



# **Universidad Tecnológica de la Mixteca**

## **Robótica de Manipuladores**

Dr. Carlos García Rodríguez

## **PROYECTO**

“Restauración del robot manipulador SCARA”

## **Ingeniería en Mecatrónica**

**Grupo:**

814-B

**Semestre 2023-B**

Huajuapan de León, Oaxaca. 03 de julio de 2023.

## **INDICE**

|     |                          |
|-----|--------------------------|
| 1.  | RESUMEN.....             |
| 2.  | INTRODUCCIÓN.....        |
| 3.  | DESARROLLO GENERAL.....  |
| 4.  | DESARROLLO POR ÁREA..... |
| 4.1 | ÁREA MECANICA.....       |
| 4.2 | ÁREA ELECTRONICA.....    |
| 4.3 | ÁREA AUXILIAR.....       |
| 4.4 | ÁREA DE GRIPPER.....     |
| 4.5 | ÁREA INTERFAZ.....       |
| 5.  | CONCLUSIONES.....        |
| 6.  | BIBLIOGRAFIA.....        |
| 7.  | ANEXOS.....              |

## **1. RESUMEN**

Este informe detalla el proceso de restauración de un robot SCARA (Selectively Compliant Articulated Robot Arm), cuyo estado original presentaba desperfectos y deficiencias en su funcionamiento. A lo largo del informe, se describirán las etapas del proyecto de restauración, los desafíos enfrentados y los resultados obtenidos.

## **2. INTRODUCCIÓN**

El robot SCARA (Selectively Compliant Articulated Robot Arm) es un tipo de robot manipulador ampliamente utilizado en la industria para diversas aplicaciones. Su diseño se caracteriza por tener articulaciones en forma de revolución y una estructura rígida en los ejes vertical y horizontal. Estas características le permiten realizar movimientos rápidos y precisos en un plano horizontal, mientras que su estructura rígida proporciona estabilidad y rigidez.

Las principales características y ventajas de los robots SCARA son las siguientes:

**Movimiento en un plano horizontal:** Los robots SCARA están diseñados para moverse principalmente en un plano horizontal, lo que los hace ideales para tareas de ensamblaje, manipulación de materiales y procesos de pick-and-place.

**Velocidad y precisión:** Gracias a su diseño y estructura rígida, los robots SCARA pueden lograr movimientos rápidos y precisos, lo que los hace eficientes para tareas que requieren alta velocidad y repetibilidad.

**Compactos:** Los robots SCARA suelen tener un diseño compacto, lo que los hace ideales para espacios reducidos en líneas de producción y laboratorios.

**Carga útil:** Los robots SCARA pueden manejar cargas útiles moderadas a pesadas, lo que los hace adecuados para tareas que involucran piezas o componentes de cierto peso.

**Configuración y programación sencilla:** Los robots SCARA son relativamente fáciles de configurar y programar, lo que los hace accesibles para usuarios con diferentes niveles de experiencia en robótica.

Las aplicaciones comunes de los robots SCARA incluyen ensamblaje de componentes electrónicos, manipulación de productos en líneas de producción, empaquetado, selección y colocación de objetos, entre otras tareas que requieren movimientos rápidos y precisos en un plano horizontal.

### **3. DESARROLLO GENERAL**

Evaluación del Estado Inicial: Se realizó una inspección detallada del robot SCARA para identificar los problemas y fallos en su estructura mecánica, sensores, actuadores y controladores.

Desmontaje y Limpieza: Se desmontaron todas las partes del robot, se limpiaron minuciosamente y se reemplazaron las piezas desgastadas o dañadas.

Reparación Mecánica: Se llevaron a cabo ajustes en los ejes, articulaciones y conexiones para garantizar el correcto movimiento del robot y evitar holguras indeseadas.

Reemplazo de Componentes Electrónicos: Se identificaron y sustituyeron los componentes electrónicos defectuosos, como placas de control, sensores y cables.

Actualización de Software: Se revisó y actualizó el software de control del robot para corregir errores y mejorar su desempeño.

Calibración y Pruebas: Una vez restaurado el robot, se procedió a su calibración y se realizaron diversas pruebas para verificar su funcionamiento en diferentes escenarios.

### **4. DESARROLLO POR AREA**

#### **4. 1 Área Mecánica**

Para empezar, se sometió el manipulador a una serie de acciones para poder observar las fallas en su funcionamiento y de esa manera poder corregirlas.

Diseño y elaboración de una nueva base.

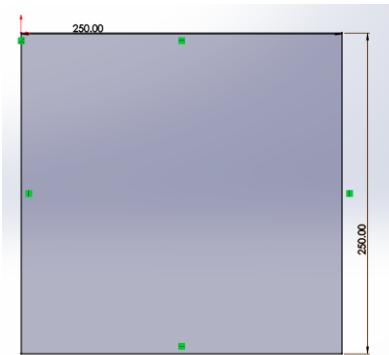
Se identificó que el manipulador presentaba problemas de desestabilización en diferentes orientaciones, especialmente cuando se encontraba en una posición de brazo extendido. Para abordar esta situación, se llevó a cabo el diseño y elaboración de una nueva base para el manipulador. La nueva base debe proporcionar una mayor estabilidad y soporte, permitiendo un rendimiento mejorado durante las operaciones del manipulador en diferentes posiciones y configuraciones.

Para poder solucionar este problema se procedió a analizar la base del manipulador y realizar mediciones exactas para el diseño de diferentes bases que fueran efectivas.

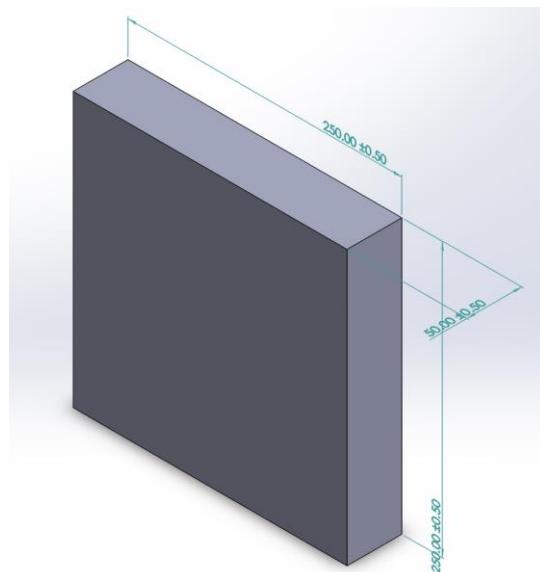
Después de tomar las medidas exactas y correctas se procedió a desarrollar la principal solución la cual consiste en:

1. Diseñar una base 25x25x5cm resistente para que el manipulador no tenga ningún problema con el peso y con diferentes configuraciones.
2. Diseñar zetas para poder asegurar la base del manipulador con la base nueva que se diseñó.
3. Hacer uso de tornillos y tuercas para acoplar la base del manipulador con la que se propuso.

Para el primer punto se desarrolló una base sencilla con las medidas dadas anteriormente, esta base debería cumplir con ser lo suficientemente resistente y a su vez no debía ser tan grande para poder trasladar el manipulador sin dificultad alguna.



*Figura 4.1.1. Diseño de la base principal del manipulador SCARA 2D.*



*Figura 4.1.2. Diseño de la base del manipulador SCARA 3D.*

Después de generar el diseño apropiado se procedió a elegir el material por el que sería construida la base, para ello hicimos uso del cemento, debido a su resistencia, peso y a su facilidad de elaboración.



Figura 4.1.3. Base del manipulador SCARA.

Para el segundo punto de la solución, nuevamente se realizaron toma de medidas en la base, esta vez para poder diseñar las zetas y la perforación para los tornillos donde se colocarían las mismas.

Se diseñaron las siguientes medidas para dos zetas diferentes:

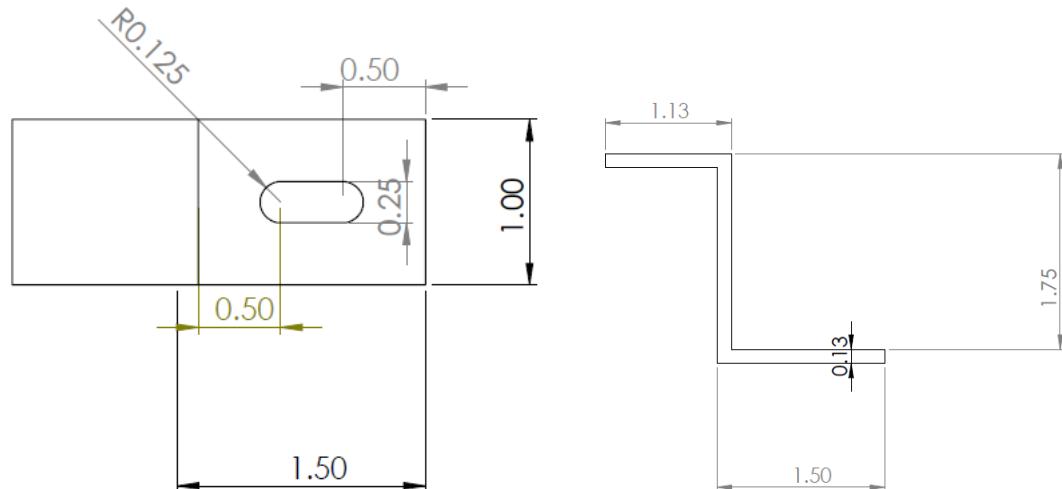
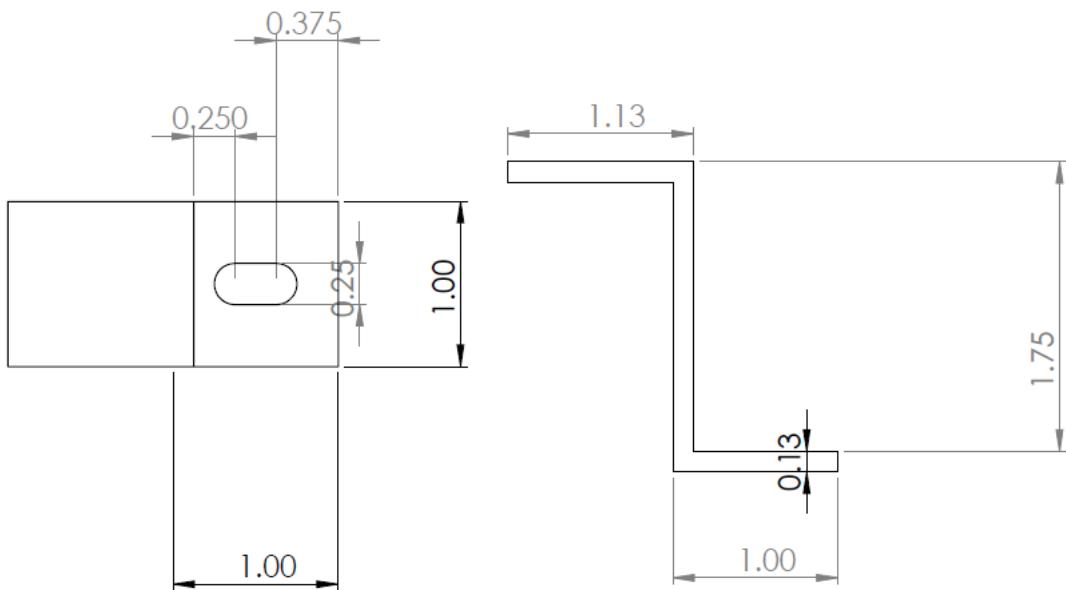
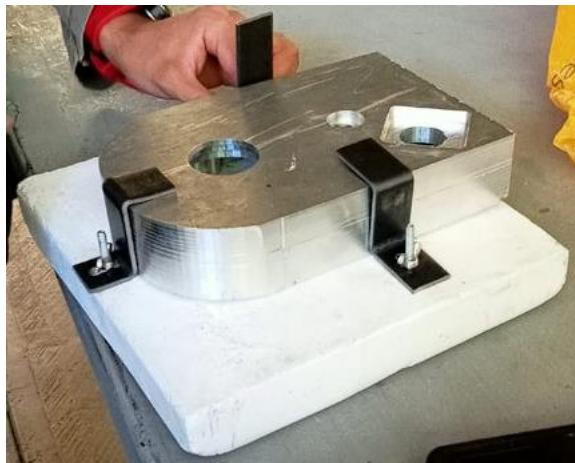


Figura 4.1.4. Diseño y medida de la zeta 1.



*Figura 4.1.5. Diseño y medida de la zeta 2.*

Finalmente, para la manufacturación de cada zeta se optó por una solera de  $\frac{1}{4}$  y tornillos de  $\frac{3}{16}$  con sus respectivas mariposas para así poder acoplar todo.



*Figura 4.1.6 Ensamble de la base del manipulador SCARA con las zetas.*

Después de verificar que esta alternativa resolvía el problema se procedió con la siguiente falla que se identificó.

- Desarrollo de una nueva pieza con contra peso.

Otro problema detectado en el manipulador fue la dificultad para controlar el movimiento ascendente y descendente debido a problemas con los rodamientos en la pieza existente. Para

solucionar este inconveniente, se desarrolló una nueva pieza con un contrapeso diseñado para equilibrar el manipulador y permitir un movimiento suave y controlado en el eje vertical. Se utilizó filamento especializado para la impresión 3D de la pieza y se realizaron los ajustes necesarios para ensamblar los nuevos rodamientos y garantizar su correcto funcionamiento.

Para el diseño de la pieza se optó por el mismo diseño del manipulador original, exceptuando el cambio en las medidas de los rodamientos y agregando la zona donde iría el contrapeso de la pieza.

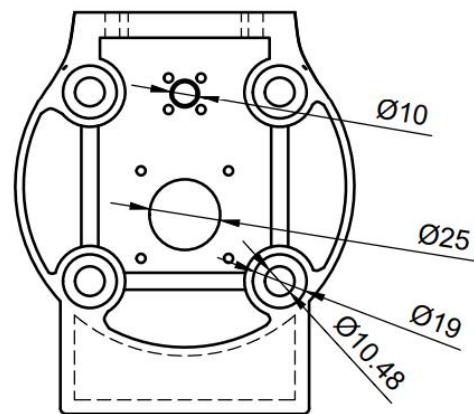


Figura 4.1.7. Diseño de la pieza con contrapeso (vista alzada).

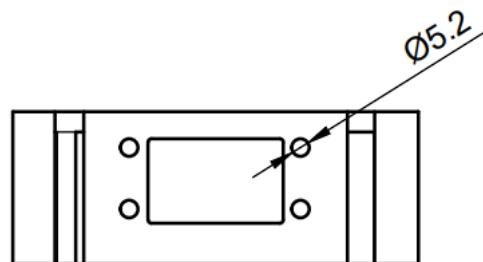


Figura 4.1.8. Diseño de la pieza con contrapeso (vista lateral).

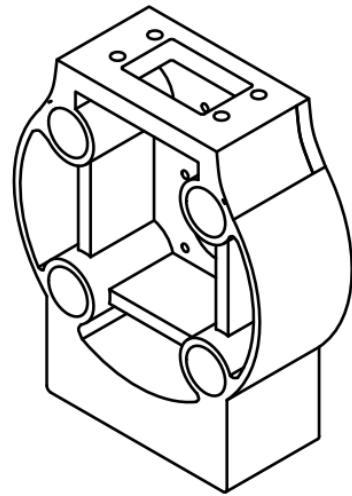


Figura 4.1.9. Diseño de la pieza con contra peso (vista isométrica).



Figura 4.1.10. Pieza con contra peso (impresión 3D).

- Construcción de un soporte para la distancia de los ejes.

Se identificaron fallos en la distancia entre los ejes verticales, lo que provocaba el calentamiento de los motores y dificultaba el movimiento ascendente y descendente del manipulador.

Para mantener una distancia igual entre los ejes verticales y asegurar un movimiento uniforme, se construyó un soporte para los ejes que se colocó estratégicamente en el manipulador. Esta araña permitió mantener la alineación adecuada de los ejes y minimizar las posibles desviaciones durante el funcionamiento del manipulador. La construcción de la araña se realizó utilizando materiales resistentes y duraderos para garantizar su estabilidad y precisión.

Con ayuda de las medidas previas de la pieza del contrapeso, se desarrolló el diseño del soporte para los ejes.

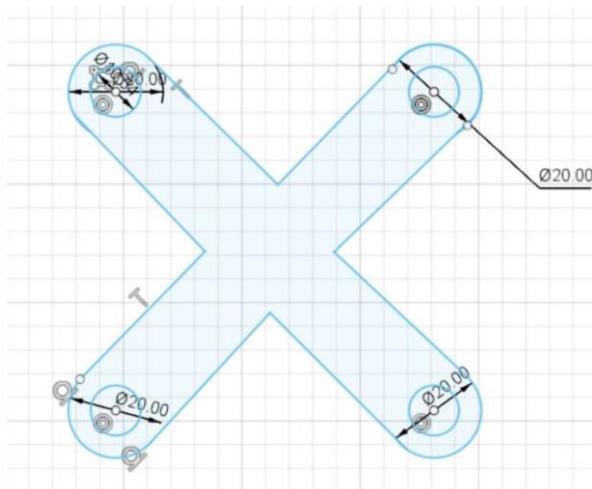


Figura 4.1.11. Soporte para los ejes (vista alzada).

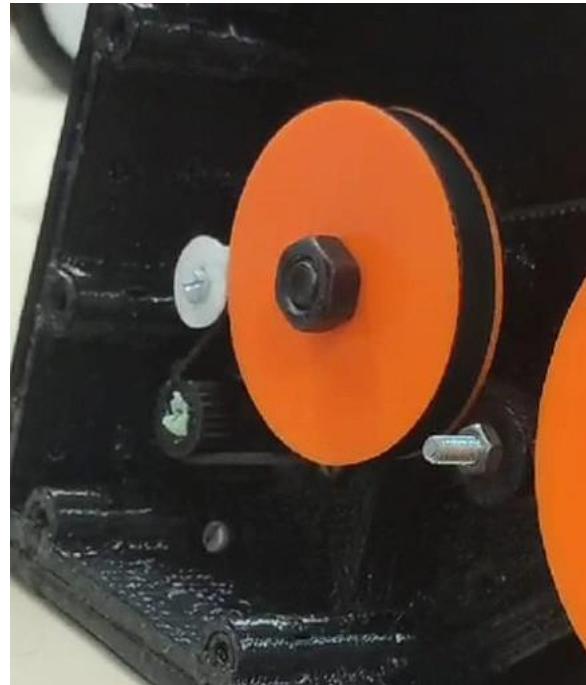


Figura 4.1.12. Soporte para los ejes impresión 3D.

- Reajuste y tensionamiento de bandas.

Se realizó un desmontaje completo del manipulador para poder analizar profundamente el funcionamiento del manipulador, se identificó que las bandas del manipulador se encontraban sueltas y no ejercían la tensión adecuada para que el manipulador pudiera efectuar diferentes

configuraciones, por lo que se requirió reajustar y tensionar las bandas. Las bandas desgastadas o mal ajustadas pueden afectar el rendimiento del manipulador y su precisión. Mediante el reajuste y el adecuado tensionamiento de las bandas, se logró optimizar el movimiento y la respuesta del manipulador.



*Figura 4.1.13. Desmontaje y reajuste de las bandas.*

- Cambio de tuercas y tornillos.

Como parte del proceso de mejora y mantenimiento, se llevaron a cabo cambios de tuercas y se procedió a apretar los tornillos en todas las conexiones del manipulador. Esto garantizó una sujeción adecuada y evitó movimientos indeseados o inestabilidad causada por conexiones flojas.

## 4.2 Área Electrónica

La construcción de un robot SCARA implica una cuidadosa planificación, diseño y desarrollo de la estructura mecánica, así como la implementación de la electrónica necesaria para su funcionamiento. La electrónica juega un papel crucial en el control y la coordinación de los movimientos del robot, la comunicación con otros dispositivos y la recopilación de datos a través de sensores. La electrónica del robot SCARA abarca componentes como controladores, motores, y sensores, que trabajan en conjunto para garantizar un rendimiento preciso y confiable. Estos componentes son diseñados y configurados de acuerdo con los requisitos específicos de la aplicación. La integración exitosa de la electrónica en el robot SCARA permite un control preciso de los movimientos y acciones del robot, lo que resulta en una mayor eficiencia, calidad y seguridad en los procesos industriales. Además, el uso de sensores electrónicos permite la detección y respuesta a condiciones variables del entorno, mejorando aún más la precisión y el rendimiento general del robot.

Una de las ventajas que se tuvieron es que no se necesitó empezar desde cero con este robot pues ya se encontraba construido, lo que se pretendía no solo en esta área es mejorar y corregir errores en el funcionamiento del robot, además de agregar componentes necesarios para que funcionara. El diagrama de conexiones es el siguiente:

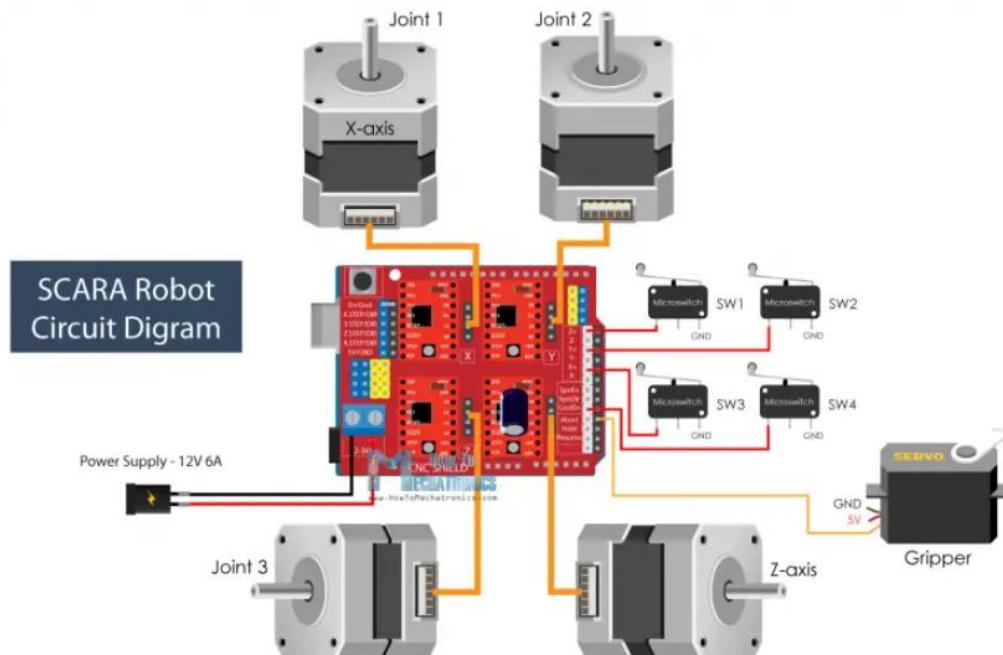


Figura 4.2.1 Diagrama de conexiones

Por la parte de electrónica la primera vez que se tuvo contacto con el robot para tratar de moverlo no se contaba con la interfaz necesaria para hacer este proceso es por eso que se necesito contactar con los compañeros que elaboraron el robot y así poder comunicarnos mediante el protocolo de comunicación UART para poder mandar las posiciones al robot y que este se moviera, como se observa en la figura 1, la interfaz tiene diferentes características como el controlar las articulaciones de manera independiente, calcular la cinemática inversa así como al posibilidad abrir y cerrar su gripper, se puede controlar la velocidad. Con el botón de guardar se guardan la posición dada y así con múltiples posiciones, una vez que se movió el robot en ciertas posiciones se puede oprimir en seguimiento de trayectoria con lo que seguirá todos los

puntos guardados. De esta manera se estuvieron probando las articulaciones, midiendo los voltajes y corrientes para verificar que todo estuviera dentro de un rango tolerable para los componentes.

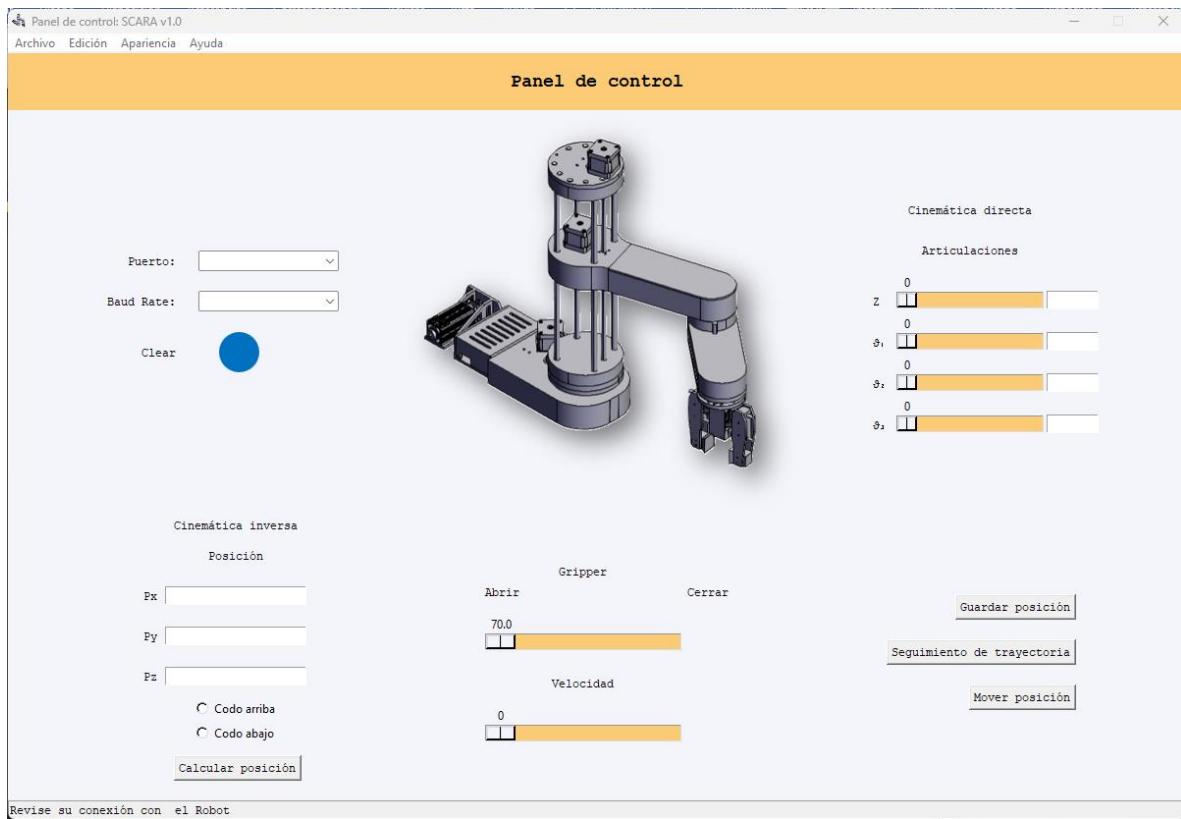


Figura 4.2.2. Panel de control

Para poder lograr hacer esto se tuvo que conseguir una fuente con la capacidad necesaria, ya que el robot no contaba con una, investigando se determinó que los motores que ocupa, es decir los motores a paso 17HS8401 (Nema 17) consumían como máximo una cantidad de 1.8 A y voltaje entre 12V a 24V DC, en total se cuenta con 4 motores y debido a la carga que se en el eje al moverse tendrían un consumo considerable, sumado a esto tenemos los drivers A4988 con una corriente continua de operación de 1.5 A, es decir que en total necesitaríamos mínimamente un voltaje de 12 V y 12 A. El amperaje era lo que limitaba a moverlo con las fuentes de alimentación de la universidad pues si máximo era de 8A, es por eso que se procedió a comprar una fuente de alimentación de una computadora la cual en sus salidas nos proporciona la alimentación necesaria además de ser una solución práctica y barata de conseguir.

