# Plotter Cnc en la plataforma Vex Robotics utilizando RobotC

(8 Junio 2017)

Brayan Camilo Lozano Polanía.
Pontificia Universidad Javeriana

***Resumen*— Se muestra la construcción de una maquina cnc, que hace dibujos en 2 dimisiones, explicando su funcionamiento desde la mecánica hasta la programación implementada en un microcontrolador con el fin de desarrollar algoritmos en el lenguaje de programación RobotC, se dan a conocer los planos y esquemáticos necesarios para poder entender más a profundidad la máquina, se muestra cómo interpretar archivos tipo Gcode con ayuda del software matemático de alto nivel llamado Matlab donde se expone el código fuente y se describe paso a paso cada línea de código y como comunicarse con un microcontrolador con este software desde un computador.**

***Palabras clave*— Cnc, Gcode, Matlab, RobotC.**

## I. Introducción

ESTE proyecto proporciona herramientas para el desarrollo de una maquina cnc (control numérico computarizado) cualquiera usando el microcontrolador VEX ARM Cortex, dando como ejemplo un plotter para su aplicación, este plotter plasma en un papel un dibujo diseñado con el lenguaje de programación de control numérico más utilizado llamado "Gcode", este lenguaje está diseñado en general para definir trayectorias ya sean rectas o curvas, donde poco a poco uniendo trayectorias milimétricas se puede llegar a definir un dibujo de calidad en coordenadas cartesianas, para poder interpretar este lenguaje se usa un software llamado "Matlab" donde su nombre significa "Laboratorio de Matrices", con este software es posible leer e interpretar la lista de trayectorias de un dibujo diseñado con el lenguaje Gcode desde un computador y poder manipular y almacenar estas coordenadas para comunicárselas a cualquier microcontrolador, en este caso el microcontrolador "VEX ARM Cortex" de la empresa de robótica educativa Vex Robotics. En la construcción de este proyecto el concepto de velocidad angular tiene una importancia tanto para la mecánica como para el control lógico, ya que este concepto es fundamental para definir la dirección de un trazo en un dibujo tipo Gcode, es fundamental porque lo que será controlado principalmente son motores paso a paso o también llamados motores de paso variando su velocidad angular. Ya que el microcontrolador "VEX ARM Cortex" está diseñado específicamente para ser usado con componentes de la plataforma "Vex Robotics", Lo que se quiere principalmente en este proyecto es demostrar el uso de este microcontrolador con diferentes componentes ajenos a la plataforma Vex, para así poder ampliar el campo de aplicación este y no tener limitantes a la hora de desarrollar soluciones donde los recursos de esta plataforma puedan escasear en una escuela o una universidad, ya que en estos lugares es donde más se pueden encontrar los componentes de esta empresa, por ser de naturaleza educativa.

## II. Desarrollo

### A. Mecánica y construcción

El funcionamiento mecánico de un plotter cnc se basa en el manejo de un plano cartesiano desde un punto de origen (0,0,0) que quiere decir : (cero en el eje X , cero en el eje Y, lápiz levantado), donde el lápiz representa un eje Z pero binario, siendo 1 cuando el lápiz hace un trazo y 0 cuando el lápiz está levantado, es decir no hace ningún trazo en el papel, vale afirmar que no se suelen usar coordenadas negativas en un dibujo tipo Gcode, ya que por lo general se pone el origen de coordenadas en la parte inferior izquierda del papel, es decir, en una esquina, lo que nos deja las coordenadas negativas fuera del papel y su superficie. Al tener 2 ejes en coordenadas desde cero a un límite, donde las unidades de estos ejes serán en milímetros, y un eje Z binario (trazar o no trazar), esto es algo que se llama un plano con 2 ejes y medio, o plano con 2.5 ejes. El área de dibujo o las dimensiones en que se puede hacer un trazo serán 400 × 400 milímetros, para poder mover un lápiz o un marcador sobre esta área, el mecanismo que se utiliza en este proyecto es un sistema de poleas y correas dentadas, las poleas necesarias son tres, una para cada eje de tres motores de paso, en las siguientes imágenes se muestra la apariencia de las poleas y como se acopla una correa a su polea respectiva.


*Imagen 1. Poleas GT2 de 20 dientes.*

*Imagen 2. Correa GT2.*

Los dientes de esta correa permiten un agarre con la polea que no genera deslizamientos, lo que ayuda a un control más preciso del movimiento.

*Imagen 3. Correa con polea GT2.*

Estas correas jalan una base sobre dos carriles, donde para el eje X hay un carril y para el eje Y otro carril independiente, y el eje Z que representa el lápiz está sobre el eje X mientas este eje se apoya en el eje Y. para este movimiento se implementan motores paso a paso, los tres mencionados anteriormente en total, uno para el eje X y dos para el eje Y, por definición un motor paso a paso es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de girar una cantidad de grados, por lo general un motor de paso da una vuelta en 200 pasos debido a engranajes internos que reducen el giro del rotor, los motores aquí utilizados son los Nema 17 que también dan una vuelta o revolución en 200 pasos.

*Imagen 4. Motor Nema 17.*

Este tipo de motor es empleado cuando se hace imprescindible controlar exactamente las revoluciones o los grados de un giro. Son utilizados, principalmente, en máquinas pequeñas de oficina, como pueden ser impresoras, fotocopiadoras, faxes, etc. También se pueden encontrar en instrumentos médicos y científicos, por estos motivos para tener mayor precisión en los dibujos deseados se utilizan motores de paso en este caso bipolares.
Un motor paso a paso bipolar en su eje de rotación tiene internamente un imán, y a su alrededor dos bobinas distribuidas de la siguiente forma:

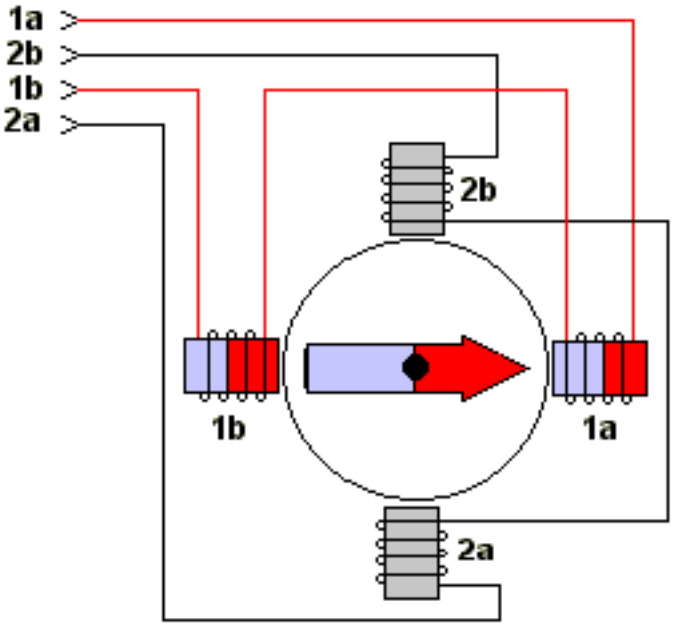


*Imagen 5. Modelo conceptual de un motor paso a paso bipolar.*

En la *Imagen 5* la flecha representa el rotor del motor que es un imán donde cada color representa un polo, el rojo el norte y el azul el sur, al ser energizada la bobina 1 el rotor es atraído por los polos generados en la bobina según el sentido de la corriente, y este queda totalmente alineado con la bobina. Para producir un giro en el eje del motor se va energizando cada bobina alternamente en una secuencia en donde los polos del rotor sean atraídos ordenadamente para producir un giro de la siguiente manera:

