UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"Interfaz gráfica generadora de trayectorias para operar robots a través de imágenes vectorizadas."

TESIS

QUE PARA OPTAR POR EL TÍTULO DE:

INGENIERO MECATRÓNICO

PRESENTA:

JHOVVANY HAZZAEL HERNÁNDEZ CLEMENTE

DIRECTOR DE TESIS

M.I. HUMBERTO MANCILLA ALONSO

MÉXICO.2014

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Dedicatoria.

A mis padres José Gildardo y Martha, que con sus consejos, apoyo incondicional, cariño y comprensión han sido siempre, el motor principal que me impulsa a ser mejor cada día.

A mí tía Lourdes que siempre me apoyo día a día para seguir adelante en este proyecto.

A mí hermana Lizbeth Nallely y prima Paulina Virginia que con su compañía y alegría me contagiaban y me motivaban a seguir adelante.

A mis abuelitos por su apoyo y comprensión.

A toda mi familia en general que ha participado directa o indirectamente a lo largo de mi formación profesional.

Agredecimientos.

Agradezco primeramente a Dios, que me dio la fortaleza y sabiduría para permitirme llevar a cabo este paso decisivo en mi vida y por haber puesto en mi camino a aquellas persona que han sido mi soporte y mi compañía en grandes momentos.

La presente tesis es un trabajo en el cual participaron varias personas directa o indirectamente, leyendo, opinando, dándome ánimos y apoyo en general. Por ello es para mí un placer utilizar este espacio para expresar mis agradecimientos hacia estas personas.

Agradezco primeramente de manera especial y sincera al M.I Humberto Mancilla Alonso por aceptarme y trabajar bajo su dirección, en la realización de esta tesis, su confianza y aporte invaluable de conocimientos ayudaron a guiar mis ideas para ser plasmadas en este trabajo.

Agradezco también al Ing. Raúl Gilberto Navarro Valdez y especialmente al Ing. Amando Sánchez Guzmán por su apoyo, al facilitarme el uso del Laboratorio de Manufactura Avanzada para llevar a cabo todas las actividades de rehabilitación e implementación durante el desarrollo de esta tesis.

A mí universidad y a mí querida Facultad de Ingeniería.

Si no somos lo suficientemente disciplinados
como para que una fábrica este impecable,
entonces no tenemos la disciplina suficiente
para que todas aquellas maquinas
funcionen correctamente.

Steve Jobs.

Índice.

Lista de Abreviaturas.

ACL - Lenguaje Moderno de Control, o Advanced Control Language.

API - Interfaz de Programación de Aplicaciones, o Application Programming Interface.

ATS - Software Moderno de Terminal, o Advanced Terminal Software.

BBRAM - Battery Backup Random Access Memory.

CADR - Diseñó Asistido por Computadora para Robótica.

DIMEI - División de Ingeniería Mecánica E Industrial.

DXF - Drawing Exchange Format.

GNU o GLP - Licencia Pública General de GNU, o GNU General Public License.

LED - Diodo Emisor de Luz, o Light-Emitting Diode.

LIMAC - Laboratorio de Ingeniería Mecánica Asistida por Computadora.

OSS - Software de Código Abierto, u Open Source Software.

PC – Ordenador Personal, o Personal Computer.

PCH o TCP - Punto Central de la Herramienta, generalmente de la pinza.

PDE - Processing Development Environment.

PDF - Formato de Documento Portátil, o Portable Document Format.

SVG - Gráficos Vectoriales Redimensionables, o Scalable Vector Graphics.

XHTML - eXtensible HyperText Markup Language.

XML - Lenguaje de Marcas eXtensible, o eXtensible Markup Language.

I. Introducción.

Debido al avance tecnológico de la robótica, el Laboratorio de Manufactura Avanzada se encuentra en la necesidad de rehabilitar y mejorar sus equipos didácticos con la finalidad de seguir a la vanguardia, con la posibilidad de economizar recursos; para tal fin, dentro de este trabajo de tesis, se encuentra el desarrollo e implementación de un nuevo sistema que busca rehabilitar tres robots de configuraciones diferentes en el laboratorio a través de la implementación de una interfaz generando trayectorias con el uso de imágenes vectorizadas.

Esté trabajo se desarrolla de la siguiente manera, como primer punto, se habla de los antecedentes del Laboratorio de Manufactura Avanzada, el uso y deshabilitación de algunos equipos; posteriormente se hace un levantamiento exclusivo del lugar, donde se muestra la localización y distribución del equipo de robótica con el que se hará la implementación, y así tener un control de la situación a la que se va a enfrentar, a su vez se explican las características y sistemas de cada una de las tres configuraciones de robots existentes en el laboratorio y además se expone de forma breve y necesaria el método de operación de los equipos. Con la información anterior se plantea una necesidad y una problemática las cuales se abordan en la etapa de desarrollo.

Como segundo punto, en la etapa de desarrollo se propone una solución ante la necesidad del Laboratorio de Manufactura Avanzada, integrando todos los elementos disponibles dentro del laboratorio y sumando el apoyo de las aplicaciones “open source”, la programación en software de código abierto, hacen de este trabajo una integración Mecatrónica de diferentes áreas de la ingeniería. La disponibilidad en la red del software de programación Processing sustenta la propuesta de solución para programar una interfaz y manejar archivos en formato vectorial y con ello generar un archivo que cumpla con las características necesarias para ser leído por el controlador del robot, la información contenida en el archivo pertenece a una serie de puntos en coordenadas cartesianas descritas por la imagen vectorizada, está información hace que el punto central de la herramienta recorra un lugar geométrico de trayectoria dentro de una área restringida delimitada por el rango de alcance del robot.

La interfaz de nombre CADR ofrece un menú de opciones para generar un archivo con información para cada configuración de robot existente en el laboratorio, y además se propone una serie de opciones para manejar la información en diferentes planos geométricos haciendo uso de orientaciones canonícas y empleando la utilización de una proyección de los puntos sobre un cilindro parabólico.

Finalmente como tercer punto se muestran los resultados de las pruebas realizadas de la propuesta de solución, validando la opción número dos y dedicando un apartado para mostrar las conclusiones y recomendación de posibles aplicaciones prácticas y de desarrollo como trabajo a futuro, además de mostrar una perspectiva de los beneficios obtenidos.

II. Planteamiento.

La falta de inversión en tecnología propia, ocasiona que en México se sufra de un rezago tecnológico que afecta distintos niveles de la sociedad, empezando con la economía.

Es triste saber que no sólo debemos competir con los obstáculos impuestos a los mexicanos, debido a la falta de interés en el desarrollo del país; sino que a veces, la tecnología sencillamente no está a nuestro alcance. Ya que sigue siendo muy costosa para la mayoría, sobre todo mantener el ritmo con la que ésta evoluciona y se actualiza. Por ejemplo si a una familia mexicana le es difícil poder comprar una computadora y pagar un acceso a internet, le es más o igual de difícil poder reemplazar la computadora por una nueva, y mantener el ritmo de los constantes cambios en estándares, sistemas operativos y hardware.

Pero el avance tecnológico que contempla la informática, la electrónica y la robótica hace que los sistemas didácticos destinados a estar en las aulas universitarias sufran de un rezago tecnológico aun mayor que el de una familia mexicana.

Con el desarrollo de este trabajo se busca generar una propuesta que logre rehabilitar el Área de Robótica Didáctica y el Área de Robótica Industrial, cuyas áreas dependen del Laboratorio de Manufactura Avanzada, el cual es parte de los Laboratorios del Departamento de Ingeniería de Materiales y Manufactura, localizados en el interior del edificio O (Laboratorios de Ingeniería Mecánica "Alberto Camacho Sánchez"), Anexo de Ingeniería conjunto Sur de la Facultad de Ingeniería de la UNAM, para esto se ha propuesto un objetivo a cumplir el cual será evaluado al final de este trabajo.

Como objetivo principal se busca rehabilitar tres robots de configuraciones diferentes en el laboratorio de manufactura avanzada a través de la implementación de una interfaz generando trayectorias con el uso de imágenes vectorizadas.

III. Antecedentes.

El Centro de Diseño y Manufactura de la Facultad de Ingeniería de la UNAM fue creado en el año de 1976 (hoy Laboratorios de Ingeniería Mecánica "Alberto Camacho Sánchez"), teniendo el doble propósito de contribuir a la formación académica de los alumnos y el desarrollo profesional de los profesores de la Facultad mediante su participación en proyectos de investigación y desarrollo.

El edificio Laboratorios de Ingeniería Mecánica "Alberto Camacho Sánchez" cuenta con laboratorios que dependen del Departamento de Ingeniería de Materiales y Manufactura, cuyo objetivo nos dice:

> "En la Ingeniería Mecánica se investiga, diseña, desarrolla, produce, construye, opera, mantiene y controla tanto las herramientas y elementos de máquina, como las máquinas y los equipos de los sistemas productivos de industrias manufactureras y de servicios. Así como los sistemas de conversión de energía, usando métodos matemáticos, físicos y químicos, adquiridos por el estudio y la experiencia, usando herramientas de Ingeniería, como el Dibujo y la Computación".

La Facultad de Ingeniería a través del Departamento de Ingeniería Mecánica ha impulsado la creación del Laboratorio de Manufactura Avanzada el cual nace como una respuesta a la gran velocidad con la que se están transformando las tecnologías asociadas a la ingeniería mecánica y al papel central que está destinada a jugar en la modernización del sector manufacturero del país.

El Laboratorio de Ingeniería Mecánica Asistida por Computadora (LIMAC) forma parte del Laboratorio de Manufactura Avanzada, el cual fue inaugurado el 22 de enero de 1991, sin embargo este laboratorio empezó a operar antes de la creación del Laboratorio de Manufactura Avanzada a mediados del mes de junio de 1990.

Ambos laboratorios comparten la misma historia de su inauguración, así como parte del equipo con el que cuentan, como se declara en una nota que la gaceta UNAM revelaba el 01 de abril del 2004, decía: "Inauguran laboratorio de ingeniería mecánica. El LIMAC opera con tecnología de frontera; es líder en su campo".

> Estos laboratorios fueron equipados con tecnología con un valor cercano a 200 mil dólares, donados por la empresa internacional SK, Engineering & Construction Co. Ltd., en cuya gestión participó Petróleos Mexicanos. Gerardo Ferrando Bravo, director de la FI, informó que el laboratorio tiene 30 computadoras, dos servidores de alta capacidad en cómputo, un microscopio metalográfico, un par de manipuladores mecánicos y equipo periférico. Este espacio, indicó, beneficiará en sus prácticas a los alumnos de más de 10 asignaturas. Cada semestre dará servicio a un promedio de dos mil alumnos de las carreras de las ingenierías mecánica, industrial, mecatrónica y robótica.
>
> Gerardo Ferrando y Jim Peom Kim, director ejecutivo de la empresa donante develaron la placa que oficializa su puesta en marcha y reconoce la generosidad de la iniciativa privada en favor de la función educativa de la Universidad.

> Ubaldo Eduardo Márquez Amador, jefe del Departamento de Ingeniería Mecánica de la FI, explicó que la misión del nuevo laboratorio es la formación de personal altamente especializado en la operación, optimización y desarrollo de tecnologías de manufactura avanzada. Ello se logrará mediante la capacitación y participación en proyectos de investigación auspiciados por la industria, a la cual se le proporciona asesoría en la instrumentación y adaptación de sistemas fabriles de vanguardia, en particular lo relacionado con diseño e ingeniería asistida por computadora. “Nuestra meta es clara: formar un laboratorio líder en su campo en México, que sume en sus instalaciones, equipo y personal al de otros centros, en búsqueda de desarrollos científicos de calidad e innovación de productos; además, mejorar los procesos de la producción nacional”, aseveró.

Ante esta perspectiva en el Laboratorio de Manufactura Avanzada con el pasar de los años, el desuso del equipo en el área de robótica y robótica industrial se fue generando paulatinamente originando cierto rezago tecnológico en el equipo, además de sumar problemas por desgaste y desajuste de los mismos, eso sin mencionar aun la rapidez con la que evoluciona la tecnología informática y sus aplicaciones; de este modo, si no somos capaces de hacer evolucionar el valor que poseemos de manera acompasada con la evolución de nuestro entorno, el valor acaba por desaparecer.

En este trabajo se presenta una propuesta que ayudará a la rehabilitación de los equipos en el área de robótica didáctica y robótica industrial, logrando con ello mantener el Laboratorio de Manufactura Avanzada a la vanguardia, reemplazando y mejorando sistemas que posibiliten economizar recursos; pero sobre todo permitiendo desarrollo de tecnologías propias que impulsen la reactivación de equipos que tengan las capacidades de seguir afrontando los retos tecnológicos que el país requiere.

1. Levantamiento.

En esta fase de investigación se busca toda la información existente acerca de los elementos que conforman el equipo de robótica didáctica y robótica industrial para su rehabilitación, también se hablará de su operación como robots de aplicación académica.

1.1 Breve introducción de la robótica.

Una definición de robot usada por The Robot Institute of America es:

"Un robot es un manipulador multifuncional y reprogramable diseñado para mover materiales, partes, herramientas, o dispositivos especiales, a través de movimientos variables programados para la ejecución de una variedad de tareas".

Con esta definición se puede afirmar que un robot es una máquina de propósito general consistente de un sistema mecánico controlado por un cerebro electrónico y que puede interactúa con su medio ambiente.

Por lo tanto el robot está constituido de sistemas, por ejemplo en la Figura 1.1 se muestra un esquema de bloques representando los sistemas constitutivos de un robot, que son los siguientes: sistema de comunicación usuario-máquina, sistema de decisión, sistema mecánico y, posiblemente, sistema de percepción. El sistema de percepción puede estar formado por sensores del tipo interno y/o del tipo externo, o simplemente no existir y, sin embargo, el sistema completo sigue siendo considerado como un robot.

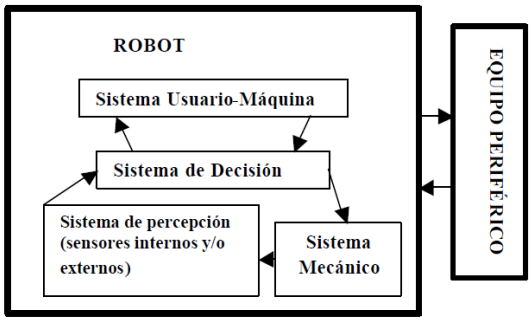


Figura 1.1 Sistemas constitutivos de un Robot.

Durante el desarrollo de este trabajo se hablará con más detalle de los robots exclusivos con los que cuenta el Laboratorio de Manufactura Avanzada así como de sus sistemas constitutivos.

1.2 Laboratorio de Manufactura Avanzada.

Como primera etapa de desarrollo de este proyecto se habló de antecedentes históricos acerca del Laboratorio de Manufactura Avanzada, así como su localización y a qué departamento de la Facultad de Ingeniería pertenece. Para complementar esta información se hablará más, acerca de las aéreas y del equipo con las que cuenta este laboratorio.

El Departamento de Ingeniería de Materiales y Manufactura ha organizado dentro del Laboratorio de Manufactura Avanzada las aéreas paras las cuales dicho laboratorio presta sus servicios a la comunidad estudiantil. Estas aéreas se han implementado dependiendo del equipo con el que se cuenta, a continuación se enlistan las aéreas disponibles y acondicionadas por el Laboratorio de Manufactura Avanzada.

- Área de entrenamiento de control numérico.
- Área de robótica didáctica.
- Área de maquinas C.N.C. industriales.
- Área de robótica industrial.
- Área del centro de manufactura integrada por computadora.

En la Figura 1.2.1 se muestra en un plano, la distribución de las áreas de trabajo dentro del laboratorio.

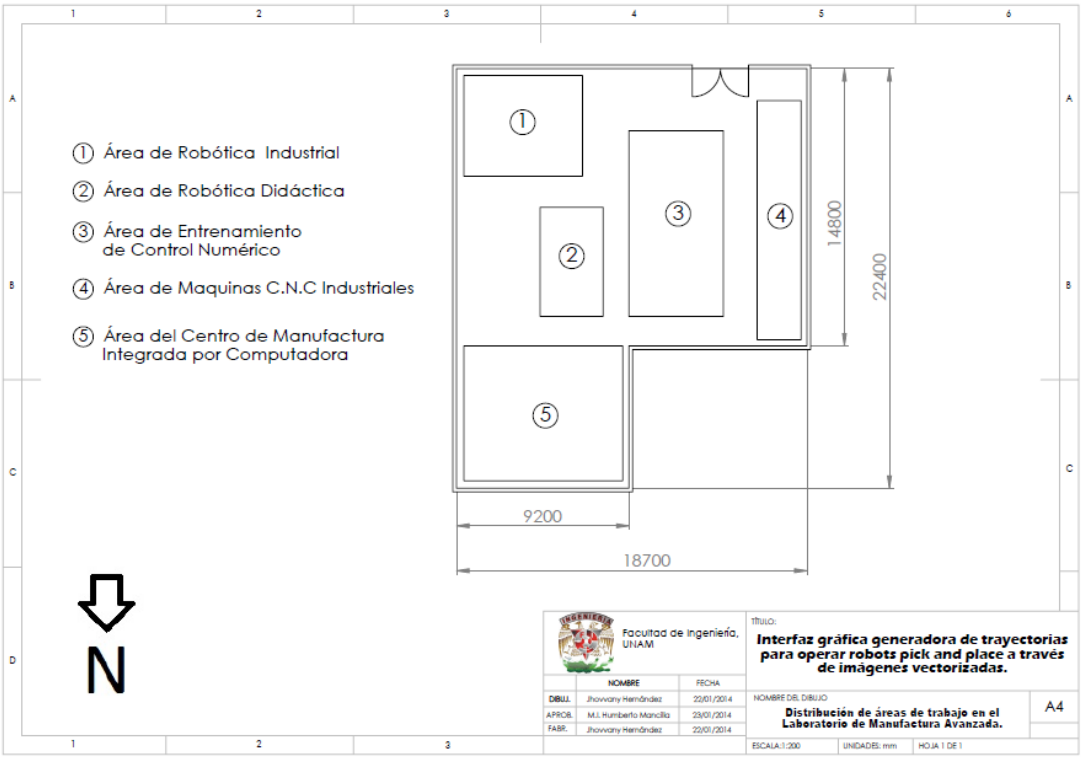


Figura 1.2.1 Distribución de áreas de trabajo en el Laboratorio de Manufactura Avanzada (Anexo 1).

El desarrollo de este trabajo se enfoca principalmente en las áreas de robótica didáctica y robótica industrial. En la Figura 1.2.2 se muestra un plano de levantamiento, la elaboración de este plano nos ayudará a determinar de manera gráfica la superficie sobre la cual esta investigación ha sido desarrollada para su posterior aplicación.

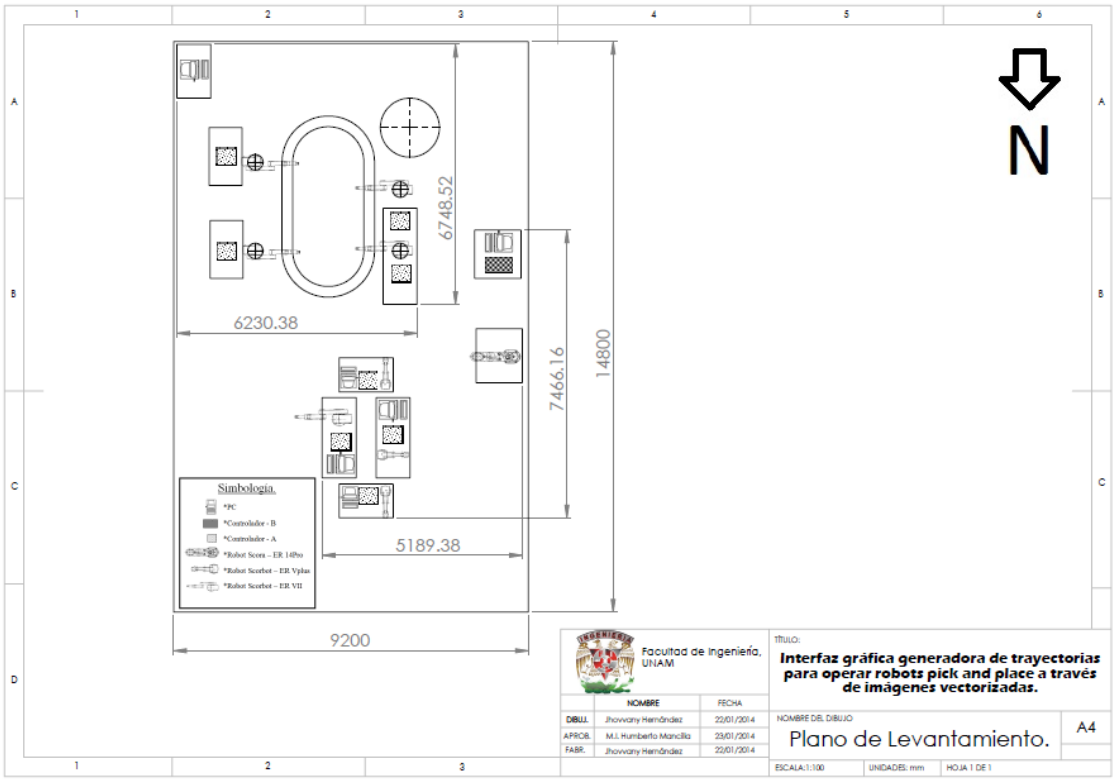


Figura 1.2.2 Área de Robótica Didáctica y Área de Robótica Industrial (Anexo 1).

Para mayor detalle acerca de los planos de la figura 1.2.1 y 1.2.2 ver Anexo 1 “Planos de Distribución y Levantamiento” al final de este documento.

La distribución y equipo de estas áreas se efectúa de la siguiente manera:

Área de Robótica Didáctica.

Esta área sirve de entrenamiento para conocer el equipo y comportamiento de los robots, se cuenta con los siguientes elementos:

- 3 Robots Scorbot-ER V Plus.
- 1 Robot Scorbot-ER VII.
- 1 Robot Scora-ER 14pro.
- 2 Controladores intelitek (Conexión USB).
- 2 Controladores –A (Conexión RS232C).
- 1 Controlador –B (Conexión RS232C).
- 5 Computadoras.
- 3 Botoneras de enseñanza.
- 2 Mesas rotatorias.
- 2 Bandas transportadoras.

Área de Robótica Industrial.

En esta área su rehabilitación ayudaría a aplicar los conocimientos de robótica didáctica, control numérico y programación de maquinas C.N.C. industriales, se cuenta con los siguientes elementos:

- 4 Robots Scorbot-ER VII.
- 4 Controladores –A (Conexión RS232C).
- 1 Computadora.
- 4 Botoneras de enseñanza.
- 1 Circuito PLC cinta transportadora.
- 1 Fresa CNC.
- 1 Torno CNC.
- 1 Mesa rotatoria industrial.

En la Figura 1.2.3 se muestra un plano de levantamiento de vista panorámica, el cual ayudará a visualizar las aéreas de robótica, así como la integración de los robots que se encuentran dentro de este laboratorio.

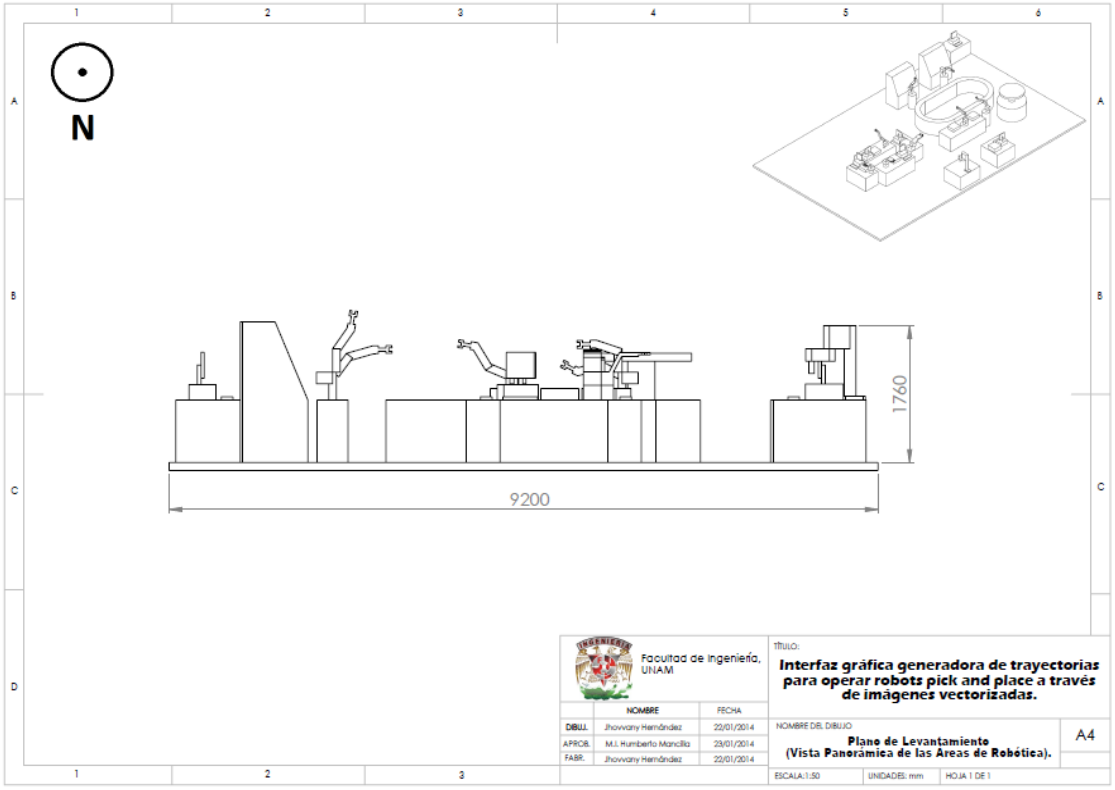


Figura 1.2.3 Vista Panorámica de las Áreas de Robótica (Anexo 1).

Ambas áreas son independientes pero comparten similitud con el equipo de control y la forma de operación, su distribución puede variar de acuerdo a la aplicación práctica que se pueda generar con la rehabilitación de estas áreas. El uso de ambas áreas se limitaba a mostrar solo el funcionamiento de los robots, su programación y funciones elementales como por ejemplo tomar piezas y dejarlas en un sitio diferente al que previamente se había tomado, por otro lado el área de robótica industrial poco a poco fue dejando de prestar servicio, una de las razones principales fue la compatibilidad de los controladores con las PC actuales, así como su software, entre otras circunstancias ajenas a este documento.

Se estima que las asignaturas beneficiadas con la rehabilitación del laboratorio serían las correspondientes a las carreras de Ingeniería Mecatrónica, Ingeniería Mecánica e Ingeniería Industrial pertenecientes a la División de Ingeniería Mecánica E Industrial (DIMEI).

En el siguiente listado se mencionan las asignaturas impartidas en la Facultad de Ingeniería que podrían ser beneficiadas de manera directa o indirecta, para generar conocimientos de índole profesional.

- Ingeniería de Manufactura.
- Robótica.
- Automatización Industrial.
- Sistemas de Manufactura Flexible.
- Procesos de Manufactura.
- Investigación de Operaciones.
- Procesos Industriales.
- Planeación y Control de la Producción.
- Sistemas de Calidad.

El beneficio de mayor importancia es el de índole profesional ya que los alumnos son los principales enriquecedores de conocimientos teórico-prácticos, ayudando así a más de 250 alumnos aproximadamente, repartidos en sus diferentes asignaturas a lo largo del semestre.

Una vez definidas las razones por las cuales la rehabilitación de las aéreas de robótica y robótica industrial favorece a la comunidad estudiantil de la DIMEI, los límites de este trabajo de investigación se basan principalmente en el desarrollo de una interfaz de fácil uso que generará un archivo, el cual contendrá las coordenadas cartesianas de una área específica de cada robot para realizar el trazo de esas coordenadas, así mismo el trazo de las coordenadas deben corresponder al dibujo de una imagen vectorizada, el archivo generado cumple con las características de los controladores para su lectura así como el tipo de robot y sus parámetros.

Ahora pasaremos a la etapa donde se definen las características de cada robot, así como la de cada controlador y su interfaz que maneja la información para ejecutar las tareas que el usuario indique.

1.3 Tipos de robots en el Laboratorio de Manufactura Avanzada.

Se tiene el registro de tres configuraciones diferentes de robots, los cuales están destinados a ser implementados de forma didáctica o industrial dentro del laboratorio, estos robots fueron adquiridos por la facultad de ingeniería gracias a la colaboración de la empresa internacional SK, Engineering & Construction Co. Ltd.

Los robots son nombrados de acuerdo a la empresa que los fabrica, el nombre de esta empresa Eshed Robotec (1982), posteriormente paso a ser conocida con el nombre de Intelitek (2001). Sin embargo, Eshed Robotec fue el fabricante de gran parte del equipo de robots y controladores con los que cuenta el Laboratorio de Manufactura Avanzada, por lo que el desarrollo de este trabajo se basa principalmente en los manuales de operación con los que la empresa Eshed Robotec contaba en ese entonces para el uso y operación de sus equipos.

1.3.1 Robot Scorbot-ER V Plus.

La arquitectura mecánica de este robot manipulador es del tipo antropomórfico (Figura 1.3.1.1), las partes que constituyen su sistema mecánico son base, cuerpo, brazo, codo, antebrazo y pinza; el sistema de percepción se caracteriza por sensores de tipo interno mediante un codificador y sensores de tipo externo con la implementación de una leva y un microinterruptor. La Figura 1.3.1.2 representa esquemáticamente la estructura y composición del Robot Scorbot-ER V Plus.

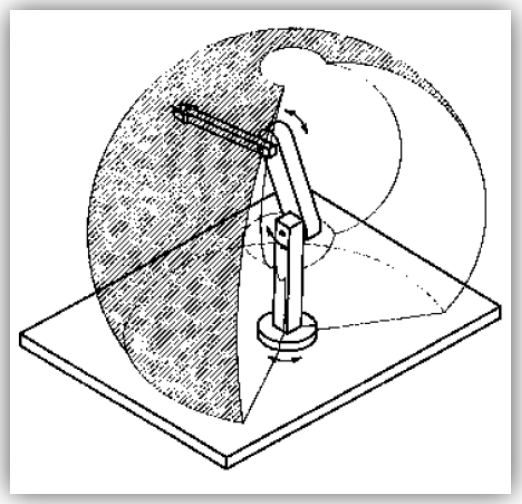


Figura 1.3.1.1 Arquitectura mecánica antropomorfa.

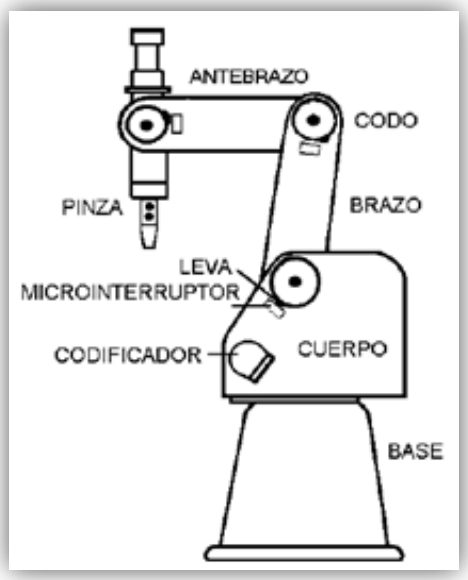


Figura 1.3.1.2 Partes del Robot Scorbot-ER V Plus.

1.3.1.1 Especificaciones.

En este tema se muestran las especificaciones y los componentes necesarios del brazo robot SCORBOT-ER Vplus.

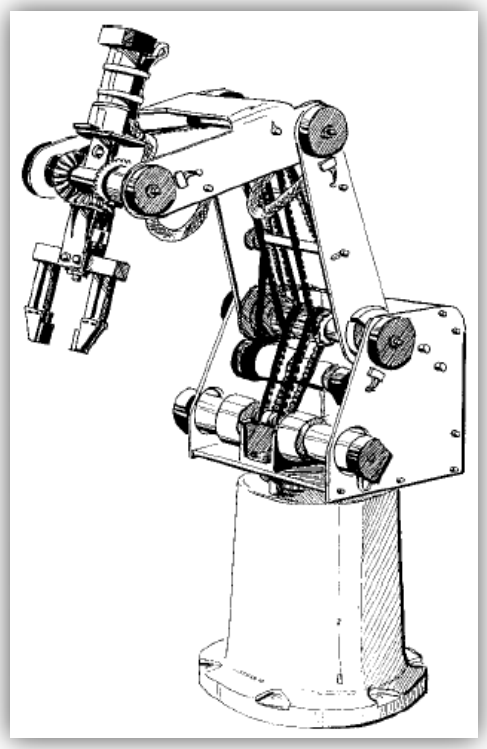


Figura 1.3.1.3 Robot Scorbot-ER V Plus.

En la Tabla 1.3.1.1 se detallan las especificaciones del brazo robot; la elaboración de esta tabla se realiza haciendo una abstracción de la información del manual de usuario, acerca de los requerimientos más importantes que sirven como preámbulo en la elaboración de este trabajo.

Tabla 1.3.1.1 Especificaciones del Robot SCORBOT-ER Vplus	
Estructura mecánica	Robot de articulación vertical
Número de ejes	Cinco ejes más pinza
Movimiento de ejes: ✸ Eje 1: Base ✸ Eje 2: Hombro ✸ Eje 3: Codo ✸ Eje 4: Inclinación de la pinza ✸ Eje 5: Giro de la pinza	 310º +130º / -35º ± 130º ± 130º Sin límite mecánico; eléctricamente, ±570°
Rango de operación	610 mm
Elemento terminal	DC pinza servo, con codificador óptico, mide el tamaño de piezas/fuerza por medio del sensor y de software
Inicio (referencia)	Posición fija en cada eje, hallada por medio de microinterruptores
Realimentación	Codificadores ópticos en cada eje
Actuadores	Servo Motores de 12 VCC
Relación de transmisión	Motores 1-3: 127:1 Motores 4-5: 65.5:1 Motor 6 (pinza) 19.5:1
Transmisión	Engranes, correas dentadas y husillos
Máxima carga de trabajo	1 Kg, incluyendo la pinza
Repetitividad	±0.5 mm
Peso	11.5 Kg
Rango de Temperatura	2° – 40°

1.3.1.2 Sistema Mecánico.

1.3.1.2.1 Estructura.

El SCORBOT-ER Vplus es un robot de estructura vertical articulada, con cinco grados de libertad. Cuando se le agrega la pinza, el robot posee seis grados de libertad. Este diseño permite a la herramienta final ser colocada y orientada arbitrariamente en un gran espacio de trabajo.

En las Figuras 1.3.1.4 y 1.3.1.5 se identifican los eslabones y las articulaciones del brazo mecánico.

Los movimientos de las articulaciones son descritos en la Tabla 1.3.1.2.

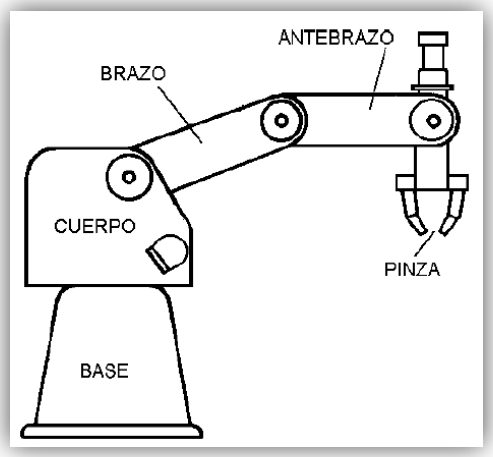


Figura 1.3.1.4 Eslabones del Robot.

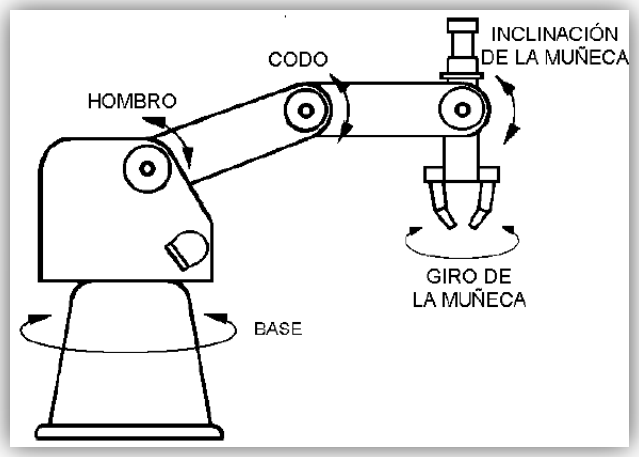


Figura 1.3.1.5 Articulaciones del Robot.

Tabla 1.3.1.2			
Eje n°	**Nombre**	**Movimiento**	**Motor**
1	Base	Rotación del cuerpo	1
2	Hombro	Sube y baja el brazo	2
3	Codo	Sube y baja el antebrazo	3
4	Inclinación	Sube y baja la inclinación de la herramienta final (pinza)	4+5
5	Giro	Gira la herramienta final (pinza)	4+5

1.3.1.2.2 Alcance de trabajo.

La longitud de los eslabones y el grado de rotación de las articulaciones determinan el alcance de trabajo del robot. Las Figuras 1.3.1.6 y 1.3.1.7 muestran las dimensiones y el alcance del SCORBOT-ER Vplus.

La base del robot se asegura normalmente a una superficie fija de trabajo. Puede, sin embargo, instalarse sobre una base lineal, lo que aumentaría de gran forma el alcance de trabajo para aplicaciones en celdas de manufactura.

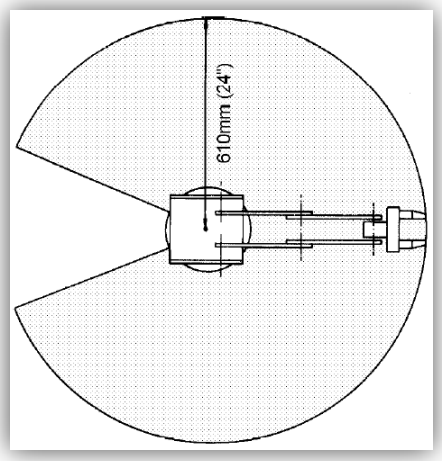


Figura 1.3.1.6 Alcance de trabajo (vista superior).

