CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

REPORTE DE PROYECTO INDUSTRIAL

"Desarrollo y puesta en marcha de un robot cartesiano, utilizando un ARM Cortex M4 para la generación de trayectorias de inspección con ultrasonido"

QUE PARA OBTENER
LA ESPECIALIDAD TECNÓLOGO EN MECATRÓNICA

PRESENTA
MIGUEL ANGEL GUALITO OLVERA

Tutor Académico
Dr. Jorge Alberto Soto Cajiga

Tutor Planta
MCyT Alejandro Gómez Hernández

QUERETARO,QRO. JULIO 2018.

Índice

Índice de figuras

Índice de tablas

RESUMEN

En este trabajo se presenta el desarrollo y puesta en marcha de un robot cartesiano, utilizando un microcontrolador con un un núcleo ARM Cortex M4, en particular un microcontrolador TM4C123GH6PM de la marca *Texas Instruments*. Este micorocontrolador fue utilizado para el control del robot y para la generación de trayectorias de inspección. El desarrollo de este robot fue necesario para la validación de un método, para el análisis de señales del ultrasonido desarrollado en el Centro de Ingeniería y Desarrollo Industrial (CIDESI). Para cumplir con los requerimientos del método, a este robot se le integro un sistema de adquisición de señales de ultrasonido para poder realizar la inspección y se genero una interfaz hombre-maquina, desde la cual se puede controlar tanto el robot como el sistema de adquisición.

1. Introducción

Para determinar el estado de un material se hace uso de los denominados ensayos no destructivos, que se define como cualquier tipo de prueba practicada a un material que no altere de forma permanente sus propiedades físicas, químicas, mecánicas o dimensionales[1]. Un ejemplo de estos es haciendo uso de señales ultrasónicas, denominado inspección con ultrasonido. Este es un método muy utilizado ya que es de tipo volumétrico; esto es, permite examinar todo el volumen o elemento que se desea inspeccionar. La aplicación de ultrasonidos se conoce sobre todo en el ámbito del diagnóstico médico. En la industria, los ultrasonidos se utilizan en numerosos procesos, por ejemplo, para limpiar, soldar, comprobar materiales, separar, mezclar, medir y mucho más.

Desde hace veinte años, el campo de las pruebas no destructivas en México, se ha encaminado a un proceso de actualización y cumplimiento de estándares internacionales. En el país existen pocas personas especializadas (en comparación con la población total del territorio mexicano), pues no rebasan la cantidad de 60 niveles III en alguna de las técnicas de inspección; esto de acuerdo con la Sociedad Americana de Pruebas No Destructivas[2][3].

Un sistema de inspección convencional por ultrasonido consiste en un único transductor piezoeléctrico emisor/receptor, el cual genera un pulso ultrasónico que se propaga en el material produciendo una reflexión en todos los puntos donde encuentre una discontinuidad, quedando éstas registradas mediante el pulso reflejado [3].

El proceso manual de sostener un trasductor en una mano y tratar de tomar los datos con la otra es un proceso muy ineficiente. El enfoque de la robótica en este campo es colocar un trasductor en un sistema automatizado y recopilar los datos de una manera estructurada, con un movimiento mucho mas rápido y preciso que con la forma convencional, esto significa que el análisis se centra puramente en los datos de inspección. Esto se traduce en ahorra de tiempo y aumento de la calidad.

2. Planteamiento del problema

En el Centro de Ingeniería y Desarrollo Industrial(CIDESI) se desarrollan diferentes proyectos en los que se involucra la inspección por ultrasonido. En uno de estos proyectos es necesario inspeccionar objetos con una separación especifica entre muestra y muestra (A y B), y con una separación (C) constante entre el transductor y el objeto inspeccionado siguiendo el esquema de la Figura 1. Estos datos posteriormente son recolectados y procesados, la inspección debe ser repetida con diferentes objetos para validar el método desarrollado.

La inspección generalmente se realiza de forma manual en donde la confiabilidad y repetibilidad de los resultados obtenidos depende en gran medida de la capacidad del usuario para aplicar correctamente la técnica; incluyendo la operación adecuada del equipo.De manera que, es necesario un equipo que pueda repetir esta inspección de manera confiable y rápida.

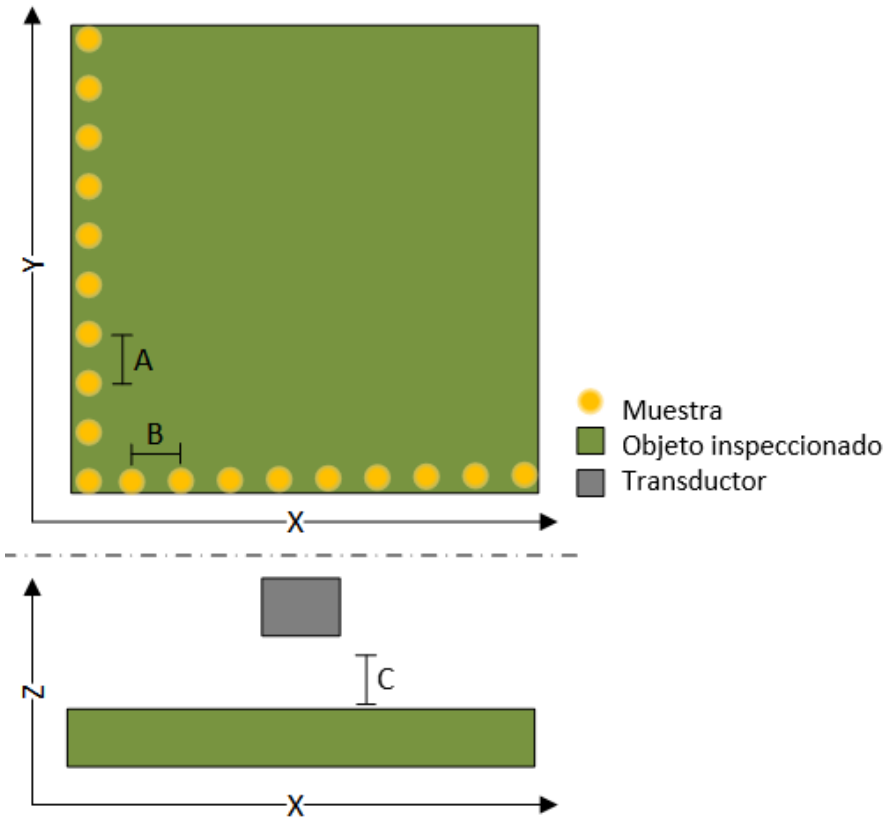

Figura 1: Esquema general del experimento.

3. Justificación

El desarrollo y puesta en marcha de un robot cartesiano capaz de seguir trayectorias de inspección con ultrasonido, aumentara la confiabilidad y repetibilidad de las inspecciones realizadas. Adicionalmente las inspecciones se realizaran con mayor velocidad sin requerir que estas sean realizadas por un experto.

Siendo el robot un sistema automatizado para la inspección por ultrasonido permite repetir el ensayo de inspección múltiples veces bajo las misma condiciones permitiendo estudiar alteraciones en las señales de ultrasonido obtenidas durante la inspección que no pueden ser adjudicadas a un error humano en la aplicación de la técnica de inspección.

Al utilizar un ARM Cortex como el procesador principal del Robot, permitirá que este sea un sistema con parámetros altamente configurables en comparación con los sistemas comerciales. Permitiendo que el robot se adapte a los diferentes paramentos de inspección requeridos por el método desarrollado en CIDESI. Ademas al ser un desarrollo propio de CIDESI existe la posibilidad de que el sistema al se multi-plataforma y no estar sujeto a una marca o lenguaje especifico, pueda integrarse el robot a futuros proyectos.

4. Objetivos

Objetivo general

Desarrollar y poner en marcha un robot cartesiano capaz de seguir y generar trayectorias de inspección con ultrasonido, utilizando un ARM Cortex M4.

Objetivos específicos

- Validar el ensamble mecánico y eléctrico del robot cartesiano.
- Desarrollar algoritmo para el control de los diferentes grados de libertad del robot.
- Desarrollar algoritmo para la generación de trayectorias para la inspección.
- Validar repetibilidad y resolución del robot cartesiano.
- Integración del equipo de adquisición de ultrasonido con el sistema del robot cartesiano.

5. Marco teórico

5.1. Inspección con ultrasonido

Los ensayos no destructivos son técnicas de inspección que se utilizan para verificar la sanidad interna y externa de piezas, componentes y materiales sin deteriorarlos ni alterar o afectar en forma permanente sus propiedades físicas, químicas y mecánicas. Los ensayos no destructivos pueden ser usados en la materia prima, durante el proceso de fabricación o en el producto final, así como para encontrar discontinuidades que se generan durante el servicio, lo cual permite la remoción previa de la pieza dañada y prevenir así un desastre. Los ensayos no destructivos son usados para producir productos más confiables y seguros. Por ejemplo, en la producción de aeronaves, en la tecnología nuclear, en la exploración espacial, en tanques y recipientes a presión, donde es esencial tener una máxima confiabilidad [4].

El ultrasonido industrial, mostrado en la Figura 2, es un método de inspección volumétrico, ya que se puede examinar todo el volumen o elemento que se desea inspeccionar. Los ensayos superficiales, como partículas magnéticas, sólo pueden detectar discontinuidades abiertas a la superficie o muy cercanas a ella. El ultrasonido son vibraciones mecánicas que se transmiten en el material por medio de ondas de la misma naturaleza que el sonido, pero con frecuencia mayor a los 20000 ciclos/segundo(Hz), es decir fuera del rango audible del oído humano [4].

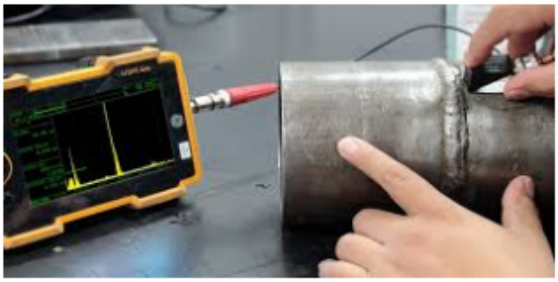

Figura 2: Prueba de Ultrasonido Industrial.[12]

El origen de la técnica de inspección con ultrasonido se remonta al conocido ensayo de percusión de la muestra con un martillo y la percepción del sonido

emitido. Por muchos años, fue común observar a los empleados del ferrocarril golpeando las ruedas de los vagones con un martillo ligero, con el fin de encontrar discontinuidades, sin embargo, los ensayos de sonoridad son muy simples para la detección de discontinuidades pequeñas en la forma precisa [4].

El uso de ondas ultrasónicas para encontrar defectos en objetos de metal fue propuesto primeramente por Sokolov en 1929. En 1930 se reconoció el uso del ultrasonido como ensayo no destructivo. Investigadores rusos y alemanes desarrollaron el método de inspección con ultrasonido con el cual se pueden detectar discontinuidades, pero una de las limitaciones que tenían estos primeros equipos era que tanto la superficie frontal como la posterior debían ser accesibles. Un acenso muy significativo se hizo cuando, casi al mismo tiempo, Firestone en América y Sproule en Inglaterra, introdujeron primeramente un detector de fallas pulso-eco, el cual requiere el acceso sólo por un lado. El sistema pulso-eco, sigue siendo el más popular en ensayos no destructivos con ultrasonido [4].

5.1.1. Aplicaciones de la inspección con ultrasonido

La inspección con ultrasonido es uno de los métodos de inspección no destructiva más ampliamente usados y encuentra aplicaciones en los siguientes ramos industriales:

- Biología: Pregerminación de semillas, putrefacción interna de troncos de árboles, medición de capas de tocino y muslos en porcino vivo.
- Comunicaciones: Señales submarinas, radar y otros sistemas de mensajes.
- Química: Preparación de reacciones y floculación.
- Medicina: Diagnósticos y exploraciones del cuerpo humano.
- Navegación y pesca: Navegación marina y sondas de profundidad.
- Industria quimica: Preparación de coloides, desagasificación de líquidos, medición de espesores en polímeros.
- Metal- Mecánica: Control de calidad en productos fundidos, maquinados, forjados, etc. Inspección de recipientes a presión, sistemas de tuberías, medición de espesores.

- Automotriz: Inspección de puntos de soldaduras, medición de espesores y control de calidad de materiales.
- Aeronáutica: Medición de espesores, control de calidad en materiales metálicos y compuestos.

La inspección con ultrasonido de metales se utiliza para la detección de discontinuidades. Este método puede utilizarse para detectar fallas internas en la mayoría de los metales y aleaciones de ingeniería. Las discontinuidades a ser detectadas incluyen poros, huecos, inclusiones, laminaciones, sopladuras, grietas, etc. Se han desarrollado procedimientos de inspección y documentación relacionada. Ejemplos de estos documentos son el código ASME, código AWS, normas API, normas ASTM [4].

