# INSTITUTO TECNOLÓGICO SUPERIOR DE TEZIUTLÁN

## Tesis

"Diseño de algoritmo de control para robot cartesiano para la impresión de sensores de carbono"


PRESENTA:

**Irwin Saidh Jiménez Serratos**

CON NÚMERO DE CONTROL

**17TE0594**


PARA OBTENER EL TÍTULO DE:

**INGENIERA MECATRONICA**

CLAVE DEL PROGRAMA ACADÉMICO

**IMCT-2010-229**

DIRECTOR (A) DE TESIS:

**M.S.C. ALFREDO CARRASCO ARAOZ**

"La Juventud de hoy, Tecnología del Mañana"

TEZIUTLÁN, PUEBLA, ABRIL 2022

# PRELIMINARES

# Agradecimientos


A MIS PADRES.

*Por darme la gran oportunidad*
*De logar uno de mis más grandes*
*Sueños que es el poder continuar*
*Mis estudios.*

A MI FAMILIA.

*Al ser uno de los principales impulsos*
*para poder seguir adelante y poder*
*cumplir con mis sueños de ser*
*un profesionista.*

A MIS COMPAÑEROS.

*Por apoyarnos mutuamente*
*A través de este viaje de conocimientos,*
*Principalmente a mi mejor amiga por su*
*Apoyo incondicional en los momentos mas*
*Difíciles.*

AL INSTITUTO TECNOLÓGICO SUPERIOR DE TEZIUTLÁN.

*Le agradezco al instituto y mis*
*profesores por ayudarme a pasar*
*por la gran carrera de Ingeniera*
*en Mecatrónica.*

# Resumen

El uso de robot cartesianos en el sector de la ingeniería electrónica es de suma importancia, con ello se puede ahorrar procesos de diseño, manufactura y ensamblaje de los circuitos electrónicos.

El presente documento muestra al lector, como se realizó el trabajo de algoritmo, diseño de interfaz y programación para el control de un sistema de un robot cartesiano de impresión de circuitos electrónicos para sensores de carbono, Su principal propósito es entregar al usuario final un entorno de trabajo amigable donde este solo tenga que dibujar, colocar una imagen o un archivo Gerber para el grabado del circuito electrónico, sin tener que tocar los parámetros de control.

Para la realización de programa de control se utilizará principalmente dos softwares, Pycharm y Arduino, los cuales tienen la funcion de ser utilizados para la creación de la interfaz gráfica y el control del robot cartesiano respectivamente. En este documento se detallará todo el proceso de realización de manera ordenada, donde se pueden obtener los apoyos de imágenes, y principalmente del código fuente del sistema de control.

Obteniendo como resultado la implementación de la interfaz gráfica con funciones como lo son cargar los archivos Gerber, imagen png, la conexión con la tarjeta de control de Arduino, el diseño de dibujo, y el desarrollo por coordenadas. Para el control del robot, se establen los parámetros de control para que funcione con solo utilizar la interfaz, sin tener que manejar la configuración del robot.

En conclusión, este proyecto nos permitirá facilitar la realización de circuitos electrónicos para los sensores de carbono, donde solo se tenga que cargar la imagen del circuito o dibujar lo mencionado, ahorrando en tiempo de producción y con mayor eficiencia a la hora de fabricarlos.

# Introducción

Un robot cartesiano es un sistema que consta de 3 ejes, XYZ. Teniendo la capacidad de desplazarse en línea recta en dichas dimensiones. Permitiendo ser increíblemente fiable y preciso, y en este caso, permitiendo realizar en la placa una grabación precisa para el sensor de carbono.

El presente documento tiene como objetivo de informar al lector de los conocimientos tanto teóricos como prácticos para la realización del sistema de control para un robot cartesiano del tipo portal de tres grados de libertad. Esto se realizará en 3 pasos. La creación de una interfaz gráfica amigable para el usuario, el control del robot mediante el uso del Gcode (Código G), y el algoritmo que controla al sistema cartesiano mediante una tarjeta Arduino.

Una interfaz gráfica de usuario (GUI, por sus siglas en inglés), es una interfaz que se utiliza para controlar dispositivos, teniendo como elementos gráficos, como lo son iconos, menús e imágenes para facilitar el manejo del usuario humano. De hecho, casi todos los programas para usuarios finales de hoy en día vienen con esta capacidad.

Tomando en cuenta las características de un código Gerber, el cual contiene la información necesaria para realizar un diagrama en 3 dimensiones, el cual tiene ficheros como: las coordenadas, velocidades, diámetros, círculos, arcos, ángulos y códigos especiales como: inicio, final, cambio de herramientas, etc.

Arduino es una tarjeta de electrónica para la creación de control mediante el uso de hardware y software de uso libre, el cual tiene las características de ser libre, flexible y fácil de utilizar para el desarrollo de dichos sistemas.

# Índice General

# CAPÍTULO I

# GENERALIDADES DEL PROYECTO

## 1.1. Descripción de la Institución

El Instituto Tecnológico Superior de Teziutlán es una institución de educación universitaria superior tecnológica ubicada en la ciudad de Teziutlán, municipio correspondiente al estado de Puebla. Como todas las instituciones de educación tecnológica en México, el Instituto Tecnológico Superior de Teziutlán se encuentra regulado por el organismo nacional conocido como Tecnológico Nacional de México.

### 1.1.1. Historia

En 1993, el Gobernador del Estado, Manuel Bartlett Díaz, escuchando la petición popular y la intervención de funcionarios públicos y empresarios interesados, gestionó ante la Secretaría de Educación Pública, dirigida por Ernesto Zedillo Ponce de León, la creación de una Institución de Educación Superior Tecnológica, acción que se vería concretada el 8 de noviembre de 1994 con la publicación del Decreto del Congreso del Estado que expide la ley que crea "Instituto Tecnológico Superior de Teziutlán", como Organismo Público Descentralizado del Gobierno del Estado, con personalidad jurídica y patrimonio propio.

El primer día del mes de septiembre de 1993 inició actividades el Instituto ofreciendo las carreras de Ingeniería Industrial y Licenciatura en Administración, siendo el primer Tecnológico Descentralizado del Estado de Puebla, junto con su similar de la Sierra Norte, designándose como primer director general a José Emilio Guillermo Ortega Balbuena. Las primeras actividades académicas se desarrollaron en el "Centro de Bachillerato Tecnológico, Industrial y de Servicios No. 44", el cual resultó insuficiente ante la aceptación de los estudiantes; por lo que apenas un semestre después el Instituto se trasladó a una granja avícola y la casa anexa.

Mientras tanto, como resultado de la donación de Jorge Barrón Levet, en ese momento Diputado Local, y de las gestiones de éste y de su hermano Samuel Barrón Levet, se formalizó la compra de 12 hectáreas de terreno a la Compañía

Minera Autlán. Ese terreno está ubicado a un costado de la antigua mina de cobre que hace 200 años había dado pie al desarrollo de la región, y que actualmente renace con la construcción de una planta hidroeléctrica.

Para el 30 de agosto de 2018, toma el cargo de la Dirección General la Mtra. Arminda Juárez Arroyo, como consecuencia de lo anterior, y con la finalidad de hacer congruente el desarrollo, integral del Instituto Tecnológico Superior de Teziutlán dentro del proceso educativo, se generó su estructura orgánica que condujo a la expedición de su Reglamento interior.

El Instituto Tecnológico Superior de Teziutlán, atento a las demandas de la sociedad, y a los principios de la Ley de Educación del Estado de Puebla, se consolida como una Institución cuyo objetivo es lograr una educación de calidad, moderna y eficaz, orientada al servicio, acercándola a las necesidades e intereses de la población, que promueva el uso transparente y eficiente de los recursos humanos, materiales y financieros de que disponga, y que cumpla puntualmente con sus programas de trabajo.

### 1.1.2. Misión y visión

Misión: El instituto Tecnológico Superior de Teziutlán tienen como Misión, formar Profesionales que se constituyan en agentes de cambio y promuevan el desarrollo integral de la sociedad, mediante la implementación de procesos académicos de calidad.

Visión: Llegar a ser la Institución de Educación Superior Tecnológica más reconocida en el Estado de Puebla, que ofrezca un proceso de Enseñanza – Aprendizaje certificado, comprometido con la excelencia académica y la formación integral del Alumno, contribuyendo al desarrollo sustentable, económico, político y social de nuestro Estado.

### 1.1.3. Área del trabajo del estudiante

Residente dentro del Instituto Tecnológico Superior de Teziutlán dentro de la carrera Ingeniería Mecatrónica bajo la asesoría del Ingeniero Alfredo Carracos Araoz, con el fin de desarrollo un proyecto definido, el cual lleva el nombre de "Diseño de algoritmo de control para robot cartesiano para la impresión de sensores de carbono", asignado bajo instancias académicas.

## 1.2. Planteamiento del problema

En la actualidad los distintos programas asignados para el control de los robots cartesianos cuentan con una interfaz de usuario de alto nivel de entendimiento, por lo que, un usuario que cuenta con poco o nulo conocimiento sobre el tema es un gran reto de aprendizaje.

El objetivo de este proyecto es desarrollar un control para un robot cartesiano, donde el usuario final solo tenga que dibujar, cargar un archivo png o Gerber de un circuito eléctrico, mediante una interfaz gráfica que contenga lo elemental para el control de dicho instrumento, sin la necesidad de que el usuario ingrese los parámetros de control como lo son los valores de velocidad, los pasos de motor, las especificaciones del tamaño de pista del circuito, facilitando la tarea del grabado del circuito.

## 1.3. Preguntas de Investigación

¿Qué características debe de tener el programa de control del robot cartesiano para ser lo más amigable con el usuario?

¿El uso de Pycharm es correcto para este proyecto?

¿La tarjeta Arduino será un buen instrumento de control para el robot cartesiano?

¿Como se puede mejorar el uso de una aplicación para el manejo de un robot cartesiano para usuarios no experimentados?

## 1.4. Objetivos

### 1.4.1. Objetivo General

Elaborar el diseño un algoritmo de control en Python con la finalidad de realizar el manejo de un robot cartesiano de 3 dimensiones, con la ayuda de una interfaz gráfica que permita al usuario un manejo más sencillo para la impresión de circuitos electrónicos de sensores de carbono.

### 1.4.2. Objetivos Específicos

- Comprender el control que debe de realizar el robot cartesiano mediante el análisis de previas investigaciones.
- Desarrollar la interfaz Gráfica de usuario, así como detallar sus funciones.
- Implementar el algoritmo en el prototipo del robot cartesiano, y corroborar el funcionamiento adecuado.
- Obtener conclusiones respecto al mejoramiento del control.

## 1.5. Justificación

La realización del presente trabajo se justifica, porque el uso de la interfaz gráfica permite al usuario manejar el control del robot cartesiano mediante el uso de botones, imágenes o código Gerber, para la creación de los circuitos electrónicos de los sensores de carbono con una mayor facilidad y mejor control de estos, sin la necesidad de establecer parámetros de configuración inicial.

# CAPÍTULO II
# MARCO TEÓRICO

## 2.1. Bases teóricas

Una vez realizado un análisis profundo sobre el tema, se logró destacar los siguientes temas:

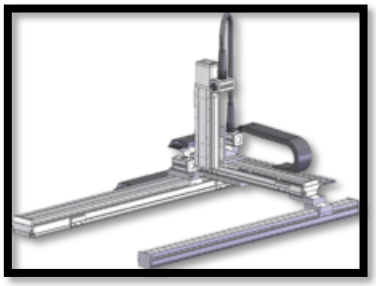
### 2.1.1. Robot Industrial

Según la International Federation of Robotics (RIA) citado por Fiestas (2017), un robot industrial es definido por el estándar ISO 8373, el cual estipula lo siguiente: "Manipulador multifuncional reprogramable y controlado automáticamente que cuenta con tres o más ejes, los cuales pueden ser fijos o movibles y son orientados a aplicaciones de automatización industrial".

Así mismo Barrientos et. al., (2007) expone que la Asociación Francesa de Normalización (AFNOR) lo define de la siguiente manera: "Manipulador automático servocontrolado, reprogramable, polivalente, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables, para la ejecución de tareas variadas. Normalmente tiene la forma de uno o varios brazos terminados en muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Normalmente su uso es el de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material." En la figura 1 se puede apreciar una representación de un robot cartesiano.

Figura 1

*Robot Industrial, del tipo cartesiano de 3 eje XYZ.*



*Fuente:* Obtenido de (Larraioz, 2020)

### 2.1.1.1. Antecedentes de los robots cartesianos

La T.M. Knasel clasifica a los robots en cinco generaciones las cuales inician con la primera generación en 1982 con el nombre *Pick and Place*, que tenían un control de final de carrera, que fue implementado en el servicio de las maquinas industriales. La segunda generación se caracterizó por contar con controles definidos como el servocontrol, un programa con parámetros y de trayectoria continua y capacidad de desplazarse sobre una vía. Estos fueron empleados en trabajos de soldadura y pintura; y aparece en el mercado en 1984.

José Luis López Segovia (2007) escribio que hacía los años de 1950 se dieron los primeros pasos para la innovación de robots con la construcción de teleoperadores o sistemas dirigidos remotamente y surgen los manipuladores programables con control. La innovación de los primeros robots industriales, con ellos aparece la primera generación de robot hacia el año de 1960, los cuales eran capaces de repetir una secuencia independientemente de que variará el entorno exterior, son los robots manipuladores programables y servo controlados.

La segunda generación de robots se caracteriza por la capacidad de interacción con el entorno de trabajo. Tienen la capacidad de realizar diferentes operaciones como procesamiento de la información captada por sensores de visión y tacto y tomar alguna decisión de ejecución, son los robots del presente.

Los robots del futuro, los cuales se están diseñando y construyendo en la actualidad, con capacidad de adaptarse al entorno, generar sus propios planes de acción, interaccionar con el entorno, además contará con un sistema de aprendizaje que le permite superar situaciones imprevistas, es el denominado robots de la tercera generación.

La cuarta generación empieza desde el año 2000. Es montado sobre ruedas o con piernas artificiales, posee sensores inteligentes, lleva como nombre móvil y se emplea en la industria de la construcción y en algunos procesos de mantenimiento en las empresas.

Los innovadores pretenden mostrar al mundo la quinta generación de robots, los cuales tendrán controladores basados en inteligencia artificial, se conocerá como un androide saltarín, el cual será empleado en la industria militar.

### 2.1.1.2. Morfología del robot

Para Rubén Darío Godoy Hernández (2007) 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 está montado sobre un riel y el brazo es el encargado de realizar las tareas ordenadas.

### 2.1.1.3. Configuración cartesiana

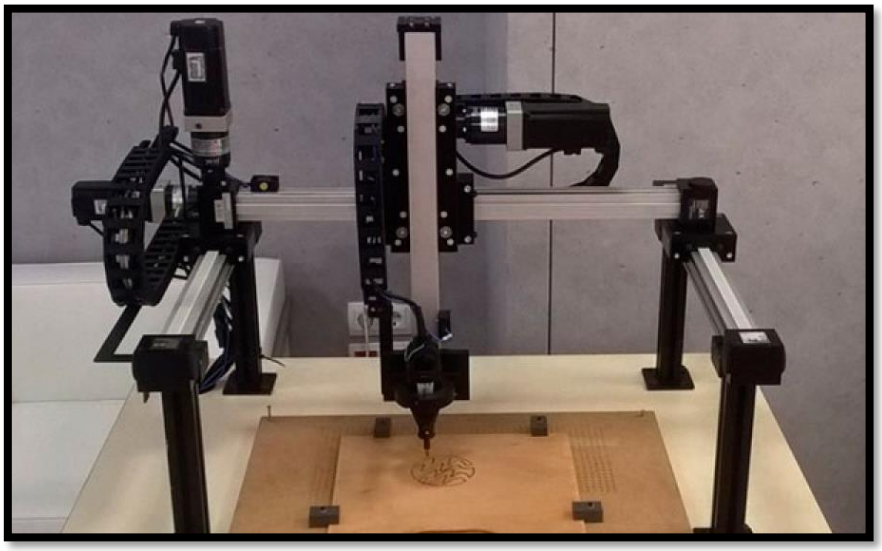
En el caso de la configuración cartesiana (PPP) que consta de una serie de eslabones prismáticos existen varias configuraciones o combinaciones en base a las necesidades de aplicación.

#### 2.1.1.3.1 Robot Portal

NÁJERA, (2016) señala que "Esta combinación esta sobre el área de trabajo para ahorrar espacio y está diseñado para transportar cargas en trayectos largos" (p. 9). Para ejemplificar mejor el modelo del Robot Portal se muestra en la siguiente figura.

Figura 2

*Robot Cartesiano del tipo Portal.*



*Fuente:* Obtenido de ( infoPLC, 2016)

## 2.1.2. Control Numérico Por Computadora

Rodríguez (2017) Es una herramienta tecnológica que sirve para gobernar el funcionamiento mediante una serie de códigos alfanuméricos, básicamente el funcionamiento de una máquina herramienta es guiar, mediante un sistema de coordenadas (x, y, z), con este método se consigue una mayor precisión y menor desgaste de material comparado con los métodos y herramientas artesanales conocidas. El CNC ha aportado mejor control en el diseño y fabricación de activos y pueden ser usadas en procesos sencillos como llevar una herramienta a una posición lineal, como también el fresado de una pieza en tres dimensiones que por

métodos convencionales resultaría demasiado costoso y de suma dificultad su realización. El tablero de una CNC se puede ver como el de la figura 3.

Figura 3

*Control Numérico por Computadora industrial.*


*Fuente:* Obtenida de (Metal y Mecanica, 2013)

### 2.1.2.1. Archivo Gerber

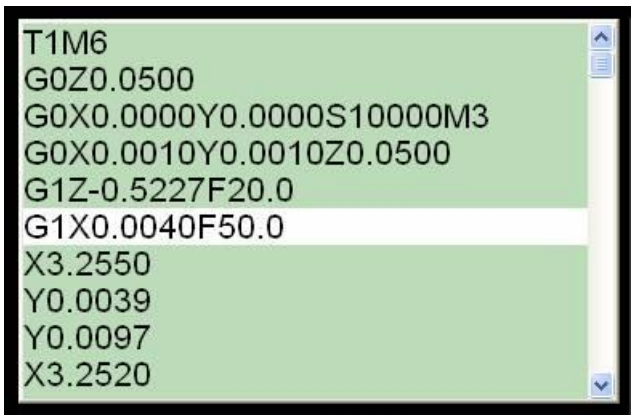
Se define como "Es utilizado en la producción de circuitos a través del maquinado, creado por (Gerber Systems Corporation), desde el año 1970 creo diversas impresoras de fotograbado de control numérico, se formalizo en 1980 por (Electronic Industries Association) y recibió el archivo de trazado EIA RS-274-D, dentro de diversos cambios hoy manejamos el archivo de trazado RS-274 X, los cuales son archivos de código ASCII que manejan coordenadas e instrucciones que interpreta el circuito a fabricar." (Tabernier, p. 58 2015, citado por Castillo 2019).