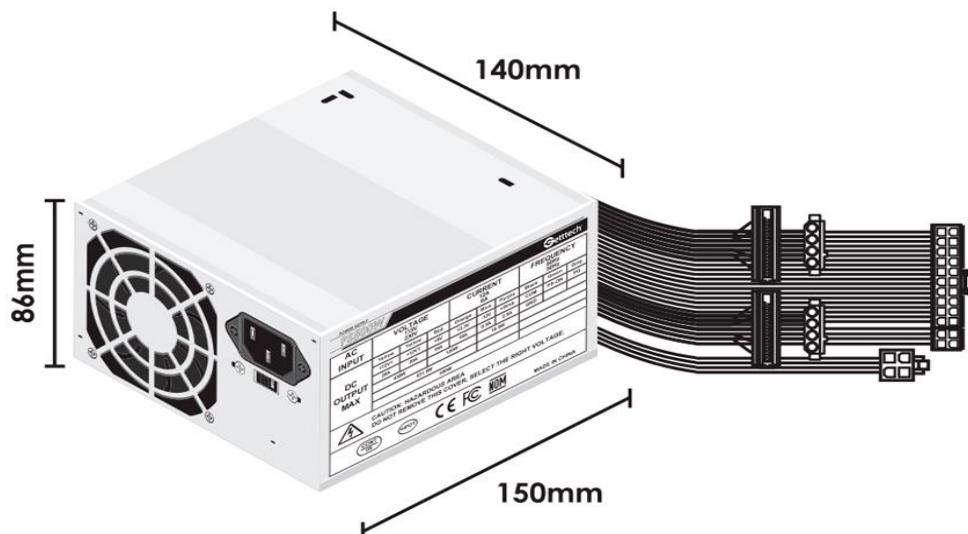


Figura 4.2.3. Fuente de alimentación

| POWER SUPPLY<br><b>PS500W</b> |                         |        |      |                      |       |        |                           |       |      | Getttech® |  |  |  |  |  |  |  |
|-------------------------------|-------------------------|--------|------|----------------------|-------|--------|---------------------------|-------|------|-----------|--|--|--|--|--|--|--|
| AC<br>INPUT                   | VOLTAGE<br>115V<br>230V |        |      | CURRENT<br>12A<br>6A |       |        | FREQUENCY<br>60Hz<br>50Hz |       |      |           |  |  |  |  |  |  |  |
|                               | Yellow                  | Yellow | Red  | Orange               | Blue  | Purple | Black                     | Green | Gray |           |  |  |  |  |  |  |  |
|                               | +12V1                   | +12V2  | +5V  | +3.3V                | -12V  | +5Vsb  | COM                       | PS-ON | PG   |           |  |  |  |  |  |  |  |
|                               | 20A                     | 20A    | 19A  | 19A                  | 0.5A  | 2.5A   | GND                       |       |      |           |  |  |  |  |  |  |  |
|                               | 430W                    |        | 120W |                      | 18.5W |        |                           |       |      |           |  |  |  |  |  |  |  |
|                               | 531.5W                  |        |      |                      |       |        |                           |       |      |           |  |  |  |  |  |  |  |
|                               | 550W                    |        |      |                      |       |        |                           |       |      |           |  |  |  |  |  |  |  |

Figura 4.2.4. Datos de la fuente

No se usaron todas las salidas, solo un par de cables correspondientes a la VCC que como lo indica la tabla de voltajes es el amarillo y el negro que es de GND, este se conectó a la shield existente además de agregar un interruptor para controlar el paso de la energía hacia la placaCNC y el encendido y apagado de la fuente.

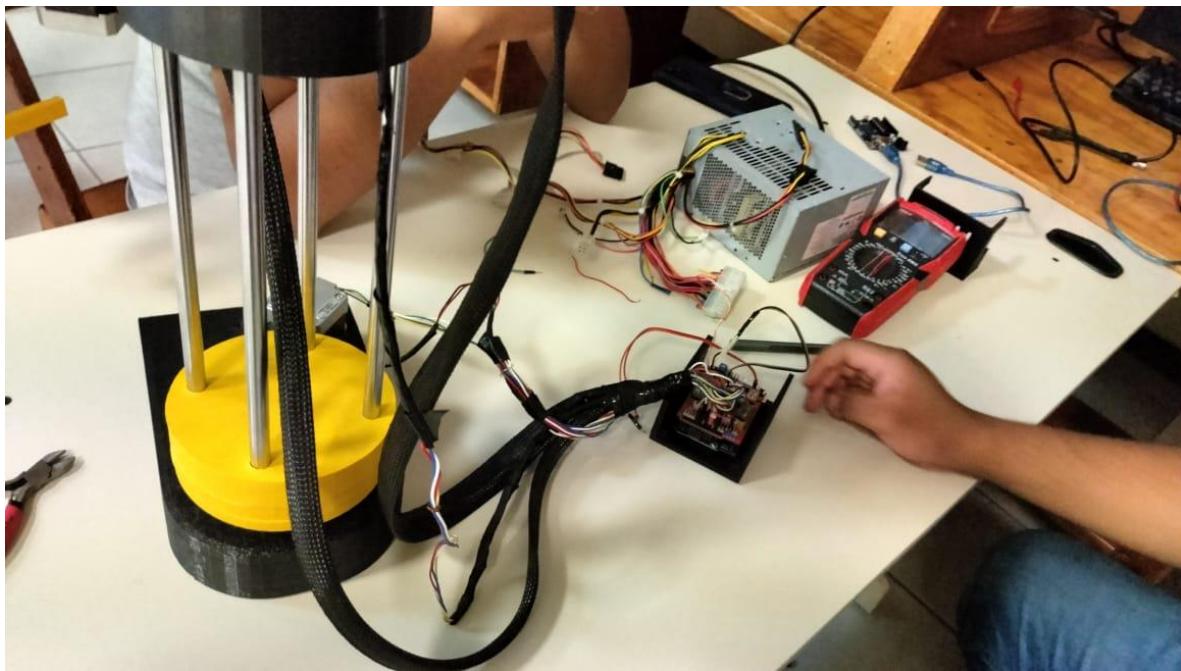


Figura 4.2.5. Conexión de la fuente

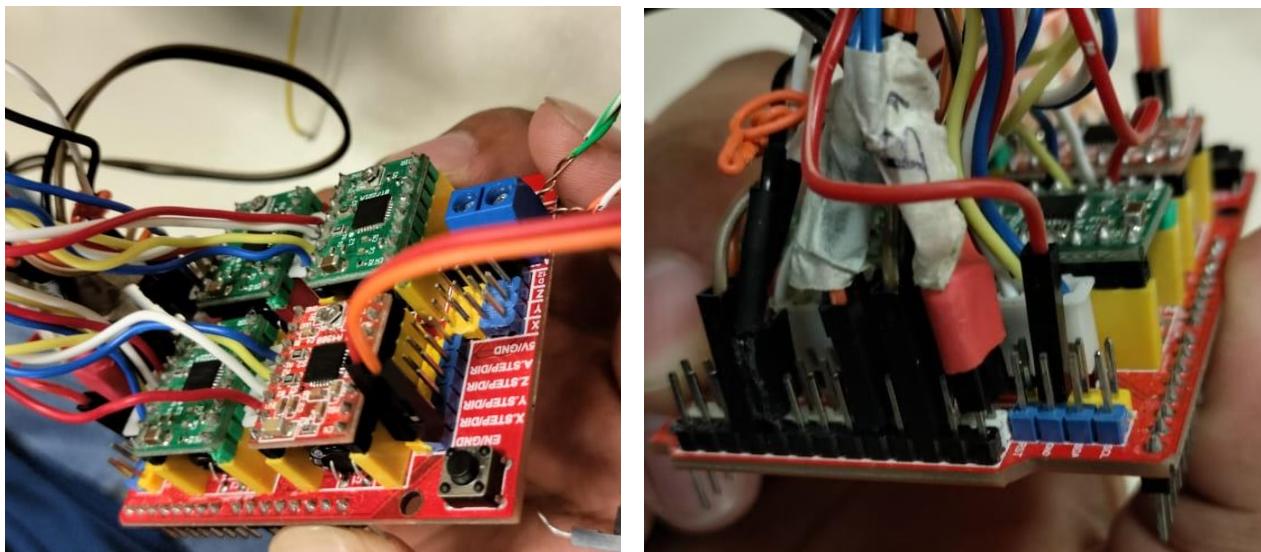


Figura 4.2.6. Conexiones no ideales

Tambien se revisaron conexiones en la placa CNC que va sobre el Arduino uno, pues había pines que estaban flojos y se salían con el mínimo movimiento, además de conectar otros que hacían falta como el diagrama de la parte inicial.

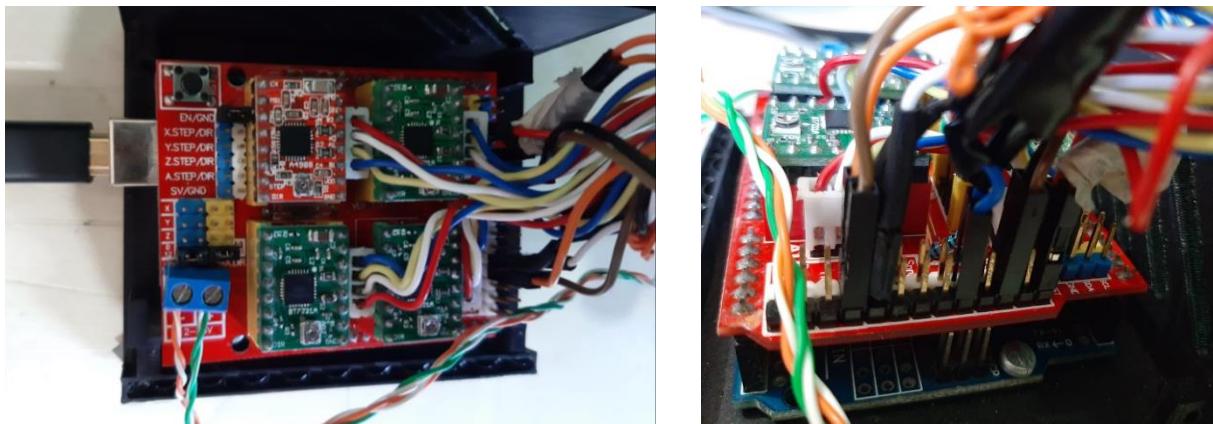


Figura 4.2.7 Conexiones arregladas



Figura 4.2.8 Prueba inicial

Una vez conseguido esto, se pudo hacer una primera prueba del robot echándolo a andar con su interfaz y cargando el programa correspondiente con Arduino. Tambien estuvieron trabajando cada área de manera simultánea, para resolver los distintos problemas que se presentaron. Cuando se movió, como se ve en la imagen de abajo el gripper no estaba funcional, por lo que esa área estuvo trabajando en solucionarlo, además los de mecánica revisarían la parte estructural ya que al moverlo se podía sentir un calentamiento considerable en los motores

a paso, es decir que en un corto periodo de tiempo estos se calentaban por lo que se tenía que dejar enfriar para poder seguir utilizando.

Con esto se pudo observar que el calentamiento en los motores era por un alto consumo de corriente para poder mover las articulaciones ya que algunas estaban pandeadas lo que forzaba al motor para moverse, es por eso que una vez se solucionó ese problema mecánico el desplazamiento en el eje Z el cual era el más preocupante se logró reducir con lo que se redujo el calentamiento en los motores y dando un rango de tiempo más amplio para esperar a que los motores se enfrien para evitar fallos. Otro problema que se detectó fue que los motores que hacen girar el gripper, es decir  $\theta_1$  y la parte de la base  $\theta_3$  no transmitían el giro de los motores, de esto se encargó el equipo de mecánica de revisar la transmisión de las poleas y bandas para que pudiera girar.

Otro detalle fue el gripper el cual se acciona mediante un servo, este estaba roto por lo que el equipo de esa área se encargó de colocar un nuevo gripper. Este detalle sumado al del motor de la base tuvo complicaciones, las cuales se explicarán a continuación.

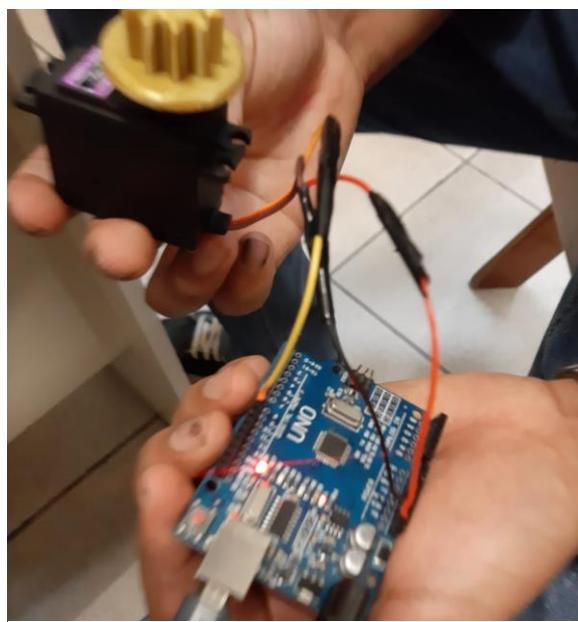


Figura 4.2.9 Prueba de servo

Empezando con el gripper, el servo que controla su apertura el equipo encargado de esta área no conectó de manera correcta los cables del servo, además aparentemente la parte estructural del gripper estaba rota, sin embargo al reemplazarla se encontró otro problema debido a que el servomotor que lo acciona tenía problemas, para determinar esto se probó independientemente con un programa sencillo en Arduino, por lo que se revisaron las conexiones y se procedió a cambiar en nuevo servo, haciendo que el gripper ahora si pudiera abrir y cerrar.

El siguiente problema fue con el motor de la base o  $\theta_3$  ya que el eje del motor se había enchuecado se cambió este por otro motor a pasos, pero al conectarse había problemas pues el modelo era diferente a los demás utilizados, por lo que se tuvo que buscar e intercambiar cables para lograr la conexión correcta. Además, el conector tenía los cables flojos, estos no se quedaban fijo en el conector y se salían a cada rato por lo que se poncharon de nuevo los conectores y de esta manera el problema quedó solucionado. Además, se cambió el final de carrera pues debido al giro, el mecanismo que lo acciona se dobló y quedó inservible lo que impedía la detección para llevar al robot a la posición inicial.

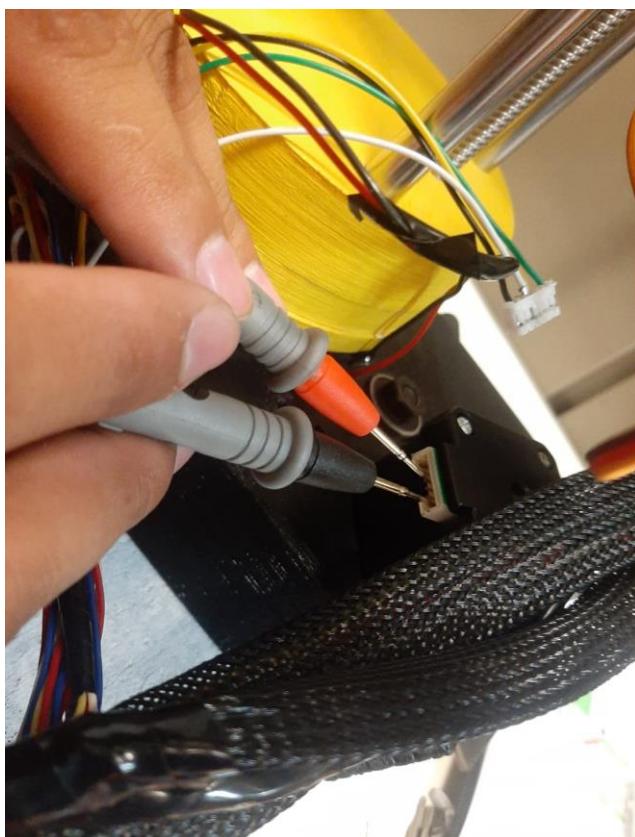


Figura 4.2.10 Verificación de las bobinas del motor

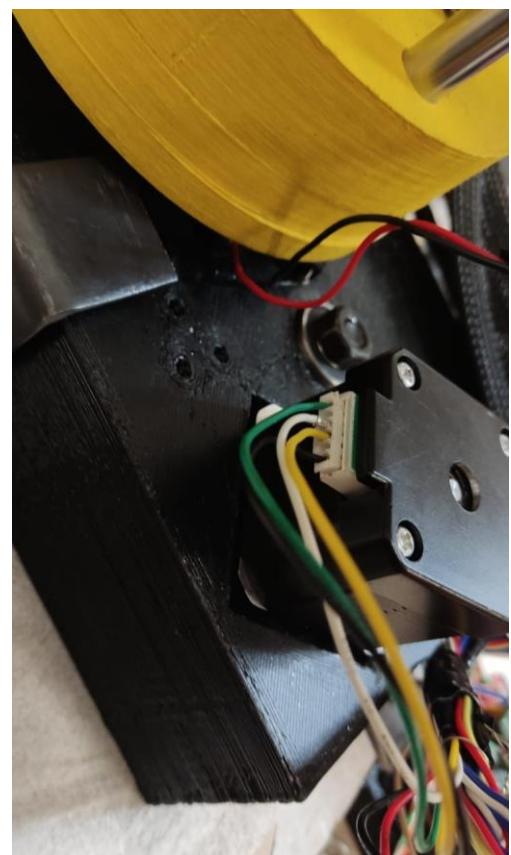


Figura 4.2.11 Conexiones reparadas

Una vez solucionados estos problemas se volvió a probar el robot con ayuda de la interfaz, con esto se pudo observar un cambio significativo en el robot, ya que este se veía más ágil y rápido para desplazar los eslabones, pues la alimentación era la correcta y se redujo el esfuerzo necesario para mover los eslabones.

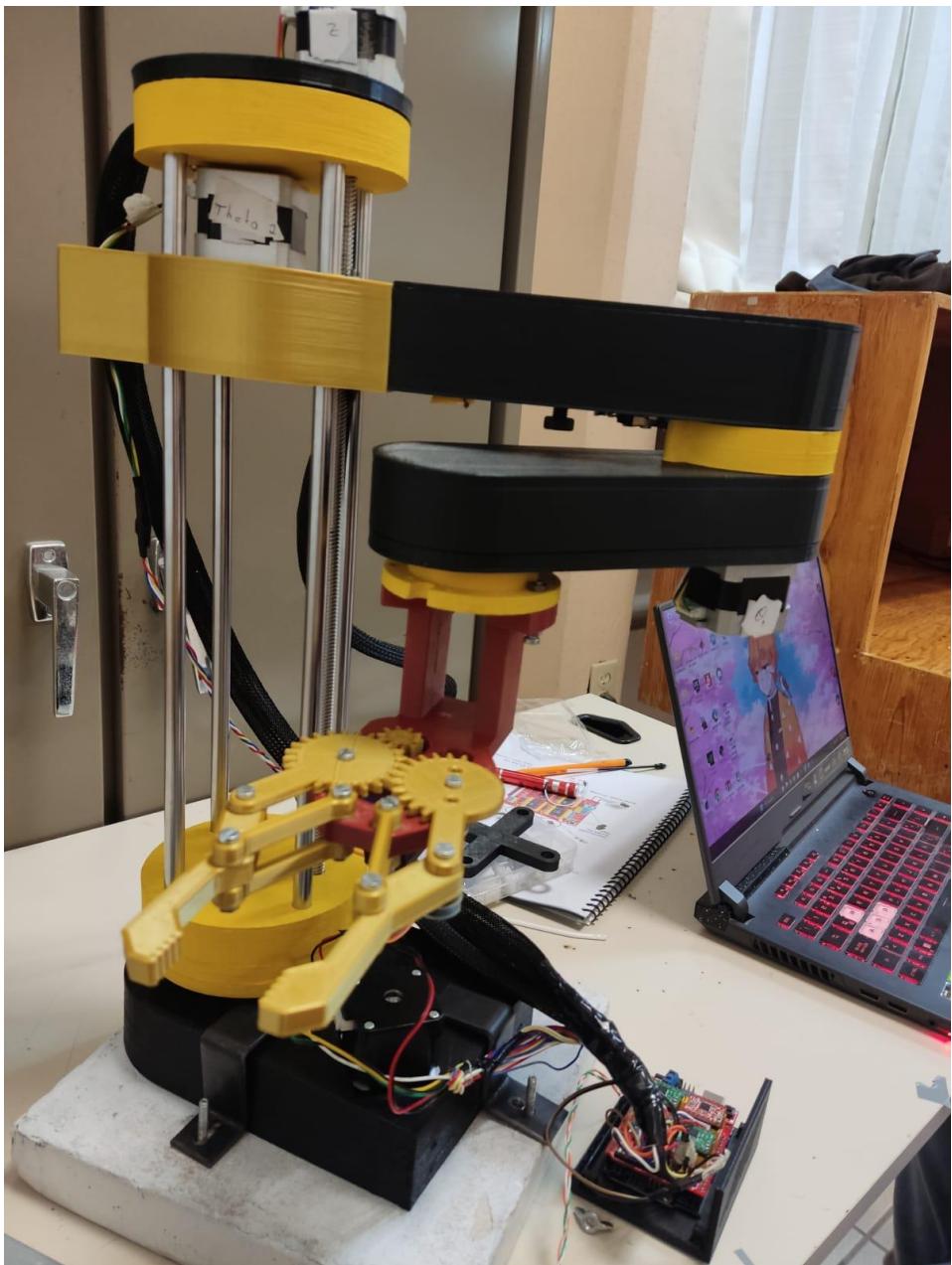


Figura 4.2.12 Prueba de movimiento del SCARA

Una vez realizando pruebas con la interfaz se caracterizaron los límites de las articulaciones respecto a los grados que se envían desde la interfaz ya que no eran del todo exactos, se observó hasta donde podía girar el robot sin presentar alguna obstrucción mecánica y sin dañar la transmisión de las poleas y bandas, además se observó que tanto podía descender el SCARA sin chocar con la base.

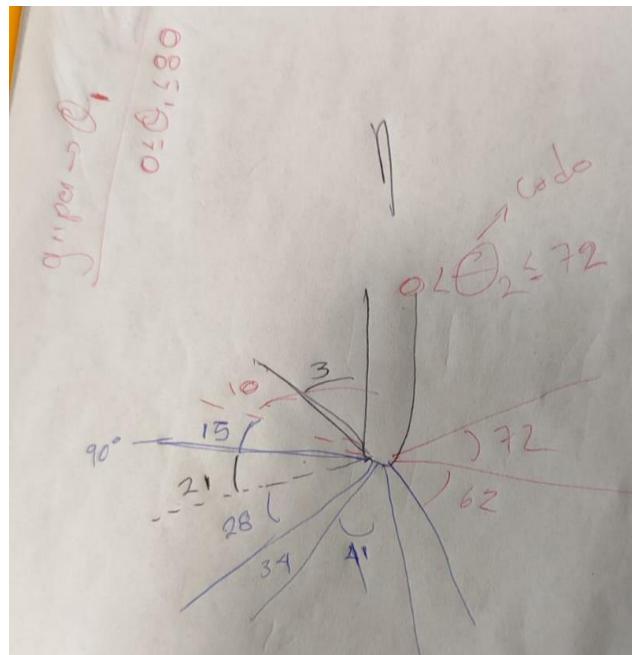


Figura 4.2.13 Rango de giro

Con ayuda de estos parámetros se pudieron determinar los límites máximos de giro y posición para poder realizar una trayectoria, el cual mueve un vaso de un lugar a otro.

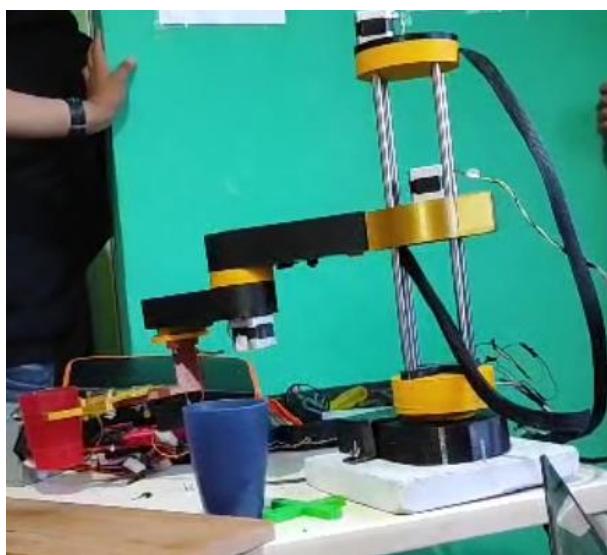


Figura 4.2.13 Posición 1 del vaso



Figura 4.2.14 Posición 2



Figura 4.2.15 Posición 3



Figura 4.2.16 Posición final

Los movimientos y trayectorias del brazo robótico SCARA han demostrado ser satisfactorios, lo que indica que el diseño mecánico y los controladores utilizados son efectivos para lograr la precisión y velocidad requeridas en las tareas de ensamblaje y fabricación.

La electrónica del robot SCARA, que incluye componentes como controladores, motores, y sensores ha demostrado ser confiable y capaz de proporcionar un control preciso del robot. En general, la combinación exitosa de un diseño mecánico sólido y una electrónica bien implementada ha llevado a movimientos y trayectorias satisfactorios en el robot SCARA.

## Interfaz

En primera parte se muestra la ventana principal que es el panel de control en donde se aprecia en la parte central una imagen del robot en 3D que se moverá de acuerdo a la posición que tenga en tiempo real, además se podrán generar archivos para guardas trayectorias deseadas, también se podrá editar la apariencia y se agregaron herramientas que más adelante se explicaran y en la parte inferior izquierda se muestra una barra de estado que indica con mensajes cuando la conexión con el robot sea exitosa, cuando no se pueda realizar una trayectoria, cuando no exista una conexión establecida, etc.

