

UNIVERSIDAD POLITÉCNICA DE VICTORIA

Brazo robótico del robot de búsqueda y rescate UPV-Transpaís 2024

REPORTE DE ESTANCIA II

Que presenta

Alexis Fernando López Salazar

Alumno de la carrera de
Ingeniería Mecatrónica.

Asesor institucional

Dr. Wilian Jesús Pech

Asesor empresarial

Dr. Rubén Machucho Cadena

Organismo receptor
Universidad política de Victoria.

Cd. Victoria, Tamaulipas, Noviembre 2023

RESUMEN

En el siguiente documento se presentará el desarrollo y elaboración del diseño mecánico, eléctrico y de programación del brazo robótico que será integrado en el robot de búsqueda y rescate para el próximo torneo mexicano de robótica en 2024, en base a los lineamientos establecidos por dicha competencia se tomaron como referencia para las tareas a realizar. El diseño se creó por medio del software SolidWorks, el cual nos permite crear, simular y evaluar las piezas mecánicas más óptimas para la creación del brazo robótico, teniendo una vista previa del ensamblaje de los componentes electrónicos y los grados de libertad que los eslabones del brazo pueden alcanzar, así como el peso total y posibles fallas mecánicas que se podrían presentar. Haciendo uso de la IDE Arduino para la programación de un control remoto para la transmisión de las señales a los motores del brazo en base a las configuraciones de los programas, lo cual es una ventaja por su facilidad en programación y accesibilidad sobre otros lenguajes en las funciones determinadas. Adicionalmente se mostrarán algunas de las adversidades que se presentaron en el desarrollo del proyecto tanto físicas como de programación.

ABSTRACT

This document will present the development and elaboration of the mechanical, electrical and programming design of the robotic arm that will be integrated into the search and rescue robot for the next Mexican robotics tournament in 2024, based on the guidelines established by competition were taken as a reference for the tasks to be carried out. The design was created using SolidWorks software, which allows us to create, simulate and evaluate the most optimal mechanical parts for the creation of the robotic arm, having a preview of the assembly of the electronic components and the degrees of freedom that the links of the arm can reach, as well the total weight and possible mechanical failures that could happen. Using the Arduino IDE for remote control programming to transmit signals to the arm motors based on the program configurations, which is an advantage due to its ease of programming and accessibility over other languages in determinate functions. Additionally, some of the adversities that occurred in the development of the project, both physical and programming will be shown.

Palabras clave: SolidWorks, Arduino, driver, joystick shield, control, adquisición, transmisión, eslabón.

ÍNDICE TEMÁTICO

I. INTRODUCCIÓN	4
II. MARCO TEÓRICO	5-9
III. JUSTIFICACIÓN	10
IV. OBJETIVOS	10
V. DESARROLLO DEL PROYECTO	11-28
Dispositivos y materiales	11
Procedimiento y metodología	11
1. Compilación de información	11-12
2. Análisis de base y eslabón 1	13
3. Análisis eslabón 2	13
4. Análisis de la garra	13-14
5. Brazo robótico ensamblado en el carro de búsqueda y rescate	15-18
6. Control remoto del brazo robótico	18-19
1) Sistema transmisor	18
2) Sistema Receptor	19
7. Programación en Arduino para los códigos transmisores, receptores y servidor web para la transmisión de video	20-24
1) Sistema transmisor. Código de configuración de mando para los joysticks shields.....	20
2) Sistema receptor, motores nema 17. Código de configuración para el control de la base y eslabones	20-21
3) Sistema receptor, servomotores. Código de configuración para el control de la garra y giro de la cámara	21-22
4) Sistema de visión. Código para la transmisión de video en vivo a un servidor web por medio de un ESP32 CAM	23-24
8. Diagrama de conexión TB6600—MOTOR NEMA 17—ARDUINO MEGA—NRF24L01	25
9. Diagrama de conexión SERVOMOTORES—ARDUINO—NRF24L01	25
10. Conexiones físicas	26
1) Sistema de control base-eslabones	26
2) Sistema de control garra-cámara	26
11. Etapa de pruebas para la competencia	27-28
1) Camino al vacío	27
2) Toma de objetos remotamente / abertura de puertas	28
VI. RESULTADOS	29
VII. CONCLUSIONES	30
VIII. BIBLIOGRAFÍA	31

I. INTRODUCCIÓN

En el siguiente proyecto se desarrolla un prototipo de brazo robótico el cual cumpla con los criterios para la competencia de búsqueda y rescate de TMR 2024, donde su función principal será servir de apoyo al carro robótico donde será instalado para su movilización. Se pretende usar el driver TB6600 para el control de motores a pasos nema 17, un puente H para un motor DC y servomotores, estos componentes serán controlados por varios módulos de radiofrecuencia NRF24L01 y la placa joystick shield, conectados a un par de arduinos que funcionarán como receptor y transmisor para la transmisión de las señales en un extenso rango de distancia para el control total del brazo robótico. Con el uso del driver TB6600 es posible la rápida configuración de los motores nema por su fiabilidad, debido que permiten un cambio efectivo en las revoluciones, velocidad y manejo de una alta corriente y voltaje que son necesarios para alimentar los motores. Se ha empleado la IDE Arduino para programar las direcciones de giro en cada motor y la transmisión de datos entre cada controlador, así como la transmisión de video por medio de un Esp 32 cam a un servidor web. La finalidad del proyecto es la creación de un brazo robótico lo suficientemente capaz y estable mecánicamente para ser controlado remotamente en un óptimo desempeño físico para la competencia.

Universidad Politécnica de Victoria

Fundamentos de calidad.

Misión.

Contribuir al progreso de México mediante la formación de profesionales altamente calificados, con sentido humano y reconocida capacidad científico – tecnológica para participar en la solución de problemas de alto impacto social.

Visión.

Ser una institución de educación superior reconocida nacional e internacionalmente por el excelente desempeño profesional de sus egresados, la capacidad y compromiso de su capital humano, la calidad de sus programas académicos y su aportación científico – tecnológica al progreso de Tamaulipas y de México.

Política de calidad.

Es una institución de educación superior que forma profesionistas competentes para el desarrollo científico-tecnológico de la región, mediante programas académicos de calidad.

Valores organizacionales

- Respeto
- Lealtad
- Responsabilidad
- Integridad

II. MARCO TEÓRICO

SolidWorks.

SolidWorks es un software de diseño CAD 3D (diseño asistido por computadora) para modelar piezas y ensamblajes en 3D y planos en 2D. El software que ofrece un abanico de soluciones para cubrir los aspectos implicados en el proceso de desarrollo del producto. Sus productos ofrecen la posibilidad de crear, diseñar, simular, fabricar, publicar y gestionar los datos del proceso de diseño. Ofrece soluciones intuitivas para cada fase de diseño. Cuenta con un completo conjunto de herramientas que ayudan a ser más eficaz y productivo en el desarrollo de productos en todos los pasos del proceso de diseño, incluye cinco herramientas clave:

1. Herramientas de diseño para crear modelos y ensamblajes
2. Herramientas de diseño para la fabricación mecánica, que automatiza documentos de inspección y genera documentación sin planos 2D.
3. Herramientas de simulación para evaluar el diseño y garantizar que es el mejor posible
4. Herramientas que evalúan el impacto medioambiental del diseño durante su ciclo de vida.
5. Herramientas que reutilizan los datos de CAD en 3D para simplificar el modo en que las empresas crean, conservan y utilizan contenidos para la comunicación técnica. (1)

Puente H.

Un puente H es un tipo de circuito electrónico que permite a un motor eléctrico de corriente directa cambiar de sentido al girar, le permite ir en ambos sentidos, en el sentido horario y anti horario. Son usados de manera muy frecuente en robótica y también son utilizados como convertidores de potencia. Este tipo de puentes están disponibles en una presentación como circuitos integrados, pero pueden también ser construidos a partir de componentes discretos.

Se le denomina puente H porque la representación gráfica de este circuito es en forma de H donde los interruptores mecánicos van ubicados a los lados y en el centro se localiza el motor. Un puente H se puede elaborar a partir de 4 interruptores mecánicos o mediante transistores. Los interruptores se configuran de manera que al pasar la corriente por dos de ellos se genera un giro en sentido positivo y al invertir el voltaje se cierran los interruptores contrarios logrando así invertir el giro del motor. (2)

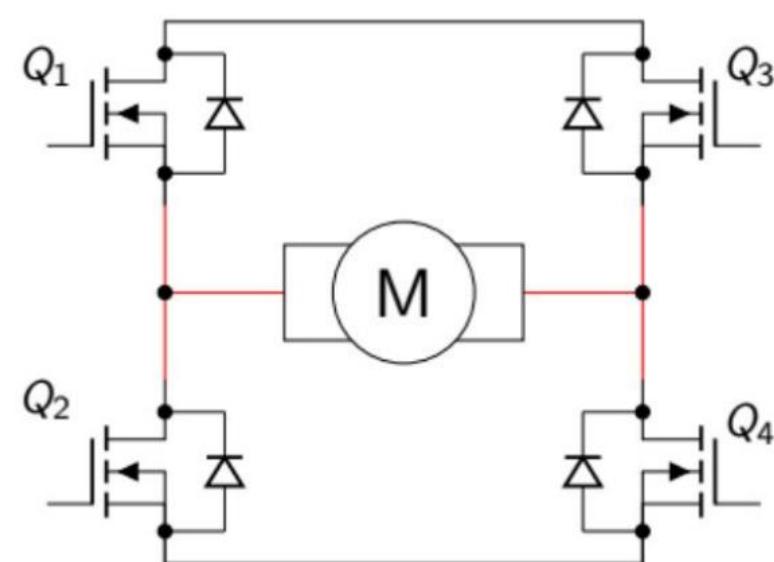


Figura 1. Diagrama eléctrico del puente H llamado así por su característica distribución de elementos.

Configuraciones puente H

Giro sentido horario.

Al encender Q1 y Q4, el motor queda alimentado directamente por lo que gira en sentido de las manecillas del reloj.

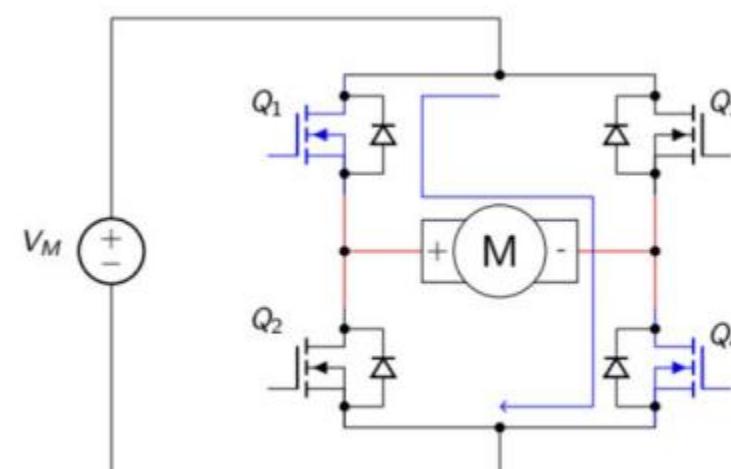


Figura 1.2

Giro sentido antihorario.

Al encender Q2 y Q3, el motor queda alimentado inversamente por lo que gira en sentido contrario de las manecillas del reloj.

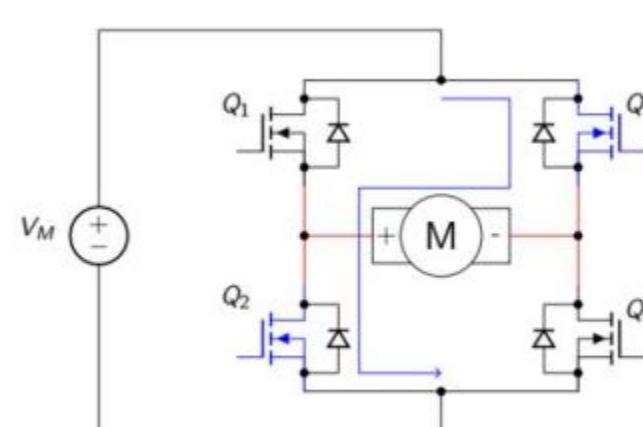


Figura 1.3

TB6600.

Este es un controlador de motor paso a paso bifásico profesional. Soporta velocidad, dirección y control. Puede configurar su micro paso y corriente de salida con 6 interruptores DIP. Hay 7 tipos de micro pasos dependiendo de la configuración seleccionada (1, 2/A, 2/B, 4, 8, 16, 32), 8 tipos de control de corriente (0.5A, 1A, 1.5A, 2A, 2.5A, 2.8A, 3.0A, 3.5A) y alcanza de 9-42 VDC. Todos los terminales de señal adoptan un optoacoplador de alta velocidad. El aislamiento mejora su capacidad anti interferencias de alta frecuencia. El controlador admite cátodo común y circuito de ánodo común, se puede optar el que según se requiera.

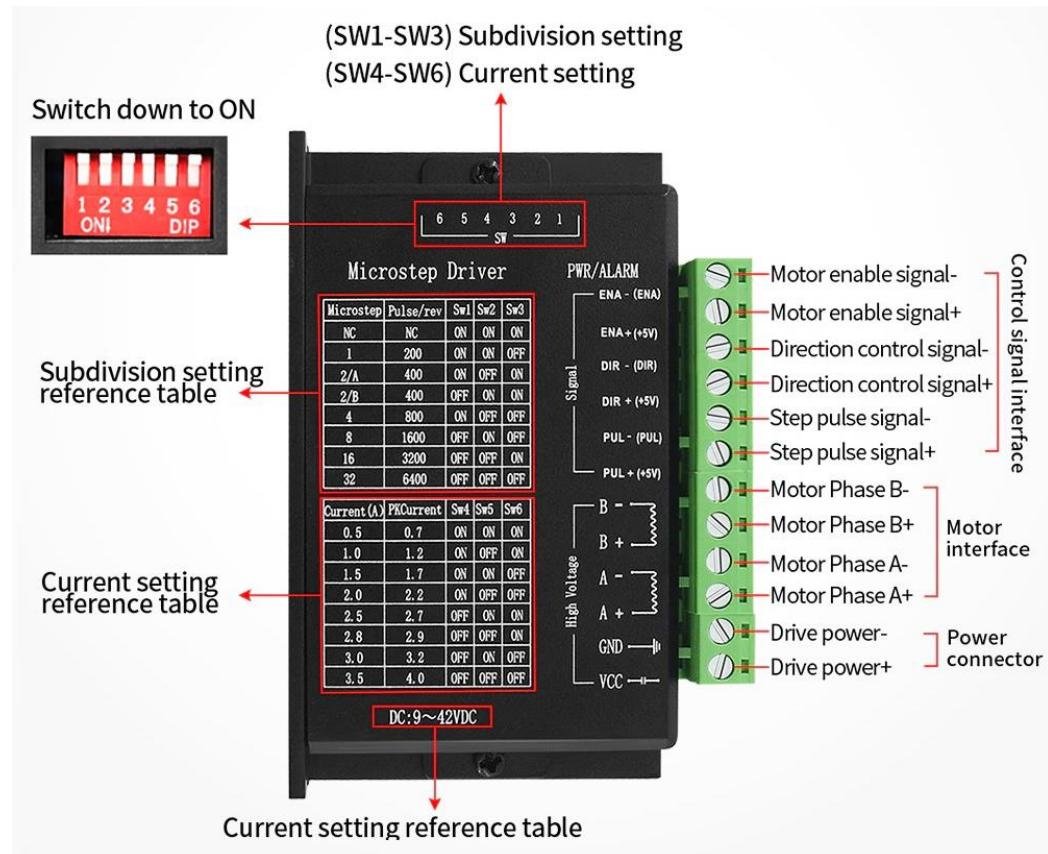


Figura 2. Descripción física del driver TB6600, slots de entrada.

Micro Step	Pulse/Rev	S1	S2	S3
NC	NC	ON	ON	ON
1	200	ON	ON	OFF
2/A	400	ON	OFF	ON
2/B	400	OFF	ON	ON
4	800	ON	OFF	OFF
8	1600	OFF	ON	OFF
16	3200	OFF	OFF	ON
32	6400	OFF	OFF	OFF

Figura 2.2 DIP de configuraciones para los micro pasos.

Current (A)	S4	S5	S6
0.5	ON	ON	ON
1.0	ON	OFF	ON
1.5	ON	ON	OFF
2.0	ON	OFF	OFF
2.5	OFF	ON	ON
2.8	OFF	OFF	ON
3.0	OFF	ON	OFF
3.5	OFF	OFF	OFF

Figura 2.3 DIP de configuraciones para la corriente.

Conexión de ánodo común:

Conecte PUL+, DIR+ y EN+ a la fuente de alimentación del sistema de control. Si la fuente de alimentación es de + 5 V, se puede conectar directamente. Si la fuente de alimentación es superior a + 5V, la resistencia limitadora de corriente R debe agregarse externamente.

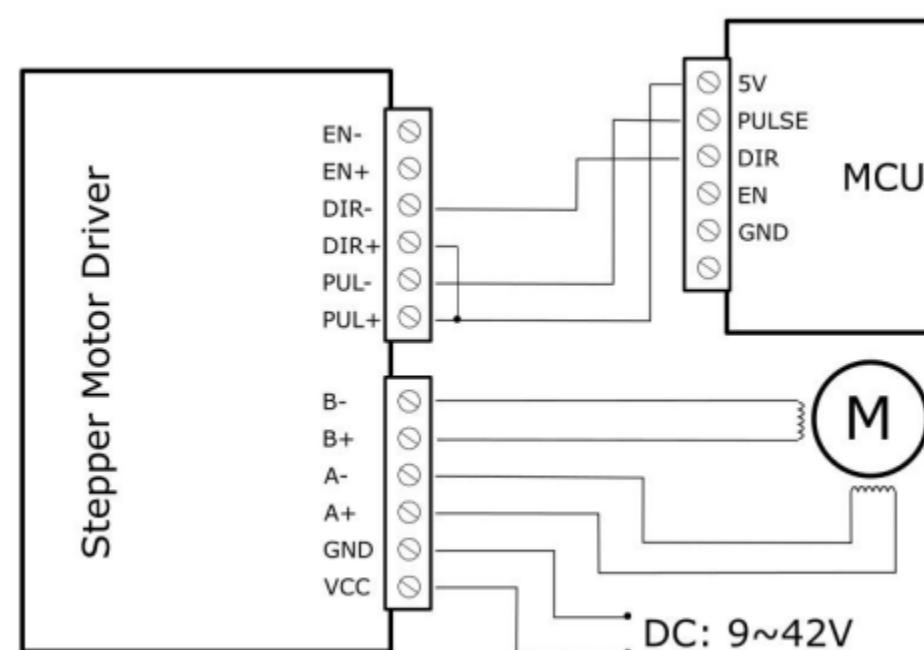


Figura 2.2. Conexión ánodo común a microcontrolador.

Conexión de cátodo común:

Se conecta PUL -, DIR - y EN - al terminal de tierra del sistema de control. La señal de pulso se conecta a PUL-; la señal de dirección se conecta a Dir-; la señal de activación se conecta a ES-. Como se muestra en la siguiente figura:

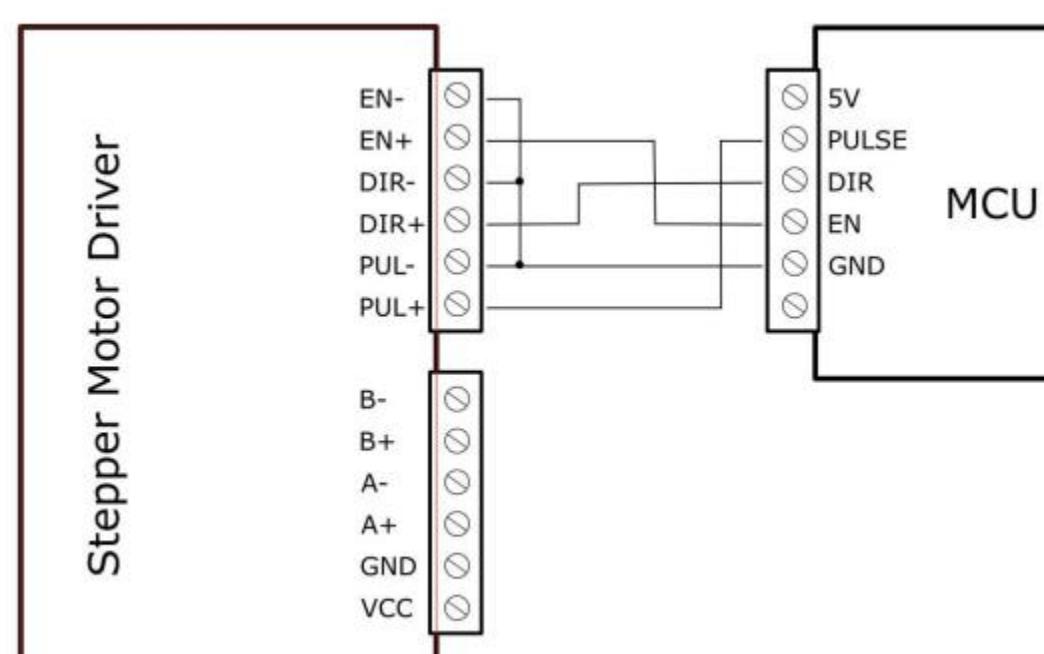


Figura 2.3. Conexión cátodo común.

Cuando "EN" está en el estado válido, el motor está en estado libre (modo fuera de línea). En este modo, puede ajustar la posición del eje del motor manualmente. Cuando "EN" es en el estado no válido, el motor estará en modo de control automático. (3)

Motor NEMA 17-HS4401.

Un motor eléctrico convencional es un dispositivo que mediante una energía eléctrica, genera un movimiento mecánico. El movimiento de un motor se genera en función a los campos magnéticos generados en las bobinas. Los motores son equipos rotatorios en donde se distinguen los dos principales elementos de un motor convencional, un estator y un rotor. Los motores bipolares, como en el caso del NEMA 17, solamente tiene 4 salidas, para controlar un motor bipolar se requiere cambiar la dirección del flujo de la corriente en las bobinas, para esto usamos un arreglo de transistores conocido como puente H. El puente H es un arreglo de transistores actualmente encontrado ya como circuito integrado, o en este caso el driver TB6600, que permite controlar los motores realizando las interrupciones de manera interna solo con activar las entradas lógicas. En general el motor NEMA 17 tiene una gran cantidad de aplicaciones tanto en la industria como en el desarrollo independiente. (4)

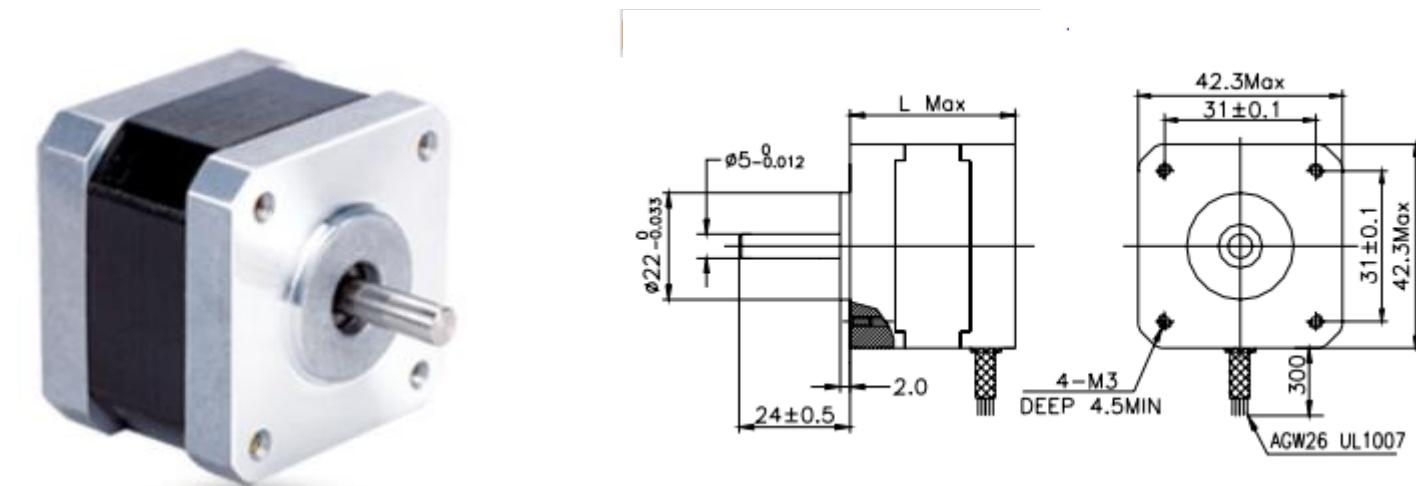


Figura 3. Representación gráfica y en planos 2D (en mm) del motor nema 17HS4401.

En la siguiente tabla se muestran las especificaciones del motor en base a una corriente de 1.7 A.

Series Model	Step Angle (deg)	Motor Length (mm)	Rated Current (A)	Phase Resistance (ohm)	Phase Inductance (mH)	Holding Torque (N.cm Min)	Detent Torque (N.cm Max)	Rotor Inertia (g.cm²)	Lead Wire (No.)	Motor Weight (g)
17HS4401	1.8	40	1.7	1.5	2.8	40	2.2	54	4	280

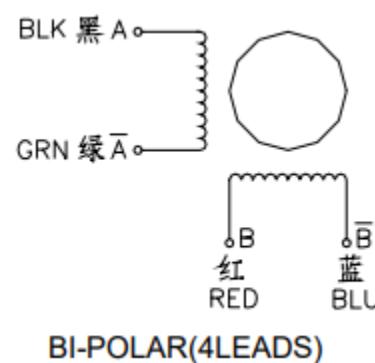


Figura 3.2. Diagrama de conexión de bobinas del motor nema 17 bipolar.

Servomotor (MG995).

Un servomotor es un tipo especial de motor que permite controlar la posición del eje en un momento dado. Está diseñado para moverse determinada cantidad de grados y luego mantenerse fijo en la posición. Existen diferentes tipos de servomotores, éstos se pueden clasificar según sus características de rotación:

- **Servomotores de rango de giro limitado:** son el tipo más común de servomotor. Permiten una rotación de 180 grados, por lo cual son incapaces de completar una vuelta completa.
- **Servomotores de rotación continua:** se caracterizan por ser capaces de girar 360 grados, es decir, una rotación completa. Su funcionamiento es similar al de un motor convencional, pero con la diferencia de que se puede controlar su posición y velocidad de giro en un momento dado.

Las señales de PWM para el circuito de control electrónico son similares para la mayoría de los modelos de servo. Esta señal tiene la forma de una onda cuadrada. Dependiendo del ancho del pulso, el motor adoptará una posición fija. El ancho de pulso sirve para lograr diferentes posiciones en un servomotor (180°, 135°, 90°, 45° hasta 0°). (5)

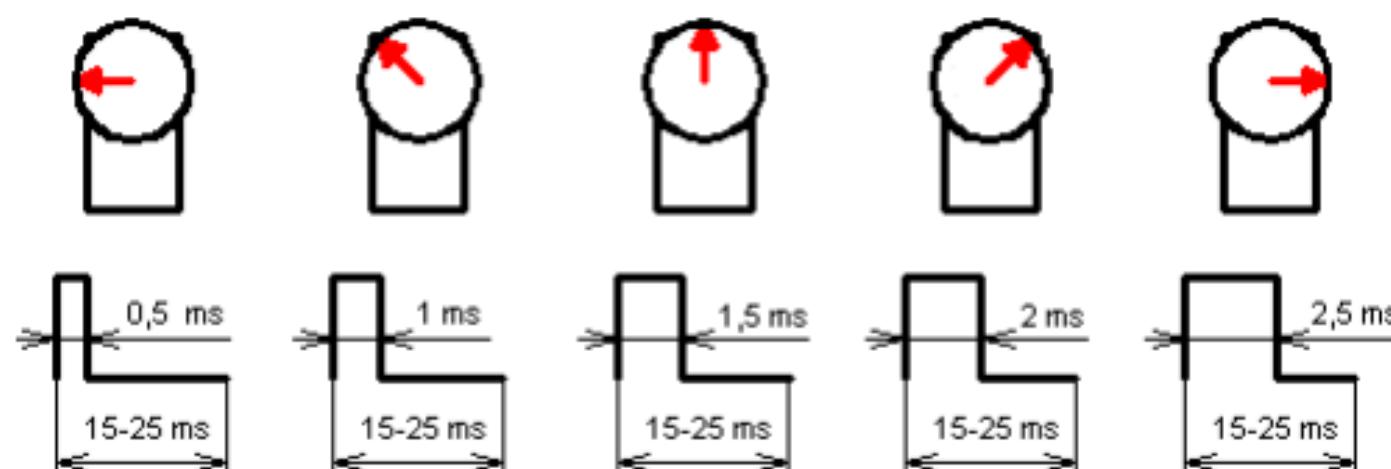


Figura 4. Las señales que se ven en la imagen son las que permiten que el eje del motor adquiera una determinada posición. Estas señales deben repetirse en el tiempo para que el motor mantenga una posición fija.