### 2.1.2.2. Código G

De acuerdo con lo expuesto "se ha generado una lista de funciones de los movimientos de la máquina entre ellos movimientos rápidos, avances, avances radiales, pausas y ciclos. Estos contienen variables (direcciones) definidas por el programador para cada función específica, maneja un lenguaje de programación vectorial básicamente se escriben segmentos de recta y arcos de circunferencia la programación se efectúa mediante un lenguaje de bajo nivel, considerando que la norma (ISO 6983-1: 2009) es el estándar con el cual se rigen las máquinas CNC y por ende el código G." (Hood y Floyd, 2009, p. 206, citado por Castillo 2019) Un ejemplo de esto se muestra en la figura 4.

Figura 4

*Ejemplo de Código G.*




*Fuente:* Obtenido de (Munoz, 2014)

### 2.1.2.3. Python

Para Van Roussum & Drake Jr (2017) Python es un lenguaje de programación poderoso y de fácil uso. Cuenta con estructuras de datos eficientes y de alto nivel, además de un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su "tipado" dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

#### 2.1.2.3.1 Transmisión Serial

De acuerdo con los autores Berrio, Arcos, Zuluaga, & Corredor (2015) Los datos entre el sistema físico y virtual deben ser transmitidos usando una tarjeta de envío de datos, encargado de enviar las señales a los controladores de los motores paso a paso, luego enviar los "pasos" respectivos a cada motor, ya sea el del eje x (2 motores), el motor del eje y o del eje z.

#### 2.1.2.3.2. Interfaz Gráfica

Para Hernández Sánchez (2019) Es una tecnología de protocolos de alto nivel, que se utiliza para poder desarrollar aplicaciones software compatibles entre sistemas distribuidos orientados al control y monitorización de procesos. Con estos protocolos se logra una comunicación abierta, independiente y estandarizada.

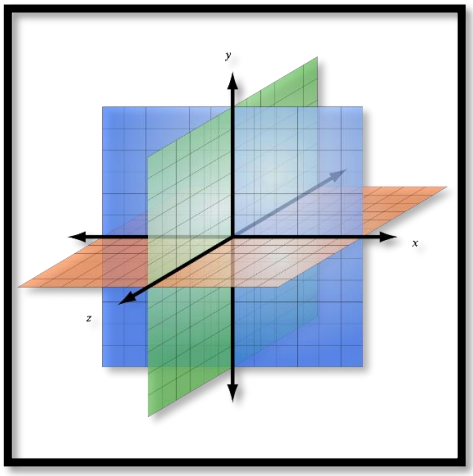
Se puede conseguir una integración total, ya que facilita que equipos de diferentes fabricantes puedan trabajar sin que se presente ningún tipo de conflicto entre ellos y facilitando que cada empresa pueda realizar su propia aplicación de control y medición adecuada a sus necesidades.

### 2.1.3. Sistemas de coordenadas

De acuerdo con Pasquel Castillo (2019) Los sistemas de coordenadas son en pulgadas o métrico, los cuales pueden ser absolutos o incrementales y son los planteados en el plano cartesiano. El cual contiene ejes de coordenadas o ejes de trabajo, en este caso formado por tres rectas perpendiculares entre sí (X, Y, Z) que se intersecan en el origen (0,0,0). Los comandos en X e Y se usan para el posicionamiento de la bancada o mesa de trabajo y el comando Z para el posicionamiento de la herramienta o Spindle. La representación de un sistema de coordenadas se puede apreciar en la imagen 5.

Figura 5

*Sistema de coordenadas.*




*Fuente:* Obtenido de (I, 2007)

### 2.1.3.1. Sistemas de coordenadas absolutas

De acuerdo con Pasquel Castillo (2019) El punto cero es el punto inicial de trabajo o punto inicial de referencia en cada diseño, lo que quiere decir que cada vez que se quiera mover el Spindle o la bancada partiremos desde al punto cero absolutos y los colocaremos en el punto deseado, pero siempre partiendo del punto cero iniciales, dicho esto el código para que la CNC nos entienda para las coordenadas absolutas es el G90.

### 2.1.3.2. Sistemas de coordenadas incrementales

De acuerdo con Pasquel Castillo (2019) Se programa la distancia que se debe desplazar el Spindle o la bancada respecto al último punto conseguido, es decir el punto cero se va modificando de acuerdo con el avance del Spindle denominándose una trayectoria en cadena la cual depende de cada punto anterior al movimiento realizado. El código en la CNC para las coordenadas incrementales es el G91.

## 2.1.4. Estructura mecánica del robot industrial

Para los autores Godoy Hernández & Rodríguez Quintero (2007). La estructura mecánica se refiere al tipo de articulación que posee el robot y el tipo de movimiento que estas generan. Los movimientos de cada una de las articulaciones asociado a los movimientos del brazo o cuerpo del robot se denominan grado de libertad. 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.

### 2.1.4.1. Movimiento lineal a lo largo de los tres ejes

"Para realizar el movimiento lineal se buscó un mecanismo capaz de lograr una buena precisión con la fuerza necesaria para poder soportar los demás ejes." (Val, 2009, p. 3)

## 2.1.5. Componentes Eléctricos del robot industrial

Para Sanchez & Rodriguez (2008) Los componentes electrónicos son los dispositivos que forman los circuitos y que hacen que aparatos que utilizamos en nuestro día a día funcionen, como móviles, televisores o secadores. Vienen encapsulados en cerámica, metal o plástico. Normalmente tienen dos o más terminales o patillas metálicas. Vivimos rodeados de ellos y solemos ser conscientes de cómo facilitan nuestra vida a corto plazo, pero sabemos poco o nada sobre cómo funcionan, de dónde proceden o cómo afecta su uso y su consumo al medioambiente.

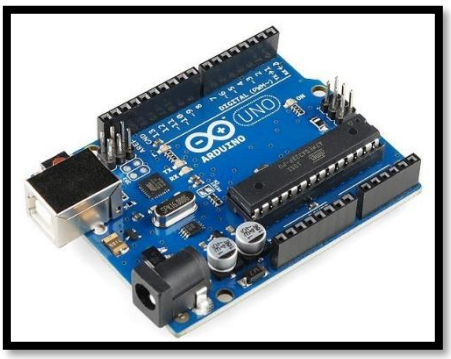
### 2.1.5.1. Tarjeta controladora

De acuerdo con el sitio web TechLib (2021). La tarjeta controladora, o simplemente "controlador", es una pieza de hardware que actúa como interfaz entre la placa base y los otros componentes de la computadora. Por ejemplo, los discos duros, unidades ópticas, impresoras, teclados y ratones requieren

controladores para funcionar. La mayoría de las computadoras tienen todos los controladores necesarios integrados en la placa base como chips, no como tarjetas de tamaño completo. Sin embargo, si agrega componentes adicionales como un disco duro SCSI, es posible que también necesite agregar una tarjeta controladora. Las tarjetas controladoras generalmente se instalan en una de las ranuras PCI de la computadora. La tarjeta de Arduino (figura 6) es un claro ejemplo de esto.

Figura 6

*Arduino uno R3.*



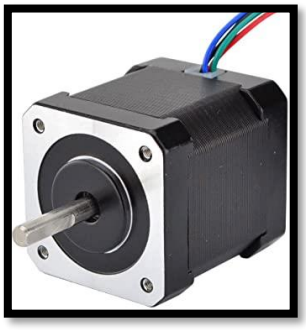

*Fuente:* Obtenida de (Arduino Inc., 2015)

### 2.1.5.2. Motores Eléctricos

Para los autores Tobón Restrepo & Agudelo García (2016). Es un dispositivo electromecánico que convierte una serie de pulsos eléctricos en desplazamientos angulares, dando como resultado un movimiento de un paso por cada pulso digital aplicado. Las bobinas del motor están en el estator y pueden ser de imán permanente o reluctancia variable. A medida que se activa los bobinados del motor en un orden predefinido permite que fluya una corriente a través de ellos con lo cual el estator se magnetiza y se generan polos electromagnéticos los cuales dan como resultado la propulsión del motor. El motor que se pretende controlar son los Nema 17 los cuales se muestran en la imagen 7.

Figura 7

*Motor Nema 17.*



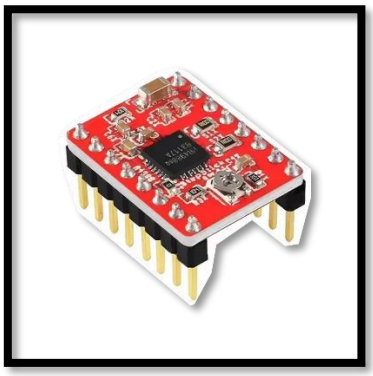
*Fuente:* Obtenido de (Leyenda Oriental, 2018)

### 2.1.5.3. Controladores (Driver)

De acuerdo con Pasquel Castillo (2019) Permite controlar altos voltajes e intensidades que requieren los motores a pasos y limitar la corriente que circula por el motor. La intensidad máxima de trabajo es de 2 A y el rango de voltaje de trabajo, 8 a 35 V, el diseño de este controlador es para motores de paso bipolares de paso completo, medio, cuarto, octavo y modos de dieciséis pasos. Esta es la definición de los controladores A4988 la cual se representa en la imagen 8.

Figura 8

*Controlador A4988.*



*Fuente:* Obtenido de (MV electronica, 2021)

### 2.1.5.4. Fuente de Poder

"Las fuentes de alimentación son equipos cuya principal función es transformar la energía. Esto con el fin de alimentar los componentes previamente mencionados y así poder realizar su función correctamente." Gastello (2020)

## 2.1.6. Fuentes de Información

Las asignaturas de diseño mecatrónico se centran en el diseño funcional de sistemas autónomos e inteligentes, aplicando diferentes disciplinas tecnológicas (dinámica mecánica, la electrónica de baja y alta potencia, electromecánica, control de movimiento, la óptica, metrología y procesamiento de señales) en un entorno bien equilibrado fortaleciéndose con la realización de prototipos. El presente artículo expone el proceso de diseño, construcción y puesta en marcha de un robot cartesiano con 3 grados de libertad y la programación de su respectivo software de interfaz con el usuario, este fue realizado en lenguaje Python donde las principales ventajas para su escogencia fueron su versatilidad, portabilidad y documentación, además de ser Open-Source. Se requería implementar un algoritmo para el reconocimiento de voz de tal forma que pudiera clasificar 3 palabras diferentes a través de redes neuronales artificiales.

Título: "Control Cartesiano de un Asistente robótico para cirugía Laparoscópica"

Autor(es): V. F. Muñoz, I. García Morales, C. Pérez del Pulgar, J. M. Gómez de Gabriel, J. J. Fernández Lozano, A. García Cerezo, C. Vara, R. Toscano

Institución: Revista Iberoamericana de Automática e Informática Industrial

Año: 2006

Este articulo estudia la planificación de movimientos en asistentes robóticos para el manejo de la cámara laparoscópica en cirugía mínimamente invasiva. Se centra en

el problema del control cartesiano de robots dotados con muñecas pasivas. A partir de un análisis cinemático de la tarea, se propone un esquema de control cartesiano y adaptativo que planifica en línea la trayectoria de la cámara y asegura que no se saturan los actuadores durante el desplazamiento. La estrategia propuesta se ha implantad en el asistente ERM, con el que se ha realizado un ensayo clínico sobre treinta y dos pacientes. 

Título: "Diseño y modelamiento de un robot cartesiano para el posicionamiento de piezas"

Autor(es): Rubén Darío Godoy Hernández, Willy Rodríguez Quintero

Institución: Universidad de la Salle

Año: 2007

En el presente proyecto se realiza el diseño y modelamiento de un robot cartesiano para el posicionamiento de piezas en el módulo de almacenamiento del laboratorio de automatización y robótica. Se recuperan, manipulan y almacenan forma automática las piezas del CIM.

El robot consta de tres grados de libertad, cada articulación es prismática, el desplazamiento en el eje X es por medio de una correa dentada conectada a un motor paso a paso. Para el desplazamiento en el eje Z se utiliza un tornillo de bolas el cual se moverá por un motor paso a paso. Para la ubicación de las piezas en el eje y se utiliza un cilindro neumático, en el cual en su extremo final tiene una pinza paralela para sujetar el palet que viene en la cinta transportadora. Esta pinza tiene una rotación alrededor del eje z.

El robot manipula el palet por lo tanto la pinza se diseña para sujetarlo en su cilindro de sujeción.

Título: "Algoritmo para obtención de índice de desempeño, en configuraciones de robots SCARA, Cartesiano y Antropomórfico"

Autor(es): Cortes Berruecos Rene, Torres Méndez Sergio Javier, Muñoz Hernández German Ardul, Reyes Cortes Fernando

Institución: Instituto Tecnológico de Puebla

Año: 2017

Se realizará el estudio de los índices de desempeño en un robot planar vertical de dos GDL, en un robot SCARA usando solo hombro y codo en forma horizontal, y un robot cartesiano solo usando los ejes XY.

Se desarrollará un algoritmo de control para cada estructura robótica, integrando funciones que contengan los modelos matemáticos de la cinemática directa, cinemática inversa, modelo dinámico, la energía y control aplicada al robot, de esta forma crear un programa principal que nos resuelva el modelo dinámico y muestre las trayectorias y valores de errores de posiciones deseadas de cada eslabón.

A partir de la norma se verificará su índice de desempeño, dependiendo de todas las variables que intervienen en cada estructura robótica das las simulaciones se desarrollaran en el editor de Matlab para validar nuestros resultados.

Título: "Control de trayectorias de un robot cartesiano"

Autor(es): Mauricio E. Badillo Nájera

Institución: Universidad Politécnica de Tulancingo

Año: 2016

En este trabajo se aborda el diseño y construcción de un robot cartesiano experimental para la generación de trayectorias predefinidas, la ventaja de estudiar los sistemas de esta manera es que proporciona las bases de la ingeniería en control.

El desarrollo de este proyecto comienza con una introducción a los sistemas de control automático y algunas investigaciones sobre sistemas cartesianos con otras técnicas de control, posteriormente el análisis cinemático de un robot de configuración cartesiana para determinar la posición de cada uno de los grados de libertad, junto con el estudio del modelo matemático del robot incluyendo fenómenos fricción y análisis del sistema de transmisión y con ayuda del diseño asistido por computadora, con el fin de observar el comportamiento en el tiempo del modelo matemático. También se implementa el sistema electrónico y un controlador PID de posicionamiento de los eslabones y terminando con el análisis de resultados del control del controlador PID en el sistema físico.

Título: "Desarrollo de un controlador para reducir el error en las trayectorias lineales de un robot cartesiano tipo Gantry"

Autor(es): Carlos Alberto Jara Ayala, Renato del Castillo Huaccha

Institución: Universidad privada Antenor Orrego

Año: 2019

Este trabajo de investigación presenta el desarrollo de un controlador de seguimiento de trayectorias lineales para una articulación con la finalidad de reducir el error en las trayectorias lineales de un robot cartesiano tipo Gantry, mediante una simulación para el movimiento coordinado de dos articulaciones

lineales y pruebas de seguimiento de trayectorias que se realicen en un sistema de control de pruebas que comprende una articulación lineal conformado por un motor de corriente continua y un husillo. Para llevar a cabo el desarrollo del controlador, se considera abordar la elección del dispositivo programable a utilizar de acuerdo con los requerimientos de un robot cartesiano tipo Gantry, una pesquisa de técnicas aplicables para la coordinación de articulaciones, y el modelamiento matemático de la articulación lineal del sistema de control. Se realizan diversas pruebas de seguimiento de trayectorias con un control en lazo abierto, así como con el controlador desarrollado, para finalmente validar la reducción del error en las trayectorias lineales de un robot cartesiano tipo Gantry mediante gráficos comparativos.

Título: "Robot Cartesiano de tres ejes controlado por computadora"

Autor(es): Christian Guillermo Val

Institución: Pontificia Universidad Católica Argentina

Año: 2009

El objetivo principal de esta obra es explicar el proyecto, diseño y la construcción de un brazo robótico en tres ejes cartesianos controlado por computadora. Este aparato comúnmente llamado "Robot Cartesiano" puede ser empleado en la manipulación de herramientas para la fabricación de piezas diversas, de manera automática por programa o de forma manual a través de una interfaz Joystick.

Se programa por medio de rutinas bajo el lenguaje Gerber, el cual es un Standard mundial para máquinas de control numérico computarizado. El proyecto tiene importantes aplicaciones como herramienta de aprendizaje para la rama de control y es una herramienta fundamental para la fabricación de prototipos en los que se necesiten realizar piezas de manera automática y precisa.

Consiste en un sistema robótico de tres carros móviles que desplazan en forma tridimensional un elemento abrasivo o cortante. La idea de construir esta máquina surgió al principio con el objeto de hacer las perforaciones de los circuitos impresos que se utilizan para hacer plaquetas electrónicas, para el montaje de sus componentes. Se buscaba posicionar la perforadora automáticamente en cualquier punto del plano XY horizontal y bajarla en la dirección vertical del eje Z para realizar la perforación.

Título: "Diseño de una fresadora CNC para PCBs mediante4 archivos Gerber y fuente excellon"

Autor(es): Carlos Roberto Pasquel Castillo

Institución: Universidad Tecnológica Israel

Año: 2019

El proyecto trata del diseño y construcción de una fresadora CNC para PCBs, la cual a través de un microcontrolador Arduino UNO controla la velocidad y posición de los motores a pasos que accionan los ejes de la fresadora, y con el arranque de viruta después del proceso de fresado y taladrado se obtiene los circuitos PCBs utilizados hoy en día en diferentes campos como la industria, educación, investigación entre otros.

Se implementó un programa para el control PWM con el software App Inventor 2 que controla vía Bluetooth la velocidad del Spindle con la interacción de un microcontrolador Arduino NANO el cual a su vez gestiona alarmas de los finales de carrera de seguridad de límites de trabajo de la fresadora CNC que bloquean el movimiento, esto es con el fin de evitar daños en las piezas de la máquina y materiales.

Título: "Diseño y construcción de un robot cartesiano de 3 grados de libertad"

Autor(es): Julie Berrio, Edgar Arcos, Juan Zuluaga, Sergio Corredor

Institución: Universidad Autónoma del Caribe

Año: 2015

La implementación es gestionada a través de un Smartphone con sistema operativo Android con la que se activa y desactiva las etapas de control principal de la fresadora teniendo un control remoto de la máquina.