5.1.2. Ventajas de la inspección con ultrasonido

- Gran velocidad de prueba: La operación electrónica proporciona indicaciones prácticamente instantáneas de la presencia de discontinuidades.
- Mayor exactitud: En comparación con los demás métodos no destructivos, la determinación de la posición de discontinuidades internas, estimando sus tamaños, orientaciones, forma y profundidad es más exacta.
- Alta sensibilidad: Permite la detección de discontinuidades a una gran profundidad.
- Buena resolución: Siendo esta característica la que determina que puedan detectarse los ecos procedentes de discontnuidades próximas a la superficie o que permita la discriminación de ecos provenientes de dos discontinuidades muy cercanas entre si.
- Acceso: La técnica pulso-eco requiere el acceso por una sola cara del material.
- Permite La interpretación inmediata, la automatización y el control del proceso de fabricación.

- No utiliza radiaciones perjudiciales para el organismo humano y no tiene efectos sobre el material inspeccionado.
- Puede dejar registro permanente de las inspecciones realizadas y evaluaciones a través de computadora.

5.1.3. Desventajas de la inspección con ultrasonido

- La inspección manual requiere mucha atención y concentración de técnicos experimentados.
- Se requiere un gran conocimiento técnico para el desarrollo de los procedimientos de inspección.
- Las piezas de geometría compleja, rugosas, de grano, grueso, porosas, demasiado ásperas, muy pequeñas, muy delgadas o no homogéneas son difíciles de inspeccionar.
- Se necesita usar patrones de referencia, tanto para calibrar el equipo como para caracterizar las discontinuidades.
- El patrón de referencia debe ser el mismo material o acústicamente equivalente al material de la pieza que se va a inspeccionar.
- Alto costo de equipo y accesorios.
- Convencionalmente, la transmisión del ultrasonido a la pieza se realiza por medio de un acoplante líquido o pastoso. La transmisión del ultrasonido a la pieza también puede realizarse a través del aire por medio de láser, presión o generando el ultrasonido con un transductor acústico electromecánico.

5.2. Robótica

La robótica se puede definir como la ciencia aplicada que combina la tecnología de informática y las máquinas-herramientas. Los robots están destinados a la fabricación flexible de productos, la definición japonesa de robot industrial se refiera a cualquier dispositivo mecánico que posee articulaciones destinado a la

manipulación. Esta definición difiere mucho de la occidental la cual ha establecido varias definiciones dados por organizaciones de la industria. La asociación De Industrias Robóticas (RIA), define el robot Industrial como un manipulador multifuncional reprogramable, capaz de mover materiales, piezas, herramientas o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas. (Barrientos. 1997) La Asociación Francesa de Normalización da una definición mas completa de un robot industrial basándose en dos definiciones, primero define manipulador y a partir de esta definición construye el concepto de robot.:

- Manipulador: mecanismo formado generalmente por elementos en serie, articulados entre si destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o un dispositivo lógico.

- Robot: Manipulador automático, servocontrolado, reprogramable, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables, para la ejecución de tareas variables. Normalmente, tiene la forma de brazos terminados en una muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Realiza una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material.

Diferentes asociaciones dan definiciones validadas acerca de los robots industriales siempre incluyendo los términos de manipulador automático o programable, cada asociación le realiza a la definición los cambios que consideran necesarios. En cuanto a la clasificación de los robots, estos se pueden clasificar con base en la tabla 1 dada por la Asociación Francesa de Robótica Industrial(AFRI)[5]

5.2.1. Morfología del robot

La morfología del robot se refiere a la constitución física del robot en la cual se observa la composición de este identificando cada una de sus partes. La configuración de los robots industriales usualmente se asemeja al cuerpo humano,

es decir, posee un cuerpo y un brazo, por lo general el cuerpo se encuentra en una parte fija de la mesa o esta montado sobre un riel y el brazo es el encargado de realizar las tareas ordenadas. Las partes que conforman un robot se pueden dividir en las siguientes[6]:

- Estructura mecánica.
- Transmisiones y Reducciones.
- Sistemas de accionamiento.
- Sistema sensorial.
- Sistema de control.
- Elementos terminales

Las articulaciones de los robots industriales realizan un movimiento relativo de las uniones contiguas, estos movimientos pueden ser lineales o rotacionales o en ocasiones una combinación de los dos, determinando diferentes tipos de configuraciones las cuales se observan en la Figura 3

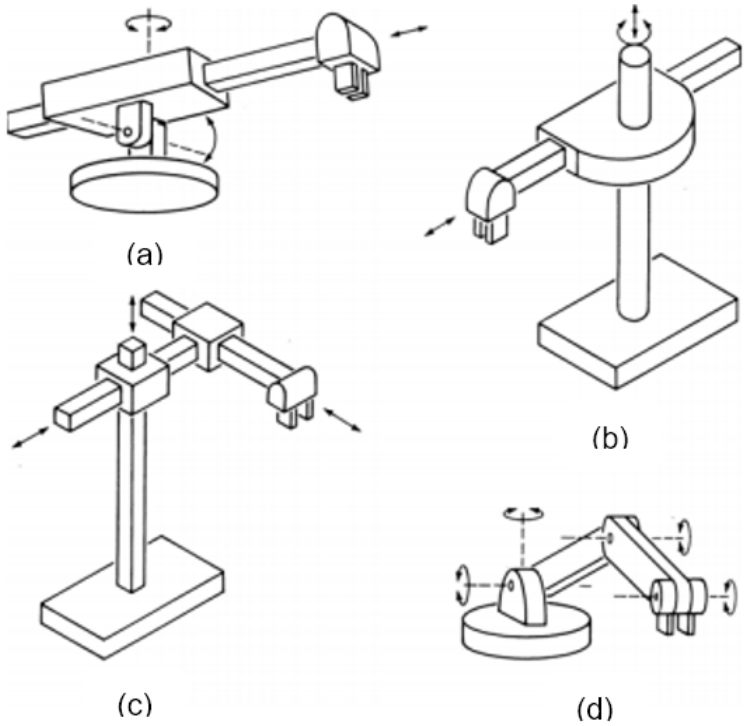

Figura 3: Configuraciones típicas de los robots industriales.[ ]

La configuración esférica como se observa en la Figura 3 a), el brazo telescopio se puede elevar o bajar alrededor de un pivote horizontal. El pivote se encuentra sobre una mesa giratoria. Esta combinación de articulaciones le permite al robot desplazar su brazo en un espacio esférico.

La configuración cilíndrica la cual se muestra en la Figura 3 b), utiliza un dispositivo deslizante que se mueve a través de una columna, este a su vez se encuentra unido a un dispositivo que le permite realizar un movimiento radial con respecto a la columna. Esta combinación de movimientos genera un espacio de trabajo aproximado a un cilindro.

La configuración cartesiana mostrada en la Figura 3 c) permite que cada una de sus articulaciones se deslice en línea recta a través de los ejes de coordenadas xyz, esta configuración forma un poliedro en su espacio de trabajo.

La configuración mostrada en la Figura 3 d), es de tipo antropomórfica, esta se asemeja al brazo humano; por lo tanto consta de antebrazo y brazo, están conectados a dos articulaciones giratorias denominadas codo y hombro. En cuanto a ventajas y desventajas de cada una de las configuraciones, estas están directamente relacionadas a su geometría. Los robots cartesianos presentan ventajas en cuanto a repetibilidad en los movimientos, ya que su estructura es fija, pero presenta desventaja en cuanto al alcance, donde las configuraciones esféricas y antropomórficas son las ideales. En cuanto a la carga de materiales y maquinaría la configuración cartesiana es la adecuada[5][6].

5.3. Motores a pasos

Un motor paso a paso es un dispositivo electromecánico que convierte una serie de pulsos eléctricos en desplazamientos angulares, lo que significa que es capaz de girar una cantidad de grados (paso o medio paso) dependiendo de sus entradas de control. Los motores paso a paso son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos. La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de 1.8°, Es por eso que ese tipo de motores son muy utilizados, ya que pueden moverse a deseo del usuario según la secuencia que se les indique

a través de un microcontrolador[7]. Estos motores poseen la habilidad de quedar enclavados en una posición si una o más de sus bobinas está energizada o bien total mente libres de corriente [8].

Figura 4: Motor a pasos.[9]

Para poder realizar los movimientos con los motores a pasos es necesario utilizar un controlador de motor a pasos, para proteger la placa de control del manejo de la corriente, el control de la secuencia de conmutación y la corriente de bobinado para el motor paso a paso.

5.3.1. Motores a pasos Bipolares

Para hacer girar un motor paso a paso bipolar, se aplican impulsos en secuencia los devanados, la secuencia de estos impulsos, se aplican externamente con un controlador electrónico. Dichos controladores, se diseñan de manera que el motor se pueda mantener en una posición fija y también para que se le pueda hacer girar en ambos sentidos. Los motores bipolares, se pueden hacer avanzar a frecuencias de audio, lo que les permite girar muy velozmente. Este es el motivo por que se suele decir que un motor “canta”, debido a la frecuencia a la que se produce la conmutación. Con un controlador apropiado, se les puede hacer arrancar y detenerse en cualquier instante y en una posición determinada. Este es en teoría, el esquema de un motor bipolar.

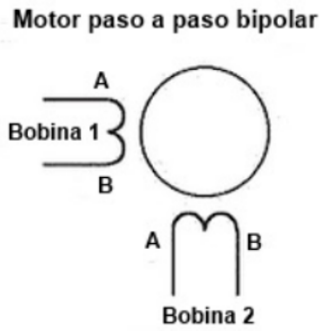

Figura 5: Motor a pasos.[9]

En la Figura 5se puede observar que estos motores, tienen varios bobinados que, para producir el avance de un paso, deben ser alimentados en una secuencia adecuada. Al invertir el orden de esta secuencia, se logra que el motor gire en sentido opuesto. El torque de detención, hace que un motor paso a paso (esto vale para los bipolares y unipolares) se mantenga firmemente en su posición, estando alimentado aun cuando no esté girando.

6. Metodología

6.1. Tareas a realizar y fechas límites

Tabla 1: Lista de actividades

ID de actividad	Nombre	Fecha
1	Definición de Proyecto	Mayo 5, 2018
2	Desarrollo de la metodología	Mayo 18, 2018
3	Validar el ensamble mecánico y eléctrico del robot	Junio 15, 2018
4	Desarrollar algoritmo para el control del robot	Junio 1, 2018
5	Desarrollar algoritmo para la generación de trayectorias	Junio 25, 2018
6	Validar repetibilidad y resolución del robot cartesiano	Julio 29, 2018
7	Integración del equipo de adquisición de ultrasonido	Agosto 3 ,2018
8	Revisión de Reporte	Agosto 10, 2018
9	Presentación	Agosto 17, 2018

Para el desarrollo del proyecto se propuso el diagrama de solucion mostrado en la Figura 6. En el se puede observar como se interconectan los diferentes dispositivos que conforman el robot y el sistema de inspección por ultrasonido, que en este caso es un equipo comercial de la marca Ultratek. Tambien se puede observar que los motores correspondientes al eje X (Motor X y Motor X') comparten un solo controlador, por lo tanto trabajaran de manera paralela.

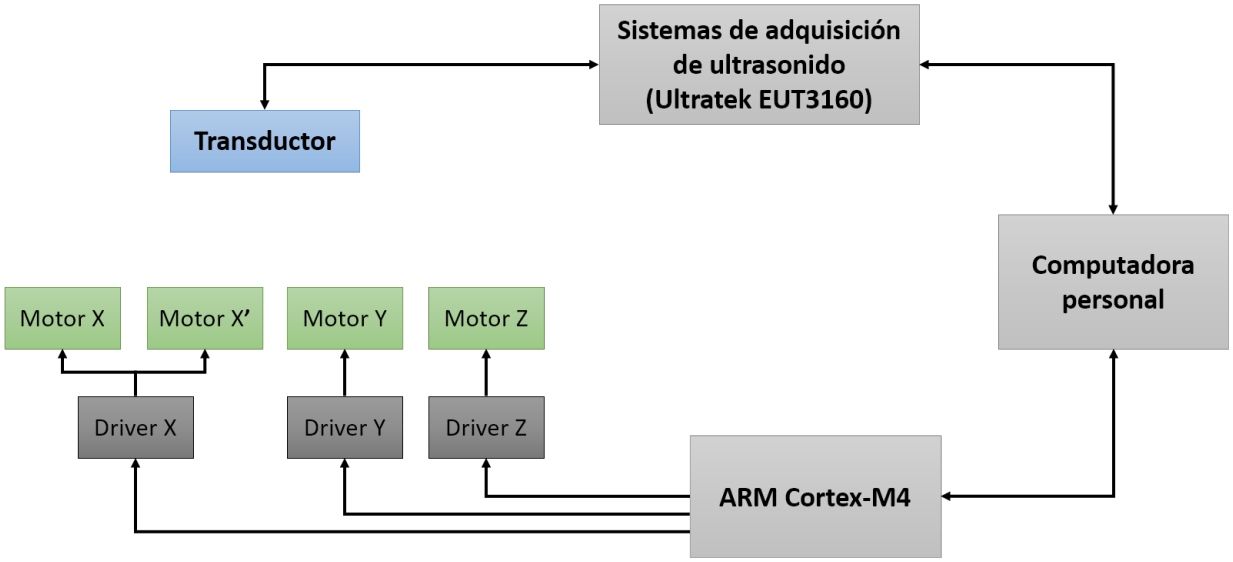

Figura 6: Diagrama general de la solución.