Voltaje positivo	Tierra (ground)	Señal de control
Rojo	Negro	Amarillo

Figura 4.1 Representación del servo y código de colores del cableado.

Arduino joystick shield.

El Arduino Joystick Shield proporciona siete pulsadores momentáneos (seis pulsadores separados y un pulsador debajo del joystick) y un joystick con dos potenciómetros (eje X, eje Y). Esta placa le da al arduino la funcionalidad para convertirlo en un mando controlador rápidamente al ser ensamblado directamente sobre él, como se muestra en la figura 4.2. El joystick se puede utilizar para controlar salidas de igual forma. Los botones para la navegación o el control. Es posible agregar un módulo de radiofrecuencia NRF24L01 y una interfaz LCD Nokia 5110. Este módulo de RF tiene un rendimiento estable, es altamente rentable y tiene un sólido soporte de biblioteca al programar. También es posible colocar un módulo de módulo bluetooth para una comunicación serial inalámbrica. El interruptor deslizante integrado puede cambiar entre 3,3 V y 5 V. (6)

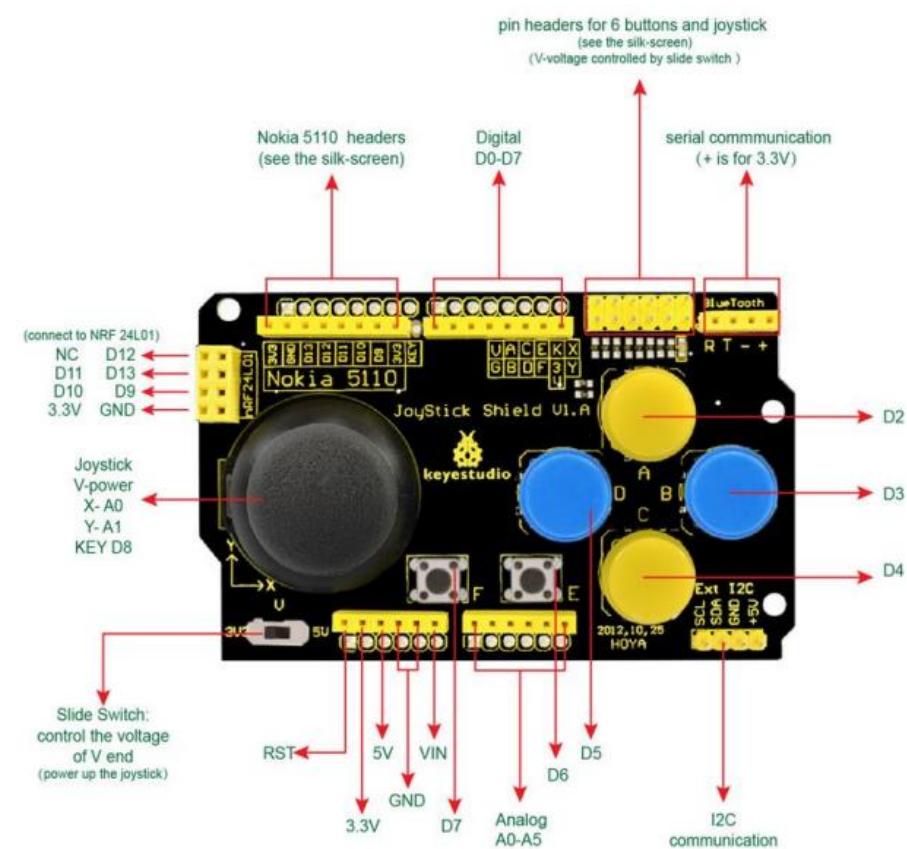


Figura 5. Descripción gráfica de la placa, componentes y distribución de pines.



Figura 5.2. Ejemplo de colocación de la placa joystick shield sobre el arduino.

NRF24L01 +PA+LNA.

El módulo transmisor-receptor de 2,4GHz NRF24L01 fabricado por Nordic Semiconductor es ideal para comunicar proyectos de forma inalámbrica, opera en la banda de 2.4GHz (Industrial, Científica y Médica) y posee un consumo ultra bajo (ULP). Esta versión de módulo posee además un circuito amplificador de potencia (PA), un circuito amplificador de bajo ruido (LNA) y una antena SMA que le permite lograr un rango de hasta 1000m en campo abierto. Es comúnmente utilizado en aplicaciones de control remoto, supervisión/monitoreo remoto, sistemas de seguridad, red inalámbrica y automatización industrial. (7)

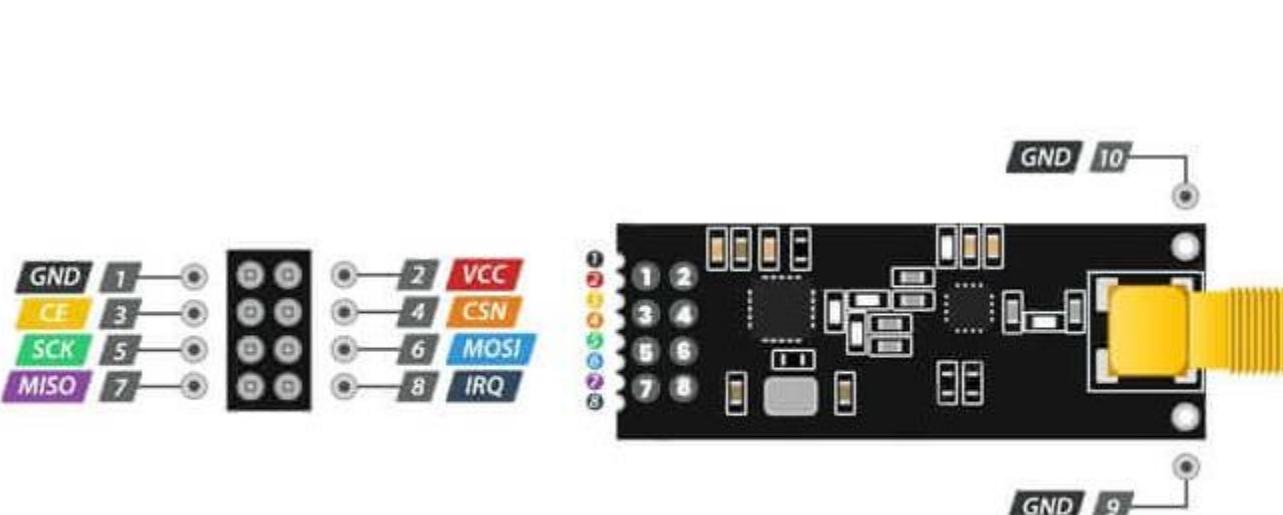


Figura 6. Descripción gráfica y distribución de pines del módulo NRF24L01 +PA+LNA.

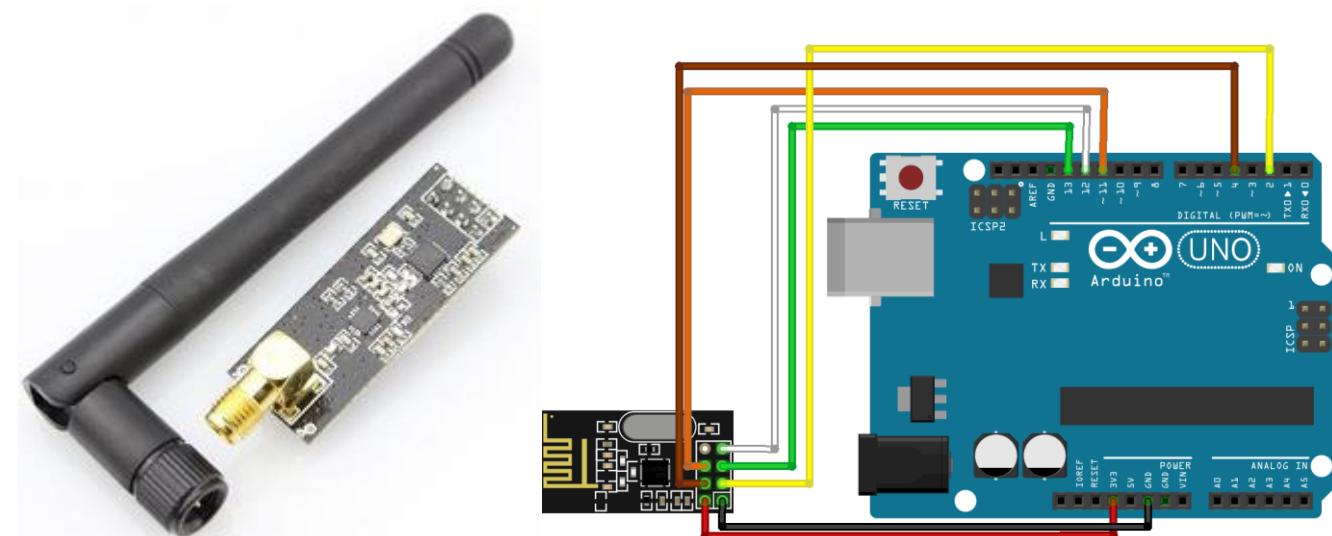


Figura 6.2 Diagrama de conexión NRF-Arduino

La siguiente tabla muestra la descripción de cada pin presente.

Name	Pin function	Description
CE	Digital Input	Chip Enable Activates RX or TX mode
CSN	Digital Input	SPI Chip Select
SCK	Digital Input	SPI Clock
MOSI	Digital Input	SPI Slave Data Input
MISO	Digital Output	SPI Slave Data Output, with tri-state option
IRQ	Digital Output	Maskable interrupt pin. Active low
VDD	Power	Power Supply (+1.9V - +3.6V DC)
VSS	Power	Ground (0V)

Arduino.

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador reprogramable y una serie de pines hembra. Estos permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables dupont). (8)

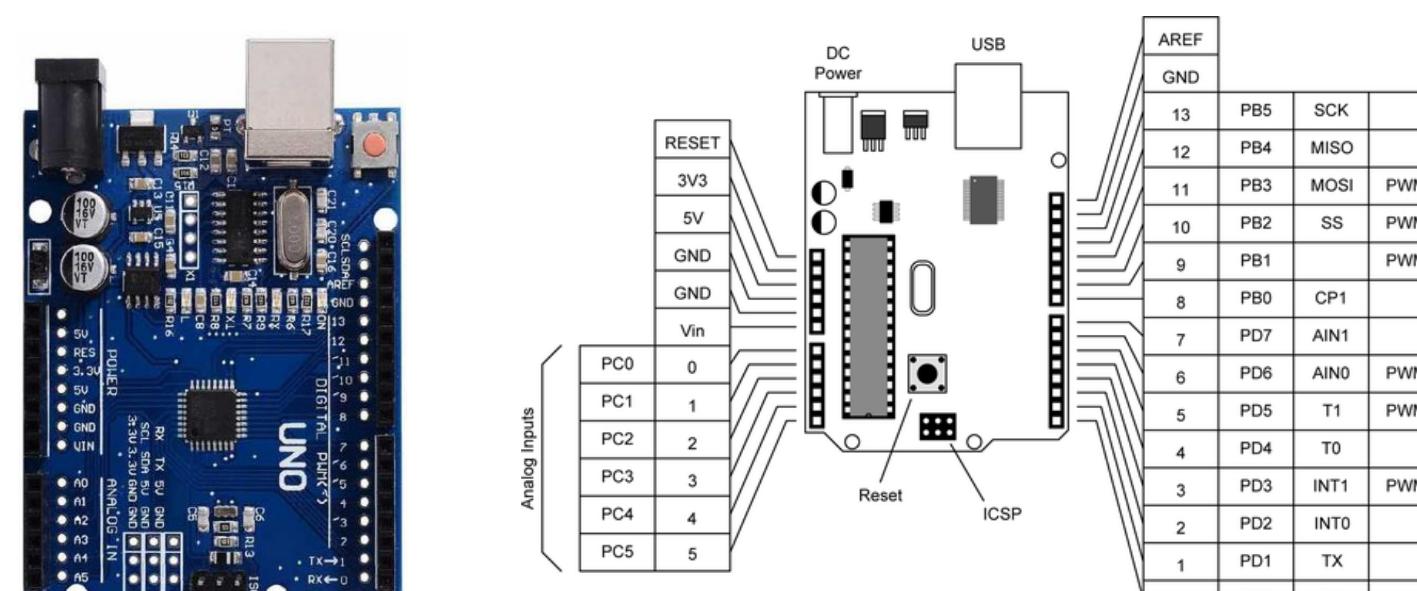


Figura 7. Representación gráfica y distribución de pines en la placa Arduino Uno.

ESP32 CAM.

El ESP32-CAM tiene un módulo de cámara de tamaño pequeño muy eficaz que puede funcionar de forma independiente como un sistema mínimo, con un tamaño de solo 27*40,5*4,5 mm y una corriente de hasta 6 mA. ESP-32CAM se puede utilizar ampliamente en diversas aplicaciones de IoT. Es adecuado para dispositivos domésticos inteligentes, control inalámbrico industrial, monitoreo inalámbrico, identificación inalámbrica QR, señales de sistemas de posicionamiento inalámbrico y otras aplicaciones de IoT. Su formato DIP permite su fácil y rápida integración en cualquier aplicación y montaje en protoboard. Es importante mencionar que a mayor resolución tiene menor cantidad de cuadros por segundo transmitidos (FPS). Al trabajar dentro del entorno Arduino se puede emplear un lenguaje de programación conocido y hacer uso de un IDE sencillo con gran variedad de librerías disponibles en internet que permiten configurar un servidor web de transmisión de video, crear un cámara de vigilancia, tomar fotografías, reconocimiento y detección de rostros, y mucho más. (9)

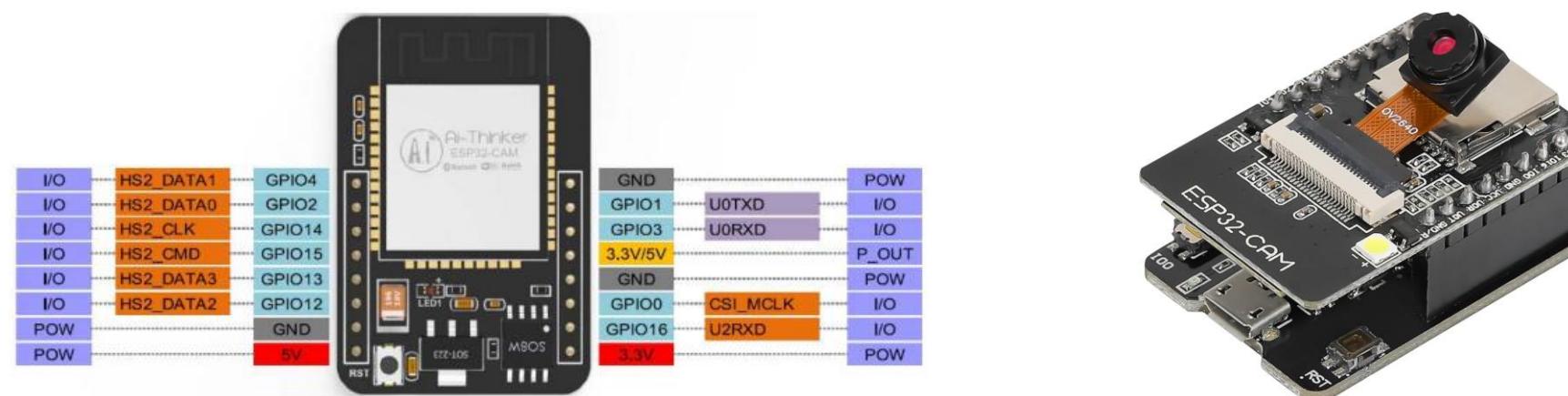


Figura 8. Representación gráfica y distribución de pines en el ESP32-CAM

Impresión 3D.

La impresión 3D es el proceso de creación de objetos mediante el depósito de capas de material unas sobre otras. La impresión 3D se denomina fabricación aditiva (AM) en lugar de los métodos sustractivos tradicionales, como el fresado CNC, cuando se utiliza para la producción industrial.

Un modelo digital en 3D se corta en cientos de capas finas mediante un software específico para exportarlo en formato de código G. Este formato de impresión 3D es un lenguaje que la impresora 3D lee para saber con precisión cuándo y dónde depositar el material. Cada capa corresponde a la forma 2D exacta de una sección o rebanada del objeto. Por ejemplo, si se imprimiera en 3D una pirámide, la primera capa (la inferior) sería un cuadrado plano, y la última capa (en la parte superior) sería un pequeño punto. Las capas se imprimen consecutivamente en 3D de una en una hasta obtener el objeto completamente impreso (figura 9).(10)

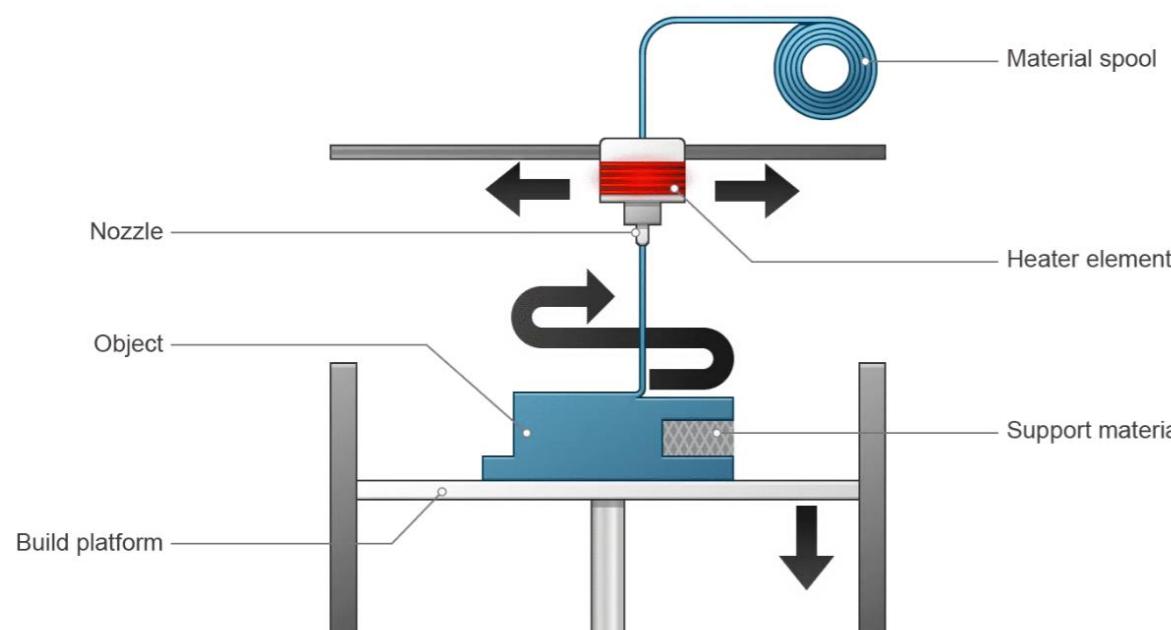


Figura 9. Ejemplo de los elementos que conforman el proceso de impresión 3D.

Filamento PETG.

El Polietileno Tereftalato (PET) es el plástico más utilizado en el mundo. El polímero se utiliza en muchas aplicaciones diferentes como la creación de botellas de agua, hasta en las fibras de tu ropa. También se utiliza ampliamente en los procesos de termoformado y puede combinarse con la fibra de vidrio para crear resinas que se utilizan en el mundo de la ingeniería. La mayoría de los alimentos y bebidas también se entregan y se envasan utilizando PET. En lo que respecta a la impresión en 3D, el material PETG es la versión particular de este plástico que se utiliza en los procesos de impresión en 3D, mientras que la G significa «modificado con glicol». Esto se añade durante la polimerización a la composición del material. La impresión 3D con filamento de PETG ofrece un resultado que da a los materiales un aspecto claro, que es menos quebradizo y más sencillo de usar que la forma base de PET. (11)

Características	
Durabilidad	Alto
Costos de material	Bajo
Fuerza	Alto
Flexibilidad	Bajo
Resistencia	
Resistencia al calor	Medio
Resistencia química	Alto
Resistencia a la fatiga	Alto
Resistencia al agua	Alto
Temperaturas	
La temperatura de transición vitrea	85 °C
La temperatura de la boquilla	210 - 250 °C
Cama calefactada	80 - 100 °C

Figura 10. Tabla de especificaciones del material.

III. JUSTIFICACIÓN

Este proyecto tiene como objetivo el diseño, programación y construcción de un prototipo de brazo robótico por medio del software SolidWorks y Arduino, este servirá de apoyo para el robot que se presentará en el próximo TMR 2024 por parte de la UPV, dicho brazo robótico cumplirá con los lineamientos establecidos por la competencia, al ser conducido por un mando de control remoto creado específicamente en base a las necesidades requeridas y monitoreado por medio de un servidor web.

El proyecto es propuesto por el Dr. Rubén Machucho Cadena, maestro de la Universidad Politécnica de Victoria quien ofreció el puesto como encargado del proyecto, junto con 5 compañeros los cuales colaboraron con el diseño y construcción del robot (carro de 6 ruedas) con la finalidad de seguir preparando mecánica, automática y estratégicamente el proyecto final que representara la UPV en la categoría búsqueda y rescate.

IV. OBJETIVOS

En el prototipo se pretende el diseño, construcción y simulación de las piezas creadas en SolidWorks para obtener los componentes más óptimos que cumplan con los estándares necesarios como de tamaño, resistencia, peso, etc. Y con el programa de Arduino realizar los bosquejos de los códigos con los que serán controlados los componentes electrónicos del brazo robótico y la transmisión de video a un servidor web. Con el fin de superar pruebas como la abertura de una puerta, hacer de puente entre un camino que conduce al vacío, servir de apoyo como gato hidráulico, recolectar objetos manualmente, entre otras tareas.

V. DESARROLLO DEL PROYECTO

Dispositivos y materiales.

- 4 drivers TB6600
- 1 puente H
- 4 NRF24L01 +PA+LNA
- 2 joystick shield
- 4 motores nema HS4401
- 3 servomotores MG995
- 1 motor CD
- Jumpers
- protoboard
- ESP 32 CAM
- 11 tornillos de 1 cm diámetro
- 11 tuercas de 1 cm diámetro
- 17 tornillos de .5 cm diámetro
- 17 tuercas de .5 cm diámetro
- 2 kg de filamento PETG para impresión 3D
- 1 power bank
- Laptop
- Impresora 3D
- 1.2 metros de perfil de aluminio 2x1 pulgadas
- Fuente de voltaje 12V—2A

Procedimiento y metodología.

1) Compilación de información.

Partiendo desde las principales tareas que el brazo robótico deberá llevar a cabo se siguió el ejemplo de varios robots japoneses ganadores de varios

torneos en la categoría robot rescue major como se muestra en los siguientes videos:

- <https://m.youtube.com/watch?v=IIPH8K0KFhE>
- https://m.youtube.com/watch?v=sI_n8XsNDao
- <https://m.youtube.com/watch?v=xpR1KR7L6iI>

Se realizo el estudio y recopilación de datos para comenzar con el diseño, debido a que debe ser simétricamente coherente con el carro robot donde será instalado para cumplir con ciertas medidas y criterios en sus eslabones, como su capacidad de movimiento sin ocasionar coaliciones entre componentes, las distancias máximas alcanzadas, posibles errores, etc. A continuación, se mostrará parte por parte los componentes mecánicos construidos y los eléctricos escogidos que integran este sistema mecatrónico.

2) Análisis de base y eslabón 1.

