DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORES:

VELASCO SÁNCHEZ, EDISON PATRICIO

MAMARANDI QUILO, JAVIER ALEXANDER

TEMA: "DISEÑO DEL SISTEMA DE CONTROL BASADO EN SOFTWARE LIBRE PARA UN BRAZO ROBÓTICO DE 6 GRADOS DE LIBERTAD CON FUNCIONALIDAD DE MECANIZADO Y PALETIZADO PARA EL LABORATORIO DE CIRCUITOS ELECTRÓNICOS DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE EXTENSIÓN LATACUNGA"

DIRECTOR: ING. RIVAS, DAVID

CODIRECTOR: ING. ÁLVAREZ, MARCELO

LATACUNGA, ENERO 2015

UNIVERSIDAD DE LA FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

DECLARACIÓN DE RESPONSABILIDAD

Nosotros: Velasco Sánchez Edison Patricio
Mamarandi Quilo Javier Alexander

DECLARAMOS QUE:

El proyecto de grado denominado "DISEÑO DEL SISTEMA DE CONTROL BASADO EN SOFTWARE LIBRE PARA UN BRAZO ROBÓTICO DE 6 GRADOS DE LIBERTAD CON FUNCIONALIDAD DE MECANIZADO Y PALETIZADO PARA EL LABORATORIO DE CIRCUITOS ELECTRÓNICOS DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE EXTENSIÓN LATACUNGA", ha sido desarrollado en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan el pie de las páginas correspondiente, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, Enero 2015

Velasco Edison	Mamarandi Javier
C.C. 050332110-1	C.C. 050295391-2

UNIVERSIDAD DE LA FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICADO

Ing. Rivas David (DIRECTOR DE TESIS)

Ing. Álvarez Marcelo (CODIRECTOR DE TESIS)

CERTIFICAN

Que el trabajo titulado "DISEÑO DEL SISTEMA DE CONTROL BASADO EN SOFTWARE LIBRE PARA UN BRAZO ROBÓTICO DE 6 GRADOS DE LIBERTAD CON FUNCIONALIDAD DE MECANIZADO Y PALETIZADO PARA EL LABORATORIO DE CIRCUITOS ELECTRÓNICOS DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE EXTENSIÓN LATACUNGA" realizado por los señores Velasco Sánchez Edison Patricio y Mamarandi Quilo Javier Alexander, ha sido guiado y revisado periódicamente y cumple normas estatuarias establecidas por la ESPE, en el Reglamento de Estudiantes de la Universidad de las Fuerzas Armadas-ESPE.

Debido a que constituye un trabajo de alto contenido de investigación y que ayuda a la formación profesional y aplicación de conocimientos, si se recomienda su publicación.

El mencionado trabajo consta de dos documentos empastado y dos discos compactos los cuales contienen los archivos en formato portátil de Acrobat (pdf). Autorizan a los señores Velasco Sánchez Edison Patricio y Mamarandi Quilo Javier Alexander que lo entregue al Ing. Silva Franklin, en su calidad de Director de la Carrera.

Latacunga, Enero 2015

Ing. Rivas David	Ing. Álvarez Marcelo
DIRECTOR	CODIRECTOR

UNIVERSIDAD DE LA FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Nosotros: Velasco Sánchez Edison Patricio
Mamarandi Quilo Javier Alexander

Autorizamos a la Universidad de las Fuerzas Armadas - ESPE la publicación, en la biblioteca virtual de la Institución del trabajo "DISEÑO DEL SISTEMA DE CONTROL BASADO EN SOFTWARE LIBRE PARA UN BRAZO ROBÓTICO DE 6 GRADOS DE LIBERTAD CON FUNCIONALIDAD DE MECANIZADO Y PALETIZADO PARA EL LABORATORIO DE CIRCUITOS ELECTRÓNICOS DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE EXTENSIÓN LATACUNGA", cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Latacunga, Enero del 2015

Velasco Edison	Mamarandi Javier
C.C. 050332110-1	C.C. 050295391-2

DEDICATORIA

Principalmente dedico este trabajo a mi familia puesto que me brindaron el apoyo y fortaleza necesarios para el desarrollo y transcurso de mi tesis, ayudándome a concluir satisfactoriamente este propósito en mi vida.

También dedico este proyecto a las futuras generaciones, que están dispuestas a aprender para superarse y aportar sus conocimientos a la sociedad.

EDISON

Todo lo que se ha logrado hasta la fecha en lo personal y profesionalmente se lo quiero dedicar a Dios y a mis padres porque han sido todo mi apoyo incondicional y el ejemplo, molde y modelo que quiero llegar a ser en el futuro. Dedico este esfuerzo también a mi hermana y amigos que a pesar de los años de trabajo si hemos caminado juntos o no, todos y cada uno han sabido inspirarme y darme valor para continuar en este largo camino, entre derrotas y victorias, tristezas y alegrías esto es un peldaño más de un largo trayecto que quiero avanzar junto a las personas que amo.

JAVIER

AGRADECIMIENTO

En primera instancia quiero agradecer a esas personas que estuvieron junto a mí no solo en la parte académica sino en cada instante de mi vida, como son mis padres Edison y Marcia, a mis hermanos Paola, Diego y Gabriela y además a mis sobrinos Ismael, Gabriel, Ana Paula, Martin y Juan José. También agradezco a todos mis compañeros de clase, amigos y amigas en especial a Alex, Luca, Darío, Darwin, Marcelo, Cristian y Javier, quienes me han apoyado en el transcurso de estos últimos años. Finalmente quiero agradecer a mis tutores, el Ingeniero David Rivas y el Ingeniero Marcelo Álvarez, quienes, mediante sus enseñanzas me ayudaron a formarme académicamente.

EDISON

Gracias a todas las personas que han estado conmigo en todo el trayecto de mi vida, a Dios y a mis padres Aníbal y Nélida que siempre estuvieron motivándome y aconsejándome con cariño y paciencia para llegar a ser la persona que soy ahora, a mi hermana Gaby que al igual que mis padres siempre me apoyó, a mis familiares que de una u otra forma pusieron su grano de arena y están presentes en todo el trabajo culminado y a mis amigos que me acompañaron en todos estos años en donde hubo buenos momentos y lo supimos disfrutar aún en los malos los pudimos superar. Muchas gracias de verdad, muchas gracias de corazón porque una persona no puede trabajar sola y siempre necesita de la compañía y el apoyo de alguien más y yo agradezco infinitamente por la familia y amigos que tengo ahora. Gracias.

JAVIER

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto centra su diseño y desarrollo en un sistema de control para un brazo robótico diseñado en la Universidad de las Fuerzas Armadas ESPE extensión Latacunga donde utiliza servomotores Dynamixel y una estructura de soporte implementada por la misma universidad; el software empleado para dicho sistema es Python que por sus ventajas y características, brinda al proyecto fiabilidad y facilidad de comunicación entre el sistema robótico que consta de un brazo de 6 grados de libertad mediante la utilización de un computador para realizar las funciones de mecanizado y paletizado, ambas procesadas y estructuradas dentro del algoritmo de control; al realizar todo este proyecto en arquitectura libre permite a los usuarios observar los códigos de las librerías y analizarlas con la finalidad de mejorarlas o modificarlas para futuros proyectos en este campo. Esto permitirá minimizar costos en varios sistemas robóticos ya que el software que viene en ellos son propietarios y representan grandes inversiones.

PALABRAS CLAVE — BRAZO ROBÓTICO; DYNAMIXEL; PYTHON; SERVOMOTOR; SOFTWARE LIBRE; RS-485; MECANIZADO; PALETIZADO; CINEMÁTICA INVERSA.

ABSTRACT

This project focuses on the design and development of a control system for a robotic arm designed at the University of the Armed Forces ESPE Extension Latacunga where using Dynamixel servo motors and a support structure implemented by the university; the software used for this system is Python that for their advantages and characteristics, provides the draft reliability and ease of communication between the robotic system that is an arm with 6 degrees of freedom with a computer to perform the functions of palletizing and machining, both structured and processed in the control algorithm; to make this whole project in a free architecture allows users to observe the codes libraries and analyze them with the finality to improve or modify them for future projects in this field. This will minimize costs in various robotic systems because the software that come with them are owners and represent large investments.

KEYWORDS — ROBOTIC ARM; DYNAMIXEL; PYTHON; SERVOMOTORS; FREE SOFTWARE; RS-485; MACHINING; PALLETIZING; INVERSE KINEMATICS.

INTRODUCCIÓN

CAPÍTULO 1

Explicación de objetivos, antecedentes y descripción del proyecto. Introducción al marco teórico de todo lo necesario para la realización del proyecto además de un resumen sobre el lenguaje de programación Python.

CAPÍTULO 2

Diseño del sistema robótico tanto en estructura como su algoritmo de control. Explicación de la forma de conexión, configuración y alimentación de los servomotores y la red formada por los mismos. Planteamiento del programa y las funciones a codificar.

CAPÍTULO 3

Implementación del sistema y codificación del algoritmo de control, explicación de las librerías realizadas para las funcionalidades de mecanizado y paletizado.

CAPÍTULO 4

Análisis de las pruebas realizadas del sistema de control programado sobre el brazo robótico, realización de diferentes ejercicios en el mecanizado de figuras y paletizado de objetos.

CAPÍTULO 1

1 GENERALIDADES

En este primer capítulo se detalla el propósito y objetivos que tiene el proyecto a realizarse, además de explicar la investigación realizada sobre el funcionamiento de un brazo robótico, servomotores, el tipo de comunicación que enlazará todo el sistema; asi como también la programación de forma general en Python y las ventajas que tiene con respecto a otros tipos de lenguaje.

1.1 Objetivo general.

Desarrollar un algoritmo en lenguaje Python para el control de un brazo robótico de 6 grados de libertad con funcionalidad de mecanizado y paletizado para el laboratorio de electrónica.

1.2 Objetivos específicos.

- Investigar las características técnicas de los servomotores que usa el brazo robótico.
- Conocer el uso del Software Python para el desarrollo del proyecto
- Ensamblar un brazo robótico utilizando servomotores Dynamixel y una estructura diseñada e implementada en la Universidad de las Fuerzas Armadas ESPE Latacunga.
- Realizar pruebas con software probado para certificar el correcto funcionamiento del robot en la parte mecánico-electrónica.
- Probar la red de servomotores implementada en el brazo robótico.
- Elaborar librerías en lenguaje Python para el uso del brazo robótico
- Diseñar la lógica de control con la cual se pueda tener movimientos deseados del brazo robótico que coincidan con la información

contenida por la interfaz gráfica con el fin de ejecutar un mecanizado y paletizado.

- Realizar un informe que contenga los resultados del proyecto que se complementen con proyectos anteriores.

1.3 Descripción del proyecto.

1.3.1 Introducción

En el campo de la robótica se puede notar grandes avances a lo largo del tiempo, vemos como robots desempeñan labores que para el humano se han vuelto muy difíciles de realizar y necesitan de estos para ingresar a ambientes peligrosos o inaccesibles, o a su vez en el campo médico donde existe la necesidad de prótesis robóticas para pacientes discapacitados; este tipo de desarrollo e investigación requiere de grandes esfuerzos e inversiones financieras, es por esa razón que los software se han vuelto propietarios y difíciles de adquirir ya que cuestan una gran cantidad de dinero, debido a esto se ha optado en investigar proyectos basados en software libre los cuales reducen costos y dan la facilidad de que cualquier persona pueda acceder a estos programas.

La necesidad de obtener más ventajas y viabilidad en el uso de sistemas robóticos empuja a encontrar maneras de manipular los mismos a través del uso de software libre, ya que con el avance de la investigación en lenguajes de programación, hacen posibles el desarrollo de nuevas herramientas, extensiones, librerías, etc. que facilitan un lazo de comunicación entre un robot y una PC por medio de un algoritmo codificado en software libre.

Por ejemplo el software libre Python es un lenguaje de programación útil y fácil de codificar, además de que tiene extensiones que son de gran ayuda para las aplicaciones en las que se quiera desarrollar un algoritmo. En este caso el establecimiento de la comunicación con una red de

servomotores Dynamixel y el desarrollo de una interfaz gráfica que permita la visualización del proceso es necesario.

El presente proyecto está afín a esta ideología por lo que se requiere desarrollar las librerías necesarias utilizando el lenguaje de programación en la plataforma Python para que una red de servomotores de un brazo robótico realice las funciones de mecanizado y paletizado, siendo el objetivo primordial del proyecto reducir costos de adquisición en software.

1.3.2 Antecedentes

La robótica industrial ha surgido por la necesidad de realizar procesos que han resultado repetitivos, tediosos e incluso difíciles para la labor humana, donde la asistencia de un mecanismo robótico y automatizado pudiere ser la solución para tales fines. El conocimiento necesario para poder entender e implementar un prototipo robótico se ve reflejado en el dominio de tres áreas fundamentales: la electrónica, la mecánica y la programación.

Por su parte la ESPE se ha inmerso en este mundo de la robótica y ha participado y realizado proyectos muy relacionados en este tema, por ejemplo en el año 2008, realizo la compra de 3 brazos robóticos a la empresa KUKA Robotics, los mismos que han sido destinados a prácticas de laboratorio para los alumnos de las Carreras de Electrónica, Mecatrónica y Electromecánica de la Universidad así como también de la Matriz.

Esto ha permitido cumplir con las expectativas esperadas en cuanto a uso, aplicabilidad y tecnología de los equipos antes mencionados; una vez analizado y comprendido el funcionamiento de los brazos robóticos, se propone realizar el modelo, diseño y construcción de un brazo robótico con 6 grados de libertad programable, utilizando software libre, teniendo como funcionalidades: el mecanizado y paletizado de objetos, con lo cual la

ESPE tendría la oportunidad de implantar robots didácticos e industriales diseñados y construidos en el país con tecnología actual y de bajo costo.

Los robots de la marca Dynamixel han tenido un gran surgimiento debido a la mecánica y electrónica que incluyen, pero, la programación se limita a las instrucciones predefinidas del software que sirve para su control. El desarrollo de un controlador libre sería de gran ayuda para facilitar el entendimiento e incrementar el desarrollo de aplicaciones que se puede dar a este tipo de robots, para poder así, explotar todas sus características.

Además poseen un software controlador propio, que convierte a todo el sistema en una caja negra, donde las líneas de instrucción para el control de motores se hacen transparentes y por tanto, se desconocen. Para tratar de pulir estas limitaciones, se cree conveniente diseñar un controlador de servomotores mediante una aplicación implementada en software libre.

1.3.3 Descripción resumida del proyecto

Uno de los puntos que más interés ha presentado la robótica industrial son los brazos robóticos, capaces de duplicar movimientos que los generaría un miembro humano, pero orientado a aplicaciones industriales en ambientes donde se expusiera a peligro al operario o tareas difíciles de ejecutar. La mayoría de robots construidos con estos fines son propietarios y difícilmente su electrónica, mecánica y algoritmo de control se distribuyen de manera que se pudiera copiarlos, dificultando así su entendimiento y teniendo como resultado la necesidad de optar por una compra que involucraría una gran cantidad de dinero.

De esta manera la realización del proyecto sobre el desarrollo de un sistema de control para un brazo robótico de 6 grados de libertad utilizando software libre permitirá resolver el problema financiero suscitado en el

párrafo anterior. Además de que permitirá a la comunidad politécnica acceder a la información teórica y práctica desarrollada para el avance de proyectos futuros relacionados con la robótica y el software libre.

El fin del proyecto realizado es el diseño de un sistema de control utilizando software libre que en este caso es el programa Python; se requiere que la interfaz de comunicación entre el computador y los servomotores interprete los datos enviados por el programa usando las diferentes librerías creadas para que la red de servomotores, que se encuentra en el brazo robótico, se muevan coordinadamente y puedan realizar funciones de mecanizado y paletizado.

Todos estos comandos que se encuentran en el programa deben ser lo más precisos y legibles posibles con el fin de crear una interfaz gráfica amigable con el usuario capaz de enlazarse con el software para el control del brazo robótico desde un computador. El brazo robótico tendrá la capacidad de interpretar las señales que serán enviadas por comunicación serial para moverse de acuerdo a estas.

1.4 Brazo robótico.

Los robots industriales son dispositivos mecánicos de funciones múltiples programables diseñados para mover materiales, piezas, herramientas o dispositivos especializados a través de movimientos variables programados para realizar varias tareas.

Un sistema robótico industrial no sólo incluye los robots industriales, sino también a todos los dispositivos y/o sensores necesarios para que el robot pueda realizar sus tareas, así como la secuenciación o monitoreo de las interfaces de comunicación. Los robots son generalmente utilizados para realizar tareas inseguras, peligrosas e incluso repetitivas para el operador. Ellos tienen muchas funciones diferentes, tales como el manejo de materiales, montaje, soldadura, carga y descarga de una máquina o

herramienta y de funciones como: pintura, pulverización, etc. La mayoría de los robots están configurados para una operación mediante la técnica de enseñanza y repetición. En este modo, un operador entrenado (programador) por lo general utiliza un dispositivo de control portátil (consola de aprendizaje) para enseñar a un robot su tarea manualmente. Las velocidades del robot durante estas sesiones de programación son lentas.

Esta técnica incluye consideraciones de seguridad necesarias para operar el robot correctamente y utilizarlo automáticamente junto con otros equipos periféricos, esto se aplica a los sistemas robóticos industriales fijos únicamente, pero existen sistemas que están excluidos de esta técnica.

Un brazo robótico es un tipo de brazo mecánico, normalmente programable, con funciones parecidas a las de un brazo humano; este puede ser la suma total del mecanismo o puede ser parte de un robot más complejo. Las partes de estos manipuladores o brazos son interconectadas a través de articulaciones que permiten, tanto un movimiento rotacional (tales como los de un robot articulado), como un movimiento translacional o desplazamiento lineal. [1]

El efector final, o mano robótica, se puede diseñar para realizar cualquier tarea que se desee como puede ser soldar, sujetar, girar, etc., dependiendo de la aplicación. Por ejemplo los brazos robóticos en las líneas de ensamblado de la industria automovilística realizan una variedad de tareas tales como soldar y colocar las distintas partes durante el ensamblaje.

En algunas circunstancias, lo que se busca es una simulación de la mano humana, como en los robots usados en tareas de desactivación de explosivos o en el campo de la medicina como prótesis para personas que han perdido alguna de sus extremidades.

1.4.1 Tipos y clasificación de los robots

Los robots industriales están disponibles comercialmente en una amplia gama de tamaños, formas y configuraciones. Están diseñados y fabricados con diferentes configuraciones de diseño y un número diferente de ejes o grados de libertad. Estos factores de diseño de un robot influyen en su área de trabajo (el volumen de trabajo o de alcance en el espacio laboral). En la Figura 1. 1 se muestran varios diagramas de las diferentes configuraciones de diseño de un robot:

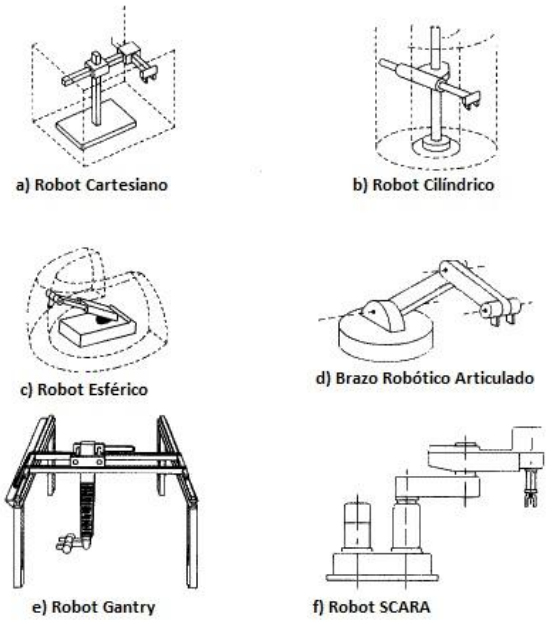


Figura 1. 1.- Configuraciones de diseño de un brazo robótico. a) Robot cartesiano; b) Robot cilíndrico; c) Robot esférico; d) Brazo robótico articulado; e) Robot Gantry; f) Robot SCARA.
Fuente: (OSHA, «INDUSTRIAL ROBOTS AND ROBOT SYSTEM SAFETY») [1]

- **Robot cartesiano:** Es un robot cuyo brazo tiene tres articulaciones prismáticas, cuyos ejes son coincidentes con los ejes cartesianos. Mostrado en la parte a) de la Figura 1. 1.

Debido a la forma que tiene se lo utiliza en trabajos de tomar y colocar.

- **Robot cilíndrico:** Usado para operaciones de ensamblaje, manipulación de máquinas herramientas, soldadura por punto, y manipulación en máquinas de fundición a presión. Es un robot cuyos ejes forman un sistema de coordenadas cilíndricas. Se lo puede apreciar en la parte b) de la Figura 1. 1.
- **Robot esférico o Robot polar:** Usados en la manipulación en máquinas herramientas, soldadura por punto, fundición a presión, máquinas de desbarbado, soldadura por gas y por arco. Es un robot cuyos ejes forman un sistema polar de coordenadas, tal como se muestra en la Figura 1. 1 parte c).
- **Robot SCARA:** Al igual que el robot cartesiano este tipo es usado para trabajos de “pick and place” (tomar y colocar), aplicación de impermeabilizantes, operaciones de ensamblado y manipulación de máquinas herramientas. Es un robot que tiene dos articulaciones rotatorias paralelas para proporcionar elasticidad en un plano. Se lo puede apreciar en el literal f) de la Figura 1. 1.
- **Robot articulado:** Usado para operaciones de ensamblaje, fundición a presión, máquinas de desbarbado, soldadura a gas, soldadura por arco, y pintado en spray. Es un robot cuyo brazo tiene como mínimo tres articulaciones rotatorias como se indica en la parte d) de la Figura 1. 1.
- **Robot paralelo:** Uno de sus usos es la plataforma móvil que manipula las cabinas de los simuladores de vuelo. Es un robot cuyos brazos tienen articulaciones prismáticas o rotatorias concurrentes.
- **Robot Antropomórfico**: Similar a una mano robótica. Se le da forma para que pueda sustituir a una mano humana, por ejemplo con dedos independientes incluido el pulgar.

1.4.2 Tipo de ruta generada.

Como ya se tiene conocimiento, un robot debe ser programado para que pueda realizar todas las tareas para las que fue diseñado, para esta labor existen diferentes plataformas de programación, como métodos, que cada una de ellas podrían albergar para transmitir esta información al robot construido. Los robots industriales pueden ser programados a distancia para realizar sus operaciones requeridas y pre-programados por tres métodos generados a través de diferentes técnicas de control los cuales son:

- **Ruta de acceso de punto a punto**. Los robots controlados de esta manera están programados para moverse de un punto discreto a otro dentro del área de trabajo del robot. En el modo automático de funcionamiento, la trayectoria exacta tomada por el robot puede variar ligeramente debido a las variaciones en la velocidad, geometrías de junta, y ubicaciones espaciales de punto. Esta diferencia de caminos es difícil de predecir y, por tanto, puede crear un potencial peligro para la seguridad personal y el equipo.
- **Ruta de acceso controlado**. La ruta de acceso o el modo de movimiento asegura que el extremo del brazo robótico seguirá un predecible (controlado) camino y orientación de cómo el robot se desplaza desde un punto a otro. Las transformaciones de coordenadas requeridas para esta administración de hardware se calculan por el ordenador del sistema de control del robot. Las observaciones que se derivan de este tipo de programación es menos probable que represente un riesgo para el personal y el equipo.
- **Ruta Continua**. Un robot cuya ruta de acceso se controla mediante el almacenamiento de una gran cantidad o sucesión de

puntos espaciales en la memoria durante una secuencia de enseñanza es un robot controlado de ruta continua. Durante este tiempo, y mientras el robot se mueve, los puntos de coordenadas en el espacio de cada eje se controlan continuamente en una base de tiempo fijo, por ejemplo, 60 o más veces por segundo, y se coloca en la memoria del ordenador del sistema de control. Cuando el robot se coloca en el modo automático de funcionamiento, el programa se repite de la memoria y se genera un camino duplicado. [1]

1.4.3 Componentes del robot.

Los robots industriales tienen cuatro componentes principales: la unidad mecánica, la fuente de energía, sistema de control y las herramientas. Cada uno de estos componentes funciona como una sola unidad para que todo el robot pueda direccionarse o posicionarse en las coordenadas dictaminadas por el operario. En forma general se muestra estos cuatro componentes en la Figura 1. 2.

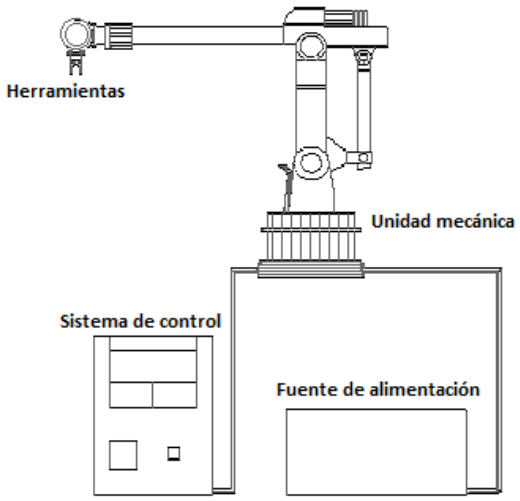


Figura 1. 2.- Principales componentes de un robot industrial.
Fuente: (OSHA, «INDUSTRIAL ROBOTS AND ROBOT SYSTEM SAFETY») [1]

a. Unidad Mecánica.

Es la estructura física del robot, la que se compone de varios elementos fabricados con disposiciones para el apoyo a la vinculación mecánica y articulaciones, guías, actuadores (lineales o rotativos), válvulas de control y sensores. Las dimensiones físicas, el diseño y la capacidad de transporte de peso dependen de los requisitos de aplicación.

b. Fuentes de alimentación

Este componente es el encargado de proporcionar energía a varios actuadores del robot y sus controladores, el tipo de energía empleado puede ser: neumática, hidráulica, o eléctrica. Los robots accionados eléctricamente son los más frecuentes en la industria y utilizan tanto corriente eléctrica AC o DC para suministrar energía a los mecanismos de accionamiento electromecánicos con motor y sus respectivos sistemas de control. El control de movimiento es mucho mejor, y en caso de emergencia un robot eléctrico puede ser detenido o se apaga de forma más segura y más rápida que los que usan energía neumática o hidráulica. Las unidades del robot son generalmente combinaciones mecánicas accionados por algún tipo de energía, y la selección se basa por lo general en los requisitos de la aplicación. Por ejemplo, la energía neumática (aire a baja presión) se utiliza generalmente para los robots de carga de bajo peso. La transmisión de energía hidráulica (aceite de alta presión) se utiliza generalmente para aplicaciones media-altas de fuerza o de peso, o donde un control más suave de movimiento se puede lograr de forma más sencilla que con la energía neumática.

c. Sistemas de Control

En la actualidad cualquiera de los equipos auxiliares o microprocesadores embebidos se utilizan para casi todo el control de robots industriales. Estos realizan todas las funciones computacionales

necesarias, así como la interfaz y el control de sensores, pinzas, herramientas y otros equipos periféricos asociados. El sistema de control realiza las funciones de secuenciación y memoria necesarios para la detección on-line, la ramificación, y la integración de otros equipos.

Se puede incorporar la capacidad de autodiagnóstico para la solución de problemas y el mantenimiento, mientras que el tiempo de inactividad del sistema del robot se reduce considerablemente. Algunos controladores de robots tienen el espacio de memoria suficiente en términos de: la capacidad de cálculo, de memoria y de entrada-salida para servir también como controladores del sistema y manejar muchas otras máquinas y procesos.

La programación de controladores y sistemas de robots no se ha estandarizado por la industria de la robótica; por lo tanto, los fabricantes utilizan sus propios lenguajes de programación propietarios que requieren una formación especial del personal.

d. Herramientas

Los robots manipuladores suelen llevar algún dispositivo que se une a la muñeca del robot para realizar una tarea determinada. Se pueden dividir en dos tipos: las pinzas, diseñadas para la manipulación, transporte y unión de objetos, y las herramientas, cada una de ellas diseñada para una función específica como soldadura, pintura, taladrar, etc.

1.4.4 Programación de un robot por métodos de enseñanza.

Un programa consta de los pasos de comando individuales que establecen la posición o función a realizar, junto con otros datos de información tales como la velocidad, torque, detención o tiempos de retardo, activación del dispositivo de entrada, activación del dispositivo de salida, ejecutar, detenerse, etc.

Cuando se establezca un programa para el robot, es necesario establecer una relación física o geométrica entre el robot y otros equipos o el área de trabajo en donde desempeñará su funcionamiento. Establecer estos puntos de coordenadas precisamente dentro del área de trabajo del robot es necesario para controlar el robot manualmente y enseñarle físicamente los puntos de coordenadas. Para hacer esto, así como determinar otra información de programación funcional, se utilizan tres técnicas de enseñanza o de programación diferentes:

a. Programación guiada por el operador.

Este método de enseñanza utiliza una unidad de programación patentado (el control del robot se coloca en modo de "enseñar") lo que permite al personal capacitado para dirigir el robot físicamente a través de la secuencia deseada de eventos mediante la activación apropiada de un interruptor. Los datos de posición y la información funcional se "enseña" al robot, y un nuevo programa está escrito (Figura 1. 3). La unidad de programación puede ser la única fuente por la cual se establece un programa, o puede ser usado en conjunción con una consola de programación adicional y/o el controlador del robot.

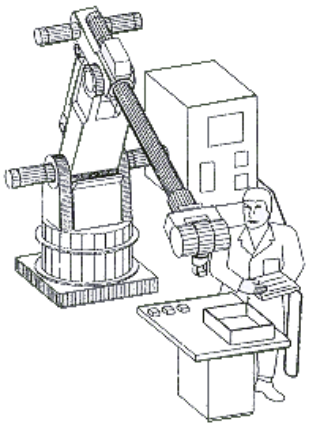


Figura 1. 3.- Programación guiada por el operador.
Fuente: (OSHA, «INDUSTRIAL ROBOTS AND ROBOT SYSTEM SAFETY») [1]

b. Programación por recorrido.

Una persona que hace la programación tiene contacto físico con el brazo robótico y de hecho gana el control y lleva el brazo del robot a través de las posiciones deseadas dentro del área de trabajo (Figura 1. 4).