6.2. Validación del ensamble mecánico y eléctrico del robot

Este trabajo parte de un robot cartesiano previamente construido como el que se observa en la Figura 7. Antes de comenzar el desarrollo del algoritmo de control para este robot, se valido el correcto funcionamiento de los ensambles mecánicos[9].

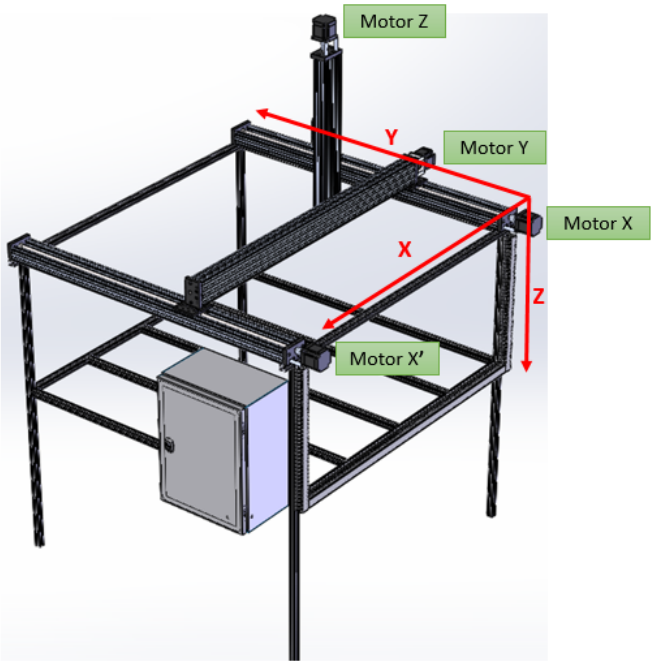

Figura 7: Modelo 3D del robot cartesiano

Durante la validación de los ensambles mecánicos y eléctrico se determino la resolución máxima. Para se alizar este calculo se analizo un solo eje como el que se muestra en le Figura 8 ,dado que todos los ejes poseen las mismas características mecánicas y eléctricas.

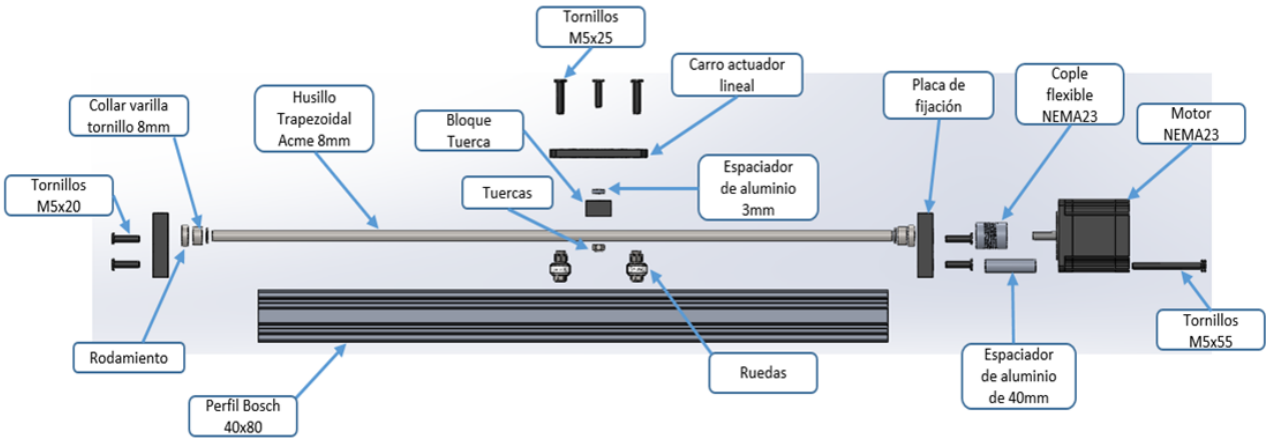

Figura 8: Despiece de un eje.

Las características de los componentes que afectan a la resolución del robot son las siguientes:

- Husillo trapezoidal Acme : 8mm por rev.
- DQ542MA Stepper Motor Driver: 1/2 pasos
- Motores a pasos NEMA23: 200 paso por rev.

Tomando en cuenta todas estas características se puede determinar que por cada 50 pasos que avance el motor el efector lineal se desplaza un milímetro, por lo tanto, la resolución máxima del robot es de 0.02mm.

6.3. Desarrollo de algoritmo para el control del robot

Para este proyecto se seleccionó el driver DQ542MA que es un controlador para un motor a pasos híbrido con una entrada de tensión de 18 a 50 VDC/VCC, diseñado para ser usado con un motor que tiene un cuerpo de 42 mm a 86 mm de ancho y está clasificado con corriente de fase de hasta 4.2A de corriente. El procesador de señal digital usado en la DQ542MA es similar a un control servo, lo que resulta en un funcionamiento más suave, menos ruidoso, menos vibración con un mayor par de retención a alta velocidad y una mayor precisión de posicionamiento. El DQ542MA puede utilizarse en dispositivos de control numérico de tamaño medio y grande, como máquinas CNC, bordadoras y empaquetadoras.

Al revisar la hoja de datos del driver DQ542MA, se puede obtener el Algoritmo 1 que sirve para que el motor gire un paso en un sentido, este sentido es determinado por el estado en que se ponga la entrada DIR del driver. Para que todo esto pueda ser efectuado el driver deber ser habilitado previamente con al menos 100us de antelación al igual que la dirección.

Algoritmo 1: Girar un paso en dirección A

```
1 ENA ← 1;
2 DIR ← 0;
3 Esperar al menos 100 uS;
4 PUL ← 0;
5 Esperar 100 uS;
6 PUL ← 1;
7 Esperar 100 uS;
8 PUL ← 0;
9 Esperar al menos 100 uS;
```

Con el algoritmo anterior es posible controlar el avance de un paso del motor y la dirección de movimiento del motor, para poder usar este algoritmo para el control del robot fue necesario generar otro algoritmo que contemplara la relación de paso y la distancia lineal desplazada por el actuado. Tomando en cuenta esta información se genera el Algoritmo 2 que ya toma en cuenta esta relación.

Algoritmo 2: Algoritmo para el desplazamiento lineal en milímetros

```
1  Pasos ← Redondear(Distancia / 50);
2  ENA ← 1;
3  DIR ← 0;
4  Esperar al menos 100 uS;
5  for i ← 1 to Pasos by 1 do
6      PUL ← 0;
7      Esperar 100 uS;
8      PUL ← 1;
9      Esperar 100 uS;
10     PUL ← 0;
11 end
12 Esperar al menos 100 uS;
```

Con el Algoritmo 2 es posible controlar el desplazamiento lineal del actuador en milímetros, en cualquier dirección. La dirección del desplazamiento depende del valor previamente colocado en la entrada DIR del driver (linea 3). Dado que, los motores a pasos requieren al menos una rampa de aceleración al iniciar el movimiento, para no perder o saltarse pasos, se tiene que modificar el Algoritmo 2 para que el motor tenga un comportamiento parecido al de la Figura 9.

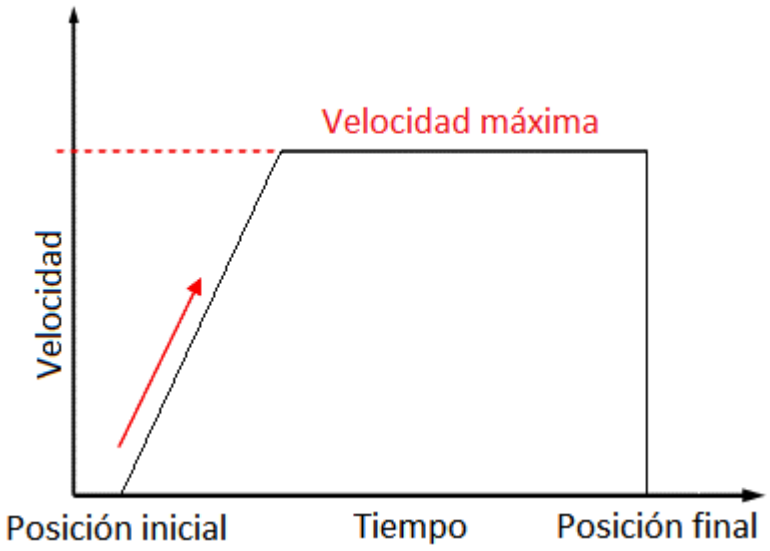

Figura 9: Rampa de aceleración.

Definiendo una velocidad máxima de $100mm$ por minuto y una aceleración: $12000mm/s^2$ sabiendo que por cada 50 pasos el efector se desplaza un milímetro, obtenemos que la cantidad máxima de pulsos que se deben generar en un minuto es de 5000, esto es aproximadamente 83 pulsos por segundo. Con esta información se puede determinar que el tiempo mínimo entre pulso y pulso es de aproximadamente $12mS$. Tomando un tiempo entre pulso y pulso máximo de $100mS$ de acuerdo a lo recomendado por el fabricante del driver. Se obtiene el Algoritmo 3 que ya contempla esta rampa de aceleración al iniciar el movimiento del motor.

Algoritmo 3: Algoritmo para el desplazamiento lineal en milímetros con rampa de aceleración

```
1  Pasos ← Redondear(Distancia / 50);
2  Velocidad ← 88;
3  ENA ← 1;
4  DIR ← Direccion;
5  Esperar al menos 100 uS;
6  for i ← 1 to Pasos by 1 do
7      PUL ← 0;
8      Esperar (12 + velocidad) mS;
9      PUL ← 1;
10     Esperar (12 + velocidad) mS;
11     PUL ← 0;
12     if Velocidad == 0 then
13         Velocidad ← 0 ;
14     else
15         Velocidad ← Velocidad - 11;
16     end
17 end
18 Esperar al menos 100 uS;
```

El Algoritmo 3 cumple con todas las características necesarias apara controlar de manera correcta cual quiera que los actuadores del robot, por lo tanto con este algoritmo se genero una función a la cual se nombro *MoverEjeX* que es la

correspondiente al eje X por lo tanto se generaron dos funciones mas, *MoverEjeY* y *MoverEjeZ* para desplazar los ejes Y y Z respectivamente. Los parámetros que reciben como entrada estas funciones son *Distancia* y *Direccion*, indicando cuanto y hacia donde se quiere desplazar el efector, que dando la función de la siguiente manera: *MoverEjeX(Distancia,Direccion)*.

6.4. Desarrollo de algoritmo para generación de trayectorias

La inspección aunque generalmente se realiza de forma manual sigue una trayectoria ordenada y continua, para cumplir con los requisitos de los proyectos que se desarrollan en CIDESI se plantea un trayectoria de inspección típica como la de la Figura 10.

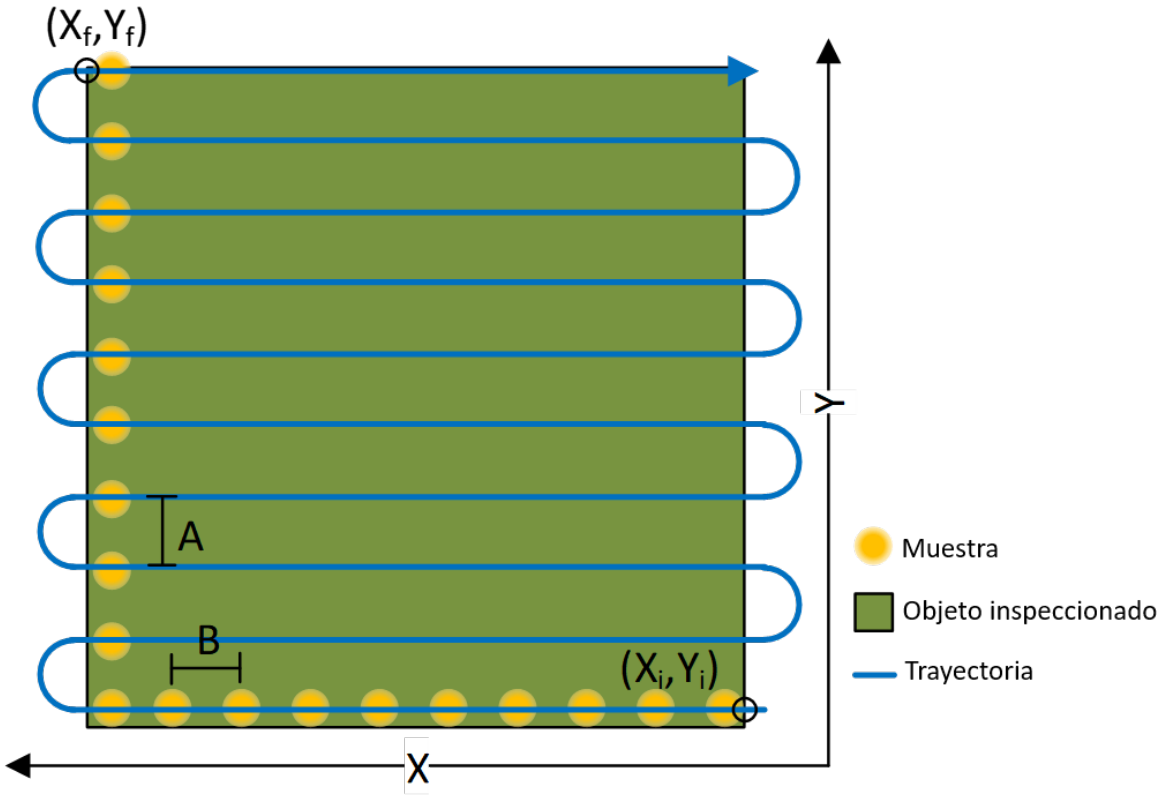

Figura 10: Trayectoria típica de inspección.

