UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO DE UN BRAZO ROBOT Y VISIÓN ARTIFICIAL CONTROLADO POR PATRONES DE VOZ PARA MANIPULACIÓN DE OBJETOS EN LA EMPRESA AUTOMYN

TESIS

PRESENTADA POR:

KENNY BRIAN TORRES LUNA

FREDY OLIVER CALCINA PAREDES

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PUNO – PERÚ

2023

Reporte de similitud

NOMBRE DEL TRABAJO

DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO DE UN BRAZO ROBOT Y VISIÓN ARTIFICIAL CONTROLADO POR PATRONES

AUTOR

KENNY BRIAN, FREDY OLIVER TORRES LUNA, CALCINA PAREDES

RECUENTO DE PALABRAS

30367 Words

RECUENTO DE CARACTERES

162890 Characters

RECUENTO DE PÁGINAS

226 Pages

TAMAÑO DEL ARCHIVO

4.6MB

FECHA DE ENTREGA

Apr 17, 2023 8:46 AM GMT-5

FECHA DEL INFORME

Apr 17, 2023 8:48 AM GMT-5

19% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base c

- 16% Base de datos de Internet
- 8% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossr
- 14% Base de datos de trabajos entregados

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO - DECANO

Dr. Ivan Delgado Huayta
DECANO FIMEES
ASESOR

V°B°

Dra. Milder Zanabria Ortega
INGENIERA DE SISTEMAS
C I P 85001

DIRECTORA INVESTIGACIÓN
FIMEES

Resumen

DEDICATORIA

Dedico este trabajo de investigación a mi familia. Principalmente, a mi madre y padre quienes han estado a mi lado, me dieron su apoyo incondicional y su guía en la formación de mi carrera universitaria.

Kenny Brian Torres Luna

DEDICATORIA

El presente trabajo está dedicado a mis padres, hermanos y a todas las personas que me brindaron su apoyo incondicional durante el desarrollo del trabajo de investigación.

Fredy Oliver Calcina Paredes

AGRADECIMIENTOS

Estoy agradecido con todas las personas que me brindaron su apoyo en esta etapa de mi formación profesional. Mi familia y amigos quienes me motivaron en el proceso de este trabajo de investigación.

Agradezco a mis docentes, que por medio de sus enseñanzas obtuve conocimientos solidos en la escuela profesional y que hoy en día me permiten desarrollarme en el aspecto laboral.

Kenny Brian Torres Luna

Doy gracias a dios por brindarme salud, bienestar en todo momento de mi vida y la sabiduría para realizar mi trabajo de investigación.

A nuestros docentes de la escuela profesional de ingeniería electrónica que nos compartieron todo el conocimiento y experiencia en la formación profesional con la cual ha sido posible la culminación de proyecto de investigación.

A los miembros del jurado evaluador de nuestro proyecto, que aportaron con su conocimiento y experiencia, también por la comprensión y el apoyo brindado.

Fredy Oliver Calcina Paredes

ÍNDICE GENERAL

CAPITULO III

MATERIALES Y MÉTODOS

CAPITULO IV

RESULTADOS Y DISCUSIÓN

Área: Control e Instrumentación

Tema: Robótica e Inteligencia Artificial

FECHA DE SUSTENTACIÓN: 21 de abril del 2023

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

AFRI:	Asociación Francesa de Robótica Industrial
AV:	Asistente Virtual
DC:	Corriente Directa
EEPROM:	Memoria de Solo Lectura Programable y borrable Eléctricamente
GDL:	Grado de Libertad
GUI:	Interfaz Gráfica de Usuario
HSV:	Alto Saturación Valor
IDE:	Entorno de Desarrollo Integrado
PETG:	Tereftalato de Polietileno de Glicol
PIP:	Programa de Instalación Preferida
PLA:	Ácido Poliáctico
PWM:	Modulación Ancho de Pulso
RGB:	Rojo Verde Azul
RIA:	Asociación de Industrias Robóticas
SRAM:	Memoria de Acceso Aleatorio Estático

RESUMEN

En síntesis, el trabajo de investigación elabora un diseño e implementa un brazo robot y visión artificial, los cuales son controlados por patrones de voz para manipular objetos en la empresa Automyn. Por lo tanto, este proyecto permite resolver los problemas de demandas productivas y riesgos de las zonas peligrosas de esta empresa. De esta manera, se desarrolló el análisis matemático y el mecanismo del brazo robot, también se realizó la detección del objeto con estimación de posición y el asistente virtual. Por su parte, la metodología describe un enfoque cuantitativo, un diseño experimental e instrumentos de recolección de datos. Por último, las pruebas realizadas en la manipulación de objetos como el control de movimiento, agarre del objeto, aproximación visual y desarrollo de software tuvieron como resultados un error considerable con el parámetro deseado y el experimental.

Palabras Clave: Brazo robot, Visión artificial, Patrones de voz, Manipular objetos.

ABSTRACT

In synthesis, the investigation's work makes a design and implement an arm robot and artificial vision, which they're controlled by voice patterns to manipulate objects in the Automyn company. Therefore, this project allows to solve the productive demands problems and dangerous zones of the risks in this company. So, it developed the mathematic analysis and the mechanism of arm robot, also it realized the object's detection with position's estimation and virtual assistant. On the other hand, the methodology describes a quantitative approach, an experimental design and data collections' instruments. Finally, the tests realized in the object's manipulation like the movement's control, object's handgrip, visual approximation and software development had such as results a considerable error with the parameter wanted and the experimental.

Keywords: Arm Robot, Artificial Vision, Voice Patterns, Manipulate Objects.

CAPITULO I

INTRODUCCIÓN

La investigación realizada consiste en el diseño, construcción y control de un sistema de brazo robótico con inteligencia artificial dedicado a la manipulación de objetos. Se abarcaron temas relacionados como el diseño mecánico, electrónico y el desarrollo de software con el fin de automatizar el sistema.

El interés principal del trabajo es que pretende responder a la necesidad de generar diseños propios y originales de brazo robótico con inteligencia artificial que puedan ser empleados en empresas como AUTOMYM y otros, ya que estos son necesarios para la realización de sus montajes en menos tiempo, por ello necesitan un mecanismo que realice un movimiento fluido y que se pueda implementar con materiales de la región. Asimismo, este prototipo será un elemento adicional para los trabajadores que realizan montajes de tableros de control y así obtener mejores resultados en la producción, mejorando el tiempo en que se realiza el montaje, permitiendo al personal de las empresas agilizar su trabajo.

La metodología de investigación a implementarse para este sistema está basada en un enfoque cuantitativo con diseño experimental y nivel descriptivo, la línea de investigación es la robótica e inteligencia artificial, además contiene un procedimiento que consiste en 4 etapas. En primer lugar, se encuentra la etapa de plan de procedimiento donde utilizando conocimientos teóricos se realizan marcos semánticos y diagramas de flujo para el desarrollo del sistema. La segunda etapa es el análisis matemático, en el área de la mecánica, electrónica y la visión por computadora. Usando softwares de simulación se determina posibles fallas. La tercera etapa es el análisis de la implementación de las

áreas mencionadas con el desarrollo de software. La última etapa es la unión de estos sistemas con el reconocimiento de voz por comandos.

La finalidad es dar a conocer la integración del sistema de visión artificial a la robótica, dicho sistema estará a su vez controlado por el reconocimiento de voz con la cual una persona interactúa a través de patrones o comandos. Para ello se planteó como objetivo general: Diseñar e implementar un brazo robot y visión artificial controlado por patrones de voz para la manipulación de objetos en la empresa Automyn. Como objetivos específicos planteamos el Diseño e implementación del análisis matemático y mecanismo de un brazo robot 5GDL con Gripper, la realización de la detección de objetos con estimación de posición para el sistema de visión artificial y el desarrollo de software de un asistente virtual para el control de voz por patrones.

La estructura del trabajo está organizada en cuatro capítulos. En el capítulo I se realiza el planteamiento del problema a través de la pregunta ¿Cómo realizar el diseño e implementación del prototipo de un brazo robot y visión artificial con los patrones de voz para la manipulación de objetos? En el capítulo II, se presenta la revisión de literatura, citando a los antecedentes y el marco teórico que sustentan el trabajo realizado. En el capítulo III, se presenta los materiales y métodos aplicados en la investigación. donde se realiza la contrastación necesaria de diversas fuentes de información que ayudan en la comprensión del desarrollo del proyecto. En el capítulo IV se hace referencia a los resultados y la discusión. que se obtuvo después de este análisis con los métodos y materiales utilizados.

1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.1 Descripción del Problema

La colaboración entre humano y máquina, es escasa en la empresa Automyn para la realización de sus montajes en menos tiempo, es necesario un mecanismo que realice un movimiento fluido y se pueda implementar con materiales de la región.

La producción de la empresa tiene una demanda baja por el tiempo que tarda un personal en el armado de tableros de control y eléctrico. La ciudad de Juliaca es competitiva en el ámbito de la elaboración de productos de grado industrial por la actividad de empresas industriales y mineras, en el cual se requieren tableros para su automatización.

Los riesgos dentro de la empresa como zonas peligrosas en donde se realizan trabajos de montaje, un personal tiende a sufrir accidentes de grado menor y problemas ergonómicos que disminuyen su capacidad de trabajo.

1.1.2 Formulación del Problema

1.1.2.1 Problema General

- ¿Cómo realizar el diseño e implementación del prototipo de un brazo robot y visión artificial con los patrones de voz para la manipulación de objetos?

1.1.2.2 Problemas Específicos

- ¿Cómo automatizar el proceso de recoger y soltar cualquier tipo de objeto?
- ¿Cómo desarrollar el sistema de visión artificial para detectar cualquier tipo de objeto en un espacio determinado?
- ¿Como determinar un control de patrones de voz para realizar la función de hombre maquina?

1.1.3 Justificación del Problema

La justificación del proyecto de investigación está desarrollada en tres puntos importantes: técnica, económica y social.

1.1.3.1 Justificación Técnica

El presente proyecto de investigación se realiza por la necesidad que tiene la empresa Automyn, ya que realiza trabajos minuciosos que requieren concentración, este prototipo será un elemento adicional para los trabajadores que realizan montajes de tableros de control y así obtener mejores resultados en la producción.

1.1.3.2 Justificación Económica

El diseño y la implementación del brazo robot y visión artificial controlado por patrones de voz permite un aumento en la producción en cuanto al montaje de tableros eléctricos y de control, mejorando el tiempo en que se realiza el montaje, permitiendo al personal de la empresa Automyn agilizar su trabajo

1.1.3.3 Justificación Social

La facilidad en el montaje de tableros eléctricos beneficia tanto al personal de trabajo como a la misma empresa, el mismo permitió un acceso rápido a herramientas y materiales en la mesa de trabajo siendo oportuna al momento en que se lo requiera. También ayuda al personal a prevenir riesgos como la ergonomía y accidentes.

1.1.4 Objetivos

1.1.4.1 Objetivo General

Diseñar e implementar un brazo robot y visión artificial controlado por patrones de voz para la manipulación de objetos en la empresa Automyn.

1.1.4.2 Objetivos Específicos

- Diseñar e implementar el análisis matemático y mecanismo de un brazo robot 5GDL con Gripper.

- Realizar la detección de objetos con estimación de posición para el sistema de visión artificial.
- Desarrollar el software de un asistente virtual para el control de voz por patrones.

1.1.5 Hipótesis

1.1.5.1 Hipótesis General

El diseño e implementación de un brazo robot y la visión artificial controlado por patrones de voz logrará realizar una manipulación colaborativa de objetos entre humano y máquina a través de una computadora.

1.1.5.2 Hipótesis Específicas

- El diseño del análisis matemático y la implementación de un brazo robot con Gripper realizará el control de movimiento y agarre del objeto.
- El desarrollo de un detector de objetos con estimación de posición logrará una aproximación visual de lo que queremos obtener en el plano coordenado del brazo robot.
- Es posible la ejecución de un asistente virtual para el control de programas por patrones de voz con el desarrollo de software en este proyecto de tesis.

La matriz de consistencia se puede observar en el Anexo 4: Documentos, matriz de consistencia.

CAPITULO II

REVISIÓN DE LITERATURA

2.1 ANTECEDENTES

En esta sección se presentan diferentes investigaciones desarrolladas en el campo internacional y nacional, que contienen una relación con el problema de investigación.

2.1.1 Antecedentes Internacionales

Bharath Sai et al. (2017) realiza una investigación sobre un robot humanoide que pueda simular la forma humana, su proyecto es capaz de detectar, reconocer diferentes caras u objetos y puede ser controlado a través de la voz. Los componentes usados son un Arduino MEGA y el módulo EasyVR, el control de usuario usa comandos de voz y se procesa por el módulo mencionado. La programación se desarrolló en Python y OpenCV para el procesamiento de imágenes, todo el sistema funciona en tiempo real.

De igual manera Medrano (2020) desarrolla un brazo robot con motores paso a paso para ejecutar funciones de escritura, en ello busca el funcionamiento autónomo para una trayectoria deseada y que realice la acción para lo que fue diseñada. Un microprocesador Arduino Mega controla el movimiento de los motores NEMA-17 con sus respectivos drivers y los sensores fin de carrera limitan su movimiento. El software grafico CorelDraw puede realizar diseños sobre la estructura del brazo con madera MDF de tres milímetros. Para evitar la fricción los engranajes y piñones están hechos con metacrilato de tres y cinco milímetros.

Así mismo Muñoz y Serrato (2018) diseñan un sistema de control de bajo costo para generar trayectorias y aplicarlo a un brazo robot llamado DM-5200 LABVOLT 2001, se observa las etapas del desarrollo en cómo se realizó el análisis de estado para el controlador y la estructura, luego el diseño del Hardware y Software para el control del

manipulador, como último el estudio cinemático para generar trayectoria automática. El resultado que se obtuvo fue un proceso iterativo de validación y permitió realizar su trayectoria, pero sobre todo con bajo costo.

Existen diversas aplicaciones que puede optar un brazo robot una de las más conocidas es la aplicación de pick and place, de acuerdo con Aparicio (2022) es posible la realización de manipulación de piezas lego con visión artificial donde estos tipos de brazos reciben el nombre de robots colaborativos, este proceso conlleva una calibración de cámara que fue posicionada arriba de los mecanismos y pruebas de procesamiento de imágenes con Merlic. El robot UR3e realiza la tarea de pick and place y que a través de protocolo TCP/IP recibe la orden para recoger piezas de distintas formas y colores. Se obtuvieron resultados como detección de megabloques con el software Merlic satisfactoriamente y con siguiendo un correcto funcionamiento tanto el robot como la cámara, los porcentajes de precisión en los métodos de clasificadores de piezas también fueron buenos.

Según Zhang et al. (2020) la tecnología de reconocimiento inteligente es un contenido de investigación muy importante en la inteligencia artificial y una parte importante en los modernos sistemas de manipulación inteligente. Por ello un método para el agarre de objetos es la detección de colores basado en OpenCV, la visión artificial tiene un rol principal que integra la estructura del manipulador de tres ejes, el reconocimiento visual, la tecnología de posicionamiento y los dispositivos del manipulador. Estos se utilizan para clasificar y transportar objetivos con diferentes características de color.

2.1.2 Antecedentes Nacionales

Bravo y Villegas (2017) realizaron una investigación sobre el diseño y la implementación de un brazo robot para manipular sustancias toxicas, su proyecto está

orientado a los laboratorios farmacéuticos con la asistencia de la visión artificial y redes neuronales en el cual tienen como objetivo diseñar el modelo mecánico y cinemático a través de planos con especificaciones técnicas, también está el procesamiento de imágenes y el control del brazo con la visión artificial con tele operación. Se tuvo unos resultados satisfactorios con los algoritmos inteligentes desarrollados logrando transformar las coordenadas de las articulaciones correspondientes a las extremidades superiores del operador, en la posición y orientación del efector final.

De igual manera Leon et al. (2020) usa la visión artificial para detectar y extracción de muestras falladas, su estudio es experimental analítico y usa un brazo robot que reconoce el color. Ellos usaron el algoritmo matemático CBIR y la teoría RGB para la identificación del color. Como resultados tuvieron una disminución en costos y esto mejoro la optimización en el control de calidad, el retardo de detección de color tuvo una notoria diferencia con respecto al ojo humano y la visión de la máquina de 17 a 0.33 segundos, también se concluye que para cada imagen se obtiene una variación en los colores.

Según Román (2021) es posible controlar mediante la voz mecanismos y que esto mejorara las máquinas de gran envergadura, su proyecto para controlar el movimiento en las piezas de un tablero de ajedrez está basado en una aplicación móvil que reconoce la voz. Este procedimiento usa el método Design Thinking para reconocer el contexto, VDI-2206 para el desarrollo del sistema mecatrónico y su respectivo análisis, VDI-2225 para la evaluación económica de los conceptos de solución. Se tuvieron resultados de un 38% de error en la detección de voz, el análisis del movimiento de piezas fue satisfactorio con el desarrollo de matrices y la matemática, eso demostró que el movimiento de la estructura fija es ideal para sus segmentos.

De acuerdo con Romero (2019) la implementación de brazos robots son posibles con un presupuesto bajo y pueden ser automatizados con teorías de control. Este procedimiento se basa en el análisis matemático como cinemática directa, cinemática inversa, la velocidad y para el control del robot se desarrolló un APK, una variación con un control manual Joystick, también está el control por medio de una computadora de escritorio. Se usaron materiales como Arduino Mega, modulo PC9685, modulo bluetooth HC05 y otros de accesibilidad económica. Los resultados obtenidos fueron que se tuvo un margen de error bajo y la APK tiene la ventaja de ser portátil.

2.2 MARCO TEÓRICO

2.2.1 Robótica

Según la asociación de industrias robóticas (RIA), un robot es un manipulador multifuncional reprogramable, capaz, de mover materias, piezas, herramientas o dispositivos especiales, según trayectoria variable, programadas para realizar tareas diversas. (Barrientos et al., 1997)

2.2.1.1 Clasificación de los Robots

La AFRI califica a los robots como:

- **Tipo A:** Manipulador con control manual o telemando
- **Tipo B:** Manipulador automático con ciclo preajustado; regulación mediante fines de carrera o topes; control con PLC; accionamiento neumático, eléctrico o hidráulico.
- **Tipo C:** Robot manipulador programable con trayectoria continúa punto a punto. Este robot carece de conocimientos de entorno.
- **Tipo D:** Robot manipulador capaz de adquirir datos del entorno, adaptando sus movimientos en función de los obstáculos.

Según su generación los robots industriales se clasifican en:

- **1° Generación:** Repite la tarea programada secuencialmente. No toma en cuenta las posibles alteraciones de su entorno.
- **2° Generación:** Se dice que adquiere información limitada de su entorno y actúa en consecuencia. Puede localizar clasificar(visión) y detectar esfuerzo y adaptar sus movimientos en consecuencia.
- **3° Generación:** Su programación se realiza mediante el empleo de un lenguaje natural. Posee capacidad para planificación automática de tareas. (Barrientos et al., 1997)

2.2.1.2 Morfología de Robots

Los elementos que forman a un robot son:

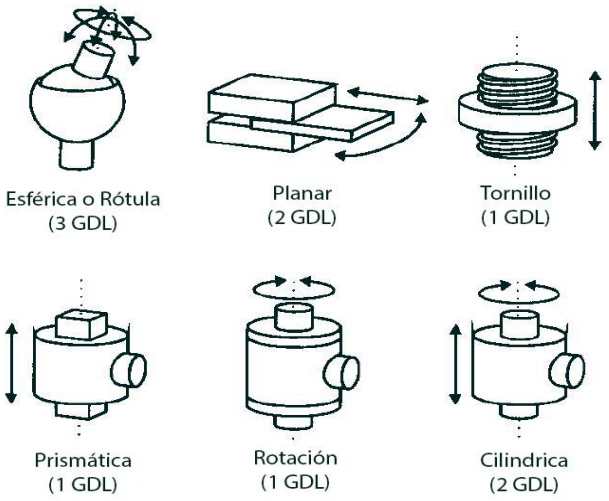
- Estructura mecánica
- Transmisiones
- Sistema de accionamiento
- Sistema sensorial
- Sistema de control
- Elementos terminales

2.2.1.3 Estructura Mecánica de un Robot

La mecánica de un robot que está unida mediante articulaciones denominadas elementos o eslabones que tiene un movimiento relativo cada dos eslabones consecutivos.

La construcción de un brazo robot físicamente guarda similitud con la anatomía del brazo humano. Se denomina grado de libertad (GDL) al movimiento independiente de cada articulación con respecto a la anterior. (Barrientos et al., 1997)

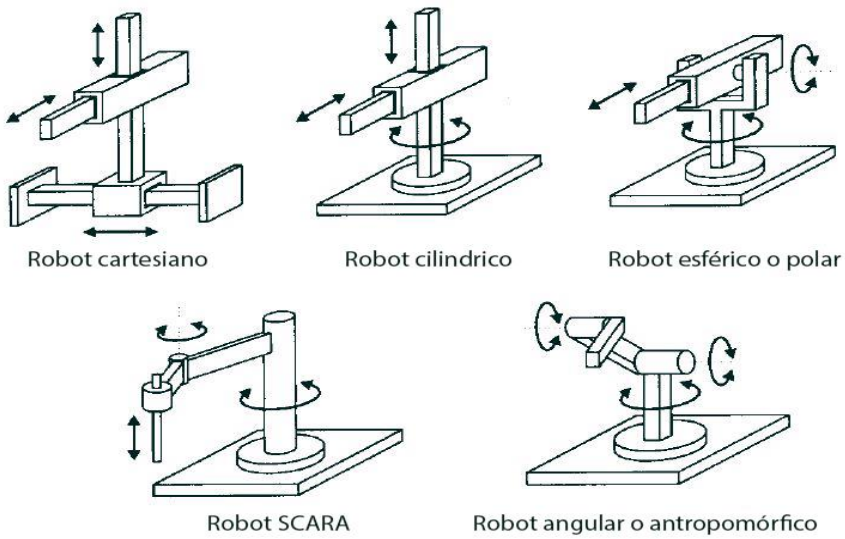
Figura 1: Distintos tipos de articulaciones para robots



Fuente: (Barrientos et al., 1997, pág. 17)

Dice que a las diferentes combinaciones de movimientos en las articulaciones de un robot se tiene diferentes configuraciones, también tiene características diferentes en diseño construcción y aplicación, pero las combinaciones más frecuentes se dan en las tres primeras articulaciones que determinan su posicionamiento en el espacio. (Barrientos et al., 1997)

Figura 2: Configuraciones más frecuentes en robots industriales



Fuente: (Barrientos et al., 1997, pág. 18)

2.2.1.4 Trasmisiones

Se dice que son elementos que se encargan de transmitir movimientos desde los actuadores hasta las articulaciones, a esto se incluyen los reductores que están encargados adaptar el par de arranque, la velocidad de cada actuador en los elementos del robot. (Barrientos y otros)

Tabla 1: Sistemas de trasmisión para robots

ENTRADA-SALIDA	DENOMINACIÓN	VENTAJAS	INCONVENIENTES
Circular-circular	Engranaje	Pares altos	Holgura
	Correa dentada	Distancia grade	-
	Cadena	Distancia grande	Ruido
	Paralelogramo	-	Giro limitado
	Cable	-	Deformabilidad
Circular-lineal	Tornillo sin fin	Poca holgura	Rozamiento
	Cremallera	Holgura media	Rozamiento
Lineal-circular	Para el articulado	-	Control difícil
	Cremallera	Holgura media	Rozamiento

Fuente: (Barrientos et al., 1997, pág. 20)

2.2.1.5 Reductores

Dice que en la robótica debido a la alta precisión y velocidad de posicionamiento los reductores deben ser utilizados de acuerdo a los requerimientos debido a la condición de funcionamiento restrictivo, las características que debe de tener un reductor se detallan en la tabla 2. (Barrientos y otros)

Tabla 2: Características de reductores para robótica

CARACTERÍSTICAS	VALORES TÍPICOS
Relación de reducción	50 – 300
Peso y tamaño	0.1 – 30Kg
Momento de inercia	10^{-4}Kgm^2
Velocidad de entrada máxima	6000 – 7000 rpm
Par de salida nominal	5700 Nm
Par de salida máxima	7900 Nm
Juego angular	0 – 2 “
Rigidez torsional	100 – 2000 Nm/rad
Rendimiento	85% -98%

Fuente: (Barrientos et al., 1997, pág. 22)

2.2.1.6 Accionamiento Directo

Generalmente utilizado en robots de accionamiento eléctrico consiste en conectar el eje del actuador directamente a la articulación sin utilizar un reductor de intermedio. Se utiliza en robots que requieren gran precisión y velocidad alta, esto debido a que los reductores tienen efectos negativos como juego angular, rozamiento y rigidez del accionador. (Barrientos et al., 1997)

El accionamiento directo tiene las siguientes ventajas:

- Posicionamiento rápido y preciso, pues se evitan los rozamientos y juegos de las trasmisiones y reductores.
- Tiene una mayor complejidad con más posibilidades de aplicación.
- Se eliminan los reductores lo que hace al sistema mecánico más simple.

2.2.1.7 Actuadores

Dice que la función principal de un actuador es generar movimiento en los elementos de un robot, la unidad de control da la orden que se va a ejecutar en cada

elemento. La robótica puede utilizar actuadores de tipo neumático, hidráulico y eléctrico. Las características a considerar son las siguientes: (Barrientos et al., 1997)

- Potencia.
- Controlabilidad
- Peso y volumen.
- Precisión.
- Velocidad.
- Mantenimiento.
- Coste.

2.2.1.8 Actuadores de Tipo Eléctrico

Se caracterizan por su sencillez, precisión y la facilidad de control, son los más usados en a la robótica industrial, entre ellos podemos distinguir de tres tipos las cuales son. (Barrientos et al., 1997)

Tabla 3: Clasificación de los actuadores eléctricos

TIPO DE MOTOR	VARIEDAD
Motor de corriente continua	Controlado por inducido
	Por excitación
Motor de corriente alterna	Síncronos
	Asíncronos
Motor paso a pasos	Motor de reclutamiento variable
	Motor hibrido
	Motor de imán permanente

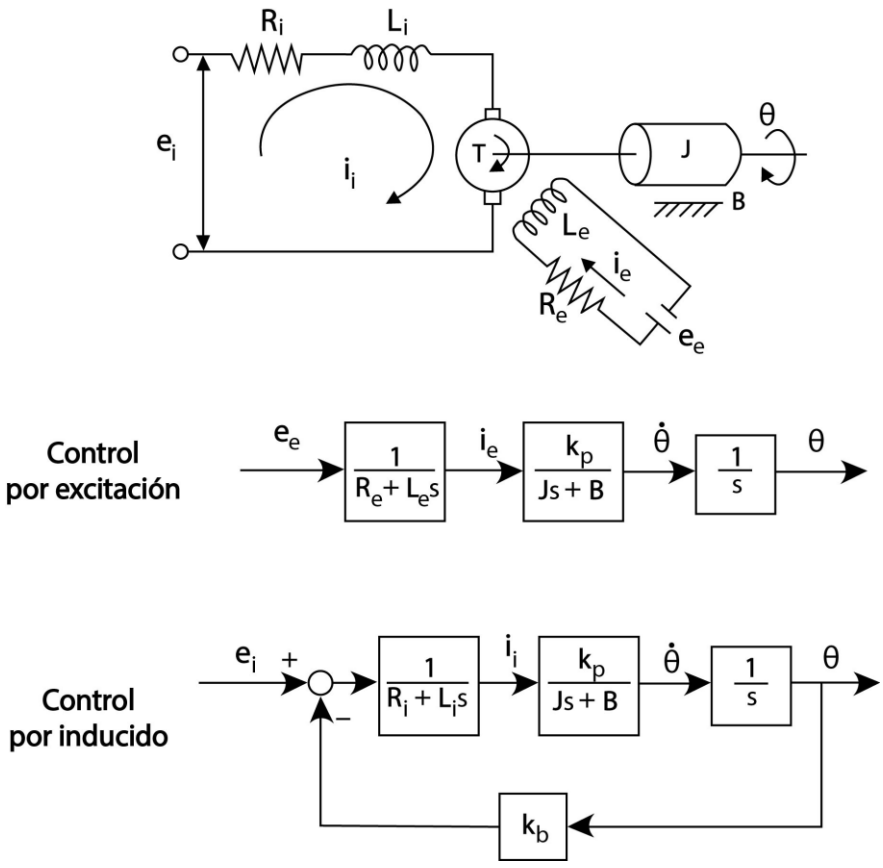
Fuente: (Barrientos et al., 1997, pág. 35)

2.2.1.8.1 Motor de Corriente Continua

Son fáciles de controlar por ese motivo son los más usados en la robótica, en ocasiones incluyen decodificadores (Encoder) lo que facilita su control. Tienen dos

devanados internos el inductor o devanado de excitación situado en el estator el cual crea un campo magnético de dirección fija, el inducido que está en el rotor su gira en el mismo sentido por la fuerza de Lorentz, recibe la alimentación de corriente continua del exterior a través del colector de delgas apoyados con unas escobillas de grafito. (Barrientos et al., 1997)

Figura 3: Motor DC esquema de funcionamiento de transferencia



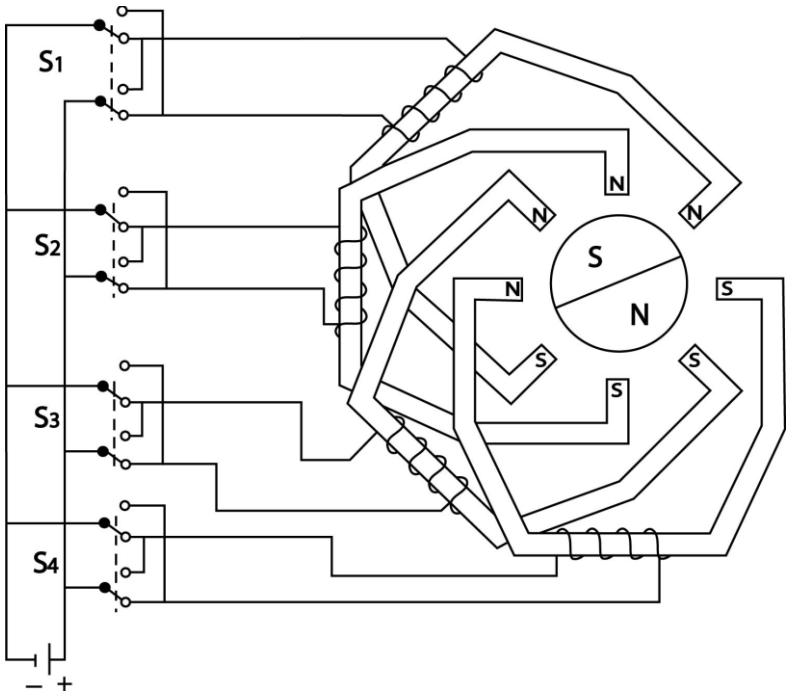
Fuente: (Barrientos et al., 1997, pág. 31)

2.2.1.8.2 Motor Paso a Paso

Establece que su funcionamiento se basa en convertir la energía eléctrica en energía mecánica y una información de posición. Formado por el rotor que tiene varios polos y los devanados del estator llamados fases. El denominado paso se da por la alimentación cíclica en sus fases que establece el funcionamiento síncrono. Generalmente el estator tiene 6 polos salientes que forman tres fases cada una tiene dos bobinas

conectadas en serie y el rotor consta de 4 polos, ambos construidos de acero blando. (Fernández Aragón, 2011)

Figura 4: Esquema básico de funcionamiento de un motor paso a paso



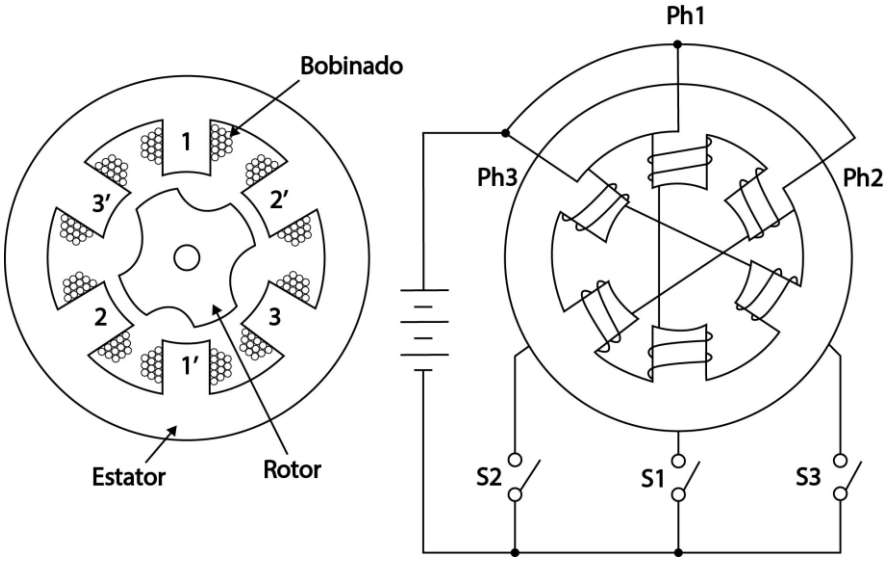
Fuente: (Barrientos et al., 1997, pág. 33)

2.2.2 Clasificación de Motores Paso a Paso

2.2.2.1 Motores de Reclutamiento Variable

Establece que la construcción del motor paso a paso de reclutamiento variable esta dado por un estator embobinado y un rotor dentado hecho de acero suave. el cual no requiere imanes permanentes, por tal motivo el motor se puede mover libremente. Para su funcionamiento se energizan los polos con corriente directa y de esa forma se establece una rotación que ocurre porque los dientes son atraídos hacia los polos del estator. Este tipo de motor se utiliza en proyectos en donde no se requiere un torque alto, también tiene deslizamiento de posición, lo cual indica que no puede ser utilizados para la aplicación industrial. (Fernández Aragón, 2011)

Figura 5: Sección de un motor paso a paso de reclutamiento variable

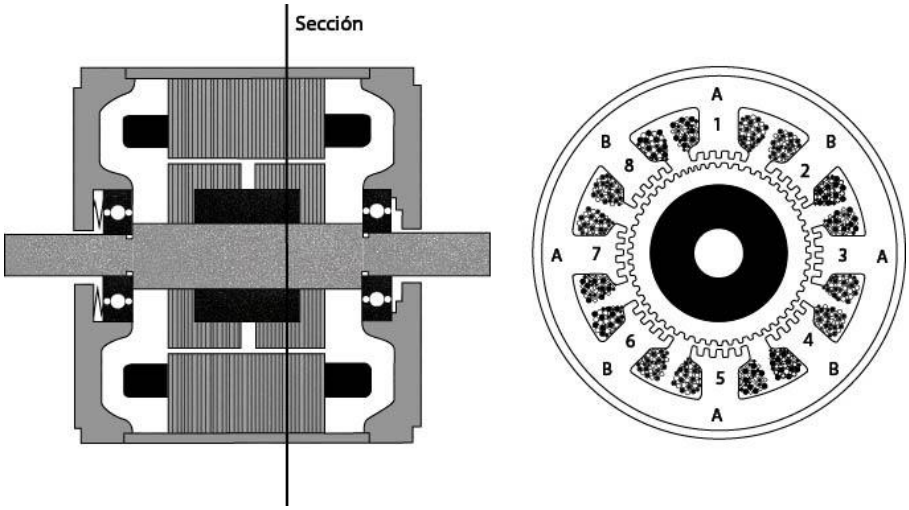


Fuente: (Fernández Aragón, 2011, pág. 13)

2.2.2.2 Motor Híbrido

Establece que su construcción es la multidentación en los polos del estator y para mejorar la orientación del campo magnético se le añadió imanes magnéticos en el rotor, lo cual tiene características dinámicas que lo distinguen de motores como reclutamiento variable e imanes permanentes. Tiene mejor desempeño en la velocidad y la resolución por tal motivo tiene un mayor costo. (Fernández Aragón, 2011)

Figura 6: Sección de un motor paso a paso hibrido



Fuente: (Fernández Aragón, 2011, pág. 14)

2.2.2.3 Motor de Imán Permanente

Establece que un motor paso a paso de imán permanente, tiene imanes en el rotor. Los ángulos de paso son grandes y es de baja velocidad, no presenta dientes en el rotor, pero están alternadas con polos norte-sur magnéticos en línea paralela con el eje como se indica en la figura 7. tales polos incrementan la intensidad del flujo magnético lo que aumenta el torque, todos los detalles antes mencionados hacen que este motor sea de bajo costo. (Barrientos et al., 1997)

Tabla 4: Relación de pasos para una vuelta

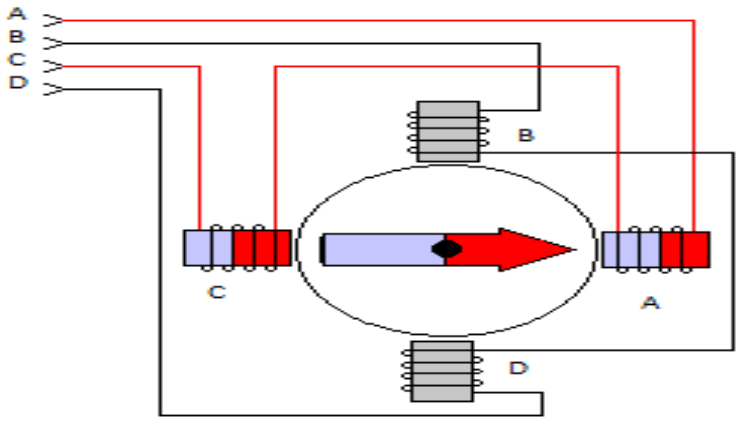
Valor de pasos	N° de pasos para una vuelta
18°	20
15°	24
9°	40
7.5°	48
3.6°	100
1.8°	200

Fuente: (Barrientos et al., 1997, pág. 34)

2.2.2.4 Motor Bipolar

Establece que un motor bipolar siempre tiene 4 cables, para hacer girar el eje de este motor se invierten los polos en su alimentación con una secuencia establecida y precisa, también para realizar su movimiento se necesita de dos puentes H. (Fernández Aragón, 2011)

Figura 7: Motor bipolar



Fuente: (*HTA 3D*, 2022) recuperado el 15 de enero de 2022 de https://www.hta3d.com/en/blog/Connect-Stepper-Motor

Secuencia de excitación de un motor bipolar.

Tabla 5: Secuencia de funcionamiento para un motor bipolar

PASO	BOBINA A	BOBINA B	BOBINA C	BOBINA D
1	+V	-V	+V	-V
2	+V	-V	-V	+V
3	-V	+V	-V	+V
4	-V	+V	+V	-V

Fuente: (Fernández Aragón, 2011, pág. 18)

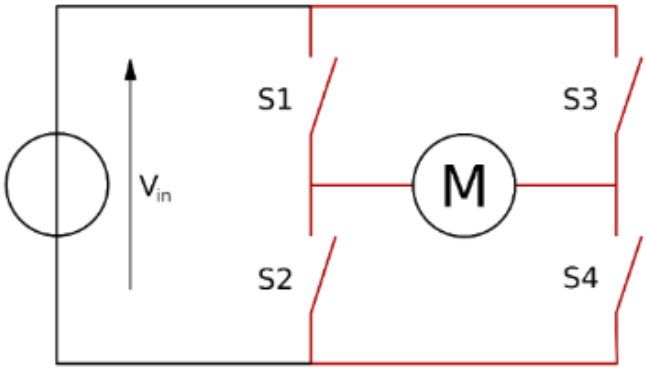
2.2.3 Controladores de Motores

2.2.3.1 Puente H

Establece que los puentes H son utilizados en la robótica porque permite a un motor DC girar en ambos sentidos, también son usados como convertidores de potencia y su construcción se puede realizar mediante circuitos integrados y componentes discreto como transistores o interruptores, su funcionamiento se puede ver en la figura 8, del que podemos decir, si S1 y S4 están cerrados, S2 y S3 estarán abiertos el motor girara en un

sentido, si S1 y S4 están abiertos, S2 y S3 están cerrados em motor gira en sentido contrario al anterior. (*Wikipedia*, 2022a)

Figura 8: Circuito eléctrico de un puente h



Fuente: (*Wikipedia*, 2022a) recuperado el 15 de enero de 2022 de https://es.wikipedia.org/wiki/Puente_H_(electr%C3%B3nica)

Establece que las acciones realizadas por el puente H pueden ser invertir el giro del motor, frenar el motor haciendo un corto circuito en las bobinas o frenar con su propia inercia, la tabla 6 muestra las diferentes acciones que puede realizar el puente H y también se puede utilizar como generador de voltaje AC para ello se toman diferentes frecuencias tomadas por el PWM que depende del Duty-Cycle. (*Wikipedia*, 2022a)

Tabla 6: Secuencia de excitación de transistores para el movimiento de motor

S1	S2	S3	S4	Resultado
1	0	0	1	El motor gira en avance
0	1	1	0	El motor gira en retroceso
0	0	0	0	El motor se detiene bajo su inercia
1	0	1	0	El motor se detiene
0	1	0	1	El motor se detiene
1	1	0	0	Cortocircuito
0	0	1	1	Cortocircuito
1	1	1	1	Cortocircuito

Fuente: (*Wikipedia*, 2022a) recuperado el 15 de enero de 2022 de https://es.wikipedia.org/wiki/Puente_H_(electr%C3%B3nica)

2.2.3.2 Controladores para Motores Paso a Paso

Existen una variedad amplia de controladores para motores paso a paso pero los controladores para utilizarlos en el presente proyecto son de carácter industrial de dos fases tales como DM320T y DM542T.

2.2.3.2.1 DM320T

Establece que es un controlador digital para motores paso a paso que muestra un diseño simple y de fácil configuración, puede alimentar motores de hasta 2 y 4 fases con un bajo calentamiento y un par optimo, también es capaz de disminuir el ruido del motor. Funciona con voltajes de 10 – 30 V CC con una corriente máxima de 2.2 A, ideal para motore NEMA 11, 14, 16, 17. (Road Zhongke, 2017a)

2.2.3.2.2 Aplicaciones del driver DM320T

Dice que el controlador puede alimentar motores paso a paso híbridos tales como NEMA 11, 14, 16, 17 que tengan 2 fases (1.8°) o 4 fases (0.9°). los cuales pueden ser usados en CNC, automatización, mesas X-Y, máquinas de grabados, máquinas de etiquetado, cortadoras laser, molinos, cortadores de plasma, etc. (Road Zhongke, 2017a)

2.2.3.2.3 DM542T

Su desarrollo se realizó con el algoritmo de control DSP, es un controlador digital que logro suavidad y un óptimo par de arranque con una inestabilidad de rango medio nulo. Tiene una función de auto identificación y auto configuración de parámetros, también posee una configuración de modos para deferentes motores el cual causa menor ruido, menor calentamiento con movimientos suaves. (Road Zhongke, 2017b)

2.2.3.2.4 Aplicaciones del driver DM542T

Su utilización se puede dar para el funcionamiento ideal de motores paso a paso NEMA 17, 23 y otros, los cuales pueden ser usados en proyectos como mesas X – Y,

máquinas de grabado, máquinas de etiquetado, cortadoras laser, etc. Adaptado para un bajo calentamiento, bajo ruido, alta velocidad y precisión. (Road Zhongke, 2017b)

2.2.4 Sensores Internos

Postula que los sensores internos son de utilidad en un robot para conseguir la información relacionado con su estado, lo cual es necesario para que el robot realice las tareas con precisión, velocidad e inteligencia. (Fu et al., 1988)

Tabla 7: Tipos de sensores internos de robots

PRESENCIA	Inductivo	-
	Capacitivo	
	Efecto hall	
	Célula Reed	
	Óptico	
	Ultrasonido	
	Contacto	
POSICIÓN	Analógicos	Potenciómetro
		Resolver
		Síncrono
		Inductosyn
		LVDT
	Digitales	Encoders absolutos
		Encoders incrementales
		Regla óptica
VELOCIDAD	Tacogeneratriz	-

Fuente: (Barrientos et al., 1997, pág. 36)

2.2.5 Elementos Terminales

Establece que son elementos que interactúan con el entorno del robot, estos elementos pueden ser herramientas o elementos de aprehensión. Los elementos terminales son llamados también como efector final. (Barrientos et al., 1997)

Tabla 8: Herramientas terminales para robots

TIPO DE HERRAMIENTA	COMENTARIOS
Pinza soldadura por punto	Dos electrodos que se cierran sobre la pinza a soldar.
Soplete soldadura al arco	Aportan el flujo de electrodo que se funde.
Cucharon para colada	Para trabajos de fundición
Atornillador	Suelen incluir la alimentación de tornillos.
Fresa - Lija	Para perfilar, eliminar rebabas, pulir, etc.
Pistola de pintura	Para pulverización de la pintura.
Cañón laser	Para corte de materiales, soldadura o inspección.
Cañón de agua a presión	Para corte de materiales.

Fuente: (Barrientos et al., 1997, pág. 46)

2.2.6 Microcontroladores

2.2.6.1 Tarjeta Arduino

Es una placa de hardware libre, flexible y fácil de utilizar, el cual se puede crear como microordenadores de una sola placa. Se le puede atribuir diferentes tipos de uso ya que cuenta con puertos GPO analógicos y digitales para realizar el control de diferentes dispositivos electrónicos. (Guerra Carmenate, 2021)

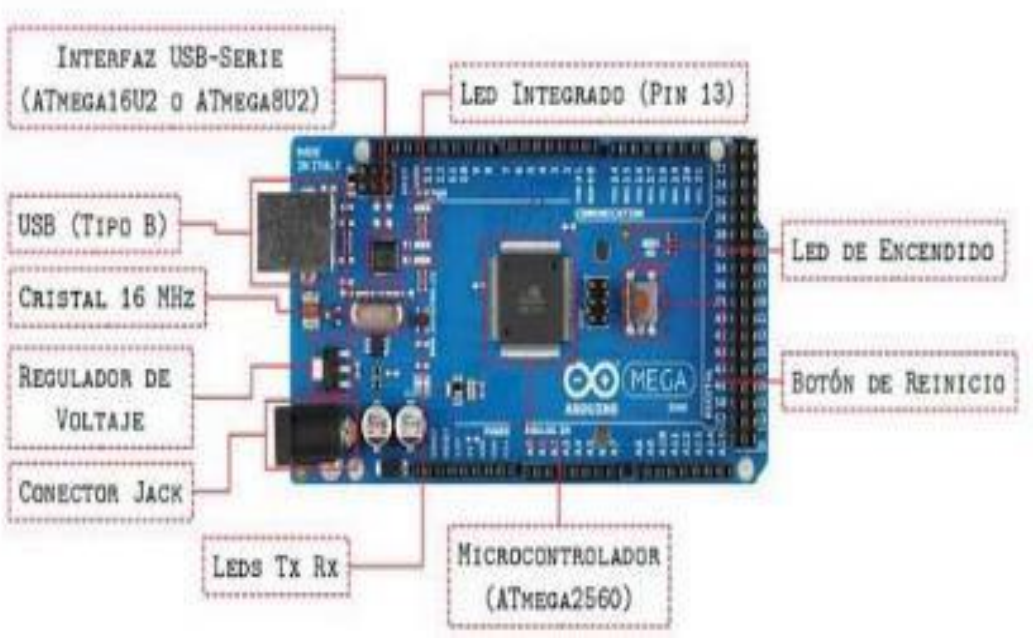
2.2.6.1.1 Arduino Mega 2560

Esta placa pertenece a la familia de placas Arduino, la placa esta desarrollada en base al microcontrolador ATmega2560, el cual se caracteriza por lo siguiente. (*Xataca*, 2021)

- 256 kB de memoria FLASH (espacio disponible para almacenar el programa o sketch)
- 8 kB de memoria SRAM (es donde se crean las variables declaradas en el programa)

- 4 kB de memoria EEPROM (permite almacenar datos que se conserven, aunque se reinició o falle la alimentación)
- Frecuencia de CPU Máxima: 16MHz (esto lo explico más adelante).
- Voltaje de Operación máximo: 6.0V (aunque se recomienda no sobrepasar los 5V)

Figura 9: Arduino mega 2560



Fuente: (Guerra Carmenate, 2021) recuperado el 28 de diciembre de 2021 de https://programarfacil.com/blog/arduino-blog/arduino-mega-2560/

2.2.7 Fuentes de Alimentación

Son tarjetas elaboradas, generalmente transforman energía de corriente alterna a corriente continua, que proporcionan energía de voltaje menores como las de 5, 9, 12, 15, 18, 24 voltios en corriente directa, pueden proporcionar configuraciones de fuentes del voltaje complementarios. (*STEPPERONLINE*, 2019c)

Figura 10: Fuente de alimentación de corriente continua

Fuente: (*STEPPERONLINE*, 2019c) recuperado el 28 de diciembre de 2021 de https://www.omc-stepperonline.com/350w-24v-14-6a-115-230v-switching-power-supply-stepper-motor-cnc-router-kits-s-350-24

2.2.8 Visión Artificial

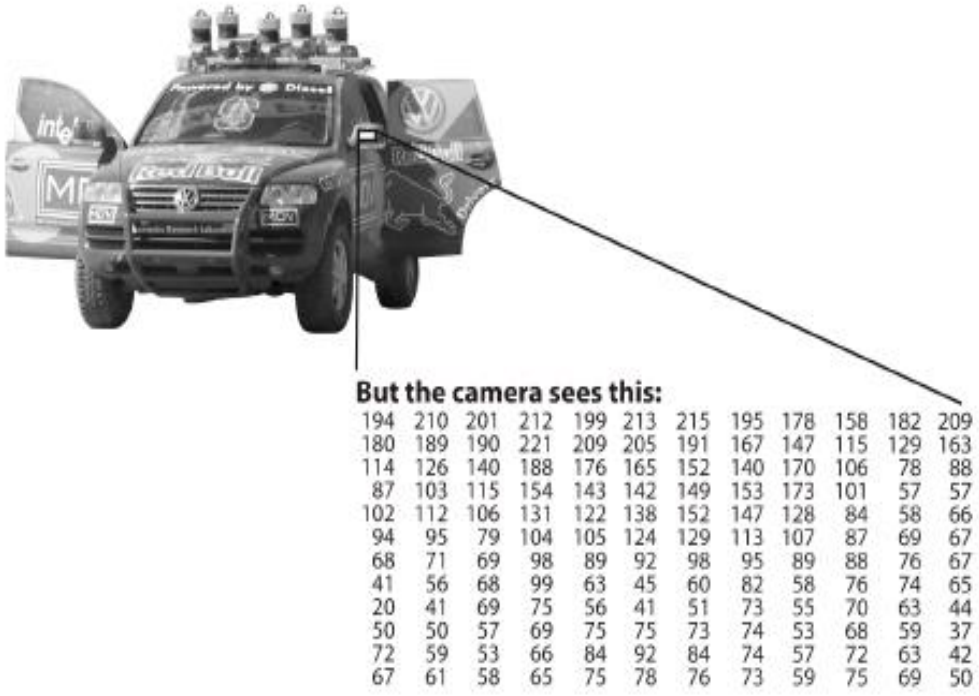
Establece que los datos obtenidos a partir de una cámara se pueden transformar en una representación o una decisión, esto para lograr un objetivo en concreto. La información contextual de entrada se puede dar por el lugar en donde está ubicado la cámara o indicar la posición de un objeto. Un ejemplo de decisión seria contar personas, objetos o animales en un lugar. Una representación también puede ser convertir una imagen en escala de grises. (Kaehler & Bradski, 2016)

En la visión por computador se puede reconstruir un modelo 3D a partir de imágenes superpuestas entre sí, para ello es necesario utilizar técnicas de desarrollo como las matemáticas. Estas técnicas se desarrollan en la física en áreas como la radiometría, óptica y diseño de sensores, también está el desarrollo en los gráficos por ordenadores. En ambos procesos se detalla el movimiento y animación de objetos, el reflejo de la luz en su superficie, la dispersión de la luz en la atmosfera, la refracción de que sucedes en la lente de la cámara y la proyección de la imagen en un plano. (Szeliski, 2022)

2.2.9 Sistema de Visión Artificial

Establece que en dicho sistema no hay reconocimiento de patrones incorporados, no se da el control automático del enfoque ni la apertura, este proceso recibe una cuadricula de números de la cámara y del disco para el ordenador. En la figura 11 se visualiza un automóvil y observamos el espejo lateral que está ubicado al lado del conductor. El ordenador solo ve una cuadricula de números que forma el espejo lateral del auto que cual tiene un componente de ruido grande, lo que india la falta de información, con la información que brinda el ordenador nuestra tarea es convertir la cuadricula ruidosa en números para el espejo lateral de la percepción. (Vélez Serrano et al., 2003)

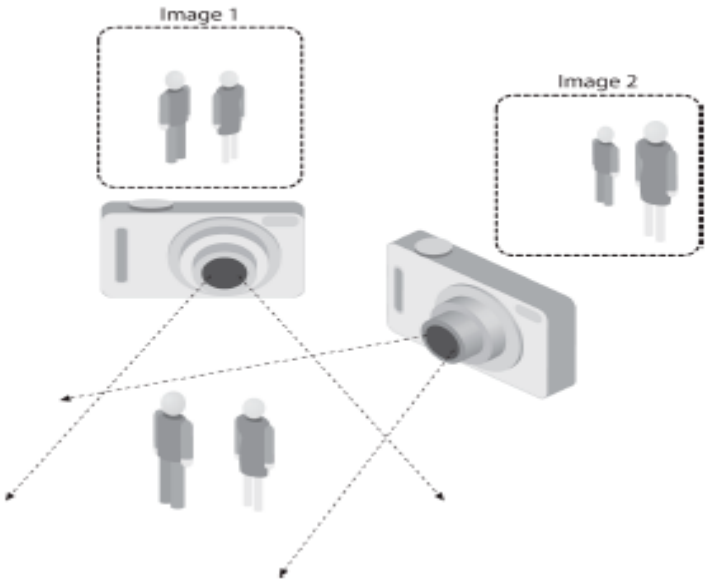
Figura 11: Vista de un ordenador



Fuente: (Kaehler & Bradski, 2016, pág. 4)

En la figura 12 se puede visualizar la vista de un ordenador, describe que el espejo lateral del coche es solo una cuadricula de números.