Dentro de las ventajas significativas esta la precisión y el corto tiempo de elaboración en una producción en serie, y una desventaja es el alto costo al adquirir un equipo de similares características con lo cual se obtuvo un ahorro del 40 % al construir la fresadora localmente.

La implementación tiene cuatro puntos principales los cuales se determinaron dentro del diseño previo a la construcción: mecánica, eléctrica, electrónica y software para el control.

# CAPÍTULO III
# DESARROLLO Y METODOLOGÍA

## 3.1. Procedimiento y descripción de actividades realizadas

Para poder realizar un buen trabajo de investigación se debe de tener un buen orden para la organización de las actividades a elaborar, y así tener un buen trabajó con los resultados esperados.

### 3.1.1. Cronograma de actividades

Tabla 1

*Cronograma de actividades.*

| Actividades | 1<br>SEP<br>13-18 | 2<br>SEP<br>20-25 | 3<br>S-O<br>27-02 | 4<br>OCT<br>04-09 | 5<br>OCT<br>11-16 | 6<br>OCT<br>18-23 | 7<br>OCT<br>25-30 | 8<br>NOV<br>01-06 | 9<br>NOV<br>08-13 | 10<br>NOV<br>16-20 | 11<br>NOV<br>22-27 | 12<br>N-D<br>29-04 | 13<br>ENE<br>10-15 | 14<br>ENE<br>17-22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bases teóricas, donde se establecio los mas: Robot Industrial, CNC, Sistema de , Estructura mecánica, Componentes | X | | | | | | | | | | | | | |
| | | X | | | | | | | | | | | | |
| el procedimiento y la descripción de a realizar para la elaboración del programa de obot cartesiano. | | | X | | | | | | | | | | | |
| | | | | X | | | | | | | | | | |
| del diseño del diagrama de flujo de control tesiano. | | | | | X | | | | | | | | | |
| ón de la interfaz grafica del control del robot con el lenguaje de programación de Python. | | | | | | X | | | | | | | | |
| | | | | | | | X | | | | | | | |
| | | | | | | | | X | | | | | | |
| | | | | | | | | | X | | | | | |
| del diseño del algorimto de control del robot iano utilizando el programa de arduino. | | | | | | | | | | X | | | | |
| | | | | | | | | | | | X | | | |
| | | | | | | | | | | | | X | | |
| | | | | | | | | | | | | | X | |
| ón de simulaciones y pruebas de la interfaz ca, junto el control del arduino, para el namiento del robot cartesiano. Así como es en los programas previamento realizados. | | | | | | | | | | | | | | X |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |

*Fuente: Irwin Saidh Jiménez Serrato, 2022*

*Nota.* Este se describe las actividades realizadas durante la elaboración del programa de control del robot cartesiano.

### 3.1.2. Estrategia global de solución

Las actividades generales que se estarán desarrollando se listan a continuación:

- Recolección de datos:

Toda aquella información reunida de las distintas fuentes de información que contengan alguna relación con el tema a investigar en este caso robot cartesianos, en estos se tomaron en cuenta conceptos, componentes, características y software utilizados para la elaboración de esta.

- Sistema de control:

Siendo la segunda meta, se realizará el algoritmo capaz de controlar el robot cartesiano, este contará con distintas funciones para controlar este dispositivo, tales como el uso de

- Simulación:

Al realizar simulaciones nos permitirán observar su correcto funcionamiento resultante del producto ensamblado en físico con el algoritmo controlador realizado, dando el resultado, ya sea con un correcto funcionamiento o presentado los errores de estos para su futura solución

- Documento:

Una vez que el programa de control y el mismo robot este en optimo funcionamiento, nos disponemos a realizar la documentación correspondiente, estos son el desarrollo realizado, y la construcción que obtengamos del proceso realizado.

## 3.2. Alcance y enfoque de la investigación

### 3.2.1. Alcance

En este proyecto se busca obtener un programa que pueda realizar el control de un robot cartesiano capaz de realizar impresiones para circuitos de sensores de carbono. Este contara con 3 opciones uno con control por medio de un archivo llamada código Gerber, uno utilizando una imagen del tipo PNG, y el control manual para realizar acabados. Una vez que el algoritmo sea capaz de realizar el control, así como conectarse con una tarjeta controladora he indicar a los

actuadores como realizar su trabajo. La principal restricción que se presenta en este proyecto es la faltade tiempo, teniendo en cuenta los factores que este conlleva.

### 3.2.2 Enfoque

El modelo de enfoque de este trabajo de investigación es del tipo cuantitativa, debido a que se formó la idea del control, seguido de esto nos planteamos el problema a resolver, realizamos una revisión de literatura y el desarrollo del marco teórico, desarrollamos la aplicación del problema a resolver, analizamos los datos obtenidos, y finalmente elaboramos el reporte de resultados.

## 3.3. Hipótesis

El desarrollo de un sistema de control del robot cartesiano por medio de una interfaz gráfica, la cual nos permita desarrollar con mayor sencillez el grabado de circuitos electrónicos de sensores de carbono, con la ayuda de una comunicación serial entre dicha interfaz y la tarjeta controladora de Arduino.

## 3.4. Diseño y metodología de investigación

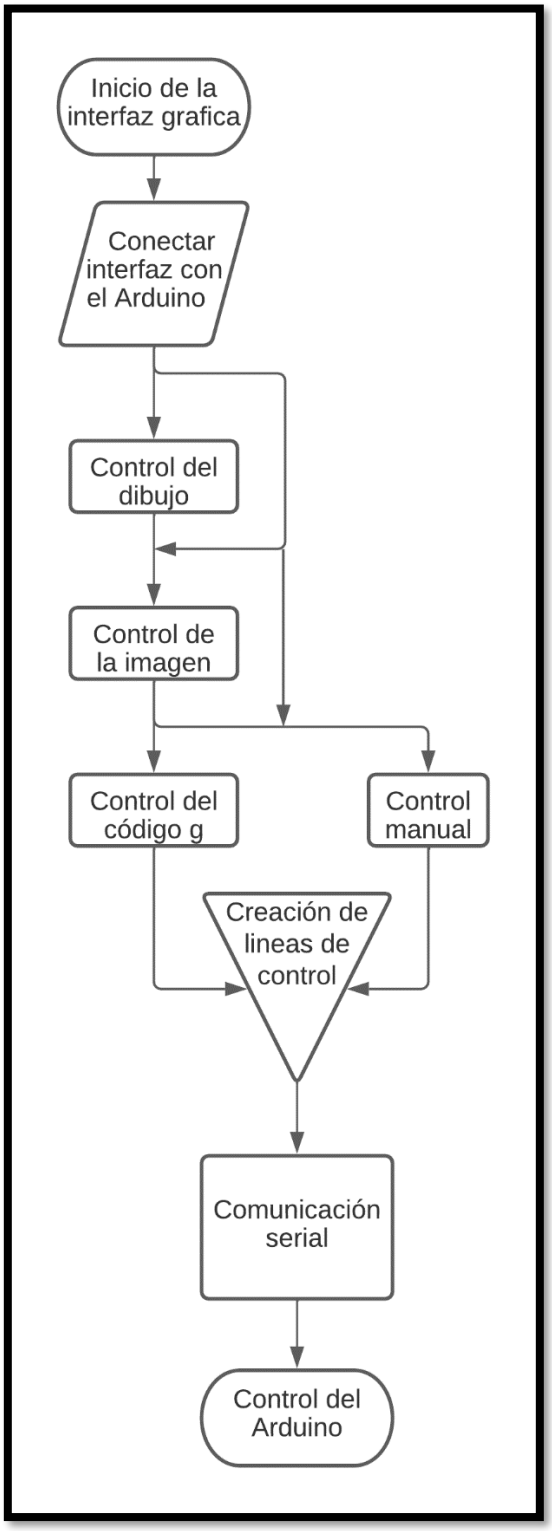
### 3.4.1. Diseño

Como bien es sabido esta es un tipo de investigación experimental, para ello se llevó a cabo una recolección de datos, en los cuales se indago las previas investigaciones donde se abarcase el uso del software de Python para controlar un sistema de un robot cartesiano.

Para ello se realizó el desarrollo del algoritmo mediante la elaboración de diagramas de flujos el cual nos permitirá realizar un mejor entendimiento de este:

Figura 9

*Diagrama de Bloque para el control del robot cartesiano.*



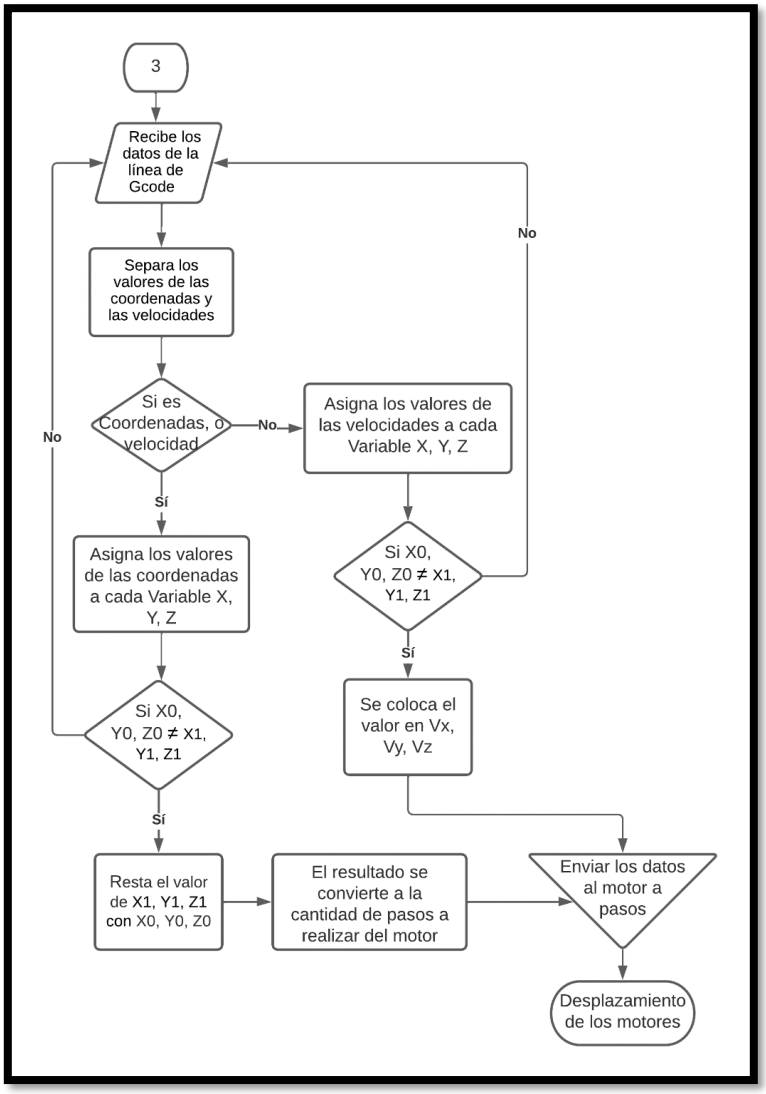

*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Para un mayor entendimiento del diagrama de bloques para la interfaz gráfica, se puede ver en el anexo 1.

Para el control del robot cartesiano se realizó el siguiente diagrama de bloques, el cual tienen la intención de explicar a grandes rasgos el funcionamiento de este, como se puede notar se realiza la comunicación entre estos.

Figura 10

*Diagrama de Bloque para el control del robot cartesiano con el apoyo del Arduino.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

## 3.4.2. Metodología

La metodología utilizada es una investigación cuantitativa, ya que nos centramos en la medición y la comprobación mediante datos numéricos, y la experimentación con el apoyo de la programación.

Para poder realizar la programación de la interfaz gráfica en Python y el control mediante Arduino, se llevó a cabo lo siguiente:

### 3.4.2.1. Interfaz gráfica

Para crear una interfaz gráfica de usuario en Python es necesario tener un lienzo en el cual desarrollar los apartados que esta contendrá, por ello se crea una ventana la cual se muestra en la figura 11.

Figura 11

*Creación de la ventana de la interfaz.*


*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Así como la creación de una barra menu la cual nos permitirá abrir los archivos como lo son el Código G y el PNG, el apartado que nos permita realizar una conexión con el puerto serial del Arduino, un botón que nos ayude a utilizar la aplicación y uno de salida de esta, mostrada en la figura 12.

Figura 12

*Barra menu de la interfaz.*


*Fuente: Irwin Saidh Jiménez Serrato, 2022*

*Nota.* Al realizar una acción en ellos se desplegará una lista con más funciones.

Para el control por Código G, tendremos una serie de botones, los cuales nos permiten cargar los archivos Gcode de la memoria, correr, parar o pausar la transferencia de datos a través del puerto serial con el Arduino. Así como, secciones de escritura, lo que permite al usuario visualizar el código G completo, así como la línea que se está transmitiendo en tiempo real. Dicha sección se encuentra en la figura 13.

Figura 13

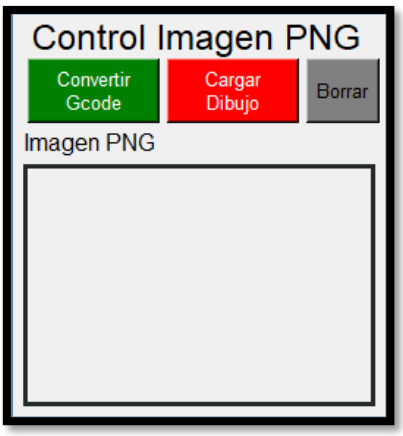
*Apartado de la interfaz gráfica de la sección del control por código G.*


*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Para el control por imagen PNG, tendremos una serie de botones, los cuales nos permiten cargar los dibujos realizados guardados en la memoria, convertir las imágenes en código G, para su posterior transferencia. Así como, sección de visualizar la imagen a convertir a código G. Se puede visualizar en la figura 14.

Figura 14

*Apartado de la interfaz gráfica de la sección de control por imagen PNG.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

*Nota.* Resultado de ejecutar los códigos mencionados con anterioridad.

Para poder realizar un dibujo dentro de la interfaz se realizó el denominado control Dibujo, el cual permite al usuario realizar el trazado del circuito para la impresión de sensores de carbono. Este apartado se encuentra en la figura 15, donde se puede visualizar distintos botones para la creación de este.

Figura 15

*Interfaz gráfica para el control por dibujo.*

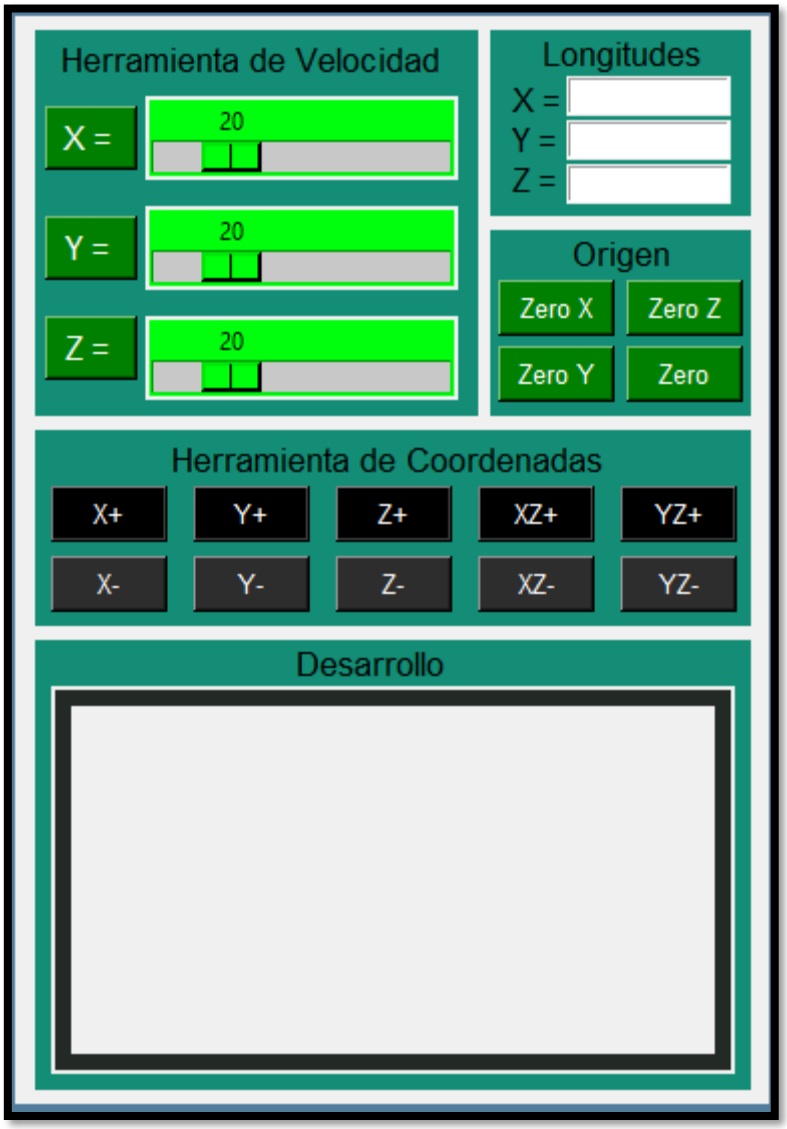

*Fuente: Irwin Saidh Jiménez Serrato, 2022*

*Nota.* Resultado de ejecutar los códigos mencionados con anterioridad.

Para poder realizar el control manual de la maquina se realizó distintos apartados en la interfaz gráfica, las cuales son del manejo de velocidad, el cambio de las coordenadas, establecer las coordenadas al origen, y una sección de desarrollo del manejo manual.

Figura 16

*Interfaz gráfica para el control manual, mediante el uso de coordenadas.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

### 3.4.2.2. Programa De Arduino

Para la realización del código para el control de la placa de Arduino. Se realizado los siguientes procedimientos:

#### 3.4.2.2.1. Definición de las constantes

Para poder hacer una definición de una constante en Arduino se hace uso de la palabra reservada de *'#define'* que nos permitiéndonos poner un nombre a la constante y el valor, en la siguiente parte se muestra las constantes con su valor a utilizar.

```
#define NUMCOMMANDS 2
#define MOTOR_STEPS 200
#define STEPS_MM 80
#define MOTOR_X_RPM 120
#define MOTOR_Y_RPM 120
#define MOTOR_Z_RPM 120
```

Donde la principal funcion que tiene la primera línea que es *'#define NUMCOMMANDS 2'* indicar el número de veces que se repetirá el código G en el grabado de la impresión.