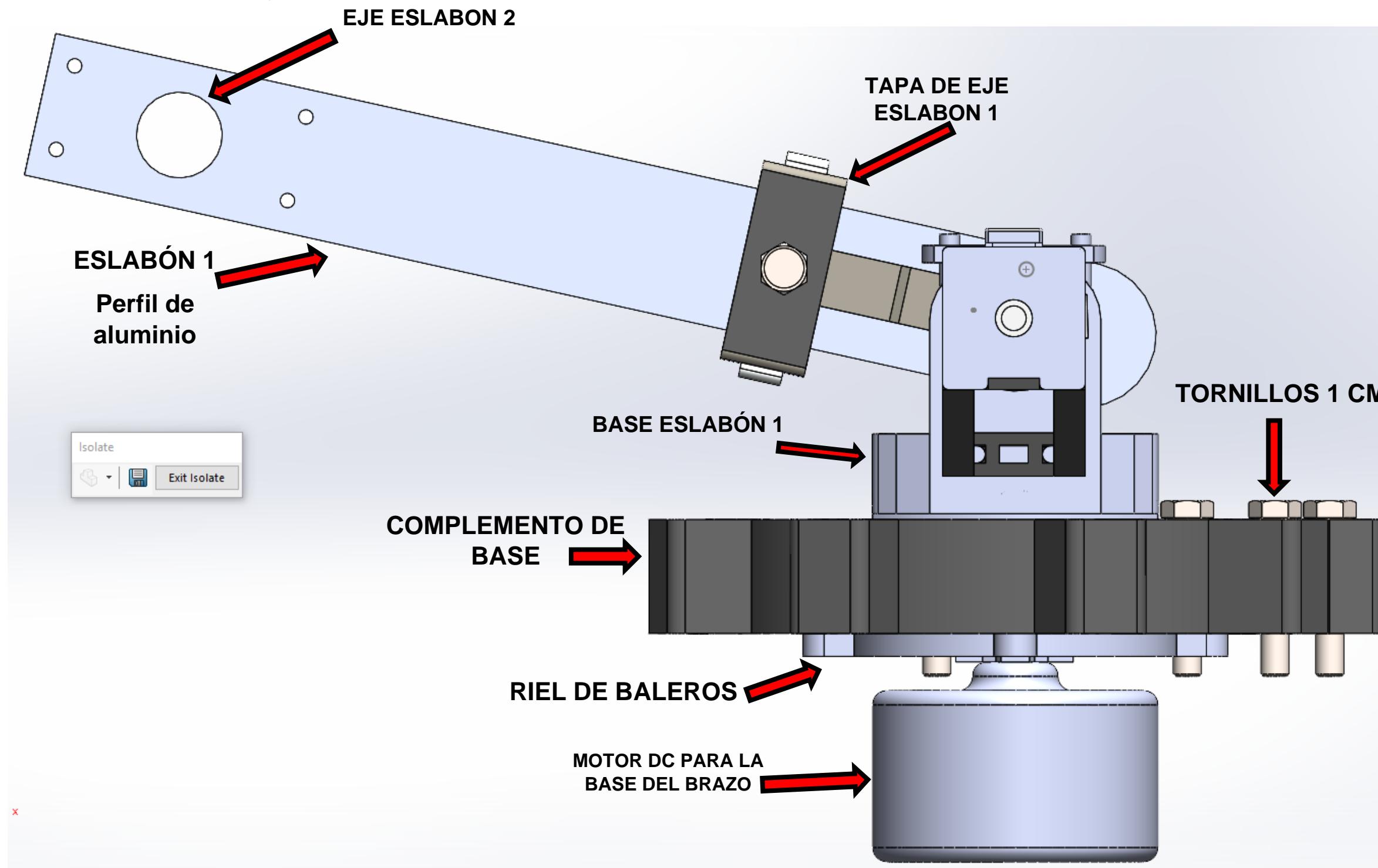
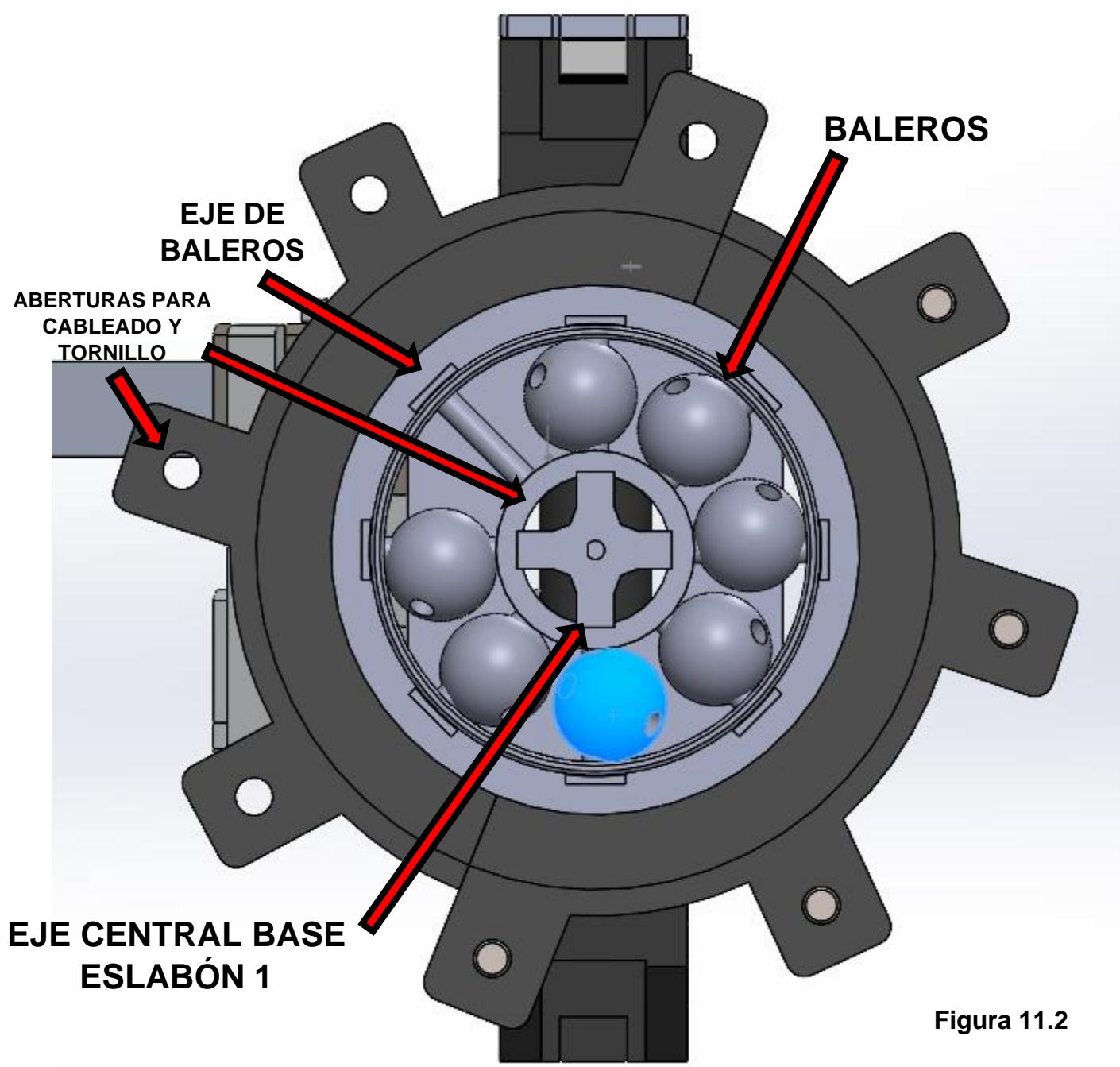
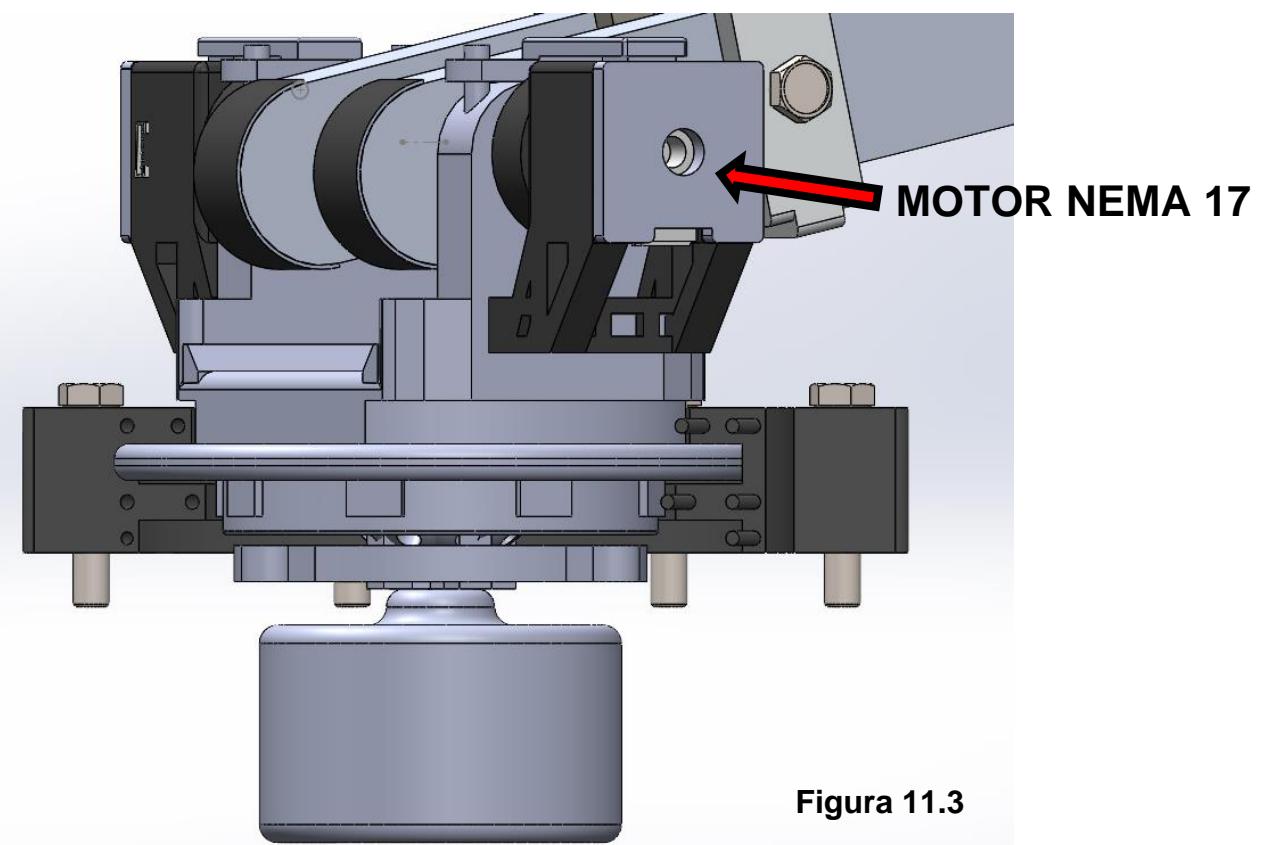


Figura 11. Se muestra la vista lateral izquierda del ensamble del brazo robótico conformado por los elementos descritos.

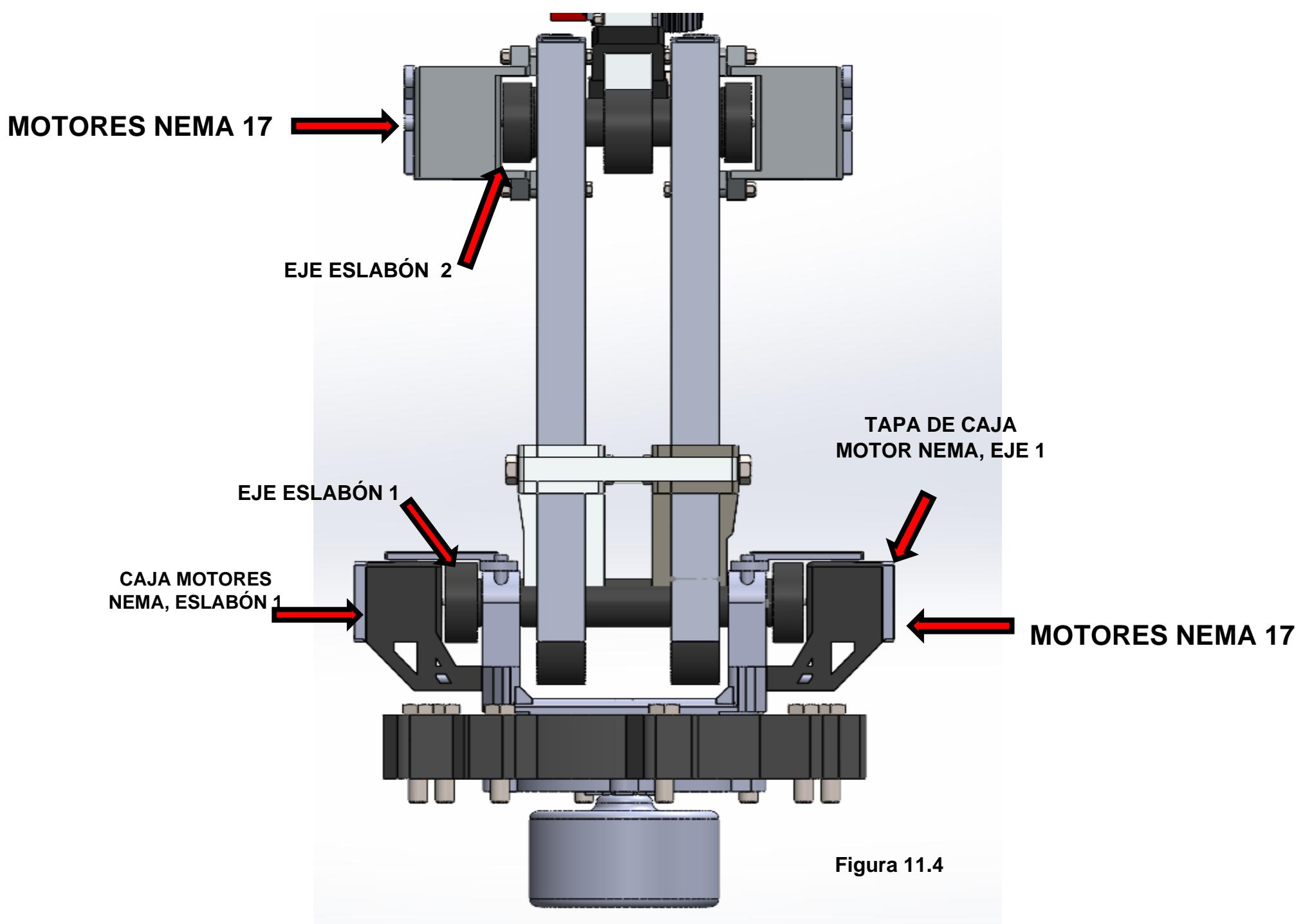


La parte media baja de la base está diseñada para ser insertada en medio de los complementos y permitir el giro 360° evitando el descarrilamiento o desnivel. (Figura 1.3)

En la parte final de la base se encuentra un sistema de rodamientos que permiten un giro más eficiente en la dirección destinada. (Figura 1.2)



En la figura 1.4 se muestra la vista trasera del brazo robótico donde se pueden observar el conjunto de los 4 motores nema 17 en su respectiva caja de posición correspondientes al eslabón 1 y 2, el primer par de motores se encarga de mover el eslabón 1, su caja está diseñada para encajar simétricamente con unas extremidades de la base y quedar enganchadas. Los motores pueden ser configurados respecto a la corriente y numero de pasos por revolución total por medio del driver TB6600, este tipo de control permite una mayor precisión en el manejo remoto del brazo robótico.



3) Análisis eslabón 2.

A continuación, se muestra el conjunto de piezas que conforman el eslabón 2, comenzando por el segundo par de motores nema posicionados en la cima de los perfiles de aluminio y sujetados por tornillo y tuerca que conforman el eslabón 1. El perfil del eslabón 2 está unido de la misma forma que sus contrapartes, sujeto de una tapa que va ensamblada al eje 2. Ambos ejes, perfiles y tapas tienen el propósito de hacer un ensamble rápido, una sujeción resistente y fluidez en el movimiento.

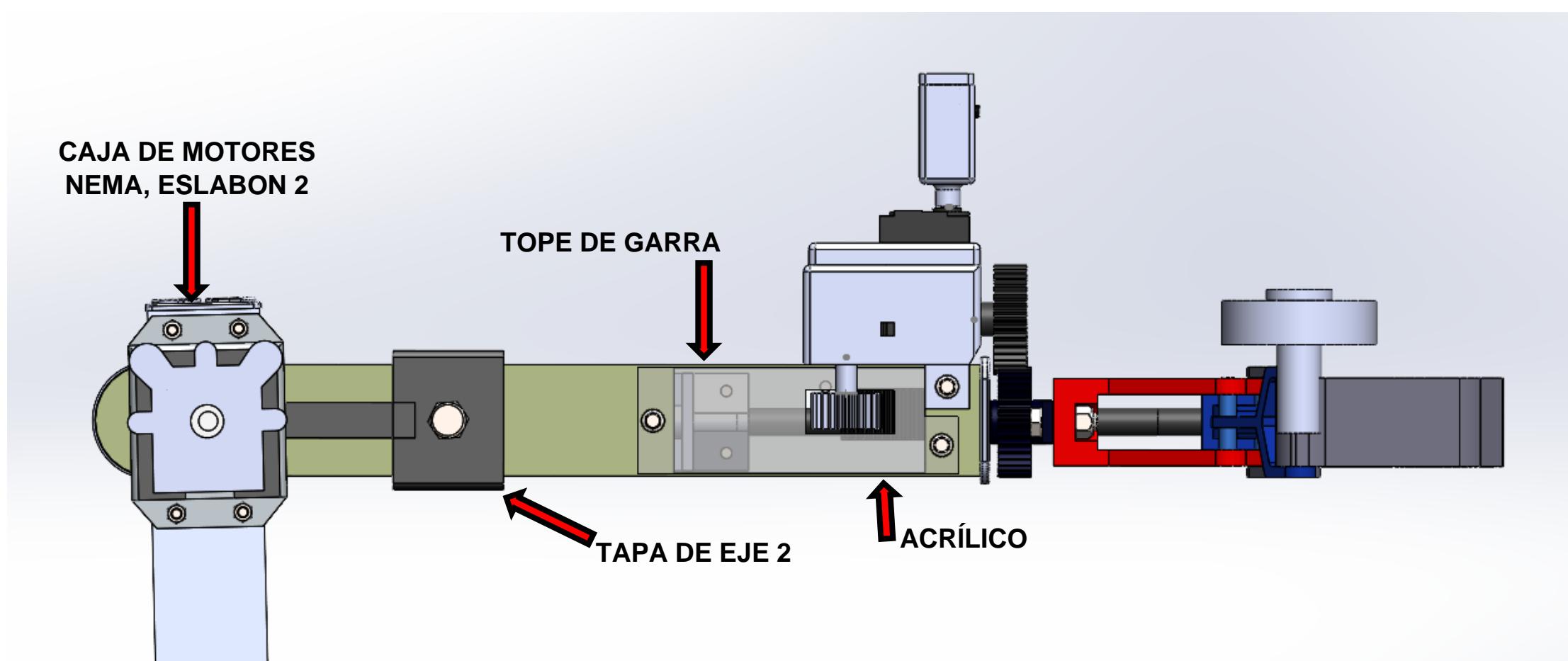


Figura 12. Vista lateral derecha del eslabón 2 y el respectivo ensamble de la garra.

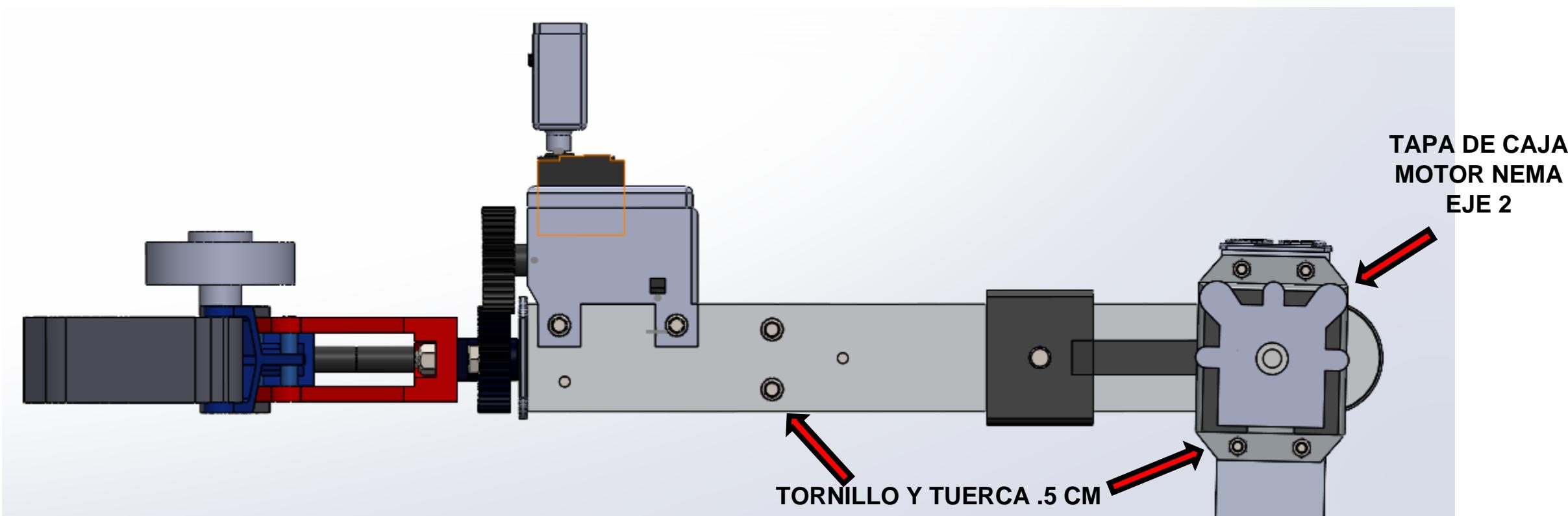


Figura 12.2. Vista lateral izquierda del eslabón 2.

4) Análisis de la garra.

Con la misión de construir una garra robótica capaz de sujetar objetos de hasta 12 cm de ancho máximo y 1 cm mínimo se trazaron las piezas correspondientes. La contracción para abrir, cerrar y hacer girar la garra 360° esta conducida por un par de servomotores a control remoto fijados en el interior de una caja especial, donde el primero tiene la capacidad de rotar de 0° a 90° y de 0 a -90° dependiendo si se tomará o soltará un objeto, el segundo cumple con la función de girar 360° hacia la dirección seleccionada para dar una mayor movilidad y precisión en la toma de objetos. La garra es capaz de girar libremente debido a que el eje de contracción no esta unido por completo al enlace de contracción y el engrane de rotación si está unido a la base de la garra. El eje de contracción funciona por medio de un sistema piñón-cremallera, por debajo tiene una ranura especial que ayuda a mantener la posición al moverse. El tercer servomotor cumple el propósito de girar la ESP 32 CAM (situada en el interior de la carcasa rotativa) hasta 360° en cualquier dirección para la visión y transmisión de imagen a un servidor web creado previamente en Arduino. La cámara será alimentada por una batería de 9v situada en el interior de la caja junto a los servos.

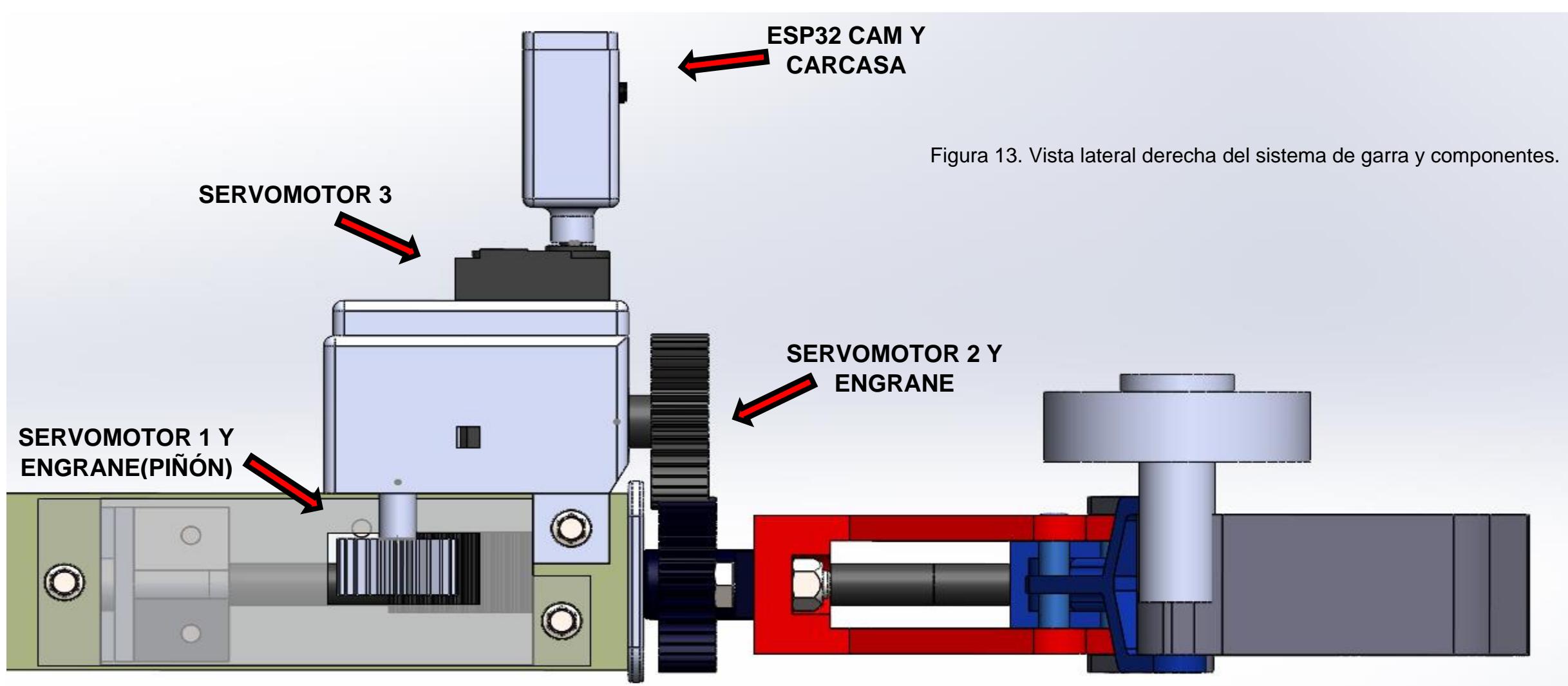
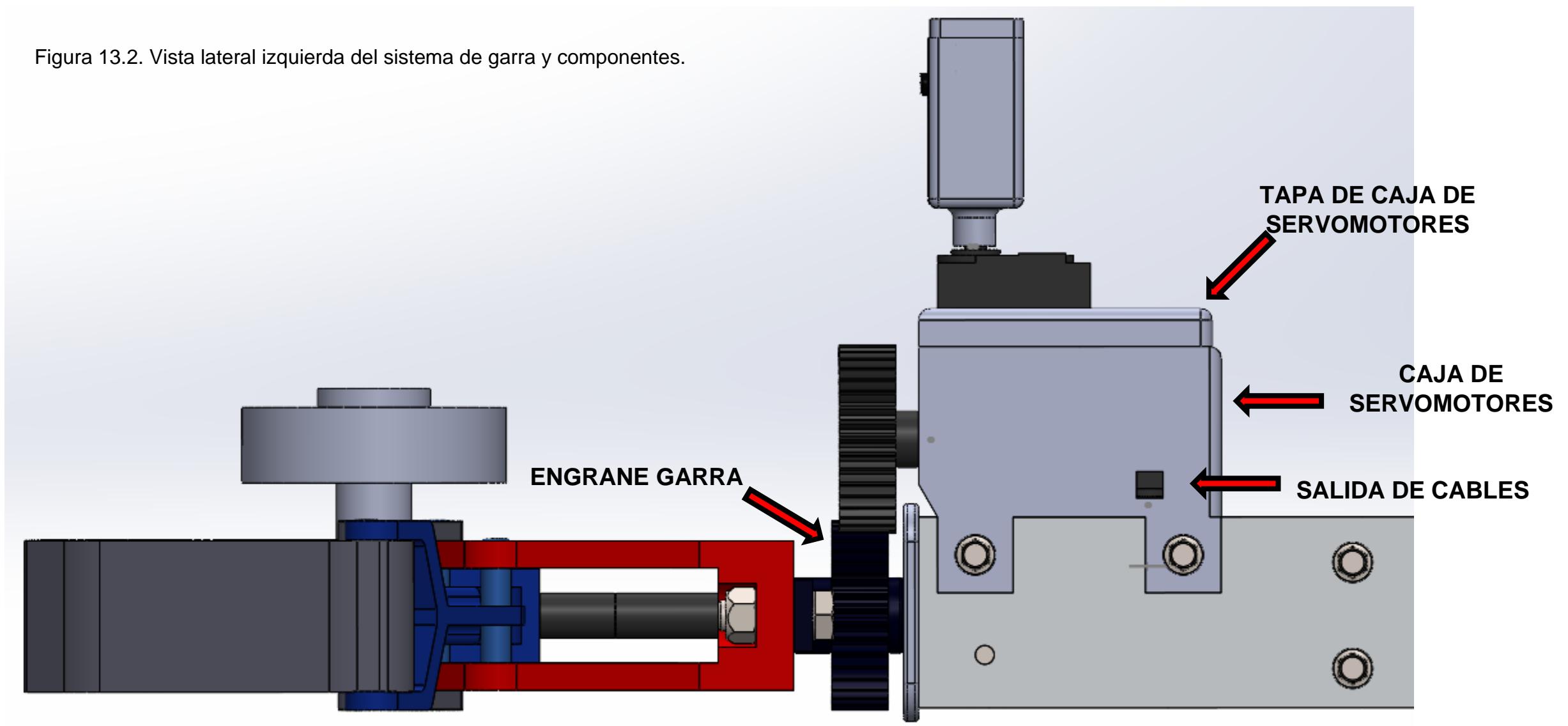


Figura 13. Vista lateral derecha del sistema de garra y componentes.

Figura 13.2. Vista lateral izquierda del sistema de garra y componentes.



