FACULTAD DE INGENIERÍA

Universidad Nacional de Lomas de Zamora

PROYECTO EN INGENIERÍA MECATRÓNICA

DOCENTES:

Ing. Cristian Leandro Lukaszewicz

Ing. Ezequiel Blanca

Ing. Juan Ignacio Szombach

D&B ROBOT

Robot jugador de puntos y cajas

ALUMNOS:

Daiana Belén Viscarra Hernández

Pedro Tagliani

FECHA DE ENTREGA: 16/12/2024

Contenido

1. Introducción 3
1.1. Motivación 3
1.2. Descripción de la propuesta 3
1.3. Puntos y cajas 3
2. Marco teórico 4
2.1. Cinemática directa 4
2.2. Cinemática inversa 7
2.3. Resolved Rate Motion Control (RRMC) 11
2.3.1. Intentos y obstáculos en su implementación 11
2.3.2. Implementación teórica 12
2.3.3. Simulación en Robotics Toolbox 14
2.4. Marcadores ArUco 17
3. Desarrollo integral del proyecto 18
3.1. Diseño mecánico 18
3.1.1. Base 18
3.1.2. Tapa con soporte principal para las articulaciones 19
3.1.3. Hombro (shoulder) 20
3.1.4. Codo (elbow) 21
3.1.5. Gripper 22
2.1.6. Ensamble completo 23
2.1.7. Mesa 23
2.1.8. Topes 24

2.1.9. Soporte para resortes 24
3.2. Diseño electrónico 25
3.2.1. Esquema del conexionado eléctrico 25
3.2.2. Fuente de alimentación 26
3.2.3. Módulo step-down 26
3.2.4. Controlador de servomotores 27
3.2.5. Servomotores 27
3.2.6. Motor paso a paso 27
3.2.7. Microcontrolador principal 28
3.2.8. Encoders magnéticos 28
3.3. Diseño de software 29
3.3.1. Calibración de la cámara 29
3.3.2. Detección del tablero 30
3.3.3. Detección de las líneas 31
3.3.4. Motor de juego 32
3.3.5. Comunicación con el brazo robótico 33
3.3.6. Control del robot 33
3.4. Funcionamiento y uso 34
4. Evaluación del proyecto 35
4.1. Resultados obtenidos 35
4.2. Costo real del prototipo 36
4.3. Tiempo invertido 37
5. Conclusiones 37

1. Introducción

1.1. Motivación

El proyecto se origina de la necesidad de desarrollar un producto capaz de proporcionar a personas de todas las edades una alternativa de entretenimiento que fomente la participación activa y el disfrute de juegos clásicos fuera de la pantalla. En un mundo cada vez más influenciado por la tecnología digital, se buscó ofrecer una experiencia lúdica que combine la nostalgia de los juegos tradicionales con la innovación tecnológica. Al proporcionar una experiencia interactiva y educativa, se pretende contribuir de manera positiva al bienestar y desarrollo de personas de todas las edades.

1.2. Descripción de la propuesta

La propuesta consiste en el desarrollo de un brazo robótico multifuncional diseñado para enfrentarse a una persona en el clásico juego de lápiz y papel conocido como puntos y cajas. Este brazo, conformado por cuatro grados de libertad, es capaz de realizar movimientos precisos y controlados. El juego se desarrolla sobre una pizarra blanca de 30x40 cm, en la cual el brazo puede desplazarse y dibujar utilizando un marcador integrado en su efector final.
Para llevar a cabo su propósito, el sistema incorpora una cámara que le permite detectar cuándo es su turno de jugar, así como analizar los movimientos realizados por el oponente. Basándose en esta información, el brazo ejecuta movimientos autónomos utilizando un algoritmo de inteligencia artificial que sigue una estrategia optimizada para maximizar su desempeño. Este enfoque asegura una experiencia de juego fluida, interactiva y competitiva.

1.3. Puntos y cajas

Se trata de un juego de lápiz y papel que combina estrategia y habilidad, diseñado para dos o más jugadores. Su objetivo principal es conectar puntos en una cuadrícula para formar cajas cerradas, acumulando puntos y superando al oponente. Este juego, que también es conocido como dots and boxes, es popular por su simplicidad en las reglas y su capacidad de fomentar el pensamiento estratégico.
El juego comienza dibujando una cuadrícula de puntos en una hoja de papel; el tamaño de esta cuadrícula puede variar según el nivel de desafío deseado. A partir de este diseño inicial, los jugadores se turnan para trazar una línea entre dos puntos adyacentes, ya sea en posición horizontal o vertical. Cada línea dibujada representa un pequeño avance hacia la creación de cajas en la cuadrícula. El desafío radica en aprovechar cada movimiento no solo para cerrar cajas propias, sino también para evitar que los oponentes logren hacerlo.
Una vez que un jugador completa el cuarto lado de un cuadrado, es decir, cierra una caja, obtiene un punto y se le permite jugar de nuevo. Esto implica que un movimiento estratégico puede

derivar en una cadena de acciones consecutivas, permitiendo al jugador cerrar múltiples cajas en un mismo turno. Este sistema de recompensas introduce un elemento de planificación y anticipación, ya que cada línea dibujada puede influir de manera significativa en las oportunidades del oponente en turnos posteriores.
El juego continúa hasta que todas las líneas posibles en la cuadrícula han sido dibujadas y todas las cajas se encuentran cerradas. Al finalizar, se realiza el conteo de puntos basado en la cantidad de cajas cerradas por cada jugador, y el ganador es quien haya acumulado el mayor número de puntos. En caso de empate, se declara un resultado igualitario.
El juego no solo depende de la suerte, sino que también requiere análisis y planificación. Los jugadores deben evitar dar oportunidades a su oponente para completar múltiples cuadros consecutivos y, al mismo tiempo, buscar maximizar su propio puntaje cerrando cuadros en cadena cuando sea posible.

2. Marco teórico

2.1. Cinemática directa

El problema cinemático directo permite determinar la posición y orientación del extremo final del robot con respecto a un sistema de coordenadas de referencia, a partir de los valores de las articulaciones y los parámetros geométricos de sus elementos. En otras palabras, se busca calcular la pose (posición y orientación) que alcanzará el robot para una configuración dada de sus articulaciones.
Para resolver este problema y obtener la matriz de transformación homogénea T, que relaciona la posición y orientación del extremo del robot con el sistema de referencia fijo situado en su base, se empleó el método de Denavit-Hartenberg. Este método consiste en una serie de transformaciones matriciales aplicadas en un orden específico, que permite modelar de manera sistemática la cinemática del robot. Las transformaciones se realizan en el siguiente orden:

$$^{i-1}A_i = Rotz(\theta_i) \cdot T(0,0,d_i) \cdot T(a_i,0,0) \cdot Rotx(\alpha_i)$$

$$^{i-1}A_i = \begin{bmatrix} cos(\theta_i) & -sen(\theta_i) & 0 & 0 \\ sen(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sen(\alpha_i) & 0 \\ 0 & sen(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{i-1}A_i = \begin{bmatrix} cos(\theta_i) & -cos(\alpha_i) \cdot sen(\theta_i) & sen(\alpha_i) \cdot sen(\theta_i) & a_i \cdot cos(\theta_i) \\ sen(\theta_i) & cos(\alpha_i) \cdot cos(\theta_i) & -sen(\alpha_i) \cdot cos(\theta_i) & a_i \cdot sen(\theta_i) \\ 0 & sen(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para aplicar el método, las ternas de referencia se ubicaron de manera sistemática, como se muestra en la siguiente imagen:

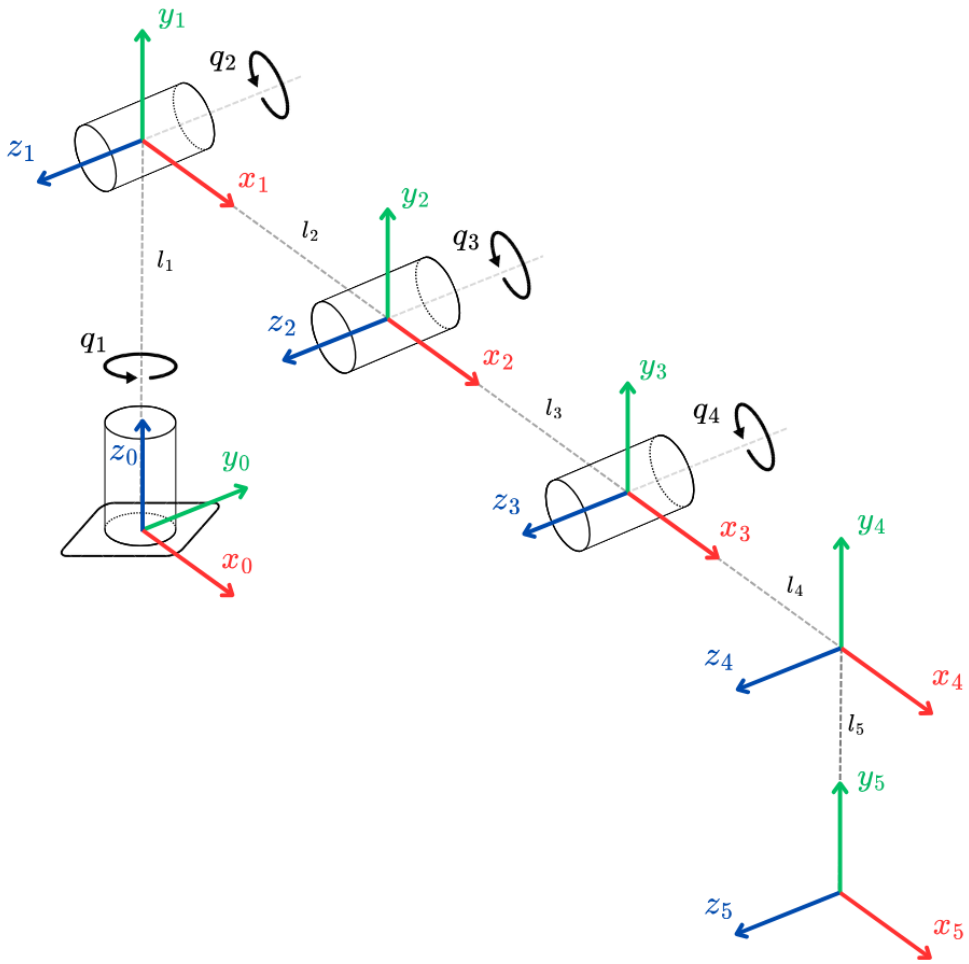

Figura 1: Configuración de ternas. Elaboración propia.

Es importante destacar que la terna $\{S_5\}$ se agrega de forma adicional, aplicando únicamente una traslación respecto al eje y de la terna anterior. Esto permite representar el marcador que estará sujeto al gripper del robot, de modo que la cinemática relacione el punto donde el marcador apoyará sobre la pizarra. Además, se debe considerar que los ángulos q_1, q_3 y q_4 serán ajustados para que sus ejes coincidan físicamente con las posiciones iniciales de sus respectivos servomotores.

Con estas consideraciones, se completa la tabla de parámetros DH y se obtienen las matrices de transformación homogénea correspondientes:

Articulación	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{a_i}$	$\boldsymbol{\alpha_i}$
1	$q_1 - 90°$	l_1	0	90°
2	q_2	0	l_2	0
3	$q_3 - 118°$	0	l_3	0
4	$q_4 - 20°$	0	l_4	0

Tabla 1: Definición de parámetros DH. Elaboración propia.

$$ {}^{0}A_1 = \begin{bmatrix} cos(q_1 - 90°) & 0 & sen(q_1 - 90°) & 0 \\ sen(q_1 - 90°) & 0 & -cos(q_1 - 90°) & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}A_2 = \begin{bmatrix} cos(q_2) & -sen(q_2) & 0 & l_2 \cdot cos(q_2) \\ sen(q_2) & cos(q_2) & 0 & l_2 \cdot sen(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}A_3 = \begin{bmatrix} cos(q_3 - 118°) & -sen(q_3 - 118°) & 0 & l_3 \cdot cos(q_3 - 118°) \\ sen(q_3 - 118°) & cos(q_3 - 118°) & 0 & l_3 \cdot sen(q_3 - 118°) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}A_4 = \begin{bmatrix} cos(q_4 - 20°) & -sen(q_4 - 20°) & 0 & l_4 \cdot cos(q_4 - 20°) \\ sen(q_4 - 20°) & cos(q_4 - 20°) & 0 & l_4 \cdot sen(q_4 - 20°) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}A_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -l_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

De este modo, es posible obtener la matriz T deseada, la cual fue calculada mediante Python. No se incluye en este documento debido a la complejidad de su expresión matricial, que resulta poco práctica para representar en este archivo. No obstante, su desarrollo completo puede consultarse en el archivo Jupyter Notebook adjunto.

$$ T = {}^{0}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3 \cdot {}^{3}A_4 \cdot {}^{4}A_5 $$

Esta matriz T, como se mencionó previamente, representa la pose del robot referida a la terna base para una configuración específica de sus variables articulares. Su componente de posición se expresa como (p_x, p_y, p_z), mientras que la orientación se describe a través de una matriz de rotación. Esta última se ha transformado a la notación de ángulos de Euler para facilitar las comparaciones con los cálculos de cinemática inversa.

Dado que este robot cuenta con 4 grados de libertad, en su configuración actual es posible controlar únicamente el ángulo de cabeceo (pitch). Para abordar esto, se diseñó un algoritmo en Python que convierte una matriz de rotación en ángulos de Euler XYZ con respecto a los ejes de la terna base (sistema fijo). De los tres ángulos obtenidos, solo se toma en consideración el que corresponde a la rotación alrededor del eje Y_0, es decir, el ángulo de cabeceo.

2.2. Cinemática inversa

El problema cinemático inverso consiste en determinar los valores de las coordenadas articulares de un robot que permiten posicionar y orientar su extremo final en una localización y orientación espacial deseada. A diferencia de la cinemática directa, que establece de manera sistemática la posición del efector a partir de las coordenadas articulares, la cinemática inversa es más compleja, ya que el procedimiento para encontrar las ecuaciones depende de manera significativa de la configuración y geometría específica del robot. Esta dependencia puede hacer que la resolución sea más desafiante y no siempre única.