*'#define MOTOR_STEPS 200'* Define los pasos que tiene que realizar el motor para dar una vuelta completa, lo que quiere decir que dará 200 pulsos de 1.8° dando un total de 360°.

Para poder saber cuántos milímetros avanza por el total de pasos se define la siguiente funcion *'#define STEPS_MM 80'*.

*'#define MOTOR_X_RPM 120, #define MOTOR_Y_RPM 120, #define MOTOR_Z_RPM 120'*, define el número total de vueltas que realiza el motor por minuto, el cual está configurado por 12 vueltas por minuto.

#### 3.4.2.2.2. Configuración de los Motores

Principalmente en esta sección se configura los pines en donde se coloca el Shield y así poder controlar los motores. En el siguiente código se muestra esta definición:

```
#define DIR_X 5
#define STEP_X 2
#define DIR_Y 6
#define STEP_Y 3
```

```
#define DIR_Z 7
#define STEP_Z 4
#define ENA 8
#define MICROSTEPS 16
```

Donde *'#define DIR_X 5, #define DIR_Y 6, #define DIR_Z 7'* configuramos los pines de salida para controlar la dirección de giro de los motores.

Donde *'#define STEP_X 2, #define STEP_Y 3, #define STEP_Z 4'*, se configura el pin de salida donde se manda el número de pasos a loa cuales se debe de dirigir.

*'#define MICROSTEPS 16'*, nos permite controlar el total de micro pasos a lo cuales se desarrolla el programa.

#### 3.4.2.2.3. Configuración de los drivers

Para poder controlar los drivers nos apoyaremos de 2 palabras reservadas las cuales son 'A4988' y 'MultiDriver', la cual nos permite configurar una variable en donde se engloba los pines de salida del motor, así como el número de controlares que tendremos en el Shield. En él se siguiente código se muestra dichos comandos.

```
A4988 stepperX(MOTOR_STEPS, DIR_X, STEP_X, ENA);
A4988 stepperY(MOTOR_STEPS, DIR_Y, STEP_Y, ENA);
A4988 stepperZ(MOTOR_STEPS, DIR_Z, STEP_Z, ENA);
MultiDriver controller(stepperX, stepperY, stepperZ);
```

#### 3.4.2.2.4. Comandos Código G

Estos comandos nos permiten configurar las entradas de los códigos G enviados a través del puerto Serial, y con el uso de la librería '*Gcode.h*'.

```
commandscallback commands[NUMCOMMANDS] = {{"G1",homing}, {"G0",
moviment}};
gcode Commands(NUMCOMMANDS, commands);
```

#### 3.4.2.2.5. Iniciación de las variables

En esta sección declaramos 3 variables para que nos apoyemos en ellas y poder guardar los datos enviados de las coordenadas en el código G. Se utiliza variables

de tipo doble para poder utilizar números positivos y negativos, así como decimales.

```
double X;
double Y;
double Z;
```

### 3.4.2.2.6. Función Inicio

Esta función se ejecuta solamente una vez, y es cuando se energiza por primera vez la tarjeta. La función se muestra a continuación:

```
void setup()
{
  Commands.begin();
  stepperX.begin(MOTOR_X_RPM, MICROSTEPS);
  stepperY.begin(MOTOR_Y_RPM, MICROSTEPS);
  stepperZ.begin(MOTOR_Z_RPM, MICROSTEPS);
  stepperX.setEnableActiveState(LOW);
  stepperY.setEnableActiveState(LOW);
  stepperZ.setEnableActiveState(LOW);
}
```

En donde se realiza la iniciación de los comandos establecidos anteriormente, así como los pasos de los motores, y se establece que los pasos están desactivados pero listos para utilizar.

### 3.4.2.2.7. Función de Repetición

Esta funcion se estará repitiendo constantemente, y estará comprobando sí, hay algún dato nuevo el puerto serial, y si esto se cumple lo manda a las siguientes funciones que se muestran.

```
void loop()
{
  Commands.available();
}
```

### 3.4.2.2.8. Función de Casa

Esta funcion permite dirigir al robot cartesiano al origen, o mejor conocido a las coordenadas cero.

```
void homing()
{
  // code to home machine
}
```

### 3.4.2.2.9. Función de Ir a la locación

Esta funcion nos ayuda a poder hacer el movimiento de los tres motores, para ello se utiliza el siguiente código:

```
void gotoLocation(double x,double y, double z)
{
  int stepsx = (x - X)*STEPS_MM; // Variacion de la distancia X
  int stepsy = (y - Y)*STEPS_MM; // Variacion de la distancia Y
  int stepsz = (z - Z)*STEPS_MM; //Variacion de la distancia Z
  stepperX.enable();// ENABLE MOTOR X
  stepperY.enable();
  stepperZ.enable();
  controller.move(stepsx, stepsy, stepsz); //Se envían al driver las
posiciones deseadas
  X = x; // Ultima Posición del Set
  Y = y;
  Z = z;
  stepperX.disable();// DISABLE MOTOR Y
  stepperY.disable();
  stepperZ.disable();
  Commands.comment("X:" + String(x) + "; Y:" +String(y) + "; Z:"
+String(z)); // DEBUG SERIAL
}
```

Para comenzar la función de dirigir los valores, se comienza calculando los pasos necesarios para conseguir llegar a donde se desea, para ello se toma la posición a la cual se dirigirá y se le resta en la posición en la cual se encuentra, se habilita los pasos para mover los motores, así como es que se envía a los drives la posición desea, después se asigna los valores de la posición deseada a la última posición, seguido de esto se desactiva los pasos para mover a los motes, y se envía al puerto serial, las posiciones a la cual está el robot cartesiano.

### 3.4.2.2.10. Función de Movimiento

Esta funcion nos permite evaluar en el puerto Serial los parámetros de las coordenadas en el código G ingresado. En el siguiente documento se evalúa dicho procedimiento:

```
void moviment(){
    double newXValue = X;
    double newYValue = Y;
    double newZValue = Z;
    if(Commands.availableValue('X')) // AÑADIDO parámetro X en Go
      newXValue = Commands.GetValue('X');
    if(Commands.availableValue('Y')) // AÑADIDO parámetro Y en Go
      newYValue = Commands.GetValue('Y');
    if(Commands.availableValue('Z')) // AÑADIDO parámetro Z en Go
      newZValue = Commands.GetValue('Z');

    gotoLocation(newXValue, newYValue, newZValue);
}
```

Para ello se crea 3 variables de tipo doble, que nos permite guardar los datos que se obtendrá a continuación. Después con una serie de preguntas, se evalúa si se encuentra un valor en el Puerto que coincida con 'X', 'Y' o 'Z', si se encuentra con uno de estos valores, a las variables declaradas anteriormente se le asigna el valor que se encuentra en esos datos. Y se llama a trabajar a la funcion de Ir a la Locación.

# CAPÍTULO IV

# RESULTADOS

## 4.1 Resultados obtenidos

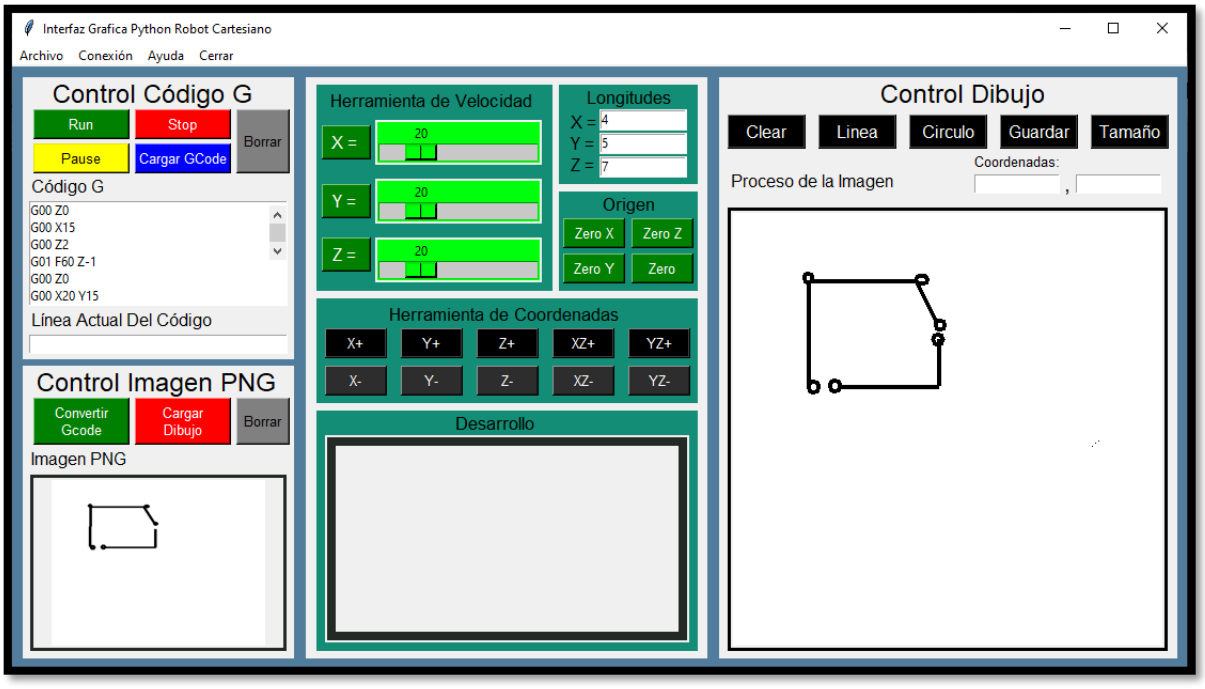
Una vez realizado los programas de la interfaz gráfica y el de control se obtuvieron los siguientes resultados:

### 4.1.1. Interfaz Gráfica

El resultado de la creación de la interfaz gráfica de usuario, que se genera con el código realizado en el entorno de programación de Pycharm, como se puede apreciar se muestra las funciones descriptas en los diagramas de flujo, estas serían realización de un dibujo, conversión de imagen PNG a código G, control mediante el código G, y el control manual del instrumento, dando salida todas las funciones enunciadas.

Figura 17

*Resultado general de la Interfaz Gráfica resultante de Python del Robot Cartesiano.*



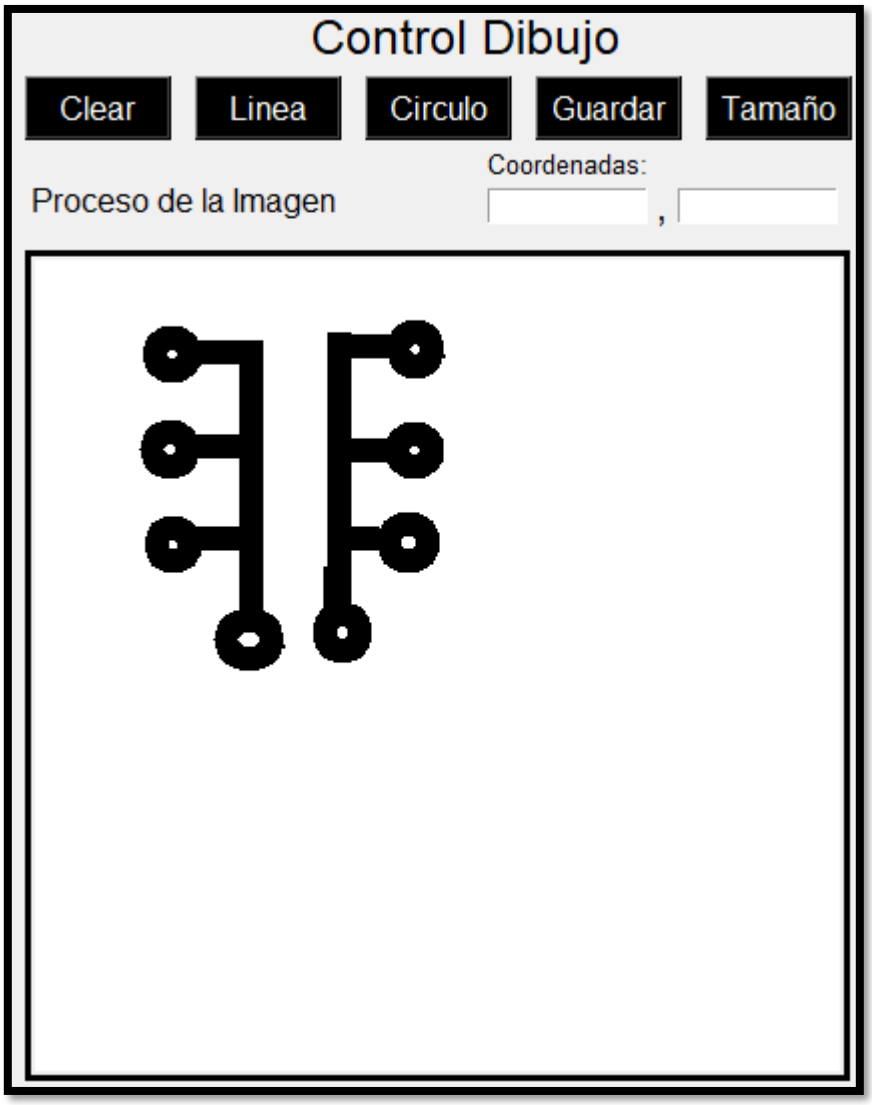

*Fuente: Irwin Saidh Jiménez Serrato, 2022*

*Nota.* Resultado de ejecutar el código mostrado en el anexo 6.

Para poder realizar la simulación del control realizaremos la simulación completa desde la elaboración del circuito con la ayuda de la herramienta de dibujo, como se muestra en la imagen 18, donde se aprecia la elaboración de este.

Figura 18

*Interfaz Gráfica de Python del Robot Cartesiano, creación del dibujo.*



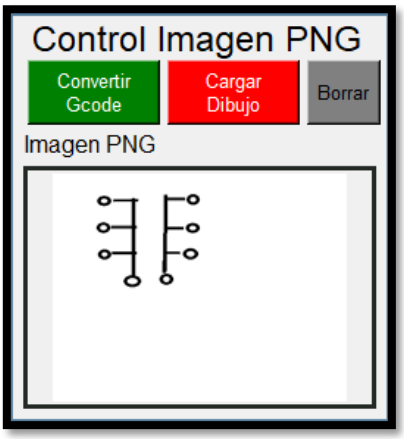

*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Al presionar el botón de guardar, permitimos al programa almacenar estos datos en la computado, para la posterior etapa, la cual permite convertir la imagen

cargada en la memoria y poder convertirlo al Código g, el cual se almacena en dicha memoria. Esto es mostrado en la figura 19.

Figura 19

*Interfaz Gráfica de Python Robot Cartesiano, conversión de imagen PNG a Gcode.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Al poder cargar el archivo de código g en la interfaz, este se muestra para poder ser utilizado en su posterior fase.

Figura 20

*Interfaz Gráfica de Python Robot Cartesiano, control del robot cartesiano por el Gcode.*


*Fuente: Irwin Saidh Jiménez Serrato, 2022*

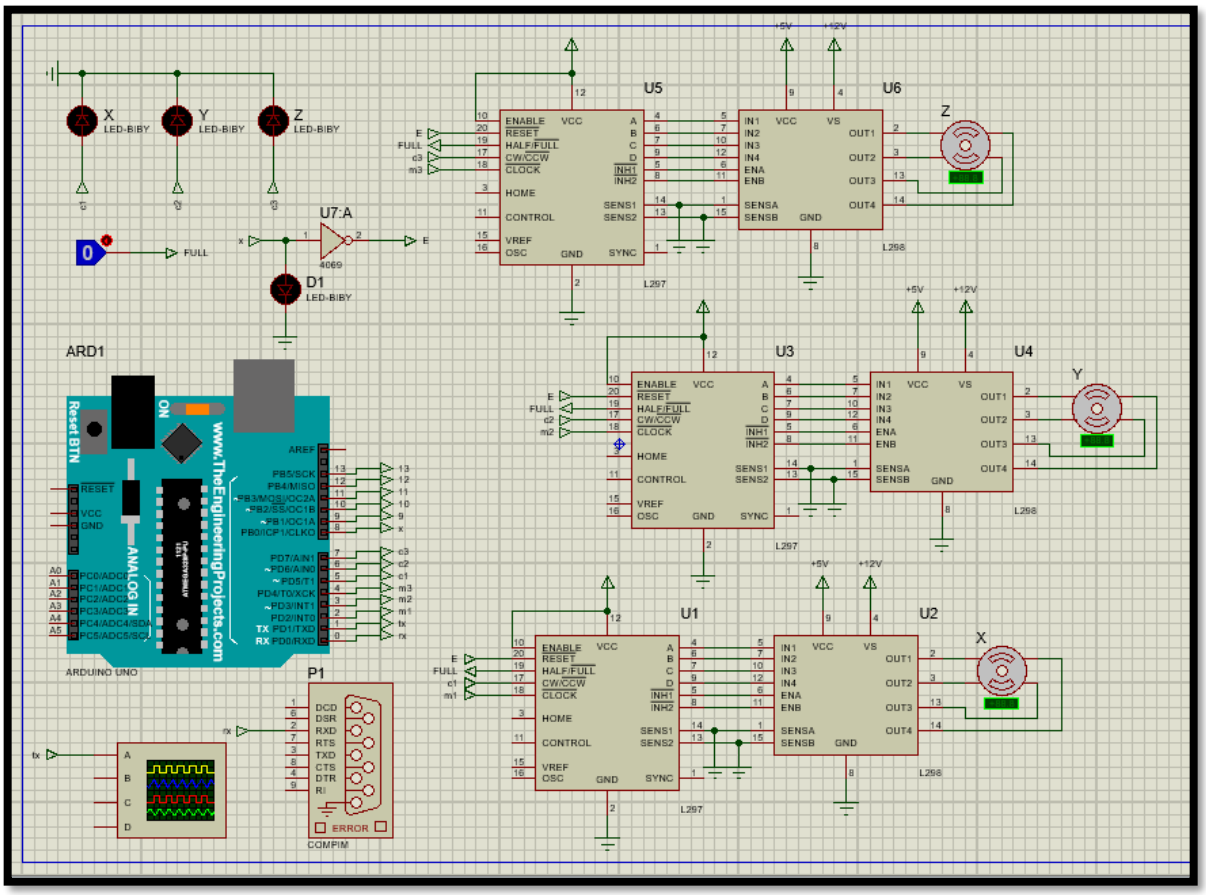
## 4.1.2. Resultado del control de Arduino

Para poder realizar la simulación general del sistema de control, necesitaremos de 3 cosas, la interfaz gráfica, un circuito de simulación, y habilitar un par de puertos seriales para la comunicación entre estos.

La interfaz gráfica se explicó en la sección ***4.1.1. Interfaz Gráfica***. Y el circuito de la simulación del robot cartesiano se muestra en la figura 21. En este se puede apreciar los motores a pasos, de los 3 ejes, así como la conexión entre el Arduino y un puerto virtual.

