

UNIVERSIDAD SAN FRANCISCO DE QUITO

Mesa robótica XY para la perforación de placas PCB con conexión USB

Guillermo Burbano B.

Tesis de grado presentada como requisito para la obtención del título de ingeniero eléctrico
electrónico

Quito, Enero 2011

© Derechos de autor

Guillermo Burbano Barzallo

2011

Resumen

La Robótica en la actualidad, se ha desarrollado de tal forma que nos permite realizar cualquier tipo de robot, que realice una o varias actividades. Generando de esta forma un proceso exacto y repetitivo sin perder la calidad de la acción y/o el producto que se esté fabricando teniendo un proceso automatizado.

Dentro de la electrónica, que es un área de estudio muy importante entre las ingenierías, existe una etapa, la cual es la fabricación de circuitos impresos. Estos son placas plásticas, con caminos de cobre, los cuales nos permiten, mediante cinco pasos implementar un circuito. De esta forma se obtiene que los dispositivos eléctricos y electrónicos que queden fijos a la placa, obteniendo seguridad y buena presentación.

En este proyecto, un robot XY para realizar los orificios de los accesorios electrónicos, se pretende que la fabricación de placas PCB, sea un proceso más rápido y efectivo. De esta forma con el uso de este robot automatizado, se pretende que el tiempo de fabricación de una placa PCB sea más rápido, pudiendo usar ese tiempo, en diferentes actividades, tales como investigación, fabricación de placas, entre otras actividades, para fomentar el mejor uso del tiempo.

Este producto se lo va a realizar a escala normal, es decir, que va a tener tamaño y dimensiones reales de acuerdo al tamaño real de las placas que se encuentran en el mercado. Usando un microcontrolador de última tecnología, nos va a permitir controlar y comunicar con la PC mediante el USB, ya que en todas las computadoras se encuentra un conector. Para realizar los movimientos, se usó motores paso a paso, de esta forma se consigue tener precisión, y facilidad de control de todos los movimientos en los diferentes ejes del robot. La interface gráfica, la cual nos va a permitir enviar los datos al microcontrolador para realizar los movimientos deseados para que este se sitúe y taladre, según el circuito impreso realizado en Proteus - Ares o Livewire - PCB Wizzard.

Se puede concluir que este robot, es una gran ayuda, para el estudiante o el profesional que necesita fabricar circuitos impresos.

Abstract

Robotics now a day, has improved as much that we can make any kind of robot that performs one or more activities. Making that the process been accurate and repeatable without losing the quality of the action and / or product being manufactured having an automated process. In electronics, that is a very important area of study among engineering, there is a stage which is the manufacture of printed circuit boards. These are plastic plates with copper paths, which allow us, through five steps to implement a circuit. By these way we find that the electrical and electronic devices are fixed to the plate, giving us security and good presentation. In this project, an XY robot for holes of electronic accessories, it is intended that the PCB plate making, been a process more rapid and effective. Thus using this automated robot, it is intended that the time of manufacture of a PCB been faster so we can use that time in different activities such as research, manufacture of plates, among other activities, to promote better use of time.

This product is going to make a normal scale, it means, that it will be actual size and dimensions according to the actual size of the plaques found on the market. Using a microcontroller of the latest technology will allow us to control and communicate with the PC by USB, because in every computer is a connector. For movements, it is used stepper motors, so it gets to be accurate, and easy control of all movements in different axes of the robot. The graphical interface, which will allow us to send data to the microcontroller to perform desired movements so it can be placed and drill, as the printed circuit board made in Proteus - Ares or Livewire - PCB Wizzard.

It can be concluded that this robot is a great help to the student or professional who needs to make printed circuit boards.

CONTENIDO

RESUMEN	3
ABSTRACT	4
LISTA DE GRAFICOS Y TABLAS	7
INTRODUCCIÓN:	9
OBJETIVOS:	12
CAPITULO 1: MARCO TEÓRICO	13
1.-EL ROBOT X-Y-Z:	13
I. ELEMENTOS ELECTRÓNICOS	14
1) <i>PIC 18F4550</i>	14
2) <i>L293 (Puente H)</i>	15
II. ELEMENTOS ELECTROMECÁNICOS	17
1) <i>Motor Paso a Paso (Mitsumi M43SP-7)</i>	17
2) <i>Motor Paso a Paso (NMB PM35S-048)</i>	19
3) <i>Motor DC Brushless</i>	20
4) <i>CONTROL MOTORES</i>	21
a. Control Motores P.A.P	21
b. Control Motor DC.....	27
III. ELEMENTOS MECÁNICOS	28
1) <i>Rieles</i>	28
2) <i>Tornillo sin Fin</i>	28
IV. PROCESO DE PRODUCCIÓN DE PLACAS PCB	29
1) <i>Generación y simulación del circuito en el software</i>	29
2) <i>Impresión y adhesión del circuito en la placa</i>	32
3) <i>Uso del ácido para retirar el cobre en exceso</i>	34
4) <i>Perforación de los orificios para inserción de los elementos eléctricos</i>	35
5) <i>Soldadura de los elementos</i>	36
V. CONFIGURACIÓN DE LA COMUNICACIÓN USB	37
CAPITULO 2: DISEÑO Y SOLUCIÓN	46
I. DISEÑO Y CONSTRUCCIÓN DEL PROTOTIPO.....	46

II.	DISEÑO DE LOS CIRCUITOS ELECTRÓNICOS Y SIMULACIÓN.....	52
1.	<i>Simulación</i>	52
2.	<i>Diseño de los circuitos</i>	54
III.	PROGRAMACIÓN	56
1)	<i>Explicación Programación del Microprocesador</i>	56
2)	<i>Diseño de la Interfaz gráfica</i>	59
3)	<i>Explicación Programación del Visual Basic</i>	61
IV.	COMUNICACIÓN FÍSICA.....	66
CAPITULO 3: FUNCIONAMIENTO Y ESPECIFICACIONES (MANUAL DE USUARIO)		68
I.	UTILIZANDO PROTEUS - ARES.....	68
II.	UTILIZANDO LIVEWIRE - PCB WIZZARD	75
III.	COMPARACIÓN DE LOS DOS PROGRAMAS.....	80
IV.	UTILIZANDO EL CONTROL MANUAL	82
ESPECIFICACIONES Y CONCLUSIONES		85
I.	ESPECIFICACIONES	85
II.	CONCLUSIONES	86
BIBLIOGRAFÍA		88
ANEXOS		90
I.	PROGRAMACIÓN PIC	90
II.	PROGRAMACIÓN VISUAL BASIC.....	90
III.	DATASHEET PIC 18F4550	90
IV.	DATASHEET LM293.....	90
V.	DATASHEET MOTORES.....	90
VI.	INSTALADOR EASY HID.....	90
VII.	EJECUTABLE APLICACIÓN	90
VIII.	SIMULACIÓN PROTEUS	90
IX.	PRESUPUESTO	90

LISTA DE GRAFICOS Y TABLAS.

Figura 1: Pic18F4550 40 pines.....	15
Figura 2: Puente H, L293D.....	16
Figura 3: Motor paso a paso Mitsumi M43SP-7 y especificación.....	18
Figura 4: Curva Velocidad Vs Torque motor Mitsumi M43SP-7.....	19
Figura 5: Motor paso a paso NMB PM35S-048 y especificaciones.....	20
Figura 6: Curva Velocidad Vs Torque motor NMB PM35S-048.....	21
Figura 7: Motor DC Brushless.....	21
Tabla 1: Ángulos y cantidad de pasos.....	22
Figura 8: rotor y estator de un paso a paso.....	23
Figura 9: hilos motores paso a paso bipolar y unipolar.....	23
Figura 10: Conexión motor paso a paso bipolar.....	24
Figura 11: Conexión motor paso a paso unipolar.....	24
Tabla 2: Tabla de verdad secuencia wave drive.	25
Tabla 3: Tabla de verdad secuencia Full Step.....	26
Tabla 4: Secuencia Half Step.	27
Figura 12: conexión motor DC.....	28
Figura 13: Configuración Rieles - Tornillo.....	29
Figura 14: Tornillo sin fin.....	30
Figura15: Simulación Circuito Proteus.....	31
Figura16: Circuito impreso Proteus.....	31
Figura 17: Simulación en Livewire - PCB Wizzard.....	32
Figura18: Circuito impreso Livewire - PCB Wizzard.....	32
Figura 19: características de impresión Proteus.....	33
Figura 20: Opción para ver los caminos en PCB Wizzard.....	34
Figura 21: Pantalla 1 del HID, cargar nombre del producto.....	38
Figura 22: Vendor Id y Product Id.....	39
Figura 23: configuración de la cantidad de bytes.....	40
Figura 24: opciones de programas y Pic´s.....	40
Figura 25: Confirmación de la generación de códigos.....	41
Figura 26: Programas creados con el HID.....	41
Figura 27: Código generado por el HID para el PBP.....	42
Figura 28: Código generado por el HID para Visual Basic.....	45
Figura 29: Dispositivo conectado con la computadora.....	45
Figura 30: Caja interior para la fuente y placa del controlador.....	47
Figura 31: vista lateral y frontal rieles y soportes.....	48
Figura 32: Plataforma movable.....	48
Figura 33: Unión base y plataforma movable.....	49
Figura 34: Soporte superior y soporte rieles.....	49
Figura 35: Soporte para el taladro y soporte para rieles superior.....	50
Figura 36: Vista lateral soporte de taladro.....	50

Figura 37: Vista lateral parte superior.....	50
Figura 38: Vista frontal completo.....	51
Figura 39: Vista lateral completa.....	51
Figura 40: Diseño Maqueta en 3D.....	52
Figura 41: Diseño Real.....	52
Figura 42: Fotos Maqueta Terminada.....	52
Figura 43: Circuito realizado en Proteus.....	53
Figura 44: Simulación del circuito Eléctrico.....	54
Figura 45: Circuito fuente y controlador de motores.....	55
Figura 46: Lógica de programación del Pic.....	57
Figura 47: Interfaz Figura.....	60
Figura 48: Lógica de Programación de Visual Basic.....	62
Tabla 5: Relación de buffers Pic – VB6.....	63
Figura 49: Archivo .txt Proteus.....	64
Figura 50: Archivo .drl Livewire - PCB Wizzard - PCB Wizzard.....	65
Tabla 6: Movimiento de acuerdo a las coordenadas.....	66
Figura 51: Cable USB.....	67
Figura 52: Conexión del dispositivo.....	69
Figura 53: Configuración Proteus.....	70
Figura 54: Área de trabajo.....	70
Figura 55: Determinación punto de origen.....	72
Figura 56: dispositivos eléctricos situados en la placa.....	72
Figura 57: Generación de caminos placa.....	72
Figura 58: Generación de archivo drill.....	73
Figura 59: archivo generado por Proteus - Ares.....	74
Figura 60: Cargar el Archivo y comenzar perforación.....	74
Figura 61: comparación puntos Proteus - Ares e interface visual.....	75
Figura 61: Confirmación Perforación finalizada.....	76
Figura 62: Configuración distancias Livewire - PCBWizzard	77
Figura 63: Configuración área de trabajo y origen de coordenadas.....	78
Figura 64: Placa Livewire/PCBWizzard.....	78
Figura 65: Generación archivo generado Livewire/PCBWizzard.....	79
Figura 66: archivo generado por generado Livewire/PCBWizzard.....	80
Figura 67: Cargar el Archivo y comenzar perforación.....	80
Figura 68: Comparación puntos Livewire/PCBWizzard e interface visual.....	81
Figura 69: Igualdad Proteus/Ares con Livewire/PCBWizzard.....	82
Figura 70: Igualdad de puntos de los dos programas.....	82
Figura 71: Control Manual.....	84
Figura 72: Confirmación de movimiento.....	84
Figura 73: Botón para ir al punto de inicio.....	85
Tabla 7: Especificaciones RobotXY.....	86

INTRODUCCIÓN:

Un robot XYZ, es aquel robot, capaz de movilizarse en los 3 ejes, de tal forma que puede realizar acciones en el plano XY. Dependiendo del posicionamiento del actuador Z, se puede realizar diferentes acciones, sin perder la precisión con la repetición de las funciones.

Este trabajo se lo realiza para el óptimo desempeño del joven estudiante, o del profesional que necesita realizar circuitos impresos. Para la construcción de los circuitos impresos se debe efectuar cinco pasos, los cuales son encadenados, es decir, que si uno no se realiza, el siguiente no se puede proceder.

Entre los cinco pasos, se ha concluido, tanto por experiencia como por investigación, que el 4to y 5to pasos son los pasos en los cuales uno va a tomar más tiempo. El cuarto paso consiste en realizar la perforación de cada orificio en la placa. El quinto paso consiste en soldar uno por uno cada elemento electrónico que se va a usar en su puesto. Este proceso, como se mencionó antes, es el que más tiempo se va invertir. Por ejemplo, si usamos un PIC16F877A, más un puente H, y los componentes respectivos, hablamos que según el diseño se debe perforar más de 60 orificios, por lo que se pierde precisión al repetir una y otra vez este proceso si se lo hace manualmente.

Mediante este pequeño robot XY, se piensa automatizar este proceso. Esto se debe a que en la actualidad existen programas de nivel avanzado para la construcción de placas PCB, tales como “Proteus” y “Livewire - PCB Wizzard”. Este par de programas, nos permiten generar, los caminos y los lugares donde se van a colocar dichos elementos electrónicos. De esta forma, podemos conseguir las coordenadas XY donde se va a realizar cada orificio del

circuito, por lo que, se puede implementar mediante programación un algoritmo que nos permita controlar el posicionamiento para la perforación.

En la actualidad, podemos encontrar todo tipo de microprocesadores. Por lo que, en el mercado se encuentra la serie 18F de microchip la cual tiene como última tecnología la posibilidad de conectarse a la PC mediante USB.

Mediante programación de PIC's, podemos controlar un puente H, que como bien dice su nombre es un puente, este nos ayuda a controlar motores. Los modelos de motores que se encuentran en el mercado, son infinitos, por lo que se debió tomar en cuenta varias características para comprar los motores adecuados.

Para el taladro, sabemos que las placas son de fibra de vidrio o de plástico, por lo que son fáciles de taladrar, y se optó por un pequeño motor DC que trabaje a 12VDC con buena robustez y giro.

Para los motores de los ejes necesitamos saber que:

- Se debe obtener precisión en los giros
- Debe ser de alto torque
- De bajo consumo eléctrico
- Fáciles de controlar

Estas características, cumplen los motores paso a paso ya que estos según el diseño del fabricante, pueden girar desde 0.72° hasta 90° permitiéndonos tener buena precisión de giro. Los motores paso a paso, están formados por 4 bobinas; si se programa que se activen dos de las cuatro bobinas, el torque del mismo va a mejorar, ya que se va a consumir más corriente que si trabajase con una de las cuatro bobinas a la vez.

Este tipo de motores, tiene bajo consumo eléctrico ya que no son muy grandes esto se debe a que las aplicaciones son más enfocadas a la robótica que a la industria. Por este mismo motivo son fáciles de controlar, a comparación de un gran motor AC no se necesitan variadores de frecuencia, ni grandes potencias para controlar.

El diseño mecánico, es un diseño que nos permita mover los ejes XY horizontalmente y el eje Z verticalmente mediante un tornillo sin fin el cual mediante el giro de los motores paso a paso va a hacer girar hacia adelante o hacia atrás el eje Y, y de derecha a izquierda el

eje X. De igual forma en el eje Z, hacia arriba y hacia abajo, permitiendo que el taladro suba y baje.

La implementación de todos los elementos, nos permite comunicar por medio del USB hacia la computadora obteniendo el control por medio de un programa generado según las necesidades.

Las especificaciones básicas implementadas son:

- Que obtenga los puntos XY por medio del Proteus y del Livewire - PCB Wizzard.
- Genere un gráfico de los puntos a ser taladrados.
- Que realice todas las perforaciones de la placa realizada en Proteus o Livewire - PCB Wizzard.
- Tenga un control manual de los movimientos.
- Muestre una interface de comunicación.

Estas especificaciones, nos van a permitir que una vez finalizada la tercera etapa de la fabricación de una placa PCB (uso del ácido férrico para remover exceso de cobre), se pueda poner en la máquina fabricada y realice los orificios en los lugares asignados por el Proteus o el Livewire - PCB Wizzard.

En el caso que se necesite realizar un orificio extra o mejorar alguna perforación, se ha diseñado un control manual que nos permite movilizarnos en el eje Horizontal y en el lugar deseado realizar una perforación extra.

Entre los resultados más importantes que se encontraron, está el control de los motores paso a paso ya que existen varias formas y programaciones, para el control. La comunicación USB, nos permite controlar el microcontrolador desde la computadora mientras exista una señal que nos indique que existe conexión.

OBJETIVOS:

1. Construir un Robot XYZ.
2. Controlar motores paso a paso unipolares y bipolares para el movimiento de las piezas mecánicas en el área de trabajo.
3. Controlar un motor DC con efector final taladro.
4. Utilizar sensores para determinar inicio de carrera en los movimientos horizontales.
5. Programar un microcontrolador para el control de los motores.
6. Realizar la comunicación con la computadora mediante comunicación USB.
7. Realizar una interface visual tal que:
 - a. Obtenga los puntos XY por medio del Proteus y del Livewire - PCB Wizzard.
 - b. Genere un gráfico de los puntos a ser taladrados.
 - c. Que realice todas las perforaciones de la placa realizada en Proteus - Ares o Livewire - PCB Wizzard.
 - d. Tenga un control manual de los movimientos.
 - e. Control de velocidad para los movimientos en el control manual.
 - f. Muestre una interface de comunicación.

CAPITULO 1: MARCO TEÓRICO

1.-EL ROBOT X-Y-Z:

El robot X, Y, Z es un robot el cual se mueve en un eje cartesiano de 3 dimensiones. Sus movimientos en el eje XY, son movimientos horizontales en el plano y en el eje Z con movimiento vertical, esto nos permite realizar acciones tales como “pick and place”, fresadora, taladro, y demás acciones dentro del área de trabajo. En nuestro caso tenemos un efector final que taladra, esto nos va a servir para realizar los orificios en la baquelita deseada.

Dentro de los modelos robóticos, existen robots, los cuales dependiendo del fabricante pueden tomar varias formas y tamaños. En nuestro caso vamos a utilizar una base móvil, debido a que el tamaño de las placas y el peso de las mismas, son insignificantes en comparación a la base. Este concepto, nos indica que el tamaño y el peso de las placas no van a afectar el movimiento de la base, ni el desgaste de los ejes para mover.

Como previamente se mencionó, según el diseño del fabricante, se puede encontrar diferentes formas, tamaños, efectores y demás accesorios. Existen robots que usan cadenas, cintas de movimiento y demás accesorios para moverse en el plano. En nuestro caso por conveniencia y por precio, se usa un tornillo sin fin el cual mediante el uso de tuercas permite que el movimiento de los motores gire el tornillo haciendo

que se mueva la base. Como las tuercas están pegadas a un sistema de la base, ésta se va a mover hacia adelante y hacia atrás, según el giro del motor.

De la misma forma se tiene en el eje X y en el eje Y. La diferencia son las dimensiones y la posición en la que se encuentran con respecto al punto (0,0).

El área de trabajo de un robot XY depende en lo que se lo vaya a usar, es decir, que si vamos a efectuar un “pick and place” de partes de un vehículo por ejemplo, se necesita tener un área de trabajo mayor, en comparación si vamos a realizar un “pick and place” de partes electrónicas.

Según la investigación realizada para encontrar el área de trabajo, se encontró varios tamaños de baquelitas. La de mayor tamaño encontrada fue de una hoja A4, es decir, 21.0 [cm] x 29.7 [cm]; de esta forma podemos construir un área de trabajo máxima de 21.0 [cm] x 29.7 [cm], con ajuste para placas más pequeñas.

I. Elementos Electrónicos

1) PIC 18F4550

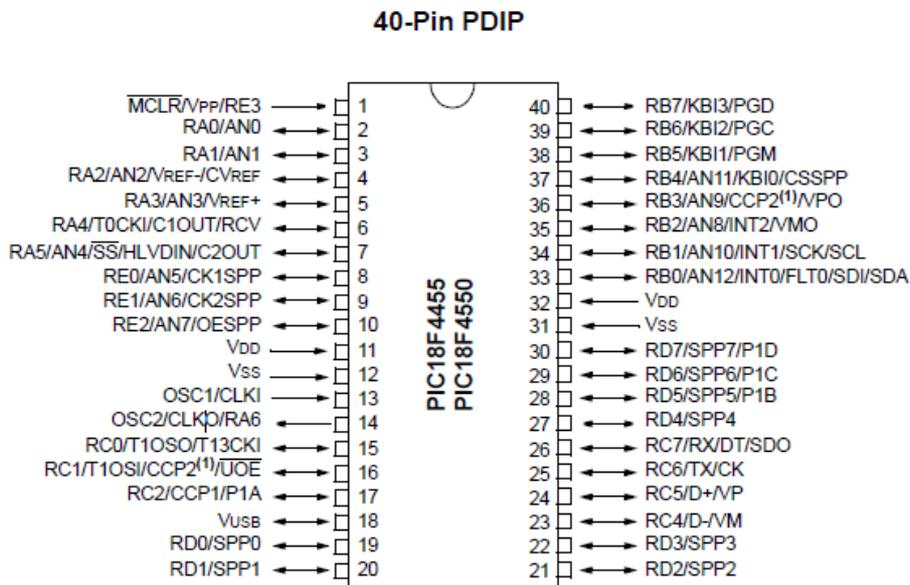


Figura 1: Pic18F4550 40 pines

La serie 18F de los microprocesadores, es la primera serie que salió con conexión USB, fabricado por la compañía Microchip¹. Esta serie nos permite utilizar de la misma forma que la serie 16F. Dentro de las características principales de este microprocesador, encontramos que:

- Conexión de alta velocidad USB 2.0 (12Mbs)
- 48 MHz de oscilación
- Entradas análogas con comparadores
- 35 entradas y salidas digitales
- 13 entradas análogas²
- Bajo consumo de energía

En nuestro caso solo se van a usar entradas y salidas digitales de los puertos B,C, D, Este microprocesador nos permite programar tanto en “assembler” como en Basic, de esta forma se puede controlar las entradas y las salidas que se van a utilizar.