En este trabajo, se abordó la resolución del problema cinemático inverso empleando el método geométrico, el cual se basa en analizar la disposición espacial del robot y descomponer el problema en relaciones geométricas simples entre sus eslabones y articulaciones. Este enfoque es especialmente útil cuando el robot cuenta con una estructura cinemática bien definida, ya que permite obtener soluciones analíticas directas en función de las características geométricas del sistema.

Para los ángulos articulares resultantes, se adoptó la configuración conocida como codo arriba (elbow-up), que es una de las múltiples soluciones posibles dependiendo de la orientación deseada del robot y de las restricciones mecánicas o de operación.

A continuación, se desarrolla en detalle el método geométrico empleado, describiendo los pasos seguidos para la resolución y los criterios utilizados para garantizar que los valores obtenidos cumplan con las especificaciones del problema planteado.

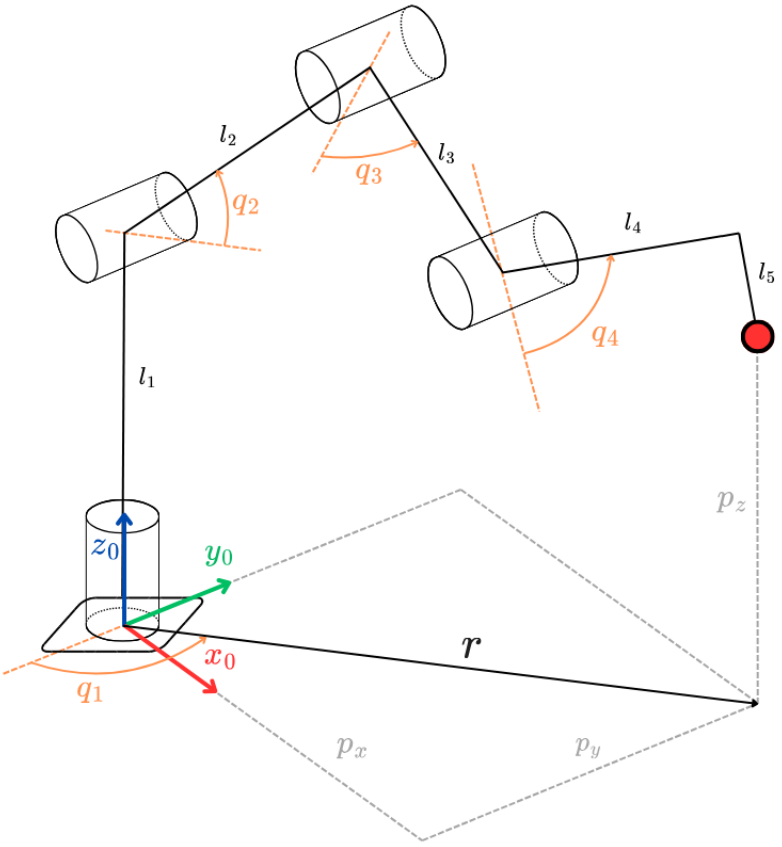

Figura 2: Configuración del brazo. Elaboración propia.

De la siguiente relación trigonométrica se obtiene el valor de q_1 y r:

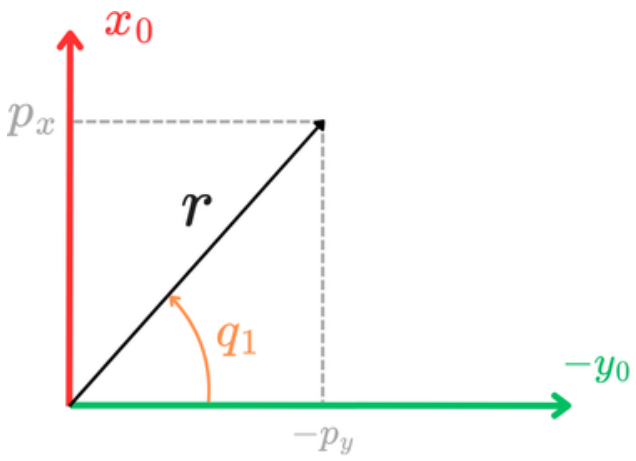

Figura 3: Análisis del plano x_0-y_0. Elaboración propia

$$\tan(q_1) = \frac{p_x}{-p_y} \rightarrow q_1 = arctan\left(\frac{p_x}{-p_y}\right) \rightarrow \boldsymbol{q_1 = arctan2(p_x, -p_y)}$$

$$r^2 = p_x^2 + (-p_y)^2 \rightarrow r = \sqrt{p_x^2 + (-p_y)^2}$$

Posteriormente, se utiliza un plano que permite definir las demás variables articulares, el cual está determinado por z_0 y r:

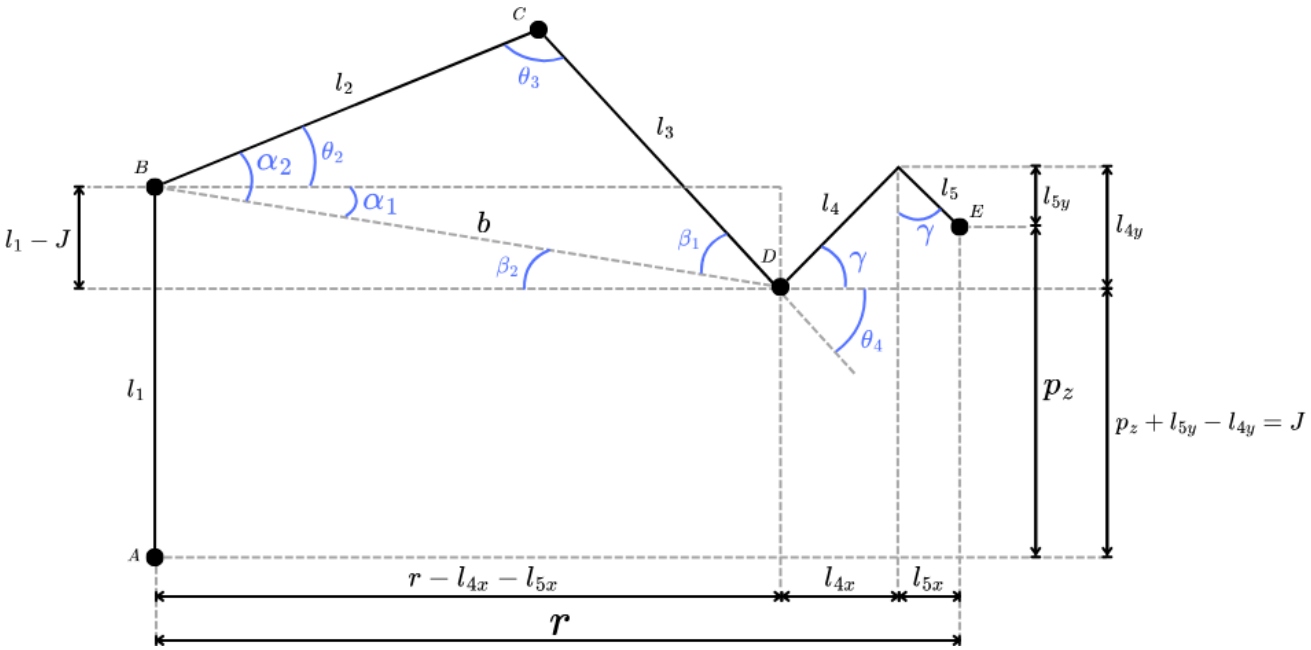

Figura 4: Análisis del plano z_0-r. Elaboración propia.

$$sen(-\gamma) = \frac{l_{4y}}{l_4} \rightarrow l_{4y} = sen(-\gamma) \cdot l_4$$

$$cos(-\gamma) = \frac{l_{4x}}{l_4} \rightarrow l_{4x} = cos(-\gamma) \cdot l_4$$

$$sen(-\gamma) = \frac{l_{5x}}{l_5} \rightarrow l_{5x} = sen(-\gamma) \cdot l_5$$

$$cos(-\gamma) = \frac{l_{5y}}{l_5} \rightarrow l_{5y} = cos(-\gamma) \cdot l_5$$

$$b = \sqrt{(l_1 - J)^2 + (r - l_{4x})^2}$$

$$tan(\alpha_1) = \frac{l_1 - J}{r - l_{4x}} \rightarrow \alpha_1 = arctan\left(\frac{l_1 - J}{r - l_{4x}}\right) \rightarrow \alpha_1 = arctan2(l_1 - J, r - l_{4x})$$

$$tan(\beta_2) = \frac{l_1 - J}{r - l_{4x}} \rightarrow \beta_2 = arctan\left(\frac{l_1 - J}{r - l_{4x}}\right) \rightarrow \beta_2 = arctan2(l_1 - J, r - l_{4x})$$

$$\theta_2 = \alpha_2 - \alpha_1$$

$$\theta_4 = \beta_1 + \beta_2$$

$$sen^2(\varepsilon) + cos^2(\varepsilon) = 1 \rightarrow sen^2(\varepsilon) = 1 - cos^2(\varepsilon) \rightarrow sen(\varepsilon) = \pm\sqrt{1 - cos^2(\varepsilon)}$$

Se aplica la Ley del Coseno en el triángulo BCD:

$$l_3^2 = l_2^2 + b^2 - 2 \cdot l_2 \cdot b \cdot cos(\alpha_2) \rightarrow cos(\alpha_2) = \left(\frac{l_2^2 + b^2 - l_3^2}{2 \cdot l_2 \cdot b}\right)$$

$$b^2 = l_2^2 + l_3^2 - 2 \cdot l_2 \cdot l_3 \cdot cos(\theta_3) \rightarrow cos(\theta_3) = \left(\frac{l_2^2 + l_3^2 - b^2}{2 \cdot l_2 \cdot l_3}\right)$$

$$l_2^2 = l_3^2 + b^2 - 2 \cdot l_3 \cdot b \cdot cos(\beta_1) \rightarrow cos(\beta_1) = \left(\frac{l_3^2 + b^2 - l_2^2}{2 \cdot l_3 \cdot b}\right)$$

Los ángulos dados se transforman en función de la arcotangente, utilizando las variables previamente calculadas:

$$sen(\alpha_2) = \pm\sqrt{1 - cos^2(\alpha_2)} \rightarrow \alpha_2 = arctan\left(\frac{\pm\sqrt{1 - cos^2(\alpha_2)}}{cos(\alpha_2)}\right)$$

$$sen(\theta_3) = \pm\sqrt{1 - cos^2(\theta_3)} \rightarrow \theta_3 = arctan\left(\frac{\pm\sqrt{1 - cos^2(\theta_3)}}{cos(\theta_3)}\right)$$

$$sen(\beta_1) = \pm\sqrt{1 - cos^2(\beta_1)} \rightarrow \beta_1 = arctan\left(\frac{\pm\sqrt{1 - cos^2(\beta_1)}}{cos(\beta_1)}\right)$$

Como se mencionó previamente, se trabajó exclusivamente con la configuración codo arriba del brazo robótico. Por esta razón, en el código se implementó únicamente la solución que corresponde a la rama positiva de los cálculos. Además, los ángulos se expresaron utilizando la función arcotangente de dos parámetros en lugar de la arcotangente estándar.
El uso de $arctan2(y, x)$ resulta especialmente útil porque, a diferencia de la función $arctan(y/x)$, esta toma en cuenta explícitamente el signo tanto del numerador (y) como del denominador (x). Esto permite determinar en qué cuadrante se encuentra el ángulo calculado. En el contexto de la cinemática inversa, esta precisión es esencial para resolver correctamente las posiciones angulares del robot y evitar ambigüedades que puedan surgir al calcular direcciones o orientaciones. Por ejemplo, $arctan(y/x)$ podría dar el mismo valor para un ángulo en el primer cuadrante que para otro en el tercer cuadrante, mientras que $arctan2(y, x)$ distingue claramente entre ambos casos.

$$\alpha_2 = arctan2\left(\pm\sqrt{1 - cos^2(\alpha_2)}, cos(\alpha_2)\right)$$

$$\theta_3 = arctan2\left(\pm\sqrt{1 - cos^2(\theta_3)}, cos(\theta_3)\right)$$

$$\beta_1 = arctan2\left(\pm\sqrt{1 - cos^2(\beta_1)}, cos(\beta_1)\right)$$

Con las variables necesarias ya establecidas, es posible definir las expresiones que determinan las restantes variables articulares del robot:

$$\boldsymbol{q_2 = \theta_2}$$

$$q_3 = 180° - \theta_3$$

$$q_4 = \theta_4 - \gamma$$

Se ajustaron los ángulos q_3 y q_4 para cambiar su origen, de modo que coincidieran con la posición inicial de sus respectivos servomotores. Esto se debe a que dichos servomotores están limitados a operar en un rango de 0° a 180°, y el objetivo es aprovechar este rango de manera eficiente.