Esta trayectoria brinda los puntos necesarios para que el robot se desplace siguiendo esa ruta, también permite determinar los puntos en lo que se deberá tomar la muestra de ultrasonido, esto se calcula con base a los parámetros A

y B. En algoritmo desarrollado se pueden configurar los siguientes parámetros parámetros:

- X_i: Coordenada inicial en X, indica la coordenada en X aparir de la cual se iniciara la inspección.
- Y_i: Coordenada inicial en Y, indica la coordenada en Y aparir de la cual se iniciara la inspección.
- X_f: Coordenada final en X, indica la coordenada en X donde finalizara la inspección.
- Y_f: Coordenada final en Y, indica la coordenada en X donde finalizara la inspección.
- A: Delta Y o incremento en Y, indica la distancia entra muestra y muestra que sera captura respecto al eje Y.
- B: Delta X o incremento en X, indica la distancia entra muestra y muestra que sera captura respecto al eje X.

Para la inspección se requiera que el transductor de ultrasonido este a una distancia contante respecto al objeto inspeccionado, por lo tanto para este caso una vez iniciada la inspección el eje Z permanecerá fijo, haciendo posible que solo se asigne una coordenada fija para el eje Z, dejando fuera esta coordenada del generador de trayectorias.

La generación de trayectorias es la representación de las coordenadas generalizadas como funciones del tiempo, qi(t) para todo i = 1, 2, $\cdots$, n, de tal manera que se satisfagan ciertas condiciones orientadas a la realización de tareas [ ]. La forma mas sencilla de generar estas trayectorias es usar una interpolación lineal para unir todos los puntos necesarios para la inspección. Para generar la trayectorias de esta inspección se puede analizar en dos partes, la generación de las partes rectas de la trayectoria y la generación de las partes curvas de la trayectoria.

Para la generación de las partes rectas de la trayectoria existen dos posibles casos, cuando se desplaza sobre el eje X de manera positiva alejando se del

origen y de manera negativa acercándose al origen. Para la generación de las partes curvas de la trayectoria, también existen dos posibles caso aun que ambos se desplazan de manera positiva alejando se del origen sobre el eje Y, en uno de los caso el giro se hace de manera horaria y el otro de manera anti horaria. Sabiendo esto la generación de la trayectoria se puede dividir en en 4 partes o 4 estados, para hacer mas fácil el calculo y el análisis de la trayectoria. Definiendo estos estados de la siguiente manera:

- Estado 1: Desplazarse de manera positiva sobre el eje X.
- Estado 2: Girar en sentido horario.
- Estado 3: Desplazarse de manera negativa sobre el eje X.
- Estado 4: Girar en sentido anti-horario.

Conociendo esta información se puede dividir el algoritmo principal para la generación de trayectorias en cuatro sub-algoritmos o sub-funciones, asi el algoritmo principal solo tendrá que hacer el llamado correspondiente al estado actual de la trayectoria. Tomando todo esto en consideración es como se genero el Algoritmo 4 para el *Estado 1* y el Algoritmo 5 para el *Estado 3*

Algoritmo 4: Desplazamiento en X positivo

```
1 Xobjetivo ← X_final;
2 while (Xactual < Xobjetivo) do
3     MoverEjeX(DeltaX,1);
4     CapturarUS;
5     Xactual ← Xactual + DeltaX;
6 end
```

Algoritmo 5: Desplazamiento en X negativo

```
1 Xobjetivo ← X_inicial;
2 while (Xactual > Xobjetivo) do
3     MoverEjeX(DeltaX,1);
4     CapturarUS;
5     Xactual ← Xactual - DeltaX;
6 end
```

Para el *Estado 2* y el *Estado 4* la función para generar estas secciones de la trayectoria que son las curvas, es un tanto mas complicas, al requerir que los dos ejes se muevan al mismo tiempo. Para facilitar el calculo de estas secciones de la trayectoria, para la parte curva solo se calcula una cantidad definida de punto, por ejemplo en la Figura 11 se muestra la parte curva seccionada en 5 puntos.

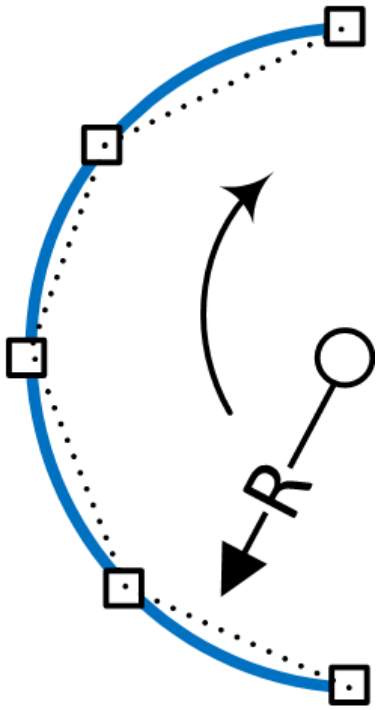

Figura 11: Generación de curvas.

Una vez que se calculan esos puntos, se puede interpolar una trayectoria recta entre cada dos puntos como la que se ve marcada con linea punteada en la Figura 11. Se puede observar que a mayor cantidad de puntos calculados para la curva, mas suavizado es este movimiento en la curva.

Para optimizar la generación de la trayectoria y reducir el tiempo de calculo se define una cantidad máxima de 20 puntos para interpolar la sección de la curva. Estos puntos se calculan a partir de las funciones seno y coseno para las coordenadas X y Y respectivamente, como si de un circulo se tratara, pero solo usando la parte izquierda del circulo para la rotación horaria y la parte derecha para la rotación anti-horaria. Por ejemplo para la rotación horaria se se calcularon los siguientes 20 puntos:

$$incrementoYhorario = \begin{pmatrix} 0{,}0061558 \\ 0{,}0183159 \\ 0{,}0300250 \\ 0{,}0409948 \\ 0{,}0509551 \\ 0{,}0596608 \\ 0{,}0668974 \\ 0{,}0724868 \\ 0{,}0762913 \\ 0{,}0782172 \\ 0{,}0782172 \\ 0{,}0762913 \\ 0{,}0724868 \\ 0{,}0668974 \\ 0{,}0596608 \\ 0{,}0509551 \\ 0{,}0409948 \\ 0{,}0300250 \\ 0{,}0183159 \\ 0{,}0061558 \end{pmatrix} \quad (1)$$

Si se suman todos los punto del arreglo incremento Y horario, tenemos como resultado un valor muy cercano a 1, puesto solo se necesita avanzar un delta Y, los puntos fueron calculados para un radio de uno, por lo tanto deben ser multiplicados por delta Y para que la trayectoria se desplace Delta y milímetros en Y.

Los datos calculados para el giro anti-horario son exactamente los mismo para el caso de el eje Y, por que también solo se requiere que avance Delta Y milímetros en esta dirección.

Para definir el movimiento el el eje X se calcularon los siguientes 20 valores:

$$incrementoXhorario = \begin{pmatrix} 0{,}0782172 \\ 0{,}0762913 \\ 0{,}0724868 \\ 0{,}0668974 \\ 0{,}0596608 \\ 0{,}0509551 \\ 0{,}0409948 \\ 0{,}030025 \\ 0{,}0183159 \\ 0{,}00615583 \\ -0{,}00615583 \\ -0{,}0183159 \\ -0{,}030025 \\ -0{,}0409948 \\ -0{,}0509551 \\ -0{,}0596608 \\ -0{,}0668974 \\ -0{,}0724868 \\ -0{,}0762913 \\ -0{,}0782172 \end{pmatrix} \quad (2)$$

Si se suman todos los punto del arreglo incremento X horario, tenemos como resultado un valor muy cercano a 0, puesto solo se necesita avanzar un delta Y, los puntos fueron calculados para un radio de uno, por lo tanto deben ser multiplicados por delta Y para que los datos corresponda a los de un semi-circulo al relacionarlos con los puntos Y.

Los datos calculados para el giro anti-horario también al ser sumados obtenemos un valor muy cercano a cero, pero para que se genere el movimiento anti horario este arreglo tiene que ser leído de manera invertida o cambiando el signo del contenido del arreglo y leerlo de manera convencional.

Usando toda esta información se obtuvieron los siguientes algoritmos para calcular las trayectorias en estas secciones.

Algoritmo 6: Giro horario

```
1  Xobjetivo ← Xactual;
2  Yobjetivo ← Xactual;
3  for i ← 0 to 19 by 1 do
4      Xobjetivo ← Xobjetivo + incrementoXhorario[i];
5      Yobjetivo ← Yobjetivo + incrementoYhorario[i];
6      while (Xactual != Xobjetivo or Yactual != Yobjetivo) do
7          if (Xactual < Xobjetivo) then
8              MoverEjeX(1,1);
9              Xactual ← Xactual + 1;
10         else if (Xactual > Xobjetivo) then
11             MoverEjeX(1,0);
12             Xactual ← Xactual - 1;
13         else
14             NOP;
15         end
16         if (Yactual < Yobjetivo) then
17             MoverEjeY(1,1);
18             Yactual ← Yactual + 1;
19         else if (Yactual > Yobjetivo) then
20             MoverEjeY(1,0);
21             Yactual ← Yactual - 1;
22         else
23             NOP;
24         end
25     end
26 end
```

Algoritmo 7: Giro Anti-horario

```
1  Xobjetivo ← Xactual;
2  Yobjetivo ← Xactual;
3  for i ← 0 to 19 by 1 do
4      Xobjetivo ← Xobjetivo − incrementoXhorario[i];
5      Yobjetivo ← Yobjetivo + incrementoYhorario[i];
6      while (Xactual != Xobjetivo or Yactual != Yobjetivo) do
7          if (Xactual < Xobjetivo) then
8              MoverEjeX(1,1);
9              Xactual ← Xactual + 1;
10         else if (Xactual > Xobjetivo) then
11             MoverEjeX(1,0);
12             Xactual ← Xactual - 1;
13         else
14             NOP;
15         end
16         if (Yactual < Yobjetivo) then
17             MoverEjeY(1,1);
18             Yactual ← Yactual + 1;
19         else if (Yactual > Yobjetivo) then
20             MoverEjeY(1,0);
21             Yactual ← Yactual - 1;
22         else
23             NOP;
24         end
25     end
26 end
```

Con estos algoritmos se puede generar la trayectoria en sus cuatro estados, agregando un *Estado 0* para llevar el robot antes de iniciar la trayectoria al punto inicial. De esta manera se genero el siguiente algoritmo principal para la generación de trayectorias.

6.5. Validación repetibilidad y resolución del robot cartesiano

Para validar la repetibilidad y resolución del robot cartesiano se realizo un experimento como el que se muestra en la Figura 12. Para que el robot al desplazarse dibujara su trayectoria en una superficie plana y así posteriormente poder medir este desplazamiento y que tan similar es este movimiento a la trayectoria planeada. Los resultados de estas medicines se muestran en la sección de resultados.

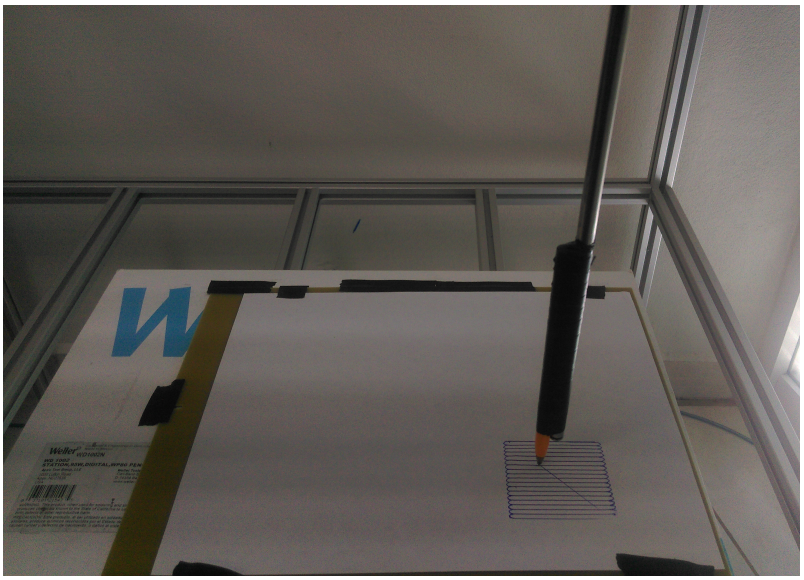

Figura 12: Experimento para validación de repetibilidad y resolución.