2) L293 (Puente H)

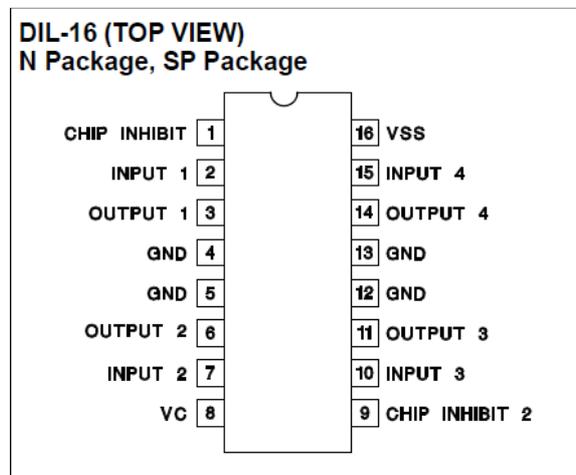


Figura 2: Puente H, L293D

¹(21)

² [34]

El L293D más conocido como puente H, es un circuito integrado, el cual nos permite manejar cargas diferentes con las que se está trabajando, esto nos permite controlar pequeños motores, relés, solenoides y demás elementos eléctricos que manejen un voltaje diferente al voltaje lógico.

En nuestro caso, el puente H, nos va permitir manejar un voltaje de 24VDC para los motores, mientras que el PIC va a manejar un voltaje lógico de 5VDC. El puente H puede manejar corrientes de hasta 1Amp. lo que nos permite manejar motores, DC, Paso a Paso y motores tipo servo.

El puente H, es muy útil ya que gracias al control TTL³, nos permite que una señal lógica active la entrada haciendo que esa salida sea activada con el voltaje diferente del canal 8;esto nos permite controlar cambio de giro de motores y generar los pasos deseados para un motor paso a paso.

³[33]

II. Elementos Electromecánicos

1) Motor Paso a Paso (Mitsumi M43SP-7)



SPECIFICATIONS

tems	M42SP-7	
Rated Voltage	DC 12V	DC 24V
Working Voltage	DC 10.8-13.2V	DC 21.6-26.4V
Rated Current/Phase	259mA	173mA
No. of Phase	4 Phase	4 Phase
Coil DC Resistance	50Ω/phase±7%	150Ω/phase±7%
Step Angle	7.5°/step	7.5°/step
Excitation Method	2-2 Phase excitation (Unipolar driving)	
Insulation Class	Class E insulation	Class E insulation
Holding Torque	49.0mN·m	52.9mN·m
Pull-out Torque	23.5mN·m/200pps	33.3mN·m/200pps
Pull-in Torque	19.6mN·m/200pps	29.4mN·m/200pps
Max. Pull-out Pulse Rate	600pps	650pps
Max. Pull-in Pulse Rate	420pps	430pps

4

Figura 3: Motor paso a paso Mitsumi M43SP-7 y especificaciones

El Mitsumi M43SP-7, es un motor paso a paso de alto torque y bajo consumo eléctrico. Este está diseñado para ser usado en impresoras, y demás objetos que necesiten precisión y buen torque. Entre las características principales tenemos que a 24VDC tiene un torque 33.3 [mN·m] a 200 pps. Como se observa en la figura 4, podemos observar la curva de trabajo de este motor.

⁴ [38]

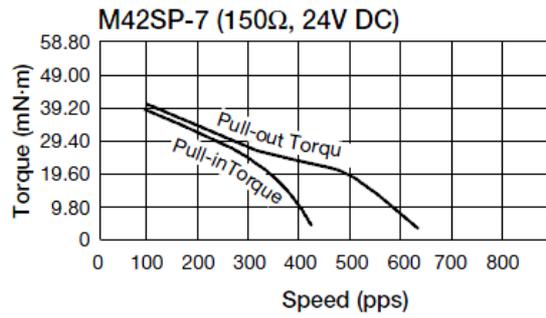


Figura 4: Curva Velocidad Vs Torque motor Mitsumi M43SP-7⁵

Entre las características de trabajo como se observa en la figura 3, se tiene que cada paso del motor es de 7.5° , lo que nos permite tener 48 pasos para conseguir mayor precisión en los movimientos. Como vamos a trabajar a 24VDC, tiene un consumo de 173 mA.

⁵ [38]

2) Motor Paso a Paso (NMB PM35S-048)



Model Specifications

Reference Characteristics		
Motor Size	PM35S-048	
No. of Steps per Rotation	48 (7.5° / Step)	
Drive Method	2-2 PHASE	
Drive Circuit	UNIPOLAR CONST. VOLT.	BIPOLAR CHOPPER
Drive Voltage	24 [V]	24 [V]
Current / PHASE		500 [mA]
Coil Resistance / PHASE	50 [Ω]	15 [Ω]
Drive IC	SMDT - 002	UDN2916B-V
Magnet Material	Ferrite plastic magnet, Polar anisotropy ferrite sintered magnet, Nd-Fe-B bonded magnet	

Figura 5: Motor paso a paso NMB PM35S-048 y especificaciones.

Al igual que el motor anterior, el NMB PM35S-048 es un motor paso a paso, de alto torque y bajo consumo eléctrico. Entre las características principales tenemos que a 24VDC tiene un torque aproximadamente 50 [mN·m] a 200 pps, según la figura 6. En la figura 6, podemos observar la curva de trabajo de este motor.

⁶ [39]

PM35S-048 BI-CHOPPER (at 24 [V], 15 [Ω], 500 [mA])

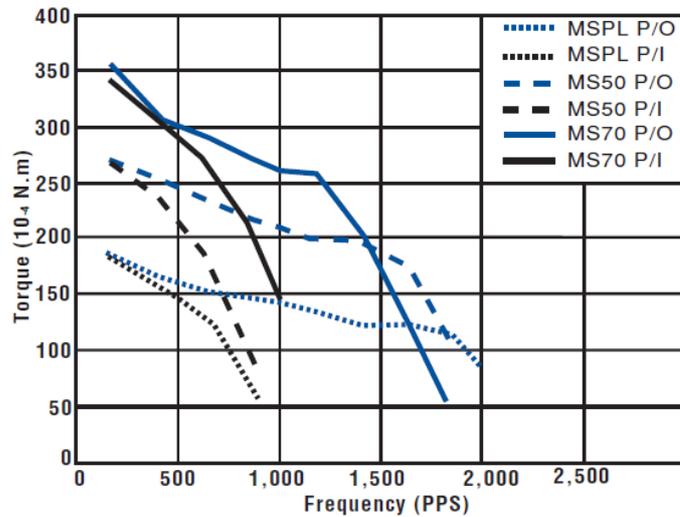


Figura 6: Curva Velocidad Vs Torque motor NMB PM35S-048

Entre las características de trabajo como se observa en la figura 5, cada paso del motor es de 7.5° , lo que nos permite tener 48 pasos para conseguir mayor precisión en los movimientos.

3) Motor DC Brushless



Figura 7: Motor DC Brushless

El motor DC Brushless, es un motor eléctrico de baja potencia, sin escobillas, por lo que le permite girar a altas revoluciones. Este tipo de motores usan corriente continua haciendo que la relación de voltaje con velocidad sea lineal. El control de éste motor se basa en cambiar la polaridad de las bobinas; este usa un voltaje

máximo de 12VDC, para obtener 12000 rpm con máxima consumo de corriente, que en este caso es 800 [mA]⁷. Este motor nos va a ayudar mediante el acople del efector final (taladro) usando la velocidad para traspasar la placa que son aproximadamente 2mm.

4) CONTROL MOTORES

Los motores tanto paso a paso como los DC, necesitan tener un control tanto de prendido y apagado, así como de velocidad.

a. Control Motores P.A.P

Los motores paso a paso como dice su nombre, son motores que dan un paso a la vez cuando un pulso es aplicado. Este paso puede variar dependiendo de la cantidad de dientes que el fabricante diseñe. Estos dientes tienen ángulos desde el 0,72 ° hasta los 90° brindando diferentes precisiones según la necesidad. Como se puede observar en la tabla 1, dependiendo del ángulo se tiene el número de pasos para dar una vuelta completa.⁸

Angulo	Cantidad de pasos
0,72°	500
1,8°	200
3,75°	96
7,5°	48
15°	24
90°	4

⁷ [32]

⁸ [42]

En la siguiente figura, figura 8, se puede observar el rotor y el estator. Tanto el rotor como el estator en un motor paso a paso tienen la misma cantidad de dientes los que dependen del número de pasos que se necesite, es decir, entre menor ángulo mayor cantidad de dientes se debe fabricar, y entre más dientes más cantidad de pasos y mayor precisión se va a obtener.

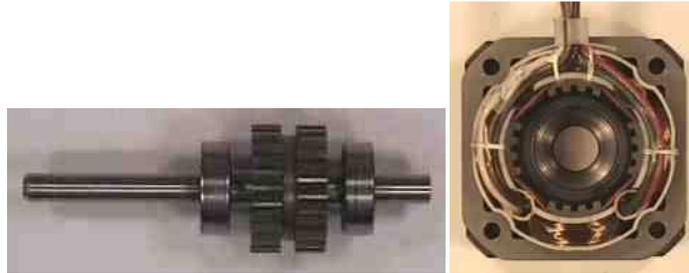


Figura 8: rotor y estator de un paso a paso

Entre los motores paso a paso, se pueden encontrar diferentes variaciones, tales como los motores paso a paso bipolares y unipolares. La diferencia entre estos dos motores es la cantidad de hilos que se encuentran. Como se observa en la figura 9, el motor paso a paso bipolar tiene 4 hilos, mientras que el unipolar presenta 6 hilos. En este tipo de motores dependiendo del fabricante podemos encontrar 5 o 6 hilos ya que el común como se observa en la figura, puede venir tanto unido como sueltos, es por aquella razón que encontramos más o menos hilos.

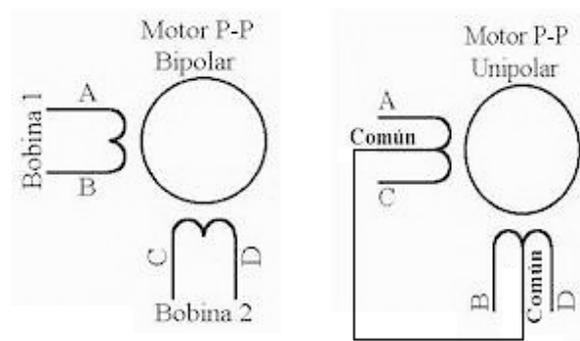


Figura 9: hilos motores paso a paso bipolar y unipolar

Para controlar un motor paso a paso, necesariamente se debe usar pulsos para activar cada bobina. Para la mayoría de casos, los motores paso a paso

son motores de baja corriente, sin embargo, si el control se lo va a realizar por medio de un microprocesador, se debe usar un puente H, ya que se debe modificar el voltaje y se debe usar corriente continua para activar cada pulso en la bobina. Como podemos observar en la figura 10 mediante el uso de un microprocesador, este envía el pulso de activación el cual activa la salida del puente H el cual activa la bobina conectada.

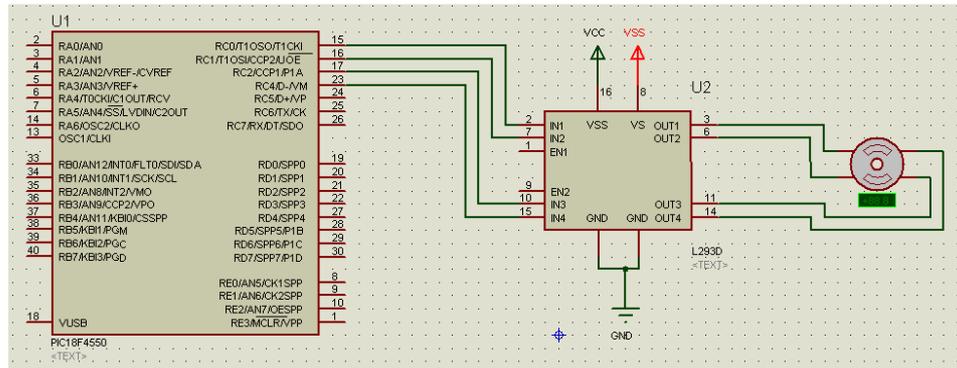


Figura 10: Conexión motor paso a paso bipolar

De igual forma para un motor paso a paso unipolar, se conecta éste al puente H y al Pic. La única variante para este motor es que los hilos comunes se conectan a la fuente de energía.

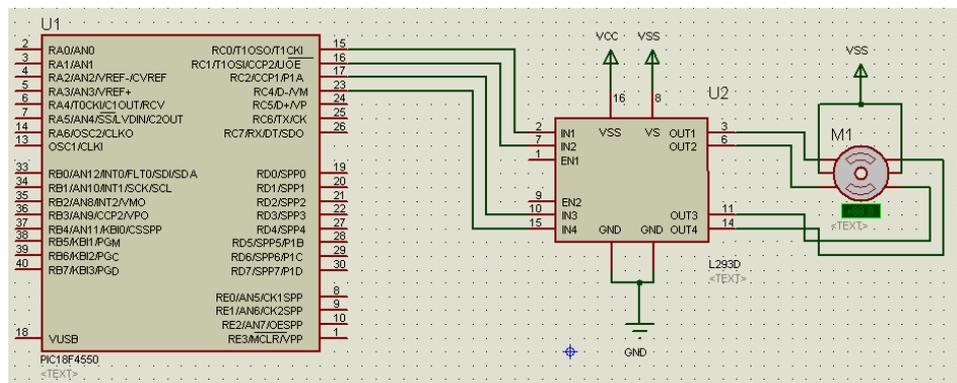


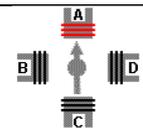
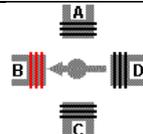
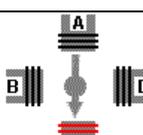
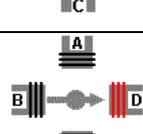
Figura 11: Conexión motor paso a paso unipolar

Existen secuencias para controlar este tipo de motores, ya que se debe activar una bobina, luego la siguiente y así sucesivamente para poder mover el rotor del motor.

Entre las secuencias que se puede generar existe:

- Wave drive
- Full Step
- Half Step

La secuencia Wave drive, es la que activa una bobina a la vez, y así sucesivamente con la siguiente. Como se puede apreciar en la tabla 2, podemos generar una tabla de verdad, la que nos indica que bobina tiene que estar activada para generar la secuencia. De esta forma podemos programar en el Pic, que puerto tiene que activar y que puerto debe desactivar. De la misma forma poniendo la secuencia al revés se obtiene el giro contrario al programado.

Tabla 2: Tabla de verdad secuencia wave drive.					
Paso	Bobina A	Bobina B	Bobina C	Bobina D	
1	1	0	0	0	
2	0	1	0	0	
3	0	0	1	0	
4	0	0	0	1	

La secuencia Full Step, es la secuencia más común, que se genera, ya que se activan dos bobinas a la vez generando más torque. Este tipo de secuencia es la recomendada por el fabricante. Como se observa en la tabla 3, de igual forma se observa la tabla de verdad, para la secuencia Full step.

Tabla 3: Tabla de verdad secuencia Full Step.					
Paso	Bobina A	Bobina B	Bobina C	Bobina D	
1	1	1	0	0	
2	0	1	1	0	
3	0	0	1	1	
4	1	0	0	1	

La secuencia Half Step, es una combinación de las dos secuencias anteriores, haciendo que el pulso que se genera camine solo medio paso. Como se observó previamente, el full step hace que el rotor camine un paso ,pero entre cada bobina activada y el wave drive camina entre cada bobina activada. Si unimos los dos, es decir, primero wave drive y luego full step, hacemos que el rotor gire medio paso haciendo que en vez de tener 4 tiempos, se hacen 8. Como se puede observar en la tabla 4, es la tabla de verdad para una secuencia Half step.

Tabla 4: Secuencia Half Step.					
Paso	Bobina A	Bobina B	Bobina C	Bobina D	
1	1	0	0	0	
2	1	1	0	0	
3	0	1	0	0	
4	0	1	1	0	
5	0	0	1	0	
6	0	0	1	1	
7	0	0	0	1	
8	1	0	0	1	

La velocidad de un motor paso a paso, depende específicamente del tiempo en que se demora en cambiar de un estado a otro, generalmente este tiempo para un giro continuo está entre los 3 y 50 ms. Esto se puede configurar

fácilmente en la programación de nuestro microcontrolador, usando la opción **PAUSE XXX**, donde XXX es la cantidad de milisegundos.

b. Control Motor DC

El control de estos dispositivos electromecánicos es más simple que cualquier otro tipo de motores ya que para obtener más velocidad se debe aumentar más voltaje y el control de giro debe cambiarse los hilos. Esto provoca que la corriente se invierta en el estator haciendo que el motor gire para el otro lado. Como se observa en la figura 12, la conexión del motor DC se la puede hacer a través de un puente H ya que como son motores pequeños de baja corriente y voltaje no se necesita un dispositivo de potencia. Si observamos la conexión, el puerto que activa la entrada del puente H, activa la salida, haciendo que éste gire para un lado mientras el otro permanece inactivado, haciendo la función de tierra. Si activamos el otro puerto va a suceder exactamente lo mismo pero al revés, provocando que el motor gire para el lado contrario.

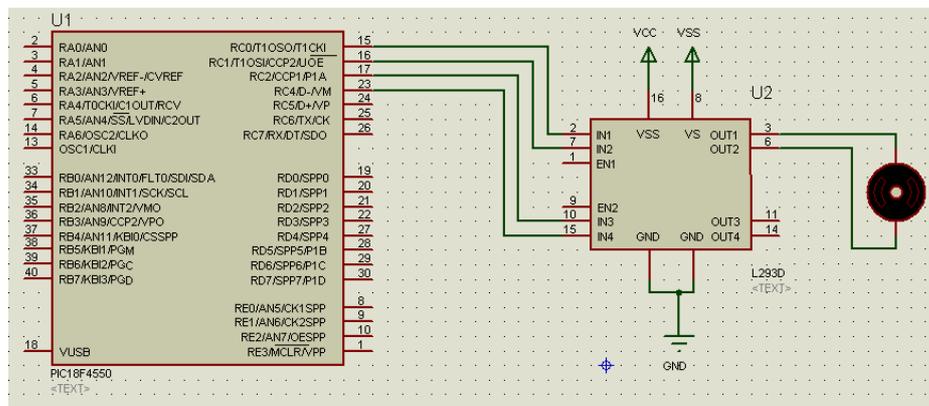


Figura 12: conexión motor DC

El voltaje de estos dispositivos varía entre 1 VDC hasta los 25 – 30 VDC, provocando que aumente la velocidad conforme se aumenta el voltaje.

III. Elementos Mecánicos

1) Rieles

Las rieles, son los elementos mecánicos sobre las cuales existe un desplazamiento, éstas nos sirven para guiar, sostener y equilibrar⁹. En nuestro caso las rieles son tubos de aluminio, los cuales están fijos a los extremos. En la base la cual va hacia adelante y hacia atrás en el eje horizontal, en el eje X que va de izquierda a derecha y en el eje Z que sube y que baja, se pueden encontrar rieles para que éstas mantengan su dirección y estabilidad.

Como se observa en la figura 13, se tiene una configuración en la que las rieles van a sostener, para que no se vaya hacia un lado o hacia otro.



Figura 13: Configuración Rieles - Tornillo

2) Tornillo sin Fin

El tornillo sin fin es un dispositivo que nos permite cambiar el movimiento de rotación a traslación, es decir, que el giro del motor va a cambiar a un movimiento horizontal¹⁰. Se lo define tornillo sin fin, ya que este nunca llega a un tope, o el giro de éste no hace que se detenga ningún mecanismo.

⁹ [5]

¹⁰ [34]

Se lo llama también varilla roscada de tipo milimétrica, ya que al ser milimétrica la distancia entre dos crestas de la rosca o entre dos de sus ranuras es de 1 mm, a ésta distancia también se la llama paso¹¹.

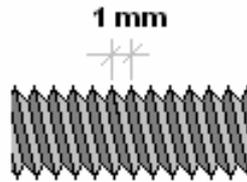


Figura 14: Tornillo sin fin

IV. Proceso de producción de placas PCB

La producción de placas PCB es un proceso de cinco pasos los cuales se componen de: la generación y simulación del circuito en el software, la impresión y adhesión del circuito en la placa, el uso del ácido para retirar el cobre en exceso, la perforación de los orificios para inserción de los elementos eléctricos y la soldadura de los elementos. Estos cinco pasos mencionados son pasos concatenados, es decir, que primero se debe realizar uno a uno para continuar con el siguiente proceso en la generación de una placa PCB, y además debe ser realizado con paciencia.; esto se debe a que poco a poco y con experiencia se va aprendiendo como realizarlas mejor.