Figura 12: Diferentes vistas 2D



Fuente: (Kaehler & Bradski, 2016, pág. 4)

2.2.10 Adquisición de Imágenes

Establece que el proceso que pasa al dominio discreto y virtual informático es la captura y la digitalización que se le aplica a una imagen tomada del mundo físico. Pasado el proceso de digitalización la imagen bidimensional que está formada por varios elementos en conjunto denominados pixeles, el cual da información sobre las regiones de la imagen. En las imágenes de color esta información representa a la intensidad de cada una de las componentes en la base de color, como ejemplo podemos tomar el RGB. Por otra parte, en las imágenes blanco y negro la información se denota en el brillo. (Vélez Serrano et al., 2003)

2.2.10.1 Iluminación

Establece que la importancia de la iluminación para la detección, extracción de los objetos y las características de interés, la luz proporciona la escena adecuada para tomar la imagen el cual favorecerá en trabajos posteriores sobre sí misma. El bajo contraste, las reflexiones, especulares sombras y detalles son detalles que se debe de tener

en el sistema de visión industrial. Para ello existen tipos de iluminación tales como incandescente, fluorescente, led, laser y la fibra óptica. (Alegre Gutiérrez et al., 2016)

2.2.10.2 Cámara

Establece que este compuesto por un juego de lentes y el diafragma, que tiene la función de construir imágenes en un plano captados por el sensor que tiene elementos fotosensibles y la digitaliza, el cual es transmitido a la tarjeta de adquisición del procesador. Para su digitalización las cámaras mantienen un formato estándar de video en el caso de que sea analógico o proporcionan información en formato digital en el caso de cámaras digitales. (Alegre Gutiérrez et al., 2016)

2.2.10.3 Lentes

Establece que según el índice de Abbe la lente se caracteriza por tener índice de refracción, reflexión y dispersión. La función principal la tiene el índice de refracción el cual tiene la capacidad de reducir la velocidad de la luz cuando pasa por ella, otra definición que se le atribuye es el cociente entre la velocidad de la luz en el vacío y la velocidad de la luz que pasa por la lente. (Alegre Gutiérrez et al., 2016)

2.2.10.4 Óptica

Establece que en un conjunto de lentes de estructura compleja que se encuentran dispuestas en un cilindro mecánico con una rosca o bayoneta el cual permite la incorporación e intercambio en la cámara, todos estos elementos se encuentran dispuestas en el objetivo. (Alegre Gutiérrez et al., 2016)

2.2.10.5 Sensor

Establece que generalmente este componente se encuentra en cámaras digitales, equipos médicos y equipos de visión nocturna, que sirve como detector para capturar la información por la que está compuesta la imagen esto se consigue gracias a la atenuación de las ondas de luz. Está compuesta por un chip formado por millones de componentes

fotosensibles a la luz como fotodiodos o fototransistores, los cuales pueden capturar la luz o las radiaciones electromagnéticas. Los sensores más comunes son el sensor CCD, SuperCCD, CMOS y Foveon X3. (*Wikipedia*, 2021b)

2.2.10.6 Apertura

Establece que en la óptica determina el ángulo en el orificio por donde pasa la luz, el haz de rayo que atraviesa la apertura con un ángulo definido se enfoca en el plano de la imagen. (*Wikipedia*, 2021c)

2.2.10.7 Diafragma

Establece que es el diafragma es un disco o sistema de aletas el cual está dispuesto en el objetivo de una cámara, que provee la cantidad de luz necesaria al objetivo, limitando la cantidad de luz que llega a los medios fotosensibles en una cámara, este componente es generalmente ajustable. (*Wikipedia*, 2021a)

2.2.10.8 Enfoque

Establece que es la acción de coincidir o dirigir los rayos de luz que un objeto o varios reflejan, al cual se pretende fotografiar con el foco de la lente. El foco en las cámaras digitales está constituido por el sensor. (*EcuRed*, 2021)

2.2.11 Procesamiento de Imágenes

2.2.11.1 Umbralización de Imágenes

Se denomina así al proceso que permite transformar imágenes establecidas en niveles como gris o color en una imagen binaria, los cuales se etiquetaran con un valor distinto en cada pixel de fondo. A la técnica utilizada en este proceso se le denomina segmentación rápida, el cual este proceso es de bajo costo computacional que también puede ser realizado en tiempo real, en la captura puede utilizarse una computadora personal de propósito general. (Vélez Serrano et al., 2003)

La umbralización de truncamiento es otro método que solo alterna valores en la imagen, el cual establece un valor umbral para todos los pixeles con un valor mayor a él. En la elección del valor umbral se toma este valor en base a una serie de características de la imagen o también de acuerdo al resultado que se desea obtener, otro método bastante utilizado es el denominado método de Otsu. (Alegre Gutierrez et al., 2016)

2.2.11.1.1 Binarización

Estas imágenes suelen aparecer después de una operación de umbralización. Los filtros lineales se utilizan a menudo para mejorar las imágenes en escala de grises y en color, pero también se utiliza mucho para procesar imágenes binarias. (Alegre Gutiérrez et al., 2016)

El umbral se identifica como un valor de intensidad en los cuales se agrupan determinados pixeles que pertenecerán a un conjunto al cual se les denominara como blancos y los pixeles sobrantes se les asigna a otros subconjuntos al que se les denomina como negros. (Alegre Gutiérrez et al., 2016)

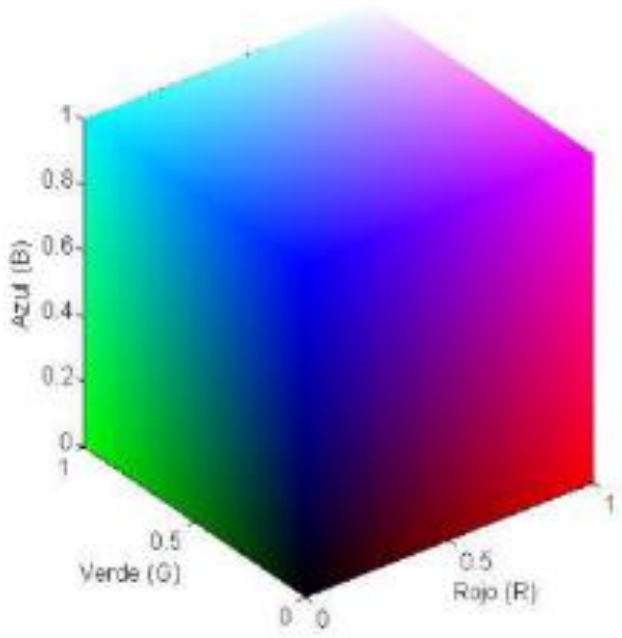
2.2.11.2 Espacios de Color

Permite describir y descomponer los colores en distintos canales los cuales están establecidas mediante un conjunto de fórmulas matemáticas. Los espacios de color que más se utilizan son el RGB puede utilizarse en periféricos como pantallas, cámaras y escáneres, otro modelo para el espacio de colores es el CMYK que es utilizado en impresoras. (Alegre Gutierrez et al., 2016)

2.2.11.2.1 Espacio Rgb

Tiene como componentes elementales a los colores rojo, verde y azul los cuales definen el espacio de color básico que está representada en forma de cubo en 3D del sensor, la variación de colores se da por las componentes pertenecientes a la superficie o también el interior del cubo. (Alegre Gutiérrez et al., 2016)

Figura 13: Espacio de color rgb



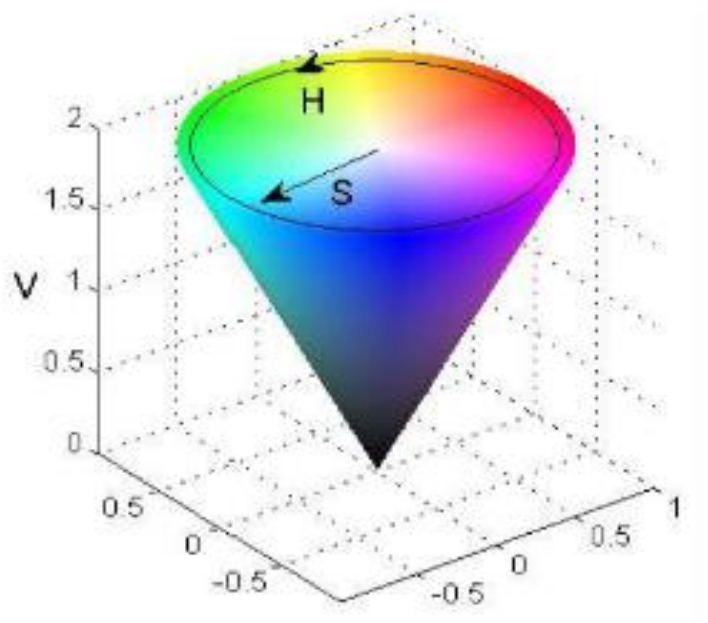
Fuente: (Alegre Gutiérrez et al., 2016, pág. 49)

Es importante tener en cuenta que OpenCV almacena los canales RGB en orden inverso. Mientras que normalmente pensamos en términos de Rojo, Verde y Azul, OpenCV realmente almacena los valores de los píxeles en orden Azul, Verde y Rojo, este ordenamiento BGR se hizo por razones históricas de los desarrolladores. (Rosebrock, 2017)

2.2.11.2.2 Espacio Hsv

Es una proyección del cubo de color RGB sobre un ángulo de croma no lineal. Un porcentaje de saturación radial y un valor inspirado en la luminancia. El valor se define como el valor medio o el valor máximo de color, la saturación se define como la distancia escalara desde la diagonal, y el tono se define como la dirección alrededor de una rueda de color. (Alegre Gutierrez et al., 2016)

Figura 14: Espacio de color hsv



Fuente: (Alegre Gutierrez et al., 2016, pág. 53)

2.2.11.3 Contornos

Son herramientas que se pueden utilizar para el análisis de formas, la detección de formas y el reconocimiento de objetos. Está representada por una curva que une los puntos continuos al largo del límite los cuales tienen el mismo color o también la intensidad. (*OpenCV Open Source Computer Vision*, 2022)

Para una mejor precisión, su usa imágenes binarias, antes de encontrar contornos, se aplica umbral o detección de bordes astutos. En OpenCV, findContours() no se modifica la imagen de origen, sino que devuelve una imagen modificada como el primero de los tres parámetros de retorno. (*OpenCV Open Source Computer Vision*, 2022)

Localizar los contornos es como buscar los objetos de color blanco en un fondo negro. De modo que el objeto a encontrar ha de ser blanco y el fondo toma el color negro. La función esta detallada de la siguiente manera por OpenCV para Python: (*OpenCV Open Source Computer Vision*, 2022)

$$cv.findContours\ (image, mode, method[, contours[, hierachy[, offset]]])$$

2.2.11.3.1 Dibujo de Contorno

Se utiliza la siguiente función cv.drawContours, otra utilidad que se le puede dar es para dibujar distintas formas siempre que contengan sus puntos de contorno. La imagen de origen considerado como el primer argumento, como segundo argumento está considerado los contornos que deben pasarse como una lista de Python. El índice de contornos útil al dibujar contornos individuales y para dibujar todos los contornos, pase -1 es considerado como el tercer argumento y los argumentos restantes son grosor, color etc. La función esta detallada de la siguiente manera por OpenCV para Python: (*OpenCV Open Source Computer Vision*, 2022)

$$cv.drawContours\ (image, contours, contourdx, color[$$

$$, thickness[, lineType[, hierachy[, maxLevel[, offset]]]]])$$

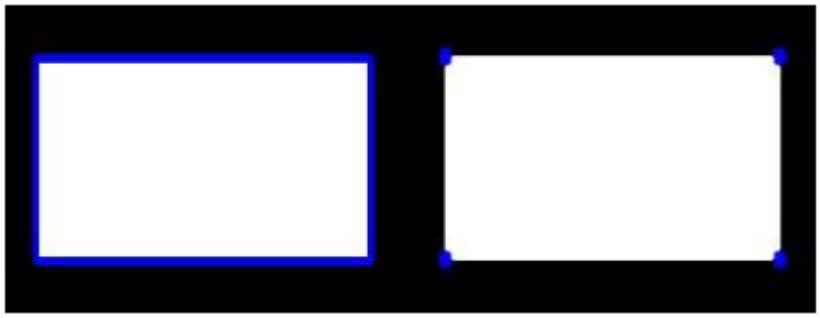
2.2.11.3.2 Métodos de Aproximación de Contorno

En el tercer argumento de la función findContours la que almacena las coordenadas (X, Y) del límite de una forma y para desarrollar esto se realiza una aproximación de contorno. (*OpenCV Open Source Computer Vision*, 2022)

El comando cv.CHAIN_APPROX_NONE, es capaz de almacenar todos los puntos de limite, si por ejemplo se encuentra el contorno en una línea recta, solo se necesitara dos puntos finales de esa línea eso es lo que realiza el comando cv.CHAIN_APPROX_SIMPLE, ahorra memoria para ello tiene que eliminar todos los puntos que son redundantes y también comprimir el contorno. Un ejemplo de esto se visualiza en la figura 15 de esta técnica, en el proceso simplemente se dibujan círculos para cada una de las coordenadas que se encuentran en la matriz de contorno (dibujado en color azul). La figura 15a encuentra los puntos con la función cv.CHAIN_APPROX_NONE utilizando 734 puntos y la figura 15b utilizando la función

cv.CHAIN_APPROX_SIMPLE utiliza solo 4 puntos. (*OpenCV Open Source Computer Vision*, 2022)

Figura 15: Adquisición de contornos



Fuente: (*OpenCV Open Source Computer Vision*, 2022) Recuperado el 02 de Febrero de 2022 de https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html

2.2.11.3.3 Momentos de una Imagen

Es el promedio ponderado que en lo particular se toman de los pixeles en una imagen, también son considerados como una función de tales momentos. En el procesamiento de una imagen los momentos se utilizan para describir objetos seguido de una segmentación. A partir de los momentos se puede obtener el centroide, el área y la orientación, también existen propiedades que derivan de los momentos espaciales de orden 0 y 1. (*OpenCV Open Source Computer Vision*, 2022)

Área de una imagen binaria es M_{00} y el Centroide esta de la siguiente forma: (*Wikipedia*, 2022c)

Ecuación 1: Fórmula general de momentos

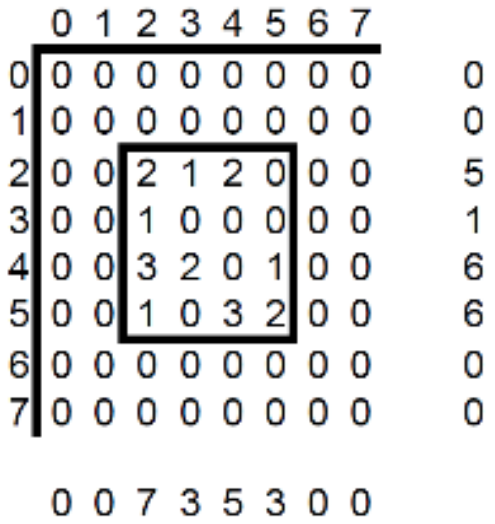
$$\{x, y\} = \left\{\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right\}$$

La función cv.moments() proporciona un diccionario de todos los valores de momento calculados. La función esta detallada de la siguiente manera por OpenCV para Python:

$$cv.moments\ (array[, binaryImage])$$

Se presenta el siguiente ejemplo de imagen en pixeles con coordenadas eje “X” en el lado horizontal y eje “Y” en el lado vertical:

Figura 16: Momentos de una imagen



Fuente: (*YouTube*, 2022) Recuperado el 2 de febrero de 2022 de https://www.youtube.com/watch?v=sPGfnYuj0-Y&t=419s&ab_channel=ProyectosJC

Ecuación 2: Momento 00

$$M_{00} = \sum_x \sum_y I(x,y)$$

$$M_{00} = 0 + 0 + 7 + 3 + 5 + 3 + 0 + 0$$

$$M_{00} = 18$$

Ecuación 3: Momento 10

$$M_{10} = \sum_x \sum_y x^1 y^0 I(x,y) = \sum_x \sum_y x I(x,y)$$

$$M_{10} = 0 * 0 + 1 * 0 + 2 * 7 + 3 * 3 + 4 * 5 + 5 * 3 + 6 * 0 + 7 * 0$$

$$M_{10} = 58$$

Ecuación 4: Momento 01

$$M_{01} = \sum_x \sum_y x^1 y^0 I(x,y) = \sum_x \sum_y y I(x,y)$$

$$M_{01} = 0 * 0 + 1 * 0 + 2 * 5 + 3 * 1 + 4 * 6 + 5 * 6 + 6 * 0 + 7 * 0$$

$$M_{01} = 67$$

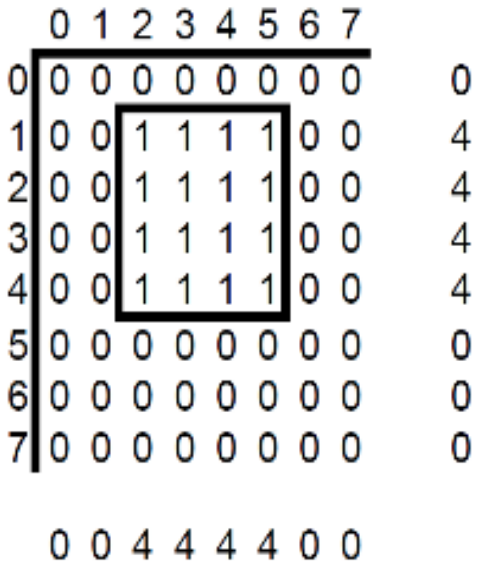
Obtenemos el centroide reemplazando en la ecuación 1.

$$x = \frac{M_{10}}{M_{00}} = \frac{58}{18} = 3.22$$

$$y = \frac{M_{01}}{M_{00}} = \frac{67}{18} = 3.72$$

Si verificamos en la figura 16 vemos que en X = 3 y Y = 3, esto es el pixel 0, que corresponde a nuestro centroide, de la misma manera en una imagen binaria como la figura 17.

Figura 17: Momentos de una imagen binaria



Fuente: (*YouTube*, 2022)Recuperado el 2 de febrero de 2022 de https://www.youtube.com/watch?v=sPGfnYuj0-Y&t=419s&ab_channel=ProyectosJC

2.2.11.3.4 Área de Contorno

El área de contorno viene dado por la función cv.contourArea() o por momentos, M[‘m00’]. La función esta detallada de la siguiente manera por OpenCV para Python:

$$cv.contourArea(contour[, oriented])$$

2.2.11.3.5 Perímetro de Contorno

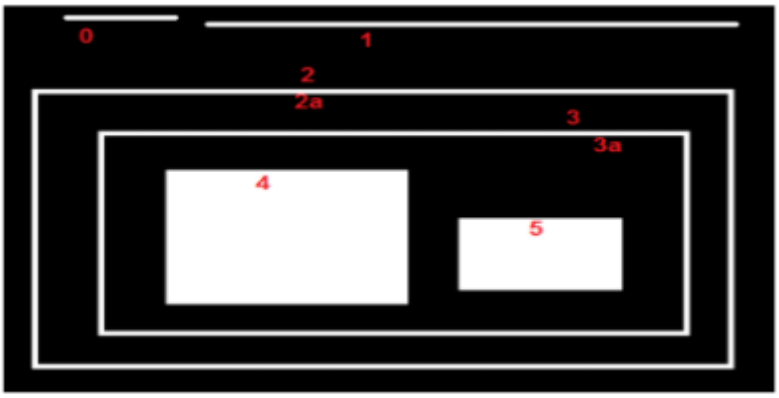
También denominado como la longitud del arco. Se puede averiguar usando la función cv.arcLenght(). Las especificaciones del segundo argumento indican si la forma es un contorno cerrado SI es igual a True, o simplemente denota una curva. La función esta detallada de la siguiente manera por OpenCV para Python:

$$cv.arcLength(curve, closed)$$

2.2.11.3.6 Jerarquía de Contornos

Conlleva la relación de padre-hijo con respecto a los contornos, para la detección de objetos en diferentes lugares al igual que figuras anidadas, el exterior se llama padre y al interior como hijo. De tal manera que los contornos de una imagen establecen una relación entre sí en cual podemos especificar como el contorno está conectado entre sí, por ejemplo, si es el hijo de otro contorno, o es un padre de este, etc. (*OpenCV Open Source Computer Vision*, 2022)

Figura 18: Contornos enumerados



Fuente: (*OpenCV Open Source Computer Vision*, 2022) Recuperado el 2 de febrero de 2022 de https://docs.opencv.org/4.x/d9/d8b/tutorial_py_contours_hierarchy.html

En la figura 18 se muestran formas enumerados desde el 0 al 5, en los cuales 2 y 2a identifican a los contornos tanto externo como interno de la caja más exterior. A los contornos 0,1 y 2 se les denomina como externos o exteriores, se puede decir que

pertenecen a una jerarquía 0 o simplemente están establecidas en el mismo nivel jerárquico.

Se considera al contorno 2a, como un hijo que está en el contorno 2, también se puede considerar de manera opuesta, en donde el contorno 2 es el padre del contorno 2ª, de tal manera se establece la primera jerarquía. De esta manera similar, denominamos al contorno 3 como hijo del contorno 2 y se establece la siguiente jerarquía. Finalmente, la denominación para los contornos 4,5 establecemos que son los hijos del contorno 3a, los cuales vienen en el último nivel de jerarquía. Por la forma en que numeré los cuadros, diría que el contorno 4 es denominado como el primer hijo del contorno 3ª, en los cuales también puede ser atribuido el contorno 5. (*OpenCV Open Source Computer Vision*, 2022)

2.2.11.3.7 Representación de jerarquía en OpenCV

Los contornos tienen su propia información establecida sobre qué jerarquía ocupan, esta información indica quién es su hijo, quién es su padre, etc. En OpenCV se representa con una matriz que tiene cuatro valores: [Siguiente, Anterior, Primer_hijo, Padre] o [Next, Previous, First Child, Parent]. La palabra Next denota el siguiente contorno el cual tiene el mismo nivel jerárquico. (*OpenCV Open Source Computer Vision*, 2022)

Un ejemplo serio, que el contorno 0 en la figura 18. ¿Quién es el próximo contorno en su mismo nivel? En el mismo nivel se encuentra el contorno 1, así que simplemente Next = 1. De manera similar para contorno 2, Next está en contorno 2, entonces Next = 2. (*OpenCV Open Source Computer Vision*, 2022)

Para el contorno 2, no existe contorno siguiente para el mismo nivel. Simplemente, Next = -1. ¿Qué pasa con el contorno 4? Se encuentra en el mismo nivel que el contorno 5, entonces su siguiente contorno es 5, entonces Next = 5. "Previous

denota el contorno anterior en el mismo nivel jerárquico". (*OpenCV Open Source Computer Vision*, 2022)

Es lo mismo que arriba. El contorno anterior al contorno 1 es el contorno 0 los cuales se encuentran en el mismo nivel. De igual manera para el contorno 2, su anterior es el contorno 1. Y para el contorno 0, no existe anterior, pero se pone como -1. "First_Child denota su primer contorno secundario". (*OpenCV Open Source Computer Vision*, 2022)

No hay necesidad de ninguna explicación. Para el nivel jerárquico del contorno 2, el hijo este asignado como contorno 2a. Entonces obtiene el valor de índice que corresponde al contorno 2a. ¿Qué pasa con el contorno-3a? Llega a tener dos hijos. Pero se toma solamente al primer hijo, que pertenece al contorno 4. Entonces utilizamos First Child = 4 para definir el nivel del contorno 3a. "Parent denota índice de su contorno padre". (*OpenCV Open Source Computer Vision*, 2022)

Aplica lo contrario que la función First Child, el contorno 3a, es el contorno principal para el contorno 4 y para el contorno 5. Para el contorno 3a, el contorno principal es contorno 3 y así sucesivamente. Si no existen hijos o padres, ese campo toma el valor de -1. (*OpenCV Open Source Computer Vision*, 2022)

2.2.11.3.8 Modo de Recuperación Retr_List

Se utiliza para recuperar todos los contornos sin crear relaciones como padre-hijo, en la que se elimina la jerarquía dando lugar a que padres e hijo son iguales en esta regla, el cual se representa como solo contornos, indicando que todos pertenecen a una misma jerarquía. Entonces, aquí, el tercer y cuarto término en una matriz de jerarquía siempre se denota por -1. Pero obviamente, los siguientes términos y los términos anteriores tendrán sus correspondientes valores. Esta opción se puede usar si no se está usando ninguna función de jerarquía. (*OpenCV Open Source Computer Vision*, 2022)

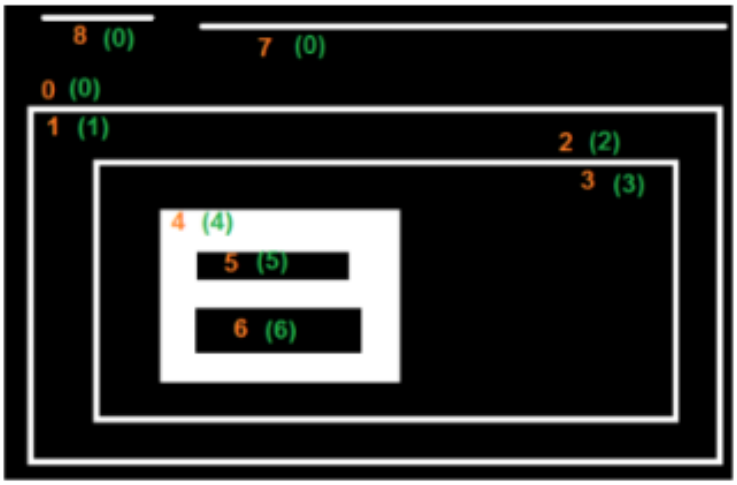
2.2.11.3.9 Modo de Recuperación Retr_External

Este modo solo devuelve banderas externas. Todos los contornos secundarios se dejan atrás. Se puede decir, bajo esta ley, Sólo se cuida al mayor de cada familia. No le importan los demás miembros de la familia. (*OpenCV Open Source Computer Vision*, 2022)

2.2.11.3.10 Modo de Recuperación Retr_Tree

Este modo recupera todos los contornos establecidos y crea un listado de jerarquía familiar completa, incluso identifica al abuelo, el padre, el hijo, el nieto e incluso más allá. (*OpenCV Open Source Computer Vision*, 2022)

Figura 19: Modo de recuperación de contorno RETR_TREE



Fuente: (*OpenCV Open Source Computer Vision*, 2022) Recuperado el 2 de febrero de 2022 de https://docs.opencv.org/4.x/d9/d8b/tutorial_py_contours_hierarchy.html

El contorno 0 se encuentra en la jerarquía 0, el siguiente contorno que se encuentra en la misma jerarquía es el contorno 7. Sin contornos previos. El hijo es contorno 1 el cual no tiene padre. Entonces la matriz es [7,-1,1,-1]. El contorno 2 está establecida en la jerarquía 1 sin más contornos en el mismo nivel, no existe anterior y el hijo es contorno 3, por lo tanto, el padre es contorno 1. Entonces la matriz es [-1,-1,3,1]. (*OpenCV Open Source Computer Vision*, 2022)

2.2.11.4 Transformaciones Geométricas de Imágenes

2.2.11.4.1 Transformación Geométrica Escalar

Se define como un cambio de tamaño en una imagen, en OpenCV se realiza con la funcion cv.resize(), puede especificarse manualmente o puede ser especificado por el factor de escala el cual determinara el tamaño de la imagen. Para ello pueden utilizarse diferentes métodos en la interpolación, los más utilizados son cv.INTER_AREA que sirve para reducir el área, cv.INTER_CUBIC y CV.INTER_LINEAR que se utilizan para hacer zoom a la imagen. La interpolación cv.INTER_LINEAR en su forma predeterminada se utiliza para todos los fines de cambio en el tamaño de la imagen. (*OpenCV Open Source Computer Vision*, 2022)

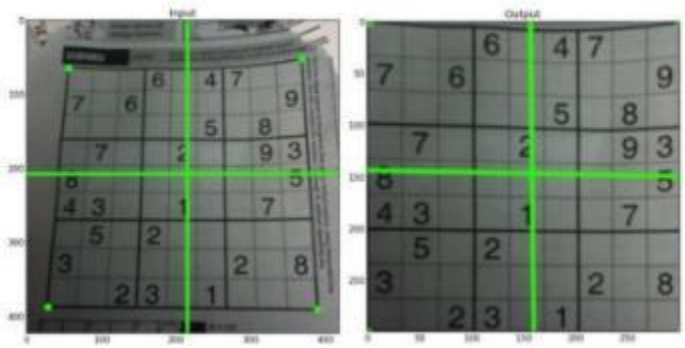
2.2.11.4.2 Transformación de Perspectiva

Para realizar una transformación de perspectiva será necesaria tener una matriz de 3x3, para ello se necesitan tomar 4 puntos en la imagen de entrada y también necesitamos tomar los puntos que corresponden en la imagen de salida. De estos 4 puntos, 3 de estos no deben de ser colineales, luego, esta matriz de transformación puede ser encontrada por la función cv.getPerspectiveTransform, para continuar se aplica la función cv.warpPerspective junto a con la matriz de 3x3. Todo el proceso indica que en una transformación de perspectiva las líneas rectas permanecen igual incluso después de la transformación. Las funciones antes mencionadas están detalladas por OpenCV para Python. (*OpenCV Open Source Computer Vision*, 2022)

$$cv.getPerspectiveTransform(src, dst)$$

$$cv.warpPerspective(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])$$

Figura 20: Transformación de perspectiva



Fuente: (*OpenCV Open Source Computer Vision*, 2022) Recuperado el 3 de febrero de 2022 de https://docs.opencv.org/4.x/da/d6e/tutorial_py_geometric_transformations. html

2.2.12 Detección de Objetos

2.2.12.1 Clasificador Cascada

El clasificador cascada se encarga de encontrar detalles de las imágenes capturadas por una cámara, como son iluminación, ángulo de visión, distancia y movimiento. Estas abstracciones o características tienen un nivel de similitud basándose en distancias euclidianas. (Minichino & Howse, 2015)

2.2.12.1.1 Detección de Cascada Haar

La detección cascada Haar está proyectada a rostros, pero también puede trabajar para objetos, no son robustas a los cambios de rotación, por ejemplo, en rostros una cara invertida no se considera familiar a una cara erguida y una cara vista de perfil no se considera similar a un rostro visto de frente. Una implementación más compleja y con más recursos podría mejorar la solidez de las cascadas de Haar considerando múltiples transformaciones de las imágenes como múltiples tamaños de ventana. (Minichino & Howse, 2015)

2.2.12.2 Entrenamiento de Clasificador en Cascada

El proceso de entrenamiento en cascada implica dos maneras calcular las compensaciones, en algunos casos, los clasificadores con más características alcanzarán

mayores tasas de detección y menores tasas de falsos positivos. Al mismo tiempo, los clasificadores con más características requieren más tiempo de cálculo.

En principio, se podría definir un marco de optimización en el que:

- El número de etapas del clasificador
- El número de características en cada etapa
- El umbral de cada etapa, se intercambian con el fin de minimizar el número esperado de características evaluadas. Desgraciadamente, encontrar este óptimo es un problema tremendamente difícil

En la práctica, se utiliza un marco muy simple para producir un clasificador eficaz que sea altamente eficiente. Cada etapa de la cascada reduce la tasa para los falsos positivos por lo cual disminuye la tasa de detección. Se selecciona un objetivo para la mínima reducción de falsos positivos y la máxima disminución de la detección. Cada etapa se entrena añadiendo características hasta que se alcanzan los índices de detección y de falsos positivos del objetivo. (Viola & Jones, 2001)

2.2.12.3 Preparación de los Datos de Entrenamiento

La preparación de datos para realizar un entrenamiento de imágenes necesita clasificadores débiles, para ello son necesarias tener unas varias muestras positivas que contengan objetos reales que se quiere detectar y también son necesarias un conjunto de imágenes negativas que tengan todo lo que no se necesita detectar. (Viola & Jones, 2001)

Las muestras negativas están dadas por un conjunto de imágenes arbitrarias los cuales no tiene registro de los objetos que se quiere detectar, a partir de las cuales se generan las muestras, deben enumerarse en un archivo de imagen negativa especial que contenga una ruta de imagen por línea las cuales pueden ser absoluta o relativa. Se debe tener en cuenta que las imágenes para las muestras negativas y las imágenes de muestra

se denominan globalmente muestras de fondo o imágenes de fondo los cuales se usan indistintamente en el presente documento. (Viola & Jones, 2001)

Las imágenes capturadas pueden ser de tamaños diferentes. De este modo cada imagen debe de tener un tamaño igual o un tamaño mayor que el que tiene la ventana de entrenamiento deseado el cual corresponde a la medida de las dimensiones del modelo a entrenar, siendo en la mayoría de las ocasiones el tamaño promedio del objeto, dado que estas imágenes también se utilizan como submuestra de las imágenes negativas las cuales están dadas por varias imágenes, estas muestras tienen el tamaño de ventana de entrenamiento. (Viola & Jones, 2001)

Las muestras positivas son usadas por el proceso de impulso para definir qué debe buscar realmente el modelo cuando intenta encontrar sus objetos de interés. La aplicación admite dos formas para generar el conjunto de datos para todas las muestras positivas. Se puede generar un montón de positivos a partir de una sola imagen de objeto positivo. Se puede proporcionar todos los aspectos positivos usted mismo y solo usar la herramienta para recortarlos, cambiarles de tamaño y colocarlo en un formato binario que será necesario para OpenCV. (Viola & Jones, 2001)

2.2.12.4 Cascade Trainer GUI

Es un software para realizar entrenamiento, esto requiere de una carpeta general, dentro de ella dos subcarpetas llamadas “n” y “p”, n para negativas y p para positivas. Las pestañas Common, Cascade y Boost se pueden usar para configurar numerosos parámetros para personalizar el entrenamiento del clasificador. La interfaz gráfica de usuario de Cascade Trainer establece la configuración más optimizada y recomendada para estos parámetros de forma predeterminada; aun así, es necesario modificar algunos parámetros para cada entrenamiento. (*AMIN Thou Shalt Programme*, 2022)

Se puede configurar el tamaño del búfer de cálculo previo para acelerar el proceso de entrenamiento, se debe asignar una cantidad de memoria elevada, pero se debe tener mucho cuidado para no asignar demasiado o muy poco. Por ejemplo, si se tiene 8 GB de RAM en una computadora, es posible configurar de manera segura los dos tamaños de búfer a continuación en 2048. (*AMIN Thou Shalt Programme*, 2022)

Tanto el ancho como la altura de la muestra, no debe ser configurado en un tamaño muy grande porque hará que su detección sea muy lenta. En realidad, es bastante seguro establecer siempre un valor pequeño para esto. La configuración recomendada tanto para el ancho como para la altura de la muestra es mantener el aspecto en 24 y configurar el otro en consecuencia. Por ejemplo, si tiene imágenes de muestra de 320 × 240, primero se calcula la relación de aspecto, que en este caso es 1,33: 1, luego se multiplica el número más grande por 24. Se obtiene 32 × 24 para el ancho y para la altura de dicha muestra. (*AMIN Thou Shalt Programme*, 2022)

Es posible establecer el tipo de función como en HAAR o LBP. Se usa HOG solo si se tiene OpenCV 3.1 o posterior. Los clasificadores HAAR son muy precisos, pero requieren mucho más tiempo para entrenarse, por lo que es mucho más inteligente usar LBP si puede proporcionar a sus clasificadores muchas imágenes de muestra. Los clasificadores LBP, por otro lado, son menos precisos, pero entrenan mucho más rápido y detectan casi 3 veces más rápido. (*AMIN Thou Shalt Programme*, 2022)

2.2.13 Reconocimiento de Voz

Es una herramienta computacional el cual puede procesar la señal de voz que es emitida por un ser humano e identificar la información que esta contiene para poder transformarla en texto o también puede emitir las ordenes que están actuando sobre un proceso. Las diciplinas como la fisiología, la acústica, la lingüística, los procesos de señal,

la inteligencia artificial y también la ciencia de la computación, son las diciplinas que intervienen en el desarrollo del reconocimiento de voz. (*Wikipedia*, 2022d)

2.2.13.1 Aplicaciones

2.2.13.1.1 Dictado Automático

Tiende a ser el uso más común que se le ha dado al reconocimiento de voz, son utilizados para el dictado de recetas y diagnósticos médicos o también en el dictado de textos legales, se utilizan corpus de manera especial las cuales incrementan la precisión del sistema. (*Wikipedia*, 2022d)

2.2.13.1.2 Control por Comandos

Es un diseño de reconocimiento de voz que se utiliza para dar órdenes a una computadora, para incrementar el rendimiento los sistemas reconocen solo vocabularios reducidos. (*Wikipedia*, 2022d)

2.2.14 Micrófono

Es un instrumento que capta el sonido del exterior y hace una conversión de ello en señales analógicas, su uso es muy común en estos días en diversas aplicaciones que trabajan el audio y viene incluido en diversos dispositivos electrónicos. (*Wikipedia*, 2022e)

2.2.14.1 Micrófono de Condensador

Este dispositivo es muy popular debido a su calidad de grabaciones, su funcionamiento se basa en dos placas una fija y otra flexible, que cuando se ingresa ondas sonoras en la placa flexible, estas vibran y varia la distancia entre ellas haciendo que haiga un cambio en la capacitancia. (*Wikipedia*, 2022e)

2.2.15 Programas y Complementos para el Desarrollo

2.2.15.1 Arduino

Es una empresa que desarrolla hardware y software libres, también cuenta con una comunidad gratuita de apoyo en algoritmos basados en varios problemas relacionados con la electrónica. Su microcontrolador puede reprogramarse y ser utilizado en otros diseños de placa, para su conexión con dispositivos de salida el microcontrolador es necesario poner pines hembra en el diseño. (Arduino, 2022)

2.2.15.2 Cascade Trainer Gui

Es un software que realiza entrenamientos de modelos con la función de clasificar imágenes en cascada para ello se hacen pruebas y mejoras de estos en el software, que resulta provechosos en la detección de objetos. Esta elaborado en base a una interfaz gráfica para que el usuario pueda determinar parámetros y usar las herramientas de OpenCV. (*AMIN Thou Shalt Programme*, 2022)

2.2.15.3 Matlab

Es un lenguaje de cálculo desarrollado por MathWorks, en ello se desarrollan algoritmos, modelos y análisis a partir de datos, por ello su importancia en las investigaciones de ingenieros y científicos. (*MathWorks*, 2022). Tiene aplicaciones como:

- IA, datos de ciencia y estadística
- Operaciones matemáticas y optimizaciones
- Procesamiento de la señal
- Procesamiento de la imagen en la visión artificial
- Sistemas de control
- Pruebas y mediciones
- Radiofrecuencia y señales mixtas
- Comunicaciones inalámbricas

- Robótica y autonomía de sistemas
- Desarrollo de dispositivos como FPGA, ASIC y SoC
- Finanzas en la computación
- Biología en la computación
- Verificaciones en el código
- Sectores aeroespaciales

En el presente caso se ha utilizado la aplicación matemáticas y optimización, que comprende las herramientas como:

- Toolbox de optimización.
- Toolbox para optimización global.
- Toolbox de símbolos matemáticos.
- Toolbox de mapeo.
- Toolbox de ecuaciones diferenciales parciales.

Los toolbox pueden ser usados de distintas formas, también desarrollan análisis y una solución de integración en tiempo directo en aplicaciones como estáticos lineales, modelamientos dinámicos estructurales, modelamiento de transferencia de calor, difusiones, electrostática y magnetismo. Pero también puede ser posible crear sus propias ecuaciones diferenciales parciales. (*MathWorks*, 2022)

2.2.15.4 Solidworks

Este software elaborado para Windows nos permite realizar diseños asistidos por computadora en 3D, en el que podemos simular piezas mecánicas 3D y ensamblarlos, también podemos diseñar planos 2D. La importancia de sus poderosas herramientas proporciona una mejor elaboración en los modelados, esto ayuda a los ingenieros y diseñadores en sus trabajos para realizar creaciones, validaciones, comunicaciones y

gestiones del proyecto, nos asegura de que se tuvo un diseño correcto antes de su fabricación. (“Solidworks,” 2021)

2.2.15.5 Python

Es un lenguaje orientado a objetos potente y fácil de aprender, su sintaxis se basa en estructuras de datos de alto nivel, la programación que se puede desarrollar con este lenguaje es ideal en varias plataformas en el que se realiza muchas aplicaciones en muchas áreas. (*Python*, 2022)

2.2.15.5.1 Módulo

Son archivos con la extensión “.py” que fueron elaborados con distintas formas en el algoritmo basados en Python, esto contiene varias definiciones y declaraciones que uno puede encontrar en la biblioteca de la página oficial. Los módulos pueden ser propios del lenguaje de programación o también con proyectos de desarrolladores independientes que aportan a la comunidad, pero uno mismo puede crear su propio modulo e importarlo desde otro programa usando el nombre “import”. (*Documentación de Python*, 2022b)

2.2.15.5.2 Date time

Es un módulo que está enfocado en implementar la extracción de los atributos al que se adecua el formato de la fecha, este algoritmo fue elaborado con clases de forma simple y compleja para obtener una manipulación de la hora y fecha. (*Biblioteca de Python*, 2022a)

2.2.15.5.3 Pip

Es el programa de instalación preferido. A partir de Python 3.4, se incluye de forma predeterminada con los instaladores binarios de Python. Su importancia se basa en la facilidad en su interfaz de línea de comandos, estos nos permiten instalar módulos, complementos de Python (*Python*, 2022)

2.2.15.5.4 Pyaudio

Este módulo tiene un enlazamiento con la biblioteca PortAudio v19, ya que esta brinda un código abierto multiplataforma para entrada y salida de audio, su importancia es que permite que el software adquiera y envíe audio en tiempo real desde una interfaz de audio del hardware de la computadora. Esto también permite interactuar con varios sistemas operativos como Microsoft, Apple, Linux y MacOS para operaciones como reproducir y grabar audio. (*Proyectos de Python*, 2022)

2.2.15.5.5 Pyttsx3

Este módulo tiene una biblioteca con variedad de voces que están enlazadas a las que por defecto tiene el sistema operativo, dependiendo de ello estos tienen la capacidad convertir el texto a voz y funcionan con o sin conexión a internet. Son compatibles con la versión 2 y la versión 3 de Python. (*Proyectos de Python*, 2022)

2.2.15.5.6 Speech Recognition

Este módulo tiene una elaboración de APIs en línea y fuera de línea, por ello está diseñado con varios motores para reconocer la voz, estos están detallados en proyectos de diferentes empresas tales como Google, IBM, etc. Y tienen un buen soporte en el procesamiento de la señal que emite la voz. (*Proyectos de Python*, 2022)

2.2.15.5.7 Subprocess

Es un módulo diseñado para reemplazar a otros módulos y funciones más antiguos, para ello se requiere una conexión a conductos como entrada, salida, error y la obtención de los códigos de retorno, por lo que se necesita generar nuevos procesos. (*Documentación de Python*, 2022a)

2.2.15.5.8 Threading

Este módulo establece en el hilo de módulos de niveles inferiores una interfaz de subprocesamiento de nivel superior, de esta manera nos permite ejecutar procesos en segundo plano. (*Biblioteca de Python*, 2022b)

2.2.15.5.9 Tkinter

En este módulo Tk se establecen conjuntos de widgets que son implementados como clases pertenecientes a la biblioteca de Python, está orientado a objetos por lo que en la ejecución de subprocesos es necesario tener un mecanismo seguro para la interacción de esto. (*Documentación de Python*, 2022b)

2.2.15.6 Numpy

Este complemento es usado en operaciones computacionales, tiene una amplia biblioteca basado en una matriz multidimensional como matrices y matrices enmascaradas, para la ejecución de operaciones rápidas se presentan una variedad de rutinas en la manipulación matemática como formas, clasificación, E/S, transformadas de Fourier, algebra lineal, estadística, simulaciones aleatorias, etc. (*Numpy*, 2022)

2.2.15.7 OpenCV

Este complemento tiene una diversa biblioteca en los métodos de visión artificial en Intel, es de código abierto y tiene una licencia BSD. Su funcionamiento está basado en sistemas operativos como Mac OSX, Windows y Linux. Si en el sistema es encontrado Intel Integrated Performance Primitives entonces el procesamiento de imagen en tiempo real es posible, estas rutinas optimizan la aceleración de estos procesos. (*EcuRed*, 2022)

2.2.15.8 Pillow

Este complemento tiene una amplia biblioteca que es compatible con varios formatos de archivo, cuenta con una potente capacidad de procesamiento en las imágenes.

El diseño para formatos de pixeles realiza un acceso rápido en el almacenamiento de datos, las herramientas para el procesamiento de imágenes proporcionan una base sólida. (*Pillow*, 2022)

2.2.15.9 Pygame

Este complemento está elaborado en videojuegos, tiene agregado funcionalidades como la biblioteca SDL. Las funciones insertadas en este lenguaje son necesarias para la creación de juegos y programas multimedia. (*PyGame*, 2022)

2.2.15.10 Pyserial

En la conexión del Arduino con Python instalado en una computadora, se necesita backends que se ejecuten en sistemas operativos como Windows, OSX, Linux, BSD o posiblemente cualquier sistema compatible con POSIX e IronPython. Por ello para la conexión llamado serial está elaborado en este complemento, esto hace una selección automática del backend apropiado. (*Pyserial*, 2022)

CAPITULO III

MATERIALES Y MÉTODOS

3.1 METODOLOGIA

En este conjunto de métodos sobre investigación tenemos que el enfoque es cuantitativo. Este sigue una secuencia de pruebas, para acortar y delimitar la información se parte de una idea, en la investigación se plantean objetivos con interrogantes. Las hipótesis se plantean y resultan variables que desarrollan el diseño, estas últimas se miden y analizan. Por último, se concluye con respecto a las hipótesis. (Hernández Sampieri, 2014)

3.2 DISEÑO DE LA INVESTIGACIÓN

En esta investigación se desarrolló un diseño experimental por que se refiere a un estudio en el que se manipulan intencionalmente una o más variables independientes, para analizar las consecuencias que la manipulación tiene sobre una o más variables dependientes dentro de una situación de control. (Hernández Sampieri, 2014)

3.3 NIVEL DE LA INVESTIGACIÓN

Sampieri (2014) busca especificar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se sometan a un análisis.

El nivel o alcance de la presente investigación es descriptivo porque busca desarrollar un tema novedoso de investigación en beneficio de la sociedad y especificar las características de ello en el proceso del análisis.

3.4 POBLACIÓN Y MUESTRA

3.4.1 Población

De acuerdo con Sampieri (2014) una población es un conjunto de todos los casos que concuerdan con determinadas especificaciones es una población.

En el presente proyecto el conjunto de datos utilizados en la investigación, son los dispositivos electrónicos y herramientas utilizados en la mesa de trabajo para el montaje de tableros eléctricos del taller de automatización industrial de la empresa Automyn.

3.4.2 Muestra

Según Sampieri (2014) para el proceso cuantitativo, la muestra es un subgrupo de la población de interés sobre el cual se recolectarán datos, y que tiene que definirse y delimitarse de antemano con precisión, además de que debe ser representativo de la población.

En el presente proyecto la muestra está representada por una herramienta como el destornillador y un dispositivo electrónico como un pulsador industrial que corresponde al conjunto de herramientas y dispositivos que conforman la mesa de trabajo del taller de automatización industrial de la empresa Automyn.

3.5 TÉCNICAS E INSTRUMENTOS PARA RECOLECCIÓN DE DATOS

3.5.1 Técnicas

Sampieri (2014) dice que medir significa “asignar números, símbolos o valores a las propiedades de objetos o eventos de acuerdo con reglas”.

Para la recolección de datos se utilizó la técnica de la observación, ya que los hechos y fenómenos investigados se evaluaron personalmente en la implementación y desarrollo del experimento.

3.5.2 Instrumentos

Así mismo Sampieri (2014) Un instrumento de medición adecuado es aquel que registra datos observables que representan verdaderamente los conceptos o las variables que el investigador tiene en mente.

En los instrumentos utilizados para la recolección de datos fueron cuaderno de notas e instrumentos de medición métrica como la mesa de trabajo milimétrica, el plano

coordenado virtual y los datos del reconocimiento de voz, el cual fue utilizada en cada uno de los experimentos.

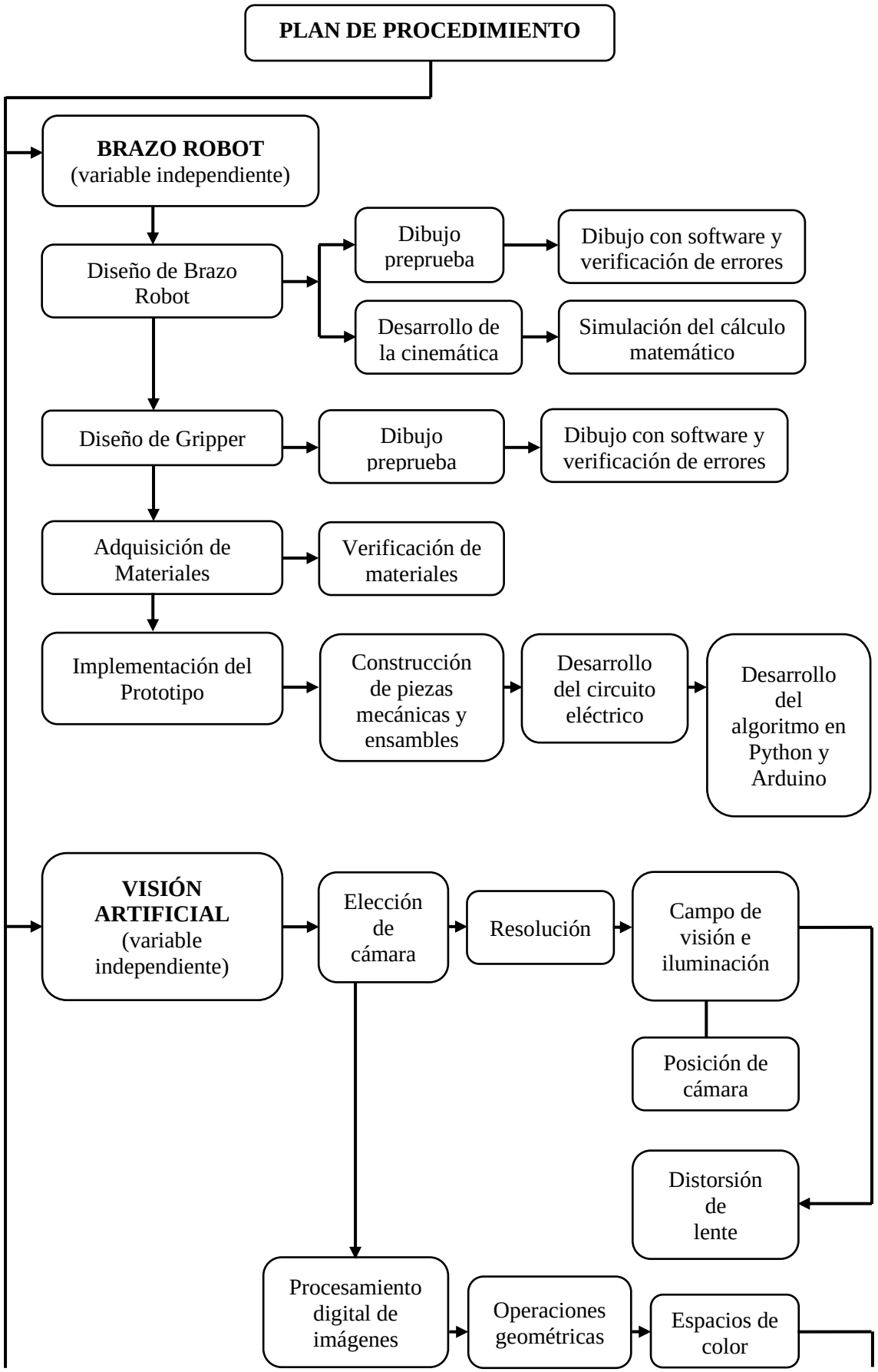
3.6 PROCEDIMIENTO DEL EXPERIMENTO

De igual manera Sampieri (2014) detalla que elaborar un plan detallado de procedimientos para reunir datos con un propósito especifico implica realizar una recolección de datos.

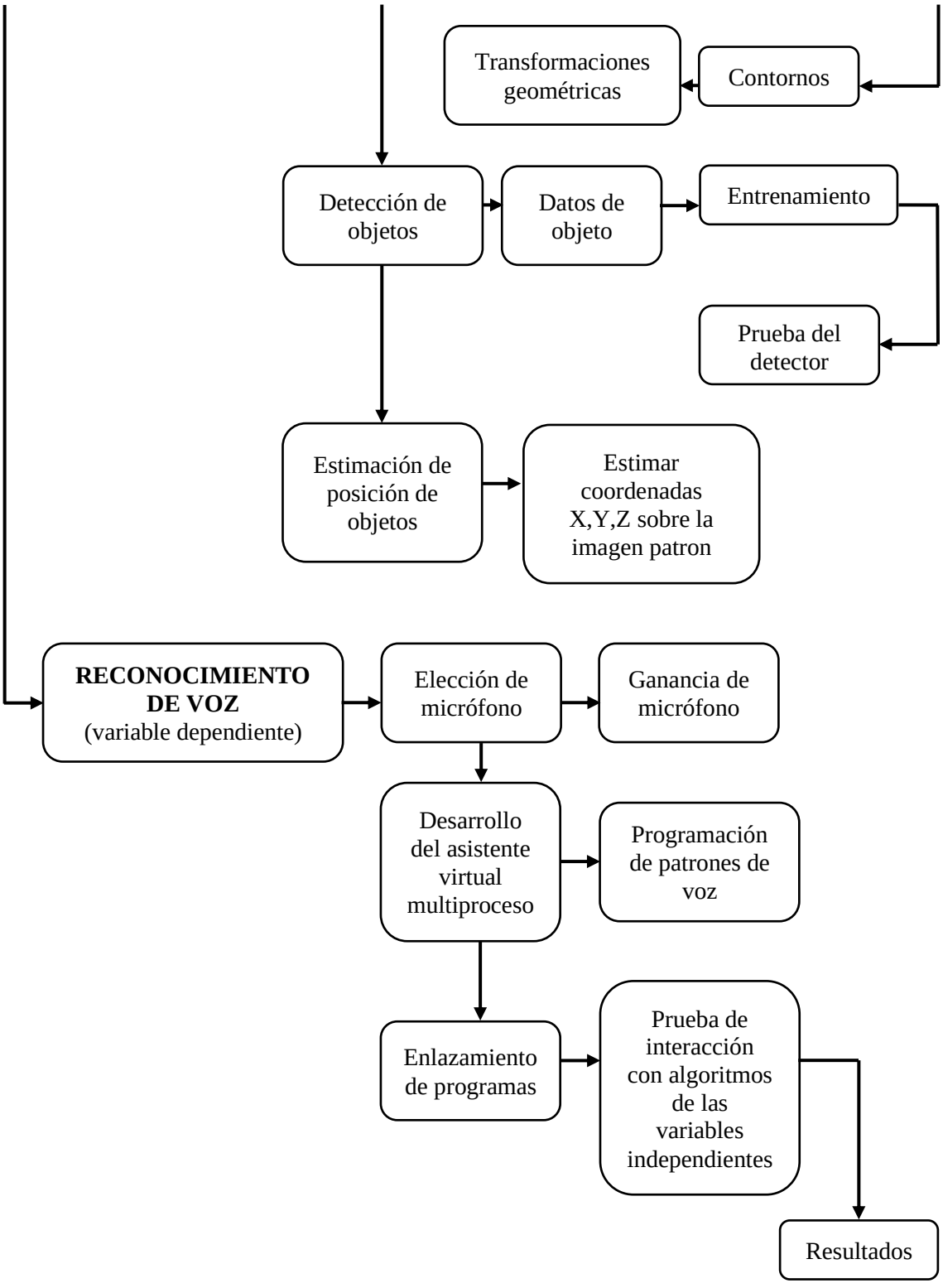
Realizamos un mapa semántico para poder entender el procedimiento y reunir datos de ello, además se desarrolló un diagrama de flujo para el procedimiento de la programación de los algoritmos.

Se muestra el mapa semántico del plan de procedimiento del proyecto.

Figura 21: Mapa semantico del plan de procedimiento



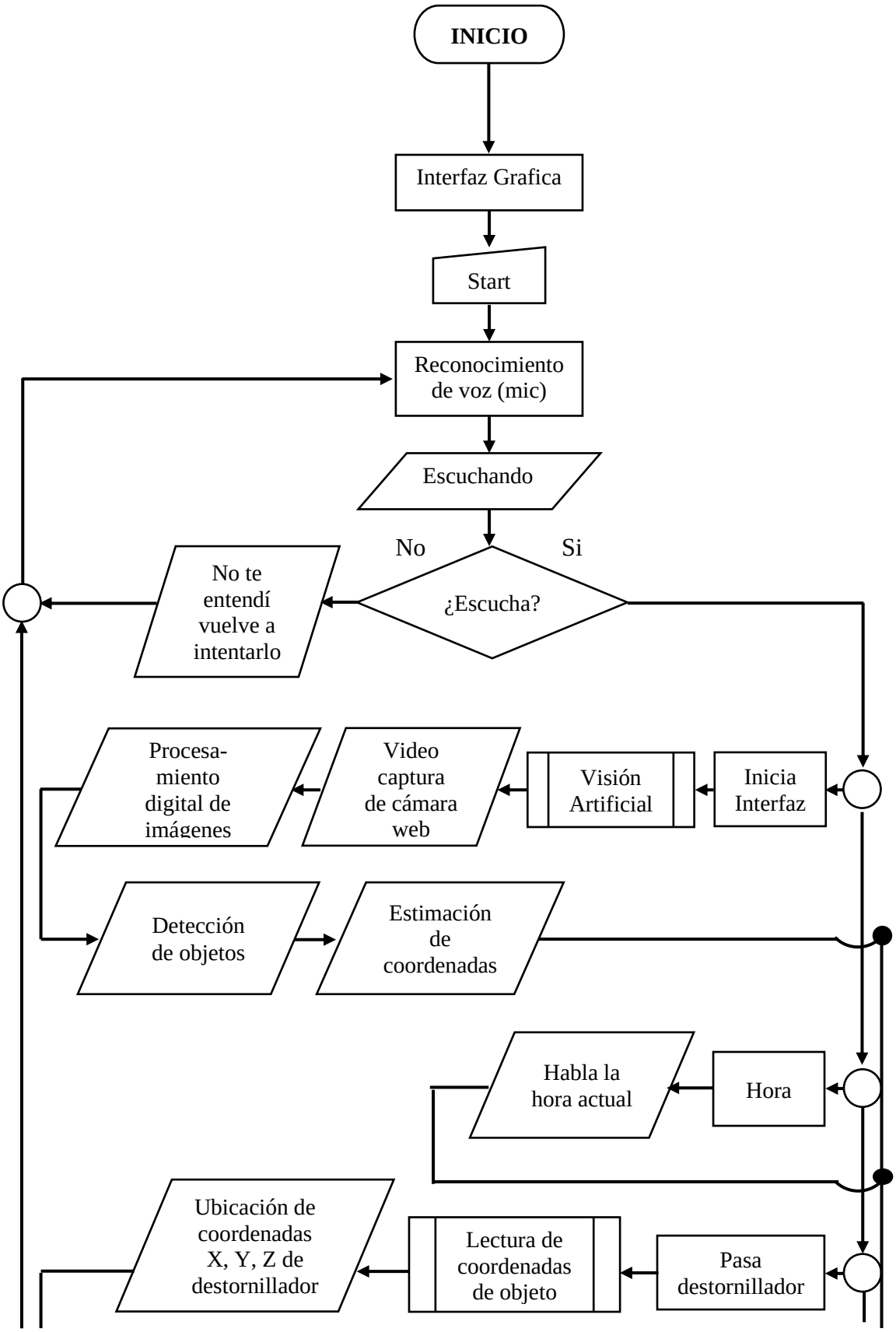
Continuación ……



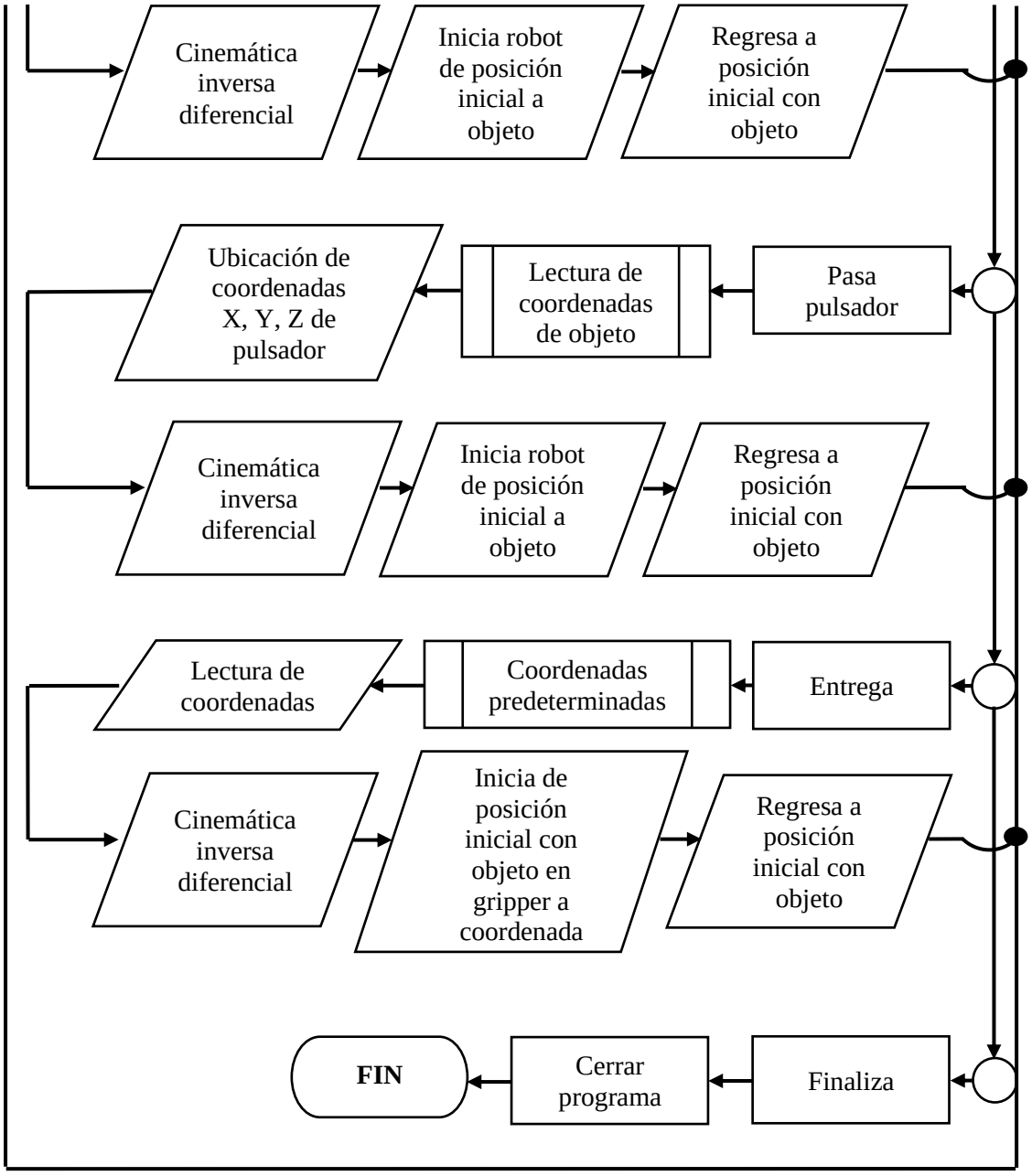
Elaborado por el equipo de trabajo

Se muestra el diagrama de flujo para la programación del algoritmo del proyecto.