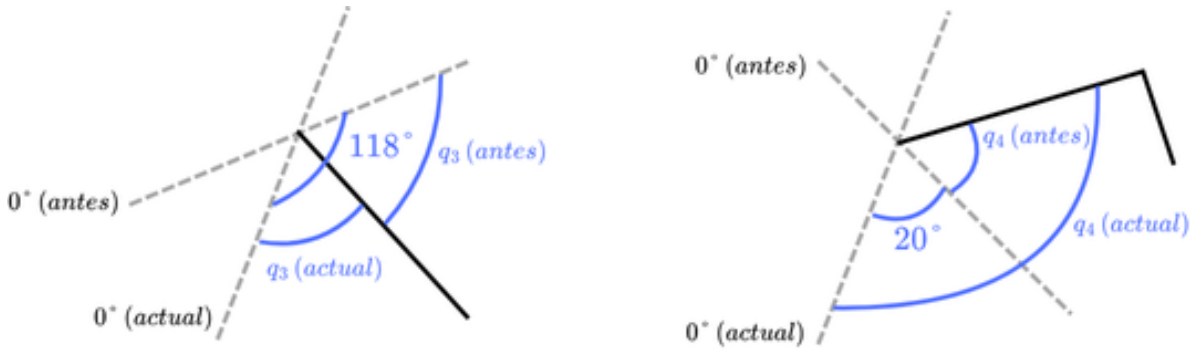

Figura 5: Corrección de ángulos. Elaboración propia.

$$q_3 = 118° - (180° - \theta_3) \rightarrow \boldsymbol{q_3 = -62° + \theta_3}$$

$$\boldsymbol{q_4 = \theta_4 - \gamma + 20°}$$

Por otro lado, es importante aclarar que el ángulo de cabeceo (γ) se incluyó como negativo en todas las expresiones en las que intervino. Esto obedece a la forma en que se definió la configuración al inicio de esta sección. La orientación del robot se planteó utilizando ángulos de Euler en el sistema XYZ, definidos respecto a la terna base (sistema fijo). Por este motivo, el ángulo de cabeceo está directamente relacionado con el eje y_0 de la terna base.
Al observar la rotación aplicada al ángulo de cabeceo en el esquema planteado, se puede notar que es negativo. Esto se debe a que el análisis se realiza considerando el eje y_0 desde una perspectiva posterior, lo cual se confirma fácilmente con la regla de la mano derecha: el giro es antihorario respecto a dicho eje.
En este contexto, cuando el ángulo de cabeceo es 0°, el gripper queda paralelo a la superficie de trabajo. Si el ángulo toma otro valor, el robot adoptará la orientación deseada respecto a la superficie, ajustando así la posición del gripper según lo requerido.

2.3. Resolved Rate Motion Control (RRMC)

2.3.1. Intentos y obstáculos en su implementación

A lo largo del desarrollo del proyecto, se buscó implementar un control a lazo cerrado para el brazo robótico, utilizando la retroalimentación proporcionada por los encoders magnéticos instalados en cada una de las articulaciones del robot. Sin embargo, a pesar de los esfuerzos realizados, no se logró implementar con éxito este método. En consecuencia, se optó por un control a lazo abierto, el cual resultó ser más fácil de aplicar y, considerando las limitaciones del sistema, más adecuado. Esta decisión fue influenciada por la complejidad inherente al diseño del brazo robótico, así como por la baja calidad de los servomotores utilizados.
Durante los intentos por implementar un control de movimiento basado en la técnica de velocidad resuelta, surgieron problemas significativos. Uno de los más críticos fue el desgaste de los engranajes que sostienen los ejes de los servomotores en las articulaciones del hombro y el

codo. Este desgaste fue causado por las variaciones en las velocidades articulares calculadas, las cuales aumentaban considerablemente conforme se acumulaba el error en el sistema. El control de velocidad en esta configuración del brazo representa un desafío notable. Con velocidades articulares muy bajas, los servomotores no eran capaces de realizar movimientos, ya que debían desplazar un ángulo diminuto en un lapso de tiempo muy breve. Debido a la baja calidad de estos motores, tales movimientos eran ignorados por los mismos, lo que impedía un control eficiente. En contraste, la articulación de la base, equipada con un motor paso a paso, ofrecía un mejor desempeño a bajas velocidades gracias a su mayor resolución. Sin embargo, esta disparidad en las capacidades de los motores utilizados (donde los servomotores presentan un desempeño ineficiente a bajas velocidades y el motor paso a paso sobresale) generó problemas en la coordinación general del sistema. Esto evidenció que los servomotores seleccionados no eran los más adecuados para el nivel de precisión requerido en un control a lazo cerrado.

Otro aspecto que condicionó significativamente la implementación del control a lazo cerrado fue el desempeño de los encoders magnéticos. Estos sensores presentaron lecturas no lineales, debido a que no estaban perfectamente centrados con respecto al eje de giro de los motores. Aunque se realizaron esfuerzos por alinear los encoders de la mejor forma posible, las limitaciones de diseño y construcción del brazo dificultaron un montaje más eficiente. Este problema fue particularmente notable en los servomotores, ya que no cuentan con un eje visible en la parte trasera, a diferencia del motor paso a paso.

Finalmente, todos estos inconvenientes (la calidad de los motores, la disparidad en las capacidades de los actuadores, las limitaciones en la alineación de los encoders, y el desgaste mecánico producido durante los intentos de implementación) condujeron a la decisión de desechar la opción de control a lazo cerrado. Si bien este método es teóricamente más robusto, las características actuales del brazo robótico hicieron inviable su aplicación. Por lo tanto, se decidió priorizar un enfoque a lazo abierto, que, aunque menos sofisticado, permitió alcanzar un desempeño funcional dentro de las limitaciones del sistema.

2.3.2. Implementación teórica

El control de movimiento por velocidad resuelta es una técnica ampliamente utilizada en la robótica para controlar de manera eficiente los movimientos de manipuladores robóticos, especialmente en sistemas con múltiples grados de libertad. Este método se basa en la resolución de velocidades articulares requeridas para que el extremo del robot siga una trayectoria deseada en el espacio cartesiano.

El principio fundamental de este control radica en la utilización de la matriz jacobiana, la cual establece una relación directa entre las velocidades articulares y las velocidades lineales y angulares del efector final en el espacio tridimensional. Con base en esta relación, y dado un conjunto de velocidades y posiciones deseadas en el espacio cartesiano, es posible calcular las velocidades que deben aplicarse a cada articulación para ejecutar el movimiento requerido.

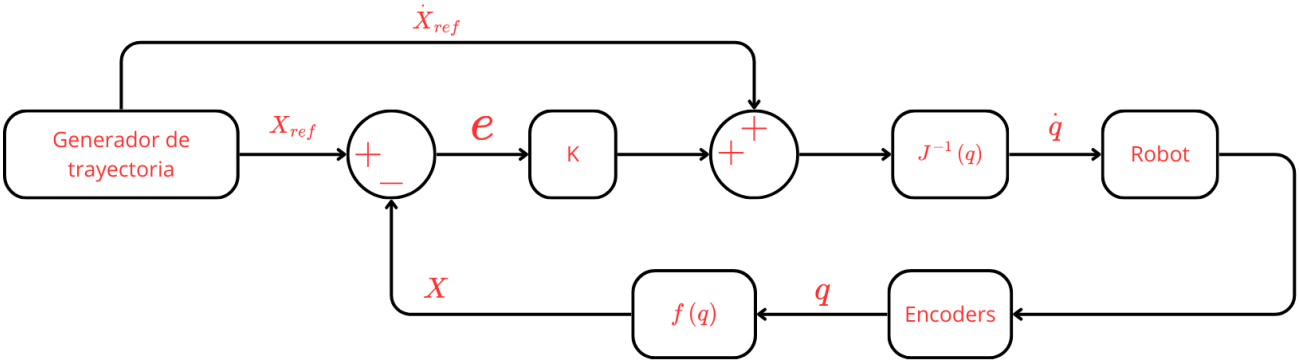

Figura 6: Diagramas de bloques para aplicar el RRMC. Elaboración propia.

Para garantizar que el robot cumpla con los requisitos del movimiento solicitado, se define una ley de control con realimentación del estado de las articulaciones en cada instante de tiempo. A través de esta ley, se obtienen las velocidades articulares necesarias para que el efector final siga una trayectoria definida en el espacio de trabajo del robot. La dinámica del error en este sistema se modela de la siguiente manera:

$$e = X_{ref} - X \;\rightarrow\; e = X_{ref} - f(q)$$

$$\dot{e} = \dot{X}_{ref} - \dot{X} \;\rightarrow\; \dot{e} = \dot{X}_{ref} - J(q) \cdot \dot{q}$$

Donde X_{ref} y $\dot{X}_{ref}$ representan, respectivamente, la pose y las velocidades cartesianas de referencia generadas por el planificador de trayectorias. Por otro lado, X es la pose actual del efector final, calculada mediante la aplicación de la cinemática directa sobre las posiciones angulares actuales de las articulaciones, medidas mediante encoders. La derivada de la pose actual, $\dot{X}$, puede representarse como el producto entre la matriz jacobiana $J(q)$ y las velocidades articulares $\dot{q}$.
El objetivo principal en este tipo de sistemas es lograr que el error e converja a cero conforme pasa el tiempo. Para ello, se impone que la dinámica del error se comporte como un sistema lineal de primer orden, definido como:

$$\dot{e} + K_p \cdot e = 0$$

Esta ecuación diferencial de primer orden tiene una solución general de la forma:

$$e(t) = e(0) \cdot e^{-K_p \cdot t}$$

La solución muestra que el error disminuye exponencialmente hacia cero a medida que transcurre el tiempo, dependiendo del valor de la ganancia proporcional K_p, la cual determina el ritmo con el que el error decae a cero. Entonces, un K_p más grande genera una convergencia más rápida del error. Es importante señalar que, al modelar el sistema como lineal de primer orden, se desprecian las no linealidades presentes en el error.

Para garantizar este comportamiento, se obtiene la ley de control que asegura la dinámica deseada del error:

$$\dot{e} + K_p \cdot e = 0$$

$$\dot{X}_{ref} - J(q) \cdot \dot{q} + K_p \cdot \left(X_{ref} - f(q)\right) = 0$$

Resolviendo para $\dot{q}$:

$$\dot{q} = J^{-1}(q) \cdot \left(\dot{X}_{ref} + K_p \cdot \left(X_{ref} - f(q)\right)\right)$$

De esta manera, para que la dinámica del error se comporte de forma deseada, las velocidades articulares $\dot{q}$ del robot deben cumplir esta ley de control en cada instante de tiempo, asegurando así que el efector final siga la trayectoria definida. Cabe destacar que la matriz jacobiana $J(q)$ debe ser invertible, lo que implica que el robot no debe encontrarse en una configuración singular.

2.3.3. Simulación en Robotics Toolbox

Aunque no fue posible implementar este método de control directamente sobre el brazo robótico construido debido a las limitaciones mencionadas al inicio de esta sección, se desarrolló un código basado en la teoría presentada en el apartado anterior. Este código fue implementado en Python y simulado utilizando la librería Robotics Toolbox. A continuación, se presenta la simulación en la que el robot describe una línea recta en el espacio de trabajo, moviéndose desde la posición inicial $(22{,}76\,cm,\ 0\,cm,\ 26{,}92\,cm)$, con una orientación definida por un ángulo de cabeceo de $-62°$, hasta la posición final $(24\,cm,\ -3{,}4\,cm,\ 2\,cm)$, con un ángulo de cabeceo de 0°. El primer paso realizado fue generar una trayectoria en el espacio cartesiano, correspondiente a una línea recta. Para ello, siguiendo los principios teóricos explicados previamente, se definieron las trayectorias en términos de sus componentes de posición y velocidad. Estas trayectorias fueron perfiladas mediante una interpolación trapezoidal entre la pose inicial y la pose final, asegurando un movimiento suave y controlado. Los resultados de esta simulación se muestran en las siguientes gráficas.

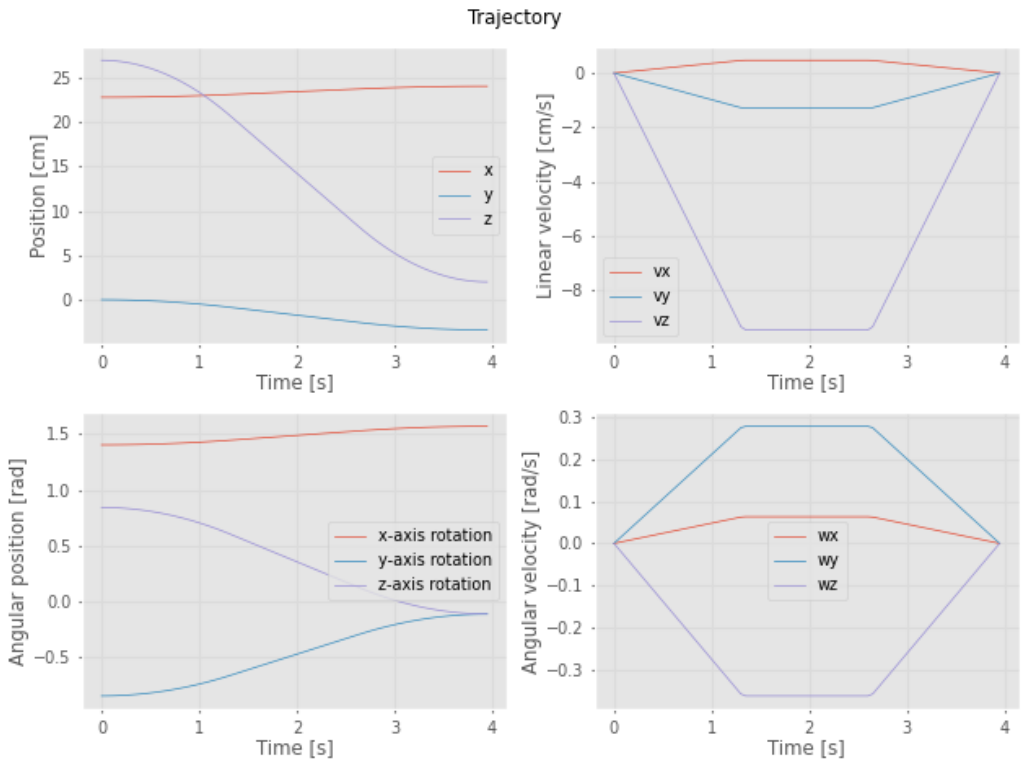

Figura 7: Generación de trayectorias en el espacio cartesiano. Elaboración propia.

De esta manera, al aplicar el método de Resolved Rate Motion Control (RRMC) con una ganancia proporcional $K_p = 6$, un tiempo total de $4\,segundos$, un intervalo de tiempo $\Delta t = 0{,}05\,segundos$ y un umbral ($threshold$) de 0,1, se realizó la simulación. En ella, se debía observar la trayectoria deseada, correspondiente a una línea recta, representada por los puntos por los que el brazo debió pasar para completar el recorrido. Estos puntos son calculados y ajustados en tiempo real conforme el efector final del brazo avanzaba hacia la consecución de la trayectoria programada.