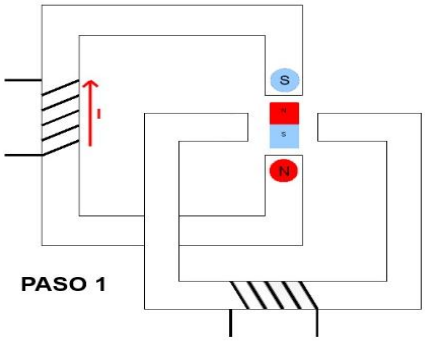


*Imagen 6. Paso 1.*

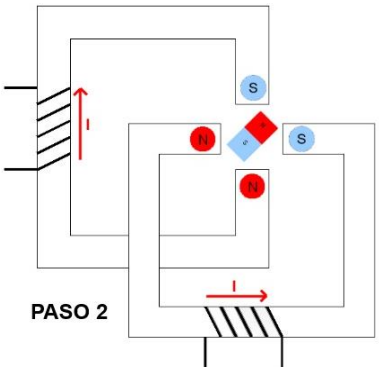


*Imagen 7. Paso 2*

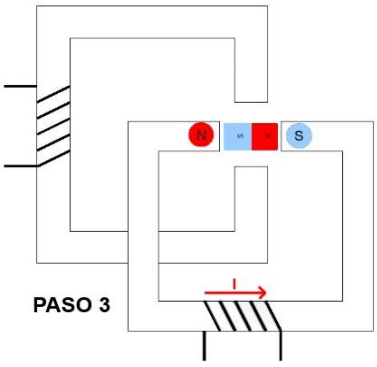


*Imagen 8. Paso 3.*

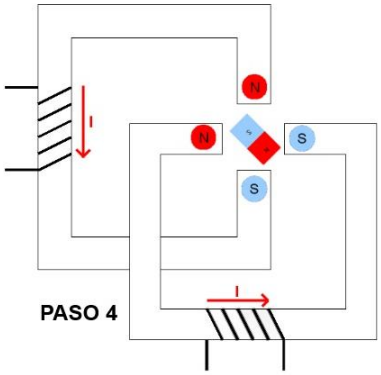


*Imagen 9. Paso 4.*

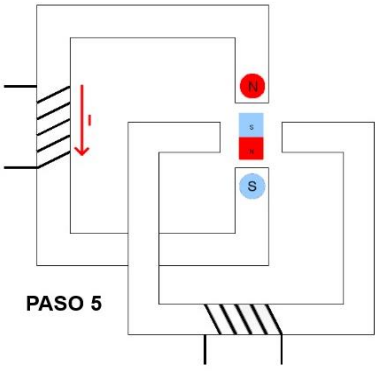


*Imagen 10. Paso 5.*

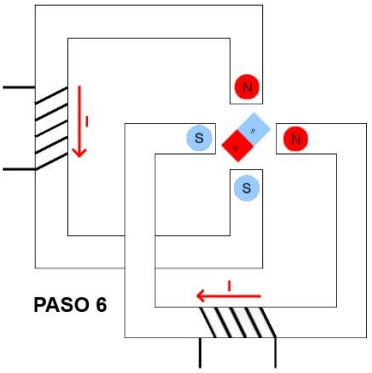


*Imagen 11. Paso 6.*

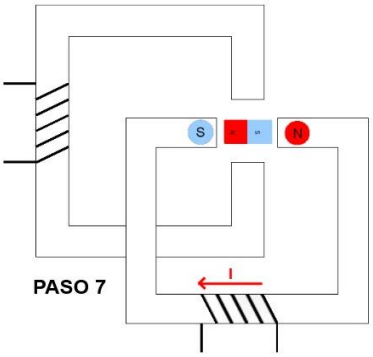


*Imagen 12. Paso 7.*

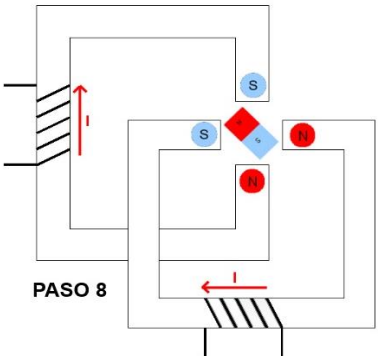


*Imagen 13. Paso 8.*

En este caso cada paso realizado es de 45 grados ya que se energizan las dos bobinas a la vez, esto se define como medio paso, un paso entero es de 90 grados cuando se energiza una a una cada bobina. Con estas secuencias y manipulando el estado de cada bobina podemos tener un control lógico del motor.

Para el montaje del eje X se tomó como ejemplo una impresora 3D diseñada por la comunidad Reprap, el eje X consta de un solo motor que jala un carro que se desliza por dos varillas lisas de 8 milímetros de diámetro gracias a una correa dentada acoplada en el motor con una polea *GT2*, el montaje del eje X es el siguiente:

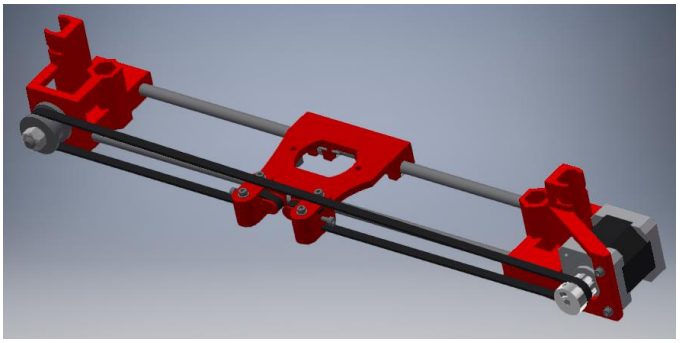

*Imagen 14. Cad eje X.*

Las piezas principales que forman la estructura se describen a continuación:

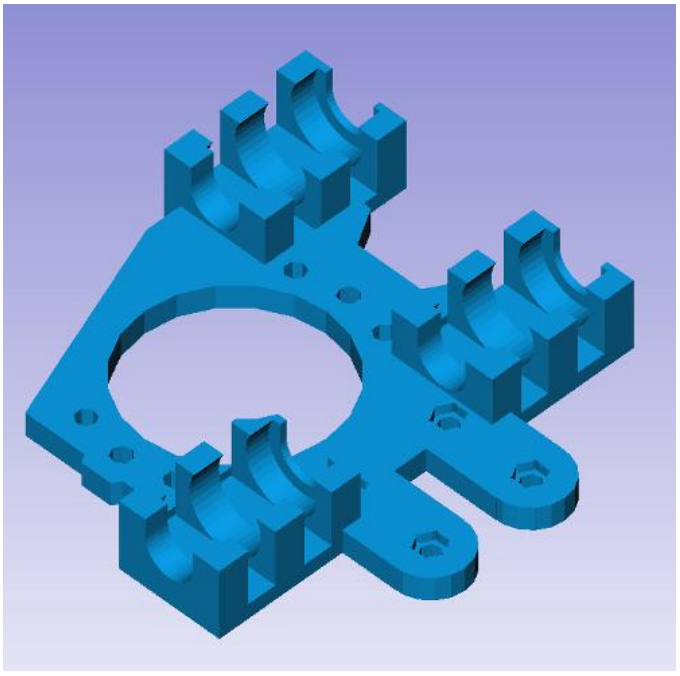

*Imagen 15. Carro del eje X (renderizado).*

El carro de la *imagen 15* es el que llevará sujeto el lápiz que hará un trazo en el papel, este carril lleva tres rodamientos lineales que le permiten deslizarse por las varillas lisas, estos rodamientos poseen balines internos que disminuyen la fricción y permiten que una varilla pase por su interior.