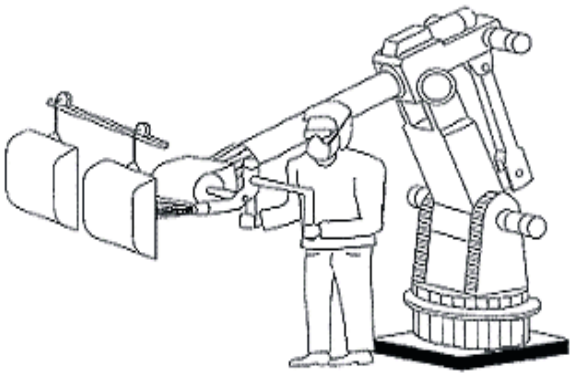


Figura 1. 4.- Programación por recorrido.
Fuente: (OSHA, «INDUSTRIAL ROBOTS AND ROBOT SYSTEM SAFETY») [1]

Durante este tiempo, el controlador del robot escanea y almacena los valores de coordenadas en una base de tiempo fijo. Cuando el robot se coloca después en el modo automático de funcionamiento, estos valores y otra información funcional se reproducen y el programa se ejecuta como se le enseñó al robot.

c. Programación off-line (fuera de línea).

La programación establece la secuencia necesaria de funcionamiento y las medidas de posición requeridas se escriben en una consola de ordenador a distancia (Figura 1. 5). Desde la consola que está distante del robot y su controlador, el programa escrito tiene que ser transferido al controlador del robot y los datos de posición precisos establecidos para alcanzar las coordenada actuales que se indicaron en la información para el robot y otros equipos. Después de que el programa ha

sido completamente transferido al controlador del robot, pueden ser usadas las técnicas tanto por programación guiada por el operador o programación por recorrido para la obtención de la información de las coordenadas de posición actual de los ejes del robot.

En la programación de robots con cualquiera de las tres técnicas mencionadas anteriormente, por lo general se requiere que el programa sea verificado por ligeras modificaciones en la información de posición hechas.

Este procedimiento se denomina programa de retoque y se lleva a cabo normalmente en el modo de operación de aprendizaje. El programador conduce o lleva el robot a través de los pasos programados manualmente.

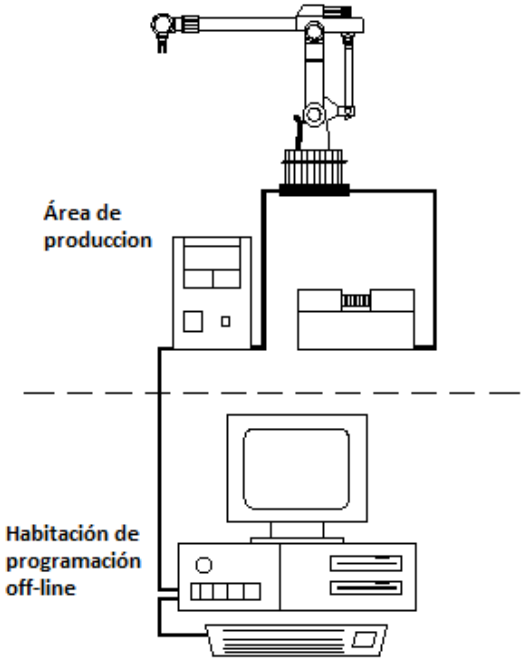


Figura 1. 5.- Programación off-line.

Fuente: (OSHA, «INDUSTRIAL ROBOTS AND ROBOT SYSTEM SAFETY») [1]

1.4.5 Grados de libertad.

Independientemente de la configuración de un robot, el movimiento a lo largo de cada eje se traducirá en ya sea una rotación o un movimiento de traslación. El número de ejes de movimiento y su disposición se denominan grados de libertad, junto con su secuencia de operación y la estructura, puede permitir movimientos del robot a cualquier punto dentro de su cobertura de trabajo.

Los robots tienen tres movimientos del brazo (arriba-abajo, dentro-fuera, de lado a lado). Todas sus áreas de cobertura se encuentran detalladas en la Figura 1. 6.

Además, pueden tener un máximo de tres movimientos adicionales de la muñeca en el extremo del brazo robótico: de guiñada (lado a lado), tono (arriba y abajo), y de rotación (en sentido horario y en sentido anti horario).

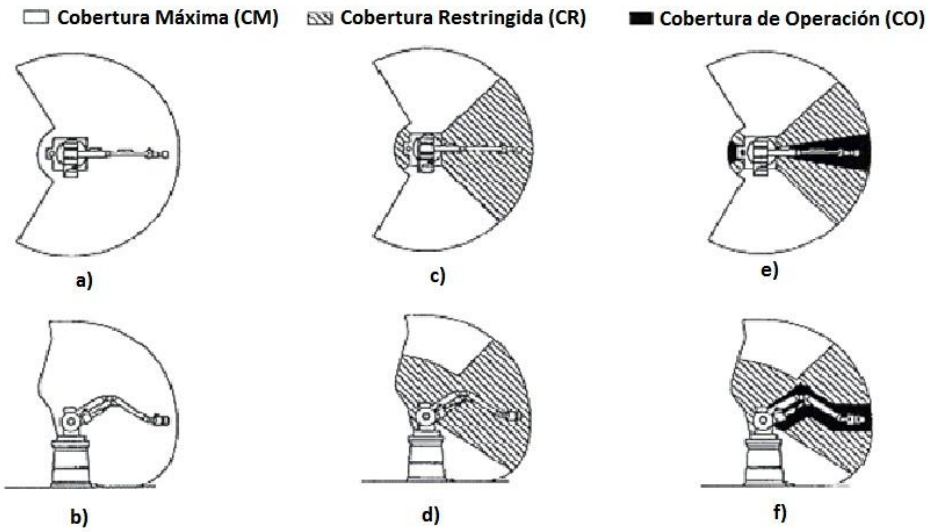


Figura 1. 6.- Áreas de cobertura de un brazo robótico. a) CM ejes XY; b) CM ejes XZ; c) CR ejes XY; d) CR ejes XZ; e) CO ejes XY; f) CO ejes XZ.
Fuente: (OSHA, «INDUSTRIAL ROBOTS AND ROBOT SYSTEM SAFETY») [1]

1.4.6 Robots fabricados, refabricados y reconstruidos.

Todos los robots deben cumplir con los requisitos mínimos de diseño para asegurar un funcionamiento seguro por el usuario. Hay que tener en cuenta una serie de factores en el diseño y construcción de los robots con los estándares de la industria. Si los robots más viejos u obsoletos son reconstruidos o remanufacturados, deben actualizarse para ajustarse a los estándares actuales de la industria.

1.4.7 Instalación

Un robot o un sistema robótico deben ser instalados por los usuarios en conformidad con las recomendaciones del fabricante y en conformidad con los estándares aceptados por la industria. Dispositivos y prácticas de salvaguardia temporales deben ser utilizados para minimizar los riesgos asociados con la instalación de nuevos equipos. Las instalaciones, equipos periféricos y las condiciones de funcionamiento que se deben considerar son:

- Especificaciones de instalación;
- Las instalaciones físicas;
- Instalaciones eléctricas;
- Acción de equipos periféricos integrados con el robot;
- Los requisitos de identificación;
- Requisitos de control y de paro de emergencia; y
- Los procedimientos o condiciones especiales de operación del robot.

Para garantizar unas prácticas de funcionamiento confiables y segura instalación de robots y sistemas robóticos, se recomienda los requisitos mínimos de la sección 5 de la norma ANSI/RIA R15.06-1992, Instalación de Robots y sistemas robotizados. [1]

1.5 Cinemática Inversa

En el campo de la robótica es la técnica que permite determinar el movimiento de una cadena de articulaciones para lograr que un actuador final se ubique en una posición concreta. El cálculo de la cinemática inversa es un problema complejo que consiste en la resolución de una serie de ecuaciones cuya solución normalmente no es única.

El objetivo de la cinemática inversa es encontrar los valores que deben tomar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial. Depende de la configuración del robot y debido a esto pueden existir soluciones múltiples. Siempre que se especifica una posición de destino y una orientación en términos cartesianos, debe calcularse la cinemática inversa del sistema para poder encontrar los ángulos de articulación requeridos.

El movimiento de una cadena cinemática ya sea si es un robot o un personaje animado es modelado por ecuaciones cinemáticas propias de la misma cadena. Estas ecuaciones definen la configuración de la cadena en términos de sus parámetros. Por ejemplo las fórmulas de la cinemática inversa permiten el cálculo de los parámetros de unión del brazo de un robot para levantar un objeto.

1.6 Mecanizado

Es un proceso de fabricación que comprende un conjunto de operaciones de conformación de piezas mediante la eliminación de material, ya sea por arranque de viruta o por abrasión. También en algunas zonas de América del Sur es utilizado el término maquinado aunque debido al doble sentido que puede tener este término (urdir o tramar algo) convendría usar el primero.

Se realiza a partir de productos semielaborados como lingotes, tochos u otras piezas previamente conformadas por otros procesos como moldeo o forja. Los productos obtenidos pueden ser finales o semielaborados que requieran operaciones posteriores.

El mecanizado en el campo de la robótica viene definido por la capacidad de un sistema robótico de fabricar piezas en algún tipo de material ya sea por el arranque o abrasión del mismo. Pero la parte fundamental o básica de este procedimiento es el recorrido que debe realizar el sistema robótico en el material para obtener una pieza de acuerdo a un modelo diseñado ya sea de manera digital o manual; es por eso que este proyecto detalla como un brazo robótico ensamblado desde cero y desarrollado todo su sistema de control mediante software libre es capaz de tomar una imagen desde un ordenador y plasmarla sobre una superficie, el mismo principio puede ser utilizado con otras herramientas en la pinza del robot con el fin de recrear el funcionamiento de una fresadora o caladora.

Todo empieza desde la programación, en cómo hacer que el brazo reconozca un diseño de una figura; los lenguajes de programación han alcanzado un desarrollo muy grande que beneficia a todo programador mediante el uso de librerías ya implementadas o extensiones en sus plataformas, una de ellas es el procesamiento de imágenes, donde Python posee muchas de ellas las cuales permiten que el programador cargue imágenes y las procese de acuerdo a sus necesidades.

El proceso de mecanizado para este brazo robótico se centra en transformar una imagen a mapa de bits es decir en formato BMP y cada pixel representará una coordenada en el plano X, Y; por lo que se requiere de la cinemática inversa para hacer que el sistema robótico recorra estas coordenadas y trace la figura que se envió desde el computador.

1.7 Paletizado

Paletizar consiste en agrupar sobre una superficie (paleta o estiba) una cierta cantidad de productos, con la finalidad de conformar una unidad de manejo que pueda ser transportada y almacenada con el mínimo esfuerzo y en una sola operación.

En este entorno surgen los sistemas de paletizado automático mediante robots, que sirven para la paletización automática en aquellas producciones con altas cadencias o bien cuando las condiciones de trabajo, ergonomía, peso de los productos, condiciones ambientales o de higiene, requieren que el trabajo no sea realizado directamente con la manipulación o intervención del hombre.

Según las unidades producidas y número de referencias que se desea paletizar, se puede elegir entre diferentes tipos de sistemas. Por otro lado, es determinante el sistema de toma de los productos, ya que una toma del producto individual o de varios productos va a determinar la eficiencia final del sistema de paletizado automático.

De esta forma debemos tener en cuenta las siguientes consideraciones:

- Los robots articulados o antropomórficos están indicados para manipular un número de unidades más reducido, en un menor espacio, en comparación con el tipo pórtico. La configuración de los palets se desarrolla alrededor del robot, no siendo lineal como ocurre en el pórtico. La regularidad o cadencia alcanzada con un robot antropomórfico suele ser mayor, en cuanto que los recorridos son menores y están más optimizados.
- Un robot pórtico (Gantry) ofrece un movimiento lineal en 3 ejes (x, y, z, definidos como alto, ancho y largo). Esta configuración permite disponer de un mayor número de estaciones de paletizado en línea,

por lo que permite paletizar más unidades para un mismo espacio y contemporáneamente con un solo robot. El número de ciclos varía en función de los recorridos a realizar, dimensiones producto y mosaico, entre otros. De esta forma, el pórtico es más flexible en cuanto al espacio útil y un solo cabezal puede recorrer distancias de 30m x 5m x 2,5m.

- La concepción de la pinza del robot depende de varios factores; combinaciones a formar, número de cajas en toma múltiple, tipo de cartón de las cajas, fragilidad del producto, distribución de su peso en el interior, calidad del precintado etc.

De manera general, podemos clasificar los sistemas de toma en:

- Pinza de palas: Formada por la combinación de palas móviles y fijas.
- Ventosas de vacío: Cada ventosa incorpora un sensor de apertura y cierre que permite realizar vacío únicamente a las ventosas que se encuentran sobre las cajas.
- Pinza de garras: Usualmente utilizadas para la toma de palets.
- Pinzas mixtas: Basadas en la combinación entre los sistemas anteriores, según la aplicación.

En el proyecto presentado, al igual que en su funcionalidad de mecanizado, el paletizado se lo realizará con un robot articulado, por medio del sistema de control desarrollado en software libre será capaz de tomar objetos en las coordenadas preestablecidas por el operario, a través del método de enseñanza que se haya utilizado para el sistema robótico, y los llevará a otro punto de coordenadas en el espacio, al realizar el mismo proceso en forma regular y sincronizada se tendrá como resultado un paletizado automático que puede ser fácilmente implementado o incluido en procesos industriales.

1.8 Servomotores.

1.8.1 Definición y estructura

Un servomotor (o servo) es un motor de corriente continua que tiene la capacidad de ser controlado en posición. Es capaz de ubicarse en cualquier posición dentro de un rango de operación por lo general de 180º y mantenerse estable en dicha posición.

Los servos se suelen utilizar en robótica, automática y modelismo (vehículos por radio-control, RC) debido a su gran precisión en el posicionamiento. En general, los servos suelen estar compuestos por 4 elementos fundamentales:

- Motor de corriente continua (DC): Es el elemento que le brinda movilidad al servo. Cuando se aplica un potencial a sus dos terminales, este motor gira en un sentido a su velocidad máxima. Si el voltaje aplicado sus dos terminales es inverso, el sentido de giro también se invierte.
- Engranajes reductores: Es un tren de engranajes que se encarga de reducir la alta velocidad de giro del motor para incrementar su capacidad de torque o par-motor.
- Sensor de desplazamiento: Suele ser un potenciómetro colocado en el eje de salida del servo que se utiliza para conocer la posición angular del motor.
- Circuito de control: Es una placa electrónica que implementa una estrategia de control de la posición por realimentación. Para ello, este circuito compara la señal de entrada de referencia (posición deseada) con la posición actual medida por el potenciómetro. La diferencia entre la posición actual y la deseada es amplificada y utilizada para mover el motor en la dirección necesaria para reducir el error.

En la Figura 1. 7 se detallan todos los componentes que se encuentran en un servomotor desde la carcasa hasta su elemento final del eje.

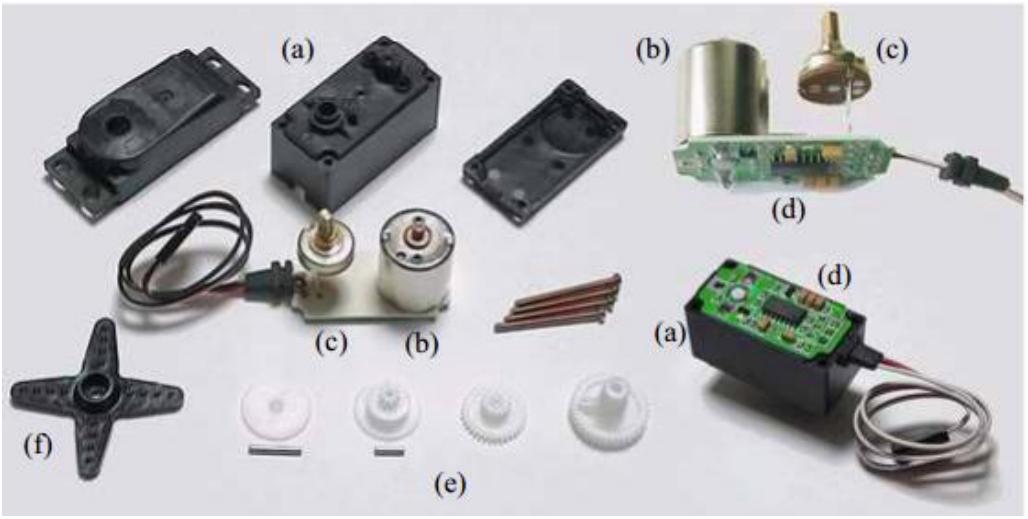


Figura 1. 7.- Componentes de un servomotor a) carcasa; b) motor DC; c) potenciómetro; d) circuito de control; e) tren reductor; f) brazo (elemento terminal del eje).

Fuente: (F. A. C. Herías, «Servomotores») [2]

1.8.2 Tipologías

Existen dos tipos de servos: analógicos y digitales. Ambos tipos de servos son iguales a nivel de usuario: tienen la misma estructura (motor DC, engranajes reductores, potenciómetro y placa de control) y se controlan con las mismas señales PWM. La principal diferencia entre ellos radica en la adición de un microprocesador en el circuito de control de los servos digitales. Este microprocesador se encarga de procesar la señal PWM de entrada y de controlar el motor mediante pulsos con una frecuencia 10 veces superior a los servos analógicos.

1.8.3 Servo y Nonservo

Dentro de lo que es la robótica industrial tenemos dos tipos de robots de acuerdo a los servomotores utilizados por lo que los robots

industriales pueden ser servo o nonservo controlados. Los robots servo son controlados mediante el uso de sensores que supervisan continuamente ejes del robot y los componentes asociados para la posición y la velocidad. Esta retroalimentación se compara a la información precargada que ha sido programada y almacenada en la memoria del robot. Los robots nonservo no tienen la capacidad de retroalimentación, y sus ejes se controlan mediante un sistema de topes mecánicos y finales de carrera. [1]

1.8.4 Características.

Los servos se utilizan frecuentemente en sistemas de radiocontrol y en robótica, pero su uso no está limitado a estos, también son de vital importancia en aeromodelismo, entre otros.

a. Servo analógico para modelismo

Estos servomotores se componen, en esencia, de un motor de corriente continua, un juego de engranajes para la reducción de velocidad, un potenciómetro ubicado sobre el eje de salida (que se usa para conocer la posición) y una plaqueta de circuito para el control. Si lo que se desea controlar es la posición de un servomecanismo, como en este caso, en lugar de un tacómetro (que es para medir velocidad) se necesita un encoder de posición. Si se habla de un servo cuyo movimiento es giratorio, será necesario un encoder (un detector que codifica la posición) que entregue un valor diferente a su salida según cual sea su posición en grados.

Los servos que se usan en modelismo son de este tipo. Como se mencionó, por lo general poseen un motor de DC, que gira a velocidad alta, una serie de engranajes para producir la reducción de velocidad de giro e incrementar su capacidad de torque, un potenciómetro conectado al eje de salida (que haría el papel del encoder) y un circuito de control de la realimentación. Estos servos reciben la señal por tres cables: dos de ellos

destinados a la alimentación para el motor y la placa del circuito del control, y una señal de control que determina la posición que se requiere. La alimentación de estos servos es normalmente, de entre 4,8v y 6v.

El estándar de la señal de control para todos los servos de este tipo, es un pulso de onda cuadrada de 1,5ms que se repite a un ritmo de entre 10ms a 22ms. Mientras el pulso se mantenga en ese ancho, el servo se ubicará en la posición central de su recorrido. Si el ancho de pulso disminuye, el servo se mueve de manera proporcional hacia un lado. Si el ancho de pulso aumenta, el servo gira hacia el otro lado. Generalmente el rango de giro de un servo de éstos cubre entre 90° y 180° de la circunferencia total, o un poco más, según la marca y modelo.

b. Servo digital para modelismo

Los servos digitales tienen, al igual que los analógicos, un motor de corriente continua, un juego de engranajes reductores, un potenciómetro para la realimentación de posición y una electrónica de control embebida dentro del servo. La diferencia está en la placa de control, en la que han agregado un microprocesador que se hace cargo de analizar la señal, procesarla y controlar el motor.

La diferencia más grande de rendimiento está en la velocidad a la que reacciona el servo a un cambio en la señal. En un mismo lapso, el servo digital puede recibir cinco o seis veces más pulsos de control que un analógico. Como resultado la respuesta del servo a un cambio en la orden de posición es mucho más veloz. Este ritmo mayor de pulsos también produce mejoras en el rendimiento electromecánico del motor (mayor velocidad y más fuerza). Esto se debe a que en cualquier servo (de ambos tipos) el motor recibe, para su control, una alimentación conmutada. En los servos analógicos, la señal está conmutada a un ritmo de entre 10ms y 22ms. Si el ajuste que se requiere es muy pequeño (un ángulo pequeño de

giro), los pulsos son muy delgados y están muy separados (10ms a 22ms). La integración de estos pulsos es la que da la alimentación de potencia al motor, y en consecuencia la que lo hace mover. Una integración de pulsos delgados y muy separados puede dar resultados erráticos. Suele ocurrir que cuando llega el otro pulso, el motor se ha pasado de la posición y deba reajustarse, algo que ocurre constantemente. En los servos digitales la señal llega mucho más seguido y por esto la integración es más estable y la variación de corriente de control es más firme.

En los servos digitales, la señal está separada por unos 3,3ms. La separación entre pulsos varía en cada marca de servo digital, pero el ritmo de llegada de los pulsos es de al menos 300 veces por segundo versus 50 a 100 veces por segundo en un analógico.

La ventaja de los digitales se reduce un poco cuando se habla de consumo (algo muy importante en, por ejemplo, un avión radio-controlado, pero también en los robots), ya que el consumo del circuito y de los ajustes más continuados produce un gasto mayor de energía, y también un mayor desgaste del motor. Los servos digitales son capaces de memorizar parámetros de programación, que varían de acuerdo a cada fabricante pero en general son:

- Se puede programar el sentido de giro como "normal" o "inverso".
- Se puede variar la velocidad de respuesta del servo.
- Se puede programar una posición central (o posición neutra) diferente, sin afectar los radios de giro.
- Se pueden determinar diferentes topes de recorrido para cada lado.
- Es posible programar qué debe hacer el servo en caso de sufrir una pérdida de señal.

- Es posible programar la resolución, es decir cuánto se mueve el control en el radio sin obtener un movimiento en el servo.

Estos valores pueden ser fijados en los servomotores utilizando dispositivos destinados a la programación, que son específicos para cada marca.

1.8.5 Funcionamiento.

Los servos disponen de tres cables (Figura 1. 8): dos cables de alimentación (positivo y negativo/masa) que suministran un voltaje 4.8v – 6v y un cable de control que indica la posición deseada al circuito de control mediante señales PWM (“Pulse Width Modulation”).

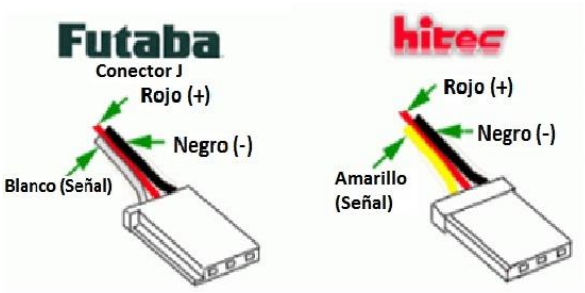


Figura 1. 8.- Código de colores de cables de los principales fabricantes de servos.

Fuente: (F. A. C. Herías, «Servomotores») [2]

Las señales PWM utilizadas para controlar los servos están formadas por pulsos positivos cuya duración es proporcional a la posición deseada del servo y que se repiten cada 20ms (50Hz). Todos los servos pueden funcionar correctamente en un rango de movimiento de 90º, que se corresponde con pulsos PWM comprendidos entre 0.9ms y 2.1ms. Sin embargo, también existen servos que se pueden mover en un rango extendido de 180º y sus pulsos de control varían entre 0.5ms y 2.5ms (Figura 1. 9). Antes de utilizar un servo habrá que comprobar

experimentalmente su rango de movimiento para no dañarlo. Para mantener fijo un servo en una posición habrá que enviar periódicamente el pulso correspondiente; ya que si no recibe señales, el eje del servo quedará libre y se podrá mover ejerciendo una leve presión.

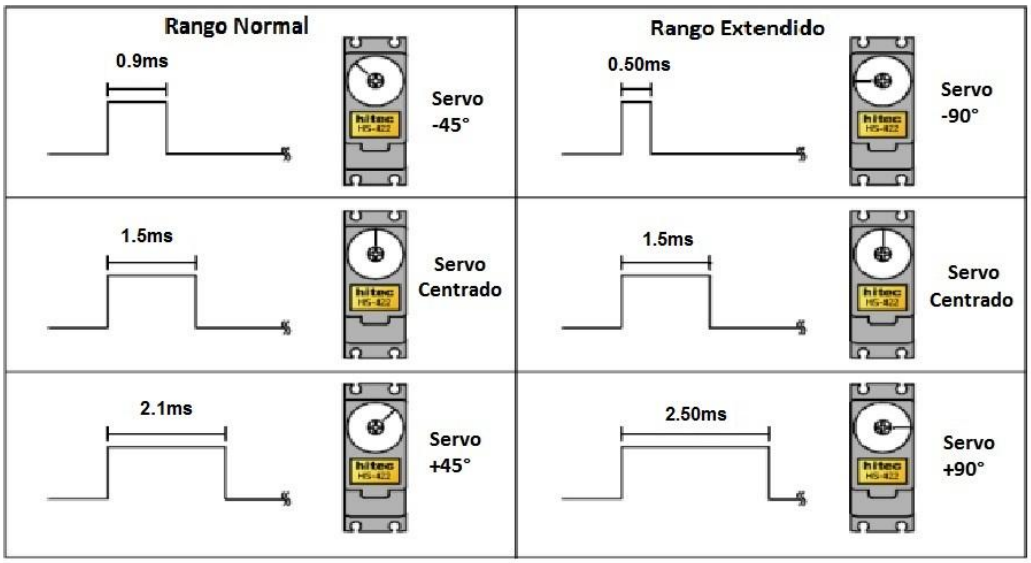


Figura 1. 9.- Pulsos PWM para controlar servomotores

Fuente: (F. A. C. Herías, «Servomotores») [2]

1.8.6 Ventajas.

Últimamente los servos han evolucionado mucho en lo que se refiere a tamaño, potencia y velocidad. Los servos digitales (Figura 1. 10) han superado con creces a los servos conocidos como “coreless” (servos sin núcleo macizo de hierro, se componen de un tubo hueco de electroimanes, lo que los hace ligeros, con poca inercia con lo que se vuelven muy rápidos).

Figura 1. 10.- Servo digital

Fuente: (F. A. C. Herías, «Servomotores») [2]

Otras ventajas que ya se han mencionado en apartados anteriores son que disminuye la zona neutra del servo y le proporciona una buena rapidez de respuesta, aceleración rápida, frenado progresivo y más eficaz con una mejor estabilidad en el posicionamiento estático. El inconveniente de los servos digitales en comparación con los analógicos es su mayor consumo de corriente eléctrica**.**

1.9 Comunicación serial.

1.9.1 Introducción

La comunicación serial es un protocolo muy común para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. La mayoría de las computadoras incluyen dos puertos seriales RS-232. La comunicación serial es también un protocolo común utilizado por varios dispositivos para instrumentación; existen varios dispositivos compatibles con GPIB que incluyen un puerto RS-232. Además, la comunicación serial puede ser utilizada para adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo.

El concepto de comunicación serial es sencillo. El puerto serial envía y recibe bytes de información un bit a la vez. Aun y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias.

Por ejemplo, la especificación IEEE 488 para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20m, con no más de 2m entre dos dispositivos; por el otro lado, utilizando comunicación serial el largo del cable puede llegar a los 1200m.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de

transmisión: (1) Tierra (o referencia), (2) Transmitir, (3) Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben datos por otra. Existen otras líneas disponibles para realizar handshaking, o intercambio de pulsos de sincronización, pero no son requeridas. Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

- **Velocidad de transmisión** (baudrate)**:** Indica el número de bits por segundo que se transfieren, y se mide en baudios (bauds). Por ejemplo, 300 baudios representa 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz. Las velocidades de transmisión más comunes para las líneas telefónicas son de 14400 baudios, 28800 baudios, y 33600 baudios. Es posible tener velocidades más altas, pero se reduciría la distancia máxima posible entre los dispositivos. Las altas velocidades se utilizan cuando los dispositivos se encuentran uno junto al otro, como es el caso de dispositivos GPIB.
- **Bits de datos:** Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5 bits, 7 bits y 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII

extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usar para referirse a todos los casos.

- **Bits de parada:** Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1 bit, 1,5 bits o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo la transmisión será más lenta.
- **Paridad:** Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de

paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados. [3]

1.9.2 Comunicación RS-232

RS-232 (Estándar ANSI/EIA-232) es el conector serial que se encentra en los computadores personales IBM y compatibles. En su momento fue utilizado para una gran variedad de propósitos, como conectar un mouse, impresora o módem, así como instrumentación industrial, aunque en la actualidad aún se lo utiliza para la comunicación en diferentes aplicaciones. Gracias a las mejoras que se han ido desarrollando en las líneas de transmisión y en los cables, existen aplicaciones en las que se aumenta el desempeño de RS-232 en lo que respecta a la distancia y velocidad del estándar. RS-232 está limitado a comunicaciones de punto a punto entre los dispositivos y el puerto serial de la computadora. El hardware de RS-232 se puede utilizar para comunicaciones seriales en distancias de hasta 15m.

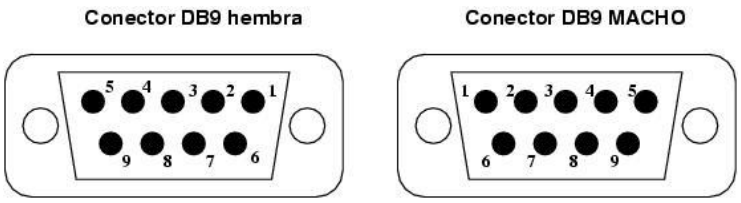


Figura 1. 11.- Pines de conector DB9

Fuente: (N. Instruments, «Comunicación Serial») [3]

- Conector externo de la computadora y expuesto del cable.

Funciones de los pines en RS-232:

- Datos: TXD (pin 3), RXD (pin 2)
- Handshake: RTS (pin 7), CTS (pin 8), DSR (pin 6), DCD (pin 1), DTR (pin 4)
- Tierra: GND (pin 5)
- Otros: RI (pin 9)

1.9.3 Comunicación RS-422

RS-422 (Estándar EIA RS-422-A) es el conector serial utilizado en las computadoras Apple de Macintosh. El protocolo RS-422 usa señales eléctricas diferenciales, en comparación con señales referenciadas a tierra como en RS-232.