Figura 22: Diagrama de flujo del plan de procedimiento



Continuación…



Elaborado por el equipo de trabajo

3.7 OPERACIONALIZACIÓN DE VARIABLES

Tabla 9: Operación de variables dependiente e independiente

Tipo	Variable	Dimensiones	Indicadores	Instrumento
Independiente	Patrones de Voz	Reconocimiento de Voz	Dispositivo de Entrada: Micrófono	Software de Desarrollo
			Dispositivo de Salida: Altavoz	
Dependiente	Brazo Robot Con Gripper	Posicionamiento del Efector Final	Cinemática Directa e Inversa	Software de Desarrollo
			Jacobiano	
		Aproximación del Objeto	Sensor de Proximidad	
		Presión ejercida para el agarre	Sensor FSR	
	Visión Artificial	Detección de Objetos	Clasificador de Imágenes	
			Entrenamiento de Imágenes	
		Estimación de Posición	Transformación de Perspectiva	

Elaborado por el equipo de trabajo

3.8 UBICACIÓN DEL PROYECTO

La ejecución del proyecto está pensada a implementar en la empresa Automyn y tiene los siguientes datos:

- RUC: 20448143661
- Latitud: 15°29′27″ sur
- Longitud: 70°07′37″ oeste
- Altitud: 3825 m. s. n. m.

Figura 23: Ubicación de la empresa en la ciudad de Juliaca

Elaborado por el equipo de trabajo

Este trabajo tuvo un cambio de ubicación debido al acceso del entorno, por tal motivo se elaboró en un laboratorio personal ubicado en la ciudad de puno, en el cual se estableció un área de trabajo similar que se asemeja al de la empresa Automyn. Por lo que el desarrollo y los resultados están establecidos para dicho laboratorio. Y tiene los siguientes datos:

- Latitud: 13°00'00" y 17°17'30" sur

- Longitud: 71°06'57" y 68°48'46" oeste
- Altitud: 3827 m.s.n.m.

Figura 24: Ubicación del laboratorio en la ciudad de Puno

Elaborado por el equipo de trabajo

3.9 ANÁLISIS DEL DESARROLLO

El procedimiento del experimento necesita un estudio por variable para su desarrollo por lo que se detalla sus análisis matemático y diseño 3D.

Para realizar el diseño de cada pieza que nos permitirá el ensamble, se utilizó el software SolidWorks, programa que permite diseños 3D con alta resolución, también permite ver una simulación de ensamble completo de todas las piezas que se diseñan.

Para realizar la simulación del cálculo matemático se utilizó el programa Matlab que permite ver el movimiento que realiza cada articulación según vamos modificando los valores deseados, también se utilizó para realizar los cálculos en multiplicación de matrices. Y también se utilizó Python para programar el controlador del brazo robot,

visión artificial, reconocimiento de voz. En la figura 3.5 se visualiza una computadora de escritorio.

Figura 25: Computadora para el desarrollo

Elaborado por el equipo de trabajo

Tabla 10: Caracteristicas de la computadora

DATOS	COMPUTADORA
Procesador	Core i5 – 2.90Hz
Núcleos / Hilos	6 / 12
Tipo de Sistema	64 bits
Memoria Ram	16 Gb
Tarjeta de Video	Nvidia Geforce GT 730 – 4Gb

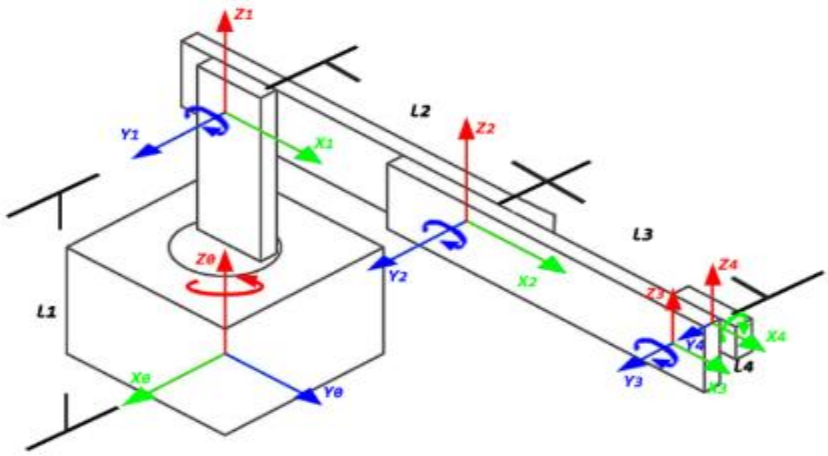
Elaborado por el equipo de trabajo

3.9.1 Análisis del Desarrollo Cinemático

3.9.1.1 Cinemática Directa

El análisis del cálculo cinemático se realiza según el posicionamiento de los ejes y la dirección del movimiento que cada una de las articulaciones realiza.

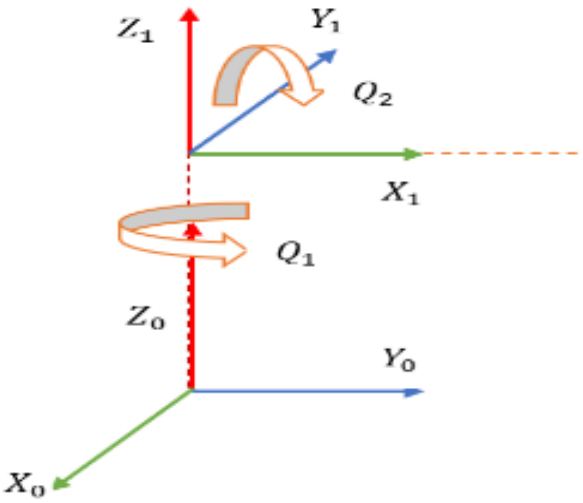
Figura 26: Estructura del robot de 5GDL



Elaborado por el equipo de trabajo

Seguidamente realizamos la asignación de coordenadas y en cada una de las articulaciones. Reconocemos los movimientos realizados para llegar desde el punto de origen a la siguiente articulación.

Figura 27: Evaluación de las dos primeras articulaciones



Elaborado por el equipo de trabajo

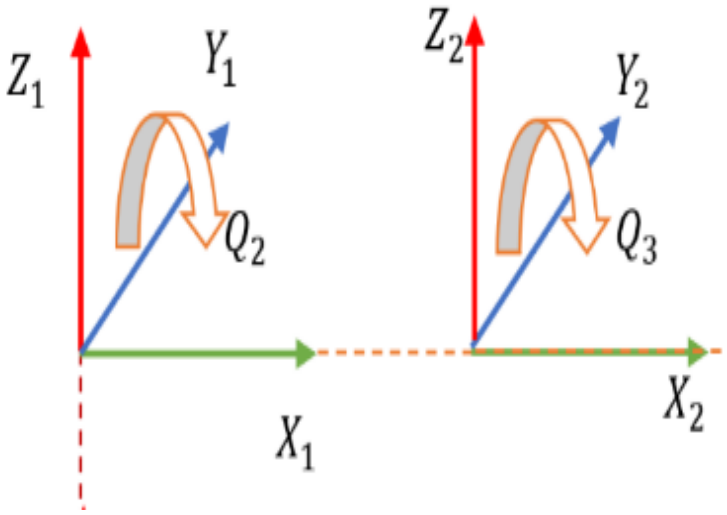
Realizando el análisis del grafico con el método de coordenadas ortonormales, en los cuales representaremos *H* como la matriz homogénea de una articulación, *R* representa a la matriz homogénea de rotación en el eje correspondiente y *T* es la matriz homogénea de traslación en el eje correspondiente, lo primero es realizar una rotación en el eje Z_0 con el ángulo Q_1 luego realizamos una translación en el mismo eje para llegar a la siguiente articulación y se obtiene la ecuación 5.

Ecuación 5: Matriz de articulación 0 a 1

$$H_1^0 = R_{z_0} T_{z_0}$$

$$H_1^0 = \begin{bmatrix} \cos Q_1 & -\sin Q_1 & 0 & 0 \\ \sin Q_1 & \cos Q_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 28: Coordenadas correspondientes a las articulaciones 2 y 3



Elaborado por el equipo de trabajo

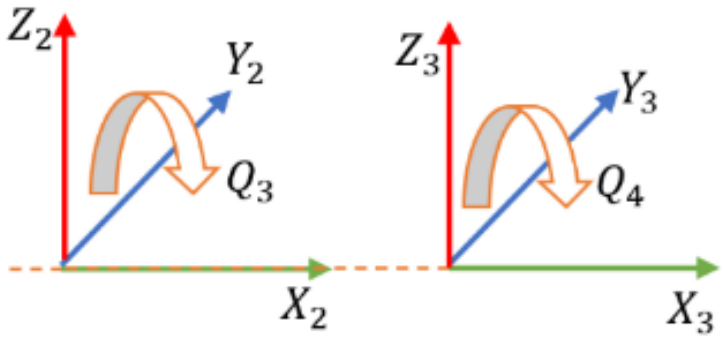
Seguidamente evaluamos la rotación que realiza Q_2 en el eje Y_1 y la translación que realiza en el eje X_1 para llegar a la siguiente articulación, de la evaluación se tiene la ecuación 6.

Ecuación 6: Matriz de articulación 1 a 2

$$H_2^1 = R_{Y_1} T_{X_1}$$

$$H_2^1 = \begin{bmatrix} \cos Q_2 & 0 & \sin Q_2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin Q_2 & 0 & \cos Q_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 29. Coordenadas correspondientes a los grados de libertad 3 y 4



Elaborado por el equipo de trabajo

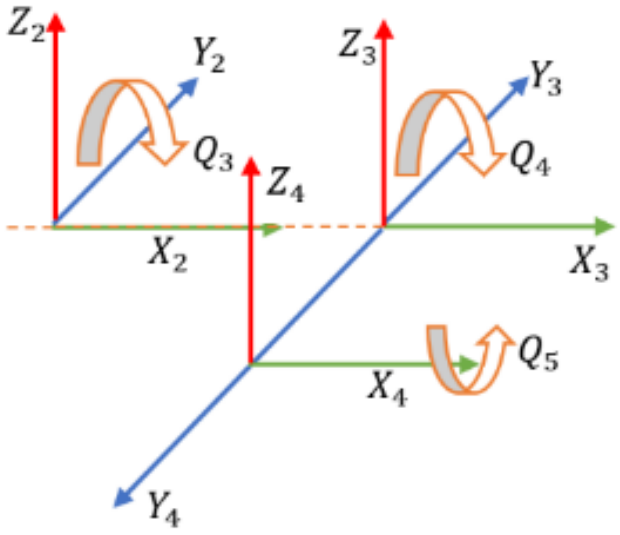
Seguidamente evaluamos la rotación que realiza Q_3 en el eje Y_2 y la traslación que realiza en el eje X_2 para llegar a la siguiente articulación, de la evaluación se tiene la ecuación 7.

Ecuación 7: Matriz de articulación 2 a 3

$$H_3^2 = R_{Y_2} T_{X_2}$$

$$H_3^2 = \begin{bmatrix} \cos Q_3 & 0 & \sin Q_3 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin Q_3 & 0 & \cos Q_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 30: Coordenadas correspondientes a los grados de libertad 4 y 5



Elaborado por el equipo de trabajo

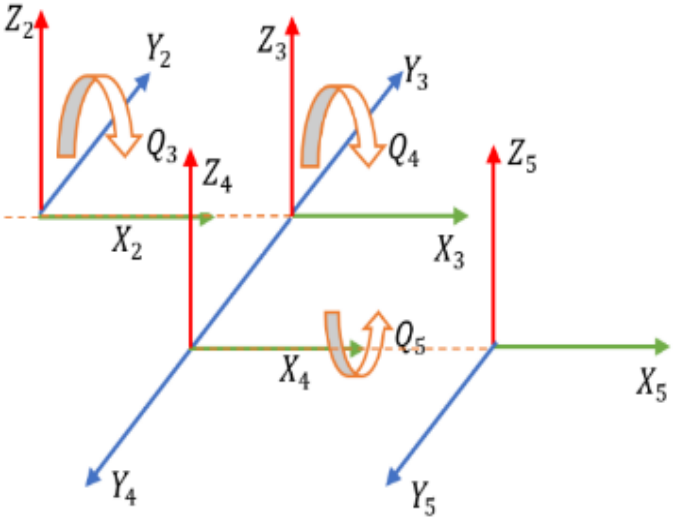
En la figura 31 primeramente, realizamos rotación en el eje Y_3 Y una translación en el eje Y_3 de la evaluación se tiene:

Ecuación 8: Matriz de articulación 3 a 4

$$H_4^3 = R_{Y_3} T_{Y_3}$$

$$H_4^3 = \begin{bmatrix} \cos Q_4 & 0 & \sin Q_4 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin Q_4 & 0 & \cos Q_4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & L_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 31: Coordenadas a los grados de libertad 5 y efector final



Elaborado por el equipo de trabajo

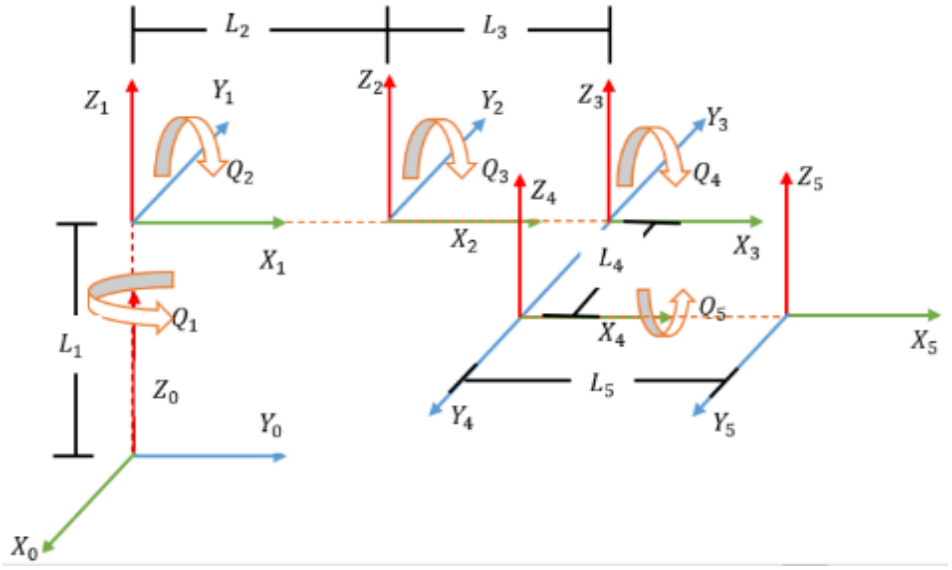
En la figura 31, realizamos una rotación en el eje X_4 luego realizamos una traslación para llegar al efector final, de la evaluación se tiene la ecuación 9.

Ecuación 9: Matriz de articulación 4 a 5

$$H_5^4 = R_{X_4} T_{X_4}$$

$$H_5^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos Q_5 & -\sin Q_5 & 0 \\ 0 & \sin Q_5 & \cos Q_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & L_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

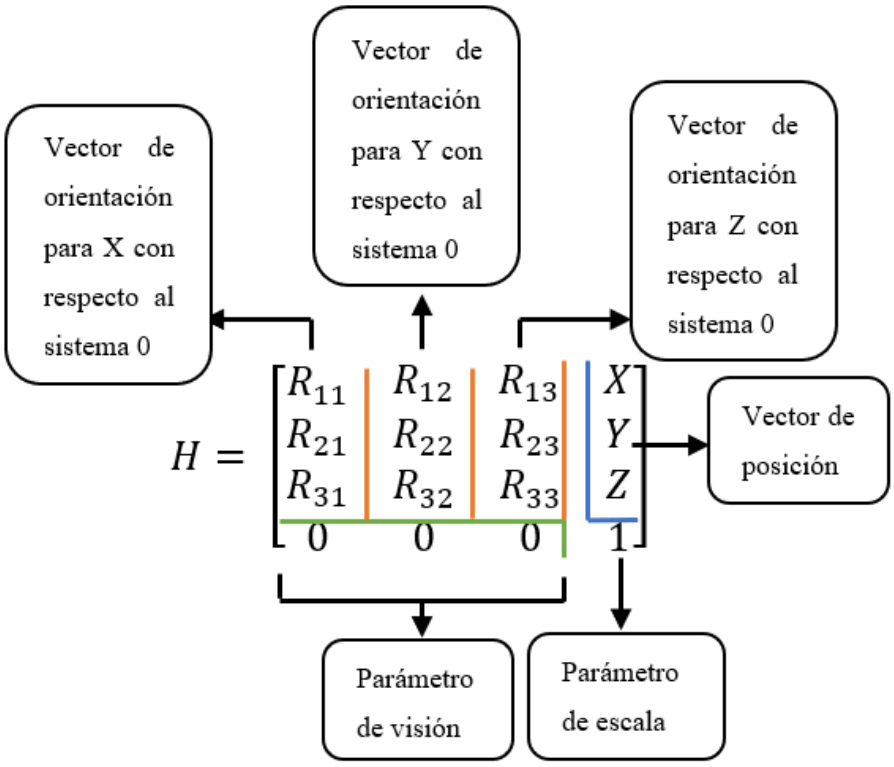
Figura 32: Coordenadas a los grados de libertad 5 y efector final



Elaborado por el equipo de trabajo

La evaluación final está dada por una matriz homogénea el cual tiene componentes como orientación, posición, visión y escala.

Figura 33: Partes de una matriz homogénea



Elaborado por el equipo de trabajo

Luego de obtener las matrices de traspaso de todas las articulaciones, tendremos una matriz general homogénea que tiene la forma de la figura 33, la matriz final lo obtenemos de la pre - multiplicación de las matrices obtenidas anteriormente.

Ecuación 10: Matriz de articulación 0 a 5

$$H_5^0 = H_1^0 H_2^1 H_3^2 H_4^3 H_5^4$$

De la matriz general homogénea tomamos solo los valores de la posición, ya que en la cinemática directa obtenemos una posición *X*, *Y*, *Z* con los valores articulares Q_1, Q_2, Q_3, Q_4, Q_5 que nosotros establecemos, la evaluación se realizó en el programa Matlab.

$$X = L_3(\cos Q_1 * \cos Q_3 * \cos(Q_2 - pi/2) - \cos Q_1 * \sin Q_3 * \sin(Q_2 - pi/2)) + L_4 * \sin Q_1 + L_5 * (\cos Q_4 * (\cos Q_1 * \cos Q_3 * \cos(Q_2 - pi/2) - \cos Q_1 * \sin Q_3 * \sin(Q_2 - pi/2)) - \sin Q_4 * (\cos Q_1 * \cos Q_3 * \sin(Q_2 - pi/2) + \cos Q_1 * \cos(Q_2 - pi/2) * \sin Q_3)) + L_2 * \cos Q_1 * \cos(Q_2 - pi/2)$$

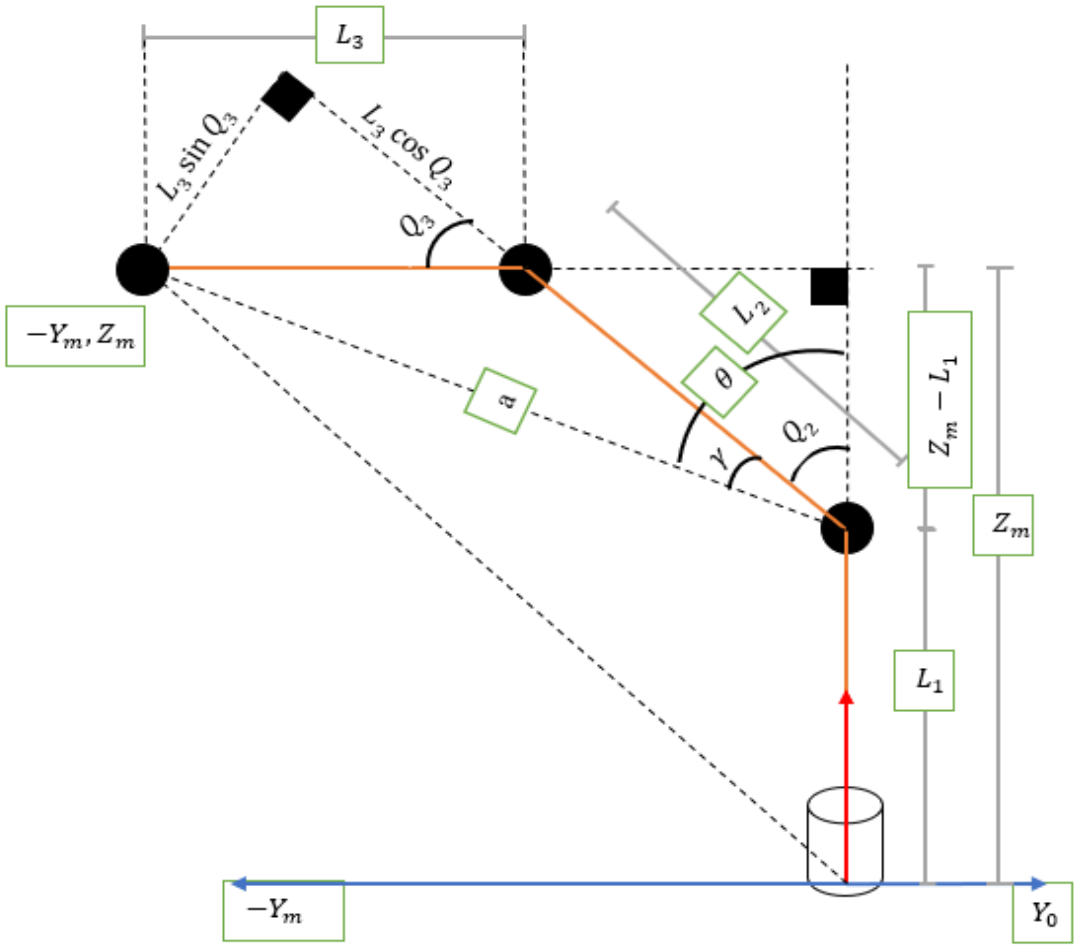
$$Y = L_3\left(\cos Q_3 * \cos(Q_2 - \frac{pi}{2}\right) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(Q_2 - pi/2)) - L_4 * \cos Q_1 + L_5 * (\cos Q_4 * (\cos Q_3 * \cos(Q_2 - pi/2) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(Q_2 - pi/2)) - \sin Q_4 * (\cos Q_3 * \sin Q_1 * \sin(Q_2 - pi/2) + \cos(Q_2 - pi/2) * \sin Q_1 * \sin Q_3)) + L_2 * \cos(Q_2 - pi/2) * \sin Q_1$$

$$Z = L_1 - L_2 * \sin(Q_2 - pi/2) - L_5(\cos Q_4 * (\cos Q_3 * \sin(Q_2 - \frac{pi}{2}) + \cos(Q_2 - pi/2) * \sin Q_3) + \sin Q_4 * (\cos Q_3 * \cos(Q_2 - pi/2) - \sin Q_3 * \sin(Q_2 - pi/2))) - L_3(\cos Q_3 * \sin(Q_2 - pi/2) + \cos(Q_2 - pi/2) * \sin Q_3)$$

3.9.1.2 Cinemática Inversa

Teniendo en cuenta que en la cinemática inversa obtenemos los valores articulares Q_1, Q_2, Q_3, Q_4, Q_5 a partir de los puntos *X*, *Y*, *Z* que nosotros establecemos. El método que se utilizó para calculo cinemático inverso fue el desacoplamiento cinemático, el cual se realiza en un análisis de dos partes, la primera es el análisis por geometría, en la esta etapa se toma solo los tres primeros grados de libertad para hacer el cálculo de las tres primeras articulaciones.

Figura 34: Asignación de ángulos en la vista vertical



Elaborado por el equipo de trabajo

En el cálculo de la cinemática inversa lo primero que se define es el X_m, Y_m y Z_m que son las coordenadas de la muñeca, para calcular estos valores se toma el valor de la orientación de la matriz de cinemática directa que hallamos anteriormente, en donde X_f, Y_f, Z_f son parámetros del vector de posición, R_{11}, R_{12}, R_{13} son parámetros del vector de orientación del eje en el que se retorna del efector final a la muñeca del brazo robot en este caso se retorna en el eje X y L_5 es la distancia entre la muñeca y el efector final, los parámetros de la muñeca se representan por las siguiente ecuaciones:

Ecuación 11: Fórmula de muñeca en eje X

$$X_m = X_f - L_5 R_{11}$$

Ecuación 12: Fórmula de muñeca en eje Y

$$Y_m = Y_f - L_5 R_{12}$$

Ecuación 13: Fórmula de muñeca en eje Z

$$Z_m = Z_f - L_5 R_{13}$$

Donde deducimos que: $Q_1 = \tan^{-1}(\frac{-Y_m}{X_m})$, luego de la figura 34 calculamos la $\tan\theta_1$ lo cual sería igual a la ecuación 14.

Ecuación 14: Identidad trigonométrica de tangente en teta 1

$$\tan\theta_1 = \frac{-X_m}{Y_m - L_1}$$

Y el valor de θ_1 se expresa en la ecuación 15.

Ecuación 15: Arco tangente en teta 1

$$\theta_1 = \tan^{-1}(\frac{-X_m}{Y_m - L_1})$$

Calculamos la $\tan\gamma$ lo cual sería igual a la ecuación 16.

Ecuación 16: Identidad trigonométrica de tangente en gamma

$$\tan\gamma = \frac{L_3 \sin Q_3}{L_2 + L_1 \cos Q_3}$$

Y el valor de γ se expresa en la ecuación 17.

Ecuación 17: Arco tangente en gamma

$$\gamma = \tan^{-1}(\frac{L_3 \sin Q_3}{L_2 + L_1 \cos Q_3})$$

Y deducimos el valor de la articulación Q_2 en la ecuación 18.

Ecuación 18: Fórmula para hallar Q2

$$\theta = Q_2 + \gamma$$

Despejando Q_2 tenemos la ecuación 19.

Ecuación 19: Fórmula de Q2

$$Q_2 = \theta - \gamma$$

Calculamos el valor de a mediante la ecuación de Pitágoras, según la figura 34 para triángulo rectángulo, obtenemos dos formas de hallar el valor de a el cual esta descrito en la ecuación 20.

Ecuación 20: Fórmula de Pitágoras para hallar A

$$a^2 = (-X_m)^2 + (Y_m - L_1)^2$$

Reemplazando las variables X_m y Y_m se obtiene la ecuación 21.

Ecuación 21: Reemplazo con ecuaciones 11 y 12

$$a^2 = (L_3 \sin Q_3)^2 + (L_2 + L_3 \cos Q_3)^2$$

Resolviendo la ecuación 21 se obtiene la ecuación 22.

Ecuación 22: Solución de ecuación 21

$$a^2 = L_2^2 + L_3^2 + 2L_2L_3 \cos Q_3$$

Luego igualamos las ecuaciones 20 y 22 para calcular el valor de Q_3 y se obtiene la ecuación 23.

Ecuación 23: Fórmula para hallar Q3

$$(-X_m)^2 + (Y_m - L_1)^2 = L_2^2 + L_3^2 + 2L_2L_3 \cos Q_3$$

Seguidamente hacemos un cambio de variable para la ecuación 23 en el cual se obtiene la ecuación 24.

Ecuación 24: Cambio de variable para tener cos Q3.

$$\cos Q_3 = \frac{X_m^2 + (Y_m - L_1)^2 - L_2^2 - L_3^2}{2L_2L_3} = D$$

De la identidad trigonométrica se tiene la ecuación 25.

Ecuación 25: Identidad trigonométrica para Q3

$$(\cos Q_3)^2 + (\sin Q_3)^2 = 1$$

Despejando $sinQ_3$ obtenemos la ecuación 26.

Ecuación 26: Fórmula para sin Q3

$$\sin Q_3 = \pm\sqrt[2]{1-(\cos Q_3)^2}$$

Seguidamente hallamos la tangente de Q_3, que es una identidad trigonométrica y se muestra en la ecuación 27.

Ecuación 27: Tangente de Q3

$$\tan Q_3 = \frac{\sin Q_3}{\cos Q_3}$$

Luego reemplazamos los valores de las ecuaciones 24 y 26 en la ecuación 27 obteniendo la ecuación 28.

Ecuación 28: Reemplazo de variables en tan Q3

$$\tan Q_3 = \frac{\pm\sqrt[2]{1-D^2}}{D}$$

Despejando Q_3 de la ecuación 28 tenemos la ecuación 29.

Ecuación 29: Fórmula para Q3

$$Q_3 = \tan^{-1}\left(\frac{\pm\sqrt[2]{1-D^2}}{D}\right)$$

El segundo método a utilizarse es algebraico, donde sabemos que una matriz homogénea esta expresada en la ecuación 30.

Ecuación 30: Matriz general homogénea

$$H = \begin{bmatrix} R_{11} & R_{12} & R_{13} & X \\ R_{21} & R_{22} & R_{23} & Y \\ R_{31} & R_{32} & R_{33} & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para hallar el valor de las articulaciones Q_4 y Q_5 debemos de tener en cuenta las ecuaciones 31 y 32.

Ecuación 31: Forma 1 de matriz de articulación de 3 a 5

$$H_5^3 = (H_3^0)^T H_5^0$$

Ecuación 32: Forma 2 de matriz de articulación de 3 a 5

$$(H_3^0)^T H_5^0 = H_4^3 H_5^4$$

Tendremos dos matrices de los cuales extraemos R_{13} de ambos casos y los igualamos:

$$-\cos Q_5 = (\sin Q_3 * \cos(Q_2 - pi/2) - \sin Q_3 * \sin(Q_2 - pi/2) * (\cos Q_3 * \cos\left(\frac{pi}{2} - Q_2\right) + \sin Q_3 * \sin\left(\frac{pi}{2} - Q_2\right) - \left(\cos Q_3 * \sin(Q_2 - \frac{pi}{2}\right) + \cos(Q_2 - pi/2) * \sin Q_3) * \cos Q_3 * \sin(\frac{pi}{2} - Q_2) - \sin Q_3 * \cos(\frac{pi}{2} - Q_2)) - L_1 + L_3 * \cos Q_3 * \sin\left(\frac{pi}{2} - Q_2\right) - \sin Q_3 * \cos(\frac{pi}{2} - Q_2) + \sin(\frac{pi}{2} - Q_2) * L_2 * \sin(Q_2 - \frac{pi}{2}) - L_1 + L_3 * (\cos Q_3 * \sin(Q_2 - \frac{pi}{2}) + \cos(Q_2 - pi/2) * \sin Q_3 = K$$

Luego tenemos la identidad trigonométrica se obtiene la ecuación 33.

Ecuación 33: Identidad trigonométrica para Q5

$$(\cos Q_5)^2 + (\sin Q_5)^2 = 1$$

Despejando $sinQ_5$ tenemos la ecuación 34.

Ecuación 34: Fórmula para sin Q5

$$\sin Q_5 = \sqrt[2]{1 - (\cos Q_5)^2}$$

Hallando la $tanQ_5$.

Ecuación 35: Fórmula para tan Q5

$$\tan Q_5 = \frac{\sqrt[2]{1 - K^2}}{K}$$

Ecuación 36: Fórmula para Q5.

$$Q_5 = \tan^{-1}\left(\frac{\sqrt[2]{1 - K^2}}{K}\right)$$

De la matriz resultante en las ecuaciones 31 y 32 extraemos $R_{13}\ y\ R_{12}$ y tenemos:

$$\tan Q_4 = ((\left(L_3 * \cos Q_1 * \cos Q_3 * \cos(Q_2 - \frac{pi}{2}\right) - \cos Q_1 \sin Q_3 * \sin(Q_2 - pi/2))$$
$$+ L_2 * \cos Q_1 * \cos(Q_2 - \frac{pi}{2}) * L_1 + L_3 * \cos Q_3 * \sin\left(\frac{pi}{2} - Q_2\right)$$
$$- \sin Q_3 * \cos(\frac{pi}{2} - Q_2) + \sin Q_3 * \sin\left(\frac{pi}{2} - Q_2\right) * L_2 + \cos Q_3$$
$$* \cos(\frac{pi}{2} - Q_2) + \sin Q_3 * \sin\left(\frac{pi}{2} - Q_2\right) * \cos Q_1 * \cos Q_3 * \sin(Q_2$$
$$- pi/2) + \cos Q_1 * \cos(Q_2 - \frac{pi}{2}) * \sin Q_3 + \cos Q_3 * \sin\left(\frac{pi}{2} - Q_2\right)$$
$$- \sin Q_3 * \cos(\frac{pi}{2} - Q_2) * \cos Q_1 * \cos Q_3 * \cos(Q_2 - \frac{pi}{2}) - \cos Q_1$$
$$* \sin Q_3 * \sin(Q_2 - pi/2) * \cos Q_3 * \cos(Q_2 - \frac{pi}{2}) + \sin Q_3$$
$$* \sin\left(\frac{pi}{2} - Q_2\right) * \cos Q_3 * \sin\left(\frac{pi}{2} - Q_2\right) - \sin Q_3 * \cos(\frac{pi}{2} - Q_2)$$
$$* \cos Q_3 * \cos(Q_2 - \frac{pi}{2}) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(Q_2 - pi/2))$$
$$+ (L_3 * \sin Q_3 * \cos(Q_2 - \frac{pi}{2}) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(Q_2$$
$$- pi/2)) + L_2 * \cos(Q_2 - \frac{pi}{2}) * \sin Q_1 * L_1 + L_3 * \cos Q_3$$
$$* \sin\left(\frac{pi}{2} - Q_2\right) - \sin Q_3 * \cos(\frac{pi}{2} - Q_2) * \sin\left(\frac{pi}{2} - Q_2\right) * L_2)$$

Hallamos Q_4 en la solución siguiente.

$$Q_4 = \tan^{-1}(\left(L_3 * \cos Q_1 * \cos Q_3 * \cos(Q_2 - \frac{pi}{2}\right) - \cos Q_1 \sin Q_3 * \sin(Q_2 - pi/2))$$

$$+ L_2 * \cos Q_1 * \cos(Q_2 - \frac{pi}{2}) * L_1 + L_3 * \cos Q_3 * \sin\left(\frac{pi}{2} - Q_2\right)$$

$$- \sin Q_3 * \cos(\frac{pi}{2} - Q_2) + \sin Q_3 * \sin\left(\frac{pi}{2} - Q_2\right) * L_2 + \cos Q_3$$

$$* \cos(\frac{pi}{2} - Q_2) + \sin Q_3 * \sin\left(\frac{pi}{2} - Q_2\right) * \cos Q_1 * \cos Q_3 * \sin(Q_2$$

$$- pi/2) + \cos Q_1 * \cos(Q_2 - \frac{pi}{2}) * \sin Q_3 + \cos Q_3 * \sin\left(\frac{pi}{2} - Q_2\right)$$

$$- \sin Q_3 * \cos(\frac{pi}{2} - Q_2) * \cos Q_1 * \cos Q_3 * \cos(Q_2 - \frac{pi}{2}) - \cos Q_1$$

$$* \sin Q_3 * \sin(Q_2 - pi/2) * \cos Q_3 * \cos(Q_2 - \frac{pi}{2}) + \sin Q_3$$

$$* \sin\left(\frac{pi}{2} - Q_2\right) * \cos Q_3 * \sin\left(\frac{pi}{2} - Q_2\right) - \sin Q_3 * \cos(\frac{pi}{2} - Q_2)$$

$$* \cos Q_3 * \cos(Q_2 - \frac{pi}{2}) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(Q_2 - pi/2))$$

$$+ (L_3 * \sin Q_3 * \cos(Q_2 - \frac{pi}{2}) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(Q_2$$

$$- pi/2)) + L_2 * \cos(Q_2 - \frac{pi}{2}) * \sin Q_1 * L_1 + L_3 * \cos Q_3$$

$$* \sin\left(\frac{pi}{2} - Q_2\right) - \sin Q_3 * \cos(\frac{pi}{2} - Q_2) * \sin\left(\frac{pi}{2} - Q_2\right) * L_2)$$

3.9.1.3 Jacobiano

Primeramente, hallamos los componentes de una matriz jacobiana expresada en la ecuación 37.

Ecuación 37: Fórmula general de jacobiano

$$J = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} \end{bmatrix}$$

Para obtener cada componente de una matriz jacobiana derivamos la posición obtenida de la cinemática directa luego tenemos:

Para $\dot{X}$ tenemos:

Ecuación 38: Jacobiano fila 1 columna 1

$$J_{11} = d\left(\frac{X}{Q_1}\right)$$

Ecuación 39: Jacobiano fila 1 columna 2

$$J_{12} = d\left(\frac{X}{Q_2}\right)$$

Ecuación 40: Jacobiano fila 1 columna 3

$$J_{13} = d(\frac{X}{Q_3})$$

Ecuación 41: Jacobiano fila 1 columna 4

$$J_{14} = d(\frac{X}{Q_4})$$

Ecuación 42: Jacobiano fila 1 columna 5

$$J_{15} = d\left(\frac{X}{Q_5}\right)$$

Para $\dot{Y}$ tenemos:

Ecuación 43: Jacobiano fila 2 columna 1

$$J_{21} = d(\frac{Y}{Q_1})$$

Ecuación 44: Jacobiano fila 2 columna 2

$$J_{22} = d\left(\frac{Y}{Q_2}\right)$$

Ecuación 45: Jacobiano fila 2 columna 3

$$J_{23} = d\left(\frac{Y}{Q_3}\right)$$

Ecuación 46: Jacobiano fila 2 columna 4

$$J_{24} = d\left(\frac{Y}{Q_4}\right)$$

Ecuación 47: Jacobiano fila 2 columna 5

$$J_{25} = d\left(\frac{Y}{Q_5}\right)$$

Para $\dot{Z}$ tenemos:

Ecuación 48: Jacobiano fila 3 columna 1

$$J_{31} = d\left(\frac{Z}{Q_1}\right)$$

Ecuación 49: Jacobiano fila 3 columna 2.

$$J_{32} = d\left(\frac{Z}{Q_2}\right)$$

Ecuación 50: Jacobiano fila 3 columna 3.

$$J_{33} = d\left(\frac{Z}{Q_3}\right)$$

Ecuación 51: Jacobiano fila 3 columna 4.

$$J_{34} = d\left(\frac{Z}{Q_4}\right)$$

Ecuación 52: Jacobiano fila 3 columna 5.

$$J_{35} = d(\frac{Z}{Q_5})$$

Derivando cada componente tenemos:

$$\dot{X} = -L_4 * \cos Q_1 - L_3 * \left(\cos Q_3 * \cos(Q_2 + \frac{pi}{2}\right) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(T_2 + pi/2)) - L_5 * (\cos T_4 * (\cos T_3 * \cos(Q_2 + \frac{pi}{2}) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(T_2 + pi/2)) - \sin Q_4 * (\cos T_3 * \cos(Q_2 + \frac{pi}{2}) + \cos(Q_2 + \frac{pi}{2}) * \sin Q_1 * \sin Q_3))) - L_2 * \cos(Q_2 + \frac{pi}{2}) * \sin Q_1 - L_3 * (\cos Q_1 * \cos Q_3 * \sin(T_2 + pi/2) + \cos Q_1 * \cos(Q_2 + \frac{pi}{2}) * \sin Q_3) - L_5(\cos Q_4 * (\cos Q_1 * \cos Q_3 * \sin(T_2 + pi/2) + \cos Q_1 * \cos(Q_2 + \frac{pi}{2}) * \sin Q_3) + \sin Q_4 (\cos Q_1 * \cos Q_3 * \cos(Q_2 + \frac{pi}{2}) - \cos Q_1 * \cos Q_3 * \sin(T_2 + pi/2))) + L_2 * \cos Q_1 * \sin(T_2 + pi/2) - L_3 * \cos Q_1 * \cos Q_3 * \sin(T_2 + pi/2) + \cos Q_1 * \cos(Q_2 + \frac{pi}{2}) * \sin Q_3) - L_5(\cos Q_4 * (\cos Q_1 * \cos Q_3 * \sin(T_2 + pi/2) + \cos Q_1 * \cos(Q_2 + \frac{pi}{2}) * \sin Q_3) + \sin Q_4 * (\cos T_1 * \cos T_3 * \cos(Q_2 + \frac{pi}{2}) - \cos T_1 * \cos T_3 * \cos(Q_2 + \frac{pi}{2}))) - L_5(\cos Q_4 * (\cos Q_1 * \cos Q_3 * \sin(T_2 + pi/2) + \cos Q_1 * \cos(Q_2 + \frac{pi}{2}) + \cos T_1 * \cos T_3 * \cos(Q_2 + \frac{pi}{2})) + \sin Q_4 * (\cos T_1 * \cos T_3 * \cos(Q_2 + \frac{pi}{2}) - \cos T_1 * \cos T_3 * \cos(Q_2 + \frac{pi}{2}))),0$$

$$\dot{Y} = L_3 * \left(\cos Q_3 * \cos(Q_2 + \frac{pi}{2}\right) - \cos Q_1 * \sin Q_4 * \sin(Q_2 + pi/2) - L_4 * \sin Q_1$$

$$+ L_5 * (\cos T_4 * \left(\cos Q_3 * \cos Q_1 * \cos(Q_2 + \frac{pi}{2}\right) - \cos Q_1 * \sin Q_3$$

$$* \sin(T_2 + pi/2)) - \cos T_4 * \left(\cos Q_3 * \cos Q_1 * \cos(Q_2 + \frac{pi}{2}\right) + \cos Q_1$$

$$* \sin(Q_2 + pi/2) * \sin Q_3)) + \cos Q_1 * \cos(Q_2 + \frac{pi}{2}), -L_3 \cos Q_1$$

$$* \sin Q_4 * \sin(Q_2 + pi/2) + \sin(T_2 + pi/2) * \cos Q_1 * \sin Q_3 *) - L_5$$

$$* (\cos T_4 * \left(\cos Q_3 * \cos Q_1 * \cos(Q_2 + \frac{pi}{2}\right) + \cos(Q_2 + \frac{pi}{2}) * \sin Q_1$$

$$* \sin Q_3) + \sin Q_4 * \cos Q_3 * \cos(Q_2 + \frac{pi}{2}) * \sin Q_1 - \sin Q_1 * \sin Q_3$$

$$* \sin(Q_2 + pi/2) * \sin Q_3)) - L_2 * \sin Q_1 * \sin(Q_2 + pi/2), -L_3$$

$$* (\cos Q_3 * \sin Q_1 * \sin(Q_2 + pi/2) + \cos(Q_2 + \frac{pi}{2}) * \sin Q_1 * \sin Q_3))$$

$$- L_5 * (\cos T_4 * (\cos Q_3 * \sin Q_1 * \sin(Q_2 + pi/2) + \cos(Q_2 + \frac{pi}{2})$$

$$* \sin Q_1 * \cos Q_3) + \sin Q_4 * \cos Q_3 * \cos(Q_2 + \frac{pi}{2}) * \sin Q_1 - \sin Q_1$$

$$* \sin Q_3 * \sin(Q_2 + pi/2))), -L_5 * (\cos T_4 * (\cos Q_3 * \sin Q_1 * \sin(Q_2$$

$$+ pi/2) + \cos(Q_2 + \frac{pi}{2}) * \sin Q_1 * \cos Q_3) + \sin Q_4 * \cos Q_3 * \cos(Q_2$$

$$+ \frac{pi}{2}) * \sin Q_1 - \sin Q_1 * \sin Q_3 * \sin(Q_2 + pi/2))),0$$

$$\dot{Z} = 0, \quad L_2\cos(Q_2 - pi/2) - \sin Q_4 * \cos Q_3 * \cos(Q_2 + \frac{pi}{2}) + \sin Q_3 * \sin(Q_2 + pi/2)) + \sin Q_4 * \cos Q_3 * \sin(Q_2 + pi/2) - \cos(Q_2 + \frac{pi}{2}) * \sin Q_3)) - L_3 * \cos Q_3 * \cos(Q_2 + \frac{pi}{2}) + \sin Q_3 * \sin(Q_2 + pi/2)), L_5 * \sin Q_4 * \cos Q_3 * \sin(Q_2 + pi/2) + \sin Q_3 * \sin(Q_2 + pi/2)) + \sin Q_4 * \cos Q_3 * \sin(Q_2 + pi/2) - \cos(Q_2 + \frac{pi}{2}) * \sin Q_3)) + L_3 * \cos Q_3 * \cos(Q_2 + \frac{pi}{2}) + \sin Q_3 * \sin(Q_2 + pi/2)), L_5 * \sin Q_4 * \cos Q_3 * \sin(Q_2 + pi/2) + \sin Q_3 * \sin(Q_2 + pi/2)) + \sin Q_4 * \cos Q_3 * \sin(Q_2 + pi/2) - \cos(Q_2 + \frac{pi}{2}) * \sin Q_3)),0$$

3.9.2 Análisis del Sistema de Visión Artificial

En este análisis estimaremos los puntos coordenados del plano fisco en una interfaz virtual mediante la visión artificial, para ello es necesario realizar una serie de pasos como la adquisición de imágenes, procesamiento y el detector de objetos para lograr así una aproximación con el plano coordenado del mundo real.

3.9.2.1 Adquisición de Imágenes del Entorno

En primer lugar, necesitamos elegir nuestra cámara web que nos permitirá capturar datos del entorno exterior, en el mercado existen varios tipos y precios de cámara con características diferentes, en la figura 35 se ve la cámara Logitech C920e con entrada USB y micrófono incluido.

Figura 35: Cámara Logitech C920e

Elaborado por el equipo de trabajo

En la tabla 11, la cámara web tienen distintas características como son la cantidad de pixeles que tendrá la foto y unos 30 fotogramas por segundo, lo cual nos servirá para construir un video a partir de imágenes, el enfoque es automático y la lente tiene dos opciones, podría ser de plástico en cámaras baratas y de cristal en cámaras mejores que ofrece una mayor capacidad de recogida de luz y reproducción de color.

Se necesita trabajar en una resolución de 640x480 pixeles para tener una mejor fluidez al realizar un video streaming.

Tabla 11: Caracteristicas de Cámara Logitech c920e

	CÁMARA LOGITECH C920E
Resolución	1080p/30fps (1920x1080) 720p/30 fps (1280 x720)
Enfoque	Automático
Megapixels	3
Lente	Cristal
Zoom digital	1.2x
Sensor	Cmos

Elaborado por el equipo de trabajo

La cámara Logitech C920e tiene una mejor elaboración en el lente que nos permite tener una mejor vista de lo queremos observar en la imagen.

Figura 36: Imagen de cámara Logitech c920e en resolución 1920x1080

Elaborado por el equipo de trabajo

Por ello usaremos una cámara Logitech c920e. En una cámara estándar se necesita corregir las distorsiones a través de un método llamado calibración de cámara, la cual se realiza por cálculos matemáticos matriciales. Las coordenadas de una cámara que esta fija en un punto, empieza su ubicación en el lado superior izquierdo, en la figura 37 se ve que la escala de medida de Xc, empieza del lado izquierdo a el lado derecho y la escala de medida de Yc, empieza del lado de arriba hacia el lado de abajo. En nuestro sistema de coordenadas del brazo robot tiene una configuración distinta, porque se encuentra en el medio del plano teniendo Xbr en el centro con valores positivos y negativos y Ybr como positivos.

Con el complemento OpenCV para Python, podemos obtener el punto del inicio de la coordenada de la cámara y transformarlo al punto del inicio de la referencia del brazo robot, para ello es necesario realizar una transformación en la forma de la imagen, mover el punto de referencia de la cámara a otro punto deseado. El plano tiene una medida de 116x83cm y los pixeles tienen que adecuarse a esta medida.

Figura 37: Sistema de referencia de cámara y brazo robot

Elaborado por el equipo de trabajo

Una vez entendido esto, tenemos que substraer la región de interés donde trabajaremos, que es el plano milimétrico detallado con medidas verídicas y está enmarcado con cinta negra en las esquinas como se ve en la figura 37.

Un método para encontrar la región de interés, es la de detección de colores, para ello usamos el espectro del espacio RGB. En la figura 38 se muestra los colores rojo, verde, azul y amarillo en forma redonda para realizar el procesamiento de imagen y hacer un seguimiento de estos objetos más adelante. Se necesita convertir al espacio de color HSV.

Figura 38: Colores rojo verde azul y amarillo en formas redondas

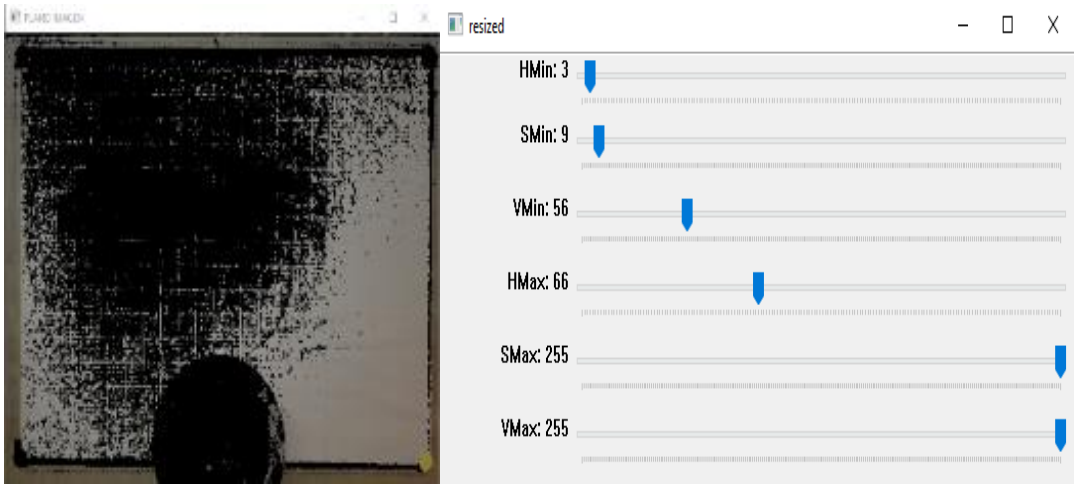
Elaborado por el equipo de trabajo

El espacio de color HSV en el complemento OpenCV se interpreta mediante un rango de valores como Hue(0-179), Saturation(0-255) y Value(0-255). Se ve que cada color se sitúa en una esquina del marco del plano milimétrico y se realiza una progresión de color en los valores mencionados.

3.9.2.2 Procesamiento de Imágenes Capturadas

En el procesamiento de imagen encontraremos el color rojo, azul, verde y amarillo en el espacio HSV. Se realiza mediante una GUI con TrackBar y un video streaming, en el configuramos los rangos de cada valor mencionado anteriormente. En la figura 39 se ve demostración de la umbralización del espacio HSV, para los valores mencionados mínimo y máximo. Después se obtiene una imagen binaria con el color detectado, se convierte en una imagen en blanco y negro, blanco es el color que queremos obtener y negro que es el fondo, a esto lo nombramos como mascara, luego aplicamos una compuerta AND en esa imagen binaria para mostrar el color en el espacio RGB y lo aplicamos a nuestra mascara.

Figura 39: Barras de espacio del color rojo, azul, amarillo y verde



Elaborado por el equipo de trabajo

Ahora necesitamos obtener el punto centro de nuestro color que tiene forma circular, para ello emplearemos la técnica de contornos en OpenCV, a través de la imagen

binaria que se obtuvo después de la umbralización, aplicamos los momentos para la descripción del color y encontramos los contornos de la máscara de color con jerarquía y aproximación de puntos. Posteriormente guardamos y buscamos el área más grande para ubicarlo como circulo, realizamos la operación de radio de circunferencia y dibujamos un círculo en él. La figura 40 se ve una demostración de lo planteado.

Figura 40: Punto centro en los colores

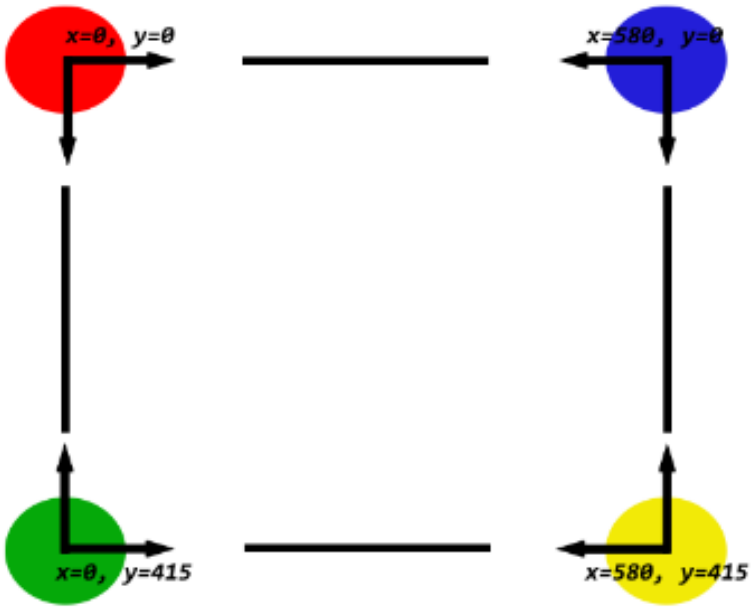
Elaborado por el equipo de trabajo

Una vez obtenido estos puntos en el color rojo, verde, azul y amarillo necesitamos hacer una transformación de perspectiva para recortar nuestra región de interés usando los puntos mencionados. La transformación de perspectiva usa una matriz de 3x3 con cuatro puntos de imagen y otros cuatro para trasformar a la imagen que se quiere recortar. Los 4 puntos de salida vienen a ser la cantidad de pixeles que tendremos según el plano coordenado, en este caso nuestro plano tiene una medida de 116cm x 83 cm o en pixeles de 116x83 pixeles.

Poner esta equivalencia en pixeles sería muy pequeña y no habría una buena apreciación, por lo que multiplicamos x5 a cada medida. La figura 41 muestra una dimensión de 580 x 415 en la imagen.

Finalmente obtenemos la región de interés donde se realizó la transformación de puntos La figura 41 muestra los nuevos puntos coordenados de la transformación de perspectiva y son (0,0), (580,0), (0,415) y (580,415), esta función tomo dos puntos superiores (0,0), (0,640) y dos inferiores (0,480), (640,480) para su ejecución.