Figura 21

*Circuito de simulación para el robot cartesiano.*

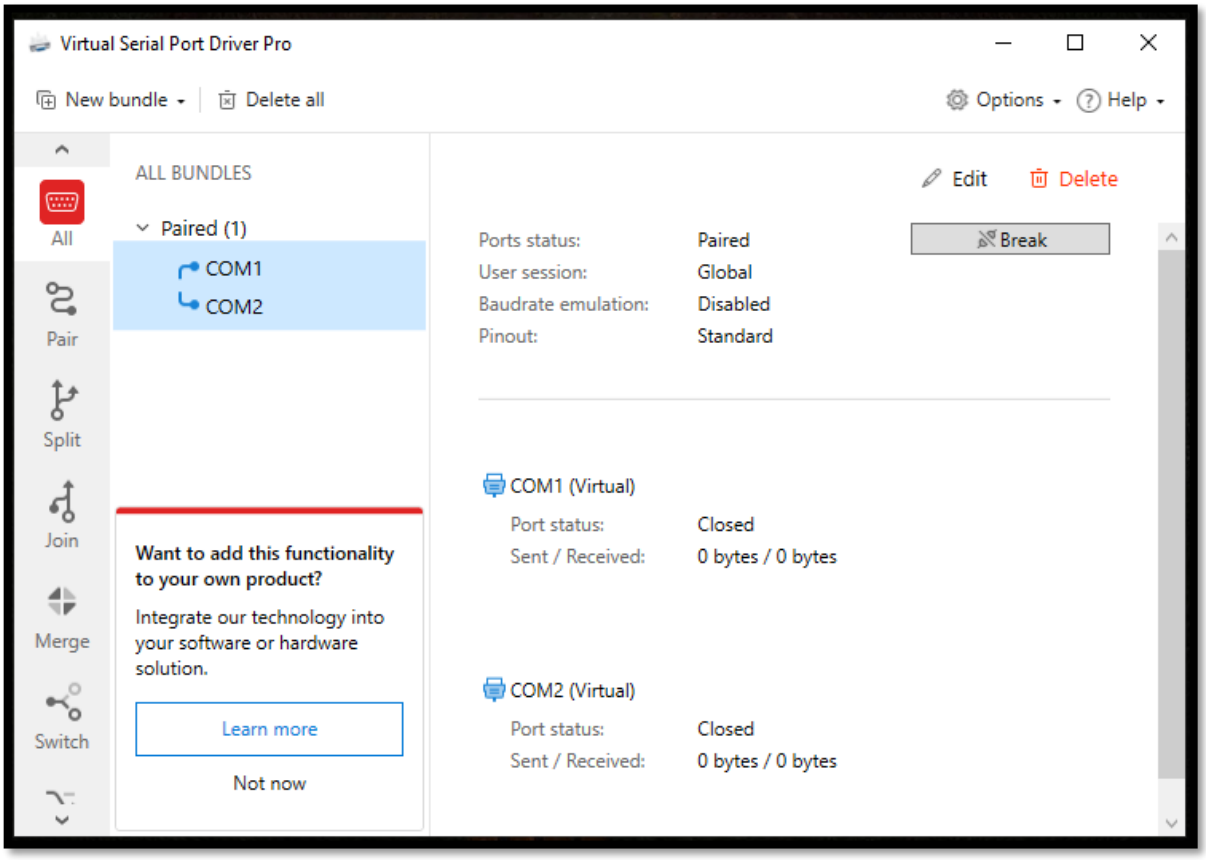



*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Para ello se habilita los puertos seriales a utilizar, por ello utilizaremos la aplicación de Virtual Serial Port Driver Pro, permitiendo crear la paridad del Puerto COM1 y COM2.

Figura 22

*Apertura de los puertos seriales.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Seguido de esto conectaremos la interfaz y el circuito de control por medio de los puertos seriales.

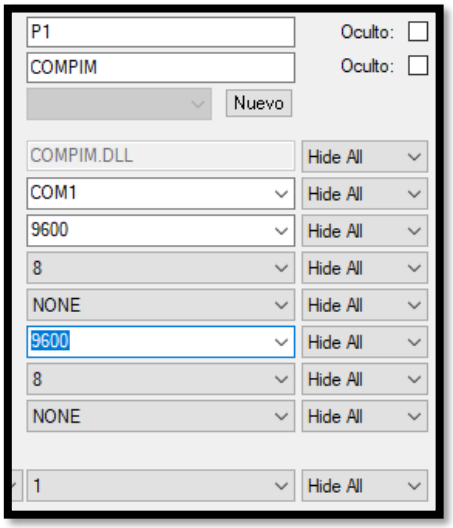
Figura 23

*Conexión del puerto serial de la interfaz gráfica.*


*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Figura 24

*Configuración del puerto serial del Arduino.*

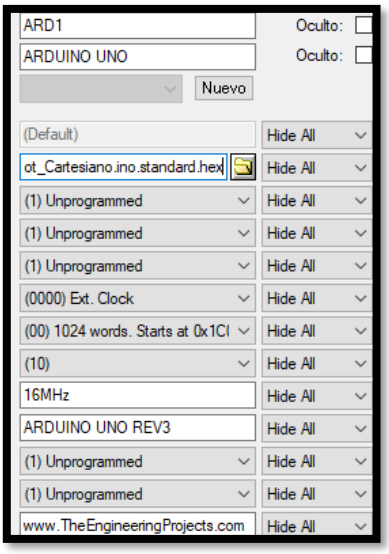



*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Seguido de esto cargamos el programa de control al Arduino y configuramos sus ciclos por segundo para un correcto desempeño

Figura 25

*Configuración del circuito simulado del Arduino.*

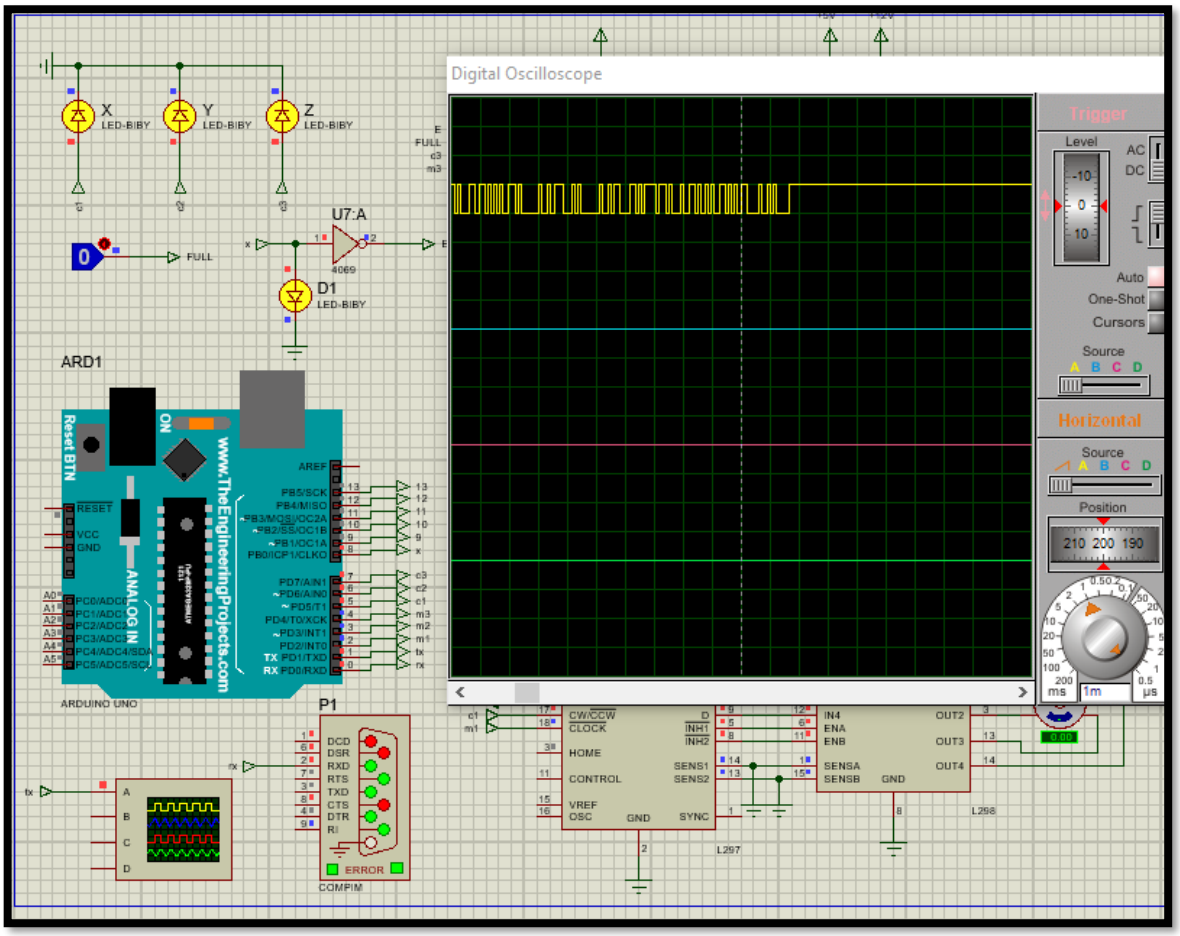



*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Una vez se realiza la comunicación entre ambas partes se puede visualizar en la simulación el intercambio de datos entre estos.

Figura 26

*Simulación del circuito controlador.*

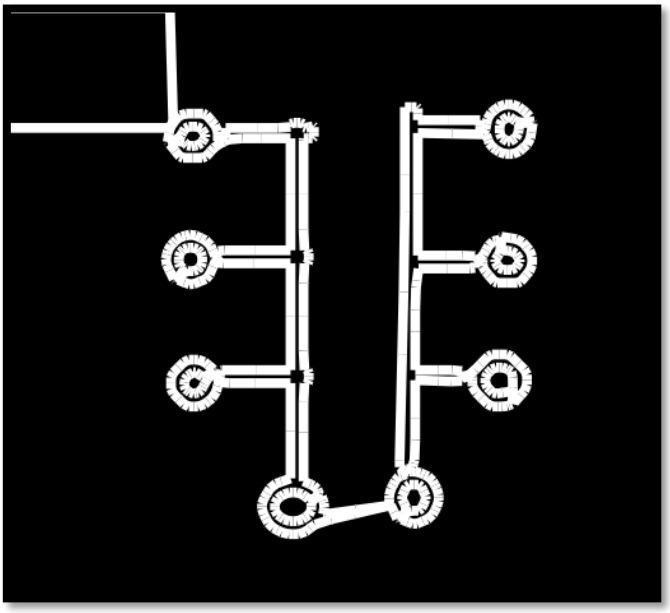



*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Para el resultado de la simulación del grabado del circuito se muestra en la figura 27, en dicha figura se puede apreciar el paso de la herramienta que realiza el trabajo del grabado.

Figura 27

*Resultado de la simulación del grabado del circuito.*



*Fuente: Irwin Saidh Jiménez Serrato, 2022*

# CAPÍTULO V

# CONCLUSIONES

## 5.1. Conclusiones del proyecto, recomendaciones y experiencia profesional y personal adquirida

Lo expuesto anteriormente permite concluir que el problema se aborda realizando una interfaz gráfica de usuario, desarrollado en el lenguaje de programación de Python, donde se contiene distintas herramientas para el control del robot cartesiano, las cuales van desde el uso de código G, creación de imágenes, control mediante los 3 ejes, el control de velocidades, y visualización de imágenes y de código G.

## 5.2. Conclusiones relativas a los objetivos específicos

Al poder consultar distintas fuentes información durante el estado del arte, nos pudimos dar una idea más clara lo que queríamos llegar al finalizar el proyecto. Permitiendo dar un mejor uso de lo previamente analizado y poder implementar con mayor facilidad las distintas herramientas. Así como el diseño y la implementación de las distintas herramientas previstas para el control del sistema mecatrónico.

## 5.3 Conclusiones relativas al objetivo general

Como bien se sabe, el objetivo específico de nuestro proyecto es el diseñar un algoritmo de control en Python para un robot cartesiano de 3 dimensiones para la impresión de sensores de carbono. Por lo tanto, se puede decir, que el principal del proyecto se cumplió, ya que se logró realizar el algoritmo de control, mediante una interfaz amigable para el usuario final.

## 5.4 Aportaciones originales

La principal contribución del proyecto es la creación de una interfaz gráfica para un usuario principiante o de poco conocimiento en la materia, limitando al usuario el uso una pantalla mediante el uso de botones, y creación de imágenes para su uso.

En el control realizado con programación en el código de Python, se produjo con un conjunto de funciones creadas a partir de módulos instalados en el IDE de Pycharm, haciendo uso de conjunciones para el correcto desarrollo del programa.

El control de la tarjeta de Arduino es una modificación de un código capaz de controlar una máquina de 2 ejes, realizando una adaptación para robot cartesiano de 3 ejes.

## 5.5 Limitaciones del modelo planteado

La utilización de las herramientas para el funcionamiento de la conversión de imágenes en código G, esto dentro del lenguaje de programación de Python, incorporando una solución más eficiente.

## 5.6 Recomendaciones

Se recomienda utilizar este programa para circuitos sencillos, ya que este es el principal objetivo, la realización de circuitos para el uso de circuitos de sensores de carbono.

# CAPÍTULO VI

# COMPETENCIA DESARROLLADA

## 6.1. Competencias desarrolladas y/o aplicadas

Al realizar este trabajo de residencia profesional, me permitió mejorar mis conocimientos, actitudes, valores y habilidades, para alcanzar mis metas. Tales atributos, características y cualidades que utilice al día a día para el desarrollo de este trabajo.

### Competencias Genéricas

Dentro de las competencias genéricas desarrolladas en el trabajo se muestra a continuación.

- El gran uso del idioma inglés debido a que la gran mayoría de fuentes de información utilizada se encontraban el idioma mencionado.
- El gran abstracción, análisis y sinterización de información obtenida mediante la búsqueda de información en los distintos canales de información de las tecnologías que tenemos al alcance.
- Al realizar nuestro algoritmo nos presentamos con distintos problemas, los cuales nos vimos en la necesidad de Identificar, planear y solucionarlos de forma efectiva y adecuada.
- Al realizar dicho trabajo, lo realizamos con el compromiso ético, al dar crédito a los autores originales.
- Este proyecto es un trabajo colaborativo, que consta de 2 partes, donde se dividió entre el trabajo de software del control y el armado de un robot cartesiano. Haciendo uso de un efectivo trabajo en equipo.

### Competencias Especificas

Las competencias especificas desarrolladas durante este desarrollo de la aplicación fueron las siguientes:

- Desarrollo de la creatividad e ingenio en el desarrollo de este algoritmo.
- Diseño, construcción y operación el sistema de control cartesiano.
- Implementación de distintos campos de interés a la carrera de ingeniería mecatrónica.

- Análisis de circuitos eléctrico, y electrónicos.
- Análisis e integración de códigos previamente realizados.
- Implementación de la automatización y sistema de control del sistema mecatrónico.
- Creando software para la manipulación de dispositivos.
- Creación de software que procesa la información de manera automatizada.
- Creación transmisión de señales mediante las telecomunicaciones de los dispositivos electrónicos.

# CAPÍTULO VII
# FUENTES DE INFORMACIÓN

infoPLC. (27 de Diciembre de 2016). *Automatizacion Industrial, Robótica e Industria*. Obtenido de https://www.infoplc.net/media/k2/items/cache/9076d455bda67229ab3d4fa5654f2d04_XL.jpg

(26 de Septiembre de 2007).

Arduino Inc. (16 de Febrero de 2015). *Bricogeek*. Obtenido de https://tienda.bricogeek.com/1157-large_default/arduino-uno.jpg

BADILLO NÁJERA, M. E. (Noviembre de 2016). CONTROL DE TRAYECTORIAS DE UN ROBOT CARTESIANO. Tulancingo de Bravo, Hidalgo, México. Obtenido de http://www.upt.edu.mx/Contenido/Investigacion/Contenido/TESIS/MAC/2016/MAC_T_2016_02_MBN.pdf

Berrio, J., Arcos, E., Zuluaga, J., & Corredor, S. (Octubre de 2015). Diseño y construcción de un robot cartesiano de 3 grados de libertad. Envigado, Colombia. Obtenido de https://revistabme.eia.edu.co/index.php/mem/article/download/815/732

Cortes Berruecos, R., Torres Méndez, S. J., Muñoz Hernandez, G. A., & Reyes Cortes, F. (2017). Algoritmo para obtención de Índice de desempeño, en configuraciones de robots SCARA, Cartesiano y Antropomórfico. Puebla, Puebla, México. Obtenido de https://www.researchgate.net/profile/Ronald-Gutierrez/publication/323295376_Robotica_Educativa_para_la_Comprension_y_Aplicacion_de_la_Ley_de_Ohm/links/5a8c9396458515a4068ae21d/Robotica-Educativa-para-la-Comprension-y-Aplicacion-de-la-Ley-de-Ohm.pdf#page=4

Gastello, E. (30 de Junio de 2020). ¿Qué es una fuente de poder? México. Obtenido de https://acmax.mx/que-es-una-fuente-de-poder

Godoy Hernández, R. D., & Rodríguez Quintero, W. (23 de Octubre de 2007). Diseño y modelamiento de un robot cartesiano para el posicionamiento de piezas. Bogotá, Colombia. Obtenido de https://ciencia.lasalle.edu.co/cgi/viewcontent.cgi?article=1049&context=ing_automatizacion

Hernández Sánchez, S. (2019). *Interfaz gráfica de aplicaciones de control desarrolladas por Python.*

I, S. (26 de Julio de 2007). *wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Sistema_de_coordenadas#/media/Archivo:3D_coordinate_system.svg

Jara Ayala, C. A., & Renato Del Castillo, H. (2019). DESARROLLO DE UN CONTROLADOR PARA REDUCIR EL ERROR EN LAS TRAYECTORIAS LINEALES DE UN ROBOT CARTESIANO TIPO GANTRY. Trujillo, Peru. Obtenido de http://www.upt.edu.mx/Contenido/Investigacion/Contenido/TESIS/MAC/2016/MAC_T_2016_02_MBN.pdf

Larraioz. (04 de 2020). *Larraioz Elektronika*. Obtenido de https://larraioz.com/wp-content/uploads/2020/04/IMG_IAI_Robot_Cartesian_XYZ_XYG-ZS-1.jpg

Leyenda Oriental. (Noviembre de 2018). *Mercado Libre*. Obtenido de https://http2.mlstatic.com/D_NQ_NP_771527-MLM28554866050_112018-O.webp

López Segovia, J. L., Alamilla Santiago, M., & Domínguez Vázquez, J. F. (Noviembre de 2007). Robot Cartesiano: Seguiento de trayectorias irregulares Arbitrarias Mediante computadora. Pachuca de Soto, Hidalgo, México. Obtenido de https://www.uaeh.edu.mx/docencia/Tesis/icbi/licenciatura/documentos/Robot%20cartesiano%20seguimiento%20de%20trayectorias.pdf

Metal y Mecanica. (29 de Enero de 2013). *wordpress*. Obtenido de https://metalymecanica.files.wordpress.com/2013/01/cnc_thumb.jpg?w=392&h=223

Munoz, M. (Febrero de 2014). *Centro de Mecanizado*. Obtenido de https://marcosmunoz.com/wp-content/uploads/2014/02/G-code.jpg

Muñoz, V. F., García Morales, I., Pérez del Pulgar, C., Gómez de Gabriel, J. M., Fernández Lozano, J. J., García Cerezo, A., . . . Toscano, R. (2006). Control Cartesiano de un Asistente robótico para cirugía Laparoscópica. *Revista Iberoamerica de Automática e Informática Industrial*, 64-75. Obtenido de https://polipapers.upv.es/index.php/RIAI/article/view/8161

MV electronica. (Agosto de 2021). *mvelectronica*. Obtenido de https://mvelectronica.s3.us-east-2.amazonaws.com/productos/A4988/60a7e180a8c21.webp

Pasquel Castillo, C. R. (2019). DISEÑO DE UNA FRESADORA CNC PARA PCBs MEDIANTE ARCHIVOS GERBER Y FUENTE EXCELLON. Quito, Ecuador. Obtenido de http://repositorio.uisrael.edu.ec/handle/47000/1966

Rodríguez Rodríguez, C. L. (2017). Diseño y contrucción de un robot cartesiano con un control de posición punto a punto. Bogotá, Colombia. Obtenido de http://repository.unilibre.edu.co/handle/10901/11249

Sanchez, J. J., & Rodriguez, G. E. (2008). *Dispositivo electrónico automatizado con electro válvulas para el control de fugas de gas domiciliario.* Corporación Universitaria Minuto de Dios.