LENTE DE CÁMARA

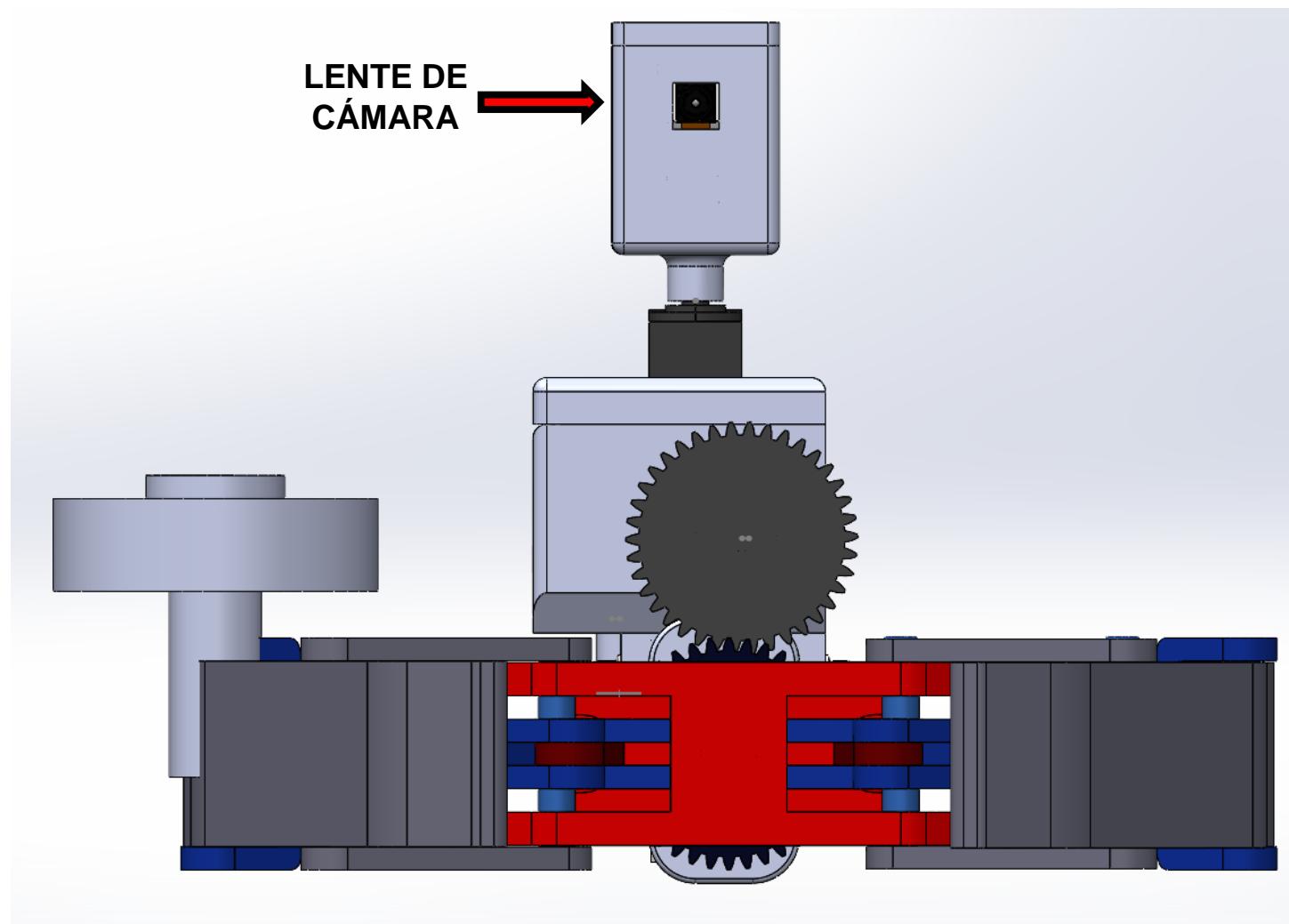


Figura 13.3. Vista frontal del sistema garra y componentes.

TAPA DE CAJA DE SERVOMOTORES
CAJA DE SERVOMOTORES
SALIDA DE CABLES

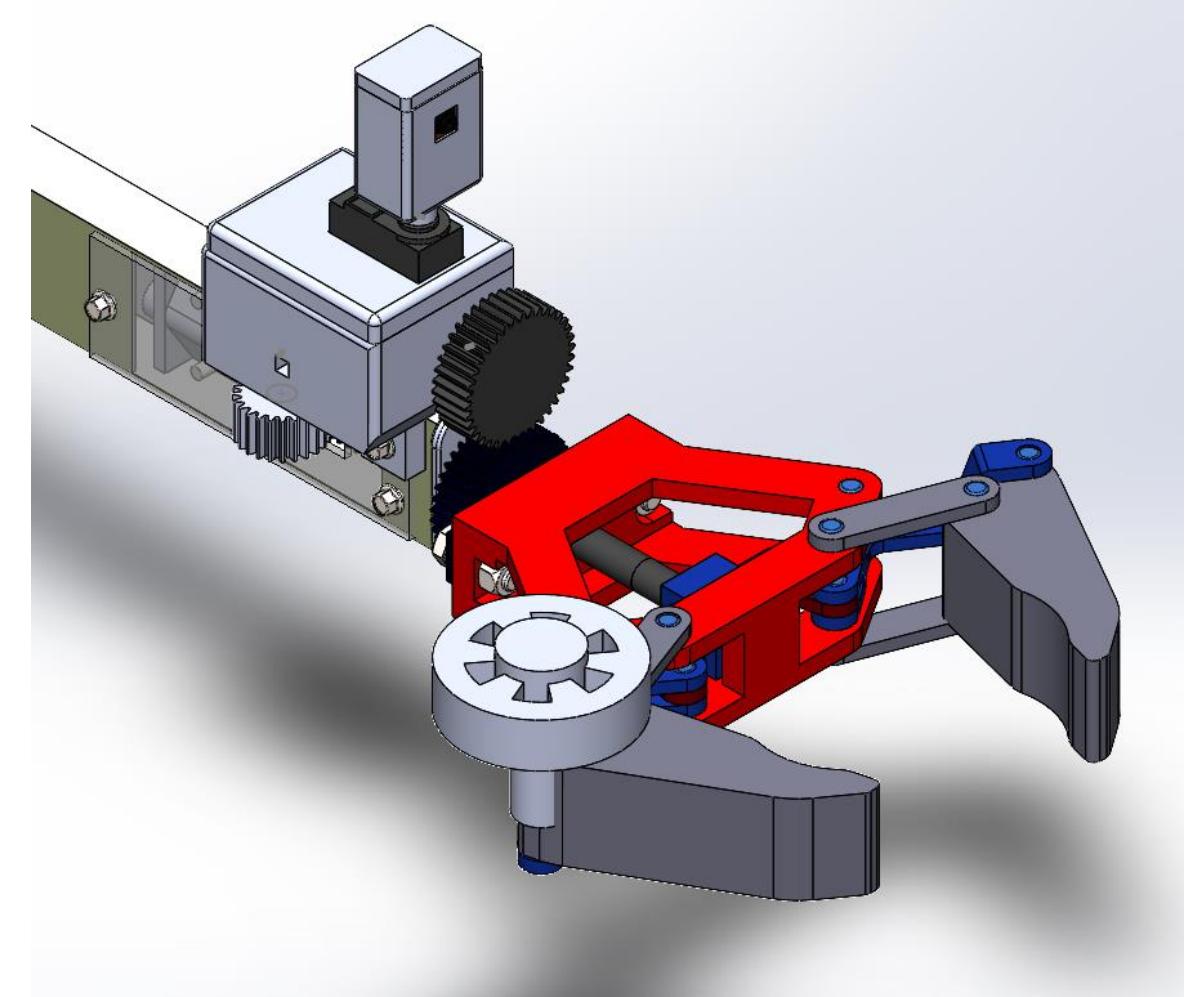


Figura 13.4. Vista isométrica del sistema garra y componentes, estado abierto.

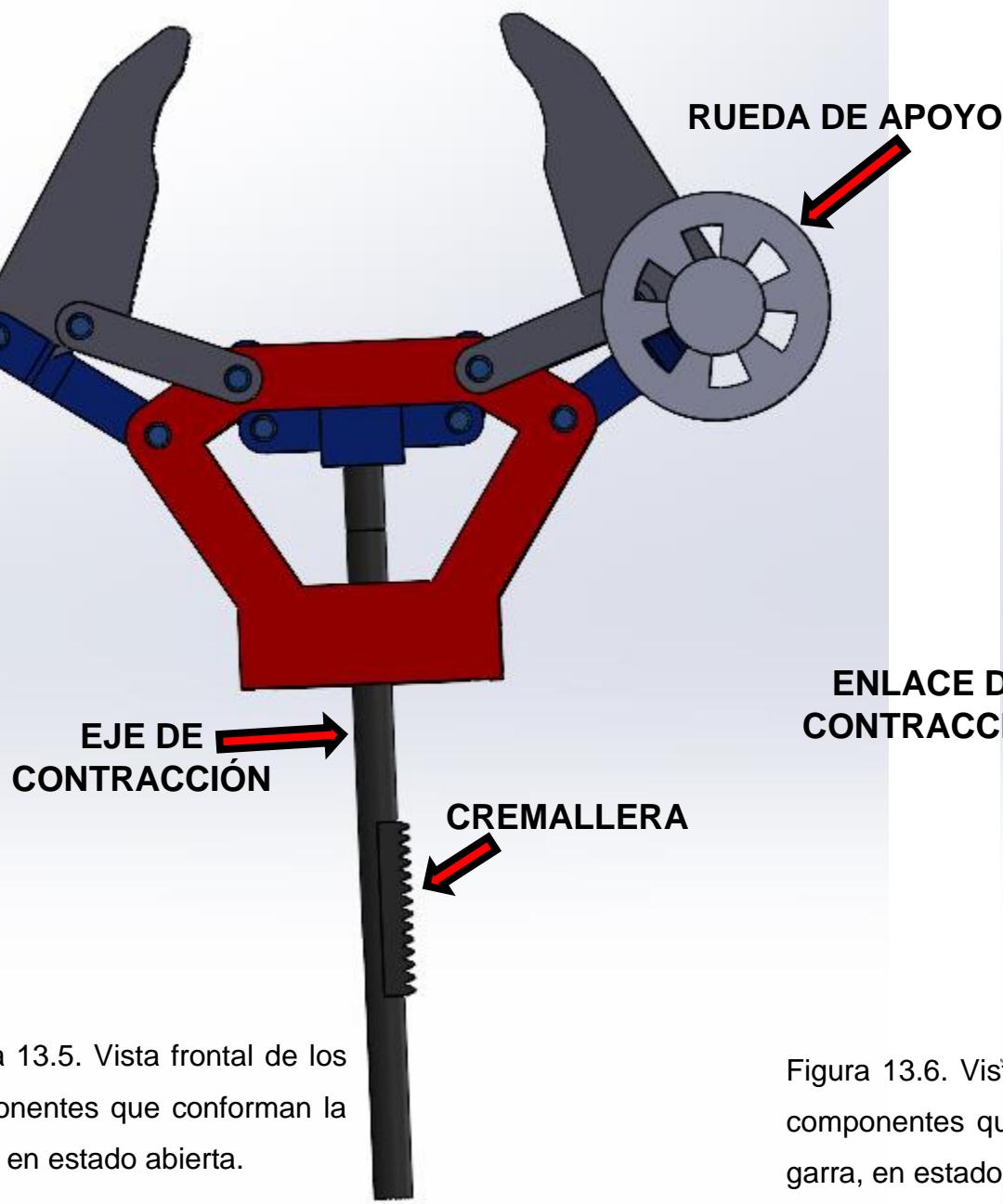


Figura 13.5. Vista frontal de los componentes que conforman la garra, en estado abierto.

Figura 13.6. Vista frontal de los componentes que conforman la garra, en estado cerrada.

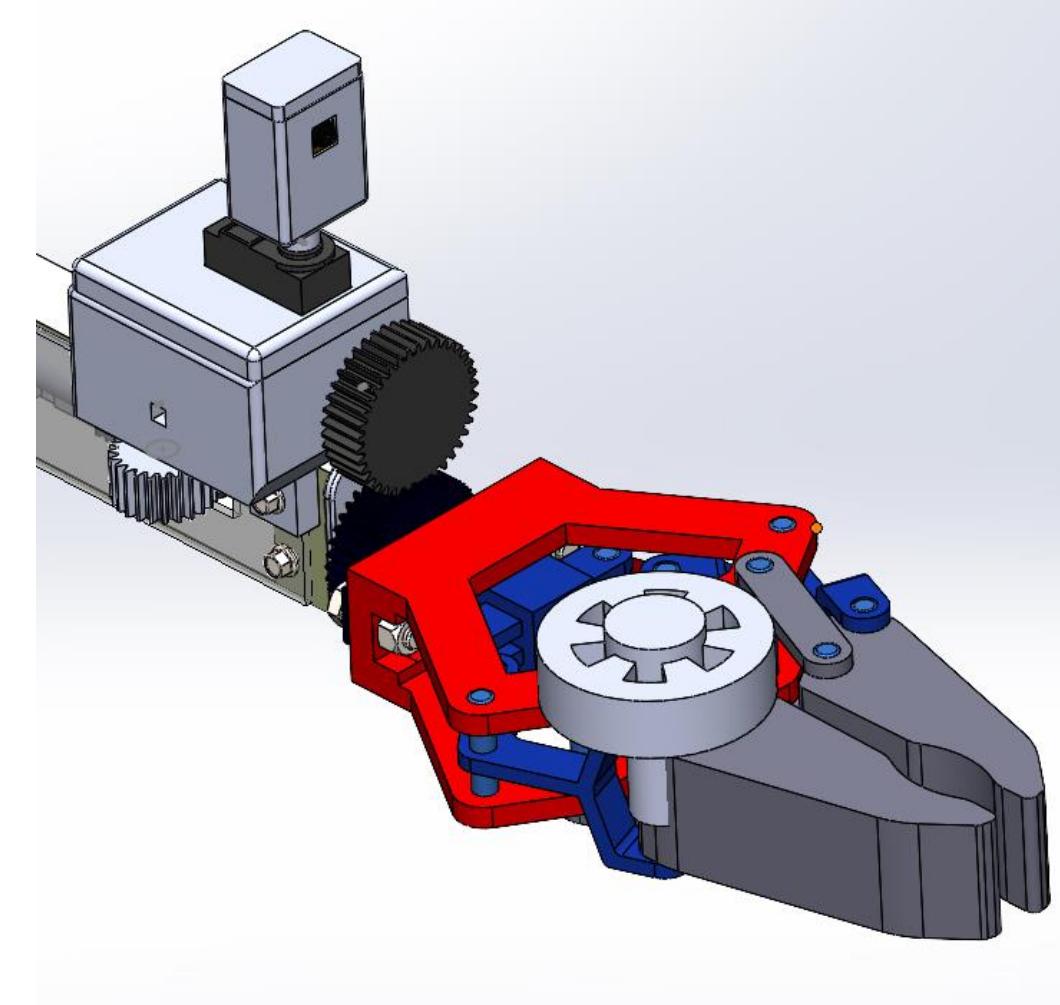
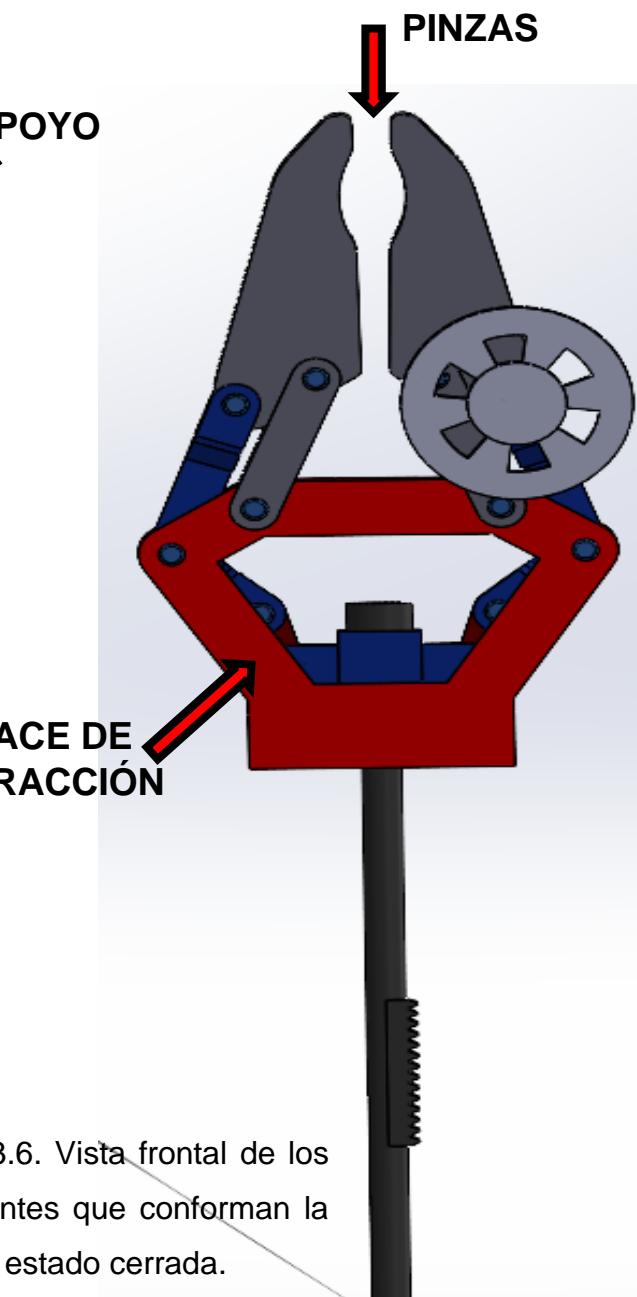


Figura 13.7. Vista isométrica del sistema garra y componentes, estado cerrada.

5) Brazo robótico ensamblado en el carro de búsqueda y rescate.

Luego de haber llevado a cabo la recopilación de información y criterios específicos para la competencia se fue realizando periódicamente el diseño de las piezas que conforman el brazo robótico, tomando como referencia los movimientos más básicos que un brazo robótico puede hacer se obtuvieron los resultados anteriores, ahora se mostrarán a continuación una serie de imágenes donde se describen gráficamente los posibles límites de distancia y posición que el brazo podría alcanzar.

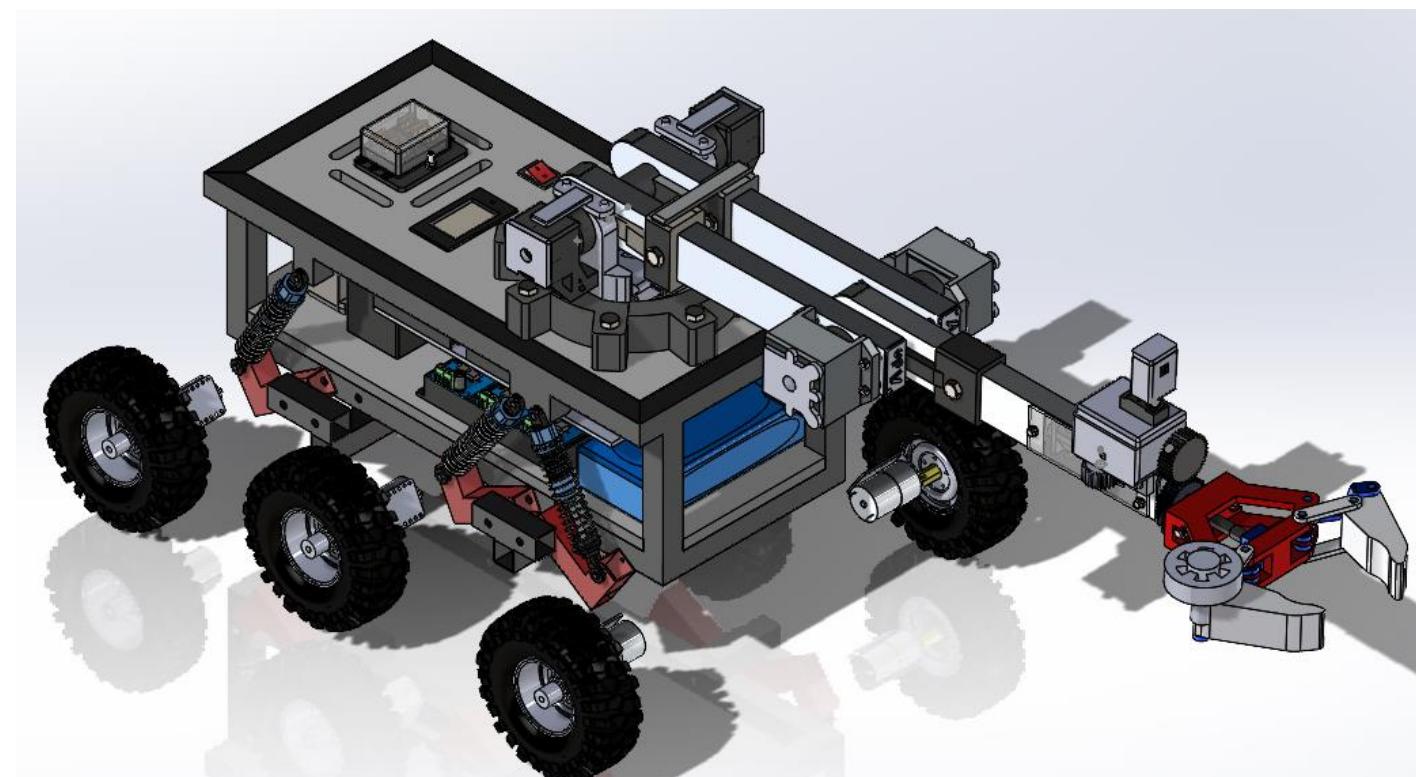


Figura 14. Vista isométrica del brazo robótico ensamblado en el modelo 3D del carro robot de rescate donde será movilizado en la competencia del TMR 2024.

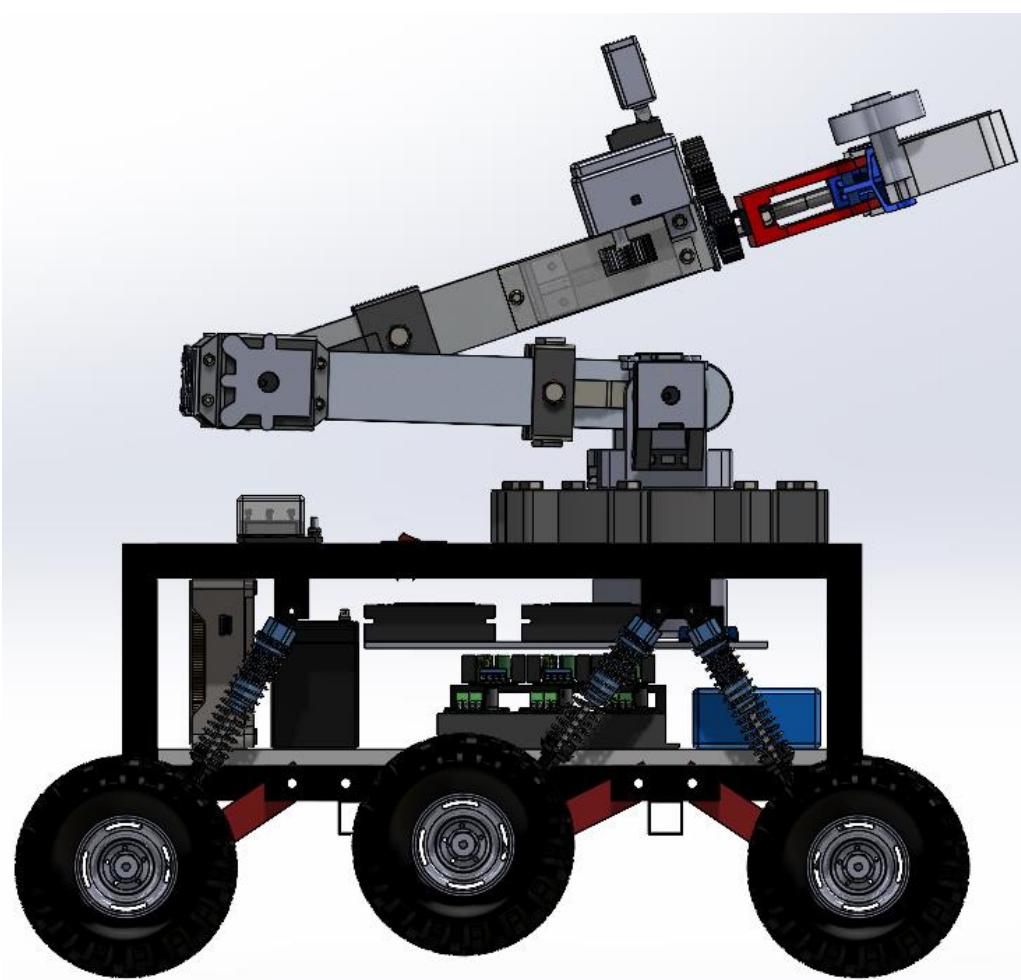


Figura 14.2. Vista lateral derecha del brazo robótico ensamblado en el carro robot de rescate.

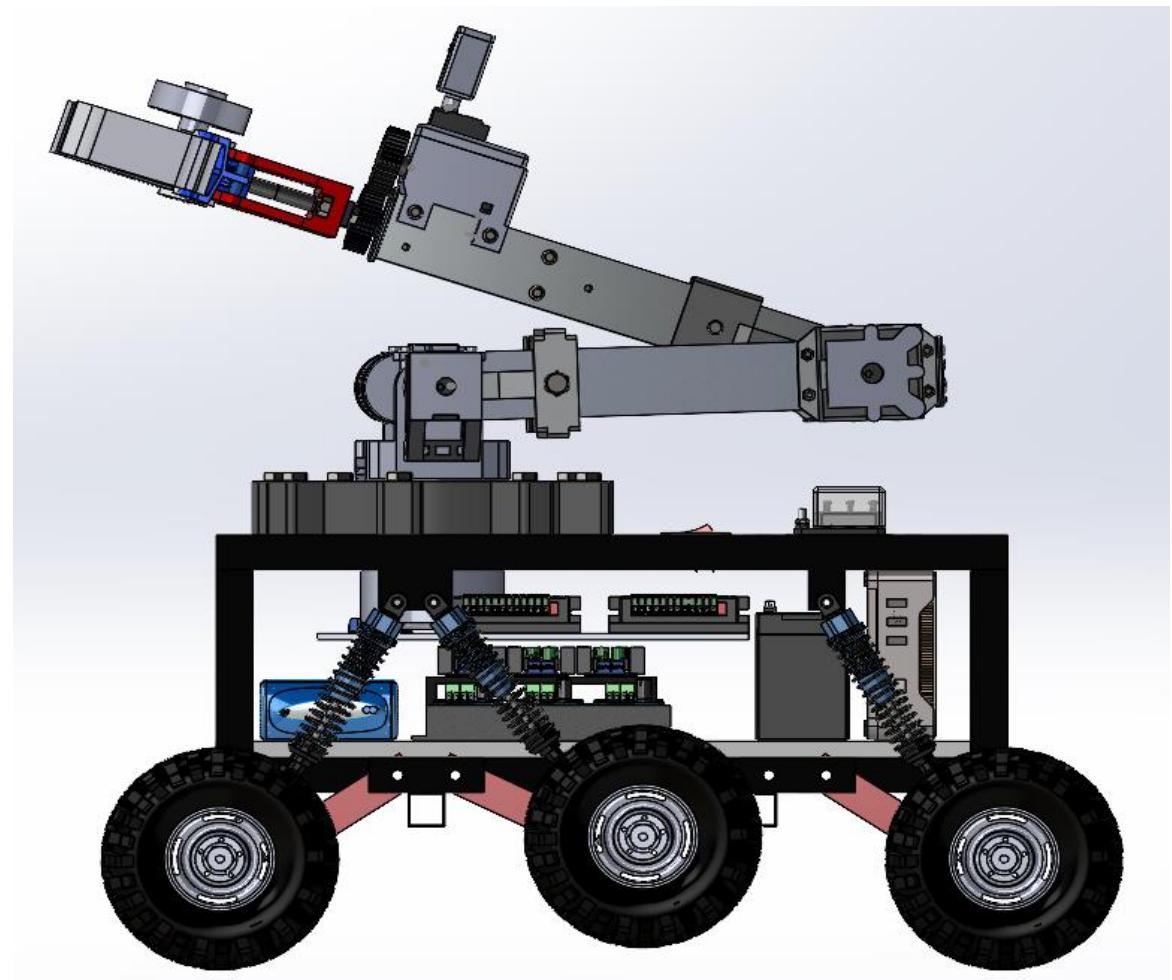


Figura 14.3. Vista lateral izquierda del brazo robótico ensamblado en el carro robot de rescate.

