# ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

## DESARROLLO DE UN SISTEMA DE CONTROL PARA UN BRAZO ROBOT QUE CLASIFIQUE OBJETOS EN BASE AL COLOR UTILIZANDO VISIÓN ARTIFICIAL

**LUIS ALFREDO BEJARANO BEJARANO**

Trabajo de Titulación modalidad: Proyectos de Investigación y Desarrollo presentado ante el Instituto de Posgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de:

MAGISTER EN SISTEMAS DE CONTROL Y AUTOMATIZACIÓN INDUSTRIAL

Riobamba - Ecuador

Noviembre 2021

**CERTIFICACIÓN:**

EL TRIBUNAL DEL TRABAJO DE TITULACIÓN CERTIFICA QUE:

El **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, titulado: DESARROLLO DE UN SISTEMA DE CONTROL PARA UN BRAZO ROBOT QUE CLASIFIQUE OBJETOS EN BASE AL COLOR UTILIZANDO VISIÓN ARTIFICIAL, de responsabilidad del señor Luis Alfredo Bejarano Bejarano, ha sido prolijamente revisado y se autoriza su presentación.

Tribunal:

Ing. Oswaldo Geovanny Martínez; M.S.c. .....

Firmado electrónicamente por: OSWALDO GEOVANNY MARTINEZ GUASHIMA

**PRESIDENTE**

Ing. John German Vera; Mtr. .....

JOHN GERMAN VERA LUZURIAGA

Digitally signed by JOHN GERMAN VERA LUZURIAGA
DN: CN=JOHN GERMAN VERA LUZURIAGA, SERIALNUMBER=010921185733, OU=ENTIDAD DE CERTIFICACION DE INFORMACION, O=SECURITY DATA S.A. 2, C=EC
Reason: I am the author of this document
Location: your signing location here
Date: 2021.11.18 17:07:09-05'00'
Foxit PhantomPDF Version: 10.1.3

**DIRECTOR**

Ing. José Alcides Rumipamba; Mtr. .....

Firmado electrónicamente por: JOSE ALCIDES RUMIPAMBA LOPEZ

**MIEMBRO DEL TRIBUNAL**

Ing. Christiam Xavier Núñez; Mag. .....

CHRISTIAM XAVIER NUÑEZ ZAVALA

Firmado digitalmente por CHRISTIAM XAVIER NUÑEZ ZAVALA
Nombre de reconocimiento (DN): c=EC, l=RIOBAMBA, serialNumber=0603964982, cn=CHRISTIAM XAVIER NUÑEZ ZAVALA

**MIEMBRO DEL TRIBUNAL**

Riobamba, noviembre 2021

## DERECHOS INTELECTUALES

Yo, Luis Alfredo Bejarano Bejarano soy responsable de las ideas, doctrinas y resultados expuestos en el **Trabajo de Titulación modalidad Proyectos de Investigación y desarrollo**, y el patrimonio intelectual generado por la misma pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

Firmado electrónicamente por:
LUIS ALFREDO
BEJARANO
BEJARANO

---

**LUIS ALFREDO BEJARANO BEJARANO**

C.I. 2100126917

## DECLARACIÓN DE AUTENTICIDAD

Yo, Luis Alfredo Bejarano Bejarano, declaro que el presente **Trabajo de Titulación Modalidad Proyecto de Investigación y Desarrollo**, es de mi autoría y que los resultados del mismo son auténticos y originales. Los textos constantes en el documento que provienen de otra fuente están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este proyecto de investigación de maestría.

Firmado electrónicamente por:
LUIS ALFREDO
BEJARANO
BEJARANO

---

**LUIS ALFREDO BEJARANO BEJARANO**

C.I. 2100126917

## DEDICATORIA

Dedico este logro a mis padres José Andrés (+) y Margarita (+) por todo el esfuerzo y sacrificio que realizaron día a día en mi enseñanza con principios, valores y sobre todo con la fe en Dios.
A mis hermanos por estar siempre a mi lado y apoyarme de manera incondicional.

***Luis***

## AGRADECIMIENTO

Ante todo, agradezco a Dios por brindarme salud y vida, para lograr mis metas propuestas.
Agradezco a la Escuela Superior Politécnica de Chimborazo, en especial al Instituto de Posgrados y Educación Continua, al personal docente quienes fueron parte de la formación profesional y humana.

En especial agradezco al Ing. John Vera Mtr. Director de tesis, por guiarme en la elaboración del proyecto, gracias a su apoyo brindado para sacar este proyecto adelante.

A mis familiares y amigos quienes brindaron su apoyo incondicional.

***Luis***

**TABLA DE CONTENIDO**

**ÍNDICE DE TABLAS**

**ÍNDICE DE FIGURAS**

**ÍNDICE DE GRÁFICOS**

**RESUMEN**

El presente documento explica el desarrollo de un sistema de control para un brazo robot que clasifique objetos en base al color utilizando visión artificial. Este documento presenta una solución al problema de manipular objetos mediante un brazo robot manipulador "xArm" de la empresa LewanSoul, el cual es controlado y guiado en base al reconocimiento de color de un objeto utilizando las librerías de visión artificial de MATLAB. Para solucionar este problema se utilizó la metodología de investigación de ensayo y error mediante el método de la espiral que se basa en resolver un problema en cuatro fases iniciando por la planificación, seguido del análisis de riesgo, la implementación y finalizar con la evaluación para observar los resultados alcanzados y empezar de nuevo con la espiral si los resultados no son los esperados. En la primera fase se *planificaron y delimitaron los objetivos*, los cuales fueron reconocer el color del objeto a manipular mediante técnicas de visión artificial con la ayuda de MATLAB y una cámara Logitech C930. Como segunda fase se *analizó* la matemática del robot para guiarlo hacia el punto que define la ubicación del objeto a recoger, de la misma manera se *analizaron* las estrategias para la creación de la interfaz gráfica y la comunicación entre software y hardware. En éste análisis se evaluaron los tipos de controles que se necesitaba para comandar el brazo, y se utilizó una GUI de MATLAB para crear la interfaz gráfica, además se escogió la comunicación serial para enviar datos hacia la tarjeta controladora de servos LewanSoul. En la tercera fase se *implementó el sistema* conectando el brazo LewanSoul con la PC de MATLAB y la cámara web Logitech C930 mediante comunicación serial. Por último, se *evaluaron* los resultados obtenidos y se encontró que el sistema presenta un 97% de fidelidad en el control de sus articulaciones desde la interfaz gráfica, un 100% en el reconocimiento de colores (rojo y verde) y un 97% en cuanto a la clasificación de objetos, con errores mínimos debido al procesador de la PC que fueron solventados al ejecutar nuevamente las acciones desde la primera fase del método de la espiral. Al final, se termina el trabajo concluyendo que el robot es fiable para utilizarlo en la industria dentro de procesos repetitivos de clasificación.

**Palabras clave**: <BRAZO ROBOT>, < VISIÓN ARTIFICIAL>, <RECONOCIMIENTO DE COLOR>, <CLASIFICACIÓN DE OBJETOS>.

**SUMMARY**

This document explains the development of a control system for a robot arm that classifies objects based on color using artificial vision. This document presents a solution to the problem of manipulating objects using a robot manipulator arm "xArm" from LewanSoul, which is controlled and guided based on the color recognition of an object using MATLAB computer vision libraries. To solve this problem, the trial and error research methodology was used through the spiral method that is based on solving a problem in four phases starting with planning, followed by risk analysis, implementation and ending with evaluation to observe the results achieved and start again with the spiral if the results are not as expected. In the first phase, the objectives were planned and delimited, which were to recognize the color of the object to be manipulated using artificial vision techniques with the help of MATLAB and a Logitech C930 camera. As a second phase, the mathematics of the robot was analyzed to guide it to the point that defines the location of the object to be collected, in the same way the strategies for the creation of the graphical interface and the communication between software and hardware were analyzed. In this analysis, the types of controls needed to command the arm were evaluated, and a MATLAB GUI was used to create the graphical interface, in addition, serial communication was chosen to send data to the LewanSoul servo controller card. In the third phase, the system was implemented connecting the LewanSoul arm with the MATLAB PC and the Logitech C930 webcam through serial communication. Finally, the results obtained were evaluated and it was found that the system presents 97% fidelity in the control of its joints from the graphic interface, 100% in the recognition of colors (red and green) and 97% in terms of to the classification of objects, with minimal errors due to the PC processor that were solved when executing the actions again from the first phase of the spiral method. In the end, the job is finished concluding that the robot is reliable for use in industry within repetitive sorting processes.

**Key words**: <ROBOT ARM>, <ARTIFICIAL VISION>, <COLOR RECOGNITION>, <CLASSIFICATION OF OBJECTS>.

## INTRODUCCIÓN

Los últimos años se ha observado varios trabajos que mezclan conceptos de manipulación de objetos mediante brazos robóticos ayudados de visión artificial, con estos trabajos se han realizado avances en el campo de las aplicaciones para el hogar y la industria, pero se ha observado a través de la historia que la industria es el campo que más apuesta a esta tecnología ya que con ella se podría automatizar procesos en la manipulación y clasificación de objetos.

Este trabajo presenta una solución al problema de manipular objetos mediante un robot manipulador, el cual es controlado y guiado en base al reconocimiento de color de un objeto. Para desarrollar este proyecto se manejó el denominado "método de espiral" (metodología utilizada para el desarrollo de software) desarrollado por primera vez por Barry Boehm en 1986 según (Desarrollo en espiral, 2018).

El método de la espiral consiste en *determinar objetivos*, *analizar riesgos*, *desarrollar y probar* el algoritmo, y finalmente *evaluar* los resultados obtenidos que de ser los esperados darían por concluido el proyecto, de lo contrario se toman en cuenta las correcciones detalladas y se vuelve a ejecutar le método una vez más desde el principio.

En esta investigación se utilizaron técnicas de visión artificial con la ayuda de las librerías de MATLAB para reconocer dos colores, rojo y verde. También se utilizaron los métodos de cinemática directa para modelar el brazo robótico de LewanSoul y los métodos de cinemática inversa mediante la grabación de las coordenadas a las que se desea alcanzar con el robot. Para controlar el brazo robot se construyó una interfaz gráfica con controles deslizantes (sliders) utilizando una interfaz o ventana gráfica de usuario (GUI) que permite crear MATLAB. Para comunicar el brazo con la PC de control se escogió la comunicación serial para enviar y recibir datos hacia y desde una tarjeta controladora de servos. Para reconocer un objeto de color se diseñó un algoritmo que reconoce el color verde y rojo y clasifica el objeto dependiendo de su color.

El presente documento se organiza de la siguiente manera. Comenzando por la primera sección, se exponen las justificaciones y objetivos del proyecto. El sustento teórico de las técnicas utilizadas se expone en la sección 2. La arquitectura y el desarrollo del sistema propuesto se detallan en la sección 3. Por último, se exponen los resultados y conclusiones en la sección 4 de las pruebas realizadas manipulando objetos mediante un algoritmo creado para este efecto.

# CAPÍTULO I

## 1. PLANTEAMIENTO DEL PROBLEMA

### 1.1. Situación problemática

En Riobamba capital de la provincia de Chimborazo no existen varias empresas que utilicen un robot manipulador con visión artificial que clasifique objetos. Las industrias que tienen algún tipo de brazo robot dentro de sus líneas de producción operan mediante sensores y actuadores. Mientras que el resto de las empresas no poseen ningún tipo de robot desarrollando sus actividades y las realizan con personal humano de forma manual. Los robots a diferencia de los humanos pueden trabajar las 24 horas al día mejorando, y reduciendo el tiempo y costo de producción en las industrias.

Pero el problema en la industria recae en el gasto económico que incurre en la contratación de personal para realizar procesos repetitivos de clasificación que no son tan complicados y que se podrían confiar a un robot ya que se realizan diariamente sin mayor complejidad, a esto se le suma el costo de perdida en la calidad de los materiales terminados y en la cantidad de producción.

### 1.2. Preguntas directrices

- ¿Cuál es el mejor sistema de control que se adapta a la manipulación del brazo robot?
- ¿Qué limitantes puede producir al realizarse el control del brazo robot mediante la visión artificial?
- ¿Cómo controlar el brazo mediante la visión artificial?
- ¿Qué tipo de algoritmo se utiliza para procesamiento de imágenes?

### 1.3. Justificación de la investigación

Una de las líneas de estudio y de desarrollo muy interesante en robótica es el uso de la Visión Artificial. Con esta técnica no sólo se espera proporcionar a una máquina la inteligencia para observar, sino de reconocer o identificar características de imágenes y posteriormente tomar decisiones. Se ha avanzado poco en materia de inteligencia y visión artificial en la ciudad de

Riobamba, esto debido a la dificultad y complejidad que presenta el desarrollo de dichos proyectos que puede conllevar un alto costo para la empresa.

Debido al problema generado por el gasto económico que incurre en la contratación de personal para clasificar objetos es importante investigar un sistema que ayude a realizar este tipo de procesos repetitivos para impulsar al sector productivo de la ciudad de Riobamba, al utilizar procesamiento de imágenes por computador, que muestran una posible solución de la manipulación de objetos con un brazo robot por medio del reconocimiento del color.

## 1.4. Objetivos de la investigación

### *1.4.1. Objetivo general*

Desarrollar un sistema de control para un brazo robot que clasifique objetos en base al color utilizando visión artificial.

### *1.4.2. Objetivos específicos*

- Desarrollar una interfaz gráfica en GUI de Matlab para el control de los servomotores del brazo robot.

- Realizar la adecuación y calibración de las articulaciones del robot para la ejecución de órdenes deseadas.

- Desarrollar un algoritmo de reconocimiento de objetos en base a su color utilizando la visión artificial, para procesarlo desde la plataforma Matlab.

- Implementar una interfaz electrónica de potencia y de comunicación para monitorear y controlar los servomotores del mecanismo del manipulador.

## 1.5. Hipótesis

¿Al Implementar el sistema de control del manipulador por medio de visión artificial, permite que el brazo manipule y clasifique objetos en procesos repetitivos con un mínimo porcentaje de error basado en la detección de un color específico dentro del área de interés?

# CAPÍTULO II

## 2. MARCO TEÓRICO

### 2.1. Sistema manipulador de 6 GDL

Para desarrollar el presente proyecto se investigó los conceptos de manipulación de objetos usando un brazo robótico de 6 grados de libertad (GDL). Luego, se analizó la matemática mediante la cinemática directa y la cinemática inversa de un robot manipulador de 6 GDL.

#### *2.1.1. Brazos robot de 6 GDL*

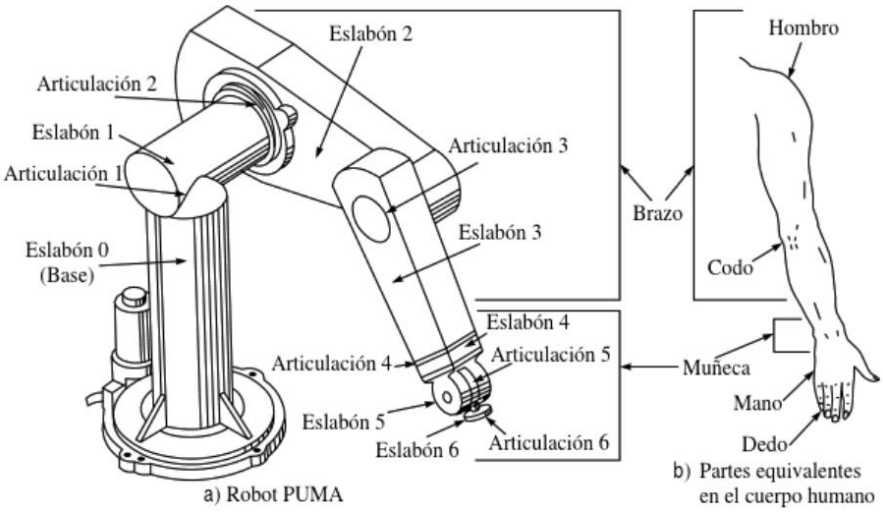



**Figura 1-2:** Robot serial manipulador equivalente a un brazo humano
**Fuente:** (Kumar Saha, 2008)

De acuerdo con Kumar Saha un robot manipulador o industrial es serial y puede ser clasificado en subsistemas como movimiento, reconocimiento, etc. (ver figura 1-2). Una de las aplicaciones de este tipo de robots es la manipulación de materiales y se lo puede clasificar también por el tipo de trabajo que realiza. También, según el autor se puede clasificar robots de acuerdo con el espacio o volumen de trabajo en que opera, es decir, de acuerdo con esto existen varias maneras de clasificar un robot. (Kumar Saha, 2008)

Kumar Saha clasifica un robot manipulador en tres subsistemas como son subsistema de: movimiento, reconocimiento y control. Según el autor el subsistema de movimiento tiene que ver con la estructura del manipulador semejante a un brazo de una persona, el subsistema de reconocimiento utiliza los datos de los sensores para recopilar información del manipulador, de los objetos a manipular y del entorno donde se desempeña el trabajo, ver figura 2-1. (Kumar Saha, 2008)

Luis Arnaez por otro lado, argumenta que la arquitectura del robot se refiere o tiene que ver directamente con el software y hardware que componen a los mismos. Por lo general el software siempre se implementa dentro de una tarjeta controladora y el hardware se refiere a la estructura que compone la máquina. Un robot que se asemeja al brazo de una persona y que sirve para tomar objetos por ejemplo es denominado "Manipulador". (Arnaez Luis, 2015)

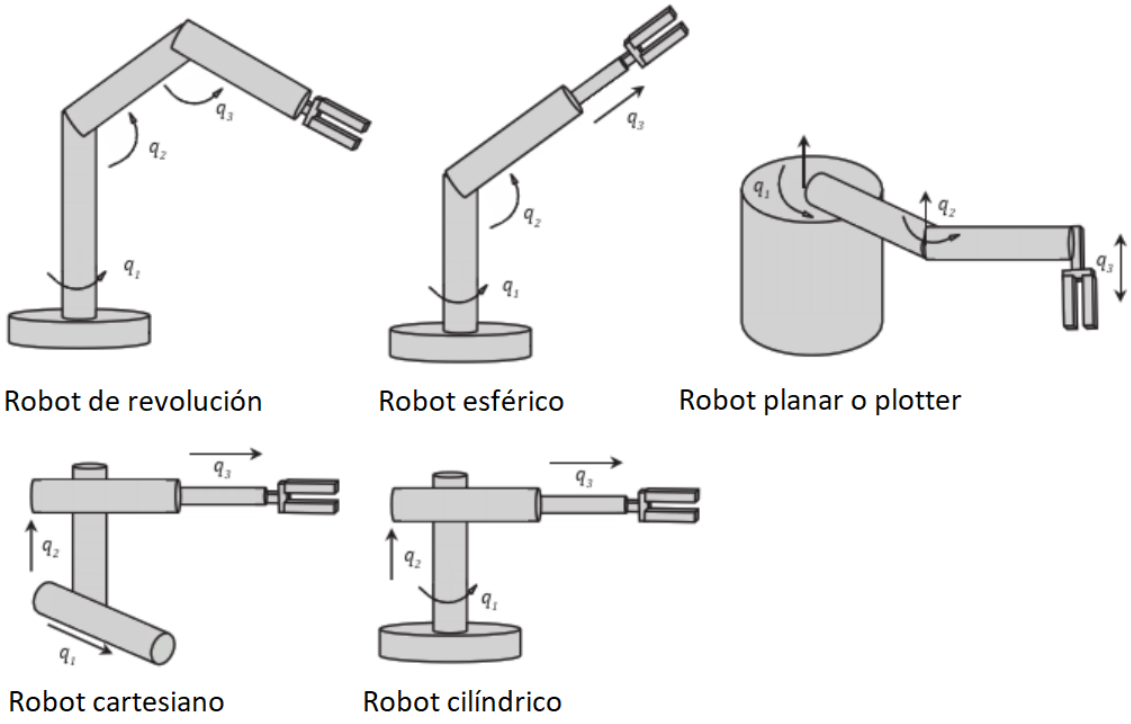



**Figura 2-2:** Tipos de Robot serial manipulador equivalente a un brazo humano
**Fuente:** (Arnaez Luis, 2015)

De acuerdo con Luis Arnaez hay algunas definiciones importantes que se deben conocer acerca de los robots, como son:

- *Grado de libertad (Degree of freedom "DOF").* - Es un movimiento elemental independiente.
- *Eslabón o link.* - Es la parte normalmente rígida que compone el robot.
- *Articulación (Joint o juntura).* - es la unión móvil entre los eslabones.
- *Mano o garra.* - Es el actuador del manipulador capaz de sujetar objetos.

Esta clasificación de manipuladores de acuerdo con el movimiento y la geometría que los robots presentan, se puede ver en la figura 2-3.