TechLib. (2021). *Tarjeta Controladora.* Obtenido de https://techlib.net/definition/controllercard.html

Tobón Restrepo, I. A., & Agudelo García, V. J. (2016). Módulo didáctico de sensorica para laboratorio de mecatrónica etapa 7. Medellín, Colombia. Obtenido de http://repositorio.pascualbravo.edu.co:8080/jspui/bitstream/pascualbravo/687/1/Rep_IUPB_Tec_Sis_Mecatr%C3%B3nicos_Sensorica.pdf

Val, C. G. (21 de Diciembre de 2009). Robot Cartesiano de tres ejes controlado por computadora. Buenos Aires, Argertina. Obtenido de https://repositorio.uca.edu.ar/handle/123456789/11220

Van Roussum, G., & Drake Jr, F. L. (2017). *El tutorial de Python.* Python Software Foundation.

# CAPÍTULO VIII

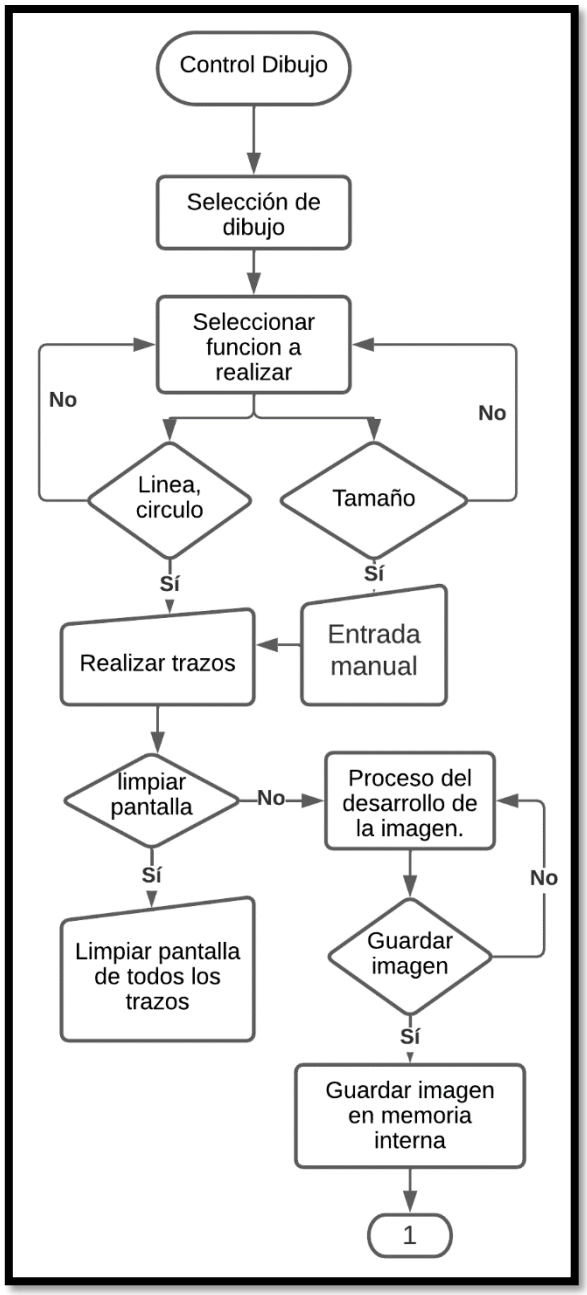
# ANEXOS

# Anexo 1

Diagramas de bloque para la interfaz gráfica, del control del robot cartesiano.

Figura 28

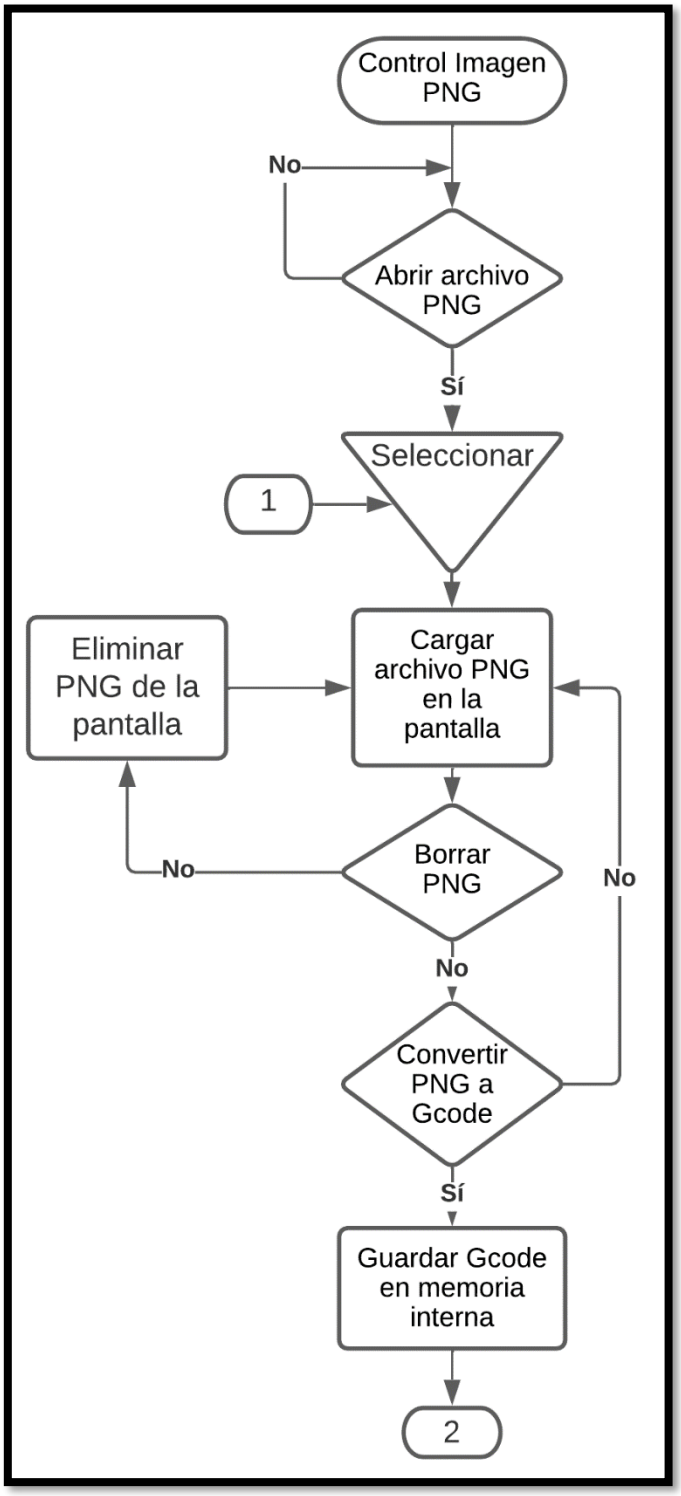
*Diagrama de Bloque para el diseño del dibujo para el circuito.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Figura 29

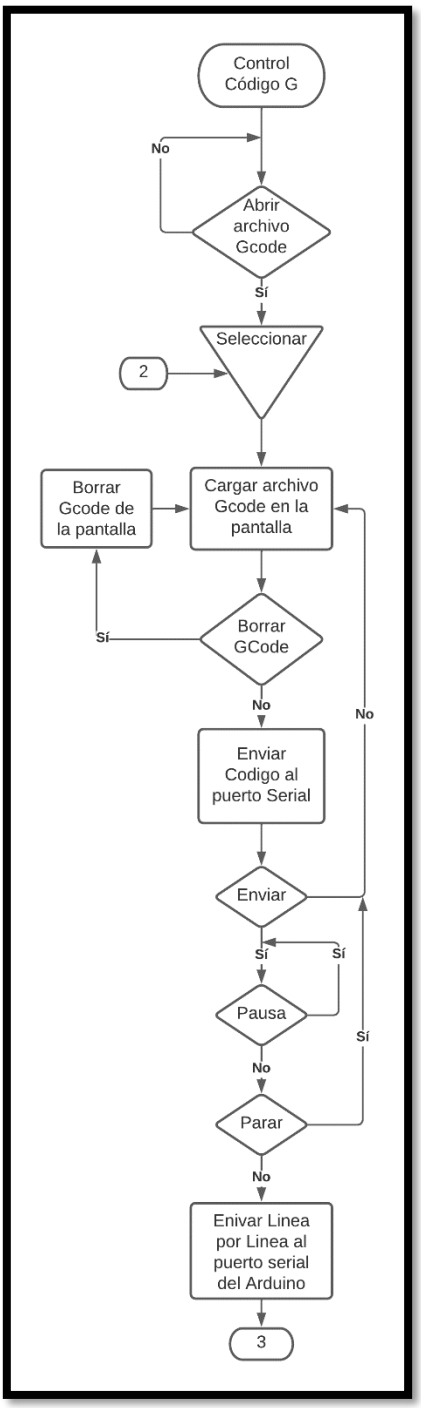
*Diagrama de Bloque para la conversión de imagen PNG a Código G para el circuito.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Figura 30

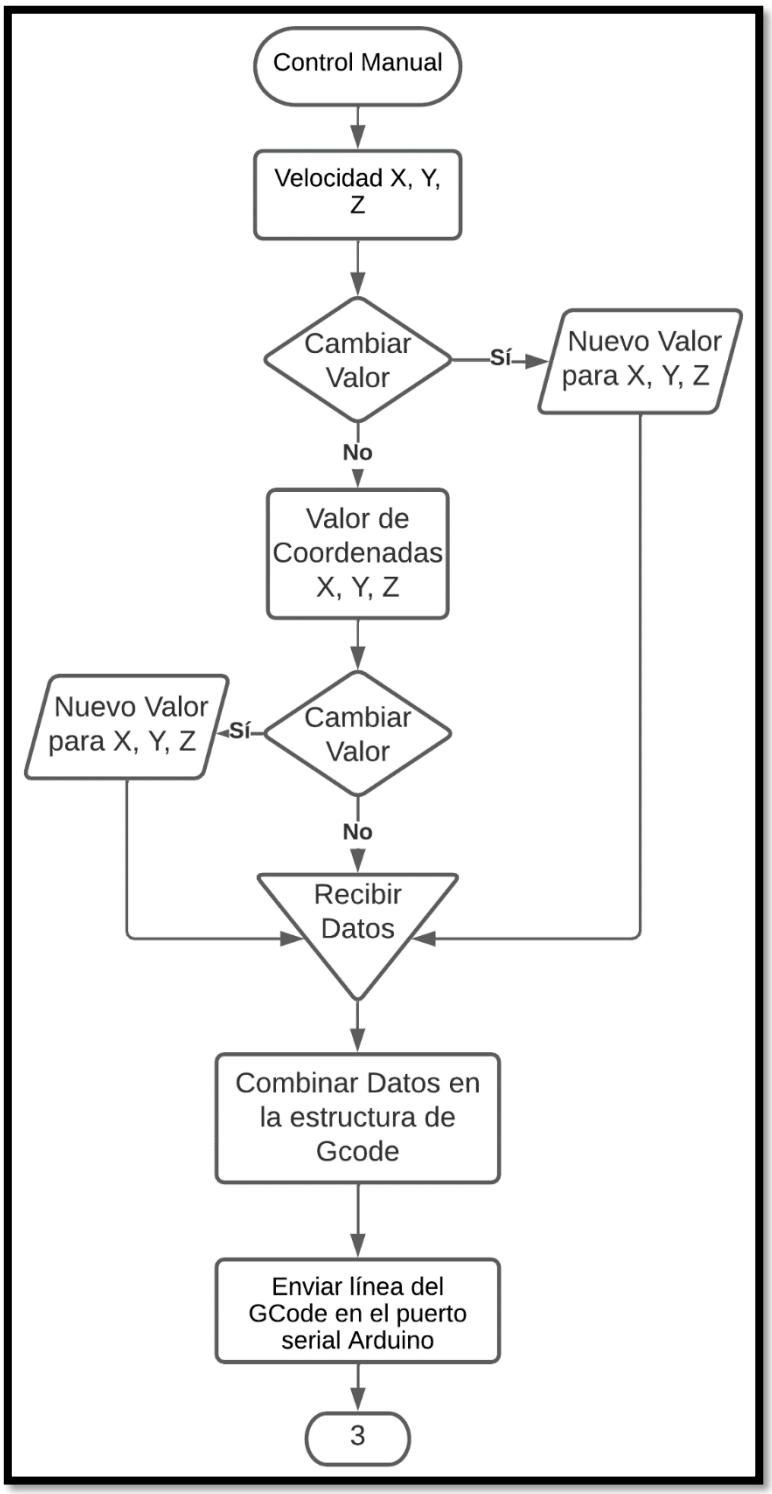
*Diagrama de Bloque para la comunicación entre la interfaz gráfica y Arduino con el uso del Código G para la impresión del circuito.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

Figura 31

*Diagrama de Bloque para el control manual para la interfaz gráfica y el programa de control del Arduino.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

# Anexo 2

Para poder descargar el software de Pycharm puedo visitar la página oficial.

Figura 32

*Página de descarga de Pycharm.*


*Fuente: Irwin Saidh Jiménez Serrato, 2022*

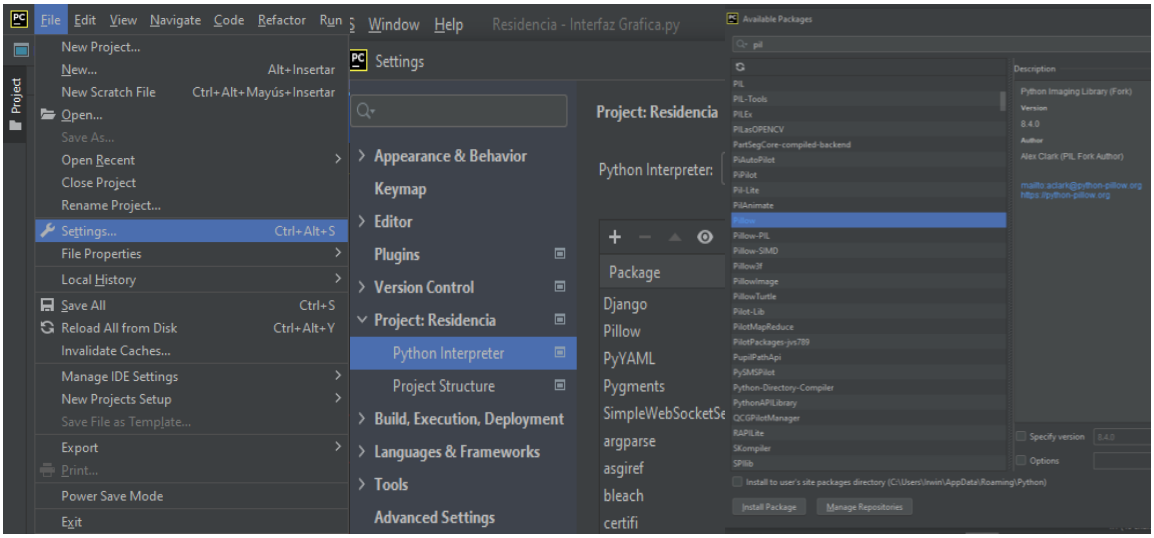
# Anexo 3

Para poder instalar un módulo en Pycharm debe seguir lo siguiente pasos:

Se dirigiré dentro de la Aplicación de Pycharm a la sección de File, en el apartado de settings, después en el Project: 'Name', se encuentra 'Python interpreter' donde encontraremos un símbolo de +, donde saltara la ventana denominada 'Available Package', donde en el buscador puedo escribir el nombre el módulo deseado y poder instalarlo con el botón 'Install Package'.

