UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO.

ESCUELA SUPERIOR DE TIZAYUCA.

"Control de Posicionamiento de un Robot Cartesiano con Herramienta de Perforación."

Tesis que para obtener el título de:

Ingeniero en Electrónica y Telecomunicaciones.

Presenta:

Jhovani Olvera Arenas.

Asesores de tesis:

M. en C. Asdrúbal López Chau.

Ing. Hugo Ruiz González.

Agradecimientos

A Dios por brindarme la oportunidad de llegar a este momento final de mi carrera.

A mis padres por el gran esfuerzo realizado para poder brindarme la oportunidad de obtener una educación de calidad.

A mi asesor de tesis, M. en C. Asdrúbal López Chau, por compartir sus enseñanzas, pero sobre todo por dedicarme gran parte de su tiempo y poder terminar este trabajo.

A mi asesor de tesis, Ing. Hugo Ruiz González, por el gran apoyo brindado para poder terminar este trabajo.

Al M. en C. Herbert Lara Ordaz por su valiosa colaboración en la corrección de estilo de este trabajo.

A todos los profesores por compartirme sus grandes conocimientos y experiencias a lo largo de mi carrera.

Resumen

En este trabajo se describe el diseñó y construcción de una propuesta básica para el control de posiscionamiento de un robot cartesiano con herramienta de perforación. Los elementos principales de este prototipo son un microcontrolador (responsable de la asignación y ejecución de las coordenadas de desplazamiento) y los motores a pasos (asociados al desplazamiento y activación del taladro de perforación).

El robot cartesiano propuesto posee una estructura mecánica la cual le permite lograr desplazamientos lineales precisos asociados a los ejes X, Y y Z. Las coordenadas para su posicionamiento son asignadas desde una computadora por medio de la interfaz o gráfica de usuario desarrollada con C++; la comunicación (interfaz) entre el software desarrollado y el robot cartesiano se lleva a cabo a través del puerto serial de la computadora y un microcontrolador AVR.

Por medio del puerto serial se envían los datos hacia el microcontrolador, estos datos son leídos por la USART (Transmisión Recepción Asíncrona Síncrona Universal) del microcontrolador, posteriormente estos datos son interpretados por el microcontrolador y son enviados hacia los motores del robot a través de sus diferentes puertos. Estas señales tienen un valor insuficiente para hacer funcionar a los motores por lo que es necesario utilizar un circuito integrado L298 el cual se encarga de conmutar los voltajes procedentes del microcontrolador por voltajes con un valor mayor, para que sean enviados hacia los diferentes motores.

Es importante mencionar que la estructura mecánica del robot cartesiano propuesto en este trabajo está realizada en forma empírica, lo que significa que no se utilizaron conceptos matemáticos de robótica y por lo tanto no se realizaron cálculos para su diseño, debido a que el uso del microcontrolador es el elemento fundamental en el desarrollo de este tipo de sistemas, el diseño de este robot se puede considerar como una comprobación de su adecuado funcionamiento.

Abstract

This paper describes the design and construction of a basic proposal for controlling posiscionamiento a Cartesian robot with tool drilling. The main elements of this prototype is a microcontroller (responsible for the allocation and implementation of the coordinates of displacement) engines and steps (associated with the movement and activation of the drill hole).

The proposed Cartesian robot has a mechanical structure which allows you to achieve precise linear displacement associated with the X, Y and Z. The coordinates for their position are assigned from a computer via the graphical user interface or software developed in C++, communication (interface) between the software developed and the Cartesian robot is performed through the serial port computer and an AVR microcontroller.

Through the serial port the data is sent to the microcontroller, these data are read by the USART (Universal Synchronous Asynchronous Transmit Receive) of the microcontroller, then these data are interpreted by the microcontroller and sent to the robot motors through its different ports. These signals have a limited value for making engines work so you must use an integrated circuit L298 which is responsible for switching the voltages from the microcontroller voltages with greater value, to be sent to different engines.

It is noteworthy that the mechanical structure of the Cartesian robot proposed in this work is done empirically, which means not used mathematical concepts in robotics and therefore there were no calculations for their design, because the use of the microcontroller is the fundamental element in the development of such systems, the design of this robot can be considered as a verification of its proper functioning.

Índice general

Índice de figuras

Índice de tablas

Introducción

Existen muchos trabajos que el ser humano no puede realizar con rapidez y exactitud, ya sea porque es complicado, peligroso o simplemente porque es una actividad tediosa. Una solución práctica es realizarlos a través de una máquina o en este caso un robot cartesiano.

Los robots cartesianos son ideales para trabajos que requieren movimientos repetitivos y muy precisos. La ventaja para las empresas es que los robots no necesitan salario, comida, dormir, y una área segura para trabajar. La fatiga y aburrimiento de los humanos afectan directamente a la producción de una compañía; con los robots no sucede esto, por lo tanto su trabajo va a ser el mismo durante el ciclo de trabajo.

Los beneficios que se obtienen al implementar un robot cartesiano son:

- Flexibilidad productiva.
- Mejoramiento de la calidad.
- Disminución de agotamiento en el proceso de producción.
- Mejoramiento de las condiciones de trabajo, reducción de riesgos personales.
- Mayor productividad.
- Ahorro de energía.

La propuesta presentada en este trabajo consiste en el desarrollo de un prototipo elemental de robot cartesiano basada en el uso y aplicación de un microcontrolador AVR que establece la comunicación entre una PC y el robot para asignar las coordenadas de desplazamiento mediante el establecimiento de una rutina específica; sin embargo, aunque la propuesta sea básica se intenta cumplir con algunos de los beneficios que aportan este tipo de robots en sus aplicaciones generales.

Justificación

Las aplicaciones de los robots cartesianos en los diferentes ámbitos resaltan su importancia, sin duda el uso de este tipo de sistemas es fundamental en muchos procesos en los cuales sería casi imposible que el ser humano ejecutara acciones repetitivas sin mostrar signos de cansancio y tedio y como consecuencia se presentarán errores o imprecisiones dentro del proceso.

Para el ingeniero en electrónica y telecomunicaciones el uso y aplicación de dispositivos electrónicos es fundamental en su formación profesional, por eso el desarrollo de un sistema que involucre conceptos relacionados con la programación y comunicación de circuitos electrónicos y la activación de distintos tipos de motores contribuye a su formación en forma importante, tal es el caso de este prototipo que permite poner en práctica todos estos aspectos.

El robot cartesiano propuesto se caracteriza por la aplicación de estos conceptos y establece la importancia de este tipo de sistemas que tienen impacto en diferentes procesos industriales, aunque el diseño propuesto es elemental se considera que cumple con las características básicas y también con la filosofía que dio origen a su diseño.

Objetivos

Objetivo general

Diseñar un robot cartesiano con herramienta de perforación a través de la implementación de las etapas de control de movimiento, comunicación con la PC y la interfaz de usuario dependientes de un microcontrolador.

Objetivos específicos

- Diseñar e implementar el hardware necesario para el sistema electrónico de control del robot cartesiano.
- Implementar los circuitos de potencia para la alimentación de motores.
- Desarrollar el firmware necesario para diseñar el hardware electrónico para el control.
- Diseñar la interfaz de usuario (software).
- Realizar la estructura mecánica del robot cartesiano para realizar pruebas de posicionamiento.
- Acoplar elementos mecánicos con el hardware electrónico para realizar pruebas de posicionamiento.

Planteamiento del problema

El robot cartesiano diseñado en esta tesis posee una estructura mecánica la cual permite realizar desplazamientos lineales asociados a los ejes X, Y y Z. Las coordenadas de posicionamiento son asignadas desde una computadora, a través de una interfaz gráfica de usuario.

El proceso de diseño y construcción del robot contempla tres etapas principales en su desarrollo, cada una de las cuales se presentan en el siguiente diagrama.

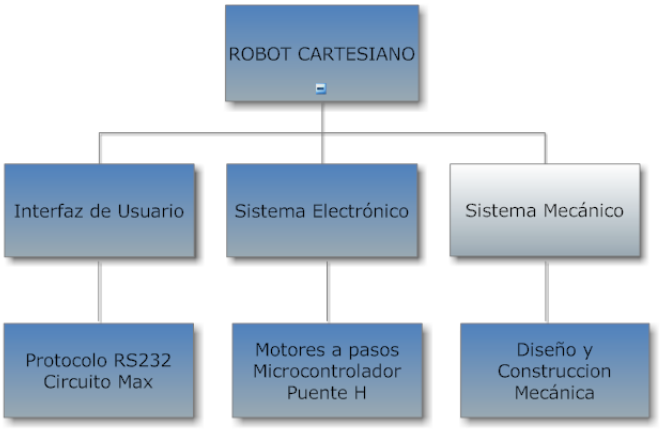

Figura 1: Descripción general del robot cartesiano.

En el diagrama de bloques mostrado en la figura 1 se muestran los puntos más importantes para el desarrollo y comprensión del funcionamiento del robot cartesiano; en el mismo bloque del diagrama se observa la relación entre la interfaz de usuario, el protocolo RS232, el circuito MAX232; posteriormente se identifica el sistema electrónico del robot conformado por los motores a pasos, el microcontrolador y los circuitos de potencia utilizados para control de los motores. El último bloque corresponde al sistema mecánico el cual involucra los elementos que conforman la estructura mecánica del robot.

Organización del trabajo.

El presente trabajo esta conformado por cuatro capítulos en los que esta descrito el desarrollo por etapas del prototipo básico del robot cartesiano propuesto, el contenido general de cada uno de los capítulos se presenta a continuación.

El capítulo uno abarca en forma general conceptos básicos acerca de la robótica, el principio de funcionamiento de los motores a pasos, los tipos de motores a paso a paso , el concepto y la arquitectura interna de un microcontrolador AVR y la definición del protocolo RS-232.

En el capítulo dos se realiza la descripción a detalle del sistema electrónico del robot cartesiano, el tipo de motores a pasos que utilizan para efectuar los desplazamientos y por ultimo la interfaz grafica de usuario la cual se desarrolló empleándo C++ y los circuitos electrónicos de control del robot cartesiano.

El capítulo tres presenta información relacionada al diseño del sistema mecánico del robot cartesiano propuesto, el proceso de diseño y la construcción de cada una de las piezas que conforman la estructura del robot.

El capítulo cuatro describe el ensamble de cada una de las piezas mecánicas del robot cartesiano, las pruebas necesarias realizadas para la comprobación del correcto funcionamiento del robot, se menciona también trabajo a futuro y las conclusiones generales.

Capítulo 1

Conceptos básicos

1.1. Introducción.

Las máquinas y la mecanización, han incrementado la fuerza muscular; la computadora ha incrementado el poder mental, los sentidos del hombre se han ampliado por medio de instrumentos y dispositivos de medición, hemos llegado a una era en la cual la tecnología, especialmente la de los robots, no sólo incrementará nuestras capacidades humanas sino que también podrá reemplazarlas por completo.

La aplicación de los robots para manejo de materiales ofrece un gran beneficio para librar al ser humano de trabajos aburridos, cansados o peligrosos.

Con una necesidad de superar la productividad las industrias manufactureras están optando más hacia la manufactura flexible auxiliada por computadora. La inflexibilidad de muchas máquinas nos ha llevado al interés del uso de robots industriales.

La robótica se relaciona en sí con el deseo de sintetizar algunos aspectos de la función humana mediante el uso de mecanismos, sensores, actuadores y computadoras.

La palabra robótica fue utilizada por primera vez por el científico y escritor de ciencia ficción Isaac Asimov en 1942. El propuso las llamadas *leyes de la robótica:*
Ley 1: Un robot no puede realizar ninguna acción, que resulte perjudicial para la humanidad, aun cuando ello entre en conflicto con las otras leyes.
Ley 2: Un robot no puede dañar a un ser humano, y no puede permitir que éste sea dañado.
Ley 3: Un robot debe obedecer las órdenes dadas por los seres humanos excepto cuando estas órdenes entren en conflicto con las leyes anteriores.

Ley 4: Un robot debe proteger su propia existencia hasta donde esta protección no entre en conflicto con las leyes anteriores.

La palabra robot fue introducida por el checo Karel Capek en 1921, y se deriva de la combinación de las palabras checas robota"que significa trabajo obligatorio y robotnik"que significa siervo.

Un robot es un manipulador reprogramable de uso general con sensores externos que pueden efectuar diferentes tareas de manejo. Con esta definición un robot debe poseer inteligencia que se debe normalmente a los algorítmos de la computadora asociados con un sistema de control.

Configuración de los robots de acuerdo a su estructura mecánica.
Cuando se habla de la configuración de un robot, se refiere a la forma física que se le ha dado al robot, el cual puede representar cuatro configuraciones clásicas:

- Angular.
- Cilíndrica.
- Polar.
- Cartesiana.

Configuración cilíndrica.
En esta configuración se pueden realizar dos movimientos lineales y un rotacional,(ver figura 1.1)lo que representa tres grados de libertad.
El robot de configuración cilíndrica está diseñado para ejecutar los movimientos conocidos como interpolación lineal e interpolación por articulación.

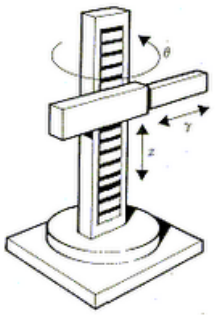

Figura 1.1: Configuración cilíndrica.

Configuración Polar.
Esta configuración tiene varias articulaciones,(ver figura 1.2) cada una de ellas realiza un movimiento distinto, que puede ser rotacional, lineal y angular.
Este robot utiliza la interpolación por articulación para moverse en sus dos primeras articulaciones y la interpolación lineal para la extensión y retracción.

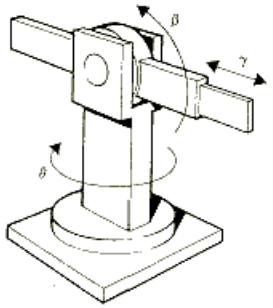

Figura 1.2: Configuración polar.

Configuración Angular.
Esta configuración presenta una articulación con movimiento rotacional y dos angulares. (ver figura 1.3). Aunque el brazo articulado puede realizar el movimiento llamado interpolación lineal (para lo cual requiere mover simultáneamente dos o tres de sus articulaciones), el movimiento natural es el de interpolación por articulación, tanto rotacional como angular.

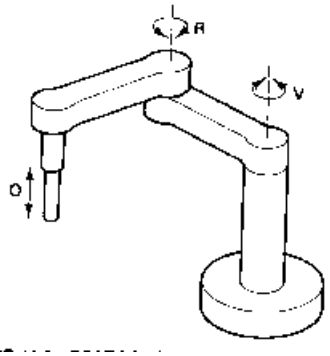

Figura 1.3: Configuración angular.

Configuración Cartesiana.
Esta configuración posee tres movimientos lineales es decir tiene tres desplazamientos lineales, los cuales son asociados a los ejes X, Y y Z (ver figura 1.4).

Los movimientos que realiza este robot entre un punto y otro son con base en interpolaciones lineales. Interpolación, en este caso, significa el tipo de trayectoria que realiza el robot cuando se desplaza entre un punto y otro.
A la trayectoria realizada en línea recta se le conoce como interpolación lineal y a la trayectoria hecha de acuerdo con el tipo que tienen sus articulaciones se llama interpolación.

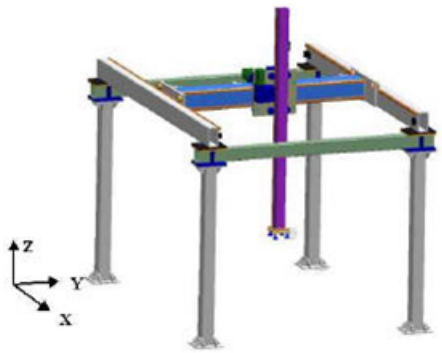

Figura 1.4: Configuración cartesiana.

Los movimientos o desplazamientos que realiza un robot cartesiano son trayectorias lineales, por lo que son ideales para realizar trabajos en donde se requiera de precisión y exactitud.
Generalmente estos movimientos realizados por el robot son ocasionados por algún dispositivo capaz de generar un desplazamiento, como puede ser un motor. El desplazamiento del motor puede ser ocasionado por medio de una señal eléctrica generada por un algún dispositivo electrónico.

En esta tesis se diseña un robot cartesiano de tres grados de libertad asociados a los ejes X, Y y Z, debido a que en este diseño se requiere de movimientos precisos y exactos, se utilizan motores a pasos de corriente directa para poder generar los desplazamientos del robot.

Los motores a pasos son dispositivos que generan desplazamientos muy precisos esto es porque internamente su estator esta formado por un grupo de bobinas, para poder ocasionar un desplazamiento en el rotor de un motor a pasos es necesario asignar un pulso en cada una de sus bobinas, de tal modo que el rotor gire hasta completar un giro o revolución, debido a esta característica el número de pasos realizados por el motor es proporcional al número pulsos asignados a el estator. A continuación se describe el principio de funcionamiento de un motor a pasos.

1.2. Motores a pasos.

En términos generales, los motores eléctricos convierten la energía eléctrica en energía mecánica rotatoria, basan su funcionamiento en las fuerzas ejercidas por un campo electromagnético, que son creadas al hacer circular una corriente eléctrica a través de una o varias bobinas. Si dicha bobina, generalmente circular y denominada estator, se mantiene en una posición mecánica fija y en su interior, bajo la influencia del campo electromagnético, se coloca otra bobina, llamada rotor, recorrida por una corriente y capaz de girar sobre su eje, esta última tiende a buscar la posición de equilibrio magnético, es decir, orientará sus polos NORTE-SUR hacia los polos SUR-NORTE del estator, respectivamente. Cuando el rotor alcanza esta posición de equilibrio, el estator cambia la orientación de sus polos, aquel tratará de buscar la nueva posición de equilibrio, manteniendo dicha situación de manera continua, se conseguirá un movimiento giratorio y continuo del rotor y a la vez la transformación de una energía eléctrica en otra mecánica en forma de movimiento circular.

En este diseño es necesario convertir la energía eléctrica en energía mecánica, para lo cual se utilizan motores de corriente continua. Pero como lo que se requiere es un posicionamiento preciso y una muy buena regulación de la velocidad, es conveniente utilizar motores a pasos.

Los motores a pasos son dispositivos que genera movimientos muy precisos, ya que por cada pulso proporcionado a las bobinas del motor, ocasiona un desplazamiento llamado paso, y en base al número de pulsos otorgados, el avance del motor es proporcional a estos. Son controlados por una serie de bobinas electromagnéticas. El centro de el eje esta constituído por una serie de imanes montados sobre el mismo, de esta forma el paso de la corriente va alternándose dependiendo de la polaridad que contengan las bobinas que rodean al eje, creando campos magnéticos de repulsión o atracción sobre el eje, causando que este gire. Estos motores utilizan esta característica del eje para producir movimientos de gran precisión, hasta una determinada posición en específico.

En la figura 1.5 se muestra el diagrama interno de un motor a pasos unipolar, el cual cuenta con cuatro bobinas conectadas en serie, a su vez forman dos grupos los cuales están conectados entre sí por un común.

1.2.1. Tipos de motores a pasos.

De acuerdo a su construcción interna, estos motores son clasificados en 3 tipos, los cuales son:

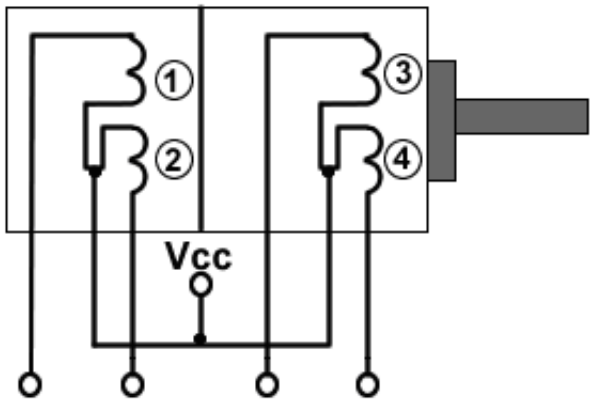

Figura 1.5: Motor a pasos unipolar

- De imán permanente.
- De reluctancia variable.
- Híbridos.

De imán permanente: El rotor es un imán permanente en el que se mecanizan un número de dientes limitado por su estructura física. Ofrece como principal ventaja que su posicionamiento no varía aún sin excitación y en régimen de carga.

De reluctancia variable: Los motores de este tipo poseen un rotor de hierro que en condiciones de excitación del estator y bajo la acción de su campo magnético, ofrecen menor resistencia a ser atravesado por su flujo en la posición de equilibrio. Su mecanización es similar a los de imán permanente y su principal inconveniente radica en que en condiciones de reposos (sin excitación) el rotor queda en libertad de girar y, por lo tanto, su posicionamiento de régimen de carga dependerá de su inercia y no será posible predecir el punto exacto de reposo.

Híbridos: Son combinación de los dos tipos anteriores; el rotor suele estar constituido por anillos de acero dentado en un número ligeramente distinto al del estator y dichos anillos montados sobre un imán permanente dispuesto axialmente.

Los motores mencionados anteriormente pueden ser de dos tipos, motores unipolares y motores bipolares, debido a que en el interior sus bobinas se encuentran conectadas de forma diferente.