*Imagen 16. Rodamiento lineal.*

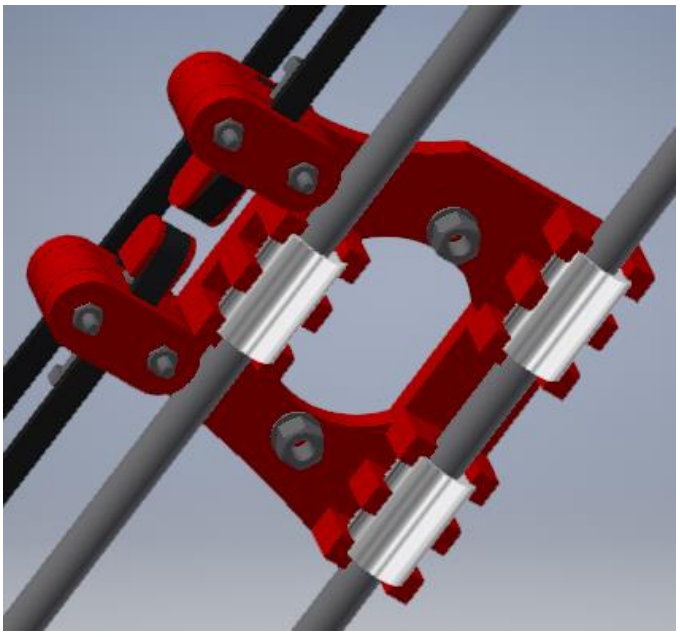

*Imagen 17. Rodamientos en el carro del eje X.*

En la *Imagen 17* se puede apreciar la distribución de los rodamientos fijados en el carro del eje X, también se muestra como las varillas lisas pasan por su interior y evidencia el moviendo que se puede realizar si la correa negra jala del carro.

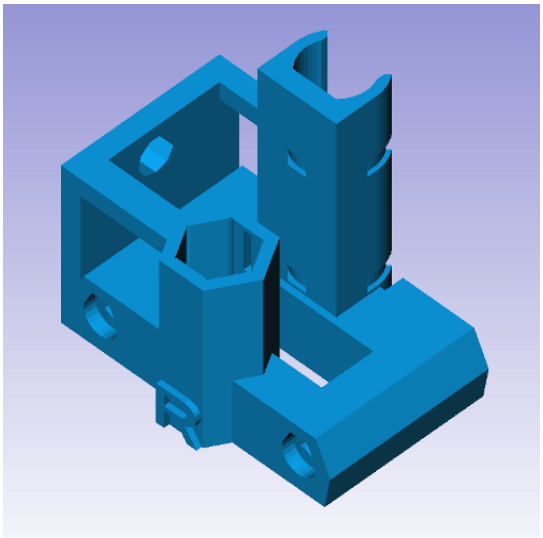

*Imagen 18. Base costado derecho (renderizada).*

La *Imagen 18* muestra una base con dos orificios en su parte más baja donde se sostienen los extremos de las dos varillas lisas y también se muestra en uno de sus costados un orificio que sirve para sostener un eje que tiene un rodamiento rígido de bolas.

*Imagen 19. Rodamiento rígido de bolas.*

En este caso el rodamiento rígido de bolas tiene un carrete que no permite que la correa dentada negra se salda del rodamiento.

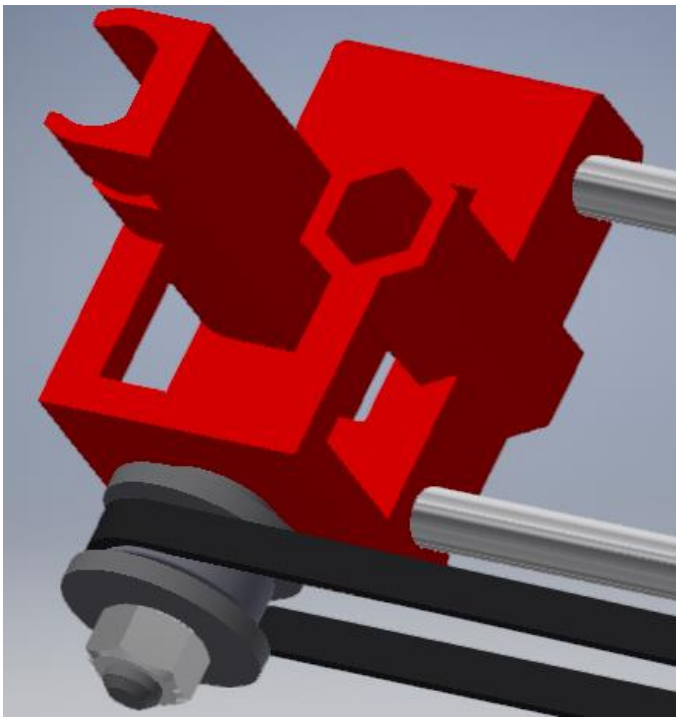

*Imagen 20. Costado derecho del eje X montado.*

En el costado izquierdo del eje X se utiliza una pieza con los mismos orificios para sostener los otros dos extremos de las varillas lisas, pero ahora la parte superior de esta pieza se concentra en sujetar firmemente un motor de paso Nema 17, Su estructura tiene tres orificios para atornillar el motor de tres de sus esquinas para quedar fuertemente sujeto y no haga movimientos de su cuerpo sino solamente de su eje o rotor.