6.6. Integración del equipo de adquisición de ultrasonido

Para integrar el equipo para adquisición de ultrasonido Ultratek EUT3160 dado que este este usa un protocolo ethernet para comunicarse, se utilizo LabVIEW como plataforma de programación, para generar una interfaz hombre-maquina desde la cual se puedieran enviar los comandos necesarios para controlar el robot y poder gestionar y visualizar la adquisicion de las muestras de ultrasonido.

La programación del microcontolador TM4C123GH6PM basado en un ARM cortex-M4 se realizo en la plataforma *Code Composer Studio* de la marca *Texas Instruments*. En este microcontrolador se implementaron todos los algoritmos antes mencionados, ademas de todos los protocolos necesarios para la comunicación con la PC. El código comentado se encuentra en la sección de anexos.

7. Resultados

Como resultado de las mediciones del primer experimento, que consistió en inspeccionar un área a de 150mm por 200mm con un *Delta X* igual a 10mm y un *Delta Y* igual a 10mm, dibujando esta trayectoria de inspección en una hoja de papel como se puede observar en la Figura 13, y posterior a esto tomar medidas con un vernier. Después de repetir el experimento 10 veces y analizar los datos se obtuvo lo siguiente:

Media aritmética (μ) : 9.88

Desviación estándar (σ) : 0.074833

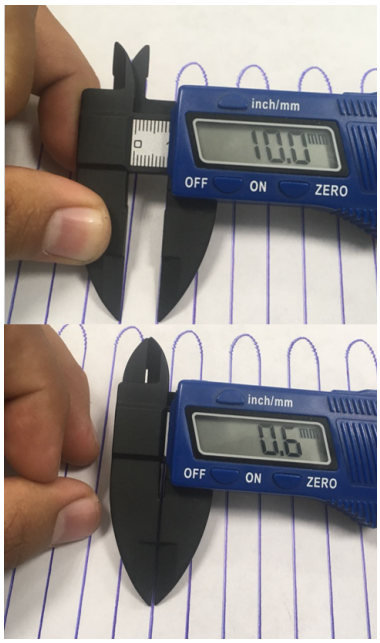

Figura 13: Resultados de experimento 1.

Repitiendo el experimento, pero ahora inspeccionando un área de 50mm por 50mm con un *Delta X* igual a 3mm y un *Delta Y* igual a 3mm, dibujando esta trayectoria de inspección en una hoja de papel como se puede observar en la Figura 14, y posterior a esto tomar medidas con un vernier. Después de repetir el experimento 10 veces y analizar los datos se obtuvo lo siguiente:

Media aritmética (μ) : 2.84

Desviación estándar (σ) : 0.048989

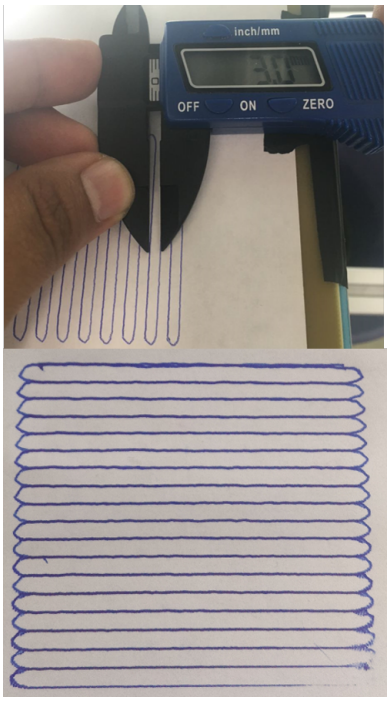

Figura 14: Resultados de experimento 1.

Como resultado de la integración del equipo de adquisición de ultrasonido y el robot, se obtuvo una interfaz hombre-maquina como la que se muestra en las Figuras 15,16 y 16. Esta interfaz esta dividida en tres pestañas principales desde las que se pueden controlar o configurar todos los parámetros del robot y del sistema de adquisición de ultra sonido.

En las Figuras 15,16 y 16 se puede observar que existe un botón *STOP* y un botón *Detener operacion* y estos son accesibles desde cual quier pestaña de la interfaz. El botón *Detener operacion* cancela la ejecución de la tarea en curso pero no inhabilita los Drivers de los motores por lo tanto el robot permanece en su posición bloqueando el giro de los motores. El boton *STOP* cancela la tarea en curso pero ademas inhabilita los Drivers de los motores haciendo que estos queden libres y puedan ser girados de manera manual. En esta interfaz tambien se puede observar un indicador de tipo texto *Estado Actual* el cual nos muestra como se encuentras el robot ademas de la tarea que esta realizando.

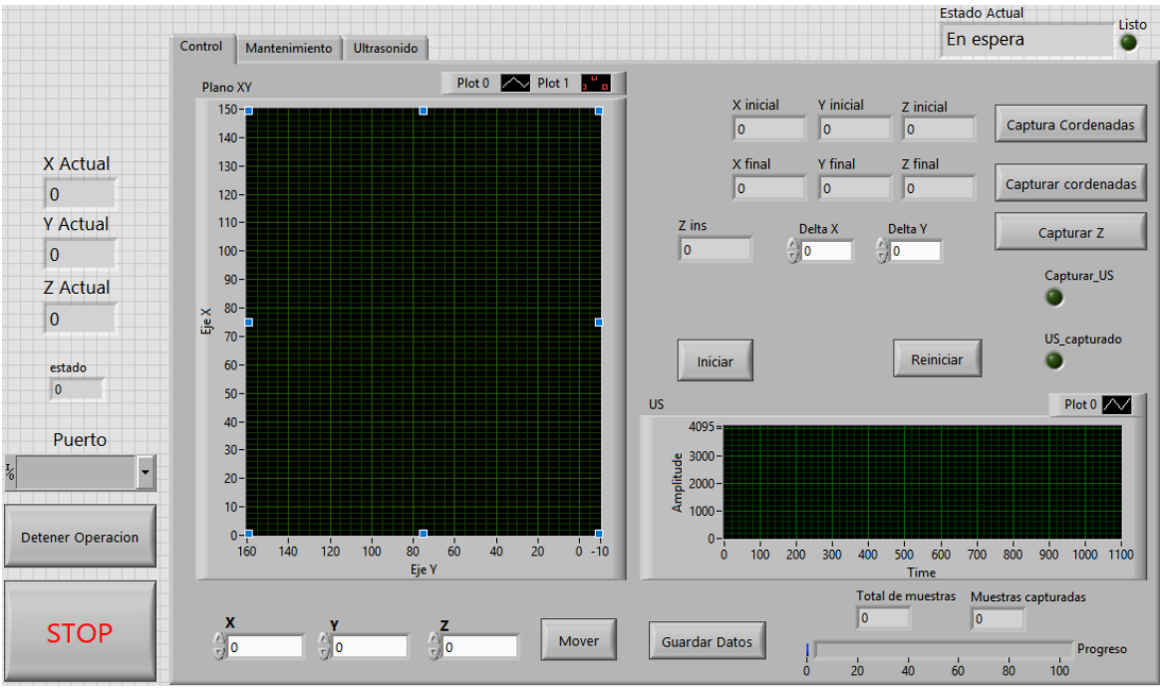

Figura 15: Pestaña Control de la interfaz hombre-maquina.

Desde la pestaña de Control, Figura 15 se le puede indicar al generador de trayectoria la información necesaria para generar la trayectoria, como son: Puntos iniciales, Puntos finales, incrementos en X y Y o la altura o coordenada en Z. En esta pestaña se puede observar la posición del robot en el plano XY de forma gráfica; y también de muestra de forma gráfica la ultima señal de ultrasonido capturada esto en el gráfico US.

Desde la pestaña de Mantenimiento, Figura 16 se puede ver el estado actual del robot en los planos XY y plano YZ. También se puede ver el estado de todos los sensores de limite ademas de poder controlar de manera manual el robot, ya sea eje a eje o mandando llamar algunas tareas como ir a HOME alguno o todos sus ejes, medir alguno o todos sus ejes o habilitar y des-habilitar cada uno de los Drivers de los motores en caso de que fuera necesario algún mantenimiento mecánico.

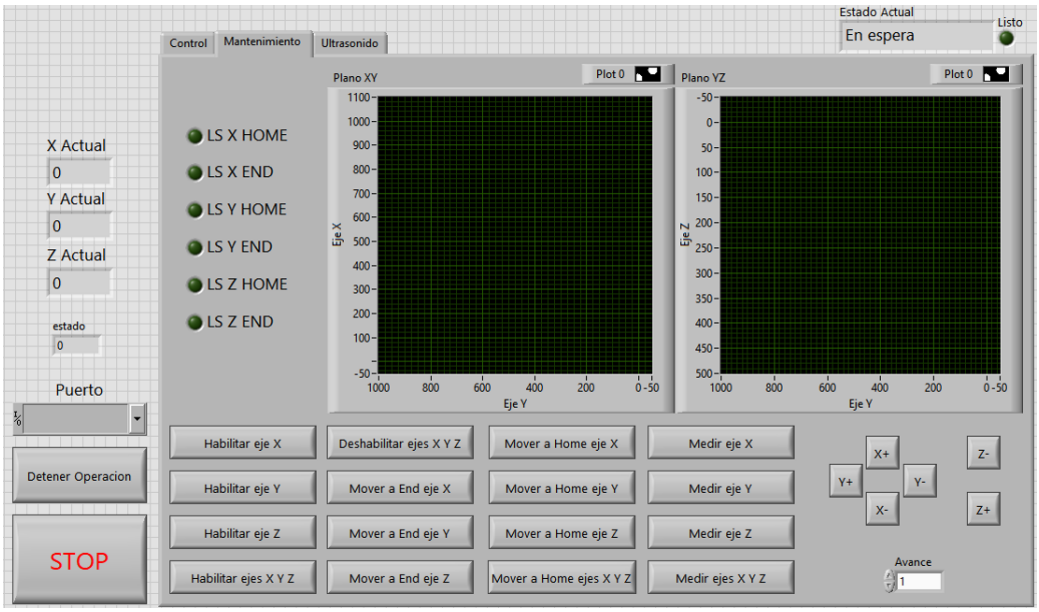

Figura 16: Pestaña Mantenimiento de la interfaz hombre-maquina.

Desde la pestaña de Ultrasonido, Figura 17, se pueden configurar todos los parámetros que permite el sistema de adquisición de ultrasonido como el voltaje con el que se excita el transductor, el tiempo que dura el pulso de excitación, filtro pasa bajas , filtro pasa bajas , velocidad de muestro, etc. También permite tomar una muestra de ultrasonido para validar las configuraciones establecidas y verificar que sean las indicadas.

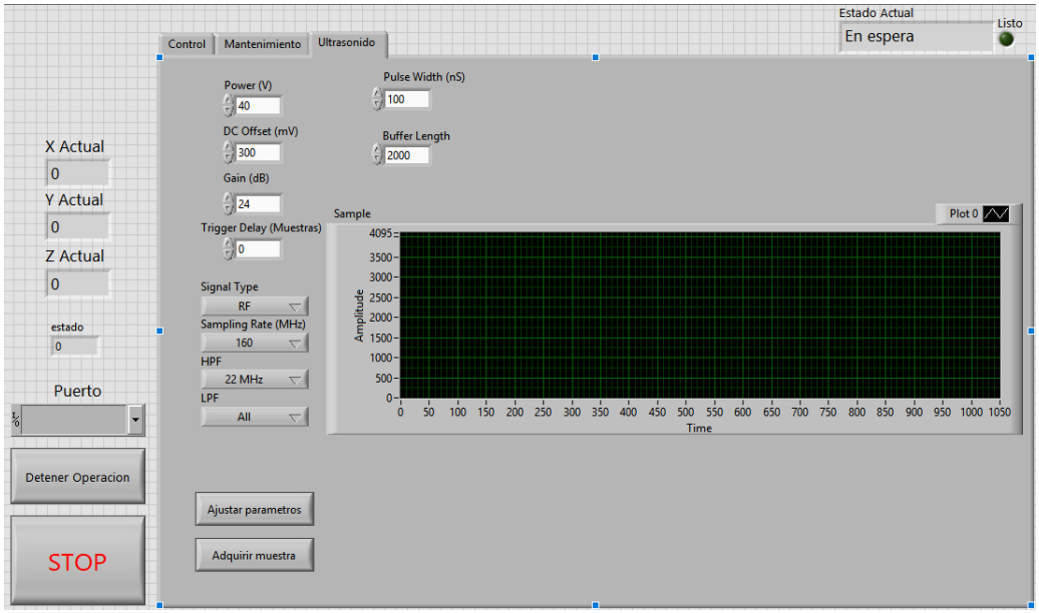

Figura 17: Pestaña Ultrasonido de la interfaz hombre-maquina.