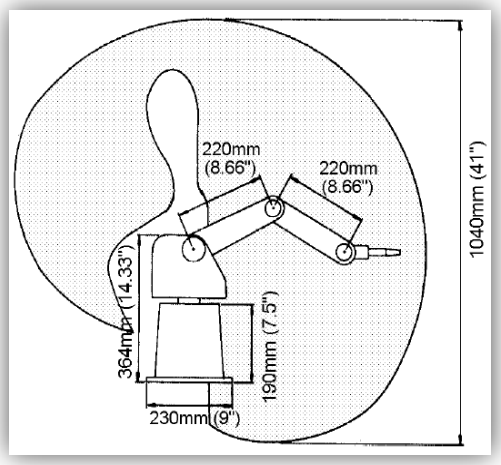


Figura 1.3.1.7 Alcance de trabajo (vista lateral).

1.3.1.2.3 Actuadores.

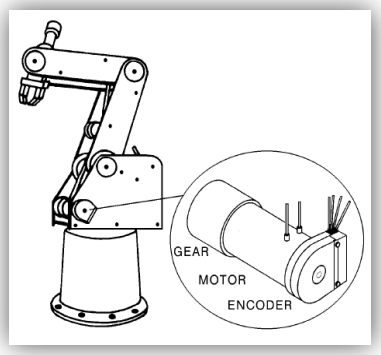


Figura 1.3.1.8 Motor.

Los cinco ejes y la pinza del robot son operados por servo motores de D.C. La dirección de revolución del motor es determinada por la polaridad del voltaje: el voltaje positivo hace girar el motor en una dirección, y el negativo en la dirección opuesta. Cada motor lleva instalado un codificador para su control en bucle cerrado.

En la Tabla 1.3.1.3 se muestran algunas especificaciones del los motores.

Tabla 1.3.1.3 Especificaciones de los Motores.	
Potencia de motores (ejes 1-6)	70W potencia en pico de par.
Velocidad máxima	600 mm/seg.

1.3.1.2.4 Transmisiones.

Para mover las articulaciones del robot se utilizan motores eléctricos y varios tipos de transmisiones que a continuación se mencionan:

- Engranes que mueven los ejes de la base y hombro.
- Poleas y correas dentadas mueven el eje del codo.
- Poleas, correas dentadas y una unidad diferencial de engranes, al final del brazo, cambian la inclinación y el giro del eje de la muñeca.
- Una transmisión de husillo abre y cierra la pinza.

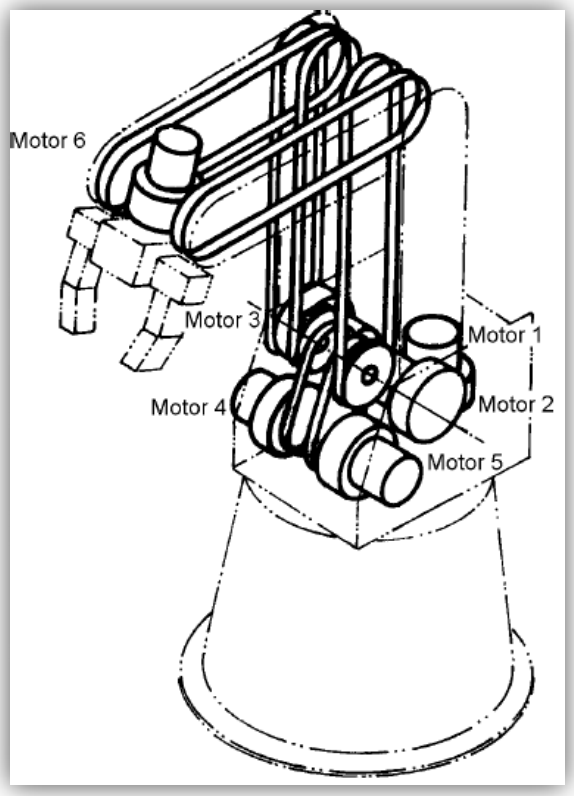


Figura 1.3.1.9 Transmisiones.

1.3.1.3 Sistema de Percepción.

1.3.1.3.1 Sensores del tipo interno.

La localización y movimiento de cada eje, se mide por un codificador electroóptico que se encuentra fijo al eje del motor. Cuando el eje del robot se mueve, el codificador generá una serie de impulsos eléctricos. El número de dichas señales es proporcional a la cantidad de movimiento del eje. El controlador lee estas señales y determina la distancia y dirección de movimiento de cada eje.

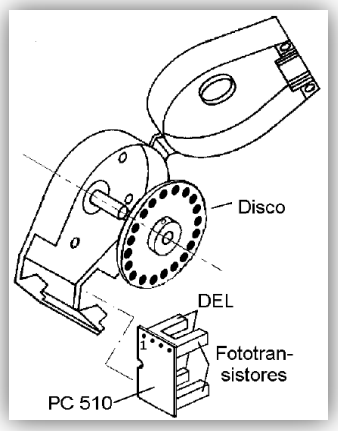
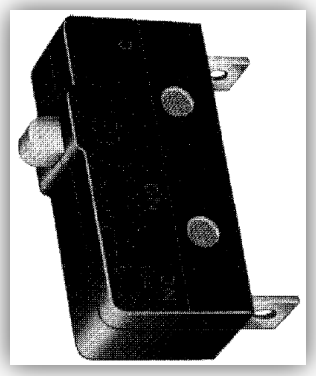


Figura 1.3.1.10 Codificador.

1.3.1.3.2 Sensores del tipo externo.

El SCORBOT-ER Vplus posee cinco microinterruptores, uno sobre cada eje. Durante el procedimiento de búsqueda de referencia o posición Inicio (Home), los ejes se mueven hasta que sus interruptores son activados.



Cuando todas las articulaciones están en la posición de referencia, el robot está en la posición Inicio. Este es el punto de referencia para la operación del robot. Cuando se activa el sistema, el robot es enviado a esa posición por medio de una rutina de software (Home).

Figura 1.3.1.11 Microinterruptor.

1.3.1.4 Equipo Periférico.

El SCORBOT-ER Vplus posee una pinza con mordazas cubiertas por almohadillas de goma. Dichas almohadillas pueden ser quitadas y reemplazadas por otras herramientas finales, tales como almohadillas de succión.

Tres engranajes cónicos forman el juego de engranajes que mueve la articulación de la muñeca. Cuando los motores 4 y 5 giran en la misma dirección, la inclinación de la muñeca sube y baja. Cuando los motores 4 y 5 giran en direcciones opuestas, la pinza gira en sentido horario o antihorario. Un husillo conectado directamente al motor 6 abre y cierra la pinza.

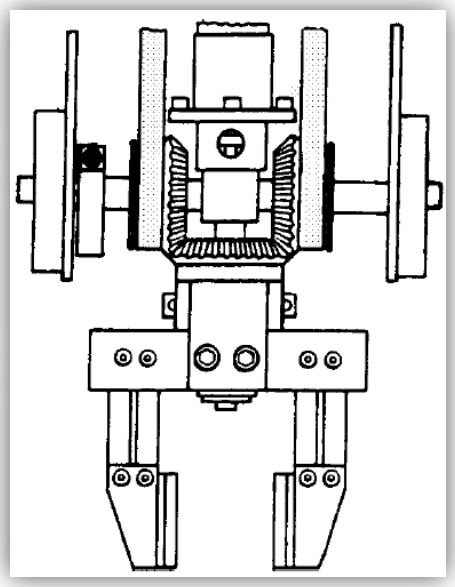


Figura 1.3.1.12 Pinza de Robot SCORBOT-ER Vplus.

1.3.2 Robot Scorbot-ER VII.

La arquitectura mecánica de este robot manipulador es del tipo antropomórfico (Figura 1.3.2.1), las partes que constituyen su sistema mecánico son base, cuerpo, brazo, codo, antebrazo y pinza; el sistema de percepción se caracteriza por sensores de tipo interno mediante un codificador y sensores de tipo externo con la implementación de una leva y microinterruptores. La Figura 1.3.2.2 representa esquemáticamente la estructura y composición del Robot Scorbot-ER VII.

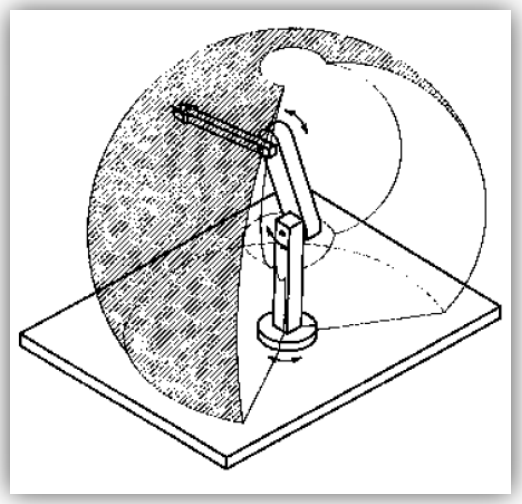


Figura 1.3.2.1 Arquitectura mecánica antropomorfa.

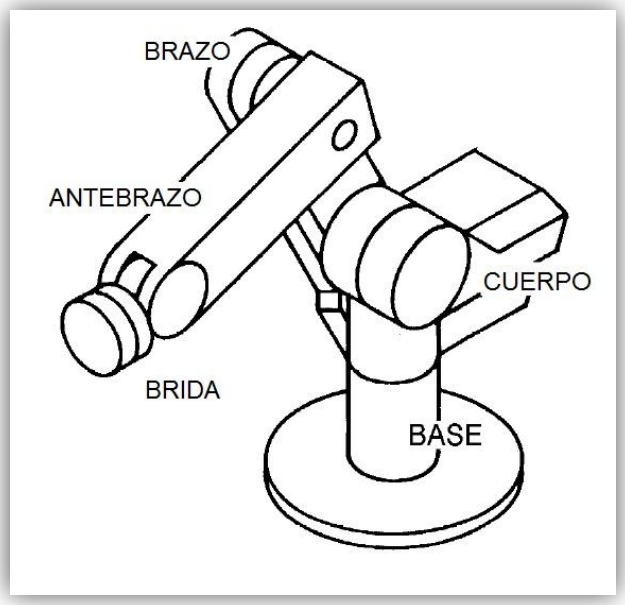


Figura 1.3.2.2 Partes del Robot Scorbot-ER VII.

1.3.2.1 Especificaciones.

En este tema se muestran las especificaciones y los componentes necesarios del brazo robot SCORBOT-ER VII.

Figura 1.3.2.3 Robot Scorbot-ER VII.

En la Tabla 1.3.2.1 se detallan las especificaciones del brazo robot; la elaboración de esta tabla se realiza haciendo una abstracción de la información del manual de usuario, acerca de los requerimientos más importantes que sirven como preámbulo en la elaboración de este trabajo.

Tabla 1.3.2.1 Especificaciones del Robot SCORBOT-ER VII	
Estructura mecánica	Robot de articulación vertical
Número de ejes	Cinco ejes más pinza
Movimiento de ejes: ✸ Eje 1: Base ✸ Eje 2: Hombro ✸ Eje 3: Codo ✸ Eje 4: Inclinación de la pinza ✸ Eje 5: Giro de la pinza	 250º; 310º programado por el usuario 170º 225º 180º 360°
Rango de operación	690 mm sin pinza
Elemento terminal	Pinza neumática Pinza eléctrica servo controlada
Inicio (referencia)	Posición fija en cada eje, hallada por medio de microinterruptores
Realimentación	Codificadores ópticos en cada eje
Actuadores	Servo Motores de D.C.
Transmisión	Sistema de engranes armónicos y correas dentadas
Máxima carga de trabajo	2 Kg, incluyendo la pinza
Repetitividad	±0.2 mm
Peso	38 Kg
Rango de Temperatura	2º - 40º

1.3.2.2 Sistema Mecánico.

1.3.2.2.1 Estructura.

El SCORBOT-ER VII es un robot de estructura vertical articulada, con cinco grados de libertad. Cuando se le agrega la pinza, el robot posee seis grados de libertad. Este diseño permite a la herramienta final ser colocada y orientada arbitrariamente en un gran espacio de trabajo.

En las Figuras 1.3.2.4 y 1.3.2.5 se identifican los eslabones y las articulaciones del brazo mecánico.

Los movimientos de las articulaciones son descritos en la Tabla 1.3.2.2.

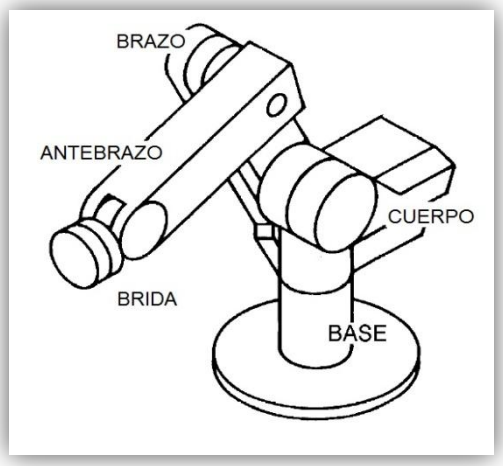


Figura 1.3.2.4 Eslabones del Robot.

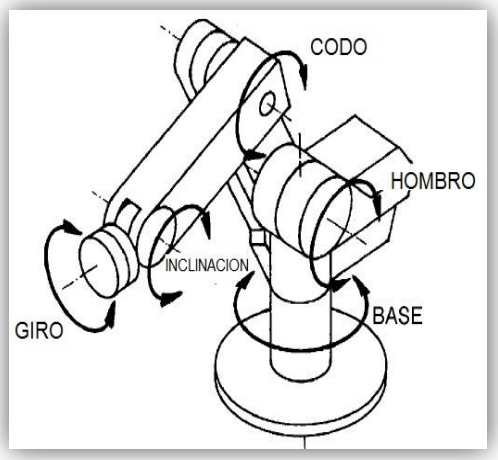


Figura 1.3.2.5 Articulaciones del Robot.

Tabla 1.3.2.2

Eje n°	Nombre	Movimiento	Motor
1	Base	Rotación del cuerpo	1
2	Hombro	Sube y baja el brazo	2
3	Codo	Sube y baja el antebrazo	3
4	Inclinación	Sube y baja la inclinación de la herramienta final (pinza)	4
5	Giro	Gira la herramienta final (pinza)	5

1.3.2.2.2 Alcance de trabajo.

La longitud de los eslabones y el grado de rotación de las articulaciones determinan el alcance de trabajo del robot. Las Figuras 1.3.2.6 y 1.3.2.7 muestran las dimensiones y el alcance del SCORBOT-ER VII.

La base del robot se asegura normalmente a una superficie fija de trabajo. Puede, sin embargo, instalarse sobre una base lineal, lo que aumentaría de gran forma el alcance de trabajo para aplicaciones en celdas de manufactura.

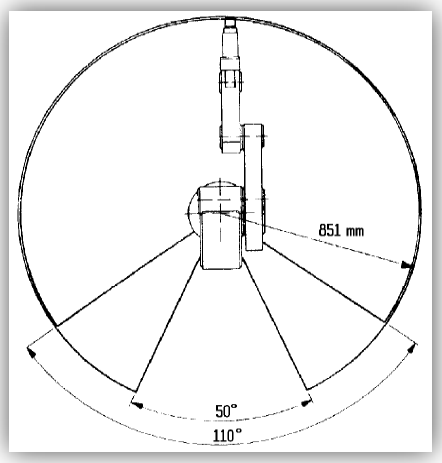


Figura 1.3.2.6 Alcance de trabajo (vista superior).

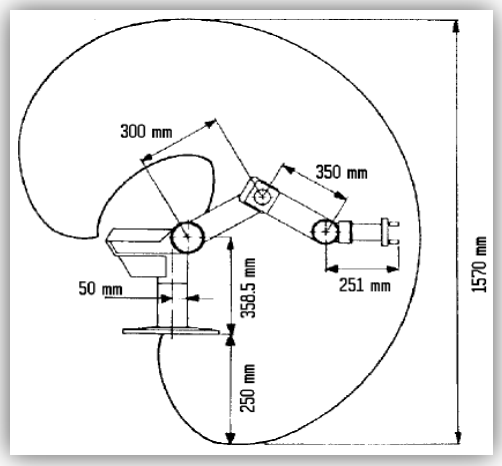


Figura 1.3.2.7 Alcance de trabajo (vista lateral).

1.3.2.2.3 Actuadores.

Los cinco ejes del robot son operados por servo motores de D.C. Estos actuadores convierten las señales del controlador (energía eléctrica) en rotaciones del eje del motor (energía mecánica).

En la Figura 1.3.2.8 se muestra la ubicación de los motores dentro del SCORBOT-ER VII. La dirección de revolución del motor es determinada por la polaridad del voltaje: el voltaje positivo hace girar el motor en una dirección, y el negativo en la dirección opuesta. Cada motor lleva instalado un codificador para su control en bucle cerrado.

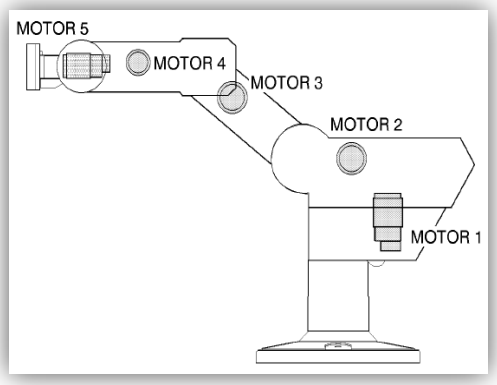


Figura 1.3.2.8 Ubicación de los Motores.

Los ejes 1, 2 y 3 del SCORBOT-ER VII son accionados por el tipo de motor de la Figura 1.3.2.9. Los ejes 4 y 5 son accionados por un motor más pequeño como el motor que se muestra en la Figura 1.3.2.10.

Estos motores son capaces de moverse a altas velocidades de revolución, a medida que las cargas en el par se incrementan; y con la implementación de un encoder adjunto, ayuda a alcanzar un nivel alto en la resolución. En la Tabla 1.3.2.3 se muestran las especificaciones de los motores.

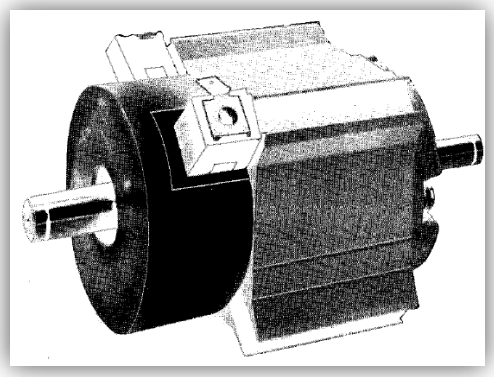


Figura 1.3.2.9 Motor de los ejes 1, 2 y 3.

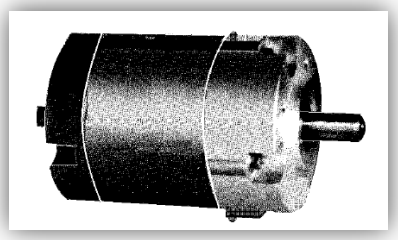


Figura 1.3.2.10 Motor de los ejes 4 y 5.

Tabla 1.3.2.3 Especificaciones de los motores.		
	Motores de Ejes 1, 2 y 3	**Motores de Ejes 4 y 5**
Par máximo	1.01 N*m	0.2 N*m
Par con Fricción	0.23 N*m	0.09 N*m
Velocidad máxima	4000 rpm	4500 rpm
Peso	1.29 Kg.	0.28 Kg.

1.3.2.2.4 Transmisión.

Los principales elementos del sistema de transmisión del SCORBOT-ER VII son los motores eléctricos, el sistema de engranes de accionamiento armónico, poleas y correas dentadas.

La Figura 1.3.2.11 muestra el sistema de accionamiento para los ejes 1 a 4 del SCORBOT-ER VII. La transmisión del eje 5 no contiene la polea y la correa dentada, sólo se utiliza el sistema de engranes de accionamiento armónico acoplado directamente al motor.

En la Figura 1.3.2.12 se muestra la transmisión de cada articulación del SCORBOT-ER VII, recordando que para el eje 5 sólo se utiliza el sistema de engranes de accionamiento armónico acoplado directamente al motor.

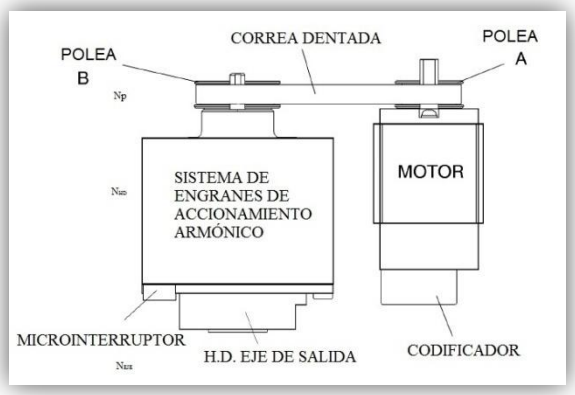


Figura 1.3.2.11 Sistema de Transmisión.

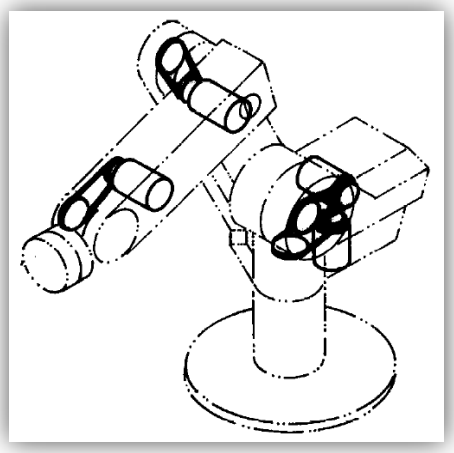


Figura 1.3.2.12 Transmisiones.

Sistema de Engranes de Accionamiento Armónico.

El sistema de engranes de accionamiento armónico, se muestra en la Figura 1.3.2.13.

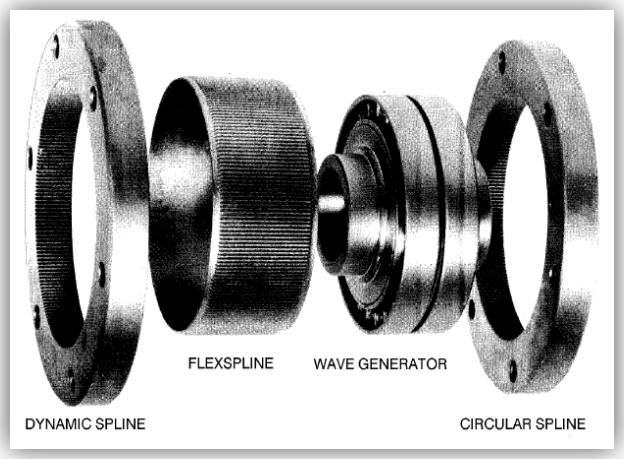


Figura 1.3.2.13 Estructura de un sistema de engranes de accionamiento armónico.

La utilización de este sistema en el SCORBOT-ER VII tiene cuatro principales componentes:

- Circular Spline: Anillo de acero, con dentado interno, normalmente fijo a la articulación del robot.
- Wave Generator: Disco rígido, ligeramente elíptico, que está conectado al eje de entrada, sobre un rodamiento.
- Flexspline: Cilindro flexible muy delgado, dentado externo, normalmente conectado al eje de salida.
- Dynamic Spline: Anillo circular sólido con dentado interno.

El dentado externo del anillo Flexspline tiene un tamaño casi igual al dentado interno del anillo Circular Spline, excepto que este dispone de dos dientes de más, y los dientes sólo engranan cuando el Wave Generator empuja el anillo Flexspline hacia fuera.

Como el Wave Generator es elíptico, el anillo Flexspline es empujado hacia afuera dos posiciones. Cuando el motor gira el eje de entrada, el Wave Generator gira con este y las posiciones donde los dientes están engranados giran con él. Sin embargo, como el anillo Flexspline tiene dos dientes menos, debe girar ligeramente hacia atrás cuando el Wave Generator gira hacia delante. En cada giro completo del eje de entrada, el anillo Flexspline se mueve dos dientes hacia atrás. Las Figuras 1.3.2.14 y 1.3.2.15 muestran diferentes pasos de este proceso.

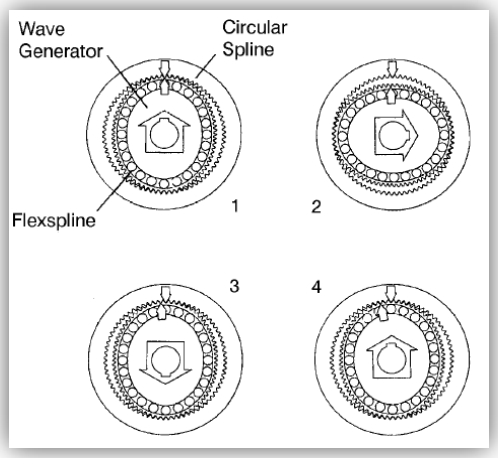


Figura 1.3.2.14 Operación del engranaje armónico.

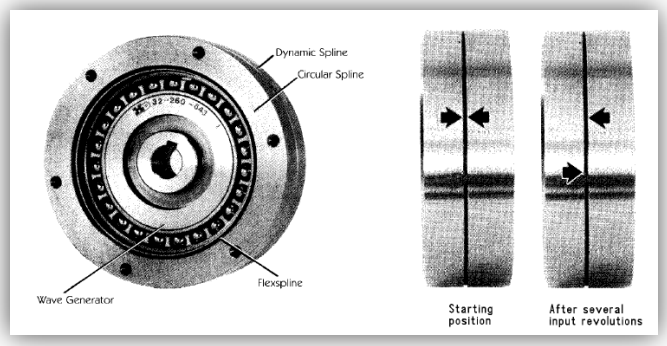


Figura 1.3.2.15 Operación del engranaje armónico.

Relación de Transmisión del Sistema Armónico.

Como en todos los engranes, existe una relación de transmisión en el sistema de engranes de accionamiento armónico; el fabricante proporciona la siguiente información en el manual de usuario.

Si el número de dientes del anillo Flexspline es N_f, entonces para varias revoluciones del eje de entrada, el eje de salida gira $2/N_f$ de una revolución. Por lo tanto:

$$Relación = \frac{1}{\left(\frac{2}{N_f}\right)} = \frac{N_f}{2}$$

La relación de transmisión de cada sistema armónico del SCORBOT -ER VII para cada eje son las siguientes:

- Eje 1 — 161:1
- Eje 2 — 160:1
- Eje 3 — 160:1
- Eje 4 — 100:1
- Eje 5 — 100:1

Relación de Transmisión de los Ejes.

Como referencia de la Figura 1.3.2.11, la transmisión de los ejes 1 a 4 consiste de dos elementos: la correa dentada y el sistema de engranes armónico.

La relación de transmisión completa del sistema de transmisión se expresa como:

$$N_T * N_{HD} = N_{EJE}$$

Donde:

N_T es la relación de transmisión en la correa dentada: $\frac{Polea\ B}{Polea\ A}$

N_{HD} es la relación de transmisión del sistema de engranes armónico.

N_{EJE} es la relación de transmisión total del sistema.

En la Tabla 1.3.2.4 se muestran las relaciones de transmisión de los ejes del SCORBOT-ER VII.

Tabla 1.3.2.4 Relaciones de Transmisión de los Ejes

Eje	N_T	N_{HD}	N_{EJE}
1	1.33 : 1	161 : 1	214.13 : 1
2	1.52 : 1	160 : 1	243.8 : 1
3	1.33 : 1	160 : 1	213.33 : 1
4	1.8 : 1	100 : 1	180 : 1
5		100 : 1	100 : 1

1.3.2.3 Sistema de Percepción.

1.3.2.3.1 Sensores del tipo interno.

La localización y movimiento de cada eje, se mide por un codificador electroóptico que se encuentra fijo al eje del motor. Como se muestra en la Figura 1.3.2.16. Cuando el eje del robot se mueve, el codificador genera una serie de impulsos eléctricos. El número de dichas

señales es proporcional a la cantidad de movimiento del eje. El controlador lee estas señales y determina la distancia y dirección de movimiento para cada eje.

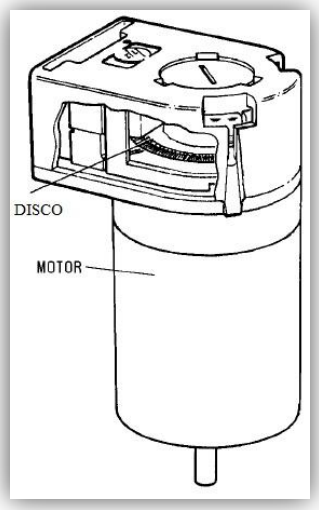


Figura 1.3.2.16 Codificador.

El codificador utilizado en el SCORBOT-ER VII contiene un diodo emisor de luz (LED) como fuente de luz. Enfrente del LED la luz es detectada por un circuito integrado receptor. Este circuito integrado contiene un conjunto de varios fotodetectores y la circuitería necesaria para producir una señal digital.

Un disco perforado gira entre el LED emisor y el receptor. Cuando el disco del decodificador gira entre el emisor y el receptor, el rayo de luz es interrumpido por una serie de ranuras y ventanas del disco, resultando una serie de impulsos recibidos por el receptor. Los discos codificadores utilizados en el SCORBOT-ER VII tienen 512 ranuras, como se muestra en la Figura 1.3.2.17. Una ranura adicional en el disco codificador sirve para generar un impulso índice (C-pulse) en cada rotación completa del disco. El impulso índice sirve para determinar la posición Home del eje.

Los fotodetectores están situados de tal manera que unas veces detectan luz y otras no. Las salidas del fotodetector dan las señales A, $\bar{A}$, B y $\bar{B}$. Los comparadores reciben estas señales y producen las señales finales de salida para los canales A y B. La salida del canal A esta en cuadratura con el canal B (desfasado 90º), como muestra la Figura 1.3.2.18.

Cuando la rotación del disco va en sentido antihorario (visto desde atrás), el canal A dirige al B. Cuando la rotación del disco es en sentido horario, el canal B dirige al A.

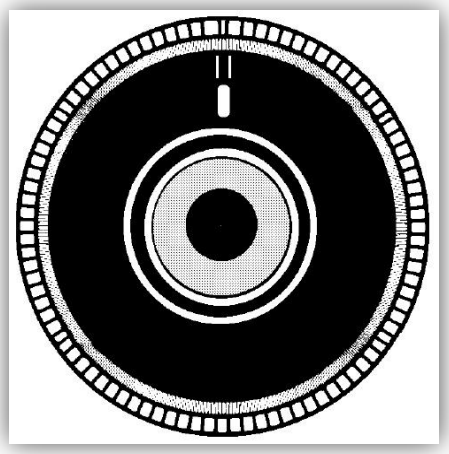


Figura 1.3.2.17 Disco codificador.

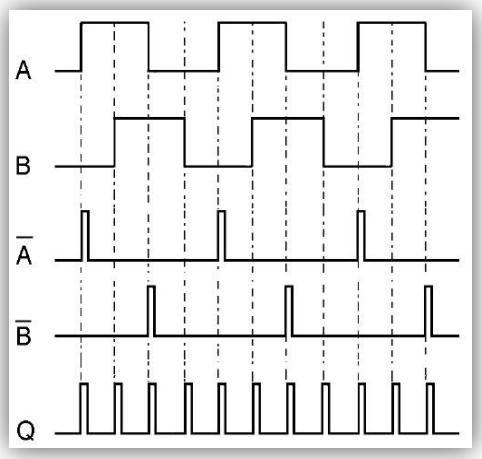


Figura 1.3.2.18 Señales de salida del codificador.

Resolución del Codificador.

De la señal recibida, el controlador del robot mide cuatro pulsos por cada ranura del codificador, lo que cuadruplica la resolución efectiva del codificador.

La resolución del codificador se expresa como:

$$S_E = \frac{360°}{n}$$

Donde:

S_E es la resolución del codificador.

n es el número de impulsos por revolución del codificador.

El codificador utilizado por el SCORBOT-ER VII tiene 512 ranuras, generando 2048 impulsos por revolución. La resolución del codificador es, por lo tanto:

$$S_E = \frac{360°}{2048} = 0.176°$$

Cuando la resolución del codificador se divide por la relación total de transmisión del eje, se obtiene la resolución del eje.

Como el codificador va fijo al eje del motor, la resolución del eje se expresa como:

$$S_{ARTICULACIÓN} = \frac{S_E}{N_{EJE}}$$

De este modo, por ejemplo, la resolución del eje 3 del SCORBOT-ER VII será:

$$S_{A3} = \frac{0.176°}{213.33} = 0.000825°$$

La resolución es el incremento mínimo posible que el controlador puede identificar y teóricamente controlar. La precisión del eje está afectada por factores tales como juego mecánico, flexibilidad mecánica y variaciones de control.

1.3.2.3.2 Sensores del tipo externo.

El SCORBOTER VII posee microinterruptores (Figura 1.3.2.19), los cuales se utilizan para limitar el movimiento del robot (Final de Carrera) y para la posición Inicio (Home) del robot (Interruptores Home). El interruptor es parte de un circuito eléctrico del brazo del robot y es independiente del controlador.

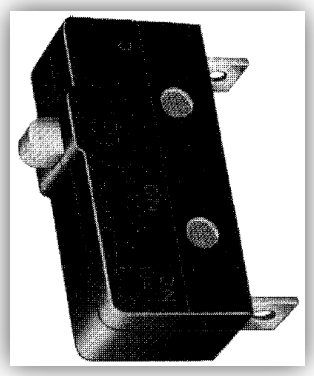


Figura 1.3.2.19 Microinterruptor.

Final de Carrera.

Los interruptores de final de carrera previenen que las articulaciones se muevan más allá de sus límites funcionales. Cuando el software de control del robot no puede detener el eje, el final de carrera sirve para detener su movimiento en los extremos del rango de trabajo.

Cada uno de los ejes del 1 al 4 dispone de dos interruptores de final de carrera, uno para cada límite del rango de movimiento. Estos interruptores están montados en un disco sujetos al bastidor del robot. En la Figura 1.3.2.20 se muestran los microinterruptores en el eje 3.

La salida del eje de la transmisión armónica tiene integrada una leva, a medida que la articulación se mueve, la leva activa o desactiva el botón del microinterruptor de final de carrera. Cuando se activa el final de carrera, se produce un error de control, resultando la activación COFF y un mensaje de protección de impacto. Se debe activar de nuevo CON y mover el brazo del robot manualmente (usando el teclado o la botonera de enseñanza) fuera de la condición de impacto.

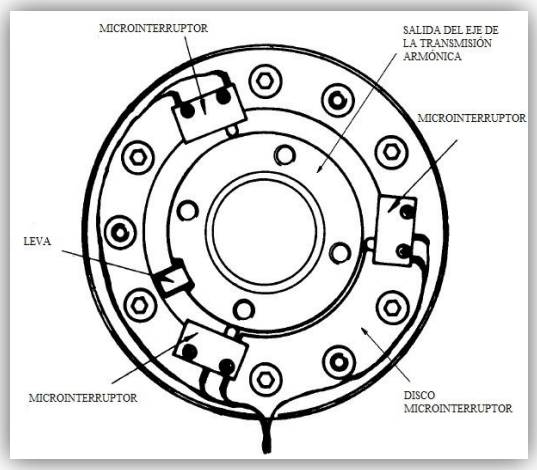


Figura 1.3.2.20 Microinterruptores en eje.

Mientras no se alcance ningún interruptor de final de carrera el eje puede girar libremente en cualquier sentido.

Antes de llegar a pulsar el interruptor límite, todos los ejes están protegidos por medio de límites software.

El eje 5 (giro) no tiene final de carrera, puede girar sin fin. Cuando un dispositivo de agarre esta en el eje 5, sus movimientos son controlados y limitados por medio de software (codificador) solamente.

Interruptores Home.

El SCORBOT-ER VII utiliza un interruptor óptico para encontrar la posición de referencia de cada eje del brazo del robot. El microinterruptor de referencia se instala en el mismo disco que los microinterruptores de finales de carrera.

Durante el procedimiento de referencia (Home), el controlador mueve un eje a la vez. El eje se mueve hasta que el interruptor óptico detecta la posición de Home y este, entonces, envía una señal al controlador. Una vez que el detector de Home ha encontrado la posición, el eje se sigue moviendo, a pequeña velocidad, hasta que el codificador produce un impulso índice. En este punto, la posición Home se ha encontrado y se detiene el eje, posteriormente el proceso Home se iniciará en el siguiente eje.

Cuando todas las articulaciones están en home, el robot está en la posición de inicio. Este es el punto de referencia para el funcionamiento del robot. Cada vez que el sistema este encendido, el robot debe ser enviado a esta posición, por medio de una rutina de software (Home).

1.3.2.4 Equipo Periférico.

El SCORBOT-ER VII puede ser equipado con un dispositivo de agarre unido a la brida en el extremo del brazo del robot, cuyo diseño se muestra en La Figura 1.3.2.21.

Los dispositivos de agarre que el fabricante recomienda son la utilización de una Pinza Neumática o Pinza Eléctrica Servo Controlada. Las cuales deben cumplir con el diseño de la brida para un correcto acoplamiento.

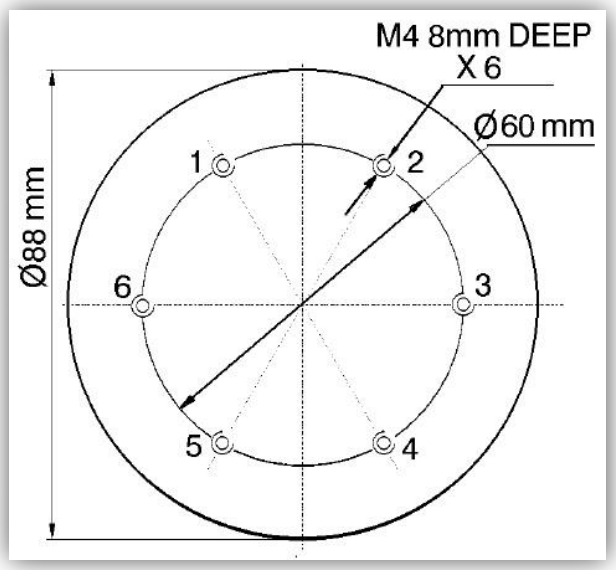


Figura 1.3.2.21 Superficie de montaje de la pinza.