Ruiz-Velasco por su parte clasifica el campo de la robótica en dos grandes grupos: Robots Industriales y Robots no Industriales. Dentro los Robots Industriales clasifica a los que tiene la capacidad de operar dentro de industrias automatizadas y en cooperación con otros robots y controlados por computadoras. Comenta que la mayoría de manipuladores que se utilizan en la

industria son de forma antropomórfica y se asemejan al brazo de un humano debido a que con ese diseño pueden manipular y trasladar objetos. (Ruiz-Velasco, 2012).

La clasificación presentada por Ruiz-Velasco de las propiedades características que puede contar un manipulador industrial, son:

- *La base del robot (hombro).* - Es el soporte de giro del brazo robótico.
- *El brazo.* - Proporciona movimiento a la mano para tomar o posicionar objetos, de éste depende el espacio de trabajo y el alcance del robot.
- *El codo.* - Articulación importante del robot, ya que gracias a la geometría de éste el robot cuenta con libertad de movimiento.
- *La muñeca*. - Permiten el movimiento de la mano en el espacio de trabajo.
- *La mano u órgano efector o terminal.* - Es el elemento que se utiliza para tomar objetos o para darle cualquier otra función al robot.

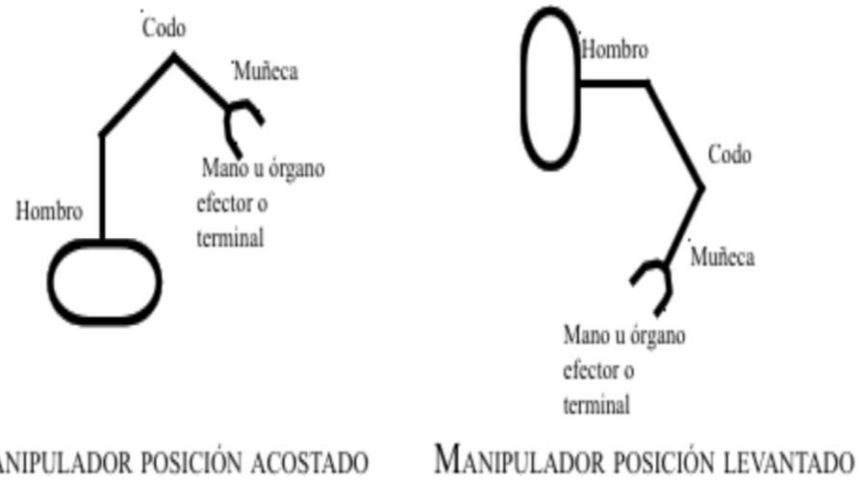



**Figura 3-2:** Manipulador industrial en diferentes posiciones
**Fuente:** (Ruiz-Velasco, 2012)

Por otro lado, Barrientos en su obra denomina brazo robot de 6 GDL al conjunto elementos que forman un robot muy parecido a un brazo humano y cuenta con seis articulaciones para maniobra. Un grado de libertad (GDL) se denomina al movimiento independiente que puede realizar cada articulación para posicionar el robot en un punto medido respecto a la anterior articulación. Cada articulación cuenta con un eje de giro y se puede rotar en torno a este (Barrientos, 2007).

Los brazos robóticos de 6 GDL permiten alcanzar muchos más puntos que otros dentro del área de trabajo, lo que los hace perfectos para trabajar en ambientes industriales como parte de un proceso de producción (Barrientos, 2007).

La figura 2-4 expone el ejemplo de brazo robótico de 6 GDL

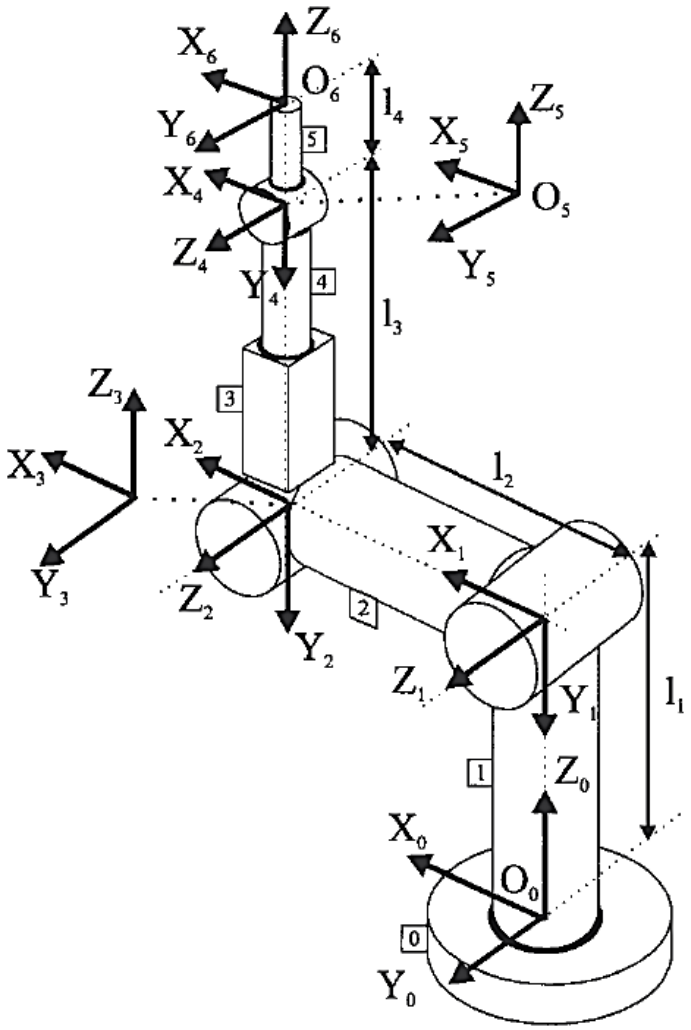



**Figura 4-2:** Ejemplo de brazo robótico de 6 DGL
**Fuente:** (Barrientos, 2007)

El concepto de GDL según Pérez Marco es muy importante en la robótica industrial, por lo cual también se cita la definición de que define un grado de libertad (DOF, a partir de sus siglas en inglés: degree-of-freedom) como una coordenada independiente sobre la que se puede modelar y estudiar de forma matemática el sistema robótico o mecatrónico. Así mismo, el autor menciona también que se puede asociar la suma de GDL en un dispositivo con la cantidad de eslabones y actuadores bajo ciertas consideraciones. Sin embargo, en la figura 2-5 se puede observar que este último párrafo se contradice ya que los robots que ahí se presentan tienen 4 articulaciones móviles, pero solo cuentan con un grado de libertad. (Pérez Marco, 2014)

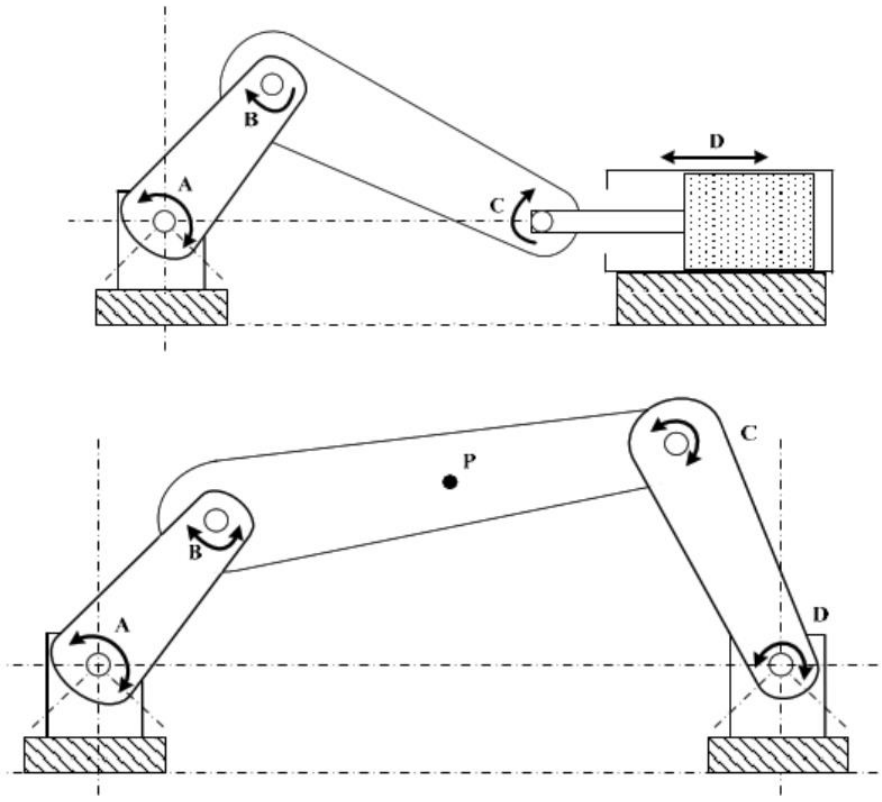



**Figura 5-2:** Ejemplo de brazos robóticos de 4 articulaciones y un grado de libertad
**Fuente:** (Pérez Marco, 2014)

Se puede definir los tipos de articulaciones más comunes y el número de DOF que puede aportar cada una según Pérez Marco tal como se observa en la figura 2-6. Desde esa óptica el autor menciona que es posible establecer las capacidades de movimiento de una articulación dependiendo del número de restricciones que impone para el libre movimiento del eslabón en el espacio. (Pérez Marco, 2014)

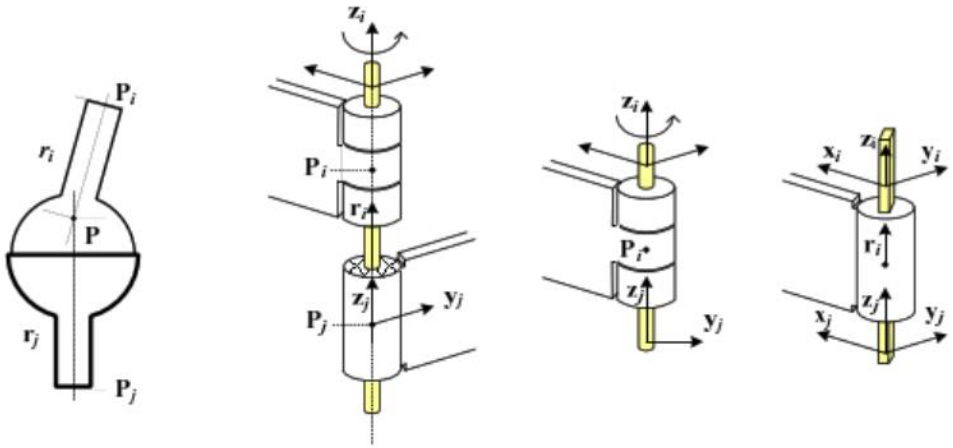



**Figura 6-2:** Cuatro tipos de articulaciones más comunes
**Fuente:** (Pérez Marco, 2014)

### *2.1.2. Cinemática directa de un robot de 6 GDL*

La cinemática directa en brazos robóticos estudia el comportamiento matemático de los mismos para analizar matemáticamente como varia la ubicación y orientación del actuador final de acuerdo con los valores adoptados por sus articulaciones. Al proporcionar diferentes valores a las articulaciones se observa el movimiento de todo el brazo robot, hecho por el cual surge la necesidad de estudiar técnicas para analizar matemáticamente como varia la ubicación y orientación del actuador final del robot. (Barrientos, 2007).

Por otro lado, Pérez Marco también se refiere a la cinemática como un área que estudia aspectos geométricos del movimiento de los cuerpos. El cálculo de la cinemática inmiscuye conceptos que definen aceleración, velocidad, orientación y posición. El modelo cinemático tiene por objetivo entender el movimiento de los eslabones en la estructura robótica, mediante el estudio de las características de movimiento que permite la arquitectura mecánica de cada articulación que se encarga de accionar los eslabones. (Pérez Marco, 2014)

Pérez Marco considera que las articulaciones y eslabones ejecutan una tarea específica, y que el análisis cinemático adquiere relevancia al permitir el estudio integral de los movimientos individuales en cada articulación y su integración dentro del sistema. Si se dispone de un modelo cinemático sobre la interacción entre los distintos componentes del sistema menciona el autor, un ingeniero puede diseñar métodos y algoritmos para controlar la operación a través del control de cada una de las articulaciones. Puede incluso cumplir las restricciones particulares que cada tarea exige, como, por ejemplo: evitar posiciones prohibidas, controlar velocidades o aceleraciones máximas, limitar fuerzas, etc. (Pérez Marco, 2014)

La posición del extremo final se puede representar en coordenadas tridimensionales que detallen los puntos *x, y, z* dentro del espacio. Por otro lado, la orientación del extremo final se puede representar con ángulos de Euler ($\phi$, $\theta$, $\psi$). Existen varias técnicas para estudiar la cinemática directa de brazos robóticos, pero la que se utilizará en este proyecto se basa en el método del cambio de sistemas de referencia mediante matrices de transformación homogénea (MTH). Las matrices MTH permiten representar las características espaciales de los eslabones como son posición, orientación. Dichas características definen la ubicación en el espacio tridimensional con respecto al origen de un sistema de referencia fijo. (Barrientos, 2007).

De acuerdo con el método escogido en el presente trabajo para analizar la cinemática directa del robot de 6 GDL es el de Denavit-Hartenberg (D-H) el cual se basa en un análisis matricial. En base a 4 reglas que el algoritmo D-H propone se obtiene la denominada matriz $^{i-1}A_i$ (2.1), que

permite ubicar un eslabón *i* en el espacio mediante la posición y orientación con respecto al de un eslabón *i-1*. El método D-H utiliza un sistema de referencia fijo y *n* sistemas móviles de referencia ubicados con las reglas D-H en cada una de la *n* articulaciones, estos sistemas (fijos y móviles) se pueden observar en la figura 2-3. Luego, en base a cuatro reglas expuestas por los autores conocidos como Denavit y Hartenberg se rellena una matriz denominada como "tabla de parámetros D-H" que se muestra en la tabla 2-1. (Barrientos, 2007).

$$ {}^{i-1}_{\square}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i S\theta_i & -S\alpha_i S\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

**Tabla 1-2:** Tabla de parámetros de Denavit-Hartenberg

| *# Articulaciones* | $\theta$ | $d$ | $a$ | $A$ |
|---|---|---|---|---|
| $i$ | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |

**Fuente:** (Barrientos, 2007)

En otro punto de vista similar Kumar Saha menciona, es necesario especificar las coordenadas del punto al que se desea llegar con el manipulador para que éste ejecute una tarea determinada. La posición se debe proporcionar con respecto al origen de un sistema de coordenadas fijo. Según el autor, la matemática del análisis de posición establece una relación que compromete las coordenadas cartesianas de la ubicación y la orientación del efector final con los valores o ángulos que toman las articulaciones de cada eslabón para alcanzar dicho punto. De lo dicho en este párrafo nacen los conceptos particulares que todos mencionan como "cinemática directa" y "cinemática inversa". La figura 2-7 representa de forma gráfica lo que se ha mencionado sobre cinemática. (Kumar Saha, 2008)

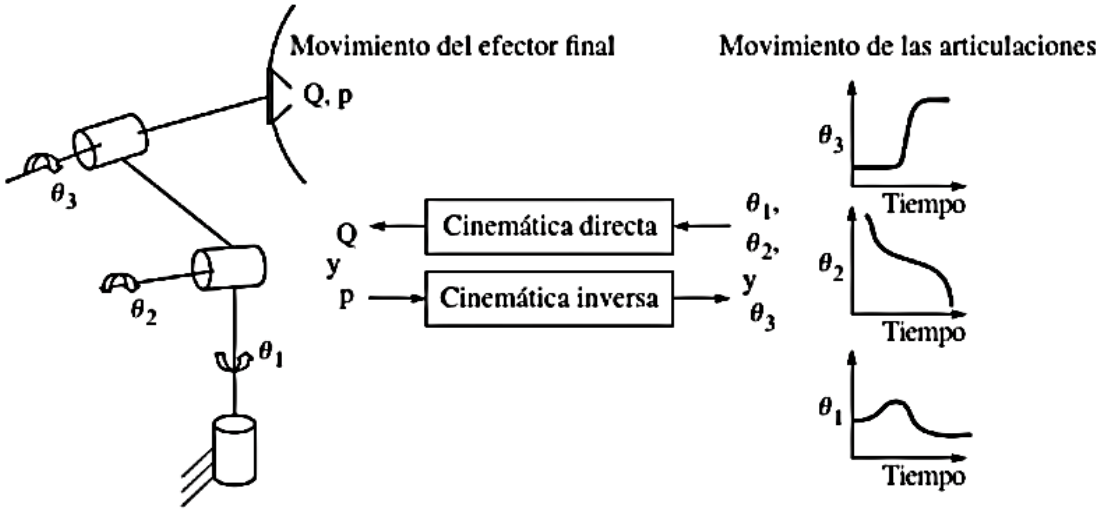



**Figura 7-2:** Diferencias entre cinemática directa e inversa
**Fuente:** (Kumar Saha, 2008)

En la cinemática directa, Kumar Saha menciona que el objetivo en éste análisis es encontrar las características de posición y orientación del extremo final del manipulador al conocer los valores de las articulaciones móviles que lo componen. Por otro lado, el autor también menciona que el objetivo de la cinemática inversa es todo lo contrario a lo mencionado anteriormente, es decir en este caso lo que se busca es encontrar los valores de las articulaciones a partir de los datos de posición y orientación que para este caso son conocidos. (Kumar Saha, 2008)

Del mismo modo, Kumar en su trabajo demuestra que la MTH denominada en este caso como T, puede representar la posición y orientación del efector final con referencia al origen. Esta matriz T se obtiene posmultiplicando las MTH individuales que se generaron al modelar el brazo con los parámetros D-H antes mencionados como se muestra en la ecuación 2.2. (Kumar Saha, 2008)

$$T = T_1 T_2 \dots T_n \tag{2.2}$$

Una vez obtenido la matriz total de la posmultiplicación de cada una de sus matrices de transformación homogénea se puede observar la rotación y la posición final del efector final. Según Pérez Marco, analizando los movimientos que componen la convención D-H es factible utilizar sus conceptos para presentar la posición y orientación de los eslabones con respecto a un sistema base. De esta forma, la convención D-H puede expresarse a través de una matriz homogénea T total, que registra cada uno de los pasos de la convención entre los dos sistemas coordenados de la articulación n-1 a la articulación n. En la figura 2.8 se puede observar la matriz generalizada según los parámetros DH para la transformada homogénea. (Pérez Marco, 2014)

$$ {}^{n-1}\mathbf{T}_n(\theta_n) = \begin{bmatrix} cos\theta_n & -sen\theta_n cos\alpha_n & sen\theta_n sen\alpha_n & a_n cos\theta_n \\ sen\theta_n & cos\theta_n cos\alpha_n & -cos\theta_n sen\alpha_n & a_n sen\theta_n \\ 0 & sen\alpha_n & cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

**Figura 8-2:** Diferencias entre cinemática directa e inversa
**Fuente:** (Pérez Marco, 2014)

De acuerdo con Marco Pérez tres parámetros de la convención DH son constantes mientras que, dependiendo de la naturaleza de la articulación, el parámetro θ o el parámetro d se consideran como variables. Con el objetivo de mantener claridad en la representación DH, una costumbre muy común es adoptar la variable $q_n$ para referirse a la variable de cada eslabón, sea el parámetro θ o el parámetro d. Por esta razón, la variable $q_n$ se denomina coordenada generalizada (dado que se puede referir a eslabones rotacionales o prismáticos), mientras que el vector que contiene el valor de todas las coordenadas generalizadas se denomina vector de estado o vector de coordenadas generalizadas (q). (Pérez Marco, 2014)

### *2.1.3. Cinemática inversa (CI) de un robot de 6 GDL*

En la cinemática inversa se observa el proceso contrario al de cinemática directa, es decir encuentra las magnitudes que deberían tomar las articulaciones del manipulador en base a un punto de ubicación y orientación proporcionado para el efector final del mismo. Existen algunas soluciones para la matemática de la cinemática inversa, pero todos ellos dependen fuertemente de cálculos matemáticos, lo cual no garantiza encontrar una única solución. (Barrientos, 2007).

Para garantizar el resultado de la cinemática inversa se puede proporcionar reglas en cuanto a los límites del área de trabajo y los límites de recorrido de las articulaciones. Si se trata de un robot de 6 GDL se puede resolver la cinemática inversa descomponiendo al robot en dos partes, es decir, analizando las tres primeras articulaciones como un robot de 3 GDL, y luego, analizando las tres últimas articulaciones que por lo general son destinadas a la orientación del extremo final del manipulador. (Barrientos, 2007).