Figura 41: Coordenadas en cada punto centro de color

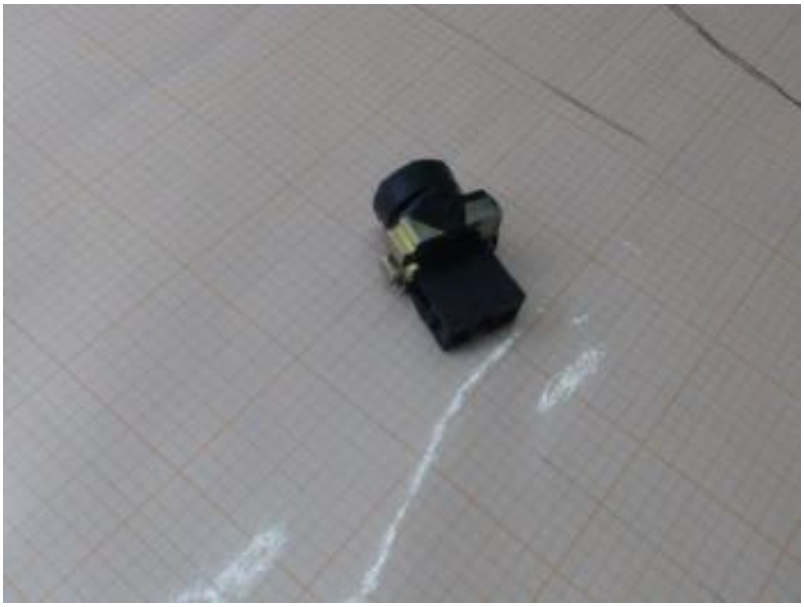


Elaborado por el equipo de trabajo

3.9.2.3 Detección de Objetos

En este paso nos centramos en obtener las características del objeto, para este caso nuestras muestras son el destornillador y el pulsador. En la figura 42 y 43 se ve las muestras que seleccionamos de nuestra población, usamos el método Haar Cascade que está basado en la unión de clasificadores débiles, en estos clasificadores cada uno realiza un análisis de una porción de la imagen, se considera débil por los falsos positivos, pero cuando se combinan los resultados en conjuntos son potentes.

Figura 42: Pulsador industrial



Elaborado por el equipo de trabajo

Las porciones de imagen del objeto deben ser solo de la muestra seleccionada en el entorno en donde se encontrará, el programa Cascade Trainer GUI será en donde se realiza el entrenamiento de estas porciones, se realiza operaciones básicas en estas imágenes captadas como recorte y escalamiento.

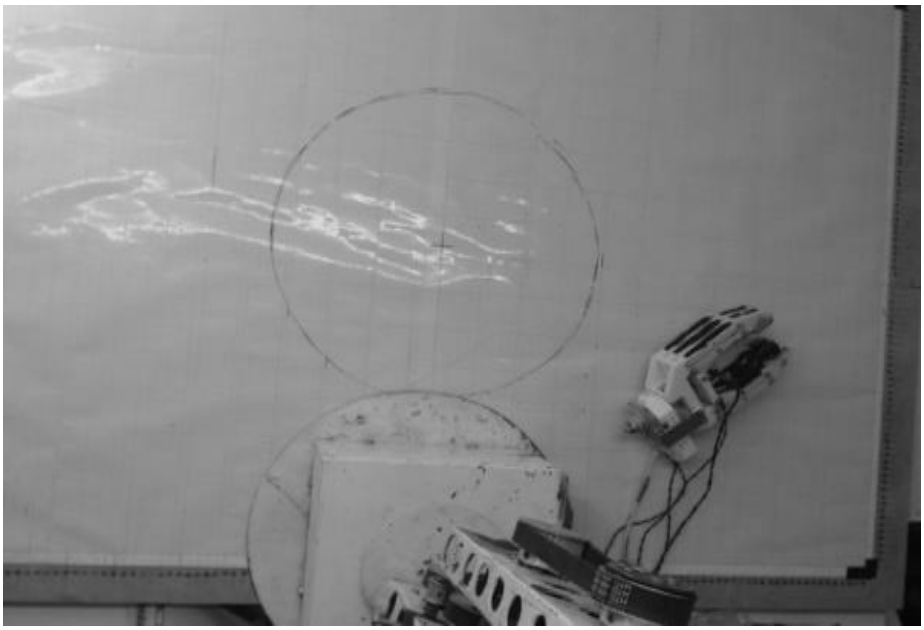
Figura 43: Destornillador estrella

Elaborado por el equipo de trabajo

El proceso del entrenamiento tiene una serie de pasos a realizar para obtener un buen resultado, se comienza primero por capturar fotos en una resolución de 640x480 pixeles, para la cantidad de imágenes positivas tomadas usamos 200 imágenes (son la imagen verdadera donde está el objeto) y 300 imágenes negativas (son imágenes del entorno en donde no se encuentra el objeto).

En la figura 44 se ve como es una imagen negativa que necesita una conversión a escala de grises y escalada en una relación de 2 a 1 por ejemplo (600x300), haremos esto hasta obtener la cantidad antes mencionada.

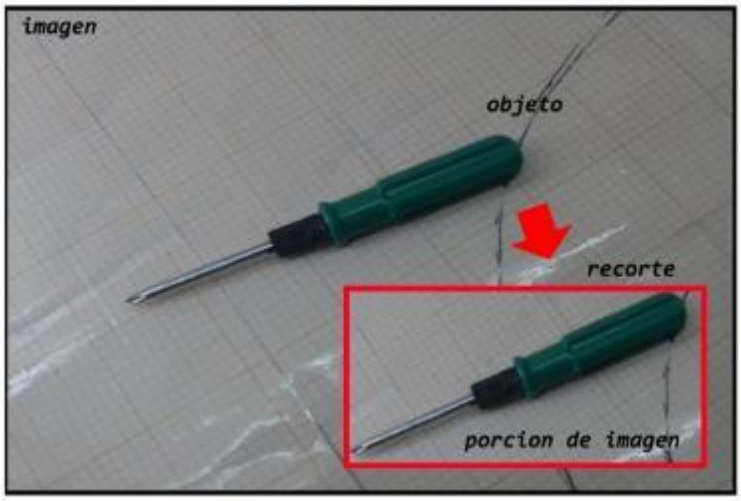
Figura 44: Imagen sin el objeto a obtener en escala de grises



Elaborado por el equipo de trabajo

Una vez tomadas las imágenes es necesario asignar una carpeta principal de cualquier nombre, dentro de ella una carpeta con nombre “p” para positivas y “n” para negativas.

Figura 45: Recorte para la porción de imagen



Elaborado por el equipo de trabajo

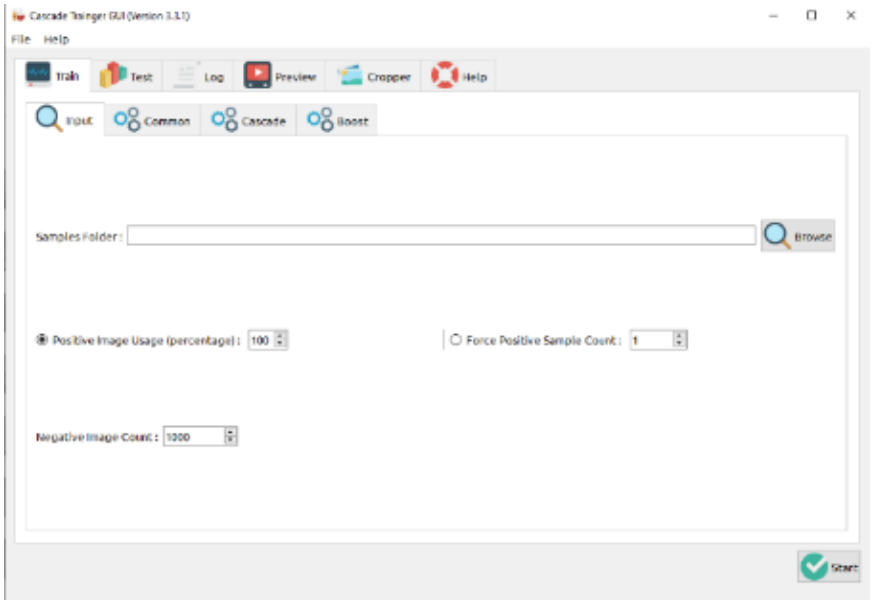
En la figura 45 se ve el proceso de recorte para la porción de imagen. Luego se realiza un escalamiento de imagen dependiendo si tu imagen es rectangular o cuadrada es recomendable usar estas medidas (32x24 y 24x24) como la figura 46. Finalmente se almacena en la carpeta "p" para cada grupo de imagen escalada.

Figura 46: Escalamiento de la porción de imagen

Elaborado por el equipo de trabajo

Terminado esto, ahora tenemos el dataset de nuestras imágenes preparadas. Seguidamente procedemos a usar el programa Cascade Trainer GUI para el entrenamiento de nuestras imágenes, nos basaremos solo en la pestaña "Train" y sus subpestañas. En la figura 47 se observa la interfaz de este programa.

Figura 47: Programa cascade trainer gui



Elaborado por el equipo de trabajo

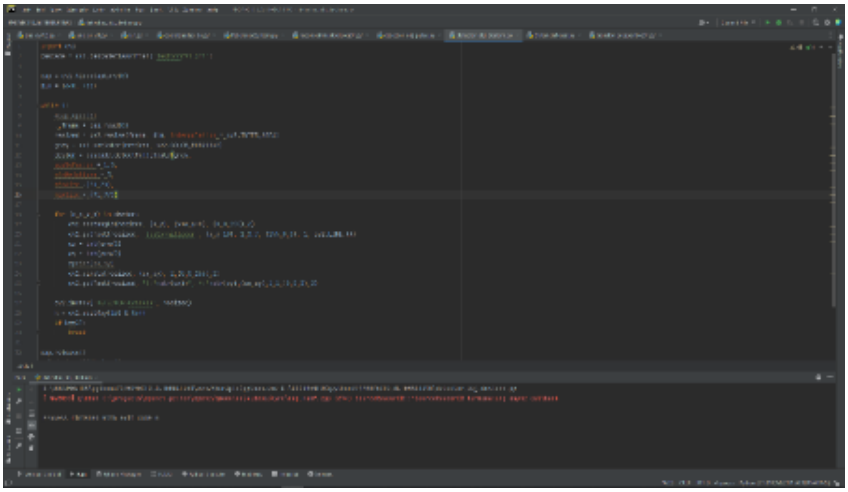
Seleccionamos en nuestra pestaña “input”, nuestra carpeta de origen donde están las subcarpetas “p” y “n”, ponemos una cantidad de imágenes positivas de 200, negativas 300. En la pestaña Common seleccionaremos la cantidad de etapas de entrenamiento que será 20 y la memoria ram que será 2048.

Luego en la pestaña Cascade seleccionamos el tamaño de nuestra escala que será de 24x24 siendo una imagen cuadrada y 32x24 siendo una imagen rectangular, seguidamente asignamos el método “HAAR” y su característica “BASIC”. El resto de pestañas que igual y empezamos con el entrenamiento.

Una vez terminado el entrenamiento de nuestro modelo, en el hay una lista de archivos en nuestra carpeta origen de las cuales en la carpeta “classifier” obtenemos nuestro archivo “cascade.xml”, que es el resultado de nuestro entrenamiento. El desarrollo de nuestra verificación del modelo entrenado se programa convirtiendo el frame en escala de grises y manipulando estos parámetros importantes como scaleFactor, minNeighbors, minSize y maxSize. Se ejecuta un “for” en donde especificamos 4

parámetros (x,y,w,h), dibujamos un rectángulo sobre la porción de imagen, obtenemos el punto centro pintándolo e imprimiendo las coordenadas con respecto a la cámara.

Figura 48: Desarrollo de algoritmo para el detector

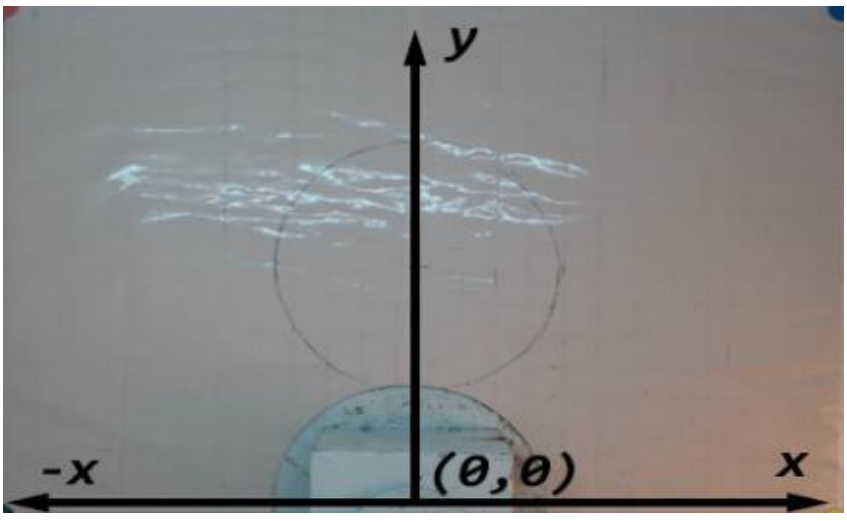


Elaborado por el equipo de trabajo

3.9.2.4 Estimación

Una vez terminado estos métodos en este proyecto como son el método de detección de colores y transformación de perspectiva, ahora que tenemos nuestra región de interés, y nos queda corregir nuestras coordenadas con respecto al brazo, con el complemento OpenCV realizamos el cambio de ubicación en nuestras coordenadas de la cámara al del brazo robot como en la figura 49.

Figura 49: Sistema de coordenadas con respecto al brazo robot



Elaborado por el equipo de trabajo

En la figura 49 el punto coordenado del frame está en (x,y), x se traslada al centro inferior de la imagen y partiéndola en dos partes para obtener coordenadas positivas y negativas, del mismo modo se realiza en el eje Y, que se traslada y cambia de orientación con respecto al brazo.

Finalmente para este análisis se incorpora el modelo entrenado de nuestro detector de objeto realizado por Haar Cascade y se pretende localizar la ubicación de los objetos que fueron entrenados y mostrar sus coordenadas (x,y) con respecto a la corrección de coordenadas. En los resultados se ve que los objetos que son diferentes al entrenado no deben encontrarse en nuestro sistema de coordenadas con estimación de posición.

3.9.3 Análisis del Sistema de Patrones de Voz

En este análisis veremos el control de los sistemas analizados anteriormente por patrones de voz, esto tiene como desarrollo un asistente virtual en Python a través de librerías como pyttsx3 y speech recognition, nos basamos en el desarrollo de software con bibliotecas que ejecutan varios procesos a la vez. Para ello realizamos paso como la adquisición de audio, desarrollo de un asistente virtual y el enlazamiento con los demás sistemas.

3.9.3.1 Adquisición de Audio

En primer lugar, necesitamos un micrófono para la entrada de voz, existen de varios tipos como los mencionados en el marco teórico. En la figura 50 podemos ver un micrófono de solapa llamado lavalier. Este es omnidireccional, claro, limpio en la entrada de la voz y posible de conseguir para el público.

Figura 50: Micrófono lavalier con cable usb

Elaborado por el equipo de trabajo

Podemos ver sus características en la tabla 12.

Tabla 12: Características de micrófono

VALORES	MICRÓFONO ESTÁNDAR
Voltaje	5V
Interfaz	USB 2.0
sensibilidad	-28dB ± 3dB
Frecuencia	50Hz-20KHz
Impedancia	≤2.2kΩ
Longitud de cable	190cm

Elaborado por el equipo de trabajo

Realizamos una prueba de funcionamiento del micrófono y verificamos su sensibilidad con una frecuencia de muestreo de 48 KHz en la página web https://es.mictests.com como en la figura 51.

Figura 51: Prueba de micrófono

Elaborado por el equipo de trabajo

Ahora podemos ubicar el micrófono en una parte cercana a nuestro plano milimétrico. El micrófono tiene una sensibilidad baja por lo que en distancias largas no se obtiene una buena captura de la voz, al final lo que se quiere es una interacción hombre-maquina con estos sistemas ejecutándose en procesos separados.

3.9.3.2 Interfaz del Asistente Virtual

El desarrollo del software se basa en una interfaz gráfica con tkinter, para que pueda haber una interacción con el usuario, primero configuramos nuestro conversor de texto a voz, le damos una variable y elegimos una voz que por defecto viene instalado en nuestra computadora, son 4 voces las que están disponibles para Microsoft (español, español-México, ingles hombre, ingles mujer).

Luego configuramos nuestro micrófono que por defecto el sistema no incluye el que vamos a usar, una vez configurado procedemos con el diseño de nuestra interfaz, que lo más importante es el botón que da inicio a nuestro programa y así evitamos el cruce con los demás programas como visión artificial y robótica al momento de llamarlos.

Un diseño que haremos con widgets como título en este caso “ANA-AV”, imagen de bienvenida y caja de texto para mostrar información, los comandos a usar como animaciones son opcionales del desarrollador para tener una mejor presentación.

Una vez entendido esto, necesitamos desarrollar el algoritmo que permita al programa entender nuestra voz y que lo convierta a texto, para ello definimos funciones como “talk” y “listen”. Para reproducir un texto como voz usamos el módulo pyttsx3 y funciona con o sin conexión a internet, en la función escuchar se hace un ajuste del ruido ambiente con el módulo “speech recognition”, y se mezcla con la función “talk”. Luego se escoge el lenguaje de reconocimiento de voz en este caso español.

Ahora se desarrolla los patrones de voz y sus respectivas acciones, cada acción viene dada con una función, en ella se especifica lo que hará, lo que necesitamos es abrir cada programa con cada acción por lo que necesitamos ejecutar el método de hilos que integra la biblioteca Python para abrir varios programas a la vez.

Para la ejecución de todas estas acciones mediante patrones, se necesita una función principal que activara el botón START, en ella se busca el patrón en la lista de ordenes cuando el usuario empieza hablar; si reconoce la voz el programa busca el patrón, ejecuta su acción, termina su acción y sigue escuchando, pero si no se reconoce la voz, el programa dice que no lo ha escuchado. De esta manera se ejecuta un bucle sin fin hasta escuchar el patrón “termina” que finaliza el ciclo o el patrón “ciérrate” que cierra la interfaz.

3.9.3.3 Enlazamiento

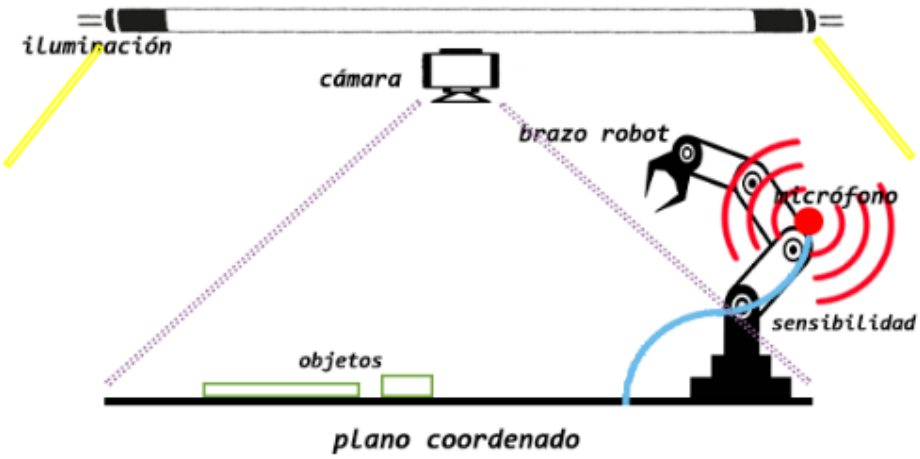
Finalmente, en este análisis tenemos que resolver los problemas de enlazamiento de programas, como la acción cuando se da el patrón “pásame el destornillador” una vez ejecutado la interfaz de la visión artificial. En este proceso está el inconveniente de mover las coordenadas a otro programa de ejecución como la cinemática inversa con conexión a Arduino, este paso podemos usar un almacenamiento de datos en un tipo de formato “destornillador.txt”, para luego abrirlo y obtener esos puntos que se insertaran en el

programa de cinemática inversa. De la misma manera para el patrón “pásame el pulsador”.

Para el patrón “regresa objeto” luego de que el brazo robot obtuvo el objeto, se llama a otro programa de cinemática inversa, este tiene un punto (x, y, z) configurado por defecto a otro lado del plano, diferente a donde se encuentran los objetos.

En la figura 53, podemos visualizar un diseño de lo que se quiere hacer con todos los sistemas vistos anteriormente, esperando ser llamados a partir de un código principal que es el asistente.

Figura 53: Esquema del resultado final



Elaborado por el equipo de trabajo

3.9.4 Análisis de Materiales

3.9.4.1 Análisis de los Motores Paso a Paso

El presente análisis se realizó un estudio de estos motores como el torque, la eficiencia y la temperatura de funcionamiento de cada motor utilizado para evitar el deterioro y también la eficiencia. Cada motor tiene una relación de reducción de engranaje y está indicado en el Anexo 2, esto será importante al momento de obtener la compensación que se debe dar al motor para obtener el control por ángulo. Del mismo modo los Micro pasos están indicados en el anexo B, el trabajo se realizó en su mayoría

con 200 micro pasos. En la formula siguiente se ve como se planteó esto y su resultado aparece en el código de programación Anexo 1: Arduino, recepción de datos y control de motores.

Ecuación 53: Fórmula de compensación para motores PAP

$$Comp. = \frac{Micropasos \times Reduccion\ del\ Gearbox}{360°}$$

Primeramente, se puso a evaluó el motor NEMA 17 que tiene una relación de transmisión 1/50, el cual posee un torque de 15.00Nm equivale a 1.5 kf, una eficiencia de 90% y con una variación de temperatura de trabajo de -10 a 50 grados Celsius, el cual es suficiente para moverla base del robot. Esto datos se obtuvo del datasheet del Anexo 2: Motor paso a paso 1.

Figura 54: Motor nema 17 con relación de transmisión 1/50

Elaborado por el equipo de trabajo

En segundo lugar, se evaluó el motor NEMA 23 que tiene una relación de transmisión 1/50, el cual posee un torque de 25.00Nm el cual equivale a 2.5 kf, una eficiencia de 90% y con una variación de temperatura de trabajo de -10 a 50 grados Celsius, el cual es suficiente para moverla el segundo grado de libertad del robot. Esto datos se obtuvo del datasheet del Anexo 2: Motor paso a paso 2.

Figura 55: Motor nema 23 con relación de transmisión 1/50

Elaborado por el equipo de trabajo

En tercer lugar, se evaluó otro motor NEMA 17 que tiene una relación de transmisión 1/20, con el cual posee un torque de 15.00Nm el cual equivale a 1.5 kf, una eficiencia de 90% y con una variación de temperatura de trabajo de -10 a 50 grados Celsius, el cual es suficiente para moverla el tercer grado de libertad del robot y su movimiento lento hace que el robot se más preciso. Esto datos se obtuvo del datasheet del Anexo 2: Motor paso a paso 3.

Figura 56: Motor nema 17 con relación de transmisión 1/20 y polea

Elaborado por el equipo de trabajo

En cuarto lugar, se evaluó otro motor NEMA 17 que tiene una relación de transmisión 1/10, con el cual posee un torque de 6.00Nm el cual equivale a 0.6kf, una eficiencia de 95% y con una variación de temperatura de trabajo de -20 a 50 grados Celsius, el cual es suficiente para moverla el cuarto grado de libertad del robot y por la velocidad de movimiento le permite ajustar la posición de la muñeca de manera rápida. Esto datos se obtuvo del datasheet del Anexo 2: Motor paso a paso 4.

Figura 57: Motor nema 17 con relación de transmisión 1/10

Elaborado por el equipo de trabajo

Para el quinto grado de libertad se evaluó el motor NEMA 14 que tiene una relación de transmisión $19\frac{38}{187}$, con el cual posee un torque de 5.00Nm el cual equivale a 0.5kf, una eficiencia de 81% y con una variación de temperatura de trabajo de -10 a 50 grados Celsius, el cual es suficiente para moverla el cuarto grado de libertad del robot y por la velocidad de movimiento le permite ajustar la posición de la del efector final de manera rápida. Esto datos se obtuvo del datasheet del Anexo 2: Motor paso a paso 5.

Figura 58: Motor NEMA 14 con relación de transmisión 1/19

Elaborado por el equipo de trabajo

3.9.5 Análisis de la Implementación del Brazo Robot

Para el análisis de la construcción del brazo robot se tomó en cuenta un modelo de brazo ya construido anteriormente. Primeramente, en la construcción del brazo se utilizará fierro corrosivo en general, el cual posee una resistividad alta y también por que el material es accesible y fácil de obtener ya que la compra se puede realizar en distintas ferreterías de esta localidad.

3.9.5.1 Base

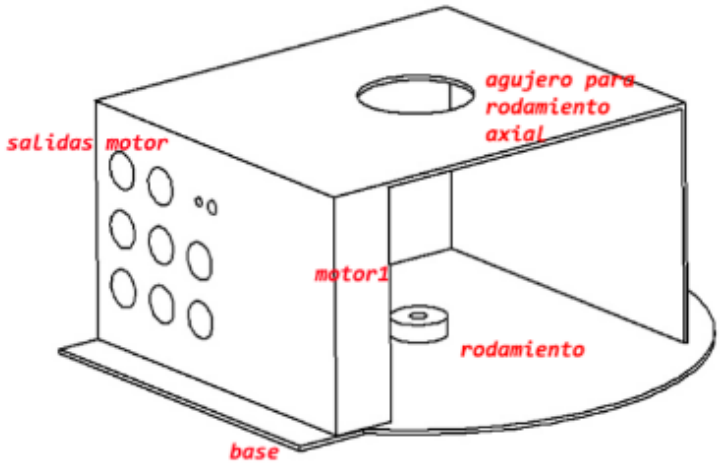
Tomando en cuenta el detalle antes mencionado la base está conformada por las siguientes partes:

- Una base plana de medidas 40x40 cm
- Un empaque para rodamiento acoplado a la base metálica de 40x40 cm para contener eje de rotación
- Empaque para acoplamiento de motor
- Tapa para contener al motor del primer grado de libertad en cual contiene agujero para rodamiento axial, agujeros de salidas para cables de motor

- Agujeros para tornillos con tuerca

El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, base del brazo robot.

Figura 59: Base para la implementación



Elaborado por el equipo de trabajo

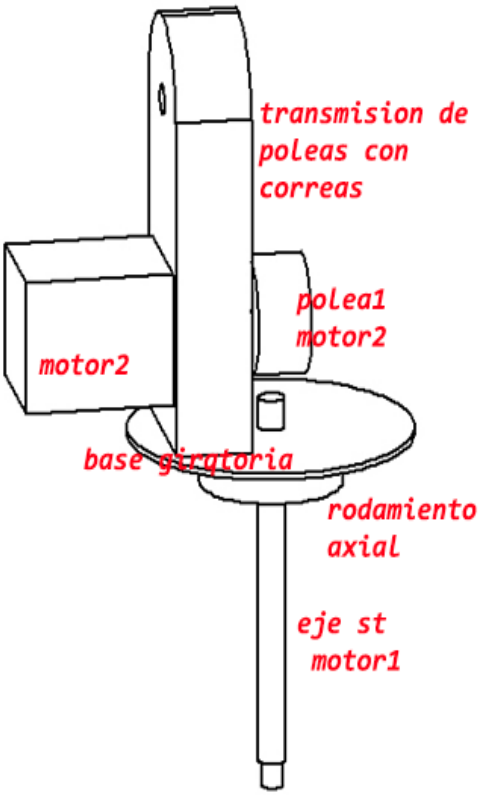
3.9.5.2 Eslabón 1

La construcción constara de las siguientes características:

- Base circular con acoplamiento para rodamiento axial y un agujero para acoplamiento de eje de transmisión
- Soporte con empaque para motor del segundo grado de libertad
- Empaques de rodamiento y con agujeros para acoplamiento de eje de transmisión
- Soporte para polea tensadora
- Agujeros para tornillos con tuerca

El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, eslabón 1 del brazo robot.

Figura 60: Eslabón 1 del brazo robot



Elaborado por el equipo de trabajo

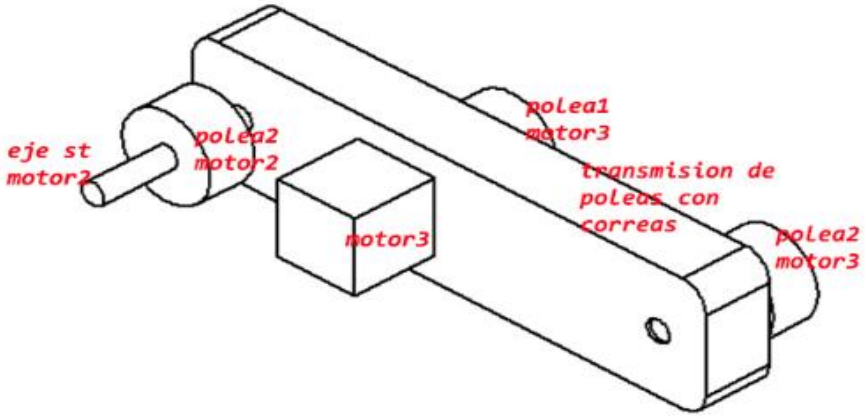
3.9.5.3 Eslabón 2

La construcción constara de un tubo rectangular con las siguientes características:

- Eje de trasmisión del eslabón uno al eslabón dos con agujeros y empaque para rodamientos
- Empaque para motor 3
- Soporte para polea tensadora
- Empaque de rodamientos y agujeros para eje de transmisión al eslabón 3
- Agujeros para tornillos con tuerca

El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, eslabón 2 del brazo robot.

Figura 61: Eslabón 2 del brazo robot



Elaborado por el equipo de trabajo

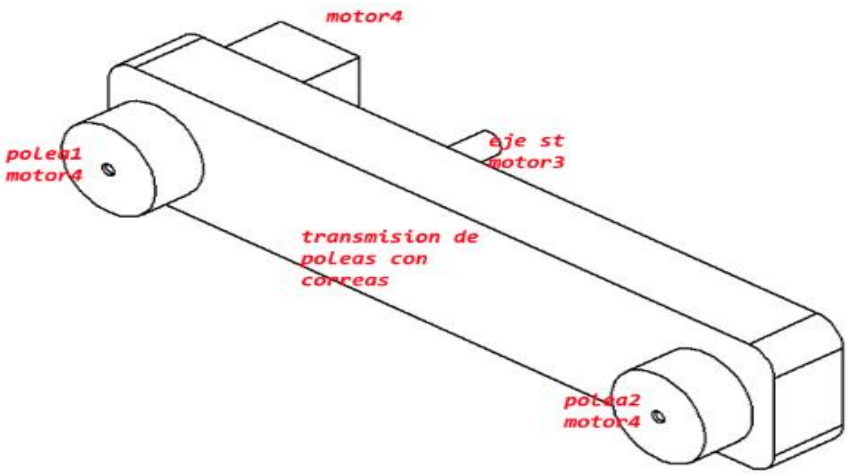
3.9.5.4 Eslabón 3

La construcción constara de un tubo rectangular con las siguientes características:

- Empaque para motor 4
- Eje de trasmisión del eslabón 2 al eslabón 3
- Empaque para rodamientos con agujeros para eje que sostiene la polea del motor 4 y eje de trasmisión al eslabón 4

El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, eslabón 3 del brazo robot.

Figura 62: Eslabón 3 del brazo robot



Elaborado por el equipo de trabajo

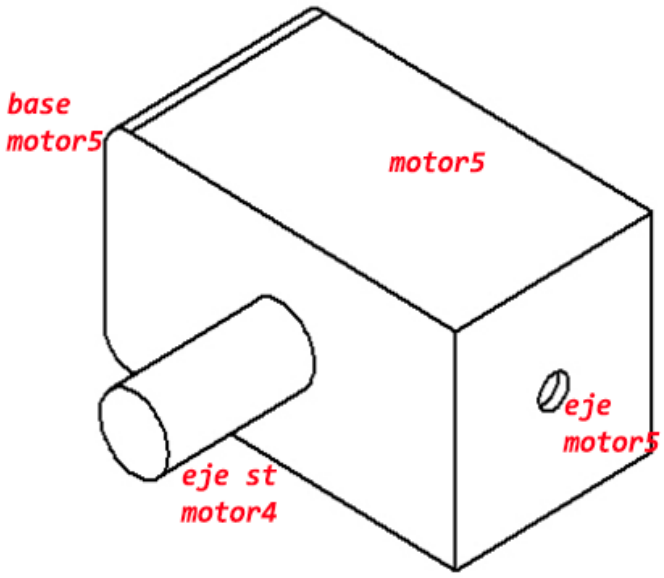
3.9.5.5 Eslabón 4

La construcción constara con las siguientes características:

- Consta de un empaque cuadrado con agujero para salida para eje de motor 5 y empalme de Gripper
- Eje de trasmisión del eslabón 3 al eslabón 4

El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, eslabón 4 del brazo robot.

Figura 63: Eslabón 4 del brazo robot



Elaborado por el equipo de trabajo

3.9.6 Análisis de la Implementación del Gripper

La implementación del Gripper o efector final se realizó de acuerdo al requerimiento establecido, se realizó las pruebas de dos modelos implementados para realizar las pruebas correspondientes.

Se necesita implementar un puente h para controlar el motor que realizara el movimiento de nuestro Gripper, para ello usaremos un circuito integrado L298N, diodos,

capacitores, resistencias, convertidores de voltaje y una fuente de 24V. Este diseño se realizó en Eagle para luego pasarlo a una placa de baquelita.

En el primer diseño de construcción del Gripper se tomó en cuenta el material de impresión, el plástico PLA es un material de resistencia moderada, que puede soportar presiones de hasta 15Nm. Una estructura conformada por una base y dos pinzas que tienen siguientes partes:

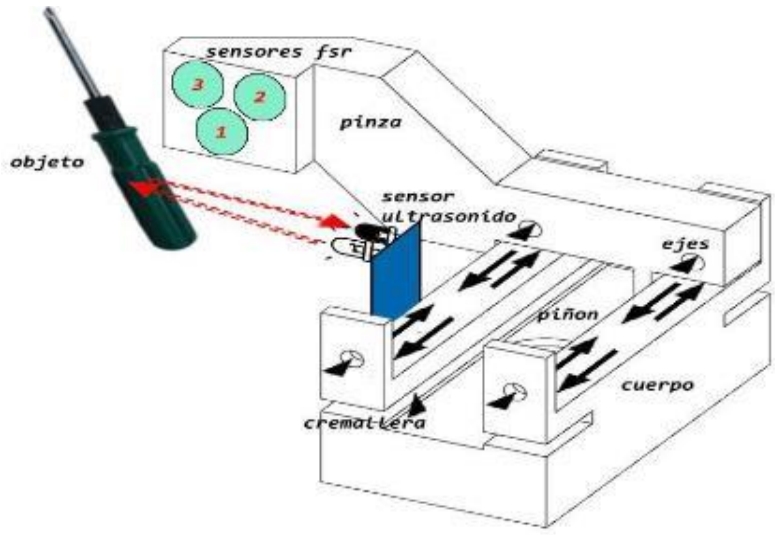
Base:

- Cavidades para rieles de engranajes y ejes
- Cavidad para cabezal de engranajes del motor
- Punto de apoyo para acoplamiento del motor
- Cavidad para sensor de proximidad
- Cavidades para tornillos
- Punto de apoyo para acoplamiento a la muñeca

Pinzas:

- Cavidad para desplazamiento de ejes
- Punto de apoyo para sensor de peso

Figura 64: Modelo 1 del gripper



Elaborado por el equipo de trabajo

Para el segundo diseño de Gripper se realizó con plástico de impresión PETG que tiene una resistencia de hasta 25Nm, las partes que lo componen son: una base que es el cuerpo fijo, una pieza móvil que llamaremos dorsal, proporciona movimiento a los dedos y uñas los cuales son de tres pares, cada una de las piezas tiene funciones o partes las cuales son:

Base:

- Cavidad para acoplamiento del motor
- Punto de apoyo para switch final de carrear
- Cavidades para ejes
- Puto de apoyo para empalme a la muñeca del robot
- Cavidad para eje de motor con acoplamiento de tornillo sin fin

El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, cuerpo del gripper.

Pieza para el control de movimiento:

- Cavidades para ejes
- Cavidades para tuerca de tornillo sin fin, en punto central y switch final de carrera
- Punto de apoyo para el sensor de proximidad

El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, dorsal del gripper.

Dedos:

- Cavidades para ejes
- Cavidades para uniones

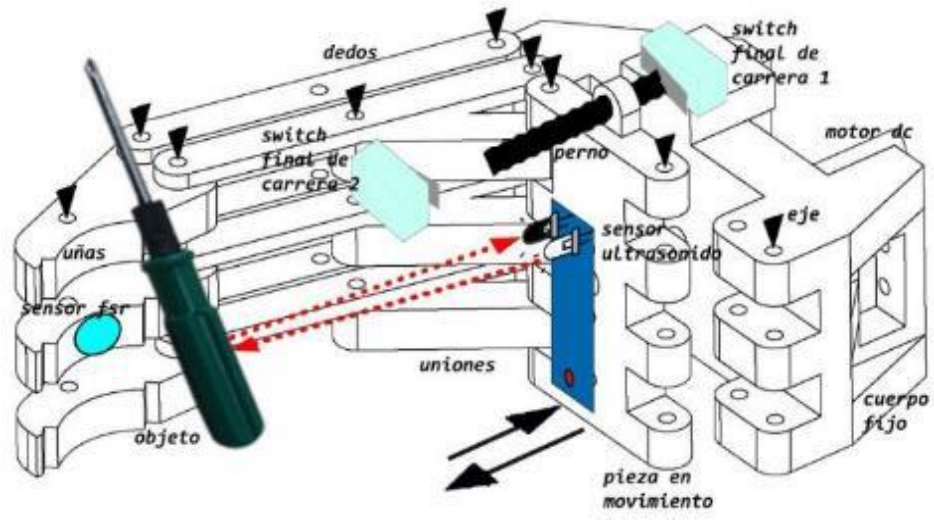
El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, articulaciones del gripper.

Uñas:

- Punto de apoyo para sensor de peso
- Ranura para zapatillas o escobillas
- Cavidad para ejes
- Ranuras dentadas para asegurar el agarre

El diseño fue desarrollado en SolidWorks y se puede observar en el Anexo 3: Solidworks, pinza del gripper.

Figura 65: Modelo 2 de gripper



Elaborado por el equipo de trabajo

El diseño final del Gripper fue ensamblado en SolidWorks y se puede observar en el Anexo 3: Solidworks-gripper.

3.9.7 Análisis de los Componentes de Control

En este análisis usaremos módulos para controlar los motores PAP, se tomó en cuenta la variabilidad en los micro pasos para así poder controlar con más precisión el movimiento de las articulaciones.

3.9.7.1 Módulo de Control Dm320t

Es un módulo de control con señal de salida PWM para motores PAP, el presente modulo es industrial lo cual permite su trabajo para tiempos extendidos. En la figura 66 se observa una tabla de micro pasos, también está la corriente de salida esto fue diseñado para motores con menos capacidad. Los datos del driver se pueden observar en el Anexo 2: Datos importantes del driver dm320t.

Figura 66: Módulo dm320t

Elaborado por el equipo de trabajo

3.9.7.2 Módulo de Control Dm542t

El dispositivo de control DM542T al igual que el anterior es un módulo industrial. En la figura 67 se observa una tabla con los micro pasos configurados y corriente, esta última en este módulo fue diseñado para motores con más capacidad. Los datos del driver se pueden observar en el Anexo 2: Datos importantes del driver dm542t.

Figura 67: Módulo dm542t

Elaborado por el equipo de trabajo

CAPITULO IV

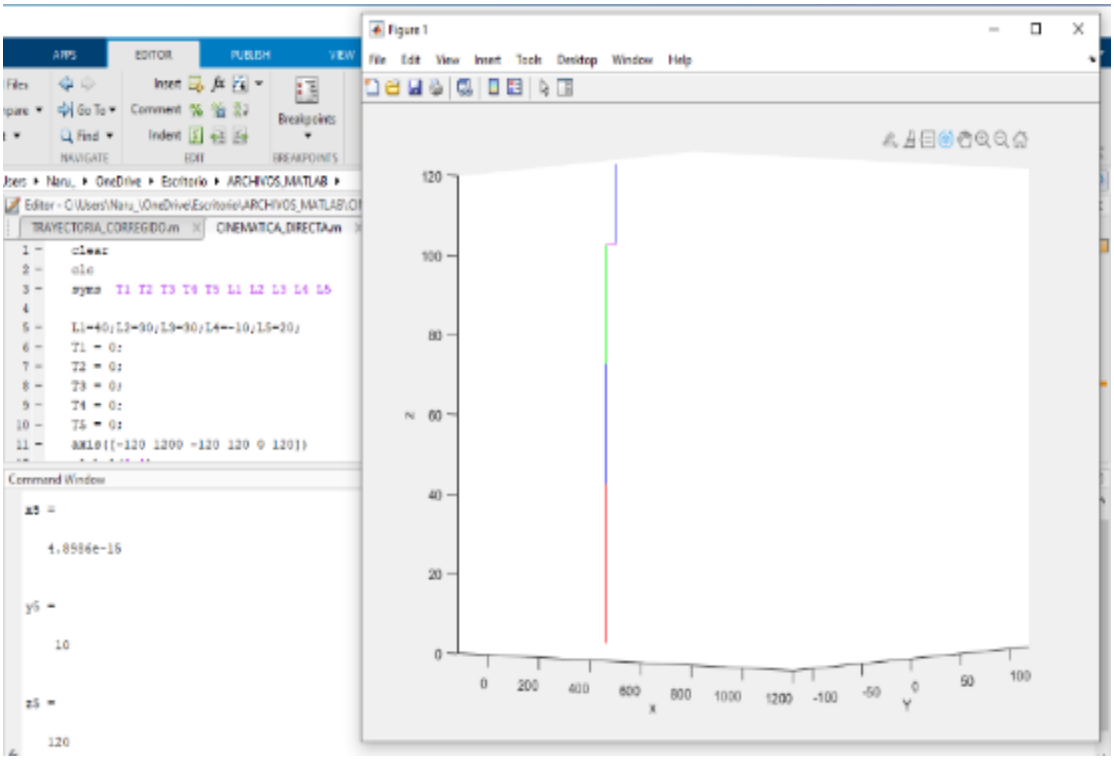
RESULTADOS Y DISCUSIÓN

4.1 RESULTADOS

4.1.1 Sistema del Control Cinemático

Como podemos observar en la figura 68 el brazo robot está totalmente en forma vertical, debido a la compensación de 90° que se le asigno al segundo grado de libertad para así considerar esa posición como punto de inicio o el punto cero en donde $T_1 = 0°, T_2 = 0°, T_3 = 0°, T_4 = 0°\ y\ T_5 = 0°$, también se considera la longitud de los eslabones $L_1 = 40cm, L_2 = 30cm, L_3 = 30cm, L_4 = 10cm\ y\ L_5 = 20cm$, las medidas de las articulaciones son fijas y no cambian con esos valores articulares el valor de los puntos cartesiano es $X = 0, Y = 0\ y\ Z = 120$. La programación se puede ver en el Anexo 1: Matlab, ecuacion de matriz final.

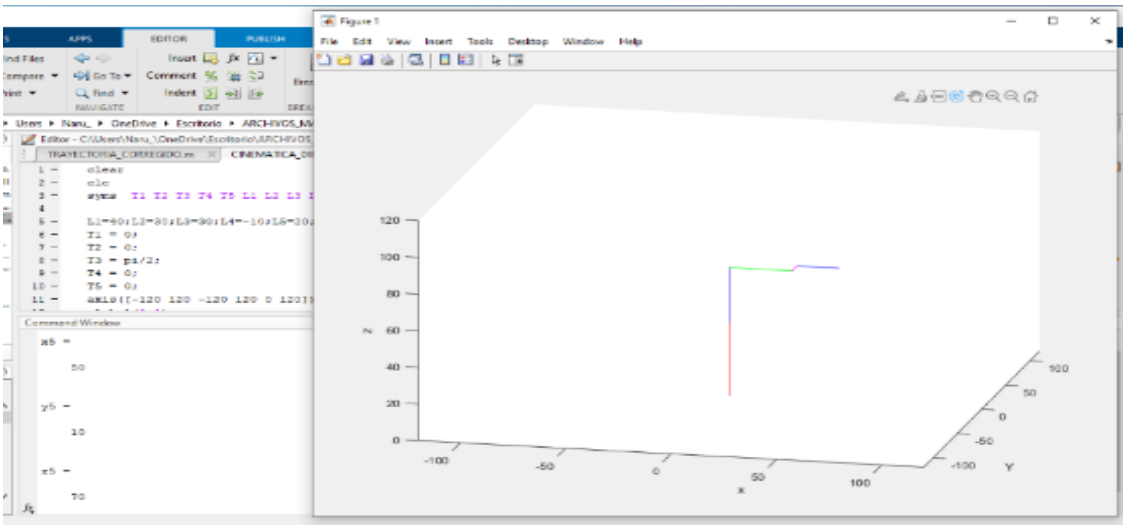
Figura 68: Simulación de la posición de inicio del robot



Elaborado por el equipo de trabajo

En la figura 69 se ve el movimiento que realiza el brazo con los valores $T_1 = 0°, T_2 = 0°, T_3 = 90°, T_4 = 0°\ y\ T_5 = 0°$ con estos valores articulares tenemos que $X = 50, Y = 10\ y\ Z = 70$. Lo cual cumple con lo observado y el cálculo matemático.

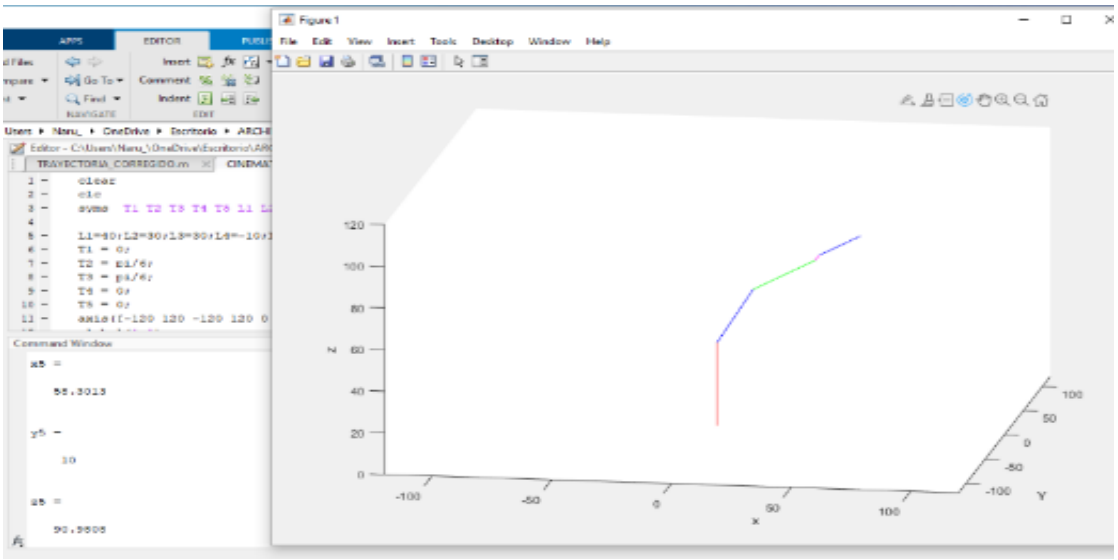
Figura 69: Simulación de movimiento del robot variando una articulación



Elaborado por el equipo de trabajo

En la figura 70 se ve el movimiento que realiza el brazo con los valores $T_1 = 0°, T_2 = 30°, T_3 = 30°, T_4 = 0°\ y\ T_5 = 0°$, con estos valores articulares tenemos que $X = 58, Y = 10\ y\ Z = 90$. Lo cual cumple con lo observado y el cálculo matemático.

Figura 70: Simulación de movimiento del robot variando dos articulaciones

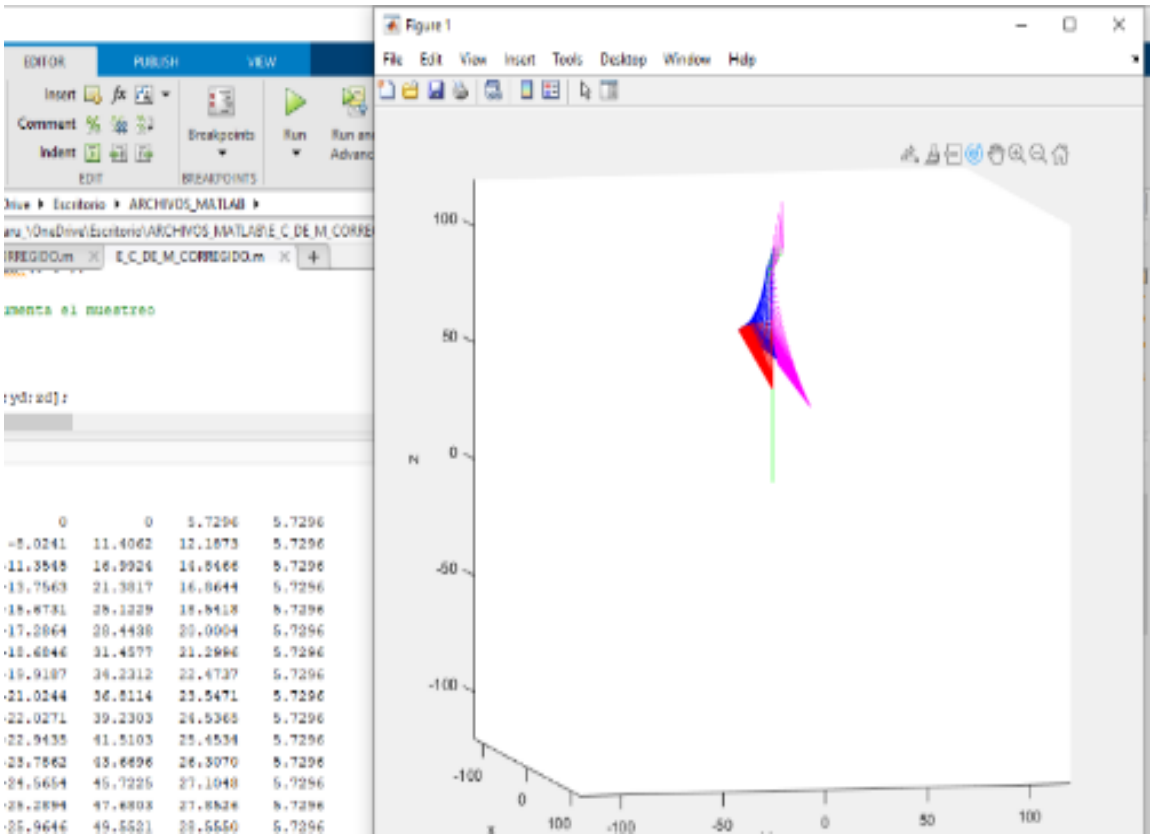


Elaborado por el equipo de trabajo

4.1.1.1 Simulación de Cinemática Inversa y Jacobiano

En la simulación de la cinemática inversa y el jacobiano se obtuvo la trayectoria que debe recorrer el brazo robot para llegar a un punto deseado y poder asignar valores articulares a cada motor, en este caso el punto deseado fue $X = 20, Y = 20\ y\ Z = 20$, el resultado demostró que llega al punto deseado. La programación de puede ver en el Anexo 1: Matlab, trayectoria de brazo robot.

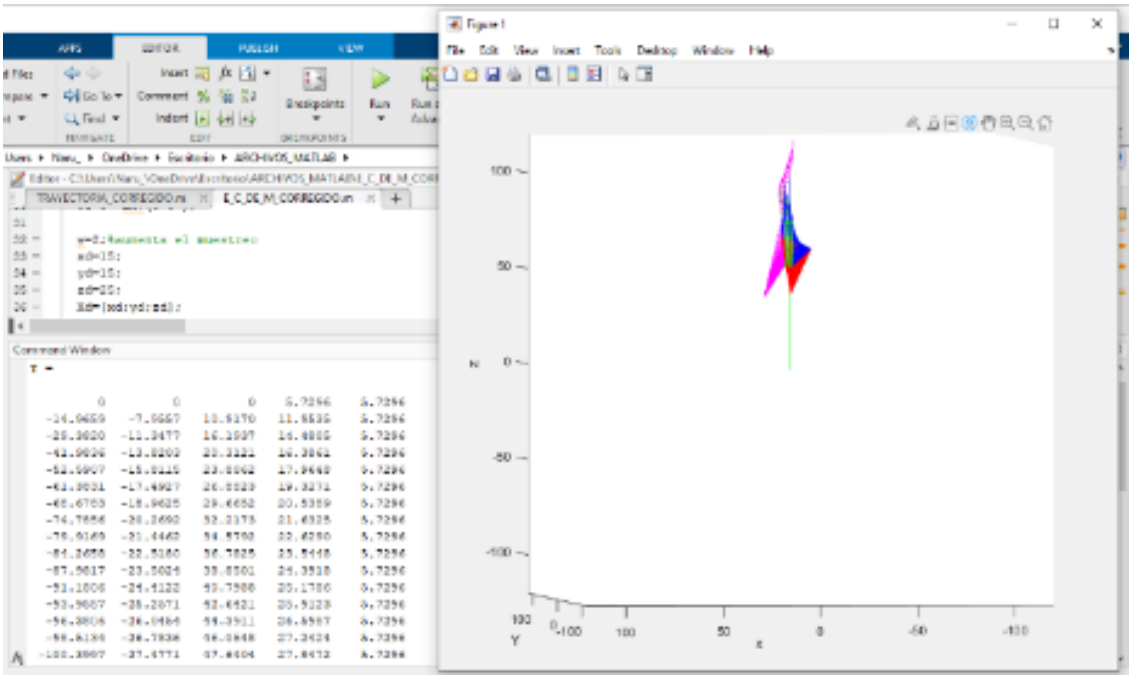
Figura 71: Trayectoria realizada con jacobiano y cinemática inversa



Elaborado por el equipo de trabajo

En la figura 72 se asignó los puntos deseados $X = 15, Y = 15\ y\ Z = 25$ el resultado fue que si llego al punto deseado.

Figura 72: Simulación de la prueba de trayectoria

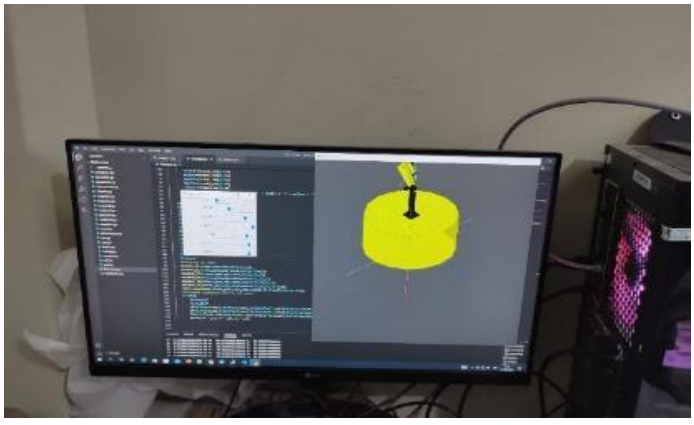


Elaborado por el equipo de trabajo

4.1.1.2 Programación en Python

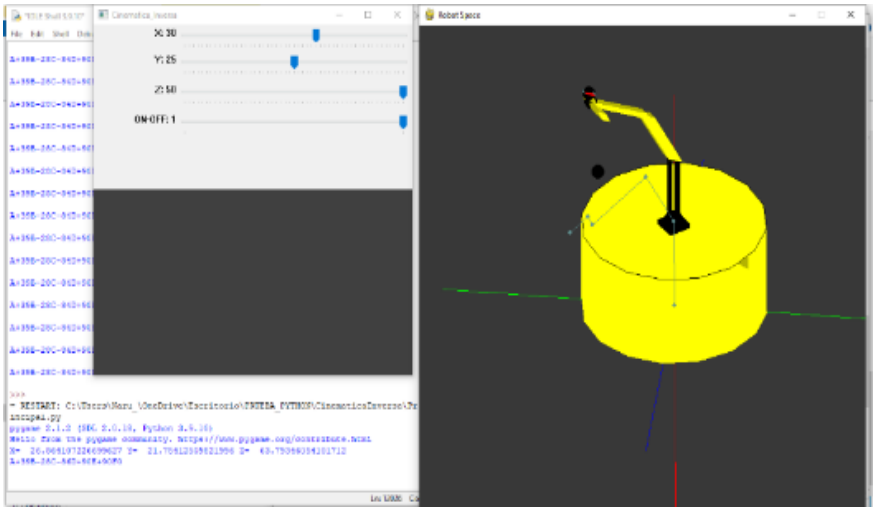
En los resultados de la programación en Python se programó un interfaz virtual para monitorear el movimiento que realiza el robot. El desarrollo se basó en diseñar una librería que se puede ver en el Anexo 1: Python, librería de la cinemática y otro programa principal que obtenga el procedimiento de la librería, se puede ver en el Anexo 1: Python, principal.