1.3.3 Robot Scora-ER 14Pro.

La arquitectura mecánica de este robot manipulador es del tipo scara (Figura 1.3.3.1), las partes que constituyen su sistema mecánico son base, columna, eslabón 1, eslabón 2, traslación y pinza; el sistema de percepción se caracteriza por sensores de tipo interno mediante un codificador y sensores de tipo externo con la implementación de una leva y microinterruptores. La Figura 1.3.3.2 representa esquemáticamente la estructura y composición del Robot Scora-ER 14Pro.

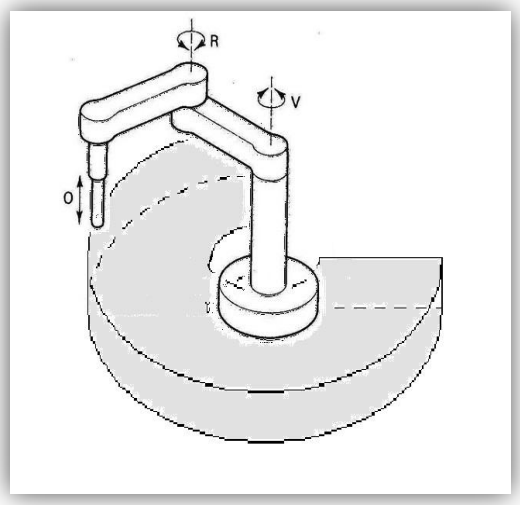


Figura 1.3.3.1 Arquitectura mecánica scara.

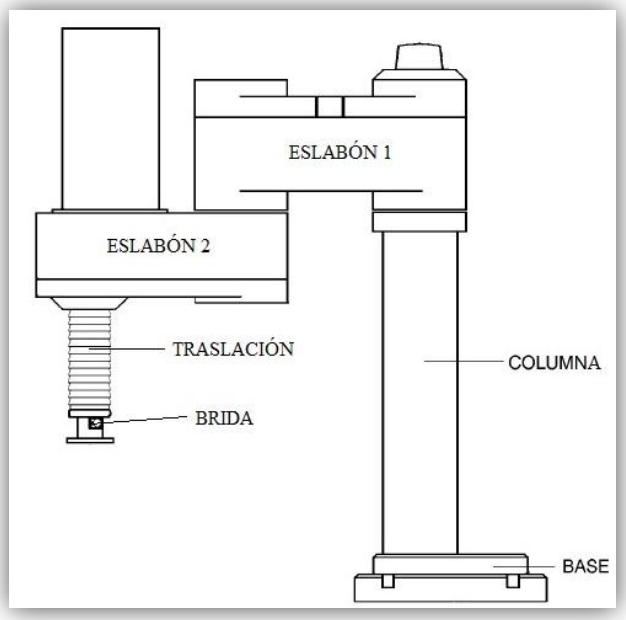


Figura 1.3.3.2 Partes del Robot Scora-ER 14Pro.

1.3.3.1 Especificaciones.

En este tema se muestran las especificaciones y los componentes necesarios del brazo robot SCORA-ER 14Pro.

Figura 1.3.3.3 Robot Scora-ER 14Pro.

En la Tabla 1.3.3.1 se detallan las especificaciones del brazo robot; la elaboración de esta tabla se realiza haciendo una abstracción de la información del manual de usuario, acerca de los requerimientos más importantes que sirven como preámbulo en la elaboración de este trabajo.

Tabla 1.3.3.1 Especificaciones del Robot SCORA-ER 14Pro	
Estructura mecánica	Robot de articulación horizontal (scara)
Longitud del brazo Eslabón 1 Eslabón 2	 270 mm 230 mm
Movimiento de ejes: ✸ Eje 1: Rotación ✸ Eje 2: Rotación ✸ Eje 3: Traslación ✸ Eje 4: Giro; sin cable en la pinza: o Giro; con cable en la pinza: Eslabón 1 y 2 combinado	 301.5º 230.5º 198 mm Sin restricción ±527º
Rango de operación	Mínimo 250 mm Máximo 500 mm
Elemento terminal	Pinza neumática Pinza eléctrica servo controlada
Inicio (referencia)	Posición fija en cada eje, hallada por medio de microinterruptores
Realimentación	Codificadores ópticos en cada eje
Actuadores	Servo Motores de D.C.
Transmisión	Sistema de engranes armónicos y correas dentadas
Máxima carga de trabajo	3 Kg, incluyendo la pinza y baja aceleración
Repetitividad	±0.05 mm
Peso	45 Kg
Rango de Temperatura	2º - 40º

1.3.3.2 Sistema Mecánico.

1.3.3.2.1 Estructura.

El SCORA-ER 14Pro es un robot de estructura horizontal articulada.

El movimiento de las dos primeras articulaciones determina la posición del efector final en el plano XY. La tercer articulación es prismática y determina la altura (coordenada Z) de la pinza.

En la Figura 1.3.3.4 se identifican las articulaciones y eslabones del brazo mecánico. Cada articulación está impulsada por un servomotor de D.C. a través de sistemas de transmisión.

Los movimientos de las articulaciones son descritos en la Tabla 1.3.3.2.

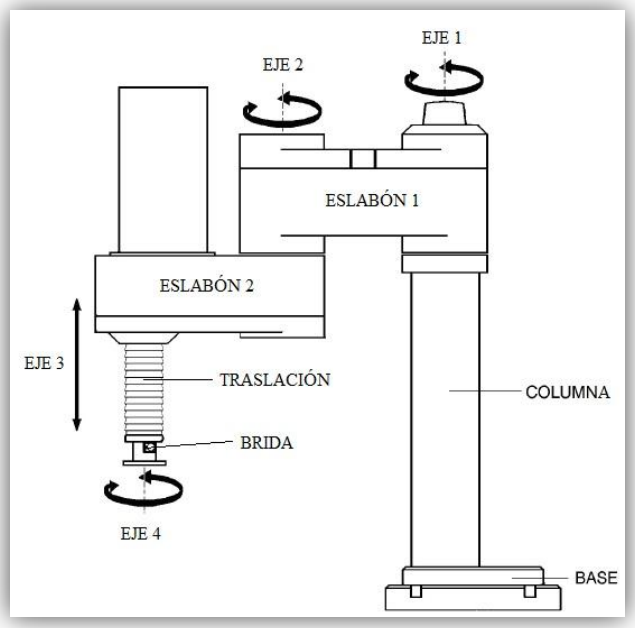


Figura 1.3.3.4 Articulaciones y eslabones del Robot.

Tabla 1.3.3.2	
Eje n°	**Movimiento**
1	Rotación del eslabón 1 en el plano horizontal (XY)
2	Rotación del eslabón 2 en el plano horizontal (XY)
3	Traslación (sube y baja) del efector final en el eje Z
4	Rotación de la herramienta final (pinza)

1.3.3.2.2 Alcance de trabajo.

La longitud de los eslabones y el grado de rotación de las articulaciones determinan el alcance de trabajo del robot. Las Figuras 1.3.3.5 y 1.3.3.6 muestran las dimensiones del SCORA-ER 14Pro, mientras que en la Figura 1.3.3.7 se puede observar una vista superior del espacio de trabajo del robot.

La base del robot se asegura normalmente a una superficie fija de trabajo.

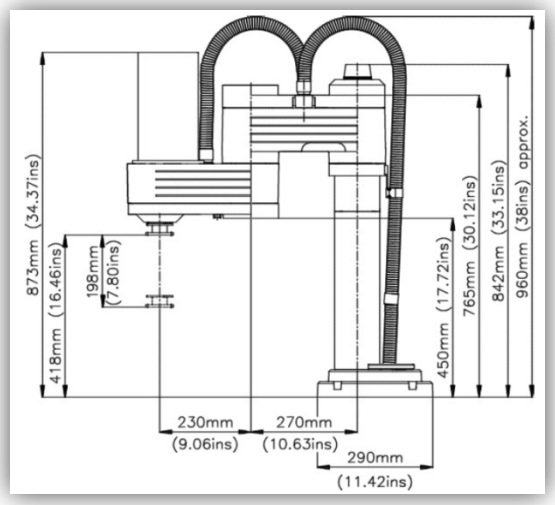


Figura 1.3.3.5 Dimensiones (vista lateral).

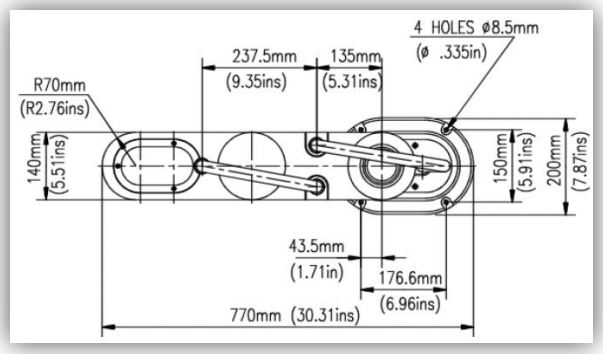


Figura 1.3.3.6 Dimensiones (vista superior).

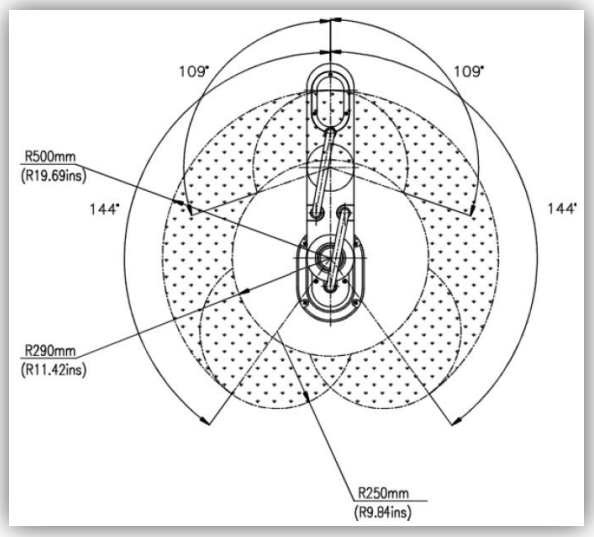


Figura 1.3.3.7 Alcance de trabajo (vista superior).

1.3.3.2.3 Actuadores.

Los cinco ejes del robot son operados por servo motores de D.C. Estos actuadores convierten las señales del controlador (energía eléctrica) en rotaciones del eje del motor (energía mecánica).

En la Figura 1.3.3.8 se muestra la ubicación de los motores dentro del SCORA-ER 14Pro. La dirección de revolución del motor es determinada por la polaridad del voltaje: el voltaje positivo hace girar el motor en una dirección, y el negativo en la dirección opuesta. Cada motor lleva instalado un codificador para su control en bucle cerrado.

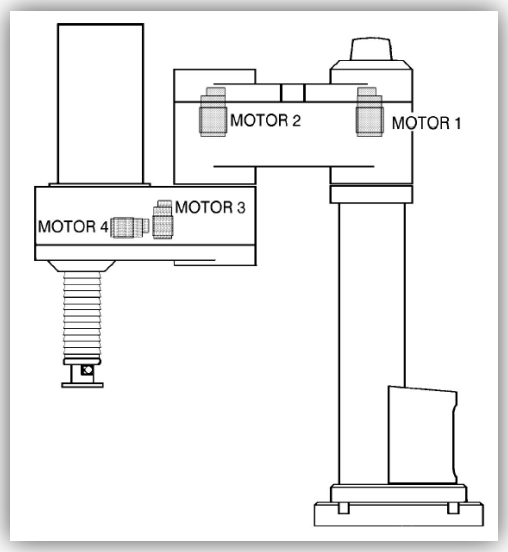


Figura 1.3.3.8 Ubicación de los Motores.

Los ejes 1 y 2 del SCORA-ER 14Pro son accionados por el tipo de motor de la Figura 1.3.3.9. Los ejes 3 y 4 son accionados por un motor más pequeño como el motor que se muestra en la Figura 1.3.3.10.

Estos motores son capaces de moverse a altas velocidades de revolución, a medida que las cargas en el par se incrementan; y con la implementación de un codificador adjunto, ayuda a alcanzar un nivel alto en la resolución. La Tabla 1.3.3.3 muestra las especificaciones de los motores.

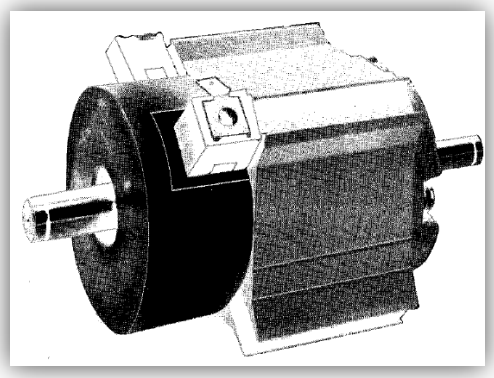


Figura 1.3.3.9 Motor de los ejes 1 y 2.

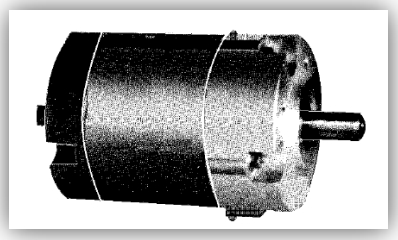


Figura 1.3.3.10 Motor de los ejes 3 y 4.

Tabla 1.3.3.3 Especificaciones de los motores.		
	Motores de Ejes 1 y 2	**Motores de Ejes 3 y 4**
Par máximo	1.01 N*m	0.2 N*m
Par con Fricción	0.23 N*m	0.09 N*m
Velocidad máxima	4000 rpm	4500 rpm
Peso	1.29 Kg.	0.28 Kg.

1.3.3.2.4 Transmisión.

El SCORA-ER 14Pro utiliza diferentes transmisiones mecánicas para transferir el movimiento de los motores a las articulaciones.

La estructura y el funcionamiento de los diversos componentes utilizados para conducir los ejes del robot se describen en este subtema.

Ejes 1 y 2.

Los principales componentes del sistema de transmisión para los ejes 1 y 2 son codificador, motor eléctrico, acoplamiento y el sistema de engranes de accionamiento armónico, en la Figura 1.3.3.11 se muestra un representación estructural de estos componentes.

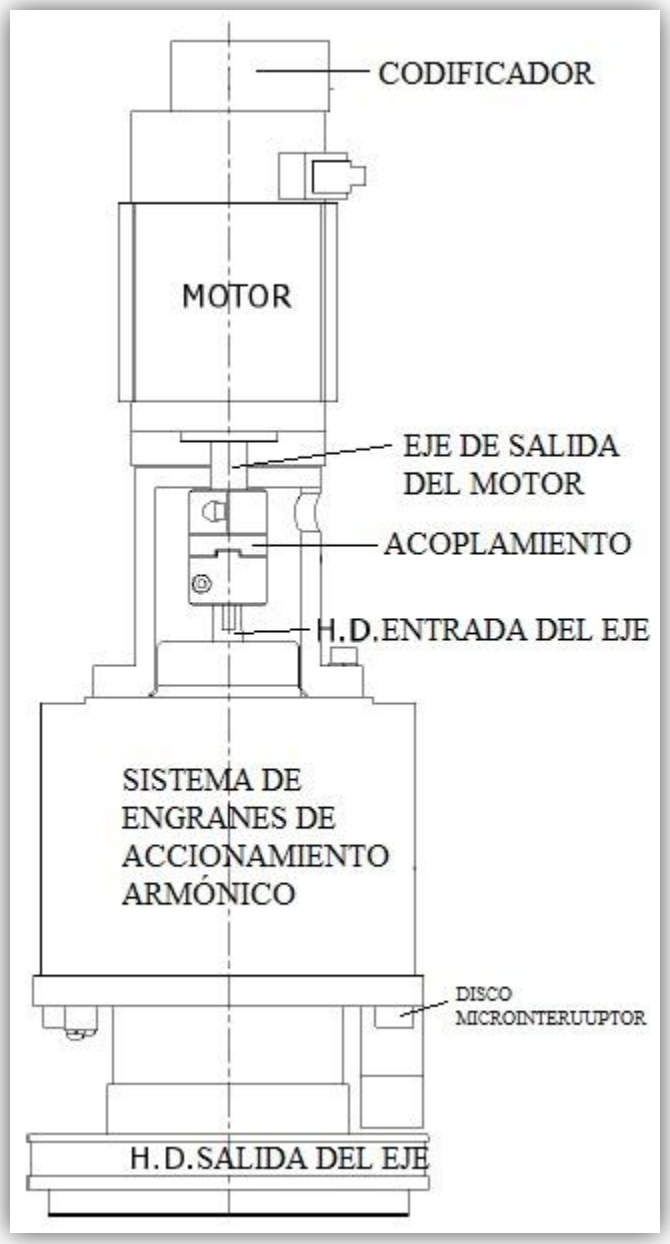


Figura 1.3.3.11 Sistema de transmisión ejes 1 y 2.

Eje 3.

El sistema de transmisión para el eje 3 produce un movimiento lineal en el eje Z. Sus principales componentes son el motor eléctrico, poleas, correa dentada, rodamientos y husillo de avance con tuerca, como se muestra en la Figura 1.3.3.12.

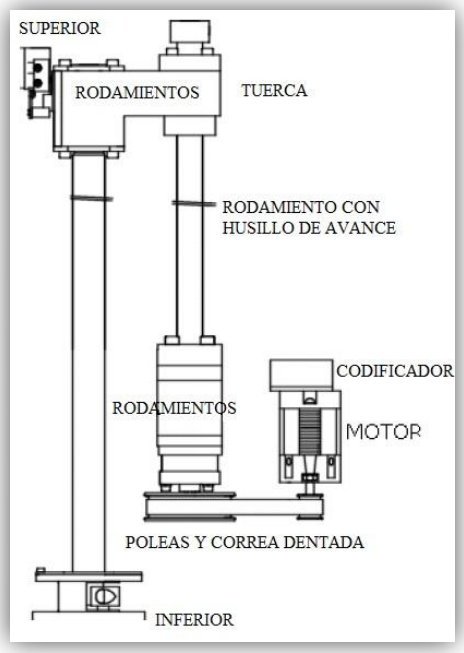


Figura 1.3.3.12 Sistema de transmisión eje 3.

Eje 4.

El sistema de transmisión para el eje 4 (Z-giro) produce una rotación en el extremo del efector sobre la brida. Sus componentes principales son el motor eléctrico, un sistema de engranes de tornillo sin fin y rueda dentada, como se muestra en la Figura 1.3.3.13.

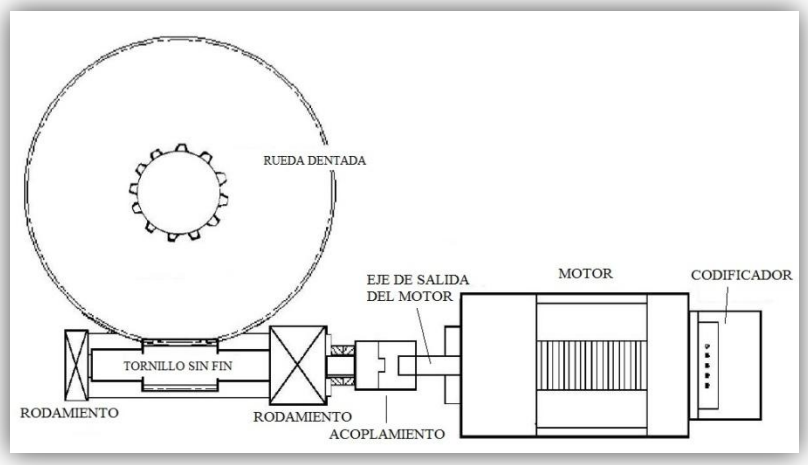


Figura 1.3.3.13 Sistema de transmisión eje 4.

Sistema de Engranes de Accionamiento Armónico.

El sistema de engranes de accionamiento armónico, se muestra en la Figura 1.3.3.14.

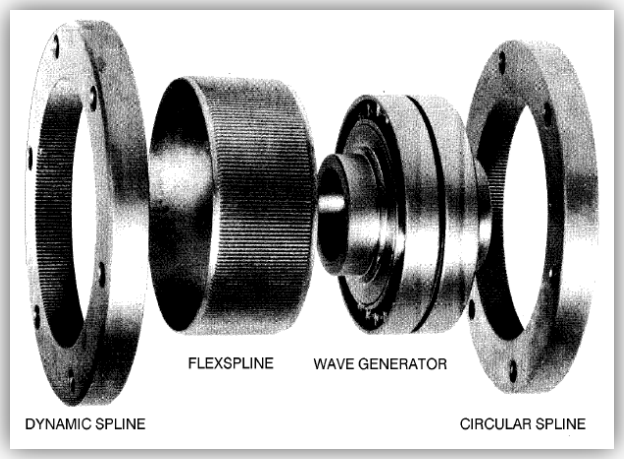


Figura 1.3.3.14 Estructura de un sistema de engranes de accionamiento armónico.

La utilización de este sistema en el SCORA-ER 14Pro tiene cuatro principales componentes:

- Circular Spline: Anillo de acero, con dentado interno, normalmente fijo a la articulación del robot.
- Wave Generator: Disco rígido, ligeramente elíptico, que está conectado al eje de entrada, sobre un rodamiento.
- Flexspline: Cilindro flexible muy delgado, dentado externo, normalmente conectado al eje de salida.
- Dynamic Spline: Anillo circular sólido con dentado interno.

El dentado externo del anillo Flexspline tiene un tamaño casi igual al dentado interno del anillo Circular Spline, excepto que este dispone de dos dientes de más, y los dientes sólo engranan cuando el Wave Generator empuja el anillo Flexspline hacia fuera.

Como el Wave Generator es elíptico, el anillo Flexspline es empujado hacia afuera dos posiciones. Cuando el motor gira el eje de entrada, el Wave Generator gira con este y las posiciones donde los dientes están engranados giran con él. Sin embargo, como el anillo Flexspline tiene dos dientes menos, debe girar ligeramente hacia atrás cuando el Wave Generator gira hacia delante. En cada giro completo del eje de entrada, el anillo Flexspline se mueve dos dientes hacia atrás. Las Figuras 1.3.3.15 y 1.3.3.16 muestran diferentes pasos de este proceso.

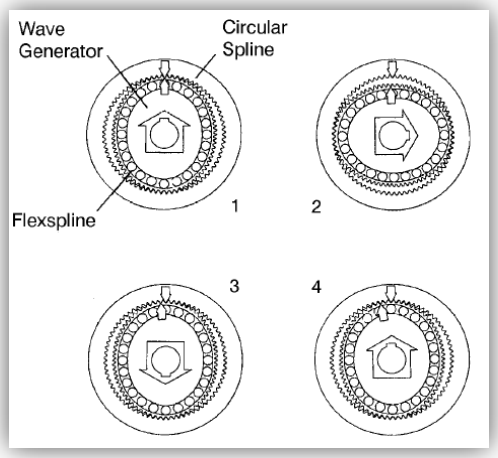


Figura 1.3.3.15 Operación del engranaje armónico.

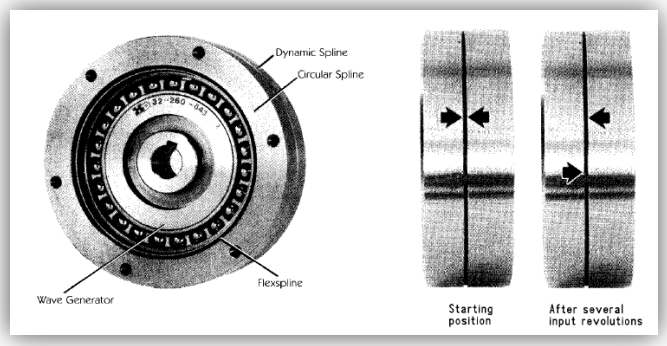


Figura 1.3.3.16 Operación del engranaje armónico.

Como en todos los engranes, existe una relación de transmisión en el sistema de engranes de accionamiento armónico; el fabricante proporciona la siguiente información en el manual de usuario.

Si el número de dientes del anillo Flexspline es N_f, entonces para varias revoluciones del eje de entrada, el eje de salida gira $2/N_f$ de una revolución. Por lo tanto:

$$Relación = \frac{1}{\left(\frac{2}{N_f}\right)} = \frac{N_f}{2}$$

La relación de transmisión de cada sistema armónico para el eje 1 y eje 2 del SCORA-ER 14Pro es:

- Eje 1 — 160:1
- Eje 2 — 160:1

Poleas y Correa Dentada.

La transmisión del eje 3 contiene dos poleas y una correa dentada, como se ilustra en la Figura 1.3.3.17. La relación de transmisión de este sistema para el SCORA-ER 14Pro es:

- Eje 3 — 80:21

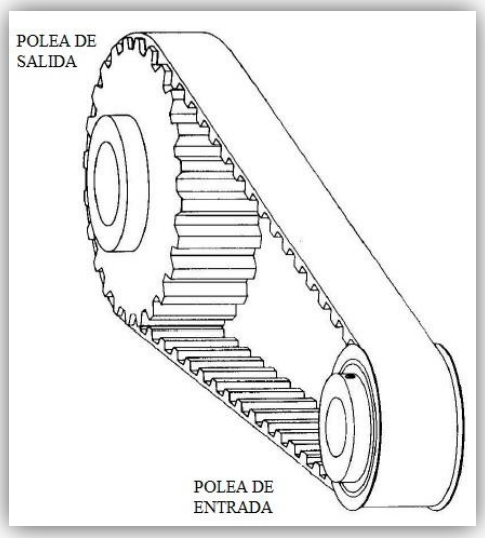


Figura 1.3.3.17 Poleas y Correa Dentada.

Rodamiento y Husillo de Avance con Tuerca.

Un rodamiento y un husillo de avance con tuerca convierten la rotación del motor en movimiento lineal para el eje Z.

En la transmisión, el husillo de avance se hace girar por el motor, y en el eje está conectada una tuerca como rodamiento, el tornillo gira y la tuerca se desplaza a lo largo de la longitud del tornillo, en la Figura 1.3.3.18 se ilustra el comportamiento de este sistema para el SCORA-ER 14Pro.

En la Figura 1.3.3.19, se puede observar el mecanismo implementado por este rodamiento y husillo de avance con tuerca. La tuerca se compone en su interior de una serie de bolas de rodamiento que circulan en un carril similar al del husillo. La tuerca a su vez transfiere desde un extremo al otro las bolas de rodamiento, por un tubo de retorno.

El husillo de avance del SCORA-ER 14Pro está equipado con un freno que detiene el movimiento en el eje Z cuando la alimentación del motor se corta.

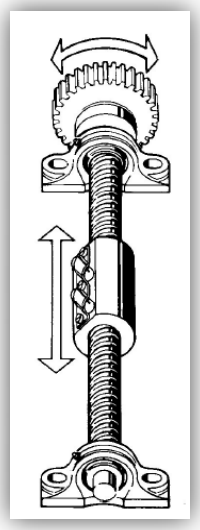


Figura 1.3.3.18 Poleas y Correa Dentada.

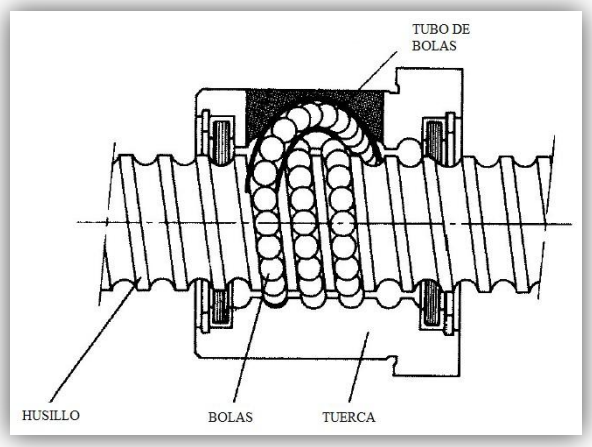


Figura 1.3.3.19 Rodamiento y Husillo de Avance con Tuerca.

Engrane Gusano.

Una transmisión de engrane tornillo sinfín, que se muestra en la Figura 1.3.3.20, se utiliza en el SCORA-ER 14Pro para transferir la rotación del motor 4 a la rotación del eje Z.

La relación de transmisión del engrane tornillo sinfín se define como:

$$\frac{\#\ Dientes\ de\ la\ rueda\ dentada}{\#\ De\ entradas\ del\ tornillo\ sin\ fín}$$

El engrane de gusano utilizado en el SCORA-ER 14Pro tiene 100 dientes y el eje tiene dos entradas. Por lo tanto, la transmisión del engrane tornillo sinfín tiene una relación de 50:1.

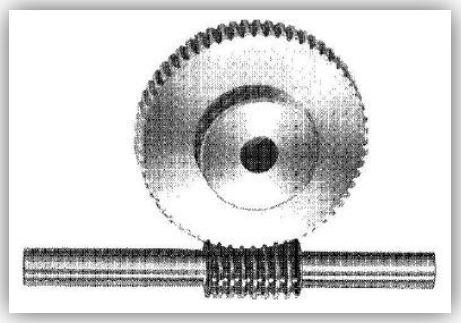


Figura 1.3.3.20 Transmisión engrane de gusano.

Rodamiento de Ranura.

El SCORA-ER 14Pro utiliza un rodamiento de ranura, que se muestra en la Figura 1.3.3.21, este sistema permite transmitir rotación en el eje Z, al mismo tiempo puede moverse linealmente sobre el eje.

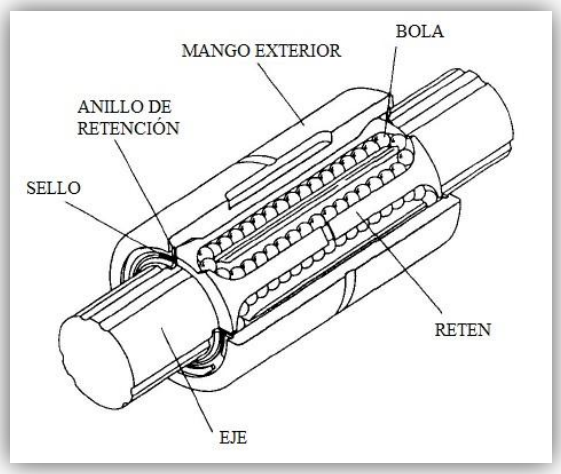


Figura 1.3.3.21 Rodamiento de Ranura.

1.3.3.3 Sistema de Percepción.

1.3.3.3.1 Sensores del tipo interno.

La localización y movimiento de cada eje, se mide por un codificador electroóptico que se encuentra fijo al eje del motor. Como se muestra en la Figura 1.3.3.22. Cuando el eje del robot se mueve, el codificador genera una serie de impulsos eléctricos. El número de dichas señales es proporcional a la cantidad de movimiento del eje. El controlador lee estas señales y determina la distancia y dirección de movimiento para cada eje.

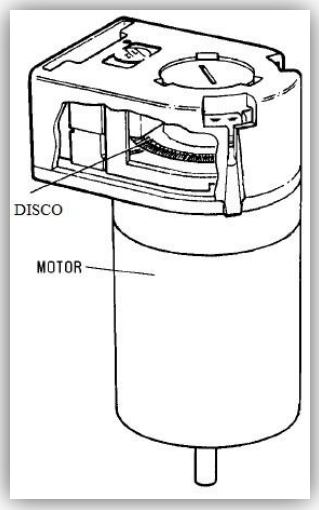


Figura 1.3.3.22 Codificador.

El codificador utilizado en el SCORA-ER 14Pro contiene un diodo emisor de luz (LED) como fuente de luz. Enfrente del LED la luz es detectada por un circuito integrado receptor. Este circuito integrado contiene un conjunto de varios fotodetectores y la circuitería necesaria para producir una señal digital.

Un disco perforado gira entre el LED emisor y el receptor. Cuando el disco del decodificador gira entre el emisor y el receptor, el rayo de luz es interrumpido por una serie de ranuras y ventanas del disco, resultando una serie de impulsos recibidos por el receptor. Los discos codificadores utilizados en el SCORA-ER 14Pro tienen 512 ranuras, como se muestra en la Figura 1.3.3.23. Una ranura adicional en el disco codificador sirve para generar un impulso índice (C-pulse) en cada rotación completa del disco. El impulso índice sirve para determinar la posición Home del eje.

Los fotodetectores están situados de tal manera que unas veces detectan luz y otras no. Las salidas del fotodetector dan las señales A, $\bar{A}$, B y $\bar{B}$. Los comparadores reciben estas señales

y producen las señales finales de salida para los canales *A* y *B*. La salida del canal *A* esta en cuadratura con el canal *B* (desfasado 90º), como muestra la Figura 1.3.3.24.

Cuando la rotación del disco va en sentido antihorario (visto desde atrás), el canal *A* dirige al *B*. Cuando la rotación del disco es en sentido horario, el canal *B* dirige al *A*.

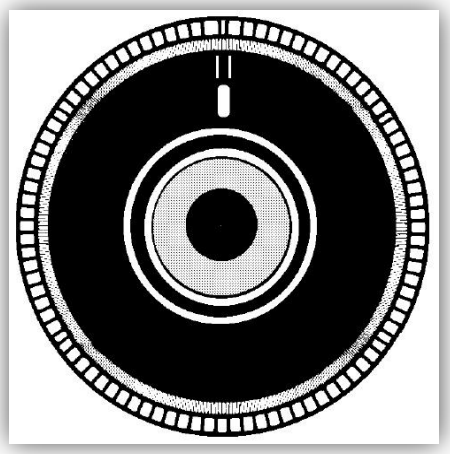


Figura 1.3.3.23 Disco codificador.

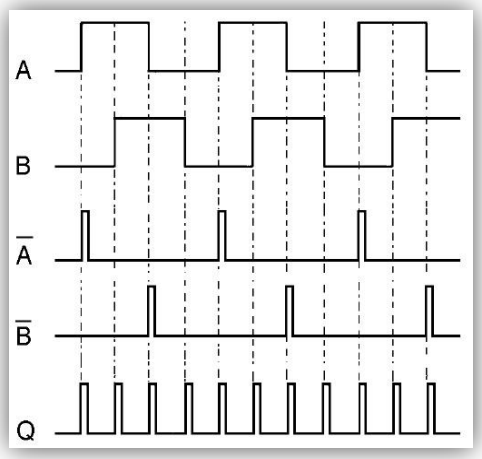


Figura 1.3.3.24 Señales de salida del codificador.

Resolución del Codificador.

De la señal recibida, el controlador del robot mide cuatro pulsos por cada ranura del codificador, lo que cuadruplica la resolución efectiva del codificador.

La resolución del codificador se expresa como:

$$S_E = \frac{360°}{n}$$

Donde:

S_E es la resolución del codificador.

n es el número de impulsos por revolución del codificador.

El codificador utilizado por el SCORA-ER 14Pro tiene 512 ranuras, generando 2048 impulsos por revolución. La resolución del codificador es, por lo tanto:

$$S_E = \frac{360°}{2048} = 0.176°$$

Cuando la resolución del codificador se divide por la relación total de transmisión del eje, se obtiene la resolución del eje.

Como el codificador va fijo al eje del motor, la resolución del eje se expresa como:

$$S_{ARTICULACIÓN} = \frac{S_E}{N_{EJE}}$$

De este modo, por ejemplo, la resolución del eje 2 del SCORA-ER 14Pro será:

$$S_{A3} = \frac{0.176°}{160} = 0.0011°$$

La resolución es el incremento mínimo posible que el controlador puede identificar y teóricamente controlar. La precisión del eje está afectada por factores tales como juego mecánico, flexibilidad mecánica y variaciones de control.

1.3.3.3.2 Sensores del tipo externo.

El SCORA-ER 14Pro posee microinterruptores (Figura 1.3.3.25), los cuales se utilizan para limitar el movimiento del robot (Final de Carrera) y para la posición Inicio (Home) del robot (Interruptores Home). El interruptor es parte de un circuito eléctrico del brazo del robot y es independiente del controlador.

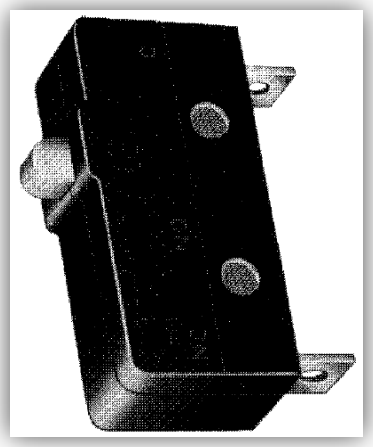


Figura 1.3.3.25 Microinterruptor.

Final de Carrera.

Los interruptores de final de carrera previenen que las articulaciones se muevan más allá de sus límites funcionales. Cuando el software de control del robot no puede detener el eje, el final de carrera sirve para detener su movimiento en los extremos del rango de trabajo.

Los ejes del 1 y 2 disponen de dos interruptores de final de carrera, uno para cada límite del rango de movimiento. Estos interruptores están montados en un disco sujetos al bastidor del robot. En la Figura 1.3.3.26 se muestran los microinterruptores en el eje 2.

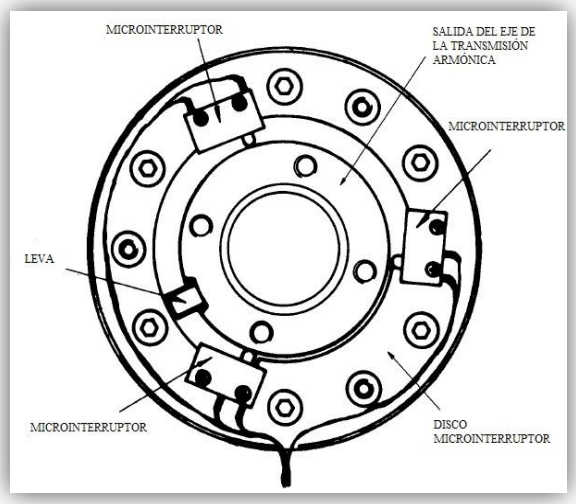


Figura 1.3.3.26 Microinterruptores en eje.