1) *Generación y simulación del circuito en el software.*

El primer paso, es la simulación del circuito en software, esto nos ayuda a confirmar el funcionamiento del circuito diseñado. Existen programas, tales como Orcad Pspice, Livewire - PCB Wizzard, Proteus, Eagle, entre otros.

Estos programas nos permiten utilizar varios componentes eléctricos y electrónicos para simular. En nuestro caso se ha usado el Proteus, ya que contiene gran cantidad de componentes al igual que el Livewire - PCB Wizzard que nos permiten simular y generar el circuito impreso.

¹¹ [44]

Como se observa en la figura 15, se observa el funcionamiento del circuito simulado en Proteus

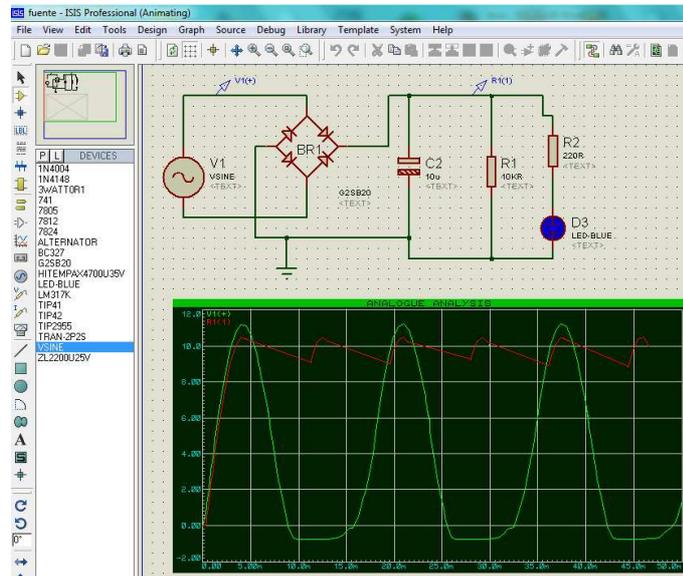


Figura15: Simulación Circuito Proteus

Después de haber realizado la simulación, el paquete de Proteus ofrece Ares, el cual es un programa complementario para la generación de circuitos impresos el cual tiene dos opciones: la primera, es realizar el circuito a mano, es decir utilizar los componentes y unirlos al gusto del usuario poniendo la posición y la forma de unión de los caminos. La segunda opción es utilizar el “autoruting” el cual es un componente de este programa que automáticamente genera los caminos y la posición de los elementos. Como se observa en la figura 16, se ha realizado con “autoruting”.

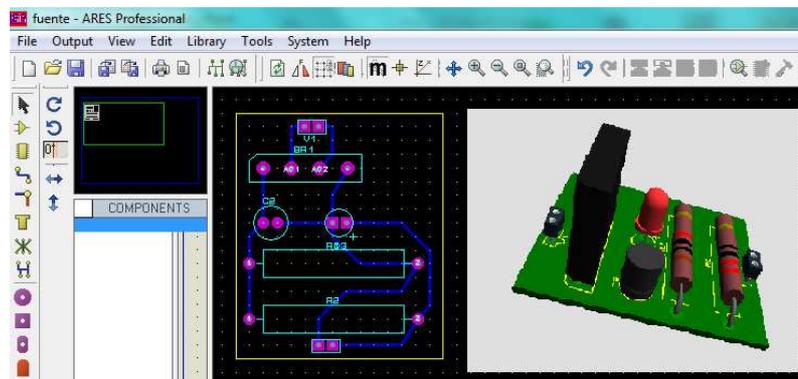


Figura16: Circuito impreso Proteus

Al igual que el Proteus, el Livewire - PCB Wizzard también tiene la opción de simular como se observa en la figura 17, observándose el funcionamiento del mismo circuito simulado en Livewire - PCB Wizzard.

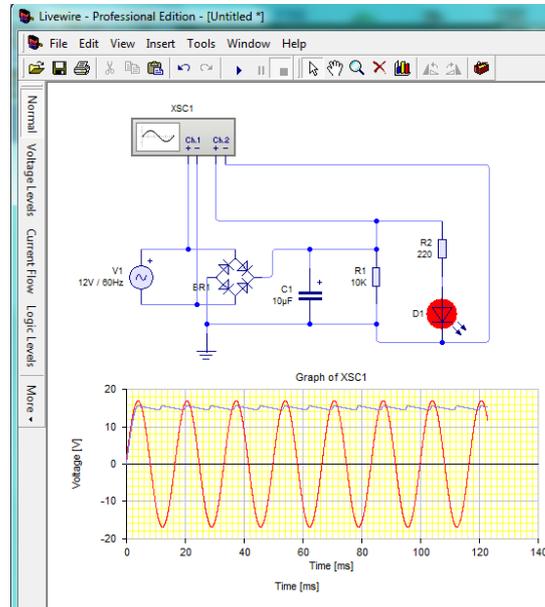


Figura 17: Simulación en Livewire - PCB Wizzard

De la misma forma que el Proteus, el Livewire - PCB Wizzard tiene igualmente un programa para generar los circuitos impresos éste se llama PCB Wizzard el cual también tiene las opciones para generar los circuitos impresos (manual y automático). Al igual que el Proteus para el mismo circuito se ha usado el autoruting, el cual nos entrega el mismo circuito, pero con los componentes en diferente sentido y con diferentes caminos. En el caso que se necesite tener la misma forma, tamaño, posición y caminos, lo más recomendable es realizar los circuitos a mano.

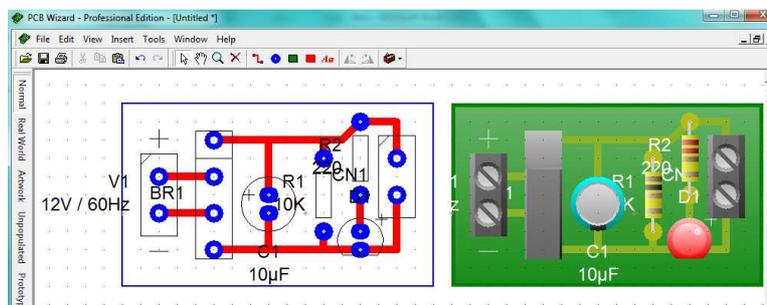


Figura18: Circuito impreso Livewire - PCB Wizzard

2) Impresión y adhesión del circuito en la placa.

La impresión y la adhesión, es la segunda etapa para la fabricación de una placa PCB. Esta etapa está dividida en dos acciones: la de imprimir en el papel transferencia y la de planchar el papel transferencia. Estas dos acciones están unidas debido a la manipulación del papel transferencia.

El papel transferencia, es un papel de transferencia pres-n-peel¹² el cual se lo puede conseguir en cualquier tienda electrónica. Este papel es un papel del tipo glossy, es decir, que tienen una capa de barniz lo que le permite que tenga aquel brillo característico.

Para realizar la impresión dependiendo del programa, se deben tomar en cuenta algunas consideraciones; en el caso del Proteus como se observa en la figura 19, podemos ver los puntos en lo que hay que poner atención para que la impresión salga correcta.

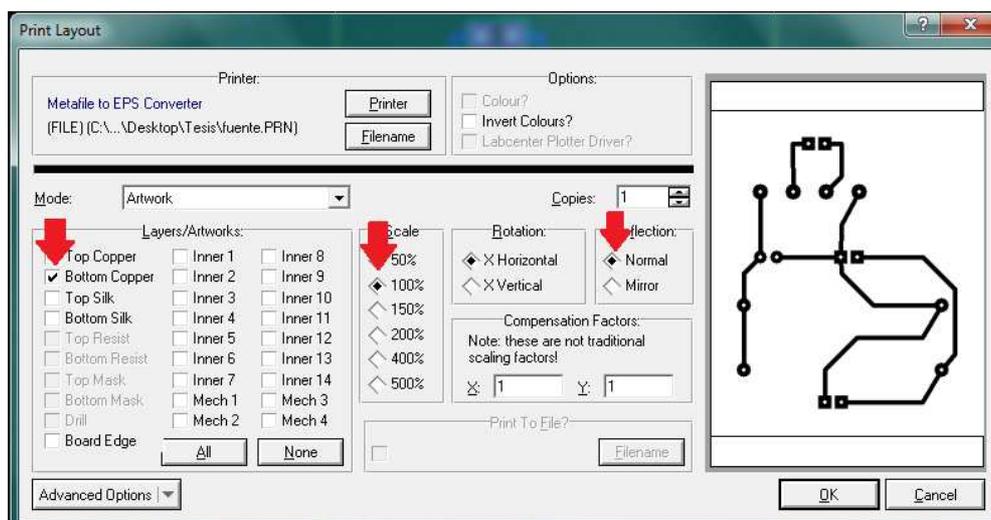


Figura 19: características de impresión Proteus

La primera flecha, de izquierda a derecha, nos indica que se va a imprimir, en éste caso el BOTTON COOPPER que es el área donde se ha realizado los caminos y los PAD's (punto en el cual se va a realizar el orificio y se va a soldar). La segunda flecha nos indica la escala, por lo general es al 100%. Este porcentaje nos indica el

¹² [42]

tamaño de la placa en la vida real. Y la tercera flecha nos indica si queremos la impresión normal o “mirror” (espejo). La impresión normal nos imprime tal cual se ha diseñado el circuito, y la opción mirror nos imprime como se lo hubiese volteado horizontalmente al circuito. En nuestro caso debido a que se va a utilizar una placa de fibra de vidrio de un solo lado de cobre se realiza la impresión utilizando la opción normal, lo que nos va a permitir tener los caminos y los PAD’s en la parte de cobre y en la que no tiene, tener los elementos.

Al igual que el Proteus, el Livewire - PCB Wizzard posee la misma opción para observar cómo quedan los caminos y los PAD’s. Como se observa en la figura 20, la flecha nos indica que haciendo clic en el botón de “Artwork”, se puede fácilmente observar cómo va a quedar la impresión.

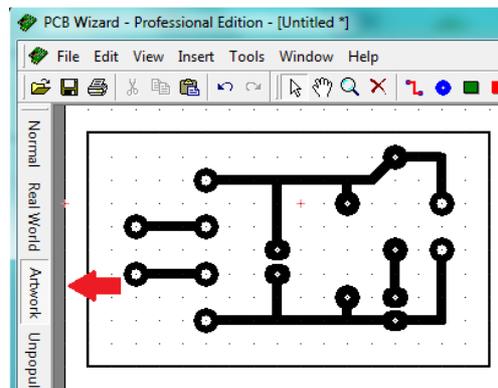


Figura 20: Opción para ver los caminos en PCB Wizzard

En el caso del Livewire - PCB Wizzard - PCB Wizzard, al momento de querer realizar la impresión, este no tiene un ambiente amigable con el cual uno pueda interactuar como el Proteus. Una vez observado la figura de cómo va a ser la impresión solo se hace clic en el botón print (imprimir), de esta forma conseguimos lo mismo que estamos observando en el papel transferencia.

Un detalle muy importante al momento de realizar la impresión, es realizarla en una impresora laser, ya que el barniz del papel transferencia que por efectos químicos se une con el polvo del tóner, permitiendo que fácilmente los caminos y los PAD’s queden levemente levantados.

Una vez que tenemos nuestra impresión, se puede proseguir a cortar la placa. Se debe adicionar 4mm más a los bordes impresos del circuito impreso. Una vez

cortado se prosigue con la transferencia térmica, este proceso consiste mediante calor hacer que los caminos impresos en la hoja sean pegados a la placa. Antes de proseguir es recomendable limpiar el cobre con una esponja de acero, ya que se deben quitar los rayones y el óxido creado en la superficie.

Una vez limpia la placa, se debe manipular por los bordes, ya que la grasa de los dedos genera óxido en la superficie de cobre; se debe colocar sobre una mesa firme la placa y sobre ésta el papel transferencia con los caminos hacia el cobre. Usando una plancha doméstica a su mayor potencia (máxima temperatura), se debe poner sobre la placa por 30 segundos haciendo máxima presión sobre ésta para que los caminos se transfieran bien hacia la placa.

Después del tiempo indicado, usando una franela, movemos la placa, y comenzamos a aplicar presión uniformemente frotándolo de un lado a otro hasta que se enfríe. Una vez terminado el proceso se retira el papel transferencia, observando que los caminos y PAD's quedaron correctamente transferidos a la placa.

3) Uso del ácido para retirar el cobre en exceso.

Cuando los caminos han sido transferidos a la placa, se debe retirar el exceso de cobre para que solo los caminos protegidos por el barniz y la tinta queden. Para esta etapa, se necesitan algunos materiales extras tales como un envase plástico o de vidrio, un par de fundas de cloruro férrico, los cuales venden en cualquier tienda electrónica, también un palillo de pinchos para poder mover y alzar la placa.

Primero se hierven dos vasos de agua, una vez que el agua está hirviendo se pone en el recipiente de plástico o de vidrio, no se recomienda usar un envase metálico ya que el ácido que se va a usar corroe el metal, haciendo que el envase usado no sirva. Una vez vertido el agua en el envase, se deja caer la placa y con mucho cuidado se va arrojando poco a poco el ácido.

En esta etapa del proceso de debe tener mucho cuidado, ya que la reacción del agua con el ácido, es de acción agresiva. De igual forma se lo debe realizar en un lugar

con buena ventilación y con ropa preferiblemente vieja, ya que el ácido salta y mancha.

Una vez que el ácido ha hecho contacto con el agua, se lo comienza a mover de lado a lado, formando un oleaje. Este nos permite acelerar el proceso, ya que el ácido en constante movimiento remueve de forma más fácil el exceso de cobre.

Debido a que es un proceso que puede tardar un poco y el ácido es oscuro, mediante el uso del palo de pincho se lo usa para sacar la placa y ver que tanto se ha avanzado en el proceso.

Una vez que se haya retirado todo el exceso de cobre, se lo debe lavar con abundante agua hasta que la placa quede limpia, y se prosigue a secarla con una toalla.

Cuando la placa está seca, se puede observar que los caminos y los PAD's tienen un color negro producto de sumersión en el ácido. Cuando se secan bien los caminos éstos toman un color blanco, para retirar estas impurezas, se tienen dos formas para limpiar. La primera utilizando acetona, tñer o cualquier disolvente con un poco de algodón. El efecto secundario de este tipo de limpieza, si bien limpia los caminos y nos permite ver el cobre, deja un color negrizo a la placa dejándola sucia y poco presentable. La segunda forma de limpiar es utilizar la misma esponja metálica remojada en agua de ésta forma, conseguimos quitar el color y pulir los caminos entregando una placa nítida y en perfecto estado.

4) Perforación de los orificios para inserción de los elementos eléctricos.

La perforación de los orificios, es la etapa en la que se hace el espacio, generalmente redondo para que el terminal metálico pase entre en la placa, permitiendo que ésta sea aislada de los demás elementos.

Existen varios tipos de taladros, los cuales nos pueden ayudar en esta etapa, entre estos tenemos:

- Taladros normales

- Taladros miniaturas o moto-tool
- Taladro industrial con pedestal

Para los dos primeros tipos de taladro, previamente se debe utilizar un clavo y un martillo para fijar el punto donde se va a realizar el orificio. De la misma forma si no se desea usar el martillo, con un cuchillo o una navaja, se puede presionar y realizar una pequeña hendidura, esto nos sirve para poder fijar la broca evitando que se mueva y fallar la perforación.

Una vez realizada las hendiduras en las que se van a realizar los orificios, usando una broca de 1mm de diámetro, se realizan los huecos que sean necesarios.

5) Soldadura de los elementos.

La soldadura de los elementos, es la unión del terminal del dispositivo eléctrico con el camino de cobre mediante el calentamiento del estaño. De esta forma conseguimos que el dispositivo quede pegado a la placa de tal forma que la corriente que pasa por los caminos al ser el estaño un elemento conductor, pase hacia el elemento eléctrico.

Una vez hechos los orificios, y con todos los elementos que son parte de la placa, se procede desde los más bajos a los más altos, por ejemplo, primero van las resistencias, los diodos y demás dispositivos que irían acostados. Luego irían los zócalos, borneras, y para finalizar los dispositivos más altos por ejemplo los capacitores, de esta forma podemos ir apoyando en la mesa sin perder el equilibrio de la placa.

Una vez que todos los dispositivos están en su sitio colocados con la polaridad correcta, en el caso de los capacitores electrolíticos y los leds, se procede a soldar. La mejor técnica de soldadura, se basa en tener limpia la punta y muy caliente el cautín, de esta forma vamos a derretir el estaño más fácilmente. Con la punta limpia y caliente se toca el camino de cobre y la base de la pata, de esta forma se va a calentar rápidamente y el estaño se va a derretir. Cuando este haya sido aplicado alrededor de la pata, se suelta y en pocos segundos se seca el estaño. Una vez

soldadas todas las puntas se prosigue a cortar las patas con un alicate, haciendo que quede todo a un mismo nivel.

V. Configuración de la comunicación USB

La comunicación USB (Universal Serial Bus) entre el microcontrolador y la computadora, es una técnica que se la viene usando desde que la serie 18F de microprocesadores salió al mercado. Esto se debe a que se pueden manipular velocidades de 1.2 Mbps. con el USB 1.0 y 480 Mbps. con el USB 2.0. También teniendo en cuenta que la mayoría de computadoras que se encuentran hoy en el mercado carecen tanto de puertos seriales como puertos paralelos lo que nos permitía antes realizar la comunicación.

Entre las muchas formas que existen para comunicar con USB, existe el EASY HID, el cual es un pequeño programa hecho por la compañía MECANIQUE, el cual nos genera dos plataformas de desarrollo. Entre estas podemos encontrar para los microprocesadores, el Pic Basic Pro y Proton que nos permite programar los Pic's de la serie 18F, tales como 18F2455, 18F2550, 18F4455 y 18F4550, y para la computadora nos genera el código en Borland Delphi, Visual C++ y Visual Basic. Este programa también nos entrega lo que son los drivers para que reconozca la computadora al Pic el momento de la primera conexión.

Cuando uno inicia el programa, la primera pantalla que nos aparece, como en la figura 21, ésta nos permite cargar el nombre del producto que se está desarrollando.

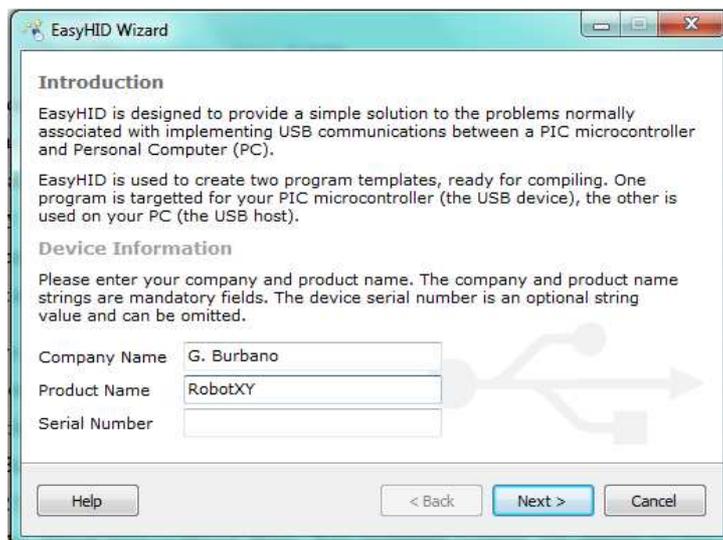


Figura 21: Pantalla 1 del HID, cargar nombre del producto

La siguiente página nos entrega lo que son el VENDOR ID y el PRODUCT ID, los cuales son números asignados por el organismo que regula la autenticidad de los productos USB donde se puede comprar por U\$S 4.000 una membresía anual con números otorgados exclusivamente al desarrollo¹³. Al ser un proyecto sin fin de lucro, se puede mantener los mismos datos, para realizar la aplicación.

