ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Mecánica y Ciencias de la Producción

" Análisis cinemático de un brazo robótico de cinco grados de libertad e implementación de un control en Matlab"

TRABAJO FINAL DE GRADUACIÓN

PROYECTO INTEGRADOR

Previo la obtención del Título de:

INGENIERO MECÁNICO

Presentado por:

Samuel Santiago Escandón Vizueta

Michelle Caroline Trujillo Torres

GUAYAQUIL - ECUADOR

Año: 2017

AGRADECIMIENTOS

Agradezco a Dios, mi familia en especial a mi mamá, a Abner Valdivieso por su apoyo incondicional y a todos los que hicieron posible esto.

Michelle Caroline Trujillo Torres

AGRADECIMIENTOS

Este proyecto es el resultado del esfuerzo conjunto de todos los que formamos el grupo de trabajo. Por esto agradezco a nuestro tutor de proyecto, PhD. Jorge Hurel, a mi compañera Michelle Trujillo y mi persona, quienes a lo largo de este tiempo han puesto a prueba sus capacidades y conocimientos en el desarrollo de este proyecto. A mis padres quienes siempre han estado apoyado y motivado mi formación académica, creyeron en mí en todo momento y no dudaron de mis habilidades. A mis profesores a quienes les debo gran parte de mis conocimientos, gracias a sus enseñanzas y finalmente un eterno agradecimiento a esta prestigiosa universidad la cual abrió abre sus puertas a jóvenes como nosotros, preparándonos para un futuro competitivo y formándonos como personas de bien.

Samuel Santiago Escandón Vizueta

DECLARACIÓN EXPRESA

"La responsabilidad del contenido desarrollado en la presente propuesta de la materia integradora corresponde exclusivamente al equipo conformado por:
Samuel Santiago Escandón Vizueta
Michelle Caroline Trujillo Torres
Jorge Luis Hurel Ezeta
y el patrimonio intelectual del mismo a la Facultad de Ingeniería Mecánica y Ciencias de la Producción (FIMCP) de la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

Samuel Escandón Vizueta
Autor #1

Michelle Trujillo Torres
Autor #2

Jorge Luis Hurel Ezeta PhD.
Tutor de materia integradora

RESUMEN

En la industria los brazos robóticos mejoran la calidad de producción, realizan tareas repetitivas y complejas sin poner en riesgo vidas humanas. Con el software Matlab se controla el brazo robótico de cinco grados de libertad de forma autónoma para desplazar objetos reconocidos por medio de visión artificial, con el fin de fomentar el desarrollo industrial en Ecuador. Se le instaló potenciómetros en el brazo robótico OWI-535 para registrar el cambio de posición en cada articulación y un pulsador final de carrera en la pinza. El uso del microcontrolador Arduino Mega 2560, los controladores de motor L298N y Motor Shield que manipulan los actuadores desde Matlab con una comunicación inalámbrica por Bluetooth HC-05. Se generó el espacio de trabajo del brazo con el análisis cinemático directo y se comprobó el comportamiento real del brazo con la simulación en Matlab garantizando su funcionamiento óptimo. Finalmente se obtuvo un brazo robótico autónomo con la resolución del análisis cinemático inverso a partir de la detección y ubicación del objeto por su color.

Palabras Clave: Matlab, brazo robótico, autónoma, desplazar objetos, Arduino, análisis cinemático.

ABSTRACT

In industry, robotic arms improve production quality, perform repetitive and complex tasks without endangering human lives. With the help of Matlab software, the robotic arm of five degrees of freedom is controlled and autonomously move objects through artificial vision, to encourage the industrial development in Ecuador. Potentiometers were installed on the robotic arm OWI-535 to record the change of position in each joint and a final push-button on the clamp. Using of Arduino Mega 2560 microcontroller, the L298N motor controllers and the Motor Shield that manipulate the actuators from Matlab with a wireless communication via Bluetooth HC-05. The workspace of the arm was generated with the direct kinematic analysis and the real behavior of the arm was verified with the simulation in MATLAB guaranteeing its optimal functioning. Finally, we obtained an autonomous robotic arm with the resolution of the inverse kinematic analysis from the detection and location of the object by its color.

Keywords: Matlab, robotic arm, autonomous, move objects, Arduino, kinematic analysis

ÍNDICE GENERAL

RESUMEN I

ABSTRACT II

ÍNDICE GENERAL III

ABREVIATURAS VI

SIMBOLOGÍA VII

ÍNDICE DE FIGURAS VIII

ÍNDICE DE TABLAS IX

CAPÍTULO 1 1

1. Introducción 1

1.1 Descripción del problema 1

1.2 Objetivos 2

1.2.1 Objetivo general 2

1.2.2 Objetivos específicos 2

1.3 Marco teórico 2

1.3.1 Matlab 2

1.3.2 Arduino MEGA 2560 3

1.3.3 Definición de robot 3

1.3.4 Tipos de robots 3

1.3.5 Robot Industrial 4

1.3.6 Brazo robótico OWI-535 4

1.3.7 Eslabones y articulaciones 5

1.3.8 Grados de libertad 5

1.3.9 Posición y Orientación 5

1.3.10 Modelo Cinemático 6

1.3.11 Cinemática Directa 7
1.3.12 Cinemática Inversa 9
1.3.13 Visión artificial 9
CAPÍTULO 2 11
2. Metodología 11
2.1 Alternativas de solución 11
2.1.1 Tarjetas de control 11
2.1.2 Sensores 11
2.1.3 Comunicación del computador al Arduino 12
2.2 Descripción y selección de la mejor alternativa 12
2.3 Diseño conceptual 14
2.4 Metodología de diseño a seguir 16
2.4.1 Visión artificial 17
2.4.2 Análisis cinemático directo 18
2.4.3 Análisis cinemático inverso 19
2.4.4 Diseño electrónico 23
2.4.5 Simulación 24
CAPÍTULO 3 26
3. Resultados 26
3.1 Visión artificial 26
3.2 Análisis cinemático directo 28
3.3 Análisis cinemático inverso 29
3.4 Espacio de trabajo 30
3.4.1 Espacio de trabajo OWI-535 30
3.4.2 Espacio de trabajo del proyecto 33
3.5 Simulación 35

3.6 Análisis de costos 37
CAPÍTULO 4 39
4. Discusión y Conclusiones 39
4.1 Conclusiones 39
4.2 Recomendaciones 40
BIBLIOGRAFÍA
APÉNDICES
APÉNDICE A
APÉNDICE B
1) Distancia en píxeles
2) Visión artificial
3) Cinemática Inversa
4) Espacio de trabajo OWI-535
5) Espacio de trabajo del proyecto
6) Simulación
7) Proyecto Arduino
8) Proyecto Matlab
APÉNDICE C

ABREVIATURAS

D	Dimensión
DC	Corriente Directa
D-H	Denavit-Hartenberg
ESPOL	Escuela Superior Politécnica del Litoral
IFR	Federación Internacional de Robótica
ISO	Organización Internacional para Estandarización
JIRA	Asociación de Robótica de Japón
MATLAB	Matriz de laboratorio
RIA	Asociación de Industrias Robóticas
RUR	Robot universal de Rossum
SCARA	Ensamblaje de brazo robótico de cumplimiento selectivo

SIMBOLOGÍA

mm	Milímetros
°	Grados
θ	Ángulo
px	Valor de eje x
py	Valor de eje y
pz	Valor de eje z
A_i^{i-1}	Matriz homogénea de eslabón
T	Matriz homogénea de brazo robótico

ÍNDICE DE FIGURAS

Figura 1.1. Brazo robótico OWI-535 5
Figura 1.2. Diagrama de relación entre cinemática directa e inversa 7
Figura 1.3. Esquema genérico de un sistema de control visual. 10
Figura 2.1. Descripción general del brazo robótico automatizado 15
Figura 2.2. Descripción general de la metodología de diseño 16
Figura 2.3. Ilustración a) sistema de referencia cámara y b) robot 18
Figura 2.4. Sistema de referencia visión artificial y Plano V del análisis cinemático. . 19
Figura 2.5. Ilustración del brazo robótico señalando cada articulación 20
Figura 2.6. Brazo robótico automatizado 24
Figura 3.1. Medición distancia en píxeles de un cuadro 26
Figura 3.2. Coordenadas de un objeto de color rojo 27
Figura 3.3. Coordenadas de un objeto de color verde 27
Figura 3.4. Coordenadas de un objeto de color azul 28
Figura 3.5. Sistema de referencia usando la convención de D-H 29
Figura 3.6. Vista lateral del a) brazo robótico y b) espacio de trabajo OWI-535 31
Figura 3.7. Vista superior del espacio de trabajo OWI-535 32
Figura 3.8. Vista lateral espacio de trabajo con $\theta 1 = \mathbf{0}$ 32
Figura 3.9. Vista frontal del espacio de trabajo del proyecto 33
Figura 3.10. Vista superior del espacio de trabajo del proyecto 34
Figura 3.11. Vita lateral del espacio de trabajo del proyecto 34
Figura 3.12. Simulación de brazo robótico 36
Figura 3.13. Comparación brazo robótico a) simulación y b) movimiento real 37
Figura 4.1. Partes brazo robótico OWI-535 44
Figura 4.2. Dimensiones brazo robótico OWI-535 45
Figura 4.3. Movimientos del brazo robótico OWI-535 46
Figura 4.4. Área de trabajo del brazo robótico OWI-535 47
Figura 4.5. Área de trabajo y brazo robótico OWI-535 automatizado 75
Figura 4.6. Sensores de posición instalados en el brazo robótico OWI-535 76

ÍNDICE DE TABLAS

Tabla 1.1 Tipos de robots. 3
Tabla 1.2 Parámetros del método de D-H. 8
Tabla 2.1 Características técnicas 11
Tabla 2.2 Ponderación para selección 12
Tabla 2.3 Matriz de decisión tarjeta de control 13
Tabla 2.4 Matriz de decisión sensor 13
Tabla 2.5 Matriz de decisión conexión Arduino y computadora 13
Tabla 3.1 Parámetro de D-H del OWI-535 29
Tabla 3.2 Puntos máximos, mínimos y ángulos de articulaciones en OWI-535 35
Tabla 3.3 Ejemplo movimiento OWI-535 36
Tabla 3.4 Detalle de recursos empleados en el proyecto 38

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Descripción del problema

Desde la primera revolución industrial, se han implementado nuevas tecnologías que han mejorado la producción, la calidad y la seguridad, entre muchos otros beneficios para la humanidad. La implementación de la robótica en el último siglo mejoró significativamente la producción y la calidad de diversos productos que fueron fabricados por muchos años por la mano del hombre.

Los robots reemplazan al humano fácilmente porque pueden realizar numerosas tareas repetitivas sin equivocarse, no son afectados por el cansancio físico y realizan trabajos peligrosos para la salud humana. Se pueden adaptar a los cambios con base en la demanda variable, además de disminuir los costos de producción de una manera significativa. Actualmente, se los utilizan en casi todos las áreas productivas y tipos de industrias como: ensamblado, soldadura, automovilística, ensamblaje de circuitos eléctricos, etc. También tienen aplicaciones no industriales como espaciales, submarinas, subterráneas, militares, médicas, agrícolas, etc.

El diseño, desarrollo e implementación de robots industriales, que se programan de manera simple, tienen como objetivo la eficiencia en las cadenas de producción, por su desempeño en trabajos de forma rápida, sin poner en riesgo la vida de personas; asimismo, realizan labores complejas que requieren de mucha precisión. Por este motivo, las empresas muestran gran interés en la inclusión de brazos robóticos para la producción de diversos productos.

En este proyecto, se realizan análisis cinemáticos de un brazo robótico de cinco grados de libertad y se lo automatiza por medio del lenguaje de Matlab,

con función en una industria de ensamblaje de objetos, para optimizar el proceso de una fábrica y sustituir la interacción del hombre en ciertas líneas de la producción, para ejecutar la tarea específica de trasladar objetos de una ubicación a otra con libertad, sin colisionar con obstáculos, y mover elementos pesados. El entorno seleccionado para desarrollar el control del brazo robótico es Matlab, dado que facilita los cálculos vectoriales y matriciales involucrados en el análisis cinemático.

1.2 Objetivos

1.2.1 Objetivo general

Automatizar un brazo robótico de cinco grados de libertad a través del análisis cinemático y la implementación de un control por medio de Matlab para desplazar objetos.

1.2.2 Objetivos específicos

Analizar y controlar el movimiento de un brazo robótico por medio del programa Matlab, localizando un objeto con visión artificial.

Definir los rangos de trabajo que el brazo robótico puede alcanzar en su entorno para evitar singularidades.

Comprobar el comportamiento del brazo robótico comparando con la simulación.

1.3 Marco teórico

1.3.1 Matlab

Matlab es un lenguaje de programación matemático de matrices: integra un entorno gráfico amigable, visualización de datos, funciones, gráficas de 2D y 3D, procesamiento de imágenes, vídeo y computación numérica para desarrollar algoritmos matemáticos con aplicaciones en ingeniería y ciencias exactas. MATLAB es un acrónimo que proviene de “Matrix Laboratory”

(laboratorio matricial) (Reyes Cortés, Matlab aplicado a robótica y mecatrónica, 2012).

1.3.2 Arduino MEGA 2560

Placa basada en el microcontrolador ATmega2560. Como características más destacables con 54 pines de entrada/salida digital (de los cuales 14 pueden ser usados como salidas analógicas PWM), 16 entradas analógicas y 4 receptores/transmisores serie TTL-UART. Consta de una memoria Flash de 256 Kilobytes, una memoria SRAM de 8 KB y una EEPROM de 4 KB. Voltaje de trabajo es 5 V (Torrente Artero, 2013).

1.3.3 Definición de robot

Organización Internacional para la Estandarización (ISO), define robot como un manipulador multifuncional reprogramable, capaz de mover objetos, a través de movimientos programados, para el desarrollo de diferentes actividades (Saha, 2010).

1.3.4 Tipos de robots

La función que desempeñará un robot influye enormemente en el diseño del mismo. Dependiendo de la función, se utiliza un tipo específico de robot. De manera general pueden ser clasificados como se muestra en la tabla 1.1 (Reyes Cortés, Matlab aplicado a robótica y mecatrónica, 2012).

Tabla 1.1 Tipos de robots.

Clasificación de robots		
Móviles	Terrestres: ruedas, patas	
	Submarinos, aéreo-espaciales	
Humanoides	Diseño complejo	
Industriales	Brazos mecánicos	Robots manipuladores

Fuente: Robótica. Control de robots manipuladores, 2011

Elaboración Cortés, F. (2011)

1.3.5 Robot Industrial

Los robots industriales son conocidos como brazos robots o brazos mecánicos, por analogía con el brazo humano. En general los robots industriales poseen los siguientes elementos:

1. Articulaciones o uniones que permiten la conexión y movimiento relativo entre dos eslabones consecutivos del robot.
2. Actuadores suministran las señales necesarias a las articulaciones para producir movimiento.
3. Sensores proporcionan información del estado interno del robot. Mejoran la capacidad de percepción del robot, le permiten responder a su entorno de manera versátil y autónoma.
4. Sistema mecánico consiste en una secuencia de eslabones rígidos de metal conectados en cadena abierta por medio de articulaciones.
5. Consola de control se compone de un sistema electrónico que incluye un dispositivo que brinda la interfaz necesaria para que el usuario se comunique con el robot a través de programación (Reyes Cortés, Robótica. Control de robots manipuladores, 2011).

1.3.6 Brazo robótico OWI-535

Es un brazo robótico controlado remotamente con cinco grados de libertad se presenta en la figura 1.1, una capacidad de elevación de 3.5oz (100 g) y una luz LED. Utilizando cinco motores de corriente directa (DC) con cajas de engranajes, el OWI-535 tiene un movimiento de muñeca de 120°, de codo de 300°, de hombro de 180°, de base de 270° y de agarre de 0-1.77 "(0-4.5 cm). Las especificaciones técnicas se encuentran en el apéndice A (OWI, 2018).

Figura 1.1. Brazo robótico OWI-535

Fuente: OWI Inc, (2017)

1.3.7 Eslabones y articulaciones

Los cuerpos individuales que conforman un robot se llaman “eslabones”. Los eslabones son cuerpos rígidos, la distancia entre 2 puntos en el cuerpo no variará, aunque este se encuentre en movimiento. La articulación une dos eslabones y proporciona las restricciones de movimiento entre los eslabones. Las articulaciones se clasifican de acuerdo con sus restricciones en: revoluta, prismático, helicoidal, cilíndrica, esférica, planar (Reyes Cortés, Matlab aplicado a robótica y mecatrónica, 2012).

1.3.8 Grados de libertad

Todo cuerpo rígido en el espacio cartesiano posee seis grados de libertad (6 GDL). La posición y orientación de un cuerpo se puede describir mediante tres coordenadas de traslación y tres ejes de rotación (Saha, 2010).

1.3.9 Posición y Orientación

El movimiento de un robot implica el movimiento espacial de cada uno de sus partes. Para que un robot puede recoger una pieza, se necesita conocer la posición y orientación de ésta con respecto a la base del robot (Barrientos, Peñin, Balaguer, & Aracil, 2007).