Motores Unipolares: En este tipo de motores, todas las bobinas del estator están conectadas en serie formando cuatro grupos. Estas a su vez, se conectan dos

a dos, también en serie, y se montan sobre dos estatores diferentes, tal y como se aprecia en la figura 1.6. Como se puede apreciar del motor a pasos salen dos grupos de tres cables, uno de los cuales es común a dos bobinados.

Las seis terminales que parten del motor, deben ser conectados al circuito de control, el cual se comporta como cuatro conmutadores electrónicos, que al ser activados o desactivados, producen la alimentación de los cuatro grupos de bobinas con las que se encuentra formado el estator. Si generamos una secuencia adecuada de funcionamiento de estos interruptores, se pueden producir saltos de un paso en el número y sentido que se desee.

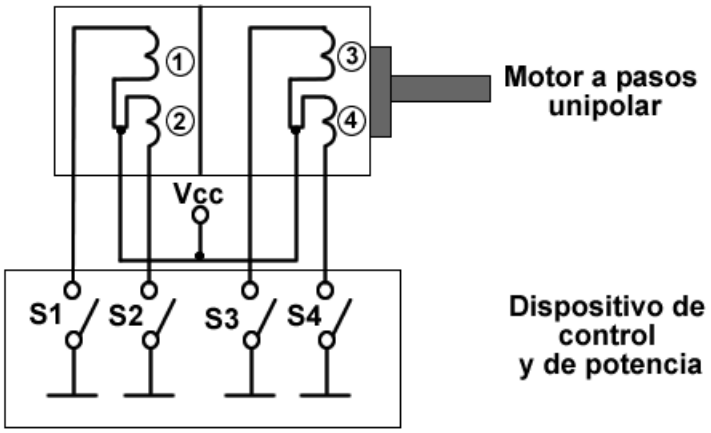

Figura 1.6: Control de un motor unipolar.

Motores Bipolares: En este tipo de motores las bobinas del estator se conectan en serie formando solamente dos grupos, que se montan sobre dos estatores, tal y como se muestra en la figura 1.7. Según se observa en el esquema de este motor salen cuatro hilos que se conectan, al circuito de control, que realiza la función de cuatro interruptores electrónicos dobles, que nos permiten variar la polaridad de la alimentación de las bobinas. Con la activación y desactivación adecuada de dichos interruptores dobles, podemos obtener las secuencias adecuadas para que el motor pueda girar en un sentido o en otro.

Para cada uno de estos tipos de motores se pueden dar pasos de ángulos muy grandes (4-24 pasos por revolución) o pasos de ángulos muy pequeños (50-200 pasos por revolución). Para altos rendimientos el máximo de pasos que se pueden dar, puede exceder a los 50 pasos por segundo (es decir, 3000 revoluciones por minuto en una maquina de 100 pasos / revolución), pero resulta más usual tener un número de pasos de grandes ángulos. Esto quiere decir que los motores a pasos son

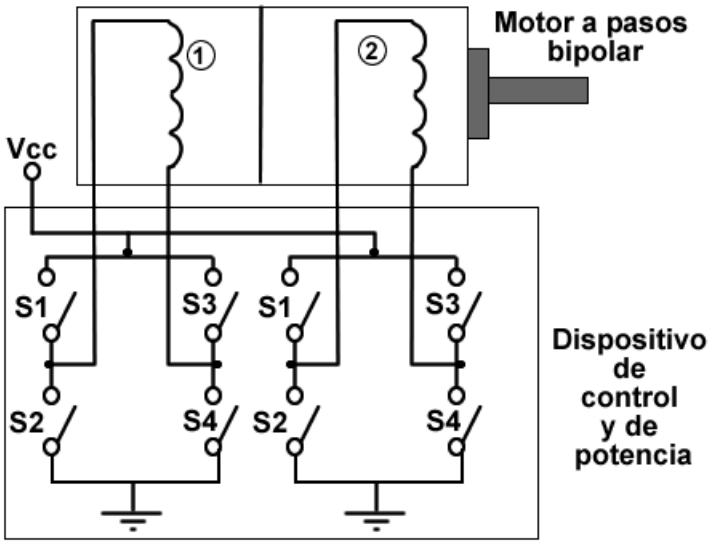

Figura 1.7: Control de un motor bipolar.

principalmente maquinas de baja velocidad con una potencia de salida muy modesta.

1.2.2. Funcionamiento de un motor a paso con imanes permanentes.

Los motores a pasos con imanes permanentes son en verdad una variedad de motores permanentes síncronos, teniendo la posibilidad de utilizar estos como motores a pasos y viceversa. La diferencia se encuentra en como el diseño lo optimiza para dos tipos diferentes de aplicaciones. La mayoría de los motores síncronos están optimizados para dar una frecuencia máxima de su tamaño.

Estos motores están diseñados para trabajar de forma continua con el rotor estacionario y generar una gran fuerza restauradora (momento de retención) si el motor se mueve de su posición. Como cualidades importantes de los motores a pasos podemos destacar la precisión en cada paso que puede dar.

Debido a que en un sólo motor a pasos se puede acelerar hasta, lo que, en efecto, es la velocidad de sincronización, se necesita un gran momento de inercia si se requiere una gran capacidad en el número de pasos. Una de las características más importantes es por lo tanto el número de pasos posibles por revolución cuando se incrementa la velocidad, para lo cual los motores se fabrican con mas de 200 pa-

sos/revolución. Este alto número de polos significa que, dado un tamaño fijo, la potencia de salida es muy pequeña. Para obtener una mejor relación entre potencia y tamaño se utilizan motores con pocos pasos / revolución. Para un uso de precisión los motores más usados son los de 50, 60, 100 y 200 pasos. Cuando el factor de potencia es mas importante se utilizan motores con ángulos más pequeños, típicamente de 6, 8, 12 o 24 pasos / revolución, el cuadro 1.1 muestra la relación entre los grados por cada impulso y el número de pasos en cada revolución.

Cuadro 1.1: Relación entre los grados por impulso de excitación y el número de pasos por revolución.

Grados por impulso de excitación	Número de pasos por revolución
0.72º	500
1.8º	200
3.6º	100
3.75º	96
6º	60
7.2º	50
7.5º	48
15º	24
30º	12
45º	8
60º	6

La mayor parte de los motores a pasos con imanes permanentes usan un rotor multipolar de imán permanente situado dentro de la bobina bifásica del arranque. Para este caso se utiliza una bobina bifásica en el arranque ya que es el mínimo número de fases necesario que hace girar al rotor en una dirección deseada. En cada posición el rotor esta en reposo respecto a los polos adyacentes opuestos, ya que es esta la posición de momento cero respecto al balance de las fuerzas magnéticas.

Conforme el rotor se mueve de esta posición se genera un momento restaurador que se incrementa de forma sinusoidal hasta llegar a un máximo que corresponde a otro de los bloques del arranque (paso completo) que se ha desplazado medio ciclo respecto a los polos magnéticos del rotor.

Si se sigue desplazando el rotor en la misma dirección, la fuerza restauradora caerá y cambiará su signo. Si se le deja libre en este punto se moverá hasta la siguiente posición de equilibrio cuatro pasos mas allá de la posición inicial.

1.2.3. Parámetros de los motores a pasos.

Desde el punto de vista mecánico y eléctrico, es conveniente conocer el significado de algunas de las principales características y parámetros que se definen sobre un motor paso a paso:

Par dinámico de trabajo (Working Torque) Depende de sus características dinámicas y 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 y dependiendo, evidentemente, de la carga.

Par de mantenimiento (Holding Torque) Es el par requerido para desviar, en régimen de excitación, un paso el rotor cuando la posición anterior es estable; es mayor que el par dinámico y actúa como freno para mantener el rotor en una posición estable dada

Par de detención (Detention Torque): Es una par de freno que siendo propio de los motores de imán permanente, es debido a la acción del rotor cuando los devanados del estator están desactivados.

Angulo de paso (Step angle): Se define como el avance angular que se produce en el motor por cada impulso de excitación se mide en grados.

Número de pasos por vuelta: Es la cantidad de pasos que ha de efectuar el rotor para realizar una revolución completa.

Frecuencia de paso máximo (Maximum pull-in/out): Se define como el máximo número de pasos por segundo que puede recibir el motor funcionando adecuadamente.

Momento de inercia del rotor: Es su momento de inercia asociado que se expresa en gramos por centímetro cuadrado.

1.2.4. Control de los motores a pasos.

Anteriormente en el control de los sistemas a pasos se utilizaban mecanismos de interrupción de conmutación para energizar los pasos de la bobinas del estator en secuencia, típicamente la transmisión bidireccional se da a través de indicadores de

rotación remota tales como los repetidores de compases.

Actualmente es usual usar controladores rápidos de alta velocidad de estado sólido, los cuales indican la secuencia de comandos al motor por medio de un reloj o contador (ver figura 1.8).

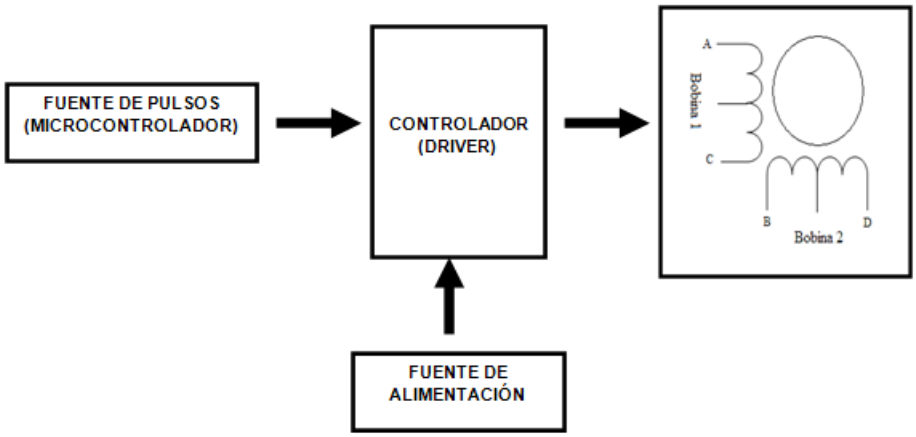

Figura 1.8: Diagrama de control de motores a pasos.

Para realizar el control de los motores a pasos, es necesario generar una secuencia determinada de pulsos. Además es necesario que estos impulsos sean capaces de entregar la corriente necesaria para que las bobinas del motor puedan generar el campo magnético necesario para hacer funcionar el motor, por lo general, el diagrama de bloques de un sistema con motores a pasos es el que se muestra en la figura 1.9.

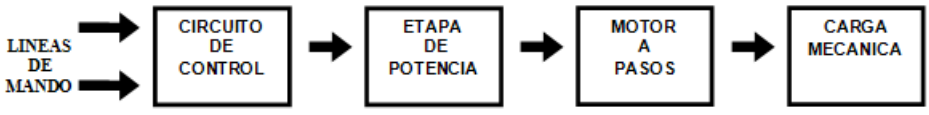

Figura 1.9: Diagrama de bloques de un sistema con motor a pasos.

En los motores a pasos las señales de control son trenes de pulsos que van actuando rotativamente sobre una serie de electroimanes colocados en el estator. Por cada

pulso recibido, el rotor del motor gira un determinado número discreto de grados. Para conseguir el giro del rotor en un determinado número de grados, las bobinas del estator deben ser excitadas secuencialmente a una frecuencia que determina la velocidad de giro. Las inercias propias del arranque y parada (aumentadas por las fuerzas magnéticas en equilibrio que se dan cuando esta parado) impiden que el rotor alcance la velocidad nominal instantáneamente, y por tanto la frecuencia de los pulsos que la fija, debe ser aumentada progresivamente.

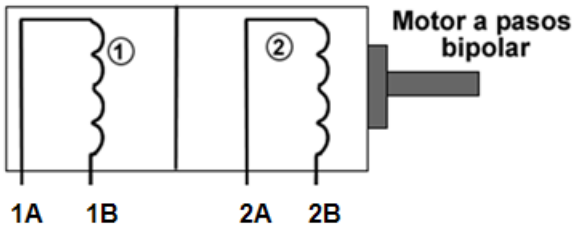

Figura 1.10: Motor a pasos bipolar.

En el cuadro 1.2 se observa la secuencia de pulsos que se debe de asignar a un motor a pasos bipolar para que ocasione un desplazamiento por cada uno de estos pulsos.

Cuadro 1.2: Excitación secuencial de motor a pasos de imanes permanentes con cuatro fases.

No. Pasos	1A	1B	2A	2B
Paso1	+Vcc	GNG	+Vcc	GNG
Paso2	+Vcc	GNG	GNG	+Vcc
Paso3	GNG	+Vcc	GNG	+Vcc
Paso4	GNG	+Vcc	+Vcc	GNG

1.3. Microcontroladores.

Hoy en día los microcontroladores están presentes en muchas aplicaciones de nuestra vida diaria, los podemos encontrar en nuestro trabajo, en los teléfonos, en los televisores de nuestro hogar, etc. Las extensas áreas de aplicación de los microcontroladores que se pueden considerar ilimitadas exigirán un gigantesco trabajo de

diseño y fabricación. Para poder aprender a manejar microcontroladores es necesario desarrollar prácticas de diseños reales.

Un microcontrolador es un circuito integrado programable que contiene todos los componentes de una computadora. Es empleado para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que controla, debido a esta característica puede ser denominado **controlador incrustado**. En su memoria solo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida pueden soportar la conexión de sensores y actuadores del dispositivo a controlar y todos los recursos complementarios disponibles tienen como única finalidad atender los requerimientos. Una vez programado y configurado el microcontrolador solamente sirve para realizar la tarea que le fue asignada a través del programa cargado. En la figura 1.11 se muestra la estructura de un sistema cerrado.

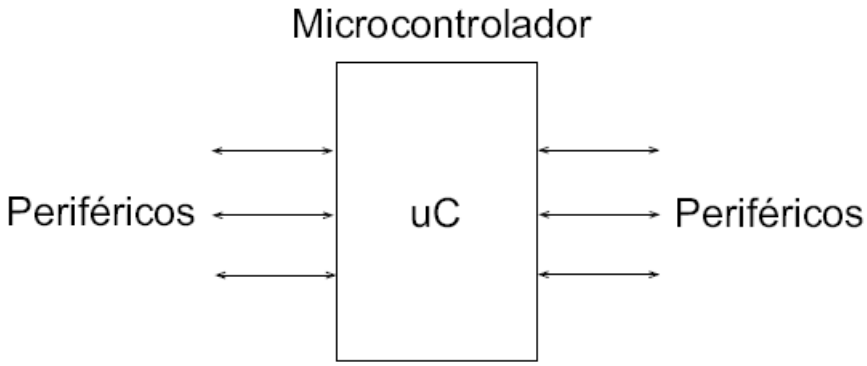

Figura 1.11: Estructura de un sistema cerrado de un microcontrolador.

El microcontrolador puede funcionar como una computadora completa, aunque de limitadas prestaciones, como lo son, que esta contenido en el chip de un circuito integrado y se destina a realizar una sola tarea.

1.3.1. Diferencia entre microprocesador y microcontrolador.

El microprocesador es un circuito integrado que contiene una Unidad Central de Proceso (UCP), también llamada procesador. La UCP esta formada por la Unidad de Control, que interpreta las instrucciones, y el camino de datos, que las ejecuta.

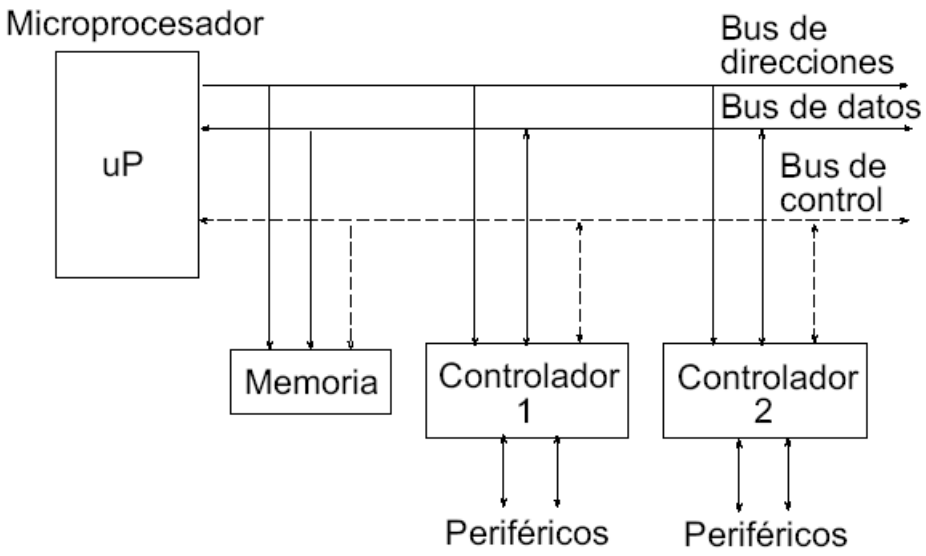

Figura 1.12: Estructura de un sistema abierto basado en un microprocesador.

Los pines del microprocesador tienen en el exterior las líneas de sus buses, direcciones, datos y control para poder permitir conectar la memoria y los módulos de entrada y salida. Se dice que un microprocesador es un sistema de arquitectura abierta, porque su configuración es variable de acuerdo con la aplicación a la que se destine.

Un microprocesador es un sistema abierto con el que se puede construir una computadora con las características que se deseen, acoplando los diferentes tipos de módulos necesarios.

Un microcontrolador es un sistema de arquitectura cerrada ya que todos los elementos necesarios para su funcionamiento correcto se encuentran dentro de el, además no pueden ser modificados estos elementos.
Un microcontrolador es un circuito integrado, el cual esta compuesto por algunos componentes similares a los de una computadora, como son UCP, memoria, entradas y salidas, etc. Es considerado de arquitectura cerrada ya que los dispositivos que necesita para su funcionamiento se encuentran internos en el.

1.3.2. Arquitectura interna de un microcontrolador.

Las partes principales de un microcontrolador son:
1. Procesador.
2. Memoria volátil para contener el programa.
3. Memoria de lectura y escritura para guardar los datos.
4. Líneas de E/S para los controladores de periféricos:

   - Comunicación en paralelo.
   - Comunicación serie.
   - Diversas puertas de comunicación (bus I2C, USB, RF, etc.

5. Recursos auxiliares:

   - Circuito de reloj.
   - Temporizadores.
   - Perro guardián (watchdog).
   - Convertidores Analógico/Digital (A/D) y Digital/Analógico (D/A).
   - Comparadores analógicos.
   - Protección ante fallos de alimentación.
   - Estado de reposo o de bajo consumo.

El microcontrolador utilizado en este trabajo es el ATMEGA8535 de la familia AVR, que es fabricado por la empresa ATMEL, este microcontrolador tiene las siguientes características:

   - Arquitectura Harvard-RISC, en ella las instrucciones se ejecutan, en su mayoría, bajo un ciclo de reloj. Debido a esta característica es necesita especificar que la frecuencia del oscilador es la frecuencia real del CPU, ya que con otras marcas de microcontroladores no sucede lo mismo.
   - Treinta y dos registros de trabajo de propósito general, conocidos como acumuladores en otros microcontroladores, estos tienen acceso directo a la ALU (Unidad Lógica Aritmética) y a la memoria SRAM.
   - Rango de operación de 2.7 a 6.0 volts.

- Gran cantidad de periféricos incluídos, como temporizadores, contadores, convertidores Analógico-Digital, comparadores analógicos, receptores y transmisores seriales y reloj de tiempo real, entre otros.
- Comparadores analógicos.
- Memoria de programa tipo FLASH con capacidad de 8 Kbytes autoprogramable en sistema, es decir sin tener que extraerlo de la tarjeta donde realiza su función.
- Memoria de datos SRAM con capacidad de 512 bytes y EEPROM con capacidad de 512 bytes dentro del chip.
- Tiene la posibilidad de ser programado en lenguaje C/C++, así como en lenguaje ensamblador.

1.3.3. Arquitectura del microcontrolador AVR.

Todos los miembros de la familia AVR tienen el mismo núcleo básico, la diferencia entre ellos se encuentra en la cantidad de memoria de programa y de datos, el número de puertos y de periféricos que contienen. La figura muestra el diagrama a bloques de la arquitectura del AVR .A continuación se explica de manera generalizada los bloques más importantes:

- Memoria SRAM (STATIC RAM). La memoria SRAM incluida en el microcontrolador se utiliza para almacenamiento de datos volátiles (variables). La longitud de cada palabra es fija (de ocho bits) pero la cantidad de palabras totales en la memoria (tamaño de memoria) varía de un dispositivo a otro. En la SRAM también se mapean los registros de propósito específico del microcontrolador, generalmente en las primeras localidades de memoria. La pila o Snack, puede programarse para comenzar en cualquier localidad del espacio de memoria SRAM.
- Memoria de programa FLASH. Se encarga de almacenar permanentemente las instrucciones que conforman el programa del microcontrolador, aunque también puede utilizarse para guardar datos permanentes, como constantes (número PI por ejemplo), caracteres, etc. Esta memoria es de tipo reprográmable con alta velocidad de acceso. Para programar el microcontrolador se requiere de un aparato externo especial.
- Contador de programa. Es un registro que se encarga de contener y actualizar la dirección de la próxima instrucción a ejecutarse. Se encuentra alojado en un espacio especial de la SRAM.

- Registro de estado y de control. En este registro, que se encuentra en la SRAM, se mantiene el estado actual del microcontrolador: algunos bits llamados banderas, indican hechos tales como el signo de la última operación realizada, si hubo acarreo en la última resta o suma, etcétera.