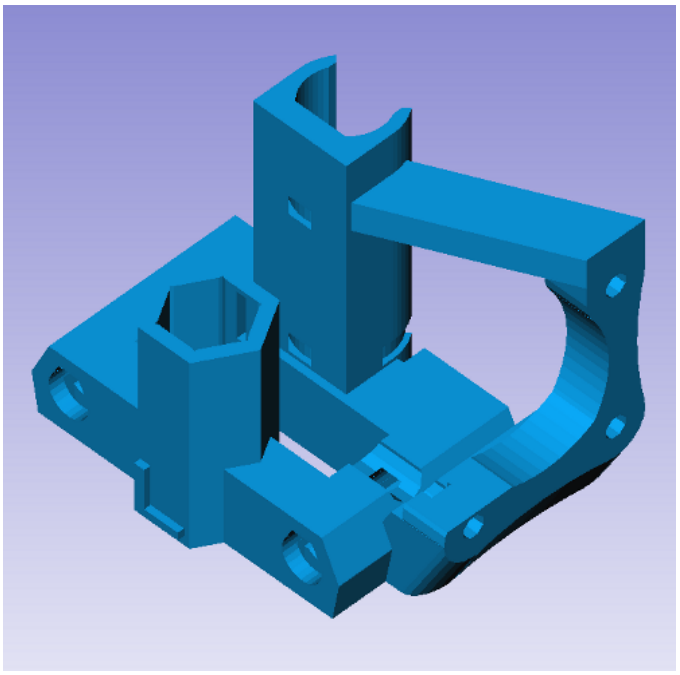

*Imagen 21. Base costado izquierdo (renderizada).*

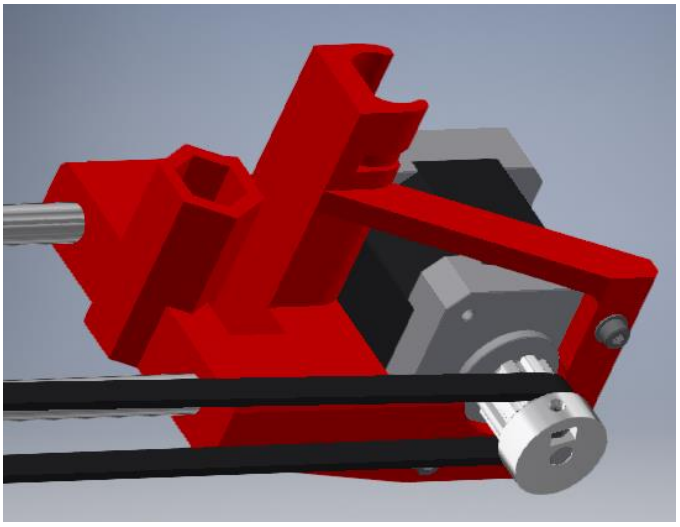

*Imagen 22. Costado Izquierdo del eje X*

En *Imagen 22* ya está la polea *GT2* aprisionada en el eje del motor y como en la *Imagen 3* también es posible ver el acople de la correa en la pelea de una manera tensa, lo que permite que todo el sistema se mueva acorde a como este girando el rotor del motor Nema 17.

El eje Y se construyó en base al eje X ya presentado, la única diferencia es que el eje Y tiene las varillas lisas con más distancia de separación entre ellas y pasa compensar un movimiento uniforme debido a la apertura de las varillas se utiliza un motor más que en el eje X, cada con su respectiva polea y correa acoplada.

*Imagen 23. Montaje eje X con eje Y.*

*Imagen 24. Vista superior de montaje eje X y eje Y.*

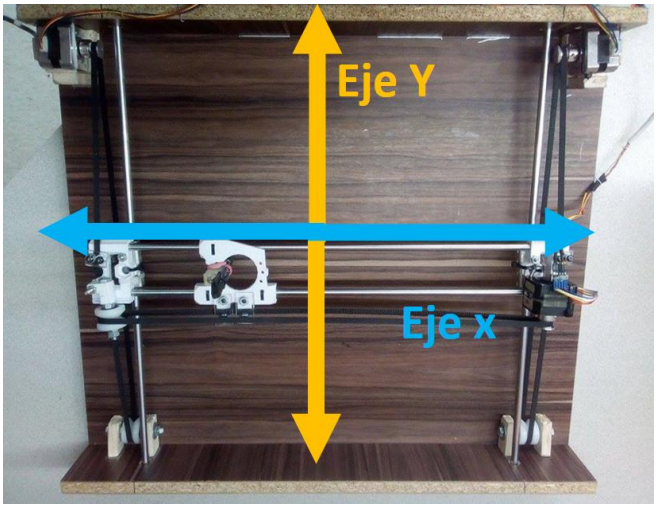


*Imagen 25. Representación de plano cartesiano.*

En las imágenes *23,24 y 25* se muestra el montaje de los dos ejes X y Y, se puede apreciar que tienen el mismo funcionamiento cada eje, lo importante aquí es que el eje X está sobre el eje Y teniendo la posibilidad de deslizar sobre él, y finalmente en la imagen *25* una representación de los ejes cartesianos formados en el montaje. La base en que se apoyan los ejes está construida con un aglomerado de 15 mm de espesor, esta base tiene un área de 60 x 60 centímetros y dos paredes paralelas con un alto de 8 centímetros.

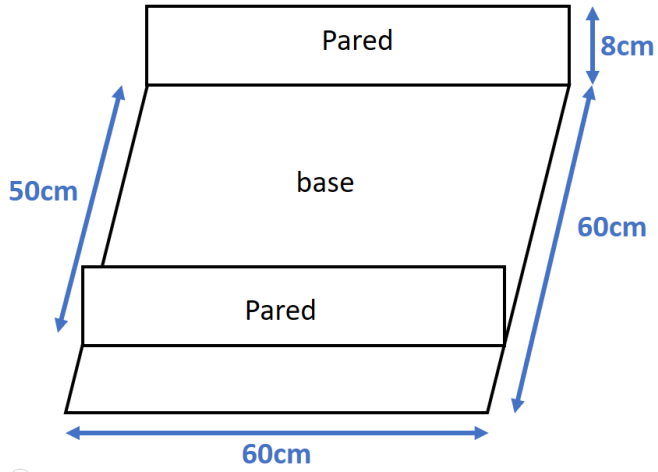


*Imagen 26. Medidas base de aglomerado.*

La distancia entre cada pared es de 50 centímetros ya que las varillas tienen de largo esta distancia, por lo que queda un área de 10 x 60 centímetros la cual se aprovecha para poner toda la parte electrónica de este plotter.

*B. Electrónica*

Para controlar los motores se utilizan controladores pololu A4988.

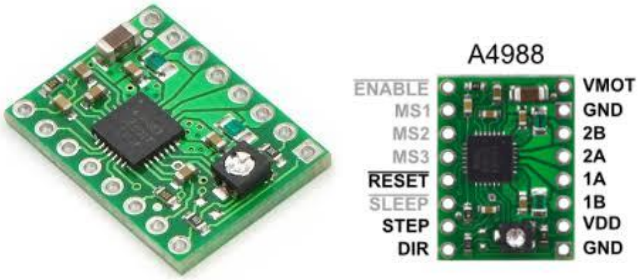


*Imagen 27. Controlador A4988.*

Estos controladores poseen internamente 2 puentes H que permiten cambiar el sentido de la corriente en cada bobina para así poder generar pasos como en la secuencia de las imágenes 6 a la 13. Un puente H es un circuito utilizado generalmente para cambiar el sentido de giro de un motor DC, que funciona abriendo y cerrando el circuito de tal forma que se invierta el sentido de la corriente que circula por el motor.

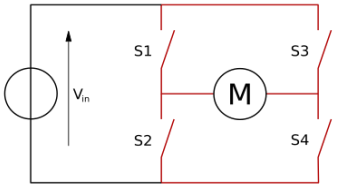


*Imagen 28. Representación conceptual de un puente H.*

En la *Imagen 28* la letra M representa el motor y los índices S1, S2, S3 y S4 interruptores, aquí el motor ha de estar quieto porque no está energizado de ninguna manera ya que el circuito está abierto en cuatro puntos, pero si se activan solo los interruptores S1 y S4 el motor girará, y de igual manera si se activan S2 y S3 girará en el sentido contrario.