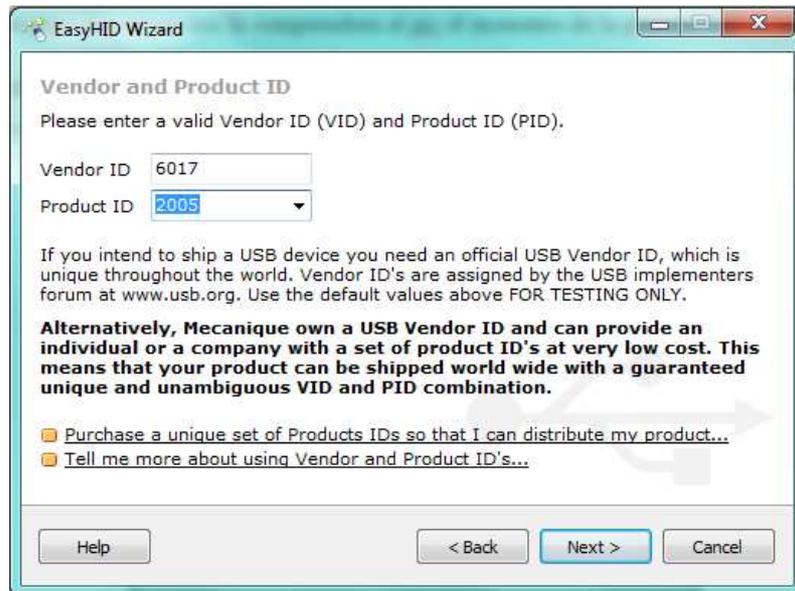


Figura 22: Vendor Id y Product Id

La tercera etapa de este programa, como se muestra en la figura 23, se observa las diferentes opciones del USB. Son casillas donde uno puede cambiar las diferentes opciones. En Polling input y polling output como se observa son datos de tiempo. El polling input es el intervalo usado por la computadora para pedir datos del Pic, mientras que el polling output es el intervalo usado para enviar datos al Pic. Como se observa, es un tiempo de 10 ms, lo que se demora en recibir y enviar datos. El bus power, es la cantidad de corriente que maneja el puerto USB, éste tiene 50 mili-amperios lo que generalmente maneja el USB de las computadoras. En Buffer input y en Buffer output, es la cantidad de bytes que se van a usar para enviar y recibir datos hacia la computadora, y se puede generar hasta 64 bytes de comunicación. La

¹³ [7]

recomendación del fabricante como se observa, es que si no está seguro de la modificación de los datos, se debe trabajar con lo pre-establecido.

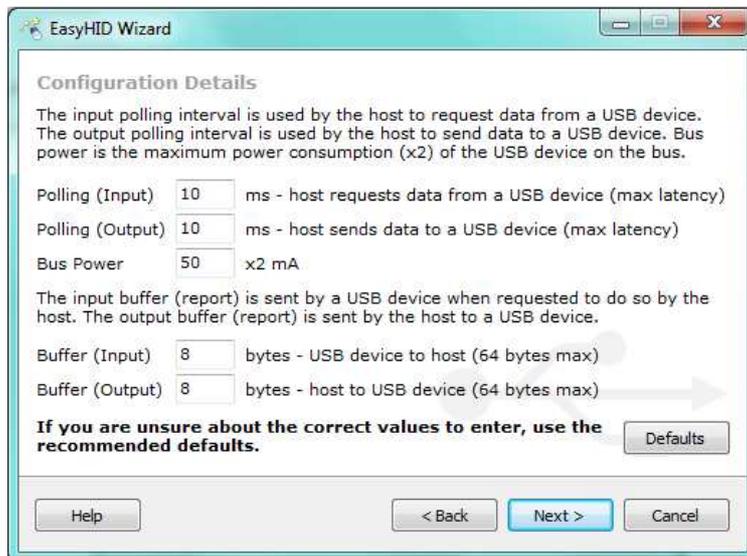


Figura 23: configuración de la cantidad de bytes

La figura 24, nos indica la pantalla más importante de todas, ya que como se observa, aquí es donde se selecciona el Pic con el que se va a trabajar, el programa con el cual se va a generar el código para el Pic, el programa que va a controlar el Pic, el nombre del archivo y su ubicación. En nuestro caso, se va a usar un Pic 18F4550, se lo va a programar en PIC BASIC, y se va a usar VISUAL BASIC 5 para poder controlar la comunicación con el Pic.



Figura 24: opciones de programas y Pic's

Una vez aceptado todo, se procede a compilar como se observa en la figura 25, se va indicar que no existen problemas y se generaron las aplicaciones correctamente.

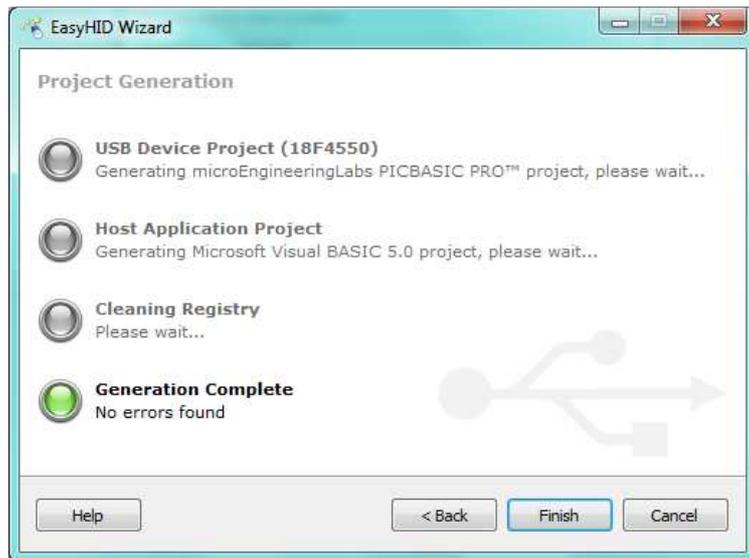


Figura 25: Confirmación de la generación de códigos

En el lugar donde se guardaron los archivos podemos observar las plataformas creadas por el programa, como en la figura 26 nos muestra con las flechas el programa generado con la extensión PBP que es para el Pic Basic Pro y VPB que es la extensión para el visual Basic.

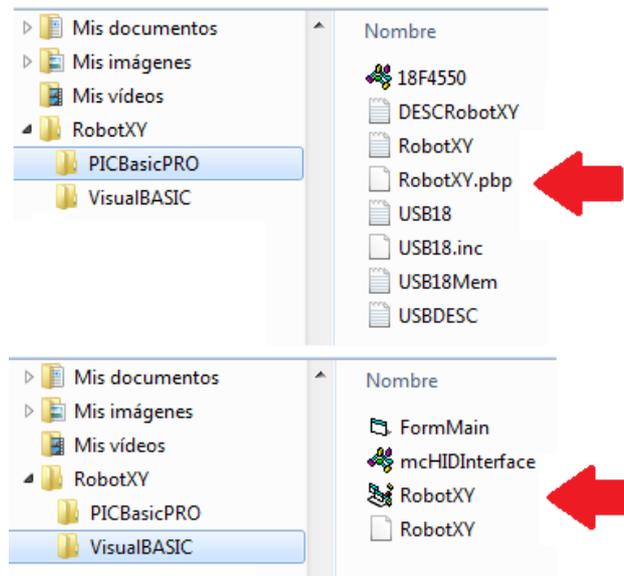


Figura 26: Programas creados con el HID

En el Pic Basic pro cuando se genera el código, se observan 3 partes importantes: la primera es donde se escribe el código, la segunda es la recepción de datos y la tercera el envío de datos.

```

1
2 DEFINE OSC 48
3 DEFINE LOADER_USED 1
4
5 USBBufferSizeMax    con 8  ' maximum buffer size
6 USBBufferSizeTX    con 8  ' input
7 USBBufferSizeRX    con 8  ' output
8
9 ' the USB buffer...
10 USBBuffer          Var Byte[USBBufferSizeMax]
11 USBBufferCount     Var Byte
12
13 ' *****
14 ' * main program loop - remember, you must keep the USB      *
15 ' * connection alive with a call to USBService every couple *
16 ' * of milliseconds or so...                                *
17 ' *****
18 usbinit ' initialise USB...
19 ProgramStart:
20   gosub DoUSBIn
21   gosub DoUSBOut
22   goto ProgramStart
23
24 ' *****
25 ' * receive data from the USB bus                               *
26 ' *****
27 DoUSBIn:
28   USBBufferCount = USBBufferSizeRX          ' RX buffer size
29   USBService                                ' keep connection alive
30   USBIn 1, USBBuffer, USBBufferCount, DoUSBIn ' read data, if available
31   return
32
33 ' *****
34 ' * wait for USB interface to attach                            *
35 ' *****
36 DoUSBOut:
37   USBBufferCount = USBBufferSizeTX          ' TX buffer size
38   USBService                                ' keep connection alive
39   USBOut 1, USBBuffer, USBBufferCount, DoUSBOut ' if bus available, transmit data
40   return

```

Figura 27: Código generado por el HID para el PBP

Entre las especificaciones del USB, previamente se había puesto que se use buffer in y out 8 bytes, esto significa que en el PBP se va a enviar y recibir datos usando 8 espacios, es decir, USB Buffer[0], USB Buffer[1] → USB Buffer[7] = 0 ó a 1. Esto nos indica que cuando el USB Buffer[x] está activado (=1) realice la acción deseada. Por ejemplo:

If USB Buffer [0] = 1 then (botón activado desde el Visual Basic activa)

High PortB.0 (Puerto b.0 se active (se prende un led))

Pause 1000 (se mantiene prendido por 1 segundo)

Low PortB.0 (Puerto b.0 se desactiva (se apaga un led))

Pause 1000 (se mantiene apagado por 1 segundo)

EndIf (finaliza la acción)

Para enviar un dato hacia el visual Basic se configura de la siguiente manera:

portB.1 = USB Buffer[x] (x es el número de buffer usado)

En la figura 28 como se puede ver, está la programación generada por Easy HID en este caso para Visual Basic. Aquí podemos encontrar 8 partes las cuales nos ayudan a mantener primero la conexión viva y segundo el envío y la recepción de datos.

Entre los detalles más importantes que se pueden resaltar está la definición de Vendor ID y del Producto ID, la cantidad de bytes de entrada y de salida que se van a usar, el llamado al driver cuando se carga el programa, e igualmente la finalización de la llamada al driver cuando se desconecta y la lectura - escritura de datos hacia el Pic.

Cuando se escriben datos al Pic, como se había definido 8 bytes de buffer in, en el Visual Basic el conteo no comienza en 0 sino en 1, lo que provoca que lo que en el PBP es 4 en el VB es 5. De tal forma que se tiene BufferOut (1), (2),... BufferOut (8).

Para escribir un dato hacia el Pic desde el visual Basic se usa:

Private Sub dere_MouseDown() (clic en un botón generado en VB)

BufferOut (2) = 1 (USB Buffer [1] en PBP es activado)

HidWriteEx VendorID, ProductID, BufferOut (0) (rutina de detección)

End Sub (finalización del envío)

De la misma forma para leer un dato desde el Pic se usa parte del código generado por el HID, el cual metiendo el buffer deseado a una variable conseguimos que el dato se traído hacia el Visual Basic desde el microcontrolador

```

'*****
'
'                               on read event...
'*****

```

Public Sub OnRead(ByVal pHandle As Long)

' read the data (don't forget, pass the whole array)...

If hidRead(pHandle, BufferIn(0)) Then

' ** YOUR CODE HERE **

Temp = BufferIn(8) (Variable igual al dato del microcontrolador)

Tactual.Caption = Temp (variable visible en visual Basic)

' first byte is the report ID, e.g. BufferIn(0)

' the other bytes are the data from the microcontrolller...

End If (fin del IF)

End Sub (Fin de la lectura)

```

' vendor and product IDs
Private Const VendorID = 6017
Private Const ProductID = 2005

' read and write buffers
Private Const BufferInSize = 8
Private Const BufferOutSize = 8
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte

' *****
' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
' *****
Private Sub Form_Load()
' do not remove!
ConnectToHID (Me.hwnd)
End Sub

' *****
' disconnect from the HID controller...
' *****
Private Sub Form_Unload(Cancel As Integer)
DisconnectFromHID
End Sub

' *****
' a HID device has been plugged in...
' *****
Public Sub OnPlugged(ByVal pHandle As Long)
If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) = ProductID Then
' ** YOUR CODE HERE **
End If
End Sub

' *****
' a HID device has been unplugged...
' *****
Public Sub OnUnplugged(ByVal pHandle As Long)
If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) = ProductID Then
' ** YOUR CODE HERE **
End If
End Sub

```

```

'*****
' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'*****
Public Sub OnChanged()
    Dim DeviceHandle As Long

    ' get the handle of the device we are interested in, then set
    ' its read notify flag to true - this ensures you get a read
    ' notification message when there is some data to read...
    DeviceHandle = hidGetHandle(VendorID, ProductID)
    hidSetReadNotify DeviceHandle, True
End Sub

'*****
' on read event...
'*****
Public Sub OnRead(ByVal pHandle As Long)

    ' read the data (don't forget, pass the whole array)...
    If hidRead(pHandle, BufferIn(0)) Then
        ' ** YOUR CODE HERE **
        ' first byte is the report ID, e.g. BufferIn(0)
        ' the other bytes are the data from the microcontrolller...
    End If
End Sub

'*****
' this is how you write some data...
'*****
Public Sub WriteSomeData()
    BufferOut(0) = 0 ' first by is always the report ID
    BufferOut(1) = 10 ' first data item, etc etc

    ' write the data (don't forget, pass the whole array)...
    hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub

```

Figura 28: Código generado por el HID para Visual Basic

Una vez que se ha entendido el funcionamiento del USB, antes de cualquier modificación al PBP se debe correr y generar el archivo .HEX es el archivo hexadecimal con el cual se programan los micro controladores. Una vez programado el Pic, se prosigue a conectar el conector USB a la computadora. Como se observa en la figura 29, el sistema operativo reconoce el dispositivo mediante el driver mcHID.dll. Este driver es instalado en la carpeta del Easy HID el momento en que se instala el programa.



Figura 29: Dispositivo conectado con la computadora

Finalizada con éxito la conexión, el microcontrolador está listo para ser programado, y poder trabajar con éste ya que la computadora lo reconoció y el plug and play de los dispositivos USB está listo para usarse.

CAPITULO 2: DISEÑO Y SOLUCIÓN

I. Diseño y construcción del prototipo

La construcción del prototipo, debe ser de tamaño real, es decir, que debe satisfacer las necesidades del mercado. Como previamente se mencionó, las placas que se encuentran en el mercado, varían entre placas de 10 [cm] x 10 [cm] hasta el tamaño A4 que es 21.0 [cm] x 29.7 [cm], de tal forma que pueda taladrar en cada punto en esta área.

Dentro del diseño se ha pensado en una caja base, de esta forma se puede poner en el interior la placa, la fuente, escondiendo la conexión de los cables y cubriendo a la placa de la humedad y la temperatura. Como se observa en la figura 30 podemos observar que la caja tiene 65,8 [cm] de largo, por 32,8 [cm] de ancho y 5 [cm] de alto. De esta forma tenemos la altura necesaria para que entre un transformador, ya que es el dispositivo más grande que debe estar allí adentro.

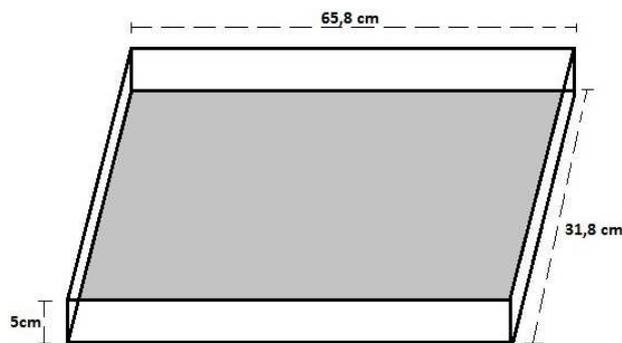


Figura 30: Caja interior para la fuente y placa del controlador

Sobre la caja, va la base la cual tiene 64 [cm] de largo, 30 [cm] de ancho y 5,5 [cm] de alto. En los bordes de la base, se va poner los soportes donde van a estar sostenidas las rieles y el motor paso a paso que va a controlar el eje Y. Los orificios diseñados están a una altura de 2,5 [cm] de esta forma la plataforma movable va a estar en las rieles a 1,5 [cm] de altura para evitar el roce.

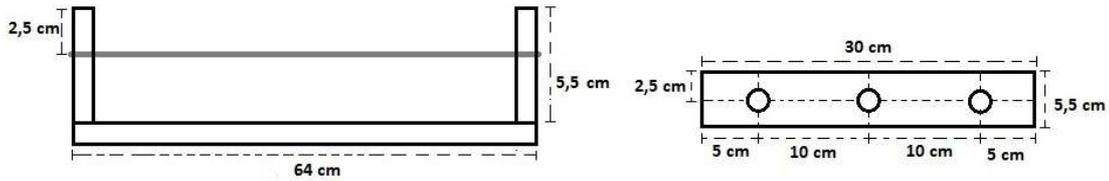


Figura 31: vista lateral y frontal rieles y soportes

La plataforma móvil como se observa en la figura 32, es una plataforma que tiene el largo y el ancho para una placa de tamaño A4. En el caso que la placa sea de menor tamaño ,se ha considerado que debe tener un ajuste para que la placa no se mueva.

En la parte inferior de la misma podemos observar que tiene 3 orificios, en los cuales los dos laterales son por donde los rieles van a pasar para mantener equilibrio, y el orificio del medio es el que contiene la tuerca la que va a trasladar el movimiento rotacional de los motores hacia el movimiento transnacional.

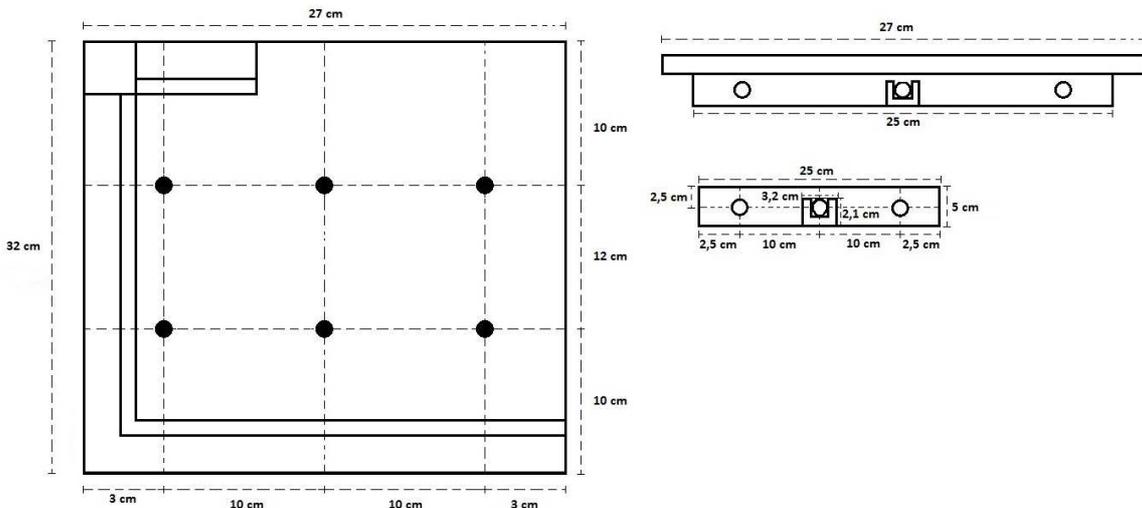


Figura 32: Plataforma móvil

En la figura 33 como se observa es la unión de la plataforma con la base, lo que nos permite observar que si la plataforma se mueve hasta el tope contrario, se mueve la mitad de la distancia de la base.

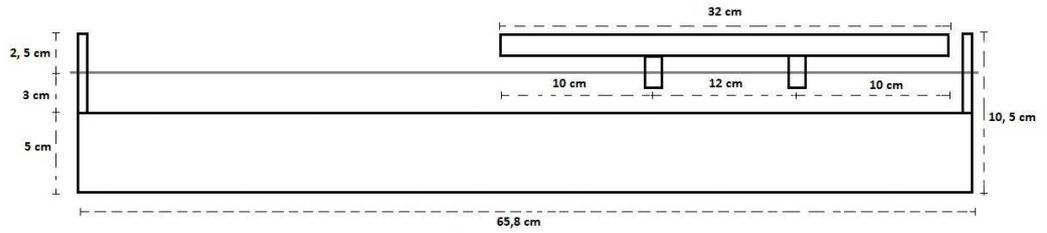


Figura 33: Unión base y plataforma móvil

Al igual que la base, se tiene un soporte superior que nos va permitir que el efector final se mueva en la dirección de X. De igual forma como se observa en la figura 34, podemos encontrar que se tienen dos rieles para mantener el equilibrio y al medio el tornillo milimétrico para poder mover traslacionalmente el efector final.

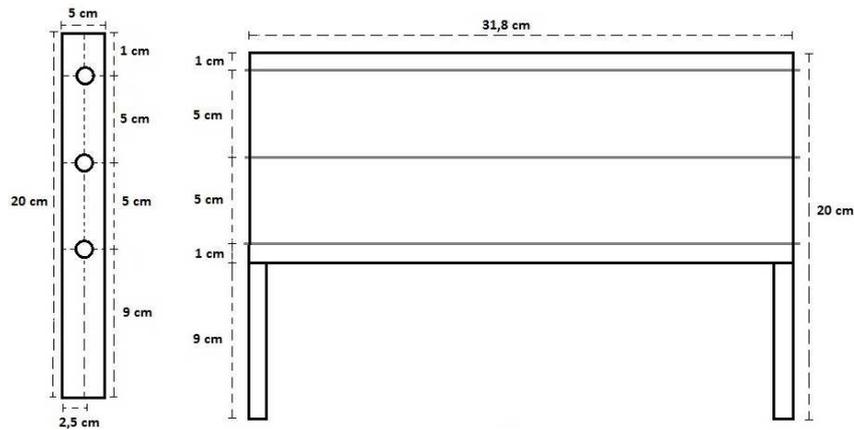


Figura 34: Soporte superior y soporte rieles

Esta sección de la maqueta, tiene por objetivo sostener el motor del eje Z. Este eje nos ayuda a mover hacia arriba y hacia abajo el efector final; de la misma manera, podemos observar que para que no pierda estabilidad está compuesto por dos rieles laterales y un tornillo sin fin para que el efector final suba y baje.