La salida del eje de la transmisión armónica tiene integrada una leva, a medida que la articulación se mueve, la leva activa o desactiva el botón del microinterruptor de final de carrera. Cuando se activa el final de carrera, se produce un error de control, resultando la activación COFF y un mensaje de protección de impacto. Se debe activar de nuevo CON y mover el brazo del robot manualmente (usando el teclado o la botonera de enseñanza) fuera de la condición de impacto.

Mientras no se alcance ningún interruptor de final de carrera el eje puede girar libremente en cualquier sentido.

Antes de llegar a pulsar el interruptor límite, todos los ejes están protegidos por medio de límites software.

El eje 3 tiene dos microinterruptores de final de carrera, uno en el límite superior y uno en el límite inferior del rango de dicho eje.

El eje 4 (giro) no tiene final de carrera, puede girar sin fin. Cuando un dispositivo de agarre esta en el eje 4, sus movimientos son controlados y limitados por medio de software (codificador) solamente.

Interruptores Home.

El SCORA-ER 14Pro utiliza un interruptor óptico para encontrar la posición de referencia de cada eje del brazo del robot.

Para los ejes 1 y 2 el microinterruptor de referencia se instala en el mismo disco que los microinterruptores de finales de carrera.

Para el eje 3 dos microinterruptores de referencia son colocados, uno en la parte superior y otro en la parte inferior del rango de dicho eje.

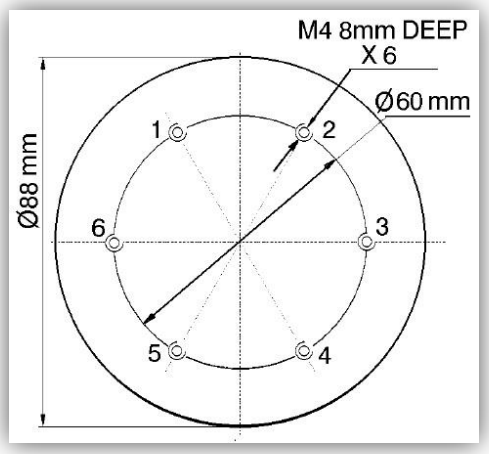
Para el eje 4 el microinterruptor de referencia se encuentra dentro de la caja de engrane gusano.

Durante el procedimiento de referencia (Home), el controlador mueve un eje a la vez. El eje se mueve hasta que el interruptor óptico detecta la posición de Home y este, entonces, envía una señal al controlador. Una vez que el detector de Home ha encontrado la posición, el eje se sigue moviendo, a pequeña velocidad, hasta que el codificador produce un impulso índice. En este punto, la posición Home se ha encontrado y se detiene el eje, posteriormente el proceso Home se iniciará en el siguiente eje.

Cuando todas las articulaciones están en home, el robot está en la posición de inicio. Este es el punto de referencia para el funcionamiento del robot. Cada vez que el sistema este encendido, el robot debe ser enviado a esta posición, por medio de una rutina de software (Home).

1.3.3.4 Equipo Periférico.

El SCORA-ER 14Pro puede ser equipado con un dispositivo de agarre unido a la brida en el extremo del brazo del robot, cuyo diseño se muestra en la Figura 1.3.3.27.



Los dispositivos de agarre que el fabricante recomienda son la utilización de una Pinza Neumática o Pinza Eléctrica Servo Controlada. Las cuales deben cumplir con el diseño de la brida para un correcto acoplamiento.

Figura 1.3.3.27 Superficie de montaje de la pinza.

1.4 Tipos de Controladores en el Laboratorio de Manufactura Avanzada.

Se tiene el registro de dos tipos de controladores diferentes, como sistema de decisión para la operación de los robots del laboratorio, este sistema de decisión es el cerebro del robot que tiene como objetivo controlar al sistema mecánico, mediante la información del sistema de percepción. Una descripción gráfica del sistema de decisión se muestra en la Figura 1.4.1.

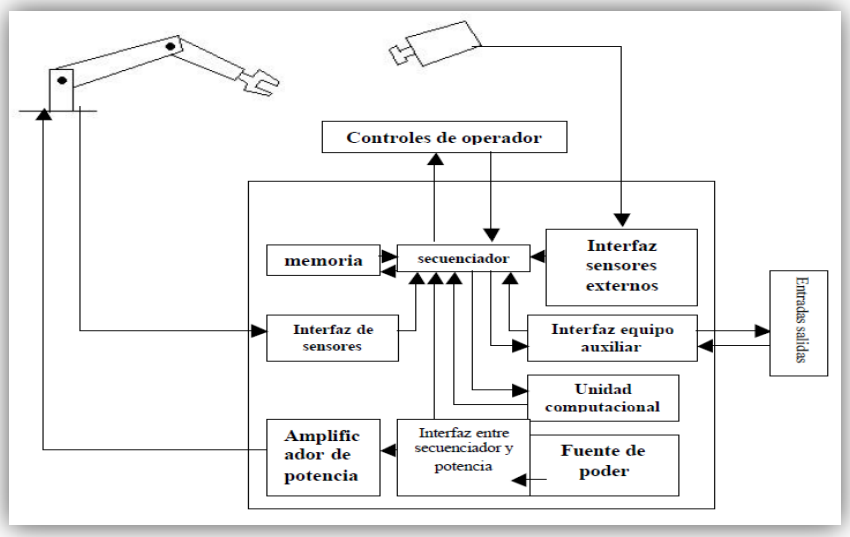


Figura 1.4.1 Sistemas constitutivos de un Controlador.

El panel de control del robot consta de:

- ✓ Una *memoria*, para guardar información que define posiciones (tales como ángulos y longitudes asociadas con las articulaciones) de donde el brazo se moverá, además de otra información relacionada a la propia secuencia del sistema (por ejemplo, el programa).
- ✓ Un *secuenciador*, que interpreta la información guardada en la memoria y que utiliza la información para interactuar con otros componentes del controlador.
- ✓ Una *unidad computacional*, que proporciona los cálculos necesarios para ayudar al secuenciador.
- ✓ Una *interfaz*, para obtener la información sensorial (tales como posición de cada articulación) hacia el secuenciador.
- ✓ Una *interfaz*, para transferir la información del secuenciador a la unidad de conversión de potencia, tal que los actuadores puedan eventualmente hacer que las articulaciones se muevan en la manera deseada.

- ✓ Una *interfaz para el equipo auxiliar*. El controlador del robot puede ser sincronizado con otras unidades externas o dispositivos de control (por ejemplo, motores y válvulas activadas eléctricamente) y/o determinar el estado de los sensores, tales como contactos límite.
- ✓ Alguna clase de unidad de control para el usuario para grabar posiciones, definir secuencias de operaciones y control del robot.

Los controladores son nombrados de acuerdo a la empresa que los fabrica, el nombre de esta empresa Eshed Robotec (1982), posteriormente paso a ser conocida con el nombre de Intelitek (2001). Sin embargo, Eshed Robotec fue el fabricante de gran parte del equipo de robots y controladores con los que cuenta el Laboratorio de Manufactura Avanzada, por lo que el desarrollo de este trabajo se basa principalmente en los manuales de operación con los que la empresa Eshed Robotec contaba en ese entonces para el uso y operación de sus equipos.

1.4.1 Controlador-A.

Este tema se detalla las especificaciones y funciones del Controlador-A, que controla el sistema robótico SCORBOT-ER Vplus y SCORBOT-ER VII en el Laboratorio de Manufactura Avanzada.

Figura 1.4.2 Controlador-A.

En la Tabla 1.4.1 se detallan las especificaciones del brazo robot; la elaboración de esta tabla se realiza haciendo una abstracción de la información del manual de usuario, acerca

de los requerimientos más importantes que sirven como preámbulo en la elaboración de este trabajo.

Tabla 1.4.1 Especificaciones del Controlador-A		
Sección	**Especificaciones**	**Notas**
Tipo de control	Autónomo Tiempo real Multitarea PID (proporcional, integral, diferencial) PWM (modulación de ancho de pulso)	PC o terminal necesario para comunicarse con el controlador
Número de ejes servo	Normal : 8 Máximo : 11	
Número de Grupos	Los 11 ejes pueden ser divididos en 3 grupos: Grupo A (brazo) Grupo B Grupo C (ejes independientes)	Cada grupo tiene control independiente. Interpolación de ejes en los grupos A y B
Tarjetas driver de ejes	PWM (modulación de ancho de pulso) 20 KHz.	
Control de movimiento	PTP (punto a punto), CP (trayectoria continua), Joint (ejes), Lineal, Circular, Predefinido, Coordenadas de la Herramienta.	Ciclo de control de 10 ms. Control de aceleración /desaceleración por medio de software.
Control de trayectoria	Paraboloide Trapezoide En lazo abierto (no para usuario)	
Control de velocidad	Control de velocidad en el trayecto.	Velocidad programable en tanto por ciento
Parámetros de control	Servo Control Velocidad, perfil de velocidad, amortiguamiento Error de posición Operación de la pinza Protección térmica, de impacto y de límite Homing Interface de codificadores Cálculos cartesianos	
Alimentación	100/110/220 VCC, 50/60 Hz, 500W max.	±5%
Fuentes de Alimentación	Motores: +24 VCC, 18 A Usuario: +12 VCC, 2 A	
Peso	19 Kg.	
Dimensiones	490 mm de largo 445 mm de ancho 150 mm de alto	
Temperatura ambiente de operación	2° – 40° C	
CPU	Motorola 68010	

EPROM	348 Kb	
RAM	Sistema: 64 Kb Usuario: 128 Kb	
Comunicación	Puerto serial RS232	
Entradas	16 entradas (con LED)	
Salidas	16 salidas (con LED)	24 VCC max.
Lenguaje de programación	ACL: Lenguaje de Control Avanzado	Usando PC
Posiciones Gravadas	Absolutas Relativas Cartesianas De articulación	Por medio de ACL, SCORBASE, botonera de enseñanza
Nº de líneas de programa	12800 líneas y 6375 posiciones (o cualquier combinación)	
Nº de programas en la RAM de usuario	Depende de la longitud de los programas	
Multitarea	Ejecuta simultáneamente hasta 20 programas independientes.	
Sistema de posicionamiento	Codificadores ópticos incrementales	
Sistemas de coordenadas	Coordenadas XYZ Coordenadas de ejes	
Indicadores LED	Rojo Alimentación principal entradas / salidas Alimentación servo Emergencia	En el panel frontal
	Verde Alimentación a ejes	En el panel dorsal
Características de seguridad	Interruptor de emergencia Interruptor de potencia en motores	En el panel frontal
	Limite de corriente ajustable protección de cortocircuito	En todos los ejes
	Protección de software térmica, de impacto y de límites	
Conectores	Entradas / salidas Alimentación para usuario	Terminales en el panel frontal
	Impulsores de eje Pinza Canal RS232 Botonera de enseñanza Robot Canales RS232 auxiliares (opcionales)	En el panel dorsal: Conectores D9 Conector D9 Conector D25 Conector D25 Conector D50 Conector D37
Botonera de enseñanza	30 teclas de función múltiple 2 líneas de pantalla LCD; 16 caracteres por línea Características de control total	

Funciones del Controlador.

El panel frontal del controlador contiene los interruptores, LED y terminales de conexión para el operador. Consultar la Figura 1.4.3.

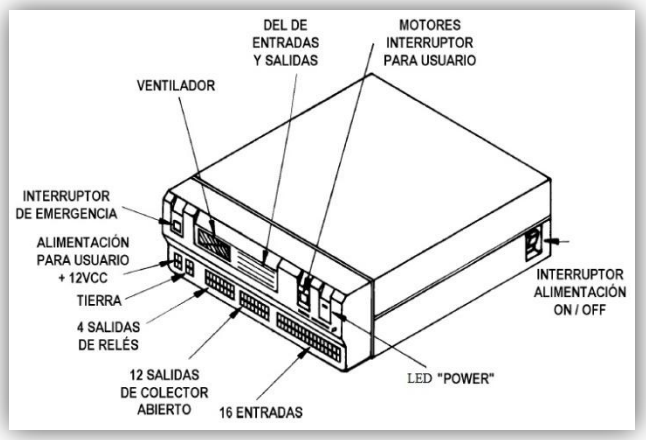


Figura 1.4.3 Panel frontal del Controlador-A.

Interruptor de Alimentación On/Off y LED.

El interruptor de alimentación del controlador, situado a su costado, conecta y desconecta el controlador a la fuente de corriente alternada.

El LED amarillo POWER en el panel frontal se enciende cuando el interruptor es activado. Indica que la alimentación llega al controlador.

Interruptor de Alimentación de Motores y Usuario, y LED.

Este interruptor conecta y desconecta voltaje de corriente continua a los motores y a la fuente del usuario. Un LED verde insertado en el interruptor se enciende cuando está conectado.

El interruptor de los motores se cierra en las siguientes circunstancias:

- Para desconectar los motores, la fuente del usuario y las entradas sin apagar el controlador.
- Para prevenir movimientos posibles de los ejes.

Cuando el interruptor de los motores es apagado, los motores y todos los ejes del robot son incapaces de moverse. Además desconecta la fuente de alimentación del usuario, convirtiendo las entradas y salidas de colector abierto en no operativas.

Interruptor de Emergencia y Lámpara.

Este interruptor detiene todas las operaciones del controlador. Una lámpara roja insertada en él se enciende cuando el interruptor está conectado. Cuando se aprieta el interruptor, sucede lo siguiente:

- La luz roja de emergencia se enciende.
- Abortan todos los programas que están siendo ejecutados.
- La alimentación de los motores se desconecta; cesan todos los movimientos; los LED verdes de los motores se apagan. Todos los LED verdes del panel dorsal se apagan.
- La alimentación del usuario se desconecta.
- Las entradas y salidas se desconectan.

Cuando el interruptor es presionado nuevamente, sucede lo siguiente:

- La lámpara roja de emergencia se apaga.
- El LED verde de los motores se enciende nuevamente.
- Los LED verdes del panel dorsal se encienden nuevamente.
- El CPU del controlador es reiniciado y aparece lo siguiente en la pantalla:

```
- - - - RAM TEST COMPLETE.
- - - - ROM TEST COMPLETE.
SYSTEM READY!
>_
```

Luego de una emergencia, el robot debe ser llevado a Inicio antes de que se pueda continuar el trabajo.

1.4.2 Controlador-B.

Este tema se detalla las especificaciones y funciones del Controlador-B, que controla el sistema robótico SCORA-ER 14Pro en el Laboratorio de Manufactura Avanzada.

Figura 1.4.4 Controlador-B.

En la Tabla 1.4.2 se detallan las especificaciones del brazo robot; la elaboración de esta tabla se realiza haciendo una abstracción de la información del manual de usuario, acerca de los requerimientos más importantes que sirven como preámbulo en la elaboración de este trabajo.

Tabla 1.4.2 Especificaciones del Controlador-B		
Sección	**Especificaciones**	**Notas**
Tipo de control	Autónomo Tiempo real Multitarea PID (proporcional, integral, diferencial) PWM (modulación de ancho de pulso)	PC o terminal necesario para comunicarse con el controlador
Número de ejes	Máximo de 12 ejes	
Número de Grupos	Los 12 ejes se pueden dividir en 3 grupos: Grupo A (brazo) Grupo B Grupo C (ejes independientes)	Cada grupo tiene control independiente. Interpolación de ejes en los grupos A y B
Tarjetas driver de ejes	Driver PWM en puente – H 33 KHz, / A. estándar, 10 A. opcional 33-42 V. (dependiendo de la carga) con realimentación	
Control de movimiento	PTP (punto a punto), CP (trayectoria continua), Joint (ejes), Lineal, Circular, Predefinido, Coordenadas de la Herramienta.	Ciclo de control 10 ms. Aceleración /Desaceleración PID controlada por software.
Control de trayectoria	Paraboloide Sinusoidal En lazo abierto	
Control de velocidad	Control de velocidad en el trayecto.	Velocidad programable en tanto por ciento
Parámetros de control	Servo Control Velocidad, perfil de velocidad, amortiguamiento Error de posición Operación de la pinza Protección térmica, de impacto y de límite Homing Interface de codificadores Cálculos cartesianos	
Requerimientos de potencia	1.500 W máx.	
Fuentes de Alimentación	Servo 33-42 V, 22 A Digital: 5 V / 12 V / 24 V Usuario: 12 V, 2 A, 24 V, 1.5 A	
Peso	36 Kg.	

Dimensiones	479 mm de largo 490 mm de ancho 206 mm de alto	
Temperatura ambiente de operación	2° – 40° C	
CPU	Motorola 68020 FPU – MC68881	
EPROM	512 KB (expandible 50 MB)	
RAM/batería	512 MB (exp. 500 MB) Sistema: 150 MB Usuario: 350 MB	
Comunicación	2 puertos RS232, expandible a 10 Puerto paralelo para imprimir	
Entradas	16 entradas	
Salidas	16 salidas	
BUS interno	32 bits de datos, dirección 24 bit	
Lenguaje de programación	ACL: Lenguaje de Control Avanzado	Usando PC
Multitarea	Ejecuta simultáneamente hasta 40 programas independientes.	
Sistema de posicionamiento	Codificadores ópticos incrementales	
Sistemas de coordenadas	Coordenadas XYZ Coordenadas de ejes	
Indicadores LED	Rojo Alimentación principal entradas / salidas Alimentación servo Emergencia	En el panel frontal
	Verde Potencia en ejes	En el panel trasero
Características de seguridad	Interruptor de emergencia Interruptor de potencia en motores	En el panel frontal
	Limite de corriente ajustable protección de cortocircuito	En el panel trasero
	Protección de software térmica, de impacto y de límites. Límite de Hardware “Watchdog”.	
Expansión	Conector para servos remotos 8 puertos RS232 BUS para periféricos inteligentes (futuro)	
Botonera de enseñanza	30 teclas de función múltiple 2 líneas de pantalla LCD; 16 caracteres por línea Características de control total	

Funciones del Controlador.

El panel frontal del controlador contiene los interruptores, LED y terminales de conexión para el operador. Consultar la Figura 1.4.5.

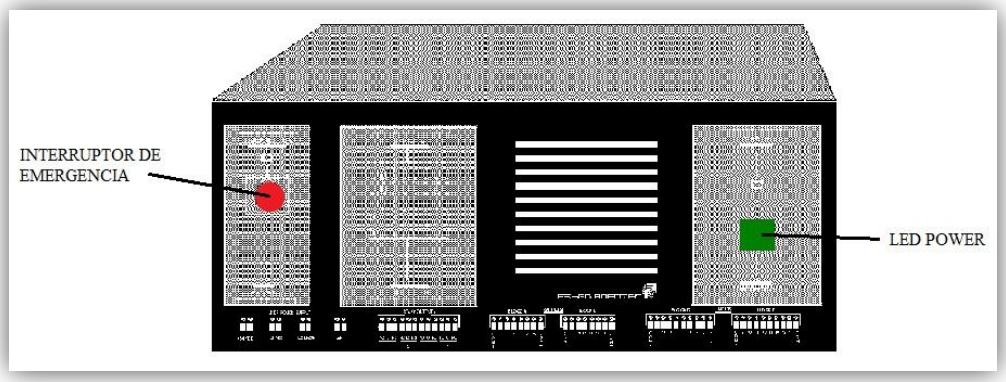


Figura 1.4.5 Panel frontal del Controlador-B.

Interruptor de Alimentación On/Off y LED.

El interruptor de alimentación del controlador, situado en el panel trasero, conecta y desconecta el controlador a la fuente de corriente alternada (Figura 1.4.6).

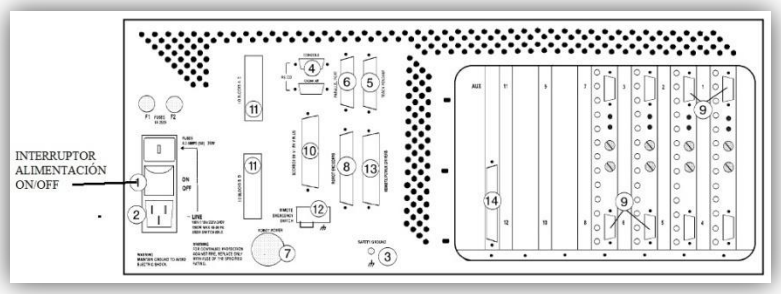


Figura 1.4.6 Panel trasero del Controlador-B.

El LED amarillo POWER en el panel frontal se enciende cuando el interruptor es activado. Indica que la alimentación llega al controlador.

Interruptor de Alimentación de Motores y Usuario, y LED.

Este interruptor conecta y desconecta voltaje de corriente continua a los motores y a la fuente del usuario. Un LED verde insertado en el interruptor se enciende cuando está conectado.

El interruptor de los motores se cierra en las siguientes circunstancias:

- Para desconectar los motores, la fuente del usuario y las entradas sin apagar el controlador.
- Para prevenir movimientos posibles de los ejes.

Cuando el interruptor de los motores es apagado, los motores y todos los ejes del robot son incapaces de moverse. Además desconecta la fuente de alimentación del usuario, convirtiendo las entradas y salidas de colector abierto en no operativas.

Interruptor de Emergencia y Lámpara.

Este interruptor detiene todas las operaciones del controlador. Una lámpara roja insertada en él se enciende cuando el interruptor está conectado. Cuando se aprieta el interruptor, sucede lo siguiente:

- La luz roja de emergencia se enciende.
- Abortan todos los programas que están siendo ejecutados.
- La alimentación de los motores se desconecta; cesan todos los movimientos; los LED verdes de los motores se apagan. Todos los LED verdes del panel dorsal se apagan.
- La alimentación del usuario se desconecta.
- Las entradas y salidas se desconectan.

Cuando el interruptor es presionado nuevamente, sucede lo siguiente:

- La lámpara roja de emergencia se apaga.
- El LED verde de los motores se enciende nuevamente.
- Los LED verdes del panel dorsal se encienden nuevamente.
- El CPU del controlador es reiniciado y aparece lo siguiente en la pantalla:

```
- - - - RAM TEST COMPLETE.
- - - - ROM TEST COMPLETE.
SYSTEM READY!
>_
```

Luego de una emergencia, el robot debe ser llevado a Inicio antes de que se pueda continuar el trabajo.

1.5 Métodos de Operación.

Los robots SCORBOT-ER Vplus, SCORBOT-ER VII y SCORA-ER 14Pro pueden ser programados y operados de varias maneras. Sin embargo en este tema se presenta el software del robot con el cual se hace la comunicación con el controlador, recordando que en el Laboratorio de Manufactura Avanzada se cuanta con dos configuraciones distintas de controladores (Controlador-A y Controlador-B), para la manipulación y ejecución de tareas en los brazos robóticos.

Para mayor referencia acerca de la operación del software, instalación de equipo y elementos periféricos consultar los manuales de operación, así como otros manuales suministrados con el sistema.

Software.

ACL

El ACL (Lenguaje Moderno de Control, o Advanced Control Language) es un lenguaje y entorno de programación robótica de tareas múltiples, desarrollado por Eshed Robotec (1982) Ltd.

El ACL es programado y está almacenado en memorias EPROM dentro de los Controladores (-A y -B), se puede acceder a él desde una PC por medio de un canal de comunicación RS232.

Es de gran importancia mencionar que la PC debe contar con comunicación RS232, ya que este trabajo de investigación se basa a partir de esta característica.

Las cualidades del ACL incluyen:

- Control directo de las articulaciones del robot por parte del usuario.
- Programación del sistema robótico por parte del usuario.
- Control de datos de entrada y salida.
- Ejecución de programas simultáneos, sincronizados e interactivos.
- Administración simple de archivos.

El ACL es descrito más a fondo en la *Guía de Referencia de ACL,* donde se explica la utilización de los comandos de comunicación y operación.

ATS

El ATS (Software Moderno de Terminal, o Advanced Terminal Software), es el entorno de interface del usuario al ACL del controlador. El ATS opera en cualquier PC que cuente con puerto de comunicación RS232. El software es un emulador de terminal que permite el acceso al ACL desde PC.

Las cualidades de ATS incluyen:

- Configuración abreviada del controlador.
- Definición de dispositivos periféricos.
- Uso de teclas para insertar órdenes.
- Editor de programa.
- Administración de respaldos.
- Administración de impresiones.

El ATS es descrito más a fondo en la *Guía de Referencia de ATS*, para su instalación y operación.

La Figura 1.5.1 muestra los componentes del sistema robótico:

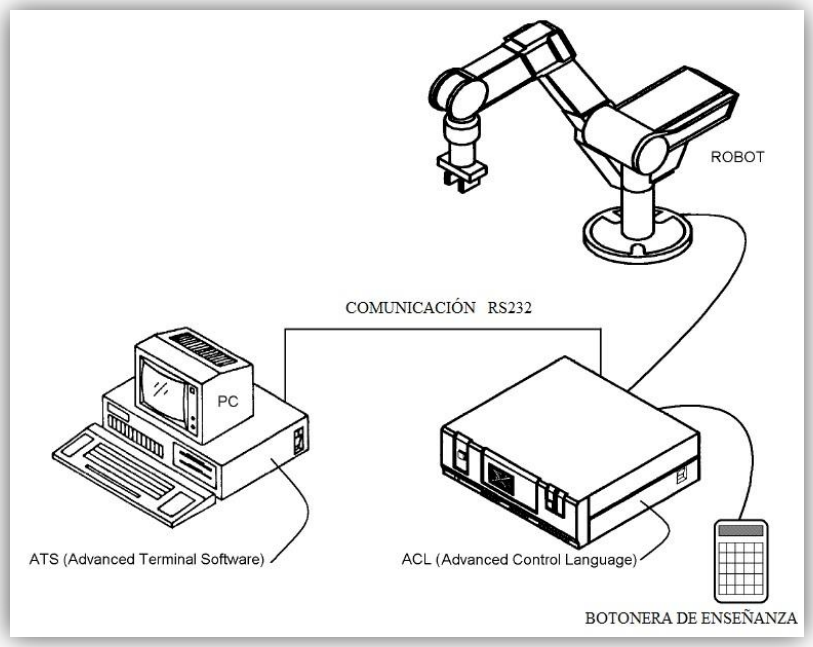


Figura 1.5.1 Sistema Robótico.

Un gestor de archivos integrado al ATS permite guardar y restaurar programas, posiciones, variables y parámetros de la memoria RAM del controlador asignada para el usuario.

ATS es un software terminal totalmente ASCII, funcionando a: 9.600 baudios, 8 bits de datos, sin paridad.

La Figura 1.5.2 muestra la vista de la ventana de trabajo del software ATS vista desde la pantalla de una PC. Lo que indica que ya se puede hacer comunicación directamente con el controlador.

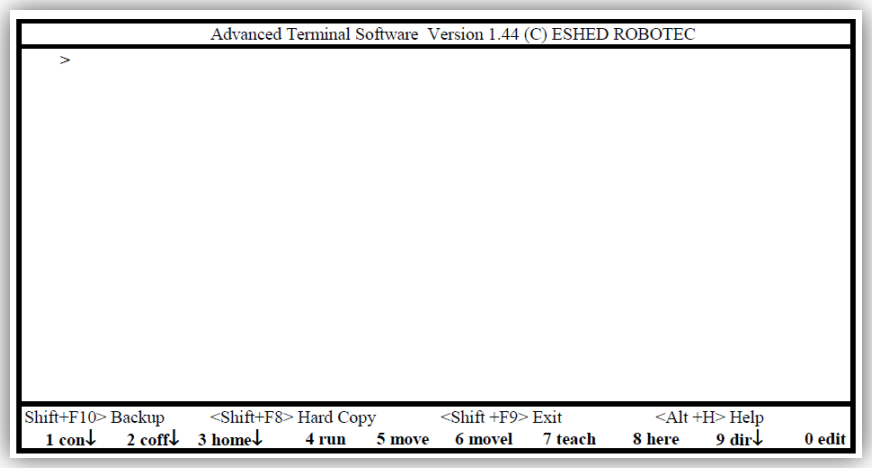


Figura 1.5.2 Pantalla principal del software ATS.

1.6 Operación de los Robots.

En este tema se presentan las órdenes básicas para operar los robots usando el software ATS.

Llevar el Robot y los Ejes Periféricos a Inicio (HOME).

La locación de los ejes del robot es controlada por los codificadores que miden la cantidad de movimiento con relación a la posición de referencia inicial, (Home). Esta posición de referencia debe ser idéntica cada vez que el robot sea usado.

Por lo tanto, cada vez que se activa el sistema se debe realizar la rutina de Ir a Inicio, que está programada internamente en el controlador.

Durante el proceso de ir a Inicio, las articulaciones del robot se mueven y buscan sus posiciones de Inicio (Home), una por una.

Los siguientes pasos harán que el robot vaya a su posición Inicial (Home).

- Activar Mayúsculas en el teclado de la computadora.
- Pulsar la teclas **<Alt+1>** y posteriormente escribir "**HOME**" o pulsar **<F3>**.

En la ventana de la Figura 1.5.2 aparecerá lo siguiente:

home <Enter>
WAIT!! HOMING...

Cuando se haya terminado la secuencia de búsqueda Home aparecerá en la venta del software

HOMING COMPLETE

Indicando que el Robot a encontrado su posición Inicial (Home) y se puede hacer uso del robot.

☞ *Para mayor referencia acerca de la operación y control de los robots, de los controladores, del software ATS y del lenguaje de programación ACL, consultar las prácticas de laboratorio del Departamento de Materiales y Manufactura referentes a la asignatura Sistemas de Manufactura Flexible, así como los manuales de operación y guías de la empresa Eshed Robotec (1982) Ltd.*

Sistemas de Coordenadas.

Los robots SCORBOT-ER Vplus, SCORBOT-ER VII y SCORA-ER 14Pro puede ser operados y programados en dos sistemas diferentes de coordenadas: coordenadas de ejes (Joints) y coordenadas cartesianas (XYZ).

En este trabajo de investigación se hace uso de las coordenadas cartesianas (XYZ), que a continuación se describe su función y localización de cada robot.

Coordenadas Cartesianas (XYZ).

El sistema de las coordenadas cartesianas, o XYZ, es un sistema geométrico usado para especificar la posición del PCH (Punto Central de la Herramienta) del robot por medio de la definición de la distancia, en unidades lineales, de su punto de origen (el centro de la base) a lo largo de los tres ejes lineales, como se ve en la Figura 1.6.1.

La Figura 1.6.1 muestra el punto de origen de cada configuración de los robots del Laboratorio de Manufactura Avanzada.

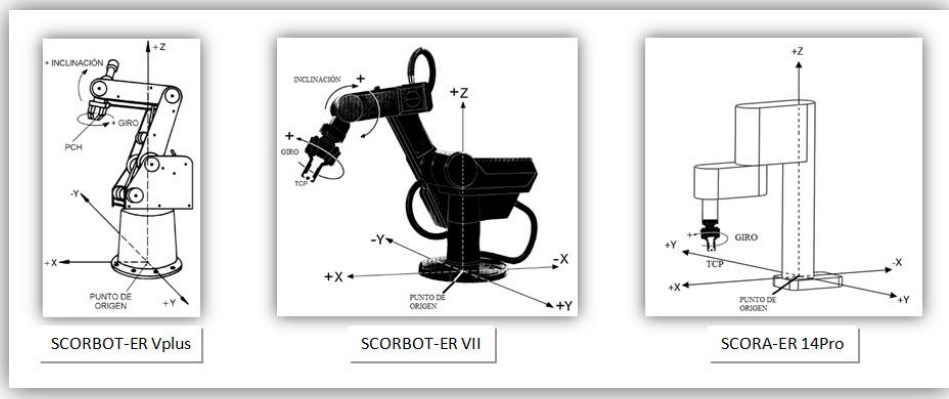


Figura 1.6.1 Coordenadas cartesianas.

En el caso de los robots SCORBOT-ER Vplus y SCORBOT-ER VII, para completar la definición de la posición, se especifica la inclinación y el giro en unidades de ángulo.

Para el robot SCORA-ER 14Pro, la definición de la posición se completa, solo con el giro en unidades de ángulo.

Cuando se ejecuta un movimiento del robot en modo XYZ, uno o todos los ejes se mueven para mover el PCH a lo largo de los ejes X, Y y Z.

Mover los Ejes.

Cuando el sistema de coordenadas está en modo XYZ, las órdenes de movimiento causan el movimiento lineal del PCH a lo largo de los ejes X, Y y Z, mientras se mantienen constantes los ángulos de inclinación y giro con respecto al punto de origen del robot.

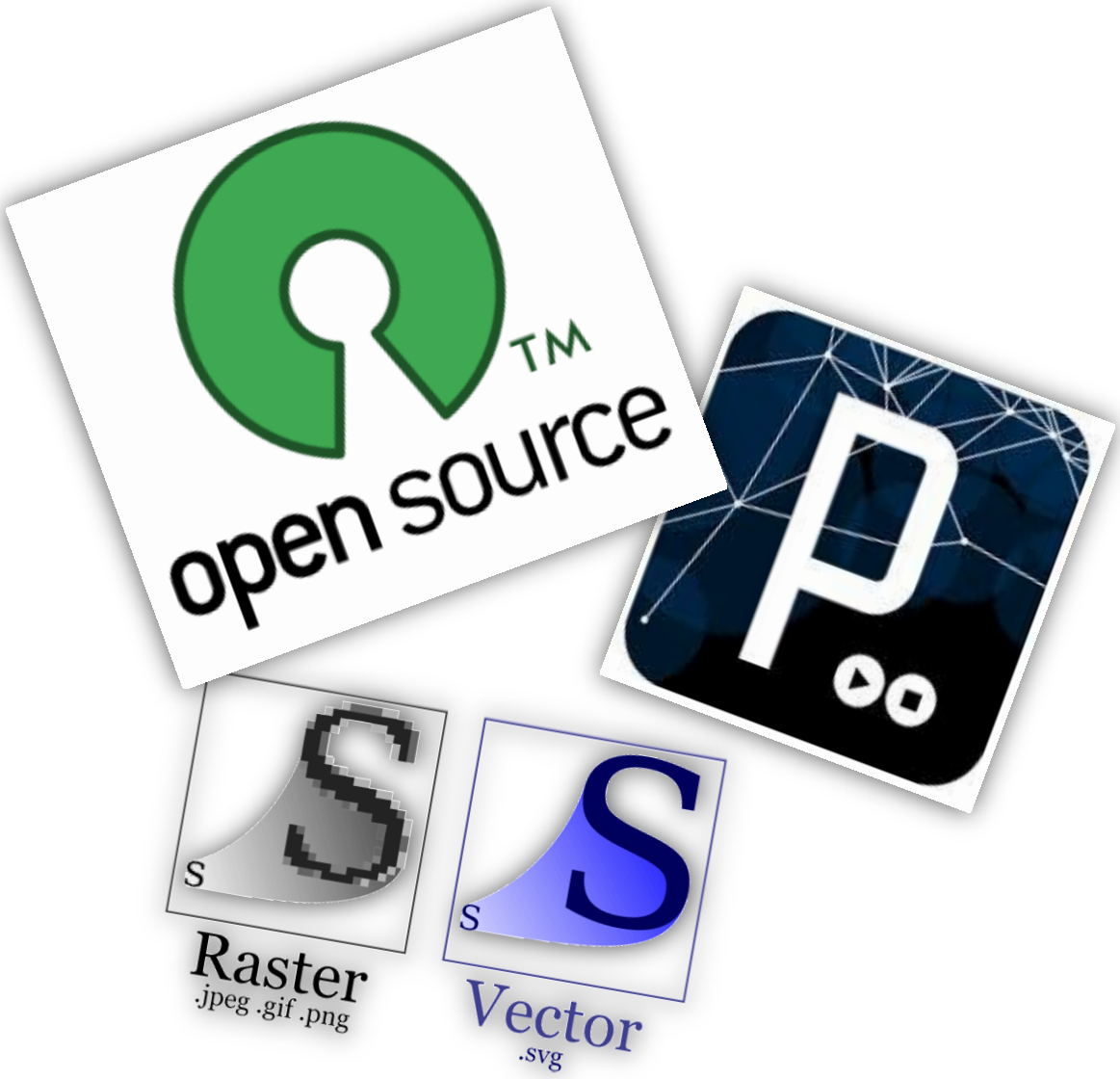
Cuando el sistema de coordenadas está en modo ejes, el robot responde a las órdenes de movimiento yendo de un punto definido a otro.

1.7 Necesidad y Problemática.

El Laboratorio de Manufactura Avanzada se ve en la necesidad de rehabilitar las aéreas de Robótica Didáctica y Robótica Industrial, además de seguir a la vanguardia con el desarrollo de la tecnología, recuperar y hacer que el entendimiento y operación de los equipos sean más fáciles de usar y con una aplicación técnico profesional.

Actualmente se ha realizado una profunda investigación, con la finalidad de tener mayor conocimiento acerca del funcionamiento del equipo en el Laboratorio de Manufactura Avanzada en las aéreas de Robótica Didáctica y Robótica Industrial como se ha mencionado en este trabajo, al hacer la investigación y al estar interactuando con los equipo de Robótica se presenta la siguiente problemática:

La robustez de los controladores implica, un mayor espacio en el área destinada para la operación y control del equipo, sin embargo reemplazar los controladores por tecnología nueva resultaría costoso poder solventar esa inversión a corto plazo, además tomando en cuenta que los controladores disponibles actualmente en el laboratorio pueden rehabilitarse considerando algunos puntos. Sin embargo una rehabilitación de controladores, así como de los brazos robóticos, se expone a sufrir daños, lo que genera una difícil tarea al tratar de conseguir refacciones y/o componentes que sean de gran importancia. Además la plataforma de uso para el sistema operativo quedo obsoleta y ha dado pie al rezago tecnológico, con lo que la utilización de esa plataforma se reserva al uso de algunas funciones básicas para el control de los robots. Comprender el funcionamiento del equipo requiere conocer y dominar por completo los manuales de usuario de cada configuración de robot y controlador, así como del software de operación y lenguaje de programación, para un correcto funcionamiento. Pero uno de los mayores problemas a los que se enfrenta es la poca información existente.



2. Desarrollo.

En este capítulo se propone una solución ante la necesidad del Laboratorio de Manufactura Avanzada, con el apoyo de las aplicaciones "open source" y la programación, haciendo de este trabajo una integración Mecatrónica de diferentes aéreas de la ingeniería.