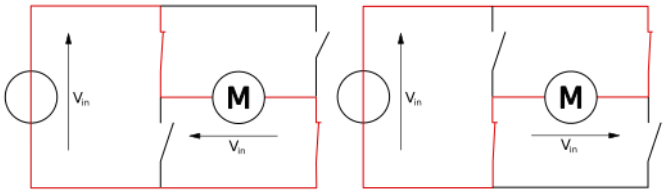


*Imagen 29. Sentido de un motor DC en un puente H.*

Cada motor paso a paso necesita un controlador ya que este tiene 2 puentes H que manipulan el estado de las 2 bobinas de cada motor paso a paso, una bobina del motor de paso se conecta a los pines 1A y 1B dando un extremo de la bobina a cada pin, y la segunda bobina a los pines 2A y 2B, en el pin VMOT se conecta la fuente de alimentación del motor y en el pin VDD la fuente para el controlador. Los pines para el control lógico del motor son STEP, DIR y ENABLE, el pin STEP es el que controla el movimiento de cada paso en el motor, el pin DIR da una dirección o sentido al giro del motor y finalmente ENABLE habilita o deshabilita el motor.

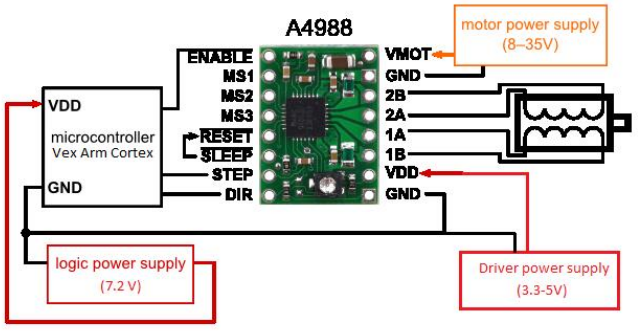


*Imagen 30. Conexión de cada motor al controlador A4988.*

Las conexiones de los pines STEP, DIR y ENABLE de cada motor van conectados a pines I/O (Input/Output) o pines lógicos del microcontrolador Vex Arm Cortex, siendo todos estos pines de salidas del microcontrolador donde alto representa 5V y bajo 0V. En la *Imagen 31* se ven las conexiones de cada fuente de alimentación para cada dispositivo donde el microcontrolador se alimenta de 7.2 voltios teniendo tierra común con cada controlador, los controladores en este caso con 5V y por último la alimentación de cada motor donde dos serán de 12V y un tercero de 9.4V, pero en el esquemático de ejemplo se toman todos los motores como iguales alimentados en un rango de 8 a 35 Voltios.

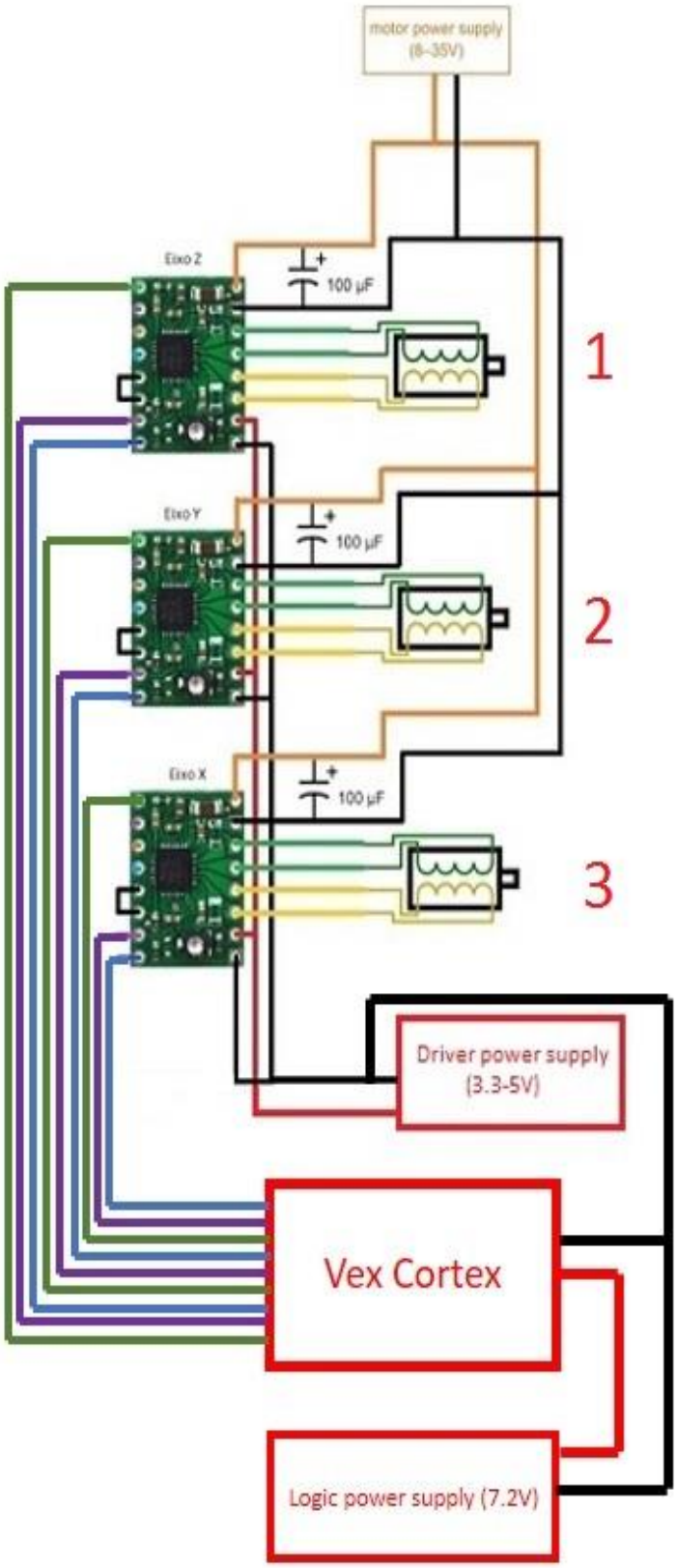


*Imagen 31. Conexión de todos los motores con su respectivo driver A4988 y fuentes de alimentación.*

El microcontrolador Vex Cortex posee pines digitales, pines analógicos, pines para comunicación serial UART e I2C y pines diseñados especialmente para el control de motores 393 de Vex, en esta aplicación para el control del plotter solo se utilizan los pines I/O, donde cada controlador A4988 se conecta como lo muestra la *imagen 32* representando los índices 1 y 2 los pines de los controladores asignados al eje Y, debido a que este eje lleva dos motores como lo muestra la imagen *25*.