Otra técnica existente y muy utilizada en la industria, es la de grabar con anterioridad los puntos a los que se desea llegar junto con el valor que adoptan las articulaciones. Estos puntos son encontrados al guiar el brazo robótico mediante un control manual ("Joystick" por sus siglas en inglés) utilizando la cinemática directa y se puede llegar a ellos calculando el recorrido más

grande que debe hacer una de las articulaciones para darle proporciones iguales de recorrido al resto de las articulaciones. (Barrientos, 2007).

Por otro lado, según Pérez Marco la solución al problema de cinemática inversa reviste una importancia especial, ya que las tareas que va a realizar el robot pueden definirse en términos de posiciones y orientaciones en el espacio. Por lo tanto, puede recurrirse a definir longitudes expresadas en metros y valores angulares definidos en radianes, lo cual facilita el diseño y desarrollo de controladores robótico. (Pérez Marco, 2014)

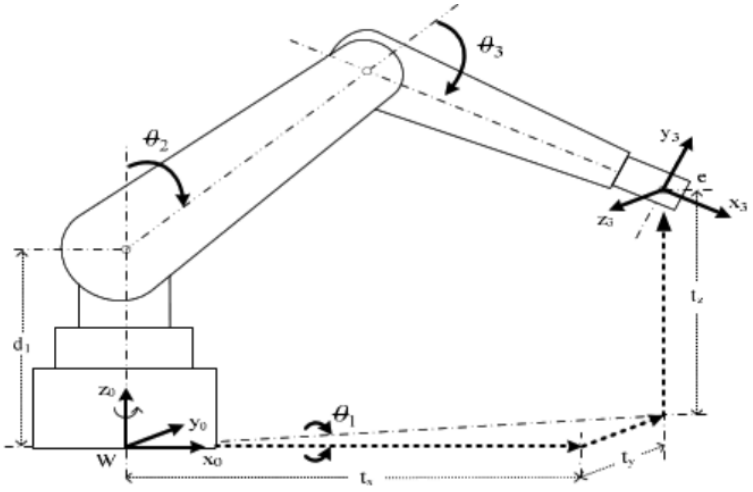



**Figura 9-2:** Representación gráfica de la cinemática inversa
**Fuente:** (Pérez Marco, 2014)

La solución a este problema de acuerdo con Pérez Marco puede también simplificar y mejorar el diseño de sistemas mecatrónicos con articulaciones de un solo grado de libertad y cuyos eslabones se estructuran en cadenas que conforman la estructura robótica. (Pérez Marco, 2014)

El concepto de cinemática directa explica Pérez Marco es la operación que permite conocer la ubicación y orientación que presenta la herramienta que sujeta el final del manipulador, considerando el estado de cada una de las articulaciones y eslabones que forman el dispositivo. Es decir, la cinemática directa define una MTH, la cual representa la ubicación y la inclinación del actuador del manipulador, que se detalla a partir de las magnitudes que presentan las coordenadas generalizadas, es decir, a partir del estado instantáneo de los eslabones que se posicionan gracias a las articulaciones que los sujetan. (Pérez Marco, 2014)

Pérez Marco explica también que la cinemática inversa será un arreglo unidimensional de coordenadas $q$ requeridas para posicionar y orientar el extremo del manipulador y representar este particular con una MTH total conocida como ${}^{0}T_{n}$. Matemáticamente, cada uno de los valores de

*q* interviene en el cálculo de cinemática, es decir, para el manipulador de la figura 2-9 dichos valores serán θ1, θ2 y θ3. Y al resolver la cinemática inversa para dicho robot la ecuación de análisis quedaría como se puede observar detalladamente en la figura 2-10. (Pérez Marco, 2014)

$$\textbf{Dado:} \quad {}^{w}T_{e}(q) = {}^{0}T_{1}(\theta_1).\ {}^{1}T_{2}(\theta_2).\ {}^{2}T_{3}(\theta_3)$$

$$\textbf{Calcular:} \quad q = [\theta_1 \quad \theta_2 \quad \theta_3]$$

**Figura 10-2:** Representación matemática de la cinemática inversa del manipulador de la figura 2-9 donde se observa la posmultiplicación de tres articulaciones

**Fuente:** (Pérez Marco, 2014)

$${}^{w}T_{e}(q) = \begin{bmatrix} n_x & o_x & a_x & t_x \\ n_y & o_y & a_y & t_y \\ n_z & o_z & a_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{0}T_{1}(\theta_1).\ {}^{1}T_{2}(\theta_2).\ {}^{2}T_{3}(\theta_3)$$

**Figura 11-2:** Representación matemática mediante la MTH total del robot de la figura 2-9
**Fuente:** (Pérez Marco, 2014)

Los parámetros conocidos son la orientación y la posición final de la herramienta del robot detallada en la figura 2-11, donde ***n*** representa la orientación del efector final con respecto al eje *x*, ***o*** representa la orientación del efector final con respecto al eje *y*, ***a*** representa la orientación del efector final con respecto al eje *z*.

En la obra de (Pérez Marco, 2014) se menciona que la solución del problema de cinemática inversa no es tan trivial, presentando algunos inconvenientes que se mencionan a continuación:

- Es posible que existan numerosas soluciones al problema de CI para una localización (ubicación e inclinación) deseada para el actuador final.
- Las ecuaciones matemáticas para definir la CI son comúnmente no lineales, y esto impide en determinados casos calcular una respuesta de forma adecuada.
- Existe también el riesgo de calcular respuestas inadmisibles para un sistema robótico en virtud de limitaciones mecánicas. Es decir, una solución de cinemática inversa puede resultar en valores de coordenadas generalizadas imposibles de cumplir para el sistema robótico, debido principalmente a sus restricciones mecánicas.

Otro método para solucionar la cinemática inversa es el método iterativo que menciona (Pérez Marco, 2014). El método iterativo de solución al problema de cinemática inversa utiliza métodos numéricos que permiten minimizar desviaciones encontradas entre configuraciones iniciales del robot manipulador F(q) con respecto a la ubicación deseada de la que se desea calcular el arreglo unidimensional de coordenadas generalizadas. El autor menciona que el algoritmo bien puede comenzar a partir de una configuración inicial con valores de cero en cada una de las coordenadas generalizadas, aunque esto, como se discute más adelante, representa algunos inconvenientes.

La cinemática deseada se puede definir mediante la matriz transformada que determina la posición objetivo, representada por *Td*, que representa dicho de otra forma el problema que se desea solucionar. En otras palabras, la respuesta que da solución a la cinemática inversa precisa la determinación del conjunto de valores que el arreglo unidimensional de coordenadas generalizadas denominado $q = [\theta 1, \theta 2, \theta 3]$ que son representan los valores que se requieren calcular para alcanzar la ubicación e inclinación deseados. (Pérez Marco, 2014)

### *2.1.4. Trayectorias coordinadas o isócronas*

Para este trabajo se ha seleccionado la técnica de control comandada por controles manuales, ya que se puede guiar al robot desde los controles de una interfaz gráfica creada en MATLAB y guardar los valores de las articulaciones al alcanzar los valores deseados para alcanzar con el brazo robot una ubicación e inclinación deseadas. El detalle de este proceso se presenta en la sección 3.

Para unir dos puntos consecutivos en una trayectoria existen diferentes métodos. Una forma es moverse de un punto a otro logrando que los servomotores giren el ángulo necesario para alcanzar la posición deseada. Este movimiento se puede hacer utilizando el concepto de trayectorias coordinadas o isócronas. Este tipo de trayectoria describe un movimiento coordinado de los motores ya que se hace un análisis previo para conocer cual motor tiene que recorrer mayor distancia. Una vez que se conoce el tiempo del motor (TM) que debe recorrer la mayor distancia se proporciona pasos a los demás motores ajustando los mismos al tiempo TM. El accionamiento de todos los motores se lo hace al mismo tiempo de forma isócrona, lo que a la vista del usuario parecerá un movimiento suave y controlado. Esto evita que cada motor alcance su posición final independientemente y describiendo un movimiento brusco del efector del manipulador robótico. (Barrientos, 2007)

## 2.2. Visión Artificial

### *2.2.1. Proceso de visión artificial*

La "visión artificial" o dicho de otra manera "visión por computador" es una ciencia que mezcla procesos para procesar, analizar y comprender lo que comprenden las imágenes del exterior con el propósito de presentar información de forma numérica para que puedan ser interpretados y tratados por un procesador. Todo esto, para crear procesos con propósito de acuerdo con el usuario que lo solicita. (Núñez, 2017)

De la misma manera que las personas utilizan los ojos para visualizar objetos y el cerebro para procesar las imágenes, la visión por computador intenta imitar este suceso para que las computadoras puedan observar e identificar una imagen o secuencia de imágenes para realizar un proceso según determinados objetivos utilizando su unidad de procesamiento computarizado o CPU. (Núñez, 2017)

La "visión artificial" o "visión por computador" se forma por varios procesos que orientan a realizar unos varios análisis o procesamiento de imágenes. Dichos procesos se denominan según (Núñez, 2017) como:

- Lectura de imágenes
- Almacenamiento de información
- Interpretación y procesado de datos

Según el mismo autor Núñez antes mencionado el proceso de la visión artificial es posible si el equipo cuenta con los siguientes módulos, ver figura 2-12.

- Sección de digitalización
- Almacenamiento de imágenes
- Dispositivo para visualización
- Procesadores de datos
- Módulo de entradas y salidas

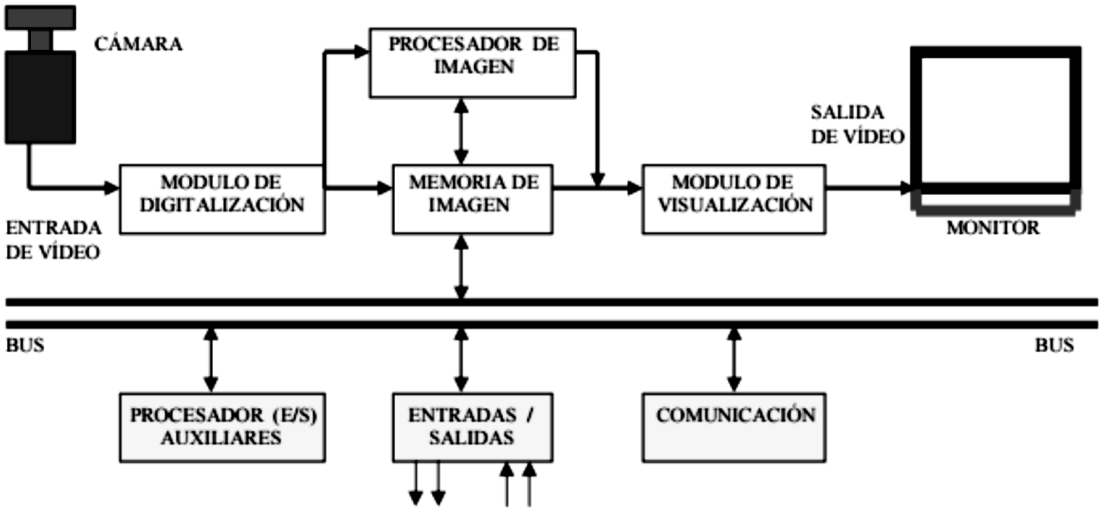



**Figura 12-2:** Representación gráfica del proceso de visión artificial
**Fuente:** (Núñez, 2017)

- *Módulo de digitalización*: Transforma señales analógicas entregadas por la cámara a señales digitales (para su posterior análisis).
- *Memoria de imagen*: Guarda las señales provenientes del módulo de digitalización.
- *Módulo de visualización*: Transforma las señales digitales guardadas en la memoria de imagen en señales de vídeo analógicas para visualizarlas en la TV.
- *Procesador de imagen*: Analiza, evalúa e interpreta imágenes recibidas por la cámara digital.
- *Módulo de entradas/salidas*: Procesa las entradas de sincronismos captadas de imágenes y las salidas que infieren a los dispositivos externos de acuerdo con el resultado de la interpretación.

Por otro lado, Núñez menciona que para el procesamiento de imágenes la "visión artificial" requiere de procesos que permiten captar e interpretar imágenes para su posterior análisis. Estos procesos se denominan como se detallan a continuación, ver figura 2-13.

- Adquisición
- Procesamiento de la imagen
- Representación y descripción (extracción de características*)*
- Reconocimiento e interpretación.

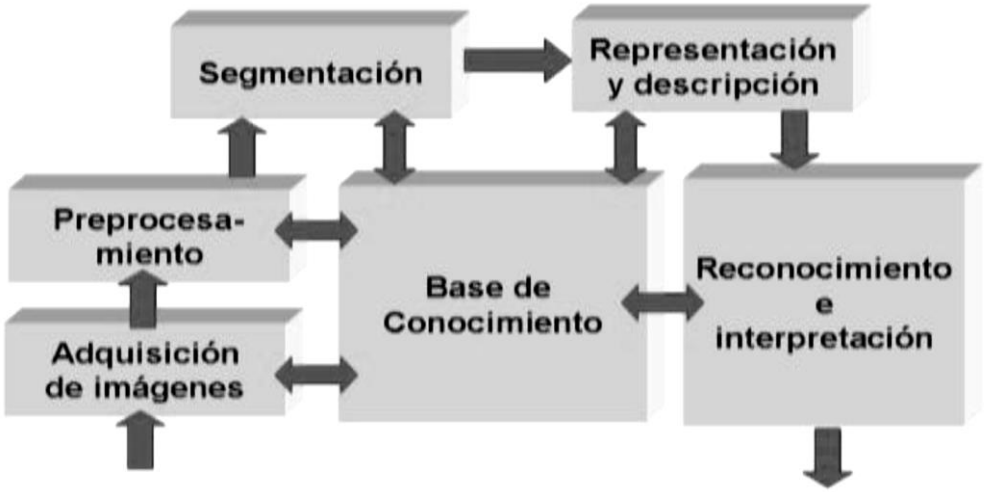



**Figura 13-2:** Representación gráfica del proceso de visión artificial
**Fuente:** (Núñez, 2017)

- *Adquisición:* Este proceso conlleva a conseguir la representación digital de una imagen obtenida e ingresada hacia un ordenador o computadora. Siendo éste proceso el que permitirá realizar las operaciones en la mencionada imagen para lograr gestionar la interpretación digital. La captura de una imagen tendrá que ser lo bastante clara y estable posible para poder realizar un adecuado procesamiento y tratamiento de dicha imagen. (Núñez, 2017)
- *Procesamiento de la imagen:* Para esta etapa la imagen es procesada con el único fin de obtener de ésta información suficiente para tratarla. La información obtenida será lo más confiable posible en el marco de lo que será el procesado y el reconocimiento. Cuenta mucho la gama del procesador o controlador del computador ya que será un efecto clave para procesar de manera rápida. Esta etapa se subdivide en las etapas de *binarización* y *segmentación*. (Núñez, 2017).
- La *binarización* es el proceso que comprende la representación de imágenes en ceros y unos, por lo que la convierte en la etapa importante donde una imagen en niveles de gris (o color) pasa a ser representada como una imagen binaria. Con este proceso se consigue también menorar el número de datos a procesar de una imagen (Núñez, 2017).
- La *segmentación* es el proceso que comprende la detección de objetos o regiones de interés en las imágenes binarizadas. Este proceso depende directamente del proceso de binarización, es decir que si la imagen fue binarizada de manera adecuada el proceso de segmentación será adecuado también. (Núñez, 2017)
- *Representación y descripción:* A este proceso se lo conoce también como la etapa donde se extraen todas las características de una imagen. Dichas características son de tipo morfológicas, es decir, puede ser el área, el perímetro, la excentricidad, los puntos inerciales, los perfiles, y además también es posible extraer ciertas características de acuerdo con la textura o el color. Una vez que se extraen dichas características cada región se puede clasificar

e interpretar para luego ser usadas con fines específicos dentro del tratamiento de las imágenes. (Núñez, 2017)

- *Reconocimiento e interpretación:* Esta última etapa comprende la interpretación y reconocimiento de las imágenes por medio del procesador de la computadora, y una vez conseguidos los datos con los procesos antes mencionados se puede realizar pruebas, mediciones y cálculos de interés. Según Núñez en la actualidad se conocen gran cantidad de algoritmos que proporcionan determinados resultados esperados, por ejemplo: (Núñez, 2017)

- Identificación de figuras
- Detección de características geométricas
- Determinación de puntos característicos a partir de patrones
- Medición de ciertas características de objetos
- Identificación de objetos especiales o difusos

Núñez en su trabajo se referirse a los "sistemas de visión artificial" como aquellos que proveen a un computador (CPU) la capacidad de entender y procesar imágenes o las características de ésta. Por lo cual se define a un "sistema de visión artificial" como aquel que puede interpretar la información del mundo real con la ayuda una cámara que captura imágenes y las analiza mediante procesador o computador. Se dice también que los elementos básicos que debería poseer un "sistema de visión artificial" son: (Núñez, 2017)

- Suplidor de luz
- Sensores que permitan captar imagen
- Sistemas que posean la capacidad de adquisición de datos
- Aplicación de algoritmos que procesen los datos

### *2.2.2. Operaciones con píxeles*

Una vez que se digitaliza una imagen se puede representar con una matriz de M columnas y N filas o por un arreglo unidimensional de N por M elementos, como se puede ver en la siguiente ecuación 2.3. (Núñez, 2017)

$$i = \begin{bmatrix} i_{11} & i_{12} & \cdots & i_{1M} \\ i_{21} & i_{22} & \cdots & i_{2M} \\ \vdots & \vdots & \cdots & \vdots \\ i_{N1} & i_{N2} & \cdots & i_{NM} \end{bmatrix} = \begin{bmatrix} i_{11} \\ \cdots \\ i_{1M} \\ \cdots \\ i_{NM} \end{bmatrix} \tag{2.3}$$

En la ecuación 2.3 cada elemento de ésta determina la posición y el valor de un pixel. Dicha matriz puede almacenar de forma digital las características de una imagen con todos sus pixeles. Una vez que se tiene el archivo se puede analizar el mismo al procesarlo matricialmente en una computadora.

Sobre el archivo digital que comprende las características digitales de una imagen se pueden guardar pixeles de forma binaria o monocromática donde cada uno tendrá un valor de 1 o 0, por otro lado, un pixel que contiene datos de una imagen a colores se representa mediante un arreglo tridimensional para almacenar en números diferentes de 0 y 1 los tres colores denominados primarios como son: rojo, verde y azul. (Núñez, 2017)

Las operaciones que se pueden realizar con los píxeles, por lo general son operaciones: *operaciones geométricas* y *aritmético-lógicas*.

Las operaciones **aritmético-lógicas** son muy utilizadas en cualquier nivel de un sistema de procesamiento de imágenes, esto debido a que son utilizadas para interpretar y generar pesos a los diferentes píxeles que se puedan tener en una imagen, entre las diferentes operaciones básicas están las denominadas, conjunción, disyunción, negación, suma, resta, multiplicación y división. (Núñez, 2017).

- *Conjunción.* - Operación lógica AND, es usada para quitar píxeles.
- *Negación.* - Inversión, es usada para proporcionar el negado de una matriz de pixeles.
- *Disyunción.* - Operación lógica OR, es usada para añadir píxeles.
- *Suma.* – Opera con los pixeles de la misma manera que la suma de dos matrices.
- *Resta.* – Opera con los pixeles de la misma manera que la resta de dos matrices.
- *Multiplicación.* – Opera con los pixeles de la misma manera que la multiplicación de dos matrices.
- *División.* – Opera con los pixeles de la misma manera que la división de dos matrices

Las operaciones **geométricas** más usadas son la traslación, escalado y rotación, mismas que se describen a continuación:

- *Traslación.* - Traslada los píxeles con un vector de traslación.
- *Escalado.* – Escala el tamaño de los pixeles.
- *Rotación.* – Gira los píxeles de acuerdo con un origen de coordenadas.

Es preciso recordar que las operaciones de multiplicación de matrices no cumplen la propiedad conmutativa. (Núñez, 2017)

### *2.2.3. Filtrado de imágenes*

Los filtros son utilizados para aumentar la calidad de las imágenes capturadas eliminando ruido de las mismas. Entre las principales clases de filtros están los filtros lineales y lo filtros no lineales que se explican a continuación.