Posición

Para ubicar un cuerpo rígido en el espacio se requiere posicionar espacialmente los puntos. En el caso de un plano existe 2 componentes individuales. Pero en un caso tridimensional será necesario 3 componentes para posicionar el cuerpo. Los métodos para coordinar un punto son: coordenadas cartesianas, polares, cilíndricas, esféricas (Barrientos, Peñin, Balaguer, & Aracil, 2007).

Orientación:

Para un cuerpo en el espacio, definir su posición en el espacio es insuficiente. Con respecto a un sistema de referencia se define su orientación. En un espacio tridimensional la orientación está definida por tres componentes linealmente independientes o tres grados de libertad.

Matriz de rotación: Uno de los métodos más extendidos para describir la orientación, debido a la facilidad que proporciona el álgebra lineal, es la matriz de rotación. Las matrices de rotación pueden componerse para expresar la aplicación continua de varias rotaciones.

Ángulos de Euler: Aunque la matriz de rotación presente muchas ventajas al momento de aplicarla, es necesario definir 9 elementos. Existe otros métodos que necesitan la utilización de solo 3 elementos y estos son los ángulos de Euler (Barrientos, Peñin, Balaguer, & Aracil, 2007).

1.3.10 Modelo Cinemático

El modelo cinemático de un robot es el análisis del movimiento del mismo con respecto a un marco de referencia. Existen dos métodos fundamentales, cinemática directa y cinemática inversa y su relación se presenta en la figura 1.2 (Barrientos, Peñin, Balaguer, & Aracil, 2007).

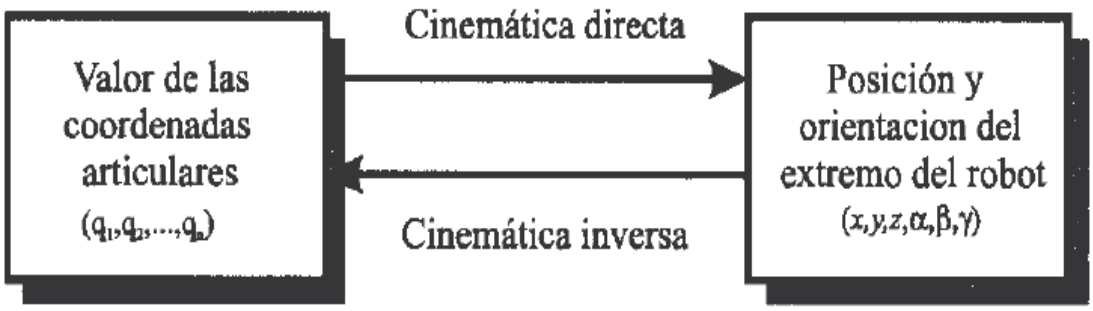


Figura 1.2. Diagrama de relación entre cinemática directa e inversa

Fuente: Fundamentos de robótica, 2007

Elaborado: Barrientos, A., Peñin, L. F., Balaguer, C., & Aracil, R. (2007).

1.3.11 Cinemática Directa

La cinemática directa representa la posición y la orientación de un objeto en el espacio, respecto a una referencia fija, utilizando básicamente algebra vectorial y matricial. Se reduce a una matriz de transformación que relacione la posición y la orientación. Si se tiene un robot de n grados de liberta, n eslabones y n articulaciones. Se puede utilizar transformaciones homogéneas que representen las posiciones y orientaciones entre los distintos eslabones que componen un robot.

Denavit-Hartenberg (D-H) es un método que relaciona eslabones adyacentes. Es posible pasar de un eslabón a otro mediante 4 transformaciones básicas que dependen exclusivamente de las características geométrica del eslabón, escogiendo adecuadamente los sistemas de coordenadas en cada eslabón. Las transformaciones básicas consisten en una sucesión de rotaciones y translaciones que permiten relacionar el sistema de referencia ente del elemento i con el sistema del elemento i-1 (Barrientos, Peñin, Balaguer, & Aracil, 2007).

Tabla 1.2 Parámetros del método de D-H.

$\boldsymbol{a_i}$	Largo del i-ésimo eslabón. Distancia a_i a la longitud de la translación x_i; vector $\boldsymbol{a}_i(0,0,a_i)$.
$\boldsymbol{d_i}$	Articulaciones lineales o prismáticas. También representa el espesor del servomotor. Translación a la longitud de z_{i-1} con valor d_i; vector $\boldsymbol{d}_i(0,0,d_i)$.
$\boldsymbol{\alpha_i}$	Ángulo entre los ejes z_{i-1} y el z_i medido con respecto al eje x_i.
$\boldsymbol{\theta_i}$	Articulaciones rotacionales; representa el ángulo entre los ejes x_{i-1} y x_i medido alrededor del eje z_{i-1}.

Fuente: MATLAB aplicado a Robótica y Mecatrónica, 2012

Elaboración: Cortés, F. (2011)

Las transformaciones se realizan en la secuencia indicada en la ecuación 1.1, debido a la propiedad no conmutativa del producto de matrices.

$$\boldsymbol{A_i} = \boldsymbol{T}(\boldsymbol{z}, \boldsymbol{\theta_i})\boldsymbol{T}(\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{d_i})\boldsymbol{T}(\boldsymbol{a_i}, \boldsymbol{0}, \boldsymbol{0})\boldsymbol{T}(\boldsymbol{x}, \boldsymbol{\alpha_i}) \qquad \text{Ecuación 1.1}$$

En la ecuación 1.2 se realiza el producto entre matrices:

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Ecuación 1.2}$$

Para obtener el modelo cinemático directo, realizar los siguientes pasos:

Enumerar los eslabones, desde el 1 hasta n. Desde la base del robot se inicia la numeración nombrando a este eslabón 0.

Identificar cada una de las articulaciones comenzando desde 1 hasta n.

Localizar el eje de giro de cada una de las articulaciones, los cuales serán los ejes $z_0, z_1, \ldots z_n$ de cada articulación.

Establecer el sistema de referencia cartesiano, siendo el origen la base del robot, armando el eje coordenado haciendo uso de la mano derecha.

Situar x_i en la línea normal común a z_{i-1} y z_i.

Situar y_i formando un sistema dextrógiro con x_i y z_i.

Obtener θ_i como el ángulo que hace x_{i-1} y x_i queden paralelos, girando en torno a z_{i-1}, parámetro variable en articulaciones en revolución.

Obtener d_i como la distancia medida a lo largo de z_{i-1} hasta intersectarse con el eje z_i.

Obtener a_i como la distancia medida a lo largo de x_i hasta el eje x_{i-1}.

Obtener $\propto_i$ como el ángulo existente entre los ejes z_{i-1} y z_i (Reyes Cortés, Matlab aplicado a robótica y mecatrónica, 2012).

1.3.12 Cinemática Inversa

Dada una determinada localización espacial para posicionar y orientar el extremo del robot se busca los valores que deben adoptar las coordenadas articulares mediante cinemática inversa.

El método geométrico permite obtener los valores de las primeras variables articulares las que posicionan el robot. Se utilizan relaciones trigonométricas y geométricas sobre los elementos del robot.

El método de desacoplamiento cinemático permite, para ciertos robots, resolver los primeros grados de libertad, dedicados al posicionamiento, de manera independiente a la resolución de los últimos grados de libertad, dedicados a la orientación. Cada uno de estos dos problemas más simples podrá ser tratado y resuelto por cualquier procedimiento (Barrientos, Peñin, Balaguer, & Aracil, 2007).

1.3.13 Visión artificial

La visión por computadora es un campo de estudio separado de la robótica, y produjo muchos algoritmos útiles para filtrar el ruido, compensar los problemas de iluminación, mejorar las imágenes, extraer formas (Murphy, 2000).

Los componentes más importantes que conforman un sistema de control visual son los que se representan en la Figura 1.4 y que a continuación se detallan dentro de una aplicación genérica de seguimiento y manipulación.

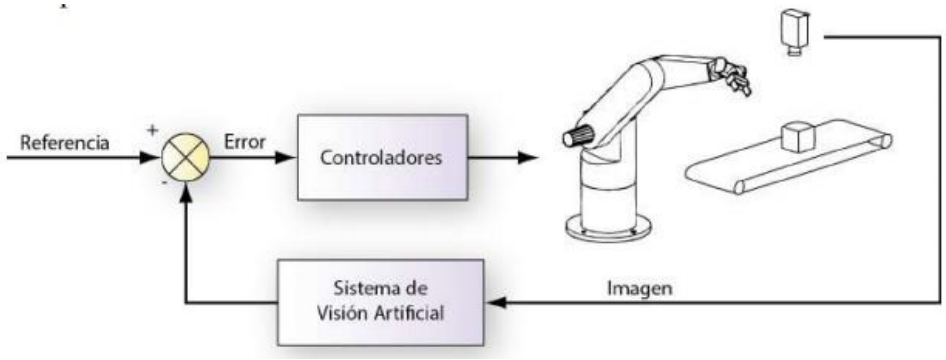


Figura 1.3. Esquema genérico de un sistema de control visual.

Fuente: Conceptos y Métodos en visión por computador, 2016

Elaborado: Alegre, E., Pajares, G., & de la Escalera, A. (2016)

Los principales elementos que aparecen en la Figura 1.3 son los siguientes:

- Referencia. Se trata de la configuración final que se desea que alcance el robot.
- Controlador. Tiene como función realizar el guiado del robot, haciendo uso de la información visual, hasta conseguir alcanzar la referencia. Para el caso indicado en la Figura 1.3, hasta que el robot agarre el objeto.
- Sistema de Visión Artificial. Este bloque representa la realimentación del sistema de control visual y se encarga de realizar la extracción de la información visual requerida para guiar al robot (Alegre, Pajares, & de la Escalera, 2016).

CAPÍTULO 2

2. METODOLOGÍA

2.1 Alternativas de solución

2.1.1 Tarjetas de control

Para el control de un robot, existe una gran variedad de tarjetas de control que se indican en la tabla 2.1:

1) Arduino; microcontrolador que tiene capacidad analógica y, en tiempo real, puede trabajar con casi cualquier tipo de sensor. Posee un lenguaje fácil de programar, una extensa información para su desarrollo y es de precio económico.
2) Raspberry Pi: computadora completamente funcional, multitarea, gran velocidad, amplia memoria, ideal para realizar aplicaciones de “software” y requiere de una variedad de programas para que pueda interactuar con sensores.
3) Orange Pi: computadora multifunción, igual de pequeña que una Raspberry Pi en ciertos detalles y tiene un precio menor a esta.

Tabla 2.1 Características técnicas

	Arduino	**Raspberry Pi**	**Orange Pi**
Precio	23	86	40
Multitarea	No	Si	Si
Voltaje de entrada	7-12V	6 V	5V
Memoria	Limitada	Expandible	Expandible
Sistema operativo	Ninguno	Distribuciones de Linux	Distribuciones de Linux
Entorno de desarrollo integrado	Arduino	Scrath, IDLE, cualquiera con soporte de Linux	Armbium

Fuente: Elaboración propia

2.1.2 Sensores

Para la medición de los ángulos entre dos eslabones, se seleccionó entre los siguientes elementos:

1) Acelerómetro: capaz de medir la aceleración en tres dimensiones (3D). Con él, se puede variar la velocidad durante el trayecto. Los acelerómetros se usan para detectar colisiones, vibraciones, cargas inerciales, aceleraciones, etc.
2) Potenciómetro: componente electrónico similar a los resistores, pero con valor de resistencia variable, lo que permite controlar la intensidad de corriente a lo largo de un circuito conectándolo en paralelo o en serie a la caída de tensión (Pinto, 2012).

2.1.3 Comunicación del computador al Arduino

Para la manipulación del robot, debe existir una comunicación entre el robot y la computadora. La que puede ser de las siguientes maneras:

1) Conexión física: Cable que va directamente desde la tarjeta hasta la computadora. Su inconveniente es que la distancia entre la computadora y el robot es limitada.
2) Conexión inalámbrica Wifi: permite transferir datos del Arduino a la computadora, en un rango de 250 m o 400 m en campo abierto.
3) Conexión inalámbrica Bluetooth permite la transferencia de datos, a 18 m de distancia.

2.2 Descripción y selección de la mejor alternativa

La ponderación para la selección de la mejor alternativa se muestra en la tabla 2.2:

Tabla 2.2 Ponderación para selección

Ponderación para selección	
0 - 0.2	Nada satisfactorio
0.2 - 0.4	Poco satisfactorio
0.4 - 0.6	Indiferente
0.6 - 0.8	Satisfactorio
0.8 - 1	Muy satisfactorio

Fuente: Elaboración propia

Tabla 2.3 Matriz de decisión tarjeta de control

Selección de tarjeta de control				
Requerimiento/ Alternativa	Costo	Características técnicas	Facilidad de uso	Total
	0,3	0,5	0,2	
Arduino	0,9	0,6	0,9	0,75
Orange Pi	0,5	0,7	0,5	0,6
Raspberry	0,3	0,7	0,5	0,54

Fuente: Elaboración propia

La mejor alternativa es el Arduino con la mayor ponderación de 0,75 entre todas las tarjetas, debido a su precio accesible; sus características técnicas cumplen con las necesidades básicas para el proyecto y su programación es muy fácil. Además, existe una gran variedad de información que puede ser usado como guía.

Tabla 2.4 Matriz de decisión sensor

Selección de tarjeta de sensor			
Requerimiento/ Alternativa	Costo	Características técnicas	Total
	0,3	0,3	
Acelerómetro	0,3	0,5	0,44
Potenciómetro	0,9	0,7	0,76

Fuente: Elaboración propia

Se seleccionó el potenciómetro por su bajo costo, linealidad y fácil instalación. Al trabajar con aceleraciones bajas funciona en sus óptimas condiciones.

Los acelerómetros son dispositivos muy sensibles a las vibraciones, presentando ruido en la medición y por lo general se debe filtrar la señal antes de usar.

Tabla 2.5 Matriz de decisión conexión Arduino y computadora

Selección de conexión				
Requerimiento/ Alternativa	Costo	Características técnicas	Facilidad de uso	Total
	0,2	0,5	0,2	
Cable	0,8	0,3	0,2	0,37
Bluetooth	0,6	0,7	0,7	0,68
Wifi	0,4	0,8	0,5	0,63

Fuente: Elaboración propia

La alternativa selecciona es el Bluetooth, con 0,68 de peso, por ser una conexión inalámbrica que funciona hasta una distancia de 10 metros, sin necesidad de internet. La conexión entre el robot y la computadora es sencilla, es un módulo de tamaño reducido con un bajo consumo.

2.3 Diseño conceptual

El brazo robótico consta de articulaciones que son: rotación de base, movimiento base, movimiento de codo, movimiento de muñeca y movimiento de pinza.

Las cinco articulaciones son giratorias, la articulación de la pinza y rotación de base giran perpendicularmente a las articulaciones de movimiento base, codo y muñeca. Cada eslabón es manipulado por un actuador eléctrico (motor DC) localizado en cada articulación.

Computador con software Matlab

Cámara web Genius Facecam	
Arduino Mega 2560	
Motor Shield	
L298N	
Potenciómetro	
Bluetooth HC-05	

Brazo robótico OWI-535

Figura 2.1. Descripción general del brazo robótico automatizado

Fuente: Elaboración propia

El kit de brazo robótico OWI-535 se automatiza como se muestra en la figura 2.1. Reconoce y desplaza objetos utilizando el software Matlab, el cual controlará las velocidades de las articulaciones individuales del robot. La transmisión de datos entre el microcontrolador Arduino y la computadora es a través del módulo bluetooth, los cinco motores se controlan con el Motor Shield y L298N y en cada articulación se instala un potenciómetro.

2.4 Metodología de diseño a seguir

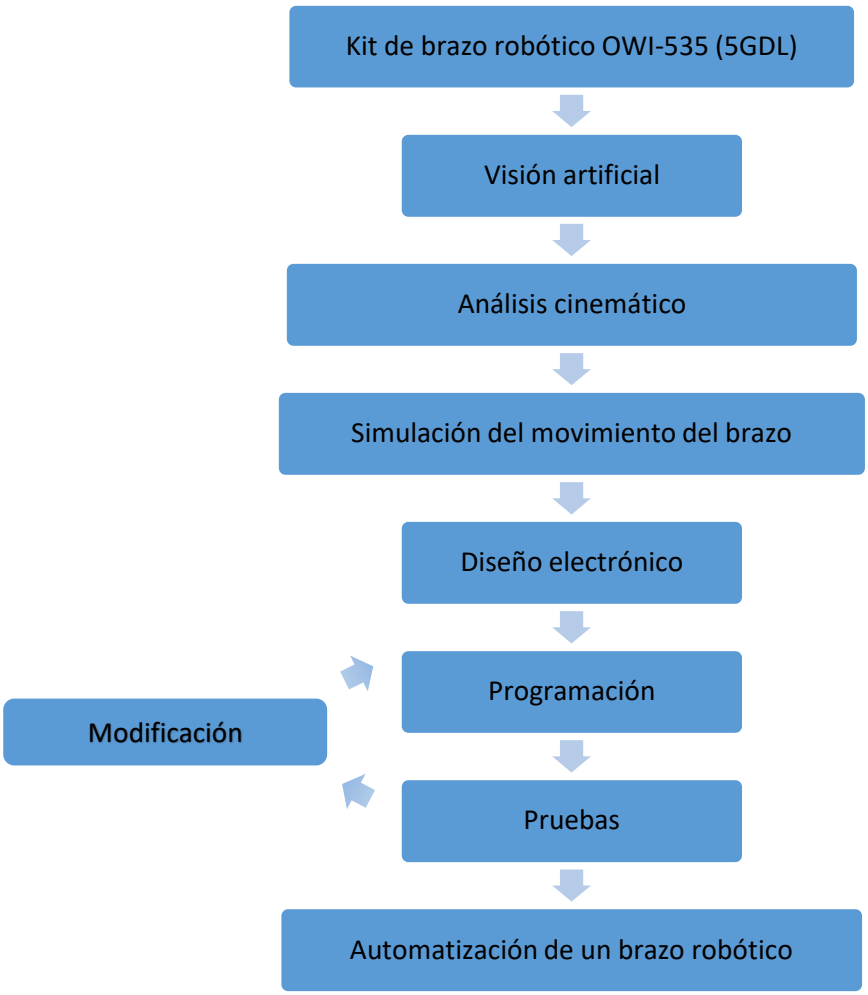


Figura 2.2. Descripción general de la metodología de diseño

Fuente: Escandón y Trujillo, 2017

Elaboración propia

El proyecto inicia con la adquisición del kit de brazo robótico (OWI-535) de cinco grados de libertad, las especificaciones técnicas indican las

restricciones de movimiento de las articulaciones. Se identifica y localiza la posición de un objeto implementando visión artificial con los “Toolboxes” Procesamiento de imagen y Adquisición de imagen en Matlab, para desplazar la pinza del robot a esta ubicación.