- Registros de propósito general. Son registros de ocho bits, que pueden ser utilizados como operandos en las operaciones aritméticas y/o lógicas, como fuentes o destinos de datos en operaciones de transferencia de datos dentro del microcontrolador y también permiten el acceso a memoria SRAM, EEPROM y FLASH.

- ALU. Es un circuito digital que realiza operaciones aritméticas y lógicas básicas, tomando como datos el contenido de los registros de propósito general o la memoria SRAM.

- Memoria EEPROM. Es un tipo de memoria de almacenamiento de datos de manera permanente, es decir, los datos son mantenidos áun si el dispositivo se encuentra sin energía. Tiene la facilidad de poder cambiar en cualquier momento el valor de los datos mediante código.

- Periféricos. Son circuitos electrónicos de propósito específico, realizan tareas como temporizaciòn, conversiones (analógico-digital) y comunicaciones seriales, entre otras.

En la figura 1.13 se muestra el diagrama de la arquitectura del microcontrolador AVR.

1.4. Protocolo RS-232.

1.4.1. Definición de interfaz.

Una interfaz puede ser definida como el dispositivo de conexión necesario para conectar dos dispositivos, en donde existen conectores, señales eléctricas y de sincronismo dentro de los circuitos de este conector, una codificación y un protocolo que permita, que estos elementos puedan comunicarse entre ellos para poder intercambiar información.

1.4.2. Norma RS-232.

La interfaz designa una norma para el intercambio en serie de datos binarios entre un equipo terminal de datos y un equipo de terminación del circuito de datos.

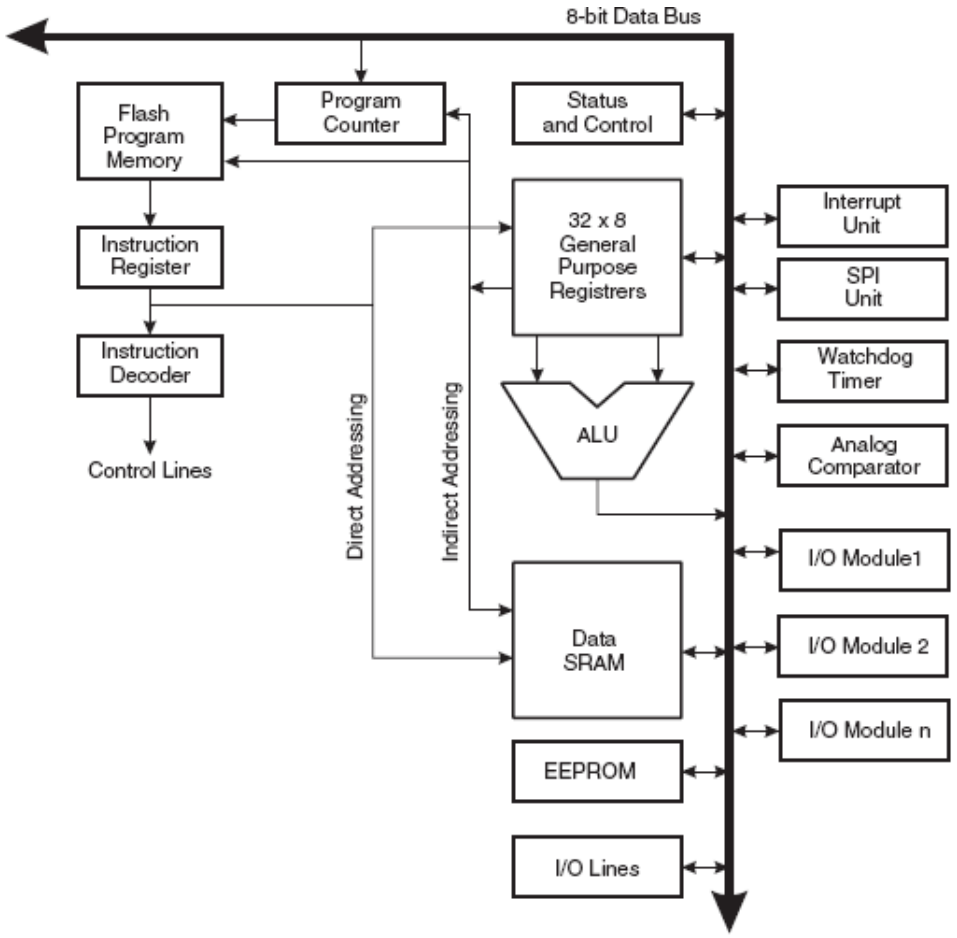

Figura 1.13: Diagrama de la arquitectura del microcontrolador.

Esta norma fue elaborada por la EIA (Electronics Industry Association) la cual define la interfase mecánica, los pines, las señales y los protocolos que debe cumplir la comunicación serial. Esta norma establece que los datos pueden ser transmitidos en grupos de 7, 8 o 9 bits, la velocidad a la que son enviados estos datos es medida en baudios (bits/segundo) y sólo son necesarios dos cables para la comunicación, uno de transmisión y otro de recepción. La comunicación se puede realiza a través del puerto serial de la computadora (DB-9) o a través del puerto paralelo (DB-25).
En este diseño la norma RS-232 permite establecer la comunicación entre la PC y el microcontrolador AVR, a través del puerto serial, el cual esta formado por dos conectores tipo DB-9, los cuales son utilizados para transmitir y recibir datos de la PC al AVR y viceversa. En la Figura 1.14 se muestra los dos tipos de conectores DB-9 (Macho y Hembra).

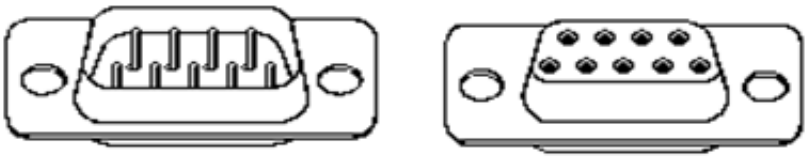

Figura 1.14: Conectores DB-9 (Macho y Hembra)

En estos conectores cada pin tiene una función diferente, en el cuadro 1.3 se muestra el número de pin, el símbolo y la función de cada uno de ellos.

Cuadro 1.3: Descripción de los pines del conector DB-9.

Pin	Símbolo	Nombre.
1	·	Chasis.
2	RxD	Recibe Datos señal de entrada(Receive Data).
3	TxD	Transmite Dados señal de salida(Trasmit Data).
4	RTS	Solicitud de envió señal de salida (Request To Send).
5	CTS	Libre para envió señal de salida(Clear To Send).
6	DSR	Equipo de datos listo señal de entrada (Data Set Ready).
7	SG	Tierra Referencia para señales(Signal Ground).
8	DCD	Detección de portadora señal de entrada (Data) Carrier Detect.
9	RI	Indicador de llamada señal de entrada (Ring Indicador).

En este diseño los bits son transmitidos en grupos de 8 bits, cada bit es enviado uno a uno a la vez, al terminar de enviar el grupo de bits, sigue un bit de paridad que puede ser opcional, con la finalidad de indicar si el número de bits transmitido es par o impar. En este tipo de transmisión sólo son necesarios tres cables, un cable para transmitir los bits, otro para recibirlos y el otro es colocado a tierra.
El modo de transmisión que se utiliza es full-duplex, debido a que permite recibir y transmitir al mismo tiempo. Las señales con las que trabaja el RS-232 son digitales, con un valor de +12V (0 lógico) y -12V (1 lógico). Dependiendo de la velocidad de transmisión empleada, es posible tener cables de hasta 15 metros.
Una vez que ha comenzado la transmisión de un dato, los bits tienen que llegar uno detrás de otro a una velocidad constante y en determinados instantes de tiempo. Debido a esto el RS-232 es asíncrono por carácter y síncrono por bit. Los pines del PIC que se encargan de portar los datos son RXD (recepción) y TXD (transmisión). Para controlar al puerto serie, la CPU emplea direcciones de puertos de E/S y líneas

de interrupción (IRQ). Mediante los puertos de E/S se pueden intercambiar datos, mientras que las IRQ producen una interrupción para indicar a la CPU que ha ocurrido un evento.

Los datos que son enviados de la computadora hacia el AVR son leídos por un circuito integrado llamado USART (Transmisor Receptor Asíncrono Síncrono Universal) que se encuentra incluido en el AVR. Este circuito se encarga de establecer la transmisión-recepción del microcontrolador, esta transmisión-recepción puede ser asíncrona o sincrona, en este diseño se utiliza la de tipo asíncrona.

1.4.3. Transmisión Serie.

La transmisión serie es aquella en la cual la información se envía por medio de bits, estos bits son enviados uno a uno a la vez, es decir uno detrás del otro hacia un determinado punto. Este tipo de transmisión es un poco lenta pero es mucho más confiable y segura, debido a esto es usada para transmisiones a larga distancia que no requieran de altas velocidades de respuesta. Esta transmisión serie esta dividida en dos tipos los cuales son: transmisión serie síncrona y transmisión serie asíncrona.

Transmisión síncrona. En este tipo de transmisión el trasmisor y el receptor se encuentran sincronizados por un reloj cada uno de ellos, el cual permite que los bits se envíen a una velocidad constante que es establecida por los pulsos del reloj.

Transmisión asíncrona. En este tipo de transmisión los bits de datos de un carácter se envían de manera independiente en el tiempo con respecto a otro carácter, precedidos por un bit de arranque y un bit de paro, de tal manera que para esta transmisión cada carácter consta de tres partes: un bit de inicio, bits de caracteres y un bit de paro. El bit de inicio siempre es cero y se utiliza para anunciar que comienza un carácter, el bit de paro siempre es 1, valor que se mantiene por al menos el tiempo correspondiente a un bit para indicar que ha culminado el carácter enviado.

1.4.4. Transmisor Receptor Asíncrono Síncrono Universal (USART).

El Transmisor y Receptor Asíncrono y Síncrono Universal (USART por sus siglas en ingles) es un diseño se comunicación serie altamente flexible que contiene el microcontrolador AVRmaga8535. Sus características principales son:

- Operación Full Duplex (Registros independientes en transmisión y recepción serie)
- Operación asíncrona o síncrona.
- Maestro esclavo en operación síncrona.
- Generador de baudios de alta resolución.
- Soporta tramas de 5, 6, 7, 8 o 9 bits de datos y 1 o 2 bits de paro.
- Generador de paridad par o impar y revisión de paridad por hardware.
- Detección de sobreflujo de datos.
- Detección de error de trama.
- Filtrado de ruido, incluído bit de detección de inicio en falso y filtro digital pasa bajos.
- Tres interrupciones independientes en transmisión completada, registro de espera de transmisión de datos y de recepción completada.
- Multi-procesador en modo de comunicación.
- Doble velocidad en modo de comunicación asíncrona.

1.4.5. Puente H (L298).

Los motores a pasos requieren del cambio de dirección del flujo de corriente a través de sus bobinas en la secuencia apropiada, dicha corriente es muy elevada con respecto a la que el microcontrolador soporta, debido a esto es necesario utilizar un puente H, por cada motor que se tenga que utilizar. El puente H (L298) es un dispositivo que permite controlar motores de CD en rangos que van desde los 9 V hasta 40 V, con consumos de hasta 4 Ampers por medio de señales de baja potencia provenientes de un microcontrolador.

Las principales características del Puente H se muestran a continuación:

- Activación de motores en un rango entre 9 y 40 voltios DC.
- Capacidad para entregar hasta 4 Ampers a la carga.
- Capacidad para activar el giro del motor en cualquiera de los dos sentidos.

El puente H es básicamente un sistema de conmutación controlado por señales digitales de baja potencia, cuando el sistema detecta un 1 digital en una de sus entradas de control y un 0 en las otras, este conecta el motor a la fuente de alimentación con determinada polaridad, si la señal de control que estaba en uno pasa a cero y la de cero pasa a uno el puente H conecta la fuente de alimentación al motor con la polaridad invertida facilitando así el giro en sentido contrario.

Capítulo 2

Diseño y desarrollo del sistema electrónico e interfaz de usuario del robot cartesiano

2.1. Sistema Electrónico.

Básicamente el robot cartesiano esta formado por dos sistemas, sistema electrónico y sistema mecánico, dentro del sistema electrónico se encuentra la interfaz de usuario.

En esta sección se describe de manera detallada el sistema electrónico diseñado para el control del robot cartesiano, en la figura 2.1 se muestra un diagrama a bloque el cual nos muestra las diferentes etapas por las que esta formado el sistema. Nos indica que las coordenadas para el posicionamiento del robot son asignadas por medio de la computadora, estas coordenadas son enviadas hacia el microcontrolador a través del puerto serial de la computadora, el microcontrolador lee estas coordenadas por medio de la USART, después de esto son enviadas hacia el robot para que sean ejecutadas por el mismo.

La interfaz de usuario está desarrollada con el software C++, ésta solicita al usuario las coordenadas para el posicionamiento del robot, éstas coordenadas son enviadas hacia el microcontrolador a través del puerto serie de la computadora.
El acoplador de señales esta formado por un circuito MAX232 y tiene la función de proteger al microcontrolador, ya que convierte los voltajes provenientes de la PC a voltajes adaptables para el microcontrolador, esto es por medio de la norma RS232.
El modelo de microcontrolador AVR utilizado en este diseño es ATmega8535, se encarga de leer las coordenadas asignadas por el usuario esto es por medio de la USART (Transmisión Recepción Asíncrona Síncrona Universal), estas coordenados

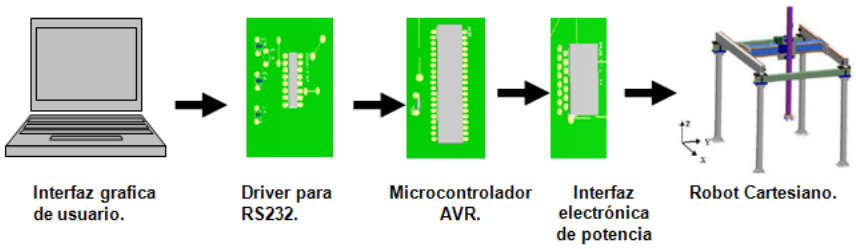

Figura 2.1: Diagrama a bloques del robot cartesiano.

son interpretadas y posteriormente enviadas hacia un circuito puente H.
Las señales provenientes del microcontrolador son recibidas por el puente H, este se encarga de conmutar estas señales por señales con un valor mayor, esto es para poder accionar los motores del robot.
Las señales provenientes del puente H, son trenes de pulsos, las cuales son asignadas a los motores del robot y en base a estas señales el robot se posiciona en las coordenadas asignadas por el usuario.

2.1.1. Interfaz de usuario.

Una interfaz se define como el elemento de conexión necesario para comunicar dos dispositivos, en este diseño se conecta la PC con el microcontrolador a través del puerto serial de la PC y la USART del microcontrolador, esta interfaz esta hecha con el software C++, a continuación se muestra el código del programa.

En esta parte del programa se declaran las librerías que son necesarias para permitir utilizar los diferentes comandos e instrucciones.

include <bios.h>
include <conio.h>
include <dos.h>
include <stdio.h>
include <stdlib.h>

En esta parte del programa se establece el nombre y el valor de las instrucciones utilizadas en este programa.

define COM1 0
define DATA READY 0x100
define TRUE 1

```
define FALSE 0
define SETTINGS ( 0xE0 | 0x03 | 0x00 | 0x00)
```

Estas son las variables utilizadas en el programa, en esta parte del programa se especifican el nombre y tipo de variable que se va a utilizar.

```
    int j;
int cont;
int q1,mq1,q2,mq2;
int dato;
int dato2;
int pasos1,pasos2;
int mili1,mili2;
int cm1,cm2;
int mm1,mm2;
int total1,total2; char t;
```

Una vez declaradas todas las variables se procede a realizar el cuerpo del programa, en esta sección se establece la longitud de las tramas a enviar, la velocidad de transmisión.

```
    bioscom(0, SETTINGS, COM1);
status=bioscom(COM RECEIVE,1, COM1);
```

Después se crea el menú de opciones que se encarga de solicitar al usuario la dirección (izquierda o derecha) de cada uno de los motores paso a paso, el número de centímetros y milímetros que se que el usuario requiere que se desplace el robot, tal como se indica a continuación.

```
    printf("MENU DE OPCIONES");
printf(Ïntroduce la dirección del MOTOR 1 ");
printf("(1) Izquierda");
printf("(2) Derecha");
printf("(3) Stop ");
scanf("d", dato);
if(dato>3)dato=0;
printf(Ïntroduce el núumero de centrimetros del MOTOR 1 ");
scanf("d",pasos1);
if(pasos1>10000)pasos1=0;
printf(Ïntroduce el número de milimetros del MOTOR 1 ");
scanf("d",mili1);
```

```
printf(Ïntroduce la dirección del MOTOR 2 ");
printf("(1) Izquierda ");
printf("(2) Derecha ");
printf("(3) Stop ");
```

En esta parte del programa se establecen los comandos necesarios para identificar en que dirección se va mover y la distancia que se va a desplazar cada uno de los motores a pasos.

```
    //comandos de los motores.

    byte1=0;
if(dato==1)
byte1=byte1|0x01;

else if(dato==2)
byte1=byte1|0x02;

if(dato2==1)
byte1=byte1|0x04;


    //numero de pasos

    byte2=0;
cm1=pasos1*314;
cm2=pasos2*314;
mm1=mili1*31;
mm2=mili2*31;
total1=cm1+mm1;
total2=cm2+mm2;
q1=total1/15;
q2=total2/15;
mq1=cm1mq2=cm2
```

Después de identificar la dirección y la distancia de desplazamiento se procede a enviar estos datos hacia el microcontrolador para que esté posteriormente intérprete estas instrucciones.

```
    status = bioscom(COM SEND,byte1, COM1);
delay(150);
status = bioscom(COM SEND,byte2, COM1);
```

```
delay(150);
printf(".");
```

En la figura 2.2 se muestra la interfaz de usuario, en ella se observa el menú de opciones que permite introducir la dirección de cada uno de los motores a pasos y la longitud de los desplazamientos.

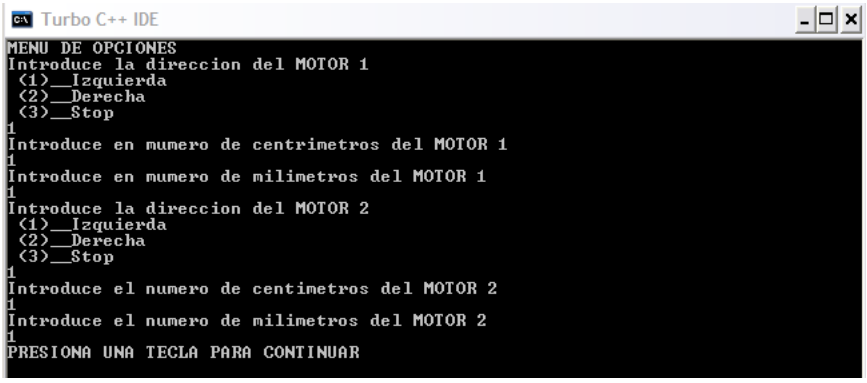

Figura 2.2: Interfaz de usuario.

2.1.2. Puerto serial.

A través del puerto serial de la PC se envían los datos (coordenadas) para posicionar al robot en el lugar solicitado por el usuario, esta comunicación se hace por medio de dos conectores tipo DB9, en estos conectores solo son utilizados los pines dos, tres y cinco. El pin dos es utilizado para recibir datos, el pin tres transmite datos y el pin cinco se encuentra conectado a tierra en la figura 2.3 se muestra como se encuentran interconectados estos conectores.

Uno de estos conectores se encuentra conectado en la PC y el otro en la USART del microcontrolador esta conexión se hace en base a la norma RS232. En la figura 2.4 se muestra la conexión entre la PC y la USART del microcontrolador.

Para que el usuario pueda enviar y recibir datos a través de la PC es necesario configurar el puerto serial, para realizar este trabajo se diseño un programa, el cual permite trasmitir los datos requeridos por el usuario, este programa fue realizado con el software C++, la figura 2.5 muestra una imagen del ambiente gráfico de C++.

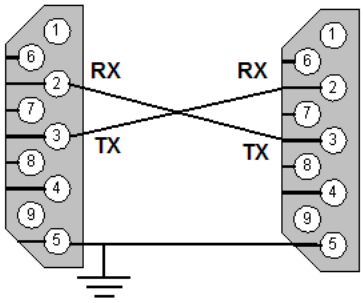

Figura 2.3: Conexión interna de los conectores DB9.

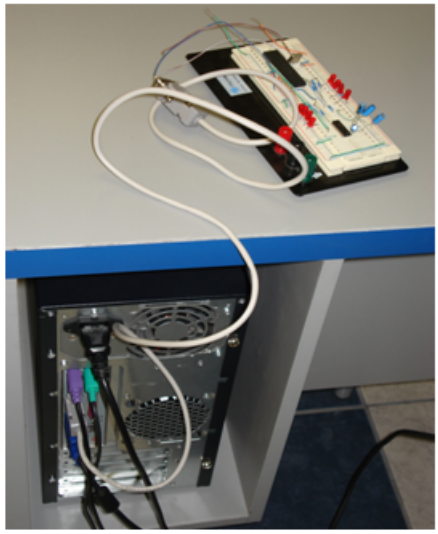

Figura 2.4: Conexión de los conectores DB9.