Dentro de los detalles de diseño, se pensó que para que el efector final no se demore mucho subiendo y bajando, se diseñó que el motor paso a paso para controlar éste, esté a un cuarto del tamaño de la placa. Por eso podemos observar que existe un hueco por el cual entra el motor paso a paso para que el rotor pueda ir casi al borde del soporte.

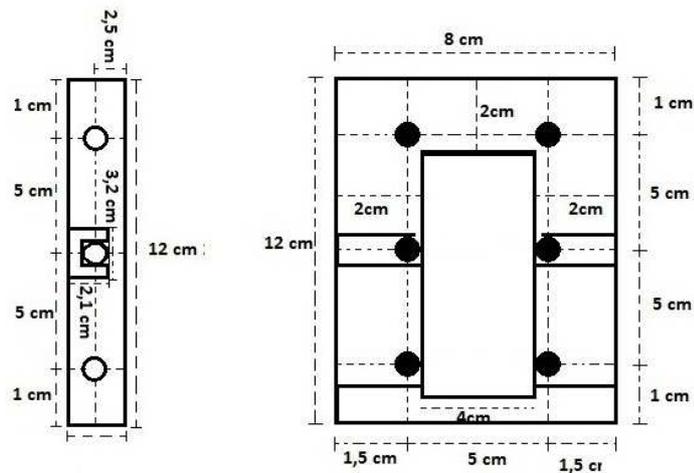


Figura 35: Soporte para el taladro y soporte para rieles superior

Como se observa en las vistas laterales superiores, figura 36 y figura 37, debido al diámetro de nuestro motor DC, el soporte de éste lo ajusta tanto en la parte superior y en la parte inferior, de esta forma ,conseguimos que existan dos puntos de sujeción para que cuando haga presión contra la placa no se salga o no se descentre.

De igual forma podemos observar que los orificios por donde van las rieles son la misma distancia obteniendo un movimiento más suave, haciendo que las rieles sirvan para estabilizar más que para guiar.

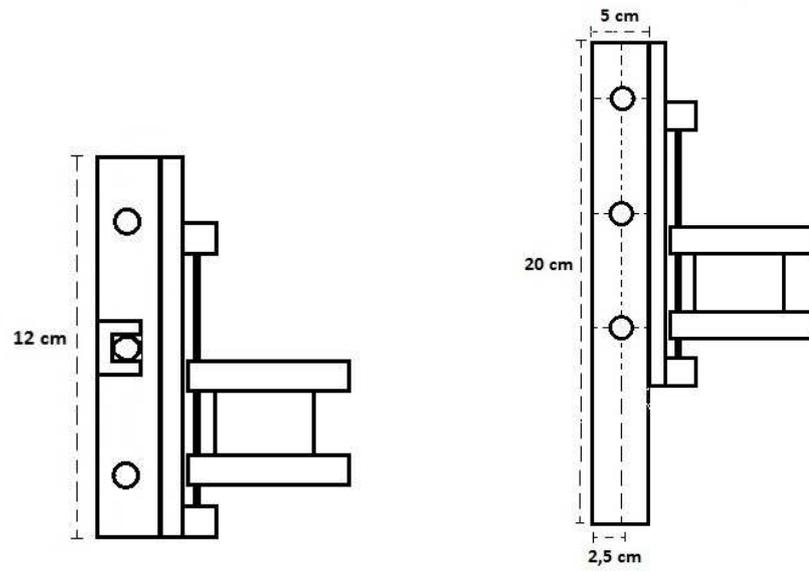


Figura 36: Vista lateral soporte de taladro Figura 37: Vista lateral parte superior

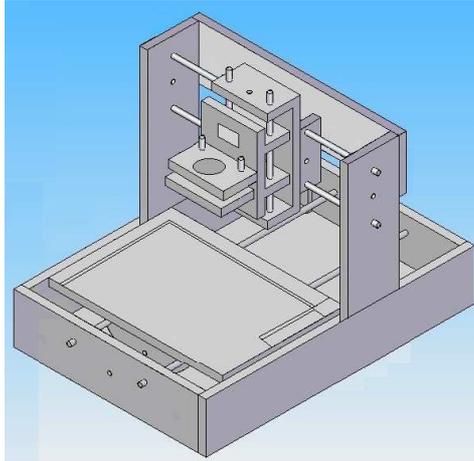


Figura 40: Diseño Maqueta en 3D

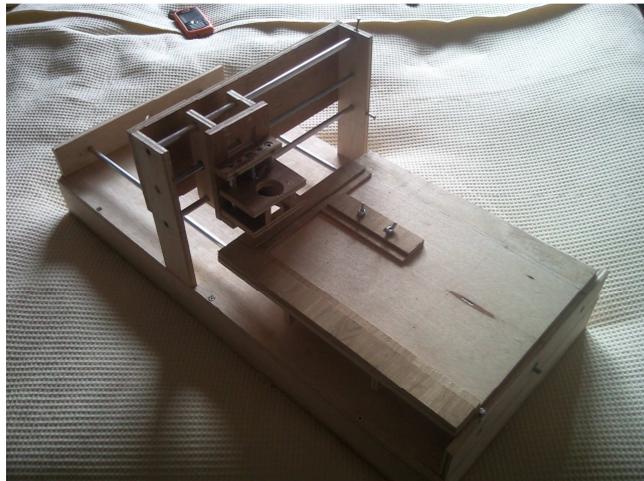


Figura 41: Diseño Real



Figura 42: Fotos Maqueta Terminada

II. Diseño de los circuitos electrónicos y simulación

1. Simulación

Una parte importante del proyecto antes de realizarlo físicamente, es probar mediante un software el funcionamiento del mismo. En el caso de un circuito eléctrico podemos simular usando Proteus. Este programa nos permite usar los dispositivos que vamos a utilizar en la vida real y simular el funcionamiento de nuestro circuito.

En nuestro caso el Proteus nos permite usar la programación realizada en el Pic, de esta forma el Pic virtual va a realizar lo mismo que el Pic físico.

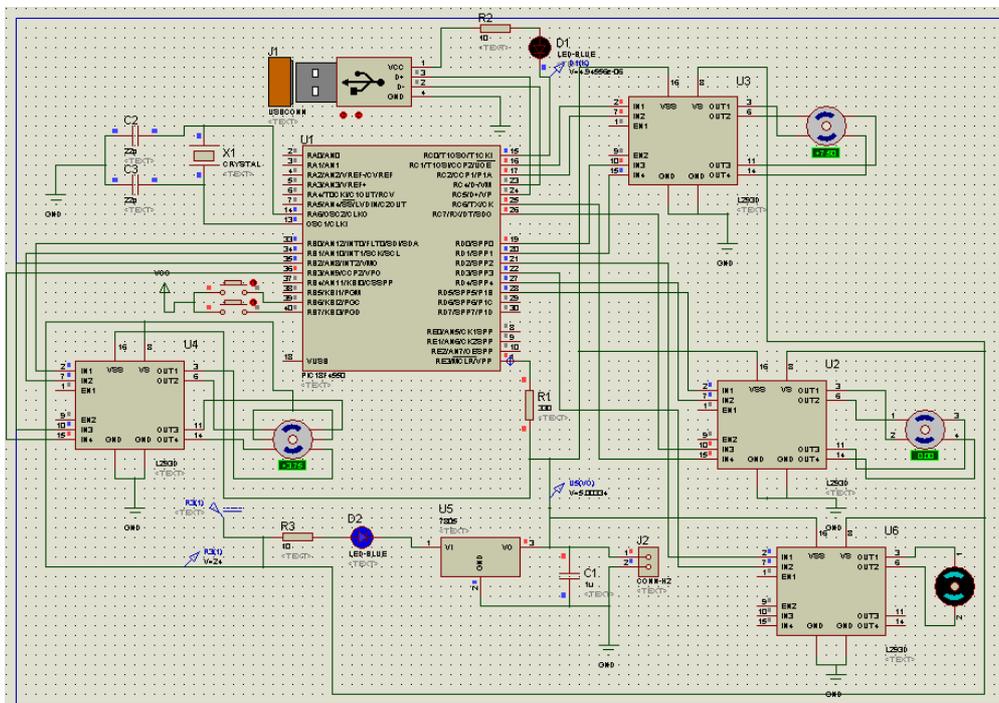


Figura 43: Circuito realizado en Proteus

Para simular el circuito, previamente se realizó la interfaz figura manual, de esta forma usando la interfaz figura podemos enviar las señales de movimiento hacia el Pic y poder observar en la simulación del Proteus. Como se observa en la figura 44, podemos observar la interfaz figura, la parte manual. De esta forma podemos usar los 5 comandos para saber que la programación de los motores está correctamente realizada.

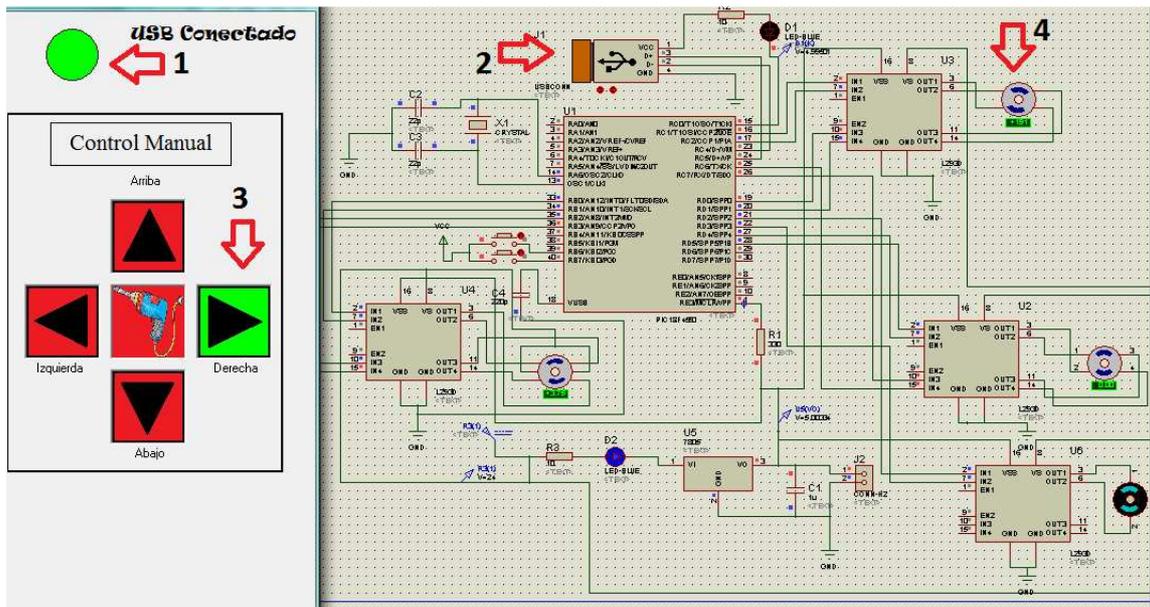


Figura 44: Simulación del circuito Eléctrico

Como podemos observar en este ejemplo:

- 1 → Confirmación de conexión USB
- 2 → USB Conectado en el circuito simulado
- 3 → Botón Derecho presionado
- 4 → Motor eje X Girando hacia la derecha

De igual forma si usamos los otros botones, como el arriba y abajo, el motor 2 se va a mover a un lado y al otro según el botón presionado. Si presionamos el botón de taladro, vamos a observar que el motor DC comienza a girar y el motor 3 (izquierda de la figura 44) va a moverse. De ésta manera, podemos comprobar la programación realizada en el Pic Basic de igual forma la comunicación, entre la computadora y el Pic virtual.

En la simulación, es posible comprobar la comunicación debido a que la misma programación del Pic real, está montada en el Pic virtual. Como se vio previamente en la programación del Pic generada por el Easy HID, nos indica el llamado al driver para que la computadora lo detecte. Así mismo cuando se corre el programa por primera vez el driver es virtualmente instalado simulando la comunicación.

2. Diseño de los circuitos

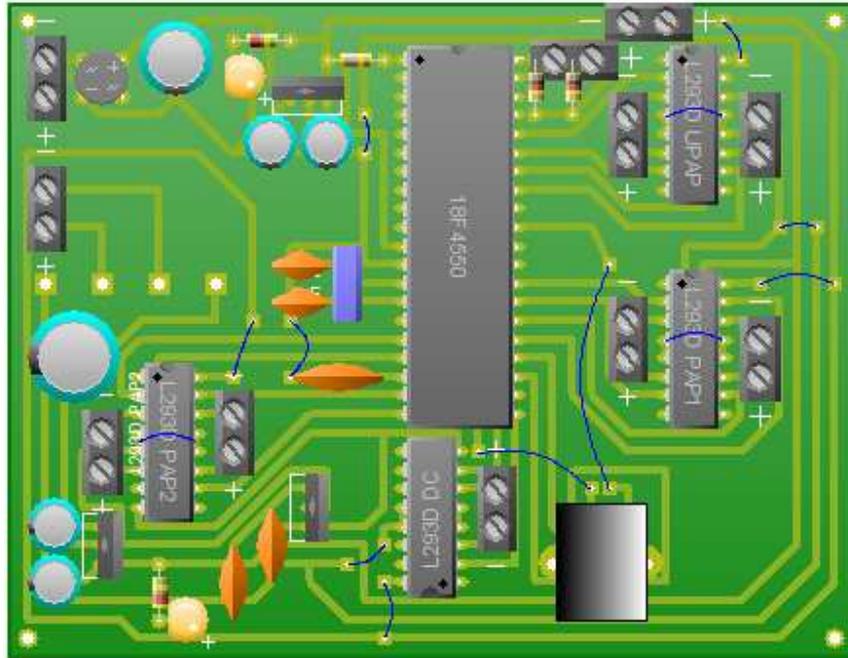


Figura 45: Circuito fuente y controlador de motores

En este circuito, podemos observar claramente 7 partes fundamentales. La primera que es la parte de la fuente en la que tenemos una bornera para la conexión de los cables del transformador 110V – 12V de 2 Amperios, luego existe un puente de diodos para rectificar la señal haciendo que deje de ser alterna haciéndola continua. Tenemos un 7824 con dos capacitores para filtrar la señal, ya que nuestros motores para óptimo funcionamiento como se observó en las hojas de datos, usan 24 [V]. En serie ponemos un LM7824 y un LM7812 ya que el motor DC no se lo puede forzar con 24 [V] debemos reducir a 12 [V] para que no se quemara.

En los anexos en la parte superior podemos encontrar el mismo circuito, pero con la pequeña modificación, que en vez de usar un LM7824, usamos un LM7805 para poder estabilizar la señal en 5 [VDC], ya que nuestro microcontrolador funciona con voltaje lógico 1, es decir, 5 voltios en corriente continua.

En nuestra tercera parte del circuito podemos observar como componente principal de toda la placa al PIC18F4550. Este dispositivo es el que va a controlar los 4

motores que por medio de programación, las entradas y salidas conectadas a este dispositivo se van a poder controlar el robot XY.

A lado del PIC se puede observar que se tiene dos puentes H, los cuales son los controladores de los motores paso a paso que se usa. La diferencia del superior con el inferior es que el uno controla un motor paso a paso bipolar (eje Z) y el otro un motor unipolar (eje Y).

En la parte interior de la placa se puede notar claramente en el circuito que existe un puente H que nos permite conectar el motor DC del taladro ya que en las especificaciones máximas de este motor, indica que el voltaje máximo debe ser 12V. Como este motor es el que va a realizar los orificios éste debe tener mayor potencia para realizar la acción deseada, es por ésta razón que se pone su uso al máximo voltaje.

Debajo de la fuente, se observa un circuito similar al del motor del eje Y. esto se debe a que es el motor del eje X ya que para evitar problemas se buscó motores con las mismas características, en el plano XY para no tener que realizar programaciones diferente y evitar utilizar distintos voltajes y otros elementos.

Adicionalmente se tiene una bornera para conectar los sensores que nos van a ayudar a identificar el inicio de carrera, es decir, con estos vamos a poder encontrar los puntos 0,0 de nuestro sistema de coordenadas en el plano XY.

Así mismo para acoplar el USB, se debe conectar el D+ y el D- del Pic; la tierra al circuito y el voltaje a un pin para poder distinguir cuando el USB está conectado o desconectado.

III. Programación

1) Explicación Programación del Microprocesador

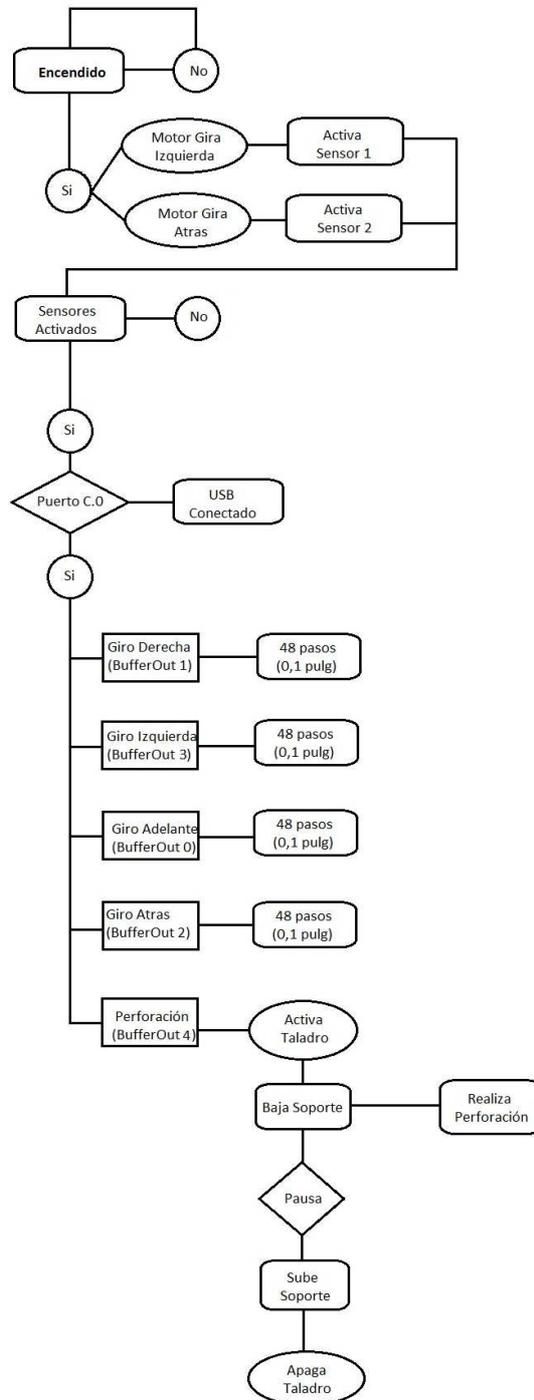


Figura 46: Lógica de programación del Pic

La programación del Pic, como se mencionó antes, se lo realiza en PicBasic Pro, que es un programador, que nos permite utilizar Basic, en vez de Assembler, para facilitar la programación.

Entre las instrucciones más utilizadas podemos encontrar:

- Definición de variables
- Subrutinas
- Rutina principal del microcontrolador.

Dado a que debemos siempre comenzar desde el punto (0,0), debemos usar sensores de micro contacto, lo que nos permite posicionar al eje X y al eje Y en los puntos de inicio. Como se observa en el diagrama de bloques, si los sensores no están activados, el motor gira hacia la izquierda cuando este llega al sensor, el motor hacia atrás se moviliza hasta activar el sensor. De esta forma conseguimos siempre tener el mismo punto de partida para todas las aplicaciones, sean manuales o automáticas.

Una vez alcanzado el inicio de carrera, podemos comprobar si existe la conexión USB, debido a que el cable USB, tiene un cable de 5v lógicos lo podemos usar para enviar a una entrada y saber que si ese cable está o no activo. Por lo tanto, si esa entrada no está activada, significa que el cable USB está desconectado, por lo que no puede haber transmisión de datos provocando que ninguna rutina comience.

Como se observa en los bloques, si el USB está conectado, el puerto C.0 se va a activar permitiendo que comiencen las rutinas.

Hemos realizado la programación para cada lado y para la perforación. Como se observa en el diagrama de bloques:

- Giro derecha → BufferOut[1]
- Giro Izquierda → BufferOut[3]
- Giro Adelante → BufferOut[0]
- Giro Atrás → BufferOut[2]
- Perforación → BufferOut[4]

Para el control de los giros derecha, izquierda, adelante y atrás, se ha programado e funciones con la programación Half Step. De esta forma podemos controlar más precisamente el movimiento. Entre las especificaciones de la fabricación de placas, el espacio mínimo entre dos puntos debe ser 2,54 [mm] o 0,1 [inches], por lo que cada movimiento del motor alcanza la distancia determinada.

Usando la programación FOR, podemos hacer que tenga repeticiones lo que provoca que el motor gire las veces que sean necesarias. En nuestro caso el motor debe dar dos vueltas para moverse 2,54 [mm]. De ésta forma podemos determinar que cada vez que se aplaste un botón en el Visual Basic, el motor va a moverse ésta distancia.

Para el motor de perforación y evitar tener doble control se ha programado de tal forma que cuando uno active la función de taladro, éste sea encendido , baje el soporte mientras baja va taladrando luego afina el orificio, es decir, hace que el orificio esté perfecto y finalmente suba el soporte apagando el motor.