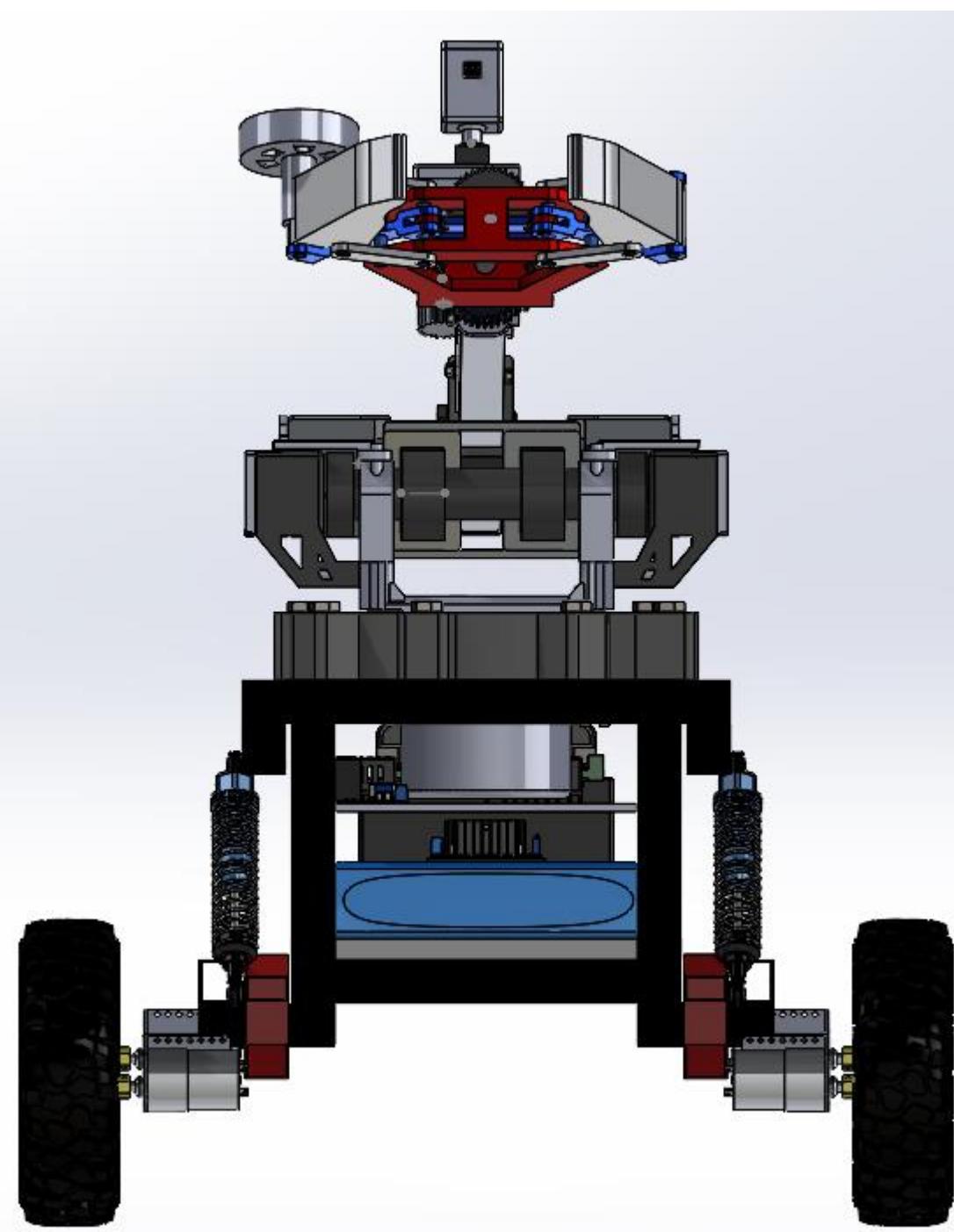


Figura 14.4 Vista frontal del brazo robótico ensamblado en el carro robot de rescate.

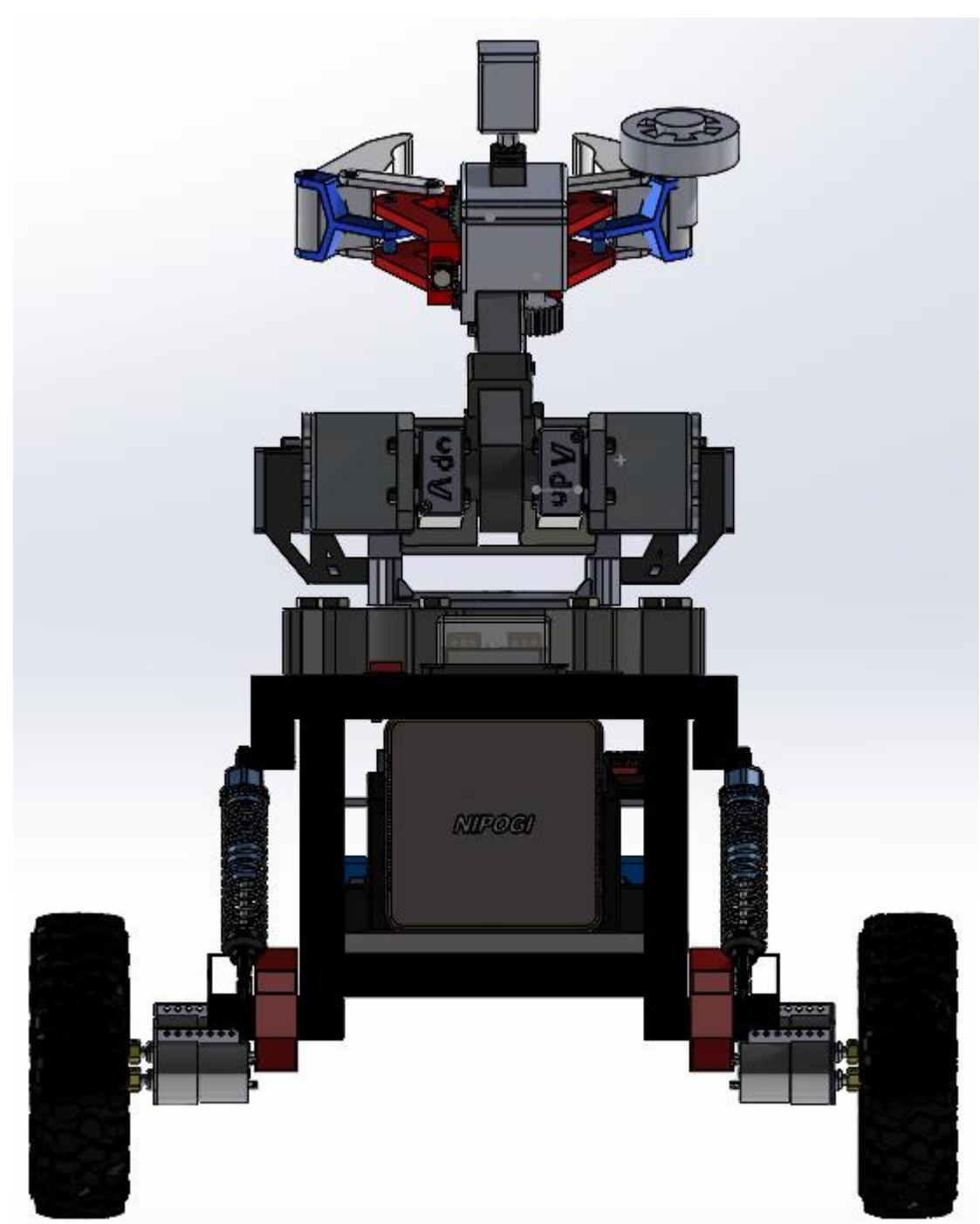


Figura 14.5 Vista trasera del brazo robótico ensamblado en el carro robot de rescate.

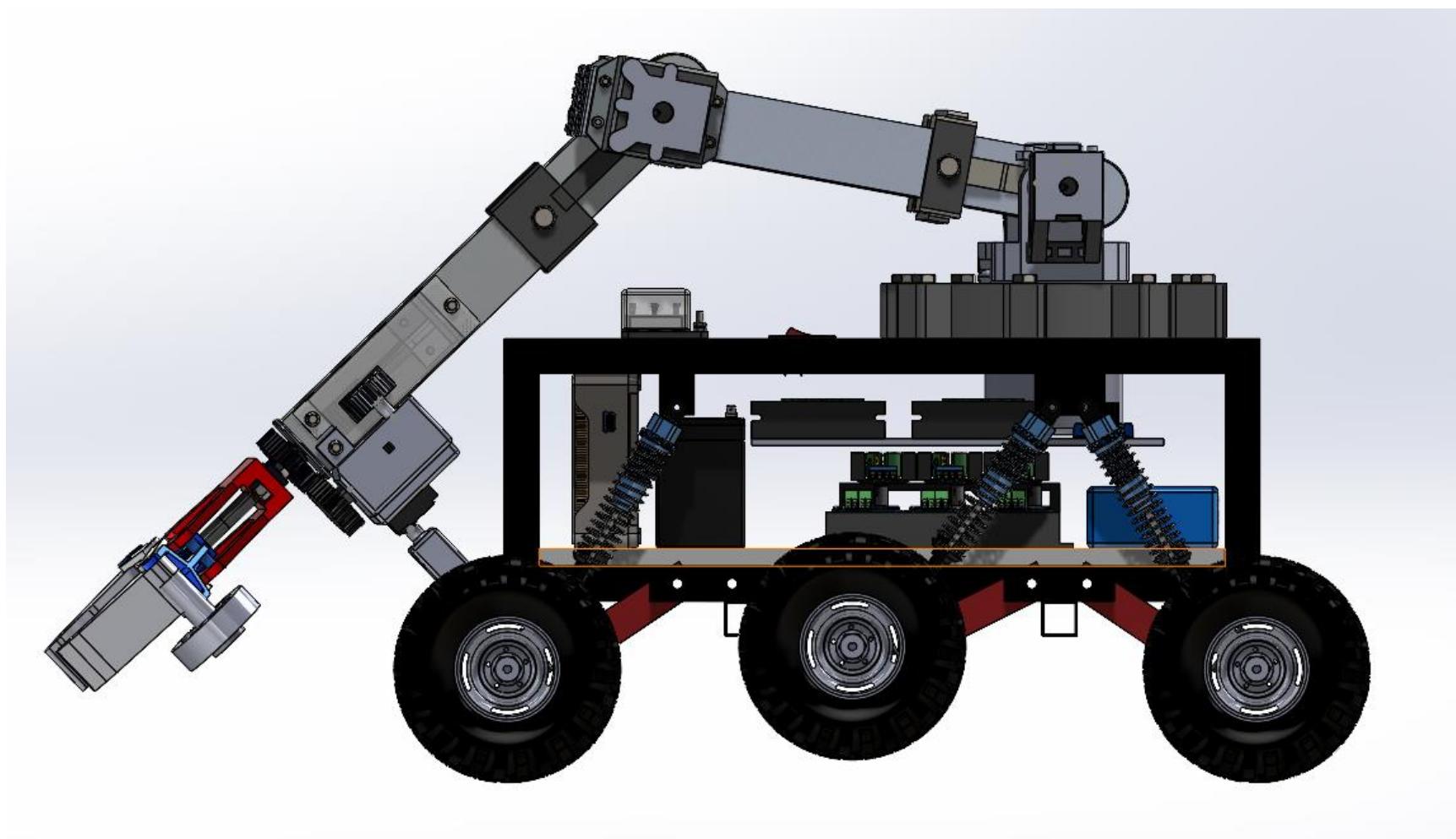


Figura 14.6. Vista lateral derecha del brazo robótico girado inversamente.

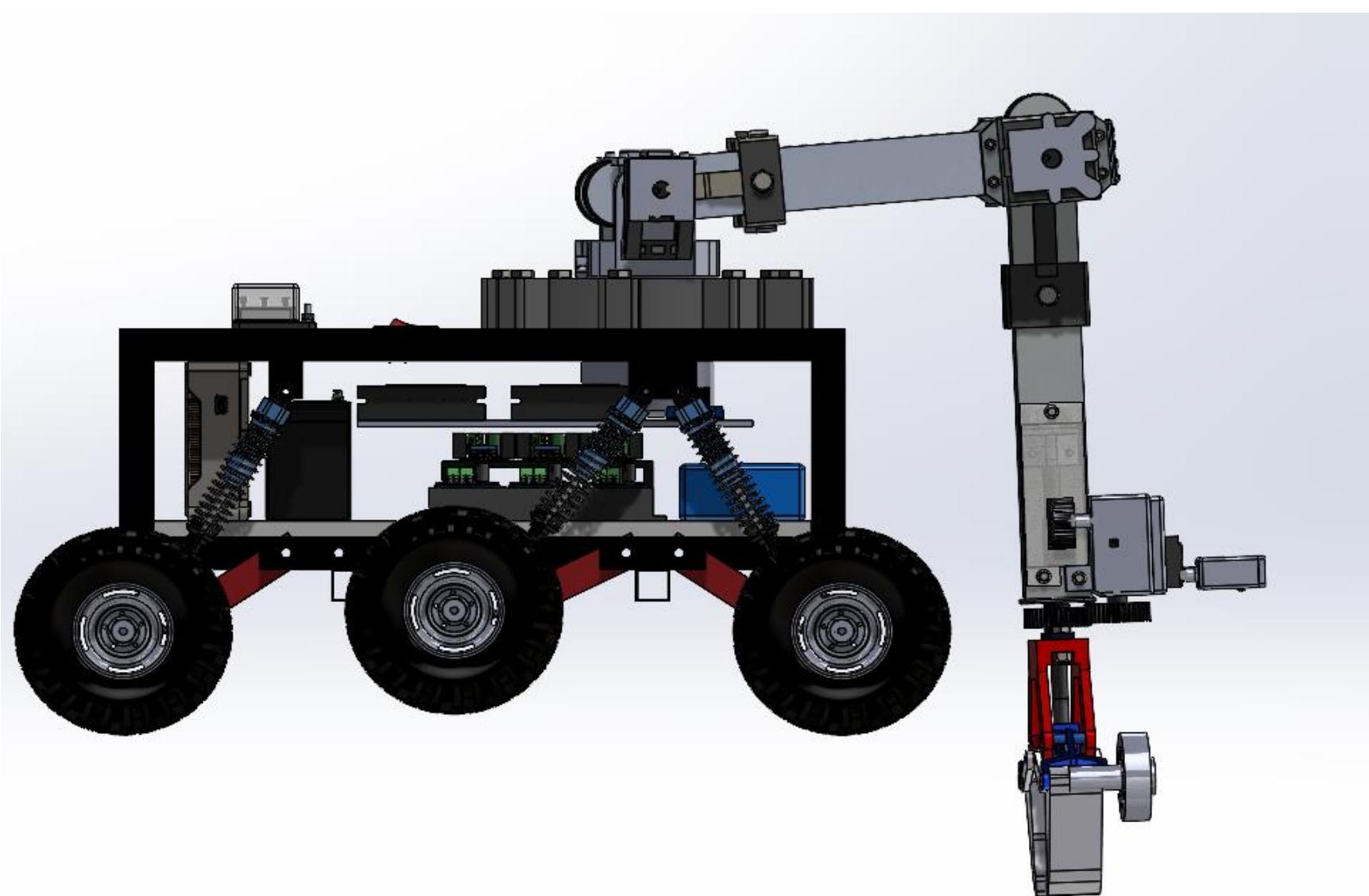


Figura 14.7. Vista lateral derecha del brazo robótico posicionado en escuadra para la toma de objetos o apoyo como función de un gato hidráulico en caso atascamiento

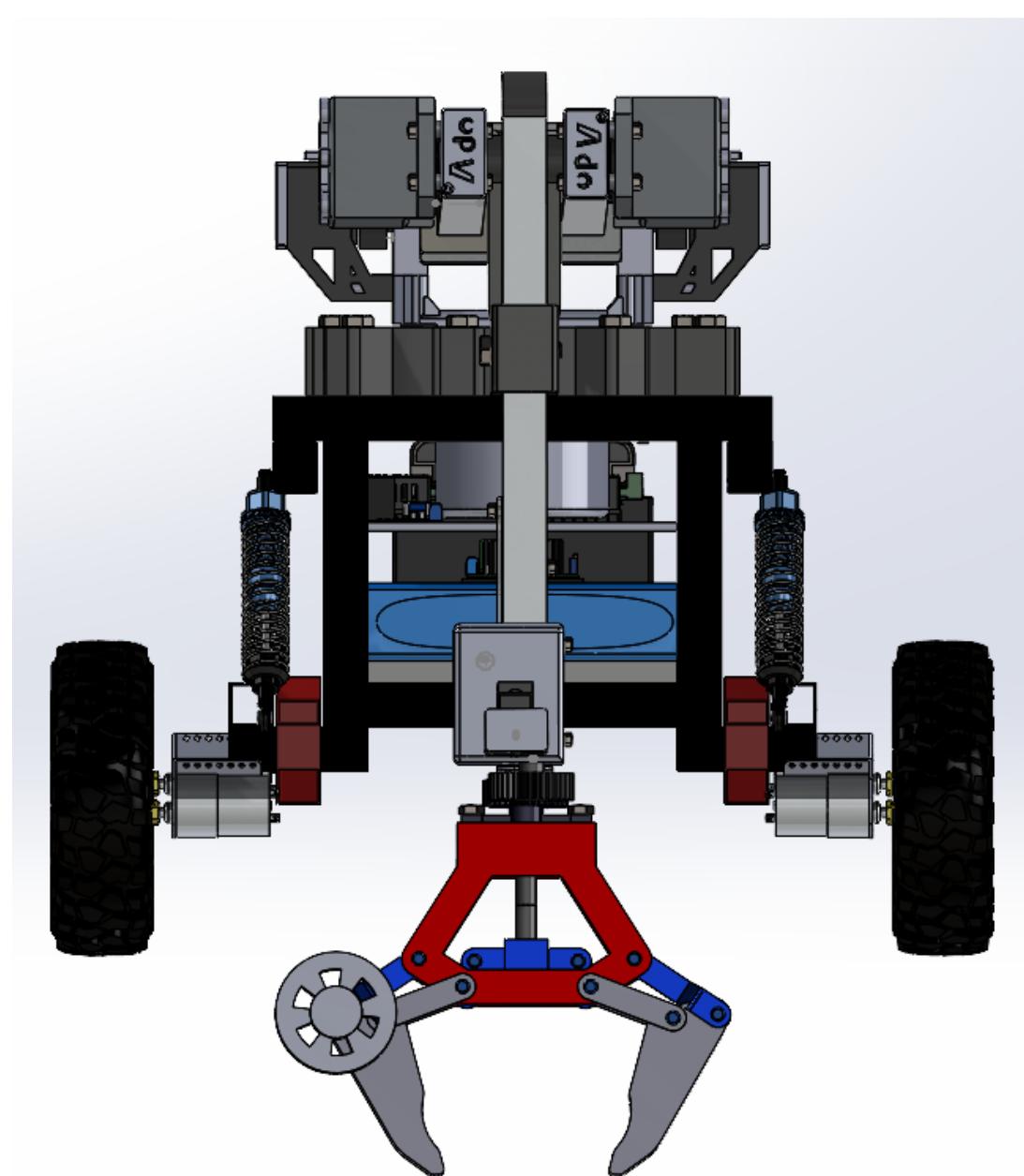


Figura 14.8. Vista frontal del brazo respecto a la posición descrita anteriormente.

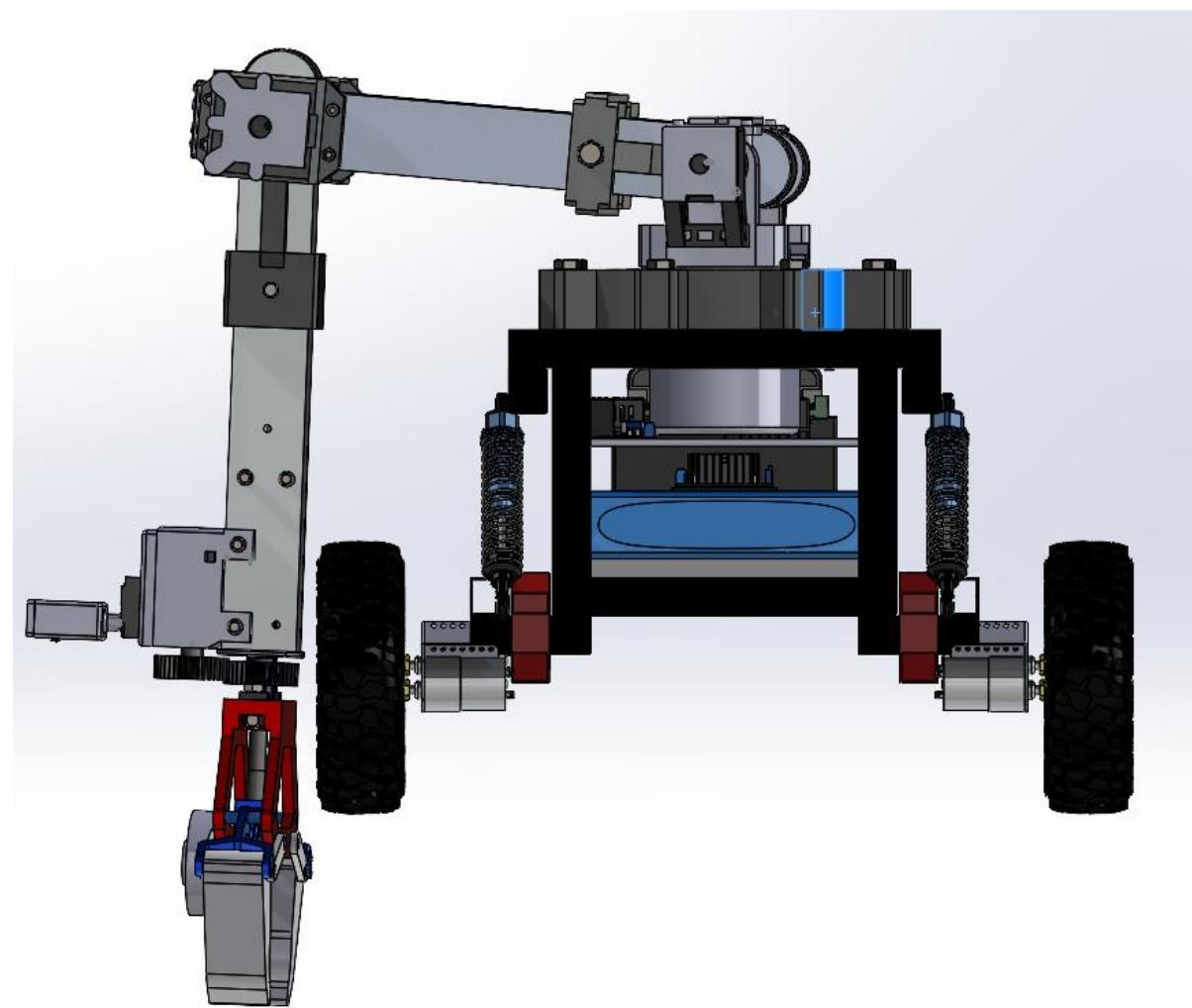


Figura 14.9. Vista frontal del brazo robótico posicionado al lado derecho en escuadra para la toma de objetos o apoyo como función de un gato hidráulico en caso atascamiento

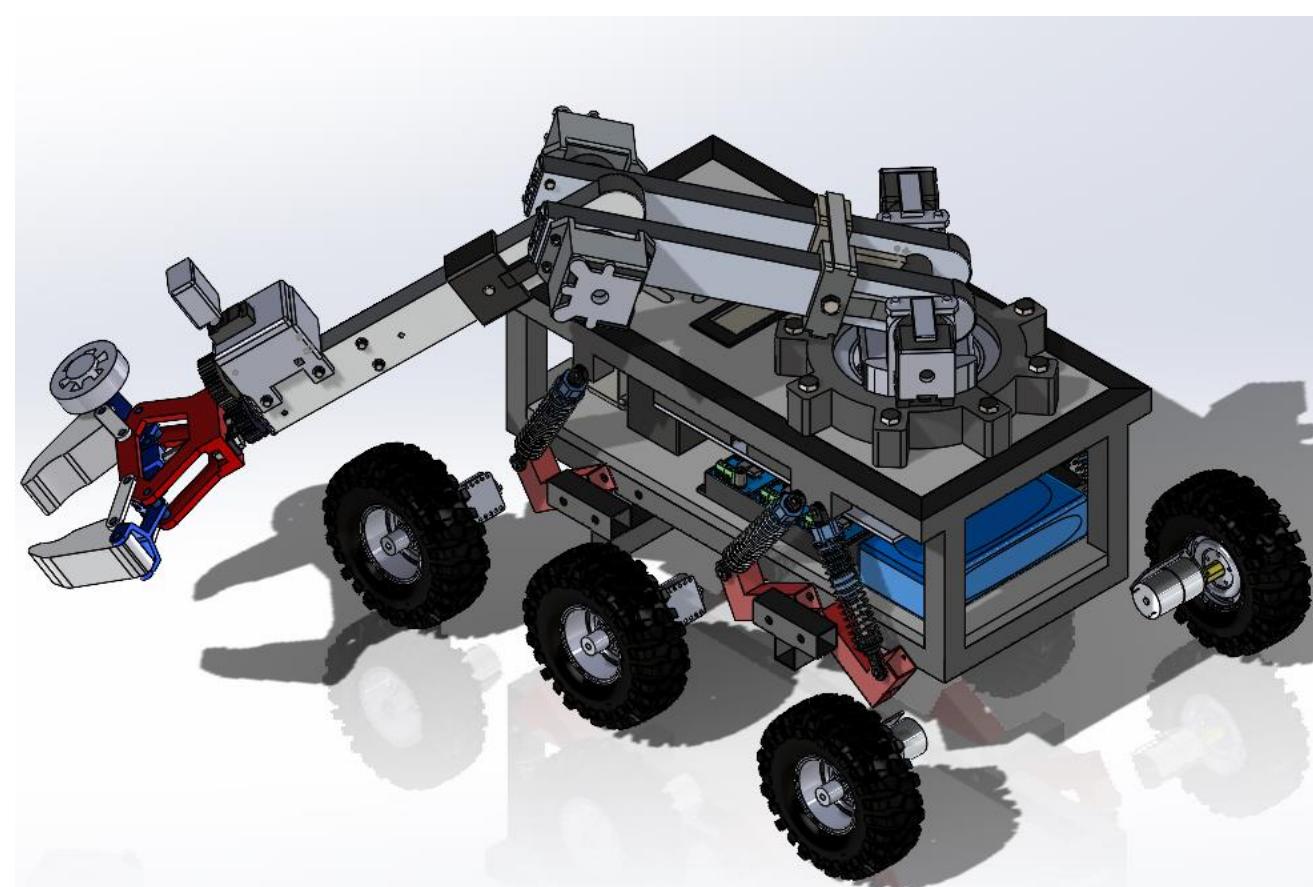


Figura 14.10. Vista isométrica del rango de distancia que el brazo podría alcanzar lateralmente hacia atrás.



Figura 14.11. Vista trasera del brazo robótico alzado hacia arriba.