La transmisión diferencial, que utiliza dos líneas para transmitir y recibir, tiene la ventaja que es más inmune al ruido y puede lograr mayores distancias que RS-232. La inmunidad al ruido y la distancia son dos puntos clave para ambientes y aplicaciones industriales.

1.9.4 Comunicación RS-485

RS-485 o también conocido como EIA-485, que lleva el nombre del comité que lo convirtió en estándar en 1983. Es un estándar de comunicaciones en bus de la capa física del Modelo OSI.

Además es una mejora sobre RS-422 ya que incrementa el número de dispositivos que se pueden conectar (de 10 a 32) y define las características necesarias para asegurar los valores adecuados de voltaje cuando se tiene la carga máxima. Gracias a esta capacidad, es posible crear redes de dispositivos conectados a un solo puerto RS-485. Esta capacidad, y la gran inmunidad al ruido, hacen que este tipo de transmisión serial sea la elección de muchas aplicaciones industriales que necesitan

dispositivos distribuidos en red conectados a una PC u otro controlador para la colección de datos, HMI, u otras operaciones. RS-485 es un conjunto que cubre RS-422, por lo que todos los dispositivos que se comunican usando RS-422 pueden ser controlados por RS-485. [3]

El hardware de RS-485 se puede utilizar en comunicaciones seriales de distancias de hasta 1220m de cable. La disposición de pines del conector es el mismo de la comunicación RS-232, un conector DB9 ilustrado en la Figura 1. 11.

- Conector externo de la computadora y expuesto del cable.

Funciones de los pines en RS-485 y RS-422:

- Datos: TXD+ (pin 8), TXD- (pin 9), RXD+ (pin 4), RXD- (pin 5)
- Handshake: RTS+ (pin 3), RTS- (pin 7), CTS+ (pin 2), CTS- (pin 6)
- Tierra: GND (pin 1)

a. Especificaciones

- Interfaz diferencial
- Conexión multipunto
- Alimentación única de +5v
- Hasta 32 estaciones (ya existen interfaces que permiten conectar 256 estaciones)
- Velocidad máxima de 10 Mbit/s (a 12m)
- Longitud máxima de alcance de 1200m (a 100 Kbit/s)
- Rango de bus de -7v a +12v

b. Aplicaciones

- RS-485 se usa con frecuencia en las UARTs para comunicaciones de datos de poca velocidad en las cabinas de los aviones. Por ejemplo, algunas unidades de control del pasajero lo utilizan, equipos de monitoreo de sistemas fotovoltaicos. Requiere el cableado mínimo, y puede compartir el cableado entre varios asientos. Por lo tanto reduce el peso del sistema.
- RS-485 se utiliza en sistemas grandes de sonido, como los conciertos de música y las producciones de teatro, se usa software especial para controlar remotamente el equipo de sonido de una computadora, es utilizado más generalmente para los micrófonos.
- RS-485 también se utiliza en la automatización de los edificios pues el cableado simple del bus y la longitud de cable es larga por lo que son ideales para ensamblar los dispositivos que se encuentran alejados.
- RS-485 Tiene la mayor parte de su aplicación en las plantas de producción automatizadas.

1.9.5 Protocolo USB

El dispositivo utilizado para la comunicación entre computador y brazo robótico convierte los datos de RS-485 a USB y viceversa por lo que es conveniente hablar algunas de las características y datos importantes del tipo de comunicación USB.

Las siglas USB (Universal Serial Bus) o “Bus Universal en Serie”, es un bus estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de

alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos.

Su desarrollo partió de un grupo de empresas del sector que buscaban unificar la forma de conectar periféricos a sus equipos, por aquella época poco compatibles entre sí, entre las que estaban Intel, Microsoft, IBM, Compaq, DEC, NEC y Nortel. La primera especificación completa 1.0 se publicó en 1996 por el Foro de Implementadores de BUS (USB Implementers Forum, USB-IF), pero en 1998 con la especificación 1.1 comenzó a usarse de forma masiva.

El USB es utilizado como estándar de conexión de periféricos como: teclados, mouse, memorias USB, joysticks, escáneres, módems, cámaras digitales, teléfonos móviles, reproductores multimedia, impresoras, dispositivos multifuncionales, sistemas de adquisición de datos, tarjetas de red, tarjetas de sonido, tarjetas sintonizadoras de televisión y grabadoras de DVD externa, discos duros externos y disqueteras externas. Su éxito ha sido total, habiendo desplazado a conectores como el puerto serie, puerto paralelo, puerto de juegos, Apple Desktop Bus o PS/2 a mercados-nicho o a la consideración de dispositivos obsoletos a eliminar de las modernas computadoras, pues muchos de ellos pueden sustituirse por dispositivos USB que implementen esos conectores. Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos:

- Baja velocidad (1.0): Tasa de transferencia de hasta 1,5 Mbps (188 KB/s). Utilizado en su mayor parte por dispositivos de interfaz humana (Human Interface Device) como los teclados, los ratones (mouse), las cámaras web, etc.
- Velocidad completa (1.1): Tasa de transferencia de hasta 12 Mbps (1,5 MB/s) según este estándar, pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones. Ésta fue la más rápida antes de la especificación USB 2.0. Estos

dispositivos dividen el ancho de banda de la conexión USB entre ellos, basados en un algoritmo de impedancias LIFO.

- Alta velocidad (2.0): Tasa de transferencia de hasta 480 Mbps (60 MB/s) pero con una tasa real práctica máxima de 280 Mbps (35 MB/s). El cable USB 2.0 dispone de cuatro líneas, un par para datos, y otro par de alimentación. Casi todos los dispositivos fabricados en la actualidad trabajan a esta velocidad
- Súper alta velocidad (3.0): Tiene una tasa de transferencia de hasta 4,8 Gbps (600 MB/s). La velocidad del bus es diez veces más rápida que la del USB 2.0, debido a que han incluido 5 contactos adicionales, desechando el conector de fibra óptica propuesto inicialmente, y será compatible con los estándares anteriores. En octubre de 2009 la compañía taiwanesa ASUS lanzó la primera placa base que incluía puertos USB 3.0, tras ella muchas otras le han seguido y actualmente se ve cada vez más en placas base y portátiles nuevos, conviviendo junto con el USB 2.0.

1.10 Lenguaje de programación Python

1.10.1 Historia

El creador del lenguaje es un europeo llamado Guido Van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica), en los Países Bajos. Hace ya más de una década que diseñó Python, ayudado y motivado por su experiencia en la creación de otro lenguaje llamado ABC. El objetivo de Guido era cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en Unix usando C. El desarrollo de Python duró varios años, durante los que trabajó en diversas compañías de Estados Unidos. En el 2000 ya disponía de un producto bastante completo y un equipo de desarrollo con el que se había asociado incluso en proyectos empresariales. Actualmente trabaja en

Zope, una plataforma de gestión de contenidos y servidor de aplicaciones para la web, por supuesto, programada por completo en Python.

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad. [4]. En los últimos años el lenguaje se ha hecho muy popular, gracias a varias razones como:

La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.

La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C. La cantidad de plataformas en las que podemos desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros. Además, Python es gratuito, incluso para propósitos empresariales.

Figura 1. 12.- Logo de Python
Fuente: (P. Org., «The Python Tutorial») [5]

1.10.2 Características del lenguaje

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo

particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones. Además usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos). Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable. Entre otras características del lenguaje se encuentran:

- **Propósito general**
  Se pueden crear todo tipo de programas. No es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.
- **Multiplataforma (Open Source)**
  Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él. Debido a la naturaleza de Python de ser Open Suorce; ha sido modificado para que pueda funcionar en diversas plataformas (Linux, Windows, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE y PocketPC). Al ser Open Source es gratuito.
- **Interpretado**
  Quiere decir que no se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero

esta se realiza de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos bytecodes que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.

- **Interactivo**

  Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudar a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.

- **Orientado a Objetos**

  La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

- **Funciones y librerías**

  Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip. Las librerías pueden ayudar a realizar varias acciones como expresiones regulares, generación de documentos, evaluación de unidades, pruebas, procesos, bases de datos, navegadores web, CGI, ftp, correo electrónico, XML, XML-RPC, HTML, archivos WAV, criptografía, GUI, y también otras funciones dependientes del Sistema.

- **Sintaxis clara**

  Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan

elementos como las llaves o las palabras clave begin y end. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

- **Simple**
  Python es un lenguaje muy simple, por lo que es muy fácil iniciarse en este lenguaje. El pseudo-código natural de Python es una de sus grandes fortalezas.
- **Propósito General**
  Usando el lenguaje Python se puede crear todo tipo de programas; programas de propósito general y también se pueden desarrollar páginas Web.
- **Lenguaje de Alto Nivel:**
  Al programar en Python no se debe preocupar por detalles de bajo nivel, (como manejar la memoria empleada por el programa).
- **Incrustable**
  Se puede insertar lenguaje Python dentro un programa C/C++ y de esta manera ofrecer las facilidades del scripting. [6]

1.10.3 Modo interactivo

El intérprete de Python estándar incluye un modo interactivo en el cual se escriben las instrucciones en una especie de intérprete de comandos, las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente, lo que da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como para los

programadores más avanzados. En la Figura 1. 13 se observa un ejemplo de programación en el modo interactivo en el cual en la primera línea se ingresa una operación matemática sencilla y al presionar la tecla ENTER el resultado aparece inmediatamente en la segunda línea; en la tercera y cuarta línea se ingresan comandos para crear un vector de 10 elementos e imprimirla, Python devuelve el resultado mostrado en la quinta línea.

```
>>> 1 + 1
2
>>> a = range(10)
>>> print a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Figura 1. 13.- Ejemplo del Modo Interactivo de Python

Elaborado por: (Mamarandi J., Velasco E.)

Existen otros programas, tales como IDLE, bPython o IPython que añaden funcionalidades extra al modo interactivo, como el autocompletado de código y el coloreado de la sintaxis del lenguaje. [6]

1.10.4 Elementos del lenguaje

Python fue diseñado para ser leído con facilidad. Una de sus características es el uso de palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos:

Tabla 1. 1.- Operadores lógicos y su equivalente en Python

C	Python
!	not
||	or
&&	and

Elaborado por: (Mamarandi J., Velasco E.)

El contenido de los bloques de código (bucles, funciones, clases, etc.) es delimitado mediante espacios o tabuladores, conocidos como indentación, antes de cada línea de órdenes pertenecientes al bloque. Python se diferencia así de otros lenguajes de programación que mantienen como costumbre declarar los bloques mediante un conjunto de caracteres, normalmente entre llaves **{ }**. Se pueden utilizar tanto espacios como tabuladores para identar el código, pero se recomienda no mezclarlos.

```c
int factorial(int x)
{
    if (x == 0)
        return 1;
    else
        return x * factorial(x - 1);
}
```

a) Función factorial en C

```python
def factorial(x):
    if x == 0:
        return 1
    else:
        return x * factorial(x - 1)
```

b) Función factorial en Python

Figura 1. 14.- a) Función factorial en C (indentación opcional); b) Función factorial en Python (indentación obligatoria)

Elaborado por: (Mamarandi J., Velasco E.)

Debido al significado sintáctico de la indentación, una instrucción debe estar contenida en línea. No obstante, si por legibilidad se quiere dividir la instrucción en varias líneas, añadiendo una barra invertida **** al final de una línea, se indica que la instrucción continúa en la siguiente.

```python
lista=['valor 1','valor 2','valor 3']
cadena='Esto es una cadena bastante larga'
```

a) Instrucciones a renglón seguido

```
lista=['valor 1','valor 2' \
      ,'valor 3']
cadena='Esto es una cadena ' \
       'bastante larga'
```

b) Uso de barra invertida en programación

Figura 1. 15.- a) Instrucciones a renglón seguido; b) Equivalente de la figura a) usando una barra invertida

Elaborado por: (Mamarandi J., Velasco E.)

a. Comentarios

Los comentarios se pueden poner de dos formas. La primera y más apropiada para comentarios largos es utilizando la notación **''' comentario '''**, tres apóstrofes de apertura y tres de cierre. La segunda notación utiliza el símbolo #, y se extienden hasta el final de la línea. El intérprete no tiene en cuenta los comentarios, lo cual es útil si deseamos poner información adicional en nuestro código como, por ejemplo, una explicación sobre el comportamiento de una sección del programa.

```
'''
primera forma de comentar
'''
# segunda forma de comentar
print "hola mundo"
```

Figura 1. 16.- Diferentes formas de usar comentarios

Elaborado por: (Mamarandi J., Velasco E.)

b. Variables

Las variables se definen de forma dinámica, lo que significa que no se tiene que especificar cuál es su tipo de antemano y puede tomar distintos valores en otro momento, incluso de un tipo diferente al que tenía previamente. Se usa el símbolo **=** para asignar valores.

c. Tipos de datos

Tabla 1. 2.- Tipos de datos utilizados en Python

Tipo	Clase	Notas	Ejemplo
str	Cadena	Inmutable	'Cadena'
unicode	Cadena	Versión Unicode de s tr	u'Cadena'
list	Secuencia	Mutable, puede contener objetos de diversos tipos	[4.0, 'Cadena', True]
tuple	Secuencia	Inmutable, puede contener objetos de diversos tipos	(4.0, 'Cadena', True)
set	Conjunto	Mutable, sin orden, no contiene duplicados	set([4.0, 'Cadena', True])
frozenset	Conjunto	Inmutable, sin orden, no contiene duplicados	frozenset([4.0, 'Cadena', True])
dict	Mapping	Grupo de pares clave:valor	{'key1': 1.0, 'key2': False}
int	Número entero	Precisión fija, convertido en long en caso de overflow.	42
long	Número entero	Precisión arbitraria	42L ó 456966786 151987643L
float	Número decimal	Coma flotante de doble precisión	3.1415927
complex	Número complejo	Parte real y parte imaginaria j.	(4.5 + 3j)
Bool	Booleano	Valor booleano verdadero o falso	True o False

Elaborado por: (Mamarandi J., Velasco E.)

- Mutable: si su contenido (o dicho valor) puede cambiarse en tiempo de ejecución.
- Inmutable: si su contenido (o dicho valor) no puede cambiarse en tiempo de ejecución.

```
x = 1

y = "texto"

z = [1,2,4]
```

Figura 1. 17.- Asignación de variables

Elaborado por: (Mamarandi J., Velasco E.)

d. Funciones

- Las funciones se definen con la palabra clave **def**, seguida del nombre de la función y sus parámetros. Otra forma de escribir funciones, aunque menos utilizada, es con la palabra clave **lambda** (que aparece en lenguajes funcionales como Lisp).
- El valor devuelto en las funciones con **def** será el dado con la instrucción **return**.

```
>>> def suma(x, y = 2):
...     return x + y # Retornar la suma del valor de la variable "x" y el valor de "y"
...
>>> suma(4) # La variable "y" no se modifica, siendo su valor: 2
6
>>> suma(4, 10) # La variable "y" sí se modifica, siendo su nuevo valor: 10
14
```

a) Instrucciones de una función utilizando **def**

```
>>> suma = lambda x, y = 2: x + y
>>> suma(4) # La variable "y" no se modifica, siendo su valor: 2
6
>>> suma(4, 10) # La variable "y" sí se modifica, siendo su nuevo valor: 10
14
```

b) Instrucciones de una función utilizando **lambda**

Figura 1. 18.- a) Instrucciones de una función utilizando **def**;
b) Instrucciones de una función utilizando **lambda**

Elaborado por: (Mamarandi J., Velasco E.)

e. Clases

- Las clases se definen con la palabra clave **class**, seguida del nombre de la clase y, si hereda de otra clase, el nombre de esta.

- En Python 2.x es recomendable que una clase herede de "object", en Python 3.x esto ya no hará falta.
- En una clase un "método" equivale a una "función", y una "propiedad" equivale a una "variable".
- "__init__" es un método especial que se ejecuta al instanciar la clase, se usa generalmente para inicializar propiedades y ejecutar métodos necesarios. Al igual que todos los métodos en Python, debe tener al menos un parámetro, generalmente se utiliza **self**. El resto de parámetros serán los que se indiquen al instanciar la clase.
- Las propiedades que se desee que sean accesibles desde fuera de la clase se deben declarar usando **self** delante del nombre.
- En Python no existe el concepto de encapsulación, por lo que el programador debe ser responsable de asignar los valores a las propiedades.

```
class PnomioC:
    def  __init__(self, ao=-1, a1=0, a2=1):
            self.ao = float(ao)
            self.a1 = float(a1)
            self.a2 = float(a2)
    def myroots(self):
            from math import sqrt
            inside = self.a1**2 - 4*self.ao*self.a2
            if inside < 0:
                    rr = sqrt(-1*inside)*1j
            else:
                    rr = sqrt(inside)
            r1 = (-self.a1 + rr)/(2*self.a2)
            r2 = (-self.a1 - rr)/(2*self.a2)
            print 'Las raices del polinomio cuadratico'
            print '(',self.a2,')X**2 + (',self.a1,')X + (',self.ao,')'
            print ' son : ',r1,' y ' ,r2
    def __call__(self, x):
            y = self.a2*(x**2) + self.a1*x + self.ao
            return y
```

Figura 1. 19.- Instrucciones de una clase

Elaborado por: (Mamarandi J., Velasco E.)

f. Condicionales

Una sentencia condicional **if** ejecuta su bloque de código interno sólo si se cumple cierta condición. Se define usando la palabra clave **if** seguida de la condición, y el bloque de código. Si se requieren condiciones adicionales estas se introducen usando **elif** seguida de la condición y su bloque de código. Todas las condiciones se evalúan secuencialmente hasta encontrar la primera que sea verdadera, y su bloque de código asociado es el único que se ejecuta. Opcionalmente, puede haber un bloque final (la palabra clave **else** seguida de un bloque de código) que se ejecuta sólo cuando todas las condiciones fueron falsas.

```
if x == y:
    print (x, "y", y, "como iguales")
else:
    if x < y:
        print(x, "es menor que", y)
    else:
        print(x, "es mayor que", y)
```

Figura 1. 20.- Código usando el condicional **if**

Elaborado por: (Mamarandi J., Velasco E.)

g. Bucle for

El bucle **for** es similar a **foreach** en otros lenguajes. Recorre un objeto iterable, como una lista, una tupla o un generador, y por cada elemento del iterable ejecuta el bloque de código interno. Se define con la palabra clave **for** seguida de un nombre de variable, depues de la palabra **in**, sucedida del iterable, y finalmente el bloque de código interno. En cada iteración, el elemento siguiente del iterable se asigna al nombre de variable especificado.

```
lista = ["a", "b", "c"]
for i in lista:
    print i
```

Figura 1. 21.- Código utilizando el bucle **for**

Elaborado por: (Mamarandi J., Velasco E.)

h. Bucle while

El bucle **while** evalúa una condición y, si es verdadera, ejecuta el bloque de código interno. Continúa evaluando y ejecutando mientras la condición sea verdadera. Se define con la palabra clave **while** seguida de la condición, y a continuación el bloque de código interno

```
numero = 0
while numero < 10:
    numero += 1
    print numero,
```

Figura 1. 22.- Código utilizando el bucle **while**

Elaborado por: (Mamarandi J., Velasco E.)

1.10.5 Sistema de objetos

En Python todo es un objeto (incluso las clases). Las clases, al ser objetos, son instancias de una clase. Python además soporta herencia múltiple y polimorfismo.