En términos de programación, como se observa en el diagrama de bloques, cuando el Buffer de salida 4 es activado, se prende el motor DC luego el motor paso a paso de Z realiza la acción FOR, determinada por tiempos para bajar, se pausa y realiza la ascensión por el mismo tiempo, una vez arriba apaga el motor del taladro y así completamos la perforación.

Adicionalmente se ha programado la velocidad de movimiento de los motores según el gusto del usuario. Si el buffer 6 es activado, la velocidad de los motores se hace más rápida, ya que la pausa entre cada intervalo se disminuye haciendo que las señales activen más rápido las bobinas y también permitiendo que los motores giren a más velocidad. . Esto nos permite tener una movilización más rápida, pero menos precisa en comparación de la movilización normal que toma un poco más de tiempo.

En los anexos, se encuentra toda la programación, debido a que son más de 400 líneas de código por lo que no se las pone en esta sección. De igual forma se encuentra explicado el funcionamiento de cada línea.

2) Diseño de la Interfaz gráfica

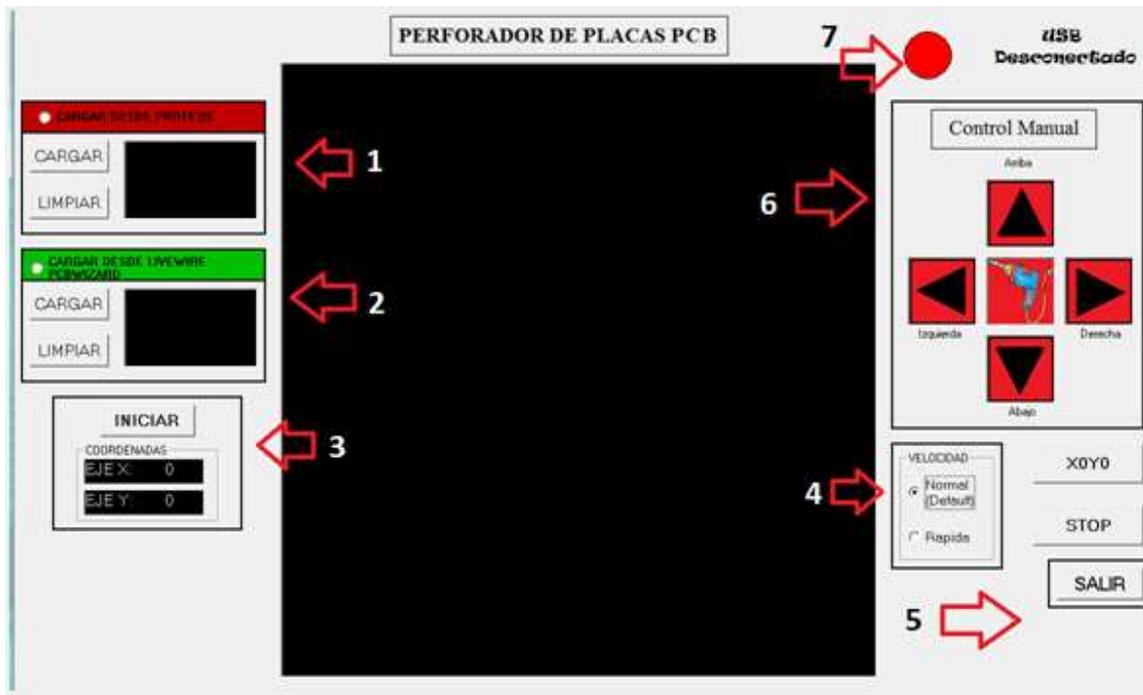


Figura 47: Interfaz Figura

Entre los objetivos planteados previamente, la interfaz figura tiene que:

- Obtener los puntos XY por medio del Proteus y del Livewire - PCB Wizzard.
- Generar un gráfico de los puntos a ser taladrados.
- Que realice todas las perforaciones de la placa realizada en Proteus o Livewire - PCB Wizzard.
- Tenga un control manual de los movimientos.
- Muestre una interface de comunicación.

Como se puede observar en la figura 47, podemos ver la interfaz gráfica generada en Visual Basic. Para la fácil comprensión de la interface se ha enumerado las partes más importantes ya que son las que nos permitirán cumplir los objetivos.

- Interface en la que nos permite cargar los datos desde el Proteus. Al momento de cargar el archivo, entramos al explorador de archivos lo que nos permite buscar el archivo texto generado por el Proteus guardado en los documentos del usuario. Una vez aceptado el código, éste se copia en el recuadro (lst1Codigo), el cual es un list que nos permite observar los datos obtenidos. El momento de aceptar el código, este

genera los puntos en el gráfico permitiéndonos observar los orificios a ser realizados. Si se oprime el botón limpiar, este borra los datos del lst1Codigo y vuelve la pantalla del picture a su estado inicial.

- 2) Interface que nos permite cargar los datos generados por Livewire - PCB Wizzard. De igual forma carga el archivo y copia los datos en el lst2Codigo, lo que nos permite observar los datos y precisar los puntos de perforación. De igual manera el botón limpiar borra los datos del lst2Codigo y pone la pantalla en su estado inicial.
- 3) El botón iniciar, comienza la perforación automática y nos permite observar las coordenadas en las que está perforando. Cuando ha llegado a un punto en el gráfico, se cambia el punto blanco a un punto rojo, lo que nos indica que la perforación ya se ha realizado.
- 4) La velocidad de perforación, nos permite mover más rápido los motores para alcanzar el punto deseado más rápido. Esta opción solo se encuentra activada para el control manual.
- 5) Botón STOP, nos permite parar la perforación, mueve los motores al punto inicial, borra el list que se esté usando y pone la pantalla en su estado inicial. Botón X0Y0 nos permite ir al punto de inicio (esta opción, está disponible solo en control manual). Botón Salir, sale de la aplicación.
- 6) Control manual, aquí podemos observar las flecha, escogiendo el movimiento de los motores. El botón central activa la función de taladrar; cuando se activan los botones, se ponen en verde indicando que el motor está en funcionamiento.
- 7) Indicación de la conexión, esta opción nos indica cuando el USB está conectado: círculo en color verde, e indicación de USB conectado. Cuando está desconectado: círculo en color rojo e indicación de USB desconectado.

El diseño de la interfaz gráfica, nos permite tanto controlar como observar el proceso de perforación. Todas las indicaciones de que realizar cuando cada botón es activado, se lo ha programado, de ésta forma podemos lograr los objetivos planteados tanto para la interfaz gráfica como para el proyecto.

3) Explicación Programación del Visual Basic

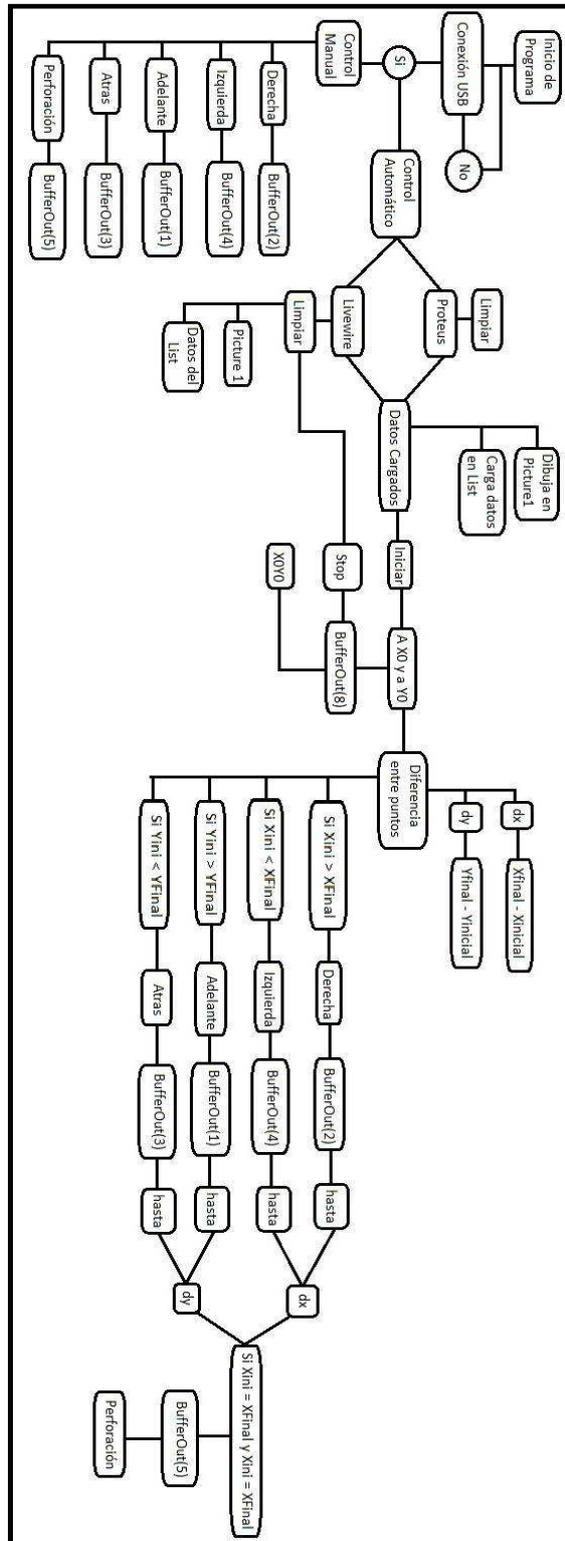


Figura 48: Lógica de Programación de Visual Basic

La programación del visual Basic, tiene dos etapas principales. La primera es el control manual y la segunda es el control automático, en el cual como se verá más adelante debe adquirir los datos, determinar cuándo se debe mover, qué motor activar, etc.

En el control manual, son 5 órdenes que se debe dar: moverse a la derecha, a la izquierda, adelante, atrás y perforar. En cada uno, cuando el botón es aplastado, cambia de color el fondo, para saber que se lo está oprimiendo y que envíe la orden por medio del buffer de activar. Cuando se suelta el botón y termina el movimiento, manda la orden de desactivar el buffer y cambie el color, lo que indica que se puede usar nuevamente. Como podemos observar en la tabla 5, es una relación entre el buffer del Pic y el buffer usado en visual Basic

Tabla 5: Relación de buffers Pic – VB6		
Buffer Pic	Acción	Buffer VB6
USB Buffer[0]	Adelante	BufferOut (1)
USB Buffer[1]	Derecha	BufferOut (2)
USB Buffer[2]	Atrás	BufferOut (3)
USB Buffer[3]	Izquierda	BufferOut (4)
USB Buffer[4]	Perforación	BufferOut (5)

Como previamente se mencionó, cada vez que activamos un desplazamiento, la distancia de movimiento va a ser de 2,54 [mm] o 0,1 [inches], y siempre va a comenzar desde el punto 0,0

La segunda parte mencionada es el control automático. Aquí tenemos dos opciones, la primera es usar los datos del Proteus o los datos del Livewire - PCB Wizzard - PCB Wizzard. Si escogemos el Proteus, tenemos la opción para cargar, lo que nos va a mandar a buscar el archivo generado por Proteus que es un archivo de texto. Este archivo de texto como podemos observar en la figura 49 nos entrega las

coordenadas XY. Podemos observar que las coordenadas comienzan a partir de la cuarta línea, lo que nos va a permitir situar los motores.

```
M48
T01C0.0300
%
T01
X+004000Y-001000
X+010000Y-008000
X+010000Y-007000
X+010000Y-006000
X+010000Y-005000
X+010000Y-004000
X+010000Y-003000
X+016000Y-011000
M30
```

Figura 49: Archivo .txt Proteus

En el código generado por Proteus, podemos encontrar varias instrucciones y detalles que nos permiten tanto comenzar la perforación como la finalización de la misma.

- M48.- Inicia el programa
- T01.- Broca que se va a usar (depende del diámetro tenemos T02, T03,...,T06)
- X+000000Y-000000.- 16 dígitos para dar el punto en el plano
- M30.- Finalización de la perforación.

Las instrucciones que se van a usar son las dos últimas, ya que por medio de programación podemos indicar que comience a leer el archivo desde la 4ta línea. La última línea nos va a indicar que termina la perforación, que ya no busque más puntos y que se movilice al punto 0,0.

Como se observa en el archivo de texto, podemos observar que se usa el 4to cuadrante, es decir, X positivo y Y negativo. Los signos de los puntos no es problema, ya que usando la opción MID de Visual Basic podemos extraer los puntos a partir del signo hasta el que uno necesite. Se ha programado que a partir del segundo dígito a partir del X, obtenga los 2 siguientes dígitos, es decir, que tome 3 números, de igual forma para Y. Esto nos va a entregar un número de tres cifras tanto para X como para Y.

De esta forma encontramos los puntos XY finales, así podemos usar un FOR para que realice las acciones hasta el último X que encuentre en el archivo. Usando estos puntos podemos graficar en la imagen usando los mismos puntos que se observa en el Proteus. Con esta opción podemos observar si el archivo cargado es el correcto, o si se necesita poner más puntos o quitar otros.

Si es que el botón limpiar es presionado, el archivo de texto el cual fue insertado en el listado y el gráfico generado, se van a borrar, de esta forma podemos insertar otros archivos.

De igual forma, tenemos la misma opción para el Livewire - PCB Wizzard, la diferencia se encuentra en el tratado del archivo. Como se observa en la figura 50, el archivo generado por el Livewire - PCB Wizzard - PCB Wizzard es muy similar al del Proteus. De igual forma tenemos el inicio de programa, el fin de programa, el tipo de broca y el punto XY.

```
M48
%
T01
X022000Y-010000
X022000Y-009000
X022000Y-008000
X022000Y-007000
X022000Y-006000
X022000Y-005000
X022000Y-004000
X022000Y-003000
X022000Y-002000
X022000Y-001000
X013000Y-021000
M30
```

Figura 50: Archivo .drl Livewire - PCB Wizzard - PCB Wizzard

El archivo generado por Livewire - PCB Wizzard, nos comienza a dar las coordenadas a partir de la 4ta línea, y en vez de ser 16 dígitos son 15 ya que obvia el signo en la coordenada de X. De igual forma usando la opción MID, tomamos 3 dígitos a partir del primer cero desde la izquierda, y para Y es desde el segundo dígito a partir del primer cero. De esta forma conseguimos los puntos XY desde el Livewire - PCB Wizzard.

De igual forma estos puntos son graficados en la pantalla, siempre como referencia para el usuario.

Una vez cargado el archivo, ya sea Proteus o del Livewire - PCB Wizard, o según el que se escoja, la opción de iniciar comienza el trabajo automáticamente.

Primero cuando uno aplasta iniciar, verifica que archivo esté cargado luego activa el BufferOut (8) el cual hace que los motores vayan directamente al punto 0,0. Como se explicó en la programación del Pic, existe una subrutina en la que mueve los motores hasta que los sensores se activen.

Una vez en los puntos 0,0 inicia buscando la primera coordenada donde realiza una diferencia del $X_{final} - X_{inicial}$ y $Y_{final} - Y_{inicial}$, lo que nos entrega la distancia entre los dos puntos; de ésta forma sabemos cuántas unidades debe moverse el motor en X y en Y hasta que consiga igualar las coordenadas finales con las iniciales.

En la tabla 6 como podemos observar, encontramos qué motor debe activarse, cuando la diferencia del final con el inicial es positiva o negativa.

Tabla 6: Movimiento de acuerdo a las coordenadas						
$X_{inicial}$	>	X_{final}	Izquierda	BufferOut (4)	Hasta	D_x ($X_{final} - X_{inicial}$)
$X_{inicial}$	<	X_{final}	Derecha	BufferOut (2)		
$Y_{inicial}$	>	Y_{final}	Adelante	BufferOut (1)		D_y ($Y_{final} - Y_{inicial}$)
$Y_{inicial}$	<	Y_{final}	Atrás	BufferOut (3)		
$X_{inicial}$	=	X_{final}	Perforación	BufferOut (5)		
$Y_{inicial}$	=	Y_{final}				

De esta forma hacemos que cuando el punto final con el punto inicial sean iguales, o la diferencia sea 0, perfore, siempre y cuando cumpla con que las dos coordenadas sean iguales.

Cuando inicia el segundo recorrido, sabemos que los motores se pararon en el punto XY anterior, por lo que ese es nuestro nuevo punto inicial, de esta forma repetimos el mismo proceso hasta que llegue al final.

De igual forma el código de programación se lo puede encontrar en los anexos, ya que posee más de 600 líneas de código. Se puede encontrar la programación con su respectiva explicación de cada línea de código.

IV. Comunicación Física

Como parte importante del proyecto, está la comunicación entre el Ordenador y el Microcontrolador para que realice las acciones deseadas.

El cable que se usa es un cable de 4 hilos blindado, como se observa en la figura 51, tiene:

- Cable rojo, voltaje lógico (+5V)
- Blanco, transmisión de datos negativo
- Verde, transmisión de datos positivo
- Negro, Tierra o común

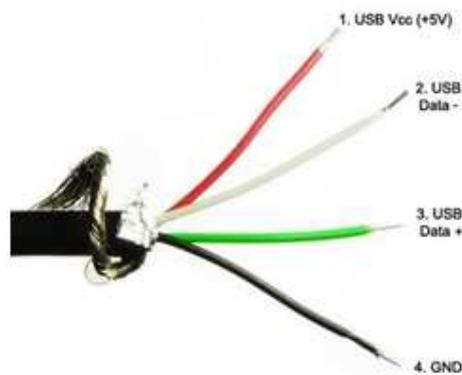


Figura 51: Cable USB

Como previamente se mencionó el cable rojo lo usamos para conectar al Pic, de esta forma cuando el USB esté conectado, el Pic sepa que puede realizar cualquier acción, ya que el USB está habilitado.

El data – y el data + se los conecta en los pines 23 y 24 del microcontrolador 18F4550 para que la información vaya y vuelva cuando se necesite intercambiar información.

El último cable, el cable ground, se lo conecta al común de toda la placa.

El cable que se va a usar, es un cable del tipo cuadrado ya que es el más común para dispositivos de control. Esto se debe a que puede existir confusión al momento de conectar el ordenador con el dispositivo y viceversa, el cable es directo e internamente se cruza. Como se observa en la figura 52, podemos observar que el cable que se va a conectar al robot, es del tipo b, es decir, cuadrado. El zócalo de conexión es cuadrado y los pines de conexión se observan en la figura.

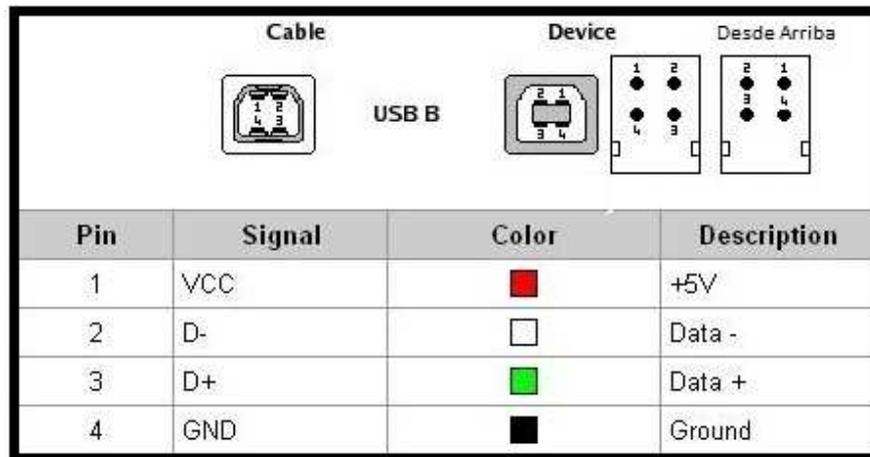


Figura 52: Conexión del dispositivo

CAPITULO 3: FUNCIONAMIENTO Y ESPECIFICACIONES (Manual de Usuario)

En esta sección, se va indicar como es el funcionamiento tanto con el Livewire - PCB Wizzard y el Proteus – Ares. De igual forma se va a comparar y se va a confirmar la igualdad en los puntos de los dos programas y se va a explicar el funcionamiento del control manual.

- Utilizando Proteus - Ares
- Utilizando Livewire - PCBWizzard
- Comparación Proteus – Ares con Livewire - PCBWizzard
- Control Manual

I. Utilizando Proteus - Ares

La configuración del Proteus se la realiza en la pantalla principal del generador de circuitos impresos. Como se mencionó antes, el editor de circuitos impresos del Proteus se llama Ares. Antes de comenzar a utilizar el Ares, como se observa en la figura 53, primero se debe determinar el espacio entre cada punto, es decir, que debemos configurar que entre punto y punto exista la distancia necesaria para que no se choquen los caminos.

La distancia mínima que se debe utilizar es de 2,54 [mm] o 0,1 [inches] como norma que se encuentra en la mayoría de editores de circuitos. De esta forma podemos configurar que la misma distancia entre puntos tanto en el Livewire - PCBWizzard como en el Proteus - Ares la cual va a ser la misma, por lo tanto la distancia de movimientos de los dos motores va a ser exactamente como se ha calibrado inicialmente para los dos programas.