2.1 Definición y Grabado de Posiciones.

Normalmente estos robots son utilizados en el Laboratorio de Manufactura Avanzada como robots pick and place en celdas de manufactura, la forma en cómo se definen y el tipo de coordenadas en que se graban las posiciones para ejecutar una tarea se explica con mayor detalle en los manuales de usuario, aquí solo se hace mención de la definición y grabado de posiciones como antecedente al desarrollo de este trabajo.

La definición de una posición reserva un espacio en la memoria del controlador, y le asigna un nombre. Al grabarse una posición, se escriben los valores de sus coordenadas en el espacio designado en la memoria del controlador.

Existen dos tipos de nombres de posiciones:

- Nombres numéricos (tales como 3, 22, 101) de hasta cinco cifras. Posiciones con este tipo de nombres no deben ser definidas antes de ser grabadas por medio de la botonera de enseñanza; la orden de grabar se define automáticamente y graba las posiciones con nombres numéricos.
- Nombres alfanuméricos (tales como P, POS10, A2). El nombre puede contener hasta cinco caracteres y debe comenzar con una letra. Estas posiciones no pueden ser accedidas desde la botonera de enseñanza.

Las posiciones pueden pertenecer a un vector; o sea, un arreglo de posiciones en el que se identifica un nombre específico y un índice; por ejemplo, PVEC[1] y PVEC[5] son posiciones en un vector llamado PVEC.

Las posiciones se graban predeterminadamente por el controlador como *coordenadas de ejes* (Joints), estas coordenadas especifican la localización de cada eje, según la cuenta de su codificador. Cuando el eje se mueve, el codificador óptico genera una serie de señales eléctricas altas y bajas alternadas. El número de señales es proporcional a la cantidad de movimiento del eje; el controlador las cuenta y determina qué distancia recorrió el eje. Similarmente, un movimiento o una posición del robot pueden ser definidos como un número específico de cuentas del codificador para cada eje, con relación a la posición de Inicio, o a otra coordenada.

El desarrollo de este trabajo se basa principalmente en definir posiciones con **Nombres Alfanuméricos**; normalmente estos robots operan de la siguiente forma, después de haber definido una posición, se hace mover el robot dentro de su área de trabajo, y se graban las coordenadas para una posición deseada. Así sucesivamente se realiza esta secuencia, definiendo y grabando posiciones en una serie de movimientos que se requieran para ejecutar una tarea.

Sin embargo como se menciono en el capitulo anterior las posiciones pueden grabarse también en *coordenadas cartesianas*, según las definidas por el usuario; grabar en *coordenadas cartesianas* no graba las coordenadas actuales del robot, si no que la orden debe ser ejecuta desde el software ATS con el comando TEACH de ACL.

Luego de haber grabado una posición, el PCH del robot puede fácilmente ir a dicha posición. Según el sistema de coordenadas activo, el movimiento del robot irá de un punto a otro dependiendo del número de posiciones grabadas.

2.2 Programas *.CBU.

Ejecutar una tarea pick and place con los robots de manera automática requiere haber definido y grabado cada posición a la cual se desea llevar el PCH, posteriormente se debe escribir un programa con la secuencia de posiciones a las que el PCH debe recorrer y así ejecutar una tarea programada.

El controlador guarda los programas, posiciones y variables, pero el software ATS permite hacer un respaldo de la información. El archivo que se genera puede ser guardado con un nombre de hasta 8 caracteres, la extensión *.CBU se agrega automáticamente, este procedimiento graba *todos* los programas, posiciones y variables que estaban actualmente en la memoria RAM del controlador al archivo.

Hasta este momento se conoce el comportamiento y funcionamiento de los robots de manera general, así también, se sabe que los programas que se generar contienen la información de una trayectoria, como se menciono anteriormente estos archivo se guardan en un formato *.CBU el cual solo reconoce el controlador, sin embargo los archivos de programas, pueden ser cargados al controlador según sea necesario, de esta forma, a manera de ingeniería inversa, sí se genera un archivo en formato *.CBU con los puntos en coordenadas a los cuales se deseará llevar el PCH, se desarrollaría una aplicación más práctica para comprender el funcionamiento y alcances, que estos robots didácticos podrían fomentar. ATS posee un Administrador de Respaldo para dicho propósito.

2.3 Opciones de solución.

La rehabilitación del equipo de Robótica Didáctica y Robótica Industrial, considera varios aspectos importantes antes de pasar al desarrollo de una aplicación práctica que ayude a comprender el funcionamiento de las diferentes configuraciones de los robots en el Laboratorio de Manufactura Avanzada; pero teniendo en cuenta que el conocimiento previo que se debe entender para manejar y operar el equipo de robótica es bastante, ya que se debe dominar la información contenida en los manuales de usuario de cada robot, así como las guías de operación del software ATS y del lenguaje de programación. Ante esta perspectiva se propone como complemento para facilitar la operación del equipo de robótica, generar un archivo que ya contenga una serie de puntos y posiciones grabadas, solo para ser ejecutadas por el controlador, el cual mandara las ordenes a los actuadores y hará que el robot siga la trayectoria contenida en el archivo.

2.4 Búsqueda de Información.

El fabricante que en la actualidad comercia este tipo de equipo en robótica didáctica (Intelitek), a realizado algunas mejoras con la implementación de equipo de control más sofisticado y de mejor calidad tecnológica, brindando mayor compatibilidad con las computadoras de nueva generación, pero como ya se menciono, solventar un gasto económico para reemplazar el equipo no brinda una solución muy favorable a corto plazo; sin embargo el fabricante no proporciona información detallada acerca de su software para poder controlar desde un archivo el robot. Contactar con el fabricante y plantear la posibilidad de realizar esta propuesta, hizo que se realizara una ardua investigación, ya que el fabricante contemplaba la siguiente solución, *"no es posible generar el archivo desde una gama de software más avanzada, por ejemplo Scorbase y Robocell, dado que los programas están diseñados para definir posiciones y grabarlas para que sean ejecutadas en un programa editado por los usuarios, por lo que el fabricante propuso que existía la posibilidad de programar, de alguna manera, una interfaz que genere este tipo de archivos con extensión *.CBU y que brindará la posibilidad de controlar el robot desde un archivo."*

Posteriormente se busco si algún programa de diseño asistido por computadora (CAD) podría brindar la solución de generar un archivo a partir de alguna trayectoria o modelo de una pieza en 2D o 3D, en la búsqueda se contemplaron varios programas de diseño entre los que destacan SolidWorks, NX (Unigraphics), CATIA y AutoCAD, además de ser grandes competidores a nivel mundial, algunos de estos paquetes de software en los Laboratorios del Departamento de Ingeniería de Materiales y Manufactura como son el

Laboratorio de Ingeniería Mecánica Asistida por Computadora y el Laboratorio de Manufactura Avanzada en el Área del Centro de Manufactura Integrada por Computadora (ver Figura 1.2.1), se brinda la posibilidad de utilizar estos programas de diseño como una herramienta a la propuesta de solución.

Algunas de las ventajas que estos programas de diseño brindan, es la amplia gama de formatos que manejan para exportar archivos, así como el fácil uso del software para generar trazos o modelar piezas u objetos en 2D o 3D; no obstante, su mayor desventaja es la licencia para su utilización en la propuesta generada para este trabajo, ya que ninguno de ellos permite generar un archivo con características y extensión *.CBU para ser leído por el controlador del robot, algunos paquetes requieren de otras aplicaciones que los proveedores ofrecen como complemento a ciertos servicios, un ejemplo de ello, el paquete de software AutoCAD facilita la exportación de coordenadas en 2D, de lo que se está diseñando, a un archivo de texto; en este caso su inconveniente es la adquisición del software y de la aplicación, para así poder exportar las coordenadas, lo que hace que la propuesta tenga un impacto económico. Ante esta perspectiva se toma la decisión de crear una interfaz gráfica de programación sencilla y en un software con características "open source", que genere trayectorias, y que a partir de esas trayectorias se pueda lograr obtener la información en coordenadas cartesianas, escritas en un archivo que pueda ser leído por el controlador del robot, todo esto a través de imágenes; las imágenes pueden contener la información de algún modelo en 2D que podría ser diseñado en algún paquete de software o simplemente una fotografía o imagen.

En resumen, el formato de imágenes o archivos de entrada a la interfaz deben cumplir con algunos puntos que más adelante se exponen, y como salida de la interfaz se obtiene un archivo con extensión *.CBU el cual además de cumplir con las características del controlador para ser leído, debe contener la información de las coordenadas cartesianas generadas por la interfaz. Esta interfaz gráfica es creada como complemento al software ATS, la interfaz ayuda a crear un archivo y el software ATS hace que la información contenida en el archivo se ejecute haciendo mover el PCH del robot; la información contenida en el archivo corresponde a la composición de la imagen en una serie de puntos que ya se encuentran definidos y grabados, de esta manera dentro del archivo se encuentra escrito un programa, el cual se ejecuta por el software ATS y el controlador envía las instrucciones correspondientes para que el robot se mueva siguiendo la trayectoria descrita en la imagen que ingresa por medio de la interfaz.

La finalidad de esta propuesta de solución contempla poder conjuntar aplicaciones mecatrónicas que tienen que ver con la integración entre software y hardware, y así tener un mayor uso en las aplicaciones, que para este caso rehabilitará el equipo de robótica en las áreas de Robótica Didáctica y Robótica Industrial del Laboratorio de Manufactura Avanzada.

2.5 Diseño de la Interfaz Gráfica de Usuario.

La interfaz gráfica de usuario, es un programa informático que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador. Una interfaz debe lograr una efectiva comunicación entre el programa y el usuario a través de sus elementos gráficos y su distribución, así mismo facilitar la realización de tareas y adecuarse a las necesidades del usuario para favorecer el aprendizaje.

El diseño de la interfaz nos ayudará a crear una comunicación entre el hombre (esto es, el usuario) y la computadora. El diseño identifica los objetos y acciones para generar un archivo *.CBU, y crear entonces un formato de pantalla que formará la base del prototipo de interfaz de usuario.

El diseño de la interfaz gráfica de usuario comienza con la identificación de los requisitos del usuario, de la tarea y del entorno. Para ésta propuesta de solución se plantea generar un archivo que contenga la información de coordenadas a través del procesamiento de imágenes.

Una vez identificadas las tareas se crea y se analiza el conjunto de objetos y de acciones para definir la interfaz. Esto es lo que forma la base para la creación del formato de la pantalla que representa el diseño gráfico y colocación de iconos, la definición de texto descriptivo en pantalla y la especificación de los elementos primarios y secundarios del menú. Esta herramienta se utiliza para generar prototipos y por ultimo implementar el modelo de diseño y evaluar la calidad del resultado.

2.5.1 Software de Código Abierto.

El software de código abierto por sus siglas OSS (en inglés Open Source Software), ofrece un software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de software compatible con la Open Source Definition o forman parte del dominio público. Esto permite a los usuarios utilizar, cambiar, mejorar el software y redistribuirlo, ya sea en su forma modificada o en su forma original. Frecuentemente se desarrolla de manera colaborativa y los resultados se publican en internet.

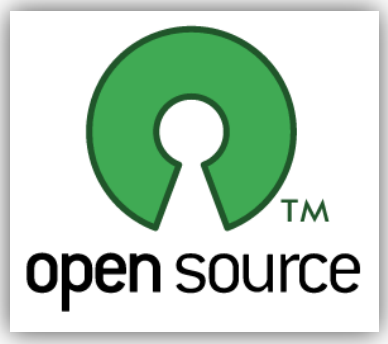


Figura 2.5.1 Logo de la iniciativa Open Source.

El movimiento del software libre surgió en 1983, y en 1998, un grupo de individuos defendieron la idea de cambiar la expresión free software (software libre) por open source software (software de código abierto). Los desarrolladores de software pueden optar por publicar su trabajo bajo una licencia de código abierto, de manera que cualquiera pueda beneficiarse de él. Por lo general, este software permite a las personas crear modificaciones del software, compatibilizarlo con otros sistemas operativos o arquitecturas de hardware, compartirlo con otras personas y comerciarlo.

La Open Source Definition introduce una filosofía en cuanto al código abierto, y además define los términos de uso, modificación y redistribución del software de código abierto. Las licencias de software otorgan a los usuarios derechos que de otro modo estarían reservados, por la ley de derechos de autor.

¿Por qué se propone usar Software de Código Abierto?

La propuesta de solución para el diseño de la interfaz, se apoya en que el software debe ser libre y de código abierto, permitiendo hacer uso de copia y redistribución, dotar de infraestructura tecnológica el Laboratorio de Manufactura Avanzada reduciendo la brecha digital con el mundo desarrollado. Además de ofrecer un software a muy bajo costo o incluso gratuito.

2.5.2 Processing.

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital, además de contar con una comunidad en línea. Fue iniciado por Ben Fry y Casey Reas desde el año 2001, processing promueve la alfabetización de software dentro de las artes visuales y la cultura visual, por medio de la tecnología. Inicialmente fue creado para servir como un software sketchbook (cuaderno de bocetos) y para enseñar los fundamentos de programación en computadoras dentro de un contexto visual, su producción evolucionó hasta convertirse en una herramienta de desarrollo para los profesionales. Hoy en día, estudiantes, artistas, diseñadores, investigadores y aficionados utilizan processing para el aprendizaje, la creación de prototipos y producción.

Se distribuye bajo la licencia GNU GPL. La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License, es una licencia ampliamente usada en el mundo del software y garantiza a los usuarios finales (personas,

organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.

Su finalidad es proteger los derechos de los usuarios finales. De esta forma, las aplicaciones instaladas en sistemas operativos, no es necesario que estén licenciadas bajo GPL o que estén distribuidas con su código fuente disponible, ya que las licencias no dependen de la plataforma. Los usuarios o compañías que distribuyen sus trabajos bajo licencias GPL, pueden cobrar o distribuirlas gratuitamente. Esto distingue las licencias GPL de las licencias software que prohíben su distribución comercial.

Al estar basado su entorno de programación en Java, puede heredar todas sus funcionalidades, convirtiéndose en una herramienta poderosa a la hora de encarar proyectos complejos.

En la Tabla 2.5.1 se muestra la información general acerca del software de programación.

Tabla 2.5.1 Información general de Processing.

Desarrollador(es) processing.org	
Paradigma	Orientado a objetos
Apareció en	2001
Diseñado por	Ben Fry and Casey Reas
Última versión estable	2.0.3 (5 de septiembre 2013)
Tipo de dato	Fuerte, estático
Influido por	Design by Numbers, Java, OpenGL, PostScript, C
Sistema operativo	Multiplataforma
☞ Descargar gratis. ☞ Creación de programas interactivos en 2D y 3D. ☞ Integración OpenGL para 3D. ☞ Más de 100 librerías amplían el núcleo del software. ☞ Foros en internet y disponibilidad de libros.	

¿Por qué Processing?

- ✓ Ambiente de programación y API (interfaz de programación de aplicaciones).
- ✓ Lenguaje y entorno de programación de código abierto, basada en JAVA.
- ✓ Creado para facilitar el desarrollo de aplicaciones visuales animadas e interactivas.
- ✓ Facilidad de esbozar código para producir rápidamente resultados visuales.
- ✓ Ambiente de desarrollo (PDE - Processing Development Environment).

- ✓ Colección de funciones (comandos o métodos) que son el corazón de la interfaz o API.
- ✓ Conjunto de librerías que soportan funcionalidades más avanzadas como dibujos en OpenGL, lectura de archivos XML, lectura de archivos PDF, varios formatos de imágenes, etc.
- ✓ Se puede combinar programación con Java y sus librerías.
- ✓ Permite 3 formas de programar: básica, procedural/estructurada y orientada a objetos.
- ✓ Processing permite programar en lenguaje 'C' (programación estructurada, a base de funciones). Pero es conveniente aprovechar su enfoque orientado a objetos (ligeras modificaciones de las clases Java).
- ✓ Programas más complejos pueden desarrollarse aplicando orientación a objetos (clases de Java).
- ✓ Genera aplicaciones listas para ser ejecutadas en las tres principales plataformas: Mac OS, Linux y Windows.
- ✓ Las aplicaciones processing también pueden generarse para su ejecución en internet (como un applet de Java).
- ✓ Es posible desarrollar aplicaciones para dispositivos móviles (http://mobile.processing.org).
- ✓ Conexión con dispositivos y prototipos electrónicos: proyectos Arduino y Wiring (http://hardware.processing.org).

La programación en *Processing* y funcionamiento de las aplicaciones mediante Java, sigue una serie de pasos, la Figura 2.5.2 muestra una representación esquemática, en diagrama de bloques, de la comunicación existente entre cada modulo en el entorno de programación del lenguaje de Processing.

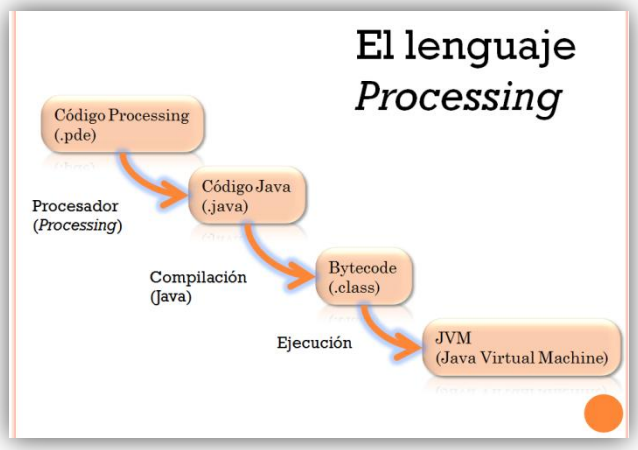


Figura 2.5.2 Diagrama de bloques lenguaje *Processing*.

En la Figura 2.5.3 se muestra la ventana principal del software de programación *Processing*, y se describen las partes principales que la componen.

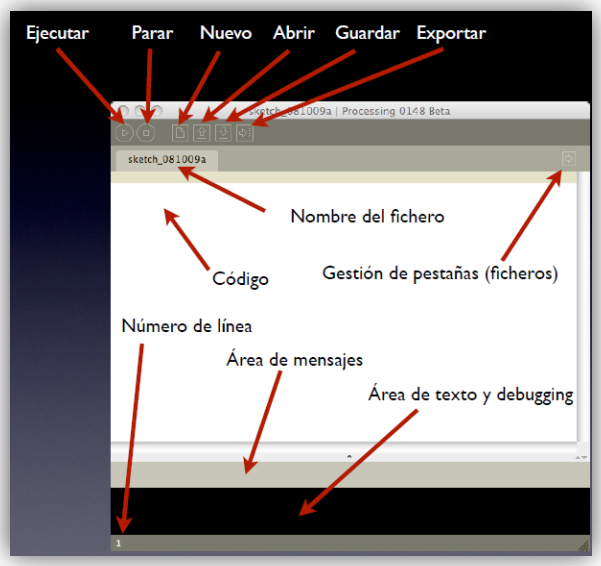


Figura 2.5.3 Ventana de programación *Processing*.

En conclusión, la propuesta de diseño de la interfaz contempla la utilización de *Processing* como software de programación para la interfaz grafica, porque los paquetes de librerías disponibles en la web manejan la integración de archivos como los siguientes:

- Importación de datos XML, SVG.
- Exportación PDF, DXF, etc.
- Video.
- Redes.
- Comunicación serial.

Y contribuciones externas como:

- Sonido: Ess, Sonia.
- Visión por computadora: JMyron, ReacTIVision, BlobDetection.
- …

Ya que processing maneja varios modos de programación, por ejemplo:

- ☞ Modo básico (dibujos estáticos).
- ☞ Modo continuo (animación).
- ☞ Modo Java (clases Java).

Se puede adaptar a la audiencia; la información que en los siguientes subtemas se exponen, hablarán de los archivos de entrada que la interfaz debe procesar para finalmente generar el archivo con extensión *.CBU.

2.6 Archivo de Entrada a la Interfaz.

Como archivo de entrada a la interfaz, se manejan imágenes vectorizadas ya que *Processing* permite la importación de datos en formato SVG. Aunque existen dos formas de representar gráficos en formato digital, como el formato raster o mapa de bits y el formato vectorial, se propone trabajar con el formato vectorial, a continuación se explica el porqué.

2.6.1 Formato Raster o Mapa de Bits.

La imagen raster, imagen digital o mapa de bits es un archivo o estructura representado generalmente en una rejilla rectangular de píxeles, o puntos de color en un monitor de un ordenador, papel u otro medio de salida.

Un mapa de bits o imagen raster se caracteriza por el ancho y el alto de la imagen en píxeles y el número de bits por píxel, el cual determina el número de colores que puede representar.

Ejemplo de mapa de bits.

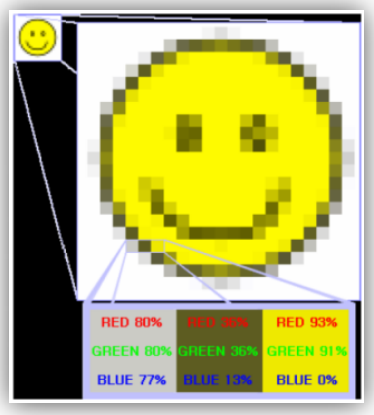


Figura 2.6.1 La cara sonriente en la esquina superior izquierda es un mapa de bits. Cuando se aumenta esta imagen aparece la gran cara sonriente a la derecha. Cada cuadrado representa un píxel. Aumentando el tamaño se observan tres píxeles, cuyos colores están construidos por valores de rojo, verde y azul.

Figura 2.6.2 Imagen ampliada mostrando el tamaño original en la esquina superior derecha.

El color de cada píxel está definido individualmente; imágenes en el espacio de colores RGB, por ejemplo, consisten de píxeles coloreados por tres bytes. Un byte para el rojo, otro para el verde y otro para el azul.

La calidad de una imagen raster está determinada por el número total de píxeles (resolución) y la cantidad de información en cada píxel (a menudo llamado profundidad de color). Los gráficos raster no pueden ser escalados a una resolución alta sin pérdida de calidad aparente. Esto es en contraste a gráficos vectoriales, el cual puede fácilmente escalar a la calidad del dispositivo en el cual está representado. Los gráficos raster son más prácticos para fotografías e imágenes, mientras que los gráficos vectoriales son más prácticos para el diseño grafico, sistemas de información geográfica o conjunto de tipografías.

Algunos de los formatos más comunes de gráficos en formato raster o mapa de bits son:

- BPM
- GIF
- PNG
- JPG
- TIFF

2.6.2 Formato Vectorial.

Los **gráficos vectoriales** (también conocidos como *modelados geométricos* o *gráficos orientados a objetos*) son los que se conforman con primitivas geométricas tales como puntos, líneas, curvas o polígonos, de igual forma, son gráficos que se construyen por ordenador basándose en ecuaciones matemáticas. Se utilizan como antónimas de aquellas imágenes que están configuradas sobre un conjunto de píxeles, que hemos visto en el punto anterior.

Todos los ordenadores actuales traducen los gráficos vectoriales a gráficos rasterizados para poderlos visualizar en la pantalla. La imagen rasterizados posee un valor determinado para cada píxel que la conforma, esta información se guarda en memoria ocupando un espacio específico.

Al contrario que un *bitmap,* una imagen vectorial puede ser escalada, rotada o deformada, sin que ello perjudique en su calidad. Normalmente, un conjunto de trazos se puede agrupar, formando objetos, y crear formas más complejas que permitan el uso de curvas de Bézier, degradados de color, etc.

Figura 2.6.3 Dos imágenes de una locomotora, la primera (izquierda) como imagen vectorial respecto de la segunda (derecha) de la que fue trazada.

Como se ha mencionado anteriormente. Las imágenes en los gráficos vectoriales no se construyen píxel a píxel, sino que se forman a partir de vectores, objetos formados por una serie de puntos y líneas rectas o curvas definidas matemáticamente.

Un ejemplo, una línea se define en un gráfico de mapa de bits mediante las propiedades de cada uno de los píxeles que la forman, mientras que en un gráfico vectorial se hace por la posición de sus puntos inicial y final y por una función que describe el camino entre ellos. Analógicamente, un círculo se define vectorialmente por la posición de su punto central (coordenadas x, y) y por su radio (r).

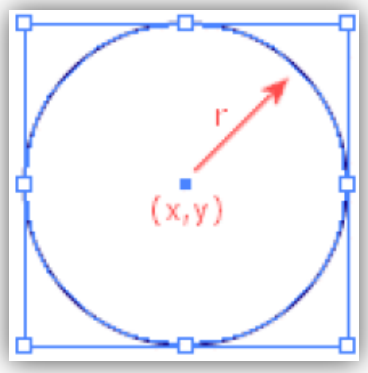


Figura 2.6.4 Vector Círculo.

Cada vector en un gráfico vectorial tiene una línea de contorno, con un color y un grosor determinados. Las imágenes vectoriales se almacenan como una lista que describe cada uno de sus vectores componentes, su posición y sus propiedades.

En cuanto a la resolución, los gráficos vectoriales son independientes de la resolución, ya que no dependen de una retícula de píxel dada. Por lo tanto, tienen la máxima resolución que permite el formato en que se almacena.

Ventajas y Desventajas.

Ventajas.

- Dependiendo del tipo de imagen, las imágenes vectoriales pueden requerir menor espacio en disco que un bitmap. Las imágenes formadas por colores planos o degradados sencillos son más factibles de ser vectorizadas. A menor información para crear la imagen, menor será el tamaño del archivo. Dos imágenes con dimensiones de presentación distintas pero con la misma información vectorial, ocuparán el mismo espacio en disco.
- No se pierde calidad al ser escalados, rotados o deformados. Ciertamente, se puede hacer zoom sobre una imagen vectorial de forma ilimitada. En el caso de las imágenes rasterizadas, llega un momento en el que el zoom revela que la imagen está compuesta por píxeles.
- Los parámetros de los objetos configurados por medio de vectores pueden ser guardados y modificados.
- Algunos formatos permiten animación. Está se realiza de forma sencilla mediante operaciones básicas como traslación o rotación y no requiere un gran acopio de datos, ya que lo que se hace es reubicar los nodos base de los vectores en nuevos puntos dentro de los ejes *x*, *y* y *z* en el caso de 3D.
- Es posible un control independiente del color, tanto del contorno como del relleno, admitiendo la aplicación de texturas, degradados, transparencias, etc.
- Se puede controlar con gran precisión la forma, orientación y ordenación de los elementos.
- Cualquier efecto que se aplique a los objetos puede rectificarse en cualquier momento, ya que el dibujo es siempre editable. Esto no ocurre en las imágenes de mapas de bit, en las que una vez pintado un elemento ya no es posible modificarlo.

Desventajas.

- Los gráficos vectoriales no son aptos para mostrar fotografías o imágenes complejas, aunque algunos formatos admiten una composición mixta (vector + imagen bitmap).
- Los datos que describen el gráfico vectorial deben ser procesados, es decir, el computador debe sumar todos los datos para formar la imagen final. Si hay demasiados datos se puede ralentizar la presentación de la imagen, incluso en imágenes pequeñas.
- Por más que se construya una imagen con gráficos vectoriales su visualización tanto en pantalla, como en la mayoría de sistemas de impresión, en última instancia tiene que hacer una traducción a sistema rasterizado.

Los gráficos vectoriales son ideales para dibujos simples y compuestos que necesitan tener formas independientes o que no necesitan tener un carácter de realismo fotográfico.

Formatos más comunes:

- ✓ POSTSCRIPT
- ✓ PDF (Es un formato derivado del formato PostScript)
- ✓ SVG (Este formato se explica en el tema 2.7 Gráficos Vectoriales Escalables).
- ✓ SWF

2.7 Gráficos Vectoriales Escalables (SVG).

En este subtema se explica el formato de imágenes SVG que el software *Processing* admite como formato de entrada para el desarrollo de la interfaz gráfica de usuario.

Scalable Vector Graphics (SVG) es un lenguaje de marcado en XML, que sirve para describir gráficos vectoriales de dos dimensione, tanto estáticos como animados. Las imágenes pueden contener hiperenlaces usando XLinks simple de salida. Es un estándar abierto creado por el consorcio del World Wide Web.

SVG permite tres tipos de objetos gráficos:

- ➢ Formas (p.e. trayectorias que consisten en líneas rectas y curvas, y áreas limitadas por ellas).
- ➢ Imágenes raster o digitales.
- ➢ Texto

Los objetos gráficos se pueden agrupar, transformar y componer en objetos previamente renderizados. El texto puede estar en cualquier namespace de XML, su uso conviene cuando se realiza búsqueda y accesibilidad de los gráficos de SVG. En la Figura 2.7.1 se muestra una comparación entre una imagen raster y una vectorial.

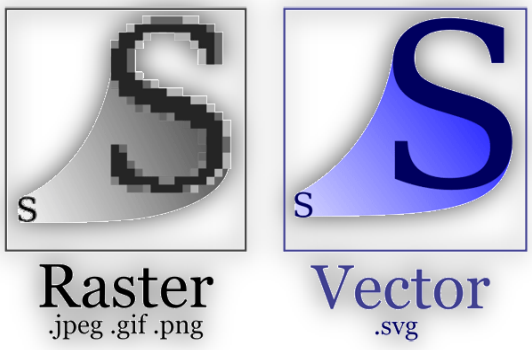


Figura 2.7.1 Está imagen muestra la diferencia entre imágenes raster y vectoriales con SVG. La imagen vectorial puede ser escalada indefinidamente sin pérdida en la calidad de imagen, mientras que el mapa de bits no.

2.7.1 Conceptos.

El significado de SVG es gráficos de vector escalable, una gramática de XML para los gráficos que permite el uso de estilos, utilizable dentro del namespace de XML.

Escalable.

Ser escalable significa aumentar o disminuir uniformemente. En términos de gráficos, escalable es que no son limitados a un solo tamaño fijo del píxel. En la Web, escalable es una terminología particular que puede ser una gran cantidad de archivos o una variedad amplia de usos. SVG, siendo una terminología de los gráficos para la Web, es escalable en ambos sentidos de la palabra.

Los gráficos de SVG son escalables a diversas resoluciones de exhibición, de modo que por ejemplo, el mismo gráfico de SVG se puede colocar en diversos tamaños en la misma página Web, y reutilizar en diversos tamaños en diversa paginas.

Vector.

Los gráficos por vector contienen objetos geométricos tales como líneas y curvas. Esto da mayor flexibilidad, comparada con los formatos de trama (tales como PNG y JPG) que tienen que almacenar la información para cada píxel del gráfico. Típicamente, los formatos de vector pueden también integrar imágenes de trama y pueden combinarse con la información del vector tal como trayectorias de truncamiento para producir una ilustración completa.

Puesto que actualmente todo está orientado a gráficos raster, SVG da control sobre el proceso de rasterización, y también proporciona filtros de efecto raster, de modo que el cambio en el formato del vector no signifique la pérdida de efectos populares tales como sombras suaves.

Gráficos.

La mayoría de las gramáticas existentes de XML representan cualquier información textual informaciones en bruto tales como información financiera. SVG proporciona una

descripción rica y estructurada de gráficos vectoriales y raster; puede ser usado independientemente, o como namespace de XML con otras características.

XML.

XML es una recomendación de W3C para el intercambio de información estructurada, ha llegado a ser extremadamente popular y su ejecución es extensa y confiable. Las gramáticas basadas en XML están abiertas a la puesta en práctica sin un esfuerzo de ingeniería inversa.

Namespace.

Es ciertamente útil tener un independiente y único visualizador de SVG. Pero SVG también puede ser utilizado como un componente en un uso de múltiples namespace XML. Por ejemplo, los gráficos de SVG se pueden incluir en un documento que utilice cualquier namespace XML, o incluso XHTML. SVG es un componente bueno, de uso general para cualquier gramática multinamespace que necesite utilizar gráficos.

Objetos Gráficos.

En cualquier gramática XML, la consideración tiene que ser dada a qué se está modelando exactamente. Por ejemplo en los formatos textuales, modelar se encuentra a nivel de párrafos y de frases, en lugar de sustantivos, adverbios o fonemas. Semejantemente, SVG modela gráficos a nivel de objetos, en lugar de puntos individuales.

SVG proporciona un elemento general de trayectoria, que se puede utilizar para crear una variedad enorme de objetos gráficos, y también proporcionar formas básicas comunes tales como rectángulos y elipses, éstos son convenientes para la codificación a mano. SVG proporciona control fino sobre el sistema coordenado en el cual se definen los objetos gráficos y las transformaciones que serán aplicadas durante la representación.

Efectos Raster.

Muchos gráficos existentes de la Web utilizan las operaciones de filtros encontrados en paquetes gráficos para crear las faltas de definición, sombras, efectos de iluminación y así sucesivamente. Con la rasterización usada con formatos de vector, SVG permite la especificación declarativa de filtros, solo o en combinación, especificando de tal manera que los gráficos sigan siendo escalables y mostrables en diversas resoluciones.

2.7.2 Convertir Imagen al Formato SVG.

Para convierte archivos multimedia de forma gratuita, rápida y en línea existe un conversor online gratuito en internet. No requiere instalación de software. Este conversor de archivos online permite la conversión de archivos multimedia fácil y rápido de un formato a otro. Da soporte a una amplia variedad de formatos de entrada.

Si se desea convertir desde imágenes raster o mapa de bits como PNG o JPG, este conversor de SVG convertirá las sombras y objetos en vectores a blanco y negro que son escalables sin pérdida de calidad. Si se requiere se pueden refinar o colorear con un programa gratuito de vectores gráficos como Inkscape, pero para fines prácticos de este trabajo no será necesario.

El conversor de Scalable Vector Graphics (SVG) permite convertir desde casi 130 formatos de origen. Para más detalles acerca del conversor consultar el link:
http://imagen.online-convert.com/es/convertir-a-svg.

2.7.3 Estructura Básica de un Documento (SVG).

SVG define un sistema de coordenadas, el área de dibujo es infinita. Los valores que pueden tomar estos atributos pueden ser simplemente números (si expresamos el tamaño en pixeles) o en unidades concretas (puntos, pulgadas, picas, centímetros, milímetros, etc.).

El origen de coordenadas comienza en la esquina superior izquierda, incrementándose la coordenada X hacia la derecha y la Y hacia abajo. El punto (0,0) se conoce como origen de coordenadas.

La Figura 2.7.2 muestra la imagen de un cuadrado en formato SVG, la estructura en formato vectorial de la imagen 2.7.2 se muestra en la Figura 2.7.3.

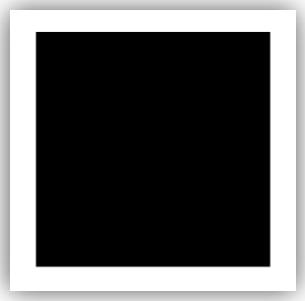


Figura 2.7.2. Imagen de un cuadrado en formato SVG.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
 width="1148.000000pt" height="560.000000pt" viewBox="0 0 1148.000000 560.000000"
 preserveAspectRatio="xMidYMid meet">
<metadata>
Created by potrace 1.11, written by Peter Selinger 2001-2013
</metadata>
<g transform="translate(0.000000,560.000000) scale(0.100000,-0.100000)"
fill="#000000" stroke="none">
<path d="M3320 2800 l0 -2350 2350 0 2350 0 0 2350 0 2350 -2350 0 -2350 0 0
-2350z"/>
</g>
</svg>

```

Figura 2.7.3 Ejemplo de código fuente de la figura vectorial 2.7.2.

Visualización del Código Fuente.

Para acceder al código fuente (source code) de la imagen se hace lo siguiente:

I. Se recomienda abrir la imagen en el navegador Google Chrome, en este navegador, las opciones de programador son de fácil acceso.

II. Entrar al código fuente de la imagen. En la parte superior derecha se encuentra un icono con tres líneas horizontales, al seleccionarlo se despliega un menú de configuración. Dentro de este menú se encuentra la parte de *Herramientas*, esta parte despliega un submenú, en el cual se encuentra la opción *"Ver código fuente"*. Al dar click sobre esta opción se desplegará una ventana nueva en el navegador la cual contiene los datos del archivo en SVG.

Al entrar en este documento XML las propiedades y parámetros que se utilizan dentro del archivo SVG se despliegan, por ejemplo en la parte inicial del código se indica el tipo de archivo, así como la versión del mismo y posteriormente se definen propiedades del archivo las cuales se pueden consultar en el link http://www.w3.org/TR/SVG11/.

2.8 Archivo de Salida de la Interfaz.

Como archivo de salida, la interfaz generada en el software *Processing*, contempla guardar la información en archivos editores de texto TXT pero con la extensión *.CBU. Éste tipo de archivo informático se encuentra compuesto únicamente por texto sin formato, sólo caracteres.

La bibliografía recomienda no usar para un archivo de texto extensiones que, están muy difundidas y son muy conocidas, porque pueden confundir tanto al usuario como al propio sistema operativo, como por ejemplo .xls, .doc, .ppt, .wav, .gif o .jpg, aunque no hay ningún impedimento real si se requieren utilizar.

Como no existe impedimento alguno para crear un archivo de texto con extensión *.CBU, dentro de la programación de la interfaz se crea un archivo con un nombre **predefinido por el programador**, está restricción se hace ya que el archivo que se genera solo se puede guardar con un nombre de hasta 8 caracteres, por lo que la extensión *.CBU se agrega automáticamente desde la interfaz. El nombre del archivo y el formato de extensión con el que se guarda la información para que el controlador pueda procesarla es "CADR.CBU". Este nombre se compone por las siglas CAD (Diseño Asistido por Computadora) y R (Robótica), seguido de la extensión CBU.

Y como ya se menciono anteriormente, el archivo debe contener la información de una serie de puntos y posiciones grabadas, solo para ser ejecutadas por el controlador, el cual mandara las órdenes a los actuadores y hará que el robot siga la trayectoria contenida en el archivo.

2.9 Interfaz CADR.

La realización de la interfaz propone pasar de la realidad virtual a la realidad, creando una relación estrecha entre hardware y software, el diseño de la interfaz se encuentra específicamente programada para el hardware; integrando una forma sencilla de utilizar el equipo de robótica.

Una vez identificadas las tareas se crea y se analizan los escenarios del usuario para definir el conjunto de objetos y acciones en la interfaz. Esto es lo que forma la base para la creación del formato de la pantalla, representa el diseño gráfico y colocación de iconos, la definición de texto descriptivo en pantalla y la especificación de los elementos primarios y secundarios del menú. Esta herramienta se utiliza para generar un prototipo y por ultimo implementar el modelo de diseño y evaluar la calidad del resultado.

2.9.1 Librerías de *Processing*.

Dos librerías a utilizar en la programación de la interfaz son:

- ✓ ***geomerative.****: Procesa la información de geometrías en 2D, es un intérprete de imágenes en formato SVG. Esta librería facilita el acceso a dibujos vectoriales en *Processing*, desarrollando una tipografía de las geometrías.
- ✓ ***controlP5.****: Emplea una gama de controladores para construir interfaces de usuario gráficas en *Processing*. Controladores tales como deslizadores, botones, perillas, campos de texto, radiobuttons, casillas de verificación, entre otros, se pueden agregar fácilmente a un esquema de processing. Se pueden organizar en grupos o pestañas.

Estas librerías están disponibles en la página principal de processing:
http://processing.org/reference/libraries/

Para agregar librerías a un sketch de processing, ir a: Sketch → Import Library → Add, buscar las librerías e instalar. Esto agrega las siguientes líneas al inicio del sketch.