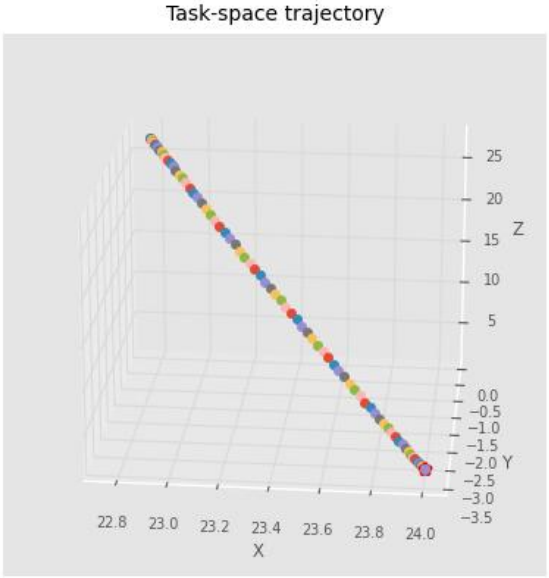

Figura 8: Aplicación del RRMC. Elaboración propia.

Como se aprecia en los resultados de la simulación, la trayectoria seguida por el efector final del brazo corresponde efectivamente al requisito de seguir una línea recta en el espacio cartesiano. Además, para verificar que se cumple con la orientación deseada en la pose final, se puede observar que la última pose calculada muestra un ángulo de cabeceo de 0°, coincidiendo con el valor esperado.

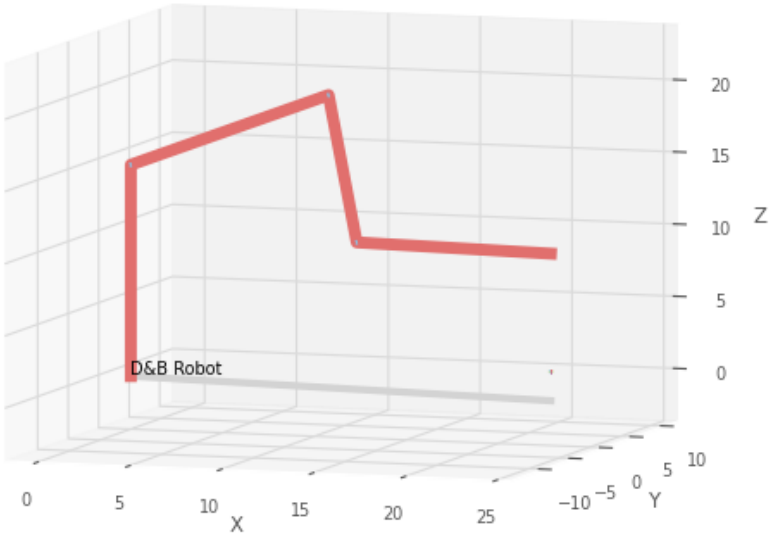

Figura 9: Última pose resultante de la simulación efectuada. Elaboración propia.

Además, resulta útil visualizar la evolución de las coordenadas articulares y cartesianas, asegurándose de que estas últimas coincidan con los valores calculados previamente.

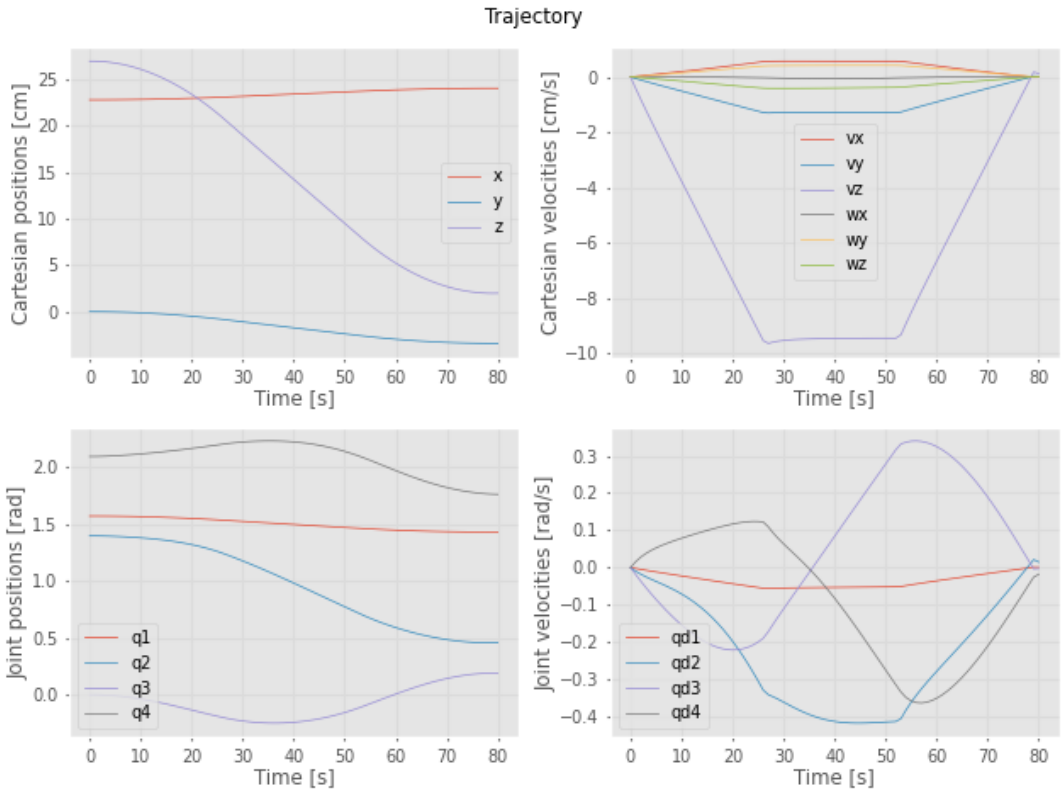

Figura 10: Evolución de las coordenadas cartesianas y articulares en la simulación. Elaboración propia.

De esta manera, se completó satisfactoriamente la simulación del método de Resolved Rate Motion Control (RRMC), demostrando su eficacia para el seguimiento de trayectorias específicas en el espacio cartesiano. Aunque, como se mencionó anteriormente, no fue posible implementarlo de manera exitosa en el robot construido, este control desarrollado representa una base sólida que podría aplicarse en futuras versiones mejoradas del brazo robótico.

2.4. Marcadores ArUco

Los marcadores ArUco son patrones bidimensionales diseñados para aplicaciones de visión por computadora, ampliamente utilizados en tareas como la localización y la navegación en robótica, realidad aumentada y calibración de cámaras. Estos marcadores consisten en un cuadrado negro con un patrón interno distintivo, generalmente codificado como una matriz binaria única. Este diseño permite que cada marcador sea identificado de manera precisa y diferenciada en una escena, incluso en entornos complejos o con ruido visual.

Una de las aplicaciones más destacadas de los marcadores ArUco es la detección y estimación de la pose del marcador respecto a una cámara. La pose se define como la posición y orientación tridimensional de un objeto en el espacio. Para lograrlo, el sistema de detección identifica las esquinas del marcador en la imagen capturada y las asocia con su configuración conocida en el espacio tridimensional. Este proceso se basa en algoritmos de visión por computadora que emplean técnicas de correspondencia geométrica y la proyección de los puntos 3D sobre el plano de la imagen, utilizando el modelo intrínseco de la cámara. Una vez identificadas las esquinas, se resuelve la relación entre el marcador y la cámara mediante el algoritmo Perspective-n-Point (PnP), lo que permite calcular la matriz de transformación que describe la posición y orientación del marcador.

El resultado de este proceso es la obtención de información de pose, típicamente representada mediante un vector de traslación y un vector de rotación, que describen cómo el marcador está ubicado y orientado en relación a la cámara. Este nivel de precisión y fiabilidad en la estimación es fundamental en múltiples escenarios. Por ejemplo, en la realidad aumentada, la correcta detección de la pose permite superponer objetos virtuales sobre marcadores físicos con gran exactitud. En robótica, la información de pose se utiliza para tareas de navegación autónoma y manipulación, permitiendo a los robots interactuar de manera precisa con su entorno. En este proyecto en particular, los marcadores ArUco se utilizaron para establecer una referencia precisa de la pizarra donde se desarrolla el juego. Con este fin, se colocaron estratégicamente cuatro marcadores sobre la superficie de la pizarra.

Los marcadores ArUco destacan además por ser ligeros en términos computacionales, de fácil implementación y altamente personalizables en cuanto a tamaño y configuración, lo que los convierte en una herramienta versátil y eficiente para aplicaciones donde la detección robusta y en tiempo real de la pose es un requisito crítico. Esto los ha consolidado como una elección preferida en desarrollos tecnológicos que dependen de la interacción entre entornos físicos y sistemas digitales.

3. Desarrollo integral del proyecto

3.1. Diseño mecánico

3.1.1. Base

La base constituye el soporte principal del brazo robot y cumple varias funciones esenciales. Fue diseñada en SolidWorks teniendo en cuenta aspectos de funcionalidad, robustez y organización de componentes. Se compone de las siguientes partes principales:

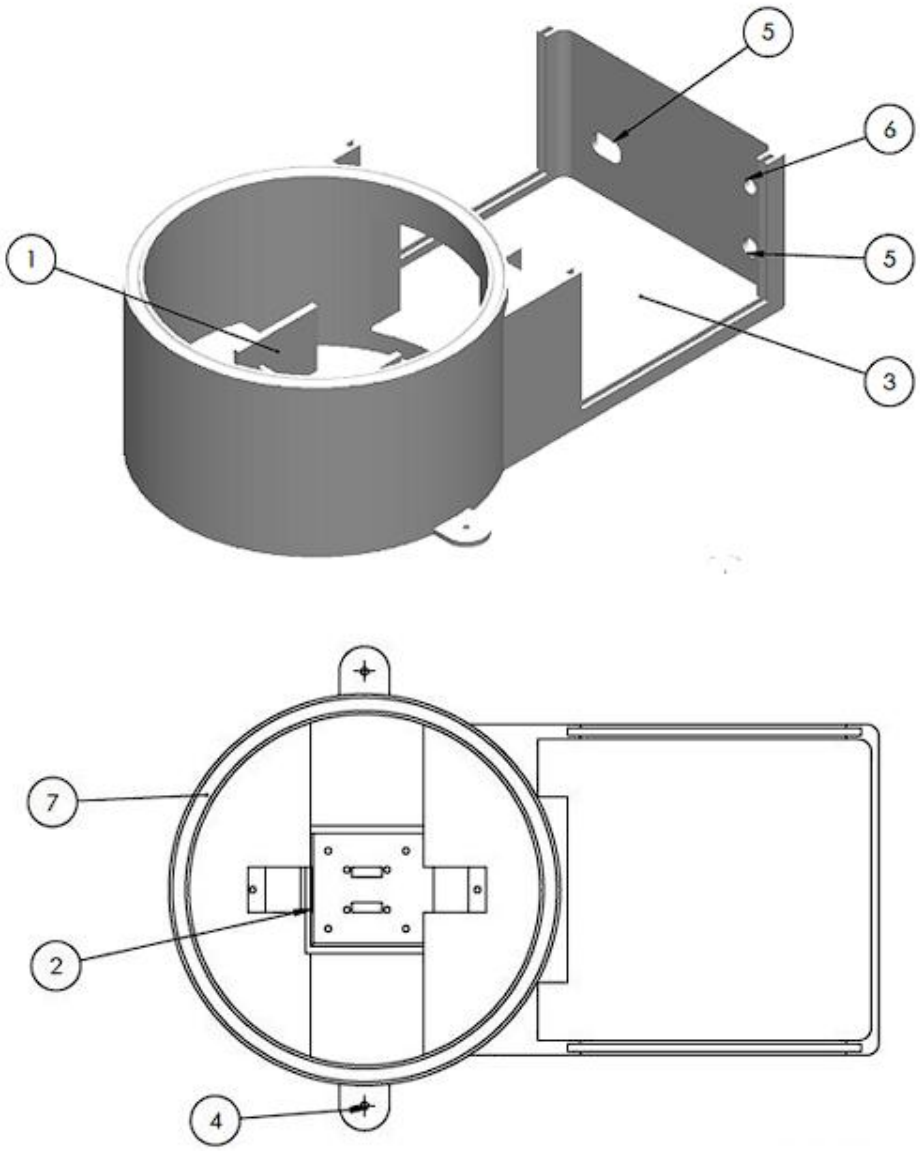

Figura 11: Diseño de la base. Elaboración propia.

1. **Soporte del motor paso a paso:** Esta sección está diseñada para fijar de manera segura el motor paso a paso que proporciona el movimiento rotacional a la base del brazo. La

geometría incluye orificios y refuerzos para asegurar el motor y evitar desplazamientos durante su funcionamiento.

2. **Zona de montaje del encoder:** En esta área se ubica el encoder, encargado de medir la posición angular del motor. El diseño contempla una colocación precisa y estable para garantizar la precisión en la detección de movimiento.
3. **Compartimento para la electrónica:** Se incluyó un espacio dedicado para ubicar los componentes electrónicos necesarios, como controladores, cables y módulos adicionales. Este compartimento cuenta con una disposición que facilita la organización y el mantenimiento del sistema.
4. **Zonas de fijación:** Para garantizar la estabilidad del brazo, la base incluye puntos de anclaje que permiten fijarla firmemente a una tabla de madera. Esto asegura que el brazo robot mantenga su posición durante la operación y minimiza las vibraciones.
5. **Entradas de conexión:** La base cuenta con dos entradas estratégicamente ubicadas: una para conectar el cable del ESP32 y otra para la fuente de alimentación. Estas entradas permiten una conexión ordenada y funcional.
6. **Espacio para la llave de encendido:** Se diseñó un espacio adicional para la instalación de una llave de encendido/apagado, proporcionando un control directo y accesible de la alimentación del sistema.
7. **Guía para rodamiento axial:** La base incluye una guía que permite integrar un rodamiento axial con la tapa de la base. Este diseño reduce la fricción y mejora el desplazamiento, asegurando un funcionamiento más fluido y eficiente del sistema rotacional.