En Matlab se calcula los valores de los ángulos articulares del robot para alcanzar la orientación y posición deseada de la herramienta, conocida la posición del extremo del robot (pinza) referido a la base mediante el método cinemático inverso. Diversos movimientos del robot se simulan con el “Toolbox” de Robótica Peter Corke en Matlab.

Para el diseño electrónico se selecciona el microcontrolador Arduino Mega 2560, el Motor Shield L293D y L298N para el control de velocidad y dirección de giro de los motores, potenciómetro para medir el ángulo de desplazamiento del eslabón. Al ejecutar el código en Matlab, se envía una señal al Arduino por conexión bluetooth, que activa los motores hasta los ángulos calculados en el análisis cinemático. Se modifica el código en Matlab al no conseguir el resultado requerido en las pruebas.

2.4.1 Visión artificial

Visión artificial permite entender las características de una escena a través de imágenes digitales.

En este proyecto se reconoce y localiza un objeto situado en una mesa de trabajo, con la adquisición, procesamiento y análisis de imagen. La imagen obtenida por la cámara web Facecam se almacena en una matriz con valores en píxeles de la cual se extrae el color del objeto a identificar, cada píxel representa la intensidad de rojo, verde y azul.

La cámara se calibra con una hoja de papel que contiene cuadros de dimensiones conocidas en milímetros, colocada en el espacio trabajo del robot. Se determina la relación de píxeles por milímetros empleando la ecuación 2.1.

$$\text{mm} = \text{píxeles} * \frac{\text{longitud del cuadro en milímetros}}{\text{longitud del cuadro en píxeles}} \qquad \text{Ecuación 2.1}$$

El sistema de referencia de la cámara está situado en la parte superior izquierda de la imagen como se presenta en la figura 2.3. Se aplica un cambio de coordenadas de un objeto visto desde la cámara a coordenadas medidas desde la base del robot, dado por las ecuaciones 2.2 y 2.3.

$$X_{robot} = X_{cámara}[\text{mm}] - 230 \qquad \text{Ecuación 2.2}$$

$$Y_{robot} = -Y_{cámara}[\text{mm}] + 345 \qquad \text{Ecuación 2.3}$$

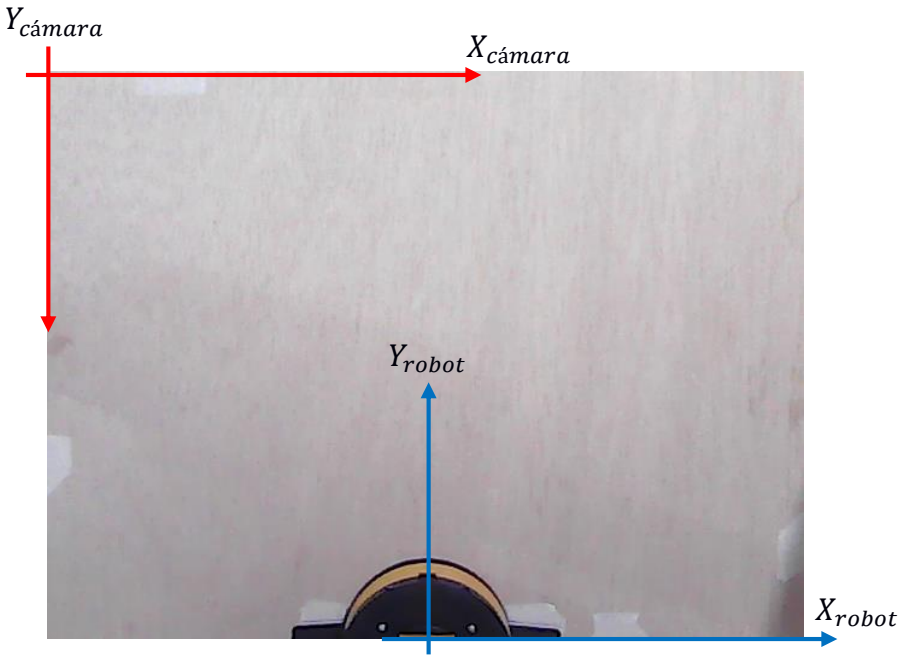


Figura 2.3. Ilustración a) sistema de referencia cámara y b) robot

Fuente: Elaboración propia

El objeto de color es identificado y señalado con un cuadro indicativo de bordes rojo, mostrando la coordenada X e Y a trasladar el extremo final del robot.

2.4.2 Análisis cinemático directo

La resolución del análisis cinemático directo describe la localización de las articulaciones con respecto a un plano de referencia. Se reduce con una

transformación matricial que relaciona las coordenadas de la unión con un marco de referencia utilizando el método de D-H.

2.4.3 Análisis cinemático inverso

Dependiendo de la configuración del brazo robótico, se realiza un estudió analítico del movimiento del brazo con respecto a un sistema de coordenadas de referencia fijo, sin considerar las fuerzas o momentos que lo originan. Relaciona la posición y orientación del extremo final del robot con los valores angulares de las articulaciones. Es decir, conocida la posición cartesiana de un objeto, se realiza la cinemática inversa del brazo robótico que determina la configuración que el robot debe adoptar para alcanzar dicho objeto con la pinza.

El análisis cinemático del robot se efectúa en el plano V de la figura 2.4.

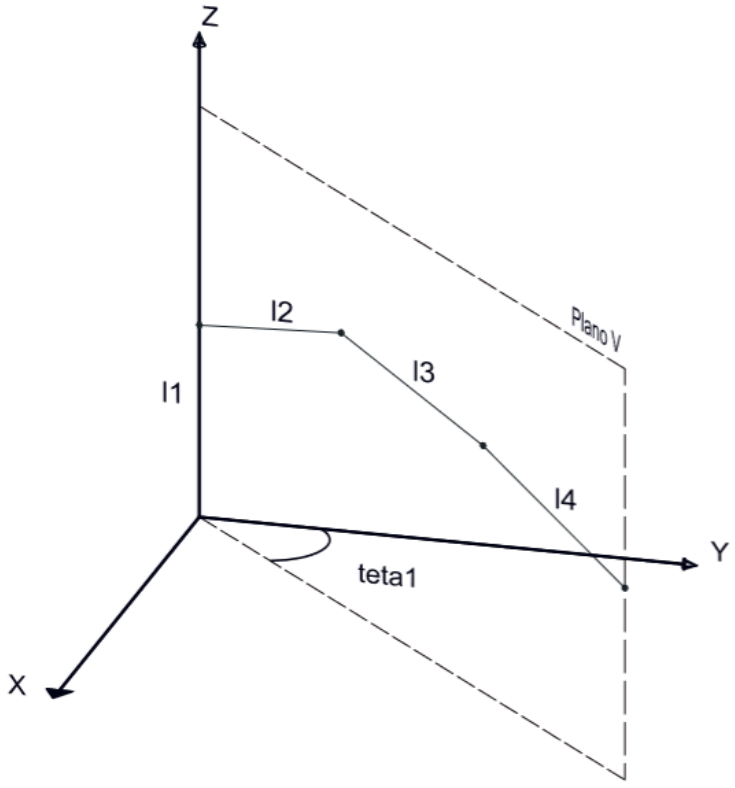


Figura 2.4. Sistema de referencia visión artificial y Plano V del análisis cinemático.

Fuente: Elaboración propia

Se realiza el análisis cinemático en dos partes, calculando los cuatro ángulos de rotación que se proporcionan al robot OWI-535 para alcanzar la posición del objeto detectado por visión artificial.

Articulación 1: Rotación de base, asociado a θ_1.
Articulación 2: Movimiento de base, asociado a θ_2.
Articulación 3: Codo, asociado a θ_3.
Articulación 4: Muñeca, asociado a θ_4.

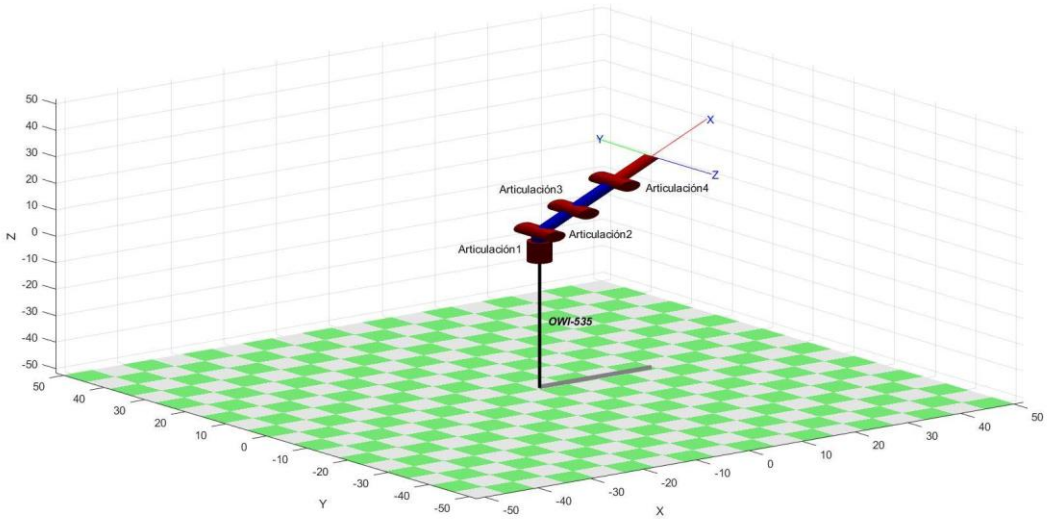


Figura 2.5. Ilustración del brazo robótico señalando cada articulación

Fuente: Elaboración propia

Parte 1:

Las coordenadas del objeto situado en el plano x-y, se obtuvo por visión artificial que son los valores x e y de la ecuación 2.4. Se calcula el ángulo de rotación de la base del robot (θ_1) de la figura 2.4 por el método geométrico.

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) \qquad \text{Ecuación 2.4}$$

Parte 2:

El cálculo de los tres ángulos restantes ($\theta_2, \theta_3, \theta_4$) se plantea en el plano V de la figura 2.4 y la solución de la cinemática inversa se convierte en un problema matemático.

Solución matemática

Empleando el método de D-H, expresar las matrices de transformación de las articulaciones 2, 3 y 4 como se muestra en las ecuaciones 2.5, 2.6 y 2,7 respectivamente y la multiplicación de estas tres matrices en la ecuación 2.8 para obtener el modelo cinemático completo.

$$A_1^0 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 90xcos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 90x\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Ecuación 2.5}$$

$$A_2^1 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 110xcos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 110x\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Ecuación 2.6}$$

$$A_3^2 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & 110xcos(\theta_4) \\ \sin(\theta_4) & \cos(\theta_4) & 0 & 110x\sin(\theta_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Ecuación 2.7}$$

$$A_3^0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Ecuación 2.8}$$

La multiplicación de matrices se iguala a la transformada de cada matriz para generar las ecuaciones que contienen los ángulos a determinar, como se muestra a continuación en la ecuación 2.9

.

$$A_3^0 = A_1^0 * A_2^1 * A_3^2 \quad \text{Ecuación 2.9}$$

Para obtener las primeras ecuaciones, se pasa la matriz A_3^2 al lado izquierdo como indica la ecuación 2.10.

$$A_3^0 * (A_3^2)^{-1} = A_1^0 * A_2^1 \qquad \text{Ecuación 2.10}$$

Se obtiene dos ecuaciones que son:

$$p_x - 110 * n_x = 90 * \cos(\theta_2) + 110 * \cos(\theta_2) * cos(\theta_3) - 110 * \sin(\theta_2) * \sin(\theta_3)$$

Ecuación 2.11

$$p_y - 110 * n_y = 90 * \sin(\theta_2) + 110 * \cos(\theta_2) * sin(\theta_3) - 110 * \cos(\theta_2) * \sin(\theta_3)$$

Ecuación 2.12

Teniendo $\theta_2\ y\ \theta_3$ como variables desconocidas, utilizamos las ecuaciones 2.11 y 2.12 para determinar los valores de estos ángulos.
Para obtener el ángulo de giro de la muñeca del robot (θ_4), realizamos la siguiente igualdad de matrices en la ecuación 2.13.

$$A_3^0 * (A_3^2)^{-1} * (A_2^1)^{-1} = A_1^0 \qquad \text{Ecuación 2.13}$$

Dando como resultado las siguientes ecuaciones:

$$p_x - 110 * n_x - 110 * n_x * \cos(\theta_4) + 110 * o_x * sin(\theta_4) = 90 * \cos(\theta_2)$$

Ecuación 2.14

$$p_y - 110 * n_y - 110 * n_y * \cos(\theta_4) + 110 * o_y * sin(\theta_4) = 90 * \sin(\theta_2)$$

Ecuación 2.15

Las ecuaciones 2.14 y 2.15 tienen solo una incógnita que es θ_4. Utilizando cualquiera de las dos ecuaciones se puede obtener el ángulo de la articulación cuatro. (Fu, Gonzales, & Lee, 1987)

2.4.4 Diseño electrónico

Matlab cuenta con una librería de compatibilidad y comunicación con Arduino, se seleccionó el microcontrolador Arduino Mega 2560 por su mayor cantidad de entradas analógicas y digitales. Previo al uso de la tarjeta Arduino en Matlab, ingresar AFMotor a la librería de Arduino y cargar el código proyecto Arduino en el apéndice B desde el software Arduino.

La comunicación del software con el robot es de forma inalámbrica. Se utiliza el Bluetooth HC-05 con un alcance de funcionamiento de 10 metros, se puede programar en modo esclavo o maestro. El modo esclavo permite recibir comunicaciones mientras que el modo maestro recibe e inicia, en este proyecto se utilizó el Bluetooth modo esclavo. Con el “Toolbox” Control de instrumentos el módulo HC-05 se puede conectar a Matlab, para permitir el envío y recepción de datos desde la tarjeta Arduino al computador.

Para el control de velocidad y giro de los 5 motores que incluye OWI-535, se utiliza un Motor Shield y un L298N, ambos compatibles con Arduino. El Motor Shield permite controlar 4 motores DC ubicados en las articulaciones de la rotación de base, movimiento de base, codo y muñeca. El L298N controla el motor DC de la pinza. Parte de automatizar el brazo robótico es instalar sensores, se añade un pulsador final de carrera en la pinza del brazo, se cierra la conexión eléctrica cuando el objeto entra en contacto con el microinterruptor deteniendo el motor de la pinza.

En las cuatro articulaciones del robot se colocó potenciómetros para medir la variación de voltaje y traducirlo en ángulos. Estos se emplean para registrar el giro de cada eslabón y llevar el robot hasta la posición final requerida, para alcanzar el objeto encontrado por visión artificial. El resultado final del diseño electrónico se muestra en la figura 2.6.

Figura 2.6. Brazo robótico automatizado

Fuente: Elaboración propia

2.4.5 Simulación

Mediante simulación se predice el movimiento del brazo robótico sin usar un prototipo físico, para comprender el movimiento del mecanismo.

El objetivo de la simulación es conocer la relación entre las coordenadas articulares y la ubicación del brazo robótico utilizando resultados numéricos y gráficos. Se busca que el robot realice la menor cantidad de movimientos al trasladar el objeto de un lugar a otro, evitando colisiones provocadas por obstáculos. Una representación sencilla para simular los movimientos del brazo robótico se consigue en Matlab por medio de una representación gráfica.

Las funciones mostradas más adelante facilitan el trabajo con robots. En Matlab se implementa el “Toolbox” de Robótica Peter Corke el cual contiene

funciones que resuelven de manera simple el análisis cinemático inverso y la simulación del robot.

La descripción de las funciones se obtuvo de Matlab con uso del comando “help” seguido del nombre de la función.
>> help nombre_de_función

>>fkine
Obtiene la cinemática directa del robot, la matriz de transformación del efector final con respecto a un inicio de coordenadas que se fija en un punto determinado.