8. Conclusiones

Debido a los resultados de las pruebas realizadas se concluye que es posible reducir en gran medida el error en la captura de las muestras de ultrasonido, al automatizar este proceso, ademas este proceso es mas rápido y repetible.

Se recomienda sustituir el sistema de adquisición de ultra sonido actual por uno en el que se puedan configurar todos los parámetros dado que el actual solo posee una lista especifica que no permite configurar valores intermedios o ajenos a esa lista.

También se recomienda migrar la interfaz hombre-maquina a una plataforma que no requiera licencia como Python o C, esto permitiría ejecutar esta interfaz en una computadora embebida eliminado el uso de una computadora personal, ademas esto abriría la posibilidad a probar nuevos métodos para analizar las señales de ultrasonido en un solo sistema.

9. Bibliografía

[1] CARRION VIRAMONTES, F. J., LOMELI GONZALEZ, M. G., QUINTANA RODRIGUEZ, J. A., & MARTINEZ MADRID, M. (2003). La evaluación no destructiva de materiales estructurales y puentes. Publicación Técnica, (231).

[2] Díaz Zavala, M. A., Rosales Alemán, C. P., & Salinas Gutiérrez, D. I. Proyecto de una empresa dedicada a efectuar pruebas NDT.

[3] Torres, F. A. (2018). Ajuste de equipo de medición por ultrasonido usando un brazo robótico. Pistas Educativas, 36(113).

[4] SAGRERO Rivera, Jorge, Ultrasonido Industrial Nivel I, Centro de Ingeniera y Desarrollo Industrial

[5] Barrientos, A. (1997). Fundamentos de Robótica. España: Mc Graw Hill.

[6] Godoy Hernández, R. D., & Rodríguez Quintero, W. (2007). Diseño y Modelamiento de un Robot Cartesiano para el Posicionamiento de Piezas.

[7] Jennings, S. (2002). Motores paso a paso. Informador Técnico, 65, 47-58.

[8] Solidbi . (2016). Solid-bi. Consultado en Mayo del 2018 de https://solidbi.es/solidworks/

[9] Ávila Díaz, C. I., & Herrera Medina, M. A. (2018). Diseño y construcción de un dispositivo absorvedor puntual tipo on-shore para estudio del potencial energético undimotriz en el departamento de córdoba (Doctoral dissertation).

[10] CIDESI, Diablo instrumentado para inspeccion de ductos obtenido el 23 de agosto de 2018 de http://cidesi.com/wsite/destacados/inspeccion-ductos.php

[11] NAVA Balanzar, Luciano, Diseno de un sistema electronico para la medicion de espesores por ultrasonido, Queretaro, Qro., Mexico, CIDESI, 2010.

[12] Reyes, F. (2011). Robótica-Control de robots manipuladores. Alfaomega Grupo Editor.

[13] Peña, E., & Ramírez, H. R. M. Aplicación de Métodos Numéricos en la Robótica para la Generación de Trayectorias.

[14]Cajiga, J. S., Vargas, J. E., & Pedraza, J. C. (2006). Generación de trayectorias para un robot manipulador utilizando procesamiento de imágenes.

[15]Caprile, S. R. (2012). Desarrollo con microcontroladores ARM Cortex-M3. Sergio R. Caprile.

[16]Martínez Arias, B. R. (2007). Arquitectura de procesadores ARM de 32 BITS, circuitos integrados y entornos de desarrollo (Bachelor's thesis, SANGOLQUÍ/ESPE/2007).

