q1 y Q4 están apagados [16, 13].

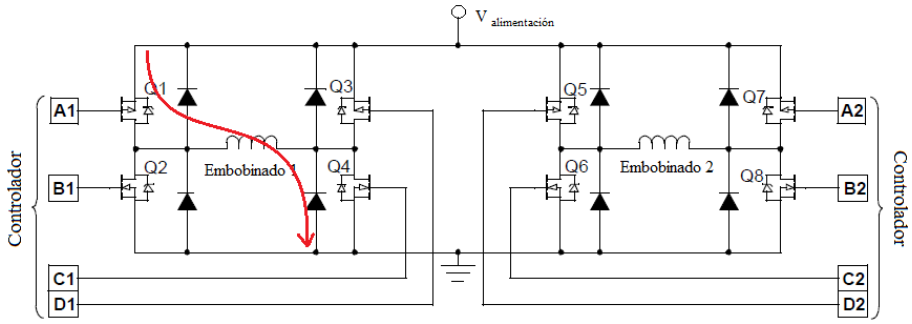


Figura A.7. Circuito de control para un motor de paso bipolar [13]

A.2.4. Parámetros de los motores de paso

El par del motor(torque) asi como el ángulo de paso son parámetros críticos en la selección de un motor de paso. El par del motor es un indicador de su fuerza mientras que el ángulo de paso determina su resolución. Los parámetros principales que caracterizan un motor paso a paso se señalan en [12, 15] como:

- Ángulo de paso (*step angle*)- Se define como el avance angular producido en el eje del motor por cada impulso de excitación. Se mide en grados.
- Par de mantenimiento (*Holding Torque*)- Es el par máximo que un motor de paso energizado puede soportar sin perder pasos
- Par de detención(*DetentTorque*)-Es el par máximo que un motor de paso no-energizado puede soportar sin perder pasos
- Frecuencia de paso máximo (*Maximum pull-in/out*). Es el máximo número de pasos por segundo que puede ejecutar el motor funcionando adecuadamente.
- Par dinámico de trabajo (*Working Torque*) Es el momento máximo que el motor es capaz de desarrollar sin perder paso, es decir, sin dejar de responder a algún impulso de excitación del estator. Es importante tener en cuenta que cuando la velocidad de giro del motor aumenta se produce una disminución en el par motor.

Anexo B

Código en C# para el procesamiento de imagen

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;


//***** Librerías de Aforte.net***********
using AForge;
using AForge.Math.Geometry;
using AForge.Imaging;
using AForge.Imaging.Filters;

namespace Sistema_Posicionador
{
 class ProcesarImagen // Se define la clase como ProcesarImagen
  {
 /* Se define un metodo(Method) que permite calcular las coordenadas en pixeles
        de los 2 "puntos" preestablecidos en la zona de trabajo.
        La imagen pasa por las siguientes 3 etapas de procesamiento:
       * (1) Filtro de Color RGB
       * (2) Escala de Grises
       * (3) BlobDetection
  * Finalmente, se realizan los siguientes cálculos:
       * (1)  coordenadas en pixeles de los dos puntos
       * (2)  distancia entre los dos puntos
        * */
  // declaracion de variables que son utilizadas fuera de esta clase(Class)
    public IntPoint P1; // coordenada en pixeles del Punto 1
    public IntPoint P2; // Coordenada en pixeles del Punto 2
    public double distanciaPixeles;

 //Método principal (main method)
  // Se declara una funcion que devuelve una imagen
   public Bitmap BuscarPuntos(Bitmap imagenCapturada)
     {
     // Paso (1)
     ColorFiltering FiltroRGB = new ColorFiltering();
```

```
    // conserva colores en este intervalo
     FiltroRGB.Red = new IntRange(152, 255);
     FiltroRGB.Green = new IntRange(0, 139);
     FiltroRGB.Blue = new IntRange(0, 155);
 // Aplicar el filtro a la imagen capturada
     Bitmap _imagen = FiltroRGB.Apply(imagenCapturada);
  // Paso (2)
  //--------Convertir a escala de grises--------------
    _imagen = Grayscale.CommonAlgorithms.BT709.Apply(_imagen);
  //------Aplicar Umbral---------
     Threshold filtroUmbral = new Threshold(34);
     filtroUmbral.ApplyInPlace(_imagen);
           // Paso (3)

      BlobCounter blobCounter = new BlobCounter();
      // opciones de filtadro
      blobCounter.FilterBlobs = true;
      blobCounter.MinWidth = 3;
      blobCounter.MinHeight = 3;
      blobCounter.MaxWidth = 20;
      blobCounter.MaxHeight = 20;

      blobCounter.ProcessImage(_imagen);
      Blob[] blobs = blobCounter.GetObjectsInformation();

      Graphics g = Graphics.FromImage(imagenCapturada);
      Pen fuente1 = new Pen(Color.Blue, 2);
      Pen fuente2 = new Pen(Color.Green, 2);