>>jtraj
Calcula una trayectoria espacial entre dos articulaciones coordinadas conjuntas.

>>link
Construye un objeto de enlace de robot. El objeto contiene parámetros cinemáticos, así como parámetros del actuador.
>>robot / plot
Crea un robot representado por una simple polilínea. Animación gráfica del robot, muestra una representación gráfica del robot dada la información cinemática del robot.

>>SerialLink
Una clase concreta que representa un robot de tipo de brazo de enlace en serie. El mecanismo se describe utilizando los parámetros de D-H, un conjunto por a

CAPÍTULO 3

3. RESULTADOS

La programación de este proyecto se realizó con el software Matlab y haciendo uso de los “Toolbox”: Procesamiento de imagen, Peter Corke y Soporte de MATLAB para Arduino.

3.1 Visión artificial

La cámara web Facecam realizó la adquisición de una imagen con dimensiones 640x480 píxeles.

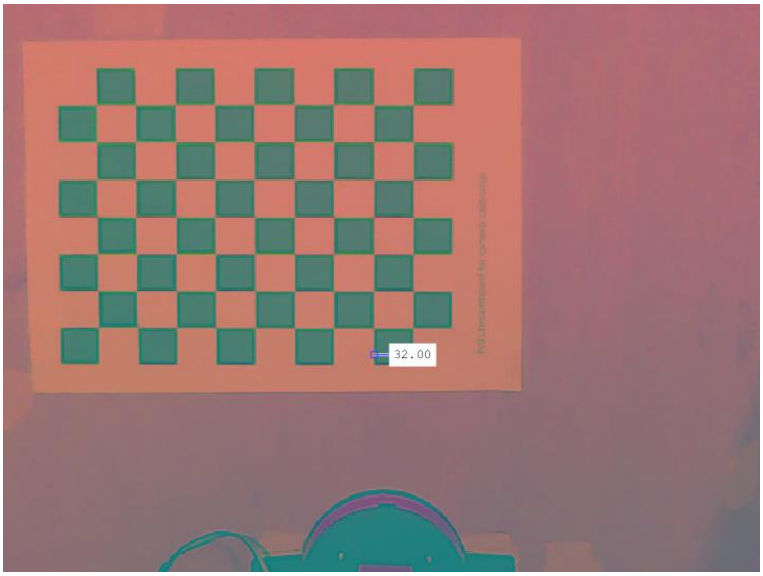


Figura 3.1. Medición distancia en píxeles de un cuadro

Fuente: Escandón y Trujillo, 2017

Elaboración propia

Se utilizó el algoritmo distancia en píxeles en el apéndice B y se obtuvo que la distancia del cuadro es de 32 píxeles como se presenta en la figura 3.1. Se conoce que los cuadros tienen 23 milímetros de lado. Con estos dos datos se planteó el factor de conversión píxeles a milímetros, mostrado en la ecuación 3.1.

$$\text{mm} = \text{píxeles} * \frac{23}{32} \qquad \text{Ecuación (3.1)}$$

Con el reconocimiento de color, el robot localizó la ubicación exacta del objeto en un plano de dos dimensiones (x,y) en unidades de milímetro.
Se realizó pruebas de reconocimiento de objetos de color rojo, verde y azul. Se utilizó el algoritmo visión artificial en el apéndice B.

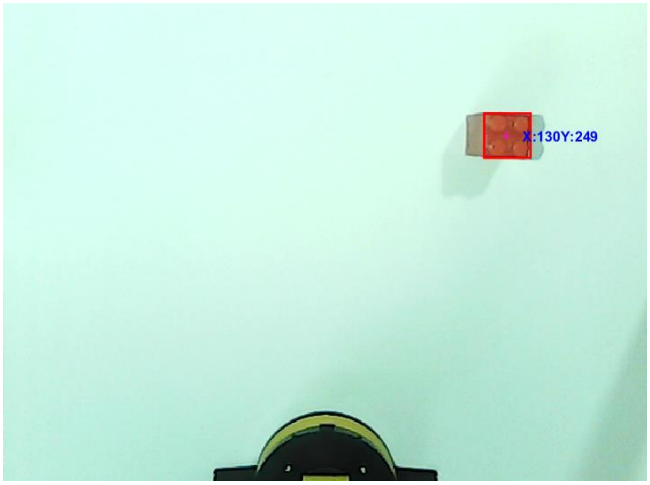


Figura 3.2. Coordenadas de un objeto de color rojo

Fuente: Elaboración propia

Prueba 1: En la figura 3.2, se detectó un objeto de color rojo de posición x=130mm, y=249mm.

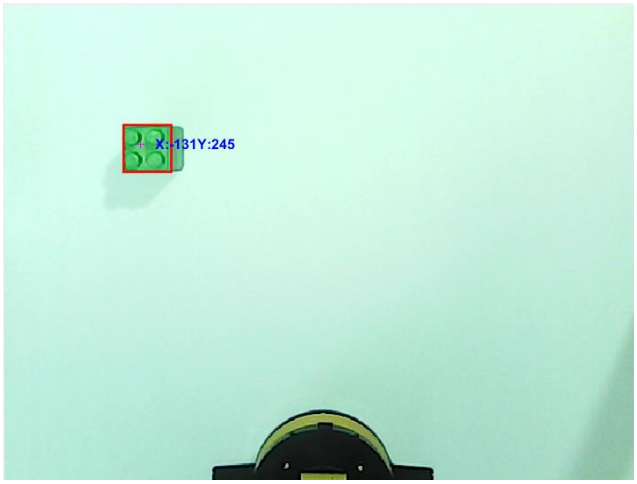


Figura 3.3. Coordenadas de un objeto de color verde

Fuente: Elaboración propia

Prueba 2: La figura 3.3 mostró la detección de un objeto de color verde con posición x=-131mm, y=245mm.

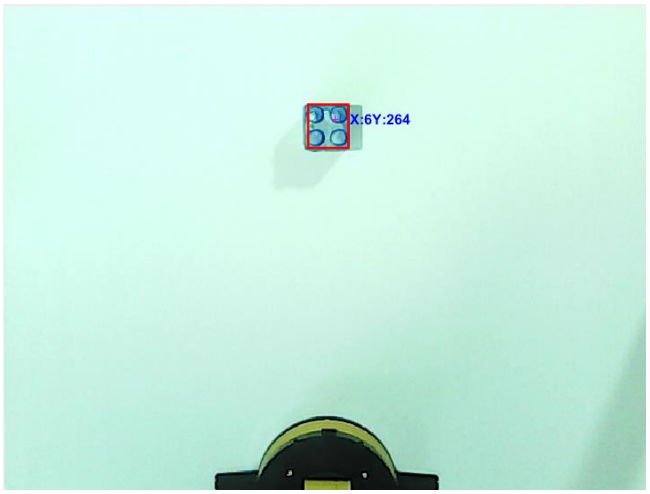


Figura 3.4. Coordenadas de un objeto de color azul

Fuente: Elaboración propia

Prueba 3: La figura 3.4, indicó un objeto de color azul con posición x=6mm, y=264mm.

3.2 Análisis cinemático directo

El análisis cinemático directo se restringió por el diseño del robot.

En la tabla 3.1 se presenta los parámetros utilizados en el método de D-H, con los ejes de referencia asignados a cada articulación del brazo indicados en la figura 3.5. El análisis cinemático completo se obtuvo con la multiplicación de la matriz de transformación de cada articulación del robot.

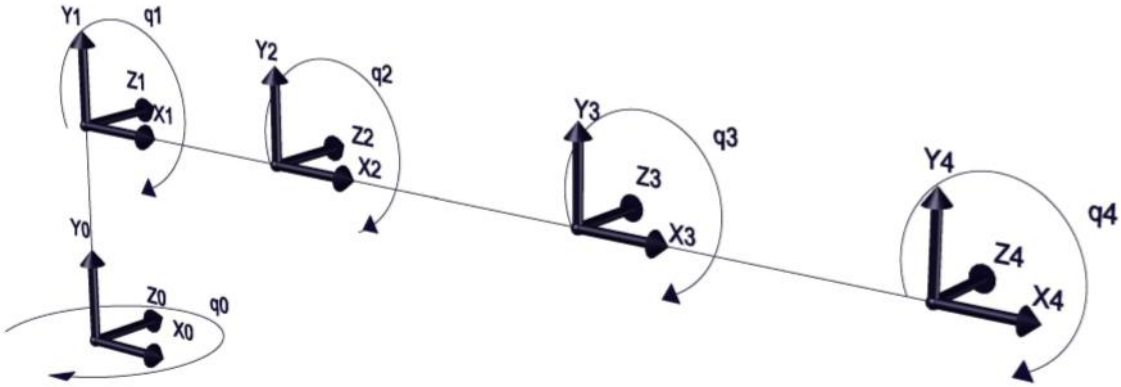


Figura 3.5. Sistema de referencia usando la convención de D-H

Fuente: Elaboración propia

Tabla 3.1 Parámetro de D-H del OWI-535

Articulación	θ_i	d_i [mm]	a_i [mm]	α_i
1	θ_1	70	0	90
2	θ_2	0	90	0
3	θ_3	0	110	0
4	θ_4	0	110	0

Fuente: Elaboración propia

Como se explicó en el capítulo anterior, se determinó los parámetros de D-H del brazo robótico como parte fundamental para análisis cinemático directo e inverso, y el empleo de estos parámetros en la simulación.

3.3 Análisis cinemático inverso

Con la representación del modelo directo se formó el modelo cinemático inverso. La solución de este análisis se ingresó en el algoritmo proyecto Matlab en el apéndice B.

Se manipuló el brazo robótico con la implementación de sensores y controladores de motor. Se encendió los motores DC y los potenciómetros entregaron la lectura analógica de su variación, registrando el cambio de los ángulos de las articulaciones. Se detuvo los motores cuando el robot llegó a los ángulos finales requeridos. La posición final del brazo robótico OWI-535

fue igual a la simulación, de esta manera se confirmó que el comportamiento del robot es correcto. Los objetos reconocidos por visión artificial fueron capturados por la pinza del robot y desplazados hasta una posición establecida.

3.4 Espacio de trabajo

El espacio de trabajo se generó con el análisis cinemático directo y se consideró de dos maneras:

1) Espacio de trabajo OWI-535, enseñó todos los puntos alcanzados por el efector final del brazo robótico condicionado por las especificaciones técnicas de fábrica.

2) Espacio de trabajo del proyecto, presentó los puntos alcanzados con las limitantes físicas que posee OWI-535 posterior a la instalación de sensores.

Las gráficas expuestas a continuación se ejecutaron con los algoritmos área de trabajo OWI-535 y área de trabajo del proyecto en el apéndice B.

3.4.1 Espacio de trabajo OWI-535

En la figura 3.6a está OWI 535 simulado en Matlab, en 3.6b y 3.7 el espacio de trabajo en vista lateral y superior respectivamente que accede el brazo robótico considerando las especificaciones de técnicas.

Los puntos del espacio de trabajo se generaron a partir de un intervalo de 10 grados en cada ángulo, optimizando la visualización del mismo y disminuyendo el cálculo computacional que presenta.

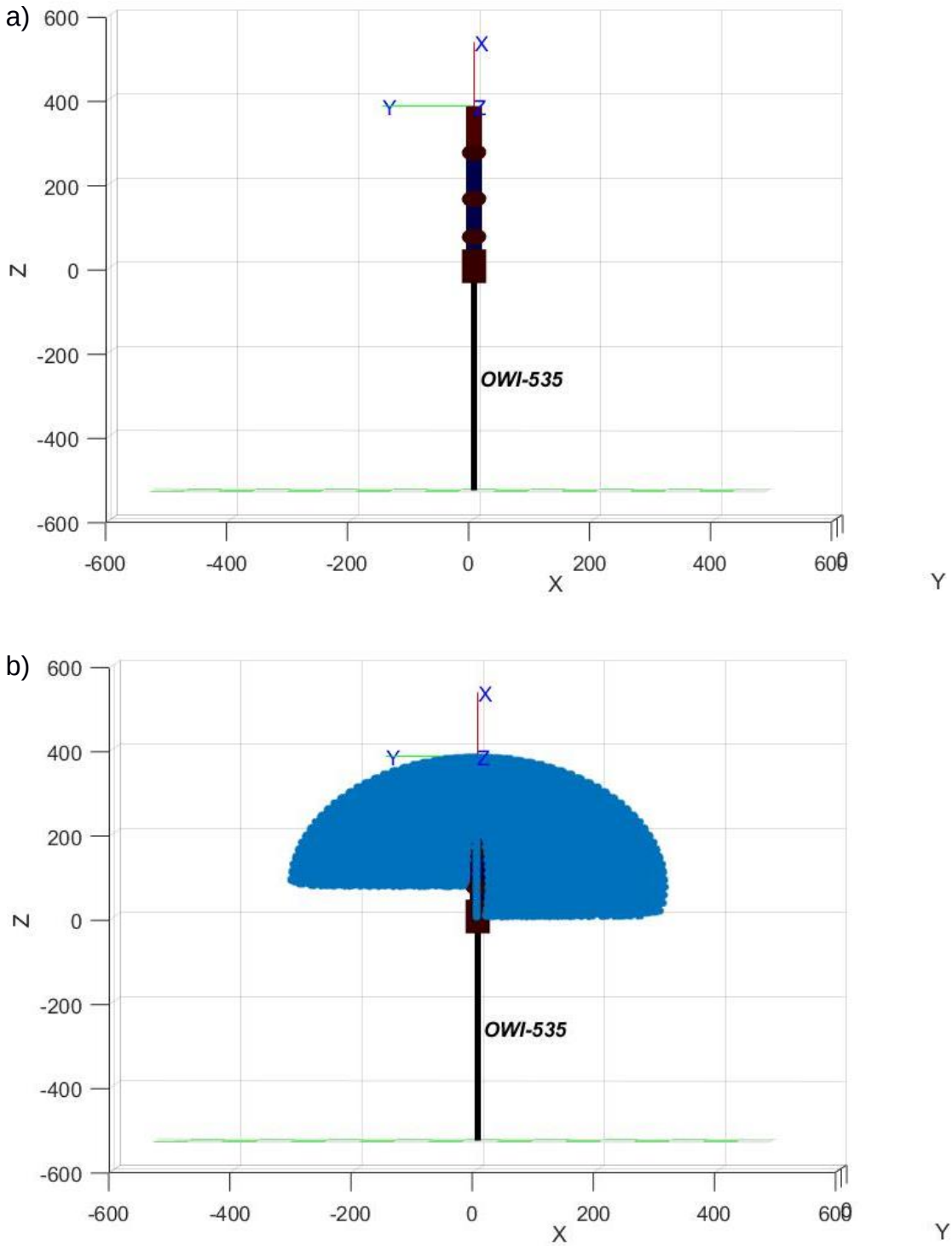


Figura 3.6. Vista lateral del a) brazo robótico y b) espacio de trabajo OWI-535

Fuente: Elaboración propia

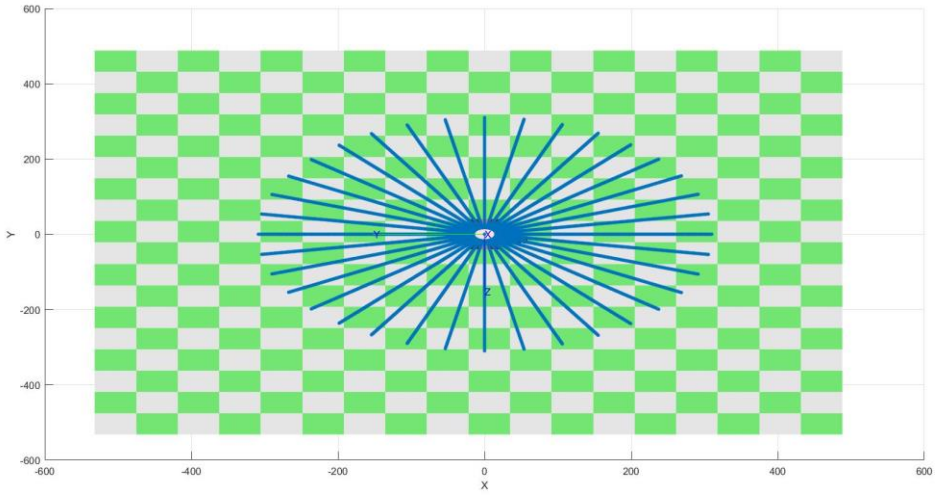


Figura 3.7. Vista superior del espacio de trabajo OWI-535

Fuente: Elaboración propia

La figura 3.8 es una vista lateral del espacio de trabajo de OWI-535 con el ángulo de la base de rotación (θ_1) igual a cero.