```
>>> cadena = "abc"
>>> cadena.upper()
'ABC'
>>> lista = [True, 3.1415]
>>> lista.append(42L)
>>> lista
[True, 3.1415000000000002, 42L]
```

Figura 1. 23.- Ejemplo de Sistema de objetos

Elaborado por: (Mamarandi J., Velasco E.)

1.10.6 Implementaciones.

Existen diversas implementaciones del lenguaje:

- CPython: es la implementación original, disponible para varias plataformas en el sitio oficial de Python.

- IronPython: es la implementación para .NET
- Stackless Python: es la variante de CPython que trata de no usar el stack de C (www.stackless.com)
- Jython: es la implementación hecha en Java
- Pippy: es la implementación realizada para Palm (pippy.sourceforge.net)
- PyPy: es una implementación de Python escrita en Python y optimizada mediante JIT (pypy.org). [5]

1.11 Comparación entre lenguajes de programación Python – C

Luego de revisar, estudiar y analizar todo lo investigado sobre el lenguaje de programación se determinó realizar una tabla de comparación entre C y Python en donde cada uno tiene sus fortalezas pero también sus debilidades, algunas de las razones por las que se escogió este último es por ser multiparadigma y de fácil entendimiento en su estructura, además de que se pueden ahorrar varias líneas de código y sus aplicabilidad se requiere en varios campos.

Tabla 1. 3.- Comparación entre lenguajes de programación C y Python

	C	PYTHON
Velocidad de ejecución	Por lo general más rápido, se puede realizar varias optimizaciones	Los paquetes son más lentos, pero con algunas extensiones como numpy y otras pueden mejorar la velocidad de ejecución.
Creación de algoritmo	Generalmente más lento y complicado al programar, pero da la ventaja de la comprobación de tipos en tiempo de compilación	A la mayoría de las personas les resulta más fácil y más rápido para crear algoritmos.
Flujo de trabajo	Difícil de entender y frágil, cuando las propiedades se deben establecer para muchos algoritmos.	Muy conciso y totalmente fuera del marco básico, así que no hay manera de romper la funcionalidad existente. Promueve la reutilización.
Operadores lógicos	Se utilizan símbolos	Se utilizan palabras
Paradigma	Imperativo (Procedural), Estructurado	Multiparadigma: orientado a objetos, imperativo, funcional, reflexivo.
Tipo de dato	Débil, estático	Débilmente tipado, dinámico.
Nivel	Alto	Muy alto

Elaborado por: (Mamarandi J., Velasco E.)

CAPÍTULO 2

2 ANÁLISIS DEL DISEÑO.

Una vez revisado toda la parte teórica es importante planear un diseño; en el capítulo 2 se observa cómo se realizó todo lo referente al ensamblaje del brazo utilizando los servos Dynamixel, la comunicación y alimentación de la red de servomotores, además como está planteado el algoritmo de programación tanto para las librerías como para la interfaz gráfica.

2.1 Servomotores Dynamixel

Como se observó en el capítulo anterior los servomotores son motores de corriente continua que poseen la electrónica que permite el control y monitoreo de varios parámetros de este como la velocidad, posición, torque, etc. debido a esto existen muchas empresas que los fabrican y cada una de ellas implementa a su manera estos mecanismos ofreciendo diferentes ventajas a los compradores.

Para realizar este proyecto se optó por adquirir los servomotores de la marca Dynamixel que han sido utilizados en laboratorios de investigación en el área de robótica en todo el mundo. Cada actuador inteligente tiene un microprocesador incorporado para facilitar la comunicación de bus, retroalimentación de la posición, la temperatura y la supervisión de la carga.

La carcasa de cada servo está construida específicamente para la robótica, proporcionando una facilidad de usar rieles de montaje y un amplio sistema de soporte disponible para la construcción de las extremidades robóticas.

Las comunicaciones serial, TTL y RS-485 permiten conexiones bus en cadena con transferencias de 1 Mbps a 3 Mbps. Además, los MCU (Microcontroller Unit / Unidades Microcontroladoras) a bordo de los servos Dynamixel tienen un conjunto de características configurables por el usuario, permitiéndoles ajustar los servos para las aplicaciones necesarias. Cada uno de estos servoactuadores tiene diferentes características según su modelo, pero por lo general son las siguientes:

- Dedicado MCU a bordo
- Torque, velocidad y respuesta ajustable
- Posición, carga, voltaje, temperatura de realimentación
- Comunicación en serie, conexión en cadena
- Amplia gama de tamaños, fuerzas y opciones de comunicación
- Diseño de montaje modular con soportes y marcos integrales
- Modelos 3D, esquemas de dimensiones, completa documentación disponible

2.1.1 Servo actuador serie MX

Los servo actuadores Dynamixel Robot MX son la última generación de actuadores Robotis Dynamixel; equipados con un procesador a bordo Cortex M3 de 32 bits y 72MHz, un codificador sin contacto magnético con una resolución de 4x sobre la serie AX/RX, y hasta 3 Mbps utilizando comunicación RS-485 en bus 2.0. Cada servo tiene la capacidad de realizar un seguimiento de su velocidad, la temperatura, la posición del eje, tensión, y la carga.

El algoritmo de control PID de reciente aplicación utilizado para mantener la posición del eje se puede ajustar individualmente para cada servo, lo que permite controlar la velocidad y la fuerza de la respuesta del motor. Todos los servos de la serie MX utilizan tensión nominal de 12v, por

lo demás la serie Dynamixel MX-R se pueden mezclar sin tener que preocuparse acerca de las fuentes de alimentación independientes.

Todo el control de la gestión y de la posición del sensor se maneja mediante una función de microcontrolador del servo. Este enfoque distribuido sale de su controlador principal libre para realizar otras funciones. Estos servos son casi idénticos a los de la serie MX-T, con la única diferencia de 4 pines de comunicación RS-485. [7]

Para mejor referencia se muestran las características técnicas de los servomotores utilizados (Figura 2. 1) en la Tabla 2. 1.

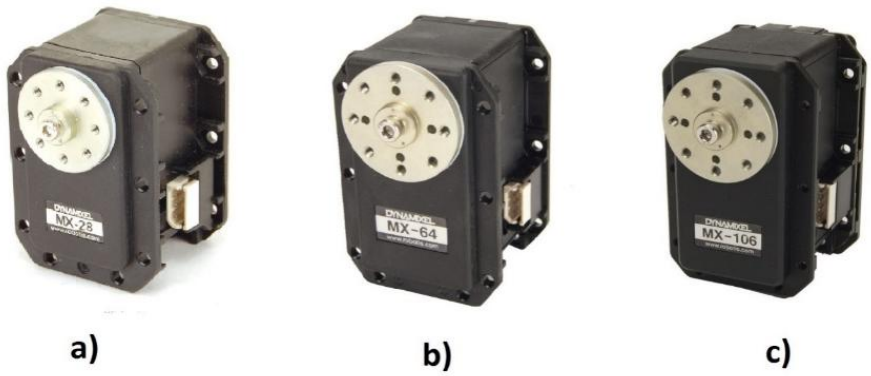


Figura 2. 1.- Servos Dynamixel; a) MX-28; b) MX-64; c) MX-106
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

Tabla 2. 1.- Datos técnicos de los servomotores utilizados.

MODELO	MX-28	MX-64	MX-106
Par de bloqueo al máximo voltaje	3,1 N.m (31,6 kg-cm)	7,3 N.m (74,4 kg-cm)	10,0 N.m (101 kg-cm)
Velocidad (RPM)	67	78	55
Voltaje nominal de operación	11,1v - 14,8v	11,1v - 14,8v	11,1v - 14,8v
Consumo de corriente de bloqueo	1,7A	5,2A	6,3A
Dimensiones (mm)	35,6x50,6x35,5	40,2x61,1x41	40,2x65,1x46
Peso	72g	126g	153g
Resolución	0,088°	0,088°	0,088°
Ángulo de operación	360	360	360
Reducción de engranaje	190:1	200:1	225:1
Material del tren de engranaje	Acero habitual	Acero habitual	Acero habitual
CPU	Cortex M3 (STM32F103C 8 a 72MHz, 32 Bit)	Cortex M3 (STM32F103C 8 a 72MHz, 32 Bit)	Cortex M3 (STM32F103C 8 a 72MHz, 32 Bit)
Sensor de posición	Encoder Magnético	Encoder Magnético	Encoder Magnético
Protocolo de comunicación	TTL/RS-485	TTL/RS-485	TTL/RS-485
Velocidad de comunicación	3Mbps	3Mbps	3Mbps
Conformidad/PID	PID	PID	PID

Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

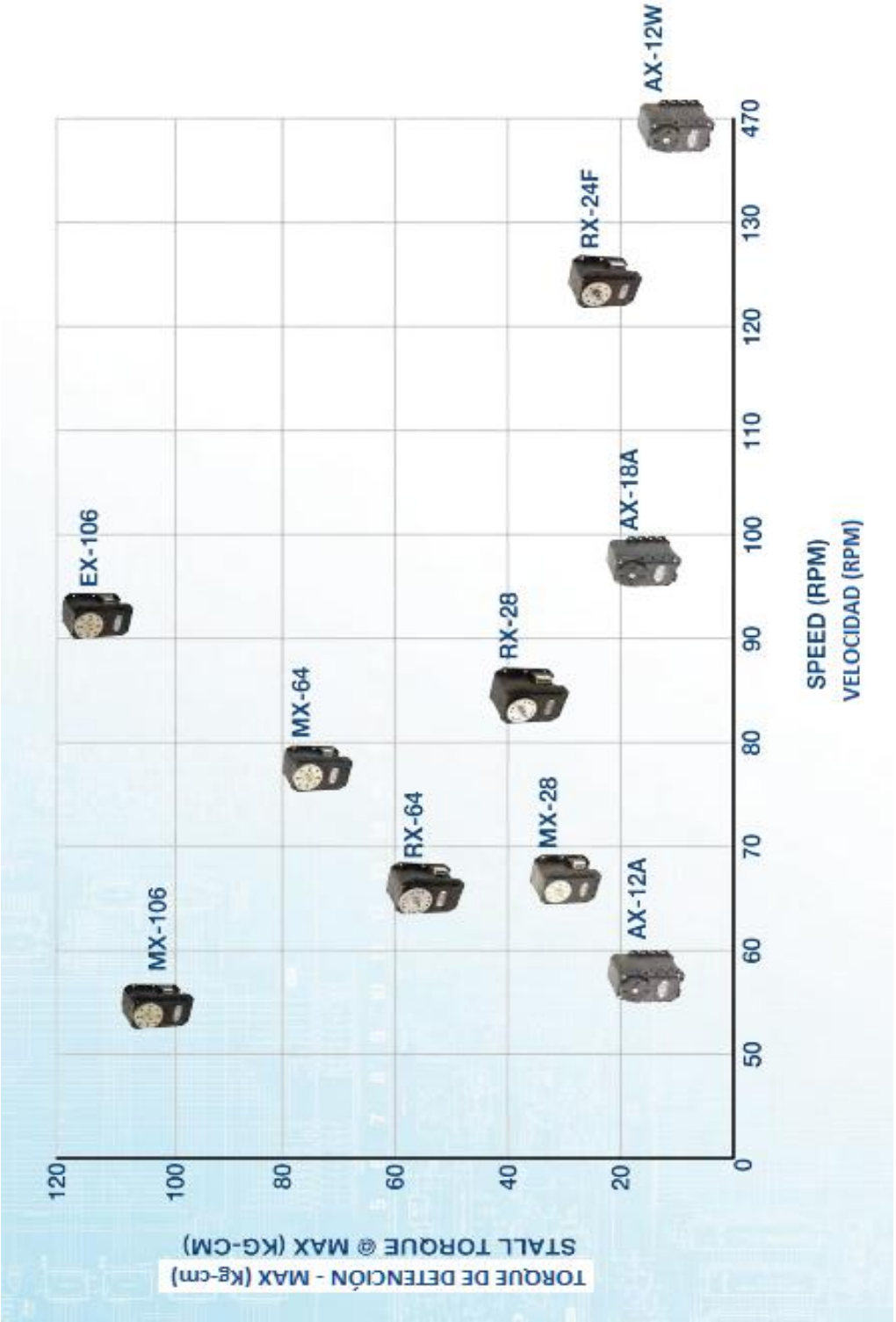


Figura 2. 2.- Gráfico fuerza-velocidad de la familia Dynamixel.

Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

2.2 Ensamblaje del sistema

Las características que poseen los servomotores Dynamixel los hace requeridos para el desarrollo de distintas aplicaciones robóticas, su control mediante una aplicación en software libre permitirá desarrollar proyectos donde el conocimiento de su control sea más fácilmente entendible y no requiera de compra de licencias para poder avanzar con una investigación futura.

El brazo robótico ensamblado para este proyecto está conformado por los servomotores Dynamixel MX-28, MX-64 y MX-106 mencionados anteriormente, para la estructura que conforma el brazo robótico se requiere o es imprescindible que las piezas que lo conforman sirvan de unión y soporte para todo el sistema, deben ser lo suficientemente resistentes para soportar el peso y fuerza de los servomotores, pero a la vez deben ser ligeras para que el peso total del brazo robótico sea óptimo y no desperdicie potencia tratando de mover su propio peso.

Por tal razón las piezas y uniones fueron diseñadas en la Universidad de las Fuerzas Armadas ESPE extensión Latacunga y fabricadas en la ciudad de Quito-Ecuador, todo esto fue trabajado en un proyecto a parte pero en la misma línea de investigación.

La Figura 2. 3 muestra el diseño final del brazo robótico en el programa Solid Work y como se veria la estructura final con las piezas manufacturas por la Universidad.

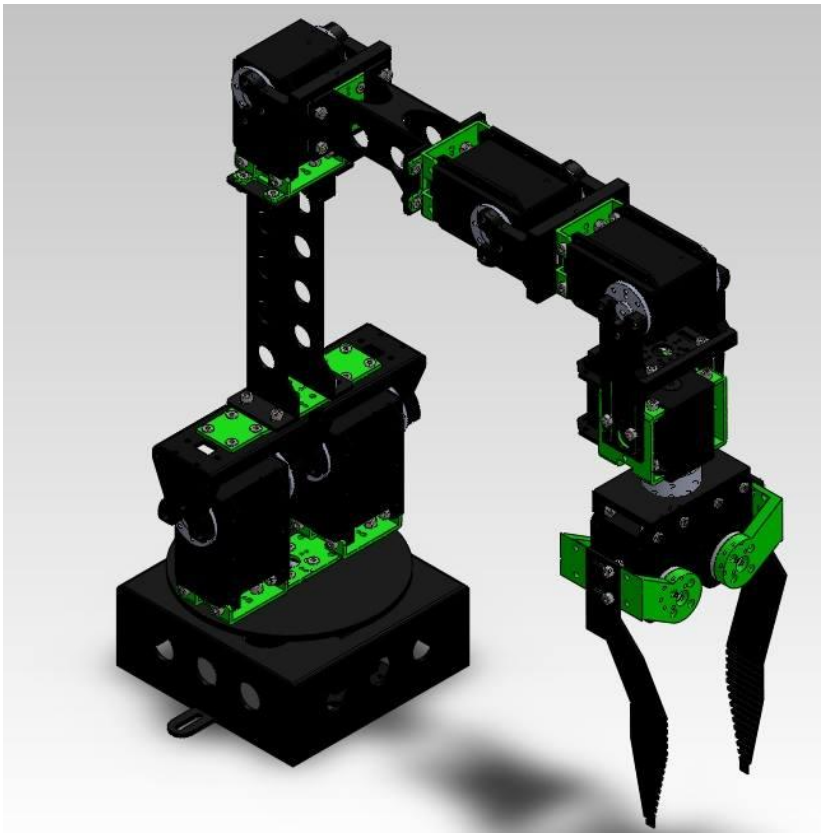


Figura 2. 3.- Diseño final del brazo robótico ensamblado

Elaborado por: (Mamarandi J., Velasco E.)

2.3 Alimentación de la red de servomotores.

Una vez que se tiene el modelo de brazo robótico ensamblado y sujeto a una base lo suficientemente fuerte y estable se procede a energizar la red se servomotores, como se pudo observar en la Tabla 2. 1, estos funcionan con un voltaje entre 11,1v a 14,8v con un consumo de corriente de 1,7A, 5,2A y 6,3A para los modelos MX-28, MX-64 y MX-106 respectivamente, la fuente de voltaje utilizada es de la marca Agilent modelo U8001A ilustrada en la Figura 2. 4 la cual es configurable en sus valores de voltaje y corriente asiéndola perfecta para este tipo de trabajo.

Figura 2. 4.- Fuente AGILENT U8001A

Elaborado por: (Mamarandi J., Velasco E.)

2.4 Comunicación USB2Dynamixel.

La interfaz USB2Dynamixel (Figura 2. 5) es un dispositivo fabricado por la empresa Robotis y es utilizada para operar con dispositivos Dynamixel directamente desde una computadora personal.

USB2Dynamixel se conecta al puerto USB del PC, y tiene dos conectores 3P (3 pines) y 4P (4 pines) que están dispuestos de manera que varios dispositivos de la misma marca pueden ser conectados a estos puertos.

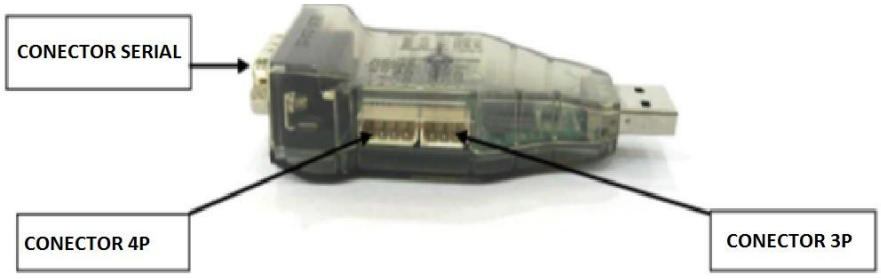


Figura 2. 5.- Interfaz USB2Dynamixel y sus puertos.

Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

El modo de conexión de la red de servomotores a una PC utilizando la interfaz USB2Dynamixel se muestra en la Figura 2. 6.

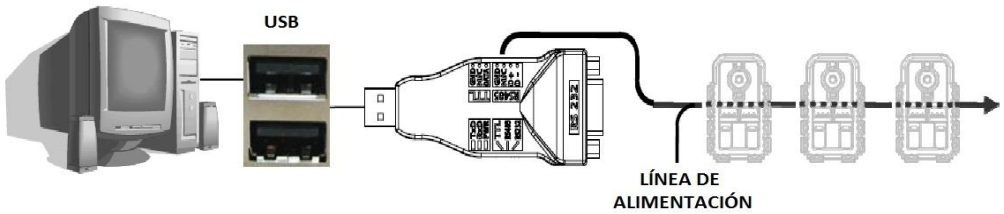


Figura 2. 6.- Conexión de una red de servomotores al PC utilizando la interfaz USB2Dynamixel

Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

También, USB2Dynamixel se puede utilizar para cambiar de puerto USB a puerto serial para usarlas en PC's sin puerto serial tales como de una laptop u otras máquinas. La función es muy útil en los casos en que controladores Dynamixel exclusivos como CM-2, CM-2+, CM-5, y la CM-510 se conectan al puerto USB, o cuando ZIG2Serial está conectado al puerto USB para controlar robots de forma inalámbrica (Figura 2. 7).

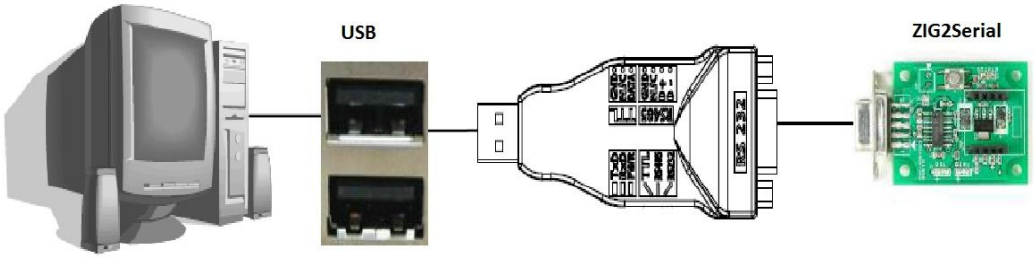


Figura 2. 7.- Conexión de un dispositivo ZIG2Serial al PC utilizando la interfaz USB2Dynamixel.

Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

La interfaz USB2Dynamixel posee en su estructura un switch que permite la elección del protocolo de comunicación entre una computadora y los diferentes dispositivos Dynamixel que existen, de esta forma la interfaz activa el puerto de conexión seleccionado y configura los niveles de voltaje requeridos en la comunicación escogida, de igual forma adapta las señales recibidas de los dispositivos para que la computadora pueda interpretarlos.

Figura 2. 8.- Ubicación del LED de indicación de estado y del switch de selección de función en la interfaz USB2Dynamixel.
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

- Pantalla de estado LED: Muestra el estado de la fuente de alimentación, TxD (escritura de datos), y RxD (datos de la lectura)
- Interruptor de selección de funciones: Selecciona el modo TTL, RS-485 y RS-232 comunica
- Conector 3P: Se conecta a AX Dynamixel serie utilizando la red TTL
- Conector 4P: Se conecta a DX o una serie Dynamixel RX utilizando RS-485
- Conector serie: Transforma un puerto USB en un puerto serie utilizando la red RS-232

2.4.1 Conexión de Dynamixel serie AX

Para la serie de dispositivos Dynamixel AX el procedimiento de conexión es el que se presenta a continuación, teniendo en cuenta que estos utilizan comunicación TTL. Ajustar el interruptor de selección de función para el modo TTL, que es en la posición que se muestra en la Figura 2. 9.

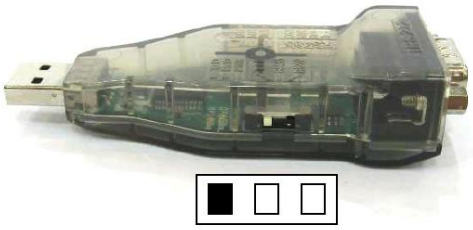


Figura 2. 9.- Posición del switch para modo TTL.
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

Conectar el cable 3P al conector 3P de la interfaz USB2Dynamixel al conector del dispositivo Dynamixel (hay dos conectores en los servo actuadores, en cualquiera de los cuales se pueden conectar).Los dispositivos Dynamixel se pueden conectar en serie mediante un cable 3P como se muestra en la Figura 2. 10, y se conecta la línea de alimentación DC de 7v a 10v al último Dynamixel.

Figura 2. 10.- Conexión en serie de dispositivos Dynamixel serie AX.
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

2.4.2 Conexión de Dynamixel serie DX/RX

La configuración del dispositivo USB2Dynamixel mostrada a continuación es la utilizada para este proyecto ya que los servomotores conectados en serie se comunican por el protocolo RS-485. Para los actuadores Dynamixel de la serie DX/RX se procede de la siguiente manera. Ajustar el interruptor de selección de función para el modo RS-485, que es en la posición que se muestra en la Figura 2. 11.

Figura 2. 11.- Posición del switch para modo RS-485.
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

Conectar el cable 4P al conector 4P de la interfaz USB2Dynamixel al conector del dispositivo Dynamixel (hay dos conectores en los servo actuadores, en cualquiera de los cuales se pueden conectar). Los dispositivos Dynamixel pueden ser conectados en serie mediante un cable 4P como se indica en la Figura 2. 12, finalmente se conecta la línea de alimentación al último Dynamixel, la fuente debe ser DC (Corriente Continua) de 10v a 12v para los modelos RX-10; DC de 12v a 16v para modelos RX-28; DC de 15v a 18v para modelos RX-64 y DC de 12v a 16v para los modelos DX-117.

Figura 2. 12.- Conexión en serie de dispositivos Dynamixel serie DX/RX
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

2.4.3 Transformación de puerto USB a puerto Serial

Como se indicó anteriormente también, USB2Dynamixel se puede utilizar para cambiar de puerto USB a puerto serial para usarlas en PC's sin

puerto serial tales como de una laptop u otras máquinas. Se posiciona el switch selector de función en el modo RS-232 (Figura 2. 13).

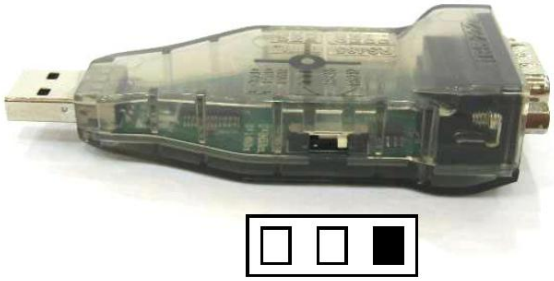


Figura 2. 13.- Posición del switch para modo RS-232.
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

El CM-5 es el controlador de otro de los productos que ofrece la empresa Dynamixel, un robot humanoide conocido como Bioloid; el CM-5 es el encargado de indicar o controlar las acciones del robot, se comunica mediante un puerto serie al PC. Cuando se utiliza el USB2Dynamixel, la red con CM-5 se puede hacer aun usando un puerto USB de un ordenador tal como se muestra en la Figura 2. 14.

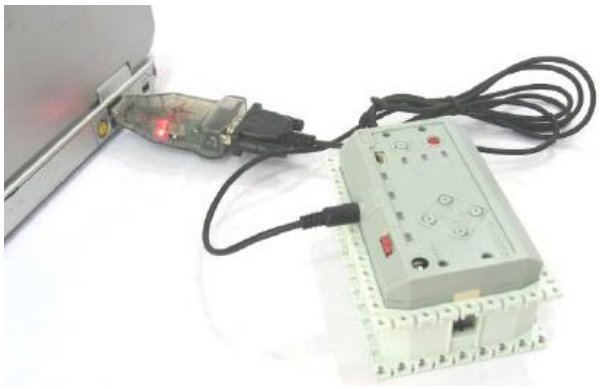


Figura 2. 14.- Conexión del controlador CM-5 a un ordenador.
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

Por otra parte el ZIG2Serial puede ser utilizado como un puerto USB utilizando USB2Dynamixel (Figura 2. 15). Para este tipo de operación, el ZIG2Serial se puede utilizar sin fuente de alimentación adicional.

Figura 2. 15.- Conexión del dispositivo ZIG2Serial al ordenador.
Fuente: (T. Robotics, «Trossen Robotics Dynamixel Guide») [7]

2.5 Cálculos de la cinemática inversa

En el proyecto realizado para conocer los ángulos de cada articulación que permitan posicionar al brazo robótico en una coordenada ya ingresada se debe trabajar en primer lugar sobre el plano YZ del brazo robótico, para mayor facilidad de los cálculos se ha establecido que este es un plano cartesiano XY, de esta manera se realizarán los cálculos en un solo plano, y no de manera espacial en los planos XYZ.

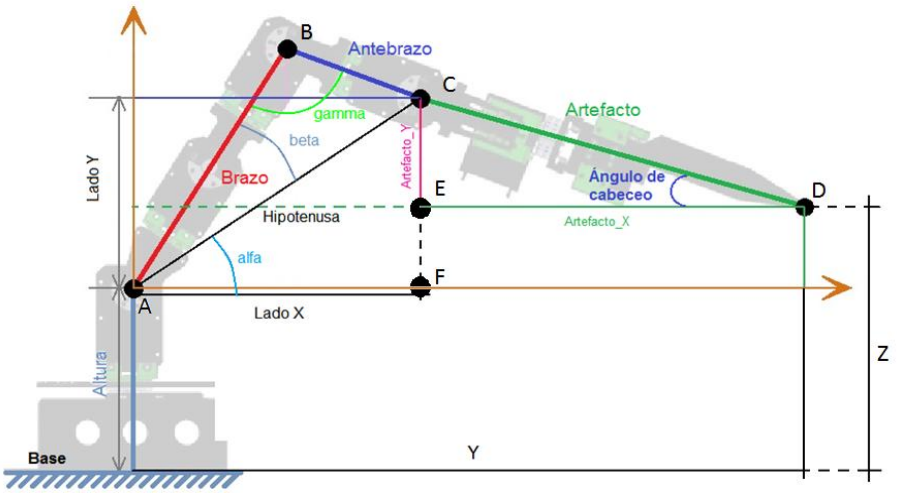


Figura 2. 16.- Componentes del brazo robótico necesarios para el cálculo de la cinemática inversa.
Elaborado por: (Mamarandi J., Velasco E.)

Se dividió la estructura en 3 vectores principales, los cuales ayudarán con los cálculos necesarios para posicionar las articulaciones del brazo robótico, estos vectores son: Brazo, Antebrazo y Artefacto, este último siendo la unión de la muñeca y la pinza.

La imagen de la Figura 2. 16 facilita el análisis, hay que tomar en cuenta que para realizar estos cálculos se debe preestablecer un ángulo de incidencia del artefacto, es decir un ángulo de cabeceo. De esta manera los ángulos a calcular son alfa (**α**), beta (**β**) y gama (**γ**). A continuación se muestra el proceso matemático de la cinemática inversa. Se debe descomponer en sus componentes X y Y al vector Artefacto, de la siguiente manera.

$$Artefacto_X = \cos\ \sigma\ * Artefacto \quad (2.\ 1)$$

$$Artefacto_Y = sin\ \sigma\ * Artefacto \quad (2.\ 2)$$

Nota: Para simplificar la formula se ha remplazado el Angulo de Cabeceo de la Figura 2. 17 con el símbolo delta (σ), este ángulo es preestablecido por el usuario. Es el ángulo con el que el brazo se va a apoyar en la coordenada espacial establecida.

A continuación se procede a calcular los elementos **Lado X** y **Lado Y** que son los compontes que ayudan a calcular el elemento **Hipotenusa** y su ángulo alfa (**α**) mostrados en la Figura 2. 16.