      IntPoint Punto1 = new IntPoint(0, 0);
      IntPoint Punto2 = new IntPoint(0, 0);
      if (blobs.Length > 0) // ¿¿se ubicaron nubes de pixeles??
        {
          Blob PrimerPunto = blobs[0]; // Primer punto de referencia
          Blob SegundoPunto = blobs[0]; // Segundo punto de referencia
// busca el primer punto de referencia (el punto más grande)
      foreach (Blob blob in blobs)
         {
    if (blob.Area > PrimerPunto.Area) // si el area del blob es mayor que el primer blob
          {
            PrimerPunto = blob; // el blob más grande se almacena en PrimerPunto
             }
               }
     foreach (Blob blob in blobs) //  busca el segundo punto de referencia
          {
   if ((blob.Area > SegundoPunto.Area) && (blob.Area != PrimerPunto.Area))
              {
               SegundoPunto = blob;
              }

          }

   // Se obtienen las coordenadas en pixeles de los Puntos de Referencia
   Punto1 = (IntPoint)PrimerPunto.CenterOfGravity;// se ubica el centroide del punto
   P1 = Punto1;
   g.DrawRectangle(fuente1, PrimerPunto.Rectangle);
```

```
   Punto2 = (IntPoint)SegundoPunto.CenterOfGravity;
   P2 = Punto2;
   g.DrawRectangle(fuente2, SegundoPunto.Rectangle);
// calcula la distancia entre los dos puntos de referencia
   double distance = Punto1.DistanceTo(Punto2);
   distanciaPixeles = distance;
   g.Dispose();
   fuente1.Dispose();
   fuente2.Dispose();
        }
     Bitmap Toma = imagenCapturada;
     return Toma; // devuelve la imagen capturada con los puntos enmarcados por un rectangulo.

       }
    }
}
```

Bibliografía

[1] Kalpakjian Schmid "*Manufactura, Ingeniería y tecnología* ",Editorial Prentice Hall, Cuarta edición, 2002

[2] http://www.parker.com , Parker Hannifin Corp 2012

[3] Matthew P. Stephen "Diseño de Instalaciones de Manufactura y Manejo de Materiales"Editorial Prentice Hall, Tercera edición, 2009

[4] Richard Buynas, Keith Nisbett "Shigley's Mechanical Engineering Design ", Editorial Mac Graw Hill, Ocatava edición, 2006

[5] Alan Overby "*CNC Machining Handbook, Builind Programming and Implementation* "Editorial MacGraw-Hill, Primera edición, 2011

[6] Jorge Rojas, Ingeborg Mahla, Gerardo Muñoz, Daniel Castro, "Diseño de un Sistema Robótico Cartesiano Para Aplicaciones Industriales "Revista Facultad de ingeniería, U.T.A (Chile), 2003.

[7] http://www.schneider-electric.com, Schneider Electric

[8] Microchip PIC18F2420/2520/4420 Enhanced Flash Microcontroller with 10-bit A /D and NanoWatt Technology, Datasheet. Microchip Technology Inc, 2004.

[9] Eduardo García Brejio "*Compilador C CCS y Simulador Proteus para microcontroladores PIC* ", Editorial Alfaomega,Primera edición, 2008

[10] Jason Price, Mike Gunderloy "*Mastering Visual C#.net*", Editorial Sybex, Primera edición, 2002

[11] http://msdn.microsoft.com/en-us/library/5b13a7k4(v=vs.100).aspx, Microsoft Developer Network,2012

[12] Matthew Grant "Quick Start for Beginners to Drive a Stepper Motor"(AN2974), Freescale Semiconductor, 2005

[13] Reston Condit "*Stepper Motor Fundamentals* "(AN907), Freescale Semiconductor,2005

[14] V.V Ahani "Stepper Motor Fundamentals,Applications and Design", Editorial New Age International, Primera edición, 2005

[15] Thomas L. Hopkins "*Stepper Motor Driver Considerations Common Problems & Solutions.* "(AN460), STMicroelectronics, 2003

[16] Padmaraja Yedamale, Sandip Chattopadhyay "Stepper Motor Microstepping with PIC18C452", Microchip, 2002

[17] H. SAX "Stepper Motor Driving "(AN235), Microelectronics,1995

[18] Andrzej M. Pawlak "*Sensors and Actuators in Mechatronics, Design and Applications* "Editorial Taylor and Francis, Primera edición, 2007

[19] Goñi Hernández Fernando,Ríos Cabrera Reyes, López Juárez Ismael "*Sistema de Posicionamiento Retroalimentado por Sensor de Visón para Sistema Flexible de Manufactura* ". XIX Congreso Nacional de Instrumentación.

[20] Ríos-Cabrera Reyes, Peña-Cabrera Mario, Goñi-Hernández Fernando,López-Juárez Ismael,"*Object Recognition Methology for Part Grasping in Manufacturing Cell* ". International Symposium on Robotics and Automation 2004.

[21] Peña-Cabrera Mario, Juárez López Ismael," *A Learnig Approach for On-Line Object Recognition in Robotic Tasks* ", Mexican International Conference on Computer Science ENC 2004.

[22] Suk-Hwan Suh, Seong-kyoon Kang, Dae-Hyuk, Ian Stroud, " *Theory and Design of CNC systems* "Editorial Springer, Primera edición, 2008

[23] Andrés García,Fernando J. Castillo García "*CIM,el computador en la automatización de la producción*