Los **filtros lineales** han sido utilizados comúnmente, pero, aunque sus resultados son adecuados en algunos sistemas los filtros lineales no tienen mucho efecto. (Núñez, 2017)

En el proceso de análisis de las características de imágenes los filtros lineales atenúan la intensidad de los contornos, no remueven completamente el ruido y ejercen poca efectividad. Además, numerosos estudios han demostrado que los primeros pasos de procesamiento de imágenes presentan características no lineales. Por esto, desde 1958 se han realizado estudios que tiene relación con las técnicas no lineales para eliminar ruido en las señales e imágenes. Es así que el filtrado no lineal ha crecido rápidamente debido a su gran popularidad en aplicaciones de telecomunicaciones y procesamiento de imágenes. (Núñez, 2017)

Gran cantidad de industrias de software que se enfocan al procesado y análisis de imágenes centran la atención en filtros no lineales que en son conocidos en algunos casos como filtros de mediana y en otros casos como filtros morfológicos. (Núñez, 2017)

Los **filtros no lineales** por otra parte no cumplen el principio de superposición y el principio de proporcionalidad, razón por la cual se limita matemáticamente en ciertos tipos de operaciones. En base a lo expresado anteriormente un filtro no lineal en la salida no es igual a la convolución de la entrada. (Núñez, 2017)

La técnica de análisis digital de las características de las imágenes utiliza convertidores que pasan de escala que representa grises $x$ a otra escala $y$ con la función $y = t(x)$. También se han dado

casos en los que se ha empleado la adaptación del histograma gráfico (figura que representa niveles de gris) que permite cambiar un nivel de gris a otro nivel representativo. (Núñez, 2017).

Un proceso donde actúa un filtro no lineal es aquel donde se evidencian comparaciones entre dos imágenes que se han sometido a diferentes ambientes de iluminación con una plantilla común (estándar) donde se observa un histograma que presenta una repartición uniforme de niveles. (Núñez, 2017)

Los tipos de filtros no lineales pueden ser varios como se detallan a continuación:

- Filtros de orden estadístico:
- Filtro de mediana.
- Filtros de pila
- Filtros híbridos de mediana
- Filtros de orden clasificado
- Filtros de medía de estado
- Filtros L
- Filtros M
- Filtros R
- Filtros morfológicos
- Filtros homomórficos
- Filtros polinomiales
- Filtros adaptivos

### *2.2.4. Algoritmos de detección de bordes*

Según Jorge Valverde un algoritmo que identifica bordes es muy útil y generalmente muy usado en la identificación de características de objetos y en ocasiones donde se requiere de la segmentación de regiones. Estos algoritmos ayudan a detectar contornos de una imagen al analizar en su matriz y encontrar en sus pixeles un cambio brusco de nivel de gris, para esto utiliza máscaras de convolución con la primera derivada. Uno de los algoritmos que se utiliza para este fin se denomina "Canny". (Valverde Rebaza, 2019)

Valverde menciona que *Canny* investigó y propuso una técnica denominada con el nombre del autor para la detección e identificación de bordes fundamentada en tres principios fundamentales conocidos como:

- *Detección.*- Evita borrar contornos relevantes y no proporcionar contornos falsos.
- *Localización.*- Minimiza la distancia entre la localización y la posición real del borde.
- *Respuesta integradora.*- Unifica el resultado a un sólo contorno.

Jorge Valverde describe que la detección de contornos o bordes utiliza métodos enfocados en la primera derivada la cual permite reconocer valores cercanos a cero en las regiones en las cuales la intensidad no cambia y presenta un valor no cambiante o constante en la etapa de intensidad. Lo anterior expresado según Jorge Valverde recae en que se puede detectar un cambio de intensidad cuando se presente un cambio repentino y brusco en el cálculo de la primera derivada, permitiendo que esta repentina característica sea utilizada como mensaje para reconocer un contorno o borde (Valverde Rebaza, 2019).

El algoritmo de "Canny" según Jorge Valverde consiste en realizar tres pasos fundamentales:

- *Cálculo de gradiente*: Encuentra la orientación y magnitud del arreglo unidimensional del gradiente en todos los pixeles.
- *Supresión no máxima*: Adelgaza el tamaño de los contornos o bordes encontrados con el cálculo del gradiente, esto lo realiza hasta obtener contornos de ancho deseado.
- *Histéresis de umbral*: Utiliza una función que se centra en dos umbrales que reducen la probabilidad de mostrar bordes falsos.

## 2.3. Interfaces de programación mediante MATLAB

En este punto se estudió el procesamiento de imágenes con MATLAB y el desarrollo de la GUI para una adecuada presentación y manejo del brazo robot.

### *2.3.1. Conceptos de MATLAB*

MATLAB es un software de cálculo utilizado por millones ingenieros y científicos. Presenta un entorno amigable con un lenguaje de programación fácil de manejar, se lo utiliza en varias ocasiones para el análisis de procesos. Cuenta con herramientas denominadas *toolboxes* para varios campos de ingeniería. Su logo representativo se muestra en la figura 2-14. (Descripción general, 2018)

Para aprender a utilizar MATLAB existen muchos sitios recomendables en la red, sin embargo (Albesa, 2015) en su trabajo escrito señala que al abrir este paquete matemático se observa las siguientes subdivisiones de ventanas que se detallan a continuación:

- La ventana Command Window (ventana central) es donde se introducirán los comandos, variables e instrucciones a realizar. Es decir, la ventana donde se trabaja.
- La ventana Current Folder (izquierda superior) indica el contenido del directorio en el que se está trabajando, y que, salvo cambio del mismo, es donde se irá guardando el archivo o las funciones que se hayan creado.
- La ventana de Workspace (derecha superior) es la ventana donde se indican las variables definidas en la sesión de trabajo o ya guardadas y cargadas de otras sesiones. Para borrar alguna de ellas, se usará el comando "clear nombredevariable" o se selecciona la variable en esa ventana y se suprime.
- La ventana de Command History muestra todos los comandos y órdenes que se han introducido, permitiendo recuperarlos o bien arrastrándolos a la Command Window (no ejecuta), o bien haciendo doble click sobre ellos (ejecuta).

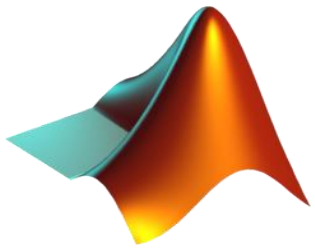


**Figura 14-2:** Logo característico de MATLAB
**Fuente:** (MATLAB, 2018)

Los comando o instrucciones en MATLAB según (Albesa, 2015) se escriben en minúscula y sus argumentos entre paréntesis. Su escritura es en inglés y se ejecutan en cuanto se presiona la tecla enter. Cuando haya dudas de los argumentos de alguna instrucción, bastará con escribir en la ventana de comandos "help nombredelcomando", por ejemplo: "help gcd" (y saldrá la ayuda de Matlab, en este caso para el cálculo del máximo común divisor). Para recuperar alguno de los comandos introducidos, hay dos formas según (Albesa, 2015), una desde la ventana de *Command History;* la otra, mediante teclas de desplazamiento. Las conocidas flechas "arriba" y "abajo" recuperan los comandos. Mientras que las de izquierda y derecha sirven para desplazarse dentro de la línea de edición para poder modificar las expresiones.

Para interrumpir el funcionamiento de una instrucción de MATLAB se pulsan las teclas Control y C a la vez y después enter. A veces, esta operación se deberá repetir. La forma de salir de Matlab

es o bien cerrar la ventana, o bien con los comandos “exit”, o “quit”. Y como siempre dándole al enter al final de cualquier instrucción para que así se ejecute. Si no se quiere que la ejecución de un comando salga por pantalla, aunque sí que sea ejecutado, bastará con poner después del comando un “;”. (Albesa, 2015)

### *2.3.2. Procesamiento de imágenes con MATLAB*

El término “Visión Artificial**”** engloba el proceso en el cual se replica electrónicamente el comportamiento del ojo humano. Dicho en otras palabras, es la capacidad de distinguir formas adquiriendo la información con la ayuda de sensores de visión que reciben la luz que incide de un objeto. Uno de estos sensores puede ser una cámara que cuenta con un agujero denominado ***pin-hole*** por el cual entra la luz y forma una imagen invertida del objeto capturado en escena. En la naturaleza el cerebro procesa la información proporcionada por los ojos, y en la **Visión por Computadora** el proceso de interpretación de datos se lo realiza con una computadora. (Dachs-Solutions, 2019)

MATLAB cuenta con una herramienta para el procesamiento de imágenes denominado *Image Processing Toolbox,* que permite segmentar imágenes y trabajar con las mismas para analizar sus características. Así también, cuenta con varias funciones creadas para capturar, binarizar, filtrar y visualizar imágenes. Con estas herramientas se puede encontrar contornos y distinguir formas aplicando cálculos matemáticos. Además, también permite detectar el color de una imagen y dibujar líneas sobrepuestas en la misma como se puede observar en la figura 2-15. (Procesamiento de imagenes, 2018)

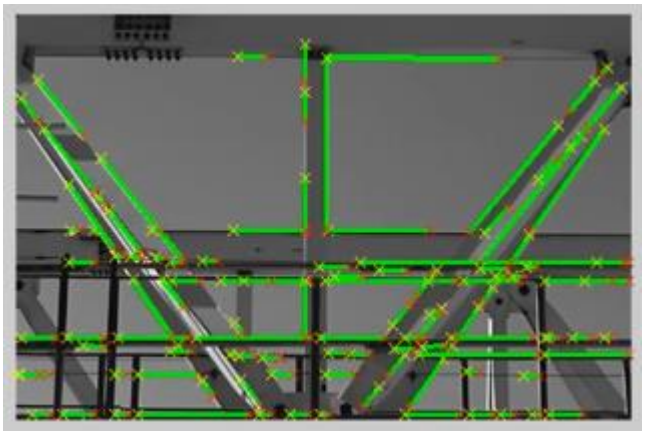


**Figura 15-2:** Ejemplo de procesamiento de imágenes en MATLAB
**Fuente:** (Procesamiento de imagenes, 2018)

En su obra (Arnaez Luis, 2015) habla sobre la importancia de esta herramienta matemática aduciendo que MATLAB (Matricial Laboratory o Laboratorio Matricial) es una combinación de un entorno amigable y un lenguaje con su propio código de programación. El elemento que hace de esta herramienta muy versátil es que el lenguaje de programación de MATLAB está diseñado para crear herramientas reutilizables propias y personalizadas en ficheros denominados "Scripts" (Archivos-M o M-Files). Así como cada uno puede generar sus M-Files y agruparlos en toolboxes procesamiento de señales, matemáticas y estadística, entre otras. La agrupación de estas herramientas en un directorio da origen al nombre de comerciales *toolboxes* debido al esfuerzo de investigadores líderes a escala mundial en diferentes campos de conocimiento.

En este trabajo se utiliza algunas técnicas de procesamiento de imágenes para capturar vídeo por medio de una cámara web y procesarla para binarizarla, filtrarla y reconocer el color rojo y verde de los objetos como se puede apreciar en la gráfica de la figura 16-2.


**Figura 16-2:** Procesamiento de imágenes con MATLAB en el presente trabajo
**Realizado por:** Bejarano, L. 2021

### *2.3.3. Creación de una GUI con MATLAB*

La denominada interfaz gráfica de usuario o como su nombre corto lo describe como "GUI" es una de las herramientas alternativas de MATLAB que se pueden utilizar para no crear todo el código que generará un proceso. Las GUI son utilizadas generalmente también para presentar al usuario una interfaz gráfica de control personalizada con la ayuda de menús, botones, barras, etc. (GUI de MATLAB, 2018)

Para crear una GUI, MATLAB proporciona al usuario un editor de diseño de GUIDE que permite crear un entorno gráfico seleccionando elementos desde un menú de opciones y arrastrándolos sobre la ventana de edición (Fig. 2-4). Las acciones mencionadas anteriormente, crean un archivo

de extensión. *fig*. Seguido de esto, se crea automáticamente un archivo de extensión *.m* en el cual se presenta el código generado por el GUIDE. Este código permite al usuario personalizar el comportamiento de la GUI con lenguaje de programación de MATLAB. La figura 2-17 muestra la ventana principal de este entorno de edición que se ejecuta al escribir la palabra reservada "guide*"* desde MATLAB con la ayuda del editor de comandos y seleccionando la opción *blanck GUI (Default)*. (GUI de MATLAB, 2018)

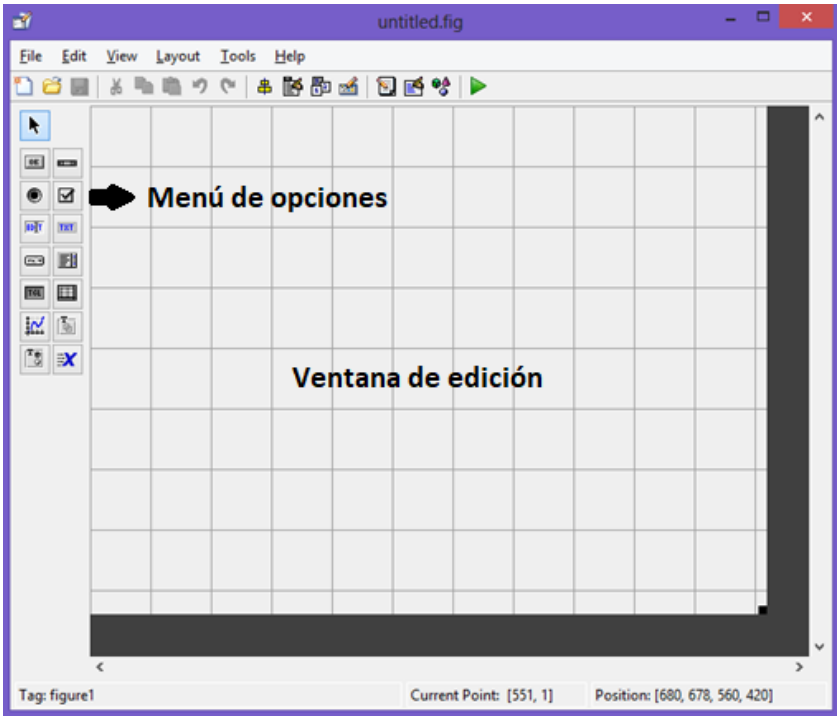



**Figura 17-2:** Ventana principal de edición de una GUI en MATLAB
**Realizado por:** Bejarano, L. 2021

## 2.4. Brazo Robótico xArm LewanSoul

A continuación, se investigó las especificaciones técnicas de la tarjeta controladora de servos que trae consigo el brazo robot seleccionado para este trabajo. Fue necesario revisar la comunicación serial entre MATLAB y la tarjeta controladora para enviar datos desde el software hacia el hardware.

### *2.4.1. Características principales*

Es un robot mecánico programable que cuenta con seis servos de alta duración que tienen indicadores RGB luminosos, pueden ser controlados mediante un bus serial inteligente LX-15D y una tarjeta electrónica propia de LewanSoul. Además, cuenta con una pinza en su extremo final que puede ser utilizada para sujetar cosas. La tarjeta controladora de servos puede conectarse a la

computadora mediante un cable de datos serial o mediante conexión bluetooth. La estructura del brazo es metálica y de alta durabilidad, ver figura 2-18. (LewanSoul, 2019)

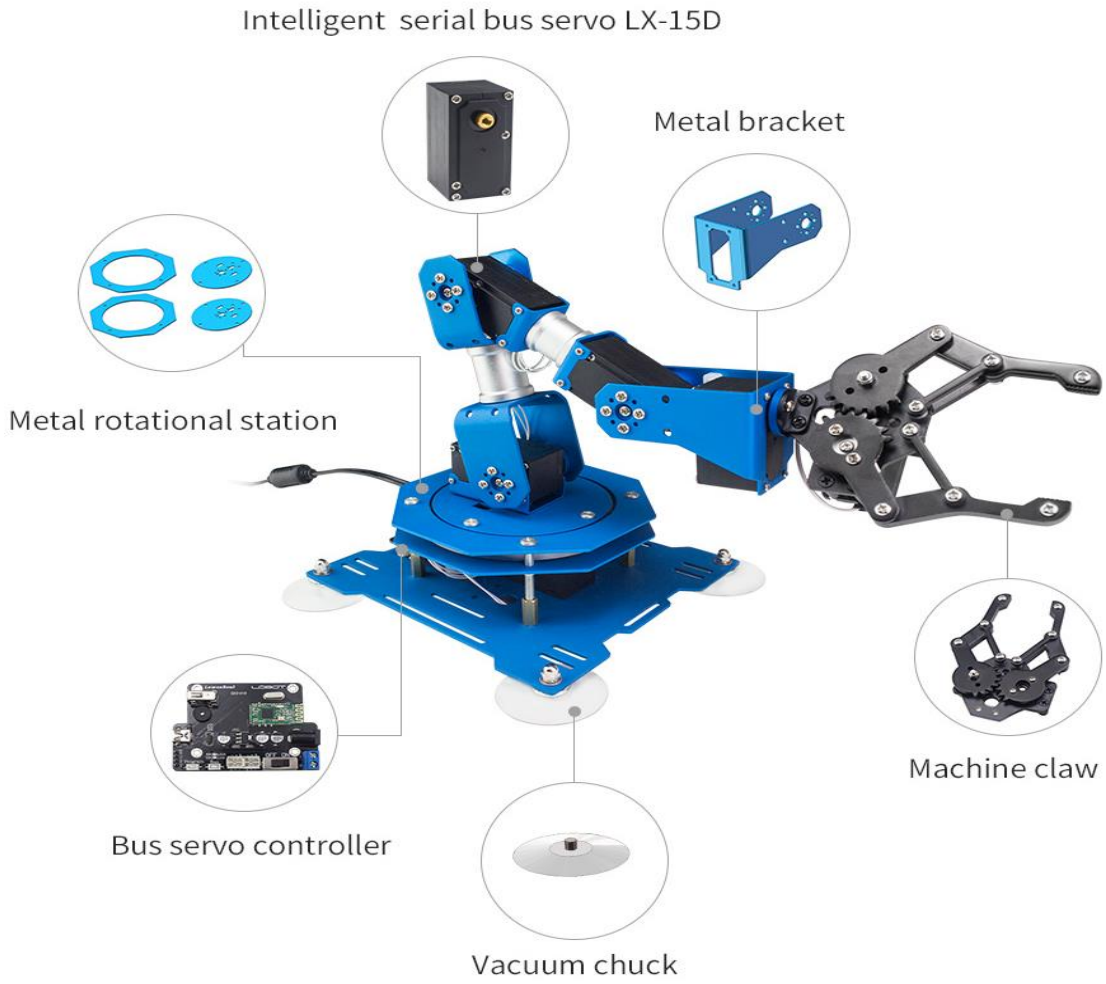



**Figura 18-2:** Brazo robótico xArm LewanSoul
**Fuente:** (LewanSoul, 2019)

Para probar el funcionamiento después de ensamblar el brazo robótico por primera vez, el producto cuenta con su propio software de control, ver figura 2-19. Con la interfaz gráfica mencionada se puede controlar la posición de los servomoteres en rangos de 0 a 1000, donde 0 significa cero grados del servomotor y 1000 180 grados del servomotor, esto de acuerdo a donde se considere cero y ciento ochenta grados en el servo motor.