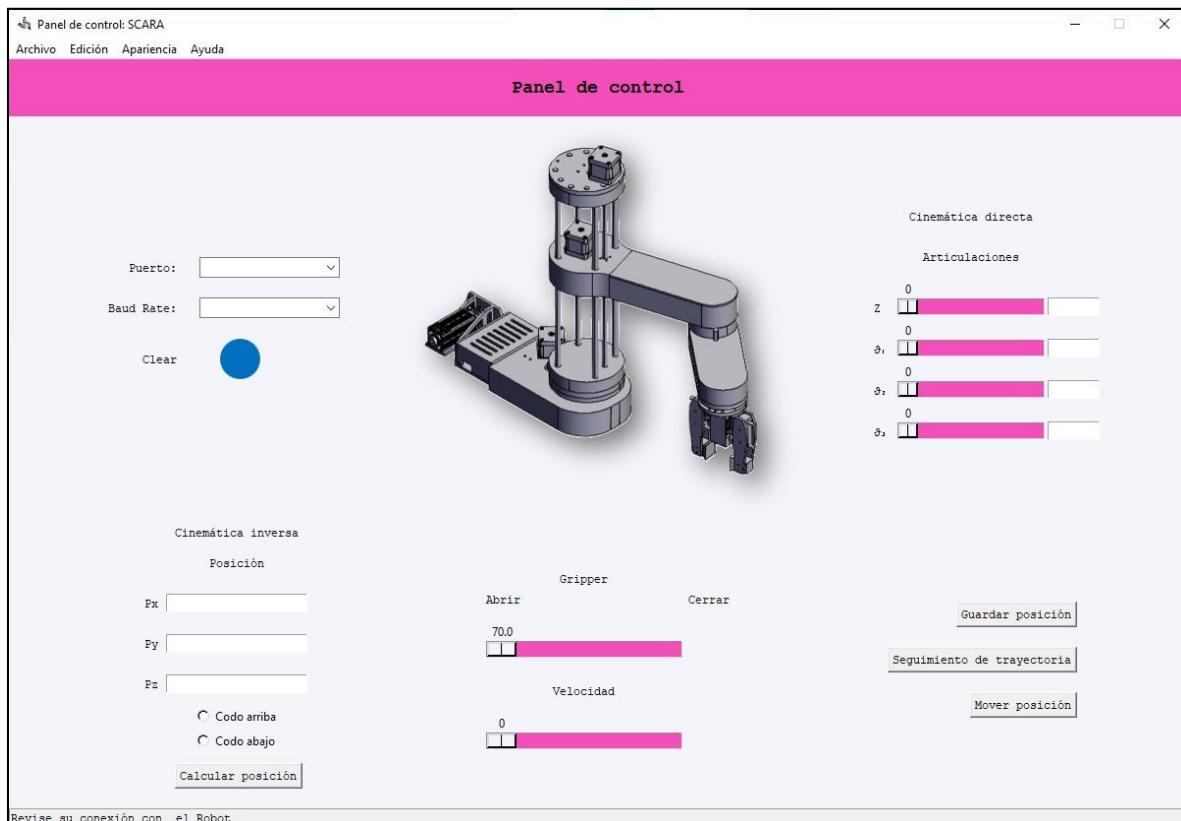


Figura 4.2.17. Interfaz completa del panel de control del robot SCARA.

## Partes de interfaz

### Botones de ventana:

Apariencia: se muestran opciones para la elección de colores para la interfaz, donde podrá seleccionar más de 2 millones de colores.



Figura 4.2.17 Selección de apariencia.

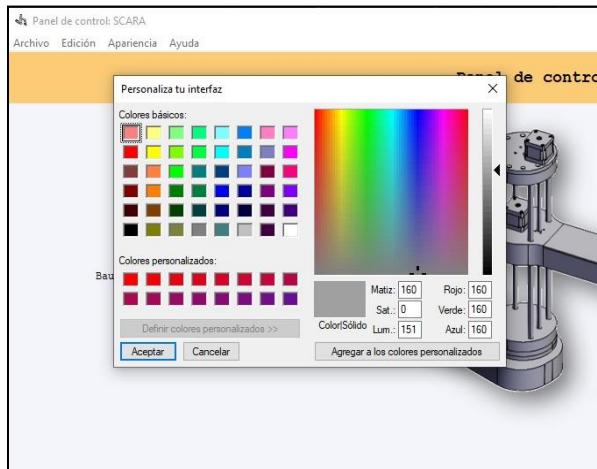


Figura 4.2.18 Ventana de colores para la interfaz.

### Ayuda:

En este apartado encuentra más información sobre el robot, desarrolladores y licencia.

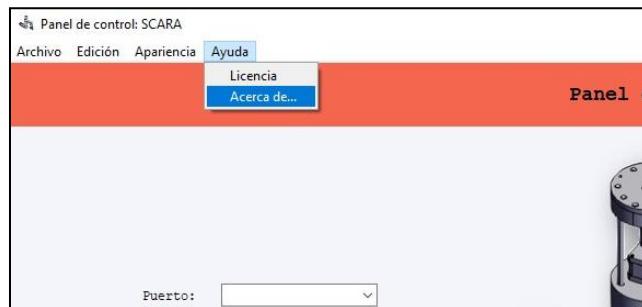


Figura 4.2.19 Apartado de ayuda.

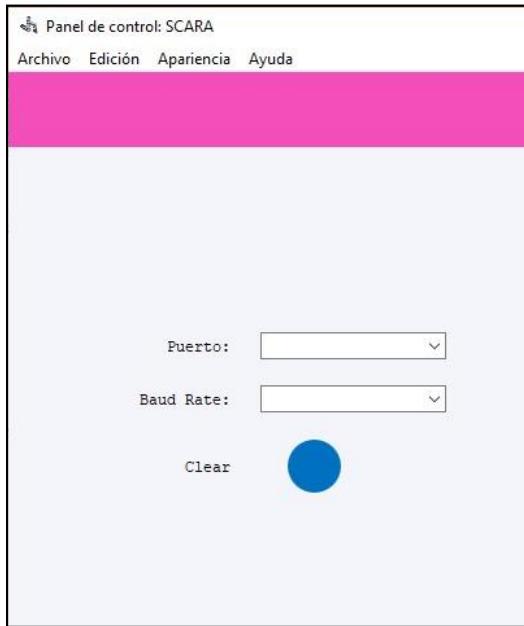
- Licencia: Permisos de licencia y condiciones de uso.
- Acerca de: se encuentra información sobre los desarrolladores del robot SACARA “Roberto”.
- 



*Figura 4.2.20 Información del desarrollador.*

### **Comandos principales:**

Con estos comandos se podrá establecer la conexión con el robot.



*Figura 4.2.21 Comandos principales para la conexión con el robot.*

Puerto: en esta barra desplegable selecciona el puerto con el que se hará la conexión, debido a que la tarjeta de programación es un Arduino UNO se tienen diferentes puertos.

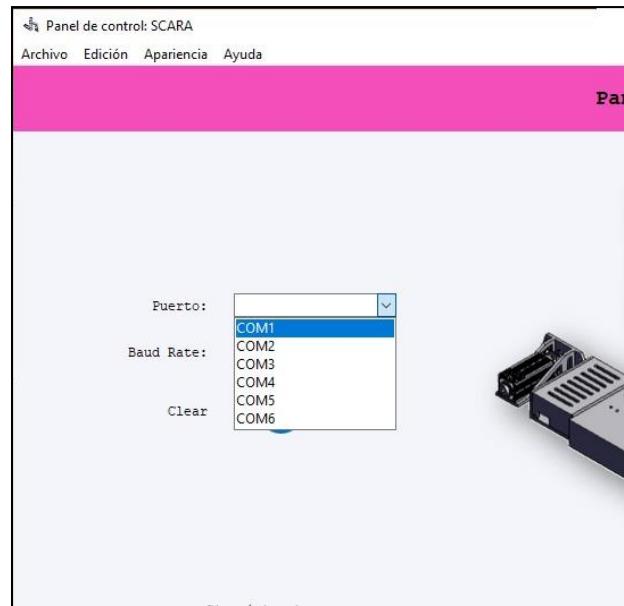


Figura 4.2.22 Selección del puerto para establecer la conexión con Arduino.

Baud Rate: se podrá seleccionar la tasa de baudios, es decir, la velocidad de transferencia de datos.

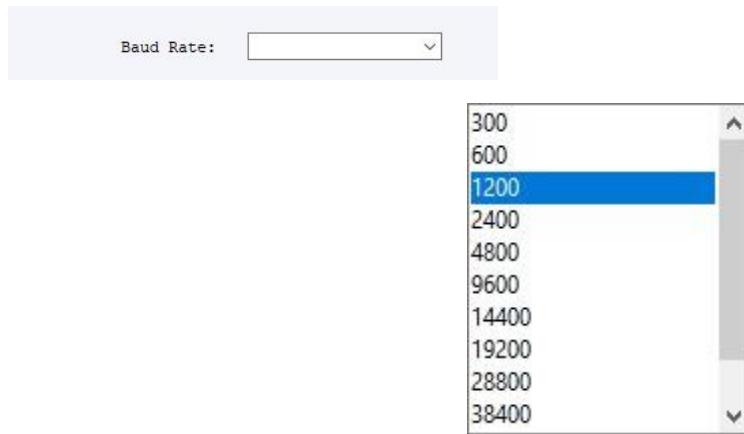


Figura 4.2.23 Selección de la tasa de transferencia de datos.

Clear: limpia los datos que contengan todos los cuadros del panel de control.

## Cinemática inversa

Determina el movimiento de las articulaciones para posicionar el efecto final.

The image shows a software window titled "Cinemática inversa". It has a section labeled "Posición" containing three input fields for coordinates: Px, Py, and Pz. Below these are two radio buttons for "Codo arriba" and "Codo abajo". A "Calcular posición" button is located below the radio buttons. At the bottom of the window is a message: "Revise su conexión con el Robot".

Figura 4.2.24 Parámetros para cinemática inversa.

Px: posición en x.

Py: posición en y.

Pz: posición en z.

Estos valores se ingresan numéricamente.

Además, se agregan las configuraciones de codo arriba y codo abajo y con el botón de **calcular posición** se inicia el movimiento para posicionar el efecto final programado por esta cinemática.

## Cinemática directa

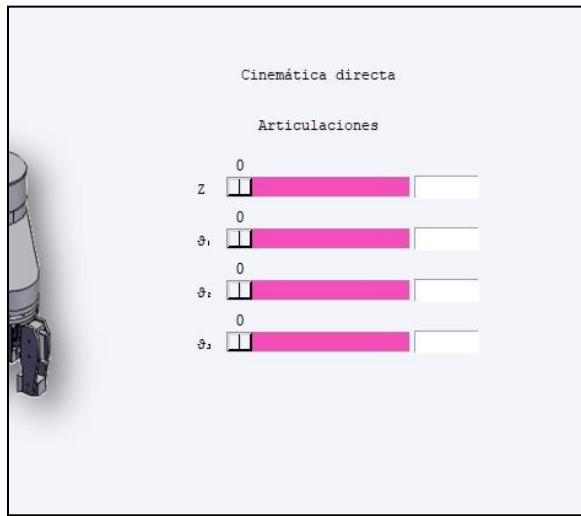


Figura 4.2.25 Parámetros para cinemática directa.

En esta cinemática se podrá controlar la posición del efecto final a través de botones deslizables y automáticamente se calcula la posición de las articulaciones.

Z: Indica la posición en el eje z (eje vertical).

Vartheta1: ángulo de posición del primer brazo del robot.

Vartheta2: ángulo de posición del segundo brazo del robot.

Vartheta3: ángulo de posición del efecto final.

### Botones de acción:

Calcular posición: se presiona una vez que se hayan configurado los parámetros en los campos necesarios.

Guardar posición: guarda la posición del robot de acuerdo a los parámetros ingresados para una posterior reproducción.

Mover posición: se mueve el robot a la posición deseada según se haya configurado en el panel de control.

Seguimiento de trayectoria: el robot seguirá la trayectoria de las posiciones previamente guardadas.



Figura 4.2.26 Botones de acción.

## Gripper

Con estos comandos se puede controlar la apertura del gripper y la velocidad del robot.

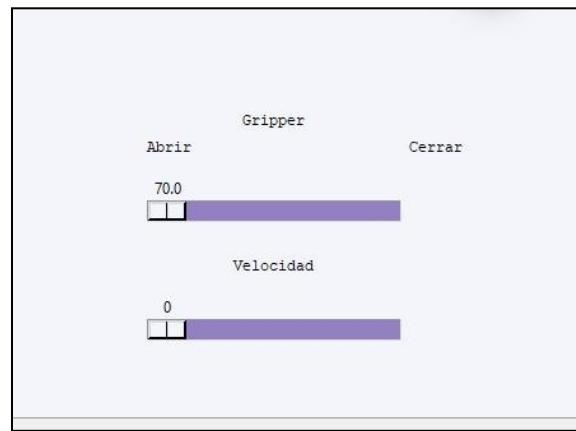


Figura 4.2.27 Configuración del gripper.

## 4.3 Área Auxiliar

### 1. Impresión de modelos 3D para adaptación al robot SCARA:

El primer paso en el proceso de cambio del motor de la base del robot SCARA fue la impresión de modelos 3D para adaptar el nuevo motor al sistema existente. Utilizando software de diseño asistido por computadora (CAD), se crearon los modelos virtuales de las piezas necesarias para esta adaptación y para la mejora de otras partes. Posteriormente, estos modelos se tradujeron en archivos STL, que son el formato estándar utilizado por las impresoras 3D.

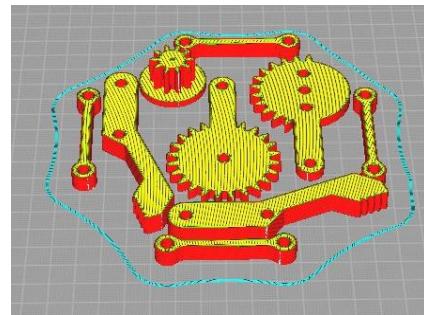


Figura 4.3.1 Robot SCARA

Se seleccionó una impresora 3D adecuada para el proyecto y se utilizó un material resistente y duradero con las siguientes especificaciones: PLA dorado seda, 40% relleno con patrón cúbico y 0.2mm de altura de capa.

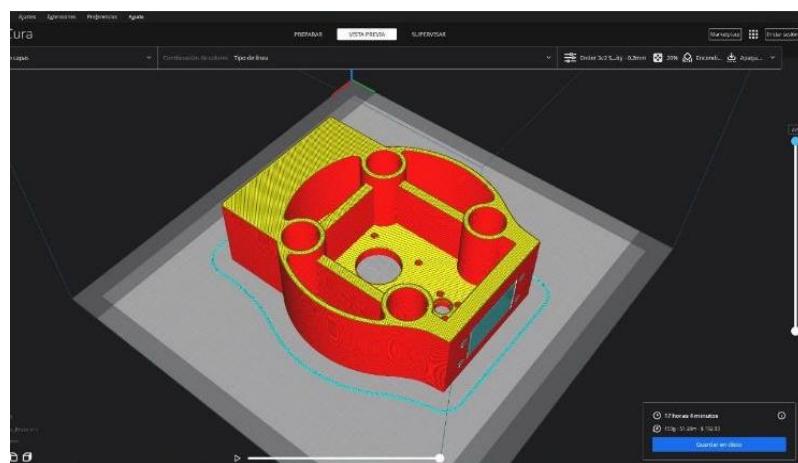
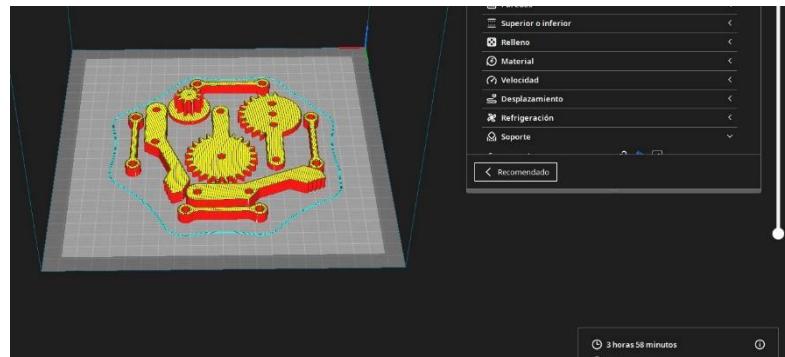


Figura 4.3.2



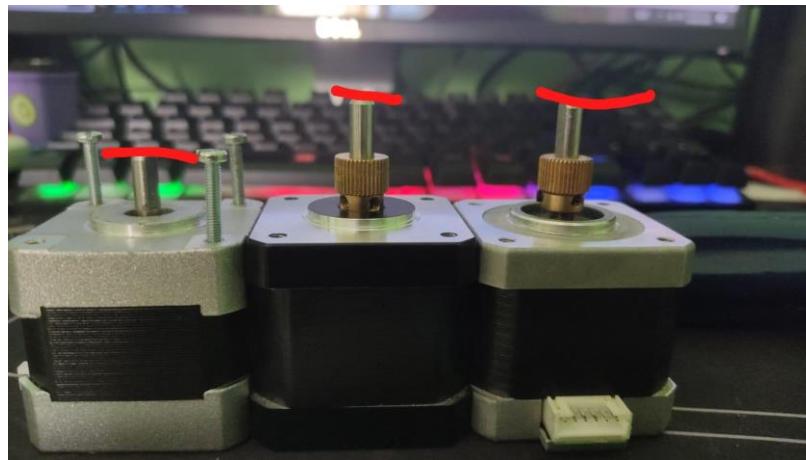
*Figura 4.3.3*

Una vez que las piezas impresas estuvieron listas, se realizaron pruebas de ajuste y compatibilidad para garantizar que se adaptarán correctamente al robot SCARA. Se realizaron ajustes y modificaciones menores cuando fue necesario.

## 2. Cambio del motor por uno con un eje más largo:

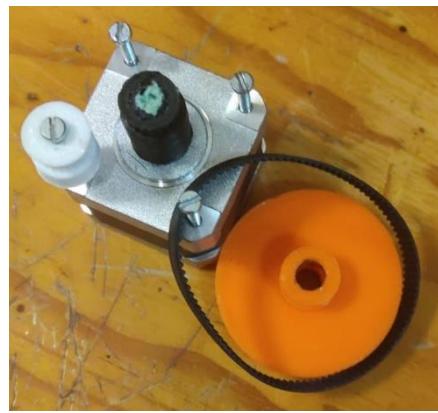
El siguiente paso crucial fue el cambio del motor de la base del robot SCARA. Se investigaron y evaluaron diferentes opciones de motores disponibles en el mercado, teniendo en cuenta criterios como la potencia, el torque, la compatibilidad con el controlador del robot y, lo más importante en este caso, la longitud del eje del motor.

Se seleccionó un nuevo motor con un eje más largo para garantizar una mejor conexión y funcionamiento óptimo en la base del robot. El eje más largo permitiría una mayor estabilidad y una fijación más segura entre el motor y la estructura de la base.



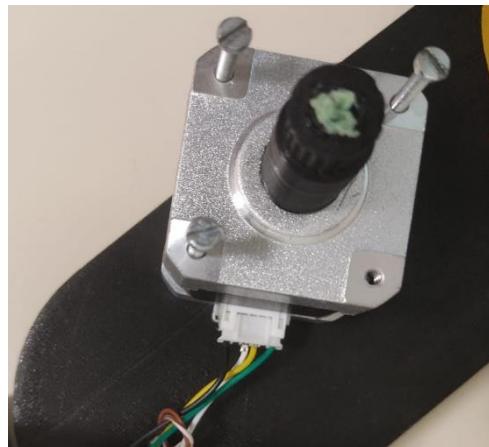
*Figura 4.3.4*

Con el motor seleccionado, se procedió a desmontar el motor existente y desacoplarse de la base del robot. Se tuvieron precauciones para desconectar adecuadamente los cables de alimentación y control antes de retirar el motor antiguo.



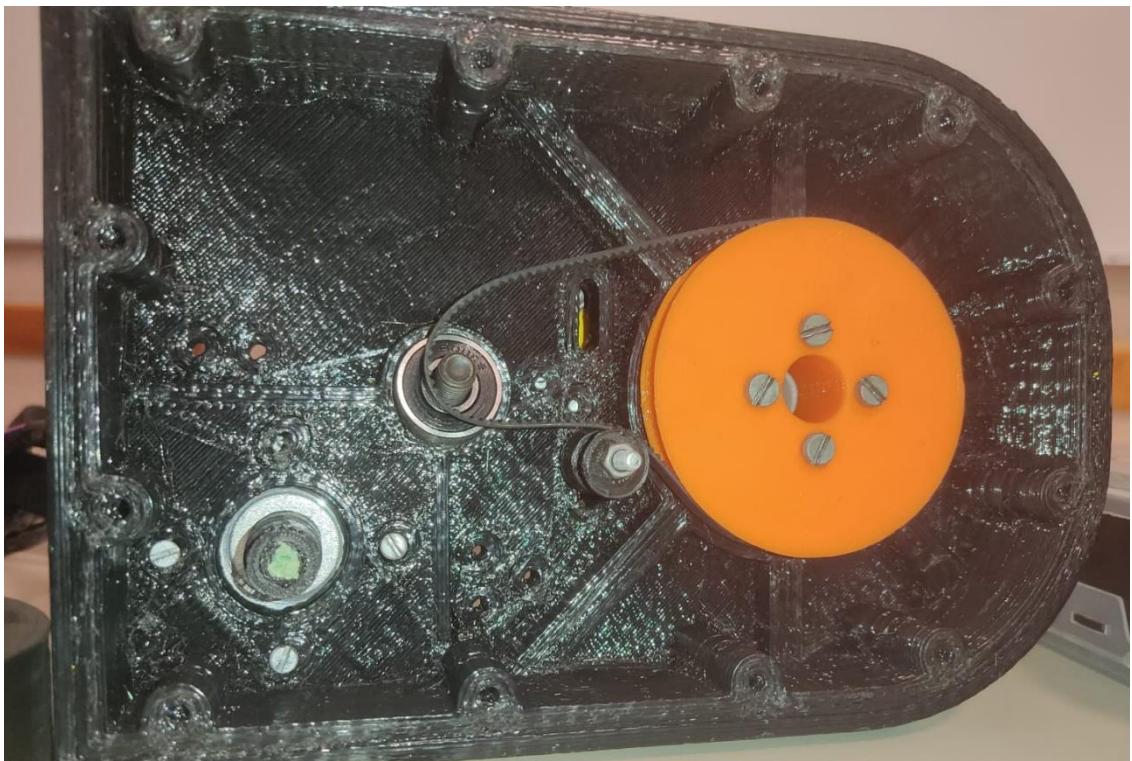
*Figura 4.3.5*

A continuación, se realizó el montaje del nuevo motor en la base del robot utilizando las piezas impresas en 3D y los adaptadores de montaje diseñados previamente. Se aseguró que el motor estuviera firmemente fijado y alineado correctamente.



*Figura 4.3.6*

Finalmente, se conectaron los cables de alimentación y control del nuevo motor al sistema existente, asegurándose de seguir las especificaciones y las conexiones correctas. Se realizaron pruebas iniciales para verificar que el motor funcionara adecuadamente y que estuviera correctamente integrado en el sistema del robot SCARA.



*Figura 4.3.7*

#### 4.4 Área de Gripper

Debido a la falta de confiabilidad del modelo anterior del Gripper, se propuso un nuevo diseño que eliminara la necesidad de las barras de acople y con elementos más confiables y que no presentaran una fractura pronto.

Como primera propuesta se pensó en un diseño basado en engranes rectos. Esto eliminaría totalmente la necesidad de usar barras de acero para una buena alineación, además que aumentaría ligeramente la fuerza de agarre.

Como añadido, se dejan las medidas de las piezas diseñadas:

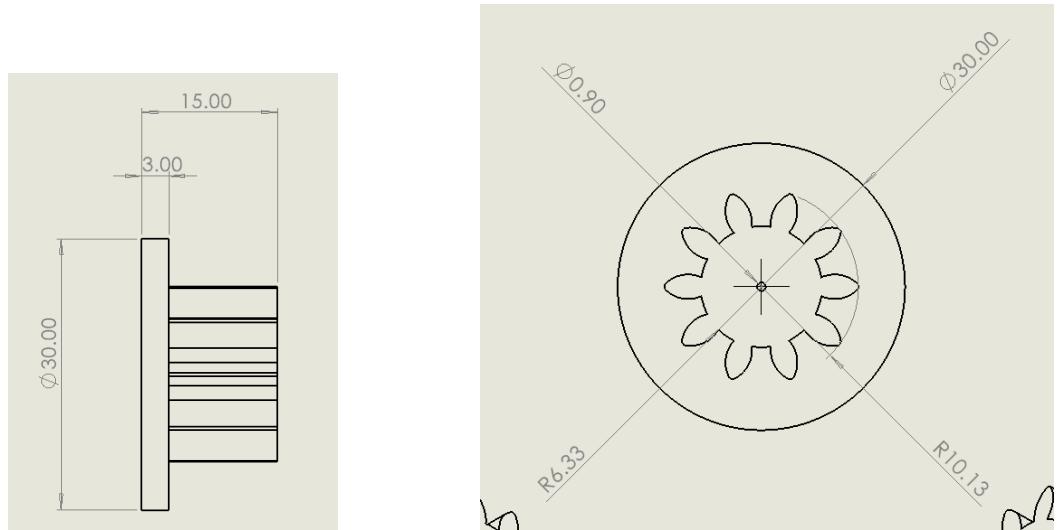


Figura 4.4.1 Medidas del engrane acoplado al servo.

Este primer piñón será el encargado de transmitir el movimiento del servo a todo el mecanismo. Esta pieza estará acoplada a la estrella del servo.