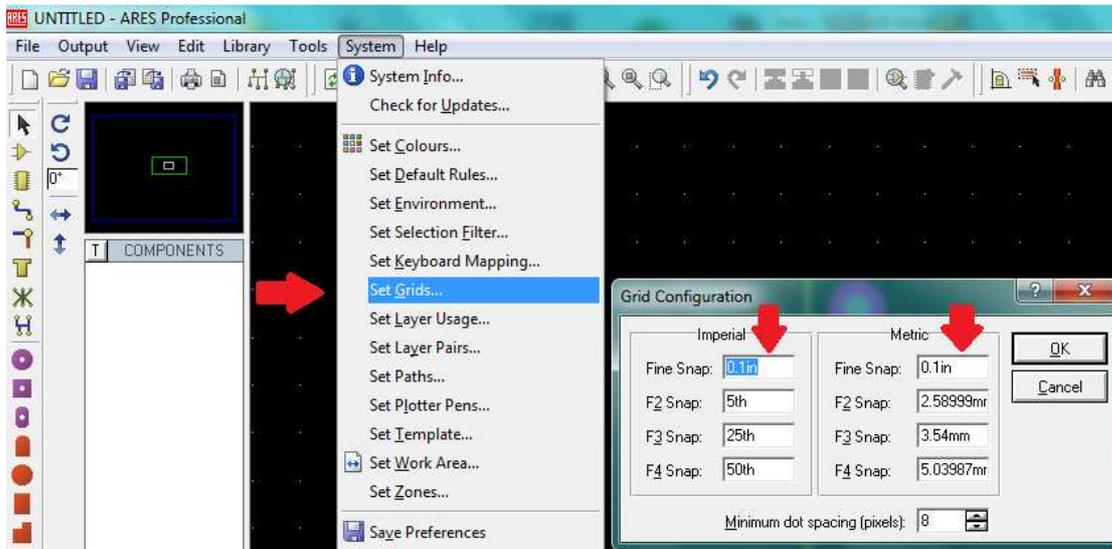


Figura 53: Configuración Proteus

Una vez configurada la distancia entre los puntos, se debe configurar el área de trabajo, esto nos sirve, ya que debemos utilizar un cuadrante para determinar: primero los puntos (0,0) y determinar los puntos (X, Y) para los componentes que se van a insertar. Como se observa en la figura 54, nos indica cómo poner el área de trabajo. Observamos que la flecha superior nos indica que se va a poner un rectángulo de área de trabajo, y de placa de caminos inferior.

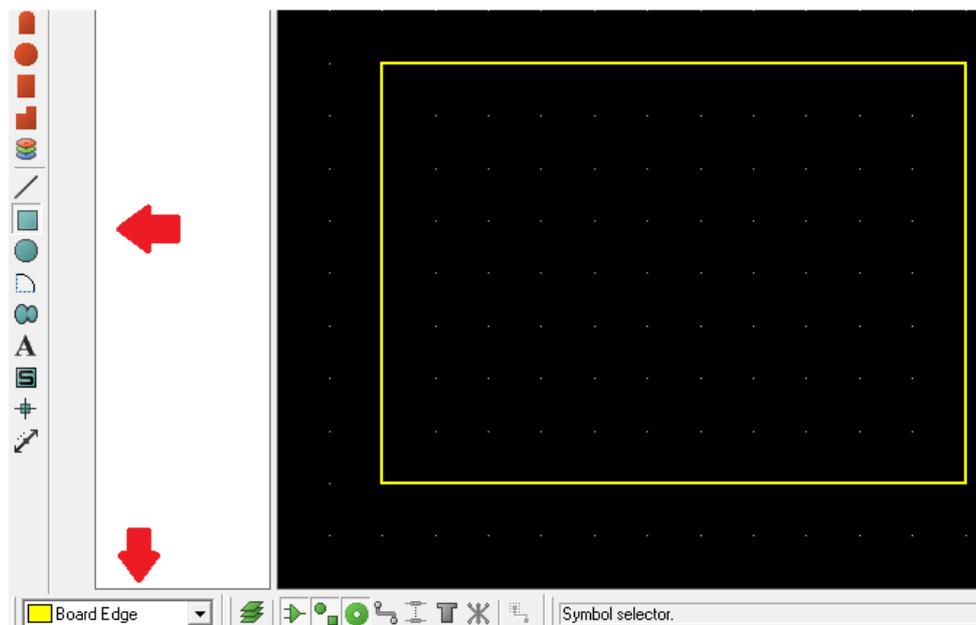


Figura 54: Área de trabajo

Una vez definida nuestra área de trabajo, debemos determinar el punto de inicio. Ya que se ha configurado nuestro robot para que trabaje desde arriba hacia abajo y se ha determinado que el trabajo a realizar sea en el cuarto cuadrante. Esto significa, que las coordenadas en el eje X son positivas y la coordenadas del eje Y son negativas. Esto significa que va a realizar un barrido de arriba hacia abajo y de izquierda a derecha.

Como se observa en la figura 55, en la opción output/ Set Output Origin → y nos da la opción de colocar el punto de origen donde uno desee. De esta forma ponemos nuestro punto de origen en la esquina superior izquierda.

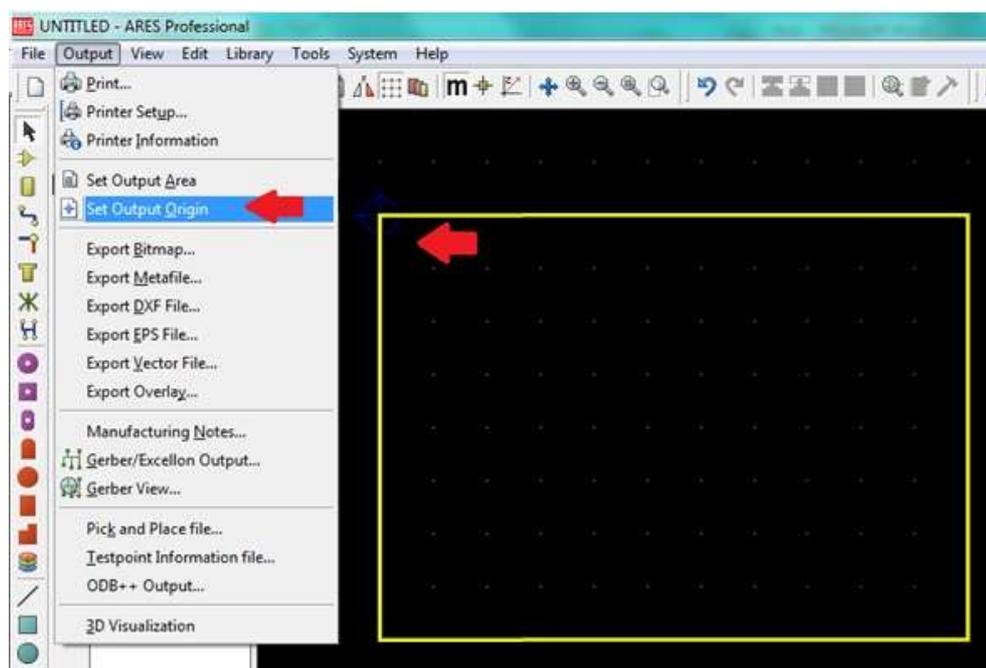


Figura 55: Determinación punto de origen

Una vez determinado nuestro punto de origen y nuestra área de trabajo podemos insertar los dispositivos eléctricos que previamente usamos en la simulación. Como se observa en la figura 56, podemos observar los dispositivos en posición para poder generar los archivos de puntos XY.

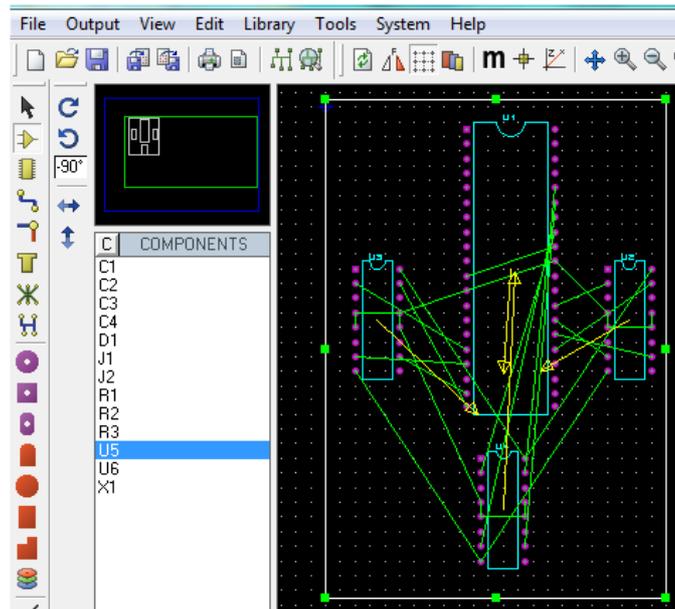


Figura 56: dispositivos eléctricos situados en la placa

Cuando se ha finalizado con todos los componentes, se puede generar el autorouting, el cual nos ayuda a generar automáticamente los caminos. Como se observa en la figura 57, observamos el botón del autorouting que nos genera los caminos automáticamente.

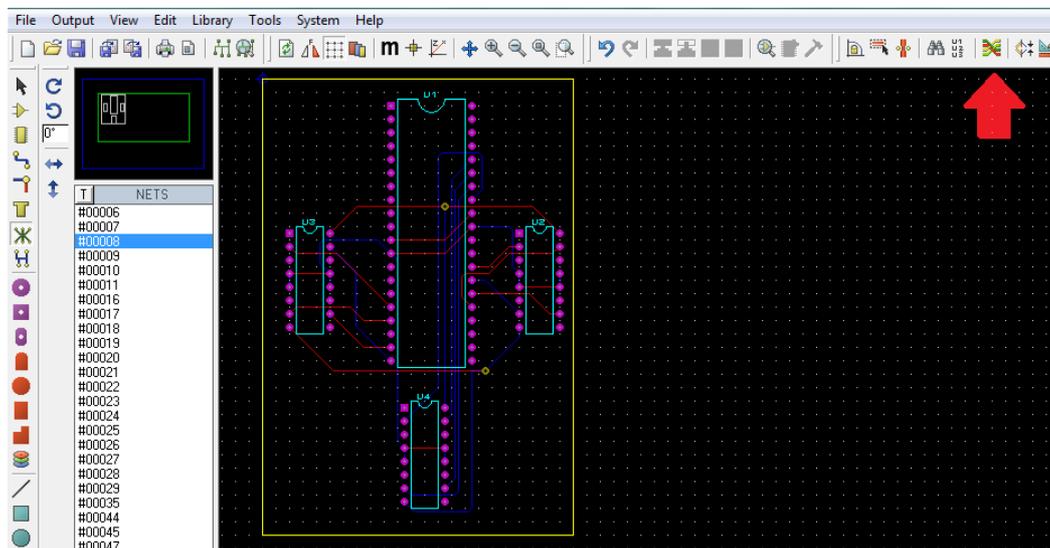


Figura 57: Generación de caminos placa

Finalizado este proceso, ya podemos comenzar tanto a imprimir como a modificar la placa.

Una vez aceptado el formato y la calidad, las uniones y todos los detalles de la placa, procedemos a crear el archivo, el cual va a contener toda la información sobre los puntos y los diámetros de los orificios a realizar. Para esto nos vamos a Output/GerberExellon Output, haciendo clic, nos entrega la pantalla que se observa en la figura 58 donde vamos a proceder a poner:

- El nombre del archivo
- El lugar donde se va a guardar el archivo
- Qué tipo de archivo se va a generar, en nuestro caso es un tipo drill
- Las unidades, imperial que nos entrega puntos XY con números en miles, lo que nos permite tener mayor precisión.

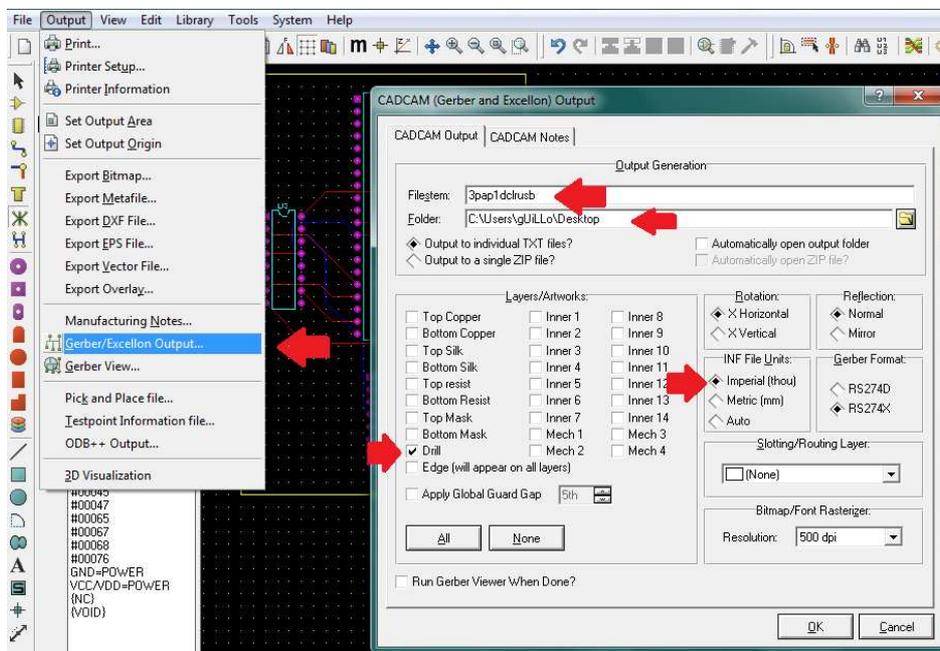


Figura 58: Generación de archivo drill

Una vez creado el archivo, podemos proceder a abrir el archivo con el block de notas, para poder observar los puntos generados. Como se observa en la figura 59, podemos ver las partes que se indicó previamente sobre este archivo, obteniendo los puntos donde se debe taladrar.

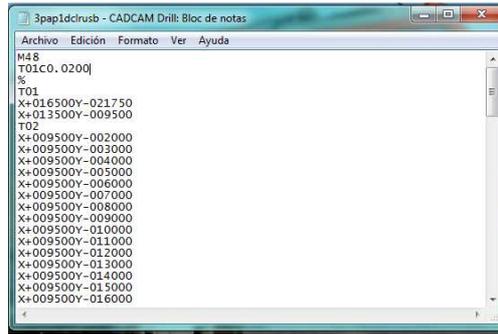


Figura 59: archivo generado por Proteus - Ares

Una vez generado el archivo procedemos a abrir la interfaz visual, es decir el programa generado en Visual Basic. En este programa como se mencionó previamente los componentes, podemos proceder a escoger:

- Clic en Cargar desde Proteus
- Buscamos el archivo generado y lo cargamos a la interfaz.

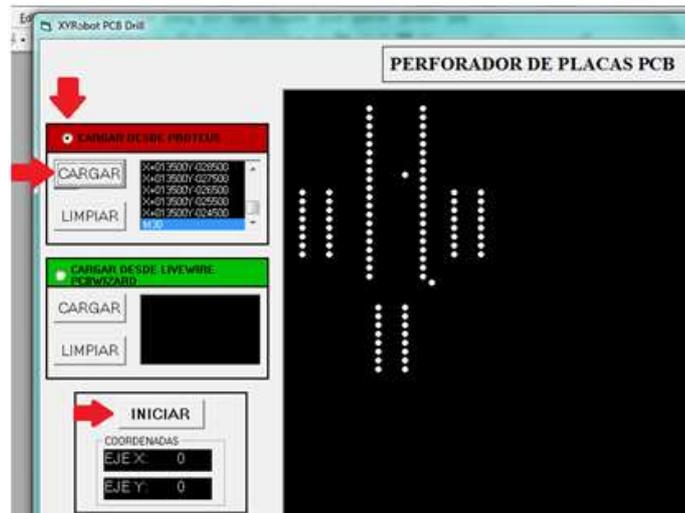


Figura 60: Cargar el Archivo y comenzar perforación

- Verificamos con el Proteus - Ares que los puntos generados con los mismos

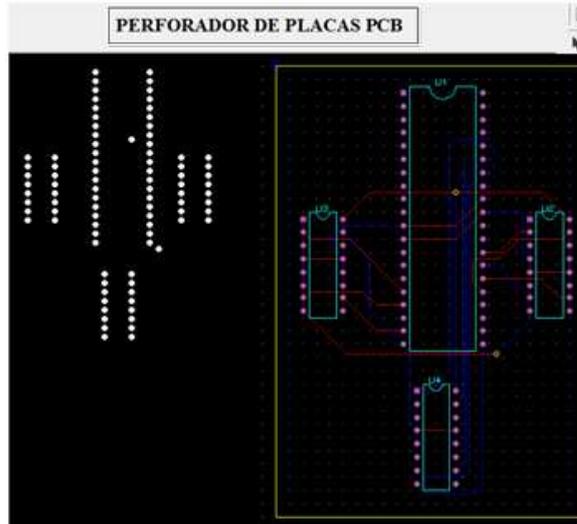


Figura 61: comparación puntos Proteus - Ares e interface visual

- Confirmamos que se encuentra en el Punto (0,0)
- Procedemos a hacer clic en Iniciar

Cuando finalice la perforación, un mensaje de confirmación nos va a mostrar para indicar que los puntos generados en el programa se han realizado con éxito. Un detalle importante, es que cada vez que se realice un orificio, la interfaz figura nos va a cambiar de color el punto taladrado. De esta forma podemos observar como procede la máquina para ubicar y taladrar punto por punto.

De igual forma debajo del botón de iniciar, nos va a indicar que coordenadas está taladrando, de esta forma si ocurre algo, podemos modificar el archivo y taladrar desde las últimas coordenadas taladradas.

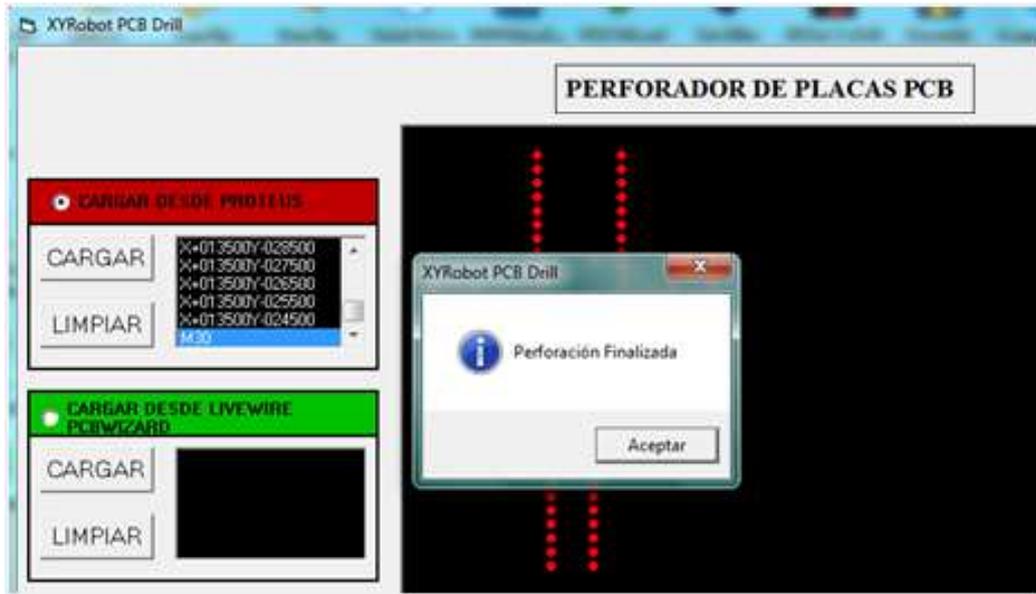


Figura 61: Confirmación Perforación finalizada

II. Utilizando Livewire - PCB Wizard

De igual manera, como se trató al archivo del Proteus - Ares, podemos tratar al archivo que nos genera el Livewire - PCBWizard, de tal forma que los dos archivos sean lo más iguales posible. Esto significa, que en preferencia tengan la misma extensión, que cada coordenada tenga la misma cantidad de dígitos, y que sean los mismos puntos.

En el Livewire - PCBWizard, de igual forma debemos trabajar con las mismas distancias y mismos detalles que el Proteus - Ares. Como se observa en la figura 62, podemos escoger la distancia entre punto y punto. Si bien recordamos en el Proteus - Ares, se puso que la distancia sea de 0,1 pulgadas, de igual manera, se puede realizar en el Livewire - PCBWizard. Si hacemos clic en VIEW/GRID_SNAP/0.1IN obtenemos que la distancia va a ser la misma que en el Proteus - Ares, por lo tanto los movimientos de los motores van a ser idénticos para cada archivo.

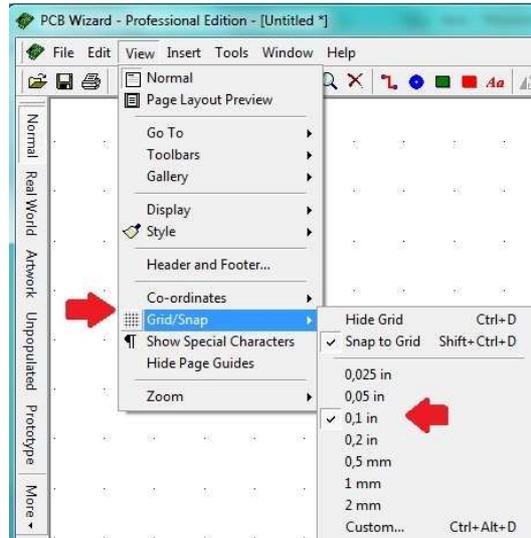


Figura 62: Configuración distancias Livewire - PCBWizzard

De igual manera, debemos configurar el área de trabajo, el punto de origen y el sistema métrico que se va a usar. Como se observa en la figura 63, entrando en la configuración de las coordenadas, confirmamos que la distancia entre punto y punto se va a utilizar pulgadas.