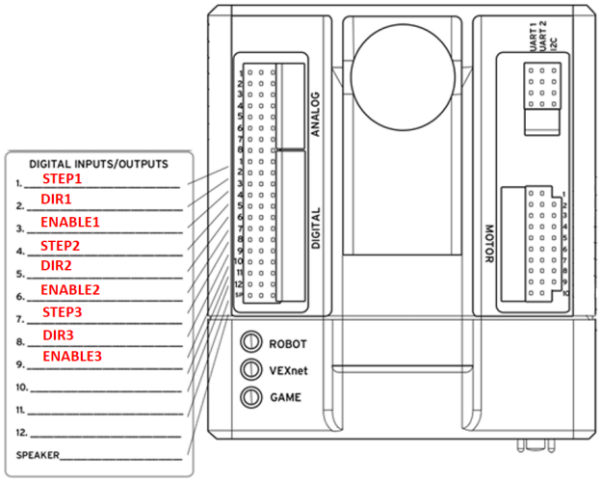


*Imagen 32. Conexión de todos los pines de cada controlador al microcontrolador Vex Arm Cortex*

*C. Control*

Para el control de los motores como se ha venido explicando las razones para su uso será el microcontrolador Vex Arm Cortex.


*Imagen 33. Microcontrolador Vex Arm Cortex desarrollado por la empresa de robótica educativa Vex Robotics.*

Aunque este microcontrolador se puede programar de distintas maneras el software de programación más usado para él es el software llamado “RobotC for Vex Robotics”, este software es comúnmente utilizado para programar robots de competencia para torneos nacionales e internacionales realizados por Vex Robotics donde su lenguaje de programación se basa en C, pero con muchas librerías incluidas que permiten el manejo de sensores, motores, comunicación serial, etc.

Este software es fácil de obtener en el sitio web oficial de Vex Robotics con licencia actualmente gratis.


*Imagen 34. Logo de software RobotC for Vex Robotics.*

Los dibujos a interpretar del lenguaje Gcode se basan en una lista de coordenadas cartesianas, definiendo punto a punto las trayectorias de cada trazo del dibujo, donde este trazo se genera uniendo estos puntos, por ejemplo, si se tiene una coordenada inicial (0,0,0) y una coordenada final (10,10,1) se generará una línea recta ya que la tercera componente es 1 quiere decir que el lápiz estará abajo listo para producir un trazo.

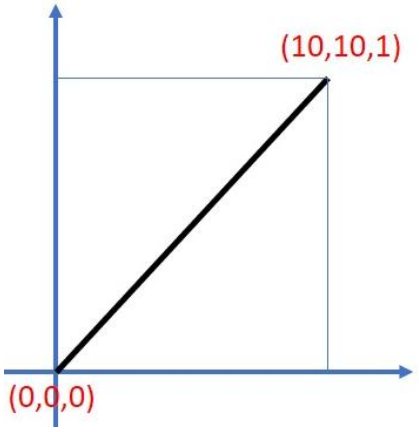


*Imagen 35. Trazo de ejemplo.*

Y si después tenemos otras dos coordenadas (20,10,0) seguida de (30,0,1) quedan dos líneas con un espacio entre ellas de 10.

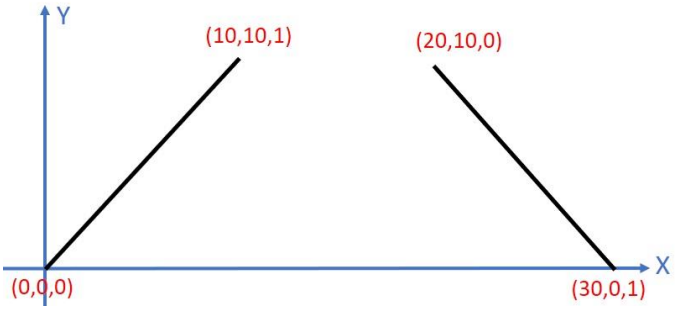


*Imagen 36. Trazo de ejemplo con espacio.*

El espacio que se aprecia en la *Imagen 36* se debe a que la coordenada (20,10,0) la tercera componente está en cero, lo que significa que el lapiza se levanta y ningún trazo, pero la siguiente (30,0,1) si tiene 1 en el eje Z que es el lápiz abajo lo que genera un trazo.

Conociendo la mecánica de los dibujos tipo Gcode, para la programación se necesita variar la velocidad de los tres motores de paso controlándolo con pausas entre cada paso donde si la pausa es muy grande el motor girara con una velocidad angular muy baja.
La Máquina de estados principal del código implementado en RobotC se muestra en la *Imagen 37.*

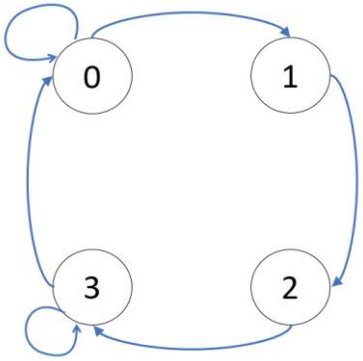


*Imagen 37. Máquina de estados finitos.*

Estados:
- Estado 0: Leer coordenadas.
- Estado 1: Definir Pendiente entre dos Puntos.
- Estado 2: Definir dirección de los motores.
- Estado 3: general línea o trazo.

Estado 0: este estado se encarga de leer de a dos coordenadas de la lista de un dibujo Gcode, donde si no termina de leer todas las componentes X, Y y Z el programa se queda en un ciclo hasta que lo haga adecuadamente.

Estado 1: Se calcula la pendiente entre las dos coordenadas leídas por el estado 0 usando la definición:

$$m = (y\text{-}yo) / (x\text{-}xo)$$

que halla la pendiente entre dos puntos donde "yo" y "xo" son las coordenadas iniciales, "y" y "x" coordenadas finales.

Estado 2: Aquí se asigna una dirección a cada motor, donde si ¨y" es mayor a "yo" los motores del eje Y tendrán sentido positivo de lo contrario será negativo, es decir en el sentido contrario e igualmente si "x" es mayor a "xo" el motor del eje X tendrá sentido positivo y si no es así será negativo.

Estado 3: En este estado se ponen en marcha los motores, dependiendo la pendiente encontrada en el estado 1, se calcula la magnitud del trazo y se definen las velocidades de cada motor que girarán durante un tiempo que definido que tan grande es la magnitud del trazo.

En RobotC la función de la que se aprovecha es la función "SensorValue[ ]" donde recibe como parámetro el nombre del pin digital y el valor que se quiere en ese pin declarado como salida. Para dar un paso en un motor paso a paso con el controlador A4988 previamente acoplado al motor y al microcontrolador se en envía un pulso de alto es decir 1, luego una pausa, después un pulso bajo es decir 0 y finalmente otra pausa. Ya en el lenguaje de RobotC seria de la siguiente manera donde la función "waitMsec( )" recibe como parámetro la cantidad de milisegundos con la que se quiere hacer una pausa:

```
SensorValue[step] = 1; wait1Msec(1);
SensorValue[step] = 0; wait1Msec(1);
```

Así que por ejemplo con un ciclo que cuente hasta 10 haría diez pasos el motor a una velocidad de 2 milisegundos por paso.

Como eje Y tiene dos motores se debe realizar un paso en un motor y luego en el otro, es decir tienen que avanzar a la vez. Teniendo las velocidades de cada eje definidas para generar la pendiente del trazo para mover los ejes a la vez ahora es necesaria otra máquina de estados que se muestra en la imagen *38.*

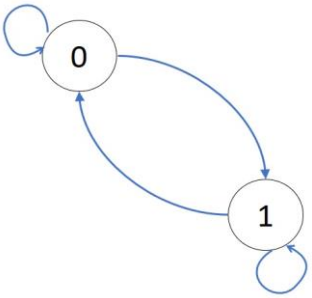


*Imagen 38. Máquina de estados para motores.*

Estados
- Estado 0: mover eje X.
- Estado 1: mover eje Y.