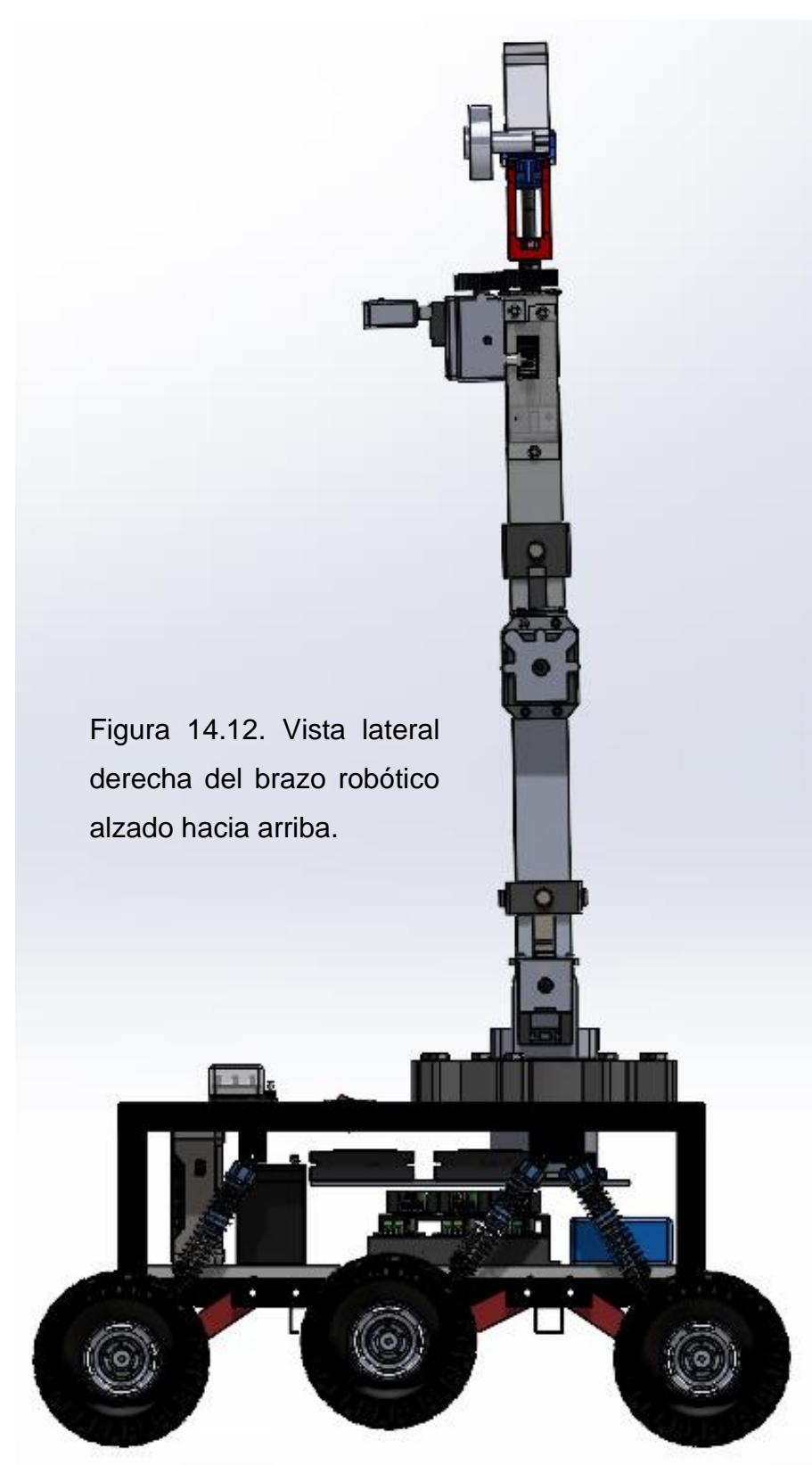
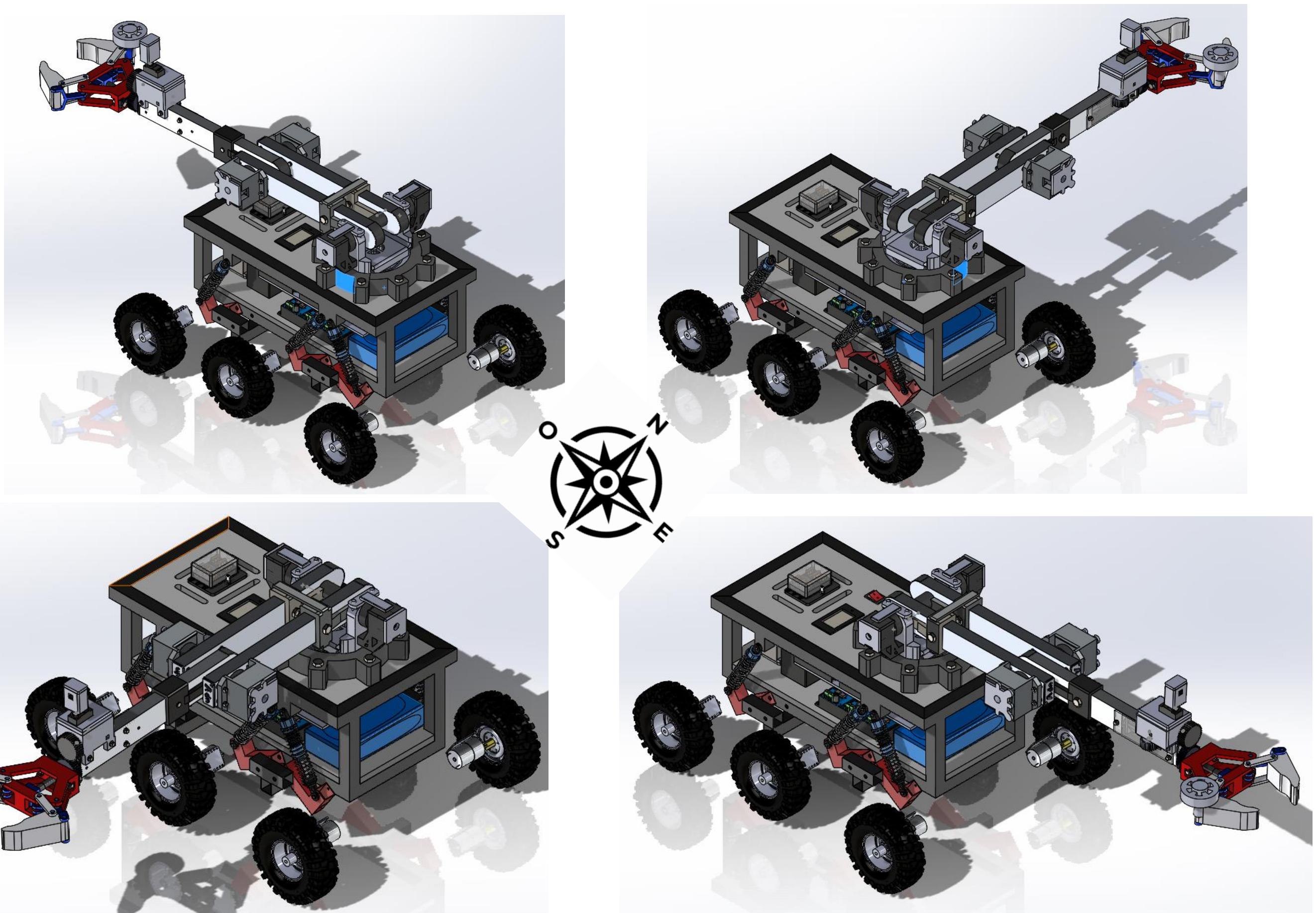


Figura 14.12. Vista lateral derecha del brazo robótico alzado hacia arriba.

Figura 14.13. Vista isométrica de las posibles posiciones alcanzadas por el brazo robótico en una forma alzada tomando como referencia los puntos cardinales.



6) Control remoto del brazo robótico.

1) Sistema receptor. Dada la encomienda de buscar una resolución al control inalámbrico de los motores nema y los servomotores se optó por crear dos sistemas independientes(receptor y transmisor, 2 códigos separados de cada uno) para después fusionarlos mecánicamente en uno mismo, para esto se ideó el primer sistema receptor que controlará los motores nema 17 de los eslabones y el motor DC para rotar la base por medio de un puente h y 4 TB600, estos drivers con función en la determinación de parámetros en la aceleración, velocidad, torque y numero de pasos por revolución, etc. Estarán colocados en la parte media del robot donde se encuentran los demás componentes eléctricos serán los encargados de suministrar la energía necesaria al robot (como se muestra en la figura 15). Las señales recibidas serán procesadas mediante los respectivos cables de señal de los motores nema y el motor DC por un arduino receptor que tendrá conectado un módulo NRF24L01 y los respectivos cables de señal de los motores nema y el motor DC. Mientras que los servomotores correspondientes al movimiento de la garra y la cámara funcionaran únicamente estando conectados los cables de señal directamente al arduino y este a otro módulo NRF24L01 (figura 15.2).

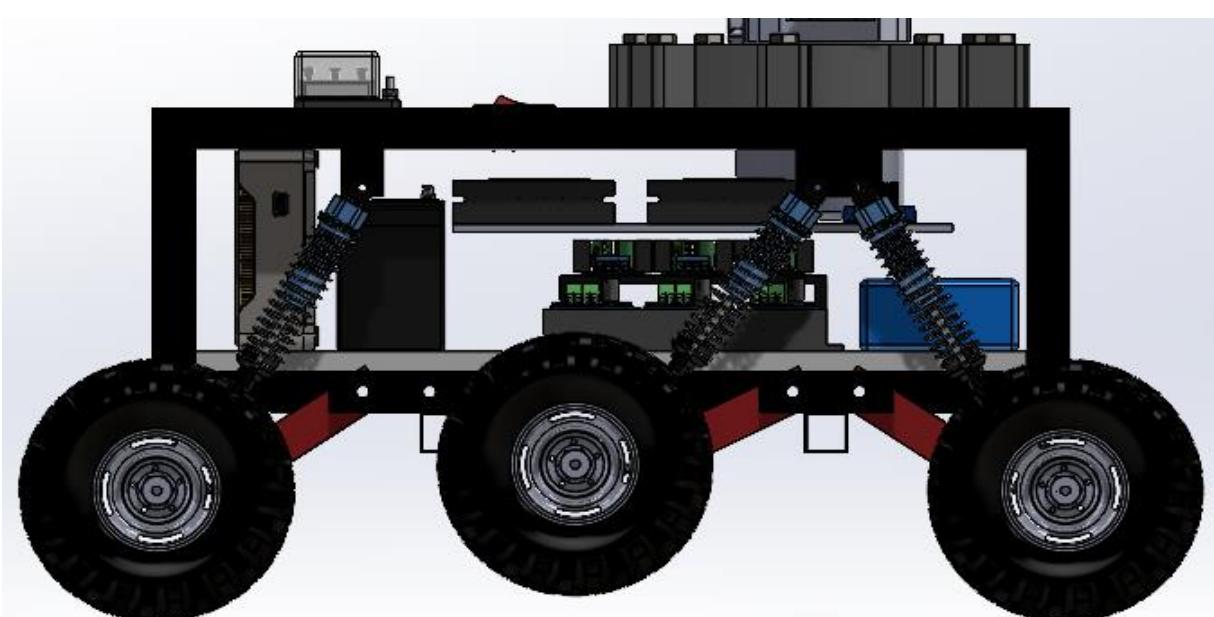


Figura 15. En la parte media de la estructura metálica del robot se puede observar la variedad de componentes que integran el control eléctrico del brazo robótico

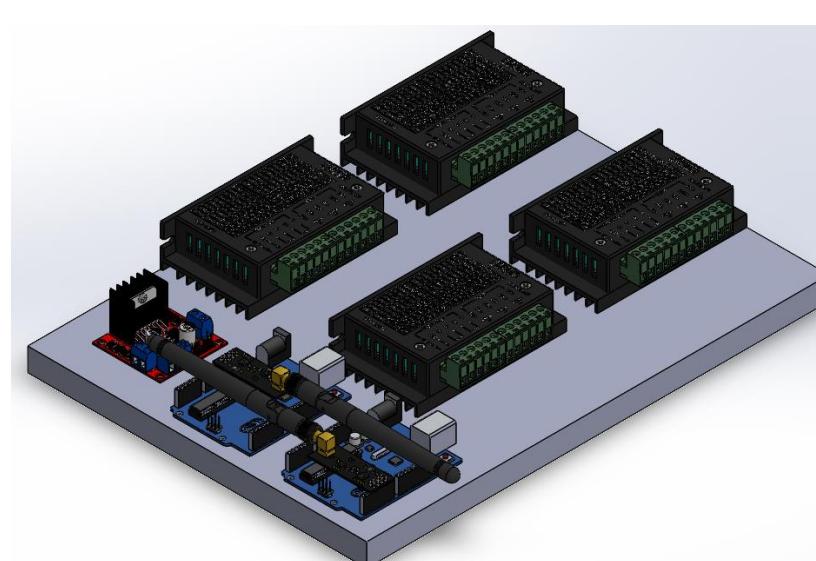


Figura 15.2. Ejemplo del tablero de control de drivers, puente h, arduinos y módulos NRF24L01.

2) Sistema transmisor. Para el sistema transmisor de señales se decidió incluir el ensamble individual de 2 joystick shield sobre 2 arduinos, cada uno con su respectivo módulo de radiofrecuencia, estarán alimentados eléctricamente por medio de un slot especial situado en la parte trasera, para integrar ya sea un power bank o una batería de 9 volts. Los botones y joysticks del mando en conjunto serán configurados en los programas creados en el software de Arduino para decidir el giro y velocidad de los motores en la dirección especificada. Seguidamente se mostrará la descripción del ensamble de los elementos que integran el mando de control remoto.

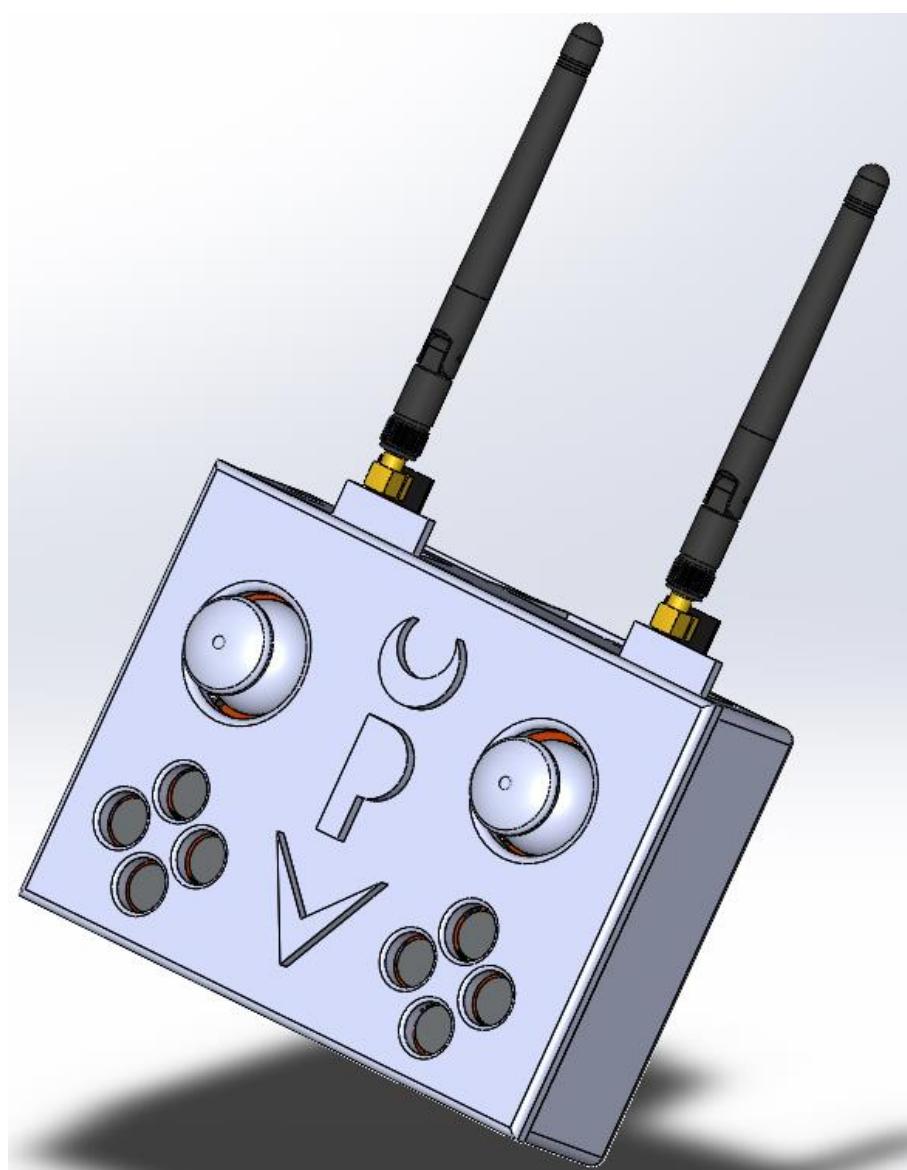


Figura 16. Vista isométrica frontal del mando remoto con todos sus componentes integrados.



Figura 16.2 Vista isométrica trasera del mando remoto.

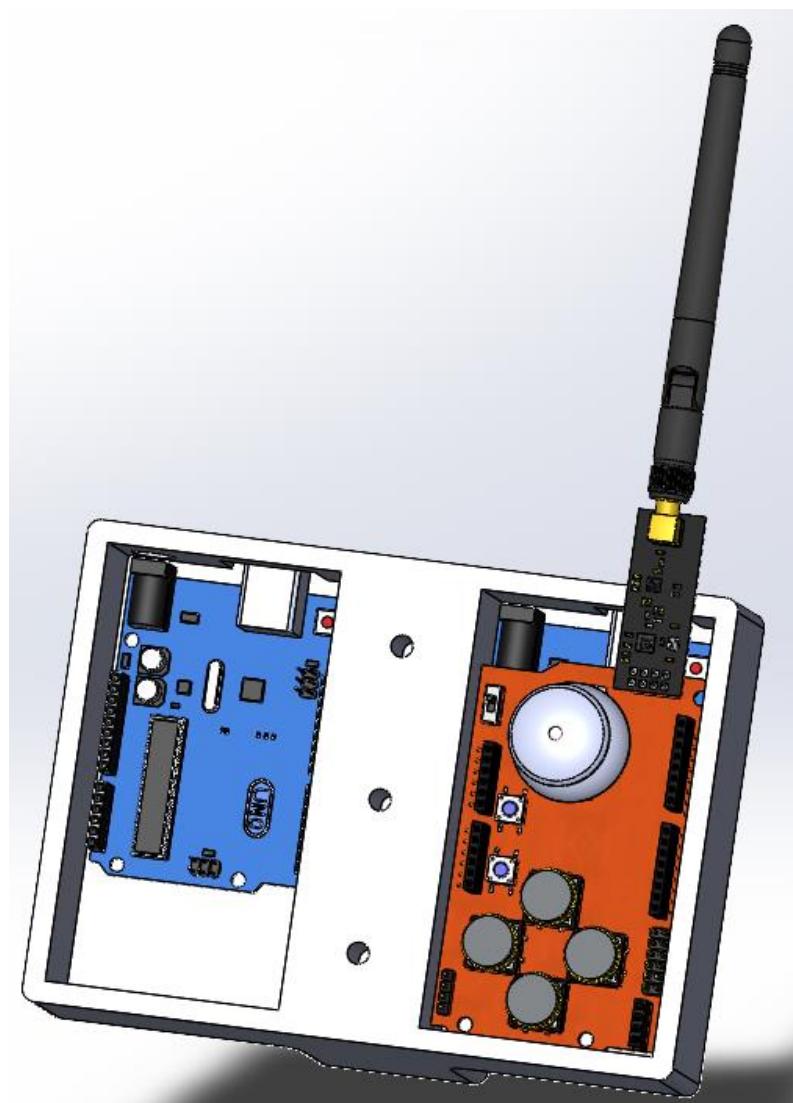


Figura 16.3 Vista hacia el interior del mando donde se pueden observar las cavidades para fijar los componentes y el orden en que estos colocados situados.

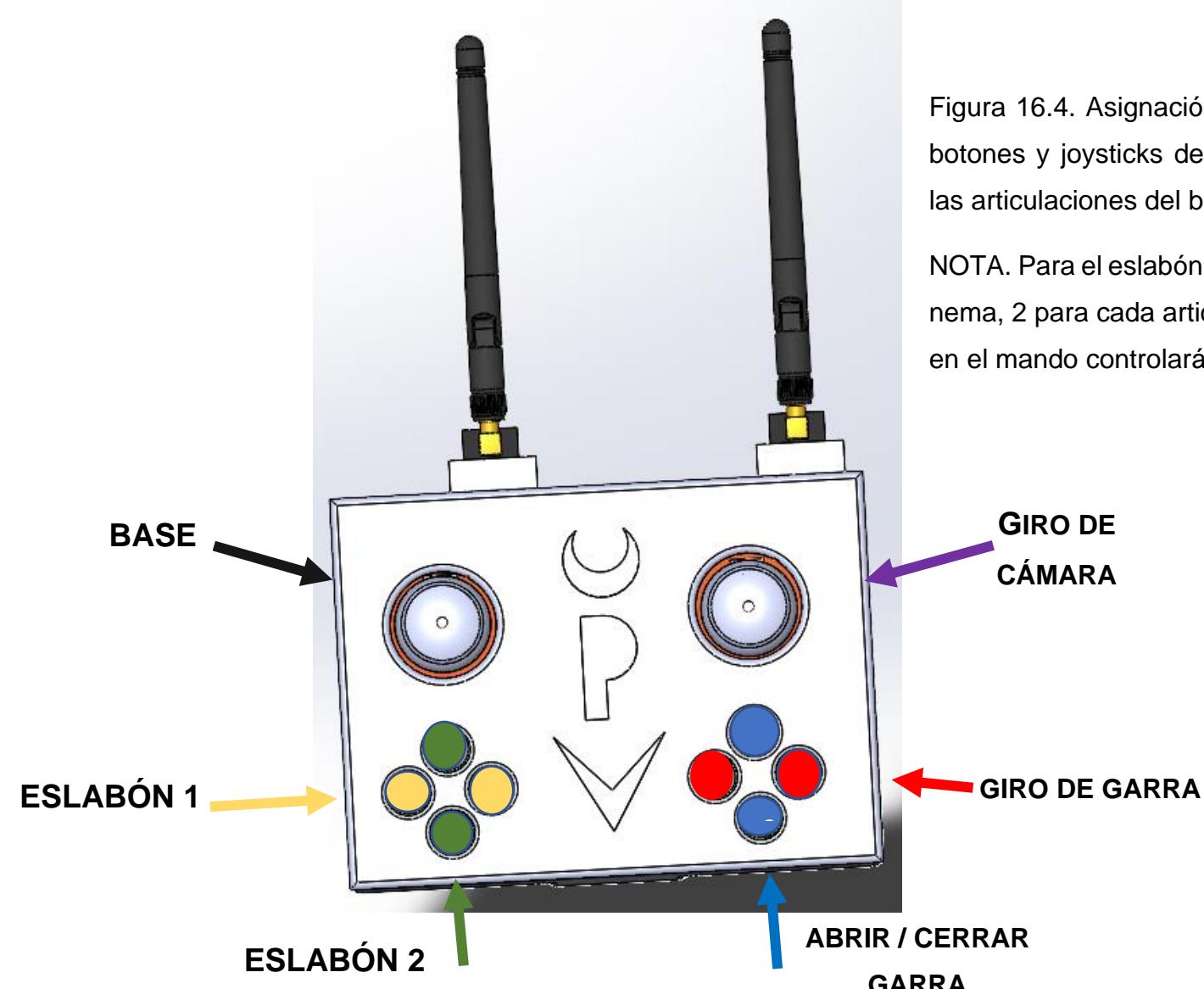


Figura 16.4. Asignación de las direcciones en los botones y joysticks del mando para el control en las articulaciones del brazo robótico.

NOTA. Para el eslabón 1 y 2 se emplean 4 motores nema, 2 para cada articulación, por lo que 1 botón en el mando controlará la dirección de 2 motores.

7) Programación en Arduino para los códigos transmisores, receptores y servidor web para la transmisión de video.

Se optó programar por medio de Arduino gracias a las bibliotecas con las que cuenta este software que son necesarias para los drivers, los NRF24L01 y la ESP32 CAM, hacen más fácil la programación de los motores nema, servos, la transmisión de señales y recepción entre controladores por medio del módulo rc. Cada placa joystick shield debe ser programada en base a las especificaciones y funciones que se les asignen a los pulsadores y joysticks, ya que como se declaró anteriormente en la figura 16.4, las direcciones de las articulaciones deben coincidir con las del código, si en el programa se declarara un pin de señal de la placa equivocado este automáticamente quedaría inhabilitado por la predeterminada asignación de pines que debe ser cumplida al ensamblar los componentes (figura 5).

1) Sistema transmisor. Código de configuración de mando para los joysticks shields.

En este caso debido a que se emplearon 2 placas joysticks con los mismos componentes, este código puede ser empleado únicamente para el envío de las señales que los pulsadores emitan hacia el módulo rc receptor. Lo que necesariamente debe ser cambiado y es de vital importancia es la dirección de escritura, que de ser la misma escrita en el programa para los sistemas transmisores y receptores y ser subida al Arduino ocasionaría una coalición y fallos, como se señala en la figura 17.2.

```
CODIGO_DE_MANDO | Arduino 1.8.19
File Edit Sketch Tools Help
CODIGO_DE_MANDO $ 
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9, 10); // Configura los pines CE y CSN del módulo NRF24L01

//CCW = COUNTER CLOCKWORK      CW = CLOCKWORK
//CONFIGURACION PREDETERMINADA DE PINES DIGITALES EN BASE A LA PLACA JOYSTICK SHIELD

#define BUTTON_MOTOR1_CW 2 // Botón CW para motor 1
#define BUTTON_MOTOR1_CCW 4 // Botón CCW para motor 1
#define BUTTON_MOTOR2_CW 3 // Botón CW para motor 2
#define BUTTON_MOTOR2_CCW 5 // Botón CCW para motor 2
int joystT = A1; // Conectado al eje Y del joystick (down)

CONTROL DE ESLABÓN 1 || CONTROL DE SERVOMOTOR 1 -- GIRO DE GARRA
CONTROL DE ESLABÓN 1 || CONTROL DE SERVOMOTOR 1 --
CONTROL DE ESLABÓN 2 || CONTROL DE SERVOMOTOR 2 -- ABRIR Y CERRAR PINZAS
CONTROL DE ESLABÓN 2 || CONTROL DE SERVOMOTOR 2 --
CONTROL DE BASE || CONTROL DE SERVOMOTOR 3 -- GIRO DE CÁMARA
```

Figura 17. Parte inicial del código donde las bibliotecas correspondientes al NRF24L01 son agregadas, así como la definición de pines para los botones que controlaran las direcciones de los motores y servos en el mando.

```
int data[5]; // Array para almacenar los datos a enviar

void setup() {
    radio.begin();
    radio.openWritingPipe(0x1); // Dirección de escritura, SE CAMBIA 0x2 PARA LA SEGUNDA PLACA DEBIDO A QUE ES UN SISTEMA DE TRANSMISIÓN DISTINTO
    radio.setPALevel(RF24_PA_HIGH); // Potencia de transmisión alta
    radio.stopListening(); // NRF configurado para transmitir datos
    pinMode(BUTTON_MOTOR1_CW, INPUT);
    pinMode(BUTTON_MOTOR1_CCW, INPUT);
    pinMode(BUTTON_MOTOR2_CW, INPUT);
    pinMode(BUTTON_MOTOR2_CCW, INPUT);
}
```

Figura 17.2. Se procede a crear un array de datos, definir la dirección de escritura para configurar el NRF como sistema transmisor y fijando los botones de la placa como entradas.

```
void loop() {
    // Leer los valores de los botones
    data[0] = digitalRead(BUTTON_MOTOR1_CW);
    data[1] = digitalRead(BUTTON_MOTOR1_CCW);
    data[2] = digitalRead(BUTTON_MOTOR2_CW);
    data[3] = digitalRead(BUTTON_MOTOR2_CCW);

    // Leer los valores del joystick
    int position_Y = analogRead(joystT);
    int angulo3 = map(position_Y, 0, 1023, 0, 180);
    data[4] = analogRead(joystT);

    // Enviar los datos al receptor
    radio.write(data, sizeof(data));
    delay(10); // Añade un pequeño retraso si es necesario
}
```

Figura 17.3. En el ciclo se enlistan los datos a enviar del array, que son las señales de los botones y joystick para ser recibidos posteriormente por Arduino+NRF del sistema receptor correspondiente, ya sea el control de los eslabones y la base o la garra y cámara.

2) Sistema receptor, motores nema 17. Código de configuración para el control de la base y eslabones.