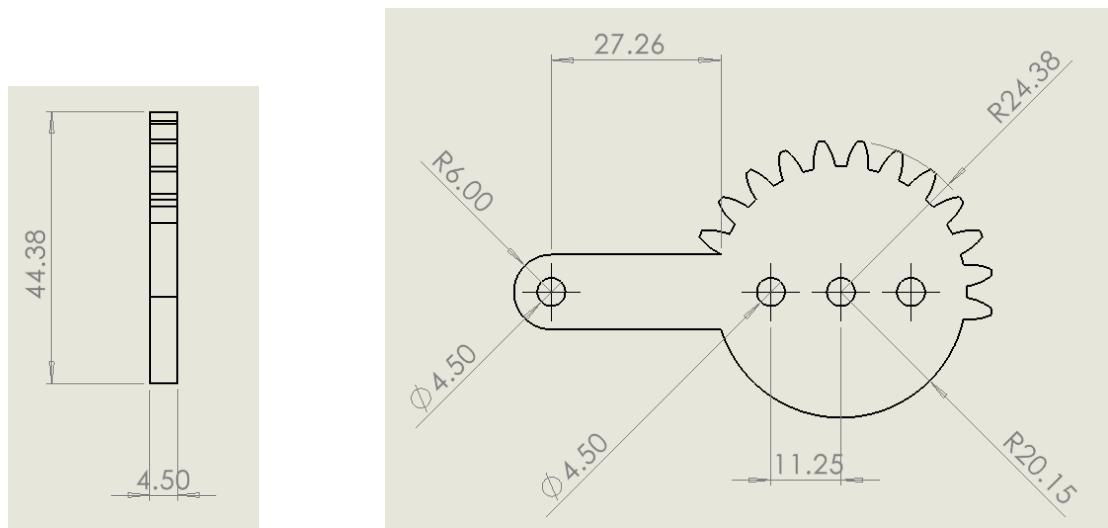
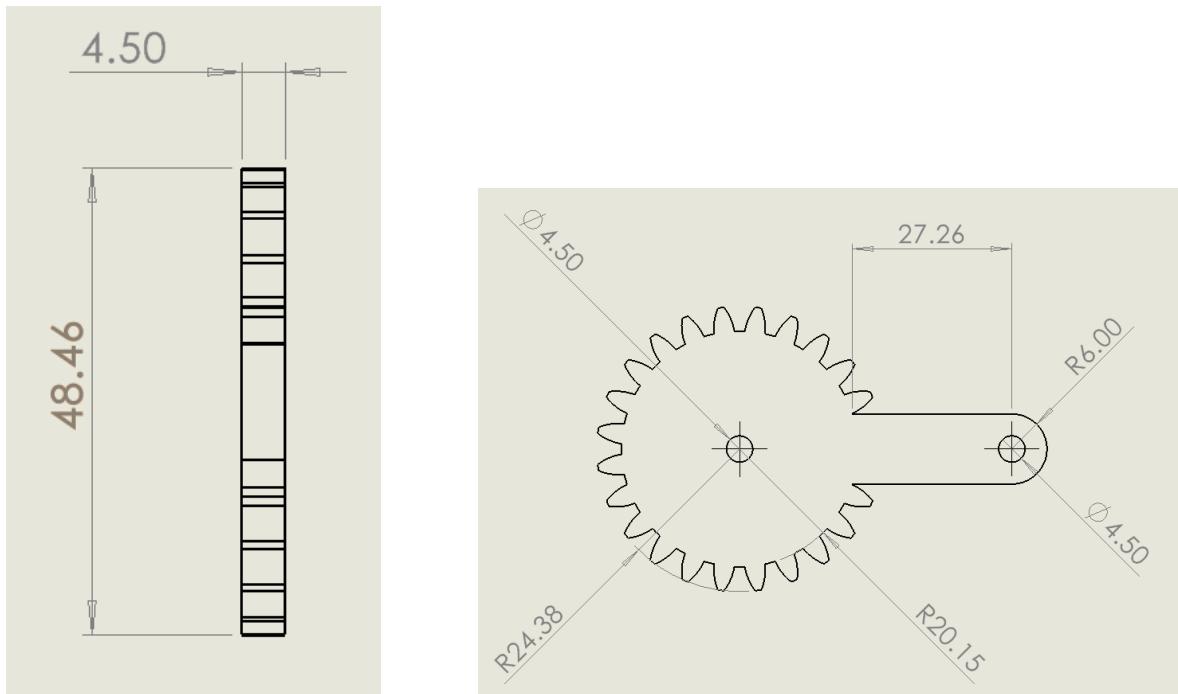


Figura 4.4.2 Medidas de la primera barra engranada.

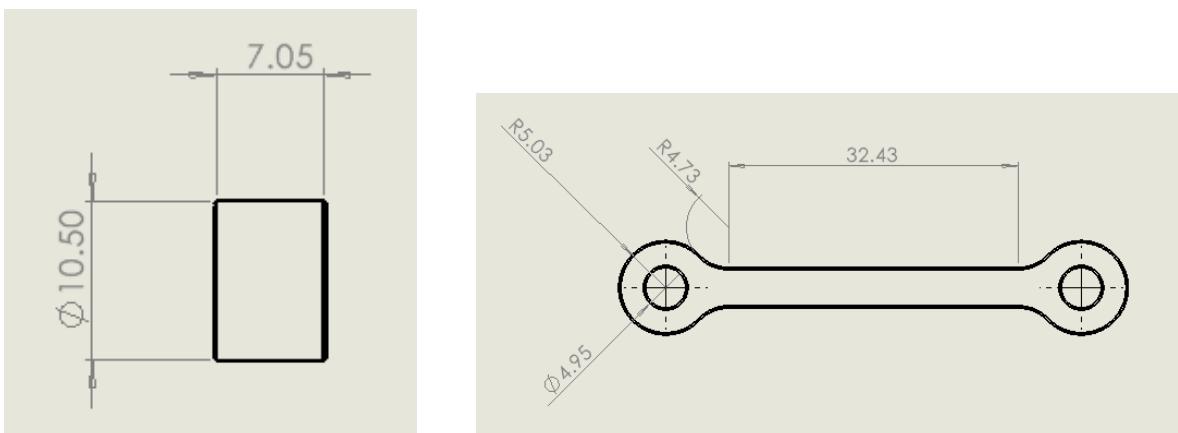
Esta barra engranada es el complemento en el que estará unida una parte de la pinza, moviéndose con ayuda del movimiento del servo y en el mismo sentido.



*Figura 4.4.3 Medidas de la barra engranada secundaria.*

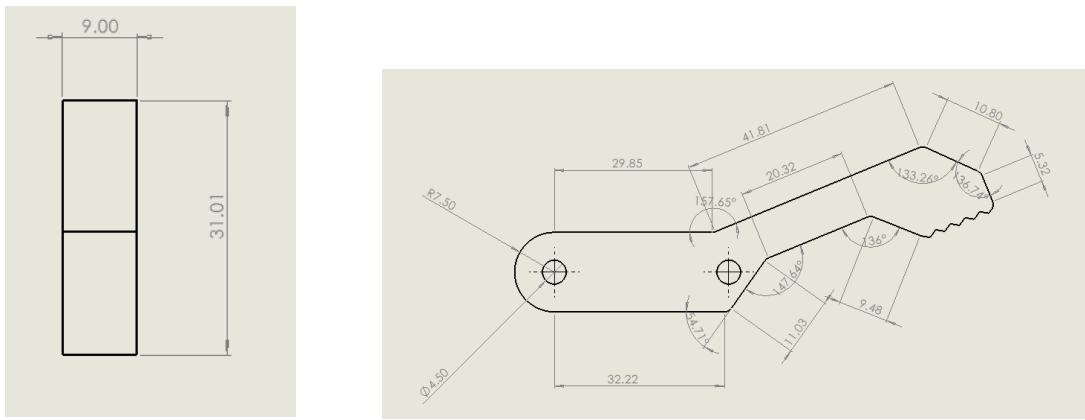
Esta barra secundaria será la encargada de recibir el movimiento del servo y transmitirlo a la barra engranada complementaria. Se puede notar que esta barra y la anterior son similares, solamente se diferencian en el número de orificios que tienen en el centro.

Al estar conectada directamente al engrane del servo, su rotación será en el sentido contrario del mismo.



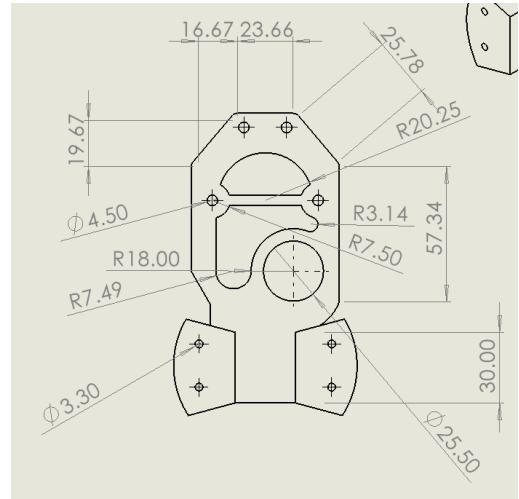
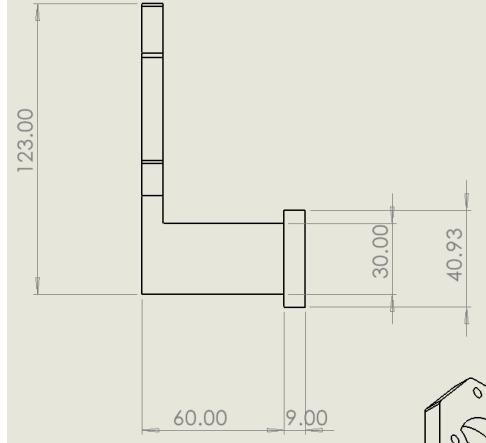
*Figura 4.4.4 Medidas de la barra de sujeción auxiliar.*

Esta pieza se encarga de guiar el movimiento de las pinzas junto con las barras engranadas. El objetivo de estas barras de sujeción es el de inducir un movimiento que siempre sea de apriete y que la orientación de las pinzas sea hacia el frente.



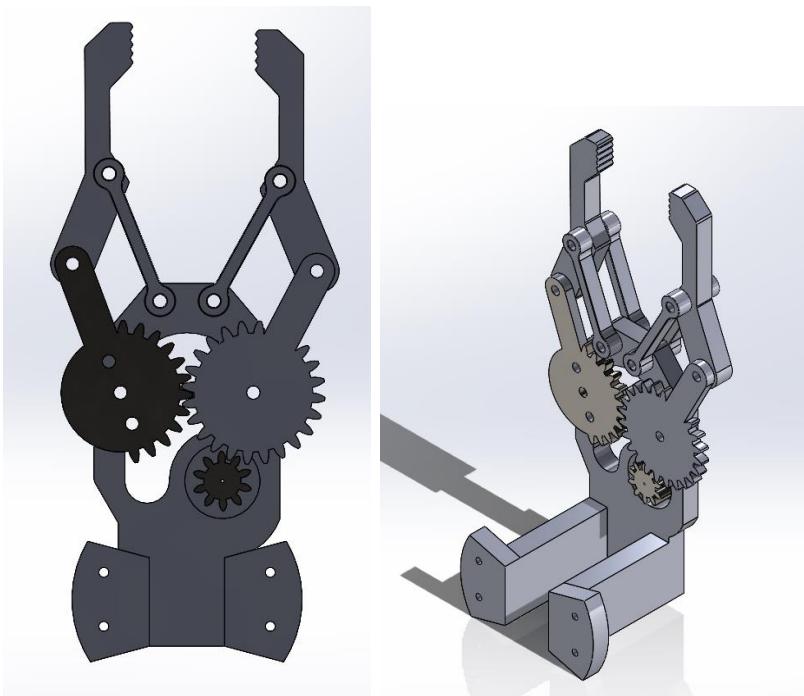
*Figura 4.4.5 Medidas de las pinzas.*

El diseño de las pinzas consiste en una barra con un final con forma de estriado para así garantizar un buen agarre entre esta pieza y el objeto a manipular.



*Figura 4.4.6 Medidas de la pieza que sujeta todo el mecanismo*

Finalmente, tenemos la pieza que soporta a todo el mecanismo ya descrito. Esta pieza conserva el diseño original utilizado para unirse al robot. Es decir, esta nueva pieza diseñada y la anterior son completamente compatibles e incluso puede intercambiarse según sean las necesidades a cumplir.



*Figura 4.4.7 Vista final frontal e isométrica del mecanismo diseñado.*

## 4.5 Área de Interfaz

Para manipular el robot scara se construyó una interfaz de usuario que, mediante comunicación serial, envíe comandos al controlador para ejecutar acciones y seguir trayectorias. La interfaz se compone de tres pestañas principales:

- **Control:** Permite controlar al manipulador eligiendo la posición mediante barras deslizantes y códigos de programación.
- **Configuración:** Establece los parámetros de operación de la interfaz.
- **Instrucciones:** Se detallan instrucciones para la operación del robot usando la interfaz de usuario.

### Pestaña de control

La pestaña de control cuenta con tres secciones:

- **Control de cinemática:** Permite establecer la cinemática directa (controlando el ángulo de giro de cada motor), la cinemática inversa (seleccionando las coordenadas de posición del eslabón final en el espacio), la apertura del gripper y la velocidad del robot . También permite el guardado de posiciones en el controlador interno del robot para su posterior ejecución.

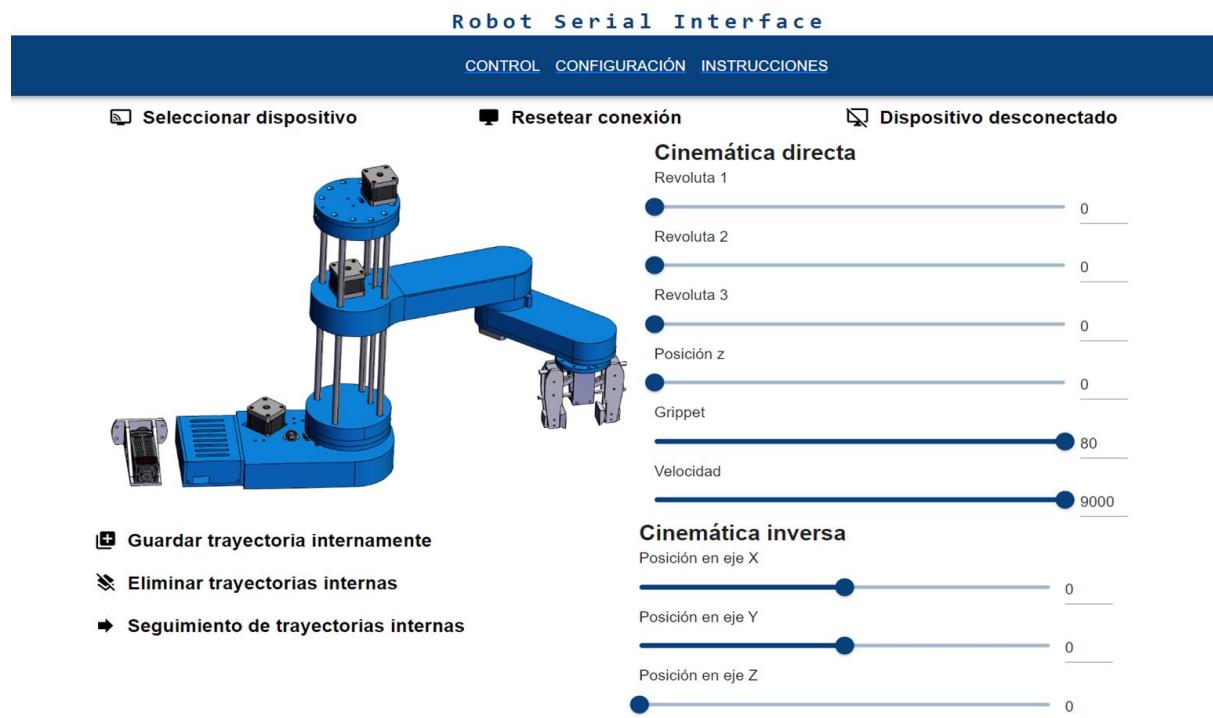


Figura 4.5.1 Interfaz de programación - Control de cinemática

The screenshot shows a software interface for programming. At the top, there is a toolbar with icons for file operations (New, Open, Save, Print, Copy, Paste, Find, Replace, Undo, Redo, Select All, Cut, Copy, Paste, Delete, Find, Replace, Undo, Redo, Select All, Cut, Copy, Paste, Delete), a search bar, and other application-specific icons. Below the toolbar is a code editor window containing the following JavaScript code:

```

1 // Código para movimientos básicos de brazo y antebrazo:
2 setTimeout(function(){
3   |   action('0,0,1,0,0,18,80,9000,0')
4   }, 1000);
5
6 setTimeout(function(){
7   |   action('0,0,40,0,0,18,80,9000,0')
8   }, 2000);
9
10 setTimeout(function(){
11   |   action('0,0,40,13,76,18,80,9000,0')
12   }, 5000);
13
14 setTimeout(function(){
15   |   action('0,0,40,13,76,28,34,9000,0')
16   }, 6000);
17
18
19
20
21
22
23

```

Below the code editor is a terminal window titled "Consola" with the following output:

```

>>
>> Conexión iniciada...
>> No se pudo realizar la acción

```

At the bottom of the interface is a text input field labeled "Ingrese un comando".

*Figura 4.5.2 Interfaz de programación - Editor y consola*

- **Editor de código:** Mediante programación en lenguaje Javascript es posible programar las acciones a ejecutar por la interfaz, por ejemplo es posible indicar una secuencia de posiciones a enviar para realizar un seguimiento de trayectorias. Las acciones de los botones se describen a continuación.
  - **Abrir archivo:** Carga el código de un archivo en el editor.
  - **Descargar:** Descarga un archivo con el código en la carpeta descargas.
  - **Enviar al editor:** Manda las posiciones actuales del manipulador al editor de código. Automáticamente las ingresa dentro de la función **action**, que recibe como parámetro un texto con el comando a enviar vía serial. Esto a su vez es envuelto en la función **setTimeOut** de Javascript para retrasar la acción durante la ejecución del programa, de lo contrario todas las **action** se ejecutarán al mismo tiempo, ocasionando un mal funcionamiento.

```
1  setTimeout(function(){ // Función para retrasar ejecución
2    |   action('0,0,1,0,0,18,80,9000,0')    // Acción a realizar
3  }, 1000);                      // Tiempo de retraso en milisegundos
```

Figura 4.5.3 Ejemplo de acción a ejecutar

- Inicia: Ejecuta el código Javascript escrito en el editor, si hay errores de sintaxis o de lógica habrá un error en tiempo de ejecución.
  - Home: Envía el robot a su posición inicial.
- 
- **Consola:** Permite enviar comandos a través del puerto serial y cuenta con un historial para registrar las acciones realizadas.

#### Pestaña de configuración

En esta pestaña se pueden cambiar los baudios de la comunicación con el robot en caso de ser necesario.

## Configuration

Robot  
scara ▾

Baud rate  
115200 ▾

**Aplica configuración**

Figura 4.5.4 Opciones de pestaña de configuración

## Instrucciones de uso de interfaz

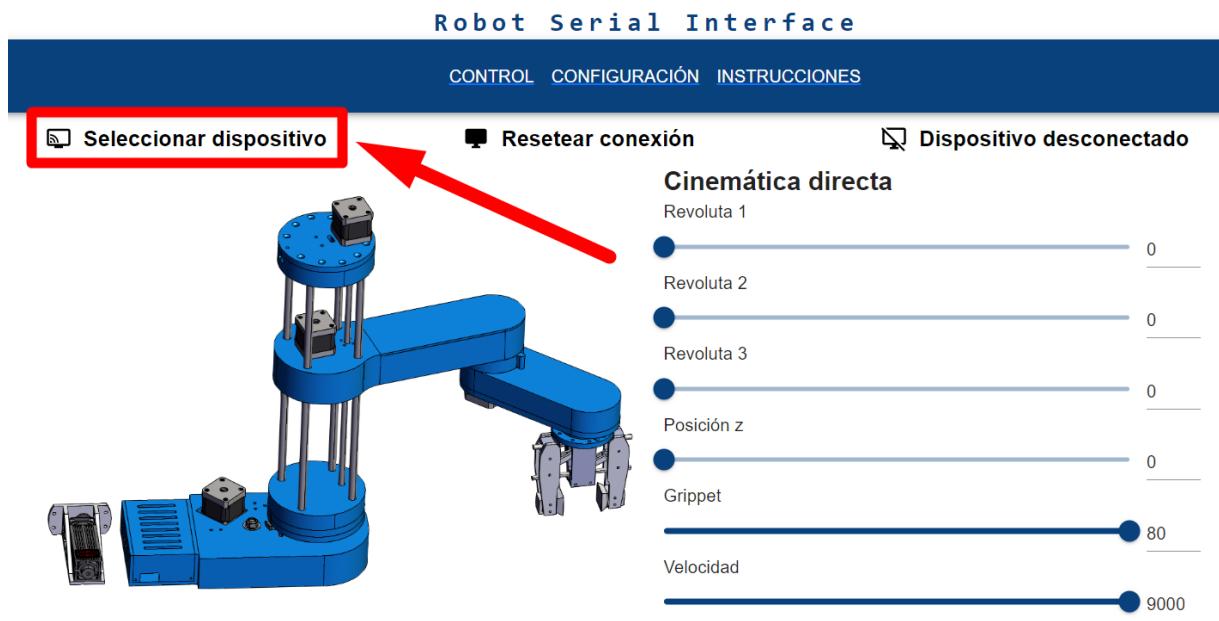


Figura 4.5.5 Presionando el botón de 'Seleccionar dispositivo'

A continuación, se dan las instrucciones de uso de la interfaz para crear un programa, con esto es posible seguir una trayectoria o programar acciones para el robot.

1. Presione el botón 'Seleccionar dispositivo' (Figura 5)
2. Seleccione el puerto de comunicación serial (COM) en el que esté conectado el dispositivo (Figura 6). En caso de estar conectado vía Bluetooth será necesario seleccionar el puerto COM de salida, esto se puede consultar en la configuración de Bluetooth de su computadora.

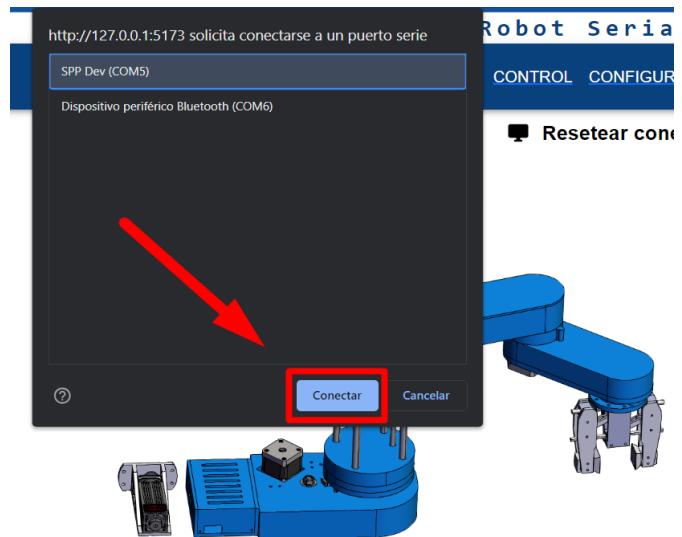


Figura 4.5.6 Seleccionando puerto de comunicación

- Verifique que el dispositivo esté conectado. Se muestra el estatus de conexión debajo del menú de la aplicación (Figura 7). También se puede verificar el estatus de la conexión en el historial de la consola como se muestra en la Figura 8.

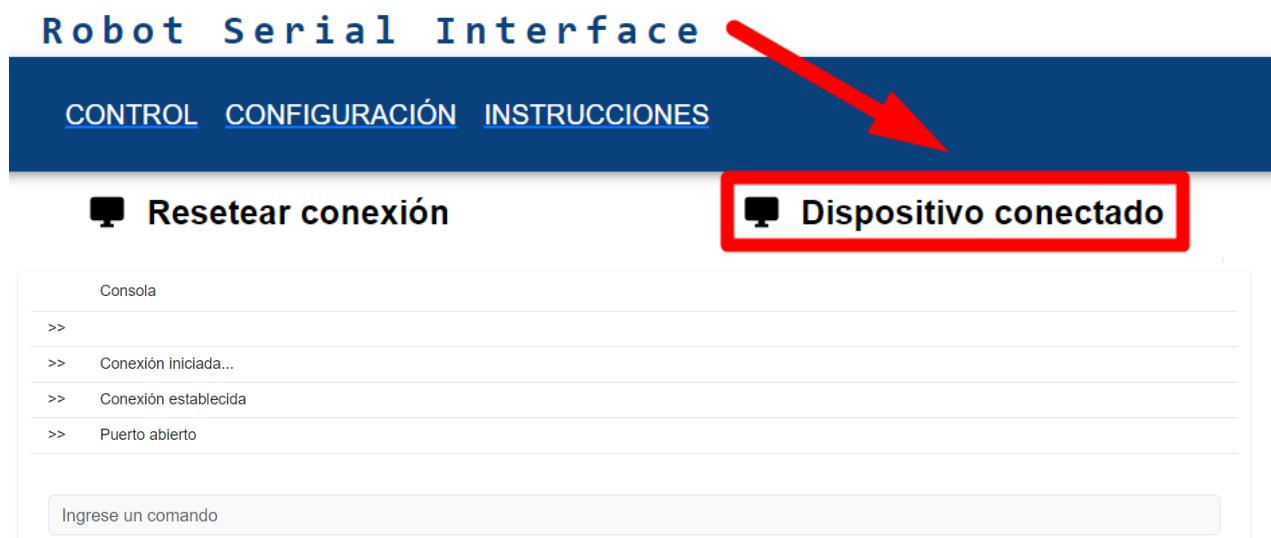


Figura 4.5.7 Estatus de conexión en la consola

- Seleccione la posición del robot, esto se puede realizar mediante cinemática directa, seleccionando los ángulos de giro de los motores, o usando cinemática inversa, fijando la posición en el espacio del efecto final. En el ejemplo de la Figura 9 se realizó mediante cinemática directa.

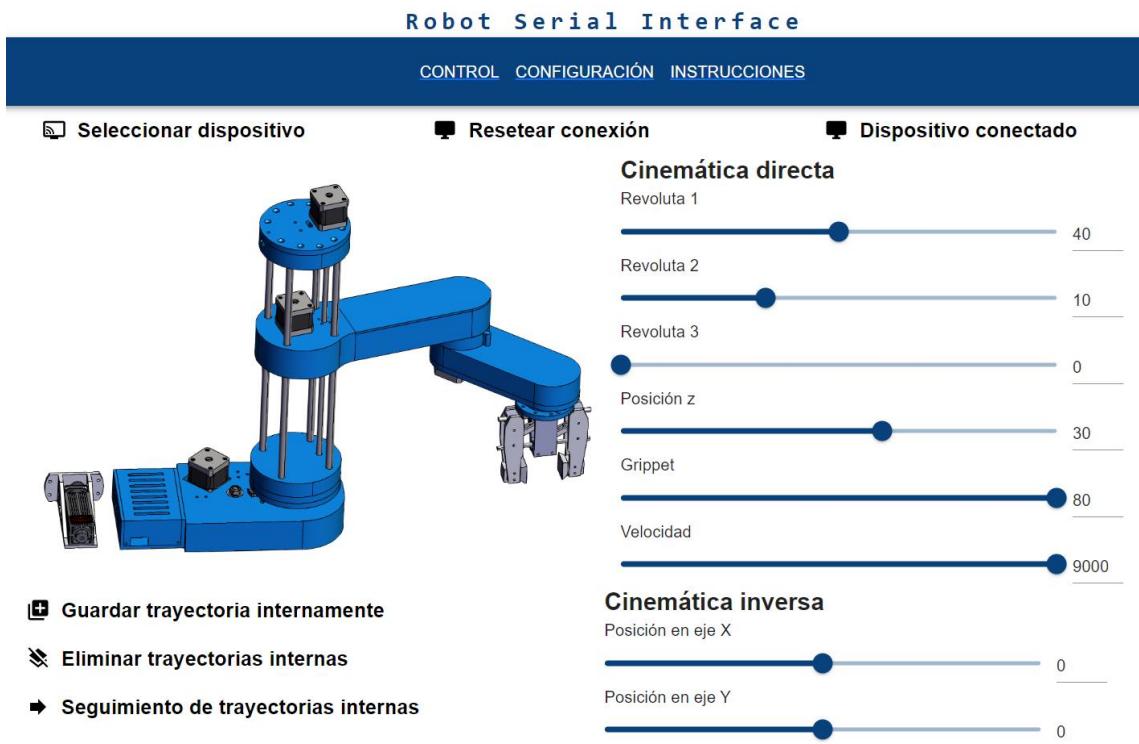


Figura 4.5.8 Seleccionando la posición del robot mediante cinemática directa

- Envíe la posición del editor de código presionando el botón como se muestra en la Figura 10.