Figura 73: Interfaz virtual con cinemática directa



Elaborado por el equipo de trabajo

Figura 74: Interfaz virtual con cinemática inversa



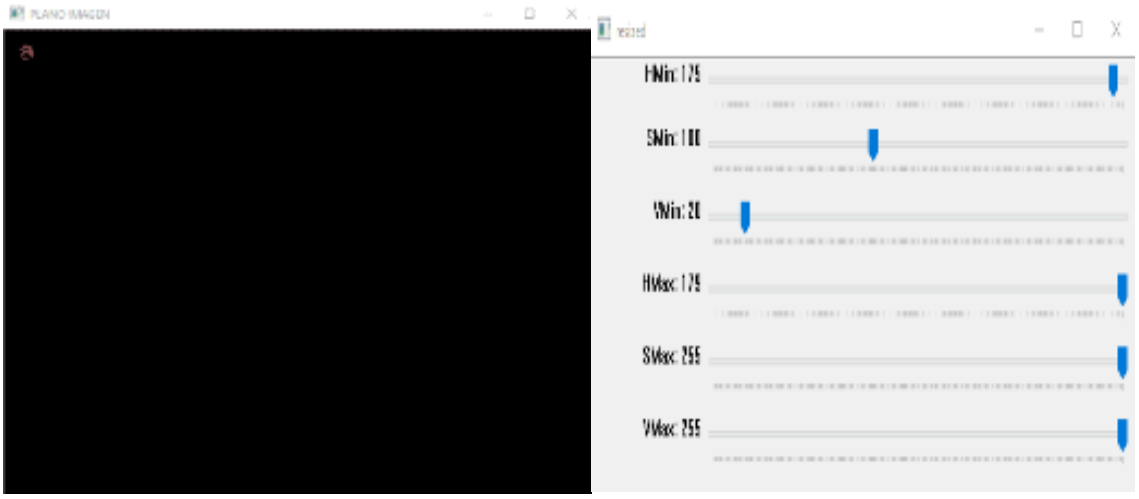
Elaborado por el equipo de trabajo

4.1.2 Sistema de Visión Artificial

4.1.2.1 Adquisición de Imágenes

En este sistema desarrollamos la adquisición de imágenes de nuestras formas circulares de colores, luego se aplicó el procesamiento de imágenes y encontramos los rangos de valores hsv para el color rojo, verde, amarillo y azul en (Hmin, Smin, Vmin – Hmax, Smax, Vmax). El color rojo tiene dos valores en el rango del espacio de color HSV, por ello se usó los dos rangos de valores para luego unirlos y obtener el color rojo.

Figura 75: Color rojo con 2 valores en el espacio hsv



Elaborado por el equipo de trabajo

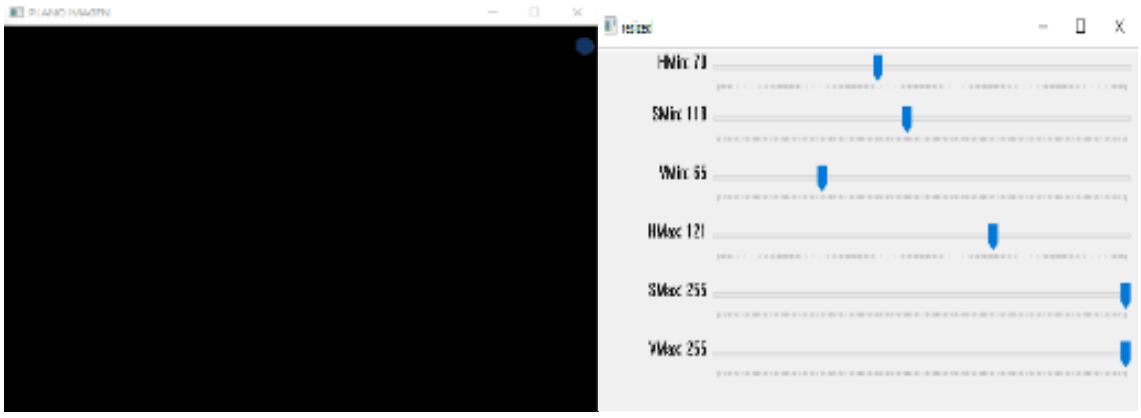
El valor 1 tiene el resultado (0,100,23 – 0,255,255) y el valor 2 (175,100,20 – 179,255,255). El valor del color verde es solo uno, está en el rango (54,68,32 – 83-255-255) como en la figura 76. La programación se puede ver en el Anexo 1: Python, detección de color HSV.

Figura 76: Color verde en el espacio hsv

Elaborado por el equipo de trabajo

El valor del color azul es solo uno, está en el rango (70, 118, 65 – 121, 255, 255) como en la figura 77.

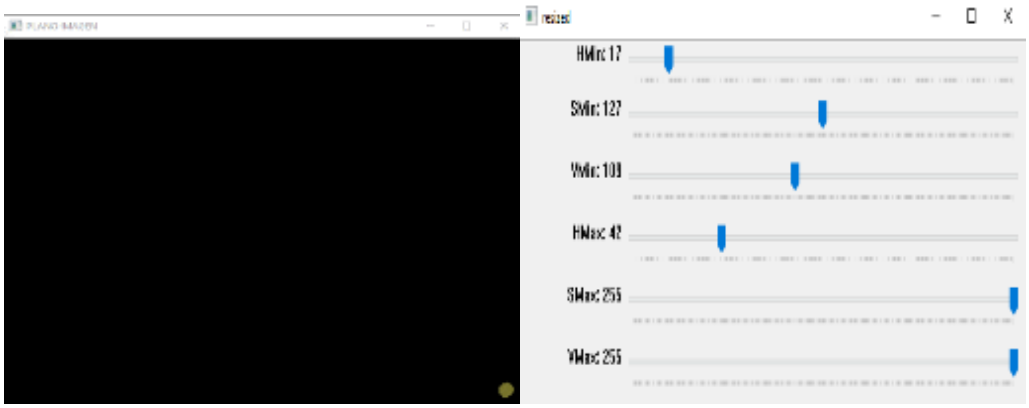
Figura 77: Color azul en el espacio hsv



Elaborado por el equipo de trabajo

El valor del color amarillo es solo uno, está en el rango (17, 127, 109 – 42, 255, 255) como en la figura 78.

Figura 78: Color amarillo en el espacio hsv



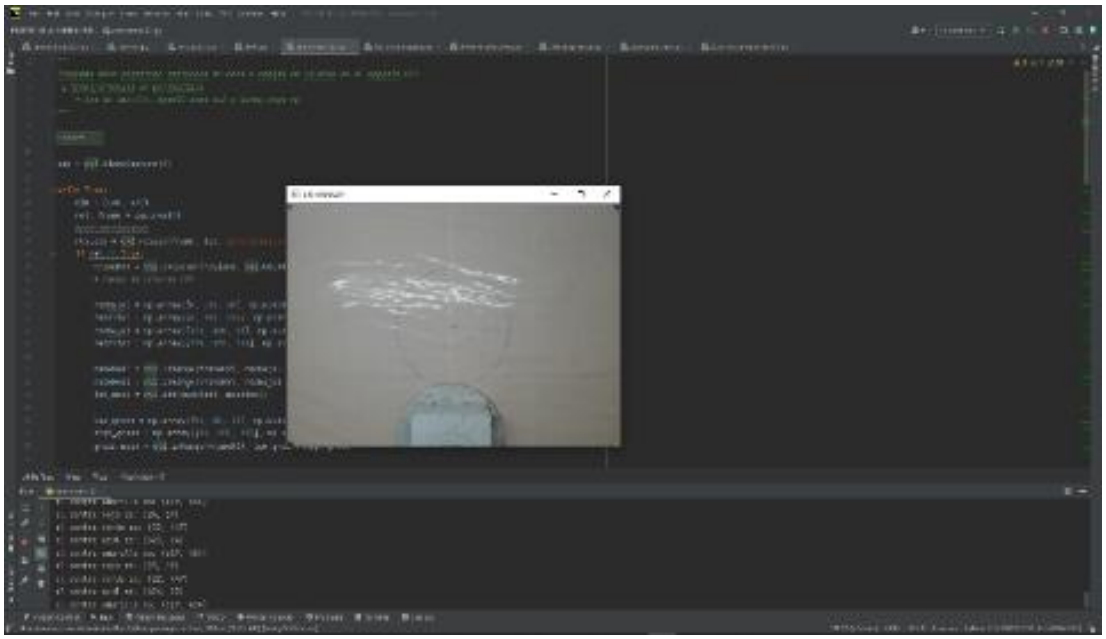
Elaborado por el equipo de trabajo

4.1.2.2 Procesamiento de Imágenes

Ahora se creo una mascara con la umbralización y aplicamos los momentos. Seguidamente se encontro los contornos y se grafico, luego se busco el radio de esa circunferencia. Despues realizamos el desarrollo del programa de obtencion de los puntos coordenados en los centros del objeto, con respecto al sistema de referencia de la cámara.

La programacion se puede ver en el Anexo 1: Python, contornos.

Figura 79: Desarrollo de la transformación de perspectiva



Elaborado por el equipo de trabajo

En la figura 80 se ve los puntos centros de cada color con los métodos antes mencionados.

Figura 80: Punto centro en cada forma de color

Elaborado por el equipo de trabajo

En la figura 81 se ve el resultado de la transformacion de perspectiva.

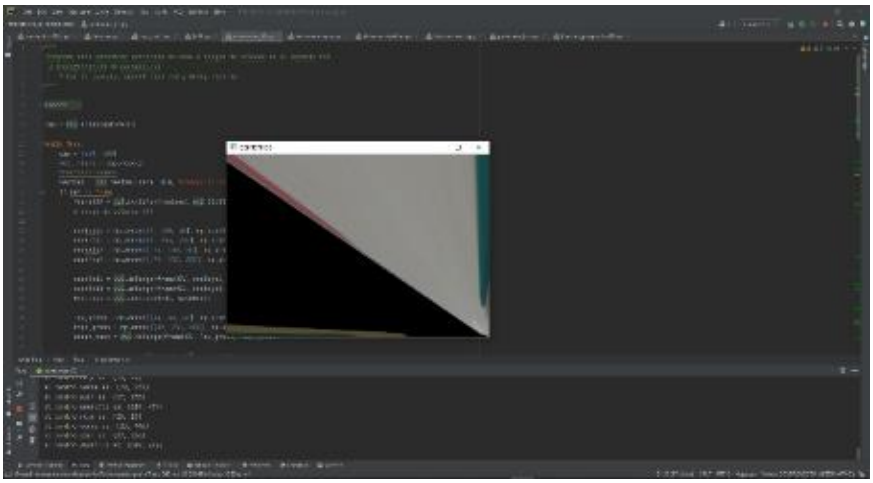
Figura 81: Puntos transformados

Elaborado por el equipo de trabajo

La transformacion de perspectiva, cuando se inserta un objeto del mismo color que el patron de color distorsiona el resultado, en la figura 82 se ve que esto nos

perjudicara mas adelante cuando realicemos el detector de objetos o poniendo cualquier otro material. La programacion se puede ver en el Anexo 1: Python, obtencion de coordenadas.

Figura 82: Perspectiva distorsionada

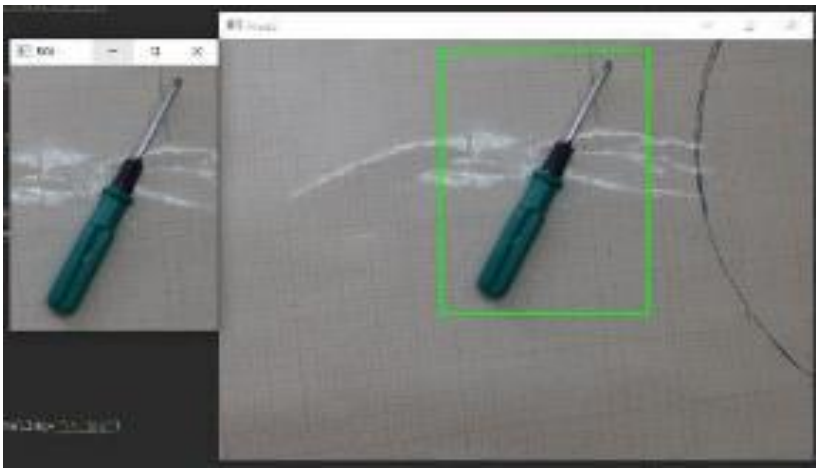


Elaborado por el equipo de trabajo

4.1.2.3 Detección de Objetos

En este paso se realizo el metodo Haar Cascade, se tomo imágenes del destornillador y el pulsador con resolución de 640x480, y luego se realizo el recorte de la porción que necesitamos. En la figura 83 se ve el proceso de recorte y en el Anexo 1: Python, recorte de imagen, se ve la programación.

Figura 83: Proceso de recorte de la porción de imagen



Elaborado por el equipo de trabajo

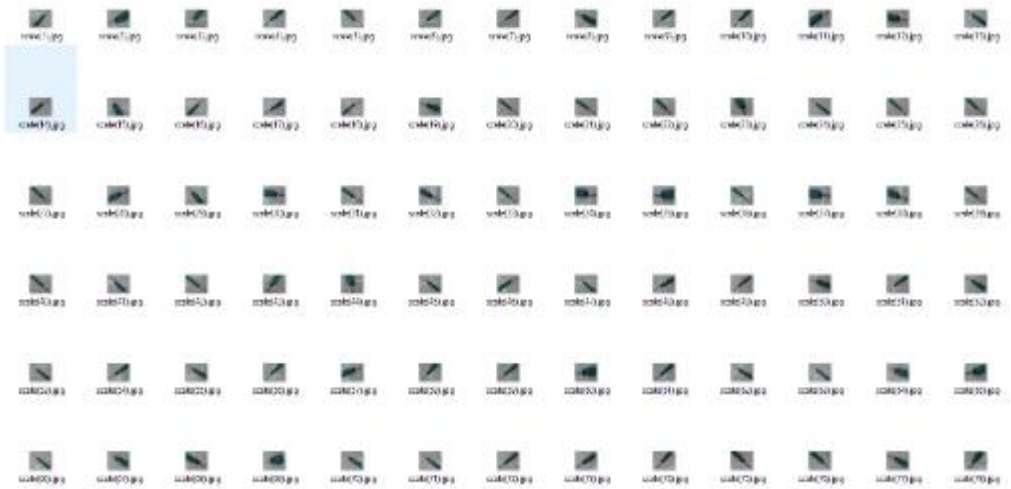
Seguidamente se realiza el escalado de los recortes de la imágenes. En la figura 84 se ve que tenemos un total de 200 imágenes escaladas (positivas) del pulsador para nuestro dataset del entrenamiento con un tamaño de 24x24 pixeles. Se crea y guarda en una carpeta “p” que esta dentro de una carpeta pulsador como se menciono anteriormente.

Figura 84: Imágenes del pulsador en la carpeta p

Elaborado por el equipo de trabajo

Tambien un total de 200 imágenes positivas del destorrnillador para nuestro dataset del entrenamiento con un tamaño de 32x24 pixeles, como en la figura 85.

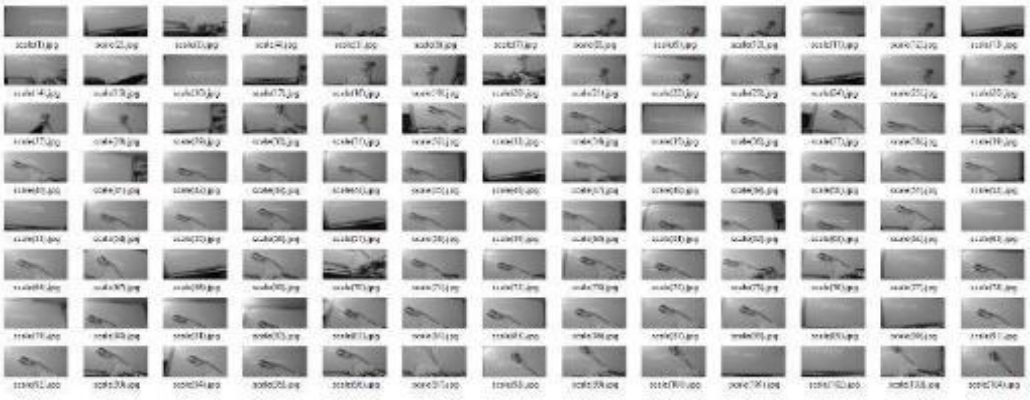
Figura 85: Imágenes del destornillador en la carpeta p



Elaborado por el equipo de trabajo

Despues se obtuvo imágenes negativas que son el entorno sin los objetos antes mencionados, se convertio a escala de grises y escaladas a 600x300 pixeles. En total son 300 imágenes negativas, se crear y se guarda en la carpeta “n”.

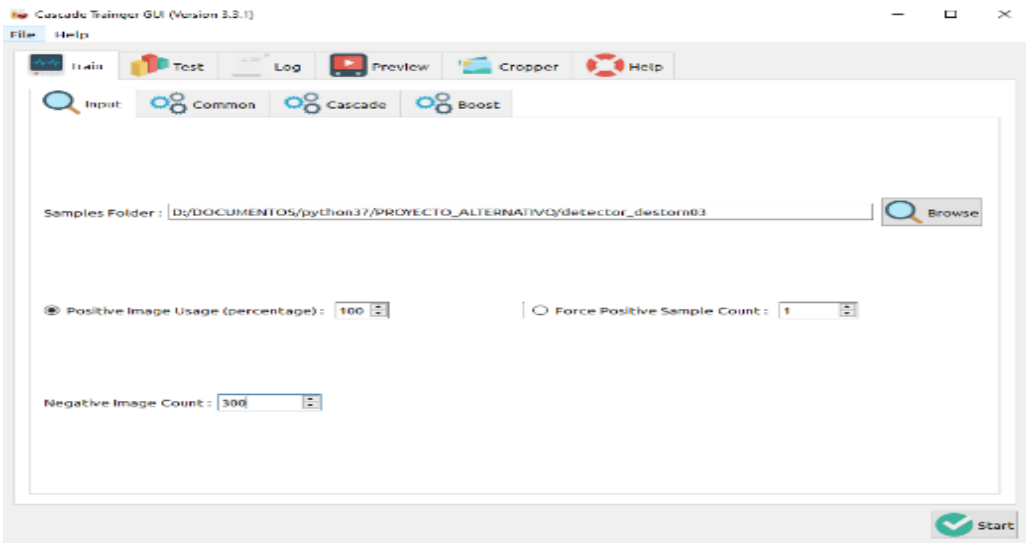
Figura 86: Imágenes negativas en la carpeta n



Elaborado por el equipo de trabajo

Una vez preparado nuestras imágenes se inicio el programa Cascade Trainer GUI, en la figura 87 se ve que en la pestaña input se ubico el origen de archivo se busca con “browse” donde esta nuestra carpeta origen, en ella estan las carpetas “p” y “n”, el porcentaje de imágenes positivas son 100%, y en cantidad de imágenes negativas se puso 300.

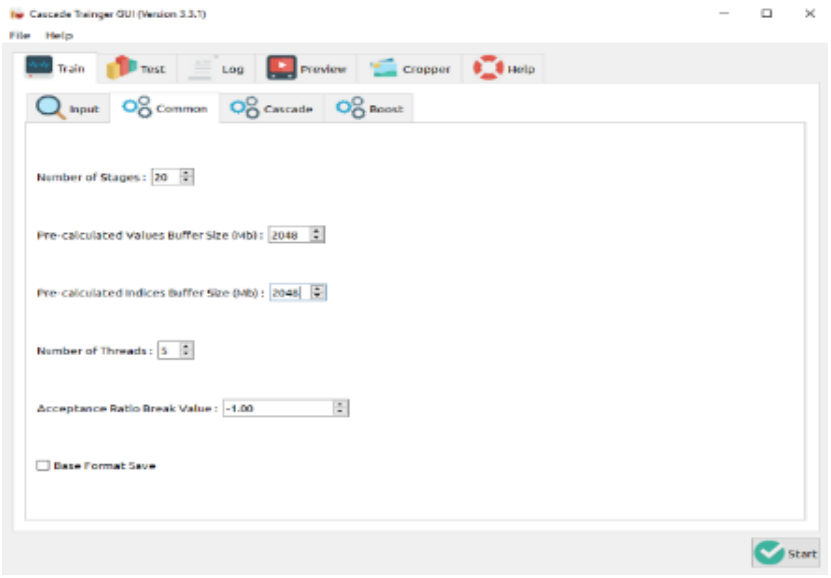
Figura 87: Cascade trainer gui - pestaña input



Elaborado por el equipo de trabajo

En la figura 88 se ve que en la pestaña “common” se configuro la etapa de entrenamiento en 20 y se especifico la memoria ram a 2048 mb, este ultimo puede cambiar dependiendo la capacidad de tu pc.

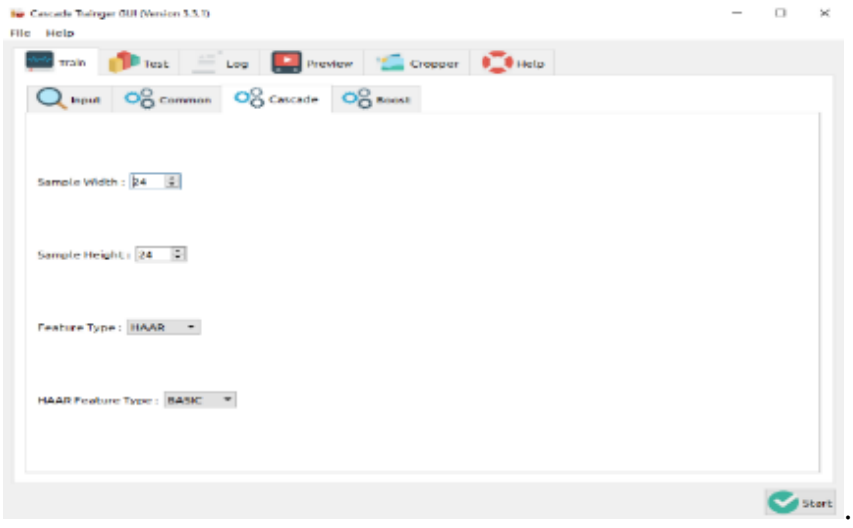
Figura 88: Cascade trainer gui - pestaña common



Elaborado por el equipo de trabajo

Luego en la pestaña “cascade”se configuro el tamaño de porcion de imagen con ancho de 24 pixeles y alto de 24 pixeles para el caso del pulsador, si es destornillador tendria un valor de 32x24, el clasificador Haar de tipo basic, esto se ve en la figura 89.

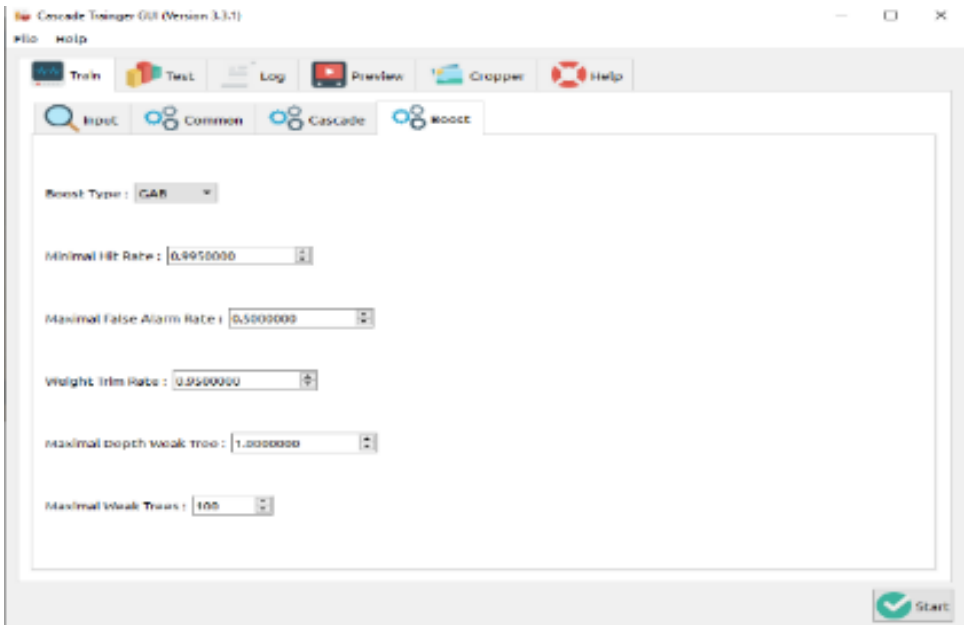
Figura 89: Cascade trainer gui - pestaña cascade



Elaborado por el equipo de trabajo

Seguidamente esta la pestaña “boost”, que se dejo la configuración por defecto, como en la figura 90.

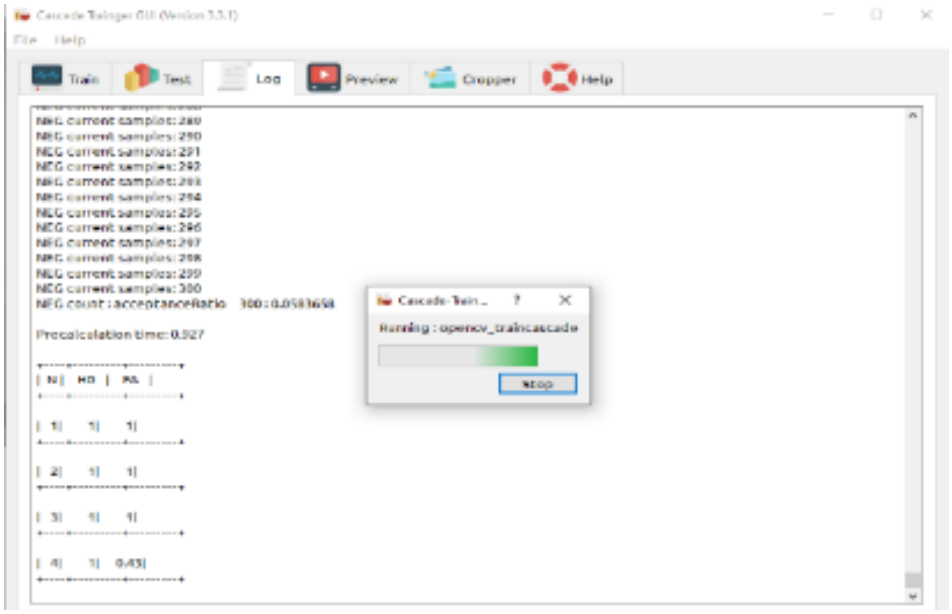
Figura 90: Cascade trainer gui - pestaña boost



Elaborado por el equipo de trabajo

Despues se comenzo el entrenamiento del clasificador de imágenes cascada, se pulso el boton “start” para comenzar.

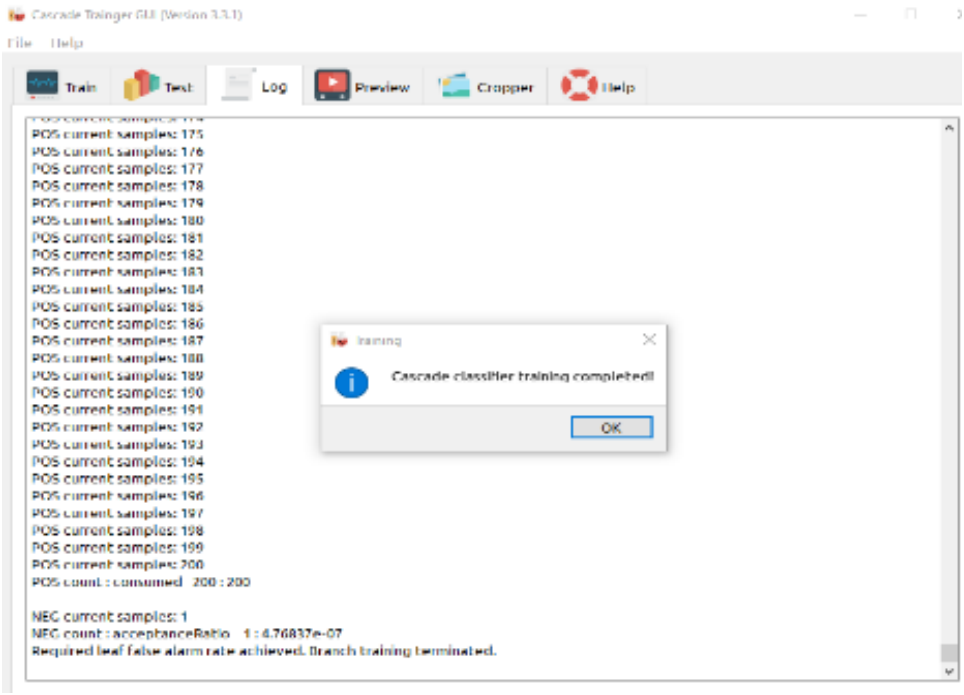
Figura 91: Cascade trainer gui - inicio de entrenamiento



Elaborado por el equipo de trabajo

Una vez termiando el entrenamiento, se observa una ventana de confirmación con el numero de etapas que se especifico anteriormente.

Figura 92: Cascade trainer gui - entrenamiento terminado



Elaborado por el equipo de trabajo

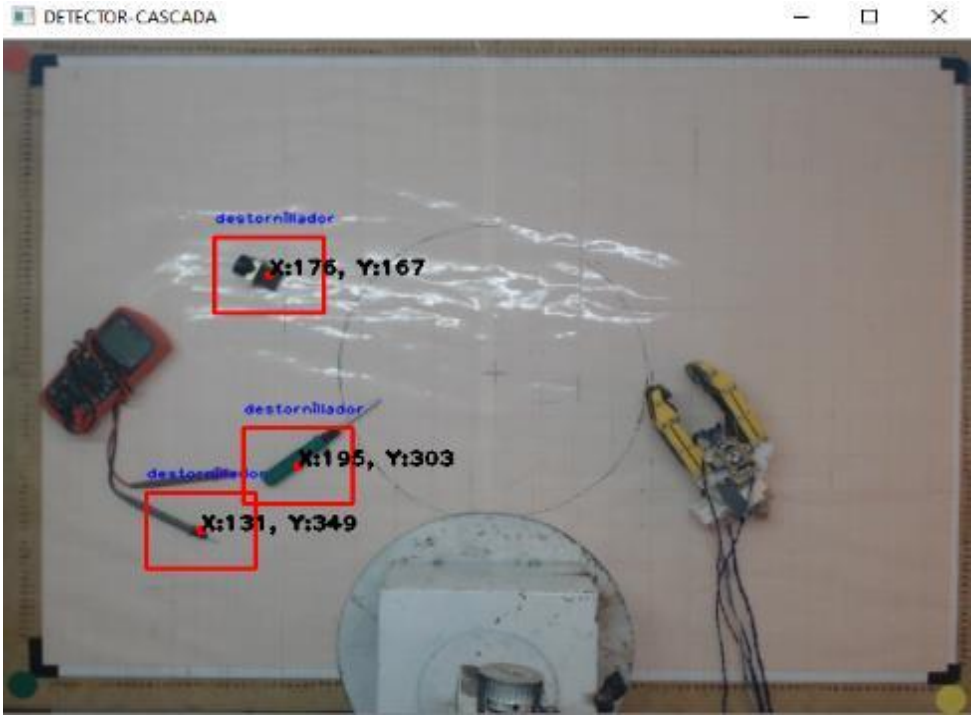
Finalmente, de este entrenamiento se obtuvo el archivo “cascade.xml” , para ello en la figura 93 se ve unos archivos despues del entrenamiento en la carpeta “classifier” para el pulsador y destornillador, del cual tomaremos el antes mencionado.

Figura 93: Archivo cascade.xml

Elaborado por el equipo de trabajo

Ahora se realiza un programa para probar los archivos entrenados, se mostro el error de detección al buscar el detornillador, se ve muchos vecinos con porciones de imagen parecidos a la muestra.

Figura 94: Error de detección del destornillador



Elaborado por el equipo de trabajo

La muestra del pulsador se mostro una detección buena con coordenadas respecto al plano imagen. Se visualiza en la figura 95 un marco verde alrededor del objeto indicando el punto centro de nuestra coordenada con parámetros como factor de escala, numero de vecinos minimos, tamaño minomo de encuadre y tamaño maximo de encuadre. Estos valores tienden a cambiar si se quiere distinguir bien el objeto, en este caso tomamos los valores siguientes: $scaleFactor = 1.1, minNeighbors = 5, minSize = (30,30) y\ maxSize = (50,50)$.

Figura 95. Pulsador encontrado

Elaborado por el equipo de trabajo

La muestra del destornillador se mostro una detección media, y coordenadas con respecto al plano imagen. Se visualiza en la figura 96, un marco rojo con nombre alrededor del objeto indicando el punto centro de nuestra coordenada, se uso parametros como: $scaleFactor = 1.1, minNeighbors = 10, minSize = (70,70) y\ maxSize = (91,97)$.

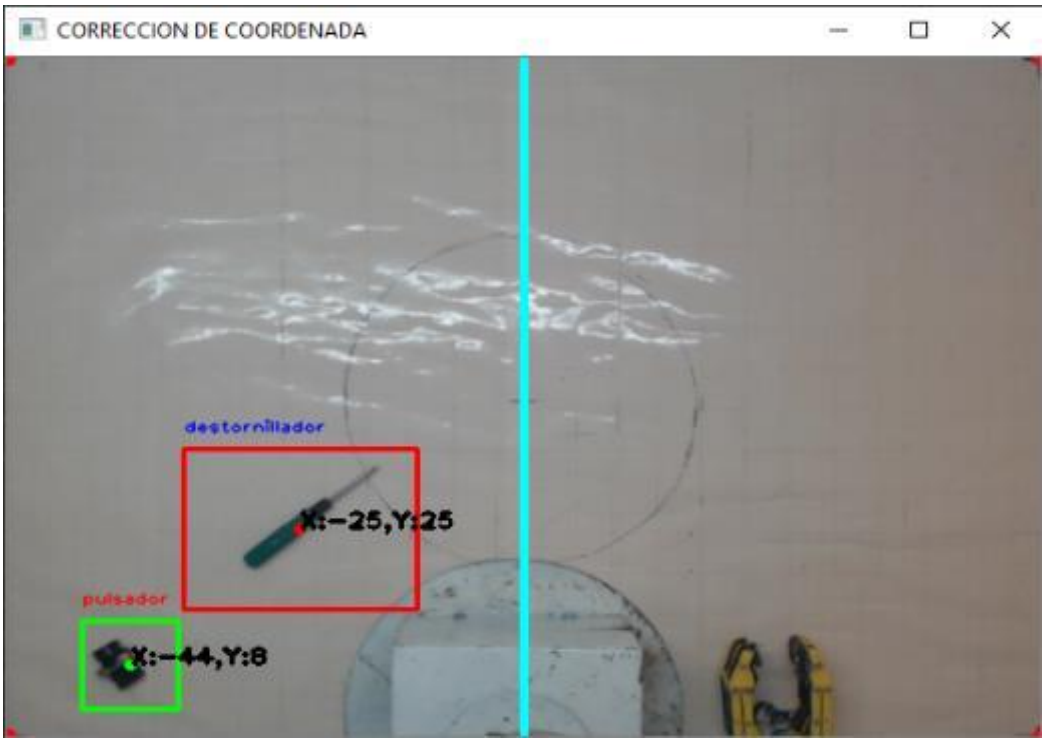
Figura 96: Destornillador encontrado

Elaborado por el equipo de trabajo

4.1.2.4 Estimación de Coordenada

La corrección de coordenada con respecto al plano coordenado, se realizó con transformación de perspectiva y añadiendo los objetos entrenados. El plano coordenado está dividido en eje $X = (-58\ a\ 0)cm\ y\ (0\ a\ 58)cm, Y = (0\ a\ 83)cm,$ en la figura 97 se visualiza estas características, se dibujó una línea celeste para separar los dos cuadrantes.

Figura 97: Corrección de coordenada

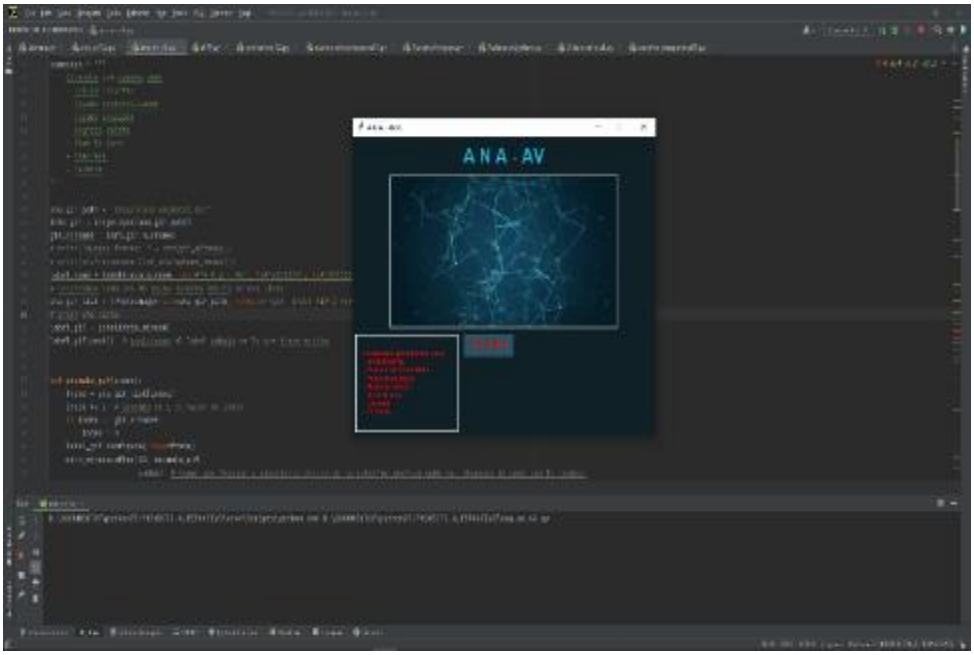


Elaborado por el equipo de trabajo

4.1.3 Sistema de Reconocimiento de Voz

Se realizo una interfaz gráfica con el módulo tkinter, se usó widgets para una mejor visualización. En la figura 98 se ve el desarrollo de la interfaz con el algoritmo del para reconocer la voz, el IDE Pycharm nos familiarizo más con el algoritmo de Python.

Figura 98: Interfaz gráfica del asistente virtual en pycharm



Elaborado por el equipo de trabajo

El asistente virtual con un título llamado “ANA – AV”, una imagen de entrada, una caja de texto para mostrar los patrones que se puede usar y un botón START para escuchar el patrón que se dará con la voz.

Se obtuvo unos bugs al momento de esperar mucho tiempo el patron, en la figura 99 se ve la corrección de estos bugs pertenecientes a el módulo Speech recognition como: “UnknowValueError” y “RequestError”. La programación se puede ver en el Anexo 1- Python, asistente virtual.

Figura 99: Código de corrección de bugs

```
except sr.UnknownValueError:
    print("No te entendí, intenta de nuevo")
except sr.RequestError as e:
    print("Could not request results from Google Speech Recognition service; {0}".format(e))
```

Elaborado por el equipo de trabajo

4.1.4 Implementación del Brazo Robot

4.1.4.1 Elaboración de Piezas para Ensamble

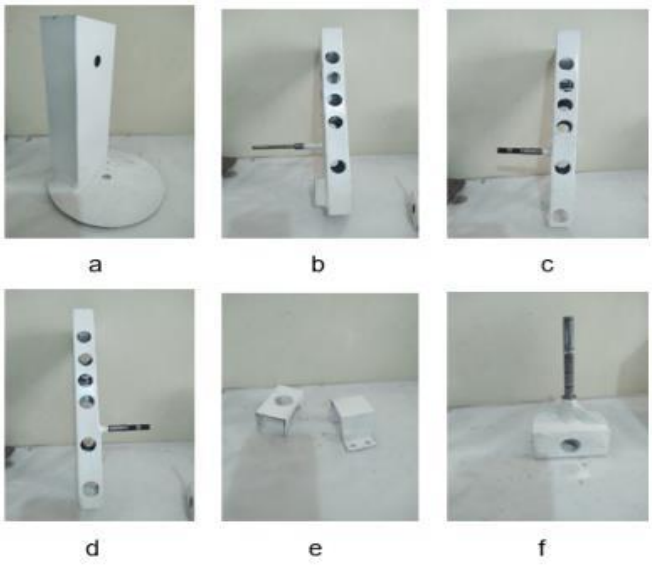
Se desarrollo las piezas para la implementación usando medidas, corte u unión con soldadura.

Figura 100: Unión de piezas

Elaborado por el equipo de trabajo

Ahora que se obtuvo las piezas, se realizó los ejes y poleas de transmisión con tornería y se realizó perforaciones. Se realizó los soportes de los motores 1, 2, 3 y 4 en las piezas. Luego se realizó el pintado de las piezas terminadas.

Figura 101: Piezas pintadas



Elaborado por el equipo de trabajo

4.1.4.2 Ensamblaje de Piezas

Una vez preparadas las piezas metálicas son soportes, se desarrolla en ensamblaje.

Figura 102: Base con plataforma

a b

Elaborado por el equipo de trabajo

Figura 103: Ensamble de base con eslabón 1

a b

Elaborado por el equipo de trabajo

En el ensamble se usó piezas fabricadas de transmisión de movimiento como en la figura 104, se usó ejes tensadores y poleas como las figura 104a y 104e. Los rodamientos de la figura 104b se usó para el movimiento de los ejes con anillos de anti rozamiento de la figura 104g. Los ejes de transmisión de movimiento usaron poleas de la figura 104c con correas de la figura 104i que se conectaron con estas mismas poleas en

los ejes de motores, se usaron anillos de sujeción de la figura 104d para sujetar a estas. También se usó pernos de la figura 104f con diferentes medidas para fijar todas las partes del sistema de transmisión y motores a las piezas. En la figura 104h se ve el rodamiento axial de bolas para el movimiento firme del eslabón 1.

El diseño se realizó para los ejes y se puede observar en el Anexo 3: AutoCAD, ejes para el sistema de transmisión del brazo robot y las poleas como se muestra en el Anexo 3: AutoCAD, poleas para el sistema de transmisión del brazo robot.

Figura 104: Partes del sistema de transmisión

Elaborado por el equipo de trabajo

En la figura 105 se ve el resultado del ensamblaje con las piezas mencionadas, el brazo robot tiene una conexión de los cables de alimentación a los motores en el gabinete de control. Esto se ha realizado sin un Gripper.

Figura 105: Ensamble del brazo robot con conexión de alimentación

Elaborado por el equipo de trabajo

4.1.5 Implementación del Gripper 1

4.1.5.1 Fabricación de Piezas

Las piezas como base y pinzas se diseñaron en el programa SolidWorks y se imprimió en una impresora 3D usando PLA como material de impresión.

Figura 106: Base impresa en 3d

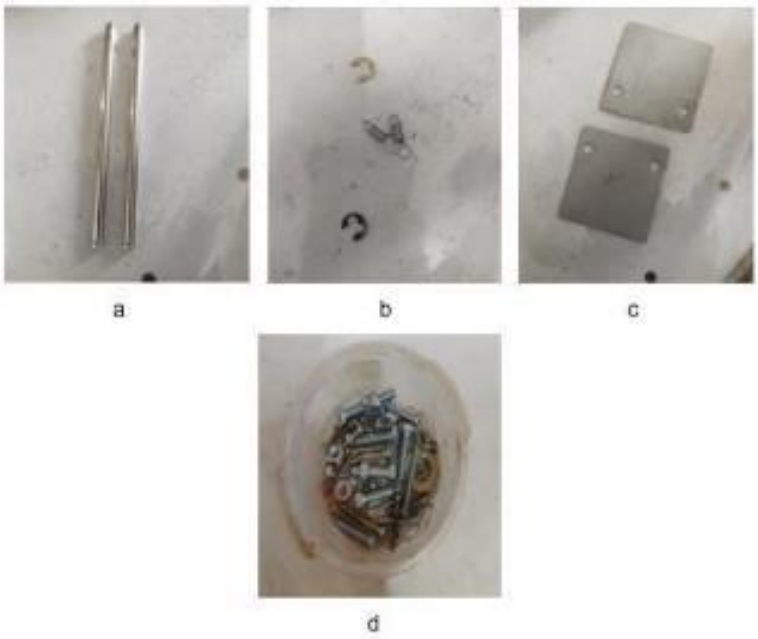
Elaborado por el equipo de trabajo

Figura 107: Pinzas impresas en 3d

Elaborado por el equipo de trabajo

La impresión tiene un acabado sin relleno, solo en los bordes con grosor de 5mm y tiende a ser débil. Además, en la figura 108 se ve los materiales adicionales que se usó como ejes, anillos de sujeción y resortes, placas de apoyo y pernos con tuercas.

Figura 108: Materiales adicionales de gripper

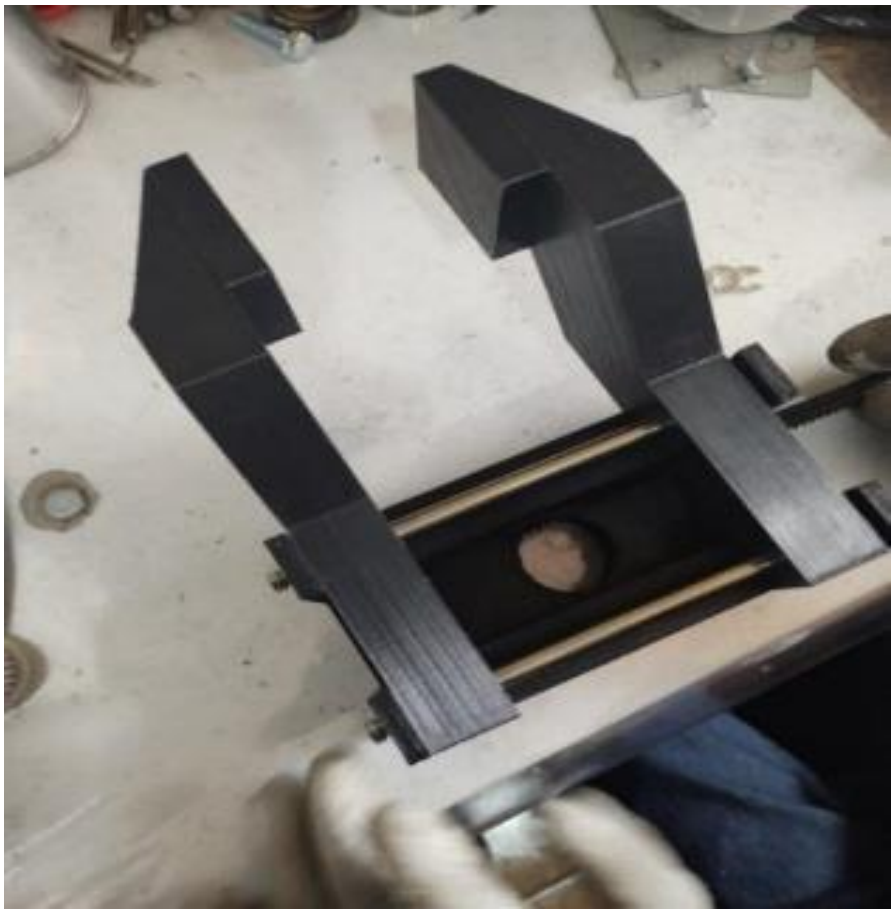


Elaborado por el equipo de trabajo

4.1.5.2 Ensamble de Piezas

Se desarrollo en ensamble con las piezas en 3d y materiales adicionales, también se usó un motor de 24V CC para el movimiento del mecanismo. Se obtuvo un agarre forzado con errores de sujeción.

Figura 109: Ensamblaje de gripper 1



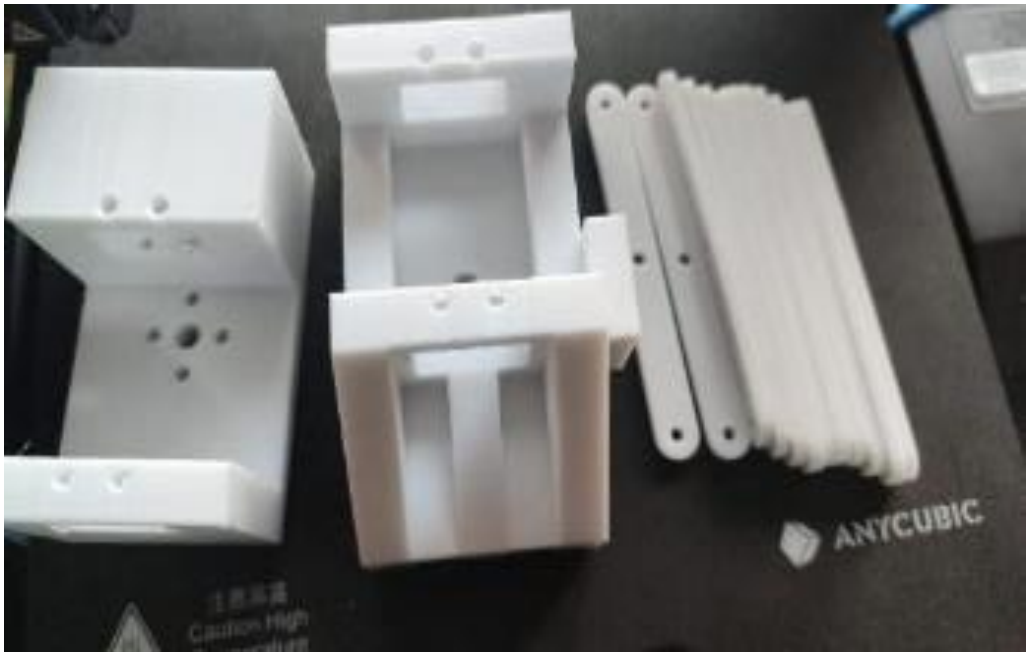
Elaborado por el equipo de trabajo

4.1.6 Implementación del Gripper 2

4.1.6.1 Fabricación de Piezas

En el diseño del gripper 2 de igual manera se usó el programa SolidWorks y se imprimió en una impresora 3D. Esta vez se realizó un diseño con más presión en las piezas, similares a una mano usando material más resistente a la presión como el PETG.

Figura 110: Piezas del gripper 2



Elaborado por el equipo de trabajo

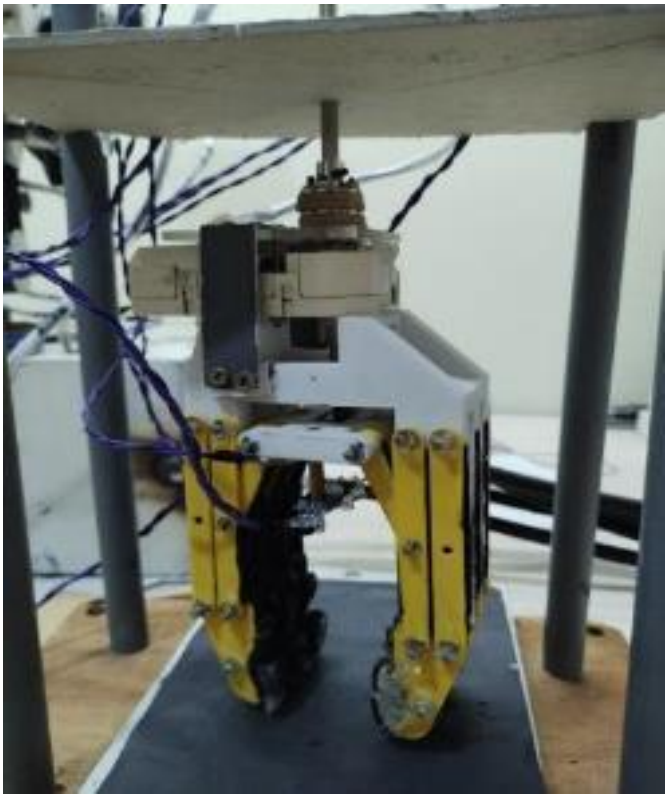
Las piezas tienen un acabado más duradero y son resistentes a las temperaturas moderadas.

4.1.6.2 Ensamble de Piezas

En el ensamblaje se usó ejes y anillos de sujeción de las figuras 108a y 108b, unas almohadillas para el agarre y pernos con tuerca de la figura 108d.

En el Gripper 2 se obtuvo un buen agarre del objeto el motor de 24V CC y da un movimiento capaz de mover todo el mecanismo, los sensores como finales de carrera, infrarrojo y fsr nos dieron un sistema eficiente. La programación con arduino para el funcionamiento de este Gripper se puede ver en el Anexo 1: Arduino, control de Gripper.

Figura 111: Conexión de motor y sensores del gripper 2



Elaborado por el equipo de trabajo

4.1.7 Sistemas de Control

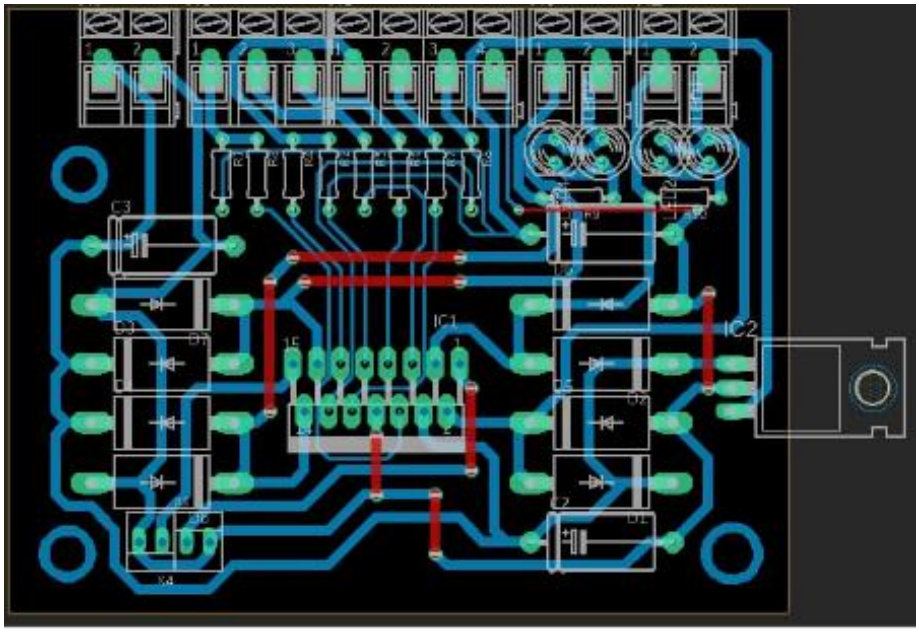
En el control de sistemas se usaron tarjetas electrónicas, estos se montaron en un gabinete de 60x60x15cm, en él se instaló un riel din, canaletas para el cableado, entrada de alimentación, borneras y salidas en la tapa para los motores y sensores.

Los controladores, módulos y dispositivos electrónicos como: Arduino mega, driver DM320T, driver DM542T, fuentes de alimentación y puente h están situados en 4 partes empezando por la alimentación, controladores, módulos y conexiones auxiliares como se muestra en la figura 114.

4.1.7.1 Desarrollo del Puente H

El desarrollo del puente H se realizó en el programa Eagle, en el cual se hizo un esquema esquemático y modelo de placa.

Figura 112: Diseño de pcb del puente h



Elaborado por el equipo de trabajo

En la figura 113 se ve la impresión en una placa de baquelita y soldadura de los componentes.

Figura 113: Conexión de motor y sensores del gripper 2

Elaborado por el equipo de trabajo

4.1.7.2 Ensamble de Gabinete y Dispositivos

Se desarrolló el gabinete para los sistemas de control y se realizó la conexión del controlador con los módulos, puente h y la alimentación de estos. Para ello se usó cables de tipo industrial para la unión de bornera a bornera. En la figura 114 se ve el peinado de cables en las canaletas y las secciones por dispositivos electrónicos, este gabinete tiene un agujero circular para una ventilación de los componentes. El plano de ello se encuentra en el Anexo 3: Diagrama de conexión, dispositivos electrónicos.

Figura 114: Cableado y conexión de los componentes de control

Elaborado por el equipo de trabajo

4.1.8 Evaluaciones Finales

4.1.8.1 Evaluación del Gripper 1

Se realizo la prueba de agarre implementado en el brazo robot con el primer diseño de Gripper.

Figura 115: Brazo robot con gripper 1

Elaborado por el equipo de trabajo

Esto nos dejó un valor diferente para cada presión establecida, en la tabla 13 se visualiza estas características con el sensor de peso FSR.