Luego de haber mostrado y analizado el código para el sistema transmisor a continuación se expondrá un boceto de lo que se planea realizar con la dirección de control en los motores nema, en la figura , esto debido a que la velocidad, aceleración, números de pasos y torque necesario aún no están bien definidos hasta que el proyecto este ensamblado físicamente en su totalidad y se hayan realizado los ensayos necesarios con el fin de determinar los parámetros más óptimos para su movilidad en la práctica.

```
SISTEMA_RECECTOR_NEMA | Arduino 1.8.19
File Edit Sketch Tools Help
SISTEMA_RECECTOR_NEMA 
#include <SPI.h>
#include <RF24.h>
#include <nRF24L01.h>
#include <AccelStepper.h>

RF24 radio(9, 10); // Configurar los pines CE y CSN del módulo NRF24L01

AccelStepper motor1(AccelStepper::FULL4WIRE, 2, 3); // Pines de control de motor 1 -- ESLABÓN 1
AccelStepper motor2(AccelStepper::FULL4WIRE, 4, 5); // Pines de control de motor 2 -- ESLABÓN 2
AccelStepper motor3(AccelStepper::FULL4WIRE, 6, 7); // Pines de control de motor 3 -- BASE
```

Figura 18. El código comienza incluyendo las bibliotecas del módulo rc y definiendo los canales de comunicación en el arduino, con la biblioteca AccelStepper la definición de pines correspondientes a las conexiones del driver con el arduino las cuales deben coincidir. (como se muestra en la figura 2.3)

```

void setup() {
    radio.begin();
    radio.openReadingPipe(1, 0x1); // Dirección de lectura, SE CAMBIA 0x2 PARA LA SEGUNDA PLACA
    radio.setPALevel(RF24_PA_HIGH); // Potencia de recepción alta
    radio.startListening(); //NRF configurado para recibir datos

    //CONFIGURACION DE VELOCIDAD Y ACCELERACION MAXIMA EN LOS MOTORES A PASOS
    motor1.setMaxSpeed(1000); //
    motor1.setAcceleration(500); //
    motor2.setMaxSpeed(1000); //
    motor2.setAcceleration(500); //
    motor3.setMaxSpeed(1000); //
    motor3.setAcceleration(500); //
}

```

Figura 18.2. Continua con la configuración del NRF receptor y los ajustes de velocidad y aceleración en los motores.

```

void loop() {
    if (radio.available()) {
        // Recibir datos desde el transmisor
        radio.read(&data, sizeof(data)); // Donde "data" es tu array de datos

        //Array para procesar las señales que controlaran los motores
        int BUTTON_MOTOR1_CW = data[0];
        int BUTTON_MOTOR1_CCW = data[1];
        int BUTTON_MOTOR2_CW = data[2];
        int BUTTON_MOTOR2_CCW = data[3];
        int joysT = data[4];
    }
}

```

Figura 18.3. Ciclo de lectura de datos procesados por el array.

```

//Array para procesar las señales que controlaran los motores
int BUTTON_MOTOR1_CW = data[0];
int BUTTON_MOTOR1_CCW = data[1];
int BUTTON_MOTOR2_CW = data[2];
int BUTTON_MOTOR2_CCW = data[3];
int joysT = data[4];

// Controlar el motor 1 -- ESLABÓN 1
if (BUTTON_MOTOR1_CW == HIGH) {
    motor1.setSpeed(100); // Velocidad positiva para sentido CW
} else if (BUTTON_MOTOR1_CCW == HIGH) {
    motor1.setSpeed(-100); // Velocidad negativa para sentido CCW
} else {
    motor1.setSpeed(0); // Detener el motor si no se presionan los botones
}

// Controlar el motor 2 -- ESLABÓN 2
if (BUTTON_MOTOR2_CW == HIGH) {
    motor2.setSpeed(100); // Velocidad positiva para sentido CW
} else if (BUTTON_MOTOR2_CCW == HIGH) {
    motor2.setSpeed(-100); // Velocidad negativa para sentido CCW
} else {
    motor2.setSpeed(0); // Detener el motor si no se presionan los botones
}

// Controlar el motor 3 -- BASE (con el joystick)
if (joysT >= 300){
    motor3.setSpeed(100); // Velocidad positiva para sentido horario
} else if (joysT <= 10) {
    motor3.setSpeed(-100); // Velocidad negativa para sentido antihorario
} else {
    motor3.setSpeed(0); // Detener el motor si el joystick no se mueve
}

// Mover los motores
motor1.run();
motor2.run();
motor3.run();
}
}

```

Done compiling.

Figura 18.4. Se pretende que el movimiento de los motores nema sea únicamente con la condición de que mientras sea pulsado un botón, estos giren en la dirección delimitada, de no ser pulsado ninguno entonces no girarán y permanecerán en ese punto. Fin del código.

3) Sistema receptor, servomotores. Código de configuración para el control de la garra y giro de la cámara.

Como se describió en la figura 13 la distribución de los servomotores para el movimiento mecánico de la garra, esta necesitará de 3 servos configurados de forma diferente en su capacidad de apertura, teniendo grados máximos y mínimos distintos.



```

3motoresNRF | Arduino 1.8.19
File Edit Sketch Tools Help
✓ ↻ ≡ ↑ ↓ Verify
3motoresNRF
#include <Servo.h>
#include <RF24.h>
#include <nRF24L01.h>
//DECLARACION DE NUMERO DE SERVOS
Servo servogarra;
Servo servo2giro;
Servo servo3cam;
RF24 radio(9, 10);
//Declaracion de pines de servos usados en el arduino
int servopin1 = 6;
int servopin2 = 7;
int servopin3 = 8;

```

Figura 19. Inclusión de bibliotecas NRF24L01 y para la cantidad de servos en el programa y los pines ocupados en el arduino.

```

int angulo = 0;
int angulo2 = 0;
int joysT = 0;
//estado de los botones motor 1
int buttonState1 = 0;
int buttonState2 = 0;
//estado de los botones motor 2
int buttonState3 = 0;
int buttonState4 = 0;

```

Figura 19.2. Se definen inicialmente la posición de reposo en 0° y el estado de los botones para cuando inicia el programa, estos ángulos podrán variar posteriormente.

```

void setup() {
  radio.begin();
  radio.openReadingPipe(1, 0x2);
  radio.setPALevel(RF24_PA_HIGH);
  radio.startListening();

  servolgarra.attach(servopin1);
  servo2giro.attach(servopin2);
  servo3cam.attach(servopin3);

  servolgarra.write(angulo);
  servo2giro.write(angulo2);
}

```

Figura 19.3. Inicio de la lectura y dirección de comunicación para el recibo de datos en el sistema receptor.

```

if (buttonState1 == HIGH)
{
  angulo++;

  if (angulo >= 180)
  {
    angulo = 180;
  }
}

if (buttonState2 == HIGH)

{
  angulo--;
  if (angulo <= 0)

  {
    angulo = 0;
  }
  servolgarra.write(angulo);
  delay(10);
}

if (buttonState3 == HIGH)
{
  angulo2++;

  if (angulo2 >= 180)

  {
    angulo2 = 180;
  }
}

if (buttonState4 == HIGH)

{
  angulo2--;
  if (angulo2 <= 0)

  {
    angulo2 = 0;
  }
}

servo2giro.write(angulo2);
delay(10);
}

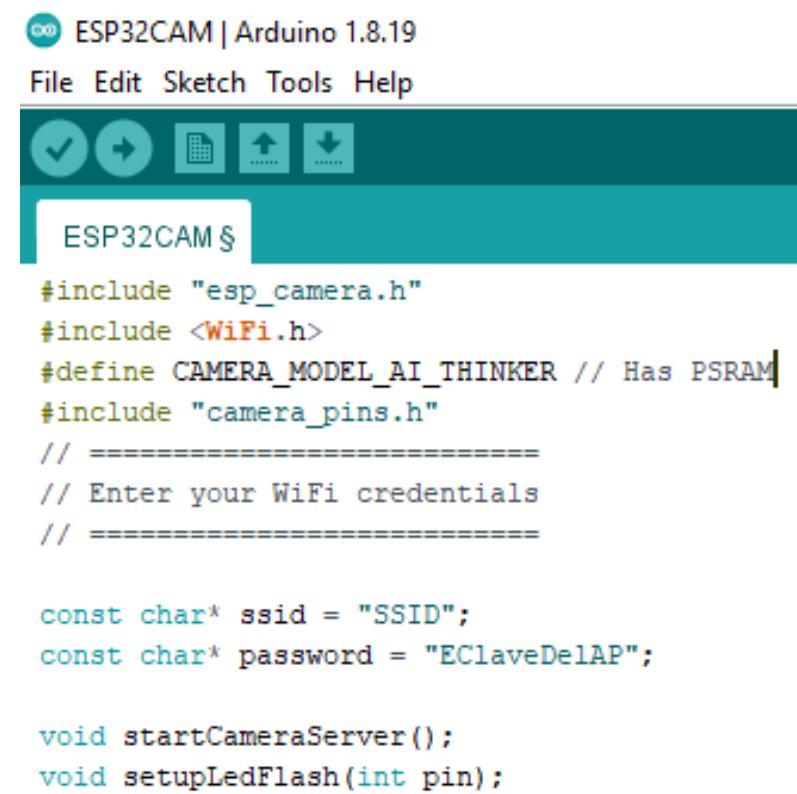
```

Done compiling.

Figura 19.4. Ciclo en donde si se recibe un pulso del mando transmisor el servomotor rotará parcialmente en la dirección seleccionada mientras este presionado su interruptor. En base a la figura 13, para el servomotor 1 se establecerá un grado de libertad máximo para que el servo no pueda seguir avanzando incluso si se sigue presionando el botón, para el servomotor 2 se le permitirá el giro 360° en cualquier dirección. En el tercer servo el joystick controlara el giro de la cámara 360°, pero de no estar en movimiento el joystick el ángulo del servomotor regresará a su estado inicial (mirando hacia el frente de la garra), tal que solo mantendrá el cambio de posición mientras haya un cambio en las lecturas analógicas.

4) Sistema de visión. Código para la transmisión de video en vivo a un servidor web por medio de un ESP32 CAM.

Gran parte importante para el control remoto del brazo es la creación de un sistema de visión transmitido al piloto del robot, por lo que la integración de una ESP32 CAM detrás de las pinzas de agarre que el brazo robótico tiene integrado para la toma de objetos y superación de obstáculos permite un rango de percepción ideal, ya que al ser tan pequeña y compacta no agrega peso extra al brazo ni tampoco carece de problemas para la transmisión de video siempre y cuando tenga una buena conexión a internet. El código que se mostrará es un ejemplo del mismo software Arduino con el propósito de crear un servidor web y transmitir video en vivo para dar acceso a todos los integrantes del equipo en el momento que se requiera.



```
#include "esp_camera.h"
#include <WiFi.h>
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
#include "camera_pins.h"
// =====
// Enter your WiFi credentials
// =====

const char* ssid = "SSID";
const char* password = "EClaveDelAP";

void startCameraServer();
void setupLedFlash(int pin);
```

Figura 20. Se incluye al inicio del código las librerías correspondientes al wifi y modelo de cámara esp, se puede definir un SSID como podría ser un teléfono, una laptop, un pc, etc. Inscripción de la respectiva clave de acceso a la red donde estará conectado para el comienzo del servidor.

```
void setup() {
    Serial.begin(115200);
    Serial.setDebugOutput(true);
    Serial.println();

    camera_config_t config;
    config.ledc_channel = LEDC_CHANNEL_0;
    config.ledc_timer = LEDC_TIMER_0;
    config.pin_d0 = Y2_GPIO_NUM;
    config.pin_d1 = Y3_GPIO_NUM;
    config.pin_d2 = Y4_GPIO_NUM;
    config.pin_d3 = Y5_GPIO_NUM;
    config.pin_d4 = Y6_GPIO_NUM;
    config.pin_d5 = Y7_GPIO_NUM;
    config.pin_d6 = Y8_GPIO_NUM;
    config.pin_d7 = Y9_GPIO_NUM;
    config.pin_xclk = XCLK_GPIO_NUM;
    config.pin_pclk = PCLK_GPIO_NUM;
    config.pin_vsync = VSYNC_GPIO_NUM;
    config.pin_href = HREF_GPIO_NUM;
    config.pin_sccb_sda = SIOD_GPIO_NUM;
    config.pin_sccb_scl = SIOC_GPIO_NUM;
    config.pin_pwdn = PWDN_GPIO_NUM;
    config.pin_reset = RESET_GPIO_NUM;
    config.xclk_freq_hz = 20000000;
    config.frame_size = FRAMESIZE_UXGA;
    config.pixel_format = PIXFORMAT_JPEG; // for streaming
```

Figura 20.2. Configuración de pines digitales de la placa ESP.

```
config.pixel_format = PIXFORMAT_JPEG; // for streaming
//config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
config.fb_count = 1;

// if PSRAM IC present, init with UXGA resolution and higher JPEG quality
// for larger pre-allocated frame buffer.
if(config.pixel_format == PIXFORMAT_JPEG){
    if(psramFound()){
        config.jpeg_quality = 10;
        config.fb_count = 2;
        config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
        // Limit the frame size when PSRAM is not available
        config.frame_size = FRAMESIZE_SVGA;
        config.fb_location = CAMERA_FB_IN_DRAM;
    }
} else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
}

#if defined(CAMERA_MODEL_ESP_EYE)
    pinMode(13, INPUT_PULLUP);
    pinMode(14, INPUT_PULLUP);
#endif
```

Figura 20.3. Configuración de la calidad de imagen con la que se transmitirá al servidor.

```

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
}

sensor_t * s = esp_camera_sensor_get();
// initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
}
// drop down frame size for higher initial frame rate
if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
}

#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(CAMERA_MODEL_M5STACK_ESP32CAM)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
    s->set_vflip(s, 1);
#endif

// Setup LED Flash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
    setupLedFlash(LED_GPIO_NUM);
#endif

```

Figura 20.4. Definición del modelo de cámara, tamaño de imagen, variables de brillo, saturación, espejo, etc.

```

WiFi.begin(ssid, password);
WiFi.setSleep(false);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

startCameraServer();

Serial.print("Camera Ready! Use 'http://");
Serial.print(WiFi.localIP());
Serial.println("' to connect");
}

void loop() {
    // Do nothing. Everything is done in another task by the web server
    delay(10000);
}

```

Figura 20.5. Encriptado SSID y contraseña de la red wifi donde se conectará, el tiempo de espera de conexión y aviso de una conexión exitosa. Fin del código.

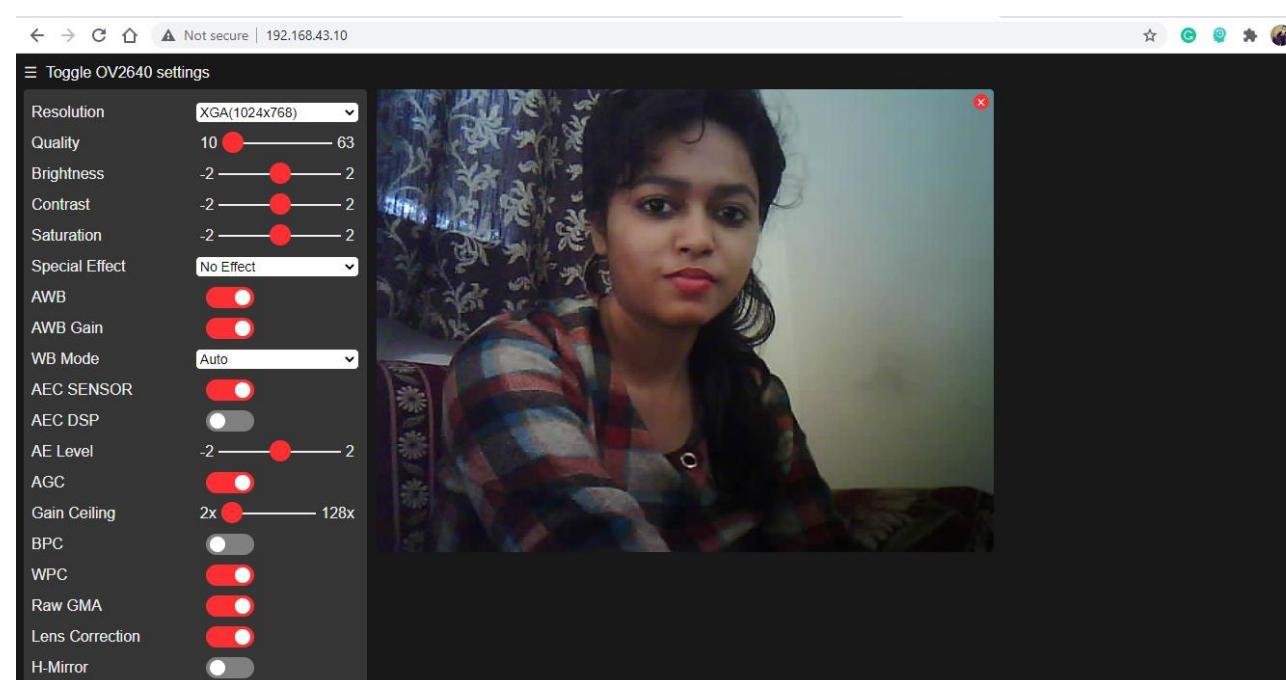


Figura 20.6. Ejemplo del streaming en tiempo real con diversas opciones para el ajuste de video que ofrece la ESP32 CAM.

8) Diagrama de conexión TB6600—MOTOR NEMA 17—ARDUINO MEGA—NRF24L01.

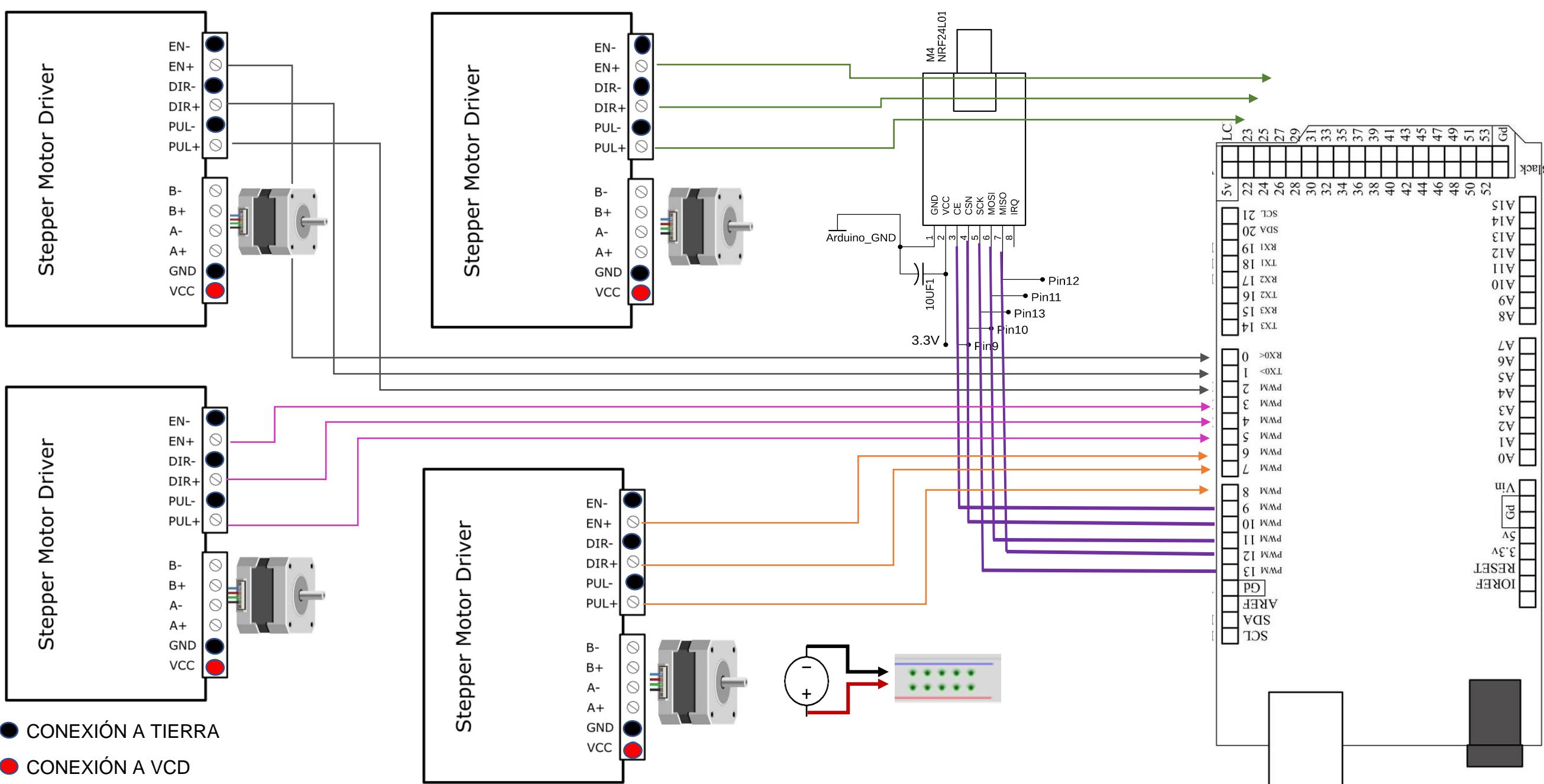


FIGURA 21

9) Diagrama de conexión SERVOMOTORES—ARDUINO—NRF24L01.

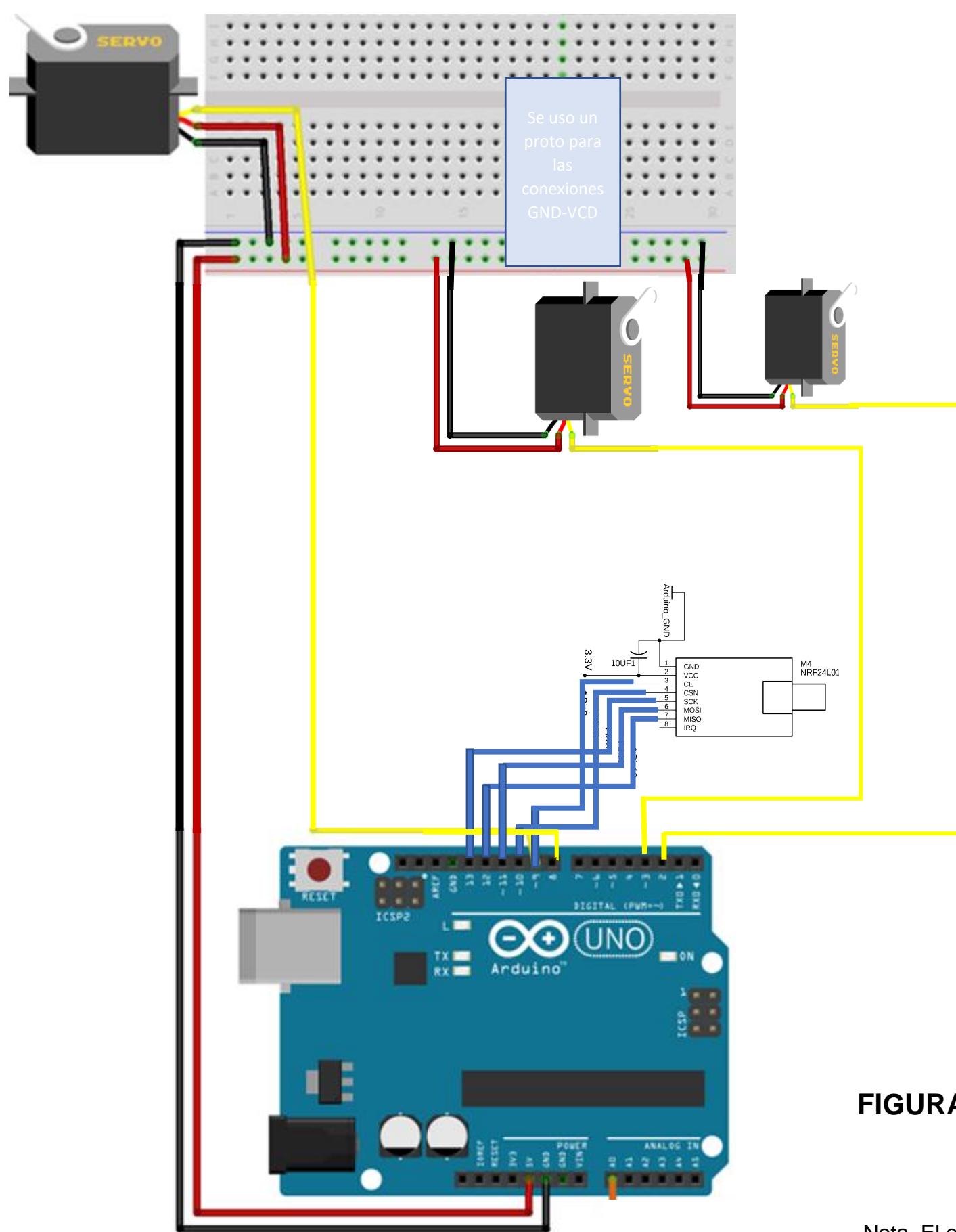


FIGURA 22

Nota. El orden de los pines digitales escogidos en el arduino para los motores puede variar.

10)Conexiones físicas.

Se mostrarán ejemplos de las conexiones que se realizaron entre los componentes, de esta manera fue probado el funcionamiento de los códigos que se hicieron en arduino para controlar los motores del brazo robótico de forma remota.

1) Sistema de control base-eslabones.

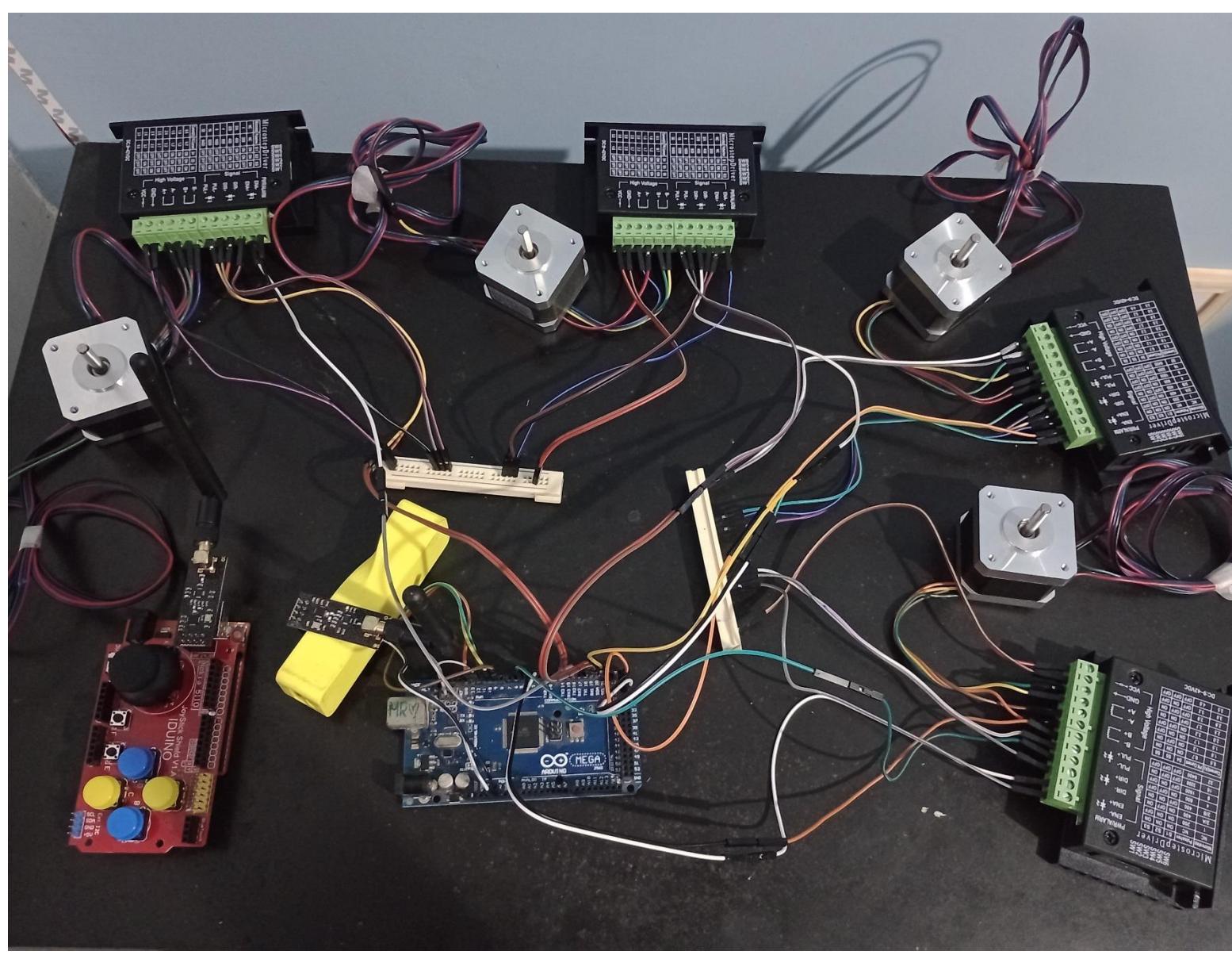


Figura 23. Conexiones en base a la figura 21.

2) Sistema de control garra-cámara.