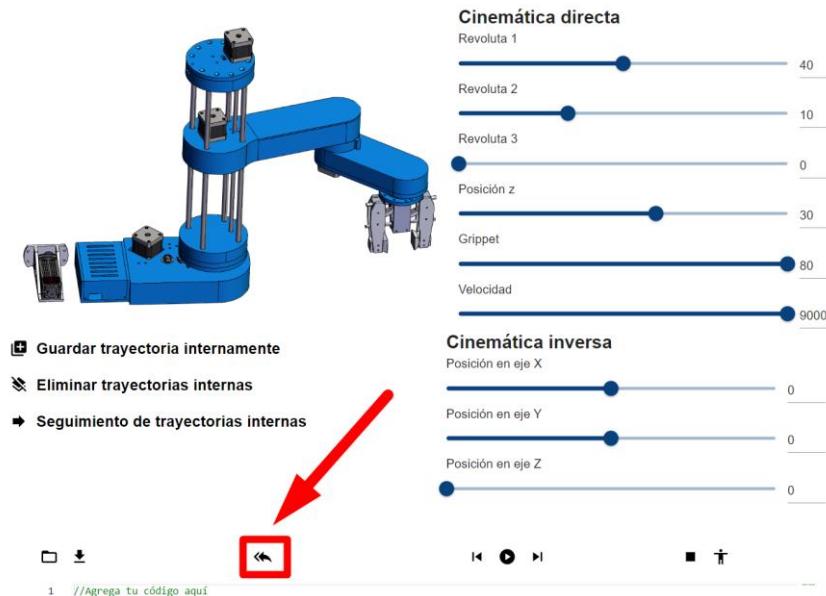
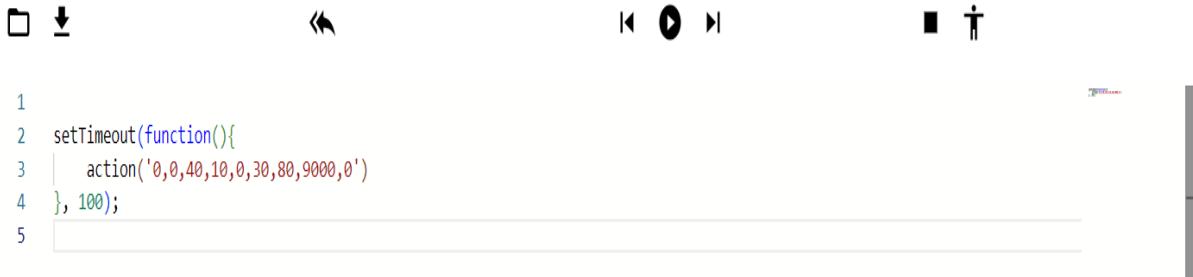


Figura 4.5.9 Enviar posiciones a editor

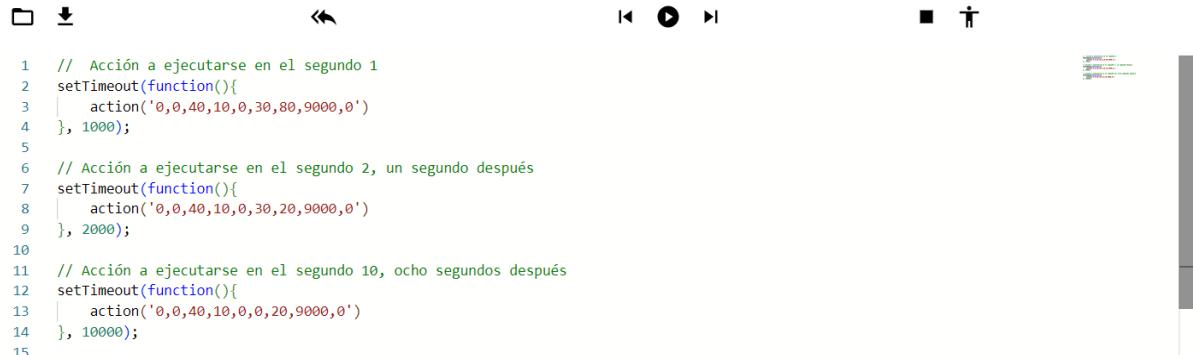
6. Se enviará al editor las posiciones como se muestra en la Figura 11. La explicación del código enviado se encuentra en la Figura 3.



```
1 settimeout(function(){
2   |   action('0,0,40,10,0,30,80,9000,0')
3   }, 100);
4
5
```

Figura 4.5.10 Código enviado al editor

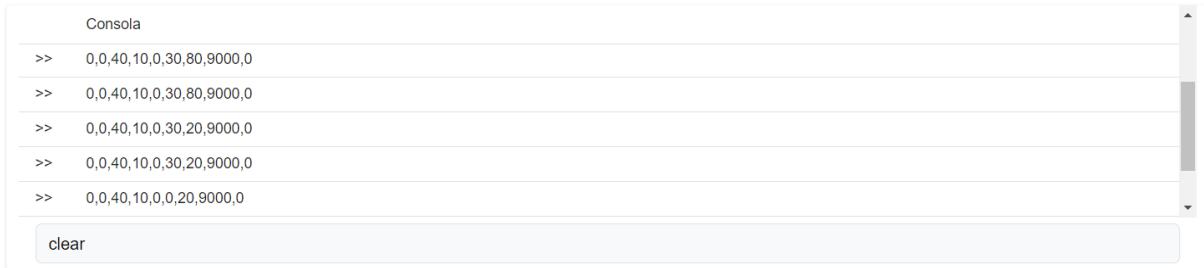
7. Repita los pasos tantas veces como sea necesario hasta definir la trayectoria a seguir. Posteriormente es posible editar el código para agregar comentarios, funciones etc. Se recomienda encarecidamente editar el tiempo de ejecución de cada acción para evitar errores. Un ejemplo de código editado se puede encontrar en la Figura 12.



```
1 // Acción a ejecutarse en el segundo 1
2 setTimeout(function(){
3   |   action('0,0,40,10,0,30,80,9000,0')
4   }, 1000);
5
6 // Acción a ejecutarse en el segundo 2, un segundo después
7 setTimeout(function(){
8   |   action('0,0,40,10,0,30,20,9000,0')
9   }, 2000);
10
11 // Acción a ejecutarse en el segundo 10, ocho segundos después
12 setTimeout(function(){
13   |   action('0,0,40,10,0,0,20,9000,0')
14   }, 10000);
15
```

Figura 4.5.11 Ejemplo de trayectoria con código editado

8. Finalmente, presione el botón ‘Iniciar’. Esto ejecutará el código escrito en el editor. Los comandos enviados vía serial se mostrarán en el historial de la consola. En caso de existir un error esto también se mostrará en el historial.



A screenshot of a terminal window titled 'Consola'. The window displays a list of commands sent via serial, each starting with '>>'. The commands are: '0,0,40,10,0,30,80,9000,0', '0,0,40,10,0,30,80,9000,0', '0,0,40,10,0,30,20,9000,0', '0,0,40,10,0,30,20,9000,0', and '0,0,40,10,0,0,20,9000,0'. At the bottom left of the window is a 'clear' button.

Figura 4.5.12 Historial con comandos enviados vía serial

9. Si desea guardar el código en la computadora presione el botón ‘Descargar’. Para volver a cargarlo deberá presionar el botón ‘Seleccionar archivo’.
10. Es posible ejecutar acciones individuales escribiendo el comando en la consola. Por ejemplo ‘clear all’ para limpiar la consola y el editor, o escribiendo la posición a tomar el robot como se muestra en este ejemplo: 0,0,40,13,76,18,80,9000,0.

#### Obtener interfaz

Para obtener la interfaz solo será necesario ingresar al siguiente link. No es necesario instalar ni descargar nada. El único requisito es que el navegador del dispositivo cuente con comunicación serial. Al momento de escritura de éste reporte esto solo se ha probado en computadoras con Google Chrome instalado.



Figura 4.5.13 Link de interfaz

[https://deschain53.github.io/Robot\\_Serial\\_Interface/](https://deschain53.github.io/Robot_Serial_Interface/)

El repositorio con todo el código de la interfaz se puede descargar en:  
[https://github.com/Deschain53/Robot\\_Serial\\_Interface](https://github.com/Deschain53/Robot_Serial_Interface)

Si desea realizar modificaciones es necesario instalar el manejador de paquetes Yarn para levantar el entorno de desarrollo y editar el código. La interfaz fué construida usando React v17, Material MUI v5 y React-Router-Dom v5.

Video de demostración del SCARA en funcionamiento:

<https://youtu.be/gqezdF-1S3w>

Integrantes externos que apoyaron esta tarea:

- Galindo Luna Areli Jaquelin
- José Armando Zafra Asunción
- Gómez Martínez José Francisco
- Jiménez Rojas Esteban
- Campechano Antonio Wilfrido

## **5. CONCLUSIONES**

- **Área Mecánica**

Los cambios realizados en el manipulador SCARA existente han demostrado ser altamente beneficiosos en términos de mejora de la estabilidad, funcionalidad y rendimiento del sistema. La nueva base proporcionó una mayor estabilidad estructural, mientras que la nueva pieza con contrapeso solucionó los problemas de los rodamientos y mejoró la capacidad de elevación del manipulador. La construcción de la araña permitió mantener una distancia igual entre los ejes verticales, reduciendo el calentamiento de los motores y permitiendo un deslizamiento fluido a través del tornillo sin fin. El ajuste y tensado de las bandas, junto con los cambios en las tuercas y los tornillos, mejoraron la transmisión de movimiento y la confiabilidad del manipulador.

En general, estos cambios han llevado a una mejora significativa en el manipulador SCARA, permitiendo un funcionamiento más eficiente, preciso y confiable. Estas modificaciones demuestran la importancia de la mecánica en los manipuladores y cómo un diseño adecuado y ajustes precisos pueden marcar la diferencia en su desempeño. Estas mejoras también subrayan la importancia de un enfoque continuo en la optimización y mantenimiento de los sistemas mecánicos para garantizar su funcionamiento óptimo en diversas aplicaciones industriales.

- **Área Electrónica**

En conclusión, la construcción de un robot SCARA en el área de la electrónica es un desafío emocionante y gratificante. Durante este proceso se tiene la oportunidad de aplicar los conocimientos teóricos adquiridos en su formación académica y adquirir experiencia práctica en el diseño, la integración componentes electrónicos.

La construcción de la electrónica del robot SCARA implica la selección y configuración adecuada de controladores, motores, sensores y circuitos de potencia. Se debe de aprender a realizar conexiones eléctricas correctas, programar controladores y desarrollar algoritmos de control para lograr movimientos precisos y coordinados del brazo robótico. A través de este proyecto, pueden explorar y comprender los principios fundamentales de la electrónica en el contexto de la robótica, lo que les proporciona una base sólida para el desarrollo de proyectos más avanzados en el futuro. Es un proyecto que fomenta la creatividad, el trabajo en equipo y el desarrollo de habilidades técnicas, preparando a los estudiantes para enfrentar los desafíos de la automatización industrial y la robótica en el mundo real.

- **Área Auxiliar**

El cambio del motor de la base del robot SCARA ha sido un proceso fundamental para mejorar el rendimiento y la funcionalidad de este sistema robótico. A través de la impresión de modelos 3D para adaptar el nuevo motor y la selección de un motor con un eje más largo, se lograron importantes avances en términos de estabilidad, precisión y eficiencia en las operaciones del robot.

La impresión de modelos 3D permitió la creación de piezas personalizadas que facilitaron la integración del nuevo motor en la base del robot. Esta tecnología ha demostrado ser una herramienta valiosa en la adaptación y modificación de sistemas existentes, proporcionando flexibilidad y precisión en la fabricación de componentes.

El cambio del motor por uno con un eje más largo ha resultado en una mayor estabilidad y una mejor conexión entre el motor y la estructura de la base. Esta mejora ha contribuido a reducir los problemas de calentamiento y a incrementar la precisión y velocidad de los movimientos del brazo del robot SCARA. Además, la selección cuidadosa del motor ha garantizado la compatibilidad con el controlador del robot y ha optimizado su rendimiento general.

Es importante destacar que este proyecto no estuvo exento de desafíos. La selección adecuada del motor y la adaptación precisa de las piezas requerían un análisis detallado y un trabajo minucioso. Sin embargo, los beneficios obtenidos a partir del cambio del motor justifican los esfuerzos realizados. En conclusión, el cambio del motor de la base del robot SCARA ha sido un proyecto exitoso que ha demostrado la importancia de la adaptación y mejora de los componentes clave en los sistemas robóticos.

- **Área de Gripper**

En conclusión, la construcción de un gripper de un manipulador requiere un enfoque cuidadoso en el diseño, la selección de materiales y componentes, la elección de mecanismos de agarre adecuados, la integración con el manipulador y las pruebas exhaustivas. Al construir un gripper con precisión y atención a los detalles, se puede lograr un manipulador eficiente y confiable para satisfacer las necesidades específicas de la aplicación.

En comparación con el modelo anterior, este diseño propuesto soluciona los problemas con las barras que provocaban un peor desgaste en el modelo anterior, con esta propuesta no ocurre eso y sigue teniendo la misma funcionalidad que antes.

- **Área de Interfaz**

En conclusión, la implementación de una interfaz para controlar un robot SCARA, abarcando el control de los eslabones, la cinemática inversa, la apertura del gripper y la velocidad, es un hito significativo en la automatización y la robótica. Esta interfaz brinda a los usuarios la capacidad de interactuar de manera intuitiva y eficiente con el robot, permitiendo un control preciso y personalizado de sus movimientos y acciones.

La posibilidad de controlar los eslabones del robot SCARA a través de la interfaz proporciona una flexibilidad y versatilidad excepcionales. Los usuarios pueden ajustar las posiciones y las trayectorias de los eslabones según las necesidades de la aplicación.

La implementación de la cinemática inversa en la interfaz es una característica clave, ya que permite al usuario definir la posición final deseada del efecto final del robot y automáticamente calcular los ángulos de los eslabones requeridos para alcanzar esa posición. Esto simplifica el proceso de programación y control del robot, ahorrando tiempo y esfuerzo a los usuarios.

Además, el control de la apertura del gripper a través de la interfaz proporciona una funcionalidad adicional para el manejo y la manipulación de objetos. Los usuarios pueden ajustar la apertura del gripper según el tamaño y la forma de los objetos que se deben agarrar, lo que permite una mayor precisión y eficiencia en las tareas de manipulación.

Por último, la capacidad de controlar la velocidad del robot a través de la interfaz es esencial para adaptarse a diferentes requerimientos de velocidad en las tareas. Los usuarios pueden ajustar la velocidad de los movimientos del robot según las necesidades de la aplicación, ya sea para aumentar la eficiencia en tareas repetitivas o para garantizar una manipulación segura y precisa de objetos delicados.

## **6. BIBLIOGRAFIA**

1. Asada, H., & Kanade, T. (1983). Robot calibration: a survey. *Robotics and Autonomous Systems*, 1(3), 213-232.
2. Kim, Y., & Suh, I. H. (2011). Robust control of SCARA robot manipulator using a sliding mode control and genetic algorithm-based PID control. *Expert Systems with Applications*, 38(5), 5926-5933.
3. Lozano-Guzmán, R. D., & Gómez-Gil, J. (2017). A robust position control of SCARA robots using adaptive fuzzy sliding mode controller. *Journal of Intelligent & Robotic Systems*, 88(1), 153-171.
4. Nagpal, R., & Jindal, G. (2016). Modeling and simulation of SCARA robot manipulator using MATLAB. *International Journal of Emerging Technology and Advanced Engineering*, 6(2), 193-196.
5. Pramanik, A., Roy, R., & Ghosh, D. (2017). Modeling and simulation of a SCARA robot for pick and place application. *Procedia Computer Science*, 105, 93-98.

## 7. ANEXOS

### Código

```
#include <Servo.h>
#include <math.h>

#define limitSwitch1 11
#define limitSwitch2 10
#define limitSwitch3 9
#define limitSwitch4 A3
#define stepper1_step 2
#define stepper1_dire 5
#define stepper2_step 3
#define stepper2_dire 6
#define stepper3_step 4
#define stepper3_dire 7
#define stepper4_step 12
#define stepper4_dire 13
#define habi 8
int MaxSpeed = 9000;
int Minwait = 800;
int wait=800;
String aux;
Servo gripperServo; // create servo object to control a servo

double x = 10.0;
double y = 10.0;
double L1 = 228; // L1 = 228mm
double L2 = 136.5; // L2 = 136.5mm
double theta1, theta2, phi, z;

int stepper1Position, stepper2Position, stepper3Position, stepper4Position;
int stepper1CurrentPosition, stepper2CurrentPosition, stepper3CurrentPosition,
stepper4CurrentPosition;
const int theta1AngleToSteps = 10;
const int theta2AngleToSteps = 36;
const int phiAngleToSteps = 44;
const int zDistanceToSteps = 100;

byte inputValue[5];
int k = 0;

String content = "";
int data[10];
int theta1Array[100];
int theta2Array[100];
```

```

int phiArray[100];
int zArray[100];
int gripperArray[100];
int positionsCounter = 0;
void setup() {
    Serial.begin(115200);

    pinMode(limitSwitch1, INPUT_PULLUP);
    pinMode(limitSwitch2, INPUT_PULLUP);
    pinMode(limitSwitch3, INPUT_PULLUP);
    pinMode(limitSwitch4, INPUT_PULLUP);

    pinMode(stepper1_step, OUTPUT); pinMode(stepper1_dire, OUTPUT);
    pinMode(stepper2_step, OUTPUT); pinMode(stepper2_dire, OUTPUT);
    pinMode(stepper3_step, OUTPUT); pinMode(stepper3_dire, OUTPUT);
    pinMode(stepper4_step, OUTPUT); pinMode(stepper4_dire, OUTPUT);
    pinMode(habi, OUTPUT);

    gripperServo.attach(A0, 600, 2500);
    // initial servo value - open gripper
    data[6] = 20;
    gripperServo.write(data[6]);
    delay(1000);
    homing();
}

void loop() {
    if (Serial.available())
    {
        ReadData();
        if (data[0] == 1)
        {
            theta1Array[positionsCounter] = data[2] * theta1AngleToSteps; //store the
values in steps = angles * angleToSteps variable
            theta2Array[positionsCounter] = data[3] * theta2AngleToSteps;
            phiArray[positionsCounter] = data[4] * phiAngleToSteps;
            zArray[positionsCounter] = data[5] * zDistanceToSteps;
            gripperArray[positionsCounter] = data[6];
            positionsCounter++;
        }
        // clear data
        if (data[0] == 2)
        {
            // Clear the array data to 0
            memset(theta1Array, 0, sizeof(theta1Array));

```

```

int phiArray[100];
int zArray[100];
int gripperArray[100];
int positionsCounter = 0;
void setup() {
  Serial.begin(115200);

  pinMode(limitSwitch1, INPUT_PULLUP);
  pinMode(limitSwitch2, INPUT_PULLUP);
  pinMode(limitSwitch3, INPUT_PULLUP);
  pinMode(limitSwitch4, INPUT_PULLUP);

  pinMode(stepper1_step, OUTPUT); pinMode(stepper1_dire, OUTPUT);
  pinMode(stepper2_step, OUTPUT); pinMode(stepper2_dire, OUTPUT);
  pinMode(stepper3_step, OUTPUT); pinMode(stepper3_dire, OUTPUT);
  pinMode(stepper4_step, OUTPUT); pinMode(stepper4_dire, OUTPUT);
  pinMode(habi, OUTPUT);

  gripperServo.attach(A0, 600, 2500);
  // initial servo value - open gripper
  data[6] = 20;
  gripperServo.write(data[6]);
  delay(1000);
  homing();
}

void loop() {
  if (Serial.available())
  {
    ReadData();
    if (data[0] == 1)
    {
      theta1Array[positionsCounter] = data[2] * theta1AngleToSteps; //store the
      values in steps = angles * angleToSteps variable
      theta2Array[positionsCounter] = data[3] * theta2AngleToSteps;
      phiArray[positionsCounter] = data[4] * phiAngleToSteps;
      zArray[positionsCounter] = data[5] * zDistanceToSteps;
      gripperArray[positionsCounter] = data[6];
      positionsCounter++;
    }
    // clear data
    if (data[0] == 2)
    {
      // Clear the array data to 0
      memset(theta1Array, 0, sizeof(theta1Array));
    }
  }
}

```

```

        memset(theta2Array, 0, sizeof(theta2Array));
        memset(phiArray, 0, sizeof(phiArray));
        memset(zArray, 0, sizeof(zArray));
        memset(gripperArray, 0, sizeof(gripperArray));
        positionsCounter = 0;
    }
    // If RUN button is pressed
    while (data[1] == 1)
    {
        wait=(MaxSpeed-data[7])+Minwait;
        // execute the stored steps
        for (int i = 0; i <= positionsCounter - 1; i++)
        {
            if (data[1] == 0)
                break;
            wait=(MaxSpeed-data[7])+Minwait;
            SteptoStep(theta1Array[i],theta2Array[i],phiArray[i],zArray[i]);
            if (i == 0)
                gripperServo.write(gripperArray[i]);
            else if (gripperArray[i] != gripperArray[i - 1])
            {
                gripperServo.write(gripperArray[i]);
                delay(800); // wait 0.8s for the servo to grab or drop - the servo is
slow
            }

            //check for change in speed and acceleration or program stop
            if (Serial.available())
            {
                ReadData();
                if (data[1] == 0)
                    break;
                // change speed and acceleration while running the program
                wait=(MaxSpeed-data[7])+Minwait;
            }
        }
    }

    stepper1Position = data[2] * theta1AngleToSteps;
    stepper2Position = data[3] * theta2AngleToSteps;
    stepper3Position = data[4] * phiAngleToSteps;
    stepper4Position = data[5] * zDistanceToSteps;

    wait=(MaxSpeed-data[7])+Minwait;

```

```

    SteptoStep(stepper1Position,stepper2Position,stepper3Position,stepper4Position);
    delay(100);
    gripperServo.write(data[6]);
    delay(300);
}

void MoveTo(int stepperPin_,int AddressPin_,int stepNumber,int enable) {
    digitalWrite(enable, LOW); // Habilita el Driver
    if(stepNumber>=0){
        digitalWrite(AddressPin_, LOW); // direccion de giro 1
        for(int i=0;i<stepNumber;i++){ // da pasos por un tiempo
            digitalWrite(stepperPin_, HIGH);
            delayMicroseconds(wait);
            digitalWrite(stepperPin_, LOW);
            delayMicroseconds(wait);
        }
    }
    else{
        stepNumber=stepNumber*-1;
        digitalWrite(AddressPin_, HIGH); // direccion de giro 2
        for(int i=0;i<stepNumber;i++){ // da pasos por un tiempo
            digitalWrite(AddressPin_, HIGH);
            delayMicroseconds(wait);
            digitalWrite(AddressPin_, LOW);
            delayMicroseconds(wait);
        }
    }
    digitalWrite(enable, HIGH); // quita la habilitacion del Driver
}

```

```

void Step(int stepperPin_,int AddressPin_,int AddressSign,int enable)
{
    digitalWrite(enable, LOW);
    if(AddressSign>=0)
    {
        digitalWrite(AddressPin_, LOW);
        digitalWrite(stepperPin_, HIGH);
        delayMicroseconds(wait);
        digitalWrite(stepperPin_, LOW);
        delayMicroseconds(wait);
    }
    else
    {
        digitalWrite(AddressPin_, HIGH);
        digitalWrite(stepperPin_, HIGH);
        delayMicroseconds(wait);
        digitalWrite(stepperPin_, LOW);
        delayMicroseconds(wait);
    }
    digitalWrite(enable, HIGH);
}

void SteptoStep(int stepper1Position,int stepper2Position,int
stepper3Position,int stepper4Position)
{
    while ( ((stepper1Position-stepper1CurrentPosition)!=0) ||
            ((stepper2Position-stepper2CurrentPosition)!=0) ||
            ((stepper3Position-stepper3CurrentPosition)!=0) ||
            ((stepper4Position-stepper4CurrentPosition)!=0))
    {
        if((stepper1Position-stepper1CurrentPosition)!=0)
        {
            if((stepper1Position-stepper1CurrentPosition)<0)
            {
                Step(stepper1_step,stepper1_dire,-1,habi);
                stepper1CurrentPosition--;
            }
            else
            {
                Step(stepper1_step,stepper1_dire,1,habi);
                stepper1CurrentPosition++;
            }
        }
        if((stepper2Position-stepper2CurrentPosition)!=0)
        {

```

```
if((stepper2Position-stepper2CurrentPosition)<0)
{
    Step(stepper2_step,stepper2_dire,-1,habi);
    stepper2CurrentPosition--;
}
else
{
    Step(stepper2_step,stepper2_dire,1,habi);
    stepper2CurrentPosition++;
}
}

if((stepper3Position-stepper3CurrentPosition)!=0)
{
    if((stepper3Position-stepper3CurrentPosition)<0)
    {
        Step(stepper3_step,stepper3_dire,-1,habi);
        stepper3CurrentPosition--;
    }
    else
    {
        Step(stepper3_step,stepper3_dire,1,habi);
        stepper3CurrentPosition++;
    }
}
if((stepper4Position-stepper4CurrentPosition)!=0)
{
    if((stepper4Position-stepper4CurrentPosition)<0)
    {
        Step(stepper4_step,stepper4_dire,-1,habi);
        stepper4CurrentPosition--;
    }
    else
    {
        Step(stepper4_step,stepper4_dire,1,habi);
        stepper4CurrentPosition++;
    }
}
}
```

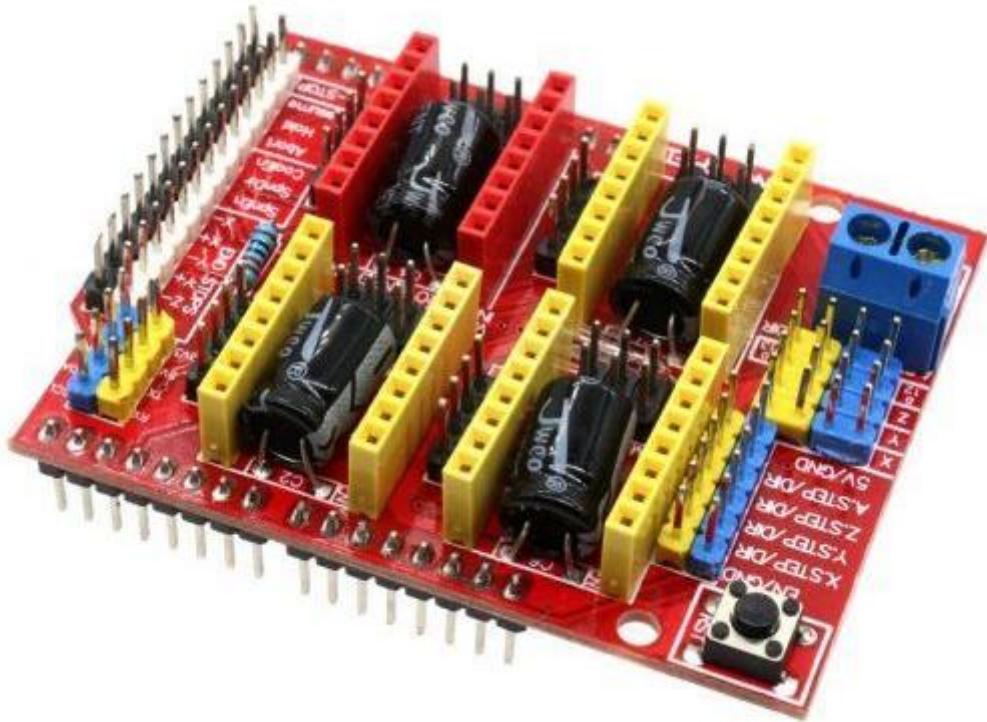
```

void homing() {
    while
    (!(digitalRead(limitSwitch4))||!(digitalRead(limitSwitch3))||!(digitalRead(limits
witch2))||!(digitalRead(limitSwitch1))) {
        if(!(digitalRead(limitSwitch1))){
            Step(stepper1_step,stepper1_dire,-1,habi);
        }
        else{
            stepper1CurrentPosition=0;
        }
        if(!(digitalRead(limitSwitch2))){
            Step(stepper2_step,stepper2_dire,-1,habi);
        }
        else{
            stepper2CurrentPosition=0;
        }
        if(!(digitalRead(limitSwitch3))){
            Step(stepper3_step,stepper3_dire,-1,habi);
        }
        else{
            stepper3CurrentPosition=0;
        }
        if(!(digitalRead(limitSwitch4))){
            Step(stepper4_step,stepper4_dire,-1,habi);
        }
        else{
            stepper4CurrentPosition=0;
        }
    }
}

void ReadData(){
    content = Serial.readString(); // Read the incoming data from Processing
    // Extract the data from the string and put into separate integer variables
    (data[] array)
    for (int i = 0; i < 10; i++) {
        int index = content.indexOf(","); // locate the first ","
        aux=content.substring(0, index);
        data[i] = aux.toInt(); //Extract the number from start to the ","
        content = content.substring(index + 1); //Remove the number from the string
    }
}

```

## Shield CNC V3



El CNC SHIELD está pensado para controlar 3 motores paso a paso mediante la adición de 3 drivers A4988 (no los incluye) de precisión para el manejo de máquinas CNC de control numérico y similares con control de 3 ejes y no demasiada potencia.