10. Anexos

Código principal del microcontrolador.

```
//#############################################################
//----------------Comunicacion Serial-------------------------
//#############################################################
/*
XXXXX -----> 0000.0mm  a  9999.9mm intervalo de operacion
LL ------> valor decimal de estado de limits(0 a 63)
E  ------> estado del control (0,1)
RESELLXXXXXYYYYYZZZZZEND
enviar
**= no importa
CMD00**************END    : Para comandos que no requieren argumento
CMD00XXXXXYYYYYZZZZZEND    : Para comandos para ajustar cordenadas
CMDs
00 = deshabilitar drivers
01 = habilitar todos los driver
02 = habilitar eje X
03 = habilitar eje Y
04 = habilitar eje Z
05 = todos los ejes a home
06 = X a home
07 = Y a home
08 = Z a home
09 = X a end
10 = Y a end
11 = Z a end
12 = cal X
13 = cal Y
14 = cal Z
15 = cal todos
16 = mover a cordenada
17 = status actual
18 = obtener area de trabajo
19 = detener operacion actual
20 = fijar cordenadas iniciales para ruta snake
21 = fijar cordenadas finales para ruta snake
22 = fijar avances para ejes X y Y de ruta snake
25 = snake travel
*/
//#############################################################
//---------------------LIBRERIAS-------------------------------
//#############################################################
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_memmap.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/rom.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/uart.h"
#include "inc/hw_gpio.h"
#include "inc/hw_types.h"
#include "inc/hw_ints.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom_map.h"
//#############################################################
//---------------------DEFINICIONES----------------------------
//#############################################################
#define LED_R   GPIO_PIN_1    //PUERTO F
#define LED_B   GPIO_PIN_2    //PUERTO F
#define LED_G   GPIO_PIN_3    //PUERTO F
```

```
#define SW_1    GPIO_PIN_4    //PUERTO F
#define SW_2    GPIO_PIN_0    //PUERTO F

#define LSX_HOME    GPIO_PIN_4    //PUERTO C
#define LSX_END     GPIO_PIN_5    //PUERTO C
#define LSY_HOME    GPIO_PIN_4    //PUERTO A
#define LSY_END     GPIO_PIN_6    //PUERTO C
#define LSZ_HOME    GPIO_PIN_7    //PUERTO C
#define LSZ_END     GPIO_PIN_6    //PUERTO D

#define ENABLED_X   GPIO_PIN_0    //PUERTO D
#define DIR_X       GPIO_PIN_7    //PUERTO D
#define PUL_X       GPIO_PIN_2    //PUERTO A
#define ENABLED_Y   GPIO_PIN_3    //PUERTO D
#define DIR_Y       GPIO_PIN_2    //PUERTO D
#define PUL_Y       GPIO_PIN_1    //PUERTO D
#define ENABLED_Z   GPIO_PIN_3    //PUERTO E
#define DIR_Z       GPIO_PIN_2    //PUERTO E
#define PUL_Z       GPIO_PIN_1    //PUERTO E
//#############################################################
//-------------_---VARIABLES GLOBALES-----------------------
//#############################################################
unsigned char SW1_state=0;
unsigned char SW2_state=0;

unsigned char LSX_HOME_state=0;
unsigned char LSX_END_state=0;
unsigned char LSY_HOME_state=0;
unsigned char LSY_END_state=0;
unsigned char LSZ_HOME_state=0;
unsigned char LSZ_END_state=0;
unsigned char LIMIT_SWITCHS_state=0;

int cnt=0,status=0;

int area_x=50000,area_y=50000,area_z=50000;
int act_x=0,act_y=0,act_z=0;
int des_x=0,des_y=0,des_z=0;
//#####################################
//----VARIANBLES PARA RUTINA SNAKE
//#####################################
int X_inicial=0,Y_inicial=0,Z_inicial=0;
int X_final=0,Y_final=0,Z_final=0;
int avance_x=3,avance_y=3,avance_z=0;;
int X_point=0,Y_point=0;
int capturar=0;
int estado=0;
int giro=0;
int nivel=0;
int iniciado=0;

const float tabla_1a[20]={0.00615583,0.0183159,0.030025,0.0409948,0.0509551,0.0596608,0.0668974,0.0724868,
                          0.0762913,0.0782172,0.0782172,0.0762913,0.0724868,0.0668974,0.0596608,0.0509551,
                          0.0409948,0.030025,0.0183159,0.00615583};

const float tabla_1b[20]={0.0782172,0.0762913,0.0724868,0.0668974,0.0596608,0.0509551,0.0409948,0.030025,
                          0.0183159,0.00615583,-0.00615583,-0.0183159,-0.030025,-0.0409948,-0.0509551,
                         -0.0596608,-0.0668974,-0.0724868,-0.0762913,-0.0782172};

const float tabla_2a[20]={0.00615583,0.0183159,0.030025,0.0409948,0.0509551,0.0596608,0.0668974,0.0724868,
                          0.0762913,0.0782172,0.0782172,0.0762913,0.0724868,0.0668974,0.0596608,0.0509551,
                          0.0409948,0.030025,0.0183159,0.00615583};

const float tabla_2b[20]={-0.0782172,-0.0762913,-0.0724868,-0.0668974,-0.0596608,-0.0509551,-0.0409948,
                          -0.030025,-0.0183159,-0.00615583,0.00615583,0.0183159,0.030025,0.0409948,
```

```
                                0.0509551,0.0596608,0.0668974,0.0724868,0.0762913,0.0782172};

char data_in[23]={"CMDNNXXXXXYYYYYZZZZZEND"};
char data_out[24] = {"RESELLXXXXXYYYYYZZZZZEND"};
char dato='0';
//#############################################################
//----------------PROTOTIPO DE FUNCIONES----------------------
//#############################################################
void delay_ms(int tiempo);
void setup();
void UARTSend(const char *charBuffer, char charCount);
void actualizar_data_out(int var_x, int var_y, int var_z);

void actualizar_limits();
void deshabilitar_todos();
void habilitar_todos();
void habilitar_x();
void habilitar_y();
void habilitar_z();
void home_z();
void end_z();
void cal_z();
void home_y();
void end_y();
void cal_y();
void home_x();
void end_x();
void cal_x();

/* Avanzar se mueve en direccion a END retroceder se mueve a HOME*/
void avanzar_x();
void retroceder_x();
void avanzar_y();
void retroceder_y();
void avanzar_z();
void retroceder_z();

//*****************************************************************************
// The UART interrupt handler.
//*****************************************************************************
void
UARTIntHandler(void)
{
    uint32_t ui32Status;// Get the interrrupt status.
    int shifter=0;

    ui32Status = UARTIntStatus(UART0_BASE, true);// Clear the asserted interrupts.
    UARTIntClear(UART0_BASE, ui32Status);// Loop while there are characters in the receive FIFO.

    while(UARTCharsAvail(UART0_BASE))
    {

        for(shifter=0;shifter<22;shifter++){

            data_in[shifter]=data_in[shifter+1];
        }

        dato=UARTCharGetNonBlocking(UART0_BASE);
        data_in[22] = dato;
    }
    if(data_in[0]=='C'&&data_in[1]=='M'&&data_in[2]=='D'&& data_in[20]=='E'&&data_in[21]=='N'&&data_in[22]=='D'){

        GPIOPinWrite(GPIO_PORTF_BASE, LED_B, LED_B);

        if(data_in[3]=='1' && data_in[4]=='6'){//ajustar cordenadas
         status=17;
```

```
 actualizar_data_out(act_x, act_y, act_z);
 UARTSend(data_out,24);
 des_x = (data_in[5]-48)*10000+(data_in[6]-48)*1000+(data_in[7]-48)*100+(data_in[8]-48)*10+(data_in[9]-48);
 des_y = (data_in[10]-48)*10000+(data_in[11]-48)*1000+(data_in[12]-48)*100+(data_in[13]-48)*10+(data_in[14]-48);
 des_z = (data_in[15]-48)*10000+(data_in[16]-48)*1000+(data_in[17]-48)*100+(data_in[18]-48)*10+(data_in[19]-48);

 if(des_x > (area_x-1)){des_x=area_x-1;}
 if(des_y > (area_y-1)){des_y=area_y-1;}
 if(des_z > (area_z-1)){des_z=area_z-1;}
}
//#############################################################################################
//----------------------------------control snake
//#############################################################################################
if(data_in[3]=='2' && data_in[4]=='0'){// fijar cordenadas iniciales
    actualizar_data_out(act_x, act_y, act_z);
    UARTSend(data_out,24);
    X_inicial = (data_in[5]-48)*10000+(data_in[6]-48)*1000+(data_in[7]-48)*100+(data_in[8]-48)*10+(data_in[9]-48);
    Y_inicial = (data_in[10]-48)*10000+(data_in[11]-48)*1000+(data_in[12]-48)*100+(data_in[13]-48)*10+(data_in[14]-48);
}
if(data_in[3]=='2' && data_in[4]=='1'){// fijar cordenadas finales
    actualizar_data_out(act_x, act_y, act_z);
    UARTSend(data_out,24);
    X_final = (data_in[5]-48)*10000+(data_in[6]-48)*1000+(data_in[7]-48)*100+(data_in[8]-48)*10+(data_in[9]-48);
    Y_final = (data_in[10]-48)*10000+(data_in[11]-48)*1000+(data_in[12]-48)*100+(data_in[13]-48)*10+(data_in[14]-48);
}
if(data_in[3]=='2' && data_in[4]=='2'){//obtener avances para ruta snake

    actualizar_data_out(act_x, act_y, act_z);
    UARTSend(data_out,24);
    avance_x = (data_in[5]-48)*10000+(data_in[6]-48)*1000+(data_in[7]-48)*100+(data_in[8]-48)*10+(data_in[9]-48);
    avance_y = (data_in[10]-48)*10000+(data_in[11]-48)*1000+(data_in[12]-48)*100+(data_in[13]-48)*10+(data_in[14]-48);
    avance_z = (data_in[15]-48)*10000+(data_in[16]-48)*1000+(data_in[17]-48)*100+(data_in[18]-48)*10+(data_in[19]-48);

    if(avance_x < 150){
        avance_x=150;
    }
    else if(avance_x > 20000){
        avance_x=2000;
    }
    else{

    }

    if(avance_y < 150){
        avance_y=150;
    }
    else if(avance_y > 20000){
        avance_y = 20000;
    }
    else{

    }

}

else if(data_in[3]=='2' && data_in[4]=='5'){//snake travel
    iniciado=0;
    status=25;
}
//#############################################################################################
//----------------------------------control snake
//#############################################################################################
else if(data_in[3]=='1' && data_in[4]=='7'){//estatus actual
    actualizar_data_out(act_x, act_y, act_z);
    UARTSend(data_out,24);
}
```

```
        else if(data_in[3]=='1' && data_in[4]=='8'){//obtener area de trabajo
            actualizar_data_out(area_x, area_y, area_z);
            UARTSend(data_out,24);
        }
        else{
            status=(data_in[3]-48)*10 + data_in[4] - 48 + 1;
        }
        GPIOPinWrite(GPIO_PORTF_BASE, LED_B, 0x00);
    }

}

//#############################################################
//--------------------FUNCION PRINCIPAL-----------------------
//#############################################################
int main(void)
 {
    setup();

    while(1)
    {
        actualizar_limits();

        SW1_state = !(bool)(GPIOPinRead(GPIO_PORTF_BASE, SW_1));
        SW2_state = !(bool)(GPIOPinRead(GPIO_PORTF_BASE, SW_2));

        if(SW1_state == 1){
            //GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
            //GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_X, ENABLED_X);
        }
        else{
            //GPIOPinWrite(GPIO_PORTF_BASE, LED_R, 0);
           //  GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_X, 0);
        }

        /*estado para maquina de estados
         0 = espera
         1 = deshabilitar drivers
         2 = habilitar todos los driver
         3 = habilitar eje X
         4 = habilitar eje Y
         5 = habilitar eje Z
         6 = todos los ejes a home
         7 = X a home
         8 = Y a home
         9 = Z a home
         10 = X a end
         11 = Y a end
         12 = Z a end
         13 = cal X
         14 = cal Y
         15 = cal Z
         16 = cal todos
         17 = mover a cordenada
         25 = ejecucion de ruta snake

*/
        ///maquina de estado?

        switch(status){
        case 0://en espera
            SysCtlDelay(5);
            GPIOPinWrite(GPIO_PORTF_BASE, LED_R, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, LED_G, 0);
            //SysCtlDelay(12000000);
            break;
```

```
case 1://deshabilitar todos los ejes
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    deshabilitar_todos();
    SysCtlDelay(1000000);
    status=0;
    break;

case 2://habilitar todos lo ejes
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    habilitar_todos();
    SysCtlDelay(1000000);
    status=0;
    break;

case 3:// habilitar eje x
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    habilitar_x();
    SysCtlDelay(1000000);
    status=0;
    break;

case 4://habilitar eje y
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    habilitar_y();
    SysCtlDelay(1000000);
    status=0;
    break;

case 5://habilitar eje z
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    habilitar_z();
    SysCtlDelay(1000000);
    status=0;
    break;

case 6://todos los ejes a home
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    home_z();
    home_y();
    home_x();
    SysCtlDelay(25000000);
    status=0;
    break;

case 7://eje x a home
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    home_x();
    SysCtlDelay(25000000);
    status=0;
    break;

case 8://eje y a home
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    home_y();
    SysCtlDelay(25000000);
    status=0;
    break;
```

```
case 9://eje z a home
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    home_z();
    SysCtlDelay(25000000);
    status=0;
    break;

case 10://eje x a end
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    end_x();
    SysCtlDelay(25000000);
    status=0;
    break;

case 11://eje y a end
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    end_y();
    SysCtlDelay(25000000);
    status=0;
    break;

case 12://eje z a end
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    end_z();
    SysCtlDelay(25000000);
    status=0;
    break;

case 13://calibrar eje x
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    cal_x();
    SysCtlDelay(25000000);
    status=0;
    break;

case 14://calibrar eje y
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    cal_y();
    SysCtlDelay(25000000);
    status=0;
    break;

case 15://calibrar eje z
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    cal_z();
    SysCtlDelay(25000000);
    status=0;
    break;

case 16://calibrar todos
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G); //amarillo
    GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
    home_z();
    home_y();
    home_x();
    cal_z();
    cal_y();
    cal_x();
```

```
        SysCtlDelay(25000000);
        status=0;
        break;

    case 17://ajustar cordenadas
        if(GPIOPinRead(GPIO_PORTF_BASE, LED_G)){
            GPIOPinWrite(GPIO_PORTF_BASE, LED_G|LED_R, 0x00);
        }
        else{
            GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G);
            GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
        }

        if(act_x < des_x){
            avanzar_x();
            act_x++;
        }
        else if (act_x > des_x){
            retroceder_x();
            act_x--;
        }
        else{SysCtlDelay(5000);}

        if(act_y < des_y){
            avanzar_y();
            act_y++;
        }
        else if (act_y > des_y){
            retroceder_y();
            act_y--;
        }
        else{SysCtlDelay(4100);}

        if(act_z < des_z){
            avanzar_z();
            act_z++;
        }
        else if (act_z > des_z){
            retroceder_z();
            act_z--;
        }
        else{SysCtlDelay(4100);}

        if(act_x == des_x && act_y == des_y && act_z == des_z){
            status=0;
        }

        break;
    case 20: // detenirndo
        //SysCtlDelay(6000000);
        status=0;

        break;
    case 21:
        SysCtlDelay(6000000);
        status=0;
        break;

    case 25: //ruta snake

        if(iniciado==0){
            X_point=X_inicial;
            Y_point=Y_inicial;
            iniciado = 1;

        }
```

```
if(GPIOPinRead(GPIO_PORTF_BASE, LED_G)){
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G|LED_R, 0x00);
}
else{
    GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G);
    //GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
}



if (estado == 0){ //ir a punto inicial de la rutina snake

        if(Y_point == act_y){
            if(capturar==1){
                capturar = 0;
            }
            else{
                capturar = 1;
            }
            //capturar=!capturar;
            Y_point = Y_point + avance_y;
        }


        if(act_y < Y_final){
            avanzar_y();
            act_y++;

        }
        else{
            des_x=act_x;
            des_y=act_y;
            giro=0;
            if(act_x >= X_final || (act_x < X_final+10 && act_x > X_final-10) ){
                estado = 4;
            }
            else{
                estado = 1;
            }
        }
        SysCtlDelay(10000);


}
else if(estado == 1){

    if(act_x < des_x){
        avanzar_x();
        act_x++;
    }

    if(act_y < des_y){
        avanzar_y();
        act_y++;
    }
    else if(act_y > des_y){
        retroceder_y();
        act_y--;
    }
    else{

    }

    if(act_x == des_x && act_y == des_y ){
```

```
            if(giro < 19){
                des_x = act_x+(tabla_1a[giro]*avance_x);
                des_y = act_y+(tabla_1b[giro]*avance_x*2);
                giro++;
            }
            else if(giro==20){

                des_y=Y_final;
                des_x=X_inicial + ((nivel+1)*avance_x);
                nivel++;
                giro++;
            }
            else{
                estado=2;
                X_point=X_inicial + ((nivel+1)*avance_x);
                Y_point=Y_final - avance_y;

                if(capturar==1){
                    capturar = 0;
                }
                else{
                    capturar = 1;
                }

            }
        }
        SysCtlDelay(10000);
    }
    else if(estado == 2){

            if(Y_point == act_y){
                if(capturar==1){
                    capturar = 0;