2.1.3. Programa en C++.

Para que exista una comunicación con la USART del microcontrolador es necesario activar el puerto serial de la computadora, esto se logra a través de la función BIOSCOM el cual tiene la función de hacer trasmitir datos hacia algún dispositivo. Esta función lleva los siguientes comandos (cmd, abyte, port).
La instrucción cmd especifica la longitud de la trama que se va enviar, el BaudRate, los bits de paro y si existe paridad, la longitud de la trama en este diseño es de 8 bits, el BaudRate es de 9600 bps, el bit de paro es uno y no existe paridad.
La instrucción port permite configura las entradas y salidas del puerto a través de la instrucción COM1 o COM2, una ves configurado el puerto esta listo para enviar datos.

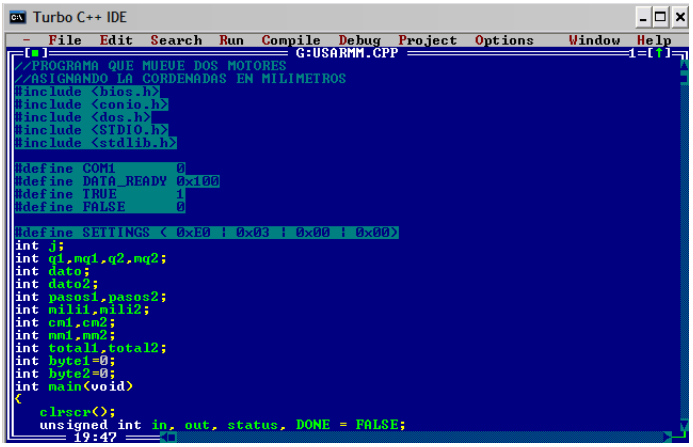

Figura 2.5: Ambiente gráfico de C++.

2.1.4. Driver para RS232.

El puerto serial tiene la característica de otorgar voltajes con valores de +12V y -12V, debido a esto es necesario reducir estos voltajes ya que el microcontrolador soporta un voltaje máximo de 6V, para poder convertir estor voltajes según la norma RS232 se utiliza el circuito MAX232 el cual es utilizado para la transmisión y recepción de información, la figura 2.6 muestra la configuración de sus terminales.

El pin 2, 3 y 5 del conector DB9 se conectan al pin 8, 14 y tierra del circuito MAX232 respectivamente, como se muestra en la figura 2.7. Los conectores que se insertan a la PC y al MAX232 tienen invertidos los pines 2 y 3 debido a esto el receptor de la PC es colocado al transmisor del MAX232 y el transmisor de la PC es se coloca en el receptor del MAX232.

La entrada T1 (pin11) esta conectada al transmisor del microcontrolador y la salida R1 (pin12) esta conectada al receptor del microcontrolador.

2.1.5. Microcontrolador AVR.

Ya terminado el diseño de la interfaz de usuario, se procedió a programar el microcontrolador a través del software AVR Studio, en la figura 2.8 se muestra el ambiente grafico de este software.

Una vez elegidos los diferentes puertos para la conexión de los motores a pasos y los motores de CD, se procedió a programar el algoritmo que se muestra en la figura

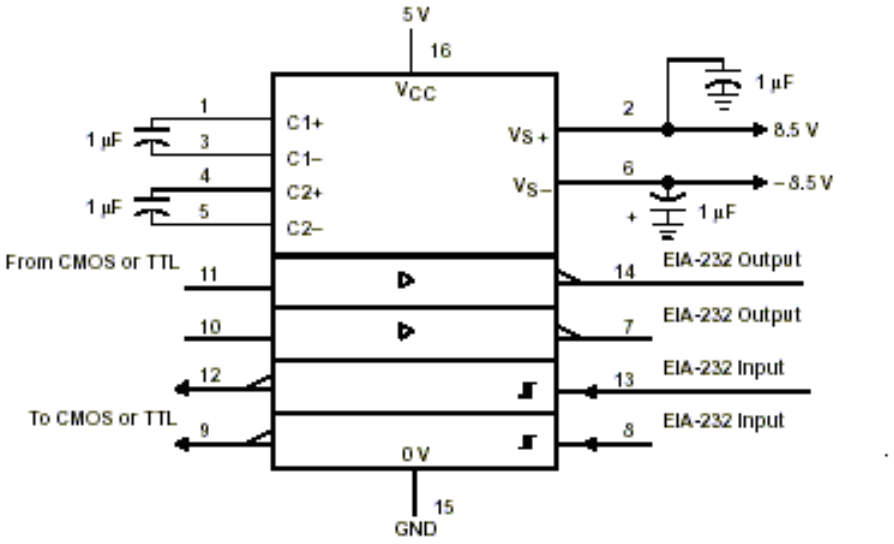

Figura 2.6: Configuración de las terminales del MAX232.

2.9.

De acuerdo con el algorítmo de la Figura 3.40 el microcontrolador debe de estar preparado para recibir un dato, si se recibió es almacenado , debido a esto lo primero que se programa es la USART del microcontrolador. Para ello es necesario limpiar todas las banderas del registro USCRA (Usart Control and Status Register A) y del registro USCRB (Usart Control and Status Register B), los cuales contienen banderas de aviso de que ha ocurrido una transmisión o recepción de datos, por lo que es necesario establecer a cero estos registros y así poder iniciar la transmisión y recepción de los datos sin ningún error.

UCSRA este registro esta formado por 8 bits ver Figura 2.10, los cuales se describen a continuación.

Bit 7 - RXC: USART Receive Complete. Este bit de bandera es de lectura se pone en 1 cuando existe un dato no leído en el receiver búffer, y se pone en 0 cuando el receiver búffer esta vacio es decir no contiene ningún dato no leído.

Bit 6 - TXC: USART Transmit Complete. En este bit de bandera se pone un 1 cuando la trama entera de información en el Transmit Shift Register se ha cambiado a salida. Y no existe nuevo dato actualmente en el transmit buffer (UDR).

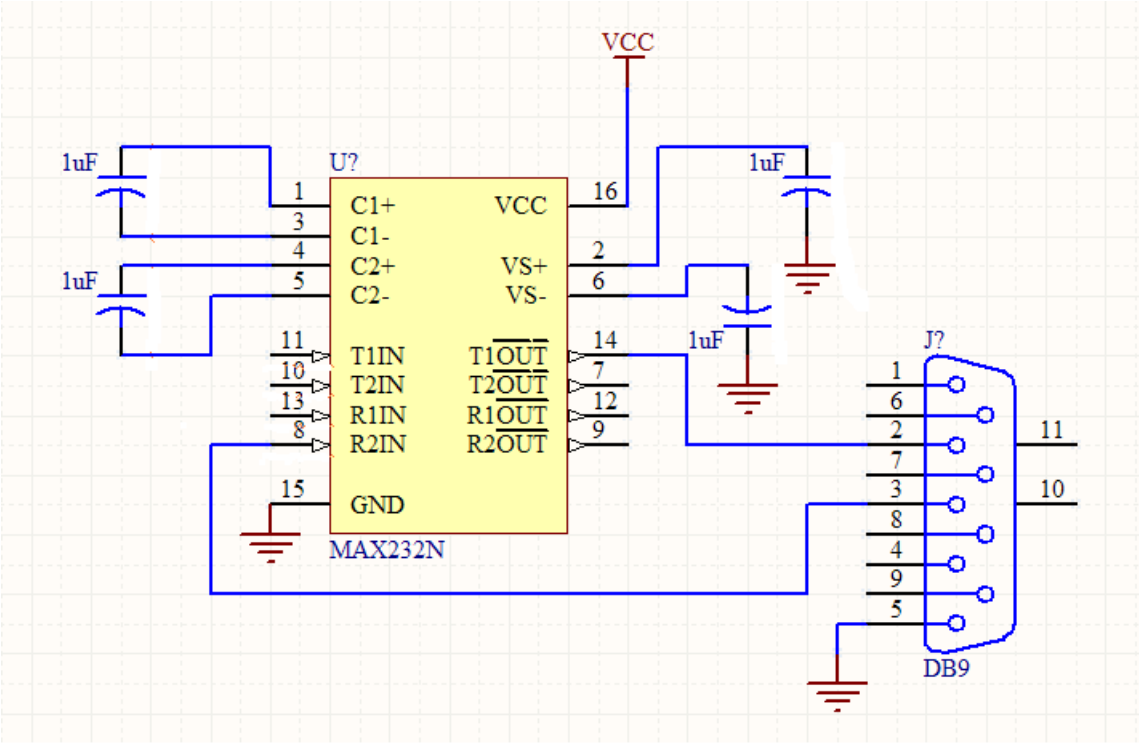

Figura 2.7: Conexión del Circuito MAX232.

La bandera TXC se limpia automáticamente cuando la interrupción de transmisión completa se ejecuta o pueden ser limpiados si se pone un 1 en estos bits. La bandera TXC puede crear una interrupción de transmisión completa.

Bit 5 - UDRE: USART Data Register Empty. La bandera UDRE indica si el transmit búffer (UDR) esta listo para recibir un nuevo dato, si UDRE es uno, el búffer esta vació, y por lo tanto esta listo para poder escribir en el. La bandera UDRE puede generar una interrupción Data Register Empty.

Bit 4 - FE: Frame Error. Este bit se pone en 1 si el siguiente carácter en el receiver buffer tiene un error de trama, cuando se recibe información.. Por ejemplo, cuando el primer bit de paro de el siguiente carácter en el receiver buffer es cero.

Bit 3 - DOR: Data OverRun.
Este bit se pone en 1 si la condición OverRun es detectada. Un datos OverRun ocurre cuando el receiver búffer esta lleno (dos caracteres). Este bit es válido sólo hasta que el reciver búffer (UDR) es leído.

Bit 2 - PE: Parity Error. Este bit se pone en 1 si el siguiente carácter en el

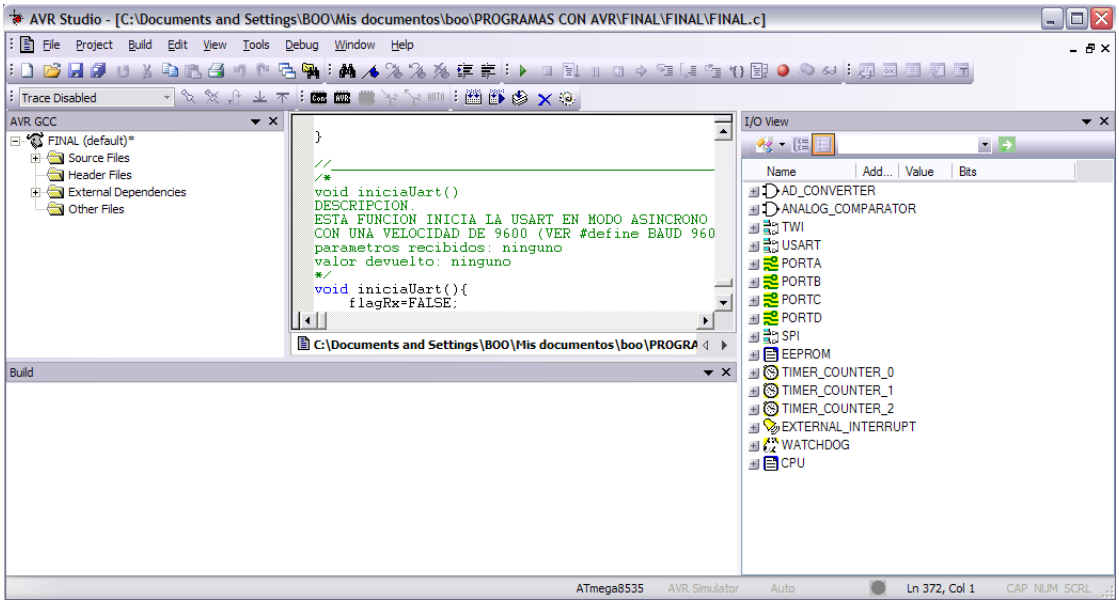

Figura 2.8: Ambiente gráfico del software AVR Studio.

reciver búffer tiene un error de paridad. Este bit es válido sólo sí el receiver búffer es leído. Siempre estará en cero este bit cuando se escribe en USCRA.

Bit 1 - U2X: Double the USART Transmission Speed. Este bit es afectado sólo para la operación asíncrona. Se escribe este bit a cero cuando se usa la operación síncrona.

Bit 0 - MPCM: Multi-processor Communication Mode. Este bit habilita el modo de Multi-processor Communication. Cuando el bit MPCM es escrito a uno, todas las tramas entrantes recibidas por la USART Receiver que no contengan información de dirección serán ignoradas.

USCRB este registro contiene 8 bits (ver figura 2.11) los cuales son configurados de la siguiente manera

Bit 7 - RXCIE: RX Complete Interrupt Enable. Si ponemos en 1 este bit, se habilita la bandera de interrupción de recepción completada, RCX de USCRA

Bit 6 - TXCIE: TX Complete Interrupt Enable. Si ponemos en 1 este bit, se habilita la bandera de interrupción de transmisión completada, TCX de USCRA

Bit 5 - UDRIE: USART Data Register Empty Interrupt Enable. Si

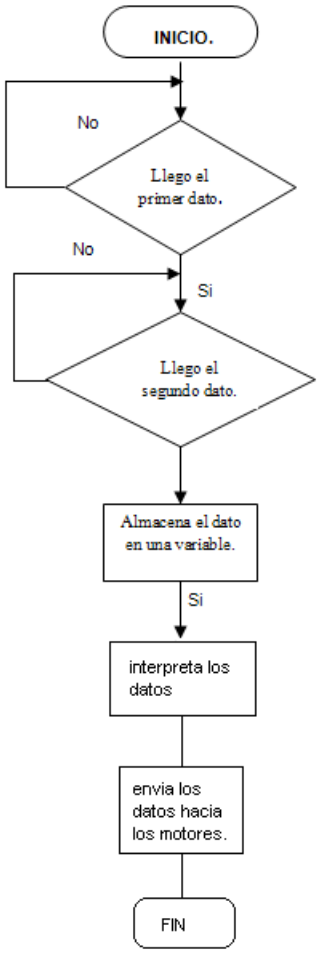

Figura 2.9: Diagrama del algorítmo del microcontrolador.

ponemos en 1 este bit, se habilita la bandera de interrupción de UDRE del registro USCRA.

Bit 4 - RXEN: Receiver Enable. Si ponemos en 1 este bit se habilita el receptor de la USART.

Bit 3 - TXEN: Transmitter Enable. Si ponemos en 1 este bit se habilita el transmisor de la USART.

Bit 2 - UCSZ2: Character Size. Este bit combinado con los bits UCSZ1:0 del registro USCRC determinan el número de bits de la tramas de información a

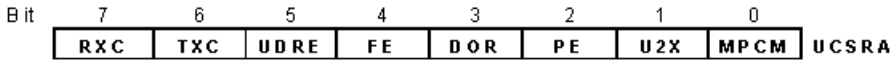

Figura 2.10: Registro UCSRA.

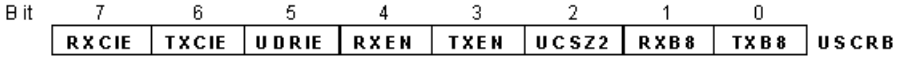

Figura 2.11: Registro USCRB.

recibir o transmitir.

Bit 1 - RXB8: Receive Data Bit 8. Se lee el noveno bit cuando se trabaja con tramas de 9 bits

Bit 0 - TXB8: Transmit Data Bit 8.
Los bits del registro USCRB quedan configurados de la siguiente manera: RXCIE=1, TXCIE=1, UDRIE=0, RXEN=1, TXEN=1, UCSZ2=0, RXB8=0 y TXB8=0.

USCRC este registro contiene 8 bits (ver figura 2.12) los cuales son configurados de la siguiente manera:

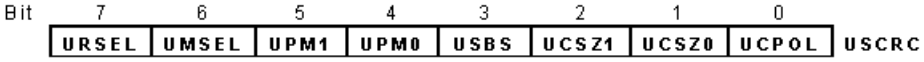

Figura 2.12: Registro USCRC.

Bit 7 - URSEL: Register Select Este bit selecciona el acceso entre los registros USCRC o el UBRRH. Este bit es leído como uno cuando se lee USCRC. El USCRC debe ser uno cuando se escribe en USCRC.

Bit 6 - UMSEL: USART Mode Select Este bit selecciona el modo de operación asíncrono o síncrono como se muestra en el cuadro 2.1.

Bit 5:4 - UPM1:0: Parity Mode Estos bits habilitan y deshabilita el tipo de paridad. Si se habilitada, el transmisor generara automáticamente y enviara la paridad a los datos transmitidos dentro de cada trama. El receptor generara un valor de paridad para los datos entrantes y lo comparara con el valor de UPM0. Si una discrepancia es detectada, el bit de bandera PE en USCRA será puesta en uno. Estos valores de este registro se muestran en el cuadro 2.2.

Cuadro 2.1: Modo de operación.

USMEL	Mode
0	Asynchronous Operation
1	Synchronous operation

Cuadro 2.2: Valores del registro UPM1.

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Bit 3 - USBS: Stop Bit Select Este bit selecciona el número de bits de parada para ser insertados por el transmisor. El receptor ignorara estos valores, los valores se muestran en el cuadro 2.3.

Cuadro 2.3: Valores del registro USBS.

USBS	Stop Bit(s)
0	1-bit
1	2-bit

Bit 2:1 - UCSZ1:0: Character Size Los bits UCSZ1:0 combinados con el bit UCSZ2 del registro USCRB, determinan el número de bits de datos en una trama para la recepción y transmisión. Estos valores se muestran en el cuadro 2.4.

Bit 0 - UCPOL: Clock Polarity Este bit es usado sólo para modo síncrono. Se escribe este bit a cero cuando se usa modo asíncrono. El bit UCPOL determina la relación entre el cambio de los datos de la salida y la muestra de los datos de entrada, esto se muestra en el cuadro 2.5.

En base a esto, los valores del registro USCRC son:
URSEL=1, UMSEL=0, UPM1=0, UPM0=0, USBS=0, UCSZ1=1, UCSZ0=1 y UCPOL=0.

Cuadro 2.4: Valores para el bit UCSZ.

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

Cuadro 2.5: Valores para el bit UCPOL.

UCPOL	TDC (Output of TxD Pin)	RDS (Input of RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

Una vez configurados estos registros se selecciona el Baud Rate de la USART, para seleccionarlo se utiliza el cuadro 2.13 en este diseño se utiliza un cristal de 3.6864 Mhz debido a que deseamos un Baud Rate de 9600 bps y en base a el cuadro 2.13 existe un 0probabilidad que exista un error en la transmisión y recepción de información.

El cuadro 2.13 indica que se coloca en el bit U2X del registro USCRA un cero y en los bits 0:7 de USART Baud Rate Registers (UBRRL y UBRRH) el valor 23.

Después de que la USART se encuentra configurada, el microcontrolador espera a que C++ envié los datos necesarios y este los almacene, estos datos son recibidos por el pin RX y son almacenados en el registro UDR (USART I/O Data Register). Estos datos son leídos e interpretados por el microcontrolador, por lo cual es necesario configurar los diferentes puertos del microcontrolador que son utilizados.

Los puertos del microcontrolador son bidireccionales debido a que pueden ser configurados como entradas o como salidas, cada uno de los puertos contiene tres registros DDR, PORT y PIN. (ver figura 2.14)

Baud Rate (bps)	fosc = 3.6864 MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	95	0.00%	191	0.00%
4800	47	0.00%	95	0.00%
9600	23	0.00%	47	0.00%
14.4K	15	0.00%	31	0.00%
19.2K	11	0.00%	23	0.00%
28.8K	7	0.00%	15	0.00%
38.4K	5	0.00%	11	0.00%
57.6K	3	0.00%	7	0.00%
76.8K	2	0.00%	5	0.00%
115.2K	1	0.00%	3	0.00%
230.4K	0	0.00%	1	0.00%
250K	0	-7.8%	1	-7-8%
0.5K	--	--	0	-7.8%
1M	--	--	--	--

Figura 2.13: Tabla para determinar el BaudRate con el cristal.

El símbolo representa el puerto que se quiere seleccionar del microcontrolador ya que pueden ser A, B, C ó D, el registro DDR es utilizado para poder configurar al puerto como entrada o como salida según se requiera. Si se escribe un *uno* en cualquier bit del registro DDR, este bit se configura como salida, si se escribe un *cero* en cualquier bit, este registro el bit es configurado como entrada.

Para poder activar a un bit como bit de salida es necesario utilizar el registro PORT, debido a que con este registro se activan los bits configurados como salidas.