Figura 33

*Proceso de instalación de módulos en Pycharm.*



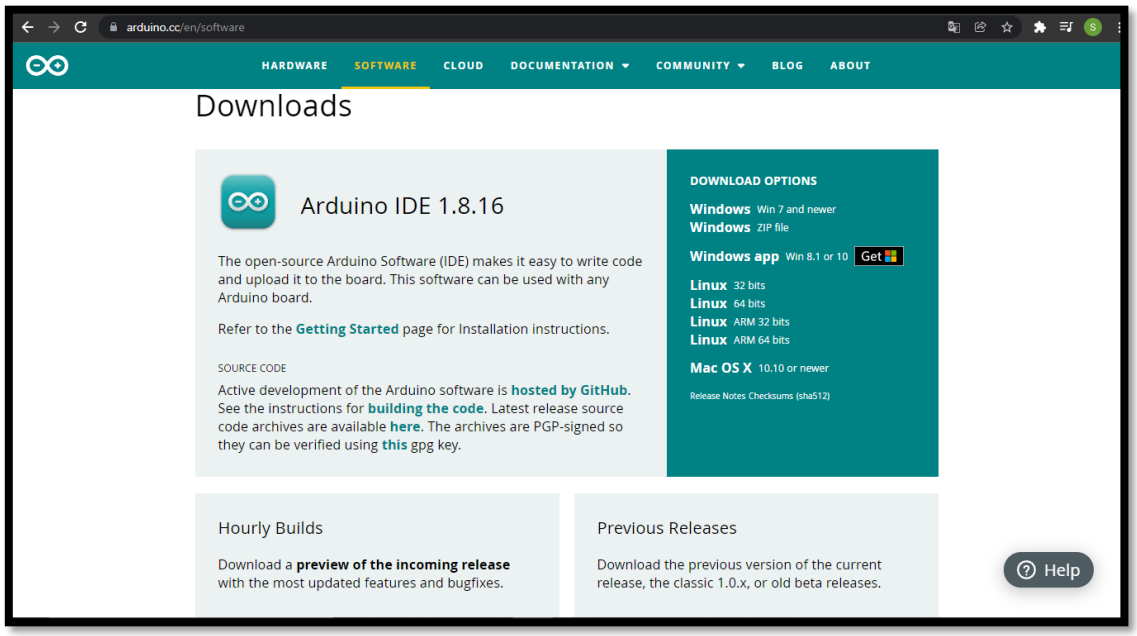

*Fuente: Irwin Saidh Jiménez Serrato, 2022*

# Anexo 4

Para poder descargar el software de Arduino puedo visitar la página oficial.

Figura 34

*Página oficial para la descarga de Arduino.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

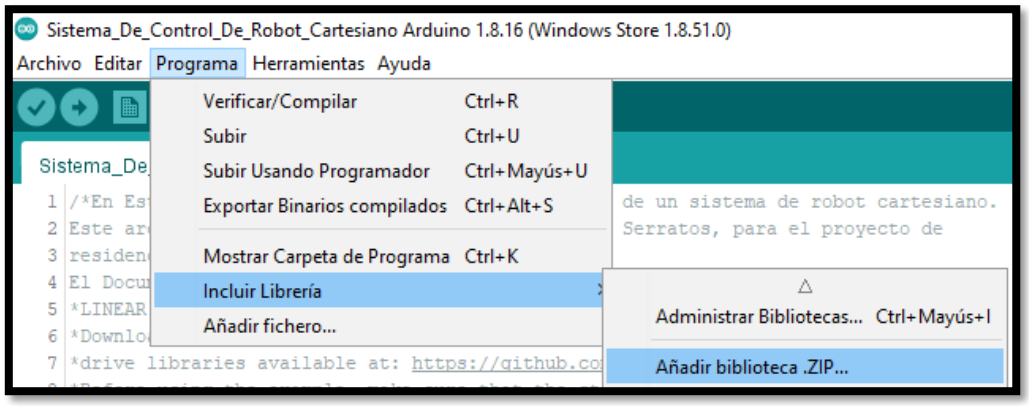
# Anexo 5

Para poder instalar una librería en Arduino siga los siguientes pasos:

Para ello se debe de dirigir a su IDE de Arduino en la barra menu llamada Programa, encontrara una sección llamada Incluir Librería, donde se diga Añadir biblioteca #. ZiP, le abrirá una ventana emergente donde tendrá que buscar su carpeta que previamente descargo y podrá incluirla.

Figura 35

*Instalación de una librería en Arduino.*




*Fuente: Irwin Saidh Jiménez Serrato, 2022*

# Anexo 6

Código realizado para la interfaz gráfica de usuario.

```
#!/usr/bin/python
# -*- coding: iso-8859-1 -*-

""" Importamos Librerías """
from tkinter import *
from tkinter import Canvas
from tkinter import filedialog
from tkinter import ttk
from tkinter import messagebox
from typing import Union
from PIL import Image
from PIL import ImageTk
from PIL import ImageDraw
import time
from serial import Serial
import serial

esRecibido = False
valor = 0.0

'''##########################################################################
##########################################'''
'''########################################DEfinimos la clase de
comunicación#########################################'''
'''##########################################################################
##########################################'''


class Comunicacion:
    puerto: Union[Serial, Serial, Serial, Serial]

    def Conexion(self, puertoUSB):
        self.puerto = serial.Serial(puertoUSB, baudrate=9600,
timeout=1.5)
        print('Conectado')
        time.sleep(1.8)

    def LeerDatos(self):
        return self.puerto.readline().decode('ascii')

    def LeerBufer(self):
        return self.puerto.readline(256)

    def EnviarX(self, dato):
        a = 'X' + str(dato)
        self.puerto.write(a.encode('ascii'))
        time.sleep(0.01)

    def EnviarY(self, dato):
```

```
        a = 'Y' + str(dato)
        self.puerto.write(a.encode('ascii'))
        time.sleep(0.01)

    def EnviarZ(self, dato):
        a = 'Z' + str(dato)
        self.puerto.write(a.encode('ascii'))
        time.sleep(0.01)

    def EnviarVelX(self, dato):
        a = 'Vx=' + str(dato)
        self.puerto.write(a.encode('ascii'))
        time.sleep(0.01)

    def EnviarVelY(self, dato):
        a = 'Vy=' + str(dato)
        self.puerto.write(a.encode('ascii'))
        time.sleep(0.01)

    def EnviarVelZ(self, dato):
        a = 'Vz=' + str(dato)
        self.puerto.write(a.encode('ascii'))
        time.sleep(0.01)

    def EnviarCodigo(self, datos):
        self.puerto.write(datos.encode('ascii'))
        time.sleep(0.5)


'''#########################################################################
##############################################'''
'''##############################Definimos la clase de Paint encargada
de hacer el dibujo############################'''
'''#########################################################################
##############################################'''


class Paint:
    def __init__(self):
        self.filename = "imagenApoyo.png"
        canvas13 = Canvas(interfaz, width=425, height=540, bg='#f0f0f0')
        canvas13.pack()
        canvas13.place(x=662, y=10)
        Label(interfaz, fg='black', font=('Arial', 18), text='Control
Dibujo').place(x=810, y=10)
        Label(interfaz, fg='black', font=('Arial', 12), text='Proceso de
la Imagen').place(x=670, y=95)
        canvas15 = Canvas(interfaz, width=412, height=415, bg='#000000')
        canvas15.pack()
        canvas15.place(x=668, y=130)
        Label(interfaz, fg='black', font=('Arial', 10),
text='Coordenadas:').place(x=898, y=78)
        Label(interfaz, fg='black', font=('Arial', 15),
text=',').place(x=982, y=95)
        lstListbox6 = Listbox(interfaz, width=13, height=1)
        lstListbox6.place(x=995, y=100)
        lstListbox7 = Listbox(interfaz, width=13, height=1)
```

```
        lstListbox7.place(x=900, y=100)
        self.canvas14 = Canvas(interfaz, width=402, height=405,
bg='#FFFFFF')
        self.canvas14.pack()
        self.canvas14.place(x=673, y=135)
        self.x, self.y = list(), list()
        self.current_y, self.current_x, self.n = 0, 0, 0
        self.exist = 0
        self.cords, self.corde, self.ocords, self.ocorde = list(),
list(), list(), list()
        self.oldx, self.oldy, self.t = 0, 0, ""
        Button(interfaz, fg='white', text='Clear', font=('Arial', 12),
width=7, height=1, background='black',
               command=self.clear_screen).place(x=670, y=45)
        Button(interfaz, fg='white', text='Linea', font=('Arial', 12),
width=7, height=1, background='black',
               command=self.line_selected).place(x=755, y=45)
        Button(interfaz, fg='white', text='Circulo', font=('Arial', 12),
width=7, height=1, background='black',
               command=self.circle_selected).place(x=840, y=45)
        Button(interfaz, fg='white', text='Guardar', font=('Arial', 12),
width=7, height=1, background='black',
               command=self.Guardar_Img).place(x=925, y=45)
        Button(interfaz, fg='white', text='Tamaño', font=('Arial', 12),
width=7, height=1, background='black',
               command=self.Size).place(x=1010, y=45)
        self.img = Image.new("RGB", (402, 405), '#FFFFFF')
        self.draw = ImageDraw.Draw(self.img)
        self.size = '4'

    def redraw(self):
        if len(self.corde):
            for i in range(len(self.corde)):
                self.canvas14.create_line(self.cords[i], self.corde[i],
width=self.size, fill="black")

        if len(self.ocorde):
            for i in range(len(self.ocorde)):
                self.canvas14.create_oval(self.ocords[i], self.ocorde[i],
width=self.size, outline="black")

    def clear_screen(self):
        self.canvas14.delete("all")
        self.current_x, self.current_y, self.n, self.oldx, self.oldy,
self.x, self.y, self.t = 0, 0, 0, 0, 0, list(), list(), ""
        self.ocords, self.ocorde, self.corde, self.cords = list(),
list(), list(), list()
        self.draw.rectangle([(40, 40), (402 - 10, 405 - 10)],
fill='#FFFFFF', outline='#FFFFFF')

    def button_released(self, event):
        self.x.extend([self.current_x, self.current_y])
        self.y.extend([event.x, event.y])
        if self.t == "l":
            self.draw.line((self.current_x, self.current_y, event.x,
event.y), fill=(0, 0, 0), width=5)
            self.img.save(self.filename)
```

```
            self.cords.append(self.x)
            self.corde.append(self.y)
        elif self.t == "c":
            self.draw.ellipse((self.current_x, self.current_y, event.x,
event.y), fill=(255, 255, 255),
                              outline=(0, 0, 0), width=5)
            self.img.save(self.filename)
            self.ocords.append(self.x)
            self.ocorde.append(self.y)
        self.exist = 1
        self.current_x, self.current_y, self.n, self.oldx, self.oldy,
self.x, self.y, self.t = 0, 0, 0, 0, 0, list(), list(), ""

    def create_circle(self, event):
        self.canvas14.delete("all")
        if self.exist == 1:
            self.redraw()
        if self.n == 0:
            self.current_x, self.current_y = event.x, event.y
            self.n = 1
        self.canvas14.create_oval(self.current_x, self.current_y,
event.x, event.y, width=self.size, outline="black")

        self.oldx = event.x
        self.oldy = event.y
        self.t = "c"
        self.canvas14.bind("<ButtonRelease-1>", self.button_released)

    def line_click(self, event):
        self.canvas14.delete("all")
        if self.exist == 1:
            self.redraw()
        if self.n == 0:
            self.current_x, self.current_y = event.x, event.y
            self.n = 1
        self.canvas14.create_line(self.current_x, self.current_y,
event.x, event.y, width=self.size, fill='#000000')
        self.oldx = event.x
        self.oldy = event.y
        self.t = "l"
        self.canvas14.bind("<ButtonRelease-1>", self.button_released)

    def line_selected(self):
        self.canvas14.bind("<B1-Motion>", self.line_click)

    def circle_selected(self):
        self.canvas14.bind("<B1-Motion>", self.create_circle)

    def Guardar_Img(self):
        self.img.save(self.filename)

    def Size(self):
        def obtener_info():
            self.size = lst_desp.get()
            print(self.size)
            interfaz3.destroy()
```

```
        interfaz3 = Toplevel(interfaz)
        interfaz3.title('Tamaño')
        interfaz3.geometry('250x80')
        lst_desp = ttk.Combobox(interfaz3)
        lst_desp.place(x=20, y=30)
        opc = ["2", "4", "6", "8", "10", "12", "14"]

        lst_desp['values'] = opc

        Button(interfaz3, text="Capturar", bg='gold',
command=obtener_info).place(x=180, y=27)


'''#########################################################################
##########################################'''
'''#########################Definimos las funciones utilizadas en la
barra Menu#####################################'''
'''#########################################################################
##########################################'''


def AbrirG():
    Arch = filedialog.askopenfilename(initialdir='/', title='Selecione un
archivo',
                                      filetypes=(('.txt', '*.txt'), ('all
files', '*.*')))
    f = open(Arch, 'r')
    for x in f:
        lstListbox1.insert(0, x)
    f.close()


def AbrirI():
    Arch = filedialog.askopenfilename(initialdir='/', title='Selecione un
archivo',
                                      filetypes=(('.png', '*.png'), ('all
files', '*.*')))
    f = Image.open(Arch)
    f = f.resize((200, 155))
    img = ImageTk.PhotoImage(f)
    Png = Label(interfaz, image=img)
    Png.image = img
    Png.place(x=35, y=385)


def Conectar():
    def Conecta():

        conectar = messagebox.askquestion('Puerto', message='Deseas
establecer la conexión')
        if conectar == 'yes':
            puertoRS.Conexion(comboBox.get())
            messagebox.showinfo('Conexión', message='Conexión lograda')
            interfaz2.destroy()

    puertos = []
    for i in range(1, 21, 1):
```

```
        COM = 'COM' + str(i)
        try:
            serial.Serial(COM, 9600)
            time.sleep(0.1)
            puertos = COM
        except:
            pass

        time.sleep(0.1)
    interfaz2 = Toplevel(interfaz)
    interfaz2.title('Selección de Puerto')
    interfaz2.geometry('280x100')
    interfaz2.focus_set()
    interfaz2.grab_set()
    comboBox = ttk.Combobox(interfaz2, width=10)
    comboBox.place(x=100, y=10)
    comboBox['values'] = puertos

    Button(interfaz2, text='Conectar', command=Conecta).place(x=100,
y=50)


'''#########################################################################
##########################################'''
'''###################Definimos las funciones para el uso de los archivos
de ayuda y de acerca de###################'''
'''#########################################################################
##########################################'''


def ArchivoHelp():
    root = Tk()
    root.title('Help')
    Lb = Listbox(root, width=95, height=27)
    Lb.configure(background='#FFFFFF')
    Lb.grid()
    f = open("Help.txt", "r")
    for x in f:
        Lb.insert(END, x)
    f.close()

    root.mainloop()


def ArchivoAbout():
    root = Tk()
    root.title('About')
    Lb = Listbox(root, width=82, height=5)
    Lb.configure(background='#FFFFFF')
    Lb.grid()
    f = open("About.txt", "r")
    for x in f:
        Lb.insert(END, x)
    f.close()
    root.mainloop()
```

```
'''#########################################################################
#########################################'''
'''#############################Definimos las funciones para el uso de
los ejes#####################################'''
'''#########################################################################
#########################################'''
Nx = 0
Ny = 0
Nz = 0


def XA():
    global Nx
    Nx = Nx + 1
    lstListbox3.insert(0, Nx)
    puertoRS.EnviarX(Nx)


def XB():
    global Nx
    Nx = Nx - 1
    lstListbox3.insert(0, Nx)
    puertoRS.EnviarX(Nx)


def YA():
    global Ny
    Ny = Ny + 1
    lstListbox4.insert(0, Ny)
    puertoRS.EnviarY(Ny)


def YB():
    global Ny
    Ny = Ny - 1
    lstListbox4.insert(0, Ny)
    puertoRS.EnviarY(Ny)


def ZA():
    global Nz
    Nz = Nz + 1
    lstListbox5.insert(0, Nz)
    puertoRS.EnviarZ(Nz)


def ZB():
    global Nz
    Nz = Nz - 1
    lstListbox5.insert(0, Nz)
    puertoRS.EnviarZ(Nz)


def XZA():
    global Nx
    global Nz
    Nx = Nx + 1
```

```
    Nz = Nz + 1
    lstListbox3.insert(0, Nx)
    lstListbox5.insert(0, Nz)
    puertoRS.EnviarX(Nx)
    puertoRS.EnviarZ(Nz)


def XZB():
    global Nx
    global Nz
    Nx = Nx - 1
    Nz = Nz - 1
    lstListbox3.insert(0, Nx)
    lstListbox5.insert(0, Nz)
    puertoRS.EnviarX(Nx)
    puertoRS.EnviarZ(Nz)


def YZA():
    global Ny
    global Nz
    Ny = Ny + 1
    Nz = Nz + 1
    lstListbox4.insert(0, Ny)
    lstListbox5.insert(0, Nz)
    puertoRS.EnviarY(Ny)
    puertoRS.EnviarZ(Nz)


def YZB():
    global Ny
    global Nz
    Ny = Ny - 1
    Nz = Nz - 1
    lstListbox4.insert(0, Ny)
    lstListbox5.insert(0, Nz)
    puertoRS.EnviarY(Ny)
    puertoRS.EnviarZ(Nz)


'''#########################################################################
##########################################'''
'''############################Definimos las funciones para los botones
de origen#####################################'''
'''#########################################################################
##########################################'''


def OX():
    global Nx
    Nx = 0
    lstListbox3.insert(0, Nx)
    puertoRS.EnviarX(Nx)


def OY():
    global Ny
```

```
    Ny = 0
    lstListbox4.insert(0, Ny)
    puertoRS.EnviarY(Ny)


def OZ():
    global Nz
    Nz = 0
    lstListbox5.insert(0, Nz)
    puertoRS.EnviarZ(Nz)


def Origen():
    global Nx
    Nx = 0
    global Ny
    Ny = 0
    global Nz
    Nz = 0
    lstListbox3.insert(0, Nx)
    lstListbox4.insert(0, Ny)
    lstListbox5.insert(0, Nz)
    puertoRS.EnviarX(Nx)
    puertoRS.EnviarY(Ny)
    puertoRS.EnviarZ(Nz)


'''#########################################################################
##########################################'''
'''#########################Definimos las funciones para los botones de
velocidad#####################################'''
'''#########################################################################
##########################################'''


def VelX():
    a = V1.get()
    print(a)
    puertoRS.EnviarVelX(a)


def VelY():
    a = V2.get()
    print(a)
    puertoRS.EnviarVelY(a)


def VelZ():
    a = V3.get()
    print(a)
    puertoRS.EnviarVelZ(a)


'''#########################################################################
##########################################'''
'''#########################Configuración de las funciones del control
de codigo G#################################'''
```

```
'''#########################################################################
#########################################'''
m = 0
size = 0


def RunG():
    global m
    global size
    size = lstListbox1.size()
    while m < size:
        lstListbox2.delete(0, END)
        Dato = lstListbox1.get(m)
        lstListbox2.insert(0, Dato)
        puertoRS.EnviarCodigo(Dato)
        m = m + 1


def StopG():
    global m
    global size
    size = 0
    m = 0


def PauseG():
    global size
    size = 0


def BorrarC():
    lstListbox1.delete(0, END)


def CargarC():
    Arch = 'C:/Users/Irwin/PycharmProjects/Residencia/GCodeApoyo.txt'
    f = open(Arch, 'r')
    for x in f:
        lstListbox1.insert(0, x)
    f.close()


'''#########################################################################
#########################################'''
'''############################Configuración de las funciones del control
por Imagen#################################'''
'''#########################################################################
#########################################'''


def BorrarI():
    canvas13 = Canvas(interfaz, width=230, height=155, bg='#f0f0f0')
    canvas13.pack()
    canvas13.place(x=20, y=385)


def CargarD():
```

```
    Arch = 'C:/Users/Irwin/PycharmProjects/Residencia/imagenApoyo.png'
    f = Image.open(Arch)
    f = f.resize((200, 155))
    img = ImageTk.PhotoImage(f)
    Png = Label(interfaz, image=img)
    Png.image = img
    Png.place(x=35, y=385)


def ImageToGcode():
    interfaz3 = Toplevel(interfaz)
    interfaz3.title('Convertir imagen a GCode')
    interfaz3.geometry('400x200')
    interfaz3.focus_set()
    interfaz3.grab_set()
    Label(interfaz3, fg='black', font=('Arial', 10),
          text='Para poder convertir la imagen, acceda al siguiente
link').place(x=9, y=10)
    Link = Listbox(interfaz3, width=63, height=1)
    Link.place(x=9, y=50)
    Link.insert(0, 'https://www.costycnc.it/cm8/')
    Nota = Label(interfaz3, fg='black', font=('Arial', 10), text='Cuando
se encuentre en la selección de '
                                                                'archivo
ingrese la\nsiguiente dirección, '
                                                                'y
seleccione la imagen de apoyo')
    Nota.place(x=9, y=80)
    Nota.pack(side="left")
    Direc = Listbox(interfaz3, width=63, height=1)
    Direc.place(x=9, y=130)
    Direc.insert(0, 'C:/Users/Irwin/PycharmProjects/Residencia/')


'''#########################################################################
##########################################'''
'''#####################################Configuración de la Interfaz
General#########################################'''
'''#########################################################################
##########################################'''
interfaz = Tk()
interfaz.title('Interfaz Grafica Python Robot Cartesiano')
interfaz.geometry('1100x562')
interfaz.configure(background='#527C9B')
'''#########################################################################
##########################################'''
'''#####################################Configuración de la Barra
menu#########################################'''
'''#########################################################################
##########################################'''
barra = Menu(interfaz)
Archivo = Menu(barra)
Archivo.add_command(label='Abrir Archivo Gerber', command=AbrirG)
Archivo.add_command(label='Abrir Archivo PNG', command=AbrirI)
barra.add_cascade(label='Archivo', menu=Archivo)
Conecion = Menu(barra)
Conecion.add_command(label='Conectar', command=Conectar)
```

```
barra.add_cascade(label='Conexión', menu=Conecion)
Help = Menu(barra)
Help.add_command(label='Ayuda', command=ArchivoHelp)
Help.add_command(label='Acerca de', command=ArchivoAbout)
barra.add_cascade(label='Ayuda', menu=Help)
barra.add_cascade(label='Cerrar', command='exit')
interfaz.configure(menu=barra)
'''#########################################################################
##########################################'''
'''##########################################Configuración Codigo
G##########################################'''
'''#########################################################################
##########################################'''
canvas1 = Canvas(interfaz, width=250, height=260, bg='#f0f0f0')
canvas1.pack()
canvas1.place(x=10, y=10)
Texto = Label(interfaz, fg='black', font=('Arial', 18), text='Control
Código G').place(x=35, y=10)
Button(interfaz, fg='white', text='Run', font=('Arial', 10), width=10,
height=1, background='green',
       command=RunG).place(x=20, y=40)
Button(interfaz, fg='white', text='Stop', font=('Arial', 10), width=10,
height=1, background='red',
       command=StopG).place(x=115, y=40)
Button(interfaz, text='Pause', font=('Arial', 10), width=10, height=1,
background='yellow',
       command=PauseG).place(x=20, y=72)
Button(interfaz, fg='white', text='Cargar GCode', font=('Arial', 10),
width=10, height=1, background='blue',
       command=CargarC).place(x=115, y=72)
Button(interfaz, text='Borrar', font=('Arial', 10), width=5, height=3,
background='gray', command=BorrarC).place(x=210,

y=40)
Texto1 = Label(interfaz, fg='black', font=('Arial', 12), text='Código
G').place(x=15, y=100)
lstListbox1 = Listbox(interfaz, width=40, height=6)
scroll = Scrollbar(interfaz, command=lstListbox1)
lstListbox1.config(yscrollcommand=scroll.set)
lstListbox1.grid()
lstListbox1.place(x=15, y=125)
scroll.place(x=240, y=130)
Texto2 = Label(interfaz, fg='black', font=('Arial', 12), text='Línea
Actual Del Código').place(x=15, y=225)
lstListbox2 = Listbox(interfaz, width=40, height=1)
lstListbox2.grid()
lstListbox2.place(x=15, y=250)
'''#########################################################################
##########################################'''
'''##########################################Configuración Imagen
PNG##########################################'''
'''#########################################################################
##########################################'''
canvas2 = Canvas(interfaz, width=250, height=270, bg='#f0f0f0')
canvas2.pack()
canvas2.place(x=10, y=280)
canvas9 = Canvas(interfaz, width=240, height=165, bg='#232924')
```

```
canvas9.pack()
canvas9.place(x=15, y=380)
canvas10 = Canvas(interfaz, width=230, height=155, bg='#f0f0f0')
canvas10.pack()
canvas10.place(x=20, y=385)
Texto3 = Label(interfaz, fg='black', font=('Arial', 18), text='Control
Imagen PNG').place(x=20, y=280)
Button(interfaz, fg='white', text='Convertir\nGcode', font=('Arial', 10),
width=10, height=2, background='green',
       command=ImageToGcode).place(x=20, y=310)
Button(interfaz, fg='white', text='Cargar\nDibujo', font=('Arial', 10),
width=10, height=2, background='red',
       command=CargarD).place(x=115, y=310)
Button(interfaz, text='Borrar', font=('Arial', 10), width=5, height=2,
background='gray', command=BorrarI).place(x=210,

y=310)
Texto4 = Label(interfaz, fg='black', font=('Arial', 12), text='Imagen
PNG').place(x=15, y=355)

'''#########################################################################
##############################################'''
'''##############################################Configuración
Manual#####################################################'''
'''#########################################################################
##############################################'''
canvas3 = Canvas(interfaz, width=372, height=540, bg='#f0f0f0')
canvas3.pack()
canvas3.place(x=275, y=10)
'''#########################################################################
##############################################'''
'''###################################Configuración Herramienta de
velocidad#################################################'''
'''#########################################################################
##############################################'''
canvas4 = Canvas(interfaz, width=221, height=193, bg='#138D75')
canvas4.pack()
canvas4.place(x=283, y=15)
V1 = IntVar()
V2 = IntVar()
V3 = IntVar()
Texto5 = Label(interfaz, bg='#138D75', font=('Arial', 13),
text='Herramienta de Velocidad').place(x=295, y=20)
VelocidadX = Scale(interfaz, bg='#00FF0C', from_=0, to=100, variable=V1,
orient=HORIZONTAL,
                   length=150)
VelocidadX.place(x=340, y=50)
VelocidadX.set(20)
VelocidadY = Scale(interfaz, bg='#00FF0C', from_=0, to=100, variable=V2,
orient=HORIZONTAL,
                   length=150)
VelocidadY.place(x=340, y=105)
VelocidadY.set(20)
VelocidadZ = Scale(interfaz, bg='#00FF0C', from_=0, to=100, variable=V3,
orient=HORIZONTAL,
                   length=150)
VelocidadZ.place(x=340, y=160)
```

```
VelocidadZ.set(20)
Button(interfaz, fg='white', text='X = ', font=('Arial', 12), width=4,
height=1, background='green',
       command=VelX).place(x=290, y=55)
Button(interfaz, fg='white', text='Y = ', font=('Arial', 12), width=4,
height=1, background='green',
       command=VelY).place(x=290, y=110)
Button(interfaz, fg='white', text='Z = ', font=('Arial', 12), width=4,
height=1, background='green',
       command=VelZ).place(x=290, y=160)
'''#################################################################
##########################################'''
'''#################################Configuración Herramienta de
Coordenadas#########################################'''
'''#################################################################
##########################################'''
canvas5 = Canvas(interfaz, width=357, height=98, bg='#138D75')
canvas5.pack()
canvas5.place(x=283, y=215)
Texto9 = Label(interfaz, bg='#138D75', font=('Arial', 13),
text='Herramienta de Coordenadas').place(x=350, y=220)
Button(interfaz, fg='white', text='X+', font=('Arial', 10), width=6,
height=1, background='#000000',
       command=XA).place(x=293, y=245)
Button(interfaz, fg='white', text='X-', font=('Arial', 10), width=6,
height=1, background='#2D2D2D',
       command=XB).place(x=293, y=280)
Button(interfaz, fg='white', text='Y+', font=('Arial', 10), width=6,
height=1, background='#000000',
       command=YA).place(x=364, y=245)
Button(interfaz, fg='white', text='Y-', font=('Arial', 10), width=6,
height=1, background='#2D2D2D',
       command=YB).place(x=364, y=280)
Button(interfaz, fg='white', text='Z+', font=('Arial', 10), width=6,
height=1, background='#000000',
       command=ZA).place(x=435, y=245)
Button(interfaz, fg='white', text='Z-', font=('Arial', 10), width=6,
height=1, background='#2D2D2D',
       command=ZB).place(x=435, y=280)
Button(interfaz, fg='white', text='XZ+', font=('Arial', 10), width=6,
height=1, background='#000000',
       command=XZA).place(x=506, y=245)
Button(interfaz, fg='white', text='XZ-', font=('Arial', 10), width=6,
height=1, background='#2D2D2D',
       command=XZB).place(x=506, y=280)
Button(interfaz, fg='white', text='YZ+', font=('Arial', 10), width=6,
height=1, background='#000000',
       command=YZA).place(x=577, y=245)
Button(interfaz, fg='white', text='YZ-', font=('Arial', 10), width=6,
height=1, background='#2D2D2D',
       command=YZB).place(x=577, y=280)
'''#################################################################
##########################################'''
'''#########################################Configuración
Longitudes#########################################'''
'''#################################################################
##########################################'''
```

```
canvas6 = Canvas(interfaz, width=130, height=93, bg='#138D75')
canvas6.pack()
canvas6.place(x=510, y=15)
Texto10 = Label(interfaz, bg='#138D75', font=('Arial', 13),
text='Longitudes').place(x=535, y=17)
Texto11 = Label(interfaz, fg='black', bg='#138D75', font=('Arial', 12),
text='X =').place(x=520, y=40)
lstListbox3 = Listbox(interfaz, width=13, height=1)
lstListbox3.place(x=550, y=40)
Texto12 = Label(interfaz, fg='black', bg='#138D75', font=('Arial', 12),
text='Y =').place(x=520, y=60)
lstListbox4 = Listbox(interfaz, width=13, height=1)
lstListbox4.place(x=550, y=62)
Texto13 = Label(interfaz, fg='black', bg='#138D75', font=('Arial', 12),
text='Z =').place(x=520, y=80)
lstListbox5 = Listbox(interfaz, width=13, height=1)
lstListbox5.place(x=550, y=84)
'''##########################################################################
##########################################'''
'''#####################################Configuración Herramienta de
Origenes#########################################'''
'''##########################################################################
##########################################'''
canvas7 = Canvas(interfaz, width=130, height=93, bg='#138D75')
canvas7.pack()
canvas7.place(x=510, y=115)
Texto14 = Label(interfaz, bg='#138D75', font=('Arial', 13),
text='Origen').place(x=550, y=117)
Button(interfaz, fg='white', text='Zero X', font=('Arial', 10), width=6,
height=1, background='green',
       command=OX).place(x=516, y=142)
Button(interfaz, fg='white', text='Zero Y', font=('Arial', 10), width=6,
height=1, background='green',
       command=OY).place(x=516, y=175)
Button(interfaz, fg='white', text='Zero Z', font=('Arial', 10), width=6,
height=1, background='green',
       command=OZ).place(x=580, y=142)
Button(interfaz, fg='white', text='Zero', font=('Arial', 10), width=6,
height=1, background='green',
       command=Origen).place(x=580, y=175)
'''##########################################################################
##########################################'''
'''#########################################Configuración del
Desarrollo#######################################'''
'''##########################################################################
##########################################'''
canvas8 = Canvas(interfaz, width=357, height=225, bg='#138D75')
canvas8.pack()
canvas8.place(x=283, y=320)
Texto15 = Label(interfaz, bg='#138D75', font=('Arial', 13),
text='Desarrollo').place(x=412, y=322)
canvas11 = Canvas(interfaz, width=337, height=190, bg='#232924')
canvas11.pack()
canvas11.place(x=293, y=345)
canvas12 = Canvas(interfaz, width=317, height=170, bg='#f0f0f0')
canvas12.pack()
canvas12.place(x=303, y=355)
```

```
puertoRS = Comunicacion()
p = Paint()
interfaz.mainloop()
```