$$Lado\ X = Y - Artefacto_X \quad (2.\ 3)$$

$$Lado\ Y = Z + Artefacto_Y - Altura \quad (2.\ 4)$$

$$Hipotenusa = \sqrt{Lado\ X^2 + Lado\ Y^2} \quad (2.\ 5)$$

$$\alpha = tan^{-1}\frac{Lado\ X}{Lado\ Y} \quad (2.\ 6)$$

Con la Ley de Cosenos se procede a realizar los cálculos de los ángulos beta (**β**) y gama (**γ**).

$$\beta = \frac{Brazo^2 - Antebrazo^2 + hipotenusa^2}{2 * Brazo * Hipotenusa} \quad (2.\ 7)$$

$$\gamma = \frac{Brazo^2 + Antebrazo^2 - hipotenusa^2}{2 * Brazo * Antebrazo} \quad (2.\ 8)$$

Al ya tener estos ángulos, se puede saber cuáles son los ángulos que deben girar las articulaciones del brazo robótico:

$$Brazo_{ángulo} = 90° + \beta + \alpha \quad (2.\ 9)$$

$$Antebrazo_{ángulo} = \gamma \quad (2.\ 10)$$

Finalmente se debe conocer el ángulo del artefacto con respecto al brazo robótico, no se debe confundir el ángulo de cabeceo con el ángulo del artefacto, ya que el ángulo de cabeceo es el ángulo con el cual el artefacto se ubica sobre la coordenada cartesiana ingresada, y el ángulo del artefacto es el ángulo calculado con el cual gira el artefacto para llegar a la coordenada deseada, esto se puede ver en la Figura 2. 17.

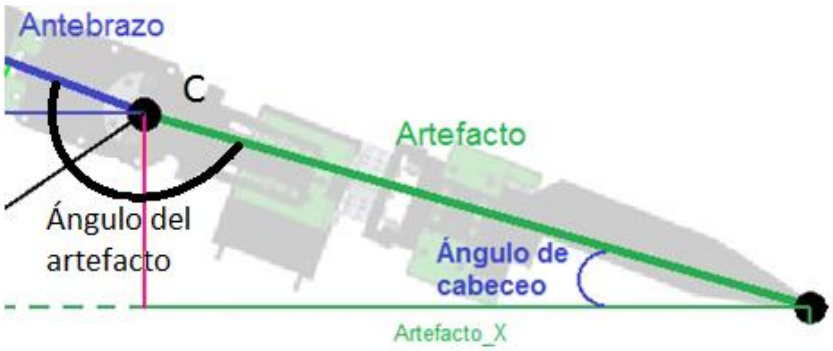


Figura 2. 17.- Diferencia entre el ángulo del Artefacto y el ángulo de cabeceo.

Elaborado por: (Mamarandi J., Velasco E.)

Para calcular el ángulo del Artefacto se deben localizar los triángulos ABC CDE y ACF, de la Figura 2. 16, con las fórmulas de ángulos complementarios de un triángulo se deduce que el ángulo del artefacto es:

$$Artefacto_{ángulo} = 180° - \beta - \gamma + 90° - \alpha + 90 - \sigma \quad (2.\ 11)$$

De esta manera se resuelve los ángulos para ubicar las articulaciones en el plano YZ, para el último dato que se necesita conocer el ángulo de giro de la base para que el brazo robótico se ubique en el espacio (X, Y, Z), para esto se debe conocer el ángulo entre los componentes X y Y del lugar en el espacio donde se van a ubicar, estos componentes no se deben confundir con los valores **Lado X** y **Lado Y** de la Figura 2. 16, ya que estos son los lados del triángulo ACF. A continuación se muestra como se debe realizar el cálculo.

$$Ángulo_{giro} = tan^{-1}\frac{X}{Y} \qquad (2.\ 12)$$

Así se puede ubicar el artefacto del brazo robótico en una coordenada espacial (X, Y, Z).

2.6 Programación en Python.

Al querer desarrollar el sistema de control basado en software libre se empieza por realizar un diagrama de flujo básico para poder entablar una comunicación entre el sistema en este caso un brazo robótico y el controlador que es un ordenador de escritorio. Todo este proceso se simplifica en el diagrama de flujo mostrado en la Figura 2. 18.

Como en todo lenguaje de programación se inicia importando librerías si se las necesita, declarando variables, llamando funciones etc.; debido a que este programa no sigue ningún método de control se dice que es una programación heurística ya que el brazo robótico ejecuta las instrucciones del programa línea por línea.

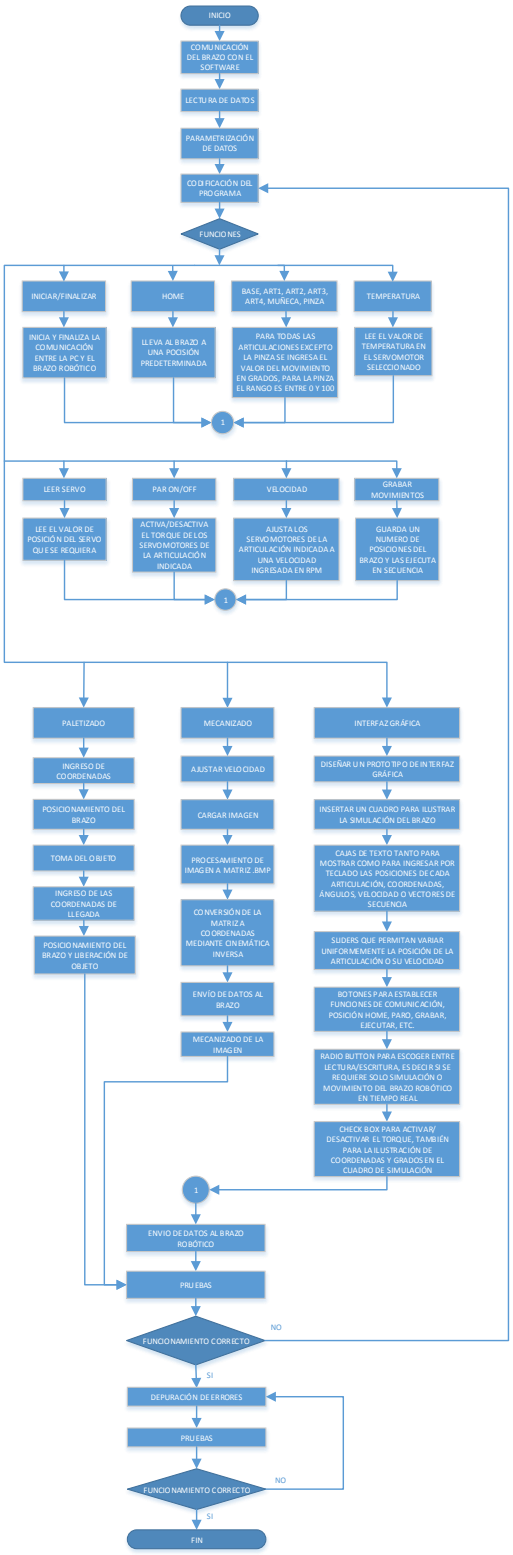


Figura 2. 18.- Diagrama de flujo, secuencia de programación

Elaborado por: (Mamarandi J., Velasco E.)

Todo este proceso se lo realizó en dos partes, una de ellas fue el desarrollo del programa encargado de la comunicación con la red de servomotores, de linealizar el número de datos total de una revolución del servomotor a un rango de valores capaz de ser interpretado por un usuario y que con el mismo pueda realizar varios cálculos y procesos; también en el que se pueda leer los datos entregados por el servomotor como el ID, valor de temperatura, torque, posición, velocidad y entre otros parámetros que dependiendo del modelo de servomotor Dynamixel que se esté utilizando puedan leerse. Por lo tanto en esta primera parte se crearon funciones de lectura y escritura, de donde directamente se puede manipular y configurar la red de servomotores o servo por servo en el brazo robótico.

2.7 Interfaz gráfica.

En la segunda parte del desarrollo se programó en la extensión de Python llamada Vidle Python, en donde tenemos la opción de crear ventanas, botones, figuras, etc. La necesidad de realizar una interfaz gráfica es sumamente importante ya que se requiere de un sistema amigable para usuarios los cuales no tengan conocimientos de programación, entonces en esta interfaz cualquier persona familiarizada con una computadora podrá manipular fácilmente el brazo robótico mediante sliders, botones, cuadros de texto, etc., y observar los cambios realizados tanto en una simulación mostrada en la interfaz (Figura 2. 19), como en el sistema robótico físico mismo.

Para poder enlazar estas dos partes se tiene que importar todas las funciones programadas en la primera parte hacia el archivo de programación de la interfaz gráfica como si se tratara de una librería. A continuación se muestra todas las funcionalidades que posee la Interfaz Gráfica a través del uso de elementos de control.

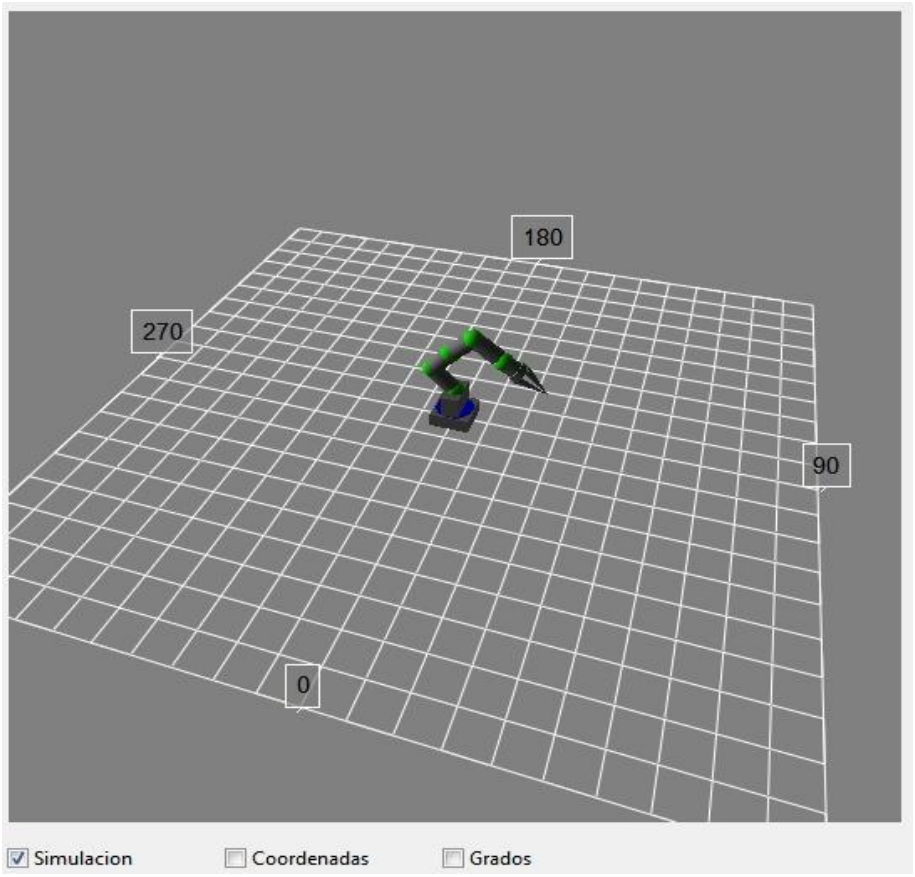


Figura 2. 19.- Simulación en la Interfaz Gráfica

Elaborado por: (Mamarandi J., Velasco E.)

El cuadro de simulación (PARTE A, Figura 2. 30) para la interfaz gráfica, posee tres Check box para habilitar o deshabilitar la simulación, mostrar las coordenadas y los grados en el plano.

Check box Coordenadas: activa cuadros de texto en el extremo de la pinza que muestran la posición de ésta en el espacio en coordenadas X, Y, Z.

En la Figura 2. 20 se indica la selección del Check box y como en el cuadro de simulación en posición extrema de la pinza se muestran las coordenadas de la misma. Estos valores están mostrados en centímetros.

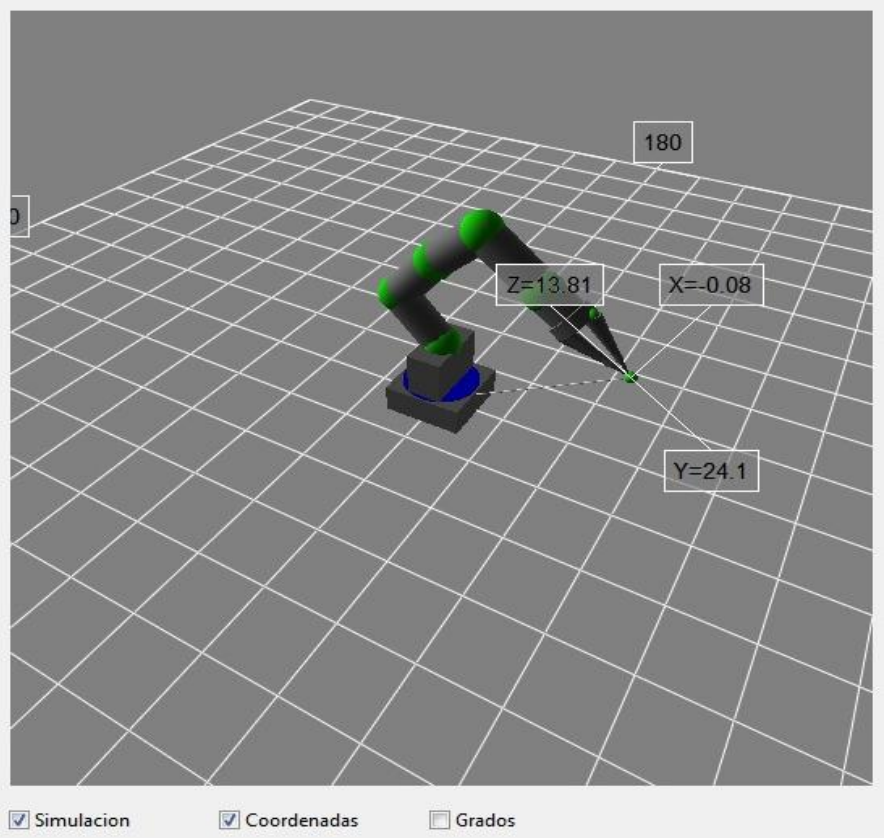


Figura 2. 20.- Interfaz Gráfica señalando las coordenadas de la pinza
Elaborado por: (Mamarandi J., Velasco E.)

Check box Grados: activa una serie de divisiones circulares en intervalos de 10° con un total de 360°, es aconsejable que es el rango máximo en el que el brazo debe trabajar sea de 300°.

En la Figura 2. 21 se observa las divisiones del área de cobertura en el plano X, Y en intervalos de 20°.

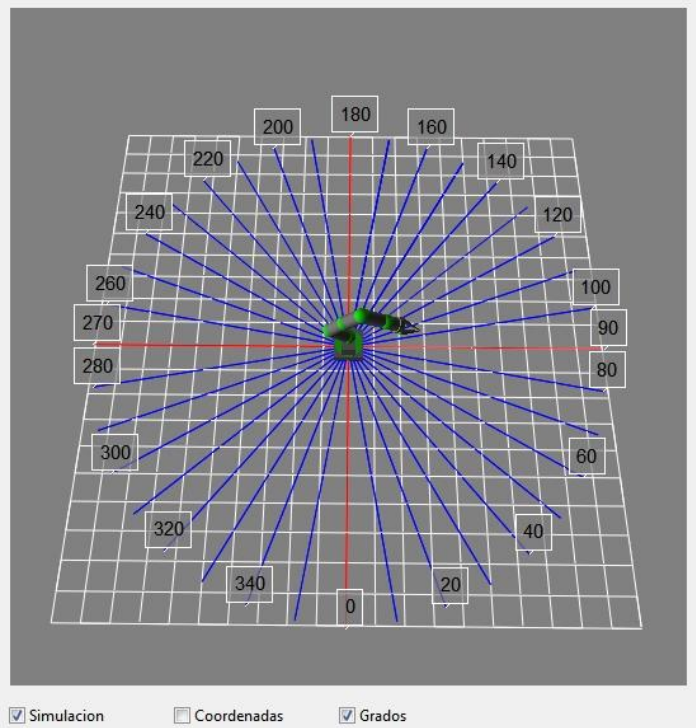


Figura 2. 21.- Interfaz Gráfica mostrando las divisiones en grados

Elaborado por: (Mamarandi J., Velasco E.)

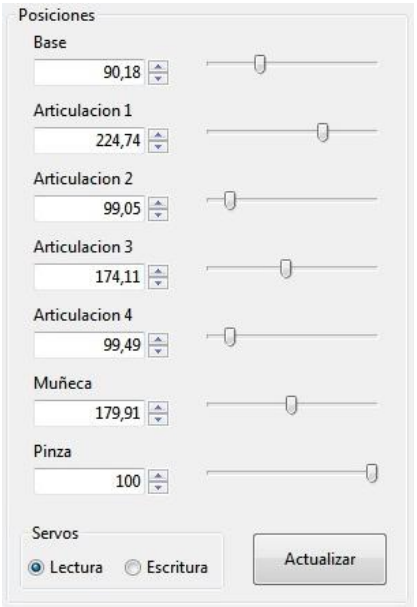


Figura 2. 22.- Funcionalidad Posiciones en la Interfaz Gráfica

Elaborado por: (Mamarandi J., Velasco E.)

En la Figura 2. 22 se muestra la funcionalidad de **Posiciones** en la que se tiene cuadros de texto y sliders que permiten manipular y visualizar la posición del brazo robótico moviendo cada uno de sus grados de libertad (articulaciones) incluyendo la pinza.

También se observa botones de selección, si se selecciona Lectura solo se puede leer los datos provenientes de los servos en el brazo robótico, si se selecciona Escritura los datos que se tiene en los sliders y cuadros de texto se transfieren a los servomotores. El botón Actualizar refresca los datos que se encuentran en ese momento en los servos y los indica en la interfaz gráfica. (PARTE B, Figura 2. 30)

Figura 2. 23.- Funcionalidad Torque en la Interfaz Gráfica

Elaborado por: (Mamarandi J., Velasco E.)

La funcionalidad **Torque** mostrada en la Figura 2. 23 posee Check box de cada articulación que permite activar o desactivar el torque individualemente o en conjunto de cada una de sus articulaciones incluyendo la pinza. (PARTE C, Figura 2. 30)

Figura 2. 24.- Funcionalidad Velocidades en la Interfaz Gráfica.
Elaborado por: (Mamarandi J., Velasco E.)

Los sliders y spinners (Figura 2. 24) que permiten regular la velocidad en términos de porcentaje en los servos de forma individual o grupal para cada articulación se encuentran en la funcionalidad **Velocidades**. (PARTE D, Figura 2. 30)

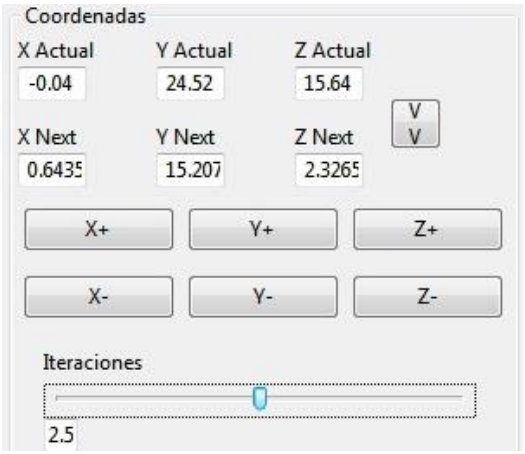


Figura 2. 25.- Funcionalidad Coordenadas en la Interfaz Gráfica.

Elaborado por: (Mamarandi J., Velasco E.)

En la Figura 2. 25 que muestra la funcionalidad de **Coordenadas** los cuadros de texto permiten ingresar valores numéricos para obtener un punto en el espacio formado por coordenadas en (X, Y, Z), los botones permiten el desplazamiento del brazo en el eje que se desea y el slider variar las iteraciones del movimiento del brazo. (PARTE H, Figura 2. 30)

Figura 2. 26.- Funciones en la Interfaz Gráfica.

Elaborado por: (Mamarandi J., Velasco E.)

Algunas de las funciones mostradas en la Figura 2. 26 se explican a continuación:

- COM: inicializa la comunicación entre el brazo robótico y el programa en la PC.
- Home: lleva al brazo robótico a una posición preestablecida, es decir una posición por default.

- Cinemática: toma los valores ingresados a las cajas de texto de Coordenadas y las transforma en valores para que los servomotores actúen y lleven a la pinza a dicha posición en términos de coordenadas de X, Y, Z.
- Paro: finaliza la comunicación establecida entre el brazo robótico y el ordenador.
- Img Líneas: permite cargar una imagen y procesarla para que el brazo robótico realice el grafico con líneas.
- Img Puntos: permite cargar una imagen y procesarla para que el brazo robótico realice el grafico con puntos por cada pixel que la imagen posea. (PARTE F, Figura 2. 30)

Figura 2. 27.- Funcionalidad Guardar Posiciones en la Interfaz Gráfica.

Elaborado por: (Mamarandi J., Velasco E.)

Una de las funcionalidades más importantes en la interfaz gráfica del brazo es la de **Posiciones Guardadas** mostrada en la Figura 2. 27 que permite, como dice el nombre guardar o almacenar posiciones preestablecidas por el operario a donde el brazo se dirigirá, los elementos de control que posee son un cuadro de texto que muestra una lista de

vectores formados por posiciones de cada articulación que el brazo sigue en forma ordenada para completar una secuencia o rutina. También se tiene botones que cumplen diferentes funciones como:

- Grabar: guarda o graba el vector en el que se encuentra posicionado el brazo.
- Borrar Todo: borra toda la secuencia almacenada.
- Borrar: borra un solo vector de la lista de secuencia que puede ser seleccionado por el spinner que se encuentra junto al botón.
- Ejecutar: ejecuta la lista de vectores grabada. Junto a este botón se tiene un spinner para aumentar o disminuir el número de veces que se ejecutará la secuencia.
- Insertar Secuencia: inserta una secuencia en la posición en la que indique el spinner que se encuentra junto al botón.
- Ejecutar Secuencia: ejecuta el vector en la posición indicada por el spinner que se encuentra junto al botón.
- Reemplazar Secuencia: Reemplaza un vector seleccionado por el spinner de la lista de una rutina por otro vector con una secuencia diferente.
- Guardar: Guarda la lista de vectores de una rutina en un archivo .txt.
- Abrir: Abre un archivo .txt que tenga guardada una lista de vectores para realizar una secuencia.
- >>>: Ejecuta toda la secuencia de vectores en orden ascendente. Permite avanzar posición a posición.
- <<<: Ejecuta toda la secuencia de vectores en orden descendente. Permite retroceder posición a posición.
- T1, T2: En estas cajas de texto ingreso los retardos que tendrá una secuencia en milisegundos, T1 es para un retardo entre movimiento de articulaciones, T2 es para un retardo entre vectores. (PARTE G, Figura 2. 30)

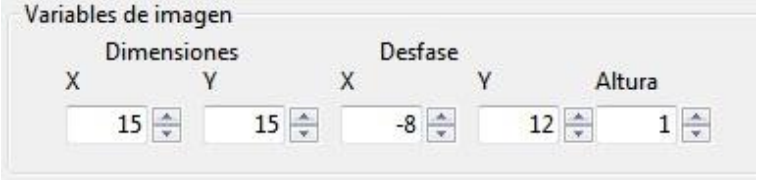


Figura 2. 28.- Funcionalidad Variables de imagen para dimensionar una imagen en la Interfaz Gráfica.

Elaborado por: (Mamarandi J., Velasco E.)

Variables de imagen (Figura 2. 28) es una funcionalidad que permite ingresar los parámetros de dimensionamiento de la imagen a mecanizar, con los spinners de Dimensiones se ingresa los valores de las dimensiones que tendrá el dibujo en centímetros para los ejes X y Y, es decir de qué tamaño será el gráfico dibujado por el brazo robótico.

Con los spinners de Desfase se ingresa la coordenada donde el brazo empieza a dibujar. Y con el spinner de Altura se ingresa el valor del eje Z de donde empezará a graficar. (PARTE I, Figura 2. 30)

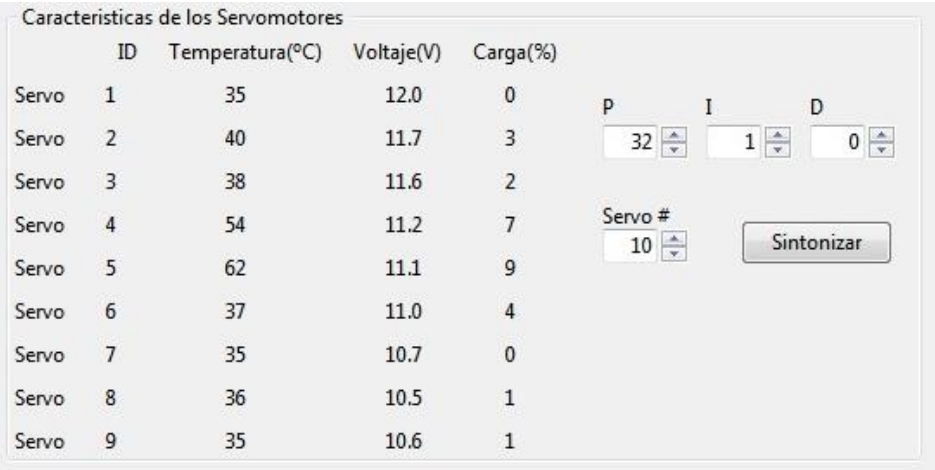


Figura 2. 29.- Tabla de características de los servomotores mostradas en la Interfaz Gráfica y sintonización PID

Elaborado por: (Mamarandi J., Velasco E.)

Características de los Servos (Figura 2. 29), en esta parte se indican las características de cada servomotor que forma parte del brazo robótico, ID, temperatura (°C), voltaje y porcentaje de carga, estos valores solo se pueden leer. Además incorpora un control PID a través de spinners para seleccionar el servomotor y variar los parámetros: proporcional, integral y derivativo y sintonizarlos. (PARTE E, Figura 2. 30)

Tras evaluar el funcionamiento de cada elemento de control programado en la Interfaz Gráfica se procede a reubicarlos en posiciones correctas en la ventana que contendrá toda la interfaz ya que por la lectura de datos que se realiza en ocasiones los cuadros de texto que poseen estos valores sobrepasan el espacio calculado y ocurren errores como que no se muestren todos los datos, texto incompleto o erróneo, entre otros. Al finalizar todo este proceso los resultados se muestran en la Figura 2. 30.

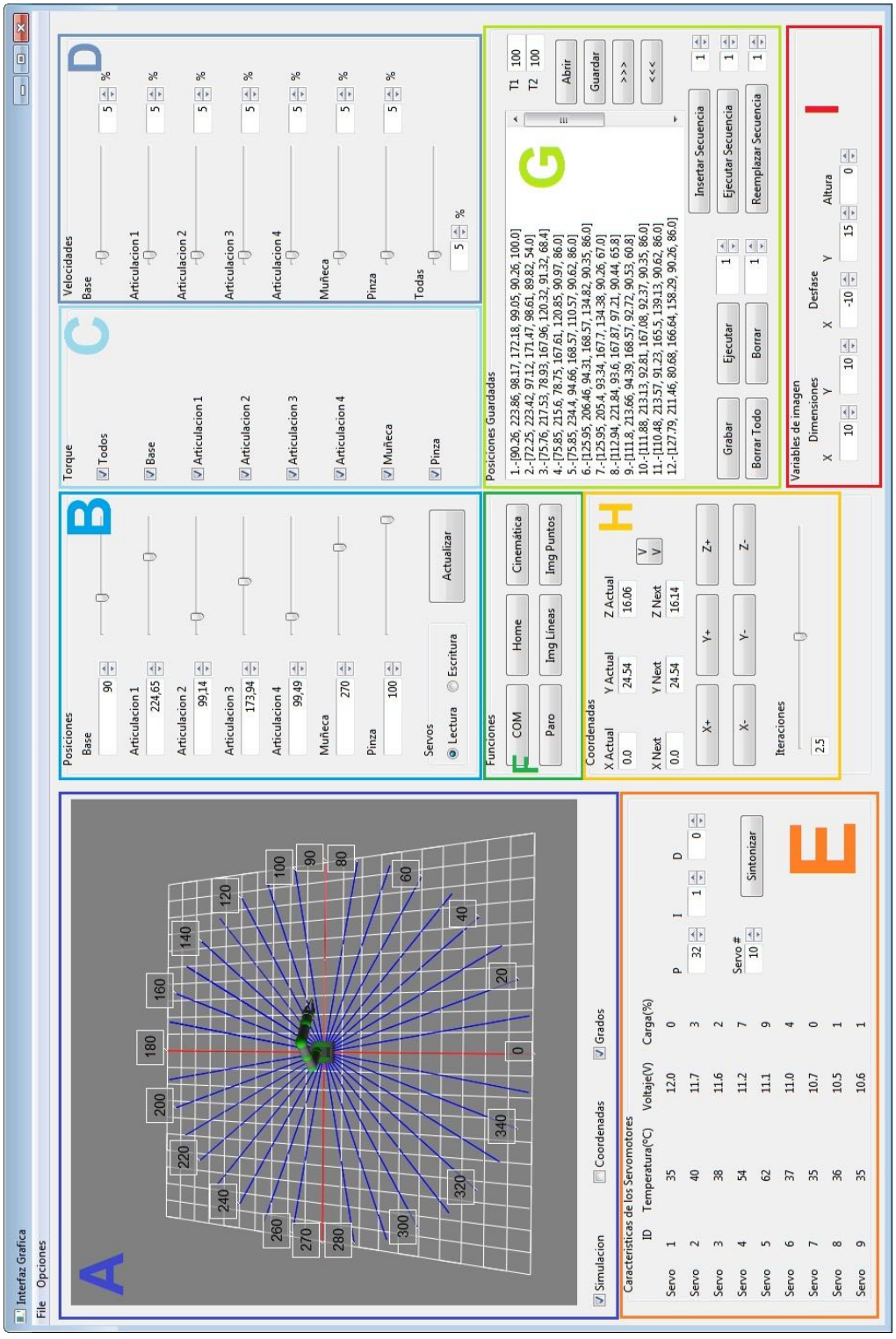


Figura 2. 30.- Interfaz Gráfica, formato final

Elaborado por: (Mamarandi J., Velasco E.)

Figura 2. 31.- Brazo robótico en comunicación con la interfaz gráfica

Elaborado por: (Mamarandi J., Velasco E.)

CAPÍTULO 3

3 IMPLEMENTACIÓN DEL SISTEMA.

El capítulo 3 se centra casi en su totalidad en lo que es la programación de forma estructurada y heurística de las librerías y del código principal en el que se encuentra desarrollada la interfaz gráfica, que permitirá el control de las funcionalidades de mecanizado y paletizado que ejerce el brazo robótico.

3.1 Pruebas de los servomotores Dynamixel.

Como se mencionó en el apartado 2.1, los servomotores Dynamixel ofrecen una gran estabilidad de trabajo y manejo de varios parámetros para su óptimo funcionamiento de acuerdo a las aplicaciones en las que se requiera su funcionamiento, para esto primero se procede a su conexión y realización de pruebas de comunicación.

Para las pruebas de los servomotores Dynamixel se debe tener en cuenta los siguientes puntos:

- Se necesita tener la interfaz USB2Dynamixel conectada a la red de servomotores y el switch en la posición adecuada de la red, en este caso se coloca para la comunicación RS-485 (esto se observa en el apartado 0).
- La fuente de alimentación debe estar conectada y encendida en la red de servomotores de manera adecuada, en esta ocasión se usa una fuente de 12v.

El software que se utiliza para verificar el funcionamiento de los servomotores, además de ayudar en la lectura y modificación de los datos de las memorias EPROM y RAM de cada servo, es el programa RoboPlus, de la organización ROBOTIS.

Los pasos para la comunicación de los servomotores con el software RoboPlus, son los siguientes:

- Iniciar el software e ingresar en Dynamixel Wizard mostrado en la Figura 3. 1.

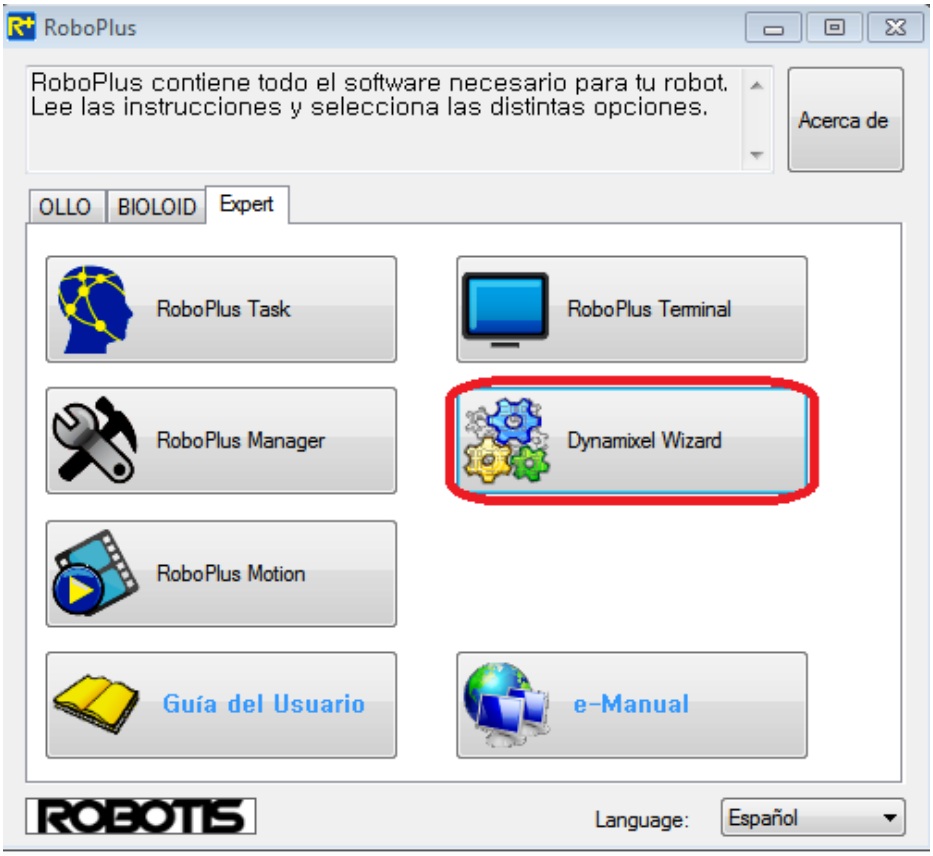


Figura 3. 1.- Ventana de selección del programa RoboPlus
Elaborado por: (Mamarandi J., Velasco E.)

- Al abrirse la ventana se selecciona el puerto COM en donde se instaló la interfaz USB2Dynamixel, y después se abre el puerto en el icono señalado a continuación en la Figura 3. 2.

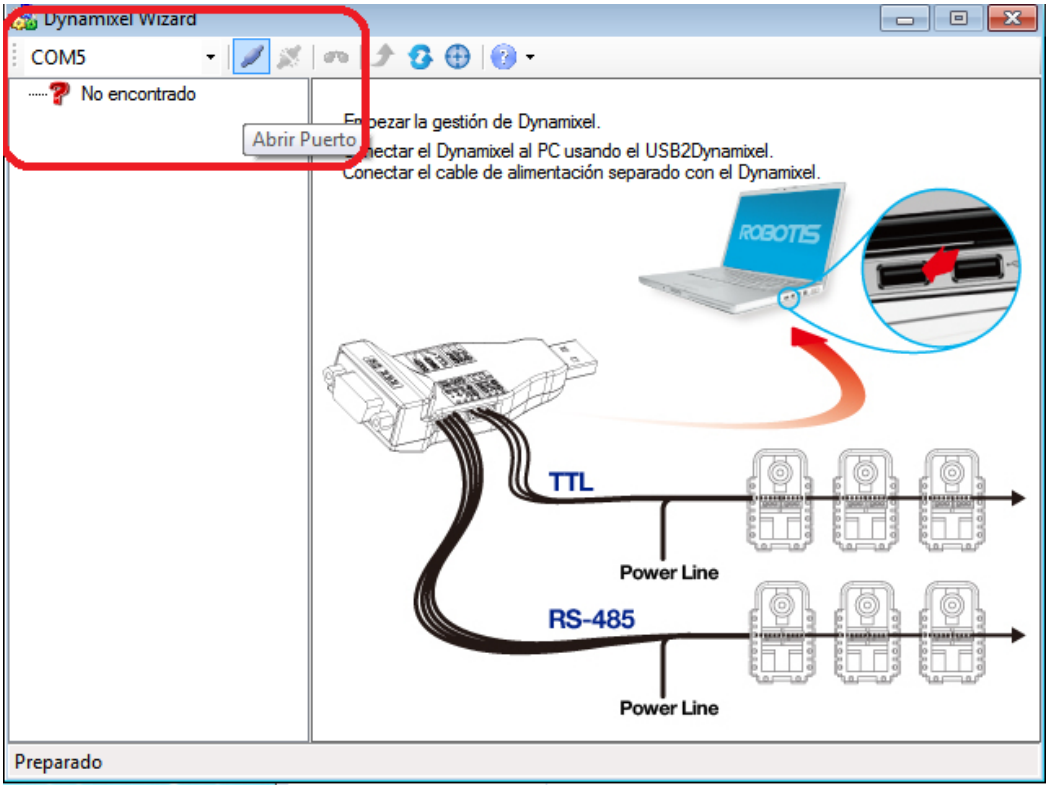


Figura 3. 2.- Selección del puerto de comunicación para la red de servomotores en el programa RoboPlus

Elaborado por: (Mamarandi J., Velasco E.)

- A continuación se coloca la velocidad de transmisión para comunicarse con el controlador, la cual está configurada en cada servomotor. Por defecto cada servomotor viene con una velocidad de 57412 bps. Al presionar en “Empezar a buscar” y seleccionar la velocidad de transmisión se detectan los servomotores conectados.

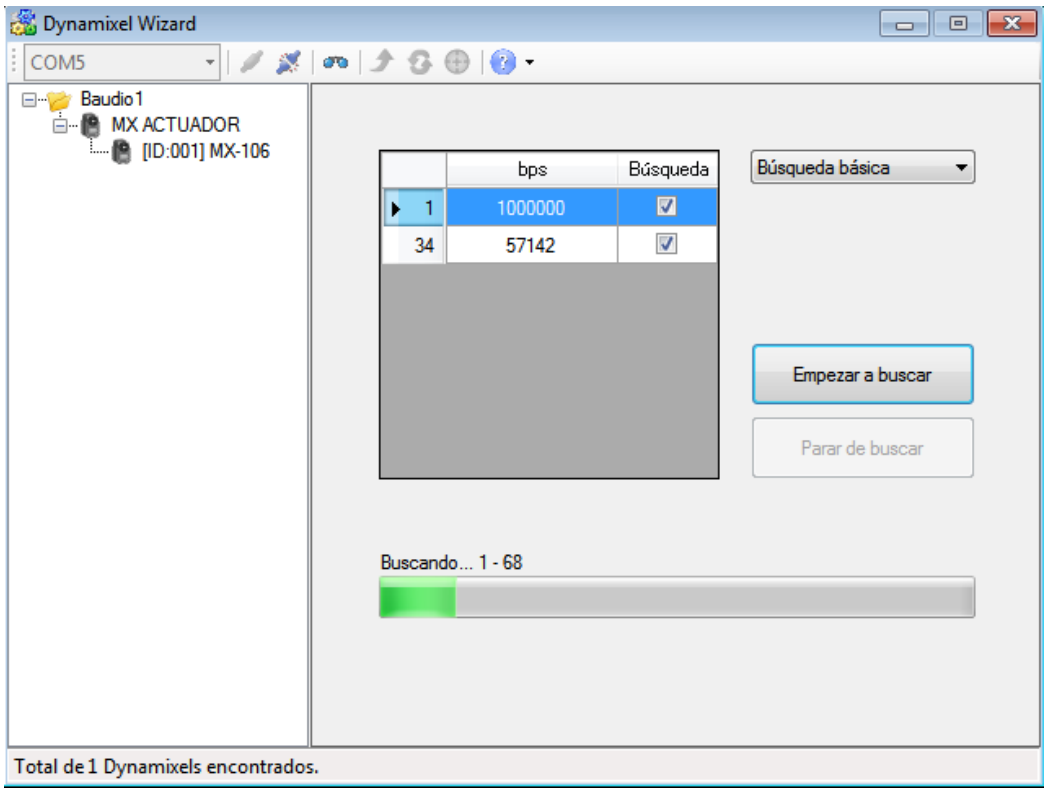


Figura 3. 3.- Detección de servomotores en la red

Elaborado por: (Mamarandi J., Velasco E.)

- Al ya detectar los servomotores conectados podemos leer y escribir los datos en las memorias de cada uno. En el caso del brazo robótico se modificó cada dirección ID de forma ascendente desde el numero 1 hasta el 9 y también se cambió la velocidad de transmisión de 57412 bps a 1000000 bps.

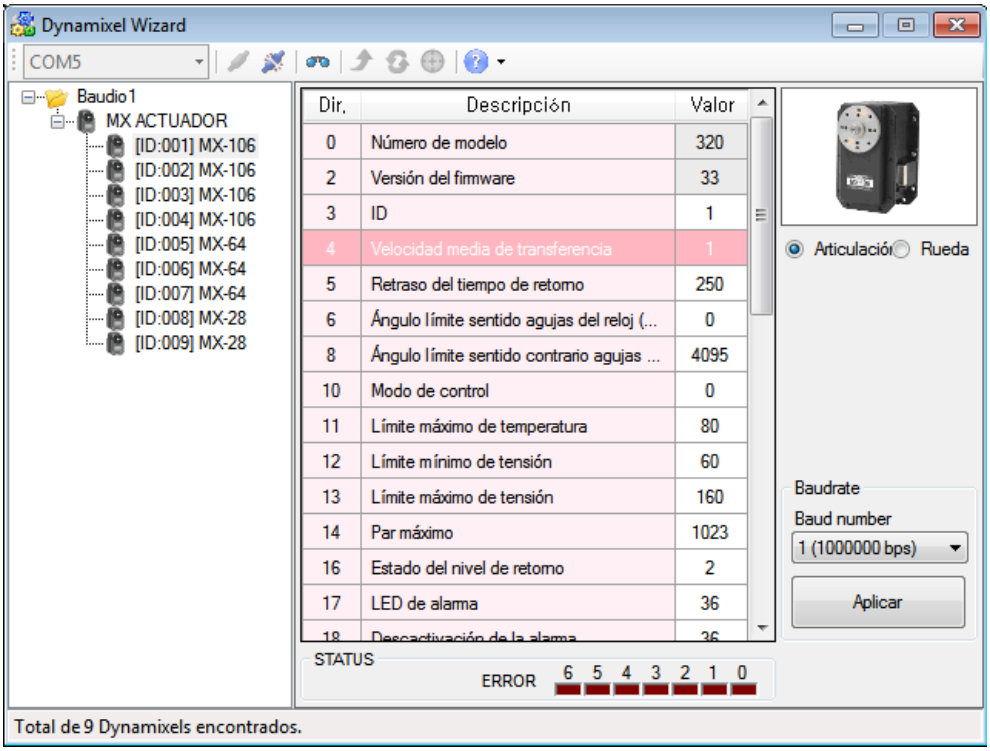


Figura 3. 4.- Lectura de las celdas de memoria de los servomotores

Elaborado por: (Mamarandi J., Velasco E.)

De esta manera se pueden modificar y leer los valores de posición, torque, velocidad de movimiento, tensión actual, etc.; y así se comprueba si los servomotores están en buenas condiciones.

3.2 Pruebas de las librerías desarrolladas en Python.

En todo código de programación existen redundancias, esto quiere decir que si no se sigue de una manera adecuada un diagrama de flujo preestablecido o una línea de secuencia de trabajo se corre el riesgo de sobrescribir el código, inutilizar líneas de programación, gastar ciclos de procesamiento, volver a repetir instrucciones codificadas con anterioridad, entre otras.

Es por esta razón que se requiere de un análisis del código a través de pruebas o chequeos visuales para corregir los errores antes mencionados.

3.2.1 Funciones programadas

Las librerías son extensiones importantes o conjuntos de implementaciones funcionales que ofrecen la posibilidad de enlazar funciones o código necesarios en algún programa. Hablando en términos generales es como llamar a una biblioteca y pedir un libro con instrucciones para poder realizar una actividad o aplicación.