Para poder activar a un bit como entrada se necesita utilizar el registro PIN, el cual se encarga de activar a los pines que fueron configurados como entradas, en este diseño los puertos A, B y C se encuentran configurados como salidas de la siguiente forma:

- Puerto A DDRA=0Xff;
- Puerto B DDRB=0xff;

Los motores a pasos del eje X y del eje Y son controlados por medio del puerto *A*, a través de este puerto se envían tramas de 8 bits hacia los motores, los primeros cuatros bits (A0, A1, A2, A3) son utilizados para controlar el motor que se encarga del movimiento del eje X y los últimos cuatro bits (A4, A5, A6, A7) son utilizados para controlar el motor del eje Y. Por lo que el puerto *A* queda configurado como

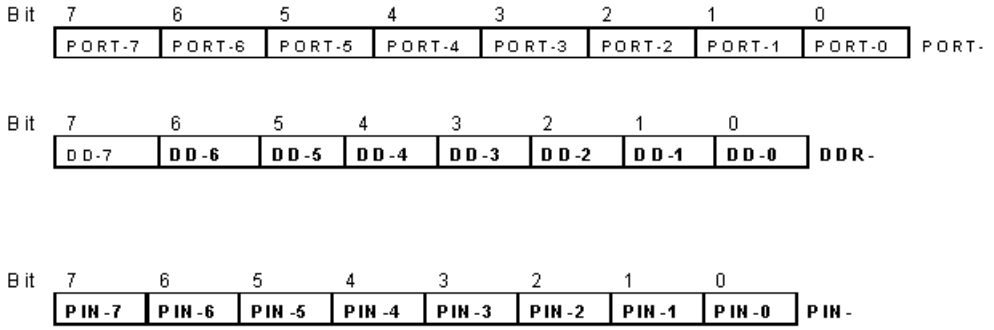

Figura 2.14: Configuración de los puertos del microcontrolador.

se muestra en la figura 2.15.

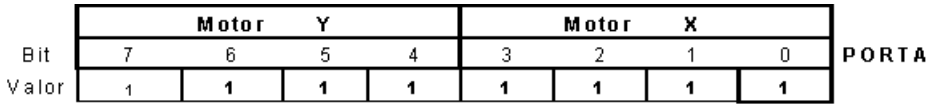

Figura 2.15: Configuración del puerto A.

Los motores del eje Z y del taladro son controlados por medio del puerto *B*, para controlar el motor del eje Z se utilizan los dos últimos bits de este puerto (B7 y B6) y para controlar el motor del taladro se utiliza el primer bit de este puerto (B0) de tal forma que el puerto B queda configurado de la siguiente manera:

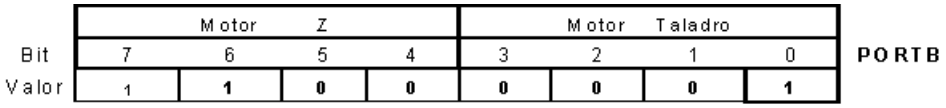

Figura 2.16: Configuración del puerto B.

De esta manera el puerto B esta configurado para poder enviar pulsos hacia los motores del eje Z y del taladro.

2.1.6. Interfaz electrónica de potencia (puente H).

La interfaz electrónica de potencia que requiere el robot cartesiano, es para el manejo de los motores a pasos y los motores de CD, se ha recurrido al puente H,

que es básicamente un conmutador de señales, las señales provenientes del microcontrolador tienen un valor máximo de 5 V, estos voltajes son insuficientes para poder accionar a los motores ya que el voltaje de los motores es de 12 V, debido a esto es necesario utilizar un puente H.
El puente H es alimentado con una fuente de 5 V, tiene la función de accionar a los motores con el voltaje necesario para que funciones.

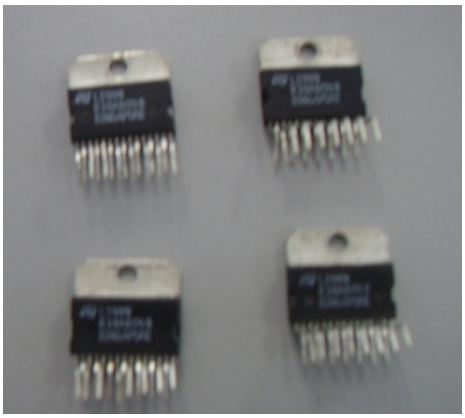

Figura 2.17: Puente H L298.

Capítulo 3

Diseño y construcción del sistema mecánico del robot cartersiano

3.1. Sistema mecánico.

En este capítulo se describen cada una de las piezas que conforman la estructura mecánica del robot cartesiano, empezando por el diseño realizado con ayuda del software AutoCAD y posteriormente por el ensamblaje de las mismas. Para la construcción física se eligieron materiales de acuerdo a la función de cada pieza mecánica. Es importante aclarar en este momento, que la implementación mecánica es solamente para demostrar que las etapas electrónicas y de comunicaciones funcionan, por lo que no hay cálculos relativos a fricción, momentos, etc.
Una vez construído el robot, se realiza la interface electrónica de potencia adecuada para los motores a pasos que se van a utilizar, esto es para manipular los motores a través del microcontrolador mediante el puente H, del cual describe su funcionamiento posteriormente.

3.1.1. Diseño y construcción mecánica.

El software que fue utilizado para realizar el diseño de las piezas mecánicas del robot fue AutoCAD 2004. Es un programa de modelado paramétrico, con el que se puede representar objetos en 3D, además es fácil de usar en aplicaciones de diseño mecánico. Este software dispone de herramientas de diseño con las cuales se pueden realizar las siguientes funciones:

- Crear piezas a partir de operaciones de boceto y predefinidas.
- Combinar piezas externas y piezas auxiliares.

- Generar ensamblajes y subensamblajes.
- Definir escenas para vistas de dibujo.
- Configurar hojas y vistas de dibujo.
- Anotar dibujos para la documentación final.
- Administrar y volver a utilizar datos de diseño.
- Migrar y modificar datos de sólidos heredados.

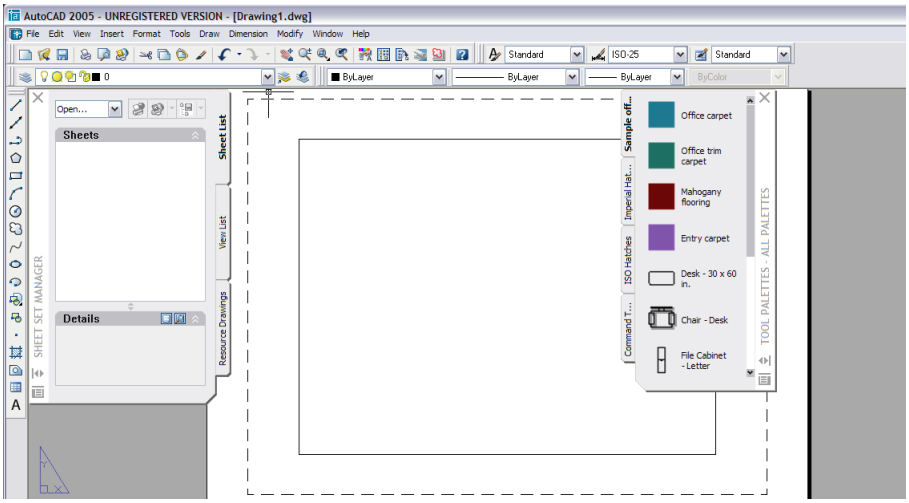

Figura 3.1: Vista del ambientede AutoCAD 2004.

Este software fue de mucha ayuda debido a que con él se diseñaron las diferentes piezas mecánicas, necesarias para la construcción del robot cartesiano, se pretende evitar que existan desajustes en los ensambles de cada una de las piezas.

3.1.2. Descripción de los ejes X, Y y Z.

Mecánicamente el robot cartesiano posee una estructura que realiza tres desplazamientos lineales asociados a los ejes X, Y y Z. Cada uno de estos ejes esta formado por un conjunto de piezas mecánicas, las cuales se describen a continuación.

Soportes del robot cartesiano.
La base es una de las piezas fundamentales, es la encargada sostener a el mecanismo y a los motores con los cuales esta formado el robot. La base debe ser firme, sólida y

resistente debido a que sostiene todo el peso del mecanismo, está formada por cuatro soportes fabricados de aluminio, se decidió utilizar este tipo de material debido a que es un material solidó, ligero y blando, en base a estas características se facilita la labor de su fabricación.

Cada uno de los soportes tiene la forma de un prisma rectangular con las siguientes medidas:
Altura 40 cm.
Base 2.54 cm.

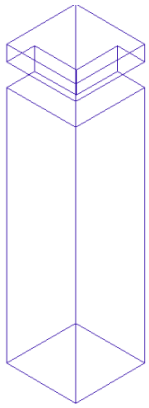

Figura 3.2: Soporte de la base del robot cartesiano.

A cada uno de los soportes se les hizo dos pequeñas ranuras, esto es para que los rieles de la base puedan tener un mejor ajuste cuando se ensamblan y evitar que exista juego en cada una de estas uniones, provocando un movimiento en la base.

Rieles de la base.
La base esta formada por cuatro rieles, los cuales están hechos de aluminio. A través de estos rieles se desplaza el robot para conseguir el movimiento asociado al eje X. En la figura 3.4se observa el diseño de los rieles sujetos a uno de los soportes.

Estos rieles se encuentran ensamblados sobre los cuatro soportes, de tal forma que juntos forman la base del robot, esta unión se hace a través de tornillos los cuales tienen las siguientes medidas:

- Diámetro 3/16 de puldada.
- Longitud 3/4 de pulgada.

En la figura 3.5 se observa como se encuentra ensamblada la base del robot.

Figura 3.3: Vista isométrica del soporte de la base.

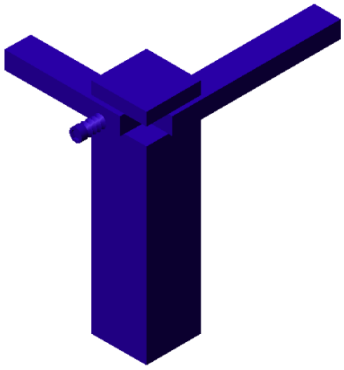

Figura 3.4: Vista isométrica de los rieles.

3.1.3. Descripción del eje X.

Este eje es el encargado de sostener al eje Y y al eje Z, ya que sobre el se desplazan estos dos ejes. Los desplazamientos del eje X y Y pueden ser simultáneos o independientes, dependiendo de lo que se requiera, las partes mecánicas por las que esta formado este eje son las siguientes:

- Tornillos sin fin.
- Tuercas.
- Chumaceras.
- Baleros.
- Spros.

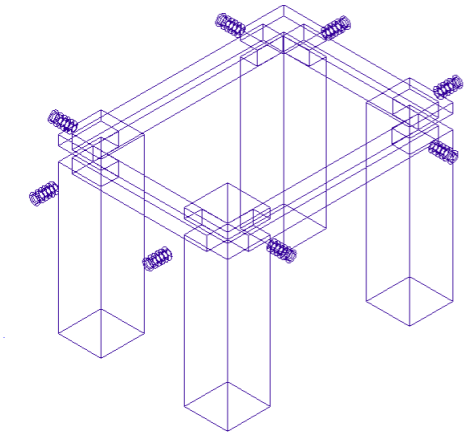

Figura 3.5: Base del robot cartesiano.

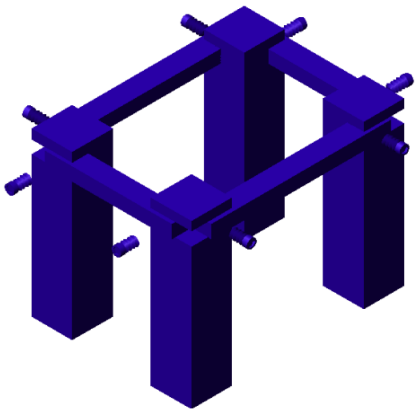

Figura 3.6: Vista isométrica de la base del robot cartesiano.

- Cadena.
- Motor a pasos.

Tornillos sin fin.
Los desplazamientos del robot en este eje son ocasionados por un motor a pasos, este motor se encuentra colocado en uno de los soportes de la base del robot, de tal manera que su flecha esta alineada de frente con la flecha del tornillo sin fin, esto es para que en el momento en que se mueva una de ellas la otra se mueva al mismo tiempo. Para realizar la unión de estas dos flechas se utiliza un cople, este cople esta fabricado de hierro, en el se encuentran incrustadas ambas flechas, en la figura 3.7 se muestra el cople que se utiliza para unir estas flechas.

Como se observa en la figura 3.7, este cople tiene un barreno en cada uno de sus extremos, estos barrenos están hechos para que se pueda incrustar un tornillo, el

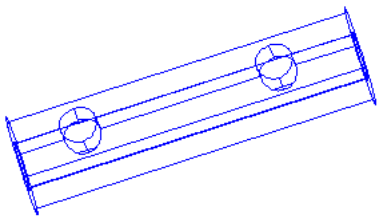

Figura 3.7: Cople para unir la flecha del motor y la flecha del tornillo sin fin.

cual tiene la función de presionar a las flechas del tornillo sin fin y del motor, esto es para evitar que estas flechas se patinen o se muevan y no exista un desplazamiento en cada revolución del motor, las dimensiones del cople son las siguientes:

- Diámetro exterior 5/16 de pulgada.
- Diámetro interior 3/16 de pulgada.
- Longitud 1 pulgada.

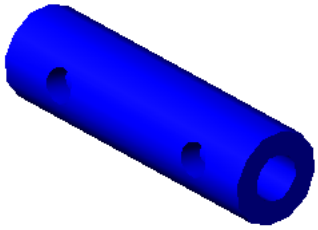

Figura 3.8: Vista isométrica del cople.

Debido a que por cada giro o revolución del tornillo sin fin, se ocasiona un pequeño desplazamiento, la dimensión de este desplazamiento esta en función de dos elementos, uno es el avance del motor por cada pulso asignado a sus bobinas, y el otro es el grosor de cada una de las cuerdas del tornillo sin fin. En la figura 3.9se observa el tipo de tornillo sin fin que se utiliza en este diseño.

El tornillo sin fin es el encargado de desplazar a la herramienta de perforación a la posición requerida por el usuario, debido a que se encuentra sujeto sobre el eje Z, pero este eje esta sujeto sobre el eje Y.

Figura 3.9: Tornillo sin fin del eje X.

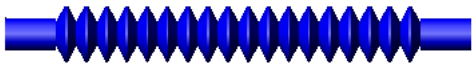

Figura 3.10: Vista isométrica del tornillo sin fin.

Tuercas.
Los tornillos sin fin son los encargados de desplazar al riel a la posición deseada por el usuario, por cada giro o revolución de los tornillos sin fin se ocasiona un desplazamiento, este desplazamiento se ve reflejado en el riel del eje X, para poder lograr este desplazamiento se utilizan dos tuercas, las cuales se encuentran incrustadas en cada uno de los extremos de este riel, como se observa en la figura 3.11.

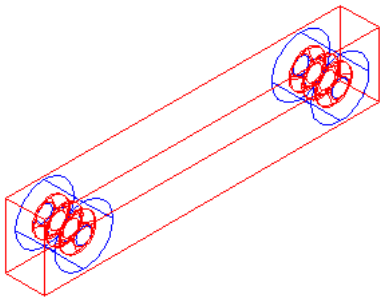

Figura 3.11: Tuercas colocadas en el riel del eje X.

Estas tuercas se encuentran incrustadas a una presión determinada, esto es para evitar que las tuercas se muevan de su posición adecuada, debido a que puede ocasionarse un desajuste en este riel y puede perderse la precisión en cada movimiento realizado. Dentro de estas tuercas se encuentran colocados los tornillos sin fin que

hacen que se mueva el riel del eje X, como se observa en la figura 3.12.

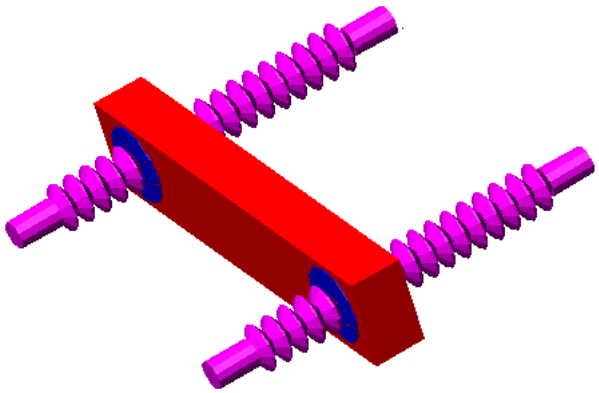

Figura 3.12: Tuercas del riel X.

Estas tuercas están hechas de un material llamado nylamin, se decidió hacer estas tuercas de este material ya que es un material blando y flexible, esto hace más sencilla la labor de su fabricación, ya que facilita el trabajo de desbaste de este material, ya que resulta complicado hacer la cuerda de estas tuercas, las medidas de estas tuercas son las siguientes:

- Diámetro exterior 7/8 de pulgada.
- Diámetro interior 3/8 de pulgada.
- Grosor de la cuerda 6.3mm.
- Longitud 1 pulgada.

Chumaceras.
Cada uno de los tornillos sin fin utilizados en el riel del eje X se encuentra sostenido por dos chumaceras, las cuales están colocadas en cada uno de los extremos de los tornillos sin fin y se encuentran fijas la base del robot cartesiano, la figura 3.14 muestra el tipo de chumaceras que se utilizan.

Estas chumaceras están hechas de un material llamado nylamin, se decidió hacerlas de este tipo de materia, debido a que es ligero, flexible, resistente y se puede desbastar con facilidad.

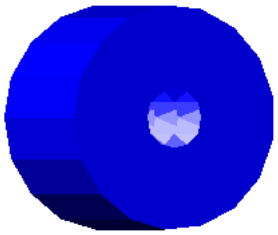

Figura 3.13: Vista isométrica de las tuercas.

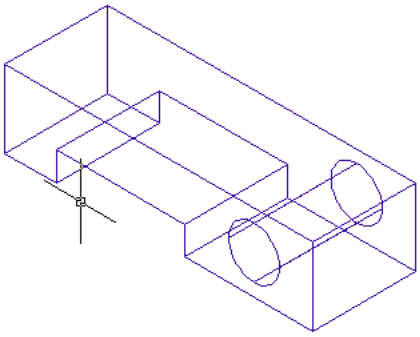

Figura 3.14: Chumaceras de los tornillos sin fin del eje X.

Baleros.
Los baleros son utilizados en este diseño debido a que es necesario reducir la fricción en los movimientos del robot, estos baleros se encuentran posicionados dentro de las chumaceras que sostienen a los tornillos sin fin, las flechas de estos tornillos están incrustadas en el centro de estos baleros, esto es para que los tornillos sin fin tengan un mejor movimiento giratorio reflejando un mejor desplazamiento. Los baleros están hechos de acero ya que es un material solidó y resistente, lo que los hace más resistentes a la fricción, las medidas son las siguientes:

- Diámetro exterior 7 mm.
- Diámetro interior 16 mm.

En la figura 3.27 se muestran los baleros que son utilizados en este diseño.
Spros.
El eje X esta formado por dos tornillos sin fin, los cuales son los encargados de

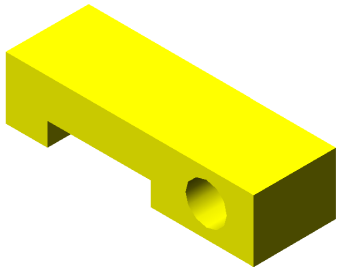

Figura 3.15: Vista isométrica de la chumacera del eje X.

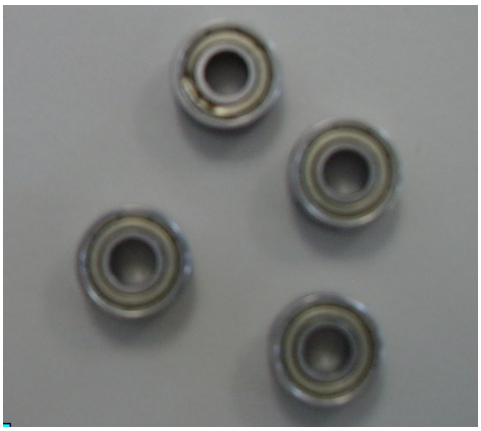

Figura 3.16: Baleros del eje X.

ocasionar el desplazamiento asociado a este eje. Estos tornillos están colocados en cada uno de los extremos del robot, para poder ocasionar un desplazamiento en este eje es necesario mover consecutivamente los dos tornillos sin fin, para poder lograr esto se utilizan dos spros que estan unidos a través de una cadena, para que al momento en que gire uno de ellos el otro tenga el mismo movimiento, en la figura 3.17 se observa como están colocados estos spros en la flecha de los tornillos sin fin.

Estos spros están fabricados de acero templado, este tipo de material es muy resistente y debido a esto es poco probable que se desgaste con facilidad, ya que los spros están en constante movimiento a través de la cadena.

Las dimensiones de estos spros son las siguientes:

- Diámetro exterior de punta a punta 27 mm.
- Diámetro interior 7 mm.
- Paso No 25.

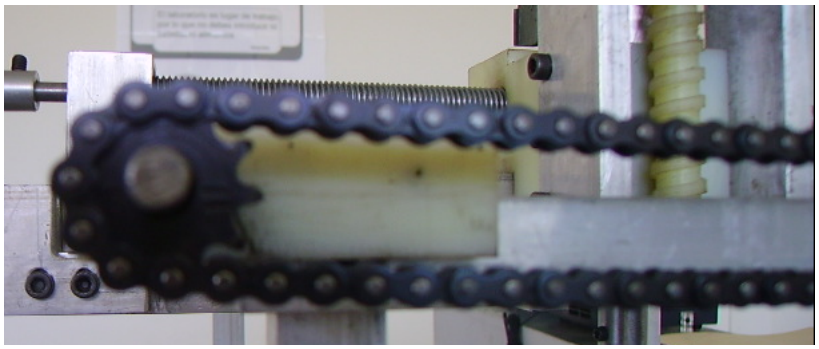

Figura 3.17: Spros.