Este **Shield CNC** utiliza los siguientes pines:

8 – EN (conductor del motor de pasos, bajo activo)

7 – Z. DIR (control de la dirección del eje Z)

6 – Y. DIR (dirección del eje y de control)

5 – X. DIR (eje X de control de dirección)

4 – Z. PASO (eje Z de control paso a paso)

3 – Y. PASO (eje de control paso a paso)

2 – X. PASO (eje X de control paso a paso)

Compatible con GRBL 0.8c. (Firmware Open Source para Arduino que convierte G-code a instrucciones para motores PAP)

Soporte para 4 ejes (X, Y, Z , A)

2 conexiones para finales de carrera para cada eje (6 en total)

Salida “Spindle enable” y “direction”

Salida “Coolant enable”

Compatible con Pololu A4988 y DRV8825

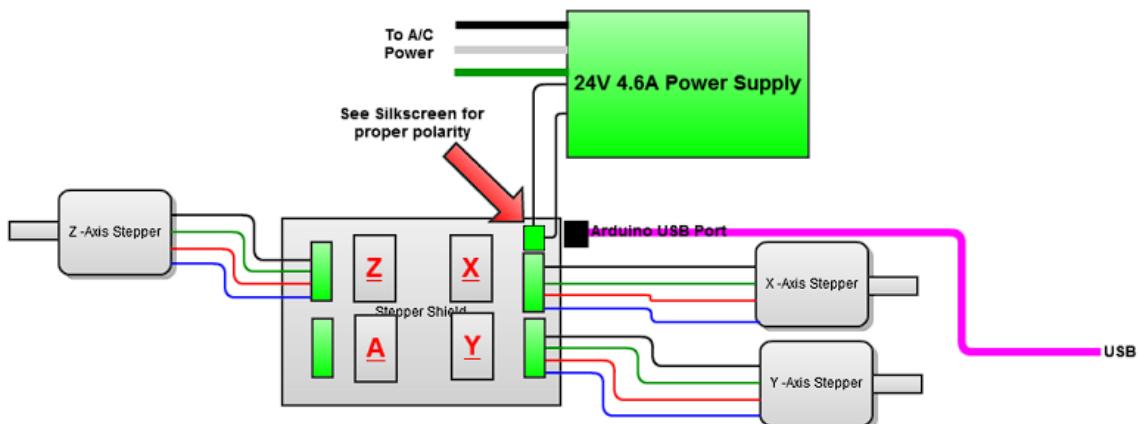
Jumpers para control de micro-stepping (Los controladores como el DRV8825 soportan hasta 1/32 para más precisión)

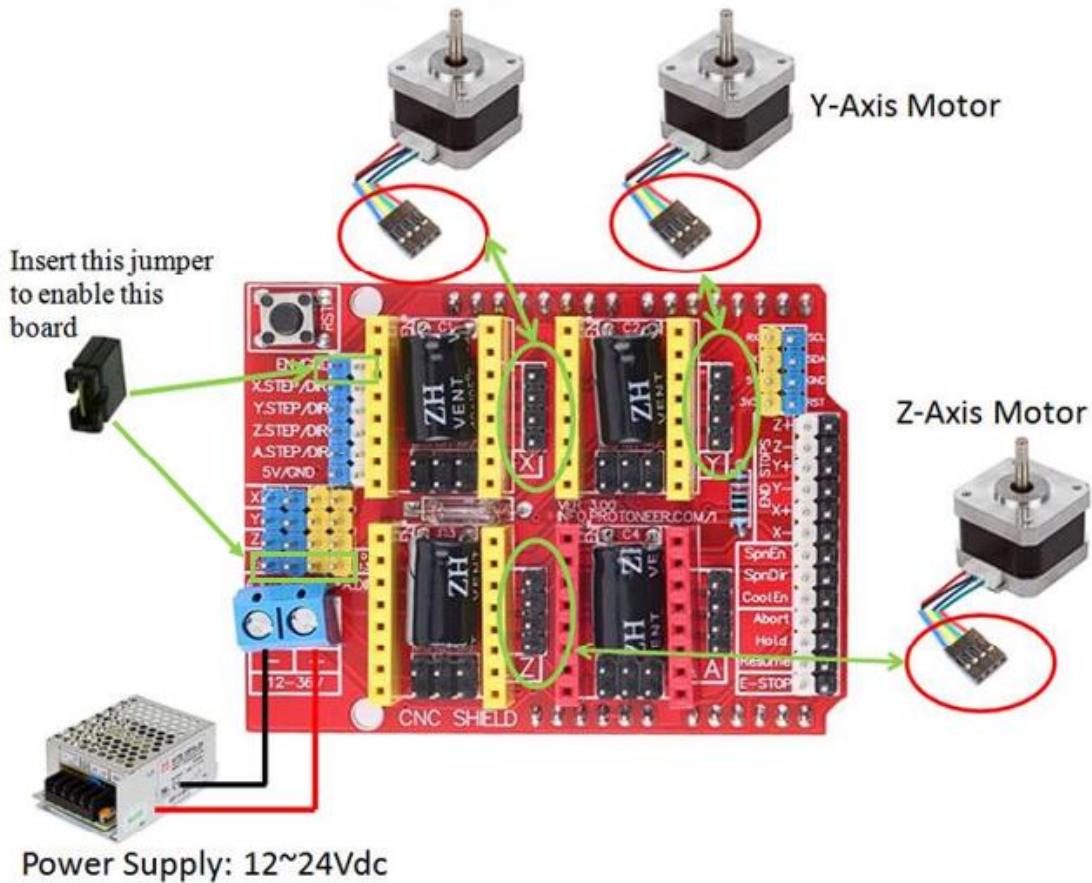
Diseño compacto

Los motores pueden ser conectados con bornes tipo Molex de 4 pines

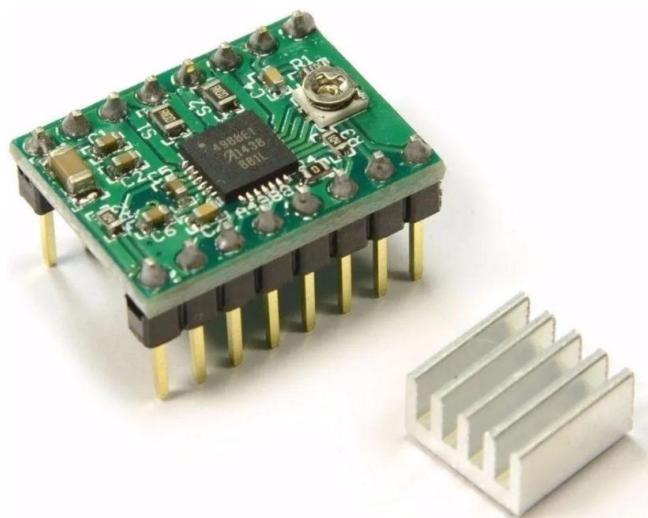
Alimentación: 12-36V DC. (Dependiendo de los controladores utilizados)

Conecte el motor paso a paso a la placa CNC Shield como se muestra en el siguiente diagrama de bloques. Del escudo CNC conectado al motor de 3 pasos:





**Driver Motor A4988**



El Stepper driver **Pololu A4988** es un complemento necesario para mover tus motores paso a paso mediante la **RAMPS 1.4**, ya que necesitas uno por cada motor que quieras manejar.

Opera desde 8 V a 35 V y puede entregar hasta aproximadamente 1 A por fase sin un disipador de calor (puede manejar hasta 2 A con un buen disipador). Cuenta con potenciómetro que permite ajustar la corriente máxima de salida y utilizar tensiones superiores a la tensión nominal para lograr mayores tasas de paso, además lleva un control inteligente que selecciona automáticamente el modo de reducción de la corriente correcta.

Diseñados para soportar las cargas de corriente que el manejo de motores de una cierta potencia como los **NEMA 17** requieren. Incluye disipador de calor.

*Características:*

Dimensiones: 15 mm x 20 mm

Peso: 1.3 Gramos

Máximo Voltaje de operación: 8 V

Mínimo Voltaje de operación: 35 V

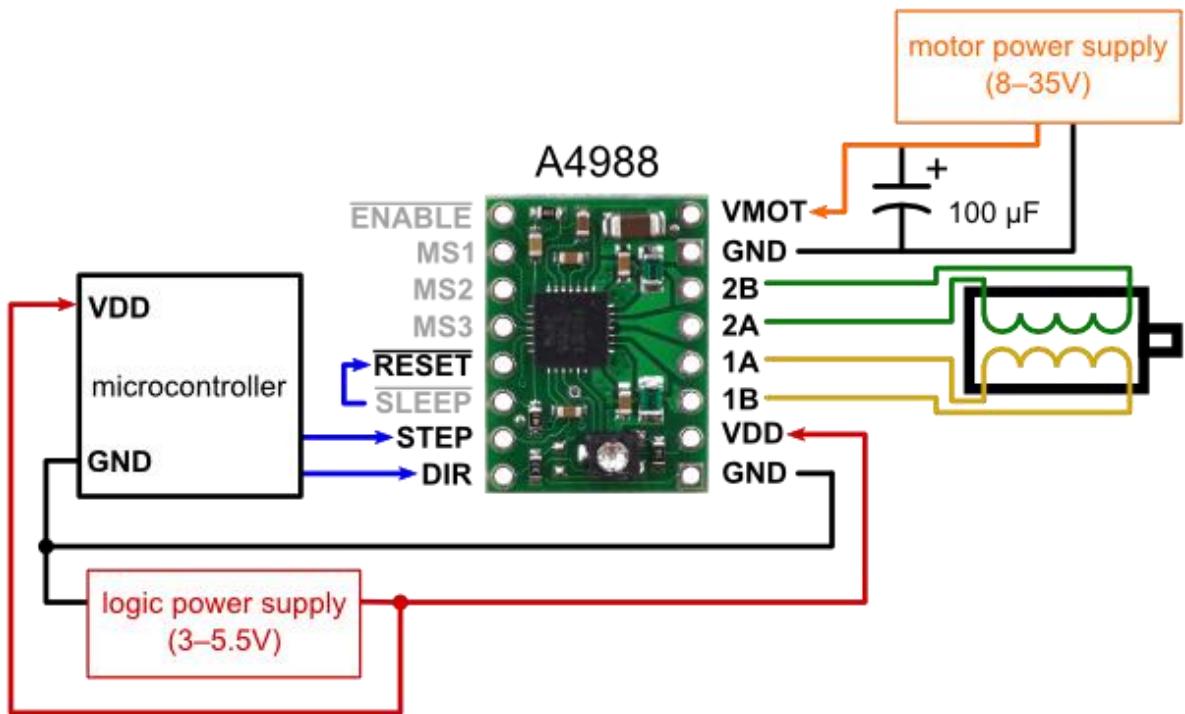
Corriente Continua de operación: 1 A

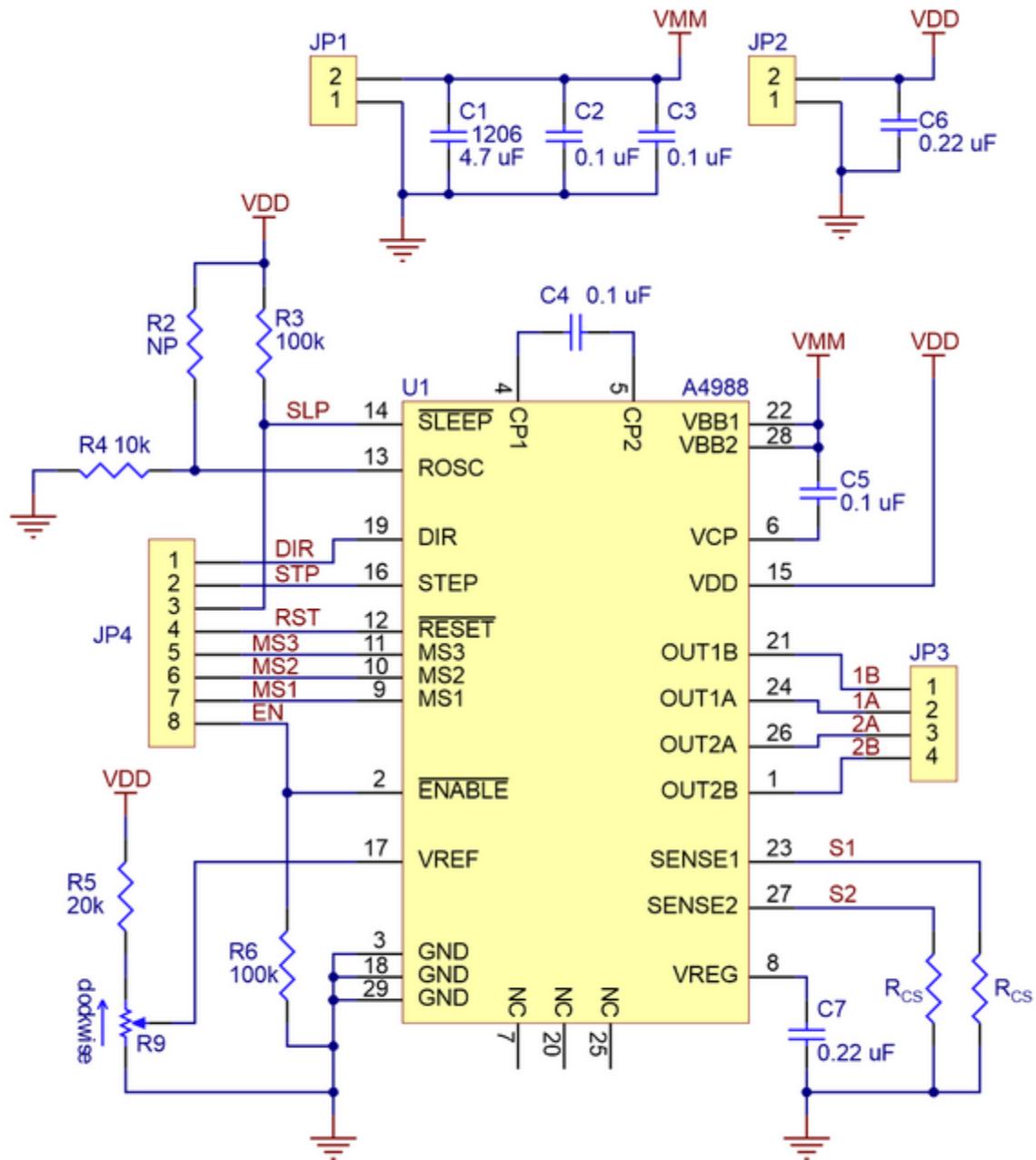
Máximo corriente de operación: 2 A

Máximo Voltaje logico: 3 V

Máximo Voltaje de logico: 5.5 V

Resolución de micropasos: Total, 1/2, 1/4,  
1/8, 1/16





$R_{CS}$  is 50mΩ for units with green resistors and 68mΩ for units with white resistors

## Nema 17 Motor a Pasos 17HS8401 1.8 A



Motor a pasos Nema 17 de 1.8Amp y 5.5Kg/cm es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que permite girar una cantidad de grados para controlar cualquier movimiento paso a paso con una rotación mínima de 1.8 grados. Este motor a pasos Nema 17 17HS8401 es de tipo bipolar, cuenta con cuatro cables de salida para conectar las dos bobinas que componen a este motor.

Este motor a pasos Nema 17 17HS8401 es ideal para construir mecanismos en donde se requieran movimientos muy precisos, es muy utilizado en impresoras 3D, Mini CNC, proyectos de robótica, iluminación de escenario, grabado láser, equipo de automatización, equipo no estándar, máquina de colocación para controlar cualquier movimiento paso a paso.

### *Ventajas:*

Bajo nivel de ruido

Bajo calor

Operación suave

Buen rendimiento de aceleración

Contamos con más motores nema solo tienes que entrar en el apartado de Robótica y seleccionar Motores.

### *Especificaciones y características*

Tipo: Motor a pasos bipolar

Nombre: Nema 17

No. del modelo: 17HS8401

Peso: 370g

Tamaño: 42 x 42 x 48 mm

Incluye: Cable 4 hilos de 100 cm

Voltaje nominal: 12V ~ 24V DC, (Voltaje recomendado 12V)

Rango de voltaje de suministro lógico: 3.3V – 5V

Corriente: 1.8 A / Fase

Fase: 2 fases

Resistencia / Fase: 1.8 Ω

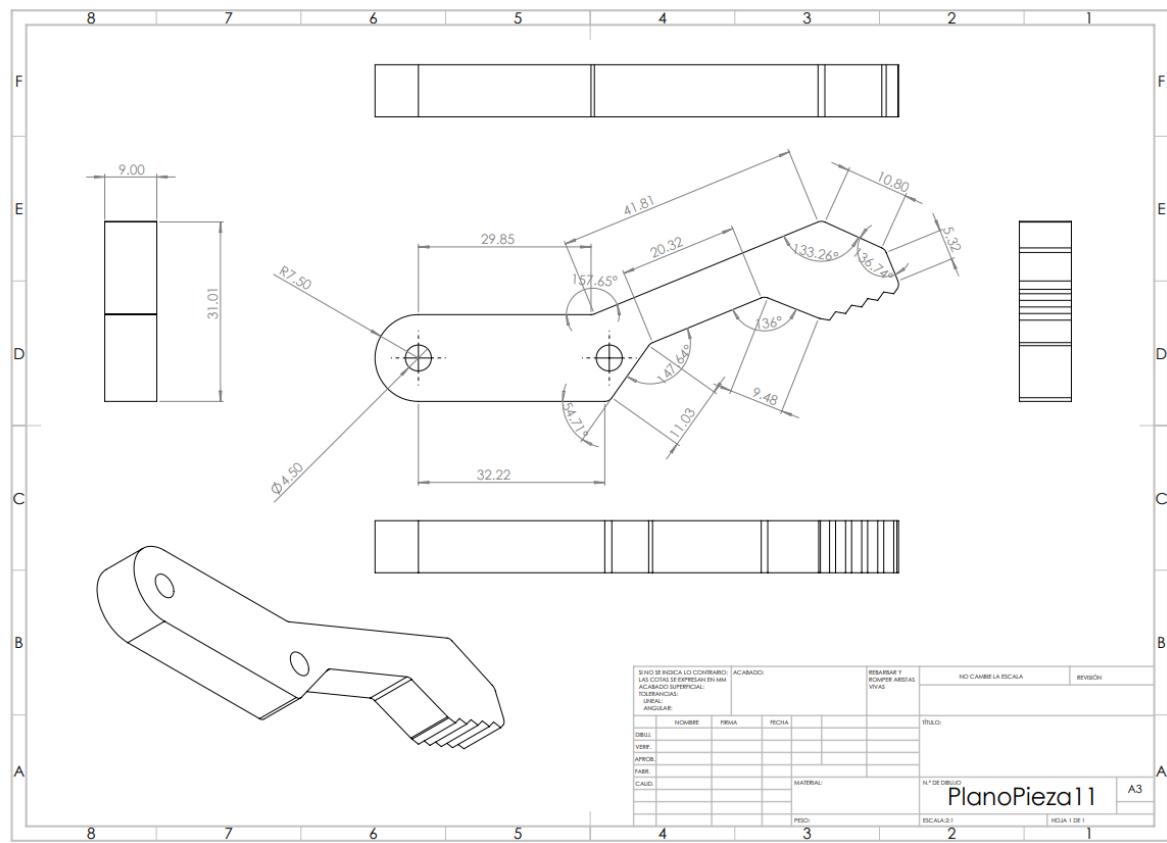
Inductancia: 3.2mH

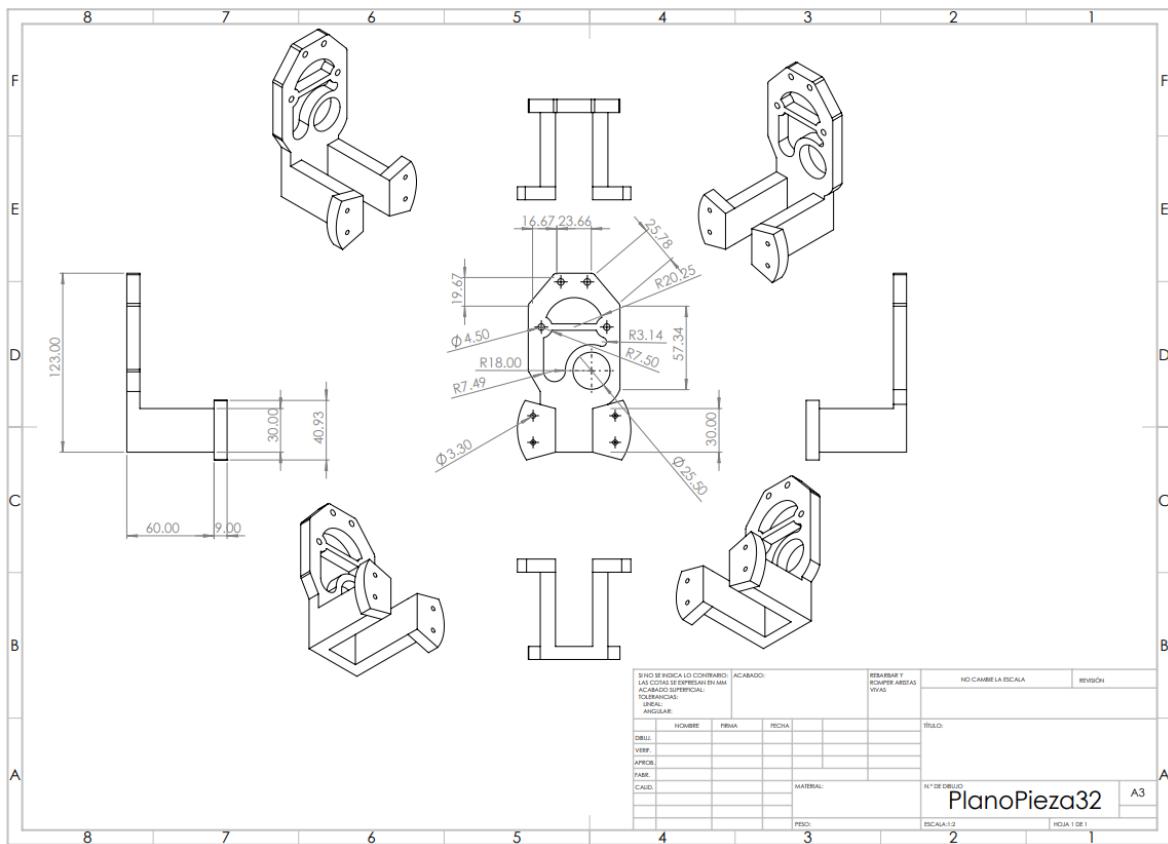
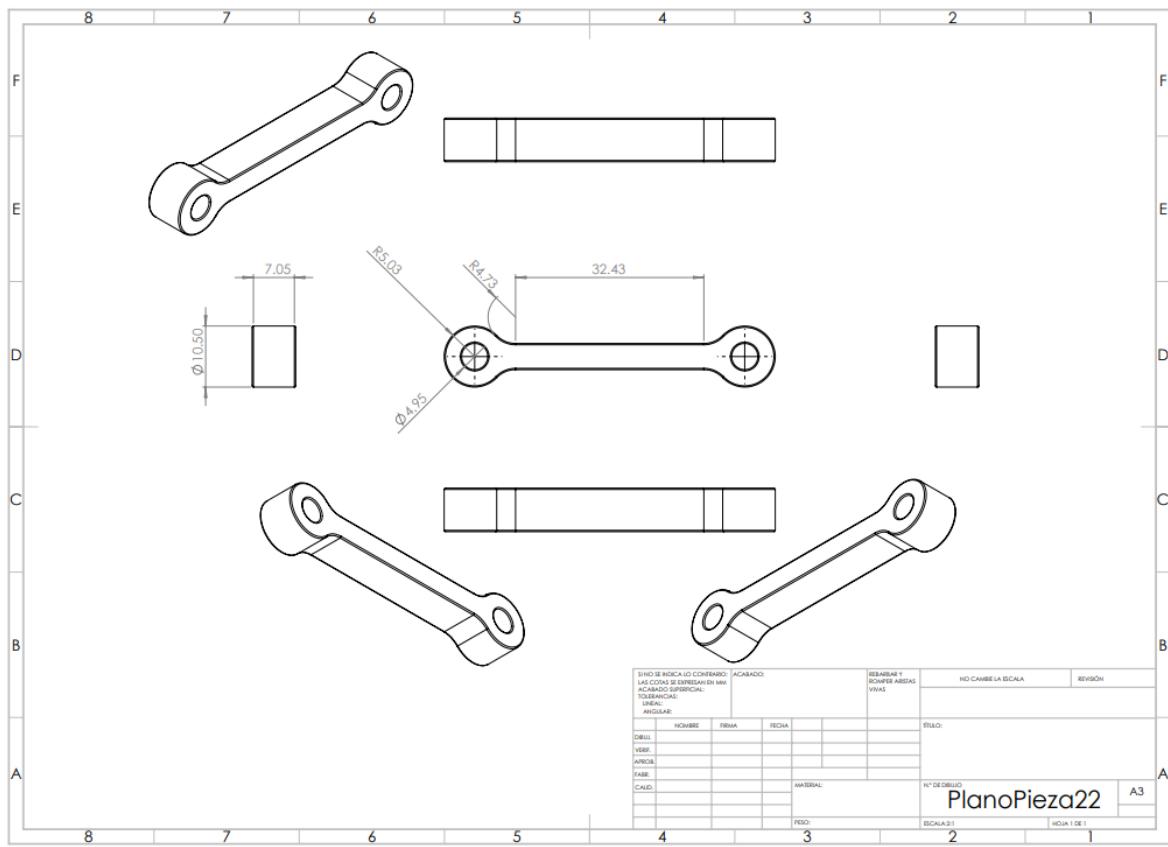
Sosteniendo Torque: 55 N.cm