El tercer paso nos indica qué icono nos permite generar nuestra área de trabajo. De igual manera que realizamos en Proteus - Ares, debemos generar un rectángulo para que las coordenadas obtenidas sean del tipo XY, es decir, coordenadas rectangulares.

Así mismo entrando en VIEW/CO-ORDINATES/ORIGIN/CHANGE ORIGIN, nos da la opción de cambiar de lugar el punto origen. Como previamente se mencionó, el punto de origen debe utilizar el cuarto cuadrante ya que el diseño mecánico se mueve hacia adelante y hacia la derecha. Esto se consigue utilizando X positivas y Y negativas. Es por esta razón que el punto de origen se lo debe poner en la esquina superior izquierda de nuestra área de trabajo. Estas especificaciones mencionadas se las puede observar más claramente en la figura 63.

Una vez que tenemos todo en su sitio, podemos proceder a crear el archivo que va a utilizar nuestra interface virtual. Dado que el Livewire/PCBWizard nos permite manipular los datos que se van a generar, es decir, poder cambiar cuantas unidades y decimales queremos.

Podemos crear una coordenada X y una Y que sea igual a la del Proteus/Ares ya que como vimos previamente al archivo del Proteus/Ares no nos deja escoger este tipo de opciones, y como resultado nos da un único archivo con las coordenadas ya listas. Para que los dos programas tengan los mismos puntos, debemos modificar el archivo del Livewire/PCBWizard. Como se observa en la figura 65 las flechas nos indican los puntos en los que debemos cambiar.

Primero, nos indica cómo se va a llamar el archivo y el lugar donde se lo va a guardar, segundo, la modificación de las coordenadas nos indica que debe tener dos unidades, y cuatro decimales, de esta forma vamos a tener una extensión de 6 dígitos por cada coordenada. Finalmente, la última flecha nos indica que no debe haber supresión de los ceros ya que si quitamos éstos en la interface gráfica, va a interpretar como otra coordenada y va a perforar otro punto muy diferente al que se lo ha diseñado.

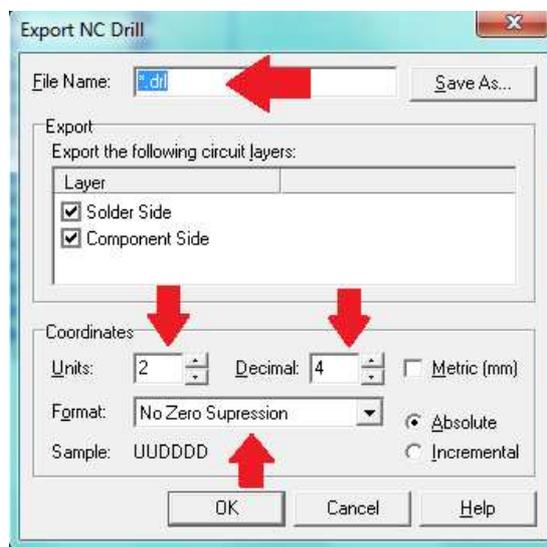


Figura 65: Generación archivo generado Livewire/PCBWizard

Del mismo modo que hicimos con el Proteus/Ares, podemos proceder a abrir el archivo con el block de notas para poder observar los puntos generados. Como se observa en la figura 66, podemos ver las partes que se indicó previamente sobre este archivo.

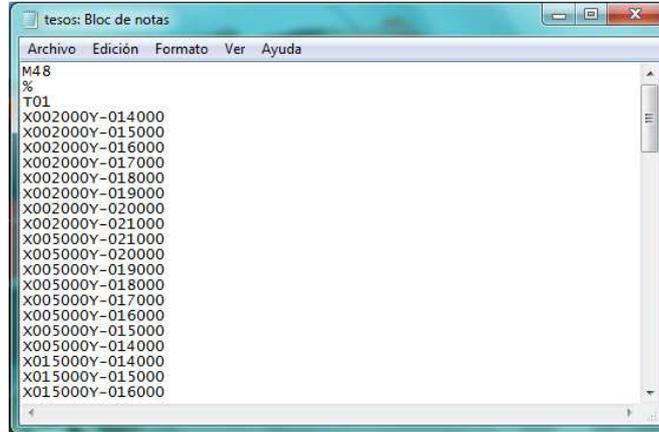


Figura 66: archivo generado por generado Livewire/PCBWizzard

Una vez generado el archivo procedemos a abrir la interfaz visual. En este programa como se mencionó previamente los componentes, podemos proceder a escoger:

- Clic en Cargar desde Livewire PCBWizzard
- Buscamos el archivo generado y lo cargamos a la interfaz.



Figura 67: Cargar el Archivo y comenzar perforación

- Verificamos con el Livewire/PCBWizard que los puntos generados con los mismos

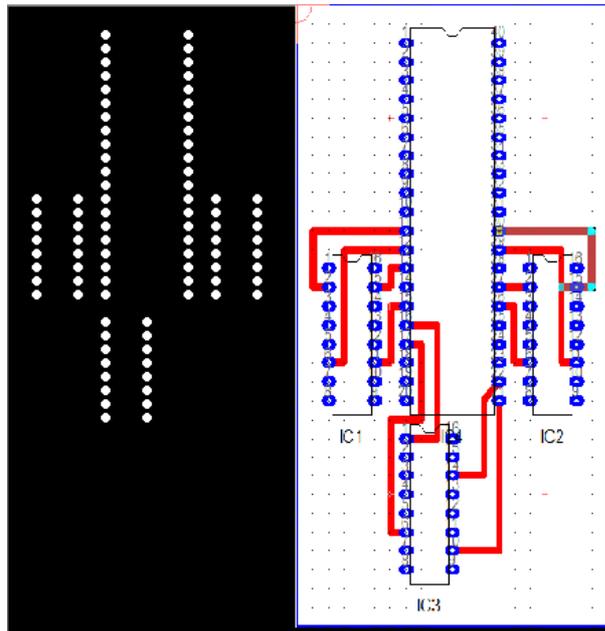


Figura 68: Comparación puntos Livewire/PCBWizard e interface visual

- Confirmamos que se encuentra en el Punto (0,0)
- Procedemos a hacer clic en Iniciar

Como previamente se indicó, cuando finalice la perforación un mensaje de finalización se va a entregar, de esta forma sabemos que la perforación está lista. De igual manera los puntos que están en perforación, se van a tornar de color rojo, para indicarnos como procede la máquina con la perforación.

III. Comparación de los dos Programas

Como se ha venido explicando, los dos programas deberían trabajar de igual forma, como se observa en la figura 64 el diseño, es el mismo del Livewire/PCBWizard al generado en Proteus/Ares, ya que de esta forma podemos didácticamente indicar al usuario que cualquiera de los dos programas pueden servir para el buen uso de la perforación automatizada.

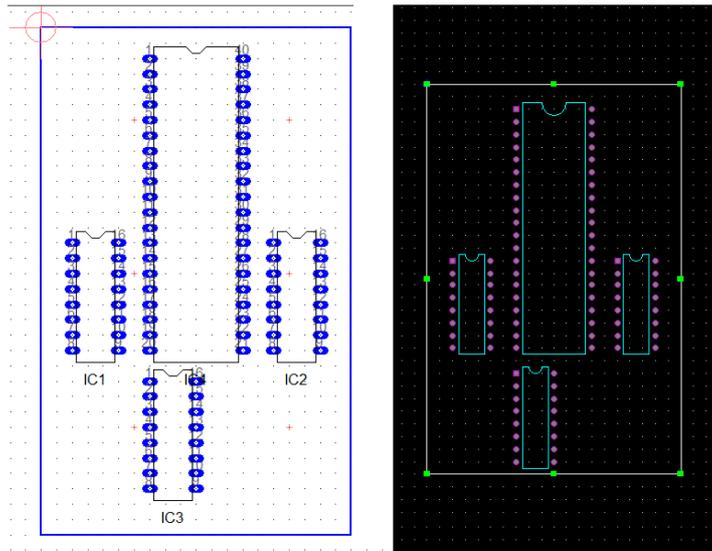


Figura 69: Igualdad Proteus/Ares con Livewire/PCBWizard

De igual forma si observamos los archivos generados por estos dos programas, nos van a indicar los mismos puntos, pero en diferentes órdenes lo cual no afecta el producto final.

Como observamos en la siguiente figura, los dos archivos están cargados y están superpuestos ya que como se indicó previamente, son los mismos puntos en diferentes órdenes, lo cual nos favorece para poder utilizar a gusto del usuario el programa que sea de mayor facilidad.

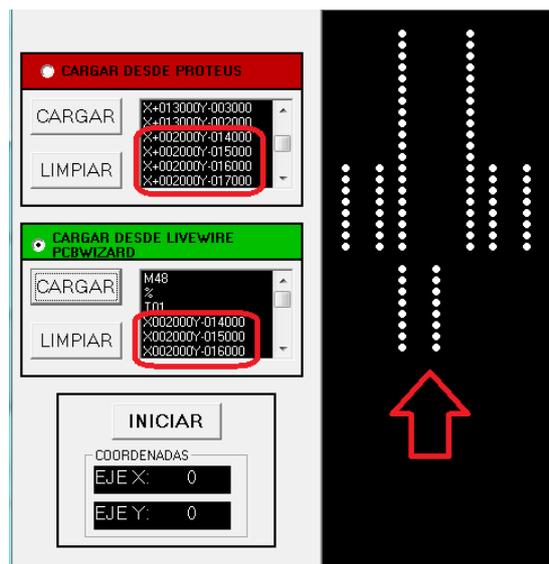


Figura 70: Igualdad de puntos de los dos programas

IV. Utilizando el Control Manual

El control manual como se mencionó previamente, es una utilidad de la interface gráfica que nos permite mover los motores como el usuario lo desee. El control manual consta de 5 botones y su objetivo principal es permitir al usuario ubicarse en algún punto requerido y perforar. Esto nos puede servir tanto para cambiar el tamaño de la broca, mejorar o crear algún orificio extra.

Como se observa en la figura 71, podemos ver el control manual en la interface gráfica. Claramente se distinguen las cuatro direcciones en las que se puede mover nuestro robot y en el centro un icono que nos permite realizar la perforación.



Figura 71: Control Manual

Dentro de los objetivos planteados de la interface gráfica, nos indicaba que debe existir un control de velocidad para el control manual. Esto significa que si lo ponemos en movimiento normal, va a tener un movimiento continuo de motores permitiéndonos llegar al punto deseado con mayor precisión. Caso contrario, si es que movemos los motores con mayor velocidad, vamos a llegar al punto deseado más rápidamente, pero con menor exactitud.

El control manual se encuentra habilitado una vez que los motores hayan llegado al inicio de carrera, de esta forma la interface de conexión, es decir el foco rojo de la esquina, se va a tornar verde, indicando que existe comunicación. De esta forma sabemos que nuestro robot está listo para comenzar a trabajar permitiéndonos utilizar el control manual o la perforación automatizada. Como se observa en la figura 72, cada vez que realizamos un clic en cada flecha, esta se torna verde, permitiéndonos confirmar visualmente que se hizo la acción deseada. Esta opción está presente en las cuatro flechas y en el icono de perforación.

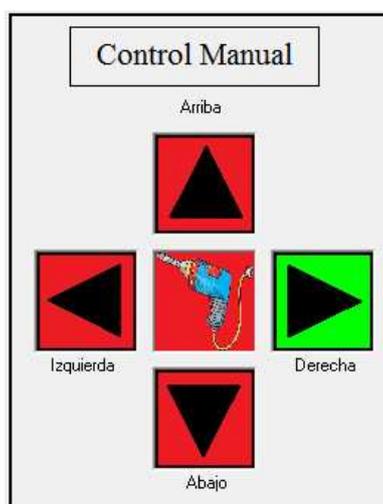


Figura 72: Confirmación de movimiento

Una vez que se haya terminado de usar el control manual, se puede apagar o se puede indicar que vaya a los puntos iniciales. Esto se debe a que como previamente vimos en la programación, una vez que se prende el robot, éste va a los puntos iniciales permitiéndonos siempre tener el mismo punto de inicio. De esta forma no hay problema de apagarlo en cualquier punto del área de trabajo, sin embargo, se ha diseñado un botón, el cual, si se lo hace clic, nos lleva a los puntos iniciales, de esta forma podemos realizar una nueva tarea de perforación, o bien simplemente apagarlo en los puntos iniciales.

Como se observa en la figura 73, observamos que el botón se encuentra debajo de las flechas de control manual. Y que cuando se hace clic una ventana de confirmación va a salir indicándonos que nos va a llevar a los puntos 0 en X y 0 en Y.



Figura 73: Botón para ir al punto de inicio

ESPECIFICACIONES Y CONCLUSIONES

I. Especificaciones

Tabla 7: Especificaciones RobotXY		
Velocidad de Desplazamiento	Eje X	1.76 [mm/seg] aprox.
	Eje Y	1.76 [mm/seg] aprox.
	Eje Z	1.76 [mm/seg] aprox.
Tiempo de Viaje Esquina Esquina	Eje X	61.8 [Seg]
	Eje Y	87.55 [Seg]
Tiempo Perforación 5 orificios	8.8 [Seg]	
Distancia de recorrido por paso	2.54 [mm] ó 0.1 [pulgadas]	
Microcontrolador	PIC18F4550	
Interface Visual	Visual Basic 6	
Comunicación	USB 2.0	
Punto Inicial	Sensores de Contactos	
Acoples	Matrimonio menor a mayor	
Cambio Rotacional a Traslación	Tornillo sin Fin con Tuercas	
Peso	6.3 [Kg]	
Dimensiones	65,8 [cm] x 25 [cm] x 31,8 [cm]	
Costo Aproximado	133,50 \$Usd	

II. Conclusiones

La fabricación de un circuito impreso, como se pudo concluir, puede ser un proceso corto o extenso según el circuito que se esté realizando. Pero gracias a la robótica y a la automatización, éste proceso puede ser más corto siempre y cuando se fomente la investigación para el buen uso del producto en función del tiempo. Esto significa que se puede generar productos que nos ayuden a reducir el tiempo de fabricación con menor costo y mayor eficiencia.

Como se pudo observar, este proyecto nos permitió mejorar el tiempo de fabricación de los orificios de una placa PCB, ya que cuando son muchos el robot automatizado realiza con continuidad y sin perder exactitud los orificios que se ha diseñado en los programas sugeridos. De esta forma podemos ayudar al usuario a que la placa realizada sea un proceso más rápido, lo que nos permite tener más tiempo para realizar diferentes actividades que pueden servir más a la sociedad.

Dentro de los aspectos más importantes que se pueden resaltar de este proyecto, es la capacidad de controlar los motores paso a paso, que mediante una comunicación de alta velocidad como es la comunicación USB 2.0 y una interface visual, figura amigable al usuario, se puede combinar para poder elaborar un robot que se mueva en los ejes XYZ. Esta fusión de los aspectos mecánicos y eléctricos, nos permitió obtener un producto final de alta fidelidad, tanto para el uso académico como investigativo.

Entre los aspectos más delicados que se han encontrado, podemos mencionar que cuando se vaya a utilizar varios motores, siempre es mejor utilizar la misma marca con las mismas características ya que en un futuro puede simplificar la resolución de detalles tales como la programación, los ajustes y la calibración. De igual manera, utilizar dispositivos que sean fáciles de adquirir, ya que en caso de que exista una falla pueda existir un repuesto fácil de encontrar, de esta forma continuar sin pausar la fabricación ni la calibración.

Futuramente a este proyecto tanto en la interface gráfica como en el diseño, se lo podría mejorar ingresando en el control manual y a las coordenadas definidas para que el usuario pueda tranquilamente ingresar a que coordenadas quiera ir y que éste se dirija. Así mismo al diseño se le puede incluir para que automáticamente vaya a cambiar de brocas y una vez

finalizada la perforación, realizar un Pick and Place de los componentes, insertarlos y soldarlos, de esta forma se completaría la automatización de la fabricación de placas PCB.

Esas recomendaciones generadas a futuro se las puede realizar tanto como continuación de proyecto final así como proyecto de las clases de automatización, robótica, scada, microcontroladores y control de máquinas eléctricas, ya que éstas son las bases que nos permiten generar el control tanto sobre los motores así como de la interface gráfica para generar un proyecto de tal magnitud.

Finalmente, se puede concluir que todos los objetivos planteados en un inicio, fueron completados con gran éxito, ya que gracias a la investigación previamente realizada, se pudo usar el microcontrolador, con la comunicación y la interfaz figura para automatizar el proceso de perforación de las placas PCB.

BIBLIOGRAFÍA

- [1] http://www.winpicprog.co.uk/pic_tutorial1.htm
- [2] http://www.microchip.com/stellent/iDCplg?IDCService=SS_GET_PAGE&nodeId=2125¶m=en542238
- [3] http://www.migsantiago.com/index.php?option=com_content&view=article&id=9&Itemid=10
- [4] <http://www.doc.ic.ac.uk/~ih/doc/stepper/>
- [5] <http://es.wikipedia.org/wiki/Riel>
- [6] <http://www.tigoe.net/pcomp/code/category/arduinowiring/51>
- [7] <http://www.neoteo.com/Portada/tabid/54/id/18223/pg/0/cp/2/Default.aspx>
- [8] <http://www.todopic.com.ar/foros/index.php?topic=20451.20>
- [9] <http://www.freewebs.com/glafebre/tp2550.htm>
- [10] <http://micros.mforos.com/1149907/7869278-controlador-usb-de-servomotores/>
- [11] <http://www.todopic.com.ar/foros/index.php?topic=20912.0>
- [12] <http://www.unpocodelectronica.net.au.net/mis-primeros-pasos-con-el-18f4550-parte5#picusb>
- [13] <http://glafebre.webs.com/>
- [14] <http://www.freewebs.com/glafebre/eleazar.htm>
- [15] <http://www.canalvisualbasic.net/foro/visual-basic-6-0/ayuda-sobre-command-butt-n-3312/>
- [16] <http://todopic.mforos.com/58732/4550883-eleazar-bot-controlado-por-pc-usb/>
- [17] <http://www.edaboard.com/ftopic313796.html>
- [18] <http://www.todorobot.com.ar/informacion/tutorial%20stepper/stepper-tutorial.htm>
- [19] <http://www.forosdeelectronica.com/f19/introduccion-motores-paso-paso-289/>
- [20] <http://www.arossini.com.ar/files/cnc/Paso%20a%20Paso.doc>
- [21] <http://www.computing.net/answers/programming/close-vb-window-button-code/13226.html>
- [22] http://www.microchip.com/stellent/iDCplg?IDCService=SS_GET_PAGE&nodeId=2042¶m=en020453
- [23] <http://www.recursosvisualbasic.com.ar/htm/tutoriales/control-commondialog.htm>
- [24] <http://www.profsr.com/vb/vbless04.htm>

- [25] <http://msdn.microsoft.com/es-es/library/kd7e4yte%28VS.80%29.aspx>
- [26] <http://www.recursovisualbasic.com.ar/htm/tutoriales/metodos-gráficos.htm>
- [27] <http://msdn.microsoft.com/es-es/library/9dtfzwyx%28VS.80%29.aspx>
- [28] <http://www.unpocodelectronica.net.au.net/generador-de-inf-para-los-drivers-usb-de-microchip>
- [29] http://users.dod.sch.gr/nichrist/pic18f4550_usb.htm
- [30] <http://espanol.answers.yahoo.com/question/index?qid=20080425091925AA8v2IZ>
- [31] <http://ladelec.com/practicas/circuitos-analogos/477-fuente-de-poder-triple-24-vDC-9-vDC-y-5-vDC-.html>
- [32] <http://ladelec.com/teoria/tutoriales-electronica/92-como-aumentar-amperaje-en-reguladores-78xx-y>
- [33] <http://www.directindustry.es/prod/johnson-motor-saia-motor/motor-electrico-brushless-DC-665-282248.html>
- [34] Dr. Ing. Laurent Sass – Clases de Robótica 2009
- [35] Datasheet - L293 (Puente H)
- [36] Datasheet – PIC18F87XA
- [37] Datasheet – TIP2293
- [38] Datasheet – LM78XX
- [39] Manual ISSIS PROTEUS
- [40] Datasheet Mitsumi M43SP-7
- [41] Datasheet NMB PM42M-048
- [42] Reyes, Carlos A., Micro controladores PIC, 2da Edición, Quito – Ecuador, 2006
- [43] Trojman Lionel, Guía de Redacción, Sensores e Instrumentación Virtual, 2010
- [44] <http://www.r-luis.xbot.es/cnc/taller01.html>

ANEXOS

I. Programación Pic

Archivo PDF Cd adjunto

II. Programación Visual Basic

Archivo PDF Cd adjunto

III. Datasheet Pic 18F4550

Archivo PDF Cd adjunto

IV. Datasheet LM293

Archivo PDF Cd adjunto

V. Datasheet Motores

Archivo PDF Cd adjunto

VI. Instalador Easy HID

Archivo EXE Cd adjunto

VII. Ejecutable Aplicación

Archivo EXE Cd adjunto

VIII. Simulación Proteus

Archivo Proteus Cd adjunto

IX. Presupuesto

Archivo Excel Cd adjunto