- Formado por 11 dientes.

Cadena.
La cadena es la encargada de unir a los dos tornillos sin fin esto es para que se muevan al mismo tiempo, esta cadena esta hecha de acero templado, se decidió utilizar esta cadena de este material ya que es un material resistente a la fricción además de ser común y fácil de conseguir. Esta cadena es de paso 25, tiene una dimensión de 25 milésimas de pulgada entre cada eslabón, esta formada por 60 eslabones, y tiene un largo de 65 cm, tal como se observa en la figura 3.18.

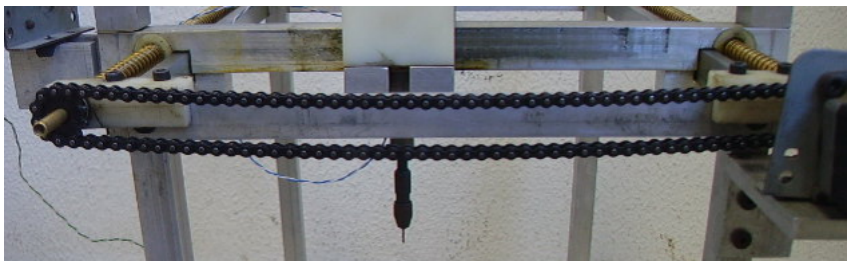

Figura 3.18: Vista de la cadena.

Motor a pasos.
Los motores a pasos son un tipo especial de motor que se caracterizan por su capacidad de precisión para posicionarse en cualquier posición dentro de su rango de operación, para ellos el motor espera un tren de pulsos que se corresponden con el movimiento a realizar. Están generalmente formados por un rotor y un grupo de bobinas en el estator.
Los motores a pasos del robot cartesiano son marca Sanyo a continuación se enumeran sus características.

- Modelo 103H7121-0140.

- Tipo Unipolar.
- 1.8°/paso.
- 18 VCD.

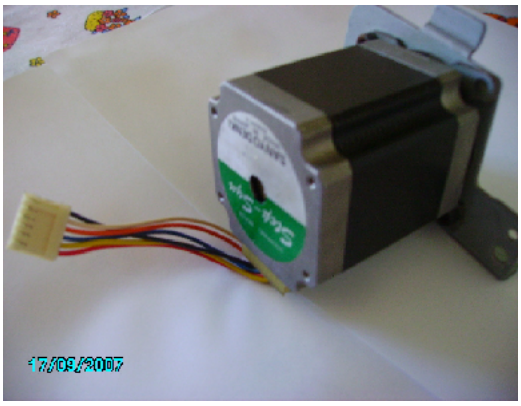

Figura 3.19: Motor a pasos modelo 103H71210140.

3.1.4. Descripción del eje Y.

El eje Y se encuentra sujeto sobre el eje X y este a su vez es el encargado de sostener al eje Z y a la herramienta de perforación, en la figura 3.20 se muestra el eje Y colocado sobre el eje X, las partes mecánicas que lo componen son las siguientes:

- Tornillo sin fin.
- Tuercas.
- Chumaceras.
- Baleros.
- Motor a pasos.

Tornillo sin fin.
Los desplazamientos de este eje son ocasionados por un motor a pasos, el cual se encuentra sujeto sobre el eje X, de tal forma que la flecha del motor esta alineada de frente con la flecha del tornillo sin fin, al igual que en el eje X en este eje existe un cople para poder unir ambas flechas de modo que al girar el motor, el tornillo sin fin produzca un desplazamiento determinado, en la figura 3.7 se puede observar como

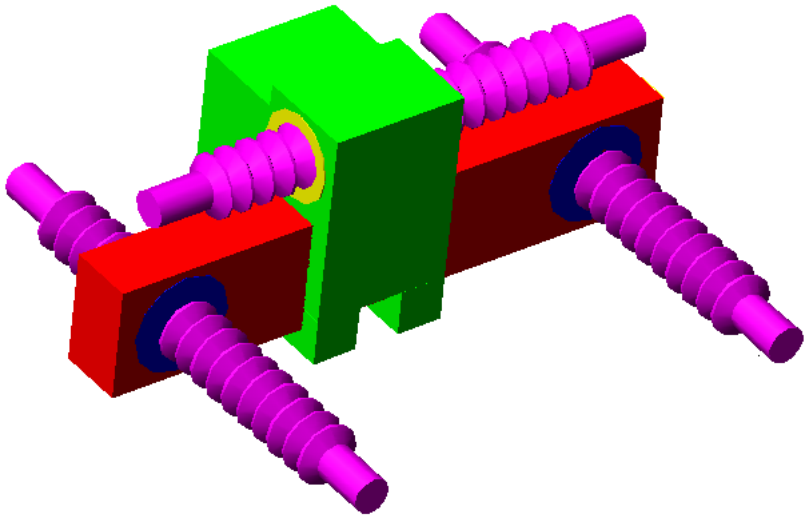

Figura 3.20: Eje X y Eje Y.

se encuentra sujeto el motor sobre el eje X de tal forma que se encuentren alineadas ambas flechas.

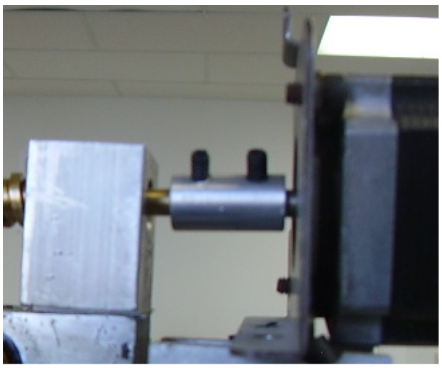

Figura 3.21: Cople para unir las flechas del motor y del tornillo sin fin.

El tornillo sin fin esta fabricado de latón, se decidió utilizar este tipo de material ya que es un material blando y flexible, lo que facilita su fabricación debido a que el desbaste es más sencillo ya que se puede flexionar, las dimensiones son las siguientes:

- Longitud 30 cm.
- Diámetro 1.27 cm.

- Grosor de cada cuerda 0.2 cm.

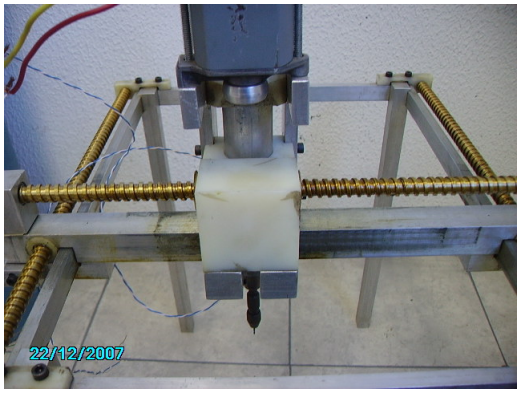

Figura 3.22: Tornillo sin fin del eje Y.

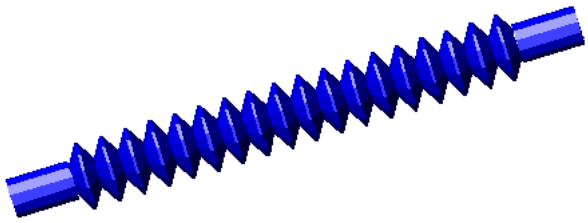

Figura 3.23: Vista isométrica del tornillo sin fin del eje Y.

Tuercas.
Los desplazamientos de este eje son ocasionados por un motor a pasos, debido a que la flecha del motor esta unida con la flecha del tornillo sin fin de esta eje, ya que por cada revolución del tornillo sin fin se produce un desplazamiento determinado, para poder lograr este desplazamiento se utilizan dos tuercas de nylamin, las cuales se encuentran incrustadas en los extremos de este eje tal y como se observa en la figura 3.24.

Al igual que las tuercas del eje X, las tuercas de este eje están hechas de nylamin debido a que es un material resistente, flexible y blando lo que hace más fácil su fabricación. Las medidas de estas tuercas son las siguientes:

- Diámetro exterior 0.875 cm.
- Diámetro interior 0.375 cm.

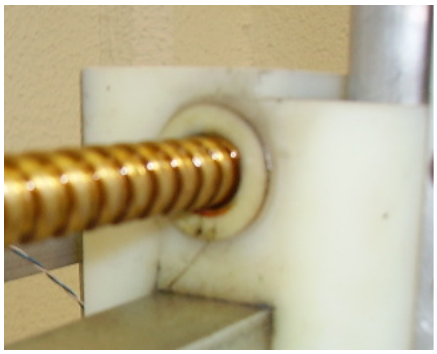

Figura 3.24: Tuercas del eje Y.

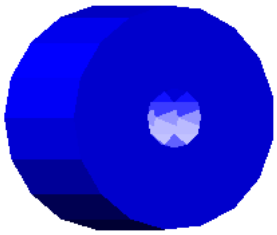

Figura 3.25: Vista isométrica de las tuercas.

- Grosor de la cuerda 0.2 cm.
- Longitud 2.54 cm.

Chumaceras.

Las chumaceras son las encargadas de sostener al tornillo sin fin de este eje, estas chumaceras se encuentran colocadas en cada uno de los extremos del eje X, en ellas se encuentra incrustado un balero, dentro de este balero es donde se introducen las flechas del tornillo sin fin, se decidió utilizar baleros para reducir la fricción y el esfuerzo del motor. Estas chumaceras están fabricadas de aluminio y sus dimensiones son las siguientes:

- Altura 2.54 cm.
- Ancho 2.54 cm.

Baleros.

Los baleros en este diseño son utilizados para obtener un mejor deslizamiento en

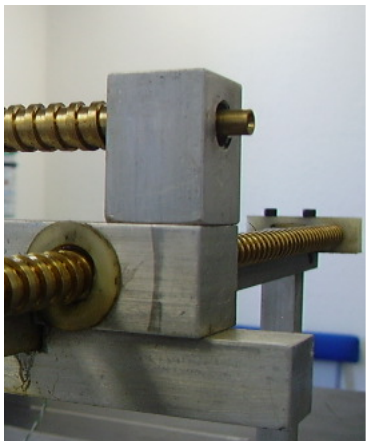

Figura 3.26: Chumaceras del eje Y.

cada uno de los movimientos que realizan los tornillos sin fin, además de reducir la fricción. En este eje se encuentran dos baleros los cuales se encuentran incrustados en cada una de las chumaceras, tienen la función de sostener al tornillo sin fin además de darle un mejor desplazamiento.

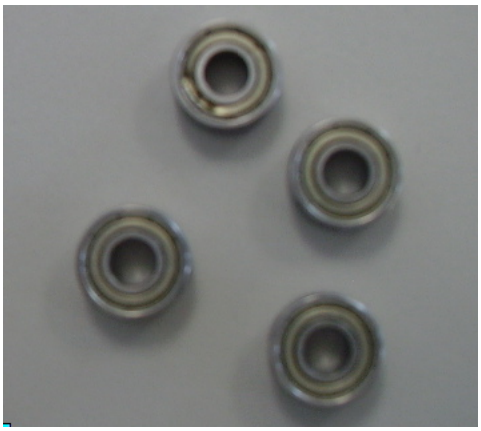

Figura 3.27: Baleros del eje Y.

Estos baleros están fabricados de acero, se decidió utilizar estos baleros debido a que son resistentes para este tipo de trabajo además de ser económicos y fácil de conseguir, las medidas de estos baleros son las siguientes:

- Diámetro interior 0.7 cm.
- Diámetro exterior 1.6 cm.

Motor a pasos.
Este motor tiene las mismas caracteristicas que el del eje X, debido a esto el desplazamiento ocasionado por cada paso es el mismo que el del otro eje.Los tornillo sin fin de este eje tiene las mismas medidas que el tornillo sin fin del eje X.

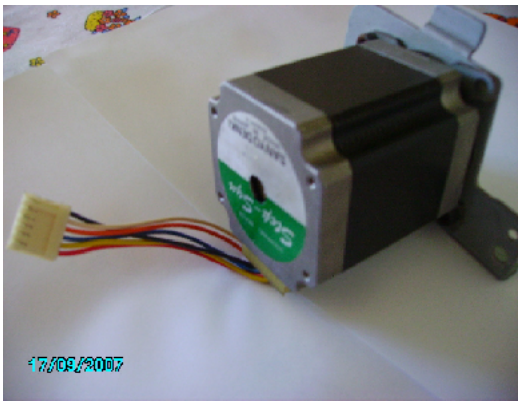

Figura 3.28: Motor a pasos del eje Y.

Este motor se encuentra sujeto sobre el riel del eje X de forma que su flecha esta alineada de frente con la flecha del tornillo sin fin, esto se hace con un cople de esta manera cuando el motor gira el tornillo sin fin ocasiona un desplazamiento de este eje.

3.1.5. Descripción del eje Z.

Este eje tiene la función de bajar y subir a la herramienta de perforación por medio de un motor de CD y un tornillo sin fin, el movimiento se realiza para que se puedan efectuar perforaciones necesarias. Este es colocado en la posición requerida por el usuario por medio de los ejes X y Y, los cuales son movidos a través de los tornillos sin fin y los motores a pasos, las partes mecánicas por la que esta formado son las siguientes:

- Tornillo sin fin.
- Tuerca.
- Motores de CD.

Tornillo sinfín.
El tornillo sin fin se encarga de bajar y subir a la herramienta de perforación , debido que este se encuentra alineado con la flecha del motor de CD de tal forma que

cuando el motor gira el tornillo sin fin ocasiona un desplazamiento, el cual hace que la herramienta de perforación se desplace hacia arriba o hacia abajo dependiendo del sentido de giro de este motor. En la figura 3.29 se observa el tornillo sin fin de este eje.

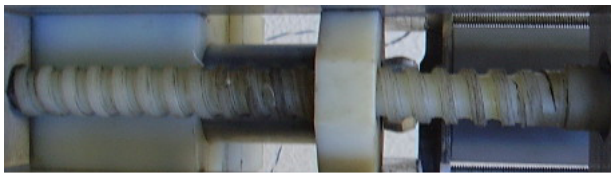

Figura 3.29: Tornillo sin fin del eje Z.

Este tornillo esta fabricado de nylamin, este material es ligero lo cual reduce el peso del mecanismo y del eje Z, las medidas de este tornillo sin fin son las siguientes:

- Longitud 15 cm.
- Diámetro 0.9 cm.
- Grosor de las cuerdas 0.2 cm.

Tuerca.
Dentro de está tuerca se encuentra colocado el tornillo sin fin, a su vez esta tuerca se encuentra fija sobre el mecanismo de la herramienta de perforacion de tal forma que cuando la tuerca es desplazada por el tornillo sin fin, el mecanismo de la herramienta de perforación es desplazado en la misma dirección que la tuerca, al igual que el tornillo sin fin esta tuerca esta hecha de nylamin en la figura 3.30se muestra la tuerca.

Figura 3.30: Tuerca del eje Z.

Motor de CD.
El motor es el encargado de hacer girar al tornillo sin fin, el cual se encuentra fijo sobre una base de aluminio de tal forma que la flecha del motor se encuentra incrustada sobre el tornillo sin fin, cuando gira el motor el tornillo sin fin gira y desplaza a la tuerca hacia arriba o hacia abajo dependiendo del sentido de giro, las características principales son las siguientes:

- Modelo LO9391E.
- 12 V a 1.5 A
- 3200 rpm.

Figura 3.31: Motor de CD.

Cada una de estas piezas es parte importante para el funcionamiento adecuado del robot cartesiano. Una de las prioridades a evitar, es que exista un desajuste entre cada ensamble de las piezas mecánicas que forman al robot, otro detalle es evitar la fricción de los elementos que hacen que se desplace el robot, en base a esto se llegó a la conclusión de utilizar baleros, ya que con estos se obtiene un mínimo de fricción a parte de ser ligeros y pequeños, esto con la finalidad de obtener un desplazamiento mejor.

Capítulo 4

Ensamblaje, Pruebas y Manual de uso

4.1. Ensamblaje del robot.

Se empezó por armar el robot desde la base, esta se encuentra sostenida sobre cuatro soportes los cuales son los encargados de sostener a la herramiena de perforación, cada soporte tiene la forma de un prisma rectangular y sus medidas son las siguientes:

- Altura 40 cm.
- Ancho 2.54 cm.
- Base 2.54 cm.

Los soportes se encuentran unidos por medio de cuatro rieles, sobre dos de estos rieles el robot se desplaza para conseguir el movimiento asociado al eje X, y los otros dos se utilizan para sostener los tornillos sin fin de este eje, en la figura 4.1 se muestra la base.

Las medidas de los rieles son las siguientes:

Rieles del eje X:

- Altura 2.54 cm.
- Ancho 1.27 cm.
- Base 40 cm.

Rieles de los tornillos sin fin:

- Altura 2.54 cm.
- Ancho 1.27 cm.
- Base 30 cm.

Figura 4.1: Base del robot cartesiano.

Una vez terminada la base se prosiguió por colocar las cuatro chumaceras que sostienen a los tornillos sin fin del eje X, estas chumaceras se encuentran colocadas en cada una de las esquinas de la base, en la figura 4.2 se muestra el tipo de chumaceras que se utilizan.

Figura 4.2: Chumaceras del eje X.

Dentro de cada una de las chumaceras se encuentra colocado un balero, en el centro del balero se colocan las flechas de cada tornillo sin fin, se utilizan baleros para reducir la fricción en cada giro y tener un mejor desplazamiento a travéz de los ejes del robot.

Posteriormente se prosiguió a colocar el riel del eje X el cual se encuentra sobre los dos rieles de 40 cm, debido a que sobre estos rieles se desplaza, este riel esta fabricado de aluminio y tiene las siguientes medidas:

- Altura 3.8 cm.
- Ancho 3.8 cm.
- Base 35 cm.

Para que el riel no tenga un desajuste en cada desplazamiento fueron colocados dos deslizadores en la parte inferior de cada uno de sus extremos, tienen la función de evitar que el riel se salga de su posición adecuada y ocasionar desajustes en los desplazamientos, en la figura 4.3 se muestra como estan colocados los deslizadores.

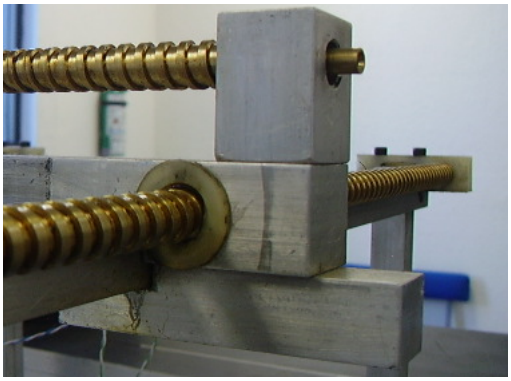

Figura 4.3: Deslizadores del riel del eje X.

Una vez asegurado el riel fueron colocadas dos tuercas en cada uno de sus extremos, están fabricadas de un material llamado nylamin. Dentro de ellas se encuentra colocado los tornillos sin fin, de tal forma que cuando giren, el riel se desplace hacia delante o hacia atrás dependiendo del sentido de giro del motor a pasos.

Posteriormente se prosiguió a colocar el motor a pasos que mueve a los tornillos sin fin del eje X (ver figura 4.4), se encuentra colocado sobre una escuadra de aluminio la cual se encuentra fija sobre uno de los soportes de la base del robot, el motor esta posicionado de tal forma que la flecha del motor se encuentra lineada de frente con la flecha del tornillo sin fin de forma que ambas flechas se mueven al mismo tiempo.

Después de colocar el motor en su posición adecuada se prosiguió a colocar la cadena y los spros la cual se encarga de unir a los dos tornillos, esto es para que se

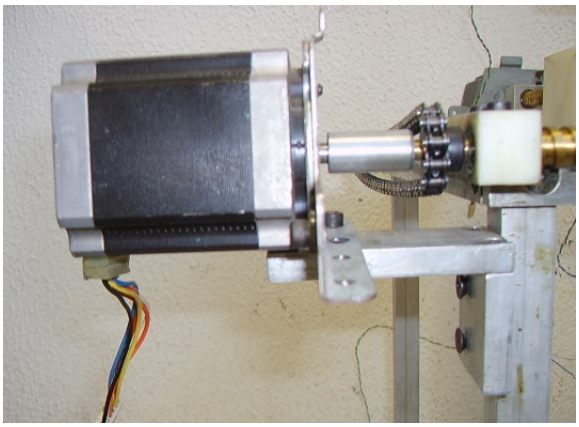

Figura 4.4: Motor del eje X.

muevan simultáneamente.

Una vez ensamblado el eje X se prosiguió a ensamblar el eje Y, este eje se encuentra colocado sobre el eje X (ver figura 4.5), de tal forma que se desplaza sobre el eje X, primero se empezó por colocar el bloque que se desliza sobre el riel, este bloque esta hecho de nylamin.

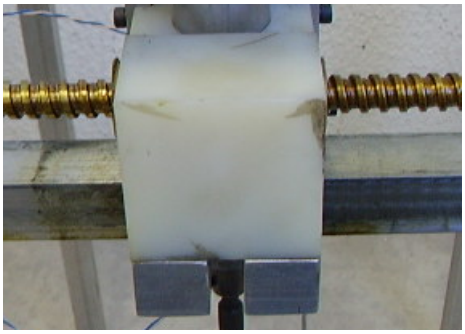

Figura 4.5: Riel del eje Y.