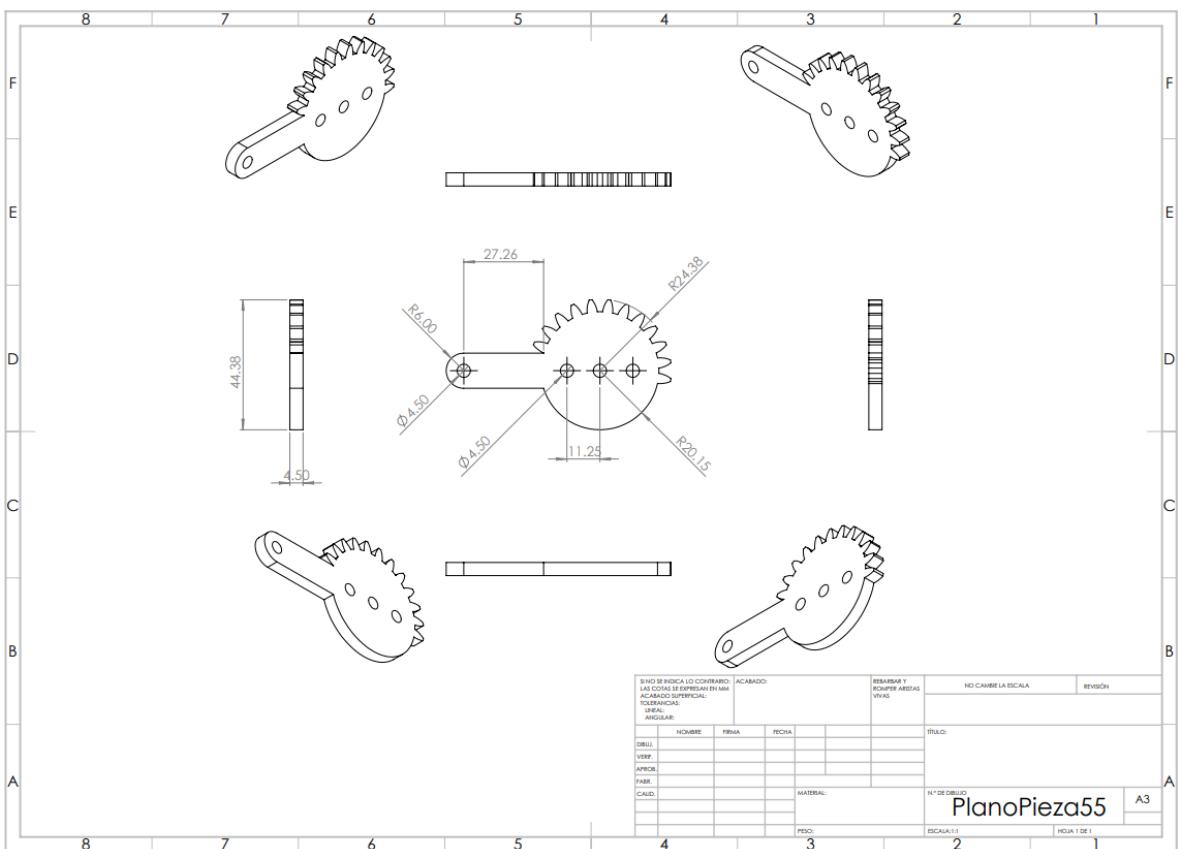
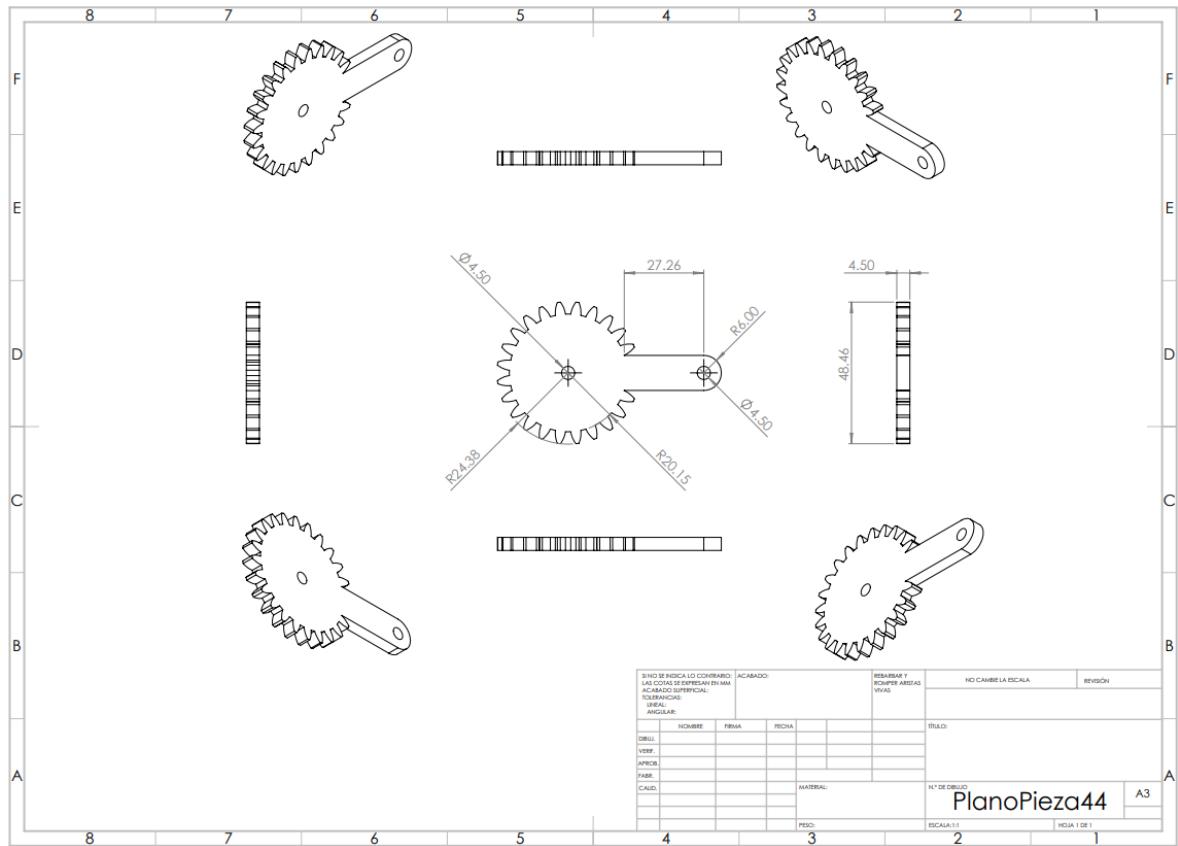
Ángulo de paso:  $1.8 \pm 5\%$  / PASO

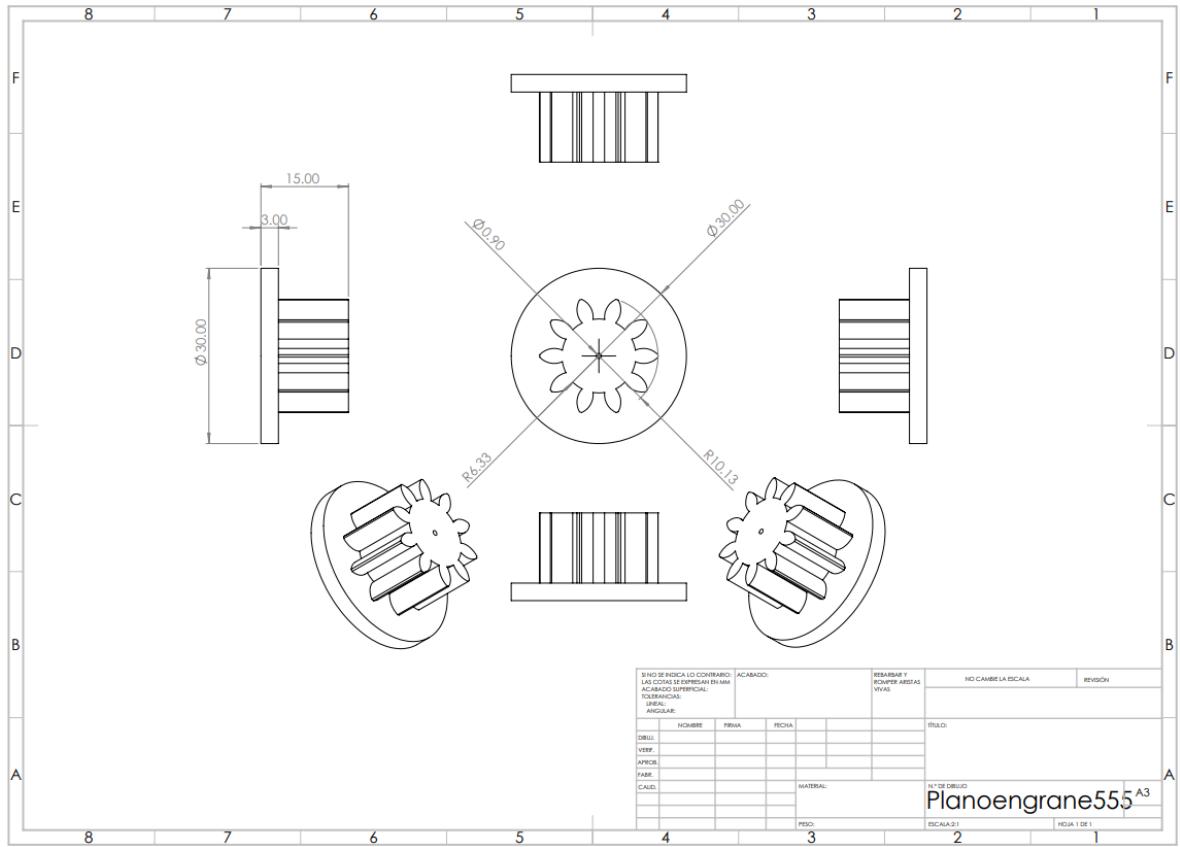
Eje: Tipo D 22 x  $\Phi$  5mm

# PLANOS









## Código de configuración

### Principal

```
1 //Funcion principal
2 //      src>Principal
3 import React from 'react'
4 import ReactDOM from 'react-dom/client'
5 import { Scara } from './Scara'
6
7 ReactDOM.createRoot(document.getElementById('root')).render(
8     <React.StrictMode>
9         <Scara />
10    </React.StrictMode>,
11 )
```

### Scara

```
1 //Scara element
2 //      src>Scara
3 import { AppRouter } from "./router/AppRouter"
4 import { AppTheme } from "./theme/AppTheme"
5
6 export const MecatronicApp = () => {
7     return (
8         <>
9             <AppTheme>
10                 <AppRouter/>
11             </AppTheme>
12         </>
13     )
14 }
```

## AppRouter

```
1 // Ruta padre de la aplicacion
2 //      src>router>AppRouter
3 import { BrowserRouter, Route, Router, Routes } from "react-
   router-dom";
4 import { WebRoutes } from "../web/routes/WebRoutes";
5
6 export const AppRouter = () => {
7
8     return (
9             <BrowserRouter>
10                <Routes>
11                    <Route path="/" element={<
12                      WebRoutes/>}/>
13                </Routes>
14            </BrowserRouter>
15        )
16    }
17 }
```

## Data validator

```
1 // objeto que se invoca para realizar la conexion via serial
2 // src>data>serialObject
3 export class SerialObject{
4
5     constructor(){
6         this.port = null;
7         this.reader = null;
8         this.writer = null;
9         this.isOpen = false;
10        this.isConected = false;
11        this.baud = 9600;
12        //this.positionArray = [];
13    }
14
15    resetConfiguration = () => {
16        try {
17            if(this.writer !== null) { this.writer.releaseLock()
18                ;
19            }
20            if(this.reader !== null) { this.reader.releaseLock()
21                ;
22            }
23            //this.port.close();
24            port = null;
25            reader = null;
26            writer = null;
27            isOpen = false;
28            isConected = false;
29            return "Reset succesfull"
30        }catch( e ) {
31            return e
32        }
33    }
34
35    setBaud = ( newBaud = 9600) => {
36        this.baud = newBaud;
```

```
33     }
34
35     selectPort = async() => {
36         if ('serial' in navigator) {
37             const port = await navigator.serial.
38                 requestPort();
39             await port.open({ baudRate: this.baud });
40             const reader = port.readable.getReader();
41             const writer = port.writable.getWriter();
42
43             const portObject = {port, reader, writer}
44             return portObject;
45         } else {
46             console.error('The Web serial API doesn\'t
47             seem to be enabled in your browser.');
48         }
49     }
50
51     setConfig = (portObject) => {
52         try{
53             this.port = portObject.port;
54             this.reader = portObject.reader;
55             this.writer = portObject.writer;
56             this.isOpen = true;
57             this.isConected = true;
58         }catch(e){
59             console.log(e)
60         }
61     }
62
63     escribe = (data = "M1,0") => {
64         try{
65             const encoder = new TextEncoder();
66             const dataArrayBuffer = encoder.encode
67                 (data);
68             console.log('serialObject > escribe :
69                 ' + data)
70             this.writer.write(dataArrayBuffer);
71         }catch(e){
72             console.log('serialObject > escribe :
73                 ','
74                 Error - Cannot read property of null')
75         }
76     }
77
78     mueveA = (motor = 1, value = 5) => {
79         this.escribe("M" + motor.toString() +"," + value.
80             toString())
81         console.log('movido a : '+value)
```

```
75     } }  
76 }
```

## Scara motors information

```
1 //Informacion los motores del robot scara (# de motor, nombre,  
2   limites y posicion home)  
3 // src>data>motors>scara_motors  
4 export const scara_position_information = [  
5     {id: 2, text:"Revoluta 1", min: 0, max: 80, default:  
6         0},    //Revoluta del extremo  
7     {id: 3, text:"Revoluta 2", min: 0, max: 30, default:  
8         0},  
9     {id: 4, text:"Revoluta 3", min: 0, max: 325, default:  
10        0},  
11     {id: 5, text:"Posicion z", min: 0, max: 50, default:  
12        0},  
13     {id: 6, text:"Gripped ", min: 1, max: 70, default:80},  
14     {id: 7, text:"Velocidad ", min: 1, max: 9000, default:  
15        9000},  
16 ]
```

## useSerial

```
1 // Se encarga de gestionar la logica de la conexion serial  
2 // src>hooks>useSerial  
3 import React, { useEffect, useState } from "react";  
4 import { SerialObject } from "../data/serialObject";  
5 import { createData } from "../data/dataValidators";  
6  
7 const historyDefault = createData('',">>>","",0);  
8  
9 export const useSerial = (serialObject = new SerialObject(),  
10   configuration={ robot: 'scara', baud: 115200,  
11       information: [], prefix:"", postfix:""}  
12   ) => {  
13  
14   //VARIABLES:  
15   -----  
16  
15   const [config, setConfig] = useState(configuration) //  
16   Configuracion del robot  
16   const {robot,baud,information:motorsInformation } = config;
```

```

18 // Variables related to internal configuration and state of
19 // serial device:
20
21 // Variables related to positions:
22 const [isConnected, setIsConnected] = useState(false);
23
24 // Establece las
25 // posiciones a enviar
26 // motorsInformation.map( information => information.
27 // default )
28 );
29
30 //const [direct, setDirect] = useState({x:0,y:0,z:0})
31
32 const prefix = config.prefix;
33 const postfix = config.postfix;
34 const positionsInformation = [positions, prefix, postfix]
35
36 // Variables related to memory and others:
37 const [history, setHistory] = useState([historyDefault]);
38 // Maneja el historial
39
40 // use effect configuration
41 useEffect(() => {
42     const newPositionInformation = config.information.map(
43         information => information.default )
44
45         if(robot == 'scara'){
46             setPositions(([0,0]).concat(newPositionInformation)
47                 .concat([0]))
48         } else {
49             setPositions(newPositionInformation)
50         }
51
52 }, [config])
53
54 // CONFIGURATION METHODS:
55 -----
56
57
58 // Set the variables related to configuration and
59 // state of serial device:
60 const setConfiguration = async() => {
61     if ("serial" in navigator) { // The Web
62         Serial API is supported.
63         addToHistory('Conexion iniciada...')
64         console.log('useSerial >
65             setConfiguration', 'Conexion

```

```
                iniciada')
54     try{
55         serialObject.setBaud(baud);
56         const p0 = await serialObject.
57             selectPort();
58
59         serialObject.setConfig(p0)
60         serialObject.escribe("hi")
61         addToHistory('Conexion
62             establecida')
63         setIsConected(true)
64         addToHistory('Puerto abierto')
65     }catch(e){
66         addToHistory(e)
67     }
68
69 } else {
70     console.log('Serial not soported')
71     addToHistory('Su dispositivo o
72         navegador no cuenta con conexion
73             serial...')

74
75     // Reset the configuration - NOT WORKING
76     const resetConfiguration = () => {
77         const resetResult = serialObject.
78             resetConfiguration()
79         if(resetResult === "Reset succesfull") {
80             setIsConected(false)
81         }
82
83         // Delete and set configuration
84         const deleteAndSetConfiguration = () => {
85             resetConfiguration()
86             addToHistory('Serial configuration deleted')
87             console.log('useSerial >
88                 deleteAndSetConfiguration','Serial
89                     configuration deleted')
90             setConfiguration()
91         }
92
93         // Check if serial is writable
94         const isPortOpen = () => {
95             try{
96                 let isPortOpen = false;
```

```

94         if(serialObject.port.writable.locked
95             == true){
96             isPortOpen = true;
97         }
98     //if(serialObject.)
99     return isPortOpen;
100 }catch(e) {
101     return false;
102 }
103
104 // Send a string through device using serialObject
105 const writte = (command = '') => {
106     //console.log(serialObject)
107     serialObject.escribe(command)
108 }
109
110
111 // MEMORY METHODS:
-----
```

---

```

112
113 // Add new entry to history
114 const addToHistory = (newEntry = "") => {
115     let dataHistoryObject
116     if(typeof(newEntry) === "string") {
117         dataHistoryObject = createData(
118             newEntry,>>,"",0)
119     } else {
120         dataHistoryObject = createData("No se
121             pudo realizar la accion",>>,"",0)
122     }
123     setHistory(history => [...history,
124             dataHistoryObject]); //Esta funciona
125 }
126
127
128 // Reset history
129 const resetHistory = () => {
130     setHistory([createData('','>>','','0)])
131 }
```

---

```

132
133 // POSITION METHODS:
-----
```

---

```

134
135 // If the serial device have motors and the chip is
136 // configured correctly it will send the command to
137 // move a specific motor to a specific position
```

```

132  const modifyPosition = (position = 0, newValue = 0) =>
133  {
134      //console.log('ModifyPosition, baud: ' +
135      //serialObject.baud)
136
137      const modify = () => {
138          let psto = positions;
139          psto[position] = newValue;
140          setPositions(psto);
141          const message = prefix + psto.toString()
142              () +postfix ;
143          write(message);
144      }
145
146      if(isConected){
147          const informationMotorPosition =
148              config.information.find( info =>
149                  info.id == position)
150          if(informationMotorPosition != undefined){
151              const minimo =
152                  informationMotorPosition.min;
153              const maximo =
154                  informationMotorPosition.max;
155
156              if ( (newValue >= minimo) || (newValue
157                  <= maximo) ){
158                  modify()
159              } else {
160                  modify()
161              }
162          }
163
164      const calculaCinematicaDirectaScara = () => {
165          const l1 = 20
166          const l2 = 8
167          const theta1 = positions[2]
168          const theta2 = positions[3]
169          const theta1F = theta1*(Math.PI/180)
170          const theta2F = theta2*(Math.PI/180)
171          const xP = Math.round( l1*Math.cos(theta1F) + l2*
172              Math.cos(theta1F+theta2F) )
173          const yP = Math.round( l1*Math.sin(theta1F) + l2*
174              Math.sin(theta1F+theta2F) )
175
176
177          return {xP,yP}

```

```

170     }
171
172
173     return {serialHookObject:{ setConfiguration,
174         resetConfiguration, deleteAndSetConfiguration,
175             isPortOpen, addToHistory,resetHistory, write,
176                 modifyPosition,
177                 isConected, serialObject, history, positions,
178                     positionsInformation,
179                         setConfig, } }
180
181 }
```

## Elementos de la interfaz

### Botón de descarga

```

1 // Icono de descarga
2 // src>interface>Buttons>DownloadButton
3 import React from 'react'
4 import FileDownloadIcon from '@mui/icons-material/FileDownload
5   ';
6 import { IconButton, Link } from '@mui/material';
7
8 const linkD = 'https://www.utm.mx/~minirobotica/formatos/
9   Formato_Registro_Talleres.docx';
10
11 export const DownloadButton = ({link = linkD}) => {
12     return (
13         <>
14             <IconButton
15                 href={link}
16
17                 >
18                     <FileDownloadIcon color="primary"/>
19                 </IconButton>
20             </>
21     )
22 }
```

### Menú de interfaz

```

1 // Barra de navegacion de interfaz
2 // src>interface>NavBar>LineMenu
```

```
3 import * as React from 'react';
4 import AppBar from '@mui/material/AppBar';
5 import { Box, Link } from '@mui/material';
6 import Toolbar from '@mui/material/Toolbar';
7 import IconButton from '@mui/material/IconButton';
8 import Typography from '@mui/material/Typography';
9 import Menu from '@mui/material/Menu';
10 import MenuIcon from '@mui/icons-material/Menu';
11 import Container from '@mui/material/Container';
12 import Button from '@mui/material/Button';
13 import MenuItem from '@mui/material/MenuItem';
14 import { NavLink } from 'react-router-dom';
15 import Grid from '@mui/material/Grid';
16 import { pages } from './pages';

17
18 export const LineMenu = ({nameLogo='name logo', nameLogoSm="N
  L Sm"}) => {
19   const [anchorElNav, setAnchorElNav] = React.useState(null);
20
21   const handleOpenNavMenu = (event) => {
22     setAnchorElNav(event.currentTarget);
23   };
24
25   const handleCloseNavMenu = () => {
26     setAnchorElNav(null);
27   };
28
29   return (
30     <>
31       <AppBar position="static" color='primary'>
32         <Container maxWidth="xl" >
33           <Toolbar disableGutters>
34             {/*PANTALLA CHICA ----- */}
35             {/*NabMenu en pantalla Chica */}
36           <Box sx={{ flexGrow: 1, display: { xs: 'flex', md: 'none' } }}>
37             <IconButton
38               size="large"
39               aria-label="account of current user"
40               aria-controls="menu-appbar"
41               aria-haspopup="true"
42               onClick={handleOpenNavMenu}
43               color="inherit"
44             >
45               <MenuIcon />
46             </IconButton>
47             <Menu
48               id="menu-appbar"
```

```
49         anchorEl={anchorElNav}
50         anchorOrigin={{
51             vertical: 'bottom',
52             horizontal: 'left',
53         }}
54         keepMounted
55         transformOrigin={{
56             vertical: 'top',
57             horizontal: 'left',
58         }}
59         open={Boolean(anchorElNav)}
60         onClose={handleCloseNavMenu}
61         sx={{
62             display: { xs: 'block', md: 'none' },
63         }}
64         //style={{ textDecoration: 'none' }}
65     >
66     {pages.map((page) => (
67         <MenuItem key={page.direction} onClick={
68             handleCloseNavMenu} >
69             <NavLink
70                 to = {page.direction}
71             >
72                 <Typography
73                     variant='h6'
74                 >
75                     {page.label}
76                 </Typography>
77             </NavLink>
78         </MenuItem>
79     )));
80     </Menu>
81     /*Logo y Marca de pagina en pantalla Grande */
82     <Typography
83         variant="h6"
84         noWrap
85         component="a"
86         href=""
87         sx={{
88             mr: 2,
89             display: { xs: 'none', md: 'none' , sm:'flex'},
90             flexGrow: 1,
91             fontFamily: 'monospace',
92             fontWeight: 700,
93             letterSpacing: '.3rem',
94             color: 'inherit',
95             textDecoration: 'none',

```

```
96      })
97      > {nameLogo} </Typography>
98 /*Logo y Marca de pagina en pantalla Chica */
99 <Typography
100     variant="h6"
101     noWrap
102     component="a"
103     href=""
104     sx={ {
105       mr: 2,
106       display: { xs: 'flex', md: 'none' , sm:'none' },
107       flexGrow: 1,
108       fontFamily: 'monospace',
109       fontWeight: 700,
110       //fontSize:'10px',
111       letterSpacing: '.3rem',
112       color: 'inherit',
113       textDecoration: 'none',
114     } }
115   > {nameLogoSm} </Typography>
116 /*PANTALLA GRANDE ----- */
117 <Grid
118   container
119   spacing={0}
120   direction="column"
121   alignItems="center"
122   justifyContent="center"
123   sx={ {
124     display: { xs: 'none', md: 'flex' },
125   } }
126   //sx= {{alignItems: {xs:"center"}}}
127   //justifyContent="center"
128   >
129   <Grid item>
130     /*Menu en pantalla Grande */
131     <Box sx={{ flexGrow: 1, display: { xs: 'none', md: 'flex' } }}>
132       {pages.map((page) => (
133         <Button
134           key={page.direction}
135           onClick={handleCloseNavMenu}
136           sx={{ my: 2, color: 'white', display: 'block'
137             }}
138           >
139             <NavLink
140               to={page.direction}
141               //className={({ isActive }) =>
```

```
141         //isActive ? activeClassName :  
142         undefined  
143     //}  
144     >  
145     <Typography  
146         textAlign="center"  
147         sx={{ color: 'white', textDecoration  
148             : 'none', }}>  
149             {page.label}  
150         </Typography>  
151     </NavLink>  
152     </Button>  
153   ) ) }  
154 </Box>  
155 </Grid>  
156 </Grid>  
157 </Toolbar>  
158 </Container>  
159 </AppBar>  
160 </>  
161 );  
162 }
```

## Navbar

```
1 // Navbar de la aplicacion  
2 // src>interface>Navbar  
3 import * as React from 'react';  
4 import { Box, Grid } from '@mui/material';  
5 import { LineMenu } from './LineMenu';  
6 import Typography from '@mui/material/Typography';  
7  
8 const nameLogo = "Robot Serial Interface";  
9 const nameLogoSm = "RSI";  
10  
11 export const Navbar = () => {  
12     return (  
13         <>  
14             <Box sx={{ flexGrow: 1 } }>  
15  
16                 <Grid  
17                     container  
18                     spacing={0}  
19                     direction="column"  
20                     alignItems="center"
```

```

21     justifyContent="center">
22         <Typography
23             variant="h5"
24             noWrap
25             component="a"
26             href="/"
27             sx={{
28                 mr: 2,
29                 display: { xs: 'none', md: 'flex' },
30                 fontFamily: 'monospace',
31                 fontWeight: 700,
32                 letterSpacing: '.3rem',
33                 color: '#09417C',
34                 textDecoration: 'none',
35                 //background:'white',
36             } }
37         >
38             {nameLogo}
39         </Typography>
40     </Grid>
41     <LineMenu nameLogo={nameLogo} nameLogoSm={nameLogoSm} />
42 </Box>
43 </>
44 )
45 }
```