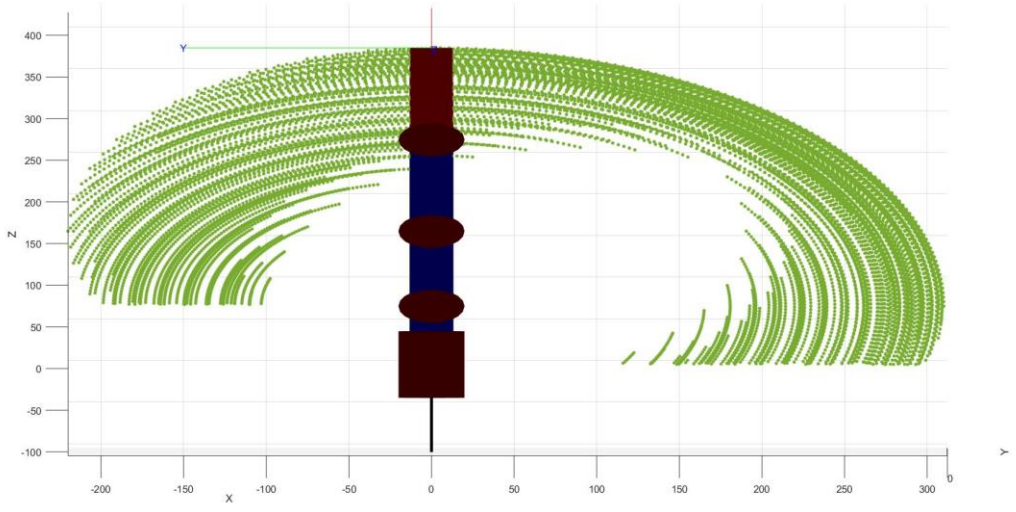


Figura 3.8. Vista lateral espacio de trabajo con $\theta_1 = 0$

Fuente: Elaboración propia

3.4.2 Espacio de trabajo del proyecto

El espacio de trabajo del brazo robótico disminuyó, en comparación al espacio con restricciones de fábrica, a causa de modificaciones aplicadas a OWI-535 indicadas en anexos apéndice C, que afectan directamente a los ángulos rotación de base (θ_1) y movimiento de base (θ_2) dando como resultado:

$$\theta_1 = -25° \; a \; 25°$$
$$\theta_2 = 0° \; a \; 90°$$

El espacio de trabajo del proyecto se visualiza en las figuras 3.9, 3.10 y 3.11.

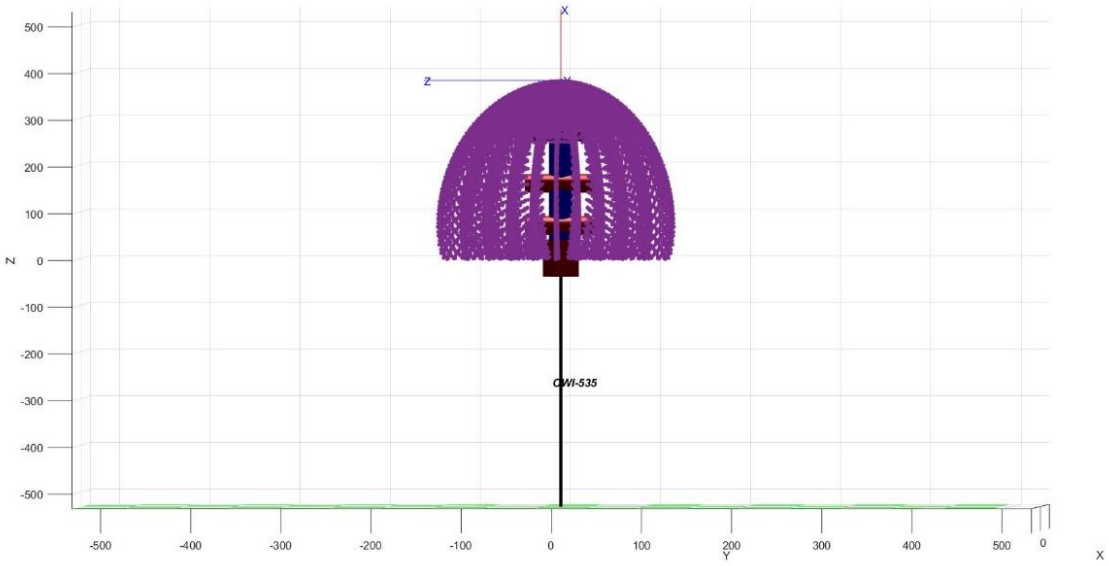


Figura 3.9. Vista frontal del espacio de trabajo del proyecto

Fuente: Elaboración propia

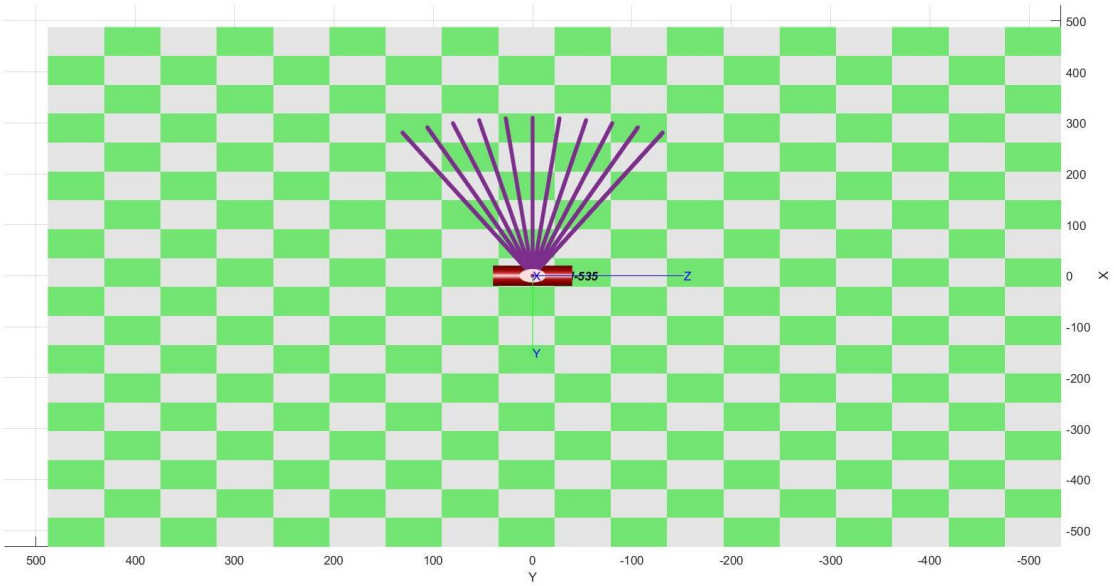


Figura 3.10. Vista superior del espacio de trabajo del proyecto

Fuente: Elaboración propia

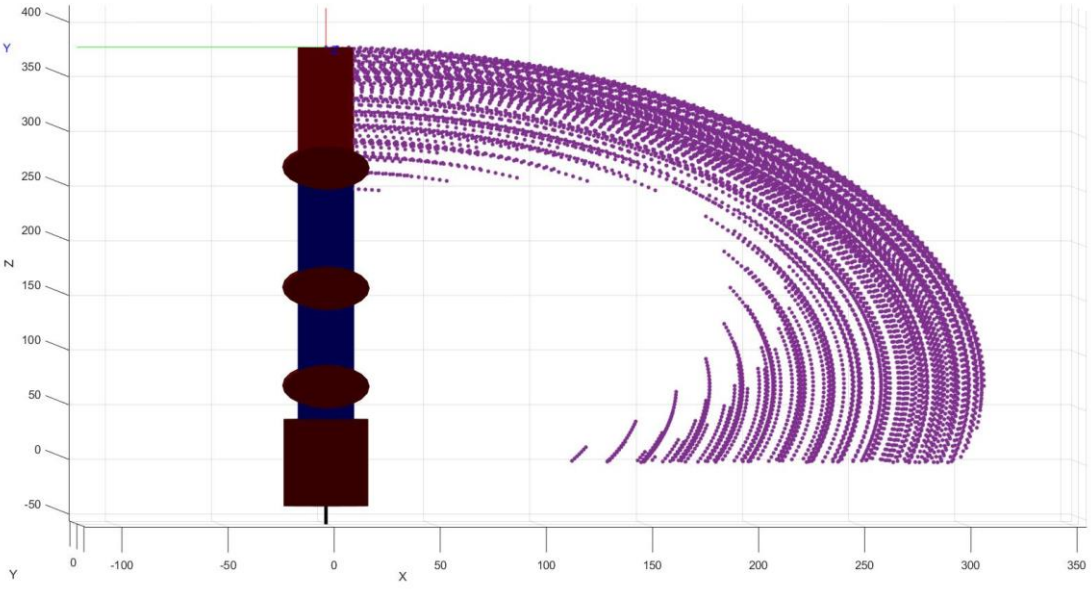


Figura 3.11. Vita lateral del espacio de trabajo del proyecto

Fuente: Elaboración propia

Las coordenadas máximas y mínimas en el centro ($\theta_1 = 0°$) y extremos del espacio de trabajo ($\theta_1 = 25° \; y - 25°$) que alcanza OWI-535 junto con la configuración de ángulos se detallan en la tabla 3.2.

Tabla 3.2 Puntos máximos, mínimos y ángulos de articulaciones en OWI-535

Puntos máximos [mm]			Ángulos de articulaciones [grados]			
x	y	z	θ_1	θ_2	θ_3	θ_4
0	0	380	0	90	0	0
310	0	70	0	0	0	0
0	0	380	25	90	0	0
281	131	70	25	0	0	0
0	0	380	-25	90	0	0
281	-131	70	-25	0	0	0
Puntos mínimos [mm]			θ_1	θ_2	θ_3	θ_4
107	0	0.16	0	88	-115	-60
97	45	0.16	25	88	-115	-60
97	-45	0.16	-25	88	-115	-60

Fuente: Elaboración propia

3.5 Simulación

Con el resultado de los ángulos del análisis cinemático inverso, se visualizó el comportamiento y movimientos del brazo robótico. Se comparó la simulación con los movimientos rotacionales del brazo robótico y se determinó que no existe problema en la programación del mismo. El entorno de la simulación se observó en la figura 3.12.

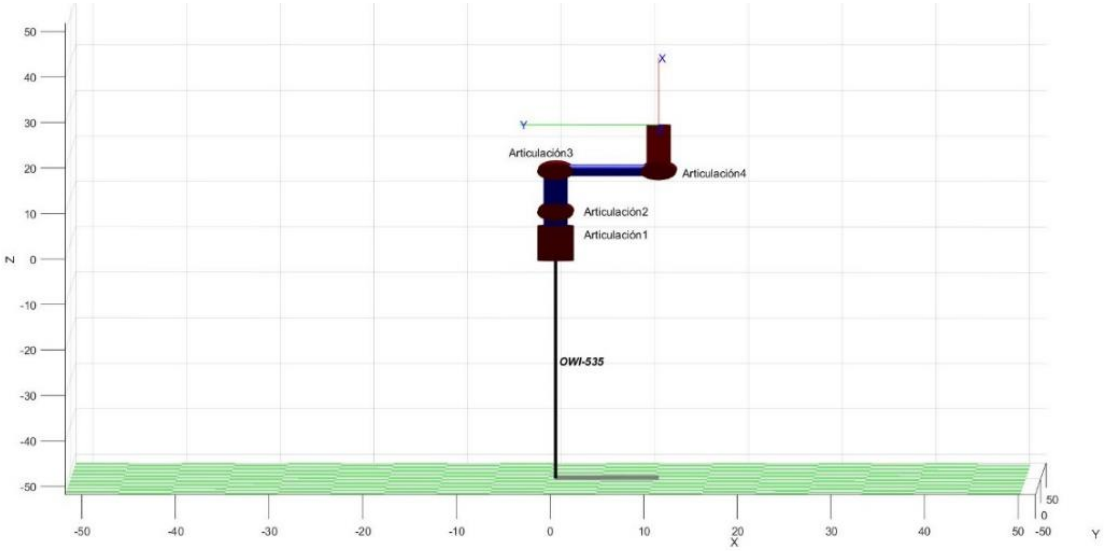


Figura 3.12. Simulación de brazo robótico

Fuente: Elaboración propia

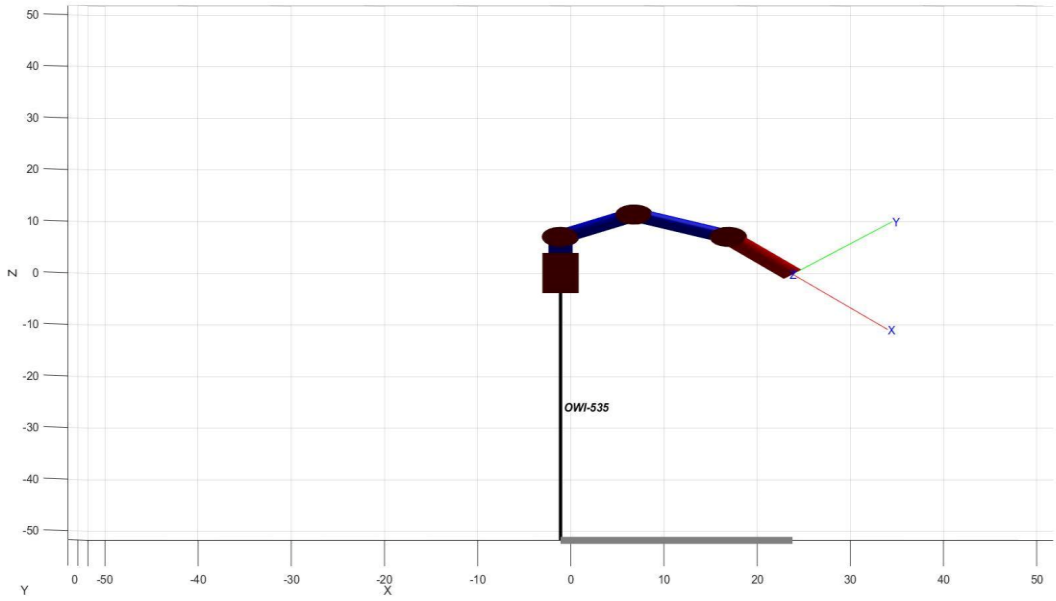
La simulación con el software Matlab se muestra en la figura 3.13a y en 3.13b la posición de OWI-535 con los ángulos tabulados en la tabla 3.3 obtenidos del análisis cinemático inverso.

Tabla 3.3 Ejemplo movimiento OWI-535

θ_i	θ [grados]
θ_1	0
θ_2	28
θ_3	-52
θ_4	-23

Fuente: Elaboración propia

a)



b)

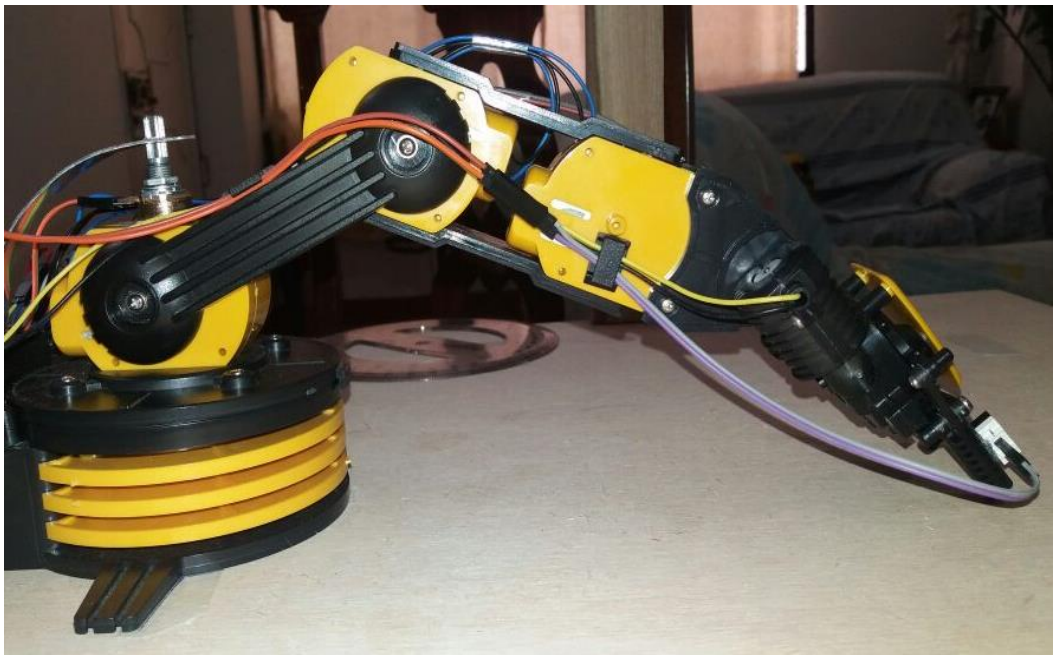


Figura 3.13. Comparación brazo robótico a) simulación y b) movimiento real

Fuente: Elaboración propia

3.6 Análisis de costos

En la tabla 3.4 se especifica los recursos necesarios para lograr automatizar el brazo robótico OWI-535. Se detalla la descripción, cantidad y los rubros de cada elemento adquirido.

Tabla 3.4 Detalle de recursos empleados en el proyecto

Descripción	Cantidad	Costo unitario	Costo
Arduino Mega 2560	1	$ 22,00	$ 22,00
Motor Shield	1	$ 8,50	$ 8,50
Cámara web Genius	1	$ 22,00	$ 22,00
L298N	1	$ 5,00	$ 5,00
Potenciómetro B10K	4	$ 0,50	$ 2,00
Bluetooth HC-05	1	$ 13,00	$ 13,00
Resistencia 1 Kohm	1	$ 0,10	$ 0,10
Resistencia 3 Kohm	2	$ 0,10	$ 0,20
Pulsador fin de carrera	2	$ 0,15	$ 0,30
Soporte de madera	1	$ 20,00	$ 20,00
Jumpers variados	62	$ 0,10	$ 6,20
		Total	$ 99,30

Fuente: Elaboración propia

El análisis de los costos relacionados para el desarrollo del proyecto tuvo un valor total de $99,30.