```
import time
from math import *
from ctypes import *

libc = windll.dynamixel # doctest: +WINDOWS
```

Figura 3. 5.- Importación de librerías

Elaborado por: (Mamarandi J., Velasco E.)

En la Figura 3. 5 se muestra un ejemplo de algunas de las librerías importadas para el desarrollo del código.

- Math.- importa todo lo relacionado a cálculos matemáticos.
- Time.- contiene funciones relacionadas al tiempo y retardos.
- Ctype.- importa el CDLL, y en Windows también windll y objetos oledll para cargar librerías dinámicas.

La variable **libc** sirve para establecer un nombre o alias a la biblioteca de enlace dinámico **windll.dynamixel** (Anexo B, Dynamixel_Lib.py, línea 5), esta biblioteca o librería permite el enlace con la memoria RAM y EPROM de los dispositivos de la marca Dynamixel de esta manera se puede leer o escribir datos de las celdas de memoria.

```
art0_cm=11.5
art1_cm=9.5
art2_cm=9.0
art3_cm=9.0
munieca_cm=10.5
pinza_cm=15.5
art4_cm=munieca_cm+pinza_cm
```

Figura 3. 6.- Declaración de constantes

Elaborado por: (Mamarandi J., Velasco E.)

Definición de constantes que establecen un alias para cada articulación del brazo robótico, en este proyecto se igualó a la distancia que tiene entre cada articulación en unidades de centímetros.

```
def iniciar():#"Iniciando comunicacion..."
    n = libc.dxl_initialize()
    if n == 0:#"Error al abrir el USB2Dynamixel!"
        exit()#"USB2Dynamixel abierto satisfactoriamente!"
def home(evt):#"Posicion Inicial"
    ejecutar_mov([[90, 225, 100, 175, 100, 180, 100]],10,0,0);
def led(ID,estado):
    if estado==0:
        libc.dxl_write_byte(ID,25,0);
    if estado==1:
        libc.dxl_write_byte(ID,25,1);
```

Figura 3. 7.- Ejemplos de funciones codificadas en el programa de control del brazo robótico

Elaborado por: (Mamarandi J., Velasco E.)

Para definir funciones como se indicó en el apartado 1.10 referente a programación en Python se observa en la Figura 3. 7 que se utiliza el término **def**.

Un ejemplo a indicar es la función **iniciar**, la cual iguala la variable **n** a la celda de memoria **libc.dxl_initialize()** y retorna un dato de dos estados siendo en este caso 0 para cuando no se ha establecido comunicación y 1 para cuando entabló comunicación con el dispositivo USB2Dynamixel; se realiza una comparación con el condicionante **if** y muestra un mensaje según el dato retornado.

También existen funciones en las cuales se requiere el ingreso de parámetros que modifiquen el funcionamiento del brazo robótico, u otras que retornen algún dato como vectores o matrices. A continuación se presenta una lista de todas las funciones desarrolladas en la librería, **Dynamixel_Lib.py**, y como es su funcionamiento:

- **iniciar():** establece la comunicación entre el programa de la PC con el brazo robótico a través de la interfaz USB2Dynamixel. (Anexo B, Dynamixel_Lib.py, línea 16)
- **home(evt):** lleva al brazo robótico a una posición preestablecida. (Anexo B, Dynamixel_Lib.py, línea 23)
- **led(ID,estado):** se ingresan los datos del ID del dispositivo Dynamixel, en este caso servomotores, y el estado comprendiéndose como 1 encender o 0 apagar el led del dispositivo indicado. (Anexo B, Dynamixel_Lib.py, línea 25)
- **leds(ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9):** esta función enciende o apaga los leds de los 9 servomotores que forman parte del brazo robótico, es decir si se ingresa leds(1,1,1,1,0,0,0,0,0) los leds de los dispositivos con ID del 1 al 4 se encienden y los dispositivos con ID del 5 al 9 se apagan. (Anexo B, Dynamixel_Lib.py, línea 31)
- **enMovimiento(ID):** al ingresar el ID del dispositivo deseado retorna un dato de dos estados en donde 1 indica que el servomotor está en movimiento y 0 que no lo está. (Anexo B, Dynamixel_Lib.py, línea 69)
- **enMovimientoAll():** lee la celda de memoria de movimiento de todos los servomotores y entrega un vector en el cual muestra si el servo se encuentra en movimiento o no. (Anexo B, Dynamixel_Lib.py, línea 74)
- **enMovimientoAll_dato(evt):** entrega un dato único y general de todo el brazo, si al menos un servomotor se encuentra en movimiento el dato retornado es 1, caso contrario, si todos los

servomotores se encuentran detenidos dato retornado es 0. (Anexo B, Dynamixel_Lib.py, línea 74)

- **temperatura(ID):** al ingresar el ID del servomotor requerido el programa retorna el valor de la temperatura a la que se encuentra dicho dispositivo. (Anexo B, Dynamixel_Lib.py, línea 77)
- **temperaturaAll():** el programa devuelve una matriz con los valores de temperatura correspondiente a cada servomotor. (Anexo B, Dynamixel_Lib.py, línea 79)
- **carga_direccion(ID):** lee la celda de memoria en la que se encuentra el valor de carga de un servomotor, dicho valor se encuentra en el rango de 0 a 2048, si el valor leído es mayor a 1023 la dirección es en sentido horario caso contrario es antihorario. (Anexo B, Dynamixel_Lib.py, línea 82)
- **baseg(grados_base):** se ingresa el dato que se encuentra en el rango de 0 a 300, y ubica la base del brazo en esa posición. (Anexo B, Dynamixel_Lib.py, línea 92)
- **art_1(grados_art1):** se ingresa el valor en grados para mover la articulación 1, por los resultados obtenidos en las pruebas se limitó el rango entre 74 y 290. Los servomotores que entran en funcionamiento para esta articulación son los que tienen ID 2 y 3. (Anexo B, Dynamixel_Lib.py, línea 101)
- **art_2(grados_art2):** se ingresa el valor en grados para mover la articulación 2, por los resultados obtenidos en las pruebas se limitó el rango entre 74 y 290. El servomotor que entran en funcionamiento para esta articulación es el que tiene ID 4. (Anexo B, Dynamixel_Lib.py, línea 112)
- **art_3(grados_art3):** se ingresa el valor en grados para mover la articulación 3, por los resultados obtenidos en las pruebas se limitó el rango entre 74 y 290. El servomotor que entran en funcionamiento para esta articulación es el que tiene ID 5. (Anexo B, Dynamixel_Lib.py, línea 121)

- **art_4(grados_art4):** se ingresa el valor en grados para mover la articulación 4, por los resultados obtenidos en las pruebas se limitó el rango entre 74 y 290. El servomotor que entran en funcionamiento para esta articulación es el que tiene ID 6. (Anexo B, Dynamixel_Lib.py, línea 130)
- **munieca(grados_munieca):** se ingresa el valor en grados para mover la articulación de la muñeca, por los resultados obtenidos en las pruebas el rango permitido es de 0 a 359. El servomotor que entran en funcionamiento para esta articulación es el que tiene ID 7. (Anexo B, Dynamixel_Lib.py, línea 139)
- **pinza(cierre):** se ingresa el valor de cierre limitado entre 0 y 100 en donde linealmente la pinza se cerrará . Los servomotores que entran en funcionamiento para esta articulación son los que tienen ID 8 y 9. (Anexo B, Dynamixel_Lib.py, línea 148)
- **delay(Tiempo):** genera una pausa en el programa, el parámetro es ingresado en milisegundos. (Anexo B, Dynamixel_Lib.py, línea 160)
- **leer_servo(articulacion):** se ingresa el nombre de la articulación de la cual se requiere la posición del o los servomotores que las componen. El valor retornado es de 0 a 360 grados. (Anexo B, Dynamixel_Lib.py, línea 167)
- **leer_encoder_all():** retorna una matriz con los valores de los encoders de todos los servomotores, estos valores se encuentran en el rango de 0 a 4096. (Anexo B, Dynamixel_Lib.py, línea 191)
- **leer_servos_all():** retorna una matriz con la posición leída de cada servomotor en unidades de grados. (Anexo B, Dynamixel_Lib.py, línea 207)
- **parOn(articulacion):** activa el torque o par motor de los servomotores que componen la articulación ingresada. (Anexo B, Dynamixel_Lib.py, línea 226)

- **parOff(articulacion):** desactiva el torque o par motor de los servomotores que componen la articulación ingresada. (Anexo B, Dynamixel_Lib.py, línea 246)
- **velocidad(articulacion,rpm):** establece un valor de velocidad en un porcentaje comprendido entre 0 y 100% de su velocidad total a los servomotores que componen la articulación indicada. (Anexo B, Dynamixel_Lib.py, línea 291)
- **velocidadAll(rpm):**establece un valor de velocidad en un porcentaje comprendido entre 0 y 100% de su velocidad total a todos servomotores que componen el brazo robótico. (Anexo B, Dynamixel_Lib.py, línea 314)
- **leerVelocidad(articulacion):** lee la velocidad a la que se encuentra una determinada articulación. (Anexo B, Dynamixel_Lib.py, línea 328)
- **leerVelocidadAll():** lee la velocidad a la que se encuentran todas las articulaciones del brazo robótico. (Anexo B, Dynamixel_Lib.py, línea 346)
- **leer_errores(evt):** determina si existe un error en la red de servomotores, estos errores pueden ser por sobrecarga, límite de ángulo, fuera de rango, error de instrucción de código, error en límites de voltaje, de corriente y de Cheksum que se utiliza para comprobar si el paquete está dañado durante la comunicación. (Anexo B, Dynamixel_Lib.py, línea 372)
- **grabar_mov(movimientos):** ingresado el número de movimientos como parámetro, la función se encarga de grabar la posición de cada articulación en la que se encuentra el brazo robótico para guardarla en una lista. (Anexo B, Dynamixel_Lib.py, línea 404)
- **ejecutar(vector):** ejecuta un vector que contiene las posiciones guardadas de las articulaciones del brazo robótico. (Anexo B, Dynamixel_Lib.py, línea 424)

- **ejecutar_mov(matriz,velocidad,tiempo1,tiempo):** esta función ejecuta una secuencia de movimientos para lo cual contiene cuatro parámetros a ingresar, los cuales son:
    - Matriz.- esta contiene una secuencia de vectores compuestos cada uno de estos de posiciones de las articulaciones del brazo robótico.
    - Velocidad.- establece una velocidad para todos los servomotores.
    - Tiempo1.- es el tiempo de retardo en milisegundos que se encuentra entre cada posición dentro de cada vector.
    - Tiempo2.- es el tiempo de retardo en milisegundos que se encuentra entre cada vector dentro de la matriz.

    (Anexo B, Dynamixel_Lib.py, línea 436)

- **ejecutar_mov2(matriz,vel_vector,tiempo1,tiempo):** esta función ejecuta una secuencia de movimientos, a diferencia de la función **ejecutar_mov,** esta puede configurar las velocidades de cada una de las articulaciones con el dato **vel_vector,** que es un vector de velocidad en valores de porcentaje. (Anexo B, Dynamixel_Lib.py, línea 458)
- **distancia():** es una función la cual retorna un vector con cuatro datos respecto a la posición del brazo robótico los cuales son: distancia en X, Y, el módulo que forman estos parámetros medidos desde el extremo de la pinza a la base del brazo y también el ángulo que forma el módulo con respecto al eje X. (Anexo B, Dynamixel_Lib.py, línea 488)
- **parOn_All():** activa el torque o par motor de todos los servomotores. (Anexo B, Dynamixel_Lib.py, línea 279)
- **parOff_All():** desactiva el torque o par motor de todos los servomotores. (Anexo B, Dynamixel_Lib.py, línea 266)

- **finalizar():** finaliza la comunicación con el dispositivo USB2Dynamixel. (Anexo B, Dynamixel_Lib.py, línea 163)

Algunas de las funciones codificadas en el programa vuelven a utilizar funciones ya programadas para optimizar el código.

La librería **imagen_dynamixel** es una librería creada que permite convertir una imagen en una matriz de coordenadas cartesianas (X, Y), indicando cada pixel de la imagen con una coordenada.

```
from numpy import *
import easygui as eg
import ctypes,cv2
```

Figura 3. 8.- Librerías importadas para el desarrollo de la librería **imagen_dynamixel.**

Elaborado por: (Mamarandi J., Velasco E.)

- Numpy.- Permite el manejo de matrices n-dimensionales, álgebra lineal, transformada de Fourier, y capacidades de generar números aleatorios. Esta librería es esencial para el funcionamiento de la librería **cv2** de OpenCV**.**
- Easygui.- es importada con la finalidad de mejorar la interfaz y ayudar con ventanas emergentes. Es importada como **eg** para facilitar su tipeado.
- Cv2.- es una librería que permite al software Python trabajar con imágenes. También conocida como librerías OpenCV que son usadas para el desarrollo de visión artificial.
- Ctypes.-está citada en la Figura 3. 5.

La librería **imagen_dynamixel** está compuesta por dos funciones que son: **dibujo_pixeles** y **dibujo_lineas**; a continuación se detalla su funcionamiento:

- **dibujo_pixeles(ancho,alto,desfaseX,desfaseY).-** La función abre una imagen RGB y la convierte en un mapa de Bits (1,0) en donde se estableció que los 1 son pixeles de color negro y los 0 son pixeles blancos.

  Cada pixel negro se convierte en una coordenada en pares ordenados con la ayuda de los índices (i,j) de la matriz de 1 y 0, dando así un puesto de cada par ordenado al igual que la posición de cada número "1" de la matriz, y con los datos de ancho y alto que se ingresan como datos de la función; además se suma a cada pixel parametrizado en una coordenada el desfase en X y Y que también son datos de la función (…**desfaseX,desfaseY).** (Anexo B, imagen_dynamixel.py, línea 5)

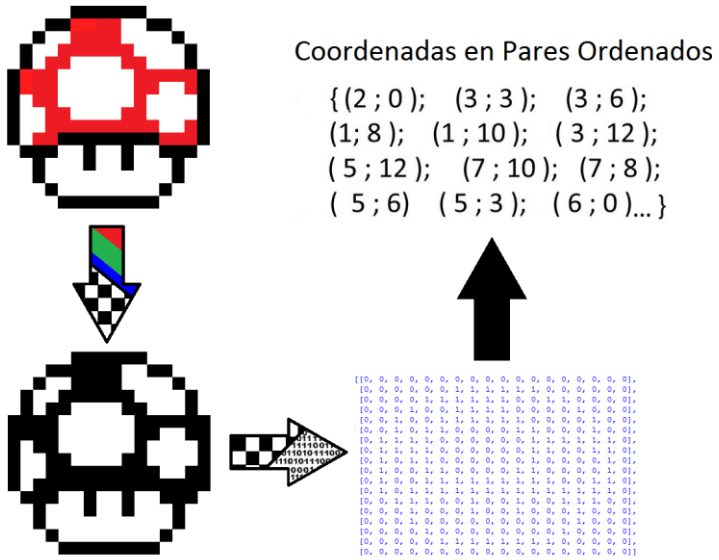


Figura 3. 9.- Explicación gráfica del procesamiento de imágenes de la librería **dibujo_pixeles.**

Elaborado por: (Mamarandi J., Velasco E.)

- **dibujo_lineas(ancho,alto,desfaseX,desfaseY).-** Esta función se basa en la ya mencionada **dibujo_pixeles,** en el procesamiento de la imagen a coordenadas en pares ordenados, pero con la diferencia de que los pares ordenados se entregan en un orden diferente, ya

no en el orden de los índices (i,j) de cada pixel, si no que se da en el orden de los pixeles de color negro, o números “1” más cercanos. De esta manera le permite al brazo realizar un dibujo por medio de líneas y no por pixeles. (Anexo B, imagen_dynamixel.py, línea 36)

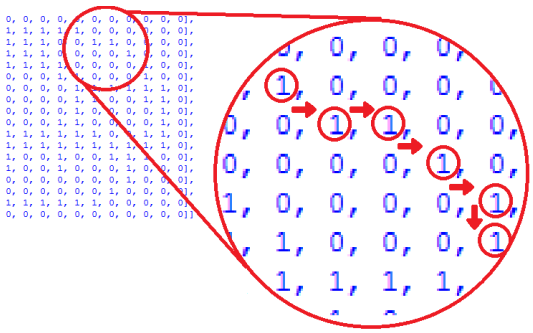


Figura 3. 10.- Proceso de orden de los pares ordenados de la librería **dibujo_linea.**

Elaborado por: (Mamarandi J., Velasco E.)

3.3 Desarrollo de la interfaz gráfica en Python.

Las librerías mostradas en la Figura 3. 5 y Figura 3. 8 son importadas desde extensiones propias de la plataforma de programación como son, **visual**, **easygui**, **ctypes**, **wx** y **math.**

- Visual.- Es una librería que está relacionada con el desarrollo de objetos en 3D, estos se crearon en la biblioteca VPython, que es un conjunto de librerías de software libre compatibles con Python.
- Wx.- Es una librería compatible con Windows y Python que permite que los botones, sliders, check box, spinners, etc. se vean igual que los entornos de Windows.

```
from Dynamixel_Lib import *
from  visual import *
from imagen_dynamixel import *
import easygui as eg
import ctypes, wx, math
```

Figura 3. 11.- Librerías importadas para el código de la Interfaz Gráfica

Elaborado por: (Mamarandi J., Velasco E.)

Las librerías **Dynamixel_lib** e **imagen_dynamixel** fueron desarrolladas durante el transcurso de este proyecto las cuales contienen las funciones antes mencionadas en el apartado 3.2.1. A continuación se detalla cómo se elaboró el programa **Interfaz Grafica.py**

```
###constantes
n=0
display_dimensiones=wx.GetDisplaySize()
pasos=[]
esfera=[(0,0),(0,0),(0,0),(0,0),(0,0),(0,0),(0,0)]
art0_cm=11.5
art1_cm=9.5
art2_cm=9.0
art3_cm=9.0
munieca_cm=10.5
pinza_cm=15.5
art4_cm=munieca_cm+pinza_cm
```

Figura 3. 12.- Constantes declaradas para la Interfaz Gráfica

Elaborado por: (Mamarandi J., Velasco E.)

Las constantes declaradas son necesarias para la creación de la pantalla de trabajo y de los objetos en 3D que vamos a utilizar para simular al brazo robótico. (Anexo B, Interfaz Grafica.py, línea 22-linea 27)

```
const_screen=display_dimensiones[0]/display_dimensiones[1]
d = 300*const_screen
d2=d/2
borde_x=12*const_screen
w = window(x=(display_dimensiones[0]-1400)/2,y=(display_dimensiones[1]-936)/2,
           width=1400, height=936, menus=True, title='Interfaz Grafica')
pantalla=display(window=w, x=borde_x, y=borde_x, width=d, height=d,background=(0.5,0.5,0.5))
```

Figura 3. 13.- Constantes declaradas para la Pantalla de trabajo y los comandos **window** y **display** para la creación de la Interfaz gráfica.

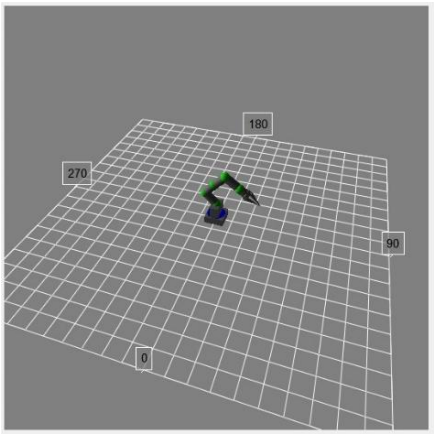
Elaborado por: (Mamarandi J., Velasco E.)

El comando **window** permite crear la ventana en donde se va a desarrollar todo la interfaz gráfica, este necesita los parámetros de **X** y **Y** los cuales son las coordenadas en donde se va a ubicar la pantalla, además de **width**, **height** para determinar el alto y ancho de la pantalla, **tittle** para nombrar a la ventana y **menus**, que habilita la opción de poseer menús en la ventana. (Anexo B, Interfaz Grafica.py, línea 26)

La línea de código con la palabra **display**, es la que crea la ventana en la cual se van a observar todos los objetos sólidos que representan el brazo robótico, los parámetros que necesitan son: **width**, **height** para determinar el alto y ancho, **X** y **Y** para ubicar a la pantalla en la ventana, **window** para saber en cual ventana se colocara esta y **background** que determina el color de fondo de la pantalla, al poner los valores en (0.5,0.5,0.5) se establece un color gris. (Anexo B, Interfaz Grafica.py, línea 27)

```
###Plano##
plane = curve(pos=[(-100,0,100), (100,0,100), (100, 0,-100), (-100, 0, -100),
                   (-100,0,100),(-90,0,100),(-90,0,-100),(-80,0,-100),(-80,0,100),
                   (-70,0,100),(-70,0,-100),(-60,0,-100),(-60,0,100),(-50,0,100),(-50,0,-100),
                   (-40,0,-100),(-40,0,100),(-30,0,100),(-30,0,-100),(-20,0,-100),(-20,0,100),
                   (-10,0,100),(-10,0,-100),(0,0,-100),(0,0,100),(10,0,100),(10,0,-100),
                   (20,0,-100),(20,0,100),(30,0,100),(30,0,-100),(40,0,-100),(40,0,100),
                   (50,0,100),(50,0,-100),(60,0,-100),(60,0,100),(70,0,100),(70,0,-100),
                   (80,0,-100),(80,0,100),(90,0,100),(90,0,-100),(100,0,-100),(100,0,-90),(-100,0,-90),
                   (-100,0,-80),(100,0,-80),(100,0,-70),(-100,0,-70),(-100,0,-60),(100,0,-60),
                   (100,0,-50),(-100,0,-50),(-100,0,-40),(100,0,-40),(100,0,-30),(-100,0,-30),
                   (-100,0,-20),(100,0,-20),(100,0,-10),(-100,0,-10),(-100,0,0),(100,0,0),
                   (100,0,10),(-100,0,10),(-100,0,20),(100,0,20),(100,0,30),(-100,0,30),
                   (-100,0,40),(100,0,40),(100,0,50),(-100,0,50),(-100,0,60),(100,0,60),
                   (100,0,70),(-100,0,70),(-100,0,80),(100,0,80),(100,0,90),(-100,0,90)],display=pantalla)
```

a)



b)

Figura 3. 14.- a) Código utilizado para definir un plano visual; b) Ventana creada a partir del código para la simulación

Elaborado por: (Mamarandi J., Velasco E.)

La matriz de coordenadas que grafica el plano en donde se realizará la simulación del brazo robótico, crea una cuadrícula como la que se muestra en la Figura 3. 14, esto se logra con el comando **curve**, que

genera una recta de 2D uniendo los puntos en el espacio (X, Y, Z) que se establezca en la línea de código. (Anexo B, Interfaz Grafica.py, línea 31). En el desarrollo de la interfaz gráfica se vio la necesidad de establecer varios elementos de control con el cual se tenga la posibilidad de manipular datos tanto en escritura como en lectura. Por tal razón a continuación se explica el algoritmo seguido e instrucciones utilizadas para crear los controles necesarios.

```
#######botones

b_Actualizar = wx.Button(w.panel,label='Actualizar', pos=(760,397),size=(100,40))
b_Home = wx.Button(w.panel,label='Home', pos=(683,478))
b_Comunicacion = wx.Button(w.panel,label='COM',pos=(588,478))
```

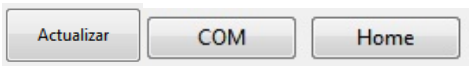


Figura 3. 15.- Declaración de botones
Elaborado por: (Mamarandi J., Velasco E.)

Para crear un botón se utiliza la siguiente estructura en donde a una variable se iguala la instrucción:

wx.Button(w.panel,label=' ',pos=(),size=( ))

Entre los apostrofes de **label** se escribe el nombre que se mostrará en el botón, entre los paréntesis de pos se ingresa las coordenadas de la posición en la que se requiere establecer y entre los paréntesis de **size** se da el tamaño que tendrá el dicho elemento. (Anexo B, Interfaz Grafica.py, línea 105)

```
#### RadioBox
t1 = wx.RadioBox(w.panel,pos=(590,390), size=(150,50),
                 choices = ['Lectura', 'Escritura'], style=wx.RA_SPECIFY_COLS,label='Servos')
```

Figura 3. 16.- Declaración de Radio Box
Elaborado por: (Mamarandi J., Velasco E.)

Para un Radio Box o botón selector se sigue casi la misma estructura que con los botones, se le da su posición y tamaño, los parámetros que cambian son, **choices** al que se le da un vector de cadena de caracteres que contenga las opciones para seleccionar, el estilo que se utilizó es **wx.RA_SPECIFY_COLS**, este estilo es por lo general el botón seleccionador en forma de circunferencia. (Anexo B, Interfaz Grafica.py, línea 135)

```
###spin
veces=wx.SpinCtrlDouble(w.panel,pos=(1085,700),size =(60,25),initial =1,)
borrar_secuencia=wx.SpinCtrlDouble(w.panel,pos=(1085,730),size =(60,25),initial =1,min=1)
```

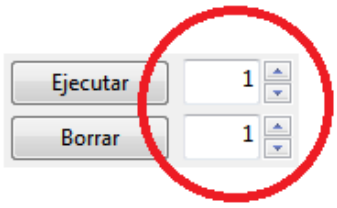


Figura 3. 17.- Declaración de Spinners

Elaborado por: (Mamarandi J., Velasco E.)

La declaración de un spinner es la siguiente:

wx.SpinCtrlDouble(w.panel,label=' ',pos=(),size=( ),initial,min,max)

Los spinners son utilizados para incrementar o disminuir un valor numérico, su estructura está compuesta por posición, tamaño y valor inicial, el valor mínimo se lo puede omitir ya que por defecto es 0, pero si se desea cambiar se utiliza el parámetro **min** y se lo iguala al valor que se requiera como mínimo.

En la interfaz gráfica se los utiliza para recorrer valores tanto enteros como decimales, todo depende del incremento que se establezca en cada spinner, por ejemplo para seleccionar un vector en una posición específica que se requiera reemplazar, borrar, insertar o ejecutar en la funcionalidad

de **Posiciones Guardadas** o para valores decimales en el incremento de la velocidad de cada servomotor. (Anexo B, Interfaz Grafica.py, línea 139)

```
###StaticBox
wx.StaticBox(w.panel,pos=(880,10), size=(190,440),label='Torque')
wx.StaticBox(w.panel,pos=(880,455), size=(485,315),label='Posiciones Guardadas')
wx.StaticBox(w.panel,pos=(580,10), size=(295,440),label='Posiciones')
wx.StaticBox(w.panel,pos=(1075,10), size=(280,440),label='Velocidades')
wx.StaticBox(w.panel,pos=(580,455), size=(295,100),label='Funciones')
wx.StaticBox(w.panel,pos=(580,560), size=(295,306),label='Coordenadas')
wx.StaticBox(w.panel,pos=(25,600), size=(530,265),label='Servos Dynamixel Caracteristicas')
wx.StaticBox(w.panel,pos=(880,776), size=(485,90),label='Variables de imagen')
```

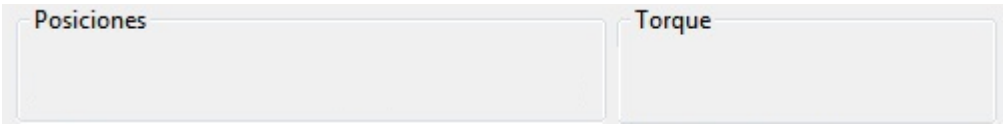


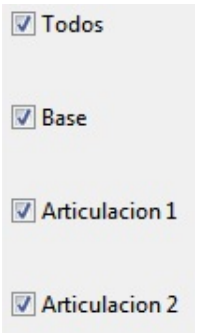
Figura 3. 18.- Declaración de Static Box
Elaborado por: (Mamarandi J., Velasco E.)

Los Static Box son cajas estáticas o contenedores que permiten el etiquetado de un conjunto de elementos de control o solamente para escribir títulos o referencias dentro de la interfaz gráfica. Su estructura está compuesta por posición, tamaño y etiqueta. (Anexo B, Interfaz Grafica.py, línea 142)

wx.StaticBox(w.panel,label=' ',pos=(),size=())