3.1.2. Tapa con soporte principal para las articulaciones

La tapa y el soporte principal forman parte fundamental de la estructura del brazo robot, proporcionando soporte y funcionalidad a la primera articulación. Este diseño fue realizado en SolidWorks, integrando las siguientes características clave:

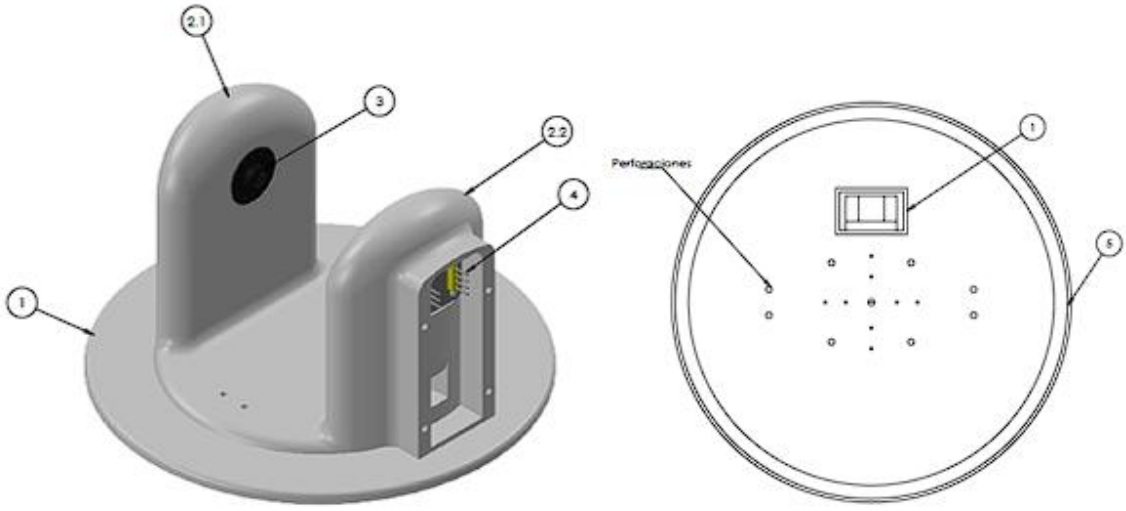

Figura 12: Diseño de la tapa y soporte principal. Elaboración propia.

1. **Tapa con perforaciones**: La tapa cuenta con perforaciones diseñadas para fijar un soporte que conecta el eje del motor de la base con la tapa. Este acople asegura un giro preciso, evitando deslizamientos o pérdida de pasos durante el movimiento. Además, cuenta con una perforación principal en donde pasan todos los cables del conexionado del brazo.
2. **Soporte principal de las articulaciones:** Este componente está dividido en dos partes:
    - La primera incluye un engranaje (3) diseñado para fijar y transmitir el movimiento del primer servomotor, correspondiente a la articulación del hombro.
    - La segunda incorpora una perforación que permite el paso ordenado de los cables hacia otras secciones del brazo. Además, en esta parte se ubica lateralmente la placa del encoder (4), facilitando su montaje y operación.
5. **Guía para rodamiento axial:** Similar a la base, la tapa incorpora una guía que permite el uso de un rodamiento axial. Esta guía está diseñada para alojar municiones de 5 mm, las cuales reducen significativamente la fricción y aseguran un movimiento suave y eficiente.

3.1.3. Hombro (shoulder)

La articulación del hombro conecta la base con las partes superiores del brazo robot y proporciona soporte y movimiento a la siguiente articulación. Este componente incluye las siguientes características:

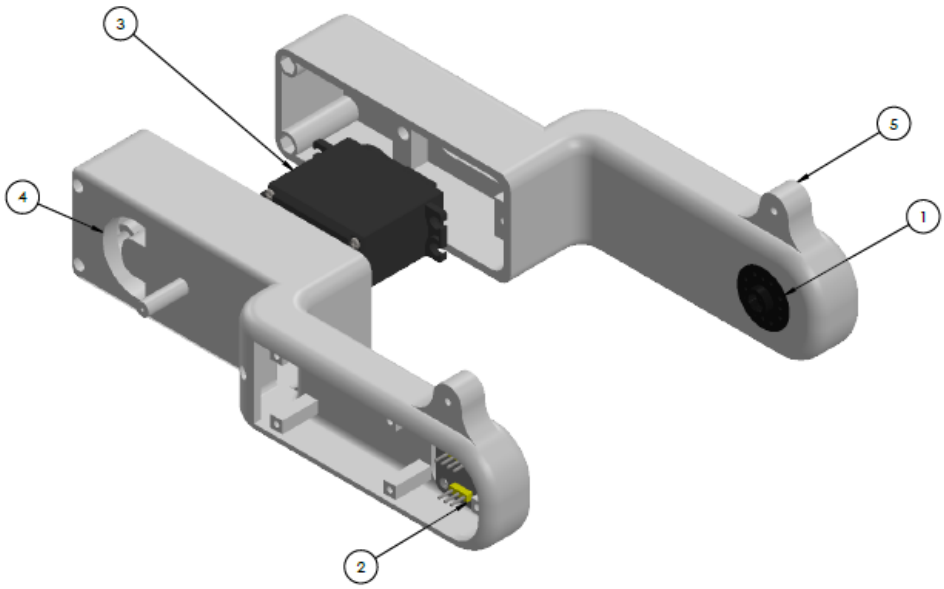

Figura 13: Diseño del hombro. Elaboración propia.

1. **Engranaje para la siguiente articulación:** Diseñado para encastrar el servomotor correspondiente, este engranaje facilita la transmisión de movimiento hacia la siguiente articulación, asegurando precisión y estabilidad.
2. **Ubicación del encoder:** La articulación incluye un espacio dedicado para la colocación del encoder, permitiendo el monitoreo exacto de la posición angular del hombro.

3. **Montaje del motor:** El diseño cuenta con un espacio específico donde el motor encastra en la pieza previamente descrita, integrándose de manera funcional con el sistema.
4. **Perforación para cables:** Se incorpora una perforación que facilita el paso de cables hacia otras partes del sistema, asegurando un diseño limpio y organizado.
5. **Zona para resortes:** La articulación cuenta con una zona especialmente diseñada para ubicar resortes. Estos ayudan a reducir la carga sobre el servomotor, evitando sobreesfuerzos y prolongando su vida útil.

3.1.4. Codo (elbow)

La articulación del codo comparte los mismos componentes principales que la articulación del hombro, incluyendo:

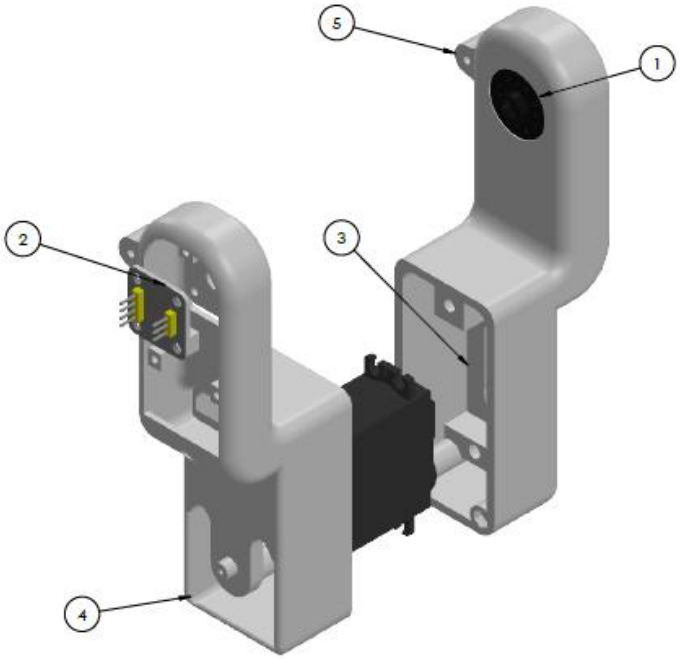

Figura 14: Diseño del codo. Elaboración propia.

1. **Engranaje para la siguiente articulación:** Diseñado para encastrar el servomotor y transmitir el movimiento hacia el siguiente segmento del brazo.
2. **Ubicación del encoder:** Proporciona una medición precisa de la posición angular del codo.
3. **Montaje del motor:** Integra un espacio específico donde el motor encastra en la pieza anterior.
4. **Perforación para cables**: Facilita el paso ordenado de cables hacia otras secciones del sistema.
5. **Zona para resortes**: Incluye un espacio para ubicar resortes que reducen la carga sobre el servomotor, asegurando un funcionamiento más eficiente.

La principal diferencia con respecto al hombro es su longitud más corta. Esto no solo reduce el peso que debe soportar el motor, sino que también mejora la maniobrabilidad del brazo robot

en espacios reducidos, permitiendo movimientos más precisos y rápidos en operaciones específicas.

3.1.5. Gripper

La última articulación del brazo robot, el gripper, está diseñada para manejar un marcador (fibrón) y garantizar su funcionamiento preciso en la tarea de trazar líneas sobre una pizarra. Sus componentes principales son:

1. **Soporte del motor:** El diseño incluye un soporte robusto que fija el motor MG996 en su lugar, garantizando estabilidad durante su funcionamiento.
2. **Motor MG996:** Este motor proporcionando la fuerza necesaria para mover el fibrón con precisión.
3. **Salida de cables:** Se incorpora una perforación para el paso de los cables del motor, garantizando un diseño limpio y ordenado.
4. **Pieza fija:** Incluye una pieza que sostiene firmemente el sistema deslizante, asegurando estabilidad y precisión en el movimiento del fibrón.
5. **Sistema deslizante:** Diseñado específicamente para alojar y mover el fibrón, este sistema permite ajustes en la posición del marcador, asegurando un contacto adecuado con la superficie de la pizarra.
6. **Resorte amortiguador:** Incluye un resorte que amortigua la presión del fibrón sobre la pizarra. Esto no solo evita daños en la punta del marcador, sino que también asegura que la línea trazada sea uniforme, incluso en superficies ligeramente irregulares.

El diseño del gripper destaca por su funcionalidad y precisión, asegurando que el fibrón pueda adaptarse a la superficie con la presión adecuada. Además, el resorte reduce el desgaste en los componentes del sistema, prolongando su vida útil y mejorando el rendimiento en aplicaciones repetitivas.

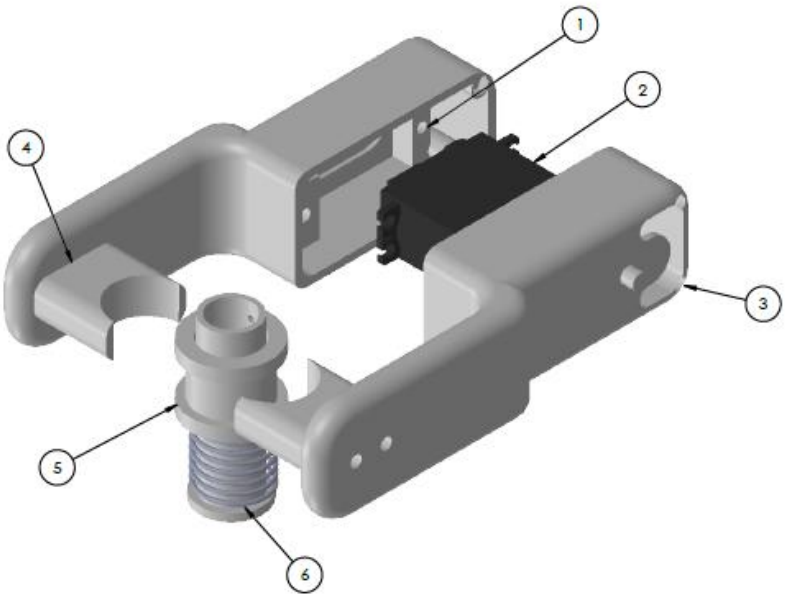

Figura 15: Diseño del efector final. Elaboración propia.

2.1.6. Ensamble completo

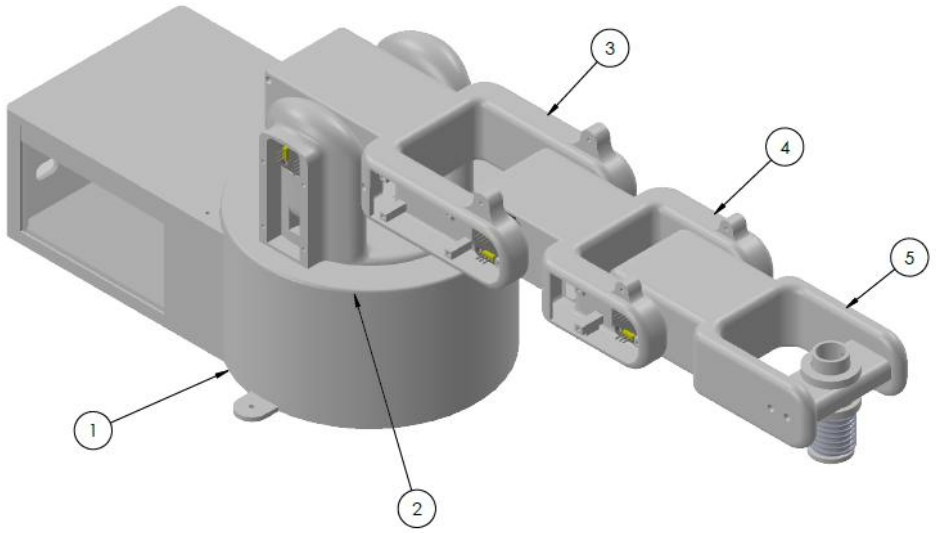

Figura 16: Ensamble completo. Elaboración propia.

1. Base.
2. Tapa con soporte principal para las articulaciones.
3. Hombro (shoulder).
4. Codo (elbow).
5. Gripper.

2.1.7. Mesa

Se diseñó y fabricó una mesa sobre la cual se apoyará la pizarra. Esta adición optimiza el funcionamiento del sistema, ya que reduce la distancia que debe recorrer la segunda articulación del brazo para trazar las líneas, lo que contribuye a alargar su vida útil al disminuir su esfuerzo. Las dimensiones de la mesa son 30 x 40 x 8.8 cm, con el ancho y largo adecuados para garantizar que la pizarra se apoye de manera estable y precisa sobre ella.