# Anexo 7

En este apartado se encuentra el código realizado para el control del Arduino.

```
/*En Este Programa de Arduino, se hace el control de un sistema de robot
cartesiano.
Este archivo es realizado por Irwin Saidh Jiménez Serratos, para el
proyecto de
Residencia profesional/Tesis
El Documento está basado en el ejemplo de la librería que se muestra a
continuación:
*LINEAR KINEMATICS
*Download the STEPPER DRIVER library as you already have the A4988 and
DRV8825
*Drive libraries available at: https://github.com/laurb9/StepperDriver
*Before using the example, make sure that the stepper motor library is
installed.
*Example by Leandro Lima / HeavyTech @2019*/

#include<A4988.h>
#include<MultiDriver.h>
#include<SyncDriver.h>
#include<gcode.h>

/*Definimos las Constantes a utilizar en la configuración de este*/
#define NUMCOMMANDS 2
#define MOTOR_STEPS 200
#define STEPS_MM 80
#define MOTOR_X_RPM 120
#define MOTOR_Y_RPM 120
#define MOTOR_Z_RPM 120

/*Definimos las Motores de los motores XYZ*/
#define DIR_X 5
#define STEP_X 2
#define DIR_Y 6
#define STEP_Y 3
#define DIR_Z 7
#define STEP_Z 4
#define ENA 8
#define MICROSTEPS 16

/*Configuración de los Driver A4988*/
A4988 stepperX(MOTOR_STEPS, DIR_X, STEP_X, ENA);
A4988 stepperY(MOTOR_STEPS, DIR_Y, STEP_Y, ENA);
A4988 stepperZ(MOTOR_STEPS, DIR_Z, STEP_Z, ENA);
MultiDriver controller(stepperX, stepperY, stepperZ);

/*Se definen las funciones Vacías*/
void homing();
void moviment();
void gotoLocation();
```

```
/*Creación de Comandos para el código G*/
commandscallback commands[NUMCOMMANDS] = {{"G1",homing}, {"G0",
moviment}};
gcode Commands(NUMCOMMANDS,commands);

/*Variables del tipo doble para Los 3 ejes*/
double X;
double Y;
double Z;

/*Iniciamos la funcion de configuración*/
void setup()
{
  Commands.begin();
  stepperX.begin(MOTOR_X_RPM, MICROSTEPS);
  stepperY.begin(MOTOR_Y_RPM, MICROSTEPS);
  stepperZ.begin(MOTOR_Z_RPM, MICROSTEPS);
  stepperX.setEnableActiveState(LOW);
  stepperY.setEnableActiveState(LOW);
  stepperZ.setEnableActiveState(LOW);
}
/*Iniciamos la funcion que se repite constantemente*/
void loop()
{
  Commands.available();
}

/*Iniciamos la funcion de casa*/
void homing()
{
  // code to home machine
}

/*Iniciamos la funcion de mover los motores a la posición deseada*/
void gotoLocation(double x,double y, double z)
{
  int stepsx = (x - X)*STEPS_MM; // Variación de la distancia X
  int stepsy = (y - Y)*STEPS_MM; // Variación de la distancia Y
  int stepsz = (z - Z)*STEPS_MM; //Variación de la distancia Z
  stepperX.enable();// ENABLE MOTOR X
  stepperY.enable();
  stepperZ.enable();
  controller.move(stepsx, stepsy, stepsz); //Se envían al driver las
posiciones deseadas
  X = x; // Ultima Posición del Set
  Y = y;
  Z = z;
  stepperX.disable();// DISABLE MOTOR Y
  stepperY.disable();
  stepperZ.disable();
  Commands.comment("X:" + String(x) + "; Y:" +String(y) + "; Z:"
+String(z)); // DEBUG SERIAL
}
/*Iniciamos la funcion de configuraciones de los motores a la posición
deseada*/
void moviment(){
    double newXValue = X;
```

```
    double newYValue = Y;
    double newZValue = Z;
    if(Commands.availableValue('X')) // AÑADIDO parámetro X en Go
      newXValue = Commands.GetValue('X');
    if(Commands.availableValue('Y')) // AÑADIDO parámetro Y en Go
      newYValue = Commands.GetValue('Y');
    if(Commands.availableValue('Z')) // AÑADIDO parámetro Z en Go
      newZValue = Commands.GetValue('Z');

    gotoLocation(newXValue, newYValue, newZValue);
}
```

# Anexo 8

Tecnológico Nacional de México
Instituto Tecnológico Superior de Teziutlán

**CARTA DE AUTORIZACIÓN DEL(LA) AUTOR(A) PARA LA CONSULTA Y PUBLICACIÓN ELECTRÓNICA DEL TRABAJO DE INVESTIGACIÓN**

El que suscribe:

| **IRWIN SAIDH** | **JIMÉNEZ** | **SERRATOS** |
|---|---|---|

| | |
|---|---|
| Con Número de Control | **17TE0594** |
| Perteneciente al Programa Educativo | **INGENIERÍA MECATRÓNICA** |

Por este conducto me permito informar que he dado mi autorización para la consulta y publicación electrónica del trabajo de investigación en los repositorios académicos.

| | |
|---|---|
| Registrado con el producto: | **TESIS** |

Cuyo Tema es:

**DISEÑO DE ALGORITMO DE CONTROL PARA ROBOT CARTESIANO PARA LA IMPRESIÓN DE SENSORES DE CARBONO**

Correspondiente al periodo:

**AGOSTO 2021-ABRIL 2022**

Y cuyo(a) director(a) de tesis es:

**M.S.C. ALFREDO CARRASCO ARAOZ**

ATENTAMENTE

IRWIN SAIDH JIMÉNEZ SERRATOS

Nombre y firma

Fecha de emisión: **04/04/2022**
c.c.p. Subdirección Académica

# Índice de figuras

# Índice de tablas