### *2.4.2. Comunicación serial con MATLAB*

La comunicación serial o comunicación secuencial es un método que se utiliza para enviar datos entre un emisor y un receptor en secuencia de bits. Esta comunicación utiliza el protocolo estándar RS-232 para intercambiar la información. (Comunicación serie, 2015)

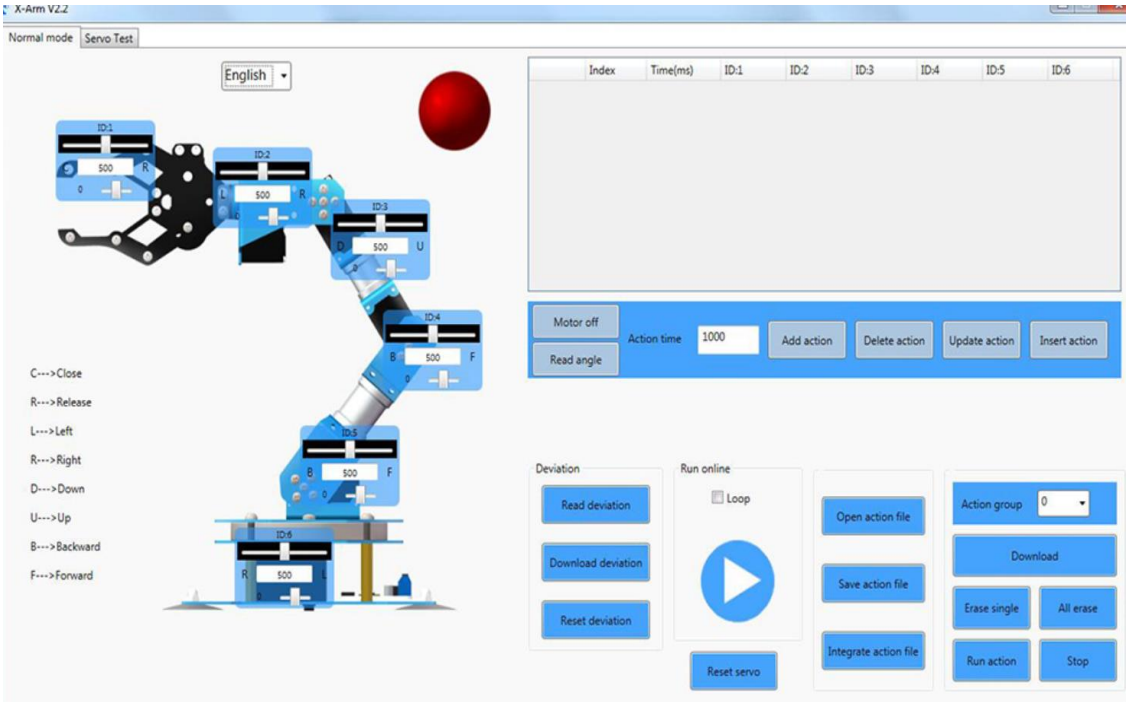



**Figura 19-2:** Interfaz gráfica del brazo robótico xArm LewanSoul
**Fuente:** (LewanSoul, 2019)

MATLAB permite utilizar esta interfaz de comunicación al definir con líneas de código un puerto de envío y una velocidad de transmisión. El puerto de envío depende en varios casos del punto donde se encuentra conectado el dispositivo al cual se desea enviar información. En este trabajo el puerto de envío depende de donde se conecte la tarjeta electrónica LewanSoul. (Use Serial Communications, 2018)

De acuerdo con el manual de usuario de LewanSoul se puede enviar comandos que permiten comandar la posición, así como la velocidad del robot mediante el accionamiento de sus servomotores. Estas órdenes se emiten por medio de la trama que se detalla a continuación:

**Tabla 2-2:** Trama para enviar órdenes a los servomotores

| Header | Data Length | Command | Parameter |
|---|---|---|---|
| 0x55 0x55 | Length | Cmd | Prm 1...Prm N |

**Fuente:** (LewanSoul, 2019)

En la tabla 2-2 se observa los distintos elementos que presenta la trama, como son el encabezado (Header) que contiene el código que señala el inicio de la trama, la longitd de los datos (Data Length) que indica la longitud de bits que se envía, el comando (Command) la orden que se quiere ejecutar en el servo motor como mover, y los parámetros que enumeran a continuación:

- Parámetro 1: Especifica el número de servo a controlar
- Parámetro 2: El mínimo valor de tiempo expresado en 8 bits
- Parámetro 3: El máximo valor de tiempo expresado en 8 bits
- Parámetro 4: Identificador del servo
- Parámetro 5: Máximo valor del ángulo de posición expresado en 8 bits
- Parámetro 6: Mínimo valor del ángulo de posición expresado en 8 bits

**Tabla 3-2:** Ejemplo de trama para mover el servo motor 1

| Header | Length | Command | Parameter |
|---|---|---|---|
| 0x55 0x55 | 0x08 | 0x03 | 0x01 0xE8 0x03 0x01 0xD0 0x07 |

**Fuente:** (LewanSoul, 2019)

Por ejemplo, para enviar una orden que mueve el servo motor a una posición 2000 con un tiempo de 1 segundo se envió desde MATLAB la trama de datos que se puede ver en la figura 2-19 donde se observa los valores enviados de acuerdo a la tabla 2.3. Para enviar la trama de datos antes expuesta como ejemplo desde MATLAB se puede utilizar el comando *fwrite()* que envía código hexadecimal convertido desde decimal hacia el puerto serial. En la figura 2-20 se puede observar un ejemplo de cómo enviar estos datos por comunicación serial.

```
fwrite(Pser,[85 85 08 03 01 16 09 01 144 01],'uint8')
```

**Figura 20-2:** Línea de código necesaria para enviar la trama de datos hacia el servo motor
**Fuente:** (LewanSoul, 2019)

### *2.4.3. Servomotor y tarjeta controladora de LewanSoul*

Se denomina servomotor al conjunto de engranes y motor de corriente continua que permiten ubicar el eje de rotación de este en un determinado ángulo de posición y mantenerse estable en dicho punto. Dicho mecanismo puede ser controlado en base a su velocidad y posición según (Servomotor, 2011). Ver figura 2-21.

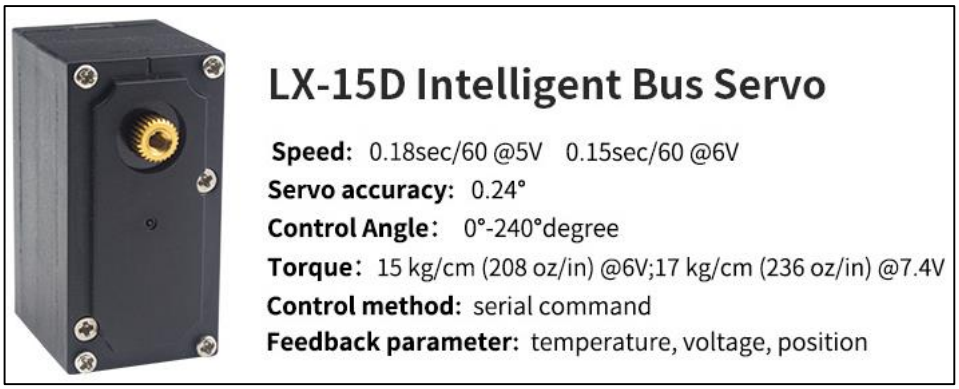



**Figura 21-2:** Servomotor del brazo robot de LewanSoul
**Fuente:** (LewanSoul, 2019).

Un servomotor por lo general cuenta con tres cables, uno para alimentación positiva, uno para alimentación negativa y uno para datos que recibe la señal de modulación de ancho de pulso (PWM). La ecuación que modela matemáticamente un servomotor según (Fernandez, 2008) se detalla a continuación en (2.3).

$$\frac{k_M}{\tau_M\, s + 1} \tag{2.3}$$

Donde,

$k_M$ es la constante de ganancia del motor en *RPM*.
$\tau_M$ es la constante de tiempo del motor en *ms*.

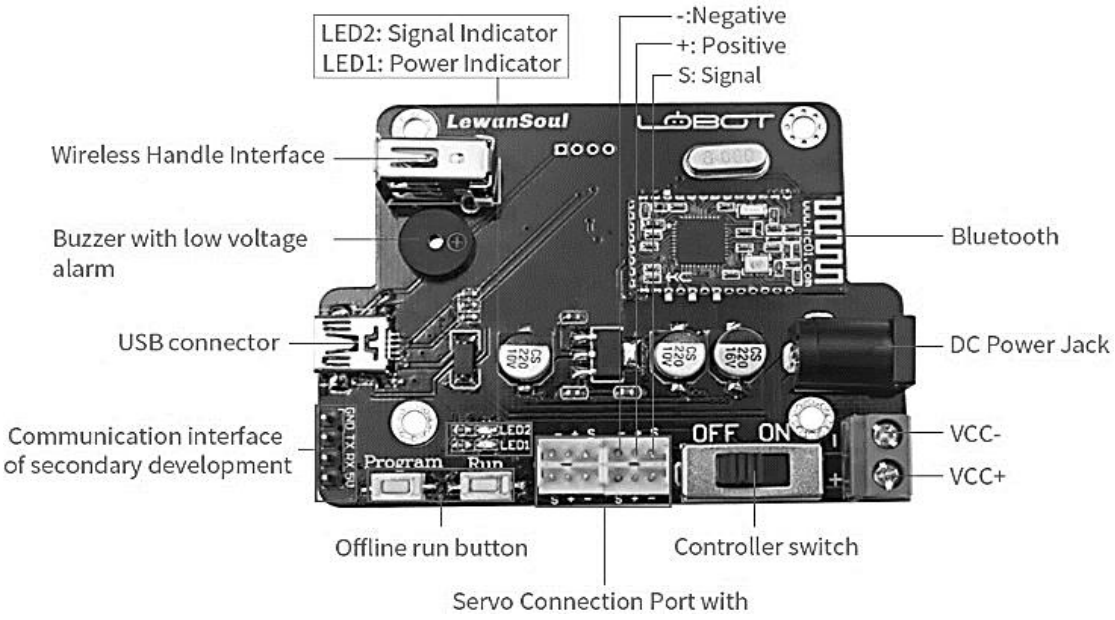



**Figura 22-2:** Tarjeta controladora de servos
**Fuente:** (LewanSoul, 2019)

Para controlar un servomotor se necesita una señal PWM, la misma que se puede ejecutar desde un micro controlador o un computador. Por otro lado, se debe mencionar que para manejar más de un servomotor se requiere una tarjeta controladora de servos. En la figura 2-22 se muestra un

ejemplo de un controlador de servos de LewanSoul que utiliza comunicación serial y es capaz de manejar 6 servos seriales a la vez.

## 2.5. Cámara web Logitech C930 HD

La cámara "Logitech C930 HD" es un tipo de "webcam" que está enfocada para ser utilizada en videoconferencias que proporcionen a las personas la sensación de charlar en una reunión real debido a su gran nitidez y definición en las imágenes y sonidos. La cámara web "Logitech C930 HD" es popular y se ha ganado su prestigio por su gran tecnología que proporciona vídeo y sonido prácticamente nítidos sin importar el entorno en el que la ocupe, es decir, no depende de la luz y ofrece buenas prestaciones en las imágenes, aunque el ambiente tenga poca luz. La resolución del vídeo cuenta con 1080 pixeles y con un vídeo en definición H.264 que permite obtener un amplio entorno de visión alrededor de un ángulo de 90 grados. La forma física de la cámara "Logitech C930 HD" se puede apreciar en la figura 2-23. (Logitech, 2019)


**Figura 23-2:** Cámara logitech C930 HD
**Fuente:** (Logitech, 2019)

La cámara web "Logitech C930 HD" se destaca y menciona entre sus principales características su campo visual de 90 grados, su capacidad de cobertura panorámica, la inclinación que permite su arquitectura de construcción y el zoom digital que puede ofrecer hasta una escala de 4x que le provee a la cámara la capacidad de encuadrar una imagen del entorno de manera personalizada. Además, gracias a su tecnología avanzada denominada "RightLight 2" y su "lente de precisión" es capaz de brindar un vídeo nítido aún en aquellas condiciones donde la luz escasea. (Logitech, 2019)

La cámara "Logitech C930 HD" es compatible con "Skype" y presenta una certificación para avalar este hecho otorgado por "Business" y "Cisco Jabber". Además, también presenta grandes mejorías para adaptarse a las tecnologías "Broadsoft", "BlueJeans", "Vidyo", "LifeSize Cloud" y "Zoom". (Logitech, 2019)

La cámara “Logitech C930 HD” es la mejor opción a la hora de realizar reuniones mediante vídeo y conexión a internet ya que gracias a su vídeo nítido los usuarios han calificado con altos puntajes el desempeño de ésta aún en la presencia de conexiones con poco ancho de banda. Esta cámara se destaca también por su tecnología que ofrece grandes prestaciones para utilizar “Skype”, además, ya que cuenta con H.264 que permite que la codificación de vídeo sea escalable. Cuenta también con codificación UVC 1.5 que le proporciona independencia de las condiciones y prestaciones de la computadora y la conexión de internet. (Logitech, 2019)

# CAPÍTULO III

## 3. METODOLOGÍA

### 3.1. Sistema desarrollado

Debido al problema generado por el gasto económico que incurre en la contratación de personal para clasificar objetos en un proceso repetitivo en el campo industrial la presente investigación se centró en desarrollar un brazo robótico que clasifique objetos reconociendo colores mediante técnicas de visión artificial. Con este proyecto se busca además impulsar al sector productivo de la ciudad de Riobamba ya que se pretende automatizar los procesos de clasificación y evitar los costos que producen la contratación de mano de obra.

Para desarrollar el presente trabajo y generar el algoritmo de reconocimiento de colores se utilizó el "método de espiral*"* (metodología utilizada para el desarrollo de software) desarrollado por primera vez por Barry Boehm en 1986 según (Desarrollo en espiral, 2018). El método de la espiral consiste en *determinar objetivos, analizar riesgos, desarrollar y probar* el algoritmo, y finalmente *evaluar* los resultados obtenidos que de ser los esperados darían por concluido el proyecto, de lo contrario se toman en cuenta las correcciones detalladas y se vuelve a ejecutar le método una vez más desde el principio.

La arquitectura de conexión del sistema propuesto en el presente trabajo se puede ver en la figura 3-1 donde se observa que está basado principalmente en la conexión entre MATLAB y el controlador de servos de LewanSoul. Para el reconocimiento del color del objeto se utiliza una cámara web Logitech C930 HD y una interfaz gráfica en MATLAB con técnicas de visión artificial.

El objeto puede ser detectado en base a dos colores diferentes (rojo o verde) utilizando la cámara web. Luego de reconocer el color del objeto se guía el brazo robot mediante un algoritmo de MATLAB para llevar el objeto desde un punto donde se encuentra al alcance del manipulador hacia un punto donde se almacena. Por otro lado, para mover el brazo se utilizan 6 servomotores los cuales proporcionan movimiento al mismo a través de un controlador de servos de LewanSoul.

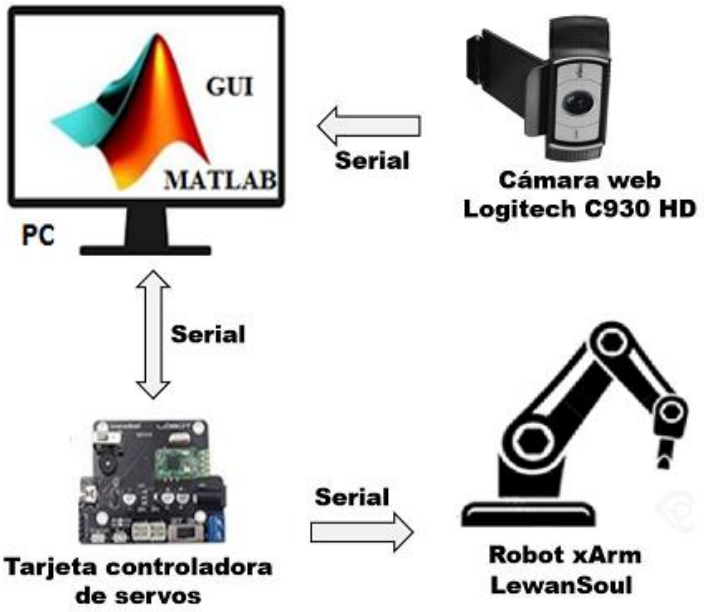



**Figura 1-3:** Arquitectura del sistema propuesto en este trabajo
**Realizado por:** Bejarano, L. 2021

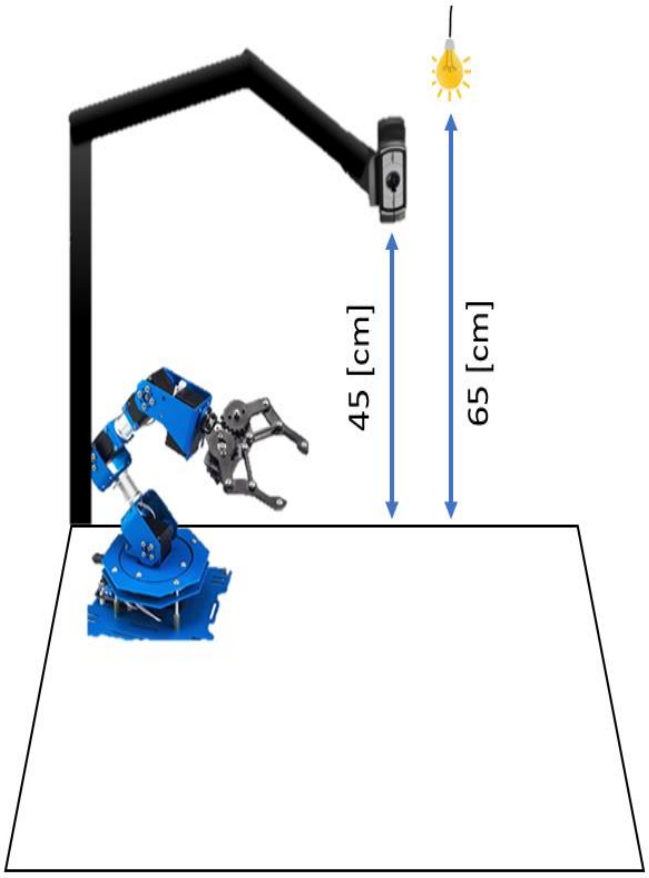



**Figura 2-3:** Arquitectura del sistema propuesto en este trabajo
**Realizado por:** Bejarano, A. 2021

## 3.2. Desarrollo matemático del brazo robot

### *3.2.1. Modelado cinemático*

El brazo robótico seleccionado para este trabajo es el robot metálico LewanSoul (LewanSoul, 2019), el cual está compuesto por 6 servomotores que le brindan movilidad a sus tres eslabones, su base y su pinza de agarre, ver figura 3-3

En base a las características de construcción del brazo robótico LewanSoul se ha modelado la cinemática de éste utilizando el principio propuesto por "Denavit-Hartenberg" (parámetros D-H). En la representación de la figura 3-4 se muestra en detalle la posición encontrada de cada uno de los sistemas de referencia móviles *(S1 a S5)* y el sistema de referencia fijo (*S0).*

**Figura 3-3:** Brazo robótico LewanSoul
**Fuente:** (LewanSoul, 2019)

Cabe mencionar que éste modelado es necesario para encontrar el punto y la orientación final del extremo del robot (S6) con respecto a los valores adoptados en cada una de las articulaciones. De acuerdo con la configuración del modelo cinemático se han encontrado los valores que se detallan en la tabla de parámetros D-H (Tabla 3-1).

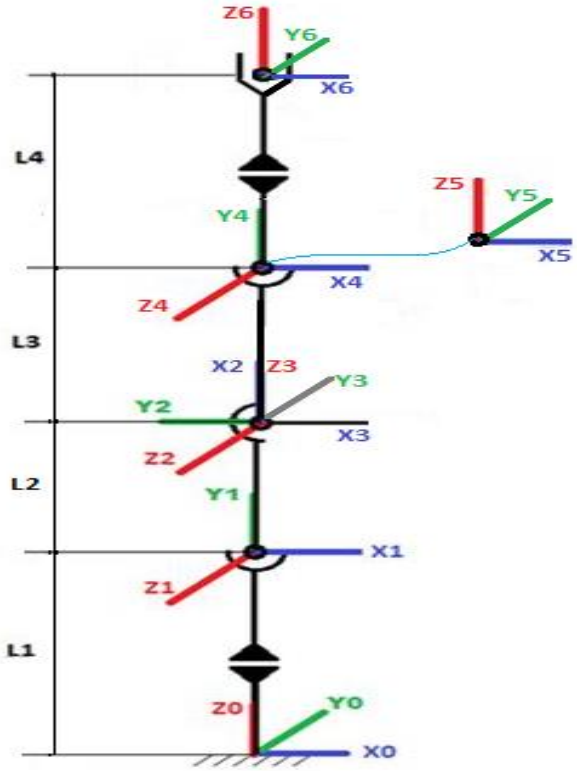



**Figura 4-3:** Modelado cinemático del robot de 6 GDL utilizando los criterios de Denavit-Hartenberg (D-H)
**Realizado por:** Bejarano, L. 2021

En la figura 3-4 se puede observar que el brazo robótico ha sido representado teóricamente con 6 articulaciones que representan los grados de libertad que le brindan los servomotores. Es decir que de acuerdo con la gráfica todas las articulaciones son rotacionales y no cuenta con articulaciones de otro tipo. De acuerdo con esto, se ha establecido la posición inicial del brazo robótico en base a los ángulos y longitudes proporcionadas por el mismo para obtener los parámetros D-H que muestran a continuación.