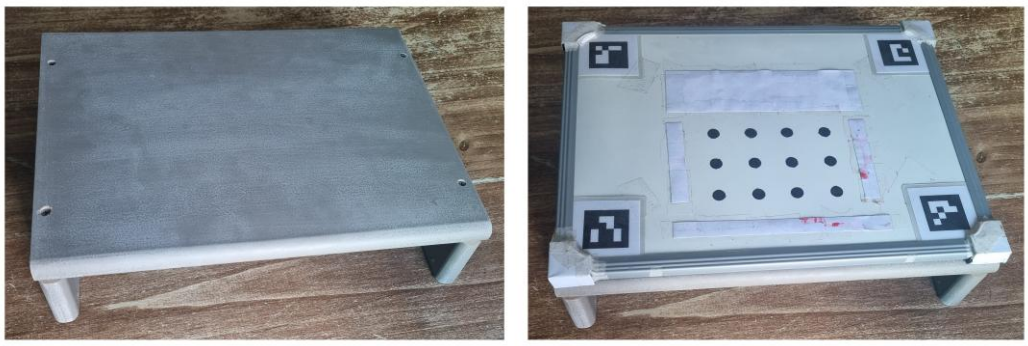

Figura 17: Diseño de la mesa. Elaboración propia.

2.1.8. Topes

Para conectar el brazo robótico con la mesa que sostiene la pizarra, se diseñaron topes que se fijan mediante tornillos. Un extremo de cada tope se atornilla a la base del robot, mientras que el otro se asegura a la mesa. De este modo, se garantiza que la referencia permanezca fija en todo momento, facilitando además el proceso de montaje del set-up.

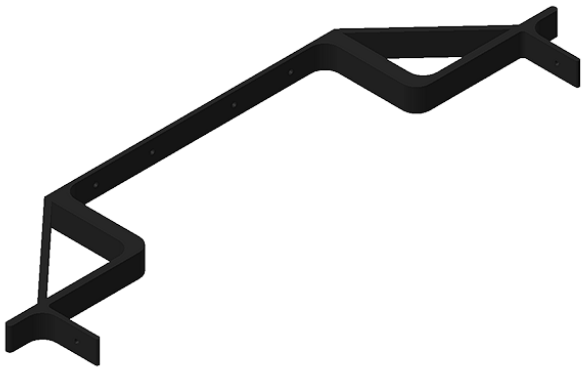

Figura 18: Diseño de los topes. Elaboración propia.

2.1.9. Soporte para resortes

Para asistir al motor de la segunda articulación del brazo robótico, correspondiente al hombro, se diseñaron e incorporaron soportes específicos para colocar resortes. Estos soportes consisten en estructuras verticales que permiten fijar los resortes de manera adecuada, asegurando que la tensión generada por los mismos sea aplicada correctamente para ayudar al servomotor durante el movimiento de elevación.

El propósito principal de los resortes es reducir la carga directa sobre el servomotor, compensando parte del peso de la estructura del brazo y cualquier carga adicional. Esto evita que el motor tenga que realizar todo el esfuerzo por sí solo, especialmente en movimientos hacia arriba.

Este sistema no solo mejora el rendimiento mecánico del brazo, sino que también protege los componentes internos del servomotor, como los engranajes de plástico, que de otro modo estarían sometidos a un mayor desgaste por el esfuerzo constante. Al minimizar la carga, los resortes prolongan la vida útil del motor y mejoran su eficiencia energética, permitiendo movimientos más suaves y controlados.

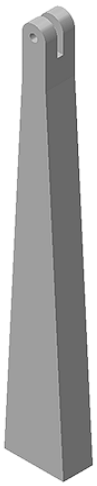

Figura 19: Diseño del soporte para resortes. Elaboración propia.

3.2. Diseño electrónico

3.2.1. Esquema del conexionado eléctrico

A continuación, se presenta el esquema de conexión eléctrica implementado para el funcionamiento del brazo robótico:

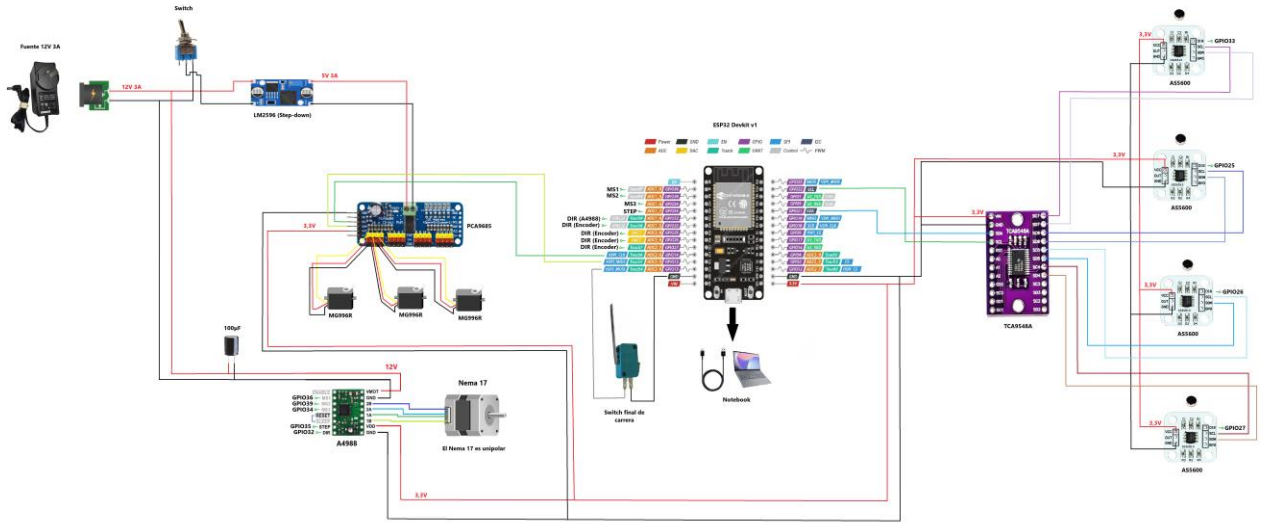

Figura 20: Esquema del conexionado eléctrico. Elaboración propia.

Los componentes utilizados para la implementación del sistema fueron:

- 1 switch normalmente abierto.
- 1 fuente de 12V 3A.
- 1 step-down LM2596.
- 1 controlador de servomotores PCA9685.

- 3 servomotores MG996R.
- 1 capacitor electrolítico de 100µF.
- 1 controlador de motores paso a paso Pololu A4988.
- 1 motor paso a paso unipolar Nema 17.
- 1 microcontrolador ESP32.
- 1 switch final de carrera.
- 1 multiplexor de I²C TCA9548A.
- 4 encoders magnéticos AS5600.

Para cada módulo empleado, se confeccionaron plaquetas de cobre perforadas, sobre las cuales se montaron los componentes. Las conexiones entre los módulos se realizaron mediante cables Dupont fabricados manualmente, asegurando la compatibilidad y robustez del ensamblaje.
Este enfoque modular no solo facilita el montaje, sino que también permite una mayor flexibilidad para realizar modificaciones o actualizaciones en el sistema, mejorando su mantenibilidad y escalabilidad en futuros desarrollos.

3.2.2. Fuente de alimentación

Se utilizó una fuente de 12V 3A para alimentar el sistema. La potencia máxima que puede suministrar esta fuente se calcula como:

$$P = V \cdot I = 12V \cdot 3A = 36W$$

Esta potencia es suficiente para alimentar los componentes principales del sistema, ya que la mayor parte de la carga será manejada por los 3 servomotores MG996R, que recibirán la energía a través de un regulador step-down.
El uso de una fuente de 3A se considera adecuado porque, en condiciones normales, los servos no están bajo máxima carga todo el tiempo. Aunque la corriente máxima de un MG996R puede llegar a 2,5A (stall current), en esta aplicación se ha observado que no supera 1A por servo. Esto se debe a que no todos los servomotores están sometidos a máxima carga simultáneamente.
Además, para facilitar la operación del sistema, se incorporó un switch de encendido/apagado en la salida de la fuente de 12V. Este switch permite interrumpir la corriente eléctrica hacia el sistema de control, garantizando seguridad y facilidad de uso.

3.2.3. Módulo step-down

El step-down es un convertidor DC-DC que reduce el voltaje de 12V a 5V, necesario para alimentar el controlador PCA9685 y los 3 servomotores MG996R. Este módulo asegura una regulación estable, ya que los servomotores requieren una alimentación de 4,8V a 6V para operar correctamente.

3.2.4. Controlador de servomotores

El controlador PCA9685 recibe la alimentación de 5V desde el regulador step-down y se encarga de enviar señales PWM (modulación por ancho de pulso) a los 3 servomotores MG996R. Este controlador permite manejar múltiples servomotores desde un solo bus I^2C, facilitando la expansión del sistema.

- Alimentación: El PCA9685 se alimenta con 5V (V+ para los servomotores) y 3,3V para la lógica (V_{cc}).
- Control I^2C: Se comunica con el microcontrolador (ESP32) a través del bus I^2C, lo que permite controlar hasta 16 servos.

El PCA9685 tiene un consumo nominal muy bajo (aproximadamente 1mA a 3mA para la parte lógica), ya que su función principal es controlar los pulsos PWM para los servos. La corriente principal la consumen los propios servomotores.

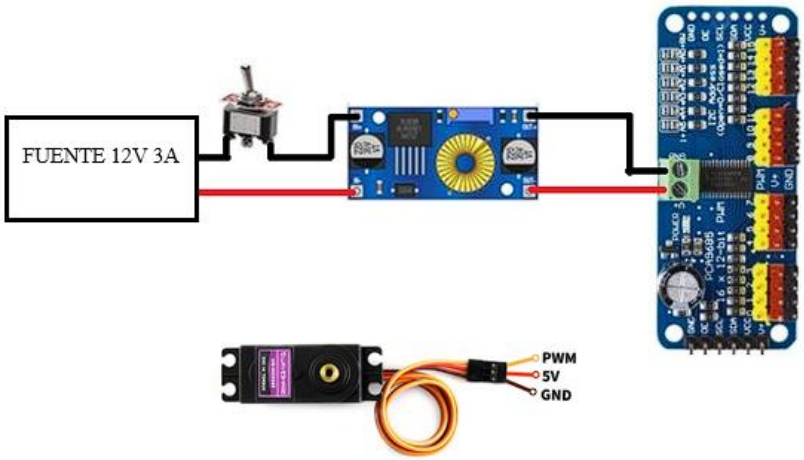

Figura 21: Conexión del controlador de servomotores PCA9685. Elaboración propia.

3.2.5. Servomotores

Los 3 servomotores MG996R reciben la señal PWM desde el PCA9685 y la alimentación desde el regulador step-down. La corriente de estos servomotores varía según la carga aplicada:

- Consumo en reposo: 100mA a 200mA.
- Consumo en operación normal: Aproximadamente 500mA a 1A (según la carga).
- Consumo máximo (stall current): 2,5A, pero este valor se presenta solo en situaciones extremas (bloqueo del motor).

3.2.6. Motor paso a paso

El sistema cuenta con un motor paso a paso NEMA 17 controlado por un driver A4988. La alimentación del driver se conecta directamente a la fuente de 12V, sin pasar por el regulador step-down.

- VMOT: 12V directamente desde la fuente.
- GND: Tierra común para todo el sistema.
- VDD: Alimentación lógica del A4988 (3,3V), que proviene del ESP32.
- STEP/DIR: Señales de control enviadas desde el ESP32 para controlar la velocidad y la dirección del motor.
- Corriente nominal del motor: 1,7A por fase.

El controlador A4988 permite regular la corriente mediante el ajuste de un potenciómetro. En esta configuración, la corriente se limita a un nivel seguro para evitar sobrecalentamiento, generalmente inferior a la corriente máxima nominal. Su cálculo es el siguiente:

$$V_{ref} = I_{max} \cdot (8 \cdot R_s) = 1{,}5A \cdot (8 \cdot 0{,}1) = 1{,}2V$$

3.2.7. Microcontrolador principal

El ESP32 actúa como el cerebro del sistema, siendo el encargado de gestionar toda la lógica de funcionamiento. Este microcontrolador se conecta y coordina con varios componentes clave para garantizar el control adecuado del sistema. Entre estos, se encuentra el módulo PCA9685, al cual se conecta a través del bus I^2C para manejar el control de los servomotores. Además, el ESP32 también interactúa con el driver A4988 mediante pines digitales específicos, denominados STEP y DIR, que permiten controlar con precisión el funcionamiento de un motor paso a paso.

La alimentación del ESP32 se realiza mediante un puerto USB que se conecta directamente a la computadora. A través de esta conexión, el dispositivo recibe un suministro eléctrico de 5 voltios y una corriente de hasta 500 miliamperios, lo cual es suficiente para alimentar la lógica de control del ESP32. Además, la bornera de conexión juega un papel crucial en la distribución de esta alimentación, ya que permite enviar la energía desde el ESP32 hacia las placas de control, específicamente el A4988 y el PCA9685.

3.2.8. Encoders magnéticos

Para ampliar la capacidad del bus I2C y permitir la conexión de cuatro encoders al ESP32, se utilizó un multiplexor. Esto fue necesario debido a que los encoders comparten la misma dirección I^2C y no es posible modificarla.

En cuanto a la alimentación, el sistema utiliza una fuente común proporcionada por el ESP32. El voltaje de operación, que puede ser de 3,3V o 5V dependiendo de los requerimientos, es suministrado por el ESP32. Asimismo, el sistema cuenta con una conexión a tierra (GND) compartida para garantizar un circuito eléctrico funcional y estable.

El control de la señal se realiza mediante los pines SDA y SCL del ESP32, que están conectados al multiplexor TCA9548A. Este componente es el encargado de habilitar la comunicación individual entre el ESP32 y cada encoder, aprovechando los canales independientes del bus I^2C proporcionados por el multiplexor.

Por último, los encoders reciben su alimentación directamente de la salida de 3,3V del ESP32, asegurando un suministro adecuado y consistente para su operación.

Figura 22: Conexión del multiplexor TCA9548A y encoders. Elaboración propia.