```
###CheckBox
rb_torque = wx.CheckBox(w.panel,pos=(890,30), size=(170,50),label='Todos')
rb_torque.SetValue(1)
rb_base = wx.CheckBox(w.panel,pos=(890,80), size=(170,50), label='Base',)
rb_base.SetValue(1)
rb_Art1 = wx.CheckBox(w.panel,pos=(890,130), size=(170,50),label='Articulacion 1')
rb_Art1.SetValue(1)
rb_Art2 = wx.CheckBox(w.panel,pos=(890,180), size=(170,50),label='Articulacion 2')
rb_Art2.SetValue(1)
```

a)



b)

Figura 3. 19.- a) Código para declaración de Check box; b) Visualización del Check box

Elaborado por: (Mamarandi J., Velasco E.)

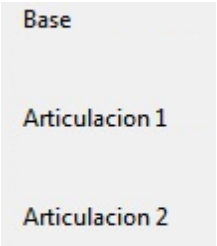
Los Check box solo tienen dos estados, activado o desactivado, estos elementos en la interfaz gráfica han sido de gran utilidad en la funcionalidad de torques para activarlos o desactivarlos en cada articulación.

Su estructura se compone de la posición, tamaño, etiqueta y el valor establecido siendo 1 para activado y 0 para desactivado. (Anexo B, Interfaz Grafica.py, línea 151)

wx.CheckBox(w.panel,label=' ',pos=(),size=( ))

```
###texto_estatico
wx.StaticText(w.panel, pos=(600,30), label='Base')
wx.StaticText(w.panel, pos=(600,80), label='Articulacion 1')
wx.StaticText(w.panel, pos=(600,130), label='Articulacion 2')
wx.StaticText(w.panel, pos=(600,180), label='Articulacion 3')
wx.StaticText(w.panel, pos=(600,230), label='Articulacion 4')
wx.StaticText(w.panel, pos=(600,280), label='Muñeca')
wx.StaticText(w.panel, pos=(600,330), label='Pinza')
```

a)



b)

Figura 3. 20.- a) Código para declaración de Static Text; b) Visualización del Static Text

Elaborado por: (Mamarandi J., Velasco E.)

El texto estático son etiquetas que se las puede posicionar independientemente de algún elemento de control, esto sirve para poner nombres, títulos, referencias dentro de la interfaz gráfica. (Anexo B, Interfaz Grafica.py, línea 175). Su estructura está compuesta de posición y etiqueta.

wx.StaticText(w.panel,label=' ',pos=())

```
###texto variables
tc_X = wx.TextCtrl(w.panel, pos=(585,597),size=(40,20))
tc_Y = wx.TextCtrl(w.panel, pos=(665,597),size=(40,20))
tc_Z = wx.TextCtrl(w.panel, pos=(745,597),size=(40,20))
tc_X1 = wx.TextCtrl(w.panel, pos=(585,647),size=(40,20),value='0')
tc_Y1 = wx.TextCtrl(w.panel, pos=(665,647),size=(40,20),value='0')
tc_Z1 = wx.TextCtrl(w.panel, pos=(745,647),size=(40,20),value='0')
tc_iteraciones=wx.TextCtrl(w.panel, pos=(600,800),size=(20,20))
```

a)

b)

Figura 3. 21.- a) Código para declaración de Text Control; b) Visualización del Text Control

Elaborado por: (Mamarandi J., Velasco E.)

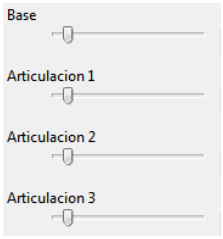
Los controles de texto son componentes de la interfaz gráfica que permiten la lectura y el ingreso de datos por teclado, en el proyecto se

puede observar que se encuentran en las funcionalidades para ingresar posiciones y coordenadas. (Anexo B, Interfaz Grafica.py, línea 241). Su estructura es general y muy parecida al de texto estático, utiliza posición y tamaño.

wx.TextCtrl(w.panel,label=' ',pos=(),size=( ),value=' ')

```
######sliders
s_base= wx.Slider(w.panel, pos=(720,45), size=(140,20),minValue=0,maxValue=300)
s_art1= wx.Slider(w.panel, pos=(720,95), size=(140,20),minValue=70,maxValue=300)
s_art2= wx.Slider(w.panel, pos=(720,145), size=(140,20),minValue=70,maxValue=300)
s_art3= wx.Slider(w.panel, pos=(720,195), size=(140,20),minValue=70,maxValue=300)
s_art4= wx.Slider(w.panel, pos=(720,245), size=(140,20),minValue=70,maxValue=300)
s_munieca= wx.Slider(w.panel, pos=(720,295), size=(140,20),minValue=0,maxValue=359.9)
s_pinza= wx.Slider(w.panel, pos=(720,345), size=(140,20),minValue=0,maxValue=100)
```

a)



b)

Figura 3. 22.- a) Código para declaración de sliders geométricos; b) Visualización de los sliders

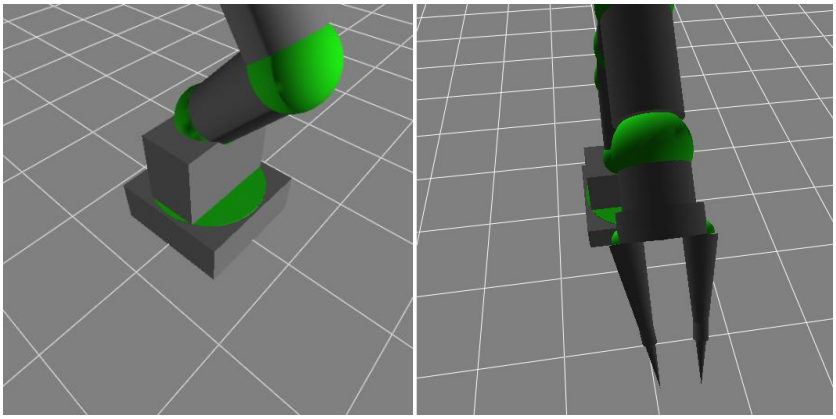
Elaborado por: (Mamarandi J., Velasco E.)

Los sliders son barras de desplazamiento que generan un valor numérico dependiendo de su posición. Estos son usados en la interfaz gráfica para permitir el movimiento de las articulaciones y modificar los valores de velocidad. (Anexo B, Interfaz Grafica.py, línea 315). Para la declaración de los sliders se necesita de la siguiente estructura:

wx.Slider(w.panel,pos=(),size=(),minvalue=##,maxValue=##)

```
pieza_1=box(pos=(0,7.85,0),length=10, height=7,  width=5 ,display=pantalla,color=(0.5,0.5,0.5))
pieza_2=cylinder(length=22.5,  radius=2.5 ,color=(0.5,0.5,0.5),display=pantalla)
pieza_pinza_2=cone(pos=(2,-4,0),radius=1.2, color=(0.5,0.5,0.5),display=pantalla)
union_2=sphere(pos=(0,22.35,0), radius=2.8,display=pantalla,color=(0.125,0.93,0.082))
```

a)



b)

Figura 3. 23.- a) Código para creación de sólidos geométricos; b) Visualización de los sólidos creados

Elaborado por: (Mamarandi J., Velasco E.)

Los sólidos creados para la ventana de simulación tienen sus propios parámetros dependiendo del tipo de cuerpo geométrico citado, por ejemplo si se crea un cono los parámetros a definir serían la altura y el radio de la base, si es una esfera definir su radio, si es un paralelepípedo o un cubo hay que definir su alto, ancho y espesor.

Para todos estos la posición es indispensable y como parámetros optativos se puede definir el color o textura.

En la Figura 3. 23 literal a) se puede observar que no se definen algunos parámetros, como su posición, en la creación de ciertos sólidos ya que al ser animados o van a estar en movimiento dichos parámetros se definirán dentro de un lazo donde se estarán calculando constantemente. (Anexo B, Interfaz Grafica.py, línea 286)

3.4 Funciones de la interfaz gráfica

```
##########FUNCIONES########

def velocidadesRPM (evt):

    velocidad('Base',(float(t_rpm_base.GetValue())))
    velocidad('Art_1',(float(t_rpm_art1.GetValue())))
    velocidad('Art_2',(float(t_rpm_art2.GetValue())))
    velocidad('Art_3',(float(t_rpm_art3.GetValue())))
    velocidad('Art_4',(float(t_rpm_art4.GetValue())))
    velocidad('Munieca',(float(t_rpm_munieca.GetValue())))
    velocidad('Pinza',(float(t_rpm_pinza.GetValue())))
```

Figura 3. 24.- Funciones que enlazan los eventos de los elementos de control
Elaborado por: (Mamarandi J., Velasco E.)

Las funciones desarrolladas en el código de la interfaz gráfica necesitan de un evento para su funcionamiento, la palabra clave usada es **evt** el cual es ejecutado al realizar una acción de algún componente de control como botones, sliders, check box, etc. en la Figura 3. 24, los encargados de realizar el evento son los spinners **t_rpm** de los cuales se toma el valor con el comando **.GetValue( )** y lo asigna a la función **velocidad** importada de la librería desarrollada **Dynamixel_Lib.**

De la misma manera en que se puede obtener datos de los diferentes elementos de control, se puede establecer datos en ellos a través del comando **.SetValue( ).**