**Tabla 1-3:** Parámetros de D-H del brazo robot de 6 GDL

| ***# Articulaciones*** | ***θ*** | ***d*** | ***a*** | A |
|---|---|---|---|---|
| *1* | $q_1$ | $L1 = 6cm$ | 0 | 90º |
| *2* | $q_2 + 90$ | 0 | $L2 = 10\ cm$ | 0 |
| *3* | $q_3$-90 | 0 | 0 | -90 |
| *4* | $q_4$ | $L3 = 9\ cm$ | 0 | 90 |
| *5* | $q_5$ | 0 | 0 | -90º |
| *6* | $q_6$ | $L4 = 12\ cm$ | 0 | 0 |

**Realizado por:** Bejarano, L. 2021

### *3.2.2. Cinemática directa del brazo robot*

De acuerdo con la tabla 3-1 de los parámetros D-H el brazo robot contaría con las siguientes matrices de transformación homogénea: La matriz T1 colocada como 3.1 es acorde con la primera

fila de la tabla 3-1 correspondiente a la primera articulación móvil, la matriz T2 mostrada como 3.2 es correspondiente con la segunda fila y sucesivamente hasta llegar a la sexta y última fila de la última articulación mostrada como 3.6.

$$ {}^{0}T_{1} = \begin{bmatrix} Cq_1 & 0 & Sq_1 & 0 \\ Sq_1 & 0 & -Cq_1 & 0 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

$$ {}^{1}T_{2} = \begin{bmatrix} C(q_2+90) & -S(q_2+90) & 0 & 10C(q_2+90) \\ S(q_2+90) & C(q_2+90) & 0 & 10S(q_2+90) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

$$ {}^{2}T_{3} = \begin{bmatrix} Cq_3 & 0 & -Sq_3 & 0 \\ Sq_3 & 0 & Cq_3 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3} $$

$$ {}^{3}T_{4} = \begin{bmatrix} Cq_4 & 0 & Sq_4 & 0 \\ Sq_4 & 0 & -Cq_4 & 0 \\ 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4} $$

$$ {}^{4}T_{5} = \begin{bmatrix} Cq_5 & 0 & -Sq_5 & 0 \\ Sq_5 & 0 & Cq_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5} $$

$$ {}^{5}T_{6} = \begin{bmatrix} Cq_6 & -Sq_6 & 0 & 0 \\ Sq_6 & Cq_6 & 0 & 0 \\ 0 & 0 & 1 & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6} $$

Una vez que se ha obtenido todas las matrices de transformación homogénea de cada articulación se encontró la matriz total al hacer la multiplicación sucesiva. El resultado en la matriz muestra la posición y orientación final que toma el centro del efector final, en este caso la pinza. La multiplicación antes explicada se obtuvo con la ecuación 3.7 y con la ayuda de MATLAB.

$$ {}^{0}T_{6} = {}^{0}T_{1}\ {}^{1}T_{2}\ {}^{2}T_{3}\ {}^{3}T_{4}\ {}^{4}T_{5}\ {}^{5}T_{6} \tag{3.7} $$

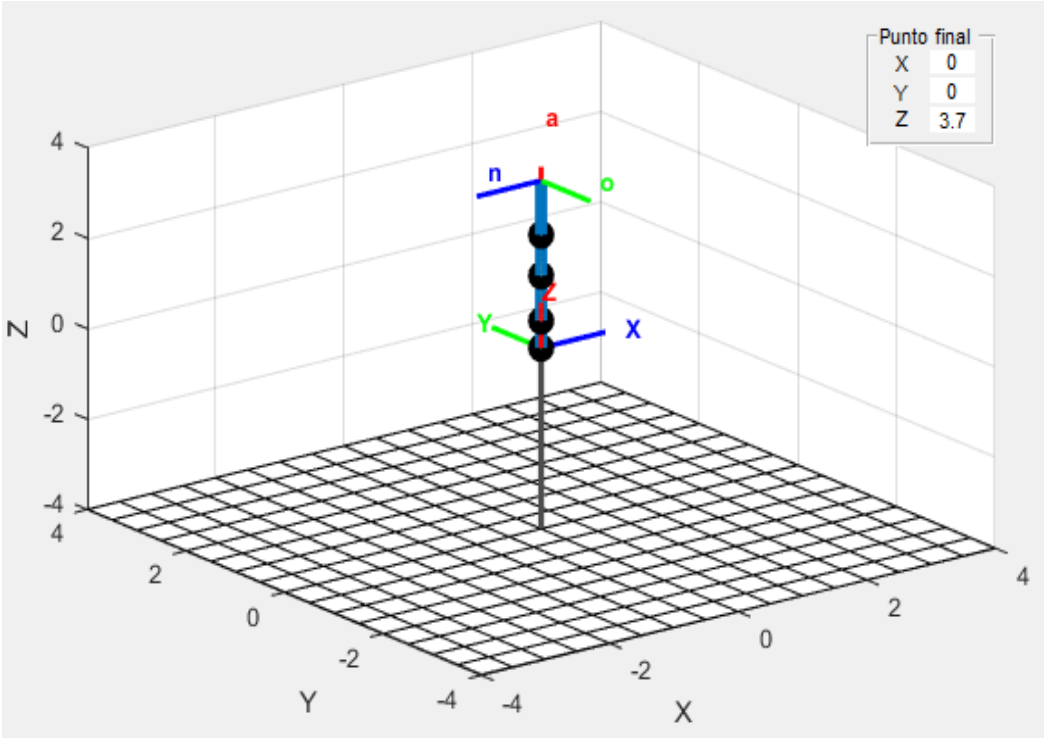



**Figura 5-3:** Posición cero del brazo robot modelado con parámetros D-H y representado en MATLAB

**Realizado por:** Bejarano, L. 2021

En la figura 3-5 se puede ver la posición final de la pinza de acuerdo a la multiplicación sucesiva de las matrices 3.1 a 3.6 cuando q1 a q6 son cero (el brazo se encuentra en la posición de reposo) y el resto de variables toman los valores presentados en la tabla 3-1 que contienen los parámetros D-H. Cada una de las matrices de transformación homogéneas de las articulaciones con los valores antes mencionados que colocan el brazo en la posición de reposo se muestra en 3.8 a 3.14, y la matriz resultante después de la multiplicación sucesiva que contiene el punto y la orientación final de la pinza se muestra en 3.14.

$$ {}^{0}T_{1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8} $$

$$ {}^{1}T_{2} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 10 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9} $$

$$ {}^{2}T_{3} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10} $$

$$ {}^{3}T_{4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11} $$

$$ {}^{4}T_{5} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.12) $$

$$ {}^{5}T_{6} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 12 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.13) $$

$$ {}^{0}T_{6} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 37 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.14) $$

En la matriz 3.14 se obtiene los datos de la posición final de la pinza con coordenadas en (x, y, z) respecto de la base del robot, en este caso las coordenadas son (0,0,37). Por otro lado, la orientación está dada por la submatriz [d11 d12 d13; d21 d22 d23; d31 d32 d33], donde la primera columna es "n" la segunda columna es "o" y la tercera columna es "a". En la figura 3.6 se puede ver la ubicación del eje "a" con respecto al eje "z", y se interpreta en la matriz que el eje "a" está ubicado totalmente en el eje "z", lo cual, de acuerdo con la figura 3-5 es correcto.

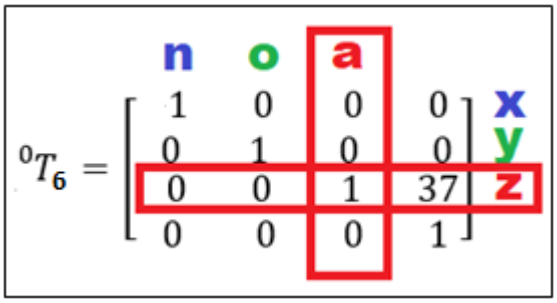



**Figura 6-3:** Matriz de orientación de la pinza
**Realizado por:** Bejarano, L. 2021

La matriz mostrada en la figura 3.6 contiene información de la posición y orientación de la pinza en el extremo del brazo robot. Esta información comúnmente es utilizada para resolver problemas de cinemática directa e inversa. Con los datos proporcionados se puede interpretar la orientación y la posición del extremo final con respecto al eje de coordenada fijo S0 que se mencionó en el párrafo anterior.

### *3.2.3. Cinemática Inversa del brazo robot*

Para resolver la cinemática inversa del brazo robot se ha utilizado el método iterativo de solución al problema de cinemática inversa, el cual utiliza los principios de los métodos numéricos para minimizar la diferencia entre la configuración inicial del robot F(q) y la posición deseada sobre

la cual se requiere determinar el vector de coordenadas generalizadas, esta técnica fue mencionada en la sección 2.1.3

De acuerdo a lo expresado anteriormente para solucionar el problema de cinemática inversa en este trabajo se guardó los puntos a los que se dese llegar junto con los valores de los ángulos de las articulaciones, y se envía mediante comunicación serial el comando adecuado para ejecutar dicha acción en el brazo robótico de LewanSoul. Para posicionar el brazo se utilizó la cinemática directa para mover el mismo mediante los controles deslizantes diseñados en MATLAB hasta los puntos de reposo, recogida y depósito respectivamente. Una vez que se ubicó el brazo en dichos puntos se guardó los valores de los ángulos que debe alcanzar cada servo para llegar a esa posición. De esta manera con la ayuda de un algoritmo de MATLAB se ordena al brazo moverse de manera adecuada para alcanzar cada uno de los puntos de reposo, recogida y depósito.

Es así que se creó un algoritmo con una secuencia de líneas que permiten desplazar la pinza del robot desde el punto de reposo hacia el punto donde recoge el objeto y luego hacia el punto donde deposita el mismo. El reconocimiento del color activa esta secuencia y decide hacía que punto dirigirse para depositar el objeto, como se observa en la figura 3-7. Una vez que la cámara se encuentra activada y la comunicación serial está establecida este proceso es factible.

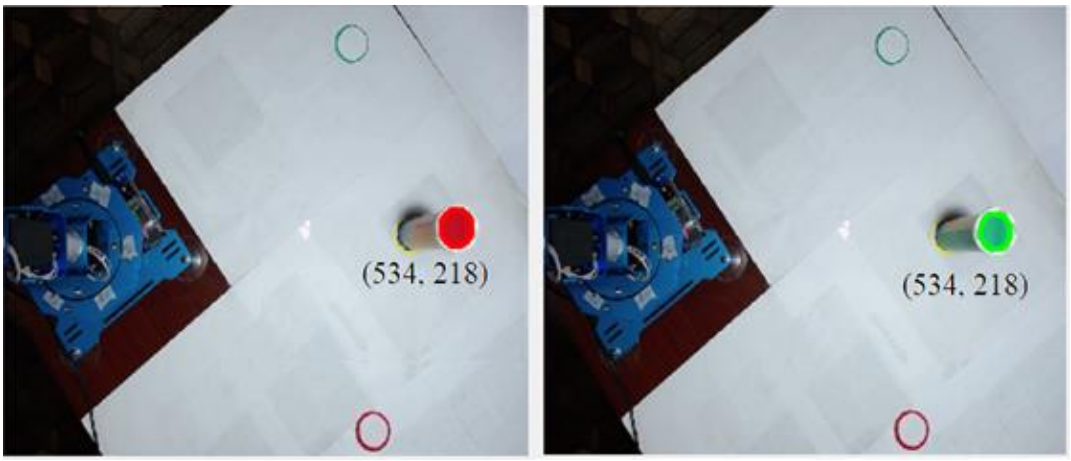



**Figura 7-3:** Reconocimiento de color para activar la secuencia de movimiento.
**Realizado por:** Bejarano, L. 2021

Para mover el brazo robot de un punto a otro se ha utilizado la técnica de trayectorias coordinadas explicada en la sección 2.1.4 Este método se centra en mover todos los servomotores al mismo tiempo, pero utilizando el tiempo del servo que tiene mayor recorrido denominado tiempo mayor (TM). Para esto, primero se ha encontrado el ángulo mayor de la articulación que debe recorrer el tramo más grande. El resultado del movimiento de las articulaciones antes descritas se observará de forma continua.

Para mover el brazo robot se ha utilizado el guiado manual en base a los controles deslizantes de la GUI de MATLAB. Para realizar esta acción se utiliza la cinemática directa calculada en la sección 3.2.2 La figura 3-10 muestra cada uno de los controles de la interfaz gráfica que son necesarios para mover una por una las articulaciones mediante un *Slider*.

## 3.3. Desarrollo y adecuación para el control del brazo robot

Para la construcción de la GUI se analizaron los requerimientos mínimos de la misma. Estos requisitos fueron tener un espacio para mostrar la simulación del movimiento del robot controlado por medio de barras deslizantes, contar con un espacio para visualizar el vídeo de la cámara, contar con un pulsador *start* y *stop* para ejecutar el proceso y detenerlo respectivamente, Esta interfaz gráfica se detalla en la figura 3-10.

En la figura 3-10 se puede observar el pulsador diseñado para arrancar el sistema pintado de color rojo, una vez que se presiona este pulsador el programa muestra un mensaje notificando que se está conectando la cámara web, ver figura 3-8. Después que se conecta la cámara web, se muestra un segundo mensaje que notifica que se está conectando de forma serial con el brazo robótico LewanSoul, ver figura 3-9. Una vez que se han conectado la cámara web y el brazo robótico mediante conexión serial el pulsador se torna de color verde y se presenta la pantalla como se muestra en la figura 3-11.

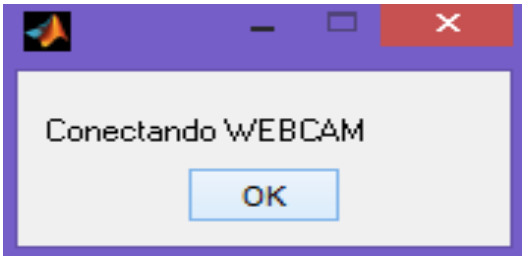



**Figura 8-3:** Cuadro de mensaje que advierte de la conexión de la cámara web.
**Realizado por:** Bejarano, L. 2021

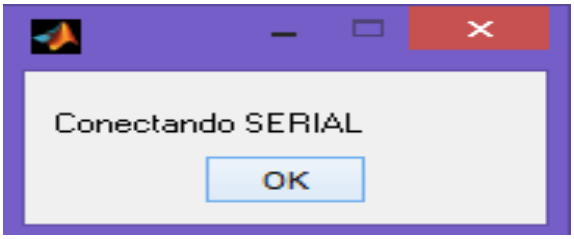



**Figura 9-3:** Cuadro de mensaje que advierte de la conexión serial con el brazo robótico.
**Realizado por:** Bejarano, L. 2021

Para mover el brazo robot se ha utilizado el guiado manual en base a los controles deslizantes de la GUI de MATLAB. Para realizar esta acción se utiliza la cinemática directa calculada en la sección 3.2.2. La figura 3-10 muestra cada uno de los controles de la interfaz gráfica que son necesarios para mover una por una las articulaciones mediante un *Slider*.

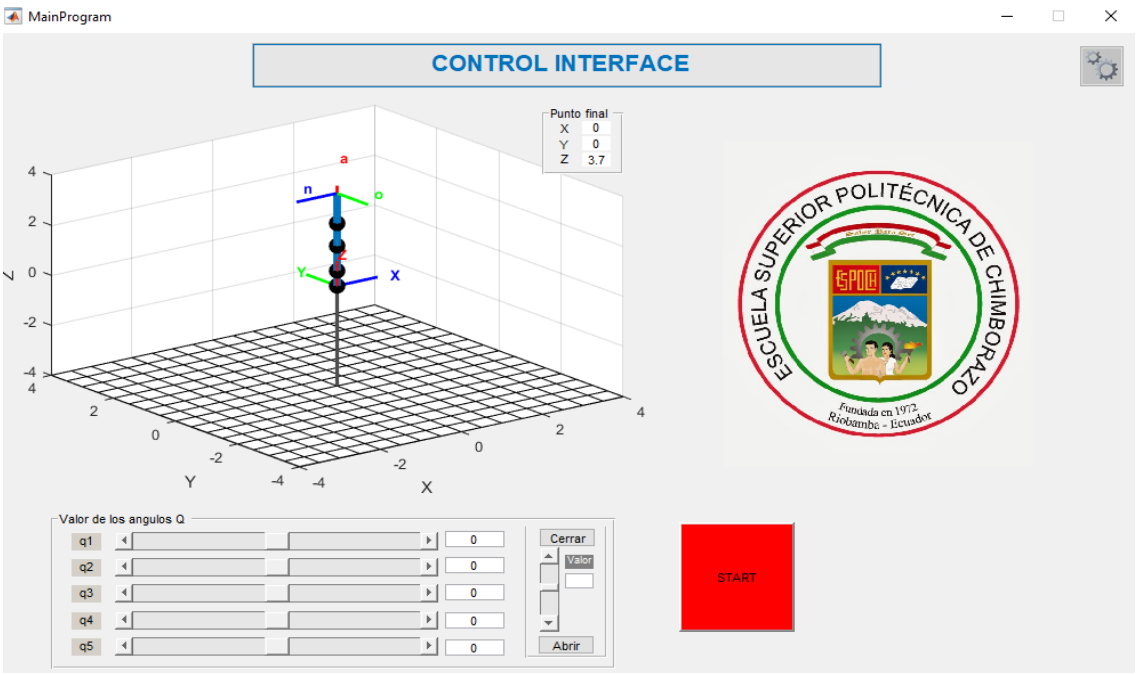



**Figura 10-3:** Interfaz gráfica creada en MATLAB para el monitoreo y control del brazo robot.
**Realizado por:** Bejarano, L. 2021

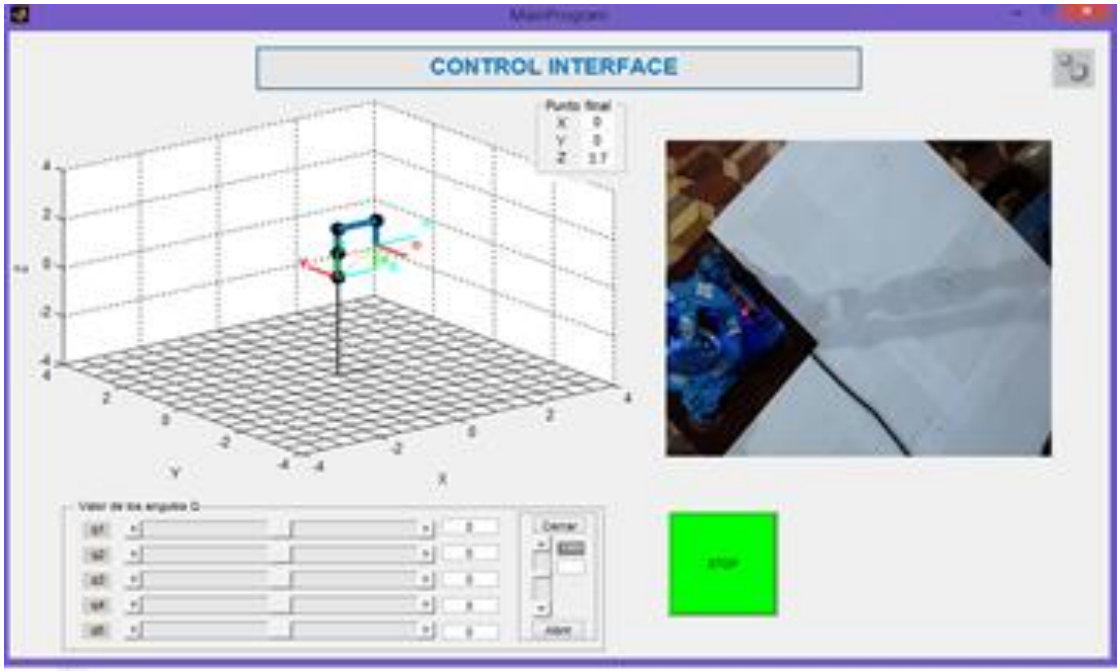



**Figura 11-3:** Interfaz gráfica creada en MATLAB para el monitoreo y control del brazo robot en estado activo.
**Realizado por:** Bejarano, L. 2021

# CAPÍTULO IV

## 4. RESULTADOS Y DISCUSIÓN

En este apartado se discuten los resultados obtenidos al realizar pruebas de manipulación de objetos de color rojo y verde con el brazo robot.

### 4.1. Pruebas de reconocimiento de color rojo y verde

En las pruebas de reconocimiento de color se encontró que el algoritmo desarrollado en MATLAB funciona correctamente. Debido a la resolución de la imagen de la cámara web Logitech las imágenes capturadas son claras y no presentan problemas de reconocimiento por iluminación. En la figura 4-1, 4-2, 4-3 y 4-4 se observa las pruebas realizadas con el reconocimiento del color.

En la figura 4-1A se presenta el reconocimiento de color rojo cuando la figura se encuentra sobre el punto de recogida. En la figura 4-1B se presenta el reconocimiento de color verde cuando la figura se encuentra sobre el punto de recogida. En la figura 4-2 se observa que el algoritmo no reconoce otro color que no sea rojo y verde. En la figura 4-3 se presenta la prueba en la que se colocó dos objetos al mismo tiempo fuera del punto de recogida, y debido a eso no se reconoce ninguno de los dos objetos, lo mismo pasa en la figura 4-4 donde no se reconoce el color del objeto porque están fuera del punto de recogida, evitando de esa forma que el brazo se active e intente clasificar el objeto.