OWI-535 se convirtió en un brazo robótico autónomo a un bajo costo, motivo por el cual la reproducción del proyecto a mayor escala es viable, con un desarrollo tecnológico que permite sustituir la mano de obra por robots autónomos que realizan tareas repetitivas en menor tiempo y sencillas de especificar en un algoritmo.

CAPÍTULO 4

4. DISCUSIÓN Y CONCLUSIONES

4.1 Conclusiones

El brazo robótico OWI-535 de cinco grados de libertad ejecuta movimientos de manera autónoma a través del análisis cinemático inverso en Matlab, alcanzando objetos en el espacio de trabajo limitado, por las modificaciones realizadas.

El reconocimiento de objetos por color funcionó correctamente empleando visión artificial, entregando la ubicación de los mismos que posterior se utilizó para el análisis cinemático inverso del robot. Se observó la correcta recolección y traslado de objetos de color rojo, verde y azul a una coordenada objetivo.

A partir del diseño del brazo robótico se determinó los parámetros de D-H, como parte fundamental para análisis cinemático directo e inverso.
La resolución del análisis cinemático inverso se separó en dos partes debido a la cadena cinemática del brazo. La primera parte calculó el ángulo de la base de rotación empleando visión artificial y la segunda los tres ángulos restantes como un problema matemático.

La simulación se realizó con la librería de robótica Peter Corke en Matlab, esencial para verificar que el movimiento del robot sea el adecuado. Se mostró la posición y orientación que tomó el robot para llegar al punto definido por visión artificial. A partir de eso, se observó que el movimiento efectuado por el brazo robótico no presentó errores y trasladó objetos de manera correcta.

La gráfica del espacio de trabajo mostró los puntos a los que puede llegar el brazo, presentando los puntos máximos y mínimos. Datos que sirvieron para diseñar el soporte de la cámara web Facecam y las dimensiones de la mesa de trabajo. El espacio de trabajo del robot para este proyecto disminuyó el

rango de giro de la rotación de la base (θ_1) de 270° a 50° y el movimiento de base (θ_2) de 180° a 90° por limitaciones físicas al instalar sensores que registren el cambio de posición, sin embargo, es satisfactorio para desplazar objetos a distintas posiciones.

4.2 Recomendaciones

Para la adquisición de datos con la cámara web se debe emplear una iluminación óptima evitando errores al interpretar las imágenes.

Los datos entregados por visión artificial son en píxeles, calcular el factor de conversión de píxeles a milímetros. De esta forma se obtiene la lectura real de las distancias.

Existen diferentes métodos para resolver el análisis cinemático inverso y se debe seleccionar el adecuado a la cadena cinemática del brazo robótico.

Utilizar la convención de D-H para encontrar los parámetros y obtener el modelo cinemático apropiado.

Comparar la simulación con el movimiento de las articulaciones del robot, si este presenta desigualdades, existe un problema, revisar las conexiones pueden estar defectuosas.

Emparejar la computadora con el módulo HC-05 configurado en modo esclavo, para que la comunicación entre dispositivos sea maestro-esclavo debido a que la comunicación maestro-maestro o esclavo-esclavo es errónea. Recordar que la comunicación tanto en esclavo como maestro es bidireccional y velocidad en baudios de configuración del bluetooth debe ser la misma de la tarjeta Arduino para poder comunicarse.

BIBLIOGRAFÍA

Alegre, E., Pajares, G., & de la Escalera, A. (2016). *Conceptos y Métodos en visión por computador.* Barcelona: Grupo de Visión del Comite Español de Automática (CEA).

Barrientos, A., Peñin, L. F., Balaguer, C., & Aracil, R. (2007). *Fundamentos de robótica.* Madrid: Mc Graw Hill.

Fu, K., Gonzales, R., & Lee, C. (1987). *Robotics: Control, Sesing, Vision, and Intelligence.* Michigan: McGraw-Hill.

Murphy, R. R. (2000). *Introduction to AI robotics.* Cambridge: The MIT Press.

OWI. (12 de 02 de 2018). *OWI Inc.* Obtenido de http://www.owirobot.com/robotic-arm-edge-1/

Pinto, J. (2012). *Fuerza Electromotriz y Circuitos.*

Reyes Cortés, F. (2011). *Robótica. Control de robots manipuladores.* México D.F.: Alfaomega.

Reyes Cortés, F. (2012). *Matlab aplicado a robótica y mecatrónica.* México D.F.: Alfaomega.

Saha, S. K. (2010). *Introducción a la robótica.* México D. F.: Mc Graw Hill.

Torrente Artero, Ó. (2013). *Arduino. Curso práctico de formación.* Madrid: RC Libros.

APÉNDICES

APÉNDICE A

Especificaciones técnicas

Brazo robótico OWI-535

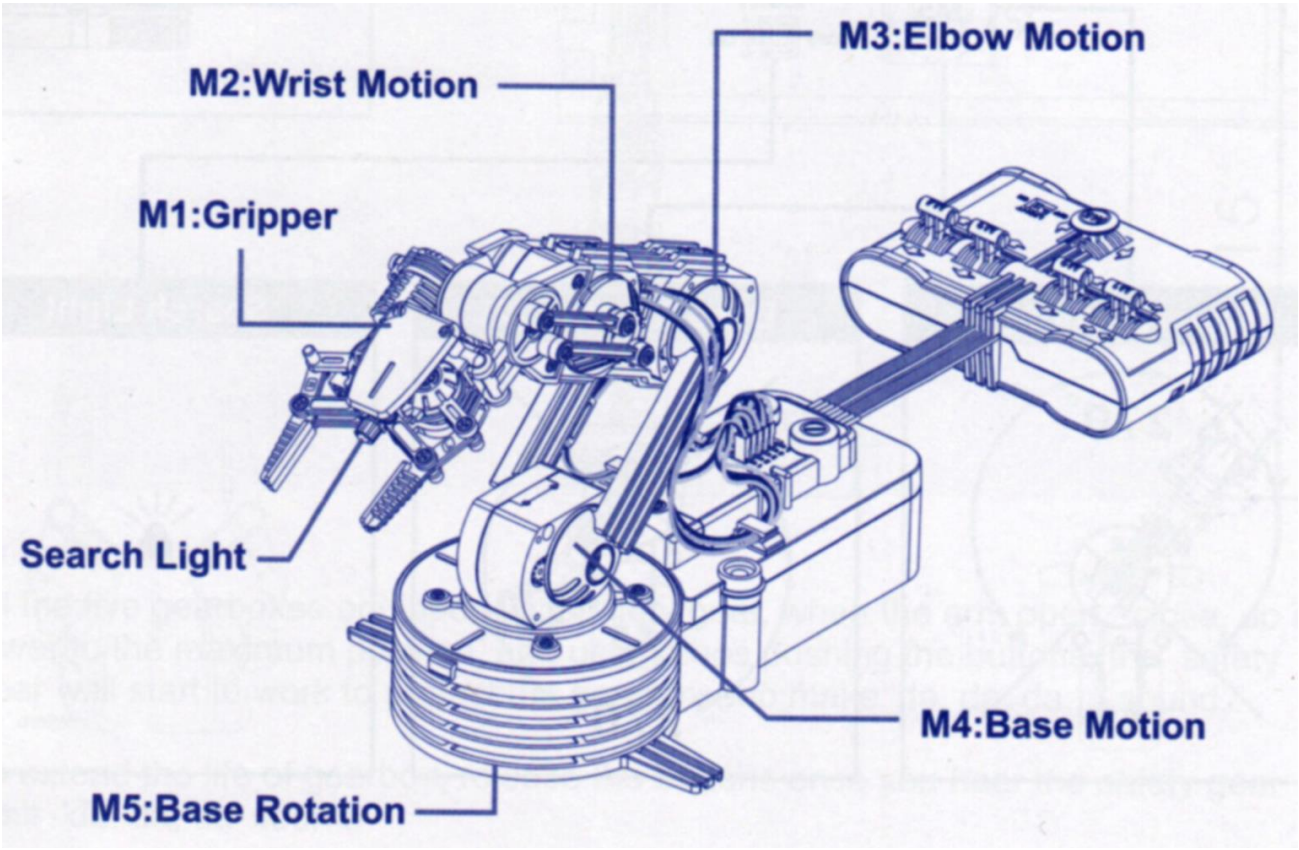


Figura 4.1. Partes brazo robótico OWI-535

Fuente: OWI Inc, 2017

Especificaciones técnicas:

- Capacidad de elevación: 100g.
- Dimensiones: 9” (22,86cm) L x 6.3” (16cm) W x 15” (38cm) H
- Peso: 1.5 libras (658 g)
- Fuente de alimentación: 4 pilas D (no incluidas)
- Alcance vertical máximo: 15 "(38 cm)
- Alcance horizontal máximo: 12.6 "(32 cm)
- Rango de movimiento de la muñeca: 120 °
- Rango de movimiento del codo: 300 °
- Movimiento de la base (hombro) Rango: 180 °
- Rango de rotación de la base: 270 °

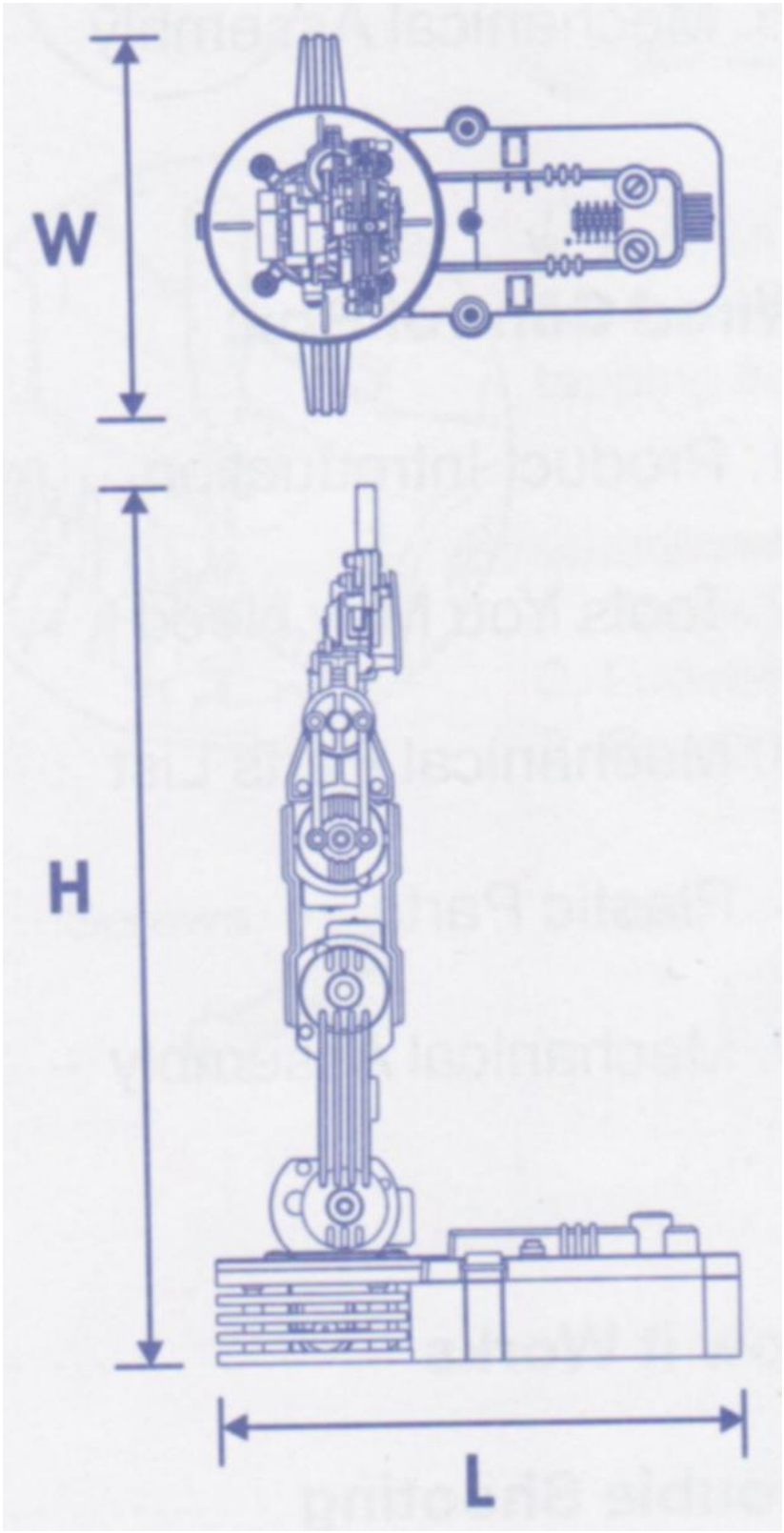


Figura 4.2. Dimensiones brazo robótico OWI-535

Fuente: OWI Inc, 2017

El brazo robótico realiza 12 diferentes movimientos los cuales se muestran a continuación:

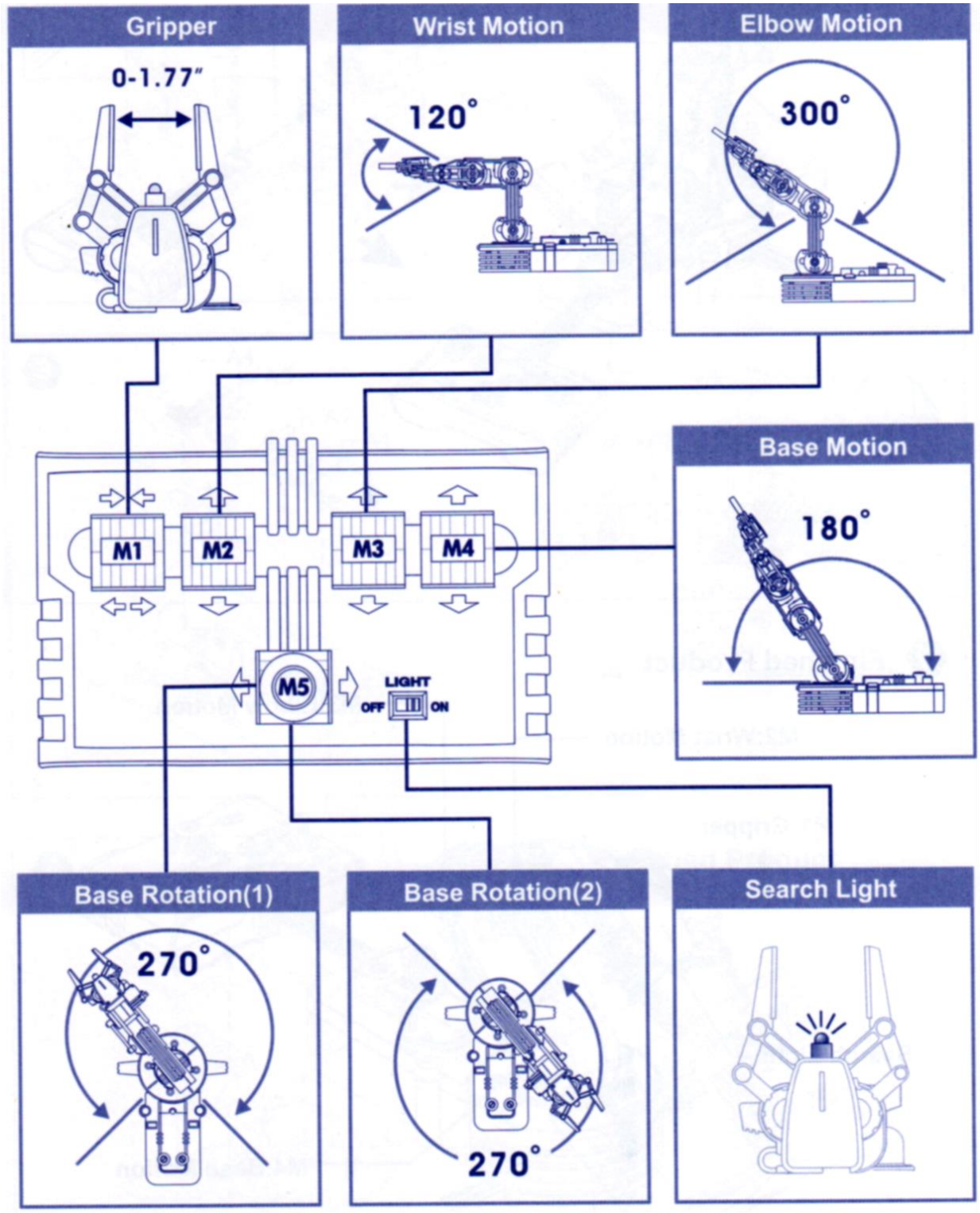


Figura 4.3. Movimientos del brazo robótico OWI-535

Fuente: OWI Inc, 2017

Área de trabajo:

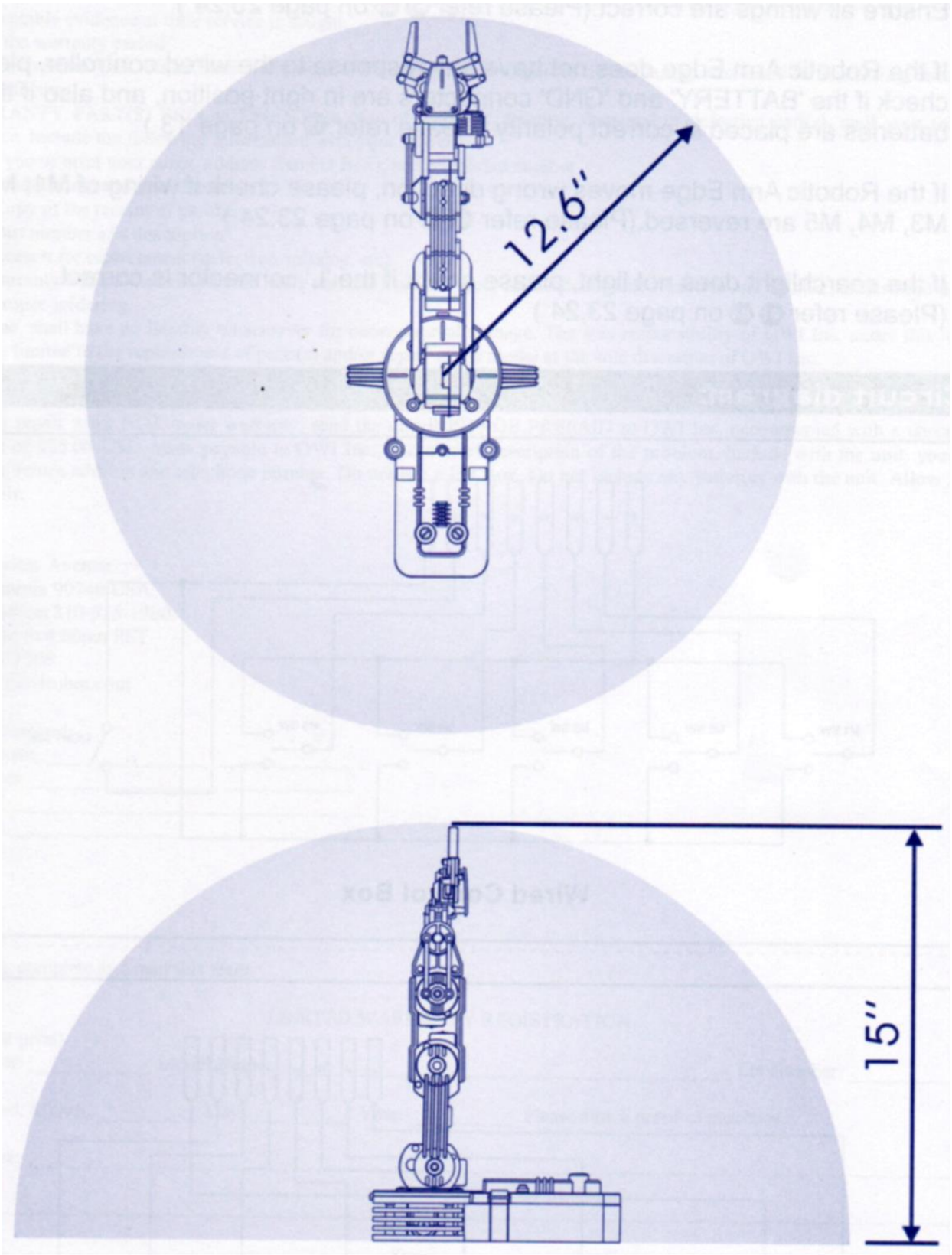


Figura 4.4. Área de trabajo del brazo robótico OWI-535

Fuente: OWI Inc, 2017

APÉNDICE B

Algoritmos:

1) Distancia en píxeles
2) Visión artificial
3) Cinemática Inversa
4) Espacio de trabajo OWI-535
5) Espacio de trabajo del proyecto
6) Simulación
7) Proyecto Arduino
8) Proyecto Matlab

1) **Distancia en píxeles**

```
clear all; close all; clc;
imaqhwinfo;
cam=imaqhwinfo;
cam.InstalledAdaptors;
vid=videoinput('winvideo',1,'YUY2_640x480');
cd('calibracion'); %ingresa a carpeta
filename = sprintf('image.tif');
imag=getsnapshot(vid);
imwrite(imag, filename);

figure, imshow('image.tif');
h = imdistline(gca);
api = iptgetapi(h);
fcn = makeConstrainToRectFcn('imline',...
                        get(gca,'XLim'),get(gca,'YLim'));
api.setDragConstraintFcn(fcn);
```

2) Visión artificial

```
clear all; close all; clc;
imaqhwinfo;
cam=imaqhwinfo;
cam.InstalledAdaptors;
vid=videoinput('winvideo',1,'YUY2_640x480');
set(vid, 'ReturnedColorspace', 'rgb'); %La propiedad ReturnedColorSpace especifica el espacio de color que desea que use la caja de herramientas cuando devuelve datos de imagen al espacio de trabajo de MATLAB®. Esto solo es relevante cuando accede a datos de imágenes adquiridos con las funciones getsnapshot, getdata y peekdata.
data=getsnapshot(vid);
%extrae el color rojo:(:,:,1), verde:(:,:,2) o azul:(:,:,3)
diff_im_r=imsubtract(data(:,:,1),rgb2gray(data));
diff_im_v=imsubtract(data(:,:,2),rgb2gray(data));
diff_im_a=imsubtract(data(:,:,3),rgb2gray(data));
%filtro para quitar ruido
diff_im_r=medfilt2(diff_im_r,[3,3]);
diff_im_v=medfilt2(diff_im_v,[3,3]);
diff_im_a=medfilt2(diff_im_a,[3,3]);
%convertir imagen a binario
diff_im_r=im2bw(diff_im_r,0.18);
diff_im_v=im2bw(diff_im_v,0.18);
diff_im_a=im2bw(diff_im_a,0.18);
%cambio de tamaño a 300 Megapixeles
diff_im_r=bwareaopen(diff_im_r, 300);
diff_im_v=bwareaopen(diff_im_v, 300);
diff_im_a=bwareaopen(diff_im_a, 300);
%nombres componentes de imagen
bw_r=bwlabel(diff_im_r, 8);
bw_v=bwlabel(diff_im_v, 8);
bw_a=bwlabel(diff_im_a, 8);
```

```
%propiedades centroide
stats_r=regionprops(bw_r, 'BoundingBox','Centroid');
stats_v=regionprops(bw_v, 'BoundingBox','Centroid');
stats_a=regionprops(bw_a, 'BoundingBox','Centroid');
imshow(data);
hold on
if length(stats_r)>0
   for object = 1.length(stats_r)
      bb=stats_r(object).BoundingBox; % stats: devuelve valores almacenados en caché y estadísticas sobre el uso de un objeto .
      bc=stats_r(object).Centroid;
      rectangle('Position',bb,'EdgeColor','r','LineWidth',2);
      plot(bc(1),bc(2),'-m+'); %mostrar rectángulo según propiedades de la imagen%X=round(bc(1));
      fc=23/32; %factor de conversión de pixeles a mm
      X=round(bc(1)*fc)-230;
      Y=-round(bc(2)*fc)+345;
      a=text(bc(1)+15,bc(2),strcat('X: ', num2str(X), 'Y: ', num2str(Y))); %round:redondea al decimal o entero más cercano. num2str:Convertir números en matriz de caracteres. strcat:Concatenar cadenas horizontalmente. text:Agregar descripciones de texto a los puntos de datos.
      set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,'Color','b');
      Entero=int8(bc(1)*0.18); %posición de imagen
   end
else
   if length(stats_v)>0
      for object = 1.length(stats_v)
         bb=stats_v(object).BoundingBox; % stats: devuelve valores almacenados en caché y estadísticas sobre el uso de un objeto .
         bc=stats_v(object).Centroid;
         rectangle('Position',bb,'EdgeColor','r','LineWidth',2);
         plot(bc(1),bc(2),'-m+'); %mostrar rectángulo según propiedades de la imagen
```

```
            fc=23/32; %factor de conversión de pixeles a mm
            X=round(bc(1)*fc)-230;
            Y=-round(bc(2)*fc)+345;
            a=text(bc(1)+15,bc(2),strcat('X:  ', num2str(X), 'Y:  ', num2str(Y)));
```
%round:redondea al decimal o entero más cercano. num2str:Convertir números en matriz de caracteres. strcat:Concatenar cadenas horizontalmente. text:Agregar descripciones de texto a los puntos de datos.
```
            set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,'Color','b');
            Entero=int8(bc(1)*0.18); %posición de imagen
        end
    else
        for object = 1.length(stats_a)
            bb=stats_a(object).BoundingBox; % stats: devuelve valores almacenados en caché y estadísticas sobre el uso de un objeto .
            bc=stats_a(object).Centroid;
            rectangle('Position',bb,'EdgeColor','r','LineWidth',2);
            plot(bc(1),bc(2),'-m+'); %mostrar rectángulo según propiedades de la imagen
            fc=23/32; %factor de conversión de pixeles a mm
            X=round(bc(1)*fc)-230;
            Y=-round(bc(2)*fc)+345;
            a=text(bc(1)+15,bc(2),strcat('X:  ', num2str(X), 'Y:  ', num2str(Y)));
```
%round:redondea al decimal o entero más cercano. num2str:Convertir números en matriz de caracteres. strcat:Concatenar cadenas horizontalmente. text:Agregar descripciones de texto a los puntos de datos.
```
            set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,'Color','b');
            Entero=int8(bc(1)*0.18); %posición de imagen
        end
    end
end
hold off;
stop(vid);
flushdata(vid); %limpia datos de imagen guardados en memoria
```

3) Cinemática Inversa

```
%Análisis cinemático inverso en el plano de los eslabones 2 al 4. Para encontrar los posibles ángulos que se puedan generar para llegar a un punto deseado.
%Antes de iniciar el programa, se necesita ingresar una matriz homogénea que se llamara TH y el punto p1.
%Se utiliza la función syms para expresar las ecuaciones que se generan.
syms  q1 q2 q3 q4 l1 l2 l3 l4 nx ny nz ox oy oz ax ay az px py pz alfa1
%Parámetros Denavit-Hartenberg del robot
teta = [q2 q3 q4];
d    = [0 0 0];
a    = [90 110 110];
alfa = [0  0  0];
%Matrices de transformación homogénea entre sistemas de coordenadas
%consecutivosT
A01 = denavitv3(teta(1), d(1), a(1), alfa(1));
A12 = denavitv3(teta(2), d(2), a(2), alfa(2));
A23 = denavitv3(teta(3), d(3), a(3), alfa(3));
%Matriz de transformación del primer al último sistema de coordenadas
A01;
A12;
A23;
TH = [nx ox ax px; ny oy ay py; nz oz az pz; 0 0 0 1];
T = A01*A12*A23;
%Se identifican lsa posibles ecuaciones con los valores de la matriz TH.
iz1 = TH*inv(A23);
iz1 = subs(iz1,cos(q4)^2 + sin(q4)^2,1);

der1 = A01*A12;

iz2 = TH*inv(A23)*inv(A12);
iz2 = subs(iz2,cos(q4)^2 + sin(q4)^2,1);
iz2 = subs(iz2,cos(q3)^2 + sin(q3)^2,1);
```

```
der2 = A01;

%Las ecuaciones de las cuatro matrices a utilizar.
E11 = iz1(1,4)==der1(1,4);
E12 = iz1(2,4)==der1(2,4);

E21 = iz2(1,4)==der2(1,4);
E22 = iz2(2,4)==der2(2,4);

%Definidas, todas las matrices, se prosigue con el reemplazo de las expresiones en el caso de ser necesario.
Me = [E11; E12; E21; E22];
Mo = Me;
Mo = subs(Mo,cos(q4)^2 + sin(q4)^2, 1);

M1 = subs(Me,l1,7);
M2 = subs(M1,l2,9);
M3 = subs(M2,l3,11);
M4 = subs(M3,l4,10);
M5 = subs(M4,px,TH(1,4));
M6 = subs(M5,py,TH(2,4));
M7 = subs(M6,pz,TH(3,4));
M8 = subs(M7,cos(q4)^2 + sin(q4)^2, 1);

%Utilizando la función se busca los posibles valores de q2 y q3, entre las ecuaciones 1 y 2.
vars1 = [q2 q3];
sol1 = solve(M8(1:2), vars1, 'Real', true);
teta2 = sol1.q2;
teta2f = double(teta2)*180/pi;
teta3 = sol1.q3;
teta3f = double(teta3)*180/pi;
```

```
ang23 = [teta2f , teta3f];

%Reemplazando los valores q2 y q3, se generan todas las combinaciones posibles.
Mf = M8(3:4);
Mf4 = subs(Mf, q2, teta2);
tyi = size(Mf4);
for i = 1:tyi(1,1)
    sol2 = solve(Mf4(i), q4, 'Real', true);
    teta4 = sol2*180/pi;
    angte = double(teta4);
    ang4(:,i) = angte;
end


%Indicador de que el brazo robótico no llega
if p1>300
    disp('No puede llegar fisicamente')
end

%Se agrupa todas las combinaciones posibles de las ecuaciones
tag4= size(ang4);
if tag4(1)==1
    tag4 = tag4(1);
elseif  tag4(1) == 0
    disp('No existe solución')
end
tag2 = size(ang23);

tiempo1 = 0;
cant1=0;
for i = 1:tag4
    tiempo1= tiempo1+1
```

```
        if ((-1))^i==1
            for j = 1:2
                cant1=cant1+1
                beta(cant1,1:3) = [ang23(2,1) ang23(2,2) ang4(j,tiempo1)];
            end
        else
            for j=1:2
                cant1=cant1+1
                beta(cant1,1:3) = [ang23(1,1) ang23(1,2) ang4(j,tiempo1)];
            end
        end
end

%Se realiza el análisis cinemático directo con los ángulos de beta, para eliminar ángulos erróneos y comprobar que llega al punto deseado.
betat=size(beta);
TH = round(TH,4);
cero = zeros(4);
ui=0;
for i=1:betat(1,1)
    teta = beta(i,1:3);
    d    = [0 0 0];
    a    = [90  110 110];
    alfa = [0  0  0];
    B01 = denavit(teta(1), d(1), a(1), alfa(1));
    B12 = denavit(teta(2), d(2), a(2), alfa(2));
    B23 = denavit(teta(3), d(3), a(3), alfa(3));
    tp = B01*B12*B23;
    tp = round(tp,4);
    rest = tp-TH;
    if rest(1,4)==0 & rest(2,4)==0 & rest(3,4)==0
        ui=ui+1;
```

```
        resp(ui,1:3) = teta;
         disp('si');
    else
        disp('no');
    end
end
%Con el grupo de ángulos obtenidos, se eliminan los valores que excedan los límites
físicos que tiene el brazo robótico, en base sus especificaciones técnicas
%Hasta llegar a la solución final.
ta1 = size(resp);
tr = 0;
for i = 1:ta1(1)
    if resp(i,1)>= 0 & resp(i,1)<= 90
        tr=tr+1;
        resp1(tr,1:3)=resp(i,1:3);
    else
        disp('nf');
    end
end
ta2 = size(resp1);


t1=0;
for i = 1:ta2(1)
    if resp1(i,2)<=150 & resp1(i,2)>=-150
        t1=t1+1;
        resp2(t1,1:3)=resp1(i,1:3);
    else
        disp('nf1');
    end
end
ta4 = size(resp2);
```

```
t2=0;
for i = 1:ta4(1)
    if resp2(i,3)>=-60 & resp2(i,3)<=60
        t2=t2+1;
        resp3(t2,1:3)=resp2(i,1:3)
    else
        disp('nf2');
    end
end
```

4) Espacio de trabajo OWI-535

%En este algoritmo se busca graficar todos los puntos posibles que puede alcanzar el brazo robótico con todos los posibles ángulos que se le pueda dar a las articulaciones. Los saltos que se generan para disminuir el cálculo computacional y para mejorar la visualización del espacio.

```
%Puntos máximos y mínimos
t1 = 0;
for l = -135:10:135
   for i = 0:10:180
      for j = -150:10:150
         for k = -60:10:60
            t1 = t1+1;
            q = [l i j k];
            TH = directDH(q);
            p2(t1,:) = [TH(1,4), TH(2,4), TH(3,4)];
         end
      end
   end
end
t3 = 0;
for i = 1:t1
   if p2(i,1)>=0
      if p2(i,3)>=0
         t3 = t3+1;
         p3(t3,1:3)=p2(i,:);
      end
   elseif p2(i,1)<-0
      if p2(i,3)>=72
         t3 = t3+1;
         p3(t3,1:3)=p2(i,:);
      end
   end
```

```
end
hold on
P = p3;
plot3(P(:,1),P(:,2),P(:,3),'.','MarkerSize',10)
xlabel('X');
ylabel('Y')
grid on

% Animation usando la libreria de Peter Corker
% Dibujo del brazo robótico
L(1) = Link([0 70 0 pi/2 0]);
L(2) = Link([0 0 90 0 0]);
L(3) = Link([0 0 110 0 0]);
L(4) = Link([0 0 110 0 0]);

% Con SerialLink agrupamos todos los eslabones
robot = SerialLink(L, 'name', 'OWI-535');

% Ángulos de inicio y final para la simulación
qf0 = [0 pi/2 0 0];
qf1 = [0 pi/2 0 0];
%% Cinemática directa por medio fkine, utilizando la función del programa
T0 = robot.fkine(qf0);
%robo.plot(qf0)
T1 = robot.fkine(qf1);

%% Generación de la animación desde el punto inicial hasta el final
tiempo = [0:0.5:10];
hold on
q1 = jtraj(qf0, qf1, tiempo);
robot.plot(q1)
pause(5)
```