Tabla 13: Valores utilizados en el griper 1

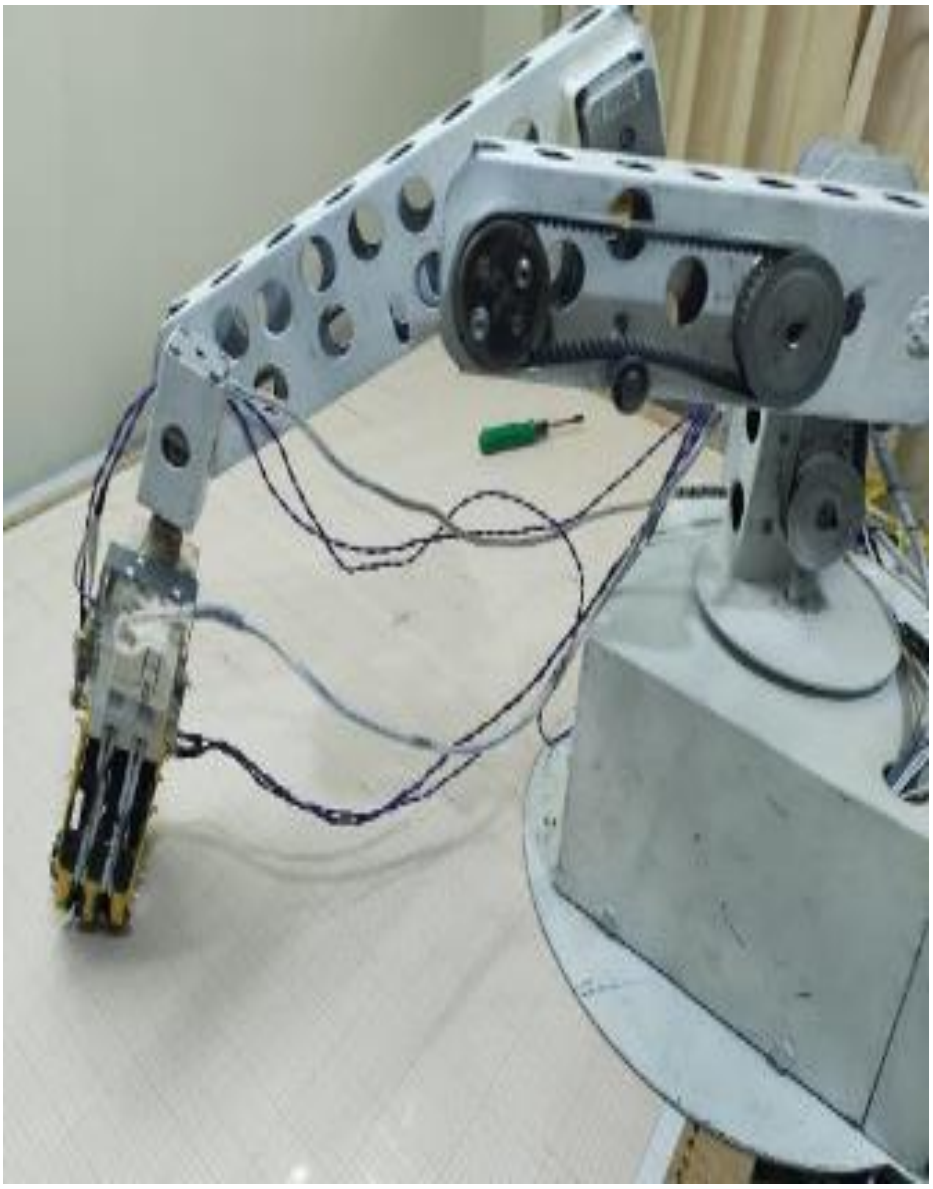
VALOR DE LA PRESIÓN (KF)	DESCRIPCIÓN
0,5	Agarra el objeto, pero no lo eleva, las pinzas están en buen estado.
0.6	Agarra el objeto, pero solo lo mueve no logra elevarlo, y las pinzas están en buen estado.
0.9	Agarra el objeto, lo eleva y el objeto se cae, las pinzas presentan una deformación ligera.
1.2	Agarra el objeto, lo eleva, pero lo mantiene en el aire con dificultad, las pinzas presentan deformaciones drásticas.

Elaborado por el equipo de trabajo

4.1.8.2 Evaluación del Gripper 2

Se realizo la prueba de agarre implementado en el brazo robot con el segundo diseño de Gripper.

Figura 116: Brazo robot con gripper 2



Elaborado por el equipo de trabajo

Los valores de presión establecidos se visualizan en la tabla 14 y fue tomadas con el sensor de peso FSR.

Tabla 14: Valores utilizados en el griper 2

VALOR DE LA PRESIÓN (KF)	DESCRIPCIÓN
0,9	Agarra el objeto, lo eleva y lo mantiene con dificultad, las pinzas están en buen estado.
1.2	Agarra el objeto, lo eleva y lo mantiene con dificultad, y las pinzas están en buen estado.
1.5	Agarra el objeto, lo eleva y mantiene, las pinzas están en buen estado.
2	Agarra el objeto, lo eleva y mantiene en el aire constante, las pinzas están en buen estado.

Elaborado por el equipo de trabajo

4.1.8.3 Evaluación del Brazo Robot

En este paso se realizó la evaluación con distintos valores para puntos deseados y se vio los resultados, en el que podemos observar el cambio en cada uno de los valores articulares con los puntos asignados, el cambio en la asignación de los ángulos para cada grado de libertad en puntos secuenciales es mínima, pero para puntos distantes cambia drásticamente. El plano del diseño final se puede ver en el Anexo 3: SolidWorks, brazo robot con Gripper.

Tabla 15: Evaluación del brazo robot

Punto deseado			Valores articulares asignados por el sistema				
Xd	Yd	Zd	Q1	Q2	Q3	Q4	Q5
-36	39	4	-45°	-27°	-30°	25°	+90°
-30	19	7	-44°	-30°	-26°	-30°	+90°
-40	51	3	-49°	-70°	-80°	80°	+90°
-24	25	8	-30°	-70°	+80°	-80°	+90°
-32	22	4	-39°	-70°	-80°	-60°	+90°
-27	34	5	-20	-70	-80°	-65°	+90°
30	35	10	40	-40°	-36°	-35°	+90°
31	24	50	43	-50°	-40°	-54°	+90°
0	0	70	0	0	80°	+90°	+90°

Elaborado por el equipo de trabajo

Tabla 16: Resultados de la evaluación del error

Valor experimental			ERROR	ERROR	ERROR
XE	YE	ZE	X	Y	Z
-37	41	4	2.63	4.87	0
-31	20	7	3.22	5.0	0
-42	49	3	4.76	4.0	0
-25	25	8	4.0	0	0
-31	21	4	3.22	4.76	0
-28	33	5	3.70	3.0	0
31	31	23	3.0	2.94	4.34
32	23	49	3.12	4.16	2.0
0	0	74	0	0	1.35

Elaborado por el equipo de trabajo

Se observo que el Gripper 2 realiza un buen agarre del objeto, la cinemática nos da el movimiento al objetivo, esto tuvo una secuencia de prueba y error para su funcionamiento.

Figura 117: Agarre de objeto con cinemática

Elaborado por el equipo de trabajo

El brazo robot comienza en un punto inicial y de ahí va a recoger el objeto, una vez encontrado retorna a su punto inicial.

Figura 118: Agarre de objeto con brazo robot en posición inicial

Elaborado por el equipo de trabajo

4.1.8.4 Evaluación de la Visión Artificial

Se obtuvo una detección buena del objeto con error de 1cm de medidas con respecto al plano coordenado.

Figura 119: Detección de objetos en el cuadrante donde se recibe

Elaborado por el equipo de trabajo

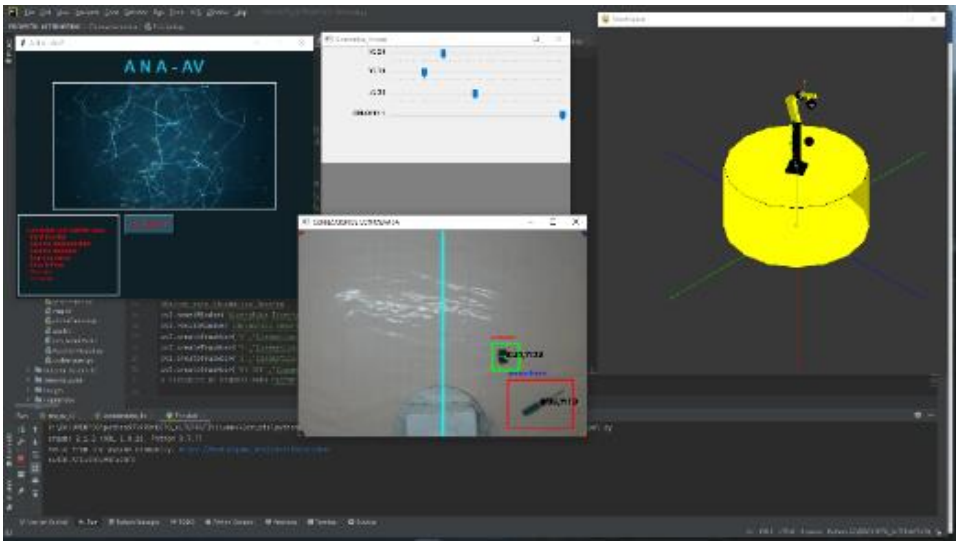
Figura 120: Detección de objetos en ambos cuadrantes

Elaborado por el equipo de trabajo

4.1.8.5 Evaluación del reconocimiento de voz con los demás sistemas

Se obtuvo una interacción de programas de los diferentes sistemas en PyCharm, en la figura 121 se visualiza diversas interfaces graficas cada una con una función independiente.

Figura 121: Control de programas a partir de un principal



Elaborado por el equipo de trabajo

4.2 DISCUSIÓN

Según Bharath Sai et al. (2017) en su artículo científico presento un robot humanoide, el cual fue capaz de detectar y reconocer diferentes caras u objetos y está controlado mediante la voz, utilizando materiales como Arduino Mega, micrófono, modulo EasyVR y un robot móvil que para su movimiento está controlado por la voz, para la programación se utilizó el lenguaje Python y su complemento OpenCV. El presente proyecto coincide con el antecedente en la programación sobre la visión artificial, reconocimiento de voz, para el control se utiliza la tarjeta Arduino Mega. Diferencian el presente proyecto presenta un robot en un punto fijo y el antecedente presenta un robot móvil, el cuanto al reconocimiento de voz el antecedente presenta la tarjeta EasyVR y el proyecto presenta un software. El antecedente referido al robot

humanoide evidencian que tiene más utilidades con respecto al brazo robot desarrollado en el proyecto el cual es de punto fijo.

De acuerdo con Medrano (2020) en su tesis tuvo como resultado un brazo robot con tres grados de libertad en el que se utilizaron materiales como motores nema 17, un Arduino mega, engranajes, piñones de metacrilato y la estructura en madera MDF tal estructura fue diseñada en el software CorelDraw. La construcción y la programación tuvo un resultado satisfactorio, ya que se obtuvo un funcionamiento optimo. A Diferencia de presente proyectos el diseño es más robusto ya que la construcción se realizó de estructura metálica con más grados de libertad. El antecedente referido a la implementación del brazo robot evidencian el funcionamiento mecánico los cuales diferencian con los tipos de materiales utilizados para su implementación, la aplicación y la complejidad en el diseño lo cual indica que el robot implementado en el proyecto tiene un avance tecnológico superior.

Según Muñoz y Serrato (2018) en su tesis tuvo como resultado un controlador de bajo costo el cual genero trayectorias en línea recta de forma automática del robot DM-5200 LABVOLT 2001. El antecedente referido tiene evidencias en el presente proyecto en la generación de trayectoria con los respectivos estudios cinemáticos que permitieron el funcionamiento satisfactorio en ambos casos y con la diferencia del error generado en el posicionamiento del plano.

De acuerdo con Aparicio (2022) en su tesis tuvo como resultado un sistema de visión artificial desarrollado con el software Mertlic con un sistema inteligencia artificial desarrollado con el software Deep Learning Tool. El antecedente referido a la detección de LEGO evidencia que para la implementación de sistemas de visión artificial existe softwares dedicados como Merlic, el cual no se implementó en el presente proyecto, pero se desarrolló un software en Python con OpenCV que fue capaz de realizar dicha tarea

satisfactoriamente y que el software dedicado como Merlic tiene mejores resultados en la detección de objetos en movimiento.

Según Zhang et al. (2020) en su artículo tuvo resultados óptimos en la adquisición del color con métodos como la conversión a HSV, umbralización y punto centro para el agarre de objetos con motores paso a paso. El antecedente referido tiene evidencias del procedimiento de detección por colores, pero es limitado con respecto a la detección especifica de un objeto con diferente forma y color. Mientras que el presente proyecto utiliza la detección de color con los métodos mencionados para establecer puntos y guardarlos, de tal manera no solo nos basamos en la detección de color si no en clasificadores de imágenes.

De acuerdo con Romero y Villegas (2017) en su tesis tuvo como resultado un funcionamiento optimo, respecto al control a distancia y la asistencia por la visión artificial teniendo. El antecedente referido coincide con el proyecto en los cálculos cinemáticos realizados para el movimiento y diferencian en la profundidad del análisis matemático ya que el antecedente tiene solo un análisis de cinemática directa. Mientras que el presente proyecto tiene un desarrollo cinemático completo.

Según León et. al. (2020) en su artículo tuvo un resultado considerable de detección y extracción de muestras fallidas con la visión artificial, para lo cual usaron teoremas de RGB para la detección del color con el fin de implementarlo a un brazo robot. El antecedente referido tiene evidencia sobre la adquisición del color para su procesamiento de imagen y diferencian en la cámara utilizada y el método de iluminación usado. Mientras que en el presente proyecto se optó por una cámara con buena captura de iluminación, sin distorsión de lente y asequible al público para realizar un procesamiento correcto de la imagen.

Según con Román (2021) en su proyecto tuvo un resultado con error de 38% para reconocer la voz, para ello uso actuadores, sensores, procesamiento de información a su sistema mecánico y una aplicación móvil que convierte la voz a texto. El antecedente referido se evidencia el control por patrones de voz a un sistema que se desea controlar, ya que este sistema tiene errores de detección del habla esto diferencia en los motores de detección implementados en la aplicación usada. Mientras que el presente proyecto cuenta con varios motores de detección que funcionan tanto con o sin conexión a internet lo que nos da una funcionalidad estable.

De acuerdo con Romero (2019) en su tesis, detalla el funcionamiento de un brazo robot con cinemática directa, inversa y jacobiano. Pero su diseño de Gripper no es funcional a diferencia del presente proyecto. Este antecedente evidencia del desarrollo matemático establecido y diferencian en los tipos de material utilizados en la implementación, siendo el material del proyecto un producto robusto.

V. CONCLUSIONES

Se diseñó e implementó el análisis matemático y mecanismo de un brazo robot 5GDL con Gripper, en los resultados se obtuvo satisfactoriamente el funcionamiento de la cinemática directa, cinemática inversa y jacobiano con los que realizo la trayectoria del robot. En la implementación mecánica del brazo robot se realizó de acuerdo a los planos diseñados, en el análisis y diseño para la implementación del Gripper se tomó en cuenta tanto el material de impresión 3D como la presión que ejerce el mecanismo del Gripper el cual tuvo un resultado optimo en el momento de realizar las pruebas.

Se realizó la detección de objetos con estimación de posición para el sistema de visión artificial obteniendo resultados satisfactorios ya que se logró detectar imágenes de las herramientas y materiales en la mesa de trabajo utilizando una cámara Logitech 920e, también se logró determinar la posición de dichos objetos. Para determinar las coordenadas de la mesa de trabajo se utilizó el método de transformación de perspectiva, el entrenador de objetos Cascade GUI tiene buenos resultados al realizar el entrenamiento con imágenes.

Se desarrolló el software de un asistente virtual para el control de voz por Patrones, logrando obtener las órdenes a través de patrones de voz en el software mediante el micrófono. En la implementación se utilizó un micrófono estándar logrando resultados satisfactorios, teniendo en cuenta la ubicación del micrófono a una distancia corta del usuario.

VI. RECOMENDACIONES

Los resultados en el diseño e implementación del análisis matemático y mecanismo de un brazo robot 5GDL con Gripper, tiene un buen funcionamiento, pero los márgenes de error aún son considerables en el posicionamiento y la trayectoria que realiza el brazo robot, debido tanto a la estructura como a los componentes físicos, la solución que se le debe dar al problema está en la programación para ello se pueden aplicar más teorías de control como un análisis matemático de la dinámica del brazo robot. Considerar la forma de aligerar el peso de las piezas, para ello ajustar los planos de dichas piezas para una impresión 3D.

En la detección de objetos con estimación de posición para el sistema de visión artificial tiene un funcionamiento optimo, resultado que se ha obtenido utilizando una cámara de calidad media Logitech 920e, pero en caso de utilizar una cámara de bajo coste este siempre presentara las distorsiones de barril o también denominado ojo de pez por lo que es necesario realizar una calibración de cámara para corregir estos errores, calibrar la cámara también permitirá mayor exactitud en los puntos establecidos del plano coordenado, para ello se sugiere utilizar un patrón como el tablero de ajedrez o circulo los cuales ya tienen una medida especifica. Para el entrenamiento de objetos se puede hacer uso de algoritmos más avanzados con redes convolucionales como el Yolov5 o Yolov7 ya que estos contienen objetos pre entrenados.

Seguir trabajando en el desarrollo del software de un asistente virtual para el control de voz por patrones, con el fin de obtener mejores resultados en el reconocimiento de voz, para ello se sugiere optar un micrófono de mayor calidad como uno de tipo condensador el cual ofrece mayor rango de recepción de la voz y mejor ganancia.

VII. REFERENCIAS BIBLIOGRAFICAS

Alegre Gutierrez, E., Pajares Martinsanz, G., & de la Escalera Hueso, A. (2016). *Conceptos y Métodos en Visión por Computador*. Grupo de Visión del Comité Español de Automática (CEA).

AMIN Thou Shalt Programme. (2022, February 29). Cascade Trainer GUI. https://amin-ahmadi.com/cascade-trainer-gui/

Aparicio González, J. (2022). *Aplicaión de "pick and place" de piezas LEGO con vision artificial en un robot colaborativo*. Universidad Politécnica de Valencia.

Aponte Castro, A. E., Castañeda Flores, E. J., & Medina Paredes, G. E. (2020). *Diseño de un algoritmo para crear y entrenar una red neuronal que permita el reconocimiento de voz*. Universidad Nacional de Piura.

Arduino. (2022). *Introducción*. https://www.arduino.cc/en/Guide/Introduction

Barrientos, A., Beñin, L. F., Balaguer, C., & Aracil, R. (1997). *Fundamentos de Robotica* (C. Fernandez Madrid, Ed.). McGraw-Hill.

Bharath Sai, U., Sivanagamani, K., Satish, B., & M, R. R. (2017). Voice controlled Humanoid Robot with artificial vision. *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, 507–208. https://doi.org/10.1109/ICOEI.2017.8300979

Biblioteca de Python. (2022a, March 5). Datetime Basic Date and Time. https://docs.python.org/es/3.7/library/datetime.html

Biblioteca de Python. (2022b, March 26). Threading Thread Based Parallelism. https://docs.python.org/3.7/library/threading.html

Bravo Romero, B., & Villegas Medina, J. E. (2017). *Diseño e Implementación de un Prototipo de Brazo Robótico (4gl) Teleoperado para Manipulación de Sustancias*

Tóxicas asistido con Visión Artificial y Redes Neuronales para Laboratorios Farmacéuticos. Universidad Catolica de Santa Maria.

Documentación de Python. (2022a, March 25). Subprocess Managment. https://docs.python.org/3.7/library/subprocess.html

Documentación de Python. (2022b, March 26). Tkinter Interface of Python. https://docs.python.org/es/3/library/tkinter.html

EcuRed. (2021, December 27). Enfoque (Fotografía). https://www.ecured.cu/Enfoque_(Fotograf%C3%ADa)

EcuRed. (2022, March 6). OpenCV. https://www.ecured.cu/OpenCV

Fernández Aragón, I. (2011). *Control de un motor paso a paso: PIC, USB y C#*. https://hdl.handle.net/2454/3547

Fu, K. S., Gonzales, R. C., & Lee, C. S. (1988). *Robotica: Control, Vision e Inteligencia*. McGraw-Hill.

Guerra Carmenate, J. (2021, December 28). *Programarfacil*. Arduino Mega 2560 El Hermano Mayor de Ardunio UNO. https://programarfacil.com/blog/arduino-blog/arduino-mega-2560/

Hernadez Sampieri, R. (2014). *Metodología de la Investigación*. McDRAW-HILL / INTERAMERICANA EDITORES, S.A. DE C.V.

HTA 3D. (2022, January 16). How to Connect the Stepper Motor - Quick Guide. https://www.hta3d.com/en/blog/Connect-Stepper-Motor

Kaehler, A., & Bradski, G. (2016). *Learning OpenCV3 Computer Vision in C++ with the OpenCV Library*. O'Reilly y Media.

Leon, R., Alvarado, E., Arevalo, K., Maldonado, A., & Polonio, A. (2020, February 4). Detección y Extracción de Muestras Falladas usando Visión Artificial y un Brazo Robotico. *Revista Ciencia y Tecnologia*.

MathWorks. (2022, February 28). Matlab. https://la.mathworks.com/products/matlab.html

Medrano Trujillo, J. I. (2020). *Brazo robot de bajo coste con motores paso a paso que escribe*. Universidad de Sevilla.

Minichino, J., & Howse, J. (2015). *Learning OpenCV 3 Computer Vision with Python* (2nd ed.). Packt Publishing Ltd.

Muñoz Giraldo, S., & Serrato Rodriguez, D. A. (2018). *Diseño de un Sistema de Control de Bajo Costo para la Generación de Trayectorias del Brazo Robótico DM-5200 LABVLOT 2001*.

Numpy. (2022, March 8). Documentación de Numpy. https://numpy.org/doc/stable/

OpenCV Open Source Computer Vision. (2022, February 2). Opencv-Python Tutorials. https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

Pillow. (2022, March 6). Documentación de Pillow. https://pillow.readthedocs.io/en/stable/

Proyectos de Python. (2022, March 25). Pyttsx3. https://pypi.org/project/pyttsx3/

PyGame. (2022, January 24). About Pygame. https://www.pygame.org/wiki/about

Pyserial. (2022, March 14). Welcome to PySerial's Documentation.

Python. (2022, March 7). The Python Tutorial. https://docs.python.org/3.7/tutorial/index.html

Road Zhongke, J. (2017a). *User manual DM320T*.

Road Zhongke, J. (2017b). *User Manual DM542T*.

Roman Gonzales, S. J. (2021). *Diseño e Implementación de un prototipo alpha de tablero de ajedrez con sistema de desplazamiento de los trebejos con comandos de voz y oponente remoto a través de una aplicación móvil*. Universidad Tecnológica del Perú.

Romero Aguirre, J. C. (2019). *Desarrollo de Sistema de Control para Prototipo de Robot Antropomórfico*. Universidad Nacional del Altiplano.

Rosebrock, A. (2017). *Deep Learning for Computer Vision* (1st ed.). PyImageSearch.

Solidworks. (2021). *Documentación*. https://solid-bi.es/solidworks/

STEPPERONLINE. (2019a). Digital Stepper Driver 1.0-4.2A 20-50VDC for Nema 17, 23, 24 Stepper Motor. https://www.omc-stepperonline.com/digital-stepper-driver-1-0-4-2a-20-50vdc-for-nema-17-23-24-stepper-motor-dm542t

STEPPERONLINE. (2019b). Digital Stepper Driver 0.3-2.2A 10-30VDC for Nema 8, 11, 14, 16, 17 Stepper Motor. https://www.omc-stepperonline.com/digital-stepper-driver-0-3-2-2a-10-30vdc-for-nema-8-11-14-16-17-stepper-motor-dm320t

STEPPERONLINE. (2019c). Digital Stepper Driver 0.3-2.2A 10-30VDC for Nema 8, 11, 14, 16, 17 Stepper Motor. https://www.omc-stepperonline.com/digital-stepper-driver-1-0-4-2a-20-50vdc-for-nema-17-23-24-stepper-motor-dm542t

Szeliski, R. (2022). *Computer Vision: Algorithms and Applications* (2nd ed.). Springer.

Vélez Serrano, J. F., Moreno Díaz, A. B., Sánchez Calle, Á., & Sánchez Marín, J. L. E. (2003). *Visión por Computador*. Vision Computacional y Vision Artificial.

Viola, P., & Jones, M. (2001). Rapid Object Detection Using a Boosted Cascade of Simple. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, 1–1. https://doi.org/10.1109/CVPR.2001.990517

Wikipedia. (2021a, January 27). Diafragma (Óptica). https://es.wikipedia.org/wiki/Diafragma_(%C3%B3ptica)

Wikipedia. (2021b, December 27). Sensor de Imagen. https://es.wikipedia.org/wiki/Sensor_de_imagen

Wikipedia. (2021c, December 27). Apertura. https://es.wikipedia.org/wiki/Apertura

Wikipedia. (2021d, December 28). Distancia Focal. https://es.wikipedia.org/wiki/Distancia_focal

Wikipedia. (2022a, January 18). Puente H (Electrónica). https://es.wikipedia.org/wiki/Puente_H_(electr%C3%B3nica)

Wikipedia. (2022b, January 25). Ramer Douglas Peucker Algorithm. https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

Wikipedia. (2022c, February 2). Momentos de Imagen. https://es.wikipedia.org/wiki/Momentos_de_imagen

Wikipedia. (2022d, March 5). Reconocimiento de Habla. https://es.wikipedia.org/wiki/Reconocimiento_del_habla

Wikipedia. (2022e, March 5). Micrófono. https://en.wikipedia.org/wiki/Microphone

Xataca. (2021, December 29). Que Es Arduino, Como Funciona y Que Puedes Hacer Con Uno. https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

YouTube. (2022, February 2). Momentos de Un Objeto. https://www.youtube.com/watch?v=sPGfnYuj0-Y&t=419s&ab_channel=ProyectosJC

Zhang, W., Zhang, C., Li, C., & Zhang, H. (2020). Object color recognition and sorting robot based on OpenCV and machine vision. *2020 IEEE 11th International Conference on Mechanical and Intelligent Manufacturing Technologies*, 125–129. https://doi.org/10.1109/ICMIMT49010.2020.9041220

ANEXOS

ANEXO 1: CÓDIGO DE PROGRAMACIÓN

Arduino - Recepción de Datos y Control de Motores

```
int PUL1 = 2;
int DIR1 = 3;
int PUL2 = 4;
int DIR2 = 5;
int PUL3 = 6;
int DIR3 = 7;
int PUL4 = 8;
int DIR4 = 9;
int PUL5 = 10;
int DIR5 = 11;
int PUL6 = 40;
int DIR6 = 41;
int SET1 = 14;
int SET2 = 15;
int SET3 = 16;
int SET4 = 17;
int SET5 = 18;
int SET6 = 19;
boolean newData = false;
String data = "";
int teta1, teta2, teta3, teta4, teta5, teta6;
int N_P1 = 0, N_P2 = 0, N_P3 = 0, N_P4 = 0, N_P5 = 0, N_P6 = 0;


void setup()
{
  Serial.begin(115200);
  pinMode(PUL1, OUTPUT);
  pinMode(DIR1, OUTPUT);
  pinMode(PUL2, OUTPUT);
  pinMode(DIR2, OUTPUT);
  pinMode(PUL3, OUTPUT);
  pinMode(DIR3, OUTPUT);
  pinMode(PUL4, OUTPUT);
  pinMode(DIR4, OUTPUT);
  pinMode(PUL5, OUTPUT);
  pinMode(DIR5, OUTPUT);
  pinMode(SET1, OUTPUT);
  pinMode(SET2, OUTPUT);
  pinMode(SET3, OUTPUT);
  pinMode(SET4, OUTPUT);
  pinMode(SET5, OUTPUT);
  pinMode(SET6, OUTPUT);
  digitalWrite(SET1, HIGH);
  digitalWrite(SET2, HIGH);
  digitalWrite(SET3, HIGH);
  digitalWrite(SET4, HIGH);
  digitalWrite(SET5, HIGH);
  digitalWrite(SET6, HIGH);

}
```

```
void loop()
{
  reconocer();
  if (newData == true)
  {
    entero();

    while (teta1 > N_P1)
    {
      digitalWrite(DIR1, HIGH);
      digitalWrite(PUL1, HIGH);
      delayMicroseconds(500);
      digitalWrite(PUL1, LOW);
      delayMicroseconds(500);
      N_P1 = N_P1 + 1;
    }
    while (teta1 < N_P1)
    {
      digitalWrite(DIR1, LOW);
      digitalWrite(PUL1, LOW);
      delayMicroseconds(500);
      digitalWrite(PUL1, HIGH);
      delayMicroseconds(500);
      N_P1 = N_P1 - 1;
    }

    while (teta2 > N_P2)
    {
      digitalWrite(DIR2, HIGH);
      digitalWrite(PUL2, HIGH);
      delayMicroseconds(500);
      digitalWrite(PUL2, LOW);
      delayMicroseconds(500);
      N_P2 = N_P2 + 1;
    }
    while (teta2 < N_P2)
    {
      digitalWrite(DIR2, LOW);
      digitalWrite(PUL2, LOW);
      delayMicroseconds(500);
      digitalWrite(PUL2, HIGH);
      delayMicroseconds(500);
      N_P2 = N_P2 - 1;
    }

    while (teta3 > N_P3)
    {
      digitalWrite(DIR3, HIGH);
      digitalWrite(PUL3, HIGH);
      delayMicroseconds(500);
      digitalWrite(PUL3, LOW);
      delayMicroseconds(500);
```

```
}

void reconocer()
{
  while (Serial.available())
  {
    //Serial.println(data);
    char character;
    char salto = '\n';
    character = Serial.read();
    if (character != salto)
    {
      data.concat(character);
    }
    else
    {
      newData = true;
    }

  }
}
void entero()
{
  if (newData == true)
  {
    char a[30];
    data.toCharArray(a, 30);
    data = "";
    if ((a[0] == 'A') && (a[4] == 'B') && (a[8] == 'C') && (a[12] == 'D') && (a[16]
== 'E') && (a[20] == 'F'))
    {
      //Serial.println("Los caracteres son validos");
      char ang1[5] = {a[1], a[2], a[3]};
      int val1 = atoi(ang1);
      char ang2[5] = {a[5], a[6], a[7]};
      int val2 = atoi(ang2);
      char ang3[5] = {a[9], a[10], a[11]};
      int val3 = atoi(ang3);
      char ang4[5] = {a[13], a[14], a[15]};
      int val4 = atoi(ang4);
      char ang5[5] = {a[17], a[18], a[19]};
      int val5 = atoi(ang5);
      char ang6[5] = {a[21], a[22], a[23]};
      int val6 = atoi(ang6);

      teta1 = (val1 * 22.22222);
      teta2 = (val2 * 55.55555);
      teta3 = (val3 * 55.55555);
      teta4 = (val4 * 11.11111);
      teta5 = (val5 * 21.33333);
      teta6 = (val6 * 15.299555555);

    }


  }

}
```

Arduino - Control de Gripper

```
int PUL = 8;
int DIR = 9;
int SET = 10;
int SW2 = 7;
int SW1 = 6;
int INF = 11;
int FSR = A0;
int lectura;
int valor = 0;
int estado;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(PUL, OUTPUT);
  pinMode(DIR, OUTPUT);
  pinMode(SET, OUTPUT);
  pinMode(SW2, INPUT);
  pinMode(SW1, INPUT);
  pinMode(INF, INPUT);
  pinMode(FSR, INPUT);
  digitalWrite(SET, HIGH);
}

void loop() {
  estado = digitalRead(SW2);
  valor = digitalRead(INF);
  lectura = analogRead(FSR) / 255;

  primero();

  if (valor == LOW) {
    digitalWrite(PUL, LOW);
    digitalWrite(DIR, HIGH);
  }

  if (lectura >= 2) {
    digitalWrite(PUL, LOW);
    digitalWrite(DIR, LOW);
    Serial.print("lectura: ");
    Serial.print(lectura);
    Serial.println();
    delay(6000);
    digitalWrite(PUL, HIGH);
    digitalWrite(DIR, LOW);
    delay(18000);

    //Serial.print("sacar el material de la garra");
  }
}

void primero() {
  if (estado == 0 && valor == 1) {


    digitalWrite(PUL, HIGH);
    digitalWrite(DIR, LOW);
  } else {
    digitalWrite(PUL, LOW);
    digitalWrite(DIR, LOW);
  }
}
```

Matlab – Ecuación de Matriz Final

```
function Y =E_C_DE_M_FINAL (t,u)
dT = zeros(7,1); %3 Grados de libertad
% Paso 1
%Declarar las dimensiones del Robot
L1=40;%cm
L2=30;%cm
L3=30;%cm
L4=5;%cm
L5=15;%cm
% Paso 2
% Entradas de la función
T1=u(1);
T2=u(2);
T3=u(3);
T4=u(4);
T5=u(5);
ti=t; %tiempo
% Paso 3
% Evaluar cinemática directa

 x=L3*(cos(T1)*cos(T3)*cos(pi/2 + T2) - cos(T1)*sin(T3)*sin(pi/2 + T2)) - L4*si
 y=L4*cos(T1) + L3*(cos(T3)*cos(pi/2 + T2)*sin(T1) - sin(T1)*sin(T3)*sin(pi/2 +
 z=L1 - L2*sin(T2 - pi/2) - L5*(cos(T4)*(cos(T3)*sin(T2 - pi/2) - cos(T2 - pi/2
 X=[x;y;z];
% Paso 4
%Declarar el Jacobiano calculado previamente(revisar scrip de jacobiano)
J =[ - L4*cos(T1) - L3*(cos(T3)*cos(pi/2 + T2)*sin(T1) - sin(T1)*sin(T3)*sin(pi
       L3*(cos(T1)*cos(T3)*cos(pi/2 + T2) - cos(T1)*sin(T3)*sin(pi/2 + T2)) - L



% Paso 5
%Calcular la pseudoInversa del jacobiano
JT=J'*inv(J*J');
% Paso 6
%Definición de la trayectoria en el espacio cartesiano
%Para este caso una linea recta en el plano z=10;
w=2;%frecuencia
xd=20;
yd=20;
zd=17;
Xd=[xd; yd; zd]; %pocision deseada

xd_d=0;
yd_d=0;
zd_d=0;
Xd_d=[xd_d;yd_d;zd_d]; %velocidad deseada
% Paso 7
%Definición de la matriz de ganancia del controlador
  K=  [.8   0   0;...
       0   .8  0;...
       0    0  .8];
% Paso 8
%Cálculo de la función  dq y declaración de salida
error=Xd-X;


salida=[X ; Xd]
dT=JT*(Xd_d+K*(error));
Y= [dT];
```

Matlab – Trayectoria de Brazo Robot

```
%Solucion de la trayectoria
clc;clear;
t0=0;  tf=2;
h=0.5;
%condiciones iniciales
T0=[0,0,.1,.1,.1];
%resolvemos la ecuacion diferencial
[t,T]=ode45(@E_C_DE_M_FINAL,[t0:h:tf],T0);

%separamos los elementos de la matriz " q " obtenida
T1=T(:,1)*180/pi;
T2=T(:,2)*180/pi;
T3=T(:,3)*180/pi;
T4=T(:,4)*180/pi;
T5=T(:,5)*180/pi;
hold on
axis([-100 100 -100 100 -100 100])
xlabel('x')
ylabel('Y')
zlabel('z')
%Graficamos la trayectoria obtenida
%Es decir la trayectoria que realiza el efector final

    L1=40;%cm
    L2=30;%cm
    L3=30;%cm
    L4=5;%cm
    L5=15;%cm
%----------Cinemática Directa de todos los eslabones del robot
x5=L3*(cos(T(:,1)).*cos(T(:,3)).*cos(pi/2 + T(:,2)) - cos(T(:,1)).*sin(T(:,3)).
y5=L4*cos(T(:,1)) + L3*(cos(T(:,3)).*cos(pi/2 + T(:,2)).*sin(T(:,1)) - sin(T(:,
z5=L1 - L2*sin(T(:,2) - pi/2) - L5*(cos(T(:,4)).*(cos(T(:,3)).*sin(T(:,2) - pi/

%plot3(x5,y5,z5)
%-----------
x4=L3*(cos(T(:,1)).*cos(T(:,3)).*cos(pi/2 + T(:,2)) - cos(T(:,1)).*sin(T(:,3)).
y4=L4*cos(T(:,1)) + L3*(cos(T(:,3)).*cos(pi/2 + T(:,2)).*sin(T(:,1)) - sin(T(:,
z4=L1 - L2*sin(T(:,2) - pi/2) - L3*(cos(T(:,3)).*sin(T(:,2) - pi/2) - cos(T(:,2

%plot3(x4,y4,z4)
%------------
x3=L3*(cos(T(:,1)).*cos(T(:,3)).*cos(pi/2 + T(:,2)) - cos(T(:,1)).*sin(T(:,3)).
y3=L3*(cos(T(:,3)).*cos(pi/2 + T(:,2)).*sin(T(:,1)) - sin(T(:,1)).*sin(T(:,3)).
z3=L1 - L2*sin(T(:,2) - pi/2) - L3*(cos(T(:,3)).*sin(T(:,2) - pi/2) - cos(T(:,2
%plot3(x3,y3,z3)
%------------
x2=L2*cos(T(:,1)).*cos(pi/2 + T(:,2));
y2=L2*cos(pi/2 + T(:,2)).*sin(T(:,1));
z2=L1 - L2*sin(T(:,2) - pi/2);
%plot3(x2,y2,z2)
%-------------
x1 =0;
y1 =0;
z1 =L1;
%-----------
```

```
x0 =0;
y0 =0;
z0 =0;
%Grafico del Robot
i=0;
%Mover Robot
%Configuracion Serial para Arduino Mega
s = serial('COM3');
flag = 1;
set(s,'DataBits',8);
set(s,'StopBits',1);
set(s,'BaudRate',9600);
set(s,'Parity','none');
fopen(s);
pause(2)

for tiempo = t0:h:tf
        i=i+1;
    a=TRAZA_FINAL(x2(i),y2(i),z2(i),x3(i),y3(i),z3(i),x4(i),y4(i),z4(i),x5(i),y
    view([i 45])

    dato1=FORMATO_90(round(T1(i)));
    dato2=FORMATO_90(round(T2(i)));
    dato3=FORMATO_90(round(T3(i)));
    dato4=FORMATO_90(round(T4(i)));
    dato5=FORMATO_90(round(T5(i)));

    salida=strcat('A',dato1,'B',dato2,'C',dato3,'D',dato4,'E',dato5,'F+90\n')
    fprintf(s, salida)

    pause(.1)
    if tiempo<tf-h
        delete(gca)
    end
end

pause(2);
fclose(s);
```

Python – Librería de la Cinemática

```
import pygame, sys,random, time , math
pi=math.pi
i=-300
gama=pi/6
inc=0
inc2=0
teta=0
beta=0

#Colores
BLACK2=(56,56,56);BLACK =(0,0,0);WHITE=(255,255,255);GREEN=(0,255,0);GREEN2
=(60,255,30)
RED=(255,0,0);RED2=(100,150,150);BLUE=(0,0,255);YELLOW=(255,255,0);
YELLOW2=(180,180,0)


#Funcion Vacia para los TrackBar
def nothing(x):
   pass

#Funciones Trigonométricas
def cos(q):
    a=math.cos(q)
    return a
def sin(q):
    a=math.sin(q)
    return a
def da_formato(a):  #ingresa un valor en grados entre -90 y 90
    #obtenemos el signo
    signo='+'
    if a>0:
        signo='+'
    if a<0:
        signo='-'
    if a==0:
        signo='+'
    b=int(a)
    #preguntamos si se trata de decenas
    if abs(b)>9.9:
        return (signo + str(abs(b)))
    #preguntamos si se trata de solo unidades
    if abs(b)<10:
        return (signo + '0' + str(abs(b)))



                #asp ,rz , rx
def pixel(x,y,z,gama,teta,beta,escala): #funcion de ubicación en la pantalla
    x=x*escala
    y=y*escala
    z=z*escala
```

```
    #transformación RZ
    x2=(x*math.cos(teta) - y*math.sin(teta))
    y2=(y*math.cos(teta) + x*math.sin(teta))
    z2=z
    #transformacion en Rx
    x3= x2
    y3= y2*math.cos(beta) - z2*math.sin(beta)
    z3= z2*math.cos(beta) + y2*math.sin(beta)


    #print(x,y,z,"Reales")
    x_screen=int(  -x3*math.cos(gama) + y3*math.cos(gama))
    y_screen=int( -x3*math.sin(gama) -y3*math.sin(gama) +z3)

    x_screen=(x_screen+400)
    y_screen=(abs(y_screen-400))

    return (x_screen,y_screen)

def cd_eslabon2(xf,yf,zf,L1,L2,T1,T2):
     x=L2*math.cos(T1)*math.cos(T2 + pi/2) - yf*math.sin(T1) + xf*math.cos(T1
)*math.cos(T2 + pi/2) - zf*math.cos(T1)*math.sin(T2 + pi/2)
     y=yf*math.cos(T1) + L2*math.cos(T2 + pi/2)*math.sin(T1) + xf*math.cos(T2
 + pi/2)*math.sin(T1) - zf*math.sin(T1)*math.sin(T2 + pi/2)
     z=L1 - L2*math.sin(T2 - pi/2) + zf*math.cos(T2 - pi/2) - xf*math.sin(T2
 - pi/2)
     return (x,y,z);
def cd_eslabon3(xf,yf,zf,L1,L2,L3,T1,T2,T3):
     x=L3*(math.cos(T1)*math.cos(T3)*math.cos(T2 + pi/2) - cos(T1)*math.sin(
T3)*math.sin(T2 + pi/2)) + xf*(math.cos(T1)*math.cos(T3)*math.cos(T2 + pi/2
) - cos(T1)*math.sin(T3)*math.sin(T2 + pi/2)) - zf*(math.cos(T1)*math.cos(T3
)*math.sin(T2 + pi/2) + cos(T1)*math.cos(T2 + pi/2)*math.sin(T3)) - yf*math.
sin(T1) + L2*math.cos(T1)*math.cos(T2 + pi/2)
     y=L3*(math.cos(T3)*math.cos(T2 + pi/2)*math.sin(T1) - sin(T1)*math.sin(
T3)*math.sin(T2 + pi/2)) + yf*math.cos(T1) + xf*(math.cos(T3)*math.cos(T2 +
pi/2)*math.sin(T1) - sin(T1)*math.sin(T3)*math.sin(T2 + pi/2)) - zf*(math.cos
(T3)*math.sin(T1)*math.sin(T2 + pi/2) + cos(T2 + pi/2)*math.sin(T1)*math.sin(
T3)) + L2*math.cos(T2 + pi/2)*math.sin(T1)
     z=L1 - xf*(math.cos(T3)*math.sin(T2 - pi/2) - cos(T2 - pi/2)*math.sin(T3
)) + zf*(math.cos(T3)*math.cos(T2 - pi/2) + sin(T3)*math.sin(T2 - pi/2)) - L2
*math.sin(T2 - pi/2) - L3*(math.cos(T3)*math.sin(T2 - pi/2) - cos(T2 - pi/2
)*math.sin(T3))
     return (x,y,z)
def cd_eslabon4(xf,yf,zf,L1,L2,L3,L4,T1,T2,T3,T4):
    x=L3*(math.cos(T1)*math.cos(T3)*math.cos(T2 + pi/2) - cos(T1)*math.sin(T3
)*math.sin(T2 + pi/2)) - L4*math.sin(T1) - yf*math.sin(T1) + xf*(math.cos(T4
)*(math.cos(T1)*math.cos(T3)*math.cos(T2 + pi/2) - cos(T1)*math.sin(T3)*math.
sin(T2 + pi/2)) - sin(T4)*(math.cos(T1)*math.cos(T3)*math.sin(T2 + pi/2) +
cos(T1)*math.cos(T2 + pi/2)*math.sin(T3))) - zf*(math.cos(T4)*(math.cos(T1)*
math.cos(T3)*math.sin(T2 + pi/2) + cos(T1)*math.cos(T2 + pi/2)*math.sin(T3
)) + sin(T4)*(math.cos(T1)*math.cos(T3)*math.cos(T2 + pi/2) - cos(T1)*math.
sin(T3)*math.sin(T2 + pi/2))) + L2*math.cos(T1)*math.cos(T2 + pi/2)
```

```
    y=xf*(math.cos(T4)*(math.cos(T3)*math.cos(T2 + pi/2)*math.sin(T1) - sin(
T1)*math.sin(T3)*math.sin(T2 + pi/2)) - sin(T4)*(math.cos(T3)*math.sin(T1)*
math.sin(T2 + pi/2) + cos(T2 + pi/2)*math.sin(T1)*math.sin(T3))) - zf*(math.
cos(T4)*(math.cos(T3)*math.sin(T1)*math.sin(T2 + pi/2) + cos(T2 + pi/2)*math.
sin(T1)*math.sin(T3)) + sin(T4)*(math.cos(T3)*math.cos(T2 + pi/2)*math.sin(T1
) - sin(T1)*math.sin(T3)*math.sin(T2 + pi/2))) + L4*math.cos(T1) + L3*(math.
cos(T3)*math.cos(T2 + pi/2)*math.sin(T1) - sin(T1)*math.sin(T3)*math.sin(T2
 + pi/2)) + yf*math.cos(T1) + L2*math.cos(T2 + pi/2)*math.sin(T1)
    z=L1 - L2*math.sin(T2 - pi/2) - L3*(math.cos(T3)*math.sin(T2 - pi/2) -
cos(T2 - pi/2)*math.sin(T3)) - xf*(math.cos(T4)*(math.cos(T3)*math.sin(T2 -
pi/2) - cos(T2 - pi/2)*math.sin(T3)) - sin(T4)*(math.cos(T3)*math.cos(T2 - pi
/2) + sin(T3)*math.sin(T2 - pi/2))) + zf*(math.cos(T4)*(math.cos(T3)*math.cos
(T2 - pi/2) + sin(T3)*math.sin(T2 - pi/2)) + sin(T4)*(math.cos(T3)*math.sin(
T2 - pi/2) - cos(T2 - pi/2)*math.sin(T3)))
    return (x,y,z)
def cd_eslabon5(xf,yf,zf,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5):
    x =yf*(math.sin(T5)*(math.cos(T4)*(math.cos(T1)*math.cos(T3)*math.sin(T2
 + pi/2) + cos(T1)*math.cos(T2 + pi/2)*math.sin(T3)) + sin(T4)*(math.cos(T1)*
math.cos(T3)*math.cos(T2 + pi/2) - cos(T1)*math.sin(T3)*math.sin(T2 + pi/2
))) - cos(T5)*math.sin(T1)) - zf*(math.cos(T5)*(math.cos(T4)*(math.cos(T1)*
math.cos(T3)*math.sin(T2 + pi/2) + cos(T1)*math.cos(T2 + pi/2)*math.sin(T3
)) + sin(T4)*(math.cos(T1)*math.cos(T3)*math.cos(T2 + pi/2) - cos(T1)*math.
sin(T3)*math.sin(T2 + pi/2))) + sin(T1)*math.sin(T5)) + L3*(math.cos(T1)*math
.cos(T3)*math.cos(T2 + pi/2) - cos(T1)*math.sin(T3)*math.sin(T2 + pi/2)) - L4
*math.sin(T1) + L5*(math.cos(T4)*(math.cos(T1)*math.cos(T3)*math.cos(T2 + pi/
2) - cos(T1)*math.sin(T3)*math.sin(T2 + pi/2)) - sin(T4)*(math.cos(T1)*math.
cos(T3)*math.sin(T2 + pi/2) + cos(T1)*math.cos(T2 + pi/2)*math.sin(T3))) + xf
*(math.cos(T4)*(math.cos(T1)*math.cos(T3)*math.cos(T2 + pi/2) - cos(T1)*math.
sin(T3)*math.sin(T2 + pi/2)) - sin(T4)*(math.cos(T1)*math.cos(T3)*math.sin(T2
 + pi/2) + cos(T1)*math.cos(T2 + pi/2)*math.sin(T3))) + L2*math.cos(T1)*math.
cos(T2 + pi/2)
    y =xf*(math.cos(T4)*(math.cos(T3)*math.cos(T2 + pi/2)*math.sin(T1) - sin(
T1)*math.sin(T3)*math.sin(T2 + pi/2)) - sin(T4)*(math.cos(T3)*math.sin(T1)*
math.sin(T2 + pi/2) + cos(T2 + pi/2)*math.sin(T1)*math.sin(T3))) + yf*(math.
sin(T5)*(math.cos(T4)*(math.cos(T3)*math.sin(T1)*math.sin(T2 + pi/2) + cos(T2
 + pi/2)*math.sin(T1)*math.sin(T3)) + sin(T4)*(math.cos(T3)*math.cos(T2 + pi/
2)*math.sin(T1) - sin(T1)*math.sin(T3)*math.sin(T2 + pi/2))) + cos(T1)*math.
cos(T5)) - zf*(math.cos(T5)*(math.cos(T4)*(math.cos(T3)*math.sin(T1)*math.sin
(T2 + pi/2) + cos(T2 + pi/2)*math.sin(T1)*math.sin(T3)) + sin(T4)*(math.cos(
T3)*math.cos(T2 + pi/2)*math.sin(T1) - sin(T1)*math.sin(T3)*math.sin(T2 + pi/
2))) - cos(T1)*math.sin(T5)) + L4*math.cos(T1) + L3*(math.cos(T3)*math.cos(T2
 + pi/2)*math.sin(T1) - sin(T1)*math.sin(T3)*math.sin(T2 + pi/2)) + L5*(math.
cos(T4)*(math.cos(T3)*math.cos(T2 + pi/2)*math.sin(T1) - sin(T1)*math.sin(T3
)*math.sin(T2 + pi/2)) - sin(T4)*(math.cos(T3)*math.sin(T1)*math.sin(T2 + pi/
2) + cos(T2 + pi/2)*math.sin(T1)*math.sin(T3))) + L2*math.cos(T2 + pi/2)*math
.sin(T1)
    z =L1 - L2*math.sin(T2 - pi/2) - L5*(math.cos(T4)*(math.cos(T3)*math.sin(
T2 - pi/2) - cos(T2 - pi/2)*math.sin(T3)) - sin(T4)*(math.cos(T3)*math.cos(T2
 - pi/2) + sin(T3)*math.sin(T2 - pi/2))) - L3*(math.cos(T3)*math.sin(T2 - pi/
2) - cos(T2 - pi/2)*math.sin(T3)) - xf*(math.cos(T4)*(math.cos(T3)*math.sin(
T2 - pi/2) - cos(T2 - pi/2)*math.sin(T3)) - sin(T4)*(math.cos(T3)*math.cos(T2
 - pi/2) + sin(T3)*math.sin(T2 - pi/2))) + zf*math.cos(T5)*(math.cos(T4)*(
```

```
math.cos(T3)*math.cos(T2 - pi/2) + sin(T3)*math.sin(T2 - pi/2)) + sin(T4)*(
math.cos(T3)*math.sin(T2 - pi/2) - cos(T2 - pi/2)*math.sin(T3))) - yf*math.
sin(T5)*(math.cos(T4)*(math.cos(T3)*math.cos(T2 - pi/2) + sin(T3)*math.sin(T2
 - pi/2)) + sin(T4)*(math.cos(T3)*math.sin(T2 - pi/2) - cos(T2 - pi/2)*math.
sin(T3)))
    return (x,y,z)



def cd_2 (T1,T2,L1,L2):
    x=L2*math.cos(T2)*math.cos(T1)
    y=L2*math.cos(T2)*math.sin(T1)
    z=L1+L2*math.sin(T2)
    return (x,y,z)
def cd_3 (T1,T2,T3,L1,L2,L3):
    x=(L2*math.cos(T2) +L3*math.cos(T2+T3))*math.cos(T1)
    y=(L2*math.cos(T2) +L3*math.cos(T2+T3))*math.sin(T1)
    z=L1+L2*math.sin(T2) +L3*math.sin(T2+T3)
    return (x,y,z)
def cd_4 (T1,T2,T3,T4,L1,L2,L3,L4):
    x=(L2*math.cos(T2)+L3*math.cos(T2+T3)+L4*math.cos(T2+T3+T4))*math.cos(T1)
    y=(L2*math.cos(T2)+L3*math.cos(T2+T3)+L4*math.cos(T2+T3+T4))*math.sin(T1)
    z=L1+L2*math.sin(T2) +L3*math.sin(T2+T3)+L4*math.sin(T2+T3+T4)
    return (x,y,z)
def cd_5 (T1,T2,T3,T4,T5,L1,L2,L3,L4,L5):
    x=(L2*math.cos(T2)+L3*math.cos(T2+T3)+L4*math.cos(T2+T3+T4)+L5*math.cos(
T2+T3+T4+T5)) *math.cos(T1)
    y=(L2*math.cos(T2)+L3*math.cos(T2+T3)+L4*math.cos(T2+T3+T4)+L5*math.cos(
T2+T3+T4+T5)) *math.sin(T1)
    z=L1+L2*math.sin(T2)+L3*math.sin(T2+T3)+L4*math.sin(T2+T3+T4)+L5*math.sin
(T2+T3+T4+T5)
    return (x,y,z)
#Librería
def robot_esqueleto(screen,gama,teta,beta,escala,T1,T2,T3,T4,T5):
    #Longitud de los eslabones
    L1=40
    L2=30
    L3=30
    L4=5
    L5=10
    #Puntos de control
    A=(0,0,0)
    B=(0,0,L1)
    C=cd_2(T1,T2,L1,L2)
    D=cd_3(T1,T2,T3,L1,L2,L3)
    E=cd_4(T1,T2,T3,T4,L1,L2,L3,L4)
    F=cd_5(T1,T2,T3,T4,T5,L1,L2,L3,L4,L5)
    #Convertimos R3 a R2 Virtual
    A2=pixel(A[0],A[1],A[2],gama,teta,beta,escala)
    B2=pixel(B[0],B[1],B[2],gama,teta,beta,escala)
    C2=pixel(C[0],C[1],C[2],gama,teta,beta,escala)
```

```
    G2= pixel(G[0],G[1],G[2],gama,teta,beta,escala)
    H2= pixel(H[0],H[1],H[2],gama,teta,beta,escala)

    I2= pixel(I[0],I[1],I[2],gama,teta,beta,escala)
    J2= pixel(J[0],J[1],J[2],gama,teta,beta,escala)


    O2= pixel(O[0],O[1],O[2],gama,teta,beta,escala)
    P2= pixel(P[0],P[1],P[2],gama,teta,beta,escala)

    A2X= pixel(AX[0],AX[1],AX[2],gama,teta,beta,escala)
    B2X= pixel(BX[0],BX[1],BX[2],gama,teta,beta,escala)
    C2X= pixel(CX[0],CX[1],CX[2],gama,teta,beta,escala)
    D2X= pixel(DX[0],DX[1],DX[2],gama,teta,beta,escala)

    #Los dibujamos
    #pygame.draw.circle(screen, RED2, (A2) , 3)
    #pygame.draw.circle(screen, RED2, (B2) , 3)
    #pygame.draw.circle(screen, BLUE, (C2) , 3)
    #pygame.draw.circle(screen, BLUE, (D2) , 3)

    #pygame.draw.circle(screen, RED2, (A2X) , 3)
    #pygame.draw.circle(screen, RED2, (H2) , 3)
    #pygame.draw.circle(screen, BLUE, (J2) , 3)
    #pygame.draw.circle(screen, BLUE, (E2) , 3)

    #dibujamos poligonos
    pygame.draw.polygon(screen,YELLOW,(A2,B2,C2,D2))
    pygame.draw.polygon(screen,YELLOW,(A2X,B2X,C2X,D2X))

    pygame.draw.polygon(screen,YELLOW2,(A2,B2,B2X,A2X))
    pygame.draw.polygon(screen,YELLOW2,(B2,C2,C2X,B2X))

    pygame.draw.polygon(screen,YELLOW2,(C2,D2,D2X,C2X))
    pygame.draw.polygon(screen,YELLOW,(D2,A2,A2X,D2X))

    #DIBUJAMOS UNION ESFÉRICA
    pygame.draw.circle(screen, BLACK, (O2) , 10)
    #DIBUJAMOS UNION ESFÉRICA FINAL
   # pygame.draw.circle(screen, BLACK, (P2) , 12)

    pygame.draw.polygon(screen,RED,(E2,F2,G2,H2))
    pygame.draw.polygon(screen,RED2,(H2,J2,E2))
    pygame.draw.polygon(screen,RED2,(F2,G2,I2))


 #Calculamos Cinematica Inversa
            #----- T1
#Grafica EL espacio Alcanzable
def Espacio_Alcanzable(screen,gama,teta,beta,escala):
    for i in range (5):
        for j in range(5):
```

```
                for k in range (5):
                    for l in range(5):
                        for m in range(5):
                          T1=-90+i*40;T1=T1*pi/180;
                          T2=-90+j*40;T2=T2*pi/180;
                          T3=-90+k*40;T3=T3*pi/180;
                          T4=-90+l*40;T4=T4*pi/180;
                          T5=-90+m*40;T5=T5*pi/180;

                          #CD
                          L1=40
                          L2=30
                          L3=30
                          L4=5
                          L5=10
                          x=(L2*cos(T2+pi/2)+L3*cos(T2+pi/2+T3)+L4*cos(T2+pi/2+T3
+T4)+L5*cos(T2+pi/2+T3+T4+T5)) *cos(T1);
                          y=(L2*cos(T2+pi/2)+L3*cos(T2+pi/2+T3)+L4*cos(T2+pi/2+T3
+T4)+L5*cos(T2+pi/2+T3+T4+T5)) *sin(T1);
                          z=L1+L2*sin(T2+pi/2)+L3*sin(T2+pi/2+T3)+L4*sin(T2+pi/2+
T3+T4)+L5*sin(T2+pi/2+T3+T4+T5);
                          #Virtualizamos el punto
                          #print(x,y,z)
                          P= pixel(x,y,z,gama,teta,beta,escala)
                          #Lo dibujamos
                          pygame.draw.circle(screen,GREEN2, (P) , 2)

def Cinematica_Inversa(x,y,z):
    L1=40
    L2=30
    L3=30
    L4=5
    L5=10
#1)Calculamos T1
    #Preguntamos por cuadrante
    if (x >= 0) & (y >= 0):
        #Octante 1,+x+y
        T1=math.acos(x/math.sqrt(x*x + y*y))
    if (x <= 0) & (y <= 0):
        #Octante 3,-x-y
        T1=math.acos(abs(y)/math.sqrt(x*x + y*y))
    if (x <= 0) & (y >= 0):
        #Octante 2,-x+y
        T1=-pi/2 +math.acos(y/math.sqrt(x*x + y*y))
    if (x >= 0) & (y <= 0):
        #Octante 2,+x-y
        T1=-pi/2 +math.acos(x/math.sqrt(x*x + y*y))
#2)Calculamos a T2
    for i in range(181):
        T2=-90 +i
        T2=T2*pi/180
        xf=(L2*cos(T2+pi/2)) *cos(T1);
```

```
        yf=(L2*cos(T2+pi/2)) *sin(T1);
        zf=L1+L2*sin(T2+pi/2) ;
        P=[xf,yf,zf]
        d1=math.sqrt(xf*xf +yf*yf +(zf-L1)*(zf-L1))
        Q=[x,y,z]
        d2=math.sqrt((xf-x)*(xf-x) + (yf-y)*(yf-y) + (zf-z)*(zf-z))
        if abs(d1-d2) <.5:
            break;
#3)Calculamos a T3
    for j in range(181):
        T3=-90 +j
        T3=T3*pi/180
        xf=(L2*cos(T2+pi/2) +L3*cos(T2+pi/2+T3)) *cos(T1);
        yf=(L2*cos(T2+pi/2) +L3*cos(T2+pi/2+T3)) *sin(T1);
        zf=L1+L2*sin(T2+pi/2) +L3*sin(T2+pi/2+T3);
        P=[xf,yf,zf]
        d1=math.sqrt(xf*xf +yf*yf +zf*zf)
        Q=[x,y,z]
        d2=math.sqrt((xf-x)*(xf-x) + (yf-y)*(yf-y) + (zf-z)*(zf-z))
        #print(d2-L4,j)
        if abs(d2-L4)<.3:
            break;
#3)Calculamos a T4
    for k in range(181):
        T4=-90 +k
        T4=T4*pi/180
        xf=(L2*cos(T2+pi/2) +L3*cos(T2+pi/2+T3)+L4*cos(T2+pi/2+T3+T4)) *cos(
T1);
        yf=(L2*cos(T2+pi/2) +L3*cos(T2+pi/2+T3)+L4*cos(T2+pi/2+T3+T4)) *sin(
T1);
        zf=L1+L2*sin(T2+pi/2) +L3*sin(T2+pi/2+T3)+L4*sin(T2+pi/2+T3+T4);
        d2=math.sqrt((xf-x)*(xf-x) + (yf-y)*(yf-y) + (zf-z)*(zf-z))
        #print(d2-L4,j)
        if abs(d2)<.3:
            break;
#3)Calculamos a T5
    for m in range(181):
        T5=-90 + m
        T5=T5*pi/180
        xf=(L2*cos(T2+pi/2) +L3*cos(T2+pi/2+T3)+L4*cos(T2+pi/2+T3+T4)+L5*cos(
T2+pi/2+T3+T4+T5)) *cos(T1);
        yf=(L2*cos(T2+pi/2) +L3*cos(T2+pi/2+T3)+L4*cos(T2+pi/2+T3+T4)+L5*cos(
T2+pi/2+T3+T4+T5)) *sin(T1);
        zf=L1+L2*sin(T2+pi/2) +L3*sin(T2+pi/2+T3)+L4*sin(T2+pi/2+T3+T4)+L5*
sin(T2+pi/2+T3+T4+T5);
        d2=math.sqrt((xf-x)*(xf-x) + (yf-y)*(yf-y) + (zf-z)*(zf-z))
        #print(d2-L4,j)
        if abs(d2)<.3:
            break;


    return (T1,T2,T3,T4,T5)


 return (T1,T2,T3,T4,T5,resultado)
```