Para evitar que el bloque se mueva de su posición adecuada se le colocaron dos deslizadores en la parte inferior de sus extremos, estos deslizadores están hechos de aluminio, en la figura 4.6 se muetra como estan colocados los deslizadores.

Después de esto se prosiguió a colocar dos tuercas en el bloque de nylamin, estas tuercas se encuentran incrustadas en los extremos del bloque ya que dentro de estas tuercas es donde se encuentra el tornillo sin fin (ver figura 4.7), el cual se encarga de deslizar al bloque hacia la izquierda o hacia la derecha dependiendo del sentido

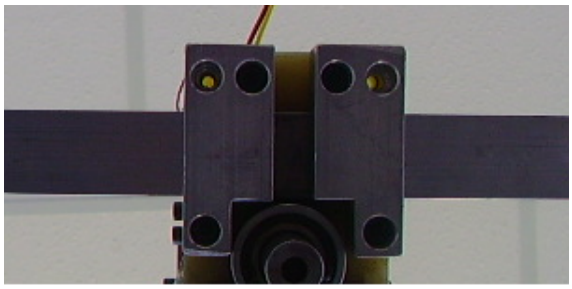

Figura 4.6: Deslizadores del eje Y.

de giro del motor.

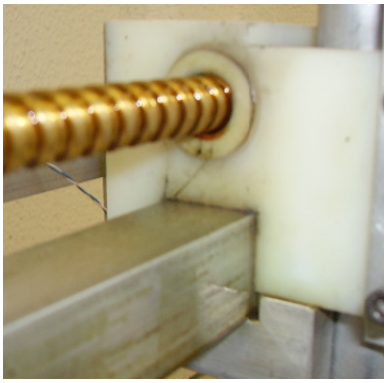

Figura 4.7: Tuercas del eje Y.

El tornillo sin fin esta sostenido por medio de dos chumaceras las cuales se encuentran colocadas sobre el eje X, cada una de ellas esta colocada en los extremos del tornillo (ver figura 4.8), las flechas del se encuentran incrustadas en el centro de un balero, el cual se encuentra dentro de la chumacera. Las chumaceras están hachas de nylamin y tienen las siguientes medidas:

- Altura 3.8 cm.
- Ancho 2.54 cm.
- Base 2.54 cm.

Una vez colocado las flechas de los tornillos en las chumaceras, se prosiguió a colocar el motor a paso que es el encargado de hacer girar al tornillo sin fin de este eje, el motor se encuentra colocado sobre dos escuadras de aluminio las cuales se

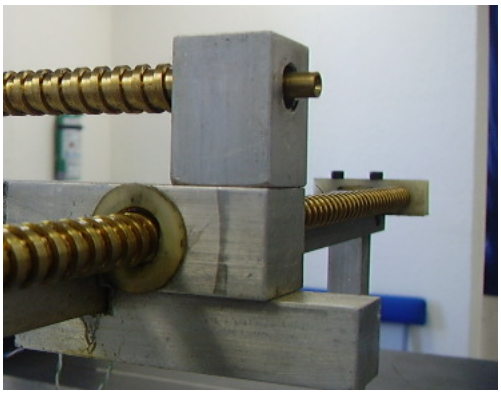

Figura 4.8: Chumaceras del eje Y.

encuentran fijas al eje X, de tal forma que las flechas se encuentren alineadas de frente una con otra, esta unión se realiza por medio de un cople de hierro, en la figura 4.9 se muestra como esta colocado el motor.

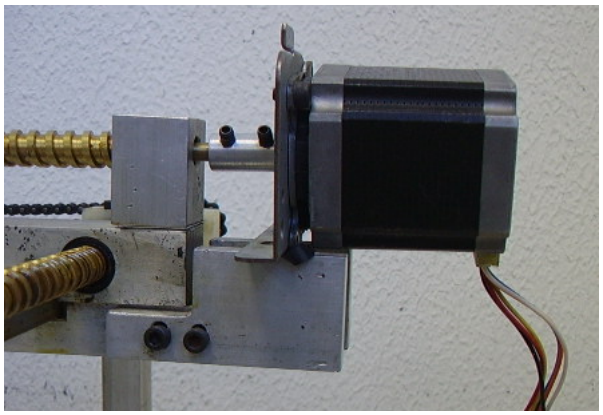

Figura 4.9: Motor a pasos del eje Y.

Una vez ensamblado el eje X y el eje Y se prosiguió a ensamblar el eje Z el cual se encuentra colocado sobre el eje Y, este eje se encarga de bajar y subir a la herramienta de perforación. Se empezó por colocar los rieles por los cuales se desliza el la herramienta (ver figura 4.10), estos rieles están hechos de aluminio y se encuentran atornillados al bloque del eje Y.

Para poder realizar el movimiento de subir y bajar la herramienta de perforación se utilizo un tornillo sin fin y un bloque de nylamin, en un extremo del bloque se encuentra colocada la herramienta y en el otro extremo una tuerca la cual esta colocada dentro del tornillo sin fin, de tal forma que cuando gira, la tuerca se desplaza

Figura 4.10: Rieles del eje Z.

hacia arriba o hacia abajo dependiendo del sentido de giro del motor, en la figura 4.11 se muestra el eje Z del robot cartesiano.

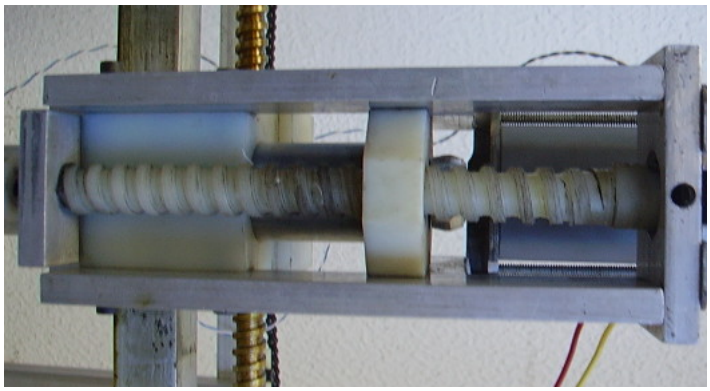

Figura 4.11: Eje Z del robot cartesiano.

El desplazamiento se ocacionado por un motor de CD el cual se encuentra fijo en la parte superior de los rieles (ver figura 4.12), de tal forma de que su flecha se encuentra alineada de frente con la del tornillo sinfín.

De esta forma es como el eje Z quedo ensamblado, después de esto se prosiguió a realizar las pruebas necesarias para comprobar el funcionamiento correcto del robot cartesiano.

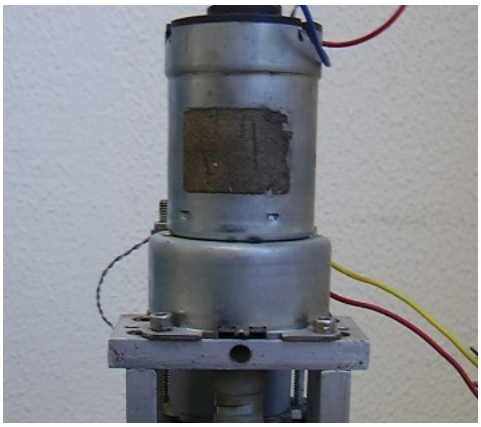

Figura 4.12: Motor del eje Z.

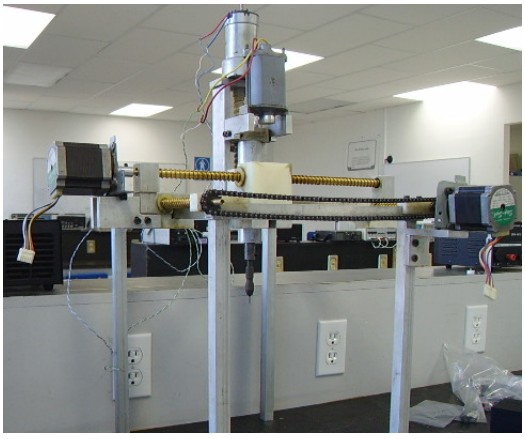

Figura 4.13: Robot Cartesiano.

4.2. Pruebas de posicionamiento.

Estas pruebas se realizaron para comprobar el funcionamiento correcto del robot, consisten en posicionar a la herramienta de perforación en diferentes puntos a partir de un punto inicial, dependiendo del eje que se trate ya que las pruebas de posicionamiento se realizaron por separado. La herramienta se posicionó en un punto inicial, en base a este punto se prosiguieron a realizar los desplazamientos correspondientes.

Primero se empezó por hacer funcionar al eje X, este eje se encarga de desplazar a la herramienta de perforación en dirección hacia el Norte o hacia el Sur dependiendo del sentido de giro del motor.

En la figura 4.14 se muestra la posición inicial, se tomo como base para la primera pureba.

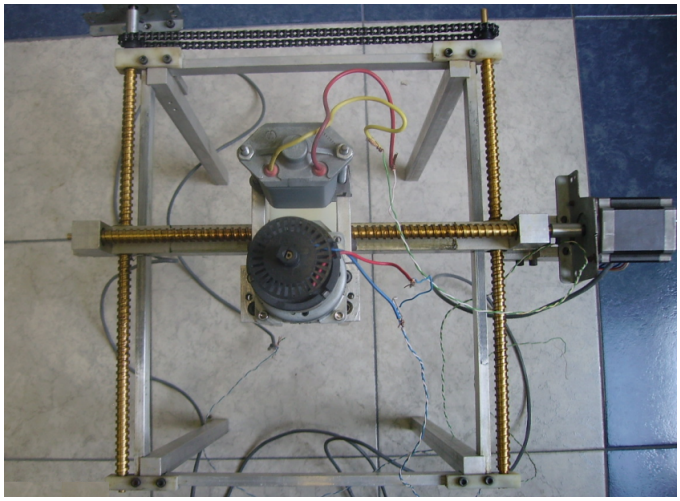

Figura 4.14: Posición inicial de la herramienta de perforación.

a) Norte.

En esta prueba se realizo el desplazamiento hacia el norte partiendo de la posición inicial, en la figura 4.15 se observa como fue este desplazamiento.

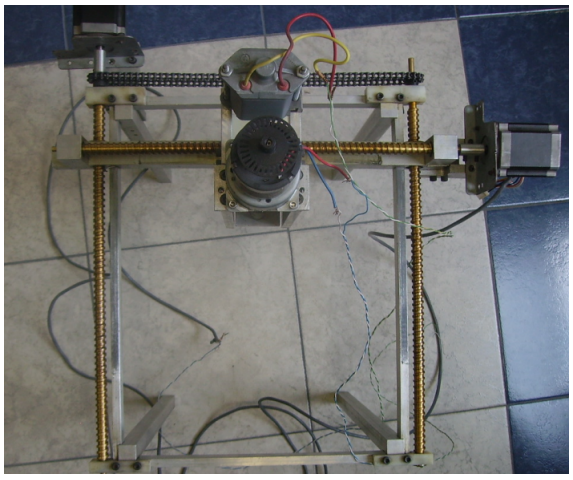

Figura 4.15: Desplazamiento en dirección Norte.

.

.

b) Sur.
En esta prueba se realizo el desplazamento hacia el sur partiendo de la posición anterior, en la figura 4.16 se observa la posición final de esta prueba.

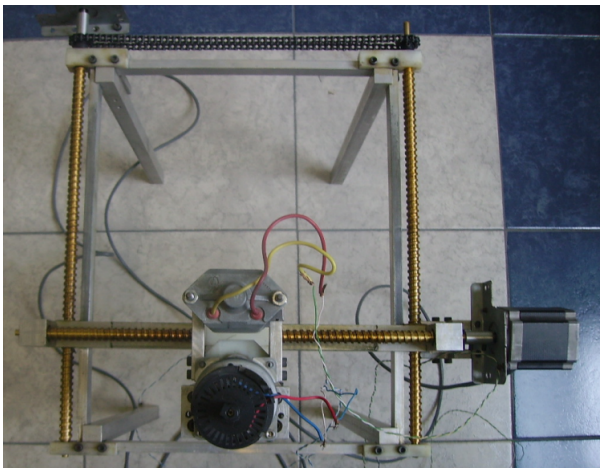

Figura 4.16: Desplazamiento en dirección Sur.

Una vez que se realizaron las pruebas del eje X se prosiguió a hacer funcionar el eje Y el cual se encarga de desplazar a la herramienta de perforación hacia el Oeste o hacia el Este dependiendo del sentido de giro del motor.

c) Este.
Al igual que en las pruebas del eje X se inicio a partir de una posición inicial, en esta prueba el desplazamiento ocurrio en derección hacia el Este, en la figura 4.17 se observa este desplazamiento.

d) Oeste.
En está prueba la herramienta se desplazo en dirección hacia el Oeste partiendo de la posición anterior, en la figura 4.18 se observa el desplazamiento.

Después de realizar la pruebas necesarias a los ejes X y Y se prosiguió a realizar las pruebas necesarias del eje Z, el cual es el encargado de desplazar a la herramienta de perforación hacia abajo y hacia arriba, esto es para que realice la perforación.

e) Perforación.
En esta prueba se desplaza hacia abajo para realizar la perforación y después de un tiempo determinado vuelve a su posición inicial, en la figura 4.19 y 4.20 se observan estas dos posiciones.

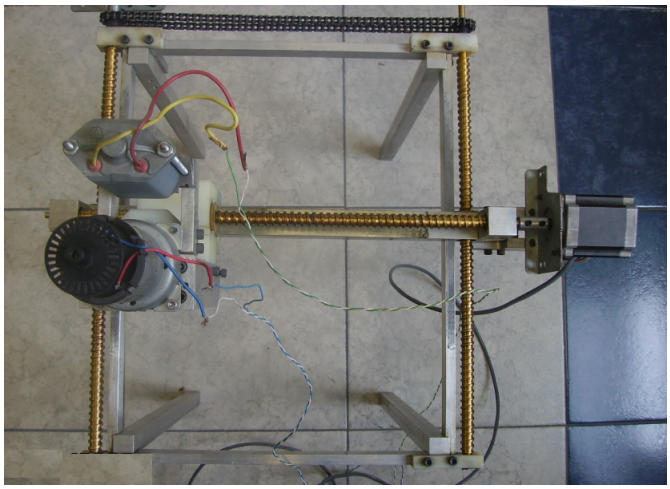

Figura 4.17: Desplazamiento en dirección Este.

f) Movimientos simultáneos.

Esta es la última prueba en la que se pusieron a funcionar simultáneamente los ejes X y Y, después de terminar su rutina cada uno de ellos el eje Z inicia su rutina que consiste en un desplazamiento hacia abajo por un período de tiempo, una vez realizada la perforación regresa a su posición inicial.

4.3. Procedimento para el uso del robot cartesiano.

4.3.1. Descripción de los componentes.

En la siguiente lista se menciona los componentes por los cuales esta formado el robot cartesiano (ver figura 4.21).

1. Motor a pasos del eje X.
2. Motor a pasos del eje Y.
3. Motor de CD del eje Z.
4. Tornillo sin fin del eje X.
5. Tornillo sin fin del eje Y.
6. Tornillo sin fin del eje Z.

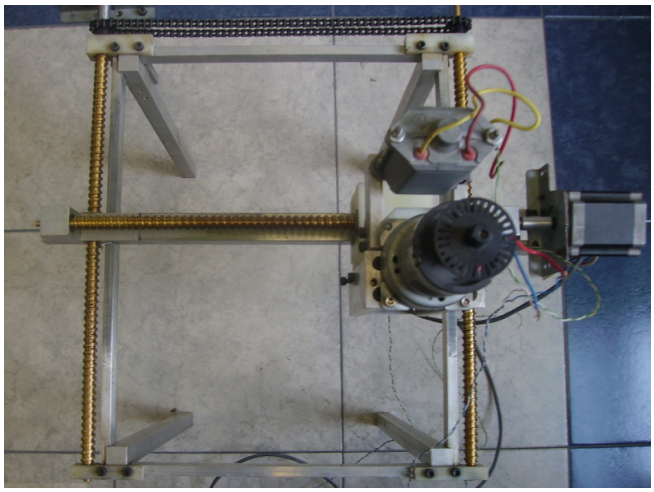

Figura 4.18: Desplazamiento en dirección Oeste.

7. Herramienta de perforación.
8. Spros.
9. Cadena.
10. Motor de la herramienta de perforación.
11. Conectores de alimentación del motor.
12. Conectores de alimentación del motor
13. Base del robot cartesiano.

4.3.2. Conexiones.

Las coordenadas de posicionamiento del robot cartesiano son asignadoas desde la PC, esto es por medio del puerto serial de la PC y la USART del microcontrolador, la interfaz entre estos dos elementos se logra por dos conectore tipo DB9 (hembra y macho) ambos se encuentran interconectados entre si, el conector tipo hembra se coloca en el puerto de la PC y el conector tipo macho se conecta al microcontrolador.

Cada uno de los motores a pasos esta formado por dos grupos de bobinas, debido a esto los motores a pasos tiene seis cables para poder ser conectados, cuatro pertenecen a las bobinas y dos a los comunes. La conexión del motor y el microcontrolador para su funcionamiento debe de ser de la siguiente forma:

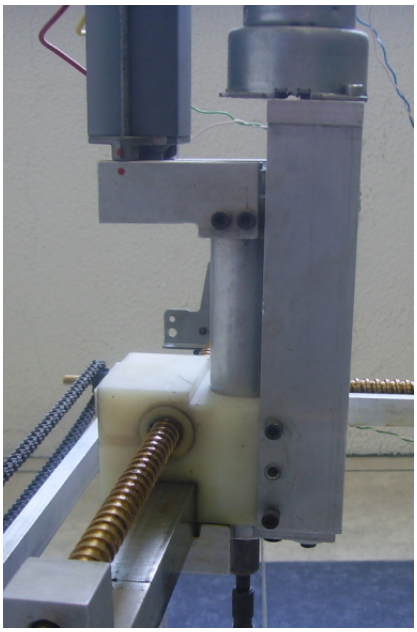

Figura 4.19: Posición inicial.

- cable blanco - comun.
- cable negro - comun.
- cable azul - pin A0.
- cable rojo - pin A1.
- cable naranja - pin A2.
- cable amarillo - pin A3.

De esta forma es como se encuentran conectados los motores a pasos del eje X y del eje Y ya que ambos motores son iguales.

4.3.3. Asignación de Coordenadas.

La asignación de las coordenadas de posicionamiento del robot cartesiano es a travéz de la interfaz de usuario, la cual esta hecha con el software C++, para poder tener acceso a ella es necesario ejecutar el programa en el que se realizo.
Una vez ejecutado se muestra un menu en el cual se deben de seguir las siguientes instrucciones.

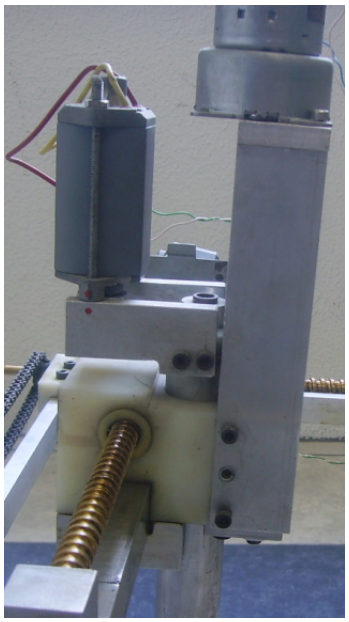

Figura 4.20: Desplazamiento de la herramienta.

1. Asignar la dirección del motor del eje X (izquierda o derecha).
2. Asignar el número de centimetros de desplazamiento del motor.
3. Asignar el número de milimetros de desplazamiento del motor.
4. Asignar la dirección del motor del eje Y (izquierda o derecha).
5. Asignar el número de centimetros de desplazamiento del motor.
6. Asignar el número de milimetros de desplazamiento del motor.
7. Presionar cualquier tecla para iniciar el proceso.

Terminado el proceso se tiene la opción de asignar otras coordenadas o de salir de la interfaz de usuario.

4.3.4. Observaciones para la asignación de coordenadas.

El área para los desplazamientos de la herramienta de perforación del robot cartesiano tiene las siguientes medidas:

- Eje X 15 cm.
- Eje Y 10 cm.

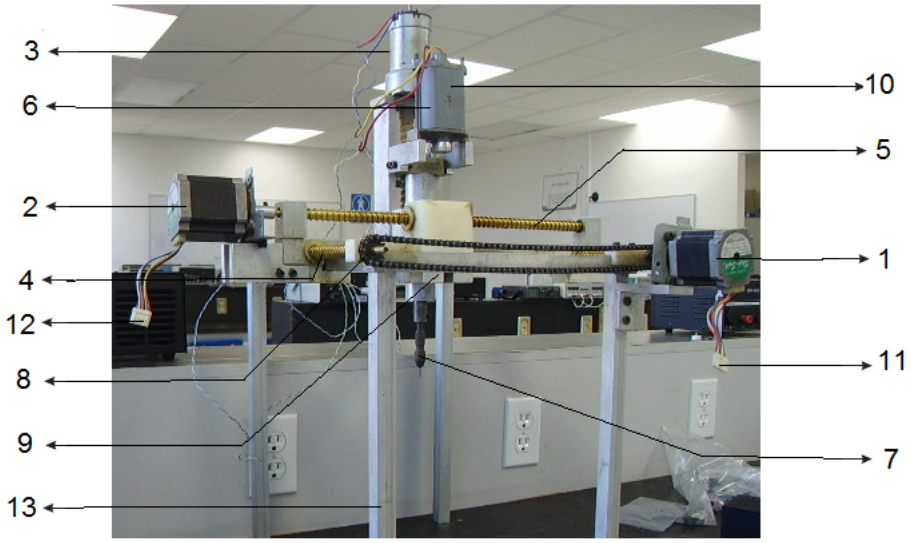

Figura 4.21: Componentes del robot.