```
import controlP5.*;
import geomerative.*;
```

2.9.2 Leyendo y Desplegando Datos.

La ventana principal de la interfaz grafica se muestra en la Imagen 2.9.1.

Imagen 2.9.1 Ambiente principal de la interfaz grafica de usuario.

El menú principal se encuentra compuesto de 4 pestañas (Imagen 2.9.2), cada pestaña es nombrada por un titulo principal en el que se indica la función que representa dentro de la interfaz.

Imagen 2.9.2 Menú principal.

En la pestaña ***Abrir Archivo SVG*** se despliega un submenú en el cual se da la opción de buscar la imagen a procesar dentro de la interfaz (Imagen 2.9.3).

Imagen 2.9.3 Abrir Archivo SVG.

Al seleccionar la opción ***Buscar*** se despliega una ventana facilitando la búsqueda del documento SVG. (Imagen 2.9.4).

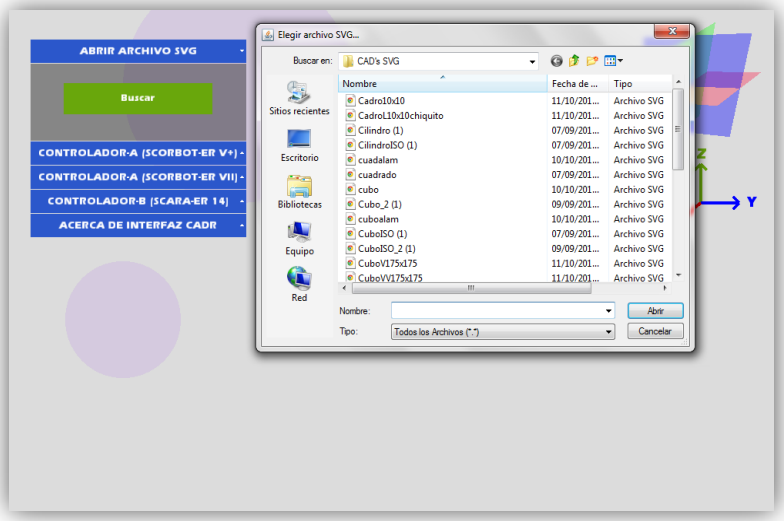


Imagen 2.9.4 Ventana de búsqueda.

Después de buscar y abrir el archivo SVG, inmediatamente la interfaz procesa la información y la despliega redibujando el archivo y encontrando los puntos de unión de los objetos gráficos y proporcionando un elemento general de trayectoria. En la Imagen 2.9.5 se muestran varias imágenes procesadas por la interfaz.

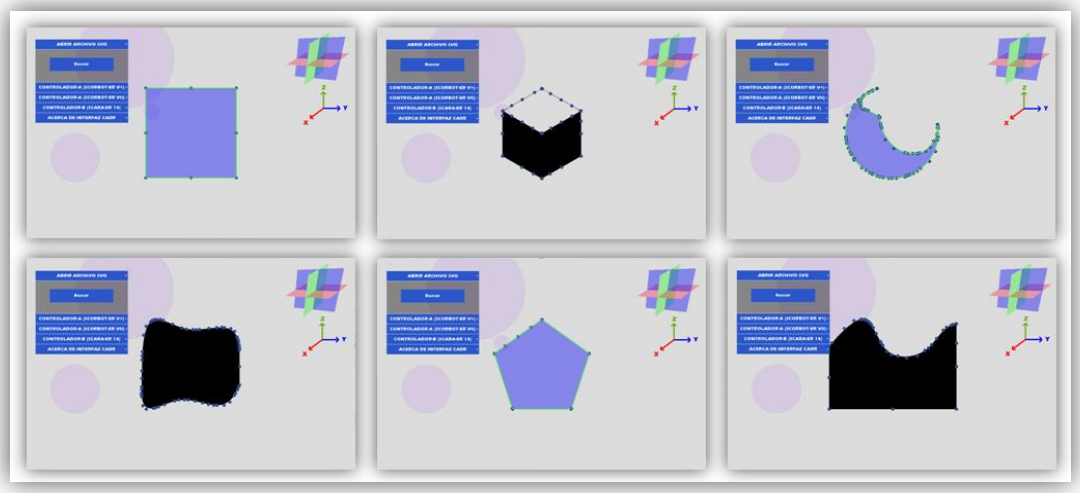


Imagen 2.9.5 Imágenes procesadas por la interfaz.

La interfaz localiza los puntos y la trayectoria generada. Es importante definir el término trayectoria. La trayectoria se divide en dos aspectos importantes:

- ***Perfil de trayectoria***, que se entiende como la evolución en el tiempo de la posición, velocidad y aceleración con que viajará el órgano terminal o el objeto de trabajo, y
- ***Lugar geométrico de la trayectoria***, que representa los puntos cartesianos por donde pasará el órgano terminal o el objeto manipulado.

El tipo de información que se genera de la interfaz proviene del formato SVG, el lugar geométrico de la trayectoria está representada por puntos que se visualizan de color azul mientras que la trayectoria es color verde (Imagen 2.9.6). La programación de la interfaz para la localización de los puntos y trayectoria, se realizo con la modificación de un programa de demostración de la librería *geomerative*, la modificación se realizó de tal modo que cumpliera con la característica de encontrar puntos en coordenadas cartesianas de la imagen en formato SVG.

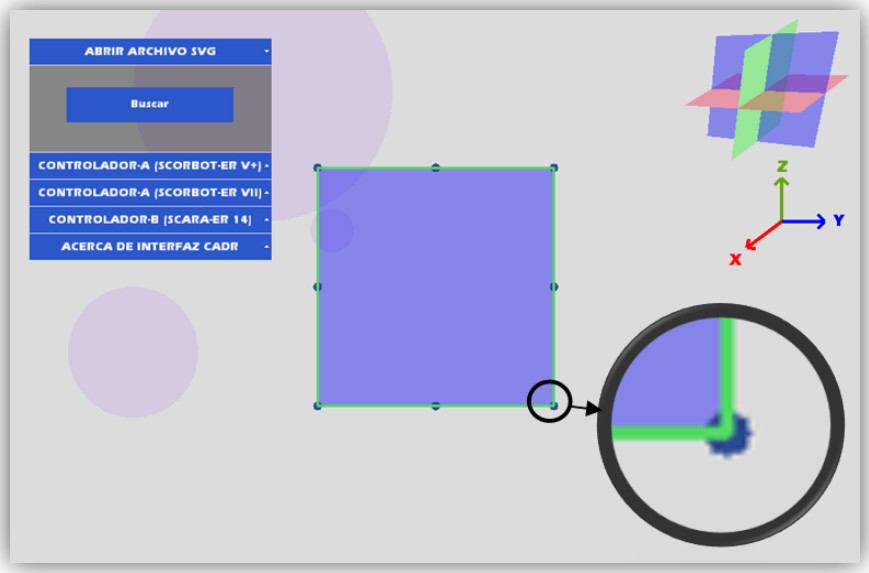


Imagen 2.9.6 Localización de puntos y trayectorias de la imagen en formato SVG.

Para explicaciones posteriores se tomará como ejemplo la imagen 2.9.6 (Cuadrado.SVG).

Las 3 pestañas siguientes del menú principal (Imagen 2.9.2), indican para qué tipo de robot y controlador se va a generar al archivo **CADR.CBU**.

2.9.3 Generando Coordenadas y Dimensión.

El lienzo SVG describe "el espacio donde se muestra el contenido SVG". El lienzo es infinito, pero se representa en una región rectangular finita. Esta región rectangular finita se llama visor de SVG, es el área de visualización en la que el usuario ve el contenido SVG.

El tamaño de la ventana gráfica SVG es decir, el ancho y la altura se determinan por un proceso, entre el documento SVG y su matriz de información, proporcionando lo siguiente:

- Un número (normalmente un número entero) representa el ancho en "píxeles" de la ventana gráfica.
- Un número (por lo general un número entero) representa la altura en "píxeles" de la ventana gráfica.
- (Altamente deseable pero no necesario), un número puede indicar el tamaño en unidades del mundo real, tales como milímetros, de un "pixel".

Con la información anterior, una imagen SVG determina, una ventana inicial para el sistema de coordenadas y un usuario inicial para el sistema de coordenadas. Uno o ambos

sistemas de coordenadas son establecidos de tal manera que el origen coincide con el origen de la ventana de visualización (esquina superior izquierda).

Las longitudes en SVG se pueden especificar como:

- (Sin identificador de unidad), valores en el espacio de usuario, por ejemplo: "15".
- (Con identificador de unidad) una longitud expresada como una unidad de medida absoluta o relativa, por ejemplo: "15mm" o "5em".

Los indicadores de unidad para una longitud en imágenes SVG son: em, ex, px, pt, pc, cm, mm, in, y porcentajes.

Un nuevo espacio de usuario, es decir, un nuevo sistema de coordenadas, se puede establecer en cualquier lugar dentro de un fragmento del documento SVG mediante matrices de transformación u operaciones de transformación simples, tales como rotación y traslación. Estás son operaciones fundamentales para gráficos en 2D y representan el método habitual de controlar el tamaño, posición, rotación de objetos gráficos.

Todas las coordenadas y longitudes en SVG pueden especificarse con o sin un identificador de unidad.

Cuando un valor de coordenadas o longitud es un número, sin identificador de unidad (por ejemplo, "25"), entonces se asume la cota como longitud en unidades de usuario (es decir, un valor en el sistema de coordenadas actual). Por ejemplo:

<text font-size="50">Text size is 50 user units</text>

Alternativamente, un valor de coordenadas o de longitud puede ser expresado como un número, seguido de un identificador de unidad. La lista de los identificadores de unidad en SVG coincide con la lista de los identificadores de unidad en Cascading Style Sheets, o sus siglas CSS: *em, ex, px, pt, pc, cm, mm* y *pulg*. El tipo <length> también pueden tener un identificador de unidad porcentual. A continuación se describe cómo se procesan los diferentes indicadores:

- Al igual que en CSS, los identificadores de unidad "*em*" y "*ex*" son relativas a la fuente de tamaño en altura de la x, respectivamente.
- Una unidad de píxeles se define para ser igual a una unidad de usuario. Por lo tanto, una longitud de "*5px*"" es lo mismo que una longitud de "*5*".
- Los otros identificadores de unidad absolutos de CSS (p.e. pt, pc, cm, mm, in) están todos definidos como un múltiplo adecuado de una unidad de píxeles. Por ejemplo, "1px" (un píxel) = "0.2822222mm". Entonces, para todo el procesamiento del contenido SVG:
  - "1pt" es igual a "1.25px" (y por lo tanto, 1.25 unidades de usuario)
  - "1pc" es igual a "15px" (y por lo tanto, 15 unidades de usuario)

- ☞ “1mm” sería “3.543307px” (3.543307 unidades de usuario)
- ☞ “1cm” es igual a “35.43307px” (y por lo tanto 35.43307 unidades de usuario)
- ☞ “1in” es igual a “90px” (y por lo tanto, 90 unidades de usuario)

- Los valores porcentuales se definen como en relación al tamaño de ventana.

Con la información anterior y la información del apartado 2.6.2 Formato Vectorial, se sabe qué, una imagen SVG normalmente se compone de un conjunto de trazos se puede agrupar, formando objetos, y crear formas más complejas a partir de la posición de sus puntos inicial y final y por una función que describe el camino entre ellos. De este modo la interfaz CADR, procesa las imágenes SVG, con indicador de unidades en “pt” ya que es el tipo de indicador que la pagina on-line http://imagen.online-convert.com/es/convertir-a-svg, asigna a las imágenes con formato SVG.

En la interfaz CADR se encuentra programada una subrutina que convierte la información de unidades “*pt*” a “*cm*”, utilizando un factor de dimensión, un fragmento del código de programación es el siguiente:

```
factdim = dimension/50;       //Factor de dimensión
CU=(((1.25/35.43307)*3.05)*factdim)*10;     //Conversión de Unidades (Pixeles vs. cm)
```

La conversión de unidades se lleva a cabo después de la utilización de una subrutina del programa que localiza los puntos y trayectorias.

2.9.4 Submenú Controlador-A (Scorbot-ER VII).

Después de haber cargado la imagen a la interfaz se debe elegir una opción de las pestañas disponibles que indican el controlador y robot para el cual se va a generar el archivo CADR.CBU. En este subtema se explica el funcionamiento del submenú ***Controlador-A (Scorbot-ER VII)***.

Lo primero que se tiene que hacer al entrar al submenú, es elegir la dimensión que se requiere para que el robot siga el *lugar geométrico de la trayectoria*, esto se hace deslizando el slider hasta la dimensión deseada, de igual modo la interfaz presenta una animación en la que se va modificando la escala del dibujo, (Imagen 2.9.8).

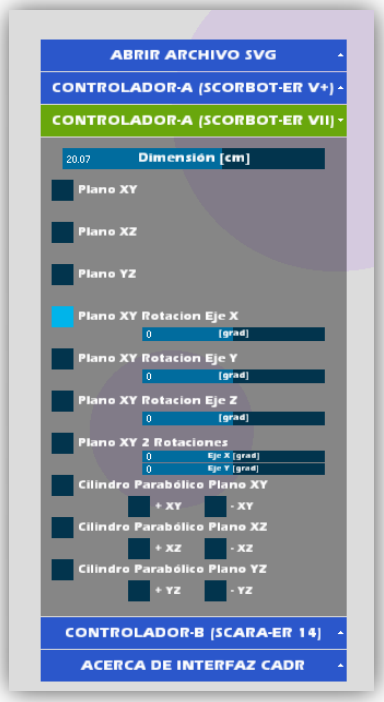


Imagen 2.9.7 Submenú Controlador-A (Scorbot-ER VII).

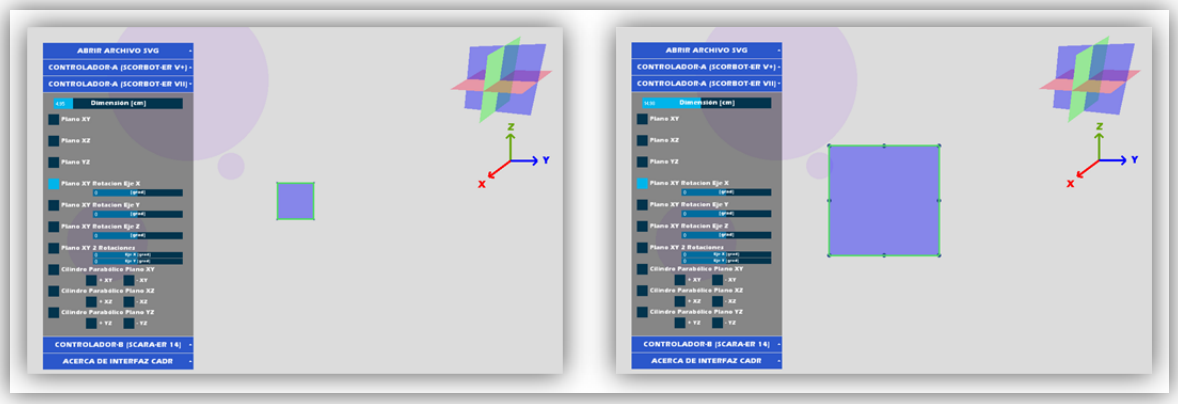


Imagen 2.9.8 (Izquierda) Cuadrado de dimensión 5cm, (Derecha) Cuadrado de 15 cm de dimensión.

Las coordenadas del *lugar geométrico de la trayectoria* se dibujan en un área restringida, como prototipo de interfaz el área se delimita dependiendo del rango de alcance del robot.

Al seleccionar el icono cuadrado de color azul (Imagen 2.9.9), se presenta una animación en la interfaz sobre que plano se van a escribir las coordenadas en el archivo CADR.CBU. La interfaz CADR solo pude generar un archivo a la vez, es decir si se selecciona *Plano XY*, el archivo CADR.CBU se genera con la información del *lugar geométrico de la trayectoria* para ese plano en el área delimitada por el programador, y si posteriormente se

elige *Plano YZ*, la información en el archivo CADR.CBU se reemplaza por el nuevo *lugar geométrico de la trayectoria* del plano seleccionado.

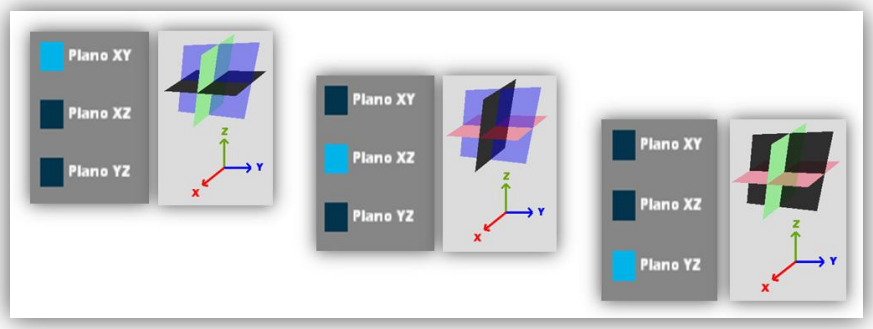


Imagen 2.9.9 Visualización del plano sobre el cual se generan las coordenadas.

Se proporcionan otras opciones más como por ejemplo rotar el *plano XY*, indicando físicamente que las coordenadas del dibujo tendrán ciertos grados de inclinación en el eje *X* y *Y* o rotación en el caso del eje *Z*, también se ofrece una opción más con dos rotaciones simultaneas. Para generar el archivo con este tipo de información se debe seleccionar a cuantos grados de inclinación o rotación se requiere. La selección se realiza deslizando el slider destinado a cada rotación predefinida, ya sea en el eje *X*, *Y*, *Z* o con dos rotaciones tanto en el eje *X* como en *Y* simultáneamente.

En la Imagen 2.9.10 se muestra algunos ejemplos de la rotación sobre el eje *X* del *plano XY*, recordando que las rotaciones programadas en la interfaz solo se realizan en este plano, el plano XY es una representación grafica ya que en él están contenidas las coordenadas del archivo SVG, finalmente el archivo CADR.CBU que se genera contiene la información seleccionada de esté submenú.

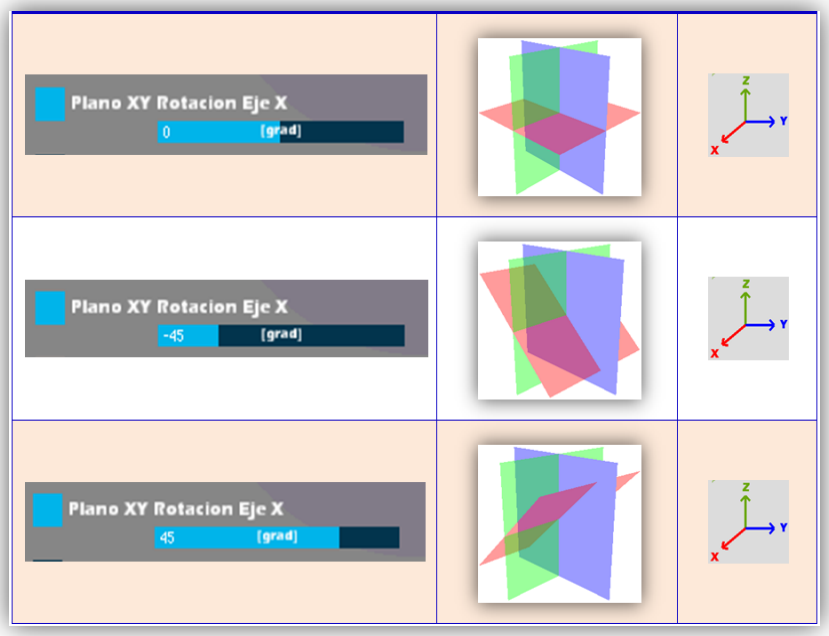


Imagen 2.9.10 Ejemplo de visualización física del lugar geométrico de la trayectoria generada por el robot, el archivo CADR.CBU contiene la información de la rotación sobre el plano XY.

Orientaciones Canónicas.

Dentro de la interfaz CADR, se genera la rotación del *plano XY* en relación a los ejes *X*, *Y*, *Z* o *XY*, este ultimo de manera simultánea. Las ecuaciones para lograr la rotación del *plano XY* se encuentra programadas en la interfaz, estás ecuaciones se denominan orientaciones canonícas, existen tres orientaciones particulares que puede guardar un sistema {B} con respecto a un sistema {A}. Estas orientaciones canonícas son las siguientes:

- Orientación alrededor del eje *x* con un desplazamiento angular α.

$$R_B^A(x,\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \in \mathcal{R}^3$$

- Orientación alrededor del eje *y* con un desplazamiento angular β.

$$R_B^A(y,\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \in \mathcal{R}^3$$

- Orientación alrededor del eje *z* con un desplazamiento angular θ.

$$R_B^A(z,\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \in \mathcal{R}^3$$

Para finalizar con la explicación del submenú *Controlador-A (Scorbot-ER VII)* se proporcionan otras opciones más, llamadas *Cilindro Parabólico*, estás opciones hacen que la información contenida en los planos *XY*, *XZ y YZ*, describan una superficie en forma de parábola positiva o negativa dependiendo de la opción a elegir. La interfaz contiene en su programación una parábola del tipo:

$$y = \pm 0.1 * (x)^2$$

Internamente la interfaz hace los cálculos para proyectar las coordenadas de la imagen de formato SVG, a una superficie conocida como cilindro parabólico, la dirección de la parábola así como la orientación sobre que plano se desea hacer la proyección, se puede elegir en la interfaz.

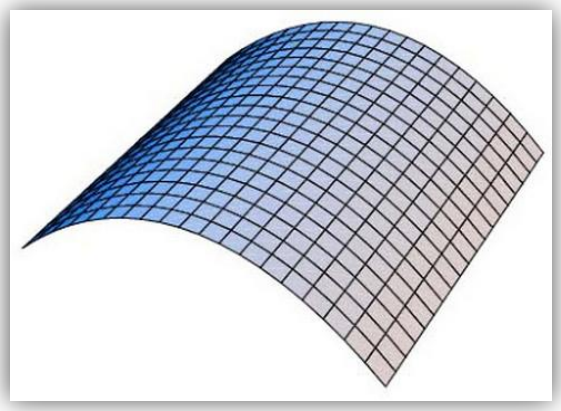


Imagen 2.9.11 Cilindro parabólico.

☞ *Nota: La explicación del submenú para el **Controlador-A (Scorbot-ER VII)** es la misma para Controlador-B (Scora-ER 14) y Controlador-A (Scorbot-ER V+). Solo que para el Controlador-A (Scorbot-ER V+) se omiten algunas opciones para generar coordenadas debido a la configuración mecánica el robot.*

2.9.5 Exportar Interfaz CADR como Aplicación.

Processing brinda la posibilidad de exportar los proyectos generando un applet de Java, esta opción permite generar una aplicación lista para ejecutarse en sistemas operativos Mac OS, Linux, Windows. Todo esto se generará en directorios dentro del directorio principal de la aplicación donde también se genera el archivo CADR.CBU dentro de la carpeta de nombre "data".

Los pasos para la exportación de una aplicación se llevan a cabo desde el software processing: File → Export Application, seleccionar la plataforma y exportar.

En un mundo globalizado por el desarrollo de la informática computacional la exportación de aplicaciones representa una gran ventaja, que para este trabajo asigna un punto más a favor en la búsqueda de poner a la vanguardia el Laboratorio de Manufactura Avanzada, integrando finalmente esta interfaz CADR como complemento a la puesta en marcha de la rehabilitación de sus equipos en las aéreas de Robótica Didáctica y Robótica Industrial.

2.9.6 Interfaz CADR.CBU y Software ATS.

Finalmente el archivo generado en la interfaz CADR.CBU debe ser leído por el controlador que enviará la información al brazo robot, esta información se carga mediante el software ATS.

Los siguientes pasos cargarán el contenido de este archivo a la BBRAM del controlador, utilizando el software ATS.

1. Desde la pantalla principal del ATS (ver Figura 1.5.2), presionar <Shift> + F10. El menú del Gestor de Archivos se activa.
2. En Backup Directory (Directorio de Respaldo): Escriba la ruta de dirección en la que se halla el archivo CADR.CBU y <Enter>. Se recomienda hacer la copia del archivo CADR.CBU a una carpeta de nombre ROBOT a la raíz del disco duro para que la ruta de dirección sea C:\ROBOT.
3. En Backup Restore usar las flechas para señalar PROGRAMS.
4. En During Restore usar las flechas para señalar ADD TO.
5. En File name (Nombre de archivo): Escriba CADR y presione <Enter>. La extensión CBU no es necesario escribirla.
6. Presione F5 para cargar (RESTORE) el archivo del disco a la BBRAM del controlador.
7. Cuando aparece “DONE”, presione <Esc> para volver a la pantalla principal de ATS.

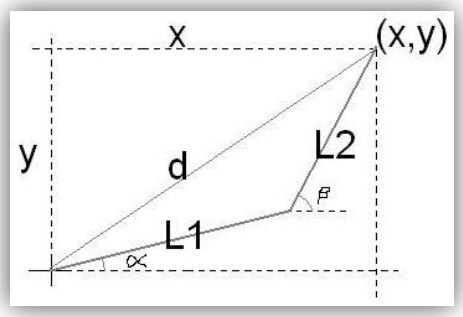
Y finalmente en la pantalla principal del ATS escribir RUN CADR <Enter>. Y el PCH del robot seguirá el *lugar geométrico de la trayectoria*, descrita por la imagen SVG.

3. Pruebas y Validación.

Esta es la etapa final en la que se muestra el lugar geométrico de trayectoria, recorrido por el punto central de la herramienta del robot, descrita por la imagen vectorizada en formato SVG.

3.1 Prueba #1 (El inicio).

Inicialmente este trabajo comenzó con la aplicación de la cinemática inversa, particularmente en el Robot Scora-ER 14Pro, el archivo CADR.CBU se creaba con información en *coordenadas de ejes (Joints)*, está información se lograba despejando el valor de los ángulos, en la que se involucran los datos de la coordenada *(X,Y)* de un punto deseado y la longitud de los eslabones del robot, (Imagen 3.1.1).



$$\alpha = \tan^{-1}\left(\frac{Ly}{Lx}\right) + \cos^{-1}\left(\frac{L1^2+Lx^2+Ly^2-L2^2}{2*L1*\sqrt{Lx^2+Ly^2}}\right) \qquad \text{ec. 3.1.1}$$

$$\beta = \pi + \cos^{-1}\left(\frac{L1^2+L2^2-(Lx^2+Ly^2)}{2*L1*L2}\right) \qquad \text{ec. 3.1.2}$$

Imagen 3.1.1 (izquierda) Representación grafica de los eslabones del robot. (Derecha) Ecuaciones que determinan el valor de los ángulos en el plano XY.

En cada punto que la imagen SVG convertía en coordenadas (X,Y), debía calcularse el valor de los ángulos, para que el robot se moviera en el plano XY, con la información de los ángulos, la relación de transmisión y la resolución del codificador, se obtenían valores en coordenadas de ejes (Joints). Los resultados fueron los siguientes:

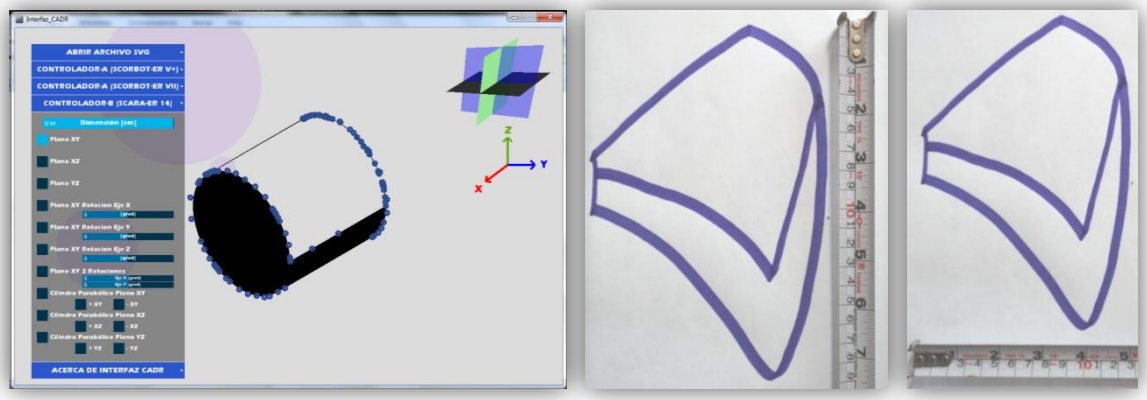


Imagen 3.1.2(a) Resultados de la utilización de las ecuaciones 3.1.1 y 3.1.2. (Izquierda) Imagen de un cilindro en vista isométrica, dimensión 18cm. (Derecha) Resultados del lugar geométrico de la trayectoria realizada por el robot Scora-ER 14Pro.

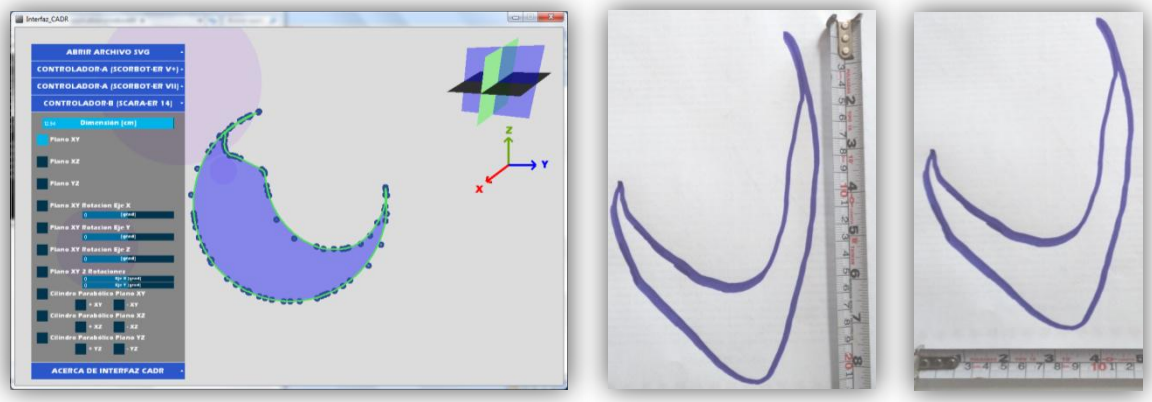


Imagen 3.1.2(b) Resultados de la utilización de las ecuaciones 3.1.1 y 3.1.2. (Izquierda) Imagen de una esfera en vista isométrica, dimensión 18cm. (Derecha) Resultados del lugar geométrico de la trayectoria realizada por el robot Scora-ER 14Pro.

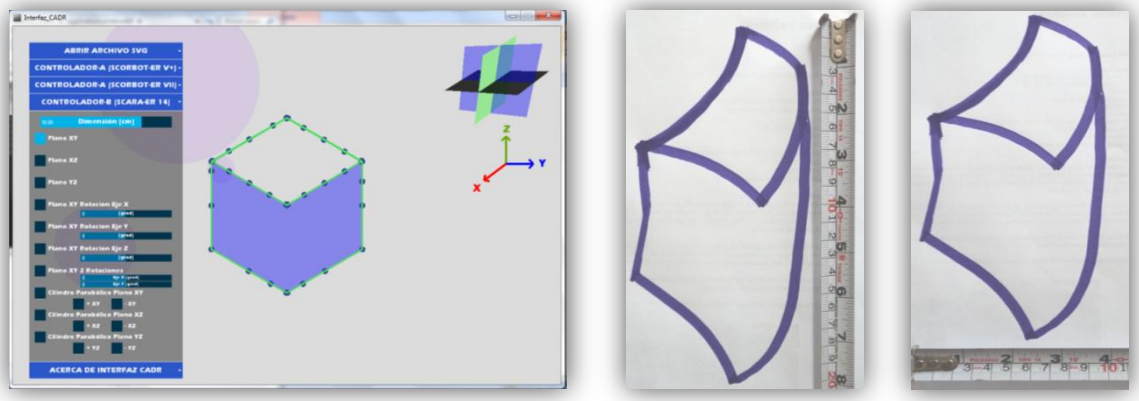
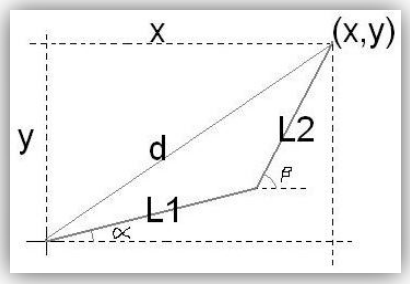


Imagen 3.1.2(c) Resultados de la utilización de las ecuaciones 3.1.1 y 3.1.2. (Izquierda) Imagen de un cubo en vista isométrica, dimensión 18cm. (Derecha) Resultados del lugar geométrico de la trayectoria realizada por el robot Scora-ER 14Pro.

Como se puede observar, las imágenes carecen de forma y de dimensión, por lo que posteriormente se implemento otra ecuación (Imagen 3.1.3), los resultados obtenidos fueron similares a los anteriores y no existieron cambios significativos.



$$\alpha = \tan^{-1}\left(\frac{Ly}{Lx}\right) - \cos^{-1}\left(\frac{L1^2+Lx^2+Ly^2-L2^2}{2*L1*\sqrt{Lx^2+Ly^2}}\right) \qquad \text{ec. 3.1.3}$$

$$\beta = \pi - \sin^{-1}\left(\frac{\sqrt{Lx^2+Ly^2}*\sin\left(\cos^{-1}\left(\frac{L1^2+Lx^2+Ly^2-L2^2}{2*L1*\sqrt{Lx^2+Ly^2}}\right)\right)}{L2}\right) \qquad \text{ec. 3.1.4}$$

Imagen 3.1.3 (izquierda) Representación grafica de los eslabones del robot. (Derecha) Ecuaciones que determinan el valor de los ángulos en el plano XY.

Algunos resultados obtenidos fueron:

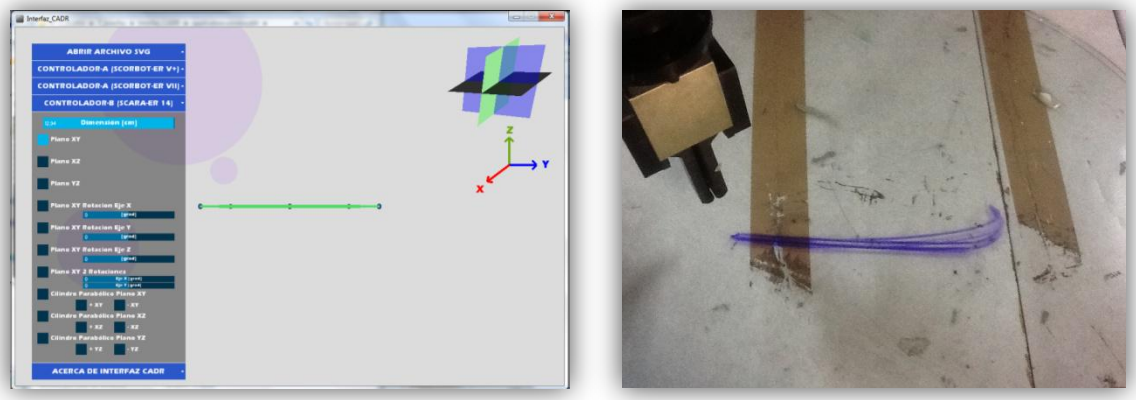


Imagen 3.1.4(a) Resultados adicionales de la utilización de las ecuaciones 3.1.3 y 3.1.4. (Izquierda) Imagen de una línea horizontal. (Derecha) Resultados del lugar geométrico de la trayectoria realizada por el robot Scora-ER 14Pro.

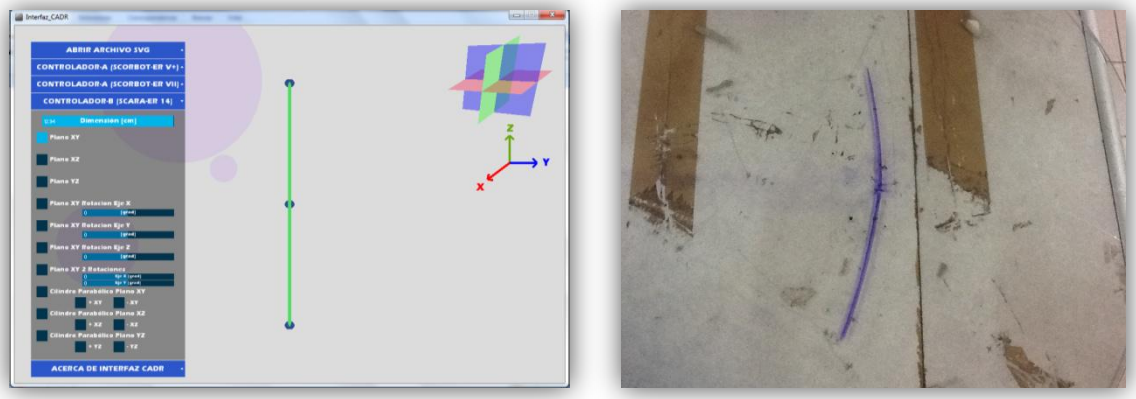


Imagen 3.1.4(b) Resultados adicionales de la utilización de las ecuaciones 3.1.3 y 3.1.4. (Izquierda) Imagen de una línea vertical. (Derecha) Resultados del lugar geométrico de la trayectoria realizada por el robot Scora-ER 14Pro.

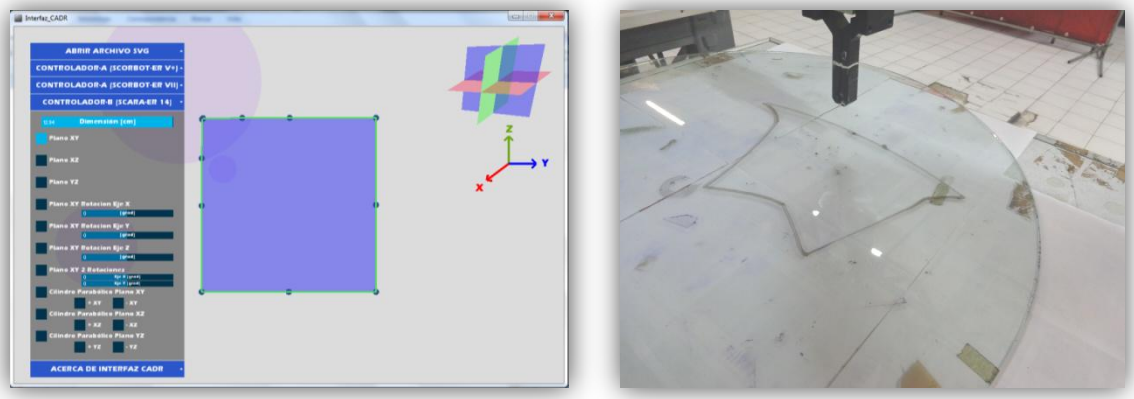


Imagen 3.1.4(c) Resultados adicionales de la utilización de las ecuaciones 3.1.3 y 3.1.4. (Izquierda) Imagen de un cuadrado. (Derecha) Resultados del lugar geométrico de la trayectoria realizada por el robot Scora-ER 14Pro.

Como se puede observa en las imágenes 3.1.2 y 3.1.4, las imágenes dibujadas por el robot carecen de forma y dimensión en relación al formato SVG, la conclusión a la que se llego fue qué los parámetros del controlador debían ser modificados, ya que las ganancias del control que determina la posición, no pueden reaccionar inmediatamente a la señal de entrada, siempre habrá un retraso entre la generación de una señal de error y la corrección actual del valor controlado. Ante esta situación solo la empresa Intelitek puede modificar los parámetros de control, que para estas pruebas se utilizo en Controlador-B y el Robot Scorbot-ER 14Pro, por lo que no es posible hacer modificaciones de los parámetros y valores en las ganancias del control. Además de existir una restricción por la empresa Intelitek, también existe otra por parte del Departamento de Manufactura Avanzada ya que se trabaja con equipo de laboratorio y solo personal calificado puede hacer modificaciones.

3.2 Prueba #2 (Validación).

Los resultados anteriores no muestra una validación integra, a si que se decide cambiar de estrategia; después de entender y comprender la *Guía de Referencia de ACL* y el método de operación del los controladores, se tiene el conocimiento de que los robots pueden operar en un sistema de *coordenadas cartesianas (XYZ)*, sí el robot se puede mover en esas coordenadas entonces se pude crear un programa en el controlador-A o –B, y el respaldo de esa información contiene las coordenadas cartesianas y su magnitud de los puntos que puede recorrer el PCH, entonces de esa manera puede restaurarse un programa al controlador, y ese programa lo genera la interfaz CADR; por lo tanto, para validar el desarrollo de esta trabajo ahora se aborda la etapa de resultados final que simplemente maneja coordenadas cartesianas.

Un aspecto importante de esta prueba de validación es que el desarrollo de este trabajo contempla las tres configuraciones de robots: Scorbot-ER V Plus, Scorbot-ER VII y Scora-ER 14Pro.

☞ Primero se validará la etapa de dimensión, eso incluye a las tres configuraciones de robots.

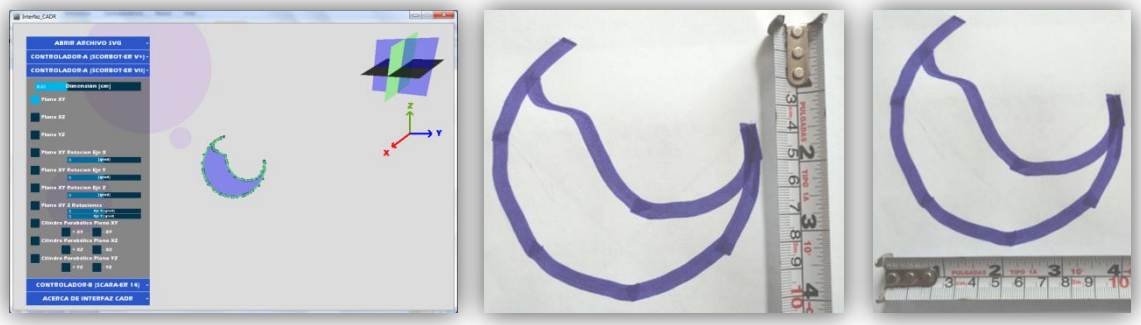


Imagen 3.2.1(a) Imagen de una esfera en vista isométrica, dimensión 10cm.

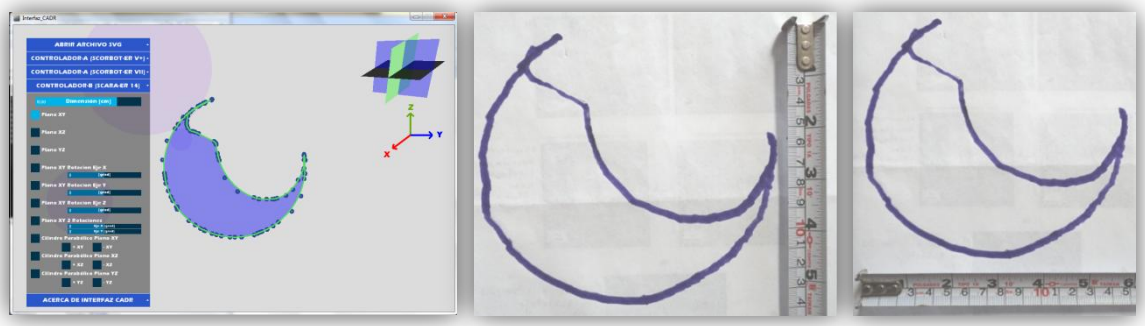


Imagen 3.2.1(b) Imagen de una esfera en vista isométrica, dimensión 15cm.

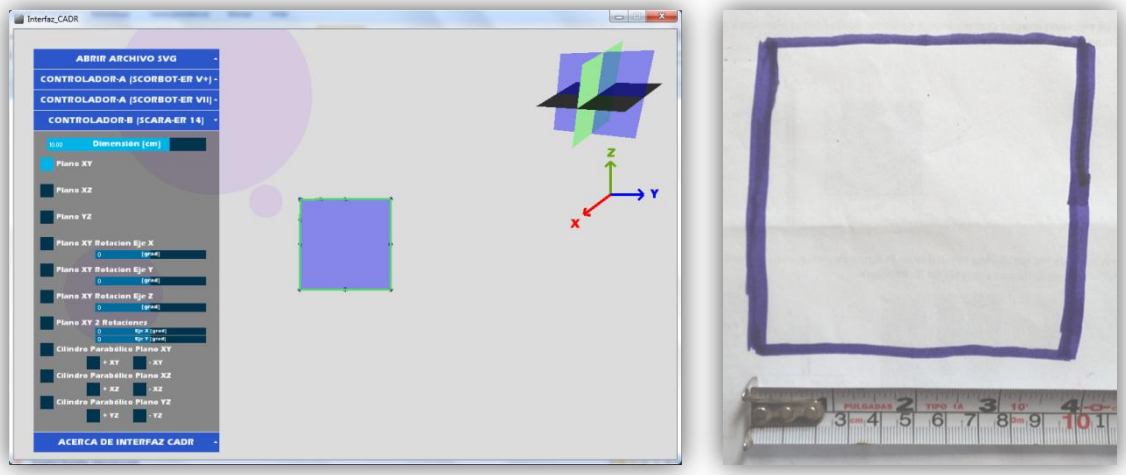


Imagen 3.2.1(c) Imagen de un cuadrado, dimensión 10cm.

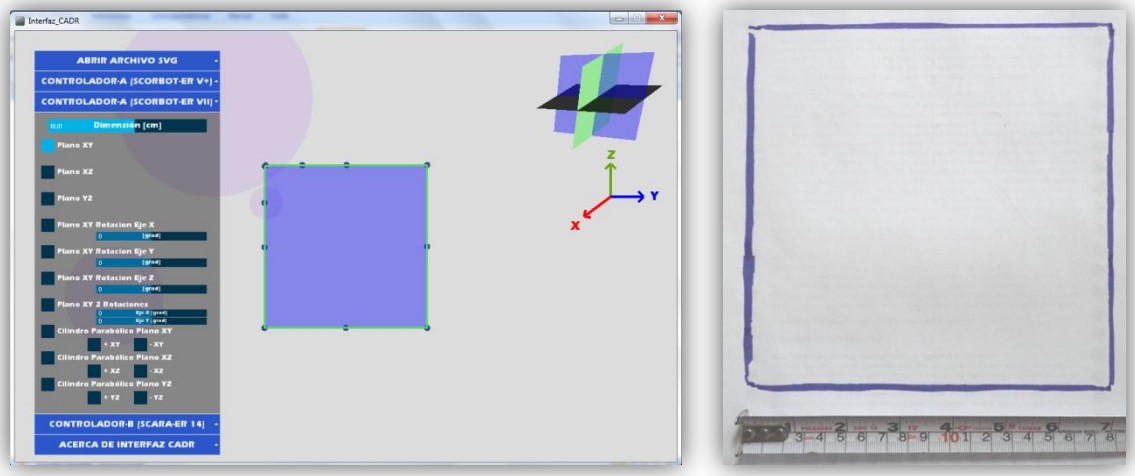


Imagen 3.2.1(d) Imagen de un cuadrado, dimensión 18cm.

Como se puede observar en las imágenes 3.2.1, se comprueba el funcionamiento de la interfaz en la etapa de dimensión, el robot recorre el lugar geométrico de la trayectoria con las dimensiones que se declaran por medio de la interfaz a lo largo y ancho de la imagen, esto funciona para las tres configuraciones de robots existentes en el Laboratorio de Manufactura Avanzada, con la conexión de sus respectivos controladores –A o –B.

A continuación se mostraran imágenes de los robots, en las cuales se muestra un dibujo trazado por el recorrido del robot a trasvés del lugar geométrico de la trayectoria descrita por una imagen SVG.

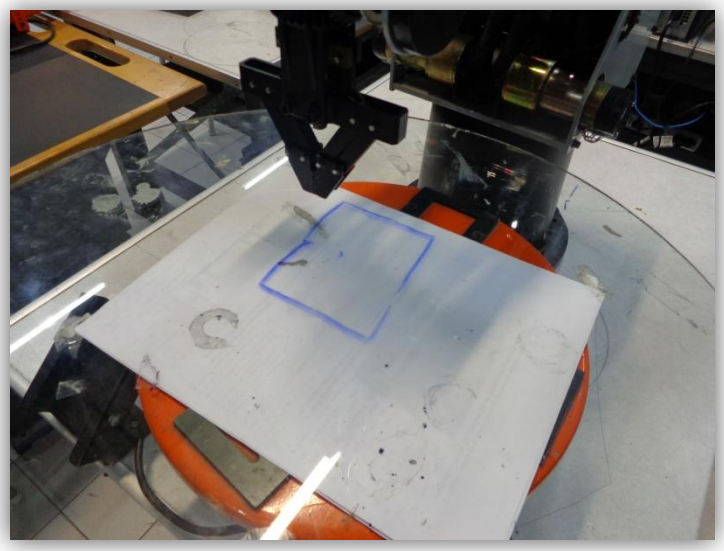


Imagen 3.2.2 Robot Scorbot-ER V Plus, dibujo en el Plano XY.

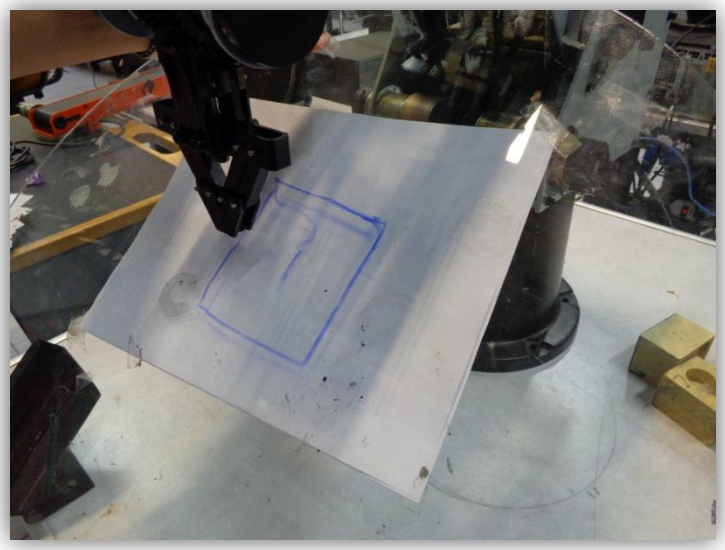


Imagen 3.2.3 Robot Scorbot-ER V Plus, dibujo en el Plano XY con rotación en el eje Y de 45°.

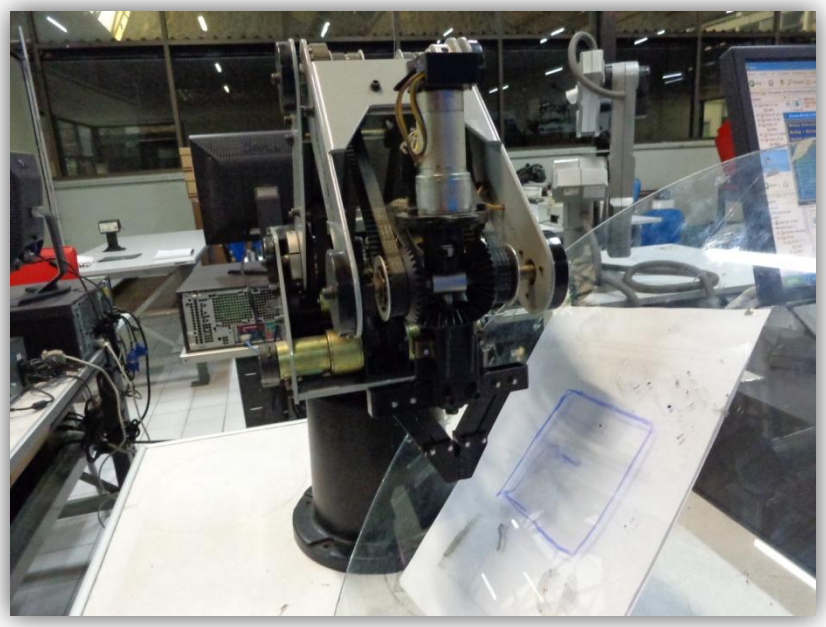


Imagen 3.2.4 Robot Scorbot-ER V Plus, dibujo en el Plano XY con rotación en el eje X de 45°.

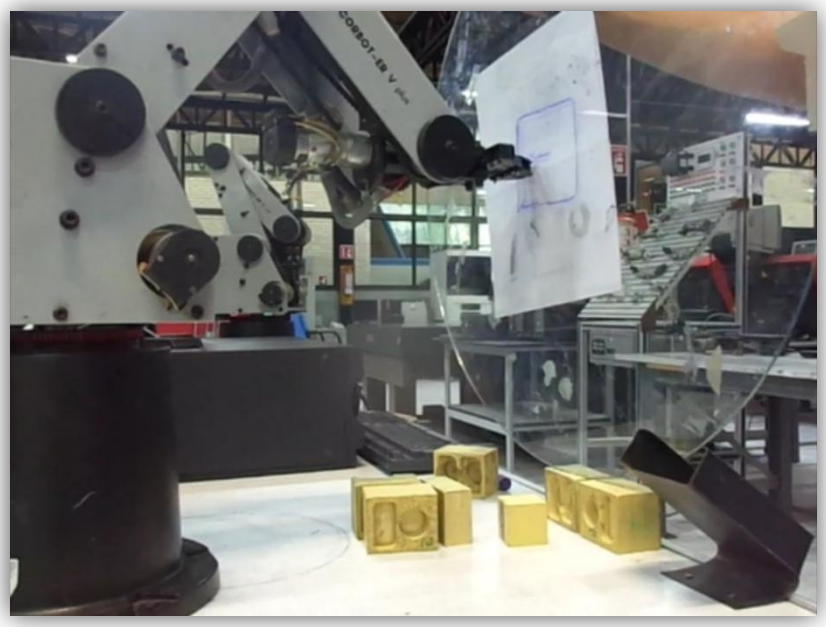


Imagen 3.2.5 Robot Scorbot-ER V Plus, dibujo en el Plano YZ.

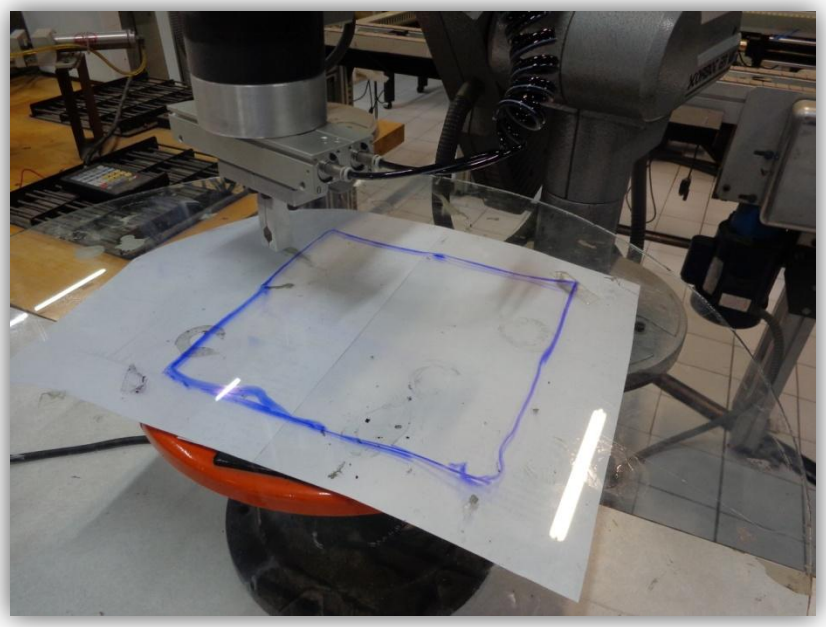


Imagen 3.2.6 Robot Scorbot-ER VII, dibujo en el Plano XY.

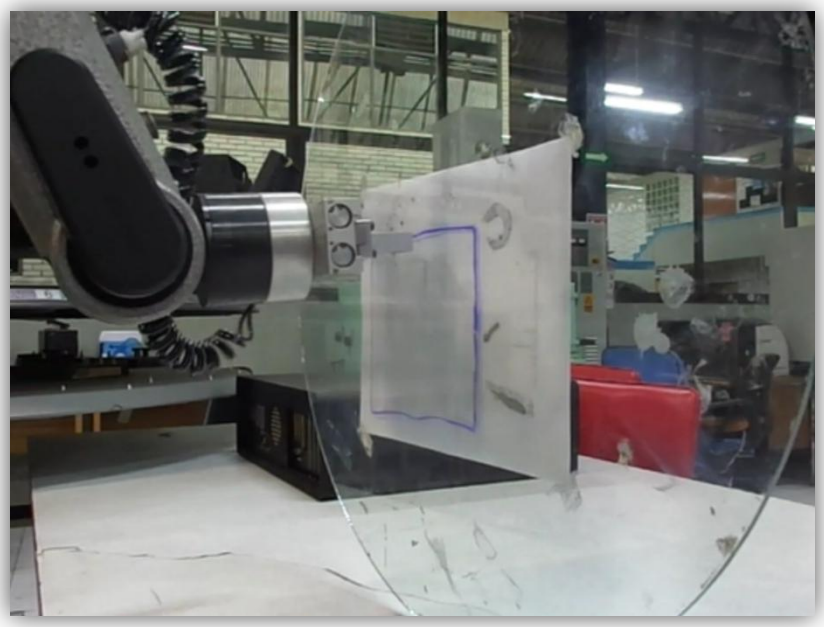


Imagen 3.2.7 Robot Scorbot-ER VII, dibujo en el Plano YZ.

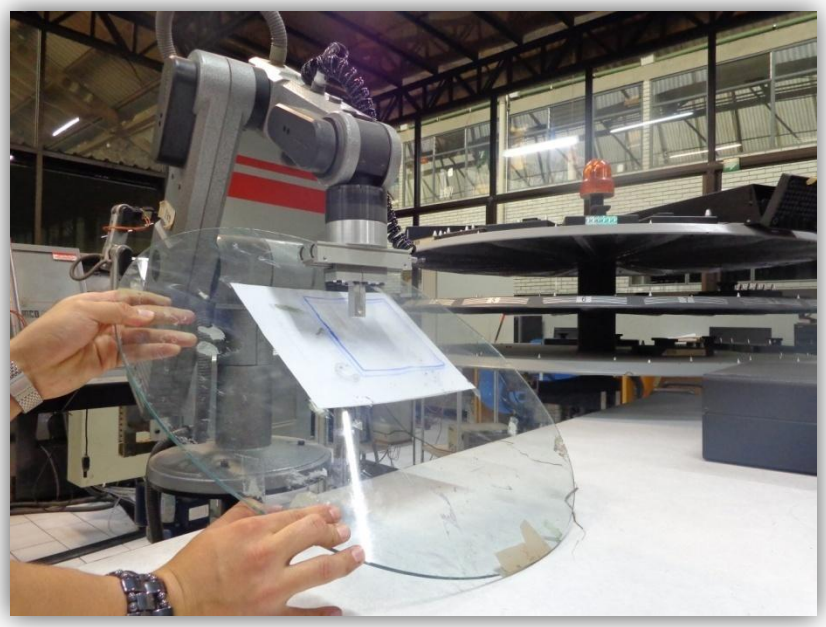


Imagen 3.2.8 Robot Scorbot-ER VII, dibujo en el Plano XY con rotación en el eje Y de 45°.

Imagen 3.2.9 Robot Scorbot-ER VII, dibujo en el Plano XY con rotación en el eje X de 45°.

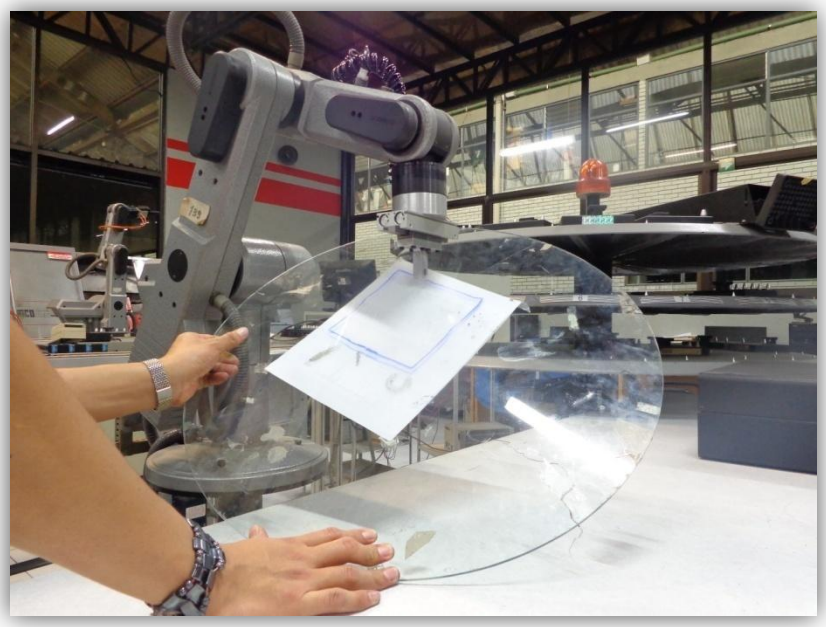


Imagen 3.2.10 Robot Scorbot-ER VII, dibujo en el Plano XY con rotaciones en X y Y de 45°.

Imagen 3.2.11 Robot Scora-ER 14Pro, dibujos en el Plano XY.

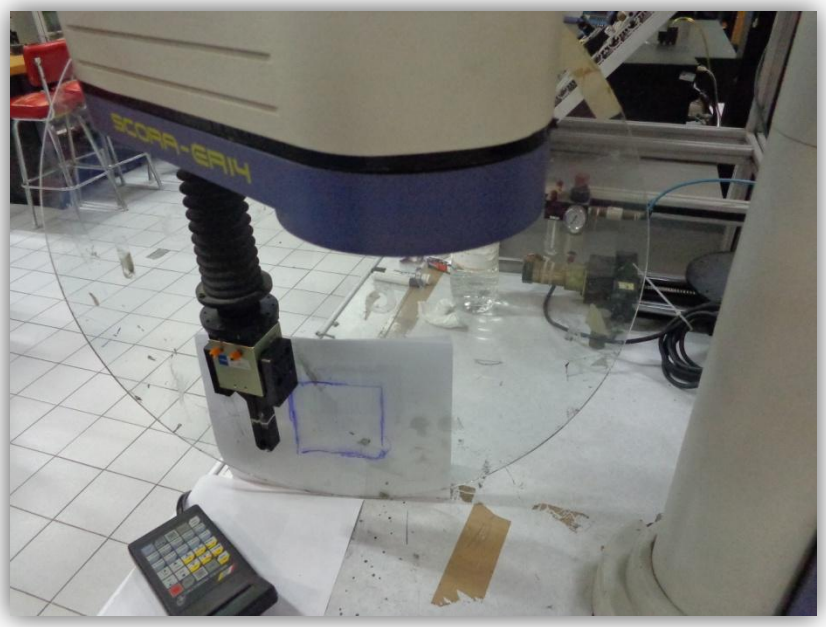


Imagen 3.2.12 Robot Scora-ER 14Pro, dibujo en el Plano XZ.

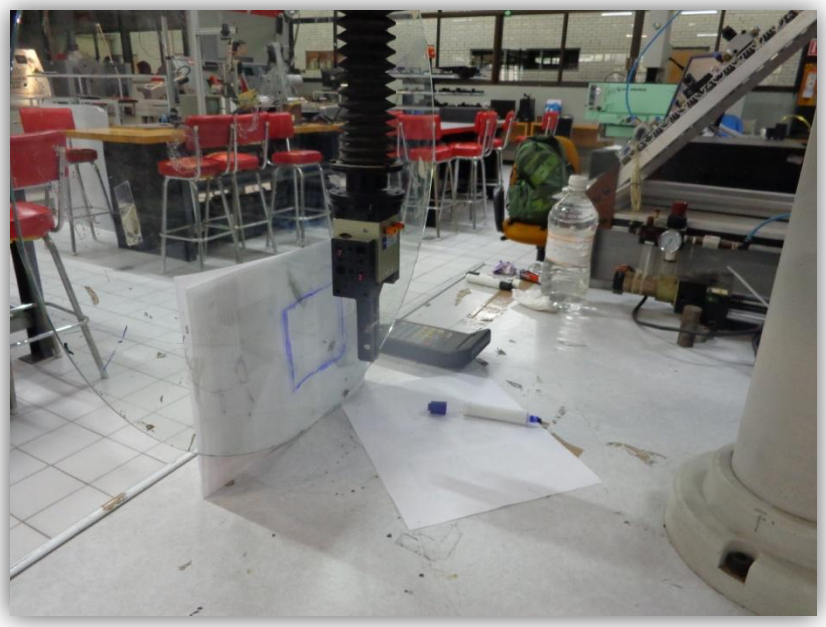


Imagen 3.2.13 Robot Scora-ER 14Pro, dibujo en el Plano YZ.

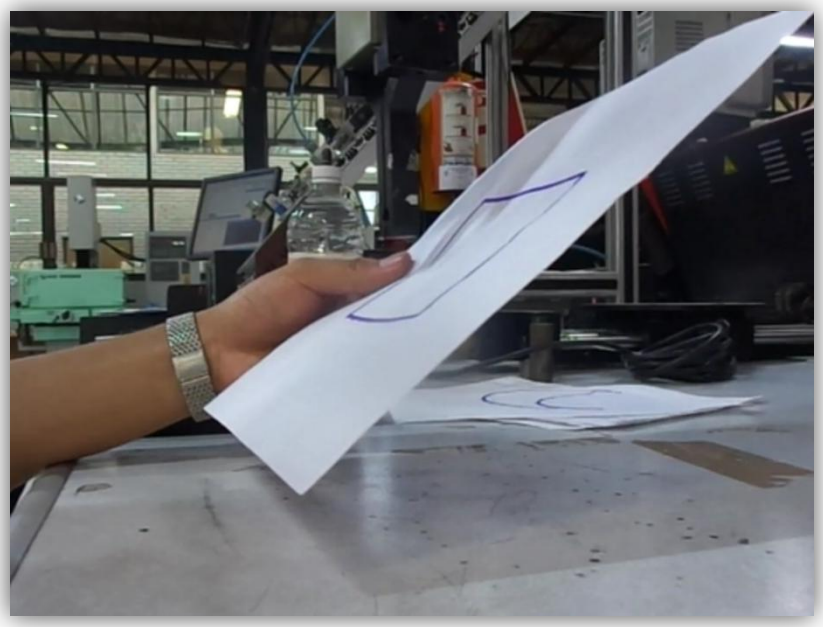


Imagen 3.2.14 Robot Scora-ER 14Pro, dibujo en el Plano XY con rotación en el eje Y de 45°.

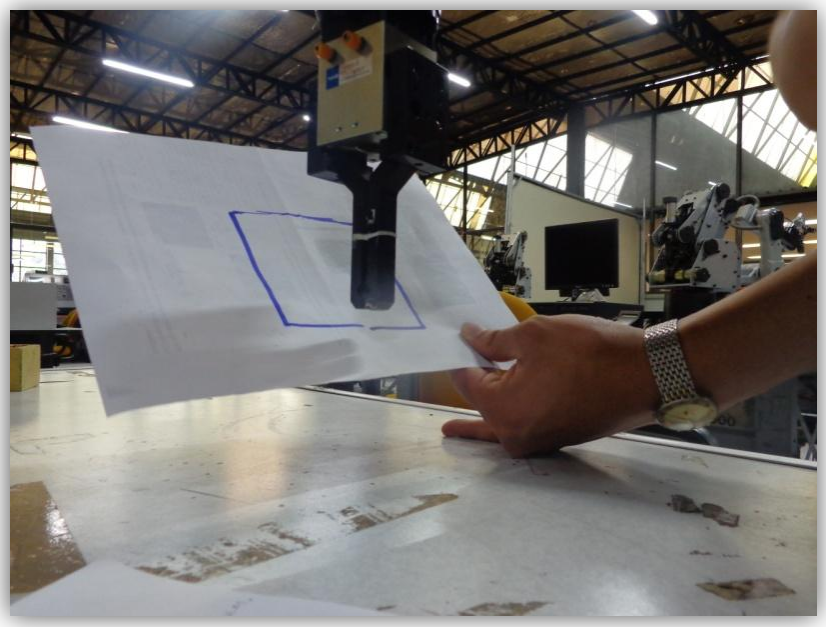


Imagen 3.2.15 Robot Scora-ER 14Pro, dibujo en el Plano XY con rotación en el eje X de 45°.

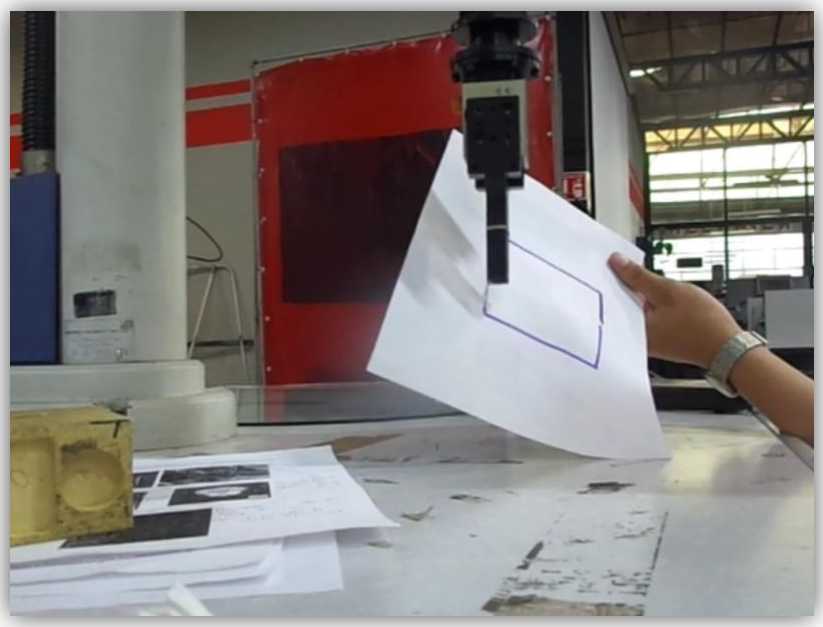


Imagen 3.2.16 Robot Scora-ER 14Pro, dibujo en el Plano XY con rotaciones en X y Y de 45°.

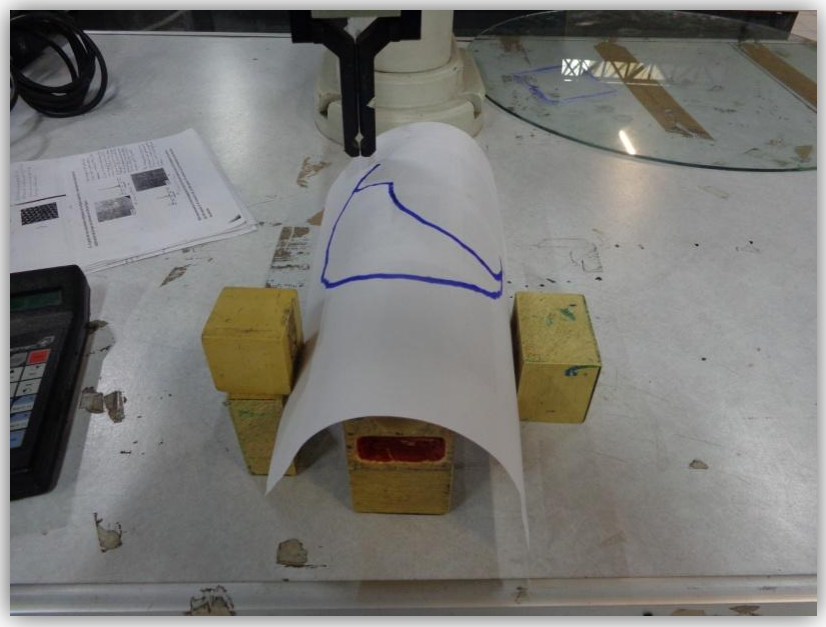


Imagen 3.2.17 Robot Scora-ER 14Pro, dibujo en el Plano XY de parábola (negativa).

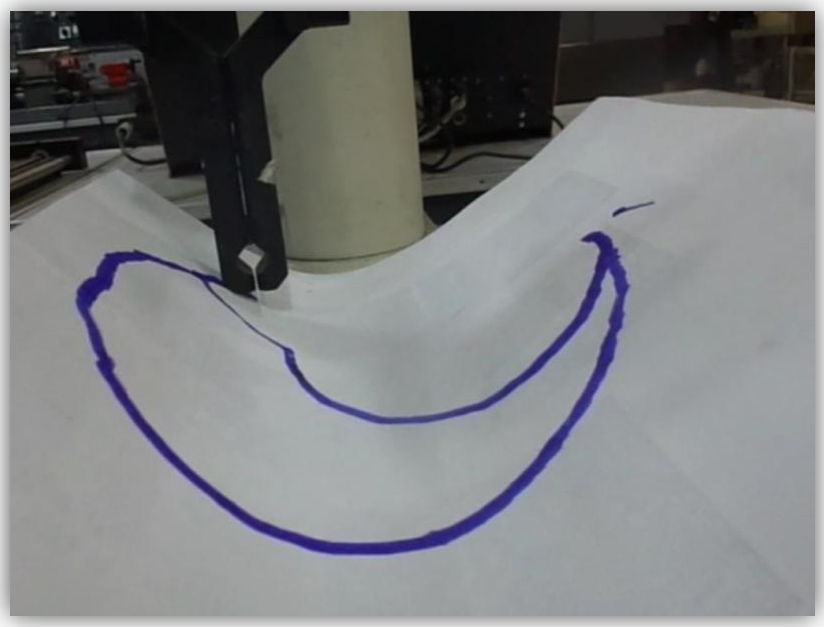


Imagen 3.2.18 Robot Scora-ER 14Pro, dibujo en el Plano XY de parábola (positiva).

Conclusiones y Trabajo Futuro.

Para finalizar este trabajo de tesis, este apartado se dedicará a mostrar las conclusiones y recomendaciones de posibles aplicaciones prácticas y de desarrollo como trabajo a futuro, además de mostrar una perspectiva de los beneficios obtenidos.

Conclusiones.

El objetivo de esta tesis fue la búsqueda de rehabilitación de tres configuraciones diferentes de robots en el laboratorio de manufactura avanzada a través de la implementación de una interfaz generando trayectorias con el uso de imágenes vectorizadas, colocando a esté laboratorio a la vanguardia, proponiendo una aplicación que ayuda a mejorar y economizar recursos, pero sobre todo permitiendo el desarrollo tecnológico propio en la facultad de ingeniería como el sistema mencionado anteriormente.

Logrando con este sistema lo siguiente:

- ✓ Una interfaz grafica más amigable con conocimiento básico acerca del uso del equipo de robótica.
- ✓ Rehabilitación de equipo que se creía obsoleto.
- ✓ Ningún costo extra por el uso de paquetería de software.
- ✓ Ampliación del enfoque acerca del uso del equipo de robótica.
- ✓ Integraciones entre software y hardware.

Un aspecto importante a resaltar en este trabajo es el funcionamiento del equipo de robótica, en el que además se colaboró con la elaboración de prácticas para el laboratorio, sin embargo, el tipo de control que implementa el fabricante, errores de redondeo, el tipo de transmisión y la antigüedad del equipo, propician las condiciones necesarias para que exista la falta de factores como: la resolución, la precisión, la exactitud y los movimientos entre cortados de un punto a otro, por mencionar algunos.

La restricción por parte del fabricante para hacer modificaciones en los parámetros del controlador o de sus equipos, crea las condiciones necesarias para investigar y desarrollar en un mundo globalizado por la tecnología, y así mejorar las aplicaciones y la rehabilitación del equipo de robótica.

Respecto la interfaz, ¿Cómo se puede estar seguro de lo que se ha hecho es lo correcto? Realmente los usuarios son los que controlan el prototipo mediante pruebas, y la respuesta obtenida se utiliza para la siguiente modificación iterativa del prototipo. En este trabajo además se busca, indirectamente, sentar las bases para seguir con modificaciones iterativas acerca de la interfaz.

Por otro lado, el manejo en la generación de coordenadas, en un principio se trato de realizar con cinemática directa solo para el Robot Scora-ER 14Pro, escribiendo un programa que encontraba los puntos de la imagen SVG y los convertía en coordenadas cartesianas, con esa información, y la longitud de los eslabones del robot, se obtenía el valor de los ángulos para cada punto del lugar geométrico de trayectoria, posteriormente el valor numérico de cada ángulo se convertían a pulsos de encoder para que el controlador leyera ese información y así lograr el movimiento en el PCH del robot. Al ser un sistema cerrado y la modificación en los parámetros de control, marcaban el camino a seguir buscando más información para proponer una solución, pero a su vez la poca información existente cerraba esas posibilidades. Sin embargo finalmente se logro aplicando otra estrategia y lo anterior sirvió en parte para demostrar el funcionamiento de ecuaciones y abrir un camino más de investigaciones posteriores.

Una vez logrado el objetivo de este trabajo, la planificación en los movimientos del robot son restringidos a una determinada área, esto se debe a que este trabajo presenta grandes posibilidades de desarrollar nuevas técnicas que puedan ser implementadas. Un ejemplo de ello es el nombre CADR.CBU que queda restringido por las características del controlador, pero esa opción puede ser depurada con una programación más robusta en la que exista una mayor interacción entre usuario e interfaz.

Trabajo a Futuro.

Como se ha mencionado, esté trabajo propicia la investigación y desarrollo en el campo de la robótica didáctica y robótica industrial, especialmente para la Facultad de Ingeniería. Ejemplo de ello generar trayectorias con el uso de geometrías para usos didácticos o industriales. Desarrollo profundo de una interfaz creando escenarios de aplicaciones para los usuarios, como una implementación de varios procesos, por ejemplo corte por plasma, corte por laser o procesos de aplicación como pintura, pegamento, etc. También se puede extender esta propuesta al desarrollo de aplicaciones móviles, y lograr manipular el robot a distancia y de una forma más simplificada trasladando toda la información existente a un entorno más fácil de uso y comprensión.

Anexo 1

Planos de Distribución y Levantamiento.

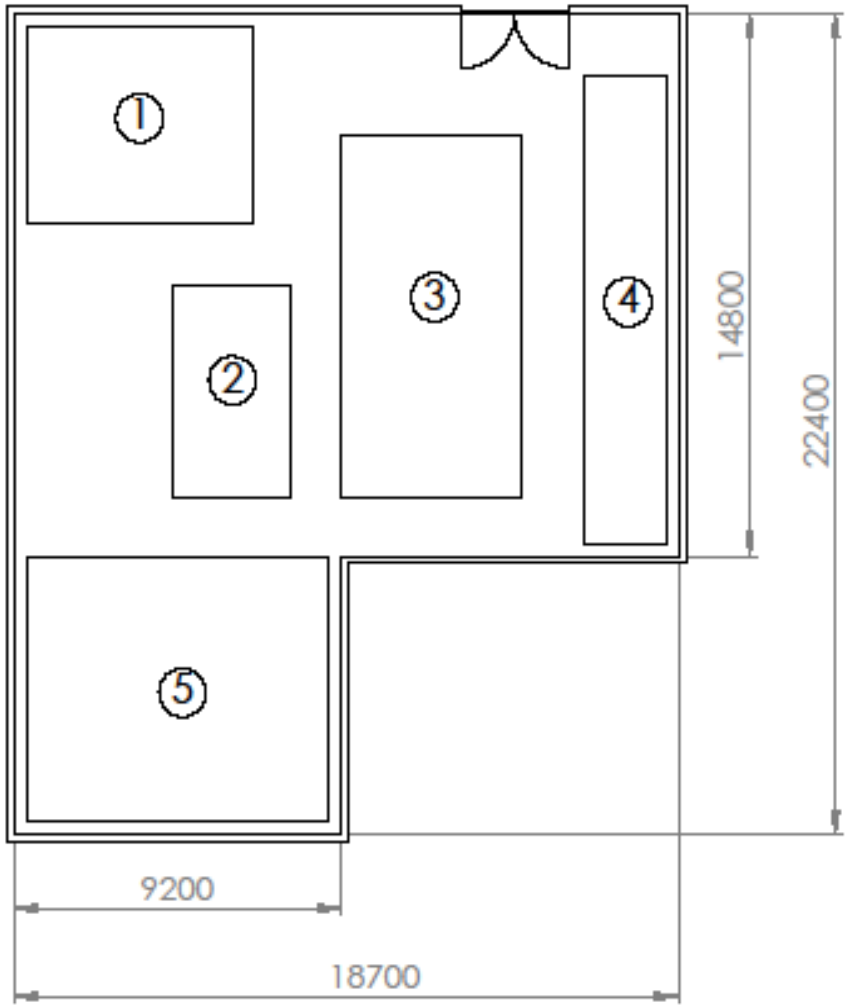
① Área de Robótica Industrial

② Área de Robótica Didáctica

③ Área de Entrenamiento de Control Numérico

④ Área de Maquinas C.N.C Industriales

⑤ Área del Centro de Manufactura Integrada por Computadora



N

Facultad de Ingeniería, UNAM

	NOMBRE	FECHA
DIBUJ.	Jhovvany Hernández	22/01/2014
APROB.	M.I. Humberto Mandilo	23/01/2014
FABR.	Jhovvany Hernández	22/01/2014

TÍTULO:

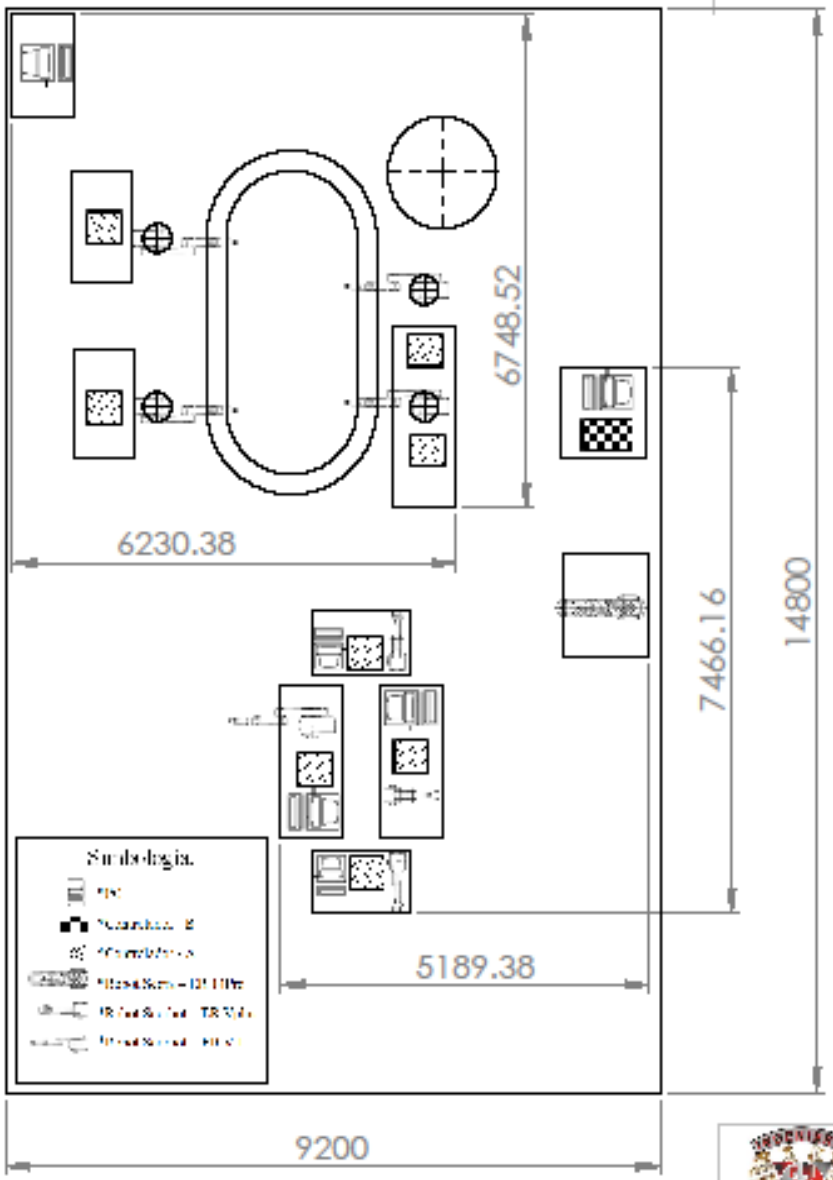
Interfaz gráfica generadora de trayectorias para operar robots pick and place a través de imágenes vectorizadas.

NOMBRE DEL DIBUJO

Distribución de áreas de trabajo en el Laboratorio de Manufactura Avanzada.

A4

ESCALA:1:200 | UNIDADES: mm | HOJA 1 DE 1



N

Facultad de Ingeniería, UNAM

	NOMBRE	FECHA
DIBUJ.	Jhovvany Hernández	22/01/2014
APROB.	M.I. Humberto Mandila	23/01/2014
FABR.	Jhovvany Hernández	22/01/2014

TÍTULO:

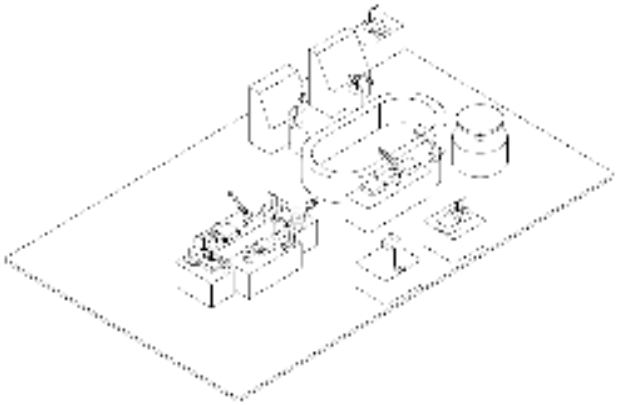
Interfaz gráfica generadora de trayectorias para operar robots pick and place a través de imágenes vectorizadas.

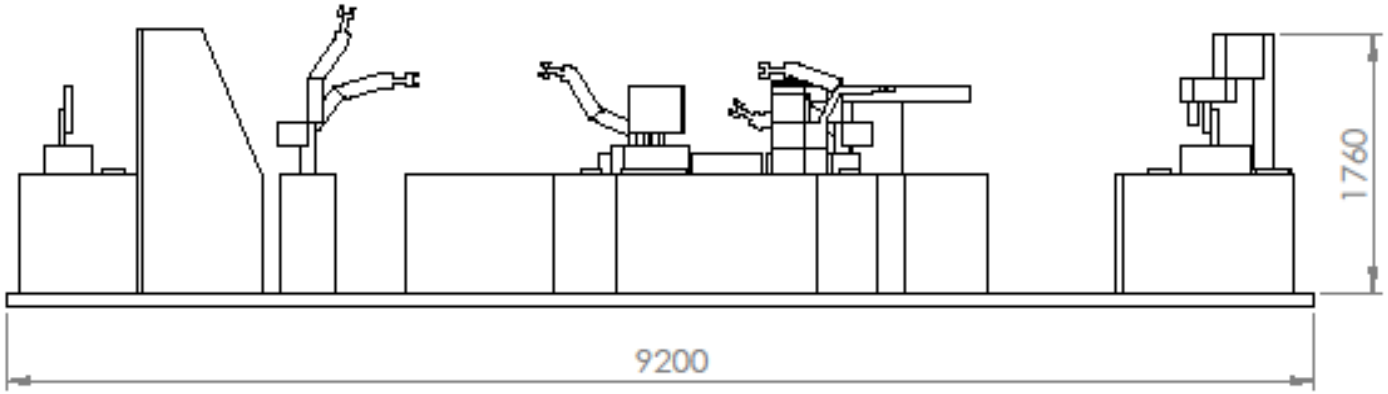
NOMBRE DEL DIBUJO

Plano de Levantamiento.

A4

ESCALA:1:100 | UNIDADES: mm | HOJA 1 DE 1





Facultad de Ingeniería, UNAM

	NOMBRE	FECHA
DIBUJ.	Jhovvany Hernández	22/01/2014
APROB.	M.I. Humberto Mandilla	23/01/2014
FABR.	Jhovvany Hernández	22/01/2014

TÍTULO:

Interfaz gráfica generadora de trayectorias para operar robots pick and place a través de imágenes vectorizadas.

NOMBRE DEL DIBUJO

Plano de Levantamiento (Vista Panorámica de las Áreas de Robótica).

A4

ESCALA:1:50 | UNIDADES: mm | HOJA 1 DE 1

Anexo 2

Guías de Usuario y Manuales de Operación.

SCORBOT-ER Vplus

Manual de Usuario

N° Cat. 100265 Rev.A

ESHED ROBOTEC

SCORBOT-ER VII

User's Manual

2nd Edition

Catalog # 100017 Rev.C

ESHED ROBOTEC

SCORA-ER 14Pro

User Manual

Catalog # 200035 Rev. B

intelitek

Controlador-B

Versión de ACL 2.28 o posterior

Manual de Usuario

Cat. # 100262 rev.01

ESHED ROBOTEC

ACL

Lenguaje de Control Avanzado

Version 1.43, F.44

Guia de Referencia

para Controlador-A

Catálogo #100266 Rev.01

ESHED ROBOTEC

ATS

Software Terminal Avanzado

Version 1.44

Guia e Referencia

para Controlador-A

Catálogo #100267 Rev.01

ESHED ROBOTEC

Anexo 3

Código fuente de la interfaz CADR.