Python – Principal

```
import cv2
import numpy as np
from Libreria import*
import serial



#arduinoMega = serial.Serial("COM5", 115200)
#time.sleep(2) # Tiempo para el enlace de Arduino

#Barras de control para mover el robot
cv2.namedWindow('Robot_space')
cv2.resizeWindow('Robot_space', 500, 500)
cv2.createTrackbar('T1','Robot_space',0,180,nothing) #-90 0 90
cv2.createTrackbar('T2','Robot_space',0,180,nothing)
cv2.createTrackbar('T3','Robot_space',0,180,nothing)
cv2.createTrackbar('T4','Robot_space',0,180,nothing)
cv2.createTrackbar('T5','Robot_space',0,180,nothing)
cv2.createTrackbar('Gripper','Robot_space',0,180,nothing)
cv2.createTrackbar('Comunication','Robot_space',0,1,nothing)# Se ocupa para
enviar informacion al robot
cv2.createTrackbar('Print','Robot_space',0,1,nothing) # Imprimir en pantalla
el punto en el espacio X,Y,Z
cv2.createTrackbar('Espacio_Alcanzable','Robot_space',0,1,nothing)

#Barras para Cinemática Inversa
cv2.namedWindow('Cinematica_Inversa')
cv2.resizeWindow('Cinematica_Inversa', 500, 500)
cv2.createTrackbar('X','Cinematica_Inversa',0,80,nothing) #-90 0 90
cv2.createTrackbar('Y','Cinematica_Inversa',0,80,nothing)
cv2.createTrackbar('Z','Cinematica_Inversa',0,80,nothing)
cv2.createTrackbar('ON-OFF','Cinematica_Inversa',0,1,nothing)#Imprimir en
pantalla el punto en el espacio X,Y,Z
# VARIABLES DE MEMORIA PARA EVITAR EL RECÁLCULO

m_direc_T1=0;
m_direc_T2=0;
m_direc_T3=0;
m_direc_T4=0;
m_direc_T5=0;
m_direc_T5=0;

m_inv_x=0;
m_inv_y=0;
m_inv_z=0;

Especial_Token=0;
T1=0;
T2=0;
T3=0;
T4=0;
T5=0;
```

```
T6=0;


pygame.init()  #iniciamos pygame
pygame.mixer.init()  #Herramienta de audio

       #x,y
size =(700,700) #definimos Tamaño de Ventana
clock= pygame.time.Clock() #Creamos el Tiempo del videojuego
screen=pygame.display.set_mode(size)
pygame.display.set_caption("Robot Space")
while True:
    for event in pygame.event.get():    #AQUI TIENE LUGAR EL Motor Gráfico
        if event.type==pygame.QUIT:     #CONDICION QUE PERMITE CERRAR EL
PROGRAMA
            sys.exit()
#-------#Eventos del Teclado-----------------------------------
        if event.type==pygame.KEYDOWN:
            if event.key==pygame.K_LEFT:
                inc=+0.01
            if event.key==pygame.K_RIGHT:
                inc=-0.01
            if event.key == pygame.K_UP:
                inc2=+0.01
            if event.key==pygame.K_DOWN:
                inc2=-0.01
            #AL DEJAR DE PRECIONAR
        if event.type==pygame.KEYUP:
            if event.key==pygame.K_LEFT:
                inc=0
            if event.key==pygame.K_RIGHT:
                inc=0
            if event.key==pygame.K_UP:
                inc2=0
            if event.key==pygame.K_DOWN:
                inc2=0
    teta+=inc; #rotaciones sobre el eje Z
    beta+=inc2; #rotacion sobre el eje X
    screen.fill(BLACK2) #Fondo del dibujo
#-------------------------------------------------
#    dibujamos los ejes de referencia
        #Puntos iniciales que definen los ejes en el espacio R3 virtual R2
    x_nega=pixel(-300,0,0,gama,teta,beta,1);x_posi=pixel(300,0,0,gama,teta,
beta,1);y_nega=pixel(0,-300,0,gama,teta,beta,1)
    y_posi=pixel(0,300,0,gama,teta,beta,1);z_nega=pixel(0,0,-300,gama,teta,
beta,1);z_posi=pixel(0,0,300,gama,teta,beta,1)
                                #x,y1    x,y2
    pygame.draw.line(screen,GREEN,x_nega,x_posi) # eje horizontal X
    pygame.draw.line(screen,BLUE,y_nega,y_posi)  # eje vertical   Y
    pygame.draw.line(screen,RED,z_nega,z_posi)  # eje vertical   Z
#-------------Aqui estamos ya Dentro del entorno virtual creado
-----------------
```

```
        #obtenemos lecturas de las barras de control
        Q1 = cv2.getTrackbarPos('T1','Robot_space')
        Q2 = cv2.getTrackbarPos('T2','Robot_space')
        Q3 = cv2.getTrackbarPos('T3','Robot_space')
        Q4 = cv2.getTrackbarPos('T4','Robot_space')
        Q5 = cv2.getTrackbarPos('T5','Robot_space')
        Q6 = cv2.getTrackbarPos('Gripper','Robot_space')
        Q7 = cv2.getTrackbarPos('Comunication','Robot_space')
        Q8 = cv2.getTrackbarPos('Print','Robot_space')
        Q9 = cv2.getTrackbarPos('Espacio_Alcanzable','Robot_space')

        #Evaluamos cinemática Inversa del Robot
        dinamica=cv2.getTrackbarPos('ON-OFF','Cinematica_Inversa')
        Xd = cv2.getTrackbarPos('X','Cinematica_Inversa')
        Yd = cv2.getTrackbarPos('Y','Cinematica_Inversa')
        Zd = cv2.getTrackbarPos('Z','Cinematica_Inversa')
    #---------------CONTROL MANUAL DEL ROBOT-------------------------
        #Lo enviamos de ser necesario por el puerto Serial
        if ((Q7>0) & (dinamica==0)): #Comunication
            #A+00B+00
            valor1=da_formato(Q1-90)
            valor2=da_formato(Q2-90)
            valor3=da_formato(Q3-90)
            valor4=da_formato(Q4-90)
            valor5=da_formato(Q5-90)
            valor6=da_formato(Q6-90)
            clave='A'+ valor1 +'B'+ valor2 +'C'+ valor3 +'D'+ valor4 +'E'+valor5+
    'F'+str(valor6)+'\n'
            #print(clave)
            #arduinoMega.write (clave.encode())
            #Las convertimos a radianes
            T1=(Q1-90)*pi/180
            T2=(Q2-90)*pi/180
            T3=(Q3-90)*pi/180
            T4=(Q4-90)*pi/180
            T5=(Q5-90)*pi/180
            #dimensiones del robot Shoubi_V3
            L1=40
            L2=30
            L3=30
            L4=5
            L5=10
            escala=4
            #Dibujamos al robot
            eslabon_1(screen,gama,teta,beta,escala)
            eslabon_2(screen,gama,teta,beta,escala,T1+pi/2)
            eslabon_T2(screen,gama,teta,beta,escala,T1,T2)
            eslabon_T3(screen,gama,teta,beta,escala,T1,T2,T3)
            eslabon_T4(screen,gama,teta,beta,escala,T1,T2,T3,T4)
            eslabon_T5(screen,gama,teta,beta,escala,T1,T2,T3,T4,T5+pi)
            robot_esqueleto(screen,gama,teta,beta,escala,T1,T2+pi/2,T3,T4,T5+pi)
            if Q9>0:
```

```
            Espacio_Alcanzable(screen,gama,teta,beta,escala)

        #Evaluamos cinemática directa del robot
        if Q8>0:
            if((T1-m_direc_T1 ==0)&(T2-m_direc_T2 ==0)&(T3-m_direc_T3 ==0)&(
T4-m_direc_T4==0)&(T5-m_direc_T5==0)):
                pass
            else :
                m_direc_T1=T1;m_direc_T2=T2;m_direc_T3=T3;m_direc_T4=T4;
m_direc_T5=T5;
                T2=T2+pi/2
                xf=(L2*math.cos(T2) +L3*math.cos(T2+T3)+L4*math.cos(T2+T3+T4
)+L5*math.cos(T2+T3+T4+T5)) *math.cos(T1)
                yf=(L2*math.cos(T2) +L3*math.cos(T2+T3)+L4*math.cos(T2+T3+T4
)+L5*math.cos(T2+T3+T4+T5)) *math.sin(T1)
                zf=L1+L2*math.sin(T2) +L3*math.sin(T2+T3)+L4*math.sin(T2+T3+
T4)+L5*math.sin(T2+T3+T4+T5)
                print("X= ",xf,"Y= ",yf,"Z= ",zf)

#-----------------TERMINA CONTROL MANUAL Y COMIENZA CONTROL POR CINEMÁTICA
INVERSA
    if dinamica>0:
        L1=40
        L2=30
        L3=30
        L4=5
        L5=10
        escala=3
        #Dibujamos al robot
        eslabon_1(screen,gama,teta,beta,escala)
        eslabon_2(screen,gama,teta,beta,escala,T1+pi/2)
        eslabon_T2(screen,gama,teta,beta,escala,T1,T2)
        eslabon_T3(screen,gama,teta,beta,escala,T1,T2,T3)
        eslabon_T4(screen,gama,teta,beta,escala,T1,T2,T3,T4)
        eslabon_T5(screen,gama,teta,beta,escala,T1,T2,T3,T4,T5+pi)
        robot_esqueleto(screen,gama,teta,beta,escala,T1,T2+pi/2,T3,T4,T5+pi)

        if((abs(Xd-m_inv_x) <.5)&(abs(Yd-m_inv_y) <.5)&(abs(Zd-m_inv_z) <.5
)):
            pass
        else:
            fi1,fi2,fi3,fi4,fi5 = Cinematica_Inversa(Xd,Yd,Zd)
            T1,T2,T3,T4,T5 = fi1,fi2,fi3,fi4,fi5
            #print(T1*180/pi,T2*180/pi,T3*180/pi,T4*180/pi,T5*180/pi)
            #dimensiones del robot
            #Actualizamos la memoria
            m_inv_x=Xd;
            m_inv_y=Yd;
            m_inv_z=Zd;
            #Enviamos por el puerto Serial
            valor1=da_formato(T1*180/pi)
            valor2=da_formato(T2*180/pi)

            valor3=da_formato(T3*180/pi)
            valor4=da_formato(T4*180/pi)
            valor5=da_formato(T5*180/pi)
            valor6=Q6
            clave='A' + valor1 + 'B'+ valor2 + 'C'+ valor3 + 'D' + valor4 + '
E' +valor5+ 'F' + str(valor6)+'\n'
            print(clave)
           # arduinoMega.write (clave.encode())


#-----------------------------------------------------------------
    #actualizar Pantalla
    pygame.display.flip()
    clock.tick(48)
#arduinoMega.closed()
```

Python – Detección de Color HSV

```
import cv2
import numpy as np

def nothing(x):
    pass

# cargar imagen
#image = cv2.imread('nombredelaimagen.jpg')
# usamos videocaptura de la camara
cap = cv2.VideoCapture(0) #numero 0 por la ubicacion en la computadora

# Creamos una ventana con nombre resized
cv2.namedWindow('resized')

# Creamos los trackbar para cada valor HSV
# Hue/Matiz tiene como rango 0-179 en Opencv
# Saturation/Saturacion tiene como rango 0-255 en Opencv
# Value/Valor tiene como rango 0-255 en Opencv
cv2.createTrackbar('HMin', 'resized', 0, 179, nothing)
cv2.createTrackbar('SMin', 'resized', 0, 255, nothing)
cv2.createTrackbar('VMin', 'resized', 0, 255, nothing)
cv2.createTrackbar('HMax', 'resized', 0, 179, nothing)
cv2.createTrackbar('SMax', 'resized', 0, 255, nothing)
cv2.createTrackbar('VMax', 'resized', 0, 255, nothing)

# Establecemos el valor por defecto de los trackbar en HSVmax
cv2.setTrackbarPos('HMax', 'resized', 179)
cv2.setTrackbarPos('SMax', 'resized', 255)
cv2.setTrackbarPos('VMax', 'resized', 255)

# Iniciamos los valores max/min en HSV
hMin = sMin = vMin = hMax = sMax = vMax = 0
phMin = psMin = pvMin = phMax = psMax = pvMax = 0

while(1):
    #dimension de resolucion para el frame
    dim = (640,480)
    ret, frame = cap.read()
    resized = cv2.resize(frame, dim, interpolation = cv2.INTER_AREA)
    # Obtenermos las posiciones actuales de todos los trackbar
    hMin = cv2.getTrackbarPos('HMin', 'resized')
    sMin = cv2.getTrackbarPos('SMin', 'resized')
    vMin = cv2.getTrackbarPos('VMin', 'resized')
    hMax = cv2.getTrackbarPos('HMax', 'resized')
    sMax = cv2.getTrackbarPos('SMax', 'resized')
    vMax = cv2.getTrackbarPos('VMax', 'resized')

    # Establecemos los valores HSV min/max a mostrar
    lower = np.array([hMin, sMin, vMin])
    upper = np.array([hMax, sMax, vMax])
```

```
    # Convertimos el espacio de color RGB a HSV
    # Convertimos el umbral de color/threshold
    hsv = cv2.cvtColor(resized, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, lower, upper)
    result = cv2.bitwise_and(resized, resized, mask=mask)

    # Imprimimos si hay un cambio en los valores HSV
    if((phMin != hMin) | (psMin != sMin) | (pvMin != vMin) | (phMax != hMax)
            | (psMax != sMax) | (pvMax != vMax) ):
        print("(hMin = %d , sMin = %d, vMin = %d), (hMax = %d , sMax = %d,
vMax = %d)"
              % (hMin , sMin , vMin, hMax, sMax , vMax))
        phMin = hMin
        psMin = sMin
        pvMin = vMin
        phMax = hMax
        psMax = sMax
        pvMax = vMax

    # Mostramos en una ventana el resultado del color encontrado
    # en una imagen
    cv2.imshow('PLANO IMAGEN', result)
    #indicamos el cierre del programa con la letra "q"
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

cv2.destroyAllWindows()
```

Python - Contornos

```
import imutils
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while True:
    dim = (640, 480)
    ret, frame = cap.read()
    #redimensionamos
    resized = cv2.resize(frame, dim, interpolation=cv2.INTER_LINEAR)
    if ret == True:
        frameHSV = cv2.cvtColor(resized, cv2.COLOR_BGR2HSV)
        # rango de colores HSV

        redBajo1 = np.array([0, 100, 20], np.uint8)
        redAlto1 = np.array([8, 255, 255], np.uint8)
        redBajo2 = np.array([175, 100, 20], np.uint8)
        redAlto2 = np.array([179, 255, 255], np.uint8)

        maskRed1 = cv2.inRange(frameHSV, redBajo1, redAlto1)
        maskRed2 = cv2.inRange(frameHSV, redBajo2, redAlto2)
        Red_mask = cv2.add(maskRed1, maskRed2)

        low_green = np.array([54, 68, 33], np.uint8)
        high_green = np.array([89, 255, 255], np.uint8)
        green_mask = cv2.inRange(frameHSV, low_green, high_green)

        low_blue = np.array([70, 118, 65], np.uint8)
        high_blue = np.array([120, 255, 255], np.uint8)
        blue_mask = cv2.inRange(frameHSV, low_blue, high_blue)

        low_yellow = np.array([17, 127, 109], np.uint8)
        high_yellow = np.array([41, 255, 255], np.uint8)
        yellow_mask = cv2.inRange(frameHSV, low_yellow, high_yellow)
        # aplicamos los momentos para cada mascara de color
        M = cv2.moments(Red_mask)
        # encontramos contornos en la mascara, aplicamos la jerarquia de
contornos
        # y realizamos un metodo de aproximacion
        cnts = cv2.findContours(Red_mask, cv2.RETR_TREE, cv2.
CHAIN_APPROX_SIMPLE)
        # cv2.drawContours(resized,cnts,-1,(0,0,255),2)
        #tomamos los contornos usando imutils
        cnts = imutils.grab_contours(cnts)
        # si tenemos algun contorno
        if len(cnts) > 0:
            # buscamos el que tenga mas area
            c = max(cnts, key=cv2.contourArea)
            # ubicamos el circulo que mas se aproxime, x,y seran las
coordenadas de la deteccion
```

```
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            # dibujamos un circulo con relleno del centro sobre la imagen
redimensionada
            cv2.circle(resized, (int(x), int(y)), 3, (0, 0, 255), -1)
        # obtenemos el centro del color y lo imprimos
        centro_rojo = (int(x), int(y))
        print(f"el centro rojo es: {centro_rojo}")

        M = cv2.moments(green_mask)
        cnts = cv2.findContours(green_mask, cv2.RETR_TREE, cv2.
CHAIN_APPROX_SIMPLE)
        # cv2.drawContours(resized,cnts,-1,(0,0,255),2)
        cnts = imutils.grab_contours(cnts)
        if len(cnts) > 0:
            c = max(cnts, key=cv2.contourArea)
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            cv2.circle(resized, (int(x), int(y)), 3, (0, 0, 255), -1)
        centro_verde = (int(x), int(y))
        print(f"el centro verde es: {centro_verde}")

        M = cv2.moments(blue_mask)
        cnts = cv2.findContours(blue_mask, cv2.RETR_TREE, cv2.
CHAIN_APPROX_SIMPLE)
        # cv2.drawContours(resized,cnts,-1,(0,0,255),2)
        cnts = imutils.grab_contours(cnts)
        if len(cnts) > 0:
            c = max(cnts, key=cv2.contourArea)
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            cv2.circle(resized, (int(x), int(y)), 3, (0, 0, 255), -1)
        centro_azul = (int(x), int(y))
        print(f"el centro azul es: {centro_azul}")

        M = cv2.moments(yellow_mask)
        cnts = cv2.findContours(yellow_mask, cv2.RETR_TREE, cv2.
CHAIN_APPROX_SIMPLE)
        # cv2.drawContours(resized,cnts,-1,(0,0,255),2)
        cnts = imutils.grab_contours(cnts)
        if len(cnts) > 0:
            c = max(cnts, key=cv2.contourArea)
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            cv2.circle(resized, (int(x), int(y)), 3, (0, 0, 255), -1)
        centro_amarillo = (int(x), int(y))
        print(f"el centro amarillo es: {centro_amarillo}")

        pts1 = np.float32([[centro_rojo], [centro_azul], [centro_verde], [
centro_amarillo]])
        pts2 = np.float32([[0, 0], [580, 0], [0, 415], [580, 415]])
        matrix = cv2.getPerspectiveTransform(pts1, pts2)
        result = cv2.warpPerspective(frame, matrix, (580, 415))

    #mostramos los puntos centros sobre la imagen redimensionada
    cv2.imshow('CONTORNOS', result)
    key = cv2.waitKey(1)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Python - Recorte de Imagen

```
import cv2
import glob
import os

def click_and_crop(event, x, y, flags, param):

    global refPt, cropping

    if event == cv2.EVENT_LBUTTONDOWN:
        refPt = [(x, y)]
        cropping = True
    elif event == cv2.EVENT_LBUTTONUP:

        refPt.append((x, y))
        cropping = False
        cv2.rectangle(image, refPt[0], refPt[1], (0, 255, 0), 2)
        cv2.imshow("image", image)



path = os.getcwd()
print(path)
# carpeta origen
originalImg = "\Dataset"
# carpeta recortada
cropImg  = "\crop"

filenames = glob.glob(path+originalImg+"\*.jpg")

image, clone = None,None

numImg = 1

for filename in filenames:

    refPt = []
    cropping = False
    cv2.namedWindow("image")
    cv2.setMouseCallback("image", click_and_crop)

    image = cv2.imread(filename)
    clone = image.copy()

    while True:

            cv2.imshow("image", image)
            key = cv2.waitKey(1) & 0xFF

            if key == ord("r"):
                    image = clone.copy()
                    refPt = []


        elif key == ord("c"):
                break
if len(refPt) == 2:
    roi = clone[refPt[0][1]:refPt[1][1], refPt[0][0]:refPt[1][0]]
    cv2.imshow("ROI", roi)
    cv2.imwrite(os.getcwd()+cropImg+"\crop("+str(numImg)+").jpg",roi)
else:
    cv2.imwrite(os.getcwd()+cropImg+"\crop("+str(numImg)+").jpg",image)

numImg = numImg+1

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Python - Obtención de Coordenadas

```
import cv2
import numpy as np

pulsador = cv2.CascadeClassifier('pulsador01.xml')
destornillador = cv2.CascadeClassifier('destorn071.xml')

cap = cv2.VideoCapture(0)
dim = (640, 480)
while True:

    ret, frame = cap.read()
    resized = cv2.resize(frame, dim, interpolation = cv2.INTER_AREA)
    # print(frame.shape)
    cv2.circle(frame, (29, 20), 5, (0, 0, 255), -1)  # primero punto
    cv2.circle(frame, (629, 19), 5, (0, 0, 255), -1)  # segundo punto
    cv2.circle(frame, (28, 446), 5, (0, 0, 255), -1) # tercero punto
    cv2.circle(frame, (625, 451), 5, (0, 0, 255), -1) # cuarto punto

    pts1 = np.float32([[29, 20], [629, 19], [28, 446], [625, 451]])
    pts2 = np.float32([[0, 0], [580, 0], [0, 415], [580, 415]])
    # transformacion de perspectiva
    matrix = cv2.getPerspectiveTransform(pts1, pts2)

    result = cv2.warpPerspective(frame, matrix, (580, 415))

    gray = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)
    pulso = pulsador.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=10
, minSize=(40, 40))

    for (x, y, w, h) in pulso:
        cv2.rectangle(result, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(result, 'pulsador', (x, y - 10), 1, 0.7, (0, 0, 255), 1
, cv2.LINE_AA)
        cx = int(x+w/ 2)
        cy = int(y+h/ 2)
        xrobot = int(cx - result.shape[1] / 2)
        yrobot = int(result.shape[0] - cy)
        xd = int(xrobot/5)
        yd = int(yrobot/5)
        #print(xd,yd)
        #print(xrobot, yrobot)
        cv2.circle(result, (cx,cy), 2,(0,255,0),2)
        #cv2.putText(result, "X:"+str(cx)+",Y:"+str(cy),(cx,cy),1,1,(0,0,0),1
)
        #cv2.putText(result, "X:" + str(xrobot) + ",Y:" + str(yrobot), (cx,
cy), 1, 1, (0, 0, 0), 1)
        cv2.putText(result, "X:" + str(xd) + ",Y:" + str(yd), (cx, cy), 1, 1
, (0, 0, 0), 2)

    destorn = destornillador.detectMultiScale(gray, scaleFactor=1.6,
minNeighbors=25, minSize=(92, 96))
```

```
    for (x, y, w, h) in destorn:
        cv2.rectangle(result, (x, y), (x + w, y + h), (0, 0, 255), 2)
        cv2.putText(result, 'destornillador', (x, y - 10), 1, 0.7, (255, 0, 0
), 1, cv2.LINE_AA)
        cx = int(x + w / 2)
        cy = int(y + h / 2)
        xrobot = int(cx - result.shape[1] / 2)
        yrobot = int(result.shape[0] - cy)
        xd = int(xrobot / 5)
        yd = int(yrobot / 5)
        cv2.circle(result, (cx, cy), 2, (0, 0, 255), 2)
        cv2.putText(result, "X:" + str(xd) + ",Y:" + str(yd), (cx, cy), 1, 1
, (0, 0, 0),2)

    cv2.line(result, (290,0), (290, 415), (255,255,0), 3) #Linea de
referencia en los cuadrantes
    # cv2.imshow("normal", frame)
    cv2.imshow("CORRECCION DE COORDENADA", result)
    key = cv2.waitKey(1)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

```
import speech_recognition as sr
import pyttsx3
from datetime import datetime
import subprocess as sub
import coordenadas #contorno en coordenadas
import cinematica1
import cinematica2
import cinematica3
from tkinter import *
from PIL import Image, ImageTk  ### utilizar imagenes dentro de python
import threading as tr

name = "ana" # indicamos nombre cualquiera
mic = sr.Microphone(device_index=2) #ubicacion de mic
# engine = None
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[3].id)
engine.setProperty('rate', 140)

main_window = Tk()
main_window.title("A N A - AVv3")
main_window.geometry("600x500")
main_window.resizable(0,0)
main_window.configure(bg='#0F2027')
# comandos a usar
comandos = """
    Comandos que puedes usar:
    - Inicia Interfaz
    - Pasame destornillador
    - Pasame pulsador
    - Regresa objeto
    - Dime la hora
    - Ciérrate
    - Termina
"""
# detalle alternativo - imagen gif
ana_gif_path = "Images/ana_animate2.gif"
info_gif = Image.open(ana_gif_path)
gif_nframes = info_gif.n_frames
# print("Numero frames: " + str(gif_nframes))
# print(sr.Microphone.list_microphone_names())
label_name = Label(main_window, text="A N A - AV", fg="#1CB5E0", bg="#0F2027"
, font=('Arial', 25, 'bold')).pack(pady=10)
# recorremos cada uno de estos numeros dentro de una lista
ana_gif_list = [PhotoImage(file=ana_gif_path, format=f'gif -index {i}') for
i in range(gif_nframes)]
# crear una lista
label_gif = Label(main_window)
label_gif.pack()  # posicionar al label debajo de lo que tiene arriba
```

```
# funcion animacion para la imagen gif
def animate_gif(index):
    frame = ana_gif_list[index]
    index += 1  # aumento en 1 el valor de index
    if index == gif_nframes:
        index = 0
    label_gif.configure(image=frame)
    main_window.after(80, animate_gif,index)
    # tomar una funcion y ejecutarlo dentro de la interfaz grafica cada ms. (
despues de cada cierto tiempo)
    # recurso de programacion "recusividad" = defines uan funcnion de un
programa dentro ejecutas esa funcion

# start = tk.Label()
# tr.Thread(target=animate_gif(0)).start()
animate_gif(0)

# Canvas en dónde se ubicarán el texto con descripcion de los comandos
canvas = Canvas(bg="#0F2027", height=160, width=200)
canvas.place(x=5, y=330)
canvas.create_text(90, 80, text=comandos, fill='red', font='Arial 8 bold')

def talk(text): #pyttsx3
    engine.say(text)
    engine.runAndWait()
    # engine.endLoop()
    # engine.stop()
    # text_info.delete('1.0', 'end')

def listen(): #speechrecogntion
    listener = sr.Recognizer()
    with mic as source:
        listener.adjust_for_ambient_noise(source)
        talk("Te escucho")
        pc = listener.listen(source)
    try:
        rec = listener.recognize_google(pc, language="es")
        rec = rec.lower()
    except sr.UnknownValueError:
        print("No te entendí, intenta de nuevo")
    except sr.RequestError as e:
        print("Could not request results from Google Speech Recognition
service; {0}".format(e))
    return rec

def cierra(rec):
    if 'ciérrate' in rec:
        talk("hasta otra!")
        sub.call('taskkill /IM pythonw.exe /F', shell=True) # acaba el
programa

def horario(rec): #funcion de prueba da la hora
```

```
    if rec:
        time = datetime.now().strftime('%H:%M')
        talk(f'Son las {time}')

def interfaz(rec):
    if rec:
        talk("Iniciando Interfaz Visual!")
        t = tr.Thread(target=coordenadas.captura())
        t.start()
    #time.sleep(5)

def destorn(rec):
    if rec:
        talk("Obteniendo destornillador!")
        t = tr.Thread(target=cinematica1.principal())
        t.start()
    #time.sleep(5)


def pulsador(rec):
    if rec:
        talk("Obteniendo pulsador!")
        t = tr.Thread(target=cinematica2.principal())
        t.start()
    #time.sleep(5)


def regreso(rec):
    if rec:
        talk("Devolviendo objeto!")
        t = tr.Thread(target=cinematica3.principal())
        t.start()
    # time.sleep(5)



key_words = {
    'ciérrate': cierra,
    'inicia interfaz': interfaz,
    'pasame destornillador': destorn,
    'pasame pulsador': pulsador,
    'regresa objeto': regreso,
    'hora': horario,
}

def run_ana():
    while True:
        try:
            rec = listen()
        except UnboundLocalError:
            talk("No te entendí, intenta de nuevo")
            continue
```

```
            # print(rec)
            if 'hora' in rec:
                key_words['hora'](rec)
            else:
                for word in key_words:
                    if word in rec:
                        key_words[word](rec)
            if 'termina' in rec:
                talk('Sesion finalizada!')
                break

        main_window.update() #actualizacion de ventana

button_listen = Button(main_window, text="S T A R T", fg="red", bg="#2C5364",
                       font=("Arial", 12, "bold"),
                       command=run_ana)  # command=tr.Thread(target=run_ana).
start()) #width=20, height=1, command=run_ana)
button_listen.place(x=220, y=330, width=100, height=40)  # side = BOTTOM,
pady=10) antes.pack