Las medidas anteriores solo aplican cuando la herramienta de perforación se encuentre colocada a la mitad de los dos ejes X y Y.

La herramienta de perforación inicia su desplazamiento partiendo de la ultima posición en la que se halla quedado, la longitud máxima que se puede desplazar en una coordenada es de 21 cm en cualquier dirección.

Debido a que el robot no cuenta con un dispositivo para monitorear los desplazamientos de la herramienta de perforación y poder desactivar el circuito de control en caso de algún percance, es necesario verificar la posición inicial del robot antes de asignar las coordenadas esto es para evitar que sobre pase el área de desplazamiento y pueda chocar con alguno de los rieles de la base.

Capítulo 5

Conclusiones y Trabajo Futuro

Conclusiones.

La finalidad de este proyecto es crear un robot cartesiano para la perforación de tarjetas de circuitos impresos, haciéndo uso principal de un microcontrolador ATmega8535 de Atmel. Se utilizo como lenguaje de programación para el desarrollo de la interfaz de usuario al lenguaje C++, es utilizada para asignar las coordenadas de posicionamiento del robot para que realice la perforación.

Con los circuitos electrónicos L298 se logró realizar la etapa de potencia para el control de los motores del robot, debido a que los voltajes provenientes del microcontrolador son insuficientes para hacer funcionar a estos motores.

En las pruebas de comunicación se observó que el microcontrolador tardaba en captar las los datos enviados de la PC debido a que el registro UDR (USART Data Register) del microcontrolador tiene un tiempo de recuperación entre cada envío. La solución a esto fue reducir la velocidad de transmisión entre una cada trama de bits provenientes de la PC.

Cuando se empezó a desarrollar esta tesis se tenía planeado utilizar un microcontolador PIC16F877 de Microchip, cuando se hicieron las pruebas de comunicación entre la PC y el microcontrolador no se pudo realizar esta comunicación debido a esto se optó por utilizar el microcontrolador ATmega8535 de Atmel.

Como se vío en el capítulo cuatro, las diferentes pruebas realizadas fueron satisfactorias para cumplir el objetivo de esta tesis.

.

.

Trabajo futuro.

El software y hardware desarrollados para el robot cartesiano, son factibles a mejoras sobre todo en la comunicación.

- Colocar un dispositivo para monitorear el correcto estado de la herramienta de perforación.
- Actualizar la interfaz para la PC a través del puerto USB.
- Generar las coordenadas de posicionamiento del robot por medio del software AltiumDXP.
- Utilizar motores de corriente directa para generar los desplazamientos del robot.

Apéndice A

Glosario

AVR Nombre que da la empresa ATMEL a sus microcontroladores. Oficialmente AVR no tiene significado.

Bandera. Señal que avisa cuando sucede algún evento o interrupción.

Bit. Unidad mínima de información.

Driver. Circuito integrado que admite o envía datos a un microcontrolador.

Firmware. Bloque de instrucciones de programa que se utiliza para propósitos específicos.

Interfaz. Dispositivo de conexión necesario para conectar dos dispositivos.

Interpolación. Tipo de trayectoria que realiza el robot cuando se desplaza entre un punto y otro.

Interrupción. Petición que se hace al CPU para que realice un evento que acaba de ocurrir.

Paso. Avance o desplazamiento del rotor de cada motor a pasos.

Protocolo. Conjunto de reglas establecidas.

Pulso. Voltaje aplicado en un período de tiempo.

Registro. Conjunto de bits que son considerados como una sola identidad.

Revolución. Giro de 360 grados que realiza el rotor de un motor.

Trama. Grupo o conjunto de bits de diferentes cantidades.

Apéndice B

Código del Microcontrolador

Aquí se muestra el código del programa del microcontrolador AVR.

```
//PROGRAMA PRINCIPAL
//PROGRAMA QUE MUEVE 2  MOTORES A PASOS
// Y 2 MOTORES DE CD A TRAVÉZ DE LA USART
// Y EL PUERTO SERIE
// Y EL PUERTO SERIE
#include<avr/io.h>
#include<avr/delay.h>
#include<avr/signal.h>
#include<avr/interrupt.h>
#include<avr/delay.h>
#define FOSC 3686400// extra
#define BAUD 9600//extra
#define TRUE 1
#define FALSE 0
#define TOTALPASOS 10

unsigned char datoRx;
unsigned char flagRx,flagTx=0;
unsigned char ctr=0,i;
unsigned char m1;
unsigned char m2;
unsigned char m3;
unsigned char paso;

void goToLeftM1();
void goToRightM1();
void goToLeftM2();
```

```
void goToRightM2();
void goToLeftM3();
void goToRightM3();
void goToM4();
void stopM3();
void stopM1();
void stopM2();
void iniciaUart();
void pausa(unsigned char p);
//-------------------------------------------

SIGNAL(SIG_UART_TRANS){
flagTx=TRUE;
}

/*ISR PARA RECEPCION DE DATOS PROVENIENTES DE LA USART

*/
SIGNAL(SIG_UART_RECV){
datoRx=UDR;//llego un dato, almacenarlo en variable datoRx y
flagRx=TRUE;//poner bandera en verdadero para avisar al programa principal
//que ha llegado un dato nuevo

}



//-------------------------------------------
/*
PROGRAMA PRINCIPAL
*/

int main(){
DDRC=0XFF;

iniciaUart();
sei();

DDRA=0xff;
DDRB=0xff;
unsigned char j=1;
```

```
unsigned char a;
unsigned char b;
unsigned char x=0;
unsigned char y=0;
unsigned char motor1,motor2;//número de pasos
unsigned char motor1Listo,motor2Listo;//número de pasos


//-----------------------------------------------
//RUTINA QUE ALMACENA LOS DATOS PROVENIENTES DE LA CPU

PORTA=0xFF;
_delay_ms(1000);
_delay_ms(1000);
_delay_ms(1000);
PORTA=0x00;

while(1){ //while principal
_delay_ms(100);
if(flagRx==TRUE){//llego dato?
j=datoRx;
UDR=j;//regresa dato
x=j;//obtiene comando.
flagRx=FALSE;
while(1){
_delay_ms(100);
if(flagRx==TRUE){//llego dato?
j=datoRx;
UDR=j;//regresa dato.
y=j;  //obtiene comando.
flagRx=FALSE;
motor1Listo=FALSE;
motor2Listo=FALSE;
break;
}
}
motor1=y&0x0F;
motor2=y&0xF0;
motor2=motor2>>4;
}//llego dato?
```

```
//-------------------------------------------------------------
//RUTINA QUE IDENTIFICA LOS DATOS PROVENIENTES DEL CPU

a=x&0x03; //solo interesa motor 1
if(a==1){ //si la entrada es 1 realiza la función goToRightM1();
if(motor1>0){
goToRightM1();
motor1--;
}
else {
motor1Listo=TRUE;
unsigned char t=PORTA;
t=t&0xf0;
PORTA=t;
}
}
else if(a==2){
if(motor1>0){
goToLeftM1();//si la entrada es 2 realiza la función goto
motor1--;
}
else {
motor1Listo=TRUE;
unsigned char t=PORTA;
t=t&0xf0;
PORTA=t;
}
}
else{   // si no es ninguna de las anteriores hace nada
}

b=x&0x0C; //solo interesa el motor 2
if(b==4){
if(motor2>0){
goToRightM2(); //si la entrada es 4 realiza la función goto
motor2--;
}
else{ motor2Listo=TRUE;
unsigned char t=PORTA;
t=t&0x0f;
PORTA=t;
```

```
}
}
else if(b==0x08){//si la entrada es 8 realiza la función goto
if(motor2>0){
goToLeftM2();
motor2--;
}
else{
motor2Listo=TRUE;
unsigned char t=PORTA;
t=t&0x0f;
PORTA=t;
}
}

else{ //si no es ninguna de las anteriores hace nada
}
//------
if(motor1Listo==TRUE && motor2Listo==TRUE){
PORTB=0x80; //encender taladro
pausa(100);
goToRightM3(); //bajar taladro
pausa(100);
pausa(100); //espera a que termine de bajar el taladro
stopM3(); //detiene el motor.
pausa(100); //espera un momento en la perforación
goToLeftM3(); //subir taladro
pausa(100);
pausa(100); //espera a que termine de subir el taladro
PORTB=0x00; //apagar taladro
motor1Listo=FALSE;
motor2Listo=FALSE;
x=0;
y=0;
}



}
return 0;
}
```

```
//RUTINAS DE LOS 3 MOTORES
//-------------------------------------------------------------------------------
void goToLeftM1() //rutina del motor 1 hacia la derecha
{
unsigned char tempo=paso&0x0f;//limpia los 4 últimos bits y deja igual los 4 primero
if(tempo==0x01){ //pregunta si la condición es correcta, si lo es ejecuta la linea s
paso=paso & 0xf0; //limpia los 4 primeros 4 bits y los 4 últimos los deja igual
paso=paso | 0x02;  //activa la bobina 2 del motor 1
}
else if (tempo==0x02){ //pregunta si la condición es correcta, si lo es ejecuta la l
paso=paso & 0xf0; //limpia los 4 primeros bits y los últimos 4 los deja igual
paso=paso | 0x04;  //activa la bobina 2 del motor 1
}
else if (tempo==0x04){ //pregunta si la condición es correcta, si lo es ejecuta la l
paso=paso & 0xf0; //limpia los 4 primeros bits y los últimos 4 los deja igual
paso=paso | 0x08;  //activa la bobina 2 del motor 2
}
else if (tempo==0x08){ //pregunta si la condición es correcta, si lo es ejecuta la l
paso=paso & 0xf0; //limpia los primeros 4 bits y los últimos 4 los deja igual
paso=paso | 0x01;  //activa la bobina 1 del motor 1
}
else {
paso=paso & 0xf0; //limpia los primeros 4 bits y los 4 últimos los deja igual
paso=paso | 0x02;  //activa la bobina 1 del motor 1
}
PORTA=paso;

}

//----------------------------------------
void goToRightM2(){ //rutina del motor 2 hacia la izquierda
unsigned char tempo=paso&0xf0;//limpia los 4 primeros 4 bits y los 4 últimos los dej
if(tempo==0x10){
paso=paso & 0x0f;  //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x80;  //activa la bobina 2 del motor 2
}
else if (tempo==0x80){
paso=paso & 0x0f; //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x40;  //activa la bobina 2 del motor 2
}
else if (tempo==0x40){
```

```
paso=paso & 0x0f; ////limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x20;  //activa la bobina 1 del motor 2
}
else if (tempo==0x20){
paso=paso & 0x0f; //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x10;  //activa la bobina 1 del motor 2
}
else {
paso=paso & 0x0f; //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x80; //activa la bobina 2 del motor 2
}
PORTA=paso;
}
//-----------------------------------------------
void goToRightM1(){ //rutina del motor 1 hacia la derecha
unsigned char tempo=paso&0x0F;//limpia los limpia los 4 últimos bits y los 4 primero
if(tempo==0x01){
paso=paso & 0xf0; //limpia los 4 primeros bits y los 4 últimos los deja igual
paso=paso | 0x08;  //activa la bobina 2 del motor 1
}
else if (tempo==0x08){
paso=paso & 0xf0; //limpia los 4 primeros bits y los 4 últimos los deja igual
paso=paso | 0x04;
}
else if (tempo==0x04){
paso=paso & 0xf0; //limpia los 4 primeros bits y los 4 últimos los deja igual
paso=paso | 0x02;
}
else if (tempo==0x02){
paso=paso & 0xf0; //limpia los 4 primeros bits y los 4 últimos los deja igual
paso=paso | 0x01;
}
else {
paso=paso & 0xf0; //limpia los 4 primeros bits y los 4 últimos los deja igual
paso=paso | 0x08;
}
PORTA=paso;
}
//-----------------------------------------------
void goToLeftM2(){ //rutina del motor 2 hacia la izquierda
unsigned char tempo=paso&0xf0;//limpia los 4 primeros bits y deja igual los 4 último
```

```
if(tempo==0x80){
paso=paso & 0x0f; //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x10;  //activa la bobina 1 del motor 2
}
else if (tempo==0x10){
paso=paso & 0x0f; //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x20;  //activa la bobina 1 del motor 2
}
else if (tempo==0x20){
paso=paso & 0x0f; //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x40;  //activa la bobina 2 del motor 2
}
else if (tempo==0x40){
paso=paso & 0x0f; //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x80;  //activa la bobina 2 del motor 2
}
else {
paso=paso & 0x0f; //limpia los 4 últimos bits y deja igual los 4 primeros
paso=paso | 0x10;  //activa la bobina 1 del motor 2
}
PORTA=paso;
}

//-------------------------------------------------
void goToRightM3() //rutina del motor 3 hacia la derecha
{
unsigned char xxx=PORTB;
xxx= xxx | 0x01;
PORTB=xxx; //pone en 1 el primer bit (bit 0)
}
void goToLeftM3() //rutina del motor 3 hacia la izquierda
{
unsigned char xxx=PORTB;
xxx= xxx | 0x02;
PORTB=xxx; //pone en 1 el primer bit (bit 0)
//PORTB=0x02; //pone en 1 el segundo bit (bit 1)
}
void stopM3() //rutina del motor 3 hacia la izquierda
{
unsigned char xxx=PORTB;
xxx= xxx & 0xFC;
```

```
PORTB=xxx; //pone en 1 el primer bit (bit 0)
//PORTB=0x02; //pone en 1 el segundo bit (bit 1)
}
//---------------------------------------------------------------------

void pausa(unsigned char p){
unsigned char i;

for(i=1;i<p;i++)
_delay_ms(100);
}

void mydelayX(){
int a,b;
unsigned char tempo;
for(a=0;a<100;a++)
for(b*=0;b<100;b++)
tempo=UDR;

}

//---------------------------------------------------------------------
/*
void iniciaUart()
DESCRIPCIÓN.
ESTA FUNCIÓN INICIA LA USART EN MODO ASINCRONO
CON UNA VELOCIDAD DE 9600 (VER #define BAUD 9600 arriba)
parámetros recibidos: ninguno
valor devuelto: ninguno
*/
void iniciaUart(){
flagRx=FALSE;
UCSRA=0x00;//LIMPIA TODAS LAS BANDERAS
UCSRB=0x00;
UCSRB=(1<<RXCIE)|(1<<TXCIE)|(0<<UDRIE)|(1<<RXEN)|(1<<TXEN)|(0<<UCSZ2)|(0<<RXB8)|(0<<
/*INT RX HABILITADA
INT TX DESHABILITADA
INT BUFFER VACIO DESHABILITADA
RX HABILITADO
TX HABILITADO
8 BITS
```

```
*/
UBRRL=23;
UCSRC=(1<<URSEL)|(0<<UMSEL)|(0<<UPM1)|(0<<UPM0)|(0<<USBS)|(1<<UCSZ1)|(1<<UCSZ0)|(0<<
// unsigned char tempo=(0<<URSEL)|(0<<UMSEL)|(0<<UPM1)|(0<<UPM0)|(0<<USBS)|(1<<UCSZ1
// UBRRH=tempo;
}
```

Apéndice C

Código de la interfaz de usuario

Aquí se muestra el código del programa que se realizo para la interfaz de usuario.

```
//PROGRAMA QUE MUEVE DOS MOTORES
//ASIGNANDO LA CORDENADAS EN MILÍMETROS
#include <bios.h>
#include <conio.h>
#include <dos.h>
#include <STDIO.h>
#include <stdlib.h>

#define COM1       0
#define DATA_READY 0x100
#define TRUE       1
#define FALSE      0

#define SETTINGS ( 0xE0 | 0x03 | 0x00 | 0x00)
int j;
int cont;
int q1,mq1,q2,mq2;
int dato;
int dato2;
int pasos1,pasos2;
int mili1,mili2;
int cm1,cm2;
int mm1,mm2;
int total1,total2;
int byte1=0;
int byte2=0;
char t;
```

```
int main(void)
{
for(;;){
clrscr();
unsigned int in, out, status, DONE = FALSE;
bioscom(0, SETTINGS, COM1);
status=bioscom(_COM_RECEIVE,1, COM1);
printf("MENU DE OPCIONES\n");
printf("Introduce la direccion del MOTOR 1 \n");
printf(" (1)__Izquierda\n");
printf(" (2)__Derecha\n");
printf(" (3)__Stop \n");
scanf("%d", &dato);
if(dato>3)dato=0;
printf("Introduce en múmero de centrimetros del MOTOR 1 \n");
scanf("%d",&pasos1);
if(pasos1>10000)pasos1=0;
printf("Introduce en número de milimetros del MOTOR 1 \n");
scanf("%d",&mili1);
printf("Introduce la dirección del MOTOR 2 \n");
printf(" (1)__Izquierda \n");
printf(" (2)__Derecha \n");
printf(" (3)__Stop \n");
scanf("%d",&dato2);
if(dato2>3)dato2=0;
printf("Introduce el número de centimetros del MOTOR 2\n");
scanf("%d",&pasos2);
if(pasos2>10000)pasos2=0;
printf("Introduce el número de milimetros del MOTOR 2\n");
scanf("%d",&mili2);
//forma comando para motores
byte1=0;
if(dato==1){
byte1=byte1|0x01;
}
else if(dato==2){
byte1=byte1|0x02;
}
if(dato2==1){
byte1=byte1|0x04;
}
```

```
else if(dato2==2){
byte1=byte1|0x08;
}
else{
}
//número de pasos
byte2=0;
cm1=pasos1*314;
cm2=pasos2*314;
mm1=mili1*31;
mm2=mili2*31;
total1=cm1+mm1;
total2=cm2+mm2;
q1=total1/15;
q2=total2/15;
mq1=cm1%15;
mq2=cm2%15;

printf("PRESIONA UNA TECLA PARA CONTINUAR\n");
getch();
printf("ejecutando\n");
for(int contador=1;contador<=667;contador++){
byte2=0;
if(q1==0&&q2==0)break;
if(q1>0){
byte2=byte2|0x0f;
q1--;
}
if(q2>0){
byte2=byte2|0xf0;
q2--;
}
status = bioscom(_COM_SEND,byte1, COM1);
delay(150);
status = bioscom(_COM_SEND,byte2, COM1);
delay(150);
printf(".");
}
byte2=0;
byte2=byte2|mq1;
byte2=byte2|(mq2<<4);
```

```
status = bioscom(_COM_SEND,byte1, COM1);
delay(100);
status = bioscom(_COM_SEND,byte2, COM1);
printf("\nOtra vez (N para salir)?");
t=getch();
if(t=='N' || t=='n')break;

}//AQUI
printf("bye\n");
getch();

return 0;

}
```

Bibliografía

[1] Ceballos F(2006).*Programación en C/C++.* (3a Ed) Alfa Omega.

[2] Francisco Javier Ceballos Sierra (2006).*C/C++ Curso de Programación .* (3a Ed) Ra-Ma.

[3] Horenstein ,M. N, (1997).*Ciscuitos y dispositivos.* (2a Ed),Prentice Hall.

[4] López Chau Asdrúbal (2006).*MICROCONTROLADORES AVR, confifuracion total de perifericos.* (1a Ed) Universidad Autónama del Estado de México.

[5] Muhammad H. Rashid,(2004), *ELECTRÓNICA DE POTENCIA, Circuitos, Dispositivos y Aplicaciones*(3a Ed), Prentice Hall.

[6] Tomasi Wayne,(1996), *Sistemas de comunicaciones electrónicas.*Edición México, Prentice Hall.

[7] J. Tocci, Ronald, (2000), *Sistemas digitales, principios y aplicaciones.*(5a Ed), Prentice Hall.

[8] Jose Maria Angulo Asategui, (2009), *Microcontroladores PIC Diseño practico de aplicaciones* (1ra Parte). Mc Graw Hill.

[9] Ignacio Angulo Amusátegui, (2007), *Microcontroladores PIC* (1ra Parte) . Mc Graw Hill.

Sitios WEB.

Atmega8535L, DATA SHEET ATMEL. Recuperado el 5 de Febrero de 2006, de `http://www.atmel.com`

Programador para AVR PonyProg. Recuperado el 10 de Febrero de 2006, de `http://www.lancos.com`

Tutorial sobre motores a paso. Recuperado el 24 de Mayo de 2006, de http://www.todorobot.com.ar/informacion/tutorial

Robot Cartesiano, Recuperado el 28 de Febrero de 2007, de http://www.grupo-maser.com/PAG_ROBOTICA/PDFs/CARTESIANOS.

Inteligencia Artificial (AUTOMATIZACION-CONTROL DE LAZO CERRADO) Recuperado el 18 de Noviembre de 2009, de http://www.dei.uc.edu.py/tai2002/IA/.

Manufactura integrada por computadora robotica, Recuperado el 18 de Noviembre de 2009, de http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r48942.PDF