## Pages

```

1 // Paginas a mostrar en el Navbar
2 // src>interface>Navbar>pages
3 export const pages = [
4     {label:'Control',direction:'/control'},
5     {label:'Configuracion',direction:'/configuration'},
6     {label:'Instrucciones',direction:'/instructions'},
7 ]
```

## Selector

```

1 // Elemento para elegir entre opciones
2 // src>interface>Selectors>SimpleSelector
3 import * as React from 'react';
4 import InputLabel from '@mui/material/InputLabel';
5 import MenuItem from '@mui/material/MenuItem';
```

```

6 import FormControl from '@mui/material/FormControl';
7 import Select from '@mui/material/Select';
8
9 export const SimpleSelector = ({data = '', setData = () => {}, 
10   legend= "", selectInfo = [{value: 1, text:''}],}) => {
11   const handleChange = (event) => {
12     setData(event.target.value);
13   };
14
15   return (
16     <div>
17       <FormControl variant="standard" sx={{ m: 1, minWidth
18         : 120 }}>
19         <InputLabel id="demo-simple-select-standard-
20           label">{legend}</InputLabel>
21         <Select
22           labelId="demo-simple-select-standard-label"
23           id="demo-simple-select-standard"
24           value={data}
25           onChange={handleChange}
26           label="Age"
27           >
28             {
29               selectInfo.map( selectedItem=> (
30                 <MenuItem key={selectedItem.
31                   value} value={selectedItem.
32                     value}>
33                       {selectedItem.text}
34                     </MenuItem>
35               ))
36             }
37           </Select>
38         </FormControl>
39       </div>
40     );
41   }

```

## Boton de leer código

```

1 // Boton para subir archivo
2 // src>web>components>Buttons>FileUploader
3 import React from 'react';
4 import FolderOpenIcon from '@mui/icons-material/FolderOpen';
5 import { IconB } from './IconB';
6
7 export const FileUploader = ({setCode = () => {}}) => {

```

```

8
9   const hiddenFileInput = React.useRef(null);
10
11  const handleClick = event => {
12    hiddenFileInput.current.click();
13  };
14
15  const readFile = (e) => {
16    const file = e.target.files[0];
17
18    if(!file) return;
19
20    const fileReader = new FileReader();
21
22    fileReader.readAsText(file);
23
24    fileReader.onload = () => {
25      setCode(fileReader.result)
26      //console.log( fileReader.result )
27    }
28
29    fileReader.onerror = () => {
30      console.log('FileUploader > ', fileReader.error)
31    }
32  }
33
34  return (
35    <>
36      <IconB Icon = {FolderOpenIcon} action = {handleClick}
37        />
38
39      <input type="file"
40          ref={hiddenFileInput}
41          onChange={readFile}
42          style={{display:'none' }}>
43      />
44    );
45  };

```

## Boton con icono

```

1 // Boton que incluye un icono
2 // src>web>components>Buttons>IconB
3 import React from 'react';
4 import IconButton from '@mui/material/IconButton';

```

```

5 import HelpOutlineIcon from '@mui/icons-material/HelpOutline';
6 import { Typography } from '@mui/material';
7
8 const defaultAction = () => console.log('?') ;
9
10 export const IconB = ({action = defaultAction ,Icon =
11   HelpOutlineIcon, legend = ''}) => {
12
13   const mode = "white";
14
15   return (
16     <IconButton onClick={ () => action() }>
17       <Icon style={{ color: mode ==='dark' ?
18         'white' : 'black' }} />
19       {
20         legend
21         ?
22           <Typography variant='h6' color
23             ='black' >
24               <div> &nbsp; {legend
25                 }</div>
26             </Typography>
27             :
28               <></>
29         }
30       </IconButton>
31     )
32   }

```

## Editor de codigo

```

1 // Maneja el texto a escribir dentro del editor
2 // src>web>components>Devmode>CodeEditor
3 import React from "react";
4 import Editor from "@monaco-editor/react";
5
6 export const CodeEditor = ({code, setCode}) => {
7
8   const handleCodeChange = (e) => {
9     setCode(e);
10   }
11
12   return (
13     <div className="container">
14       <div className="row">
15         <Editor

```

```
16          height="50vh"
17          language="javascript"
18          value={"" || code}
19          onChange={(e) => handleCodeChange(e) }
20          theme={'dark'}
21        />
22      </div>
23    </div>
24  );
25 };
```

## Botones de editor de código

```
1 // Botones del editor de codigo
2 // src>web>components>Devmode>CodeEditor>EditorButtons
3 import React, { useState } from 'react'
4 import PlayCircleIcon from '@mui/icons-material/PlayCircle';
5 import AccessibilityIcon from '@mui/icons-material/
  Accessibility';
6 import StopIcon from '@mui/icons-material/Stop';
7 import FileDownloadIcon from '@mui/icons-material/FileDownload
  ';
8 import SkipNextIcon from '@mui/icons-material/SkipNext';
9 import SkipPreviousIcon from '@mui/icons-material/SkipPrevious
  ';
10 import ReplyAllIcon from '@mui/icons-material/ReplyAll';
11
12 import {saveAs } from "file-saver";
13 import { FileUploader } from '../../../../../Buttons/FileUploader';
14 import { IconB } from '../../../../../Buttons/IconB';
15
16 export const EditorButtons = ({
17   code="", setCode = () => {}, actionCommand = () => {}
18   , positionsInformation = [position=[0,0,0],prefix=""
19   ",postfix=""]
20 }) => {
21
22   const [positions, prefix, postfix] =
23     positionsInformation;
24   const [contador, setContador] = useState(100); //Para
25     linea de tiempo
26
27   const action = actionCommand; //NO ELIMINAR: controla
28     acciones a ejecutar en editor
29
30   //Ejecuta codigo dentro de editor
```

```
26     const evaluaFuncion = () => {
27         try{
28             console.log('Codigo escrito: ' + code)
29             eval(code);
30         }catch{
31             console.log('Error en evaluacion de codigo')
32         }
33     }
34
35     // Manda el array de posiciones de useSerial al editor
36     const sendPositionsToEditor = () => {
37         const newCommand = `action('${prefix+
38         positions.toString() + postfix}\')`;
39         const newcode = `setTimeout(function(){\n' +'\t' +
40             newCommand + '\n}, '+ contador + '); \n'
41         setContador(contador + 500);
42         setCode(code + '\n' + newcode);
43     }
44
45     const writeFile = () => {
46         const blob = new Blob([code], { type: 'text/plain;
47             charset=utf-8'});
48         const name = 'naab';
49         const extension = 'dimmex'
50         saveAs( blob, name + '.' + extension);///.txt');
51     }
52
53     return (
54         <>
55         <div className="row mx-auto">
56             <div className="col">
57                 <FileUploader setCode = {setCode} />
58                 <IconB Icon = {FileDownloadIcon} action = {
59                     writeFile}/>
60             </div>
61             <div className="col">
62                 <IconB Icon = {ReplyAllIcon} action={ () =>
63                     sendPositionsToEditor() }/>
64             </div>
65             <div className="col">
66                 <IconB Icon = {SkipPreviousIcon} />
67                 <IconB Icon = {PlayCircleIcon} action = {
68                     evaluaFuncion}/>
69                 <IconB Icon = {SkipNextIcon} />
70             </div>
71             <div className="col">
72                 <IconB Icon = {StopIcon} />
73             </div>
74         </div>
75     )
76 
```

```

68         <IconB Icon = {AccessibilityIcon} action =
69             {() => action('home')}/>
70         </div>
71     </div>
72   )
73 }
74 }
```

## Linea de comandos

```

1 // Linea de comandos de consola
2 // src>web>components>Devmode>Consola>CommandLine
3 import Table from '@mui/material/Table';
4 import TableBody from '@mui/material/TableBody';
5 import TableRow from '@mui/material/TableRow';
6 import TableCell from '@mui/material/TableCell';
7 import { EditableCell } from './EditableCell';
8
9 const languaje = 'spanish';
10
11 export const ComandLine = ({action}) => {
12
13   return (
14     <Table size="small" aria-label="simple table">
15       <TableBody key="bodyCommandLine">
16         <TableRow key={"rowCommandLine"}>
17           <TableCell key="comandTable">
18             <EditableCell
19               key={'comandLine'}
20               valueInitial={''}
21               actionOnSubmit={(c) =>
22                 action(c)}
23               autoset={true}
24               placeholder={languaje ===
25                 "spanish" ? "Ingrese un
26                 comando" : "Writte a
27                 command"}/>
28             </TableCell>
29         </TableRow>
30       </TableBody>
31     </Table>
32   )
33 }
```



```
        ))}
      </TableRow>
    </TableHead>
    <TableBody >
      {history
        .map((row) => {
          return (
            <TableRow key = {'consoleRow
              -${row.id}'}>
              <TableCell key
                ={`'
                  identacion-
                  ${row.id}`}
                align={"left"}
                style={{width: 1
                }}> {row.
                  identacion
                }<
                TableCell>
              <TableCell key
                ={`texto-${{
                  row.id}`}
                align={"left"}> {
                  row.texto
                }<
                TableCell>
              <TableCell key
                ={`estatus-
                  ${row.id}`}
                align={"left"}> {
                  row.estatus
                }<
                TableCell>
            </TableRow>
          );
        )})
      </TableBody>
    </Table>
  }
  </TableContainer>
  <ComandLine
    action = {action}
  />
</Paper>
```

```
65     );
66 }
```

## Barras deslizantes cinematica directa

```
1 // Barras deslizantes para cinematica directa
2 // src>web>components>Devmode>Controls
3 import React, { useEffect } from 'react'
4 import { SliderMotor } from './SliderMotor';
5 import { Grid, Typography } from '@mui/material';
6
7 export const Controls = ({motorsInformation = [], writteSerial
8   , robot = "scara"}) => {
9
10   return (
11     <>
12       <Typography variant='h5'>Cinematica directa</
13         Typography>
14       <div>
15         {motorsInformation.map(motorInformation => (
16           <SliderMotor key={motorInformation.id} info
17             = {motorInformation} action = {
18               writteSerial}/>
19             )));
20       </div>
21     </>
22   )
23 }
24 }
```

## Barras deslizantes cinematica inversa

```
1 // Barras deslizantes de cinematica inversa y algoritmo
2 // src>web>components>Devmode>Controls>InverseCinematicScara
3 import React, { useEffect, useState } from 'react'
4 import { SliderMotor } from './SliderMotor'
5 import { Typography } from '@mui/material'
6 const L1 = 20
7 const L2 = 10
8
9 export const InverseCinematicScara = ({modifyArray}) => {
10
11   //Aregar useState X Y y pasarlos al action del sliderMotor
12   const [X, setX] = useState(0);
13   const [Y, setY] = useState(0);
```

```

14  const [Z, setZ] = useState(0);
15
16  const infoX = {id:1,min:-50, max:50, default:0, text:""
17      Posicion en eje X"}
18  const infoY = {id:2,min:-50, max:50, default:0, text:""
19      Posicion en eje Y"}
20  const infoZ = {id:3, text:"Posicion en eje Z", min: 0, max:
21      50, default:0}
22
23
24
25
26
27  const actionX = (id,value) => {
28      //console.log(id,value)
29      if(id==1){ setX(value) }
30      if(id==2){ setY(value) }
31      if(id==3){ setZ(value) }
32
33
34
35
36
37
38 // Angles adjustment depending in which quadrant the final
39 // tool coordinate x,y is
40     if (x >= 0 & y >= 0) {           // 1st quadrant
41         thetal = 90 - thetal;
42     }
43     if (x < 0 & y > 0) {           // 2nd quadrant
44         thetal = 90 - thetal;
45     }
46     if (x < 0 & y < 0) {           // 3d quadrant
47         thetal = 270 - thetal;
48         //phi = 270 - thetal - theta2;
49         //phi = (-1) * phi;
50     }
51     if (x > 0 & y < 0) {           // 4th quadrant
52         thetal = -90 - thetal;
53     }
54     if (x < 0 & y == 0) {
55         thetal = 270 + thetal;
56     }

```

```

56         return {theta1, theta2}
57     }
58
59
60     useEffect(() => {
61         const {theta1,theta2} = inverseKinematics(X,Y)
62         //console.log('X,Y,Z', X,Y,Z)
63         console.log('theta1: ',theta1)
64         console.log('theta2: ',theta2)
65
66         if(theta1 != NaN && theta2 != NaN){
67             modifyArray(2,Math.round(theta1))
68             modifyArray(3,Math.round(theta2))
69         }
70
71     }, [X,Y])
72
73     useEffect(() => {
74         modifyArray(5,Z)
75     }, [Z])
76
77
78     return (
79         <>
80             <Typography variant='h5'>Cinematica inversa</
81                 Typography>
82             <SliderMotor key="x" info = {infoX} action = {
83                 actionX}/>
84             <SliderMotor key="y" info = {infoY} action = {
85                 actionX}/>
86             <SliderMotor key="z" info = {infoZ} action = {
87                 actionX}/>
88         </>
89     )
90 }

```

## Botones de guardar, eliminar y seguir trayectoria interna

```

1 import React from 'react'
2 import { IconB } from '../Buttons/IconB'
3 import AddToPhotosIcon from '@mui/icons-material/AddToPhotos';
4 import LayersClear from '@mui/icons-material/LayersClear';
5 import FowardIcon from '@mui/icons-material/Forward';
6
7 export const ScaraAditionalControls = ({modifyArray = () =>{}}
8 ) => {

```

```

8
9     const guardarTrayectoria = () => {
10         modifyArray(0,1)
11         setTimeout(function(){
12             modifyArray(0,0)
13         }, 1600);
14     }
15
16     const eliminarTrayectoria = () => {
17         modifyArray(0,2)
18         setTimeout(function(){
19             modifyArray(0,0)
20         }, 1600);
21     }
22
23     const seguimientoTrayectorias = () => {
24         modifyArray(0,1)
25         //setTimeout(function(){
26         //    modifyArray(0,0)
27         //}, 1600);
28     }
29
30     return (
31     <>
32         <IconB legend="Guardar trayectoria
33             internamente" Icon ={AddToPhotosIcon}
34             action={ guardarTrayectoria }/>
35         <IconB legend="Eliminar trayectorias internas"
36             Icon ={LayersClear} action={
37             eliminarTrayectoria}/>
38         <IconB legend="Seguimiento de trayectorias
39             internas" Icon ={FowardIcon} action={

40             seguimientoTrayectorias }/>
41     </>
42   )
43 }

```

## Componente de barra deslizante

```

1 // Componente de barra deslizante
2 // src>web>components>Devmode>Controls>SliderMotor
3 import React, { useEffect, useState } from 'react';
4 import { styled } from '@mui/material/styles';
5 import Box from '@mui/material/Box';
6 import Grid from '@mui/material/Grid';
7 import Typography from '@mui/material/Typography';

```

```
8 import Slider from '@mui/material/Slider';
9 import MuiInput from '@mui/material/Input';
10
11 const Input = styled(MuiInput)`
12   width: 42px;
13 `;
14
15 export const SliderMotor = ({info = {id:0,min:-90, max:90,
16   default:0, text:""}, 
17   step = 1, action = (c) => console.log(c)}) => {
18
19   const [value, setValue] = useState(info.default);
20   const [isFirstRender, setIsFirstRender] = useState(true);
21
22   useEffect(() => {
23     if(!isFirstRender)
24       action(info.id, value)
25   }, [value])
26
27   useEffect(() => {
28     setIsFirstRender(false)
29   }, [])
30
31   const handleBlur = () => {
32     if (value < info.min) {
33       setValue(info.min);
34     } else if (value > info.max) {
35       setValue(info.max);
36     } else {
37       setValue(value)
38     }
39   };
40
41   return (
42     <Box sx={{ width: {md:400,lg:500} }}>
43       <Typography id="input-slider" gutterBottom>
44         {info.text}
45       </Typography>
46       <Grid container spacing={2} alignItems="center">
47
48         <Grid item xs>
49           <Slider aria-labelledby="input-slider"
50             value= {typeof value === 'number' ?
51               value : info.default}
52             onChange={ e => setValue(e.target.
53               value === '' ? '' : Number(e.target
54                 .value))}>
```

```

51         defaultValue={info.default} min={info.
52                         min} max = {info.max}/>
53     </Grid>
54     <Grid item>
55         <Input
56             sx={{ width: 50 }}
57             value={value}
58             onChange={ e => setValue(e.target.
59                             value === '' ? '' : Number(e.target
60                               .value))}
61             onBlur={handleBlur}
62             inputProps={{
63                 step: step,
64                 min: info.min ,
65                 max: info.max,
66                 type: 'number',
67                 'aria-labelledby': 'input-slider',
68             }}
69         />
70     </Grid>
71 </Box>
72 );
73 }

```

## Sección de programación

```

1 // Seccion de programacion
2 // src>web>components>Devmode>Programer
3 import { useState } from "react";
4 import { CodeEditor } from "./CodeEditor/CodeEditor";
5 import { Consola } from "./Consola/Consola";
6 import { EditorButtons } from "./CodeEditor/EditorButtons";
7
8 export const Programer = ({ history, addToHistory,
9                           resetHistory, writte, positionsInformation, }) => {
10
11     const [code, setCode] = useState("//Agrega tu codigo
12                                   aqui");
13
14     //Funcion para modificar consola y escribir en serial
15     // de acuerdo al comando ejecutado
16     const actionCommand = (c='') => {
17         switch (c) {
18             case 'clear':
19                 resetHistory();
20         }
21     }
22
23     return (
24         <div>
25             <div>
26                 <CodeEditor
27                     value={code}
28                     onChange={setCode}
29                     height="300px"
30                     width="100%" />
31             </div>
32             <div>
33                 <Consola
34                     value={positionsInformation}
35                     height="300px"
36                     width="100%" />
37             </div>
38         </div>
39     );
40 }

```

```
17         break;
18
19     case 'clear dev':
20         setCode('');
21         break;
22
23     case 'clear all':
24         setCode('');
25         resetHistory();
26         break;
27
28     case 'home':
29         writte(
30             '0,0,-90,0,0,0,0,90,0,0,0,0,0,0,0,70,90,0,-80,-90,0
31             '); //Home de Naab
32         //addToHistory(c)
33         break;
34
35     default:
36         writte(c);
37         addToHistory(c)
38         break;
39     }
40 }
41
42 return (
43     <div className="container">
44         <div className="row mt-5">
45             <EditorButtons
46                 code = {code}
47                 setCode={setCode}
48                 actionCommand = {actionCommand}
49                 >
50                 positionsInformation = {
51                     positionsInformation}
52                 />
53             </div>
54             <div className="row mt-3">
55                 <CodeEditor
56                     code={code}
57                     setCode={setCode}
58                     >
59             </div>
60             <div className="row mt-2 pb-3">
61                 <Consola
62                     action = {actionCommand}
63                     history ={history}/>
64             </div>
65         </div>
66     </div>
67 )
```

```
61         </div>
62     )
63 }
```

## Pestañas

### Configuración

```
1 // Ventana de configuracion
2 // src>web>pages>Configuration
3 import React, { useEffect, useState } from "react"
4 import { WebLayout } from "../layout/WebLayout"
5 import {naab_motors_information} from '../../../../../data/motors/
    naab_motors'
6 import { scara_position_information } from '../../../../../data/motors/
    scara_motors';
7 import scaraImg from '../../../../../img/scara.png'
8 import naabImg from '../../../../../img/naab.jpg';
9 import { SimpleSelector } from '../../../../../interface>Selectors/
    SimpleSelector';
10 import { IconB } from "../components/Buttons/IconB";
11 import { Container } from "@mui/material";
12
13 export const Configuration = ({setConfiguration = () => {}},
14   configuration= {
15     robot: 'naab', baud: 115200, information:
16       naab_motors_information, imgRobot: naabImg,
17       prefix: "posiciones,",postfix:""},
18   ) => {
19
20   const [robot, setRobot] = useState(configuration.robot);
21   const [baud, setBaud] = useState(configuration.baud);
22
23   const selectRobot = [
24     {value: 'naab',text:'naab'},
25     {value: 'scara',text:'scara'},
26   ]
27
28   const selectBaud = [
29     {value: 300,  text:300},
30     {value: 600,  text:600},
31     {value: 1200, text:1200},
32     {value: 2400, text:2400},
33     {value: 9600, text:9600},
34     {value: 14400, text:14400},
```

```
34         {value: 19200, text:19200},
35         {value: 28800, text:28800},
36         {value: 38400, text:38400},
37         {value: 115200, text:115200},
38     ]
39 }
40
41 const getInformation = () => {
42     switch (robot) {
43         case 'scara':
44             return scara_position_information;
45             //break;
46
47         default:
48             return naab_motors_information;
49             //break;
50     }
51 }
52
53 const getImg = () => {
54     switch (robot) {
55         case 'scara':
56             return scaraImg;
57             //break;
58
59         default:
60             return naabImg;
61             //break;
62     }
63 }
64
65 const prefixNew = robot === "naab" ? "posiciones," : "";
66 const postfixNew = robot === "naab" ? "" : "";
67
68 const handleClick = () => {
69     setConfiguration(
70     {
71         robot,
72         baud: robot==='naab' ? 9600 : 115200 ,
73         information: getInformation(),
74         imgRobot: getImg(),
75         prefix: prefixNew,
76         postfix: postfixNew
77     }
78 )
79 }
```

```

82    return (
83      <>
84        <WebLayout/>
85        <Container>
86          <div>Configuracion</div>
87          <div className="window-app">
88            <h1>Configuration</h1>
89            <SimpleSelector legend="Robot" data={robot}>
90              setData={setRobot} selectInfo={
91                selectRobot}/>
92            <SimpleSelector legend="Baud rate" data={baud}>
93              setData={setBaud} selectInfo={
94                selectBaud}/>
95            <IconB action={handleClick} legend="Aplica
96              configuracion"></IconB>

```

## Control

```

1 // Ventana de configuracion
2 // src> web>pages>Control
3 import React, { useState } from 'react';
4 import { Box, Container, Grid } from '@mui/material';
5 import { WebLayout } from '../layout/WebLayout';
6 import { Programer } from '../components/Devmode/Programer';
7
8 import { Controls } from '../components/Devmode/Controls/
8   Controls';
9 import { IconB } from '../components/Buttons/IconB';
10
11 import ConnectedTvIcon from '@mui/icons-material/ConnectedTv';
12 import DesktopAccessDisabledIcon from '@mui/icons-material/
12   DesktopAccessDisabled';
13 import DesktopWindowsIcon from '@mui/icons-material/
13   DesktopWindows';
14 import { ScaraAditionalControls } from '../components/Devmode/
14   Controls/ScaraAditionalControls';
15 import { InverseCinematicScara } from '../components/Devmode/
15   Controls/InverseCinematicScara';
16
17 export const Control = ({configuration={ robot: 'naab', baud:
18   9600, information: [], imgRobot},

```

```
18     serialHook}) => {
19
20     const {baud, information, imgRobot, robot} = configuration;
21
22     return (
23       <>
24         <WebLayout/>
25         <Container >
26
27           <Box component="div">
28             <Grid container spacing={2} alignItems="center">
29               <Grid item xs={12} md={4} >
30                 <IconB legend="Seleccionar dispositivo" Icon =
31                   ={ConnectedTvIcon} action={ serialHook.
32                     setConfiguration }/>
33               </Grid>
34               <Grid item xs={12} md={4} >
35                 <IconB legend="Resetear conexion" Icon ={(
36                   DesktopWindowsIcon) action={ serialHook.
37                     resetConfiguration }/>
38               </Grid>
39               <Grid item xs={12} md={4} >
40                 <IconB
41                   legend= {serialHook.isConected ? "
42                     Dispositivo conectado" : "Dispositivo
43                     desconectado"}
44                   Icon ={serialHook.isConected ?
45                     DesktopWindowsIcon :
46                     DesktopAccessDisabledIcon}
47                   action={ serialHook.isPortOpen }
48                   />
49                 </Grid>
50               </Grid>
51             </Box>
52
53             <Box component="div">
54               <Grid container spacing={2} alignItems="center">
55                 <Grid item xs={12} md={6} >
56                   <img src = {imgRobot} className='img-fluid
57                     max-width: 50%' />
58                 </Grid>
59                 <Grid item xs={12} md={6} sx={{tb:5}}>
60                   <Controls
61                     robot = {robot}
62                     motorsInformation={information}
63                     writteSerial = {serialHook.
64                       modifyPosition}
65                   />
66                 </Grid>
67               </Box>
68             </Grid>
69           </Container>
70         </WebLayout>
71       </>
72     </Container>
73   </>
74 </WebLayout>
```

```
56         </Grid>
57     {
58         configuration.robot == "scara" ? (
59             <Grid container sx={{tb:5}}>
60                 <Grid item xs={6} md={6} >
61                     <ScaraAdditionalControls
62                         modifyArray = {serialHook
63                             .modifyPosition}/>
64                 </Grid>
65                 <Grid item xs={6} md={6} >
66                     <InverseCinematicScara
67                         modifyArray = {serialHook
68                             .modifyPosition}/>
69                 </Grid>
70             ) : (<></>)
71         }
72     </Grid>
73 </Box>
74
75 <div className="container">
76     <div className="row">
77         <Programmer
78             history = {serialHook.history}
79             addToHistory = {serialHook.
80                 addToHistory}
81             resetHistory = {serialHook.
82                 resetHistory}
83             writte = {serialHook.writte}
84             positionsInformation = {serialHook.
85                 positionsInformation}
86         />
87     </div>
88 </div>
89 </Container>
90 </>
91 )
92 }
```

## CÓDIGO USADO EN LA PRUEBA

```
// Gripper abierto  
setTimeout(function(){  
    action('1,0,0,0,0,80,9000,0')  
}, 0);
```

```
// Revoluta 2 a posición 13  
setTimeout(function(){  
    action('1,0,13,0,0,80,9000,0')  
}, 1000);
```

```
// Descenso en eje z a 8  
setTimeout(function(){  
    action('1,0,8,0,0,8,80,6800,0')  
}, 4000);
```

```
// Cierre del gripper --  
setTimeout(function(){  
    action('1,0,8,0,0,8,0,3000,0')  
}, 7000);
```

```
// Ascenso en eje z a 6 --  
setTimeout(function(){  
    action('1,0,6,0,0,8,0,6800,0')  
}, 10000);
```

```
// Angulo theta2 a 46 --  
setTimeout(function(){
```

```
action('1,0,6,46,0,8,0,9000,0')
}, 13000);

// Descenso en eje z a 10 --
setTimeout(function(){
    action('1,0,0,46,0,10,0,6800,0')
}, 16000);

// Abrir gripper
setTimeout(function(){
    action('1,0,0,46,0,10,80,3000,0')
}, 18000);

// Subir en eje z a 6
setTimeout(function(){
    action('1,0,0,46,0,10,80,9000,0')
}, 21000);

// Volver a home
setTimeout(function(){
    action('1,0,0,0,0,80,9000,0')
}, 23000);

// Ejecutar trayectoria constantemente
setTimeout(function(){
    action('2,0,0,0,0,80,9000,0')
}, 26000);
```