Estos dos estados controlan cuando mover cada eje o más precisamente cuando dar un paso en cada eje lo que significa que define la cantidad de pasos por milisegundos, en conclusión, velocidad.

Estado 0: espera una cantidad de milisegundos asignados para una pausa entre cada pulso que acciona un paso en el motor del eje X.
Estado 1: espera una cantidad de milisegundos asignados para una pausa entre cada pulso que acciona un paso en cada motor del eje Y.
Siendo RobotC un software basado en C para hacer operaciones matemáticas se digita de igual manera que en el lenguaje C, por ejemplo, para hallar la pendiente se utiliza la siguiente sintaxis:

```
float m=0; m = (y - yo) / (x - xo);
```

donde se declara "m" como flotante y seguidamente se opera con los signos correspondientes a resta y división de C asignando o igualando el resultado a la variable m.

Para definir en qué momento accionar un paso en cada eje es necesario el uso de los "Timers" del microcontrolador, donde el microcontrolador Vex Arm Cortex posee cuatro, pero en este caso solo se utiliza el T1 y el T2. Para el uso de estos se debe antes limpiar los Timers con la función "ClearTimer( );" que recibe como parámetro el Timer escogido y lo reinicia es decir deja su contador en cero donde este cuenta con unidades de milisegundos:

```
ClearTimer(T1); ClearTimer(T2);
```

El timer T1 está asignado para el eje X y el timer T2 para el eje Y, por ello para preguntar si es el momento de dar un paso en un eje se revisa en que conteo se encuentra el timer con la función "time1[ ]" donde el parámetro es un timer escogido y esta retorna el número de milisegundos que han transcurrido desde la última vez que se reinició:

```
if ( time1[T1] >= tiempo_por_paso){ }
```

En la línea de código anterior se pregunta con una sintaxis de C fundamental, pero con la función "time1[]" de RobotC en la condición y un variable "tiempo_por_paso" que simplemente define cada cuanto dar un paso donde esta variable debe estar en milisegundos.

Con estas funciones descritas y la sintaxis de programación fundamental de C es posible programar todos los requerimientos de un plotter cnc basándose en una máquina de estados general como la que muestra la imagen *37*, requerimientos como velocidad por paso de cada eje, cálculos de pendientes, cálculo de tiempos y pausas, dirección de cada motor, etc. Sin importar si se usa una lógica distinta este microcontrolador cuenta con las herramientas necesarias para desarrollar proyectos tales como un plotter y yendo más allá incluso se puede llegar a controlar una impresora 3D con el Vex Arm Cortex basándose en máquinas de estados y lógica de programación en C aprovechando todas las librerías del software para competición "RobotC for Vex Robotics".

*D. Comunicación entre Microcontrolador y un computador.*

El propósito de comunicar un computador con el microcontrolador Vex Arm Cortex en esta aplicación es poder enviarle todas las coordenadas de un dibujo en lenguaje Gcode una a una ya que el microcontrolador Vex Arm Cortex al igual que muchos mas no puede almacenar demasiadas coordenadas en su memoria interna, un dibujo Gcode no con muchos detalles puede llegar a tener como mínimo mil coordenadas o si te quiere decir puntos que definen su forma con trayectorias rectas, siendo un dibujo tan pesado de coordenadas la memoria de un microcontrolador común puede colapsar, por ello se utiliza un computador que puede almacenar una gran cantidad de coordenadas debido a que posee una memoria mucho más grande o con más espacio a comparación de la de un microcontrolador común.
La solución que se da a poder interpretar archivos Gcode y poder enviar por medio de comunicación serial cada coordenada una a una de un dibujo hacia el microcontrolador Vex es por medio del software matemático de alto nivel llamado Matlab, este software tiene su propio lenguaje de programación utilizando funciones de complejidad muy avanzada para aplicaciones en diferentes campos de la ciencia como estadística, ingeniería, física, etc. Con este es posible leer e interpretar cualquier archivo tipo Gcode, es decir, las coordenadas de cualquier dibujo, desde el computador se le puede programar como específicamente se quiere leer cada coordenada con lógica de vectores y matrices generando ciclos y recorriendo posiciones, en esto nos basaremos para la lectura e interpretación de cada dibujo.
Como los archivos tipo Gcode son prácticamente un archivo de texto que contiene la lista de coordenadas del dibujo, este archivo se puede leer y convertir en un vector de caracteres en el software Matlab de la siguiente manera utilizando la sintaxis de su propio lenguaje que es llamado lenguaje M:

```
fid = fopen('Dibujo_ejemplo.gcode','r');
Data = fread(fid);
charData = char(Data);
fclose(fid);
```

En la primer línea la función "fopen" recibe como parámetro el nombre del archivo que contiene el dibujo y lo que hace es abrir este archivo y guardarlo en "fid", la segunda con la función "fread" leer los datos del archivo en formato binario y los guarda en "Data" luego en la línea tres convierte esos datos en una cadena de caracteres, lo que significa que todos los caracteres del archivo se han guardado ahora en un vector para así poder recorrerlo y poder interpretar cada carácter del archivo y finalmente en línea cuatro la función "fclose" cierra el archivo para y no lo manipula más.
Teniendo ya los caracteres del archivo ordenadamente guardados en n vector es posible por medio de ciclos recorrer este vector e ir sacando lo que nos interesa que son todas las coordenadas del dibujo, estas coordenadas se encuentran cuando se localicen los dos caracteres "G1" o por separado "G" y "1" juntos ya que después de estos es donde se encuentra una coordenada con sus componentes en la misma línea de texto cuando lo visualizamos el archivo, por ejemplo, si se abre el archivo con un visualizador de texto desde el computador un archivo Gcode veremos algo como esto:

```
%
(Header)
(Generated by gcodetools from Inkscape.)
(Using default header. To add your own header create file "header" in the output dir.)
M3
(Header end.)
G21 (All units in mm)
Start cutting path id: rect7379)
(Change tool to Default tool)
G0 Z5.000000
G0 X2.989792 Y22.073966
G1 Z-1.000000 F100.0(Penetrate)

G1 X2.989792 Y3.183819 Z1.000000 F400.000000
G1 X21.200291 Y3.183819 Z1.000000
G1 X21.200291 Y22.073966 Z1.000000
G1 X2.989792 Y22.073966 Z1.000000

G0 Z5.000000
(End cutting path id: rect7379)
(Footer)
M5
G0 X0.0000 Y0.0000
M2
(Using default footer. To add your own footer create file "footer" in the output dir.)
(end)
%
```