**Figura 1-4:** Pruebas realizadas con el algoritmo de reconocimiento de color.
**Realizado por:** Bejarano, L. 2021

**Figura 2-4:** Pruebas realizadas con otro color diferente al rojo o verde. En este caso no se detecta ningún color.
**Realizado por:** Bejarano, L. 2021

**Figura 3-4:** Pruebas realizadas con el algoritmo de reconocimiento de color con los colores rojo y verde al mismo tiempo.
**Realizado por:** Bejarano, L. 2021

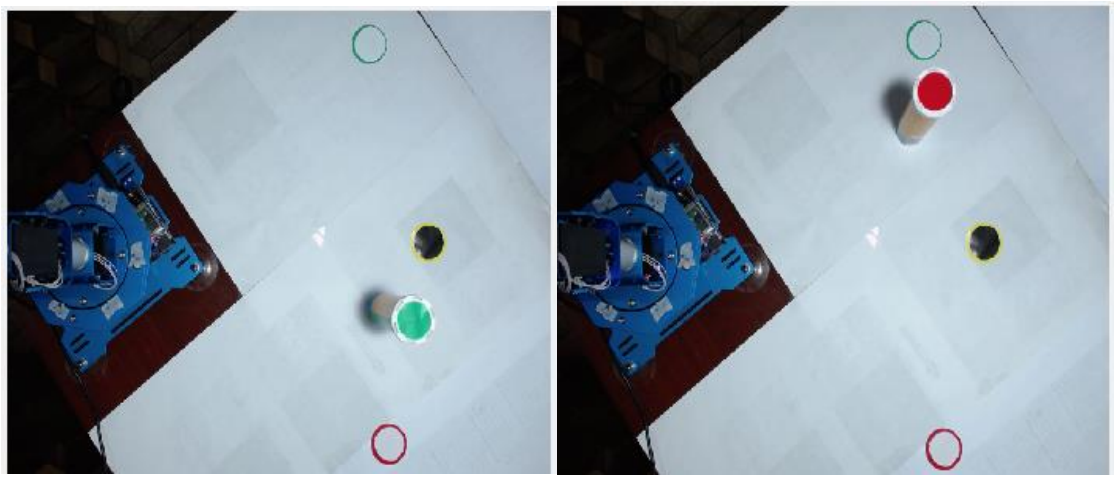


**Figura 4-4:** Pruebas realizadas con el algoritmo de reconocimiento de color con el objeto fuera del punto de recogida.
**Realizado por:** Bejarano, L. 2021

Para realizar estas pruebas se utilizó un proceso estadístico con muestreo aleatorio simple que se define en la ecuación 4.1. Para encontrar la población se midió el tiempo que tarda el brazo robot en clasificar un objeto de acuerdo al diseño de la mesa de trabajo de este proyecto. El tiempo medido fue 24 segundos el cual en hora arroja una cantidad de 150 veces que el robot puede clasificar un objeto. Con estos datos se obtuvo la población de 150 repeticiones del robot en una hora y se aplicó la ecuación 4.1 para encontrar la muestra cómo se presenta en el siguiente apartado.

$$n_o = \frac{Z^2 PQ}{e^2} \tag{4.1}$$

Donde:

**Z:** es el valor constante obtenido mediante niveles de confianza que para este caso es 1.96

**P:** es la probabilidad con la que se presente el fenómeno que para este caso es 50%

**Q:** es también la probabilidad con la que se presente el fenómeno que para este caso es 50%

***e*:** es el margen de error permitido que para este caso es de 3%

Con la ecuación 4.1 se determinó que el tamaño de la muestra es

$$n_o = \frac{Z^2 PQ}{e^2} = \frac{(1.96^2)(0.5)(0.5)}{(0.03)^2} = 1067$$

Como se sabe la población se puedo hacer una corrección al tamaño de la muestra anterior.

$$nc = \frac{n}{1 + \frac{n-1}{N}}$$

$$nc = \frac{1067}{1 + \frac{1067 - 1}{150}} = 131.62 \cong 132$$

Con la muestra definida se realizaron 132 repeticiones en la clasificación de objetos de color rojo y 132 repeticiones en la clasificación de objetos de color verde. En la figura 4-5A se presenta una de las ocasiones en que el brazo clasificó objetos de color rojo, en la figura 4-5B se presenta una de las ocasiones en que el brazo clasificó objetos de color verde.

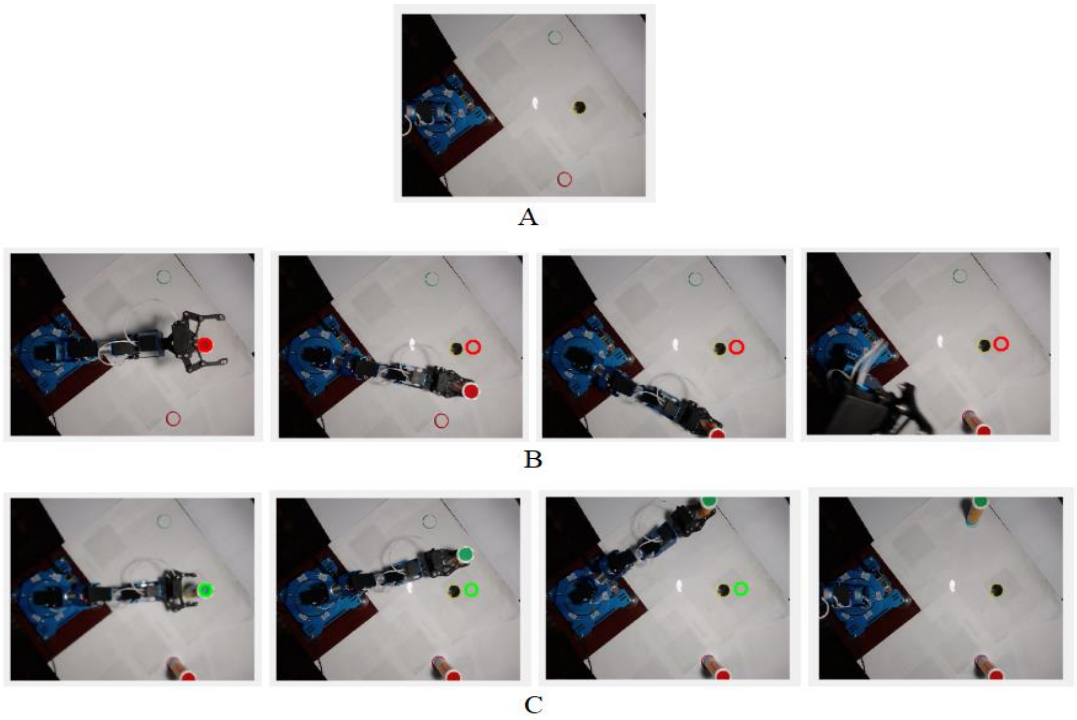



**Figura 5-4:** Pruebas realizadas en la clasificación de objetos de color rojo y verde.
A Imagen sin objetos, B Clasificación de color rojo, C Clasificación de color verde

**Realizado por:** Bejarano, L. 2021

**Tabla 1-4:** Tabla de resultados de aciertos en la clasificación de objetos de acuerdo con su color

| ***Pruebas realizadas*** | ***132*** |
|---|---|
| *Número de aciertos* | 132 |
| *Número de desaciertos* | *0* |

**Realizado por:** Bejarano, L. 2021

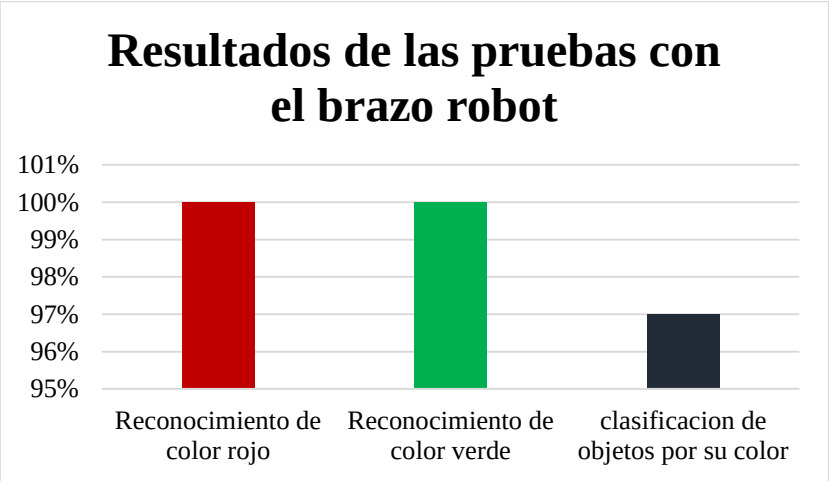



**Gráfico 1-4:** Resultado de clasificación y reconocimiento de colores.

**Realizado por:** Bejarano, L. 2021

Se evaluó la respuesta de una secuencia repetida de 132 ocasiones de clasificación, los datos obtenidos en estas pruebas se presentan en la tabla 4-1 y en el grafico 1-2 donde se puede observar que los aciertos son del 100%. Se evidenció que se ejecutó el programa de forma adecuada y se corroboró que el software y el hardware respondieron como se esperada sin presentar errores que necesiten intervención humana en las pruebas.

### 4.2. Pruebas de control de las articulaciones

Igual que en el caso anterior para realizar estas pruebas se utilizó un proceso estadístico con muestreo aleatorio simple que se define en la ecuación 4.1. Para encontrar la población se midió el tiempo que tarda el brazo robot en clasificar un objeto de acuerdo al diseño de la mesa de trabajo de este proyecto. El tiempo medido fue 24 segundos el cual en hora arroja una cantidad de 150 veces que el robot puede clasificar un objeto. Con estos datos se obtuvo la población de 150 repeticiones del robot en una hora y se aplicó la ecuación 4.

$$n_o = \frac{Z^2 PQ}{e^2} = \frac{(1.96^2)(0.5)(0.5)}{(0.03)^2} = 1067$$

Como se sabe la población se puedo hacer una corrección al tamaño de la muestra anterior.

$$nc = \frac{n}{1 + \frac{n-1}{N}}$$

$$nc = \frac{1067}{1 + \frac{1067-1}{150}} = 131.62 \cong 132$$

Para realizar estas pruebas se evaluó la respuesta de una secuencia repetida de 132 órdenes enviadas por conexión serial desde la slider uno hacia el servomotor de la articulación uno. A medida que se fueron haciendo las pruebas se fue evaluando el número de aciertos que presentaba el algoritmo en el movimiento de la primera articulación, de los datos recogidos se presenta la siguiente tabla 4-2 de datos.

**Tabla 2-4:** Tabla de resultados de aciertos en control de la articulación 1

| ***Pruebas realizadas*** | ***132*** |
|---|---|
| *Número de aciertos* | 128 |
| *Número de desaciertos* | *4* |

**Realizado por:** Bejarano, L. 2021

De acuerdo con los datos mostrados en la tabla 4-2 y 4-7 el control de la articulación presenta un 97% de aciertos y un 3% de desaciertos, estos últimos causados por la velocidad de procesamiento de la máquina utilizada que en ocasiones provocaba que el programa se detenga. Después de estos intentos se realizaron pruebas con las demás articulaciones y se observó que en los primeros intentos el control respondía de forma adecuada por lo que se pudo decir que si se realizada una evaluación se habría tenido un porcentaje de aciertos aceptable como dio las pruebas que se muestran a continuación en las tablas 4-3, 4-4, 4-5,4-6 y que determinaron un 98% de fidelidad en el control de las articulaciones desde la interfaz gráfica diseñada para este objetivo.

**Tabla 3-4:** Tabla de resultados de aciertos en control de la articulación 2

| ***Pruebas realizadas*** | ***132*** |
|---|---|
| *Número de aciertos* | 129 |
| *Número de desaciertos* | *3* |

**Realizado por:** Bejarano, L. 2021

**Tabla 4-4:** Tabla de resultados de aciertos en control de la articulación 3

| ***Pruebas realizadas*** | ***132*** |
|---|---|
| *Número de aciertos* | 132 |
| *Número de desaciertos* | *0* |

**Realizado por:** Bejarano, L. 2021

**Tabla 5-4:** Tabla de resultados de aciertos en control de la articulación 4

| ***Pruebas realizadas*** | ***132*** |
|---|---|
| *Número de aciertos* | 131 |
| *Número de desaciertos* | *1* |

**Realizado por:** Bejarano, L. 2021

**Tabla 6-4:** Tabla de resultados de aciertos en control de la articulación 5

| ***Pruebas realizadas*** | ***132*** |
|---|---|
| *Número de aciertos* | 132 |
| *Número de desaciertos* | *0* |

**Realizado por:** Bejarano, L. 2021

**Tabla 7-4:** Tabla de resultados de aciertos en control de la articulación 6

| ***Pruebas realizadas*** | ***132*** |
|---|---|
| *Número de aciertos* | 128 |
| *Número de desaciertos* | *4* |

**Realizado por:** Bejarano, L. 2021

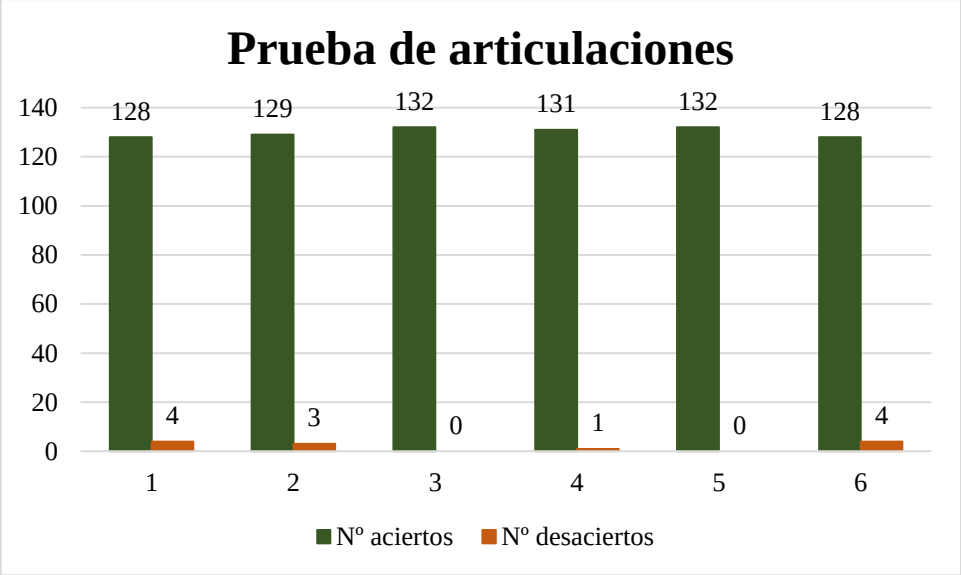



**Gráfico 2-4:** Resultado de evaluación del movimiento de las articulaciones.
**Realizado por:** Bejarano, L. 2021

### 4.3. Prueba de Hipótesis

¿Al implementar el sistema de control del manipulador por medio de visión artificial, permite que el brazo manipule y clasifique objetos en procesos repetitivos con un mínimo porcentaje de error basado en la detección de un color específico dentro del área de interés?

De acuerdo con las pruebas realizadas se observó que el brazo robótico trabajó de forma repetitiva al reconocer las piezas de color rojo y verde y clasificarlas con un mínimo de 3% de error, sin necesidad de que un operador tenga que manipular ningún control y sólo supervise dicha tarea.

# CAPÍTULO V

## 5. IMPLEMENTACIÓN

### 5.1. Implementación del algoritmo de visión artificial

El programa es automático y cada vez que se coloca un objeto en el punto de recogida el brazo actúa y lo mueve al punto de depósito de acuerdo a su color. Para lograr esto se creó un algoritmo en base a la lógica que se presenta en la figura 1-5.

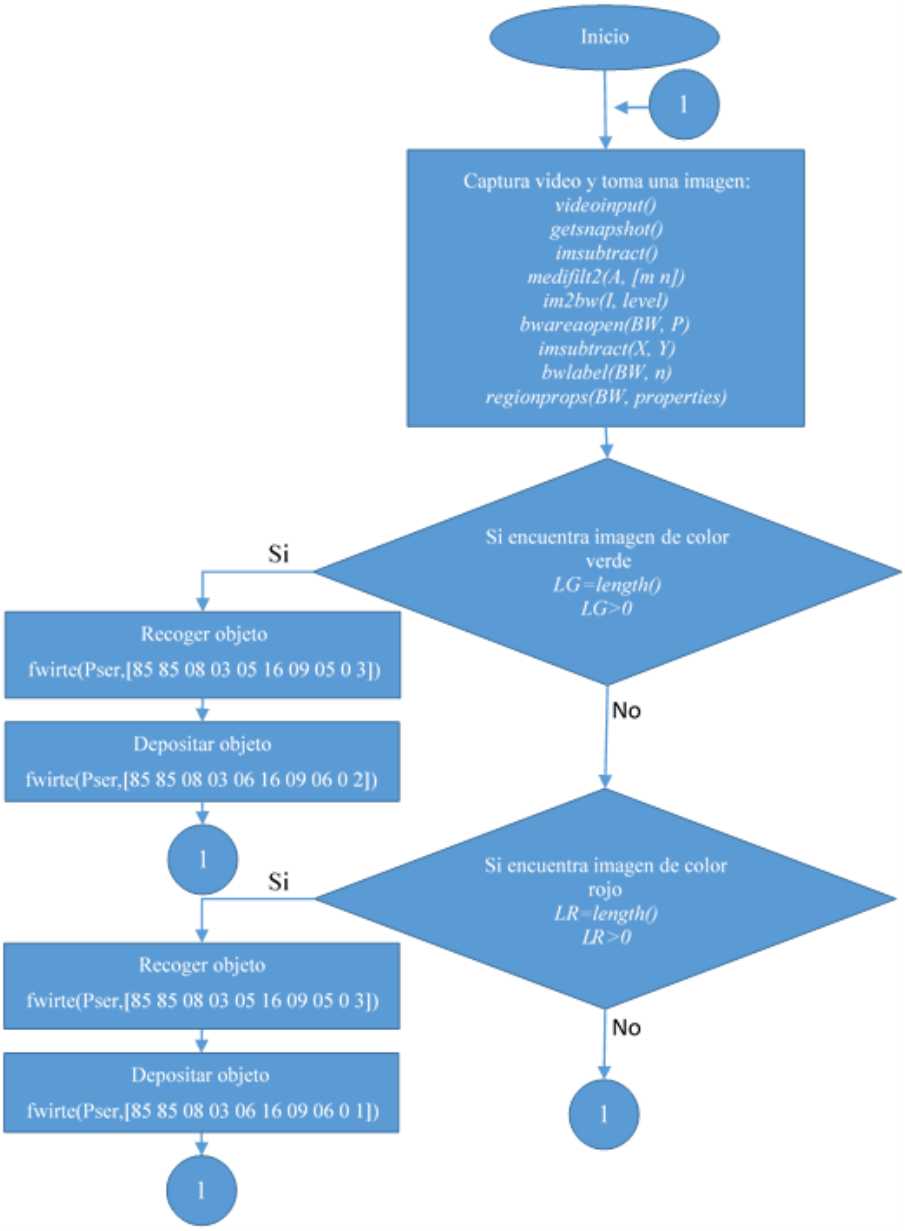



**Figura 1-5:** Interfaz gráfica creada en MATLAB para el monitoreo y control del brazo robótico LewanSoul en estado activo.
**Realizado por:** Bejarano, L. 2021

Una vez que se captura la imagen desde la cámara web se procesa la imagen mediante visión artificial para detectar si el objeto está marcado con color rojo o color verde. Antes de emitir un resultado de color el algoritmo evalúa si la imagen detectada se encuentra dentro de un rango específico en coordenadas (x, y), esto último es necesario para que el sistema no detecte una imagen por fuera del punto de recogida y el robot no se mueva por error.

Después de reconocer el color y de corroborar que el objeto reconocido está en el punto de recogida el brazo robótico se mueve desde el punto de reposo hacia el punto de recogida con los parámetros de la trama adecuados y presentados en la sección 2.4.2. Luego, una vez que recoge el objeto lo lleva y lo deposita en el punto de depósito, lugar designado para dicho objeto de acuerdo a su color.

### *5.1.1. Codificación en MATLAB*

Para detectar el color rojo o verde de un objeto se utilizó los comandos de procesamiento de imágenes de MATLAB que se explican a continuación. Para abrir la cámara web del computador y crear una variable de *video* se utilizó el comando *videoinput.* Luego con la ayuda del comando *getsnapshot* se guardó una imagen en una variable llamada "data" en formato RGB.

La figura 3-13 muestra la captura de la imagen en RGB sin filtrarla y sin procesarla mediante técnicas de visión artificial. Los filtros para que discriminar ruido no deseado y detectar el color adecuado se explica en el siguiente apartado.

**A** **B**

**Figura 2-5:** Imagen capturada y presentada mediante *imsubtract* de MATLAB.
A) Imagen en escala de grises. - B) Imagen RGB
**Realizado por:** Bejarano, L. 2021

### *5.1.2. Filtrado y binarización de la imagen*

Una vez que la imagen fue transformada a escala de grises se aplicó *medifilt2(A, [m n])* que es un comando de MATLAB para utilizar un filtro de operación no lineal que generalmente se aplica en imágenes para reducir el ruido denominado "sal y pimienta". Este filtro opera sacando el valor medio de una matriz interna *m* por *n* de dos dimensiones que define el usuario, es decir, pone a cada uno de los pixeles de una región *m* por *n* con valores medios y rellena con ceros los contornos de las imágenes a procesar.