```
b_Actualizar.Bind(wx.EVT_BUTTON,actualizar)
b_Home.Bind(wx.EVT_BUTTON, home_brazo)
s_base.Bind(wx.EVT_SCROLL, slider_base)
s_art1.Bind(wx.EVT_SCROLL, slider_art1)
rb_torque.Bind(wx.EVT_CHECKBOX , torque_todos)
rb_base.Bind(wx.EVT_CHECKBOX , torque_base)
```

Figura 3. 25.- Ejecución de los elementos de control
Elaborado por: (Mamarandi J., Velasco E.)

La Figura 3. 25 muestra la estructura utilizada en las instrucciones, las cuales son llamadas al dar un clic en ellas pero cada una necesita de su

propio evento característico de la librería **wx**, por ejemplo para la ejecución de un botón se necesita del evento **wx.EVT_BUTTON.**

A continuación se cita en la Tabla 3. 1 los eventos para cada elemento empleado en la interfaz gráfica.

Tabla 3. 1.- Eventos de ejecución de funciones

Elemento de control	Evento para la ejecución de una función
Botón	wx.EVT_BUTTON
Slider	wx.EVT_SCROLL
Spinner	wx.EVT_SPINCTRLDOUBLE
Check Box	wx.EVT_CHECKBOX

Elaborado por: (Mamarandi J., Velasco E.)

La estructura de cada llamada a la función por el clic de un elemento de control es:

Elemento_de_control.Bind(evento_del_elemento, funcion)

3.4.1 Lista de Funciones de la interfaz gráfica

Las funciones que componen al programa de interfaz gráfica para el manejo del brazo robótico de 6 grados de libertad están integradas con los botones, sliders, spinners, etc. en la parte de control ya que estos llamaran a estas funciones al momento de ser ejecutados.

a. home_brazo(evt):

Esta función nos permite llevar al brazo a la posición **HOME** gracias a la función **home(evt)** de la librería **Dynamixel_lib.py**, también activa los valores de torques en ON y configura las velocidades de las articulación al 5% de su velocidad máxima. (Anexo B, Interfaz Grafica.py, línea 356)

b. cargar_matriz(evt):

La función **cargar_matriz (evt):** está directamente relacionada con el botón **ABRIR** de la Interfaz gráfica, esta función lo que hace es abrir un archivo .txt previamente generado o guardado y lo convierte en pasos, es decir, en una matriz de posiciones en grados de cada articulación. (Anexo B, Interfaz Grafica.py, línea 378)

c. guardar_matriz(evt):

Esta función lo que permite es guardar un archivo .txt con los movimientos que se encuentran en la ventana de texto **tc2** en el recuadro de **Posiciones Guardadas,** este archivo generado puede después ser abierto con el botón **ABRIR** de la Interfaz Gráfica. (Anexo B, Interfaz Grafica.py, línea 415)

Este archivo puede ser modificado en el bloc de notas, ya que está configurado de una manera intuitiva.

d. actualizar(evt):

Esta función está relacionada con el botón **ACTUALIZAR,** lo que esta hace es iniciar la comunicación con el brazo robótico y leer todos los datos de los spinners de cada articulación para luego ingresarlos en la función **ejecutar()** de la librería **Dynamixel_lib.py**, después actualiza cada slider de las articulaciones dependiendo de su posición. (Anexo B, Interfaz Grafica.py, línea 422)

e. Comunicacion(evt):

La función comunicación está relacionada con el botón **COM** que se encuentra en el conjunto de botones de funciones, esta permite comunicar al brazo con la Interfaz Gráfica abriendo el puerto COM del dispositivo de conexión USB2dynamixel con la ayuda de la función **iniciar()** de la librería

Dynamixel_lib.py, además se lee la posición de cada articulación para actualizar los spinners y los sliders de la interfaz gráfica. (Anexo B, Interfaz Grafica.py, línea 434)

Esta función posee la variable global n que se la declara como 1 cuando se inicia la comunicación, esto permite saber si el dispositivo esta comunicado con la interfaz y saber reconocer futuros errores en la red de servomotores.

f. Paro(evt):

Esta función se ejecuta al hacer clic en el botón **PARO** la cual hace que se finalice la comunicación del brazo robótico con la Interfaz Gráfica, además asigna el valor de 0 a la variable global **n** indicando que se finalizó la comunicación. (Anexo B, Interfaz Grafica.py, línea 459)

g. Grabar(evt):

Mediante esta función se puede guardar un vector de posiciones que se seguirán mostrando en la ventana de texto en el recuadro de **Posiciones Guardadas** numeradas en el orden en el que se hayan sido guardadas, generando así una matriz de movimientos. (Anexo B, Interfaz Grafica.py, línea 465)

Para crear esta matriz se cogen los valores de los spinners de las articulaciones creando un vector que se añade de manera continua a la variable global **pasos**; después se ejecuta la función **imprimir_paletizado(evt)** que permite visualizar los movimientos guardado en la variable pasos.

h. Ejecutar_B(evt):

Permite ejecutar un conjunto de movimientos ya grabados en la matriz **pasos** un número n de veces según se establezca en el spinner

veces que se encuentra junto al botón **EJECUTAR** en la zona de **Posiciones Guardadas,** el cual es el encargado de llevar a cabo esta función. (Anexo B, Interfaz Grafica.py, línea 471)

i. Secuencia(evt):

Ejecuta solo el número del vector de movimientos de la matriz **pasos** según el dato numérico del spinner **ts_k,** de esta manera se puede ejecutar solo una secuencia y verificar las posiciones de las articulaciones. (Anexo B, Interfaz Grafica.py, línea 478)

j. Borrar_secuencia(evt):

Borra solo el número de movimiento de la variable **pasos** según el dato numérico del spinner **borrar_secuencia,** y recorre la matriz el espacio para no dejar ningún elemento vacío, esto se logra con la función **remove** propia de Python para el manejo de arreglos. (Anexo B, Interfaz Grafica.py, línea 483)

k. Borrar(evt):

Declara la variable global **pasos** como un vector vacío [ ], y borra la ventana de texto con el comando **tc2.SetValue('')**. (Anexo B, Interfaz Grafica.py, línea 489)

l. Reemplazar(evt):

Esta función fue creada para facilitar el cambio de una posición del brazo robótico en la matriz de datos **pasos**, de esta manera se puede modificar un solo vector de posiciones de cada articulación en uno nuevo. La función **reemplazar(evt)** está relacionada con el botón **REEMPLAZAR** del cuadro **Posiciones Guardadas** de la Interfaz Gráfica. (Anexo B, Interfaz Grafica.py, línea 494)

m. Funciones: anterior(evt) y siguiente(evt):

Ambas funciones permiten un manejo de los movimientos que se quieren ejecutar de la matriz **pasos** de forma secuencial. Así se puede avanzar o retroceder en la matriz y verificar si las posiciones del brazo son las adecuadas. (Anexo B, Interfaz Grafica.py, línea 498 y línea 505)

Para saber cuál es el paso que se está ejecutando se debe saber el valor del spinner **ts_k,** que está junto al botón **EJECUTAR**.

Esta función se relaciona al presionar los botones **>>>** que corresponde a **siguiente(evt)**, y el botón **<<<** que llama a la función **anterior(evt).**

n. insertar(evt):

Esta función logra la facilidad de ingresar un movimiento extra entre movimientos ya grabados sin modificar los ya existentes. Esta función es llamada con el botón **INSERTAR,** y es ingresado en la posición dependiendo del valor numérico del spinner **ts_Insertar** y la función **pasos.insert**, la cual permite ingresar un valor en la matriz **pasos**. (Anexo B, Interfaz Grafica.py, línea 513)

o. imprimir_paletizado(evt):

Lo que esta función realiza es convertir la matriz **pasos** en texto para luego ser visualizada en la ventana de texto **tc2,** de esta manera se puede conocer la cantidad de movimientos que se tienen guardados y los valores en grados de cada articulación. (Anexo B, Interfaz Grafica.py, línea 518)

p. cinematica(evt):

Para poder ubicar al artefacto del brazo robótico, en este caso la pinza, en un lugar en el espacio dado por cualquier sistema de coordenadas se debe conocer el ángulo de cada articulación y su dimensión, además de cómo se comporta el brazo robótico, para este caso es un brazo robótico articulado. (Anexo B, Interfaz Grafica.py, línea 524)

El proceso para calcular los ángulos de cada articulación en un brazo robótico articulado se conoce como cinemática inversa.

Esta función permite hacer este proceso, conociendo el punto en el espacio dado por coordenadas cartesianas donde se requiera posicionar el artefacto y entrega los grados de cada articulación.

A continuación se describe con detalles cómo esta función se llevó a cabo.

- Se calcula el módulo entre las coordenadas ingresadas X,Y; leyendo los valores con **tc_X1.GetValue()** y **tc_Y1.GetValue(),** y con la siguiente formula:
  x=sqrt(float(tc_X1.GetValue())2+float(tc_Y1.GetValue())**2)**
- Se obtiene el módulo de la coordenada en el espacio X,Y,Z con el comando:
  modulo=sqrt(float(tc_X1.GetValue())2+float(tc_Y1.GetValue())**2+float(tc_Z1.GetValue())**2)**
  Esto permite saber si el brazo puede llegar a ese lugar en el espacio.
- Se procede a preguntar si no existen inconvenientes para el cálculo de la cinemática inversa, los posibles errores que se pueden dar son: un módulo mayor a 60cm o menor a 10cm, ya que no trabaja en estas zonas; y se verifica si no existe ningún error en la red de

servomotores. Si existe uno de estos errores, la Interfaz Gráfica muestra una ventana emergente indicando cual es el inconveniente, esto se logra con la instrucción:

ctypes.windll.user32.MessageBoxW (0, “Mensaje”u"Titulo", 0)

- Para la determinación de la cinemática inversa se realiza un lazo while donde se calcula los posibles ángulos de cada articulación y este se rompe al encontrar un ángulo de cabeceo menor a 360º, y si los ángulos de las articulación están dentro de la zona del trabajo del bazo robótico. Dentro del lazo while se encuentra el cálculo de las componentes en X, Y de cada articulación, esto se logra con las funciones **cos()** y **sin()** que calculan el coseno y seno de un valor respectivamente.

  munieca_x=cos(ang_cabezeo*pi/180)*art4_cm

  munieca_y=sin(ang_cabezeo*pi/180)*art4_cm

  Al ya tener estos datos se puede calcular los ángulos de las articulaciones con teorema de cosenos, ya que se conoce los lados de los triángulos que forma cada articulación, esto se hace en la siguiente línea de código.

  beta=

  ((brazo2-ante_brazo**2+hipotenusa**2)/(2*brazo*hipotenusa))**

  gamma=

  ((brazo2+ante_brazo**2-hipotenusa**2)/(2*brazo*ante_brazo))**

- Finalmente la función calcula el ángulo de giro del brazo robótico, de esta manera puede ubicar la pinza en el lugar del espacio ingresado, esto se logra calculando la tangente de las componentes en X y Y.

  angulo_giro=

  atan(float(tc_Y1.GetValue())/float(tc_X1.GetValue()))*180/pi

  Al ya tener los ángulos calculados, se designa a cada articulación su ángulo correspondiente. En la siguiente instrucción se ve los valores, en color rojo, que se van a ejecutar para la ubicación del brazo robótico.

ejecutar([float(angulo_giro),float(ang_brazo),180,float(ang_ante_brazo),float(ang_munieca),float(t_munieca.GetValue()),float(t_pinza.GetValue())])

q. torque_todos(evt):

Esta función permite encender o apagar el torque de todas las articulaciones, esta es llamada al leer el valor del check box **rb_torque.GetValue()**.(Anexo B, Interfaz Grafica.py, línea 600)

r. Funciones: torque_base(evt), torque_art1(evt), torque_art2(evt), torque_art3(evt), torque_art4(evt), torque_munieca(evt), torque_pinza(evt):

Estas funciones permiten encender o apagar el torque de cada articulación dependiendo del check box que sea activado. (Anexo B, Interfaz Grafica.py, línea 615 - línea 665)

s. Funciones: Xmas(evt), Xmenos(evt), Ymas(evt), Ymenos(evt), Zmas(evt), Zmenos(evt):

Estas funciones están relacionadas con los botones, **X+ X- Y+ Y- Z+ Z-,** del static box **Cuadro de Coordenadas**. Estas permiten aumentar o disminuir el valor de cada coordenada dependiendo del valor del slider **s_iteraciones,** de esta manera se puede manejar el brazo en los planos X,Y,Z de manera intuitiva. (Anexo B, Interfaz Grafica.py, línea 832 – línea 849)

t. error():

Fue elaborada para conocer en la Interfaz Gráfica, mediante una ventana emergente, si existe un error en la red de servomotores; esta función adquiere el vector de errores entregado por la función **leer_errores(evt)**. (Anexo B, Interfaz Grafica.py, línea 932)

Los errores que pueden existir son: sobrecarga, límite de ángulo, fuera de rango, error de instrucción de código, error en límites de voltaje, de corriente y de Cheksum

Además esta función permite conocer el lugar del error cambiando de color verde a rojo la articulación en la Interfaz Gráfica. También se puede saber si existe un error en el recuadro **Características de Los Servomotores**.

3.4.2 Funciones para mecanizado

Para poder ver la funcionalidad de mecanizado en el brazo robótico se crearon 2 funciones llamadas **imagenes(evt)** e **imagenes_2(evt)**. Dichas funciones emulan el procedimiento que debería seguir un sistema robótico para realizar la funcionalidad de mecanizado que industrialmente se traduce como el posicionamiento del sistema en coordenadas preestablecidas por el operario para dar forma a una pieza por el desgaste del material, para resumir el funcionamiento de cada una es el siguiente:

a. imagenes(evt):

Esta función es llamada al presionar el botón **IMG PUNTOS,** que está en el recuadro **Funciones**. La función establece un valor de 180º a la muñeca para realizar el mecanizado. (Anexo B, Interfaz Grafica.py, línea 851)

A continuación se llama a la función **dibujo_pixeles()** de la librería **imagen_dynamixel.py**, esta devuelve un vector de coordenadas que representa a los pixeles de una imagen, ordenada de manera de los índices (i,j) de la imagen, como se puede ver en la imagen Figura 3. 9.

Después con una instrucción **for** se recorre todo el vector de coordenadas para entrar a la secuencia de movimientos del brazo y crear una imagen pixelada que la función **imagen_dynamixel.py** adquirió.

b. imagenes_2(evt):

Esta función es llamada al presionar el botón **IMG LINEAS,** que está en el recuadro **Funciones**. La función setea un valor de 180º a la muñeca para realizar el mecanizado. (Anexo B, Interfaz Grafica.py, línea 883)

Se adquiere el vector de coordenadas entregada por la función **dibujo_lineas()** de la librería **imagen_dynamixel.py,** este vector esta ordenado de forma de los pixeles de color negro más cercanos como se explica en la imagen Figura 3. 10.

Ya teniendo este vector de coordenadas en pares ordenados como ya se explicó, el brazo realiza líneas continuas en cada pixel, es decir une los pixeles de la imagen. Si la función no encuentra ningún pixel cercano en el vector se lee el valor de "**a**" que indica que el brazo debe levantarse para buscar otro pixel. Al ya recorrer todos los pixeles de la imagen el brazo regresa a su posición **HOME.**

CAPÍTULO 4

4 PRUEBAS, ANALISIS DE RESULTADOS, CONCLUSIONES Y RECOMENDACIONES.

4.1 Pruebas

4.1.1 Pruebas de mecanizado y Paletizado

Luego de elaborar las librerías en el software Python y poder implementarlas para crear una Interfaz Gráfica capaz de poder manipular los servomotores Dynamixel y su estructura, se lleva a cabo el desarrollo del paletizado y el mecanizado del brazo robótico de 6 grados de libertad, para esto se realizaron varias pruebas en donde se observó el desempeño del software implementado.

a. Pruebas de Paletizado

Para las pruebas de paletizado del brazo robótico hay que tener en cuenta la zona de trabajo del mismo, además de los objetos a paletizar, ya que estos no deben ser muy pesados debido a que los servos de las articulaciones del brazo robótico pueden sufrir errores de sobrecarga y apagarse.

En el paletizado se estableció una secuencia de pasos con las funciones y los botones del cuadro de **Posiciones Guardadas**, el cual se sabe que permite guardar posiciones en el espacio de trabajo del brazo robótico, además se puede guiar mediante la simulación grafica en 3D para saber dónde se encuentra el brazo robótico.

Tras grabar un número adecuado de posiciones del brazo robótico se crea una secuencia de paletizado en donde se va a mover 1 objeto, como muestra la Figura 4. 1, de un lugar A hacia un lugar B.

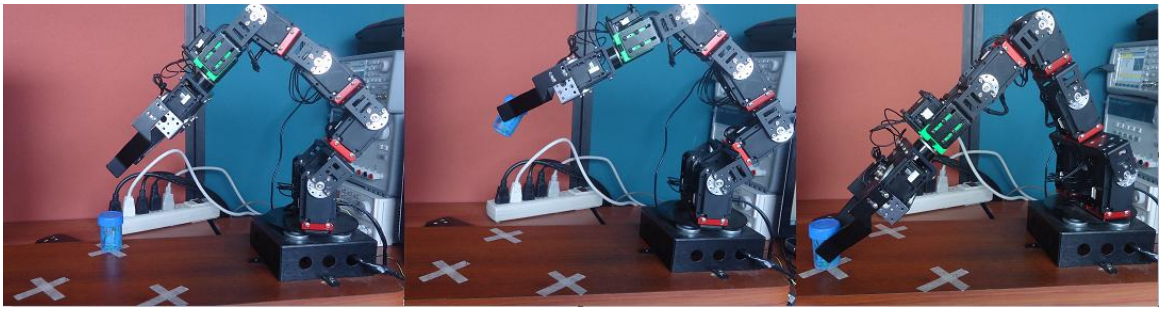


Figura 4. 1.- Proceso de paletizado de un objeto de un punto A hacia un punto B

Elaborado por: (Mamarandi J., Velasco E.)

Mientras se realizó el paletizado, se observó que las posiciones de cada articulación difería con las posiciones guardadas, esto se debe a que las articulaciones del brazo poseen cada una de estas una carga que hace que el brazo marque un error, por lo cual se tuvo que sintonizar cada servomotor del brazo robótico, variando las constantes P, I, D de cada articulación con los spinners mostrados en la Figura 4. 2, por lo que se realizaron diferentes pruebas que se detallan en el Anexo C (**¡Error! No se encuentra el origen de la referencia.**, **¡Error! No se encuentra el origen de la referencia.**, **¡Error! No se encuentra el origen de la referencia.**, **¡Error! No se encuentra el origen de la referencia.**).

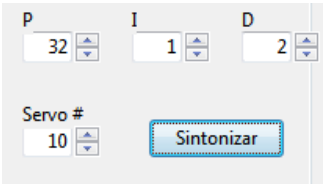


Figura 4. 2.- Spinners de las constantes P, I, D y número de servomotor para su sintonización

Elaborado por: (Mamarandi J., Velasco E.)

En estas pruebas se configuraron diferentes valores de PID de cada articulación, para de esta manera decrementar el error que posee cada una de estas, así se establecieron las diferentes ganancias en el sistema PID de los servomotores mostradas en la Tabla 4. 1.

Tabla 4. 1.- Constates de sintonización para cada articulación del brazo robótico

Articulación	P	I	D
Base	32	1	5
Articulación 1	32	2	2
Articulación 2	32	2	5
Articulación 3	32	2	1
Articulación 4	40	5	2
Muñeca	35	3	2
Pinza	35	5	2

Elaborado por: (Mamarandi J., Velasco E.)

Al tener sintonizado el brazo robótico, se logró mayor exactitud en el posicionamiento en el espacio, en la Figura 4. 3 muestra como el sintonizar el brazo mejora el paletizado.

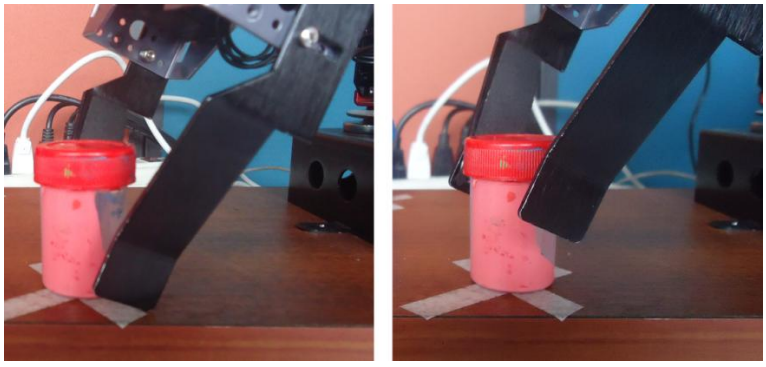


Figura 4. 3.- Imagen izquierda (brazo con control P), Imagen derecha (brazo con control PID)

Elaborado por: (Mamarandi J., Velasco E.)

Como se puede apreciar en la Figura 4. 3, al no tener sintonizado el brazo, pueden ocurrir colisiones del mismo, o también no poder paletizar los objetos de manera correcta.

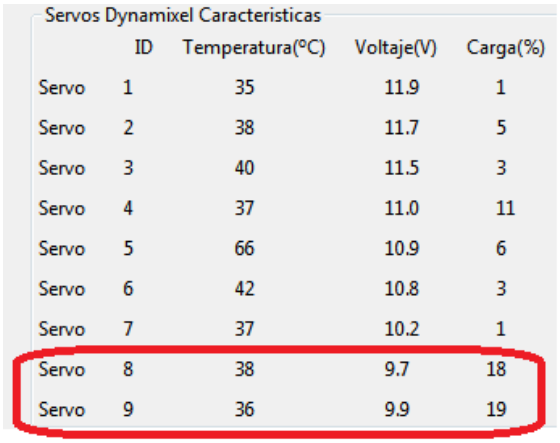
Con la secuencia ya guardada y con los servomotores ya sintonizados (recordar que la sintonización se debe hacer cada vez que los servomotores sean energizados) procedemos a realizar las pruebas finales con los objetos a paletizar.

Tras grabar 12 posiciones del brazo robótico se crea una secuencia de paletizado en donde se va a mover 1 objeto, previamente antes de ejecutar la secuencia completa, se procedió a realizarla paso a paso para poder verificar sus movimientos. Con la ayuda de los botones **<<<** de anterior y **>>>** de siguiente, se verifica si la secuencia posee algunos movimientos innecesarios lo caules son borrados con el botón **BORRAR** y con el spinner **borrar_secuencia** mostrados en la Figura 4. 4.

Figura 4. 4.- Botón **Borrar** y spinner **borrar_secuencia**, necesarios para eliminar posiciones innecesarias del paletizado.
Elaborado por: (Mamarandi J., Velasco E.)

De esta manera al borrar algunas secuencias, se redujo las posiciones del brazo robótico para su paletizado de 12 a 9 movimientos.

En los movimientos ya guardados se debe tener cuidado con los que están involucrados en el agarre del objeto, más aún cuando la pinza se cierra para la sujeción del objeto; ya que si existe mucha carga en la pinza, en los servomotores 8 y 9, puede generarse un error de sobre carga y apagarlos. Para lo cual es necesario que cuando se graben estos movimientos se manipule el spinner de la pinza para poder saber cuál es el valor exacto con la que sujeta con firmeza al objeto, y no generar ningún error, es sugerible que el valor de carga este entre el 15% y 25% como se muestra en la Figura 4. 5.



Servos Dynamixel Caracteristicas

	ID	Temperatura(°C)	Voltaje(V)	Carga(%)
Servo	1	35	11.9	1
Servo	2	38	11.7	5
Servo	3	40	11.5	3
Servo	4	37	11.0	11
Servo	5	66	10.9	6
Servo	6	42	10.8	3
Servo	7	37	10.2	1
Servo	8	38	9.7	18
Servo	9	36	9.9	19

Figura 4. 5.- Cargas de los servomotores del brazo robótico. Servomotores 8 y 9 con una carga de 18% y 19% de su máximo respectivamente.

Elaborado por: (Mamarandi J., Velasco E.)

En el paletizado de los 3 objetos también se tomó en cuenta cuales eran las velocidades de cada servomotor, las pruebas realizadas mostradas en el Anexo C (**¡Error! No se encuentra el origen de la referencia.**) muestran cuales son las velocidades adecuadas para que el brazo pueda trabajar sin nungun inconveniente. Asi se estableció que las velocidades se modifiquen a: la base en 28%, las articulacion 1 en 5,6%, la articulación 4 en 10,8%, la muñeca y pinza del brazo robótico se las dejo a su máxima velocidad del 100%; de esta manera el brazo pudo realizar con éxito en un tiempo óptimo el paletizado del objeto de un opunto A hacia un punto B. Hay que tomar en cuenta en este caso que las articulaciones 2 y 3 no están directamente relacionadas en los movimientos de paletizado, por lo cual su velocidad no altera en los resultados del brazo robotico.

Tras configurar las velocidades de paletizado y tener sintonizado el brazo robotico, se realizaron 100 pruebas en donde se midio la desviación en milímetros del objeto ya situado en el punto B, estos datos se muestran en el Anexo C (**¡Error! No se encuentra el origen de la referencia.**).

Como se observa en el anexo mencionado, la desviación del objeto paletizado es mínima y no altera en la funcionalidad del brazo. En la Figura

4. 6, el valor promedio de desviación de un sitema de paletizado es de 1mm.

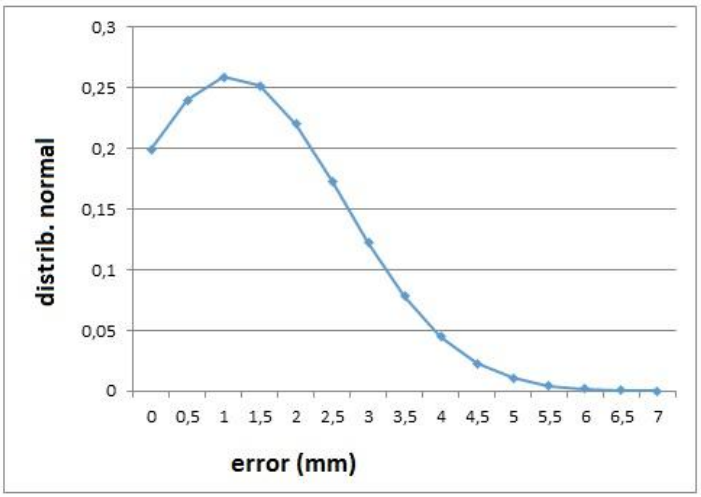


Figura 4. 6.- Curva de distribución normal del margen de error en las pruebas de paletizado

Elaborado por: (Mamarandi J., Velasco E.)

Ya demostrado que el brazo robótico puede realizar funcionalidades de paletizado, se realizó una prueba del movimiento de 3 objetos, como se muestra en la Figura 4. 7.

Figura 4. 7.- Proceso de Paletizado de 3 objetos.

Elaborado por: (Mamarandi J., Velasco E.)

b. Pruebas de Mecanizado

Para saber si el brazo robótico es capáz de realizar un mecanizado se ha ideado que realice una figura con un marcador sostenido por la pinza sobre una superficie cualquiera, esto da a la idea de que el brazo puede utilizar otras herramientas en su pinza con el fin de recrear el proceso realizado por una fresadora o caladora y así verificar si es apto para la funcionalidad de mecanizado.

Para que el brazo pueda trazar esta imagen, se trabajará con la librería **imagen_dynamixel** y la función **cinematica**, ya detalladas en el apartado 3.2.1.

Así una imagen en mapa de bits, es convertida en coordenadas y posteriormente el brazo, con estas coordenadas, es posicionado en el espacio generando así la imagen del ordenador a una superficie.

Se realizaron dos pruebas de mecanizado, una con la función **dibujo_pixeles** y otra con **dibujo_lineas**, ambas funciones de la librería **imagen_dynamixel**.

Con la primera función, **dibujo_pixeles**, se generó una imagen en forma de pixeles, para esto se deben establecer los valores de las dimensiones del dibujo y la altura del marcador en conjunto con la pinza que se las llamara artefacto, debido a que en el mecanizado el artefacto es el cual genera las piezas mecanizadas.

Los spinners del cuadro **Variables de Imagen**, de la Figura 4. 8, son usados para la configuración de los parámetros de dimensiones del dibujo y la altura del artefacto.

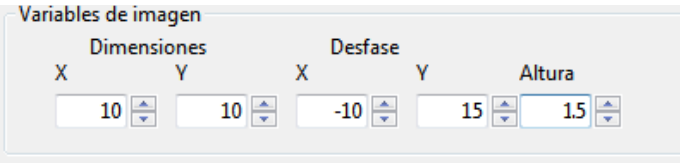


Figura 4. 8.- Spinners para calibrar las dimensiones, el desfase en X, Y y la altura del artefacto.

Elaborado por: (Mamarandi J., Velasco E.)

En este caso se estableció que el dibujo que se va a generar tenga las dimensiones de 10cm x 10cm con un desfase en el plano de -10cm en X y con 15cm en el eje Y, para la precisión del artefacto en la superficie a dibujar se estableció una altura de 1,5cm ya que este es el tamaño del

marcador que sobresale de la pinza al ya estar sujeta en el brazo robótico, esto se muestra en la Figura 4. 9.

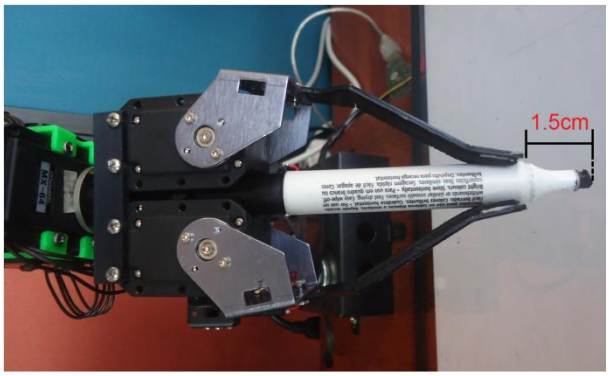


Figura 4. 9.- Pinza sujetando el marcador con una altura sobresaliente de 1,5cm.

Elaborado por: (Mamarandi J., Velasco E.)

Se debe tomar en cuenta, al igual que en el paletizado, que la carga de la pinza este en los valores de 15% y 25% al sostener el marcador; así no existirá errores de sobrecarga.

Después de ya tener calibrado las dimensiones a mecanizar la imagen, se procede a establecer una velocidad de trabajo no mayor al 10% en todas las articulaciones, ya que para velocidades altas se puede tener imperfecciones en el dibujo debido al movimiento brusco en las articulaciones, además de que en esta funcionalidad se crea la imagen por pixeles haciendo un punto con el artefacto y es recomendable que la velocidad sea baja ya que existiría vibraciones en la estructura por su movimiento de martilleo. Al igual que en el paletizado, para una mejor precisión, se debe establecer los valores PID de los servomotores según los valores de la Tabla 4. 1.

Ya teniendo calibradas las dimensiones y velocidades, también sintonizado el brazo robótico con los PID adecuados, se procede a presionar el botón **Img Puntos**, el cual abrirá una ventana para cargar el archivo imagen que se quiere mecanizar. La imagen puede ser de formatos

JPG, BMP o PNG. Después aparecerá el dibujo que se va a realizar junto un mensaje emergente que preguntará si se quiere realizar el mecanizado.

Al finalizar esa prueba de mecanizado se estableció que los puntos del dibujo sean parejos, así se sabe que todos los puntos estuvieron sujetos a la misma precisión en coordenadas y presión dada por el artefacto, si los pixeles del dibujo no son los adecuados, se debe calibrar la altura del artefacto. En la Figura 4. 10 se muestra como se realizó un mecanizado de un dibujo y como los pixeles poseen las mismas características.

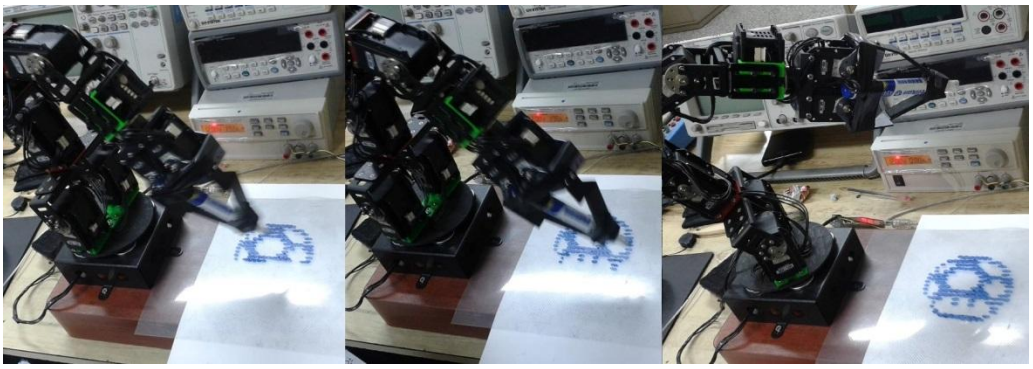


Figura 4. 10.- Proceso de mecanizado de un dibujo por pixeles
Elaborado por: (Mamarandi J., Velasco E.)

Con la segunda función, **dibujo_lineas**, se realizó otro tipo de mecanizado, en donde no se dibujan las imágenes pixel a pixel, sino que une estos pixeles en trayectorias, de esta manera se tiene una mejor apreciación del dibujo mecanizado, esto se habla en el desarrollo de la función **dibujo_lineas** explicada en el apartado 3.2.1.

Al igual que en el mecanizado anterior con la función **dibujo_pixeles**, se estableció los valores de altura de artefacto, dimensiones del dibujo y desfase en el plano, también los valores de velocidad de las articulaciones y carga en la pinza. Para esta prueba se tomaron los mismos valores ya establecidos anteriormente como se muestra en la Figura 4. 8.

A continuación al presionar el botón **Img Líneas** se puede cargar la imagen que se desea mecanizar. El mecanizado realizado con esta función determinó que se pueden mecanizar imágenes de mayor número de pixeles como se muestra en la Figura 4. 11.

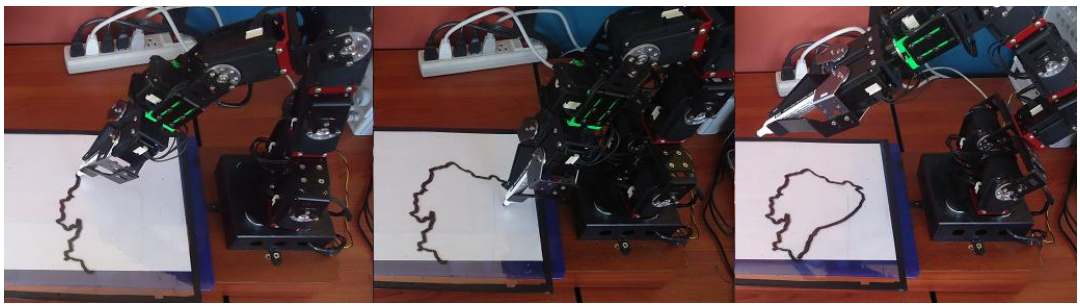


Figura 4. 11.- Proceso de mecanizado de una imagen por líneas
Elaborado por: (Mamarandi J., Velasco E.)

Como se observa en la Figura 4. 11, en este tipo de mecanizado se genera una imagen con mejor detalle. Se debe tener en cuenta que este proceso es algo lento, debido a que la función **dibujo_lineas** convierte cada pixel en una coordenada, entonces el brazo robótico debe ubicarse en cada uno de estas coordenadas. De esta manera se pueden mecanizar imágenes más complejas con una mayor cantidad de pixeles.

Para determinar si el mecanizado del sistema es el ideal, se realizaron tres tipos de pruebas, mostrados en el Anexo C (**¡Error! No se encuentra el origen de la referencia.**), donde se variaron los valores del los spinners de la Figura 4. 8. Asi se llegó a la conclusión que en el mecanizado se tiene un error de 1 mm para el eje Y y un error de 1,3 mm en el eje X. Tambien se realizaron pruebas en la que se vario las dimensiones del dibujo a mecanizar Anexo C (**¡Error! No se encuentra el origen de la referencia.**), aquí se observó que entre mas grande es el dibujo, tiene una mayor desproporcion en el eje X, dando un valor de error promedio de 3mm.

4.2 Análisis de Resultados

Con la finalidad de lograr los objetivos planteados en la tesis, se desempeñaron varias pruebas tanto en la parte del software como en el hardware. Se ocuparon diversos criterios para aprobar el software realizado y conocer su desempeño. Entre los criterios de aceptación del software para la interfaz gráfica se tienen los siguientes literales.

4.2.1 Análisis de la Interfaz Gráfica

a. Facilidad de manejo.

El software posee múltiples elementos los cuales producen el control de los servomotores, como son el caso de sliders y spinners que manipulan el movimiento de los servomotores. Al comparar con el software RoboPlus, este posee similares manejos de los servomotores.

A diferencia de RoboPlus, el software desarrollado genera un mejor entendimiento en velocidades, posiciones en grados, voltajes de consumo, temperatura, etc.; debido a que estos datos, están parametrizados, con unidades de grados (radianes), porcentaje de velocidad y carga, °C, etc. Y así facilita que el usuario sepa las capacidades de trabajo de cada servomotor.

Además el poder saber simultáneamente el estado de cada servomotor facilita el desempeño del brazo robótico y esto ayudó a la creación de un juego de objetos en 3D que simulan el trabajo del brazo robótico en el espacio.

b. Trabajo en tiempo real.

La medición simultanea de los datos de cada servomotor leído en cada ciclo del programa principal en la interfaz gráfica, permite conocer el

estado de cada uno de estos y así saber cómo están en tiempo real, siempre y cuando el brazo robótico este en estado estático.

Uno de los inconvenientes que se generó en la interfaz gráfica es el no poder monitorear el brazo robótico mientras este desempeña un lazo de movimientos, esto se debe a que trabaja de forma paralela el monitoreo y el desplazamiento del brazo. La interfaz gráfica genera una ayuda de manipulación del brazo robótico, mas no un monitoreo en tiempo real de trabajo del mismo.

Para la solución de este inconveniente se debe trabajar en Python con multi-hilos, o multi-tramas, que permiten la ejecución de programas en paralelo, es decir que se pueden ejecutar funciones del programa y no esperar que se termine dicha función para generar otra, sino que se puede ejecutar otra en cualquier momento.

c. Interfaz Usuario Máquina interactiva e intuitiva.

El establecer un grupo de sólidos que simulen el brazo robótico ensamblado, mejora el manejo del mismo, debido a que se sabe en qué lugar en el espacio se encuentra el brazo y si este puede generar colisiones con el plano XY. Gracias a la cinemática directa, se puede conocer en qué lugar del espacio se encuentra el artefacto del brazo robótico y nos enseña una ubicación del mismo.

Además los diferentes textos de la interfaz ayudan al usuario en saber que ocurre con cada articulación y en que ubicación en grados se encuentra. También un sistema de detección de errores en la red, con la ayuda de varias funciones, permiten al usuario saber donde existen inconvenientes en los servomotores, de una manera intuitiva cambiando las articulaciones del brazo robótico simulado a una tonalidad roja y generando mensajes emergentes del tipo de error y en donde se generó.

d. Software con arquitectura abierta.

La mayoría de software de control para un sistema robótico es de carácter propietario, esto genera inconvenientes al momento de realizar operaciones fuera del límite permitido del software, generando así problemas en el usuario. Además si se requiere actualizaciones del software mencionado, se necesita el servicio de la empresa que lo desarrolló y esto añade un gasto más al usuario generando posibles inconformidades. La facilidad de Python es que es un software libre como ya se aclaró, esto permite que el programa sea de arquitectura abierta, admitiendo así cualquier cambio en las líneas de código y poder desarrollar futuros proyectos. De esta manera el usuario puede moldear el programa a su manera y generar las actualizaciones que vea necesarias. También al ser un software multiplataforma el usuario puede trabajar en Windows o Linux dependiendo de sus necesidades.

Por estas razones la interfaz gráfica creada en Python tuvo un buen desempeño al emplearlo como un sistema de control. Este software tiene la capacidad de generar un entorno manejable donde el usuario pueda manipular el hardware, en este caso el brazo robótico, mediante objetos comunes como spinners, check box, botones, etc.

4.2.2 Análisis de las Funcionalidades de Paletizado y Mecanizado

Para las funcionalidades del brazo robótico de paletizado y mecanizado se determinaron su desempeño según las pruebas realizadas en el apartado 4.1.1.

a. Paletizado

En las pruebas de paletizado se observó que el brazo robótico trabaja de manera adecuada cuando este está sintonizado con valores de PID establecidos en la Figura 4. 2, de esta manera el paletizado de los

objetos que se realizaron en las pruebas fueron exitosos inclusive después de ejecutar 100 pruebas de paletizado igual.

En el paletizado de los objetos pueden existir inconvenientes con las cargas o peso que soporta el brazo, esto se debe tener muy presente porque pueden apagar los servomotores e inhabilitar el brazo robótico, gracias a la visualización del porcentaje de carga de los servomotores se puede tener una idea del trabajo que ejerce el brazo robótico mientras este está paletizando. Por esto se debe realizar una prueba paso a paso del proceso que se quiere ejecutar y observar en toda la secuencia estos porcentajes de carga, principalmente las cargas en los servomotores 8 y 9 que son los que se encuentran en la pinza y los encargados de sujetar los objetos a paletizar.

b. Mecanizado

Para conocer si el brazo robótico es capaz de realizar un proceso de mecanizado se realizaron dos tipos de pruebas explicadas en el apartado 4.1.1.

Estas pruebas de mecanizado dieron a conocer que el brazo puede ubicarse en el espacio de trabajo con cualquier coordenada que ingrese el usuario, como ejemplo se creó el mecanizado de imágenes como se muestran en la Figura 4. 10 y la Figura 4. 11.

En las pruebas de mecanizado con la funcionalidad **dibujo_pixeles**, que genera una imagen en la superficie a través de puntos de un marcador como pixeles, se observó que las imágenes deben tener una dimensión no mayor a 20x20 pixeles, esto debido a que se realiza el mecanizado de todos estos pixeles, y por generase este proceso de manera de martilleo; un gran número de pixeles esto puede hace tardar mucho el proceso.

Al igual que en el paletizado, es primordial conocer la carga de sujeción del objeto, en este caso un marcador, si esta se encuentra fuera del rango de sujeción generaría que el artefacto se moviese o que los servomotores se apaguen por el exceso de carga.

En las pruebas de mecanizado con la funcionalidad **dibujo_lineas**, se tuvo como resultado que las imágenes pueden tener un mayor número de pixeles, las pruebas que se realizaron fueron con imágenes de hasta 700x700 pixeles, esto mejora la funcionalidad anterior de solo por pixeles, ya que esta nueva genera trazos entre pixeles para así componer la imagen. Además la funcionalidad de mecanizar una imagen partiendo desde un pixel y seguir a su más cercano en sentido horario, como se explica en la Figura 3. 10, ayuda a que el mecanizado se realice de manera más rápida de lo que se haría con la función **dibujo_pixeles**.

4.3 Conclusiones.

- El brazo robótico ensamblado responde correctamente al algoritmo de control en las funcionalidades de mecanizado y paletizado.
- El software libre ofrece posibilidades de adaptar los programas a las necesidades concretas del proyecto.
- Al tratarse de un sistema de control desarrollado en software libre permite que cualquier persona con conocimientos de programación pueda realizar cambios en las librerías para mejorarlas, estudiarlas o basarse en ellas para futuros proyectos.
- La implementación del sistema de control basado en software libre reduce costos de manera significativa cuando se trata de la inversión de sistemas robóticos con software propietario.
- De acuerdo al tipo de servomotores Dynamixel utilizados se tiene una configuración diferente de comunicación de la red de servo actuadores con la PC mediante la interfaz USB2Dynamixel.

- El protocolo de comunicación RS-485 permite comunicar los nueve dispositivos (servomotores) que conforman el brazo robótico en una red asegurando los valores de voltaje necesarios para que funcione normalmente el sistema robótico cuando estos trabajan al máximo.
- El software utilizado Python posee grandes ventajas y múltiples extensiones que permitió el desarrollo de la interfaz gráfica en este proyecto, pero también facilita su aplicabilidad en diversos campos de la ingeniería.
- Python permite el manejo de los servomotores Dynamixel, mediante librerías creadas por medio de una comunicación por el puerto USB y el conversor USB2Dynamixel.
- El software de diseño y simulación de elementos mecánicos permite usar técnicas de ingeniería inversa.
- Los servomotores con comunicación serial presentan grandes ventajas al poseer monitoreo y control interno.
- Con la programación realizada tanto en funciones como en interfaz gráfica, se requirió del estudio de otros campos como la cinemática inversa, matemática de vectores, linealización de rangos, procesamiento de imágenes, comprensión de la mecánica en robots, etc. Y así poder relacionar todo en un solo sistema.
- Se realizaron funciones de movimiento, ajuste de torque, velocidad, lectura y escritura de datos, entre otras; todas estas dependiendo de los parámetros o datos que se podían leer del servomotor.
- Para observar el comportamiento del robot antes de manipularlo se requirió de una función de simulación, gracias a esto se pueden evitar varios accidentes, ya que si un usuario ingresa valores de movimiento a una gran velocidad al ejecutar los comandos se corre un riesgo de golpes a las personas cercanas al brazo.
- Se trabajó mucho en lo que es la parametrización de coordenadas, ya que para manipular el brazo robótico al principio solo se ingresaban valores para cada servomotor, manipulando así sus

grados de libertad de uno en uno hasta llegar a la posición en el espacio deseado; pero no es lo más óptimo, por ende para movilizar al brazo hasta una posición requerida se tiene que trabajar en conjunto como un sistema en sincronización y utilizando el razonamiento matemático de la cinematica inversa, lo cual logró que el brazo llegue a una posición únicamente ingresando los valores en los ejes cartesianos X, Y y Z.

- Para la función del mecanizado se tuvo que realizar el estudio e investigación de la cinemática inversa, al comprender este tema la idea fundamental era leer la imagen que se trataba de dibujar y transformarla a una matriz de coordenadas para que el brazo robótico recorra dichos puntos y trace la figura cargada al programa. Para esto lo que se realizó es procesamiento de imágenes en Python y una relación de escalado de la matriz obtenida de la imagen en unos y ceros a una matriz de coordenadas por donde el brazo robótico recorrería en el espacio.
- Las pruebas realizadas han arrojado resultados favorables y satisfactorios como se esperaban, cada una de las funciones programadas cumplen con sus funciones a cabalidad, una de las funciones principales es la que emula la Programación por recorrido de un robot, esta es una técnica en la cual el usuario, que hace la programación, tiene contacto físico con el brazo robótico obteniendo el control y llevando el brazo del robot a través de las posiciones deseadas dentro del área de trabajo, esto es muy usado en la industria para que un robot haga una rutina laboral.

4.4 Recomendaciones.

- Cada servomotor que se usa en la estructura del brazo robótico poseen en su memora RAM los datos de ganancias para un control PID, para poder sintonizarlos y así tener un menor error de ángulo

en cada uno de ellos. Por defecto cada servomotor viene con los valores de ganancia de 32 para el Proporcional, 0 para el Integral y 0 para el Derivativo; estos datos hacen que los servos posean un error de 2 bits en su posición, es decir de 0,017°. Al ocurrir esto se da un error en la posición del brazo robótico, dependiendo de la articulación se pueden tener desviaciones en el rango de los centímetros. Por lo cual se debe sintonizar los servomotores con los valores de ganancia adecuados de acuerdo a la aplicación requerida. También hay que tomar en cuenta que como estos datos están en la memoria volátil de los servomotores, no quedarán grabados y se reiniciarán a los valores iniciales (P=32, I=0, D=0) por lo que se debe sintonizar al brazo robótico cada vez que se encienda el sistema.

- La estructura del brazo robótico no admite que este gire indefinidamente en su propio eje, debido a sus cables de comunicación. Por esto se debe establecer una zona de trabajo no mayor a los 359°, además se debe tomar en cuenta que el plano Z no se admiten valores negativos porque la base del brazo robótico se encuentra en un plano horizontal.
- La interfaz gráfica que se desarrolló posee un sistema de detecciones de errores, uno de los más comunes que ocurre mientras se está controlando el brazo robótico es el error de sobre carga, esto se debe a que los servomotores están sometidos a una gran fuerza, la cual el servomotor no puede soportar. En el monitoreo se puede ver el porcentaje de carga de cada servomotor, lo recomendable es que este porcentaje no exceda el 30%, debido a que puede activar la alarma de Overload, que posee cada servomotor, haciendo que este se apague.
- El programa de control desarrollado en el lenguaje de programación Python, no puede ejecutar varias tareas a la vez, por lo cual no puede monitorear una secuencia del brazo robótico en su interfaz

gráfica. Para solucionarlo se debe implementar un sistema multi-hilo o multi-tarea en el programa, esto permitirá ejecutar una secuencia de instrucciones mientras se desarrollan otras. Lamentablemente el realizar un programa multi-hilos conlleva a generar otra clase de código en la programación, por lo cual no se optó en realizar este avance ya que implicaría el desarrollo de un nuevo código en el software, es recomendado, que para futuros controles, se empleen programas multi-tareas o multi-hilos.

REFERENCIAS BIBLIOGRÁFICAS

[1] OSHA, «INDUSTRIAL ROBOTS AND ROBOT SYSTEM SAFETY,» 1 Enero 1999. [En línea]. Available: https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html. [Último acceso: 5 Mayo 2014].

[2] F. A. C. Herías, «Servomotores,» 20 Septiembre 2007. [En línea]. Available: http://www.aurova.ua.es:8080/proyectos/dpi2005/docs/publicaciones/pub09-ServoMotores/servos.pdf. [Último acceso: 8 Mayo 2014].

[3] N. Instruments, «Comunicación Serial: Conceptos Generales,» 6 Junio 2006. [En línea]. Available: http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1. [Último acceso: 29 Agosto 2014].

[4] M. A. Alvarez, «Desarrollo Web-Python,» 19 Noviembre 2003. [En línea]. Available: http://www.desarrolloweb.com/articulos/1325.php. [Último acceso: 2 Septiembre 2014].

[5] P. Org., «The Python Tutorial,» 5 Septiembre 2014. [En línea]. Available: https://docs.python.org/2/tutorial/index.html#tutorial-index. [Último acceso: 12 Septiembre 2014].

[6] C. Tecnologías, «Principales características del lenguaje Python,» 1 Septiembre 2011. [En línea]. Available: http://www.cuatrorios.org/index.php?option=com_content&view=article&id=161:principales-caracteristicas-del-lenguaje-python&catid=39:blogsfeeds. [Último acceso: 20 Agosto 2014].

[7] T. Robotics, «Trossen Robotics Dynamixel Guide,» 2006. [En línea]. Available: http://www.trossenrobotics.com/p/MX-64R-dynamixel-robot-actuator.aspx. [Último acceso: 20 Junio 2014].

[8] R. V. M. Augusta y J. H. F. Valencia, «Construcción de prototipo de brazo robótico controlado por un ordenador,» Escuela Politécnica Nacional, Quito, 2012.

[9] E. L. Pérez, «Temario RS-232,» [En línea]. Available: cselectrobomba.googlecode.com/files/Serial_RS232.pdf. [Último acceso: 29 Agosto 2014].

[10] O. Ocampo, «Programacion en Castellano-Guiía de aprendizaje de Python,» 2014. [En línea]. Available: http://programacion.net/articulo/guia_de_aprendizaje_de_python_65. [Último acceso: 20 Agosto 2014].

[11] C. John, ROBÓTICA, Tercera Edición ed., México: Pearson Educación, 2006, pp. 109 - 117.

[12] K. S. Fu, R. C. Gonzáles y C. S. G. Lee, ROBÓTICA: CONTROL, DETECCIÓN, VISIÓN E INTELIGENCIA, Primera Edición ed., México: Grupo Impresa S.A. de C. V., 1993, pp. 112 - 119.