Lo resaltado en azul es lo que contiene un archivo tipo Gcode en su interior, donde “G1” es un comando utilizado para mover un trazo a una coordenada con componentes X,Y y Z, pero es fundamental saber que también se tienen en cuenta los comandos “G0”, estos comando hacen únicamente un cambio de posición del lápiz pero sin hacer ningún trazo, en este caso este es un archivo que describe las trayectorias para dibujar un cuadrado pero aquí la componte Z no representa un lápiz sino un cabezal de un mototul, por ello la componente Z tiene valores mayores a 1 en algunos momentos por que se usan para perforar tablas de madera subiendo y bajando el cabezal. Pero en nuestro caso se interpretarán las coordenadas mayores a 1 como un trazo con el lápiz abajo. Las cuatro líneas centrales espaciadas por un salto de línea son las cuatro trayectorias principales para definir este cuadrado y se evidencia que la componente Z es siempre 1 diciendo que se debe hacer un trazo siguiendo estos cuatro puntos. Los demás comandos no se tienen en cuenta como lo son M5, M2, M3, F, etc. Únicamente lo comandos “G1” y “G0”.
Así que la lógica para leer el vector formado es simple, si se encuentran los caracteres “G1” o “G0” se leerán las coordenadas siguientes hasta encontrar un salto de línea y a medida que se va leyendo cada coordenada le envía por comunicación serial al microcontrolador de Vex.
Para iniciar la comunicación serial se con el lenguaje M se debe hacer lo siguiente en la ventana de comandos del software:

```
close all;
com='COM4';
delete(instrfind({'Port'},{com}));
puerto_serial=serial(com);
puerto_serial.BaudRate=9600;
puerto_Serial.Paritiy = 'none';
puerto_Serial.DataBits = 8;
puerto_Serial.StopBits = 1;
puerto.Timeout = 1;
set(puerto_serial,'Timeout',0.1);
puerto.Terminator ='LF';
warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
```

En estas líneas de comandos inicializan todas condiciones para la comunicación serial, se define el puerto, la velocidad o el BaudRate que son los baudios por segundo a que se va a comunicar el computador con el microcontrolador, se define la comunicación sin paridad, el número de bits de cada dato en este caso de 8 bits, se define un bit de parada y un tiempo de un segundo para leer lo que pueda enviar el microcontrolador al computador.

Habiendo definido una lógica para leer y recorrer el vector que contiene todos los caracteres del archivo al tener una coordenada leída y se quiera enviar se debe primero abrir el puerto seleccionado en este caso de ejemplo el COM4 con el siguiente comando:

```
fopen(puerto_serial);
```

Donde “puerto_serial” contiene todos los parámetros de comunicación como una estructura y se los pasa a la función fopen para poner en acción la comunicación.
Una vez abierto el puerto se puede enviar una a una cada componente con la función “fwrite” que recibe la estructura con los parámetros iniciales de comunicación y una variable que se quiera enviar en este caso se envían dos caracteres el “1” y el “0” que representarían un numero 10 al interpretarlo y después se cierra el puerto y se limpia para que no cause problemas en el funcionamiento del computador:

```
Variable = '10';
fwrite(puerto_serial, variable);
fclose(puerto_serial);
delete(puerto_serial)
clear puerto_serial;
```

La configuración en RobotC al programar el microcontrolador es la siguiente:

```
// Setup the two UART ports
configureSerialPort(uartOne, uartUserControl);
configureSerialPort(uartTwo, uartUserControl);
setBaudRate(uartOne, baudRate9600);
setBaudRate(uartTwo, baudRate9600);
```

*Imagen 39. Configuración comunicación serial en RobotC.*

En la imagen *39* se define el UART1 y UART2 del microcontrolador Vex Arm Cortex como control de usuario, es decir sin ninguna configuración para algún sensor o dispositivo de la plataforma Vex, simplemente una comunicación para recibir byte por byte, y finalmente se configura la misma velocidad de 9600 baudios por segundo para comunicarse con el software de Matlab.
En el lenguaje de RobotC cuando se quiere enviar algo por serial se usa el siguiente comando:

```
sendChar(uartOne,'T');
```

Esta función recibe el puerto que se quiere usar en este caso el UART1 y como segundo parámetro un carácter a enviar donde aquí como ejemplo enviaría el carácter "T".
Para recibir se usa la función "getChar( )" donde se le pasa únicamente a la función el puerto a leer y esta retorna el carácter que ha leído y se guarda en una variable tipo char que hemos llamado "incomingChar" en este ejemplo eligiendo el UART2 para ser leído:

```
incomingChar = getChar(uartTwo);
```

Con esto es suficiente para hacer una comunicación entre el software de Matlab y el microcontrolador de Vex, estos comandos son los principales en esta comunicación, ya podemos enviar coordenadas una a una desde un computador al microcontrolador Vex Arm Cortex definiendo que hacer con estas coordenadas en la parte de control con máquinas de estados y el lenguaje C fundamental a la hora de programar el microcontrolador. En nuestro caso toda la parte de comunicación se definiría como el estado 0 de la maquina general mostrada en la imagen *37*.

## III. Conclusiones

El uso del microcontrolador Vex Arm Cortex desde su configuración básica es posible manejarse para diferentes proyectos que involucren un control lógico usando los pines digitales de este declarándolos como salidas o entradas para manipular y leer diferentes dispositivos que sean ajenos a la plataforma de Vex.

A la hora de crear una solución con el microcontrolador de Vex y un dispositivo de mayor capacidad es mejor dar la mayor carga a este dispositivo ya sea en almacenamiento o procesamiento para no colapsar ningunos de los periféricos del microcontrolador como en el caso especial de un plotter de manejo de más de mil coordenadas que el microcontrolador no soporta.

La comunicación serial del microcontrolador de Vex es útil para el enviar y recibir información importante en un sistema, donde es posible enviar y recibir estados de sensores, control de actuadores, datos representados en cadenas de caracteres, etc.

Teniendo en cuenta la mecánica y el montaje realizado, la realización de una impresora 3D con el microcontrolador Vex Arm Cortex funcionaria con la misma metodología y manejo de ejes cartesianos del plotter realizado y.

Cambiando el lápiz planteado en el diseño del plotter por cualquier otro elemento ya sea un mototul, un extrusor, un marcador o cualquier aparato que pueda marcar una trayectoria ampliaría las aplicaciones de este proyecto en diversidad de campos.

Conociendo varios lenguajes de programación es posible maximizar las soluciones para crear sistemas con más complejidad utilizando y aprovechando la mayor cantidad de herramientas posibles.

La construcción mecánica tiene un mismo peso al compararse con la parte electrónica y el control ya que si se tiene un control demasiado bueno pero una mecánica pobre, esto no sacara adelante la realización de un plotter y de igual manera se tiene una mecánica perfecta pero un control torpe.

## Referencias


[1] C. Reprap, «Prusa i2,» RepRap, España, 2011.

[2] Wikipedia, «G-code,» Wikimedia Foundation, 2017.

[3] MathWorks, «Serial Comunication,» MathWorks, 2017.

[4] RobotC, «Serial Link Functions,» ROBOTC API Guide, 2016.

[5] R. a. C. p. l. f. robotics, «VEX2 Sensors Overview,» ROBOTC API Guide, 2017.

[6] A. D. Academy, «Get free video training in Inventor®,» Autodesk, 2017.

[7] Allegro, «DMOS Microstepping Driver with Translator,» 2014.

[8] P. L. a. p. b. CO, «Stepper Motor NEMA 17,» PBC LINEAR a pacific bearing CO, 2017.

[9] R. forums, «Vex Serial Port usage. (SerialTransmitting function),» RoboMatter forums, 2008.