Se realizaron pruebas de filtrado con una matriz de efecto 1 por 1 y se observan los resultados en la figura 3-14B y con una variación en la matriz de 10 por 10 se observa en la figura 3-14C. La acción y efecto del filtro que se observa en las imágenes B y C de la figura 3-14 muestran gráficamente el resultado obtenido con diferentes matrices de filtrado. Debido a estos resultados se probó el funcionamiento con una matriz 5 por 5 con el reconocimiento de colores y se explica en el siguiente apartado.


**Figura 3-5:** Imagen capturada y filtrada mediante medifilt2(A, [m n]) de MATLAB.
A) Imagen RGB - B) Imagen filtrada con matriz 1 x 1 - C) Imagen filtrada con matriz 10 x 10.
**Realizado por:** Bejarano, L. 2021

Después de aplicar el filtro con una matriz 5 por 5 antes mencionado se binarizó la imagen utilizando el comando *im2bw(I, level)*, que se encarga de convertir una imagen de escala de grises a imagen binaria. La función *im2bw(I, level)* proporciona una matriz de pixeles con un valor de 1 a aquellos que superan un nivel establecido y un valor de 0 a aquellos que están por debajo de dicho nivel. El nivel puede tomar valores entre 0 y 1, es decir que si se establece un valor de 0.5 se estaría estableciendo un valor medio entre blanco y negro.

En la figura 4-5 se observa los resultados al aplicar la función *im2bw(I, level)* con un nivel de 0.1 a 0.5 de la cual se ha seleccionado el nivel 0.3 que es donde se reduce el ruido al mínimo.

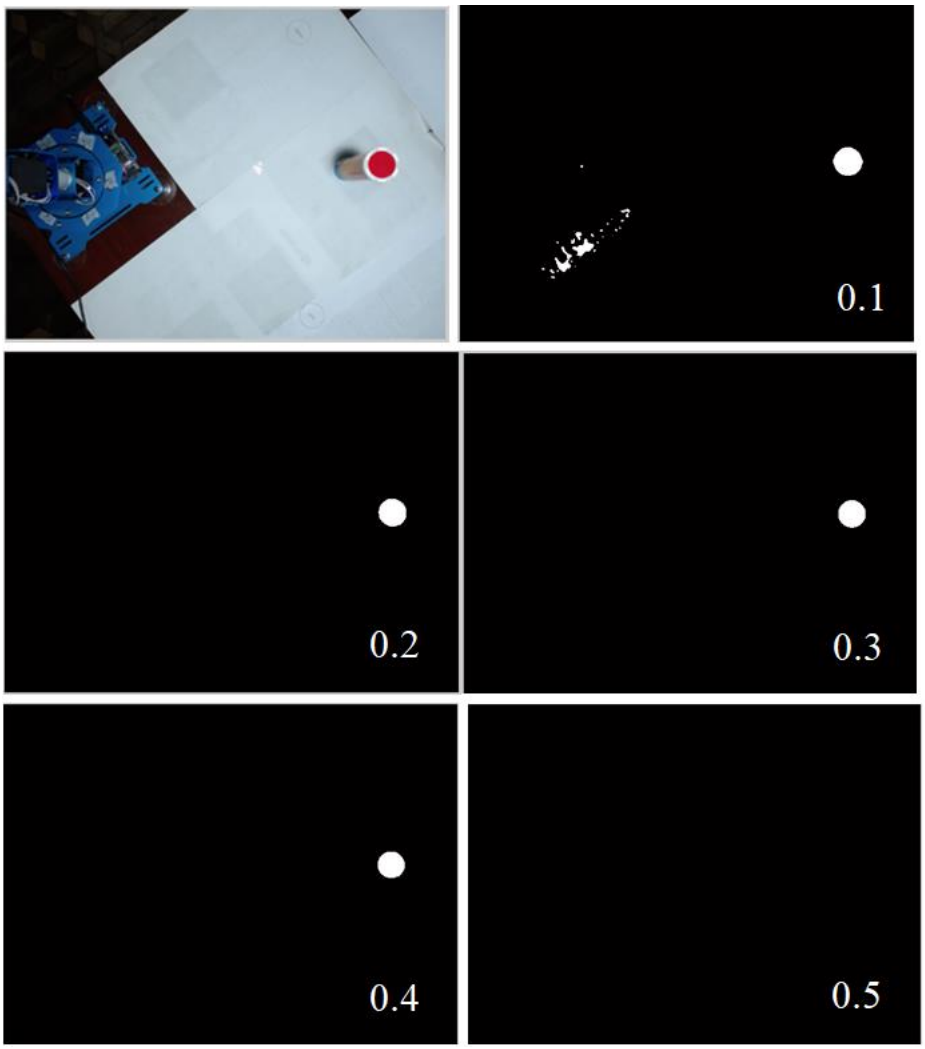
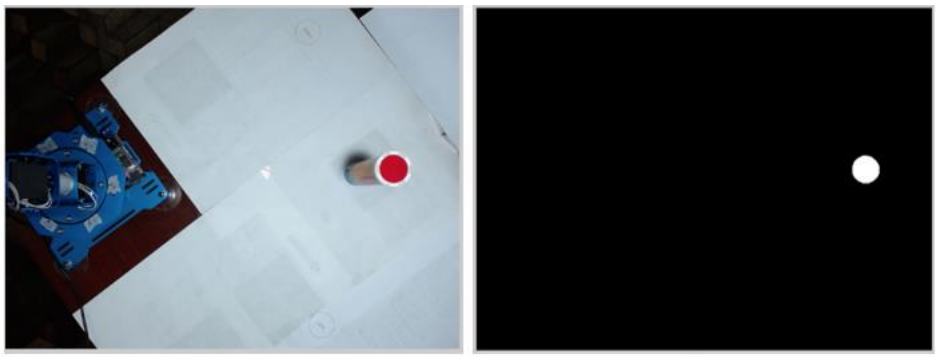



**Figura 4-5:** Imagenes binarizadas con niveles de 0.1 a 0.5 en la función im2bw(I, level).
**Realizado por:** Bejarano, A. 2021

Después de binarizar la imagen se utilizó una operación morfológica de erosión mediante el comando *bwareaopen(BW, P)* de MATLAB, con la que se remueve objetos con un conjunto de pixeles menores a P.

En la figura 3-16 se observa el resultado de utilizar la función *bwareaopen(BW, P)* con un conjunto de pixeles igual a 50.



A B

**Figura 5-5:** Imagen erosionada mediante *bwareaopen(BW, P)* con un valor de P=50.
**Realizado por:** Bejarano, L. 2021

### *5.1.3. Detección de color verde o rojo*

El reconocimiento del color mediante MATLAB depende directamente de la función *imsubtract(X, Y)* que resta los valores de cada elemento de una matriz Y señalados en la correspondiente matriz X. Con lo que si se proporciona una matriz X con los elementos que determinan una imagen de color rojo se conseguirá obtener una matriz de ese color. Para este trabajo se proporcionó la función con los valores *imsubtract(data(:,:,1), Y)*, donde los datos que se obtuvieron de la imagen capturada se restaran los que sean de color rojo.


A B

**Figura 6-5:** Reconocimiento de color con el algoritmo creado en coordenadas específicas.
A) Reconocimiento del color rojo. B) Reconocimiento del color verde.
**Realizado por:** Bejarano, L. 2021

Luego, con la ayuda de la función *bwlabel(BW, n)* de MATLAB se obtuvo las características de *n* objetos conectados dentro la imagen. Después de obtener las características de los objetos de las imágenes con la ayuda de *regionprops(BW, properties)* de MATLAB se obtuvo las propiedades del objeto detectado en la imagen. Entre estas propiedades se encuentran el área, contornos o bordes, centroide, área convexa y demás.

Con el centroide del objeto se dibujó un círculo de acuerdo al color del objeto con *rectangle.* Esto último es posible solamente si el centroide se encuentra dentro del rango de coordenadas específicas que determinan que el objeto está en el punto de recogida, y así el brazo robótico se active de forma automática, lo recoja y clasifique de acuerdo con su color. El resultado de reconocimiento de color se puede apreciar en la figura 3-17.

### *5.1.4. Conexión serial entre MATLAB y el controlador de servos*

La conexión serial entre el MATLAB y el controlador de servos se realizó en base al protocolo de comunicación proporcionada por el fabricante (LewanSoul, 2019). La trama de comunicación que se observó en la tabla 2-2 tiene los siguientes componentes:

- Los dos primeros bits (0x55 0x55) se utilizan para avisar del inicio de una nueva trama de comunicación.
- El siguiente bit muestra la longitud de los datos que se van a recibir.
- La sección comando indica la acción de control a realizar.
- Por último, los parámetros muestran el servomotor que se va a manejar, el ángulo que se va a mover y el tiempo en el que se ejecutará dicha acción.

Mediante MATLAB se envió la trama de datos por el puerto serial para realizar las primeras pruebas de movimiento, es así que, para mover el servo uno a una posición 2000 con un tiempo de 1 segundo se envió desde MATLAB la siguiente trama: 0x55 0x55 0x08 0x03 0x01 0xE8 0x03 0x01 0xD0 0x07 de acuerdo como se muestra en la figura 3-18.

```
fwrite(Pser,[85 85 08 03 01 16 09 01 144 01],'uint8')
```

**Figura 7-5:** Trama de prueba para enviar ordenes al brazo robótico.
**Realizado por:** Bejarano, L. 2021

Luego, para realizar movimientos con todos los servos motores del brazo se envió la siguiente secuencia de tramas que lo llevaban desde el punto inicial hacia el punto de recogida del objeto, y luego hacia el punto de depósito.

A continuación, se muestran las tramas que mueven los servos motores hacia los puntos escogidos como punto de reposo del robot, punto de recogida, punto de depósito del objeto verde y punto de depósito del objeto rojo respectivamente de acuerdo a las especificaciones del brazo robótico de LewanSoul:

*Tramas de datos para mover el brazo robótico al punto de reposo*

```
fwrite(Pser,[85 85 08 03 06 16 09 06 115 1],'uint8');
fwrite(Pser,[85 85 08 03 05 16 09 05 0 2],'uint8');
fwrite(Pser,[85 85 08 03 04 16 09 04 0 2],'uint8');
fwrite(Pser,[85 85 08 03 03 16 09 03 0 2],'uint8');
fwrite(Pser,[85 85 08 03 02 16 09 06 0 2],'uint8');
fwrite(Pser,[85 85 08 03 01 16 09 06 0 2],'uint8');
```

*Tramas de datos para mover el brazo robótico al punto de recogida*

```
fwrite(Pser,[85 85 08 03 05 16 09 05 0 3],'uint8');
fwrite(Pser,[85 85 08 03 04 16 09 04 0 1],'uint8');
fwrite(Pser,[85 85 08 03 03 16 09 03 110 1],'uint8');
fwrite(Pser,[85 85 08 03 01 16 09 01 90 2],'uint8');
fwrite(Pser,[85 85 08 03 03 16 09 03 90 1],'uint8');
```

*Tramas de datos para mover el brazo robótico al punto de depósito de color rojo*

```
fwrite(Pser,[85 85 08 03 06 16 09 06 115 1],'uint8');
fwrite(Pser,[85 85 08 03 03 16 09 03 110 1],'uint8');
fwrite(Pser,[85 85 08 03 01 16 09 01 30 1],'uint8');
fwrite(Pser,[85 85 08 03 05 16 09 05 0 2],'uint8');
fwrite(Pser,[85 85 08 03 04 16 09 04 0 2],'uint8');
fwrite(Pser,[85 85 08 03 03 16 09 03 0 2],'uint8');
fwrite(Pser,[85 85 08 03 06 16 09 06 0 2],'uint8');
```

*Tramas de datos para mover el brazo robótico al punto de depósito de color verde*

```
fwrite(Pser,[85 85 08 03 06 16 09 06 140 2],'uint8');
fwrite(Pser,[85 85 08 03 03 16 09 03 110 1],'uint8');
fwrite(Pser,[85 85 08 03 01 16 09 01 30 1],'uint8');
fwrite(Pser,[85 85 08 03 05 16 09 05 0 2],'uint8');
fwrite(Pser,[85 85 08 03 04 16 09 04 0 2],'uint8');
fwrite(Pser,[85 85 08 03 03 16 09 03 0 2],'uint8');
fwrite(Pser,[85 85 08 03 06 16 09 06 0 2],'uint8');
```

### *5.1.5. Algoritmo implementado en MATLAB*

El algoritmo creado en un *Script* de MATLAB para reconocer color rojo se presenta a continuación en la figura 3-19. El algoritmo para reconocer color verde queda de la misma

manera, pero se debe cambiar los parámetros para detectar características verdes como se explica en la sección 3.4.3.

```
while (video == 1)
    //Se captura la imagen desde la cámara web
    set(vid, 'ReturnedColorSpace', 'RGB');
    data = getsnapshot(vid);
    // Se captura características de la imagen con color rojo
    diff_im = imsubtract(data(:,:,1), rgb2gray(data));
    // Se aplica filtro y se binariza la imagen
    diff_im = medfilt2(diff_im, [5 5]);
    diff_im = im2bw(diff_im,0.3);
    diff_im = bwareaopen(diff_im,50);
    // Se obtiene las propiedades (centroide) de la imagen
    bw = bwlabel(diff_im, 8);
    stats = regionprops(bw, 'BoundingBox', 'Centroid');
    // Se verifica si hay objetos de color rojo o verde
    LR=length(stats);
    if LR>0
        // Se obtiene el centroide del objeto
        bb = stats(1).BoundingBox;
        // Se verificasi el centroide está dentro del área de recogida
        if bb(1)>494 && bb(1)<532 && bb(2)>190 && bb(2)<221
            Pser=handles.Serial;
            //Se dibuja un circulo de color rojo sobre el contorno
            rec=rectangle('Position',bb,'EdgeColor','r',
            'LineWidth',2,'Curvature',[1,1]);
            //Se envian las tramas necesarias para mover el robot
            fwrite(Pser,[85 85 08 03 05 16 09 05 0 3],'uint8');
            fwrite(Pser,[85 85 08 03 04 16 09 04 0 1],'uint8');
            fwrite(Pser,[85 85 08 03 03 16 09 03 110 1],'uint8');
            pause(4)
            fwrite(Pser,[85 85 08 03 01 16 09 01 90 2],'uint8');
            pause(2)
            fwrite(Pser,[85 85 08 03 03 16 09 03 90 1],'uint8');
            pause(3)
            fwrite(Pser,[85 85 08 03 06 16 09 06 115 1],'uint8');
            pause(3)
            fwrite(Pser,[85 85 08 03 03 16 09 03 110 1],'uint8');
            pause(4)
            fwrite(Pser,[85 85 08 03 01 16 09 01 30 1],'uint8');
            pause(2)
            fwrite(Pser,[85 85 08 03 03 16 09 03 0 2],'uint8');
            pause(2)
            fwrite(Pser,[85 85 08 03 06 16 09 06 0 2],'uint8');
            delete(rec)
        End
    End
End
```

**Figura 8-5:** Algoritmo para enviar ordenes al brazo robótico.
**Realizado por:** Bejarano, L. 2021

## CONCLUSIONES

La interfaz creada en una GUI de MATLAB nos ayuda al control de los servomotores del brazo robot y permite el monitoreo del mismo con esto se logra que los servomotores no giren fuera de los rangos no permitidos.

Con la adecuación y calibración de las articulaciones se obtiene que el brazo robot sea más preciso al momento de realizar el agarre de los objetos para su posterior clasificación.

El reconocimiento de colores con las herramientas de procesamiento de imágenes de MATLAB se obtuvo un 100% en reconocimiento de acuerdo al color. Con estos resultados se puede decir que el robot trabajará de forma adecuada en una industria para realizar tareas de clasificación repetitivas.

Con la implementación electrónica de potencia y comunicación se realiza la integración entre el software y hardware pero siempre hay retardos de tiempo en la comunicación, y más aún si se deben hacer cálculos como es el caso del presente trabajo. Sin embargo, se consigue realizar un control en un 97% que permite realizar tareas de clasificación con el brazo.

Se observó en las pruebas su correcto funcionamiento en base a los objetivos planteados. Es decir, el sistema reconoce objetos de acuerdo a su color y los clasifica.

## RECOMENDACIONES

Como una recomendación es necesario mencionar que los objetos a manipular con el brazo robot no debe exceder el peso de torque de los servomotores que este caso es 15 kg/cm. Y ya que la primera articulación del robot debe soportar el peso de los eslabones y la pinza no se recomienda pasar de los 2 kg/cm para que no se afecte el adecuado funcionamiento del sistema.

Para realizar un buen reconocimiento de objetos se recomienda trabajar en un entorno con luz adecuada para que la cámara pueda captar imágenes claras y diferenciar el color de los objetos enfocados.

Se recomienda conectar el brazo robótico con su cargador propio de LewanSoul ya que debe proporcionar la corriente adecuada que en determinado momento el brazo lo requiera para realizar esfuerzos en la traslación de objetos.

Se recomienda probar el funcionamiento del brazo con la interface gráfica proporcionada por el fabricante para verificar su adecuado trabajo. No es recomendable conectarlo a MATLAB directamente para realizar estas pruebas ya que el protocolo de comunicación serial cuenta con una notación definida por el fabricante.

**BIBLIOGRAFÍA**


Albesa, L. A. (2015). *Matlab para matemáticas en ingenierías.* Valencia: Editorial de la Universidad Politécnica de Valencia.

Arnaez Luis, E. B. (2015). *Enfoque práctico de la teoría de robots.* Lima: Universidad Peruana de Ciencias Aplicadas (UPC).

Barrientos, A., Peñin, L., Balaguer, C., & Aracil, R. (2007). *Fundamentos de Robótica.* Madrid-España: S.A. MCGRAW-HILL / INTERAMERICANA DE ESPAÑA.

Comunicación serie. (12 de noviembre de 2015). *Comunicación serie*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Comunicaci%C3%B3n_serie

Desarrollo en espiral. (20 de febrero de 2018). *Desarrollo en espiral*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Desarrollo_en_espiral

Descripción general. (2018). *Descripción general*. Obtenido de MathWorks: https://la.mathworks.com/products/matlab.html

Dorf, R., & Bishop, R. (2007). *Sistemas de control moderno.* Madrid-España: PERSON EDUCACIÓN S.A.

Fernandez, R. (26 de mayo de 2008). *Control de posición de un servomotor de corriente contínua.* Obtenido de UPC: https://ocw.upc.edu/sites/all/modules/ocw/estadistiques/download.php?file=11608/2011/1/53987/sec_p7_control_de_posicion_de_un_servomotor_de_cc_0708b-5205.pdf

Fernandez-Pacheco, A. S., Flor, F. R., Rodríguez, R. F., Gutiérrez, I. P., & Oliver, A. A. (2015). *Robótica Educativa.* Madrid: RA-MA Editorial.

GUI de MATLAB. (2018). *Creación de apps con interfaces gráficas de usuario en MATLAB*. Obtenido de MathWorks: https://es.mathworks.com/discovery/matlab-gui.html

Kumar Saha, S. (2008). *Introducción a la Robótica.* México: McGRAW-HILL/INTERAMERICANA EDITORES, S.A. DE C.V.

LewanSoul. (2017). *LeArm 6DOF Robotic Arm*. Obtenido de LewanSoul: http://www.lewansoul.com/product/detail-135.html

Logitech. (2019). *Cámara Web*. Obtenido de Logitech: https://www.logitech.com/es-es/product/c930e-webcam

MATLAB. (04 de junio de 2018). *MATLAB*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/MATLAB

Nasir, S. Z. (15 de noviembre de 2015). *Color Detection in Images using MATLAB*. Obtenido de https://www.theengineeringprojects.com/2015/11/color-detection-in-images-using-matlab.html

Pérez Marco, A. C. (2014). *Fundamentos de robótica y mecatrónica con Matlab© y Simulink©.* Madrid: RA-MA Editorial.

Procesamiento de imagenes. (2018). *Procesamiento, análisis y desarrollo de algoritmos de imágenes*. Obtenido de MathWorks: https://es.mathworks.com/products/image.html

Ruiz-Velasco, E. S. (2012). *Los Robots.* Madrid: Ediciones Díaz de Santos.

Servomotor. (26 de enero de 2011). *Servomotor*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Servomotor

Use Serial Communications. (2018). *Use Serial Communications with Arduino Hardware*. Obtenido de MathWorks: https://es.mathworks.com/help/supportpkg/arduino/ug/use-serial-communications-with-arduino-hardware.html