main_window.mainloop()
```

ANEXO 2: DATASHEET DE DISPOSITIVOS ELECTRÓNICOS

Datos Importantes de Driver DM320T

Microstep	Steps/rev.(for 1.8°motor)	SW4	SW5	SW6
2	400	ON	ON	ON
4	800	OFF	ON	ON
8	1600	ON	OFF	ON
16	3200	OFF	OFF	ON
32	6400	ON	ON	OFF
64	12800	OFF	ON	OFF
20	4000	ON	OFF	OFF
40	8000	OFF	OFF	OFF

7.2 Current Configurations

For a given motor, higher drive current will make the motor to output more torque, but at the same time causes more heating in the motor and drive. Therefore, output current is generally set to be such that the motor will not overheat for long time operation. Since parallel and serial connections of motor coils will significantly change resulting inductance and resistance, it is therefore important to set drive output current depending on motor phase current, motor leads and connection methods. Phase current rating supplied by motor manufacturer is important in selecting drive current, however the selection also depends on leads and connections.

The first three bits (SW1, 2, 3) of the DIP switch are used to set the dynamic current. Select a setting closest to your motor's required current.

7.2.1 Dynamic Current Configurations

Peak Current	RMS Current	SW1	SW2	SW3
0.3A	0.21A	ON	ON	ON
0.5A	0.35A	OFF	ON	ON
0.7A	0.49A	ON	OFF	ON
1.0A	0.71A	OFF	OFF	ON
1.3A	0.92A	ON	ON	OFF
1.6A	1.13A	OFF	ON	OFF
1.9A	1.34A	ON	OFF	OFF
2.2A	1.56A	OFF	OFF	OFF

Notes: Due to motor inductance, the actual current in the coil may be smaller than the dynamic current setting, particularly under high speed condition.

7.2.2 Standstill Current Configuration

The standstill current is set to be 50% of the selected output current. It means standstill current automatically reduced to 50% of the selected dynamic current 0.4 second after the last pulse.

7.3 Automatic Motor Matching & Self Configuration

When powered on a DM320T will automatically configure itself with the best settings to match the driven stepper motor for optimal performance. No action is needed.

Datos Importantes del Driver DM542T

Microstep	Steps/rev.(for 1.8°motor)	SW5	SW6	SW7	SW8
2	400	OFF	ON	ON	ON
4	800	ON	OFF	ON	ON
8	1600	OFF	OFF	ON	ON
16	3200	ON	ON	OFF	ON
32	6400	OFF	ON	OFF	ON
64	12800	ON	OFF	OFF	ON
128	25600	OFF	OFF	OFF	ON
5	1000	ON	ON	ON	OFF
10	2000	OFF	ON	ON	OFF
20	4000	ON	OFF	ON	OFF
25	5000	OFF	OFF	ON	OFF
40	8000	ON	ON	OFF	OFF
50	10000	OFF	ON	OFF	OFF
100	20000	ON	OFF	OFF	OFF
125	25000	OFF	OFF	OFF	OFF

Current Settings

For a given motor, higher drive current will make the motor to output more torque, but at the same time causes more heating in the motor and drive. Therefore, output current is generally set to be such that the motor will not overheat for long time operation. Since parallel and serial connections of motor coils will significantly change resulting inductance and resistance, it is therefore important to set drive output current depending on motor phase current, motor leads and connection methods. Phase current rating supplied by motor manufacturer is important in selecting drive current, however the selection also depends on leads and connections.

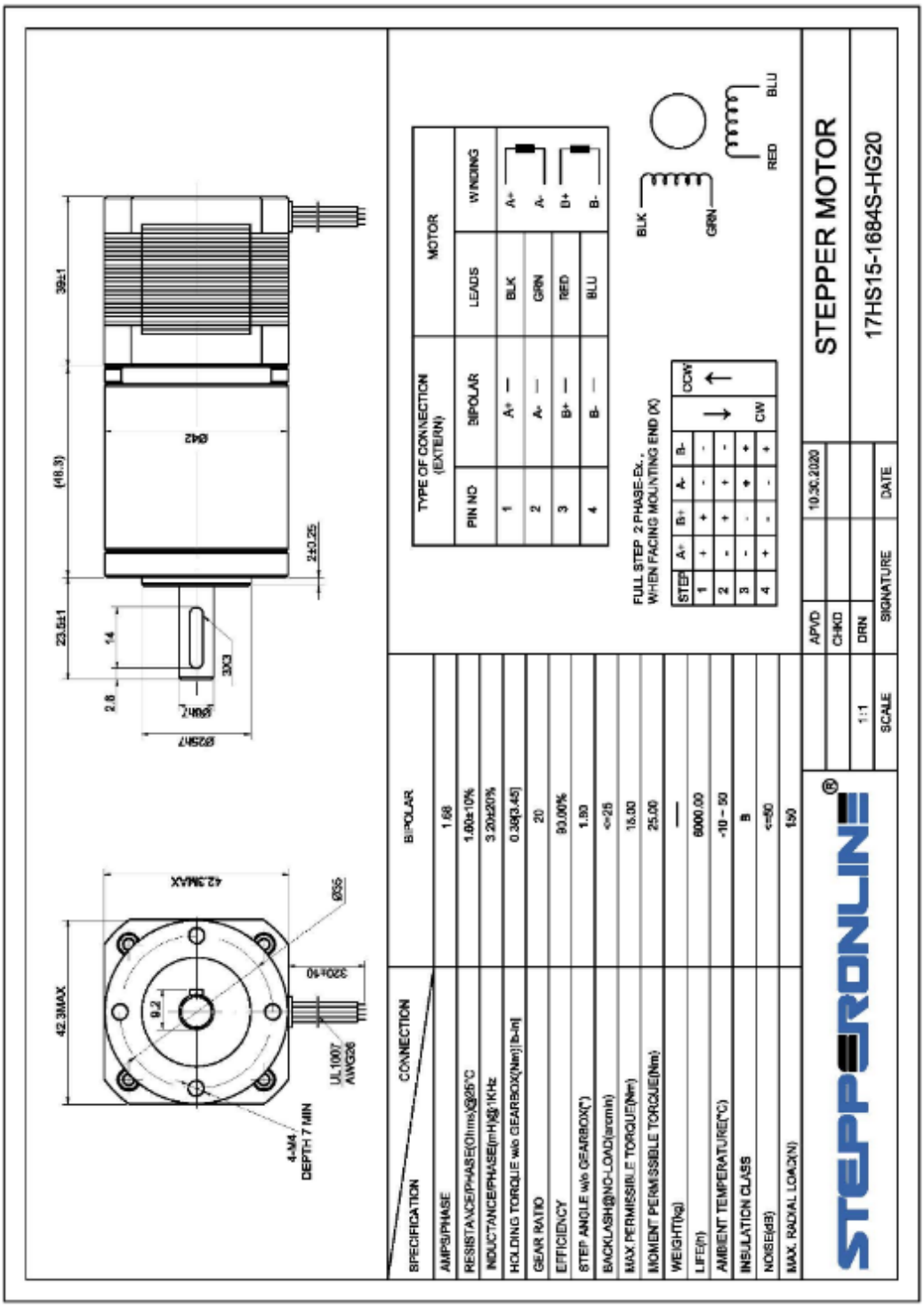
The first three bits (SW1, 2, 3) of the DIP switch are used to set the dynamic current. Select a setting closest to your motor's required current.

Dynamic Current Setting

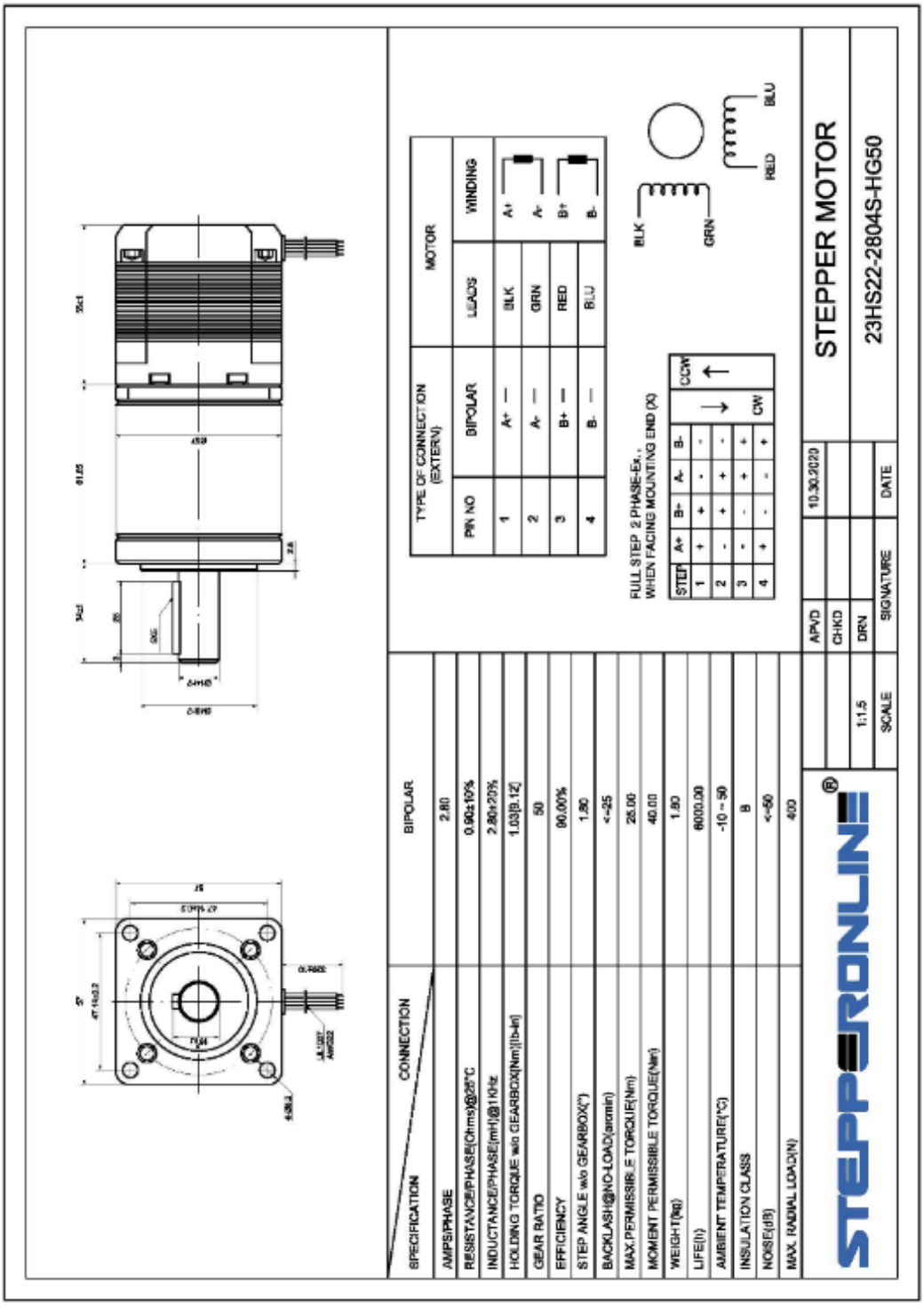
Peak Current	RMS Current	SW1	SW2	SW3
1.00A	0.71A	ON	ON	ON
1.46A	1.04A	OFF	ON	ON
1.91A	1.36A	ON	OFF	ON
2.37A	1.69A	OFF	OFF	ON
2.84A	2.03A	ON	ON	OFF
3.31A	2.36A	OFF	ON	OFF
3.76A	2.69A	ON	OFF	OFF
4.20A	3.00A	OFF	OFF	OFF

Notes: Due to motor inductance, the actual current in the coil may be smaller than the dynamic current setting, particularly under high speed condition.

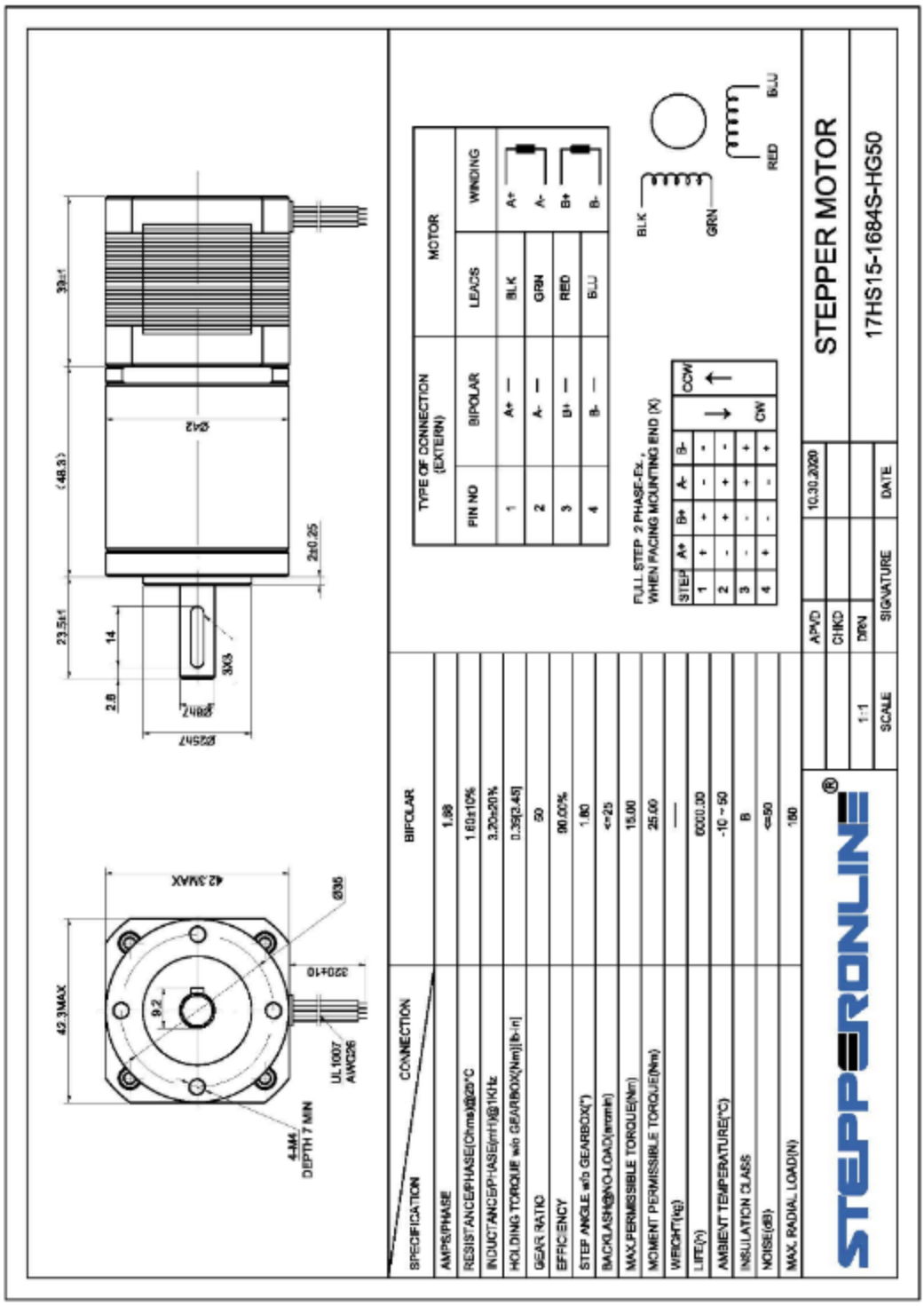
Motor Paso a Paso 1



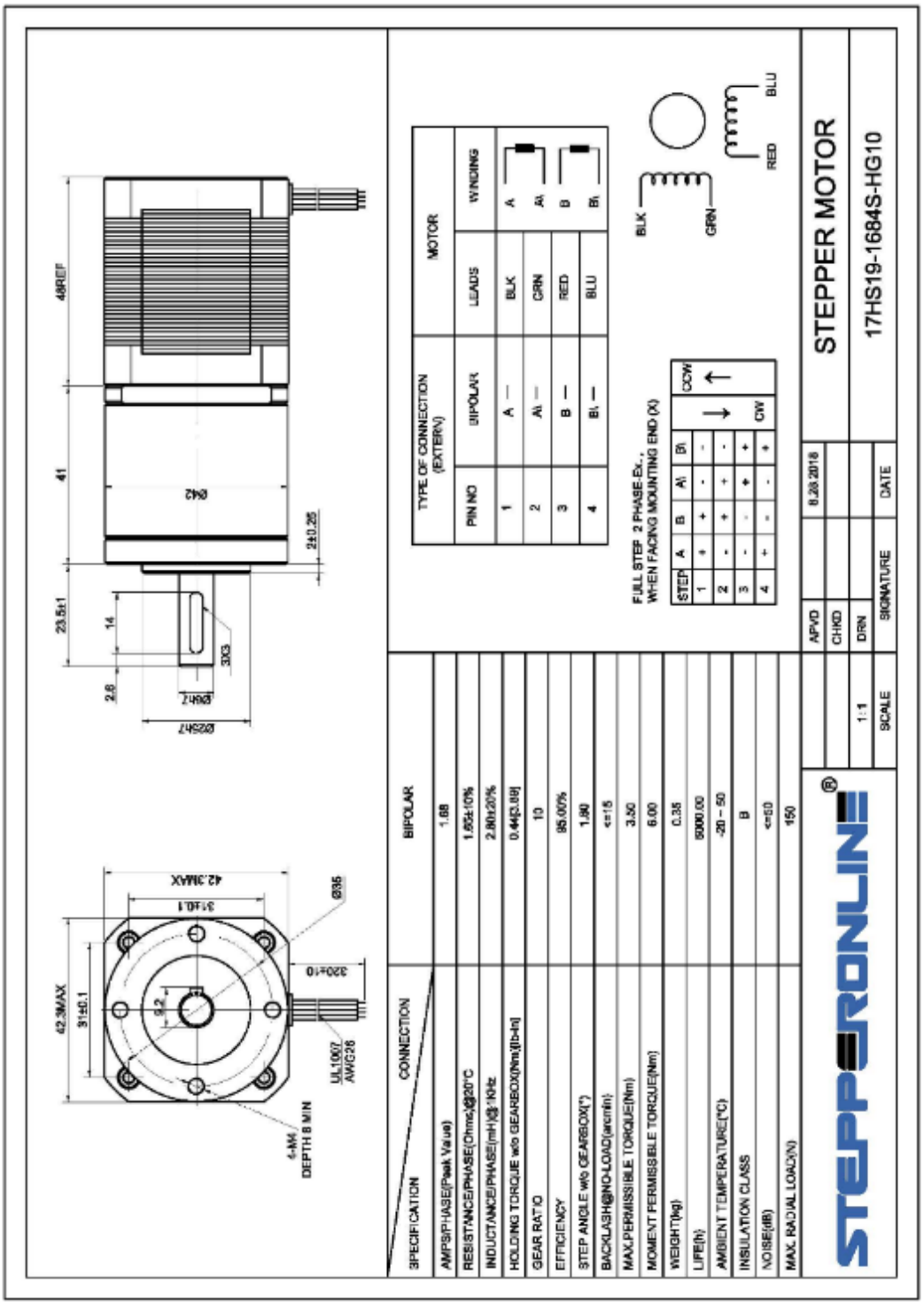
Motor Paso a Paso 2



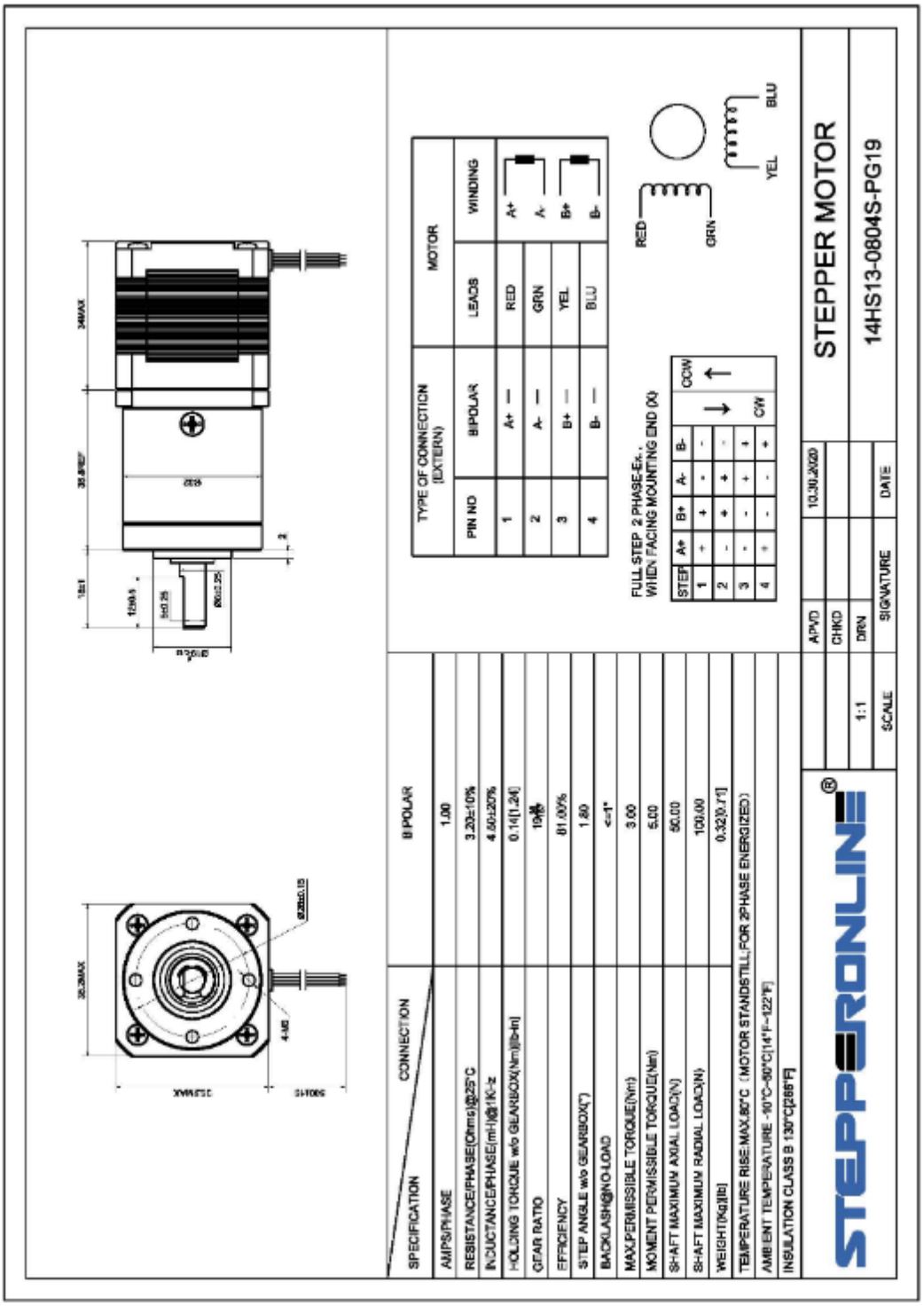
Motor Paso a Paso 3



Motor Paso a Paso 4

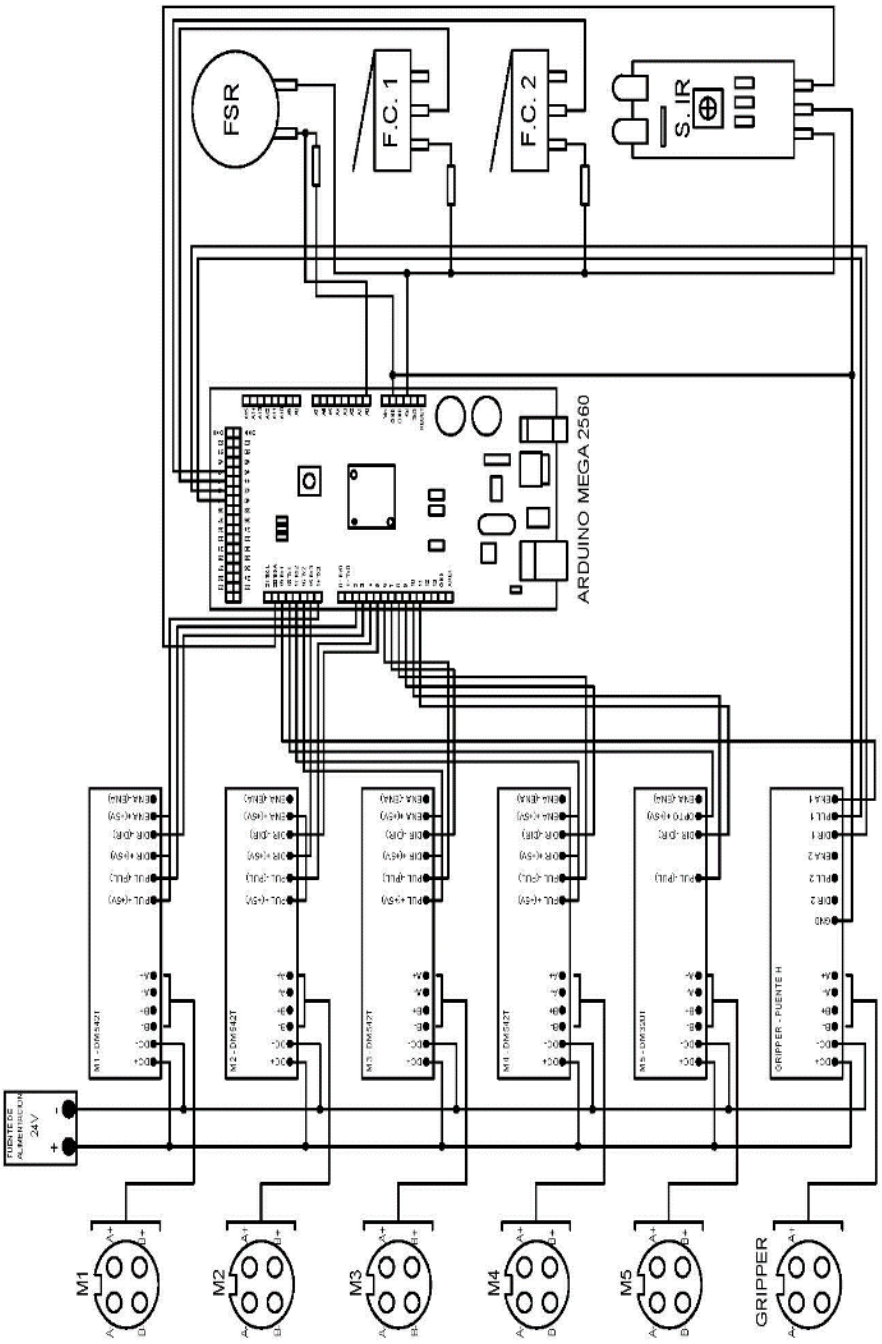


Motor Paso a Paso 5

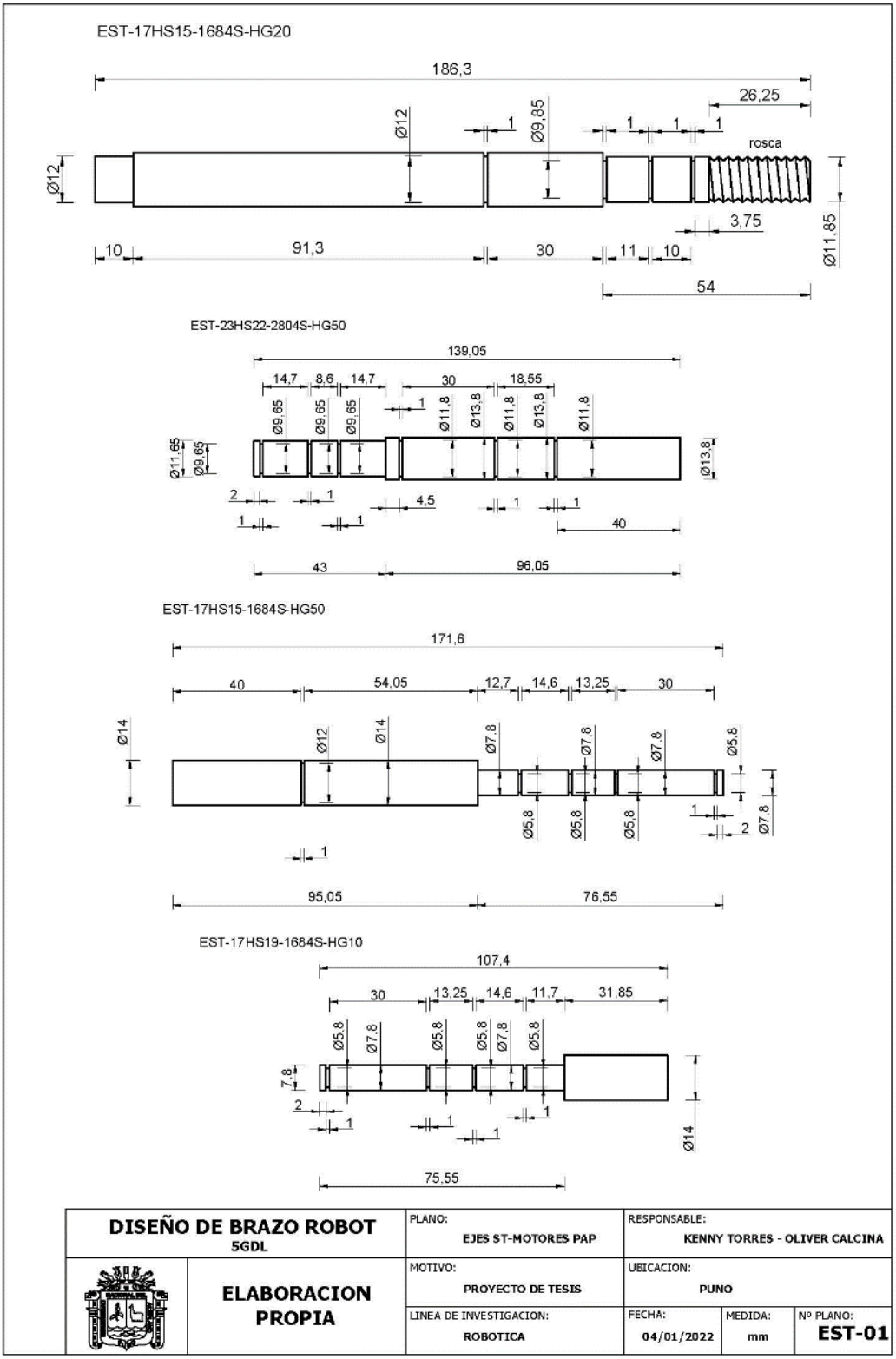


ANEXO 3: PLANOS DEL PROYECTO

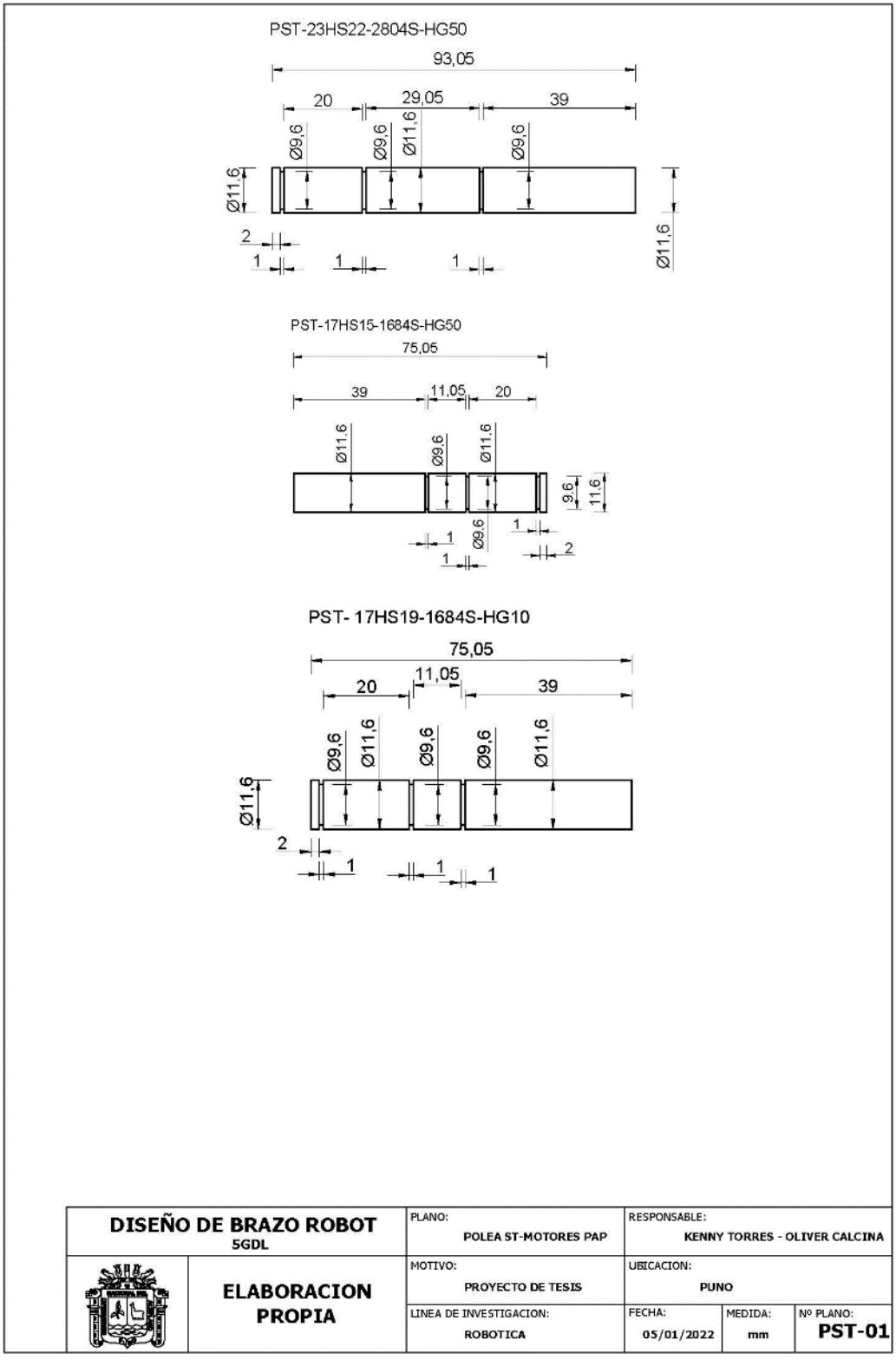
Diagrama de Conexión – Dispositivos electrónicos



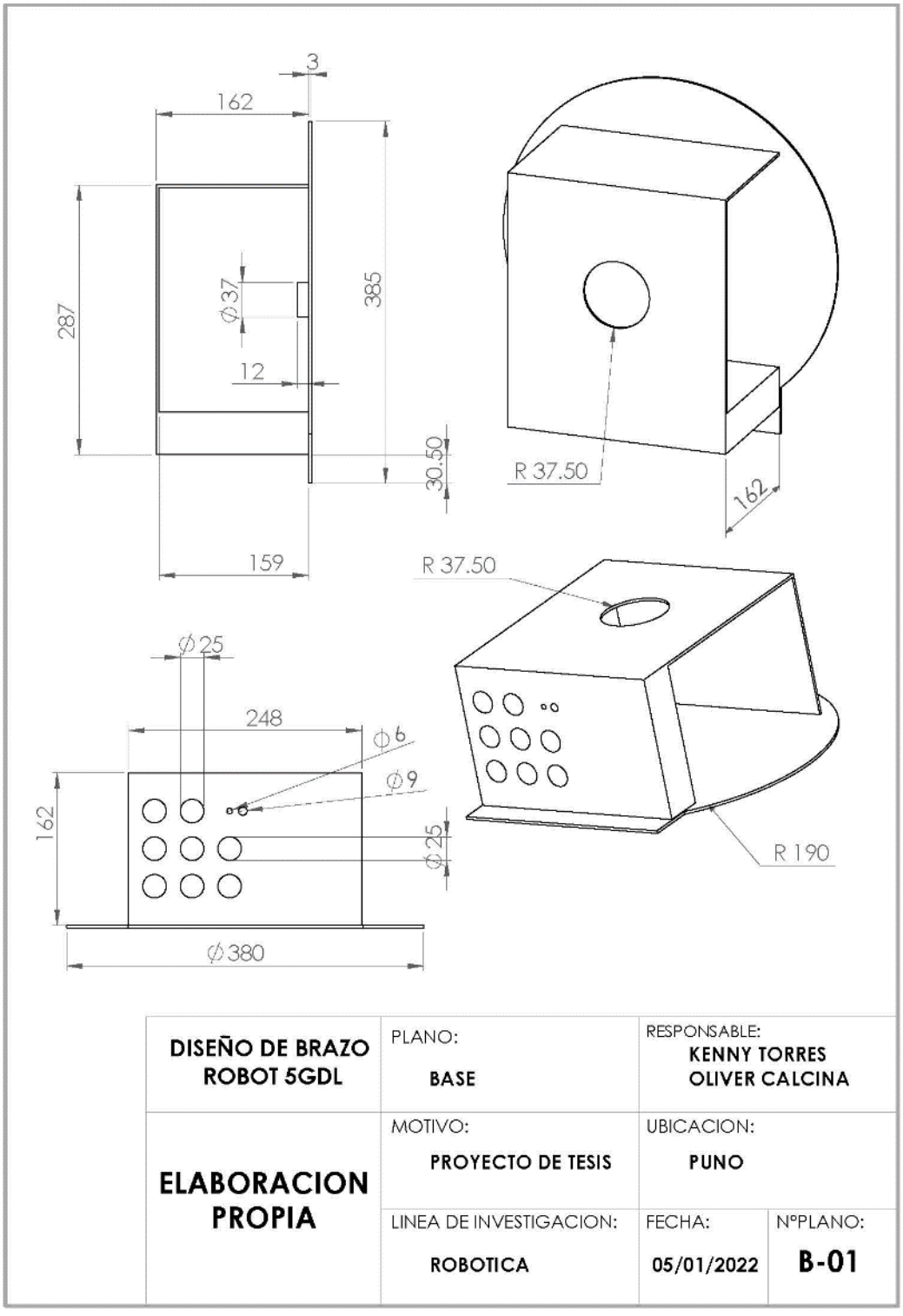
AutoCAD – Ejes para el Sistema de Transmisión del Brazo Robot



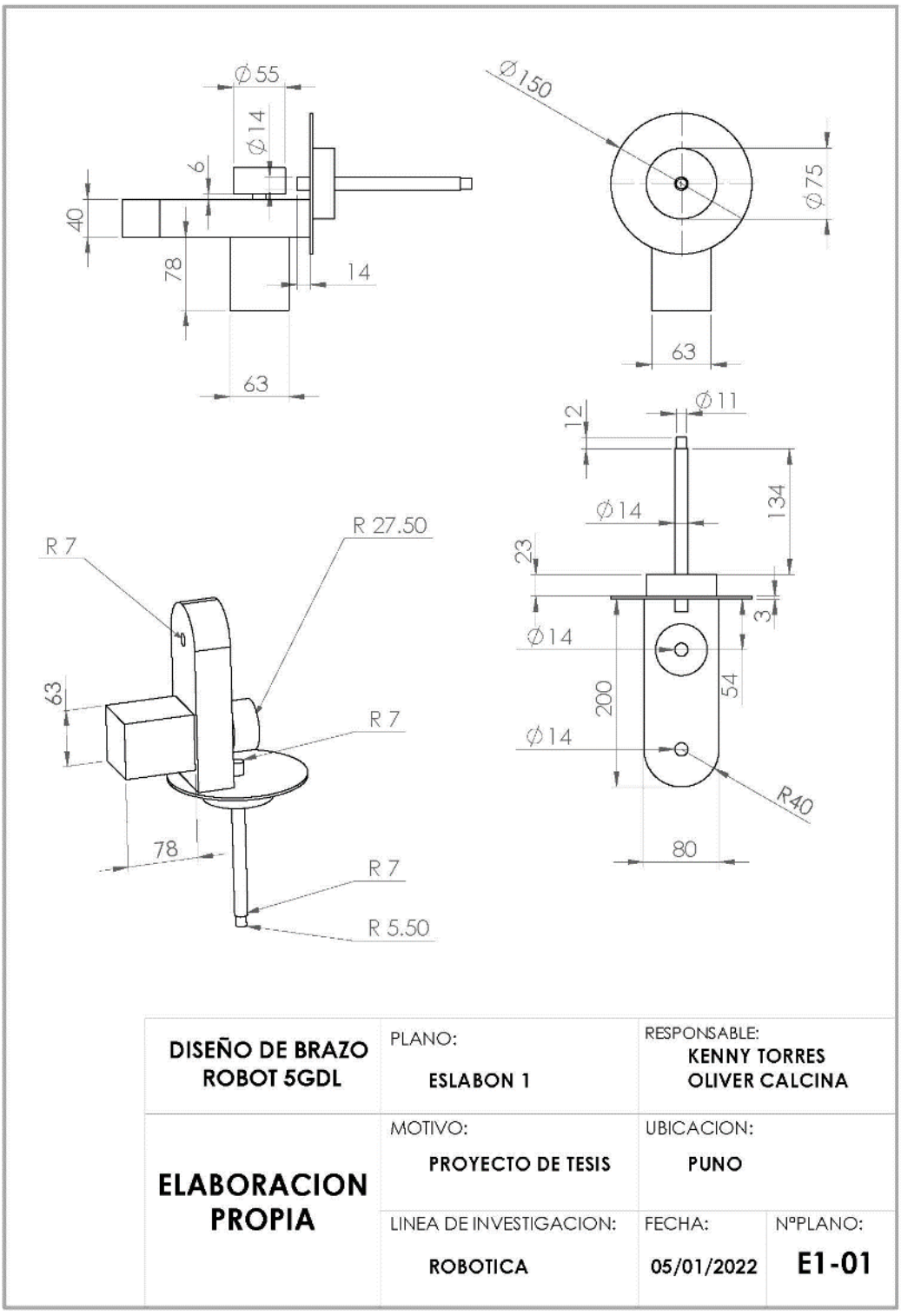
AutoCAD – Poleas para el Sistema de Transmisión del Brazo Robot



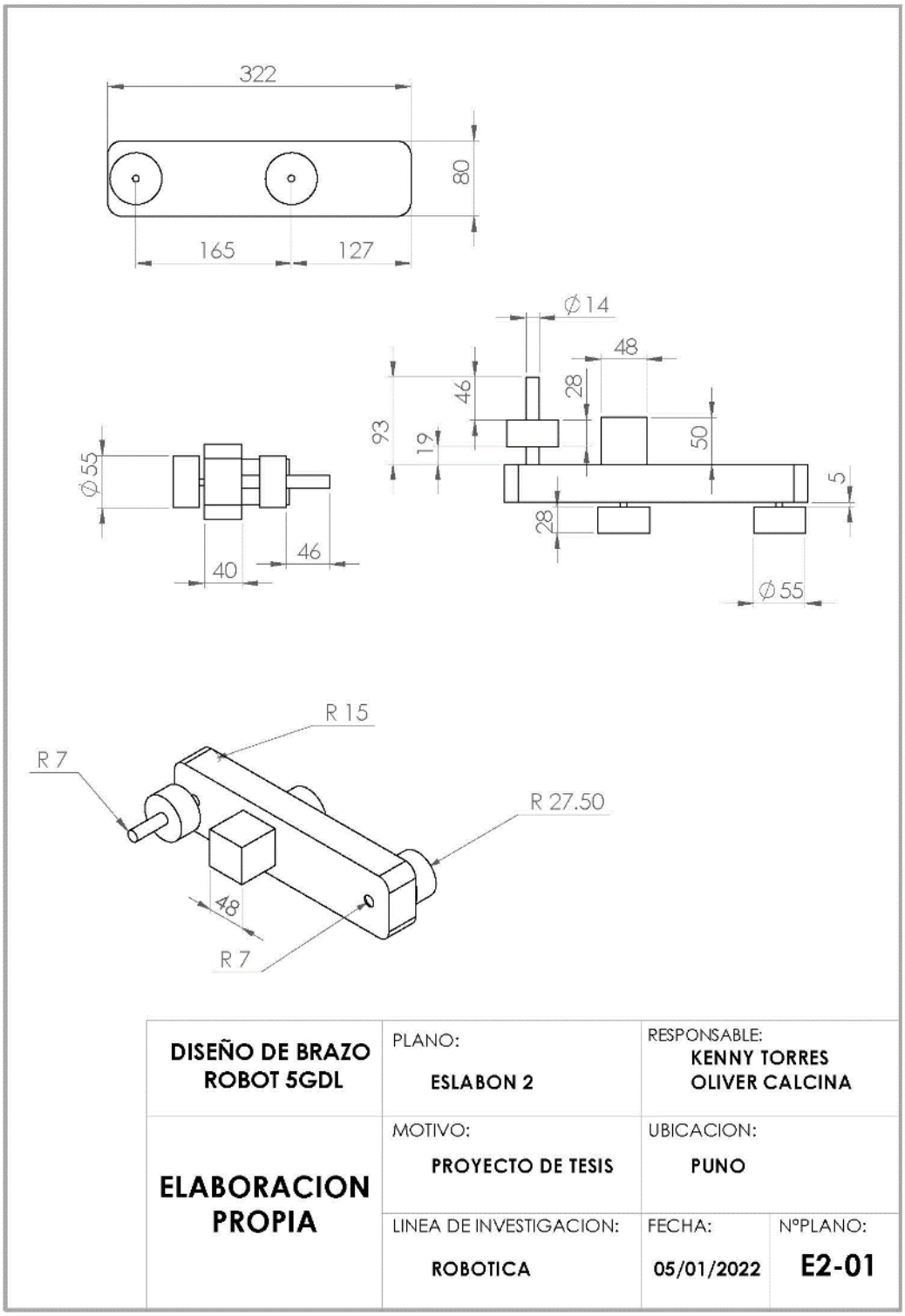
SolidWorks - Base del Brazo Robot



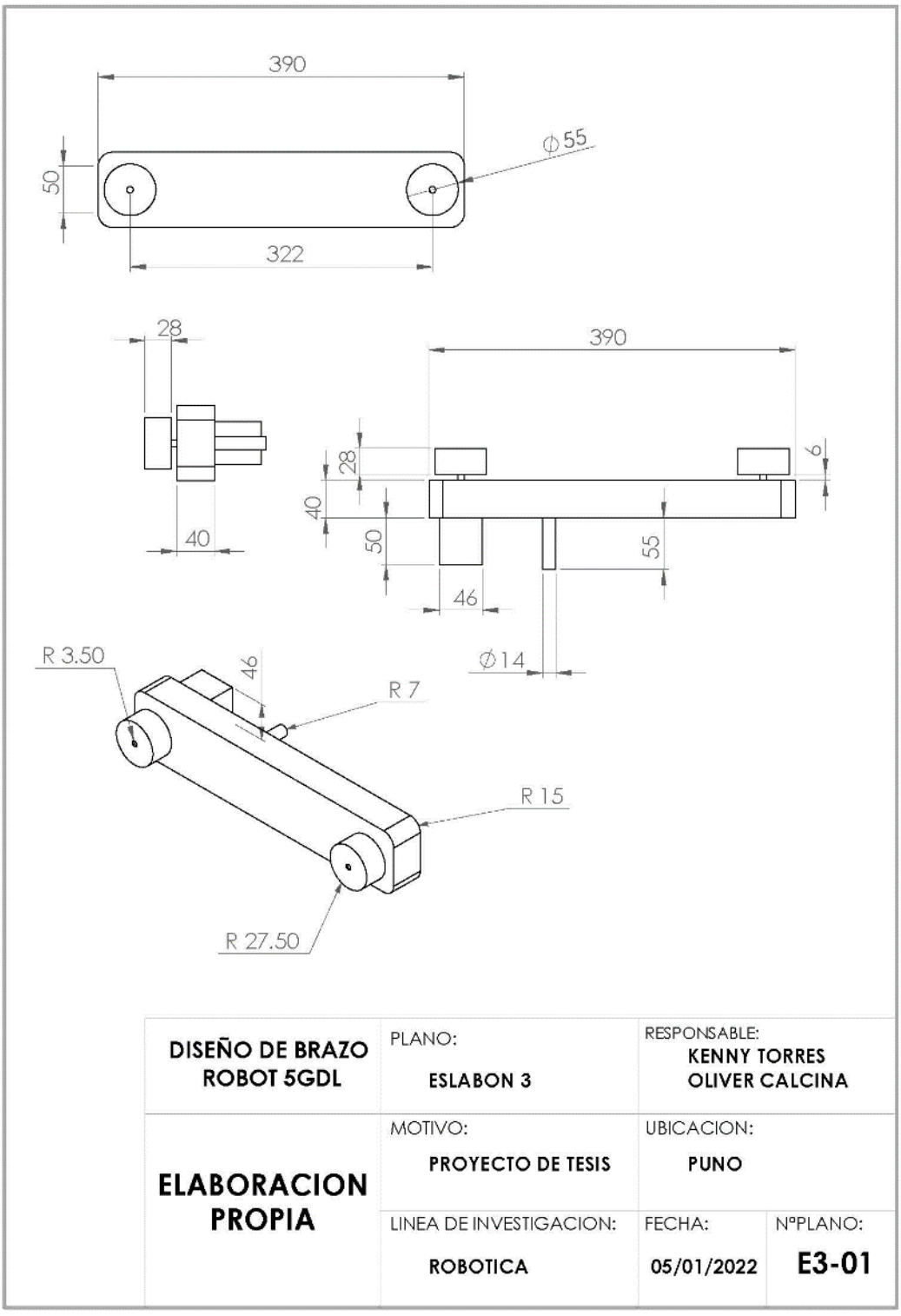
SolidWorks – Eslabón 1 del Brazo Robot



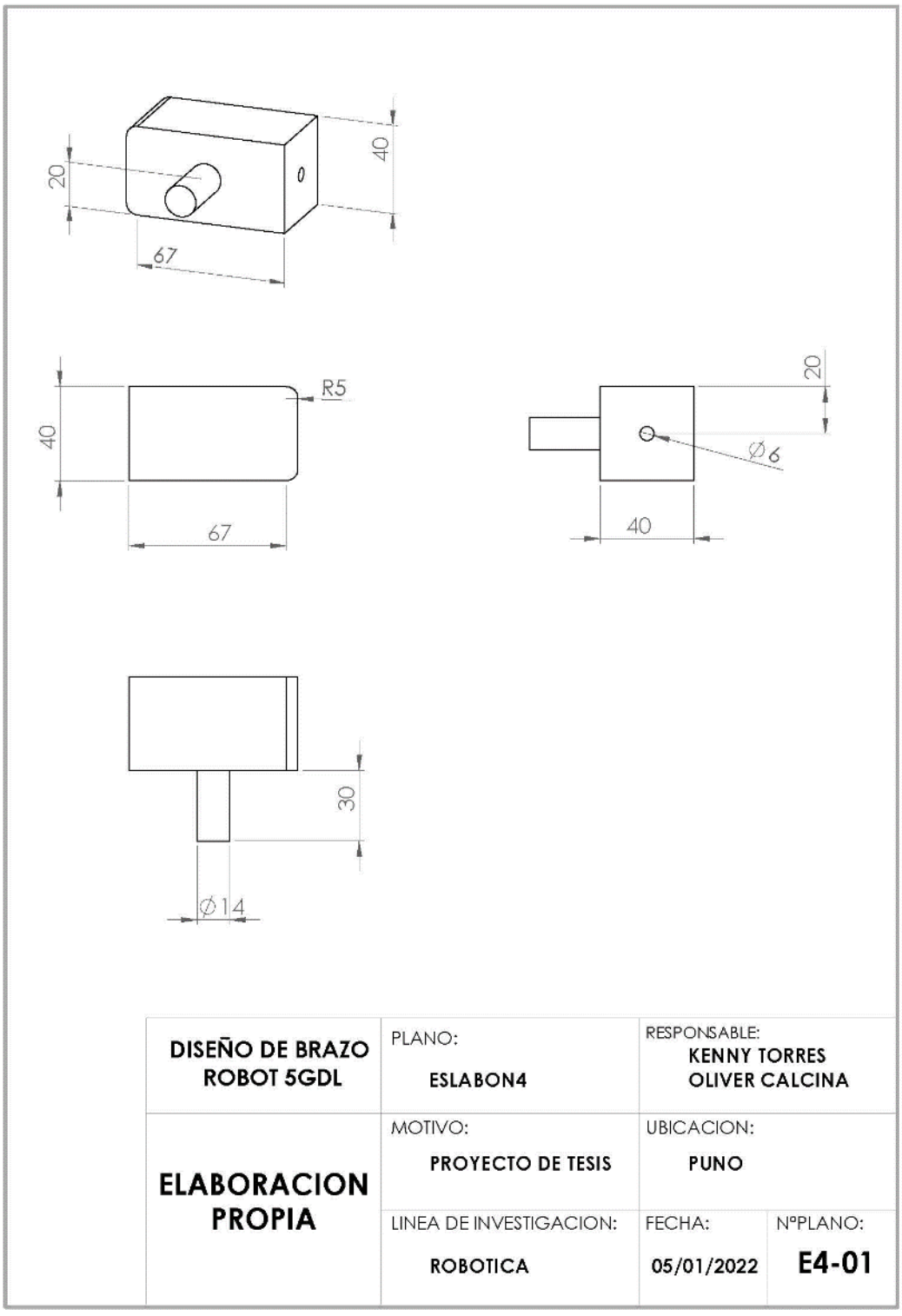
SolidWorks – Eslabón 2 del Brazo Robot



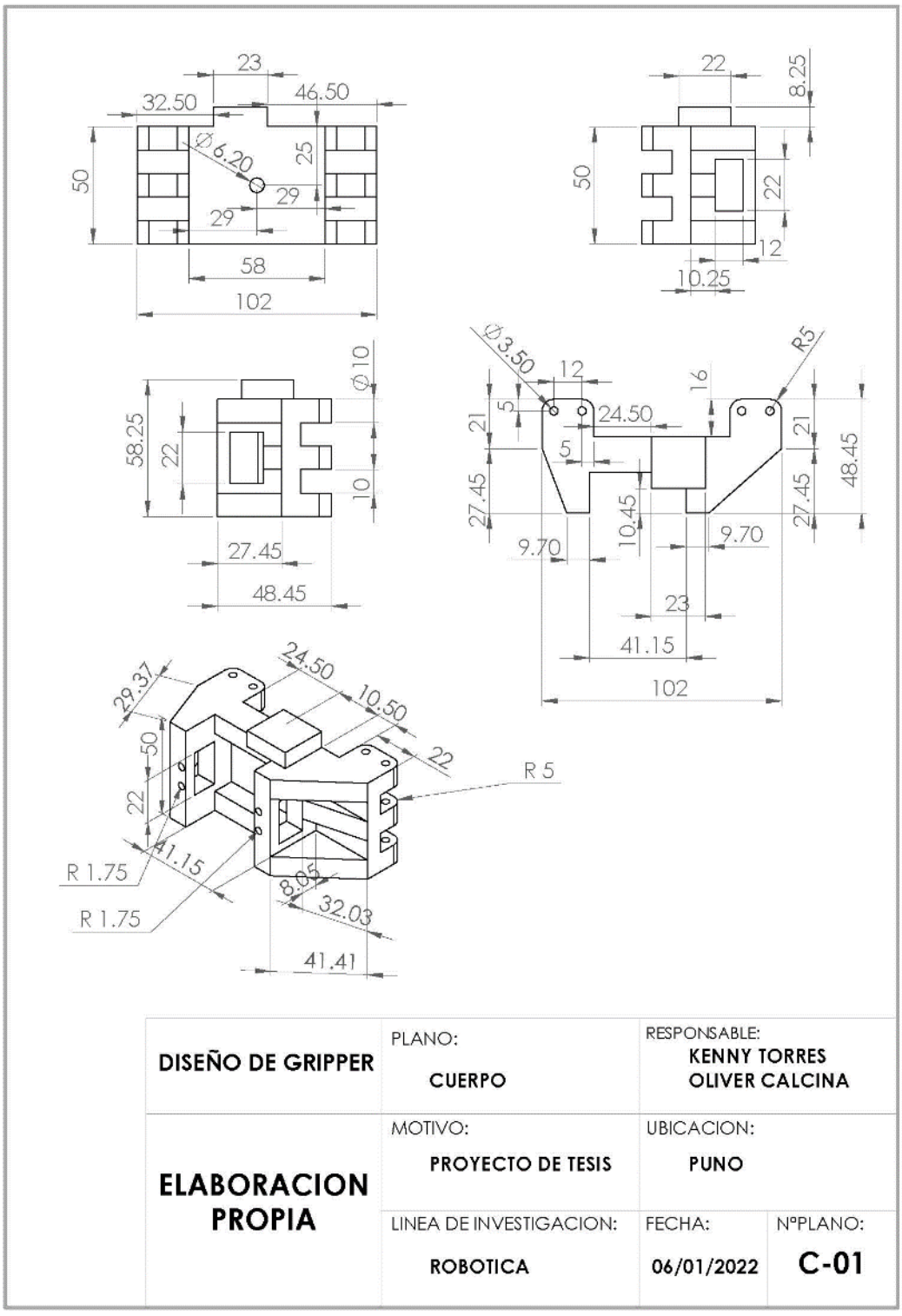
SolidWorks – Eslabón 3 del Brazo Robot



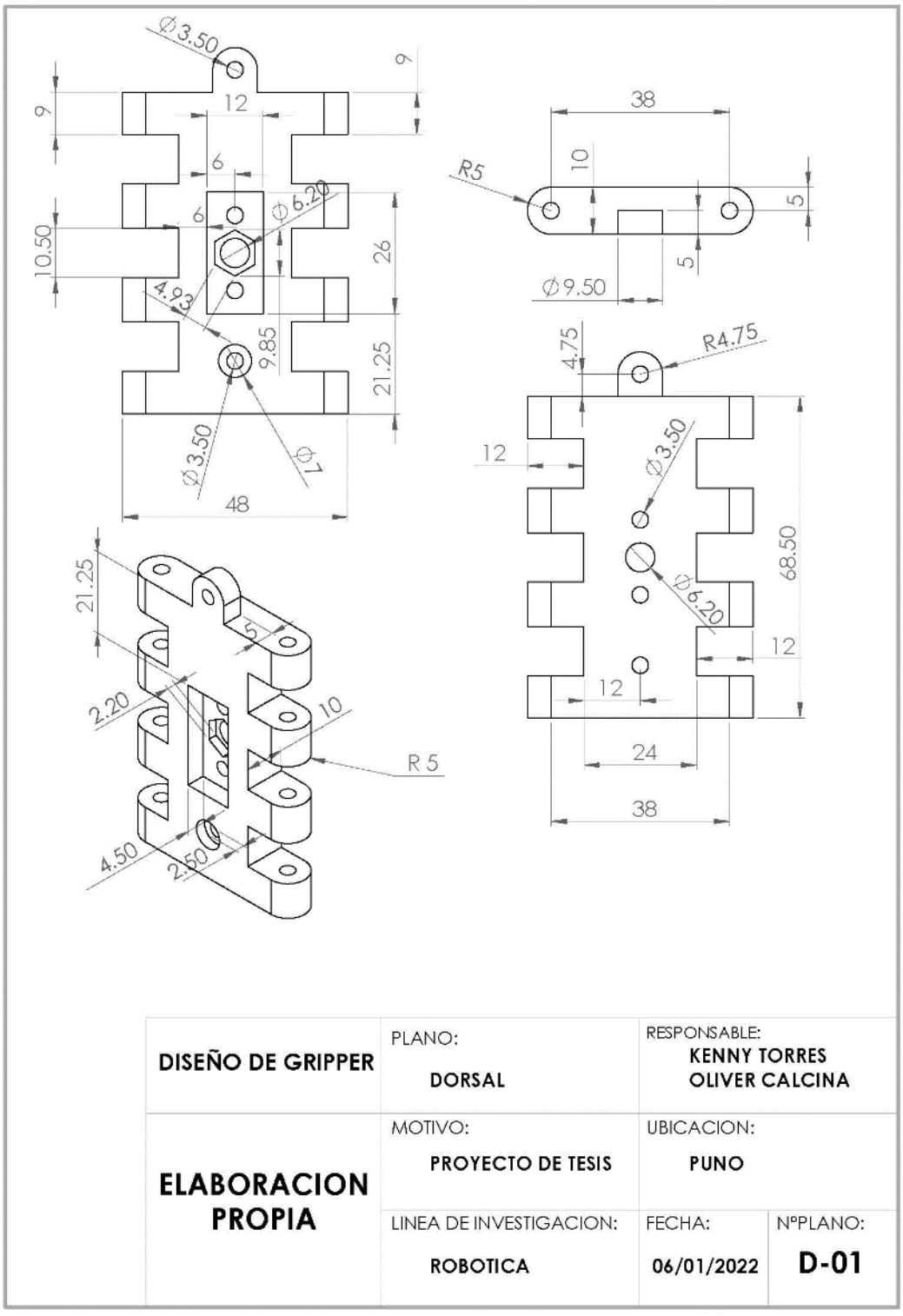
SolidWorks – Eslabón 4 del Brazo Robot



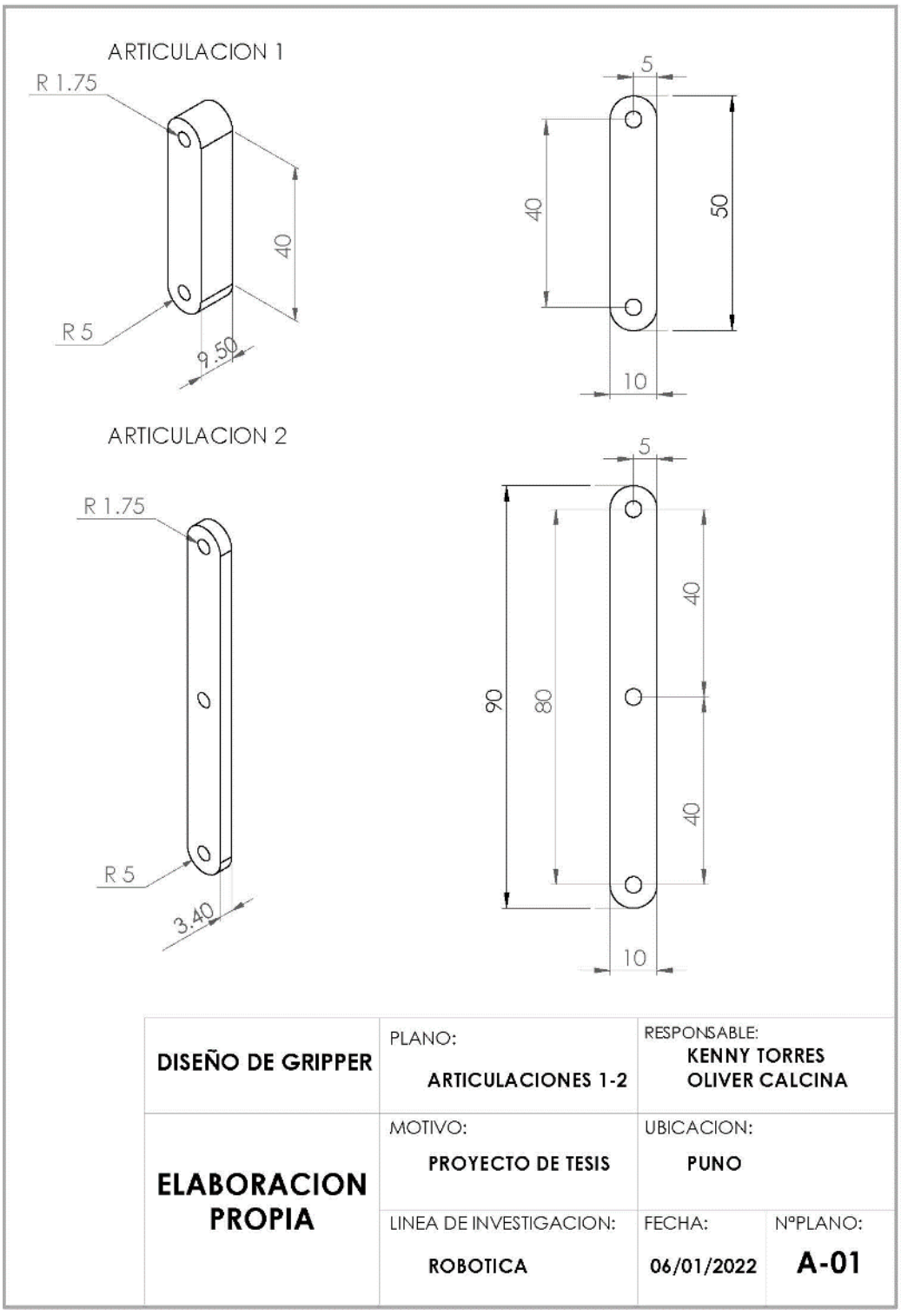
SolidWorks – Cuerpo del Gripper



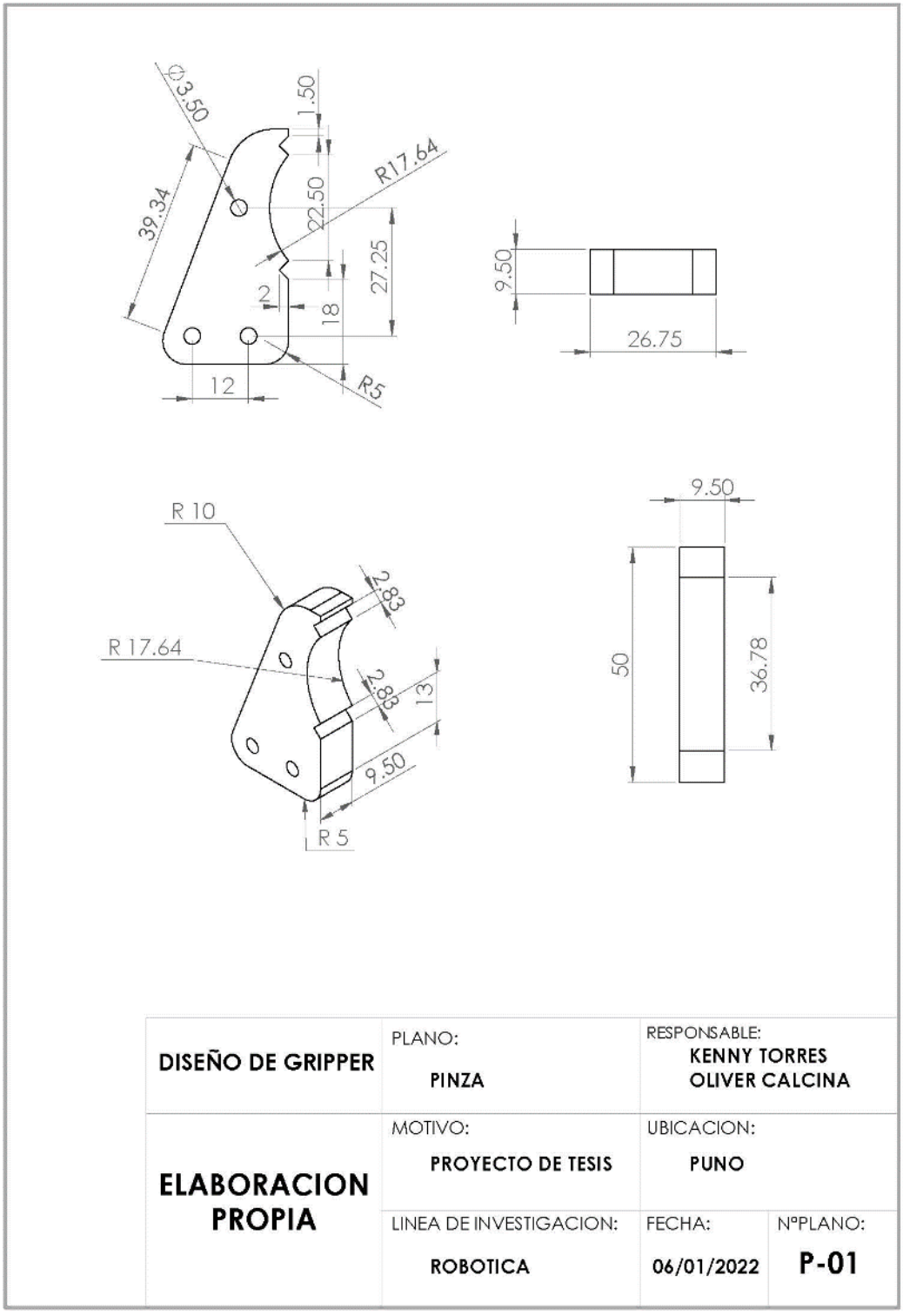
SolidWorks – Dorsal del Gripper



SolidWorks – Articulaciones del Gripper

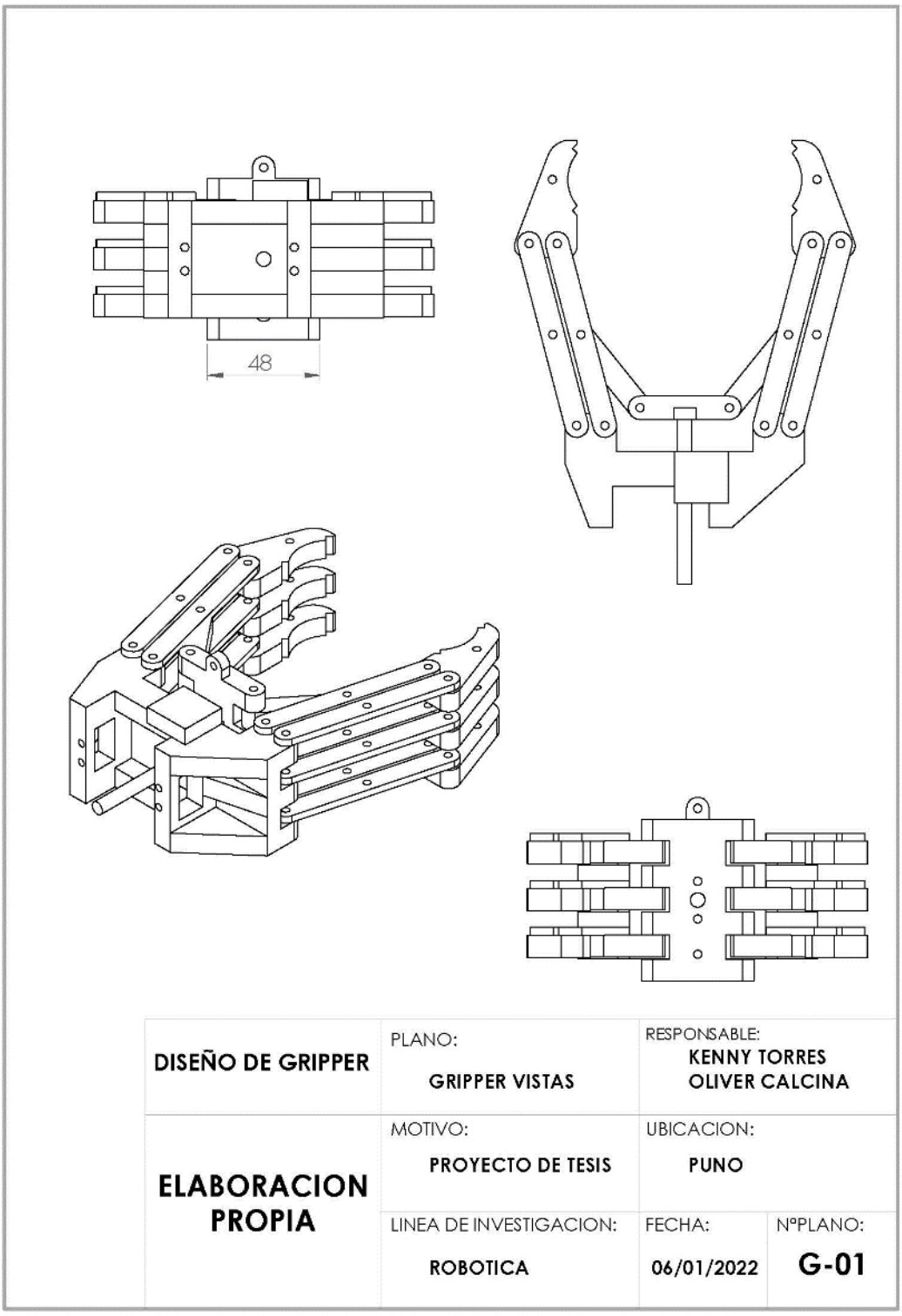


SolidWorks – Pinza del Gripper

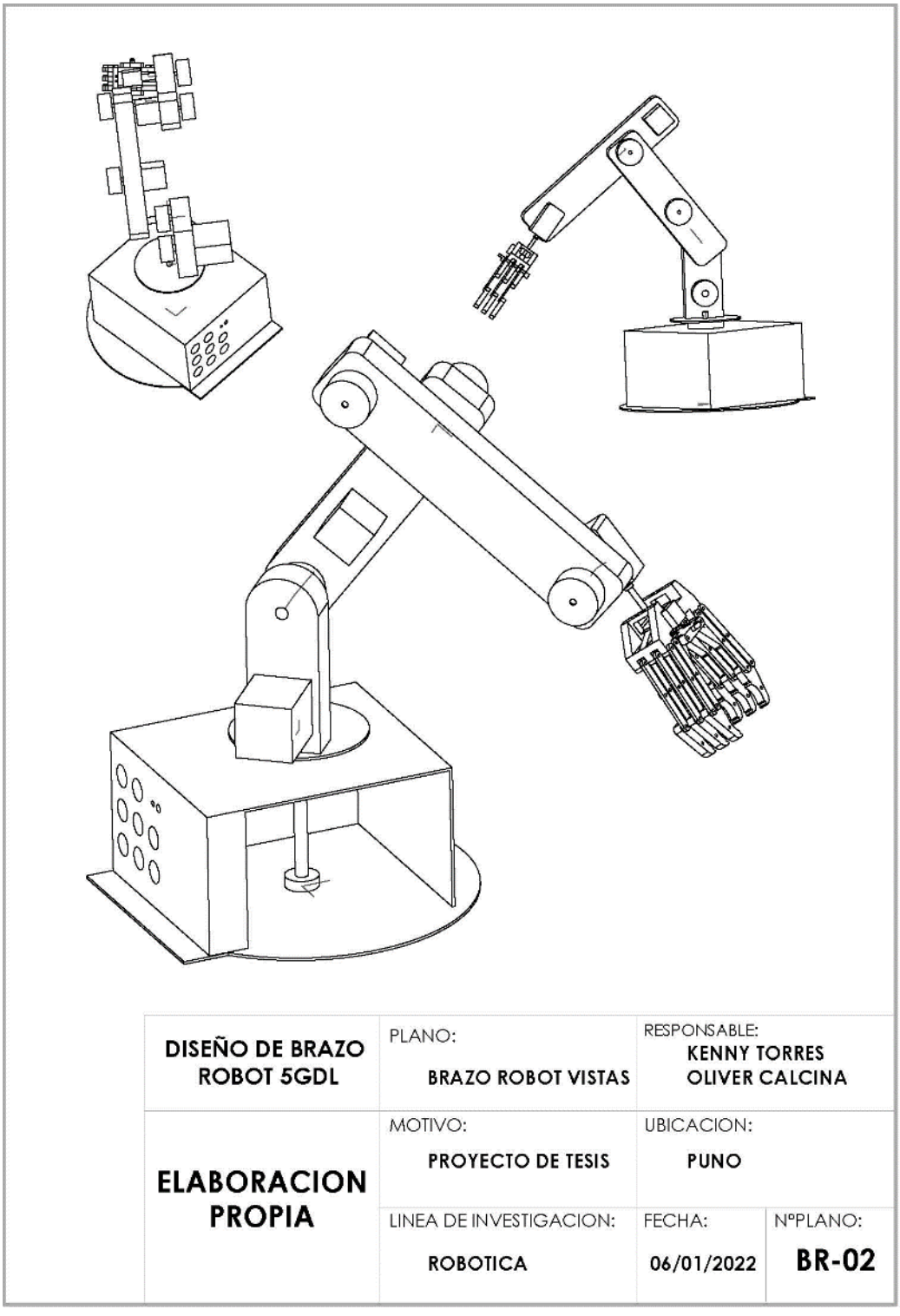


DISEÑO DE GRIPPER	PLANO: PINZA	RESPONSABLE: KENNY TORRES OLIVER CALCINA	
ELABORACION PROPIA	MOTIVO: PROYECTO DE TESIS	UBICACION: PUNO	
	LINEA DE INVESTIGACION: ROBOTICA	FECHA: 06/01/2022	NºPLANO: P-01

SolidWorks – Gripper



SolidWorks – Brazo Robot con Gripper



ANEXO 4: DOCUMENTOS

Matriz de Consistencia

DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO DE UN BRAZO ROBOT Y VISIÓN ARTIFICIAL CONTROLADO POR PATRONES DE VOZ PARA MANIPULACIÓN DE OBJETOS EN LA EMPRESA AUTOMYN				
FORMULACION DEL PROBLEMA	**OBJETIVOS**	**HIPOTESIS**	**VARIABLES**	**METODOLOGIA**
Problema General ¿Cómo realizar el diseño e implementación del prototipo de un brazo robot y visión artificial con los patrones de voz para la manipulación de objetos?	**Objetivo General** Diseñar e implementar un brazo robot y visión artificial controlado por patrones de voz para la manipulación de objetos en la empresa Automyn.	**Hipótesis General** El diseño e implementación de un brazo robot con visión artificial controlado por patrones lograra realizar una manipulación colaborativa de objetos entre humano y máquina a través de una computadora.	**Variable Independiente** Patrones de voz **Dimensiones** Reconocimiento de voz **Indicadores** Dispositivo de entrada audio	**Tipo** cuasi experimental - enfoque cuantitativo **Nivel** Descriptivo **Población** Dispositivos electrónicos, actuadores, controladores
Problema Especifico ¿Cómo automatizar el proceso de recoger y soltar cualquier tipo de objeto? ¿Cómo desarrollar el sistema de visión artificial para detectar cualquier tipo de objeto en un espacio determinado? ¿Como determinar un control de patrones de voz para realizar la función de hombre maquina?	**Objetivos Especificos** Diseñar e implementar el mecanismo del brazo robot de 5GDL con Gripper. Realizar la detección de objetos con estimación de posición para el sistema de visión artificial. Desarrollar el software de un asistente virtual para el control de voz por patrones.	**Hipótesis Especificas** El diseño del análisis matemático y la implementación de un brazo robot con Gripper realizará el control de movimiento y agarre del objeto. El desarrollo de un detector de objetos con estimación de posición logrará una aproximación visual de lo que queremos obtener en el plano coordenado del brazo robot. Es posible la ejecución de un asistente virtual para el control de programas por patrones de voz con el desarrollo de software en este proyecto de tesis.	**Variable Dependiente** Brazo robot **Dimensiones** Movimiento Control de Gripper **Indicadores** Cinemática inversa diferencial Control de sensores y actuadores Visión artificial **Dimensiones** Detección de objetos Estimación de posición **Indicadores** Clasificador de imágenes Distancias, parámetros de imágenes.	**Muestra** Se usará materiales descritos en la población **Técnicas** Se usará técnicas de observación y redacción de información **Instrumentos** softwares de diseño, internet, simuladores, papel, lápices.

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo **KENNY BRIAN TORRES LUNA - FREDY OLIVER CALCINA PAREDES**, identificado con DNI **72732006 – 48098783** en mi condición de egresado de:

☒**Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado** **INGENIERÍA ELECTRÓNICA**, informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO DE UN BRAZO ROBOT Y VISIÓN ARTIFICIAL CONTROLADO POR PATRONES DE VOZ PARA MANIPULACIÓN DE OBJETOS EN LA EMPRESA AUTOMYN"

Es un tema original.

Declaro que el presente trabajo de tesis es elaborado por mi persona y **no existe plagio/copia** de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como suyas las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error u omisión en el documento, así como de las connotaciones éticas y legales involucradas.

En caso de incumplimiento de esta declaración, me someto a las disposiciones legales vigentes y a las sanciones correspondientes de igual forma me someto a las sanciones establecidas en las Directivas y otras normas internas, así como las que me alcancen del Código Civil y Normas Legales conexas por el incumplimiento del presente compromiso

Puno 14 de abril del 2023

KENNY BRIAN TORRES LUNA

FREDY OLIVER CALCINA PAREDES

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

AUTORIZACIÓN PARA EL DEPÓSITO DE TESIS O TRABAJO DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL

Por el presente documento, Yo **KENNY BRIAN TORRES LUNA - FREDY OLIVER CALCINA PAREDES**, identificado con DNI **72732006 – 48098783** en mi condición de egresado de:

☒ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

INGENIERÍA ELECTRÓNICA, informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:

"DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO DE UN BRAZO ROBOT Y VISIÓN ARTIFICIAL CONTROLADO POR PATRONES DE VOZ PARA MANIPULACIÓN DE OBJETOS EN LA EMPRESA AUTOMYN"

para la obtención de ☐**Grado,** ☒ **Título Profesional** o ☐ **Segunda Especialidad.**

Por medio del presente documento, afirmo y garantizo ser el legítimo, único y exclusivo titular de todos los derechos de propiedad intelectual sobre los documentos arriba mencionados, las obras, los contenidos, los productos y/o las creaciones en general (en adelante, los "Contenidos") que serán incluidos en el repositorio institucional de la Universidad Nacional del Altiplano de Puno.

También, doy seguridad de que los contenidos entregados se encuentran libres de toda contraseña, restricción o medida tecnológica de protección, con la finalidad de permitir que se puedan leer, descargar, reproducir, distribuir, imprimir, buscar y enlazar los textos completos, sin limitación alguna.

Autorizo a la Universidad Nacional del Altiplano de Puno a publicar los Contenidos en el Repositorio Institucional y, en consecuencia, en el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto, sobre la base de lo establecido en la Ley N° 30035, sus normas reglamentarias, modificatorias, sustitutorias y conexas, y de acuerdo con las políticas de acceso abierto que la Universidad aplique en relación con sus Repositorios Institucionales. Autorizo expresamente toda consulta y uso de los Contenidos, por parte de cualquier persona, por el tiempo de duración de los derechos patrimoniales de autor y derechos conexos, a título gratuito y a nivel mundial.

En consecuencia, la Universidad tendrá la posibilidad de divulgar y difundir los Contenidos, de manera total o parcial, sin limitación alguna y sin derecho a pago de contraprestación, remuneración ni regalía alguna a favor mío; en los medios, canales y plataformas que la Universidad y/o el Estado de la República del Perú determinen, a nivel mundial, sin restricción geográfica alguna y de manera indefinida, pudiendo crear y/o extraer los metadatos sobre los Contenidos, e incluir los Contenidos en los índices y buscadores que estimen necesarios para promover su difusión.

Autorizo que los Contenidos sean puestos a disposición del público a través de la siguiente licencia: Creative

Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional. Para ver una copia de esta licencia, visita: https://creativecommons.org/licenses/by-nc-sa/4.0/

En señal de conformidad, suscribo el presente documento.

Puno 14 de abril del 2023

KENNY BRIAN TORRES LUNA

FREDY OLIVER CALCINA PAREDES