3.3. Diseño de software

El desarrollo del código se llevó a cabo principalmente en Python, mientras que el control de los motores y sensores del brazo se implementó en C++, utilizando la extensión PlatformIO. El sistema está compuesto por varias etapas, por lo que el código se estructura en diferentes secciones, cada una con un propósito específico. El código completo está disponible en el siguiente repositorio de GitHub:

- https://github.com/pedrotagliani/dots-and-boxes-robot

3.3.1. Calibración de la cámara

Para garantizar que las medidas obtenidas de la pose de los marcadores ArUco con respecto a la cámara sean lo más precisas posible, resulta fundamental realizar una calibración previa de la cámara. Este proceso se llevó a cabo implementando un código desarrollado con la librería OpenCV, ampliamente utilizada en aplicaciones de visión por computadora.
La calibración se basó en la captura de imágenes de un tablero ChArUco, un patrón híbrido que combina características de los tableros de ajedrez y los marcadores ArUco, lo que permite una detección precisa y robusta. Para ello, se imprimió el tablero y se tomaron múltiples imágenes desde diferentes ángulos y distancias utilizando la cámara en cuestión. Estas imágenes se ingresaron al algoritmo de calibración, el cual utiliza un enfoque basado en mínimos cuadrados para ajustar los parámetros intrínsecos y de distorsión de la cámara.
Como resultado de este proceso, se obtuvieron los parámetros intrínsecos (focal, centro óptico) y los coeficientes de distorsión radial y tangencial de la cámara. Estos parámetros son esenciales para corregir distorsiones en las imágenes capturadas y para calcular con precisión la pose de los marcadores en futuras operaciones.

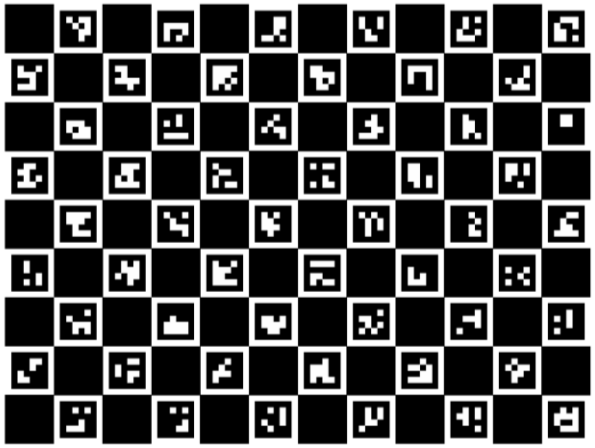

Figura 23: Tablero ChArUco de 9x12. Generado en calib.io.

3.3.2. Detección del tablero

Esta sección del código constituye uno de los componentes más críticos del sistema, ya que garantiza el correcto aislamiento del tablero de juego dentro de la imagen capturada. Para lograrlo, se utilizan los cuatro marcadores ArUco que delimitan la pizarra, estableciendo referencias que permiten identificar y extraer el tablero de manera precisa.
Cada marcador posee un ID único, lo que resulta esencial para evitar falsos positivos durante el proceso de detección. Conocer de antemano estos IDs asegura que el sistema reconozca únicamente los marcadores correctos, ignorando otros elementos similares presentes en la escena.
Cuando los cuatro marcadores son detectados simultáneamente, se recorta la imagen basándose en las posiciones de sus vértices, generando una nueva imagen que contiene exclusivamente el tablero de juego, eliminando elementos externos que puedan interferir en el análisis.
Posteriormente, se procesan los puntos del tablero, utilizando para ello la matriz de transformación homogénea asociada a cada marcador respecto a la cámara. Esto permite calcular con precisión la posición de cada marcador y, por extensión, la de los puntos en el tablero. La matriz de transformación actúa como un puente entre el espacio físico del tablero y el sistema de coordenadas de la cámara, lo que facilita ubicar los puntos con alta precisión.
Con esta información y sabiendo que la distancia entre los puntos del tablero es fija (3,6 cm tanto en las direcciones vertical como horizontal), se calculan las coordenadas de todos los puntos del tablero primero en referencia a los marcadores y, posteriormente, respecto a la cámara.
Es importante destacar que la detección de los marcadores ArUco puede fluctuar ligeramente entre capturas debido a variaciones en las condiciones de iluminación, ángulo de visión o ruido en la imagen. Estas fluctuaciones pueden afectar negativamente la precisión en la detección del tablero, ya que los puntos calculados mediante las matrices de transformación homogénea podrían no coincidir completamente con las posiciones reales.
Para mitigar este problema, se implementó un enfoque basado en el promedio de múltiples mediciones. Al calcular los puntos utilizando varias capturas consecutivas y promediar sus valores, se logró reducir significativamente el impacto de las variaciones en la detección. Como

resultado, los puntos obtenidos presentaron una alta correspondencia con las posiciones reales, siendo suficientemente precisos para tareas como la detección de las líneas dibujadas en el tablero.

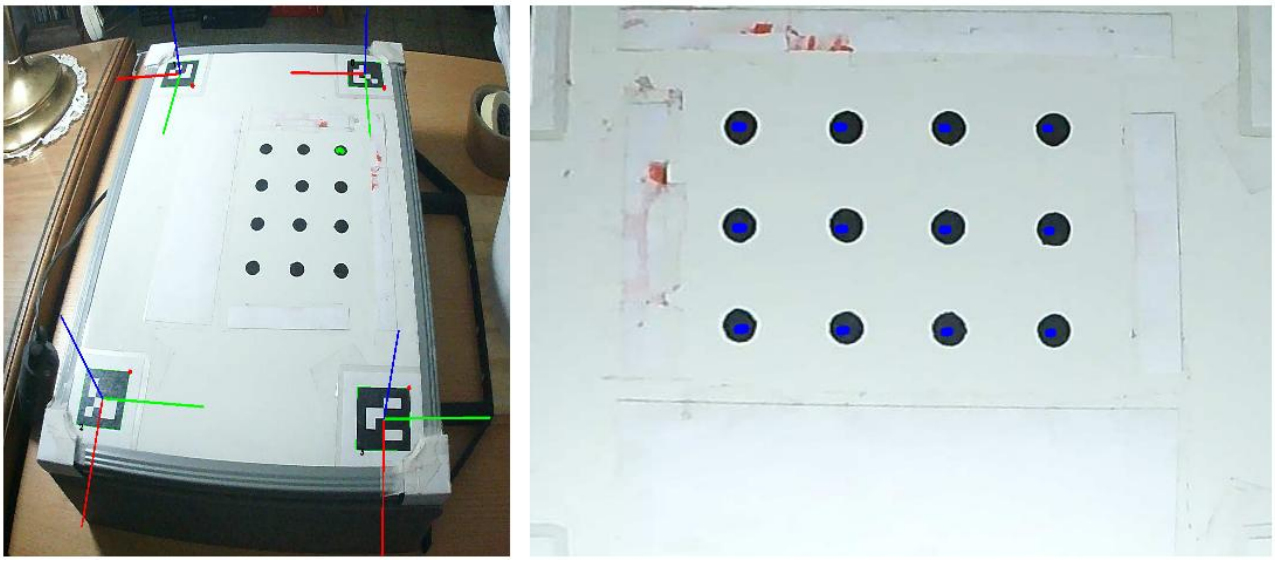

Figura 24: Detección del tablero. Elaboración propia.

3.3.3. Detección de las líneas

Una vez obtenida la matriz con los puntos promediados del tablero, es posible proceder con la detección de las líneas. En este proceso, las líneas se detectan de manera individual, una a la vez. Inicialmente, se identifican todas las líneas horizontales y luego las verticales, aunque el método de detección utilizado es idéntico para ambos casos.

Para detectar una línea, el primer paso es seleccionar los dos puntos que la delimitan. Esto es posible gracias a que en esta etapa ya se dispone de la matriz de puntos del tablero, cuyos valores promediados garantizan una proximidad adecuada a las posiciones reales. Con los dos puntos seleccionados, se realiza un recorte del tablero para aislar la región correspondiente a esos puntos del resto de la imagen. Este recorte se define añadiendo un padding (margen) tanto en la altura como en la anchura, generando un rectángulo que abarca la línea a detectar.

Dentro de este rectángulo, se aplica un proceso de thresholding y, posteriormente, se utiliza la función *HoughLinesP* de la librería OpenCV para realizar la detección de la línea. En esta función, es necesario ajustar parámetros clave, como la longitud mínima de la línea, la cual se configuró para ser al menos el 70% de la dimensión más larga del rectángulo. Esta dimensión coincide con la orientación de la línea que se pretende identificar. Si la línea es detectada de manera satisfactoria, se registra y se actualiza el estado del juego en consecuencia.

El método funciona de manera eficiente para líneas dibujadas en color negro. Sin embargo, no es igualmente eficaz para detectar líneas de color rojo, que es el utilizado por el robot en su efector final. Para solucionar este problema, antes de aplicar el proceso de thresholding, se realiza una detección de color mediante máscaras. Estas máscaras se configuran con límites inferiores y

superiores que corresponden al color de interés; en este caso, el rojo. Este enfoque permitió detectar de manera confiable las líneas dibujadas por el robot.
Además, esta técnica es adaptable a otros colores. Modificando las máscaras, se pueden configurar los parámetros necesarios para detectar líneas de distintos colores según el tono utilizado por el efector final del robot. Este nivel de flexibilidad amplía las capacidades del sistema, haciéndolo robusto para diversas configuraciones y aplicaciones.

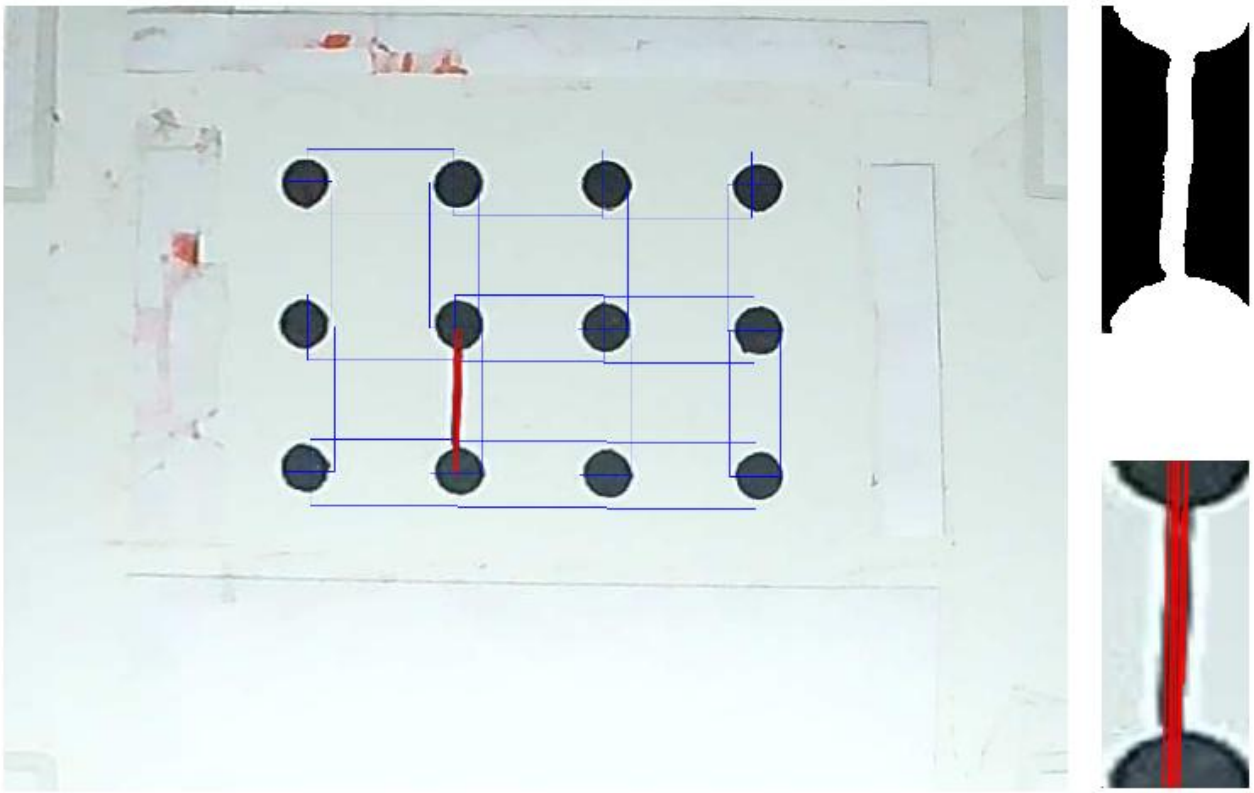

Figura 25: Detección de líneas. Elaboración propia.

3.3.4. Motor de juego

Para gestionar los movimientos del brazo robótico durante el juego, se empleó un motor de juego de puntos y cajas desarrollado en Python llamado dnbpy, el cual está disponible en GitHub. Este motor incluye cuatro tipos de agentes con diferentes niveles de dificultad, detallados a continuación:

- Jugador aleatorio: selecciona un borde al azar.
- Jugador heurístico nivel 1: elige un borde que complete una caja si es posible; en caso contrario, selecciona uno al azar.
- Jugador heurístico nivel 2: selecciona un borde que complete una caja si es posible; de no ser factible, elige un borde que no permita al oponente completar una caja; si ninguna de estas opciones está disponible, selecciona uno al azar.

- Jugador con algoritmo Minimax nivel 3: utiliza una búsqueda minimax con poda alfa-beta limitada en profundidad para estimar el valor de una posición del tablero.

Esta librería demostró ser muy versátil y se integró con éxito en el sistema desarrollado para el control del brazo robótico. Sin embargo, fue necesario realizar diversos cálculos y adaptaciones para convertir la convención utilizada en este proyecto para detectar y referenciar las líneas del tablero al formato estándar del motor de juego.

Una vez implementada correctamente, la librería permitió aprovechar múltiples funcionalidades, como consultar el estado del tablero, realizar movimientos basados en el agente seleccionado, visualizar el marcador, entre otras características.

3.3.5. Comunicación con el brazo robótico

Se implementó un sistema de comunicación serial como enlace entre el código desarrollado en Python, encargado de generar los comandos de movimiento para el brazo robótico, y el código alojado en el microcontrolador, programado mediante la extensión de PlatformIO. Este último se encarga de controlar los motores y sensores del brazo, permitiendo una ejecución precisa de las acciones requeridas.