                }
                else{
                    capturar = 1;
                }
            //capturar=!capturar;
            Y_point = Y_point - avance_y;
            }

            if(act_y > Y_inicial){
                retroceder_y();
                act_y--;

            }
            else{
                des_x=act_x;
                des_y=act_y;
                giro=0;
                if(act_x >= X_final || (act_x < X_final+10 && act_x > X_final-10)){
                    estado = 4;
                }
                else{
                    estado = 3;
                }
            }
            SysCtlDelay(10000);

    }
    else if(estado == 3){

            if(act_x < des_x){
                avanzar_x();
                act_x++;
            }
```

```
            if(act_y < des_y){
                avanzar_y();
                act_y++;
            }
            else if(act_y > des_y){
                retroceder_y();
                act_y--;
            }
            else{
            }
            if(act_x == des_x && act_y == des_y ){

                if(giro < 19){
                    des_x = act_x+(tabla_2a[giro]*avance_x);
                    des_y = act_y+(tabla_2b[giro]*avance_x*2);
                    giro++;
                }
                else if(giro==20){
                    des_y=Y_inicial;
                    des_x=X_inicial + ((nivel+1)*avance_x);
                    nivel++;
                    giro++;
                }
                else{
                    estado=0;

                    X_point=X_inicial + ((nivel+1)*avance_x);
                    Y_point=Y_inicial + avance_y;

                    if(capturar==1){
                        capturar = 0;
                    }
                    else{
                        capturar = 1;
                    }

                }
            }
            SysCtlDelay(10000);

    }
    else if(estado == 4){
        GPIOPinWrite(GPIO_PORTF_BASE, LED_R|LED_G|LED_B, 0x00);
        int cnt_del=0;
        for(cnt_del=0;cnt_del<15;cnt_del++){

            GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);

            SysCtlDelay(5000000);

            GPIOPinWrite(GPIO_PORTF_BASE, LED_R, 0x00);

            SysCtlDelay(5000000);
        }
        estado=0;
        status=0;
    }
    else{

    }


     break;


default:
```

```
            GPIOPinWrite(GPIO_PORTF_BASE, LED_G, 0x00); //rojo
            GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
            break;
        }

    }//while(1)
}//main


//#############################################################
//-------------- FUNCION PARA CONFIGURACION--------------------
//#############################################################
void setup(){

        SysCtlClockSet(SYSCTL_SYSDIV_4|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN); //50MHz

        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
        while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOA));
        SysCtlDelay(16);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
        while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOB));
        SysCtlDelay(16);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
        while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOC));
        SysCtlDelay(16);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
        while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOD));
        SysCtlDelay(16);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
        while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOE));
        SysCtlDelay(16);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
        while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF));
        SysCtlDelay(16);

        SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
        IntMasterEnable();
        GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
        UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 460800,(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |UART_CONFIG_PAR_NONE));
        IntEnable(INT_UART0);
        UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);

        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, LED_R | LED_G |LED_B);
        GPIOPinWrite(GPIO_PORTF_BASE, LED_R | LED_G |LED_B, 0x00);

       HWREG(GPIO_PORTF_BASE+GPIO_O_LOCK) = GPIO_LOCK_KEY;
       HWREG(GPIO_PORTF_BASE+GPIO_O_CR) |= GPIO_PIN_0;
        GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, SW_1);
        GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, SW_2);
        GPIOPadConfigSet(GPIO_PORTF_BASE, SW_1,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
        GPIOPadConfigSet(GPIO_PORTF_BASE, SW_2,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

        GPIOPinTypeGPIOInput(GPIO_PORTC_BASE, LSX_HOME|LSX_END|LSY_END|LSZ_HOME);
        GPIOPadConfigSet(GPIO_PORTC_BASE, LSX_HOME,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
        GPIOPadConfigSet(GPIO_PORTC_BASE, LSX_END,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
        GPIOPadConfigSet(GPIO_PORTC_BASE, LSY_END,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
        GPIOPadConfigSet(GPIO_PORTC_BASE, LSZ_HOME,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, LSY_HOME);
        GPIOPadConfigSet(GPIO_PORTA_BASE, LSY_HOME,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

        GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, LSZ_END);
        GPIOPadConfigSet(GPIO_PORTD_BASE, LSZ_END,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

        SysCtlDelay(16);
```

```
        HWREG(GPIO_PORTD_BASE+GPIO_O_LOCK) = GPIO_LOCK_KEY;
        HWREG(GPIO_PORTD_BASE+GPIO_O_CR) |= GPIO_PIN_7;
        GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, PUL_X);
        GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, ENABLED_X);
        GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, DIR_X);
        GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, ENABLED_Y);
        GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, DIR_Y);
        GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, PUL_Y);

        GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, ENABLED_Z);
        GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, DIR_Z);
        GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, PUL_Z);

        GPIOPinWrite(GPIO_PORTA_BASE, PUL_X, 0x00);
        GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_X, ENABLED_X);
        GPIOPinWrite(GPIO_PORTD_BASE, DIR_X, 0x00);
        GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_Y, ENABLED_Y);
        GPIOPinWrite(GPIO_PORTD_BASE, DIR_Y, 0x00);
        GPIOPinWrite(GPIO_PORTD_BASE, PUL_Y, 0x00);
        GPIOPinWrite(GPIO_PORTE_BASE, ENABLED_Z, ENABLED_Z);
        GPIOPinWrite(GPIO_PORTE_BASE, DIR_Z, 0x00);
        GPIOPinWrite(GPIO_PORTE_BASE, PUL_Z, 0x00);



        GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
        SysCtlDelay(1500000);
        GPIOPinWrite(GPIO_PORTF_BASE, LED_R, 0);
        SysCtlDelay(1500000);
        GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
        SysCtlDelay(1500000);
        GPIOPinWrite(GPIO_PORTF_BASE, LED_R, 0);
        SysCtlDelay(1500000);
        GPIOPinWrite(GPIO_PORTF_BASE, LED_R, LED_R);
        SysCtlDelay(1500000);
        GPIOPinWrite(GPIO_PORTF_BASE, LED_R, 0);
        SysCtlDelay(1500000);
        GPIOPinWrite(GPIO_PORTF_BASE, LED_G, LED_G);
        SysCtlDelay(6000000);
        GPIOPinWrite(GPIO_PORTF_BASE, LED_G, 0);
}
//####################################################################
//---------------FUNCION PRA GENERAR RETARDOS EN MILISEGUNDOS -------
//####################################################################
void delay_ms(int tiempo){
    long int ciclos=0;
    long int cnt=0;
    ciclos = (tiempo * 5000);
    for(cnt=0; cnt < ciclos; cnt++);
}
//####################################################################
//---------------FUNCION PARA actualizar estado de limites ----------
//####################################################################
void actualizar_limits(){
  LSX_HOME_state = (bool)(GPIOPinRead(GPIO_PORTC,LSX_HOME)&&GPIOPinRead(GPIO_PORTC,LSX_HOME)&&GPIOPinRead(GPIO_PORTC,LSX_HOME));
  LSX_END_state = (bool)(GPIOPinRead(GPIO_PORTC,LSX_END)&&GPIOPinRead(GPIO_PORTC,LSX_END)&&GPIOPinRead(GPIO_PORTC,LSX_END));
  LSY_HOME_state = (bool)(GPIOPinRead(GPIO_PORTA,LSY_HOME)&&GPIOPinRead(GPIO_PORTA,LSY_HOME)&&GPIOPinRead(GPIO_PORTA,LSY_HOME));
  LSY_END_state = (bool)(GPIOPinRead(GPIO_PORTC,LSY_END)&&GPIOPinRead(GPIO_PORTC, LSY_END)&&GPIOPinRead(GPIO_PORTC,LSY_END));
  LSZ_HOME_state = (bool)(GPIOPinRead(GPIO_PORTC,LSZ_HOME)&&GPIOPinRead(GPIO_PORTC,LSZ_HOME)&&GPIOPinRead(GPIO_PORTC,LSZ_HOME));
  LSZ_END_state = (bool)(GPIOPinRead(GPIO_PORTD,LSZ_END)&&GPIOPinRead(GPIO_PORTD,LSZ_END)&&GPIOPinRead(GPIO_PORTD,LSZ_END);

  LIMIT_SWITCHS_state=LSX_HOME_state+(LSX_END_state<<1)+(LSY_HOME_state<<2)+(LSY_END_state<<3)+(LSZ_HOME_state<<4)+(LSZ_END_state<<5);
}
//#####################################################################
//---------------FUNCION PARA deshabilitar todos los drivers ---------
//#####################################################################
```

```
// solo cuando los driver estan deshabilitado se pueden mover de manera manual(mecanicamente) los ejes
void deshabilitar_todos(){
    GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_X, ENABLED_X); //deshabilitar eje X
    GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_Y, ENABLED_Y); //deshabilitar eje Y
    GPIOPinWrite(GPIO_PORTE_BASE, ENABLED_Z, ENABLED_Z); //deshabilitar eje Z
    SysCtlDelay(16);
}
//#####################################################################
//----------------FUNCION PARA habilitar todos los drivers -----------
//#####################################################################
void habilitar_todos(){
    GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_X, 0); //deshabilitar eje X
    GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_Y, 0); //deshabilitar eje Y
    GPIOPinWrite(GPIO_PORTE_BASE, ENABLED_Z, 0); //deshabilitar eje Z
    SysCtlDelay(16);
}
//#####################################################################
//----------------FUNCION PARA habilitar driver de eje X -------------
//#####################################################################
void habilitar_x(){
    GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_X, 0); //deshabilitar eje X
    SysCtlDelay(16);
}
//#####################################################################
//----------------FUNCION PARA habilitar driver de eje Y --------------
//#####################################################################
void habilitar_y(){
    GPIOPinWrite(GPIO_PORTD_BASE, ENABLED_Y, 0); //deshabilitar eje Y
    SysCtlDelay(16);
}
//#####################################################################
//----------------FUNCION PARA habilitar driver de eje Z --------------
//#####################################################################
void habilitar_z(){
    GPIOPinWrite(GPIO_PORTE_BASE, ENABLED_Z, 0); //deshabilitar eje Z
    SysCtlDelay(16);
}
//#####################################################################
//----------------FUNCIONES PARA DESPLAZAR EJE X ----------------------
//#####################################################################
void avanzar_x(){
    GPIOPinWrite(GPIO_PORTD_BASE, DIR_X, DIR_X); // direccion a END
    SysCtlDelay(16);

    //for(cnt=0;cnt<5;cnt++){ //0.1mm
    GPIOPinWrite(GPIO_PORTA_BASE, PUL_X, PUL_X);//generar paso
    SysCtlDelay(5000);
    GPIOPinWrite(GPIO_PORTA_BASE, PUL_X, 0);//generar paso
    //SysCtlDelay(5000);
    //}
}

void retroceder_x(){
    GPIOPinWrite(GPIO_PORTD_BASE, DIR_X, 0); // direccion a HOME
    SysCtlDelay(16);

    //for(cnt=0;cnt<5;cnt++){ //0.1mm
    GPIOPinWrite(GPIO_PORTA_BASE, PUL_X, PUL_X);//generar paso
    SysCtlDelay(5000);
    GPIOPinWrite(GPIO_PORTA_BASE, PUL_X, 0);//generar paso
    //SysCtlDelay(5000);
    //}
}

void home_x(){
    habilitar_x();
```

```
    do{
        actualizar_limits();
        retroceder_x();
        act_x--;
        SysCtlDelay(5000);
    }while(LSX_HOME_state != 1 && status != 20);
    if(LSX_HOME_state == 1)
        act_x=0;
    SysCtlDelay(10000000);
    //deshabilitar_todos();
}

void end_x(){
    habilitar_x();
    do{
        actualizar_limits();
        avanzar_x();
        act_x++;
        SysCtlDelay(5000);
    }while(LSX_END_state != 1 && status != 20);
    SysCtlDelay(10000000);
    //deshabilitar_todos();
}

void cal_x(){
    end_x();
    area_x=0;
    SysCtlDelay(10000000);
    while(LSX_HOME_state != 1 && status != 20){
        retroceder_x();
        area_x++;
        act_x--;
        SysCtlDelay(5000);
        actualizar_limits();
    }
    if(LSX_HOME_state == 1)
        act_x=0;
    SysCtlDelay(10000000);
}
//#####################################################################
//----------------FUNCIONES PARA DESPLAZAR EJE Y --------------------
//#####################################################################
void avanzar_y(){
    GPIOPinWrite(GPIO_PORTD_BASE, DIR_Y, DIR_Y); // direccion a END
    SysCtlDelay(16);

    //for(cnt=0;cnt<5;cnt++){//0.1mm
    GPIOPinWrite(GPIO_PORTD_BASE, PUL_Y, PUL_Y);
    SysCtlDelay(4100);
    GPIOPinWrite(GPIO_PORTD_BASE, PUL_Y, 0);
    //SysCtlDelay(4100);
    //}
}

void retroceder_y(){
    GPIOPinWrite(GPIO_PORTD_BASE, DIR_Y, 0); // direccion a HOME
    SysCtlDelay(16);

    //for(cnt=0;cnt<5;cnt++){//0.1mm
    GPIOPinWrite(GPIO_PORTD_BASE, PUL_Y, PUL_Y);
    SysCtlDelay(4100);
    GPIOPinWrite(GPIO_PORTD_BASE, PUL_Y, 0);
    //SysCtlDelay(4100);
    //}
}
```

```
void home_y(){
    habilitar_y();
    do{
        actualizar_limits();
        retroceder_y();
        SysCtlDelay(4100);
        act_y--;
    }while(LSY_HOME_state != 1 && status != 20);
    if(LSY_HOME_state == 1)
            act_y=0;
    SysCtlDelay(10000000);
    //deshabilitar_todos();
}

void end_y(){
    habilitar_y();
    do{
        actualizar_limits();
        avanzar_y();
        SysCtlDelay(4100);
        act_y++;
    }while(LSY_END_state != 1 && status != 20);
    SysCtlDelay(10000000);
    //deshabilitar_todos();
}

void cal_y(){
    end_y();
    area_y=0;
    SysCtlDelay(10000000);
    while(LSY_HOME_state != 1 && status != 20){
        retroceder_y();
        area_y++;
        act_y--;
        SysCtlDelay(4100);
        actualizar_limits();
    }
    if(LSY_HOME_state == 1)
        act_y=0;
    SysCtlDelay(10000000);
}

//#################################################################
//----------------FUNCIONES PARA DESPLAZAR EJE Z -------------------
//#################################################################
void avanzar_z(){
    GPIOPinWrite(GPIO_PORTE_BASE, DIR_Z, DIR_Z); // direccion a END
    SysCtlDelay(16);
    GPIOPinWrite(GPIO_PORTE_BASE, PUL_Z, PUL_Z);//generar paso
    SysCtlDelay(4100);
    GPIOPinWrite(GPIO_PORTE_BASE, PUL_Z, 0);//generar paso
}

void retroceder_z(){
    GPIOPinWrite(GPIO_PORTE_BASE, DIR_Z, 0); // direccion a HOME
    SysCtlDelay(16);

    GPIOPinWrite(GPIO_PORTE_BASE, PUL_Z, PUL_Z);//generar paso
    SysCtlDelay(4100); //1us
    GPIOPinWrite(GPIO_PORTE_BASE, PUL_Z, 0);//generar paso
}

void home_z(){
    habilitar_z();
    do{
        actualizar_limits();
```

```
        retroceder_z();
        act_z--;
        SysCtlDelay(4100);
    }while(LSZ_HOME_state != 1 && status != 20);
    if(LSZ_HOME_state == 1)
        act_z=0;
    SysCtlDelay(10000000);
}

void end_z(){
    habilitar_z();
    do{
        actualizar_limits();
        avanzar_z();
        act_z++;
        SysCtlDelay(4100);
    }while(LSZ_END_state != 1 && status != 20);
    SysCtlDelay(10000000);
}

void cal_z(){
    end_z();
    area_z=0;
    SysCtlDelay(10000000);
    while(LSZ_HOME_state != 1 && status != 20){
        retroceder_z();
        SysCtlDelay(4100);
        area_z++;
        act_z--;
        actualizar_limits();
    }
    if(LSZ_HOME_state == 1)
        act_z=0;
    SysCtlDelay(10000000);
}
//#####################################################################
//----------------FUNCIONES para enviar cadena por UART ------------
//#####################################################################

void UARTSend(const char *charBuffer, char charCount)
{
    while(charCount--)
    {
        UARTCharPut(UART0_BASE, *charBuffer++);
    }
}

void actualizar_data_out(int var_x, int var_y, int var_z){

    actualizar_limits();

    data_out[20] = (var_z %10) + 48;
    data_out[19] = (var_z %100)/10 + 48;
    data_out[18] = (var_z %1000)/100 + 48;
    data_out[17] = (var_z %10000)/1000 + 48;
    data_out[16] = (var_z /10000) + 48;

    data_out[15] = (var_y %10) + 48;
    data_out[14] = (var_y %100)/10 + 48;
    data_out[13] = (var_y %1000)/100 + 48;
    data_out[12] = (var_y %10000)/1000 + 48;
    data_out[11] = (var_y /10000) + 48;

    data_out[10] = (var_x %10) + 48;
    data_out[9]  = (var_x %100)/10 + 48;
    data_out[8]  = (var_x %1000)/100 + 48;
```

```
    data_out[7]  = (var_x %10000)/1000 + 48;
    data_out[6]  = (var_x /10000) + 48;

    data_out[5]  = (LIMIT_SWITCHS_state %10) + 48;
    data_out[4]  = (LIMIT_SWITCHS_state /10) + 48;

    data_out[3]  = status + 48;
    data_out[2]  = capturar + 48;

}
//############################################################
```