5) Espacio de trabajo del proyecto

%En este algoritmo se demostrar gráficamente cual es el área de trabajo que el brazo robótico tendrá las limitaciones físicas que presenta OWI-535 posterior a la automatización

```
%Puntos máximos y mínimos del espacio de trabajo
t1 = 0;
for l = -25:5:25
    for i = 0:5:90
        for j = -150:10:150
            for k = -60:10:60
                t1 = t1+1;
                q = [l i j k];
                TH = directDH(q);
                p2(t1,:) = [TH(1,4), TH(2,4), TH(3,4)];
            end
        end
    end
end
t3 = 0;
for i = 1:t1
    if p2(i,1)>=0
        if p2(i,3)>=0
            t3 = t3+1;
            p3(t3,1:3)=p2(i,:);
        end
    end

end
hold on
P = p3;
plot3(P(:,1),P(:,2),P(:,3),'.','MarkerSize',10)
xlabel('X');
```

```
ylabel('Y')
grid on
k = boundary(P);
hold on
trisurf(k,P(:,1),P(:,2),P(:,3),'Facecolor','red','FaceAlpha',0.1)
```

6) Simulación

```
% Animación usando la librería de Peter Corke
% Usando Link, se crea los eslabones que existen en el brazo robótico.
L(1) = Link([0 70 0 pi/2 0]);
L(2) = Link([0 0 90 0 0]);
L(3) = Link([0 0 110 0 0]);
L(4) = Link([0 0 110 0 0]);

% Con SerialLink agrupamos todos los eslabones
robot = SerialLink(L, 'name', 'OWI-535');

% Ángulos de inicio y final para la simulación
qf0 = [0 pi/2 0 0];
qf1 = [0 pi/2 0 0];

%% Cinemática directa por medio fkine, utilizando la función del programa
T0 = robot.fkine(qf0);
%robo.plot(qf0)
T1 = robot.fkine(qf1);

%% Generación de la animación desde el punto inicial hasta el final
tiempo = [0:0.5:10];
q1 = jtraj(qf0, qf1, tiempo);
robot.plot(q1)
pause(5)
```

7) Proyecto Arduino

```
#include <AFMotor.h>
float A; //almacenar datos Potenciómetro 1
float B; //almacenar datos Potenciómetro 2
float C; //almacenar datos Potenciómetro 3
float D; //almacenar datos Potenciómetro 4
int serie; //guardar datos de Matlab a Arduino a través de Bluetooth
AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);
int IN1 = 22; // variables de tipo entera
int IN2 = 24;
int inputPin = 26;
float value;

void setup()
{
  Serial.begin(115200);
  Serial1.begin(9600);
  motor1.setSpeed(200);
  motor2.setSpeed(200);
  motor3.setSpeed(200);
  motor4.setSpeed(200);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(inputPin, INPUT);
}


void loop()
{
```

```
if ( Serial1.available()) // puerto serial disponible
{ serie = Serial1.read(); //almacenar dato leido o enviado por el módulo Bluetooth
  if (serie == 1) { //dato obtenido en Matlab
    A = analogRead(A8);
    Serial1.println(A);
  }
  if (serie == 2) {
    B = analogRead(A9);
    Serial1.println(B);
  }
  if (serie == 3) {
    C = analogRead(A10);
    Serial1.println(C);
  }
  if (serie == 4) {
    D = analogRead(A11);
    Serial1.println(D);
  }
  if (serie == 5) {
    motor1.run(FORWARD);
    delay(100);
  }
  if (serie == 6) {
    motor1.run(RELEASE);
    delay(100);
  }
  if (serie == 7) {
    motor1.run(BACKWARD);
    delay(100);
  }
  if (serie == 8) { //dato obtenido en Matlab
    motor2.run(FORWARD);
```

```
  delay(100);
}
if (serie == 9) {
  motor2.run(RELEASE);
  delay(100);
}
if (serie == 10) {
  motor2.run(BACKWARD);
  delay(100);
}
if (serie == 11) { //dato obtenido en Matlab
  motor3.run(FORWARD);
  delay(100);
}
if (serie == 12) {
  motor3.run(RELEASE);
  delay(100);
}
if (serie == 13) {
  motor3.run(BACKWARD);
  delay(100);
}
if (serie == 14) { //dato obtenido en Matlab
  motor4.run(FORWARD);
  delay(100);
}
if (serie == 15) {
  motor4.run(RELEASE);
  delay(100);
}
if (serie == 16) {
  motor4.run(BACKWARD);
```

```
      delay(100);
    }
    if (serie == 17) { //dato obtenido en Matlab
      digitalWrite(IN1, HIGH); //pin encendido, emite 5V
      digitalWrite(IN2, LOW);   //pin apagado, emite 0V. Apagado es diferente de desactivado que significa que no emite nada.
    }
    if (serie == 18) {
      digitalWrite(IN1, LOW); // pin apagado, emite 0V. Apagado es diferente de desactivado que significa que no emite nada.
      digitalWrite(IN2, HIGH);  //  pin encendido, emite 5V
    }
    if (serie == 19) {
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
    }
    if (serie == 20) {
      value = digitalRead(inputPin);  //lectura digital de pin
      Serial1.println(value);
    }
  }
}
```

8) Proyecto Matlab

Vision_Artificial

```
% Angulo1
% py0 es el punto en el y-x 0
% px0 es el punto en el y-x 0
% p1 es el punto px en la matriz de ang12
% pz es el punto py en la matriz de ang12
px0=X;
py0=Y;
th1f = atan(px0/py0)*180/pi
p1 = sqrt((px0^2)+(py0^2));

TH=[1,0,0,p1;0,1,0,0;0,0,1,0;0,0,0,1];

Cinematica_Inversa; % Análisis Cinemático Inverso

angulos234=resp3(1,1:3)
ang2=resp3(1,1) %ángulo eslabón 2
ang3=resp3(1,2) %ángulo eslabón 3
ang4=resp3(1,3) %ángulo eslabón 4

Simulacion_Proyecto; %Simulación

%Movimiento de Motores
delete(instrfind); %delete connected ports
b=Bluetooth('btspp://98D331F523F3', 1);
fopen(b);
%%
%Rotación de base
P1=0;
a1=abs(th1f);
while P1<a1
```

```
    fwrite(b,1);
    P_M1 = fscanf(b,'%e'); %lectura potenciómetro 1
    if th1f<=0
        fwrite(b,5); %rotar en sentido antihorario
        pause(0.2)
        fwrite(b,6);
    else
        fwrite(b,7); %rotar en sentido horario
        pause(0.2)
        fwrite(b,6);
    end
    fwrite(b,1);
    P_M11 = fscanf(b,'%e');
    P1ant=P1;
    P1=abs(P_M11-P_M1)*290/1024
    P1=P1+P1ant
end

%Muñeca
P4=0;
a4=abs(ang4);
while P4<a4
    fwrite(b,4);
    P_M4=fscanf(b,'%e');
    if ang4<=0
        tic
        fwrite(b,14);
        pause(0.2);
        toc
        fwrite(b,15);
    else
        tic
```

```
        fwrite(b,16);
        pause(0.2);
        toc
        fwrite(b,15);
    end
    fwrite(b,4);
    P_M41 = fscanf(b,'%e');
    P4ant=P4;
    P4=abs((P_M41)-(P_M4))*290/1024
    P4=P4+P4ant
end

%Movimiento de base
P2=0;
a2=110-ang2;
while P2<=a2
    fwrite(b,2);
    P_M2 = fscanf(b,'%e');
    tic
    fwrite(b,8);
    pause(0.2)
    toc
    fwrite(b,9);
    fwrite(b,2);
    P_M21 = fscanf(b,'%e');
    P2ant=P2;
    P2=abs(P_M21-P_M2)*290/1024;
    P2=P2+P2ant
end

%Codo
P3=0;
```

```
a3=abs(ang3);
while P3<a3
    fwrite(b,3);
    P_M3 = fscanf(b,'%e');
    if ang3<=0
        tic
        fwrite(b,11);
        pause(0.2)
        toc
        fwrite(b,12);
    else
        tic
        fwrite(b,13);
        pause(0.2)
        toc
        fwrite(b,12);
    end
    fwrite(b,3);
    P_M31 = fscanf(b,'%e');
    P3ant=P3;
    P3=abs(P_M31-P_M3)*290/1024;
    P3=P3+P3ant
end

%Pinza
tic
fwrite(b,17); %Cerrar pinza
pause(0.6);
toc
fwrite(b,19); %Detener pinza

%Movimiento de base
```

```
P2=0;
tic
fwrite(b,10);
pause(0.5)
toc
fwrite(b,9);

%Rotación de base
P1=0;
th1f=25;
a1=abs(th1f);
while P1<a1
    fwrite(b,1);
    P_M1 = fscanf(b,'%e'); %lectura potenciómetro 1
    if th1f<=0
        tic
        fwrite(b,7); %rotar en sentido horario
        pause(0.2)
        toc
        fwrite(b,6);
    else
        tic
        fwrite(b,5); %rotar en sentido antihorario
        pause(0.2)
        toc
        fwrite(b,6);
    end
    fwrite(b,1);
    P_M11 = fscanf(b,'%e');
    P1ant=P1;
    P1=abs(P_M11-P_M1)*290/1024
    P1=P1+P1ant
```

```
end
tic
fwrite(b,5); %rotar en sentido antihorario
pause(1)
toc
fwrite(b,6);

%Movimiento de base
P2=0;
tic
fwrite(b,8);
pause(0.3)
toc
fwrite(b,9);

tic
fwrite(b,18); %Abrir pinza
pause(0.6);
toc
fwrite(b,19); %Detener pinza
```

APÉNDICE C

Brazo robótico OWI-535 automatizado

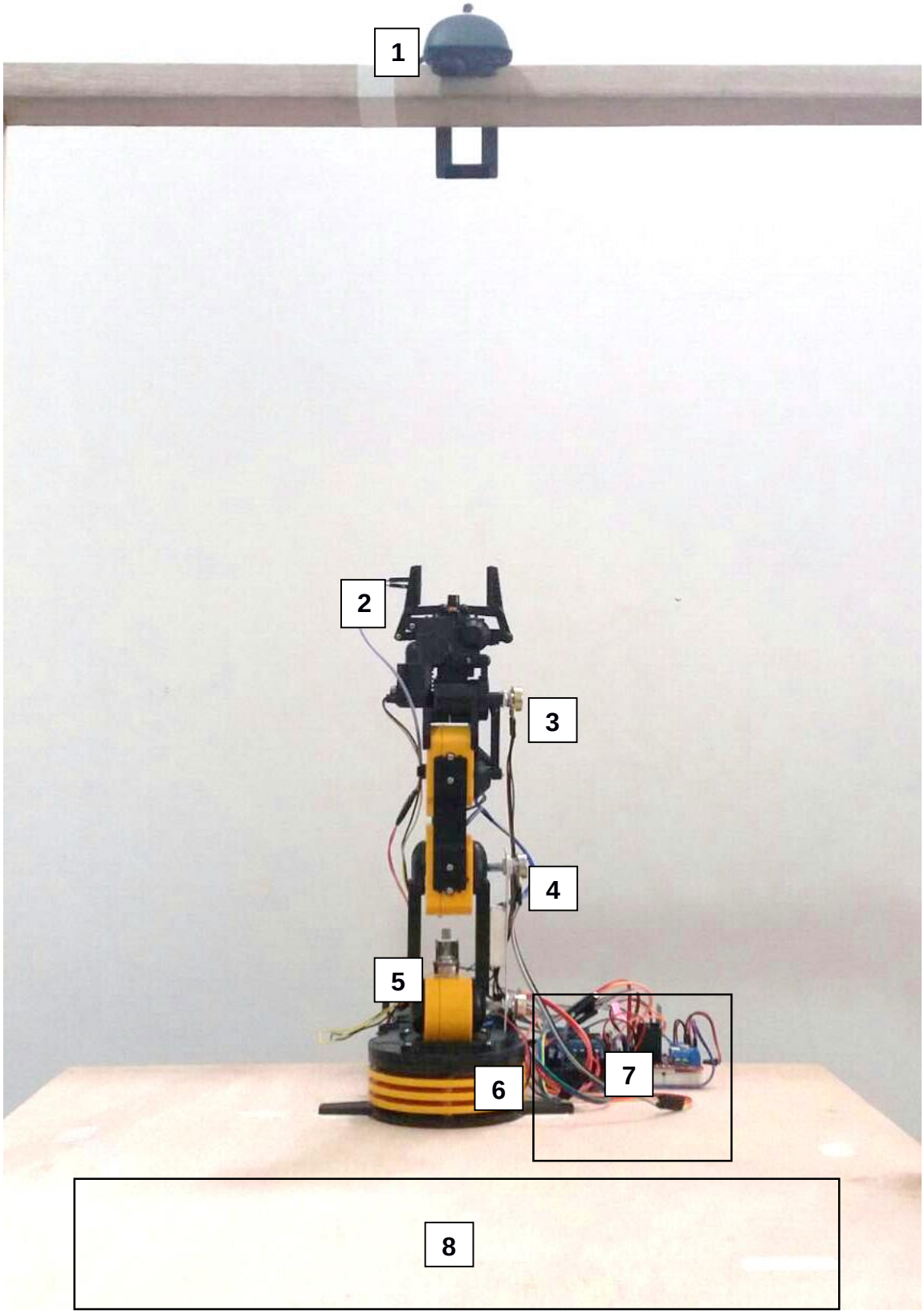


Figura 4.5. Área de trabajo y brazo robótico OWI-535 automatizado

Fuente: Escandón y Trujillo, 2017

Elaboración propia

A continuación, se detallan las partes numeradas en la figura 4.5. Con estos elementos se consiguió controlar mediante computador con el software Matlab el brazo robótico OWI-535, desplazando objetos.

1) Cámara web Genius Facecam
2) Pulsador final de carrera
3) Potenciómetro 4
4) Potenciómetro 3
5) Potenciómetro 1
6) Potenciómetro 2
7) Conformado por: Arduino Mega 2560
   Motor Shield
   L298N
   Bluetooth HC-05
8) Área de trabajo

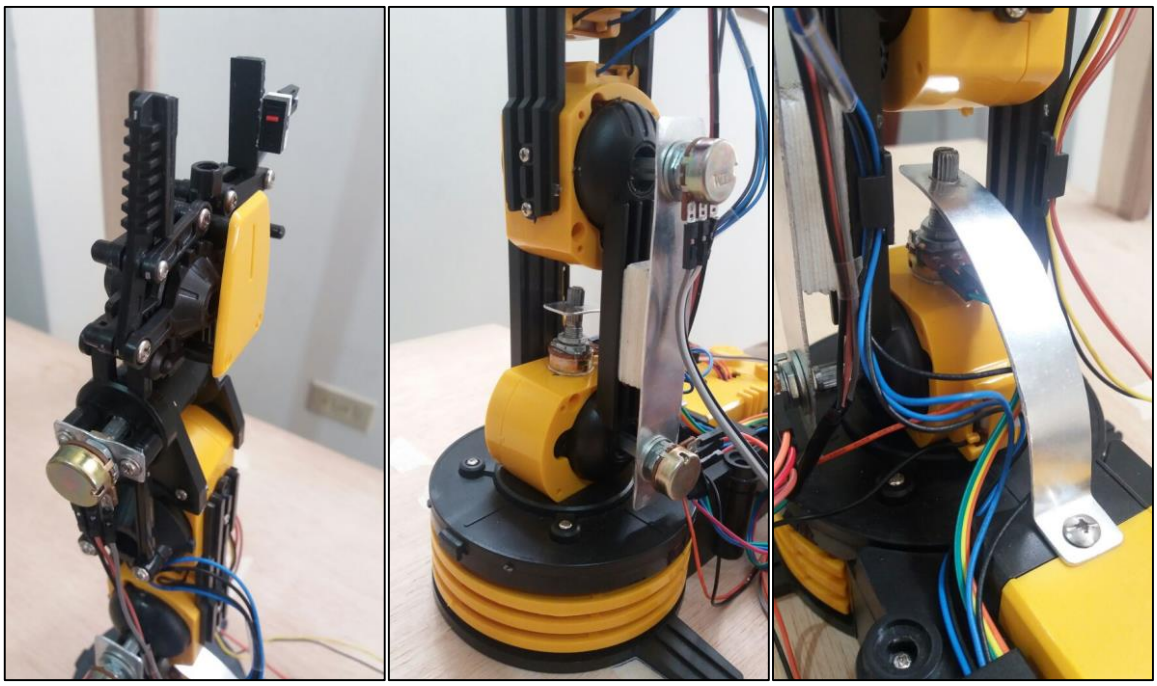


Figura 4.6. Sensores de posición instalados en el brazo robótico OWI-535

Fuente: Escandón y Trujillo, 2017

Elaboración propia