El esquema de comunicación se basa en el envío de caracteres específicos desde Python a través del puerto serie. Cada carácter corresponde a una instrucción particular que es interpretada por el microcontrolador para realizar acciones como el desplazamiento de un motor, la lectura de un sensor, o la ejecución de secuencias predefinidas. Este diseño modular permite que el sistema cuente con múltiples opciones de operación, habilitando una interacción dinámica entre las capas de software.

La comunicación es bidireccional, lo que significa que no solo se transmiten comandos desde Python hacia el microcontrolador, sino que este también puede enviar información de retorno. Esto incluye datos de estado del sistema, lecturas de los sensores, o confirmaciones de ejecución de comandos. Este flujo de datos en ambas direcciones asegura una sincronización efectiva entre los dos componentes, permitiendo la detección y corrección de posibles desajustes durante la operación.

El uso de esta estrategia de comunicación garantiza una integración fluida entre los módulos de control y hardware del brazo robótico, maximizando la versatilidad del sistema. Además, el enfoque implementado facilita futuras expansiones, como la incorporación de nuevos comandos o la mejora de la resolución y precisión en el intercambio de datos, todo sin comprometer la robustez del sistema actual.

3.3.6. Control del robot

El proceso de inicialización del brazo robótico comienza al encenderlo, momento en el cual se dirige automáticamente a su posición home. Los servomotores alcanzan esta posición de manera inmediata, gracias a que cuentan con encoders internos que proporcionan la retroalimentación necesaria para guiar el movimiento. En contraste, el motor paso a paso de la base no dispone de

un sistema de posicionamiento interno, por lo que se incorporó un switch final de carrera normalmente abierto para establecer el punto de referencia inicial. En este proceso, el motor paso a paso comienza a girar en sentido antihorario a velocidad constante hasta que activa el switch final de carrera, definiendo este punto como los 0° del eje de rotación de la base. Una vez alcanzado este punto, el motor avanza hasta los 90°, que corresponden a la posición home de la base. Una vez que todo el brazo se encuentra en esta posición inicial, el sistema queda a la espera de recibir las instrucciones enviadas desde el programa en Python.

Los movimientos necesarios para que el brazo participe en el juego se controlan utilizando librerías especializadas que optimizan su funcionamiento. La librería AccelStepper se utiliza para gestionar el motor paso a paso, permitiendo un control preciso y confiable de su movimiento. Por otro lado, los servomotores se controlan mediante la librería ServoEasing, que facilita la interpolación suave entre dos puntos de movimiento. Esta última ofrece configuraciones completamente ajustables, lo que elimina la necesidad de realizar interpolaciones manuales. La implementación de estas dos librerías permite programar funciones tanto bloqueantes como no bloqueantes, lo que brinda la flexibilidad de operar los motores de manera simultánea o independiente, según sea más conveniente para los movimientos requeridos.

Además, se añadieron encoders externos que verifican si el brazo se posiciona correctamente en home al inicio del proceso. Este mecanismo actúa como una capa adicional de seguridad para detectar y corregir posibles fallas en el posicionamiento inicial, incrementando la confiabilidad del sistema.

3.4. Funcionamiento y uso

El proceso para jugar una partida contra el brazo robótico fue diseñado con el objetivo de ser lo más intuitivo y sencillo posible, ofreciendo al usuario una experiencia guiada a través de mensajes informativos impresos en la terminal. Estos mensajes facilitan la interacción y aseguran que el juego se desarrolle de manera fluida.

Al inicio de la partida, se solicita al usuario que verifique las condiciones de iluminación y la calidad de captura de la cámara. Es fundamental que los marcadores ArUco sean claramente visibles en la imagen. En la imagen brindada se muestra en tiempo real la cantidad de marcadores detectados, permitiendo al usuario confirmar que los cuatro marcadores están siendo reconocidos de manera constante. Esto asegura que el sistema está en condiciones óptimas para comenzar. Como recomendación, se sugiere colocar la cámara lo más cerca posible del tablero, ya que una menor distancia mejora significativamente la precisión de las mediciones obtenidas.

Una vez confirmada la correcta detección de los marcadores, el algoritmo verifica el estado inicial del tablero. Si se encuentra completamente vacío, el juego puede comenzar. En caso contrario, se solicita al usuario que limpie cualquier interferencia presente, como líneas dibujadas previamente, garabatos o elementos ajenos al tablero que puedan afectar la detección. Solo cuando el tablero está despejado, el juego se inicia según lo configurado, ya sea con el robot o el jugador humano realizando el primer movimiento.

Durante el desarrollo de la partida, el sistema alterna los turnos entre el robot y el jugador. Cada vez que un jugador completa una caja, el resultado parcial del marcador se actualiza y se muestra en la terminal. Al finalizar la partida, se imprime el resultado final, ofreciendo un resumen del desempeño de ambos participantes.
Es importante mencionar que el sistema detecta una línea a la vez, lo que implica que se debe esperar a que el algoritmo confirme la detección antes de continuar con el siguiente turno. Este proceso también se comunica claramente a través de la terminal, manteniendo al usuario informado en todo momento sobre el estado del juego y asegurando una experiencia transparente y comprensible. Este enfoque estructurado garantiza un desarrollo ordenado de la partida, promoviendo una interacción efectiva entre el usuario y el sistema.

4. Evaluación del proyecto

4.1. Resultados obtenidos

Después de realizar diversas pruebas en el sistema completo, evaluando tanto el brazo robótico como el código para medir su desempeño en partidas reales, los resultados obtenidos fueron satisfactorios. El brazo demostró ser capaz de dibujar con precisión las líneas correspondientes, cumpliendo con las acciones programadas. Asimismo, el código, incluyendo tanto el algoritmo de detección como la implementación del motor de juego, funcionó de manera efectiva, logrando un desempeño acorde a las expectativas.
En promedio, una partida tiene una duración aproximada de poco más de 2 minutos, un tiempo considerado aceptable y que evidencia tanto la velocidad de acción del brazo como la eficacia del sistema de detección del tablero. Este último se comportó de forma consistente, permitiendo el desarrollo normal de las partidas sin interrupciones.

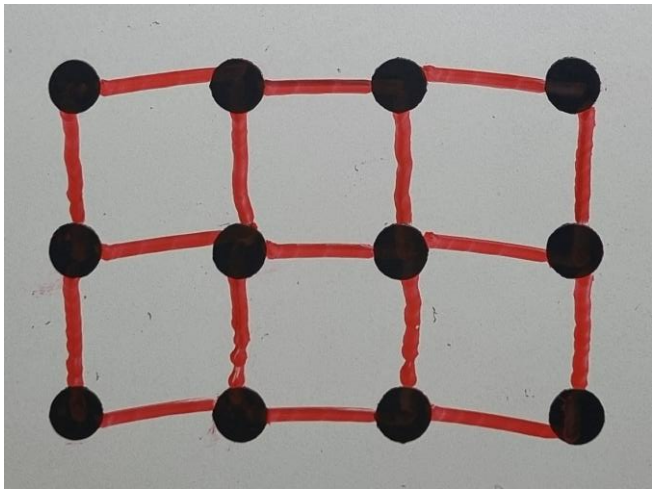

Figura 26: Líneas dibujadas por el robot. Elaboración propia.

4.2. Costo real del prototipo

A continuación, se presenta el desglose de los componentes adquiridos junto con sus respectivos costos, los cuales representan el costo real asociado al desarrollo del proyecto:

Componentes	Unidades	Precio unitario [ARS]	Precio total [ARS]	Precio total [USD]
Pizarra blanca Galaxia 30x40cm	1	9.299,80	9.299,80	8,86
Step-down LM2596	1	4065	4.065,00	3,87
Controlador de servos PCA9685	1	12730	12.730,00	12,12
Llave switch	1	2100	2.100,00	2,00
Servomotor MG996R	4	11200	44.800,00	42,67
Capacitor electrolítico 100µF 16V	1	450	450,00	0,43
Driver Pololu A4988	3	2349,06	7.047,18	6,71
Motor paso a paso Nema 17	1	17910	17.910,00	17,06
Microcontrolador ESP32	2	9310	18.620,00	17,73
Switch final de carrera	1	2499	2.499,00	2,38
Multiplexor I2C TCA9548A	1	9130	9.130,00	8,70
Encoder magnético AS5600	5	9100	45.500,00	43,33
Marcador rojo Pelikan 442	1	1.093,60	1.093,60	1,04
Marcador negro	2	1590	3.180,00	3,03
Borrador para pizarra	1	1720,5	1.720,50	1,64
Cable de 0,5mm	2	2650	5.300,00	5,05
Conector jack hueco 2,1mm	1	100	100,00	0,10
Tornillo M3 cabeza fresada	10	800	8.000,00	7,62
Tornillo M2 x 10mm	10	300	3.000,00	2,86
Rollo PLA	3	18000	54.000,00	51,43
Tuercas M3 y M2	20	100	2.000,00	1,90
Placa de Pertinax 5x5	1	700	700,00	0,67
Estaño	1	2100	2.100,00	2,00
Impresiones 3D	-	-	27.200,00	25,90
Resortes 6cm	4	840	3.360,00	3,20
Barras de silicona 12u	1	2880	2.880,00	2,74
Cable USB a USB-C	1	5500	5.500,00	5,24
Balines de acero de 5mm	70	300	21.000,00	20,00
Plastificación de marcadores	4	1000	4.000,00	3,81
Costo total del prototipo			319.285,08	304,08

Tabla 2: Costos reales del prototipo. Elaboración propia.

Como se observa, el costo real del prototipo fue de USD 304,08, lo cual representa más del doble del presupuesto estimado al inicio del proyecto, que fue de USD 135,45, según se detalla en los archivos relacionados a la propuesta final. Esto implica un incremento del 124,50% respecto a lo planteado inicialmente.

Este aumento se debe principalmente a los cambios y ajustes que se implementaron a lo largo del desarrollo del proyecto. Dichos cambios derivaron en la necesidad de adquirir nuevos componentes y herramientas adicionales que no estaban contemplados en la planificación inicial. Entre los factores que influyeron, se destacan la incorporación de sistemas de mayor precisión, como los encoders magnéticos y el motor paso a paso de la base.

Este análisis de costos pone de manifiesto la importancia de realizar una planificación presupuestaria flexible, considerando un margen para imprevistos, especialmente en proyectos de investigación y desarrollo. Además, recalca la relevancia de documentar cada etapa del proceso, ya que permite identificar con claridad las áreas en las que se pueden optimizar los recursos en futuros desarrollos. A pesar del incremento en el costo, los resultados obtenidos

justifican la inversión realizada, ya que se logró un prototipo funcional que cumple con los objetivos establecidos, sentando las bases para posibles mejoras y aplicaciones en futuros proyectos.

4.3. Tiempo invertido

Al analizar el diagrama de Gantt adjunto, se evidencia una notable discrepancia entre las horas netas estimadas y las horas netas reales. Mientras que las horas estimadas inicialmente ascendieron a 508 horas, las horas reales totalizaron 787 horas, lo que representa un incremento del 54,92%. Este aumento significativo se atribuye a los diversos inconvenientes surgidos a lo largo del proyecto, así como a la implementación de ideas que, a pesar de los esfuerzos realizados, no lograron alcanzar los resultados esperados.
Un ejemplo representativo de este último punto es la incorporación de los encoders al sistema. Estos dispositivos fueron añadidos con el objetivo de implementar un control a lazo cerrado que mejorara la precisión y confiabilidad del sistema. Sin embargo, su ajuste y calibración no pudieron completarse de manera efectiva debido a las limitaciones físicas del sistema, las cuales han sido detalladas en secciones previas del informe.
Estos contratiempos reflejan los desafíos inherentes a la integración de nuevas tecnologías en un diseño ya existente, destacando la importancia de considerar las restricciones físicas y técnicas durante la planificación inicial del proyecto. A pesar de estas dificultades, los aprendizajes obtenidos de estas experiencias son valiosos y contribuyen al desarrollo de habilidades para futuros proyectos de ingeniería.

5. Conclusiones

Se llevó a cabo con éxito el desarrollo de un brazo robótico de cuatro grados de libertad capaz de participar en partidas del juego puntos y cajas contra un contrincante humano. El proyecto abarcó de manera integral todas las etapas necesarias para su realización: diseño mecánico, diseño eléctrico, desarrollo del software, implementación del sistema de control, y pruebas funcionales. Cada una de estas fases representó un desafío técnico que fue resuelto mediante la aplicación de conocimientos adquiridos en diferentes materias de la carrera, consolidando así las habilidades prácticas y teóricas desarrolladas a lo largo del trayecto académico.
El diseño mecánico se enfocó en garantizar que el robot tuviera la precisión y estabilidad necesarias para interactuar con el tablero de juego, mientras que el diseño eléctrico aseguró una comunicación eficiente entre los sensores, actuadores y microcontroladores. En cuanto al software, se desarrolló un sistema de control robusto que integra algoritmos para la detección visual, planificación de trayectorias y ejecución precisa de movimientos, permitiendo al robot identificar las líneas dibujadas, calcular su siguiente movimiento y ejecutarlo de manera eficiente. Adicionalmente, se implementaron estrategias de optimización en el código para mejorar la respuesta del robot y garantizar una experiencia fluida durante el juego.

Una de las contribuciones más significativas del proyecto fue la implementación de un sistema de visión artificial que utiliza marcadores ArUco para referenciar y detectar líneas en el tablero. Este sistema, combinado con un control mecánico preciso, permitió una interacción efectiva entre el robot y el tablero, garantizando que los movimientos realizados fueran exactos y coherentes con las reglas del juego. También se incorporaron librerías de inteligencia artificial y algoritmos de decisión que dotaron al robot de la capacidad de competir contra el jugador humano en diferentes niveles de dificultad, logrando adaptarse a la estrategia de su oponente.

El éxito de este proyecto no solo radica en haber construido un robot funcional, sino en la integración interdisciplinaria de conocimientos de mecánica, electrónica y programación para resolver un problema específico de manera creativa y eficiente. Más allá de la aplicación directa en el juego de puntos y cajas, las soluciones desarrolladas pueden ser adaptadas y extendidas a otros contextos que requieran la combinación de sistemas robóticos y visión artificial.