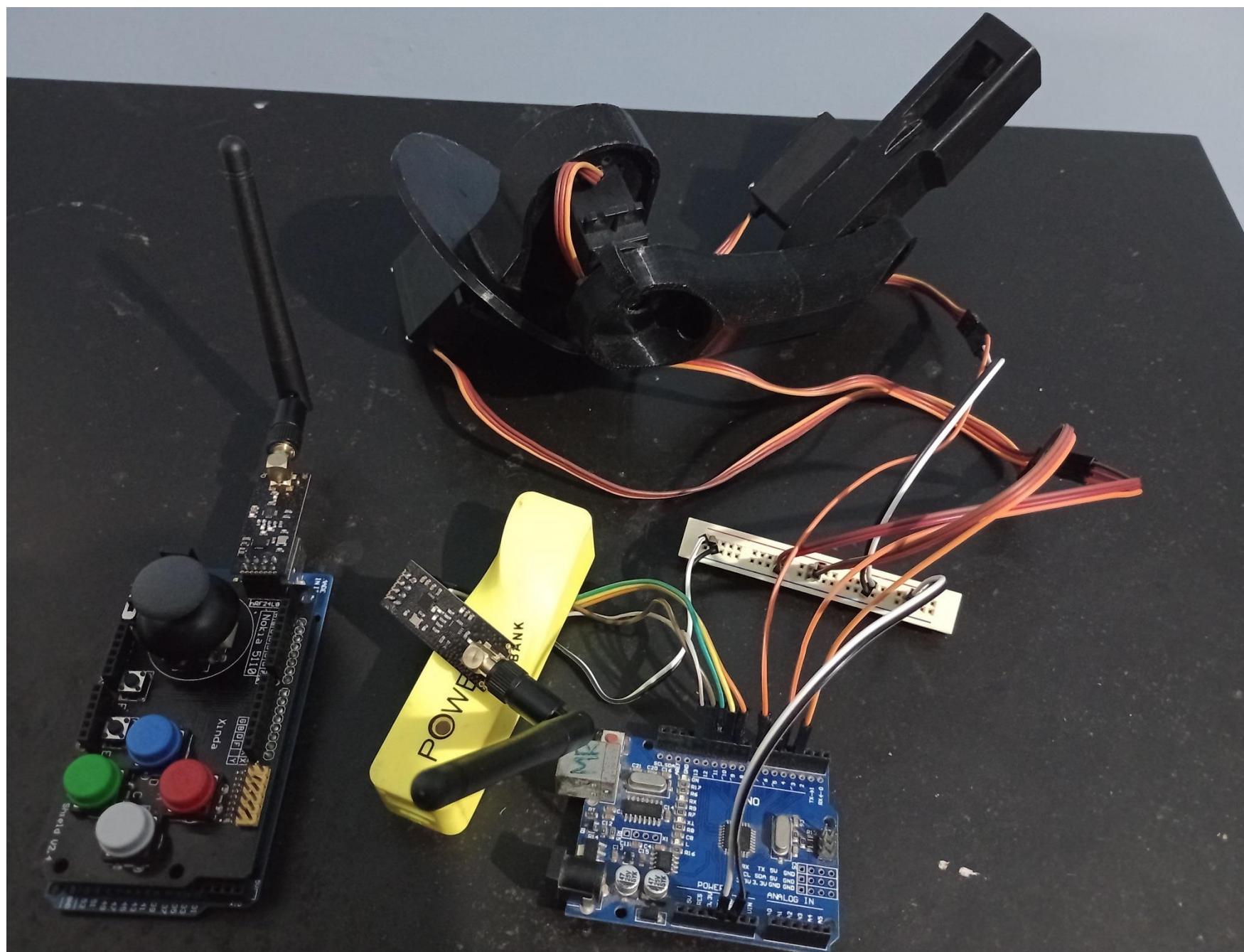


Figura 24. Conexiones en base a la figura 22. Se probó el movimiento remoto de los servos conectándolos a una base y eslabones de otro brazo robótico más pequeño.

11) Etapa de pruebas para la competencia.

Finalmente se evaluará la factibilidad del prototipo de brazo robótico construido y mostrado en las anteriores secciones, una vez ensamblado en el robot rescue se prevé que se puedan presentar varios de los siguientes posibles escenarios que el robot deberá superar para la acreditación de puntos en la competencia del TMR 2024.

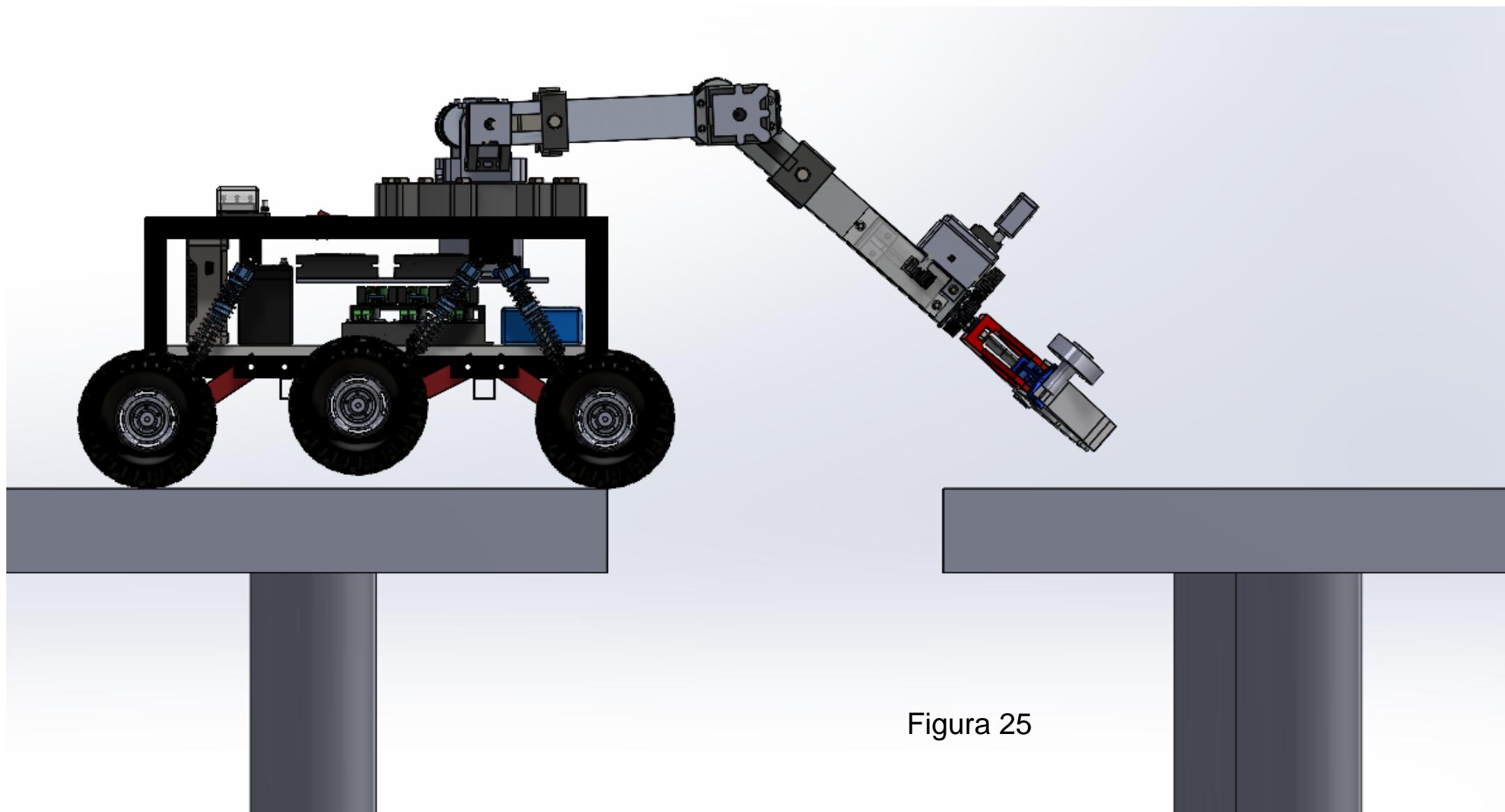


Figura 25

1) Camino al vacío. Uno de los obstáculos que la categoría proyecta será avanzar por un camino que conduce al vacío y para no caer se plantea la solución de usar el brazo robótico como punto de apoyo, de esta forma, logra el avance lento y seguro del robot rescue hasta el otro extremo, la ESP32 CAM asiste con la transmisión de video para saber el lugar donde sea seguro colocar el brazo, como se muestra en la figura 25.

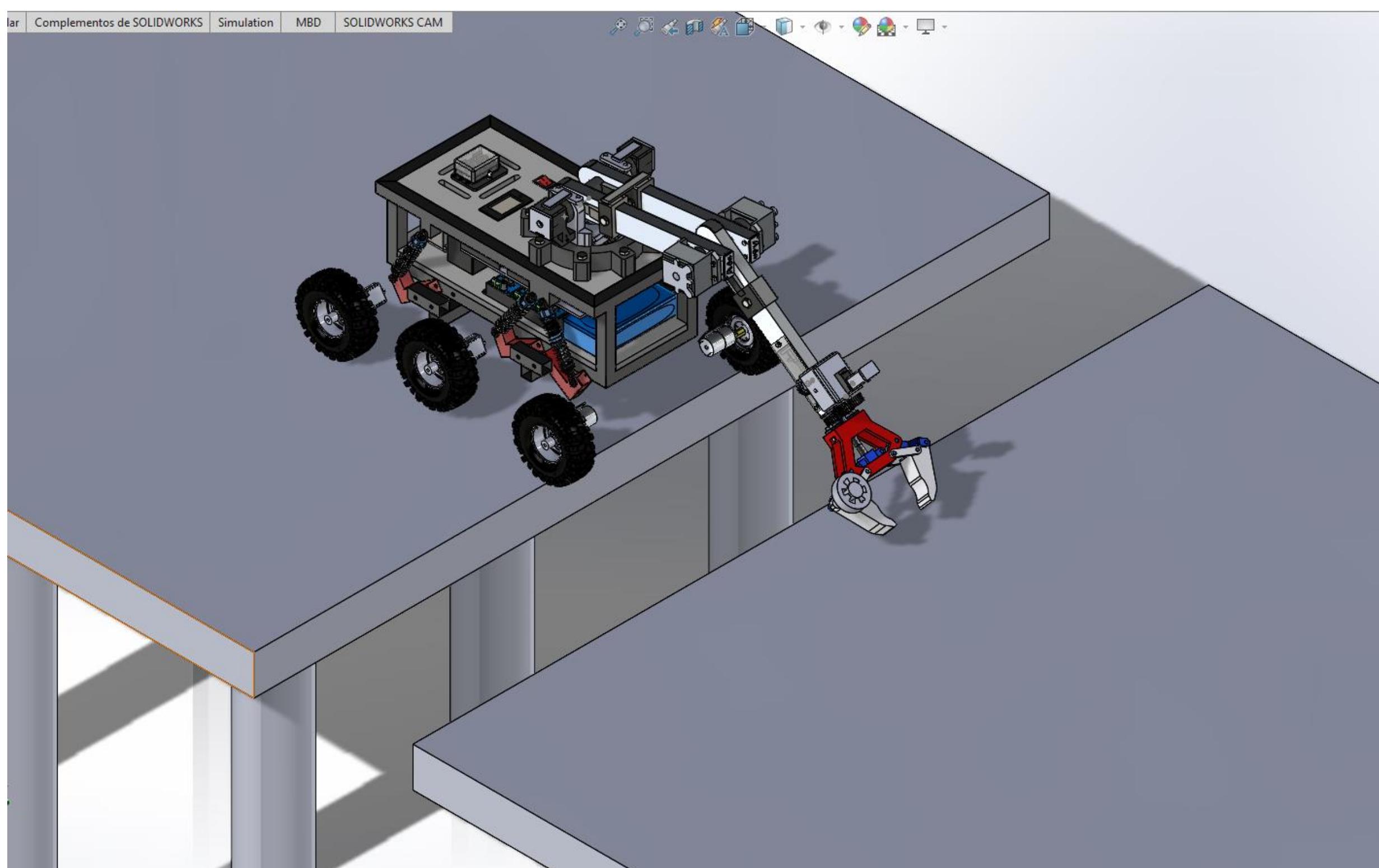


Figura 25.2. Vista isométrica del robot rescue atravesando la prueba de camino al vacío.

2) Toma de objetos remotamente / abertura de puertas. Otra de los posibles impedimentos que el robot de rescate enfrentará será abrir puertas por medio de la destreza mecánica que el control remoto de la garra y la transmisión de video deben secundar, con la conducción exacta en la posición de los servomotores y el monitoreo visual esta tarea podrá ser dominada a la perfección. Gracias al video transmitido en vivo al servidor es posible ver con exactitud que movimientos se deben producir.

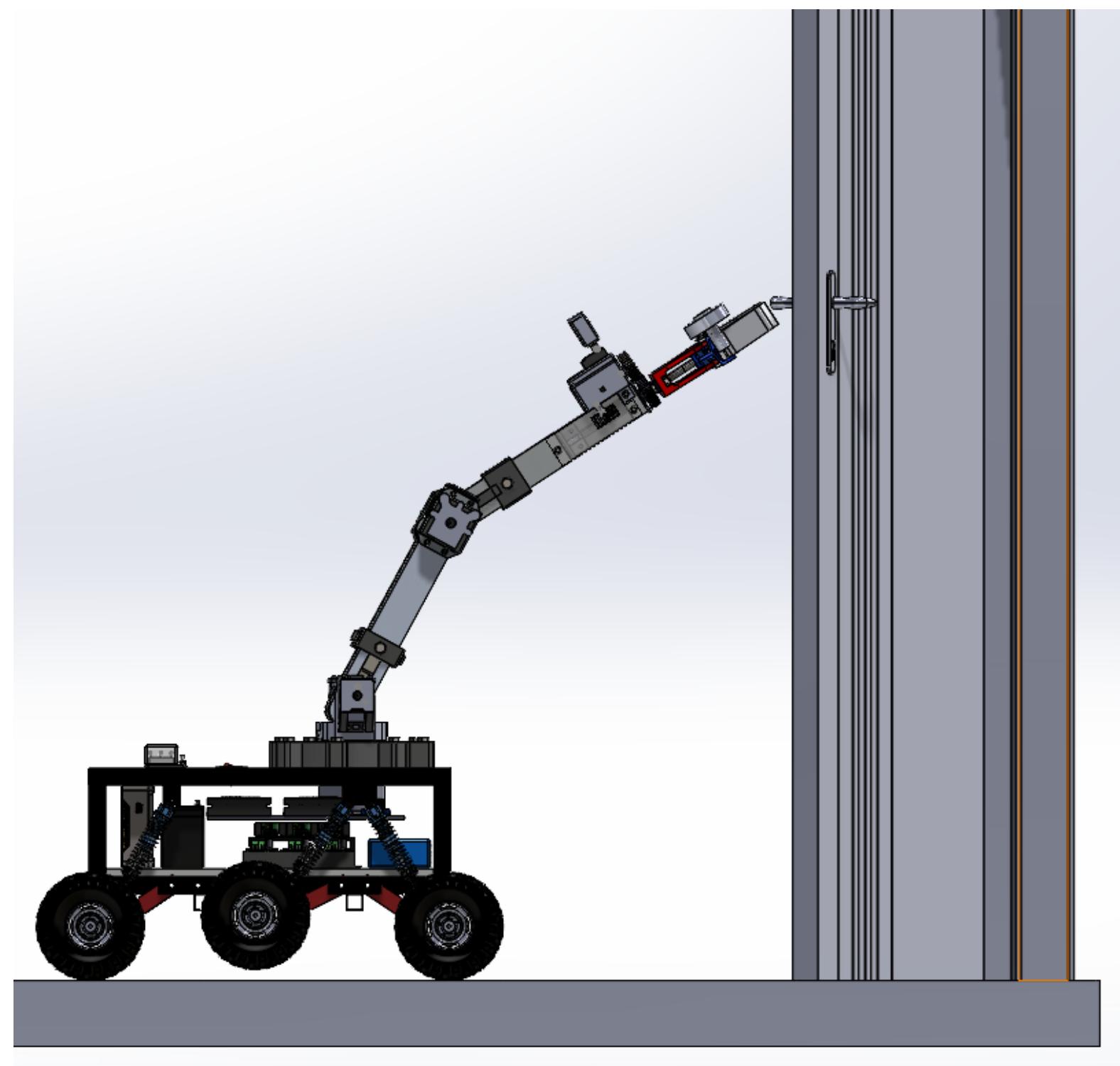


Figura 26. Vista isométrica del robot rescue y el brazo robótico en posición de tomar la manija, posteriormente girar la garra y ser capaz de abrir la puerta para el avance en su trayecto.

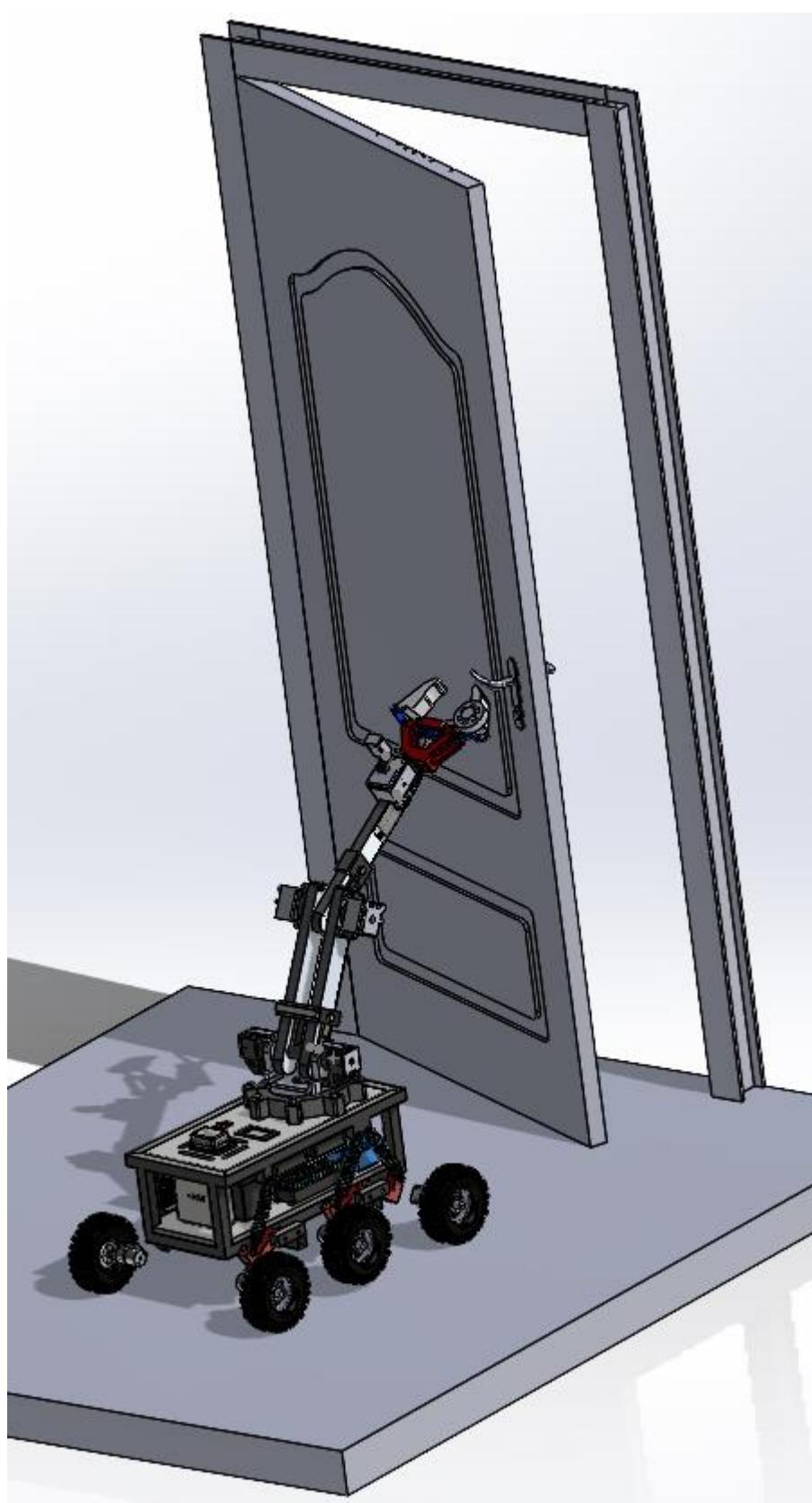


Figura 26.2. Vista isométrica trasera derecha del robot de rescate y la puerta abierta.

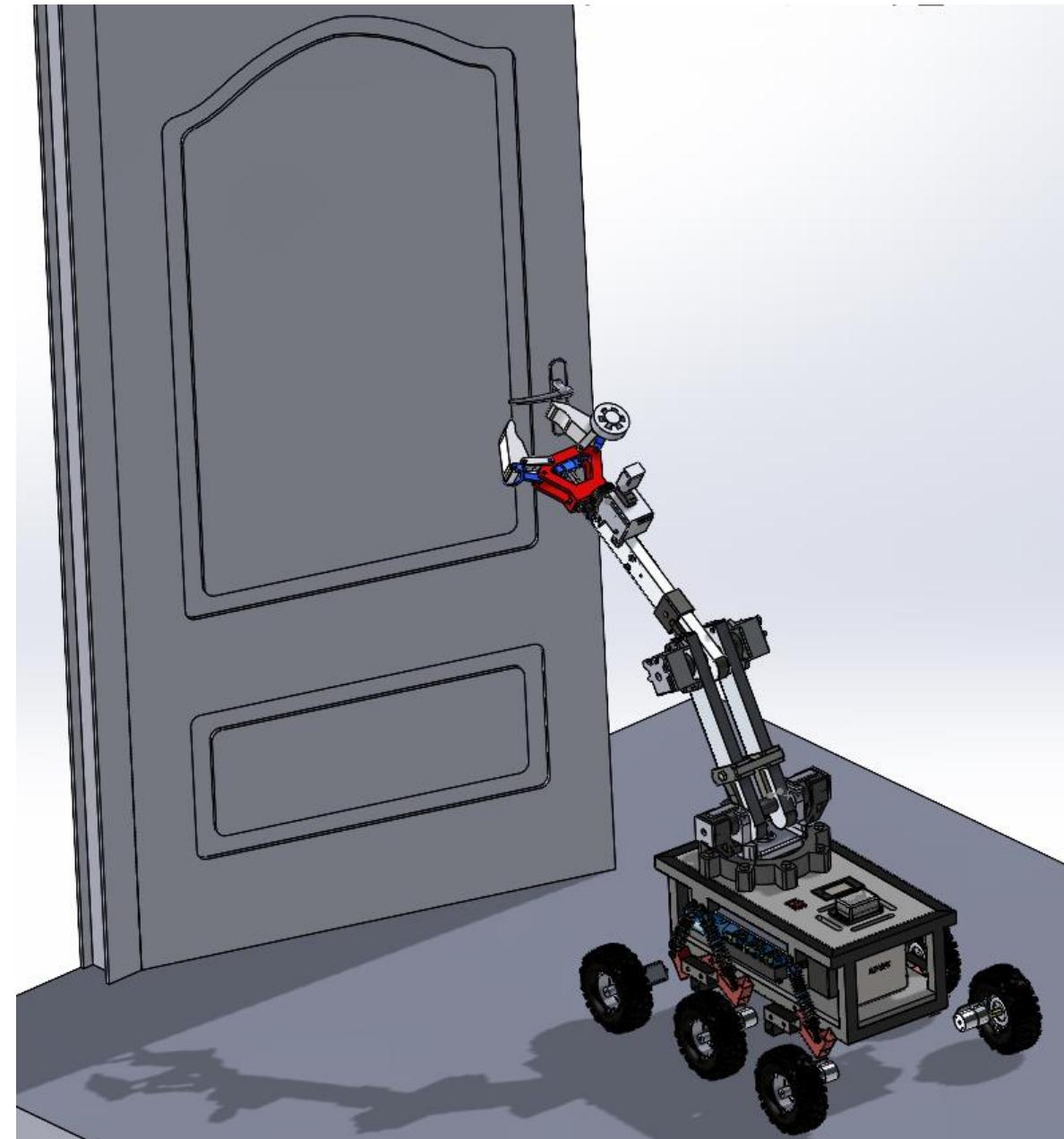


Figura 26.3. Vista isométrica trasera izquierda del robot de rescate y la puerta abierta.

VI. RESULTADOS

A pesar de contar con muy poco tiempo para las exigencias que el desarrollo del proyecto demanda se pudo crear un prototipo de brazo robótico teóricamente funcional para el robot rescue con el que la UPV podría participar en el TMR 2024, ya que como se demostró anteriormente este brazo robótico cumple con muchos de los puntos necesarios que la categoría podría presentar a sus concursantes, debido a que su diseño tiene una altura, peso, fuerza y destreza ideales para la gran variedad de diferentes tareas a realizar. Se creo un prototipo para todos los sistemas mecánicos, eléctricos y de control que con el avance del tiempo y dedicación en culminar y mejorar aún más por completo el proyecto se obtendrán un robot operativo que pueda superar cualquier obstáculo en su paso.

VII. CONCLUSIONES

El prototipo de brazo robótico parece ser perfectamente funcional en las simulaciones al rotar sus articulaciones como se señaló anteriormente, y los códigos creados en arduino permiten el desarrollo de un sistema de control remoto más accesible a los operadores. Sin embargo, al no contar con una impresora 3D que cumpla con las características de impresión para el filamento PETG, la continuidad y pruebas físicas se vieron afectadas ya que por el momento no se han podido imprimir las piezas mecánicas que componen el proyecto. Una vez superado este inconveniente se planea el ensamblaje de todas las piezas y componentes para realizar los ajustes necesarios en los próximos ensayos y simulaciones.

Durante el desarrollo de este proyecto se desarrollaron nuevas habilidades de programación sobre el lenguaje c++ en Arduino para el manejo de señales de radio, el diseño mecánico a través del software de SolidWorks, los conocimientos en sistemas electrónicos y de control fueron reforzados y un nuevo interés por el control autónomo de brazos robóticos ha comenzado. Próximamente se continuará trabajando en el proyecto con la meta de que pueda estar operativo a finales de año. Y perfectamente funcional, incluso automático para la fecha del TMR 2024. Nuevas propuestas en el prototipo serán necesarias con la evolución de este proyecto.

Este prototipo aportará al proyecto y equipo didáctico para prácticas de programación (ya que se pueden reprogramar sus componentes) en brazos robóticos, para los futuros nuevos estudiantes de mecatrónica en la Universidad Politécnica de Victoria.

VIII. BIBLIOGRAFÍA

- 1 SOLIDWORKS, “Que es y para que sirve”, noviembre 2023, URL: <https://solid.bi.es/solidworks/#:~:text=Definici%C3%B3n,proceso%20de%20desarrollo%20del%20producto>.
- 2 Wikibooks, “Robótica/Puente H”, 2021, URL: https://es.wikibooks.org/wiki/Rob%C3%A9tica/Puente_H
- 3 Makerguides, “TB6600 Stepper Motor Driver User Guide”, URL: <https://www.makerguides.com/wp-content/uploads/2019/10/TB6600-Manual.pdf>
- 4 datasheet, “PDF 17HS4401 Data sheet (Hoja de datos)”, URL: <http://www.datasheet.es/PDF/928661/17HS4401-pdf.html>
- 5 bolanosdj, “¿Qué es y cómo funciona un servomotor?”, URL: <https://www.bolanosdj.com.ar/MOVIL/ARDUINO2/IntroServos.pdf>
- 6 keyestudio, “Ks0153 keyestudio Joystick Shield”, URL: https://wiki.keyestudio.com/Ks0153_keyestudio_JoyStick_Shield
- 7 ProductSpec Duchess, “nRF24L01 Preliminary Product Specification”, URL: https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf
- 8 Arduino, “¿Qué es Arduino?”, URL: <https://arduino.cl/que-es-arduino/>
- 9 Ai-Thinker, “ESP32-CAM Module”, URL: <https://loboris.eu/ESP32/ESP32-CAM%20Product%20Specification.pdf>
- 10 DassaultSystemes, “¿Qué es la impresión 3D?”, URL: <https://www.3ds.com/es/make/guide/process/3d-printing>
- 11 tractus3D, “Filamento de PETG”, URL: <https://tractus3d.com/es/materials/petg/>