```
//***** Librerías *****
import controlP5.*;       //Controla botones
import geomerative.*;     //Controla el formato de imágenes SVG

//***** Variables de Librerías *****
ControlP5 cp5, cp5g2, cp5g3, cp5g4;     //Variables de control para los grupos del acordeón
RShape shp;                      //Variable que controla el dibujo de las imágenes SVG
RShape shp2;                      //Variable que controla el dibujo de las imágenes SVG
RPoint[] ps;                     //Variable que determina el rango de coordenadas

//***** Variables de Accordion *****
Accordion accordion;     //Variable para cargar los grupos de los menús al acordeón

//***** Declaración del Objeto para guardar los puntos *****
float [][] dataRS;       //Variable de control para manejar los datos en coordenadas

//***** Variables globales para la rango de dimensión del dibujo real = Slider *****
int RDmCBSER14 = 0;                 //Rango de Dimensión mínima para Controlador-B (Scara-ER 14)
int RDMCBSER14 = 13;                 //Rango de Dimensión Máxima para Controlador-B (Scara-ER 14) ---> Utilizarlo como base (Sobre este valor se sumara o
restara el rango de dimensión de alcance de los otros robots)
int RDmCASERVP = RDmCBSER14 + 0;       //Rango de Dimensión mínima para Controlador-A (Scorbot-ER V+)
int RDMCASERVP = RDMCBSER14 + 10;      //Rango de Dimensión Máxima para Controlador-A (Scorbot-ER V+)
int RDmCASERVII = RDmCBSER14 + 0;      //Rango de Dimensión mínima para Controlador-A (Scorbot-ER VII)
int RDMCASERVII = RDMCBSER14 + 20;     //Rango de Dimensión Máxima para Controlador-A (Scorbot-ER VII)

//***** Variables globales para la rango de rotación de los ejes del dibujo real = Slider *****
int Rotm = -90;     //Rango de Rotación mínima de los ejes para el dibujo real
int RotM = 90;      //Rango de Rotación Máxima de los ejes para el dibujo real

//***** Variables globales para generar los archivos CBU *****
float dimension = 10;     //Dimensión del dibujo
float factdim;           //Factor de dimensión
float CU;                //Conversión de unidades (Pixeles vs. cm)

//***** Variable de factor de escala para dibujar en el Sketch *****
float s1= 1;     //Escala inicial al cargar la imagen

//***** Variables para el dibujo de los ejes *****
float angX = radians(-50);     //Eje X
float angY = radians(-50);     //Eje Y
float angZ = radians(-20);     //Eje Z
float ang1 = radians(90);      //Plano XZ
float ang2 = radians(90);      //Plano XY

//***** Variable de control para el color de las figuras de fondo (Ambiente de la interfaz) *****
color c = color(150,38,255,25);     //Asignación de color por default de los Círculos

//***** Variables de control del color de los planos (Ambiente de la interfaz) *****
color cXY = color(255,0,50,80);        //Asignación de color por default del plano XY
color cXZ = color(105,167,12,120);     //Asignación de color por default del plano XZ
color cYZ = color(0,0,255,100);        //Asignación de color por default del plano YZ

//***** Variables Booleanas *****
boolean ignoringStyles = false;     //Funciona solo con el movimiento del mouse

//***** Variables para generar lo archivos *****
int PPx, PPy, PPz, G, P, R;     //Variables que corresponden a los ejes

//***** Variables para los puntos home de los robots *****
int PHX, PHY, PHZ, PHP, PHR;     //Variables que corresponden al home del robot

//***** Variables de la rotación para manejar números enteros *****
public int RXCASERVP, RYCASERVP, RZCASERVP, DRXCASERVP, DRYCASERVP;         //Variables de CCASERVP
public int RXCASERVII, RYCASERVII, RZCASERVII, DRXCASERVII, DRYCASERVII;     //Variables de CCASERVII
public int RXCBSER14, RYCBSER14, RZCBSER14, DRXCBSER14, DRYCBSER14;         //Variables de CBSER14

//***** Vector para el manejo de datos *****
int [][] arrayMI;     //Arreglo para el manejo de la información

//***** Variable para cargar el signo de la parábola *****
float SigP;     //Se utiliza para el manejo del signo de la parábola

//***** Programa *****
void setup(){
```

```
 size(1000, 650,P3D);     //Tamaño de ventana
 frameRate(40);          //Configuración de los frames por segundo
 noStroke();             //Desactiva el dibujo de contorno
 smooth();               //Activa el suavizado de imágenes
 gui();                 //Llamado a subrutina

 //Se inicializa la biblioteca antes de usarla
 RG.init(this);                    //Subprograma lectura archivo SVG *
 RG.ignoreStyles(ignoringStyles);     //Subprograma lectura archivo SVG *
}

//***** Subrutina de Controles *****
void gui(){                   //Asignación de los grupos para las funciones de ControlP5
 cp5 = new ControlP5(this);
 cp5g2= new ControlP5(this);
 cp5g3= new ControlP5(this);
 cp5g4= new ControlP5(this);

 // Grupo 1, Abrir archivo
 Group g1 = cp5.addGroup("g1")                        //Agregar Grupo
          .setColorBackground(color(43, 87, 203))       //Color del letrero
          .setColorForeground (color(105, 167, 12))     //Color cuando esta el mouse
          .setHeight(30)                          //Largo del título en Y
          .setBackgroundColor(color(0,100))           //Color del resto del acordeón
          .setBackgroundHeight(100)                  //Largo del MENU en Y por default 100
          ;
    // Botón 1, Abrir
    cp5.addButton("Abrir")                      //Agregar Botón
      .setPosition(43,25)                     //Posición inicial del Cuadrito
      .setSize(193,40)                        //Largo X y ancho Y del botón
      .setColorBackground(color(43, 87, 203))       //Color del botón
      .setColorForeground (color(105, 167, 12))     //Color cuando esta el mouse
      .moveTo(g1)                             //Mover el botón al Grupo 1
      .plugTo(this,"file")                     //Cargar el evento de la subrutina file
      ;

 // Grupo 2, Controlador-A (Scorbot-ER V+)
 Group g2 = cp5.addGroup("g2")                        //Agregar Grupo
          .setColorBackground(color(43, 87, 203))       //Color del letrero
          .setColorForeground (color(105, 167, 12))     //Color cuando esta el mouse
          .setHeight(30)                          //Largo del título en Y
          .setBackgroundColor(color(0,100))           //Color del resto del acordeón
          .setBackgroundHeight(375)                  //Largo del MENU en Y por default 100
          ;
    // Dimensión [cm]
    cp5.addSlider("DCASERVP")              //Nombre del Slider
      .setPosition(20,10)                 //Posición inicial del Cuadrito
      .setSize(240,20)                    //Largo X y ancho Y del Slider
      .setRange(RDmCASERVP,RDMCASERVP)     //Rango de Dimensión MODIFICAR
      .moveTo(g2)                         //Mover el botón al Grupo 2
      .setValue(10)                      //Es desde donde comenzara el valor del Slider
      .plugTo(this,"DDCASERVP")           //Llamado de subrutina
      ;
    // Rotación X [grad]
    cp5.addSlider("RXCASERVP")     //Nombre del Slider
      .setPosition(93,140)        //Posición inicial del Cuadrito
      .setSize(167,13)            //Largo X y ancho Y del Slider
      .setRange(Rotm,RotM)        //Rango de Dimensión MODIFICAR
      .moveTo(g2)                 //Mover el botón al Grupo 2
      ;
    // Rotación Y [grad]
    cp5.addSlider("RYCASERVP")     //Nombre del Slider
      .setPosition(93,180)        //Posición inicial del Cuadrito
      .setSize(167,13)            //Largo X y ancho Y del Slider
      .setRange(Rotm,RotM)        //Rango de Dimensión MODIFICAR
      .moveTo(g2)                 //Mover el botón al Grupo 2
      ;
    // Rotación Z [grad]
    cp5.addSlider("RZCASERVP")     //Nombre del Slider
      .setPosition(93,220)        //Posición inicial del Cuadrito
      .setSize(167,13)            //Largo X y ancho Y del Slider
      .setRange(Rotm,RotM)        //Rango de Dimensión MODIFICAR
      .moveTo(g2)                 //Mover el botón al Grupo 2
      ;
```

```
    // Doble Rotación X [grad]
    cp5.addSlider("DRXCASERVP")     //Nombre del Slider
      .setPosition(93,257)        //Posición inicial del Cuadrito
      .setSize(167,11)            //Largo X y ancho Y del Slider
      .setRange(Rotm,RotM)        //Rango de Dimensión MODIFICAR
      .moveTo(g2)                 //Mover el botón al Grupo 2
      ;
    // Doble Rotación Y [grad]
    cp5.addSlider("DRYCASERVP")     //Nombre del Slider
      .setPosition(93,269)        //Posición inicial del Cuadrito
      .setSize(167,11)            //Largo X y ancho Y del Slider
      .setRange(Rotm,RotM)        //Rango de Dimensión MODIFICAR
      .moveTo(g2)                 //Mover el botón al Grupo 2
      ;
    // Menú de Planos
    cp5g2.addRadioButton("CASERVP")     //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
       .setPosition(10,40)          //Posición inicial de la cadena de cuadritos
       .setItemWidth(20)            //Ancho de los cuadritos
       .setItemHeight(20)           //Largo de los cuadritos
       .addItem("Plano XY", 0)      //Asignación de Nombre y Valor del caso
       .addItem("Plano YZ", 1)
       .addItem("Plano XY Rotación Eje X", 2)
       .addItem("Plano XY Rotación Eje Y", 3)
       .addItem("Plano XY Rotación Eje Z", 4)
       .addItem("Plano XY 2 Rotaciones", 5)
       .addItem("Cilindro Parabólico Plano XY", 6)
       .addItem("Cilindro Parabólico Plano YZ", 7)
       .setColorLabel(color(255))    //Color de letra de los .addItem
       .activate(3)                 //Que icono quiero que este activado inicialmente
       .moveTo(g2)                  //Mover el botón al Grupo 2
       .toUpperCase(false)          //Cambia mayúsculas false
       .setSpacingRow(20)           //Espacio entre cuadritos
       ;
    cp5g2.addRadioButton("POSCASERVP")     //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
       .setPosition(80,300)         //Posición inicial de la cadena de cuadritos
       .setItemWidth(20)            //Ancho de los cuadritos
       .setItemHeight(20)           //Largo de los cuadritos
       .addItem("+ XY", 0)          //Asignación de Nombre y Valor del caso
       .addItem("+ YZ", 1)
       .setColorLabel(color(255))       //Color de letra de los .addItem
       .moveTo(g2)                  //Mover el botón al Grupo 2
       .toUpperCase(false)          //Cambia mayúsculas false
       .setSpacingRow(20)           //Espacio entre cuadritos
       ;
    cp5g2.addRadioButton("NEGCASERVP")     //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
       .setPosition(150,300)        //Posición inicial de la cadena de cuadritos
       .setItemWidth(20)            //Ancho de los cuadritos
       .setItemHeight(20)           //Largo de los cuadritos
       .addItem("- XY", 0)          //Asignación de Nombre y Valor del caso
       .addItem("- YZ", 1)
       .setColorLabel(color(255))       //Color de letra de los .addItem
       .moveTo(g2)                  //Mover el botón al Grupo 2
       .toUpperCase(false)          //Cambia mayúsculas false
       .setSpacingRow(20)           //Espacio entre cuadritos
       ;

// Grupo 3, Controlador-A (Scorbot-ER VII)
Group g3 = cp5.addGroup("g3")                   //Agregar Grupo
         .setColorBackground(color(43, 87, 203))       //Color del letrero
         .setColorForeground (color(105, 167, 12))     //Color cuando esta el mouse
         .setHeight(30)                        //Largo del título en Y
         .setBackgroundColor(color(0,100))           //Color del resto del acordeón
         .setBackgroundHeight(455)                 //Largo del MENU en Y por default 100
         ;
    // Dimensión [cm]
    cp5.addSlider("DCASERVII")              //Nombre del Slider
      .setPosition(20,10)               //Posición inicial del Cuadrito
      .setSize(240,20)                  //Largo X y ancho Y del Slider
      .setRange(RDmCASERVII,RDMCASERVII)     //Rango de Dimensión MODIFICAR
      .moveTo(g3)                       //Mover el botón al Grupo 3
      .setValue(10)                     //Es desde donde comenzara el valor del Slider
      .plugTo(this,"DDCASERVII")           //Llamado de subrutina
      ;
    // Rotación X [grad]
```

```
cp5.addSlider("RXCASERVII")     //Nombre del Slider
  .setPosition(93,180)         //Posición inicial del Cuadrito
  .setSize(167,13)            //Largo X y ancho Y del Slider
  .setRange(Rotm,RotM)         //Rango de Dimensión MODIFICAR
  .moveTo(g3)                 //Mover el botón al Grupo 3
  ;
// Rotación Y [grad]
cp5.addSlider("RYCASERVII")     //Nombre del Slider
  .setPosition(93,220)         //Posición inicial del Cuadrito
  .setSize(167,13)            //Largo X y ancho Y del Slider
  .setRange(Rotm,RotM)         //Rango de Dimensión MODIFICAR
  .moveTo(g3)                 //Mover el botón al Grupo 3
  ;
// Rotación Z [grad]
cp5.addSlider("RZCASERVII")     //Nombre del Slider
  .setPosition(93,260)         //Posición inicial del Cuadrito
  .setSize(167,13)            //Largo X y ancho Y del Slider
  .setRange(Rotm,RotM)         //Rango de Dimensión MODIFICAR
  .moveTo(g3)                 //Mover el botón al Grupo 3
  ;
// Doble Rotación X [grad]
cp5.addSlider("DRXCASERVII")     //Nombre del Slider
  .setPosition(93,297)          //Posición inicial del Cuadrito
  .setSize(167,11)             //Largo X y ancho Y del Slider
  .setRange(Rotm,RotM)          //Rango de Dimensión MODIFICAR
  .moveTo(g3)                 //Mover el botón al Grupo 3
  ;
// Doble Rotación Y [grad]
cp5.addSlider("DRYCASERVII")     //Nombre del Slider
  .setPosition(93,309)          //Posición inicial del Cuadrito
  .setSize(167,11)             //Largo X y ancho Y del Slider
  .setRange(Rotm,RotM)          //Rango de Dimensión MODIFICAR
  .moveTo(g3)                 //Mover el botón al Grupo 3
  ;
// Menú de Planos
cp5g3.addRadioButton("CASERVII")     //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
   .setPosition(10,40)             //Posición inicial de la cadena de cuadritos
   .setItemWidth(20)               //Ancho de los cuadritos
   .setItemHeight(20)              //Largo de los cuadritos
   .addItem("Plano XY", 0)         //Asignación de Nombre y Valor del caso
   .addItem("Plano XZ", 1)
   .addItem("Plano YZ", 2)
   .addItem("Plano XY Rotación Eje X", 3)
   .addItem("Plano XY Rotación Eje Y", 4)
   .addItem("Plano XY Rotación Eje Z", 5)
   .addItem("Plano XY 2 Rotaciones", 6)
   .addItem("Cilindro Parabólico Plano XY", 7)
   .addItem("Cilindro Parabólico Plano XZ", 8)
   .addItem("Cilindro Parabólico Plano YZ", 9)
   .setColorLabel(color(255))      //Color de letra de los .addItem
   .activate(3)                    //Que icono quiero que este activado inicialmente
   .moveTo(g3)                     //Mover el botón al Grupo 3
   .toUpperCase(false)             //Cambia mayúsculas false
   .setSpacingRow(20)              //Espacio entre cuadritos
   ;
cp5g3.addRadioButton("POSCASERVII")     //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
   .setPosition(80,340)              //Posición inicial de la cadena de cuadritos
   .setItemWidth(20)                 //Ancho de los cuadritos
   .setItemHeight(20)                //Largo de los cuadritos
   .addItem("+ XY", 0)               //Asignación de Nombre y Valor del caso
   .addItem("+ XZ", 1)
   .addItem("+ YZ", 2)
   .setColorLabel(color(255))        //Color de letra de los .addItem
   .moveTo(g3)                      //Mover el botón al Grupo 2
   .toUpperCase(false)               //Cambia mayúsculas false
   .setSpacingRow(20)                //Espacio entre cuadritos
   ;
cp5g3.addRadioButton("NEGCASERVII")     //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
   .setPosition(150,340)             //Posición inicial de la cadena de cuadritos
   .setItemWidth(20)                 //Ancho de los cuadritos
   .setItemHeight(20)                //Largo de los cuadritos
   .addItem("- XY", 0)               //Asignación de Nombre y Valor del caso
   .addItem("- XZ", 1)
   .addItem("- YZ", 2)
```

```
      .setColorLabel(color(255))        //Color de letra de los .addItem
      .moveTo(g3)                       //Mover el botón al Grupo 2
      .toUpperCase(false)               //Cambia mayúsculas false
      .setSpacingRow(20)                //Espacio entre cuadritos
      ;

// Grupo 4, Controlador-B (Scara-ER 14)
Group g4 = cp5.addGroup("g4")                          //Agregar Grupo
         .setColorBackground(color(43, 87, 203))       //Color del letrero
         .setColorForeground (color(105, 167, 12))     //Color cuando esta el mouse
         .setHeight(30)                                //Largo del título en Y
         .setBackgroundColor(color(0,100))             //Color del resto del acordeón
         .setBackgroundHeight(455)                     //Largo del MENU en Y por default 100
         ;
   // Dimensión [cm]
   cp5.addSlider("DCBSER14")              //Nombre del Slider
     .setPosition(20,10)                  //Posición inicial del Cuadrito
     .setSize(240,20)                     //Largo X y ancho Y del Slider
     .setRange(RDmCBSER14,RDMCBSER14)     //Rango de Dimensión MODIFICAR
     .moveTo(g4)                          //Mover el botón al Grupo 4
     .setValue(10)                        //Es desde donde comenzara el valor del Slider
     .plugTo(this,"DDCBSER14")            //Llamado de subrutina
     ;
   // Rotación X [grad]
   cp5.addSlider("RXCBSER14")     //Nombre del Slider
     .setPosition(93,180)         //Posición inicial del Cuadrito
     .setSize(167,13)             //Largo X y ancho Y del Slider
     .setRange(Rotm,RotM)         //Rango de Dimensión MODIFICAR
     .moveTo(g4)                  //Mover el botón al Grupo 4
     ;
   // Rotación Y [grad]
   cp5.addSlider("RYCBSER14")     //Nombre del Slider
     .setPosition(93,220)         //Posición inicial del Cuadrito
     .setSize(167,13)             //Largo X y ancho Y del Slider
     .setRange(Rotm,RotM)         //Rango de Dimensión MODIFICAR
     .moveTo(g4)                  //Mover el botón al Grupo 4
     ;
   // Rotación Z [grad]
   cp5.addSlider("RZCBSER14")     //Nombre del Slider
     .setPosition(93,260)         //Posición inicial del Cuadrito
     .setSize(167,13)             //Largo X y ancho Y del Slider
     .setRange(Rotm,RotM)         //Rango de Dimensión MODIFICAR
     .moveTo(g4)                  //Mover el botón al Grupo 4
     ;
   // Doble Rotación X [grad]
   cp5.addSlider("DRXCBSER14")    //Nombre del Slider
     .setPosition(93,297)         //Posición inicial del Cuadrito
     .setSize(167,11)             //Largo X y ancho Y del Slider
     .setRange(Rotm,RotM)         //Rango de Dimensión MODIFICAR
     .moveTo(g4)                  //Mover el botón al Grupo 4
     ;
   // Doble Rotación Y [grad]
   cp5.addSlider("DRYCBSER14")    //Nombre del Slider
     .setPosition(93,309)         //Posición inicial del Cuadrito
     .setSize(167,11)             //Largo X y ancho Y del Slider
     .setRange(Rotm,RotM)         //Rango de Dimensión MODIFICAR
     .moveTo(g4)                  //Mover el botón al Grupo 4
     ;
   // Menú de Planos
   cp5g4.addRadioButton("CBSER14")    //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
      .setPosition(10,40)             //Posición inicial de la cadena de cuadritos
      .setItemWidth(20)               //Ancho de los cuadritos
      .setItemHeight(20)              //Largo de los cuadritos
      .addItem("Plano XY", 0)         //Asignación de Nombre y Valor del caso
      .addItem("Plano XZ", 1)
      .addItem("Plano YZ", 2)
      .addItem("Plano XY Rotación Eje X", 3)
      .addItem("Plano XY Rotación Eje Y", 4)
      .addItem("Plano XY Rotación Eje Z", 5)
      .addItem("Plano XY 2 Rotaciones", 6)
      .addItem("Cilindro Parabólico Plano XY", 7)
      .addItem("Cilindro Parabólico Plano XZ", 8)
      .addItem("Cilindro Parabólico Plano YZ", 9)
      .setColorLabel(color(255))     //Color de letra de los .addItem
```

```
      .activate(3)               //Que icono quiero que este activado inicialmente
      .moveTo(g4)                //Mover el botón al Grupo 4
      .toUpperCase(false)        //Cambia mayúsculas false
      .setSpacingRow(20)         //Espacio entre cuadritos
      ;
    cp5g4.addRadioButton("POSCBSER14")    //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
      .setPosition(80,340)         //Posición inicial de la cadena de cuadritos
      .setItemWidth(20)            //Ancho de los cuadritos
      .setItemHeight(20)           //Largo de los cuadritos
      .addItem("+ XY", 0)          //Asignación de Nombre y Valor del caso
      .addItem("+ XZ", 1)
      .addItem("+ YZ", 2)
      .setColorLabel(color(255))       //Color de letra de los .addItem
      .moveTo(g4)                //Mover el botón al Grupo 2
      .toUpperCase(false)          //Cambia mayúsculas false
      .setSpacingRow(20)           //Espacio entre cuadritos
      ;
    cp5g4.addRadioButton("NEGCBSER14")    //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
      .setPosition(150,340)        //Posición inicial de la cadena de cuadritos
      .setItemWidth(20)            //Ancho de los cuadritos
      .setItemHeight(20)           //Largo de los cuadritos
      .addItem("- XY", 0)          //Asignación de Nombre y Valor del caso
      .addItem("- XZ", 1)
      .addItem("- YZ", 2)
      .setColorLabel(color(255))       //Color de letra de los .addItem
      .moveTo(g4)                //Mover el botón al Grupo 2
      .toUpperCase(false)          //Cambia mayúsculas false
      .setSpacingRow(20)           //Espacio entre cuadritos
      ;

// Grupo 5, Acerca de Interfaz CADR
Group g5 = cp5.addGroup("g5")                  //Agregar Grupo
        .setColorBackground(color(43, 87, 203))      //Color del letrero
        .setColorForeground (color(105, 167, 12))    //Color cuando esta el mouse
        .setHeight(30)                 //Largo del título en Y
        .setBackgroundColor(color(0,100))        //Color del resto del acordeón
        .setBackgroundHeight(250)              //Largo del MENU en Y por default 100
        ;
    // Información
    cp5.addRadioButton("INFO1")        //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
      .setPosition(10,20)          //Posición inicial de la cadena de cuadritos
      .setItemWidth(20)            //Ancho de los cuadritos
      .setItemHeight(20)           //Largo de los cuadritos
      .addItem("Interfaz CADR.", 0)    //Asignación de Nombre y Valor del caso
      .addItem("Versión 1.0", 1)
      .addItem("14 - 01 - 2014", 2)
      .setColorLabel(color(255))       //Color de letra de los .addItem
      .moveTo(g5)                //Mover el botón al Grupo 4
      .toUpperCase(false)          //Cambia mayúsculas false
      .setSpacingRow(3)            //Espacio entre cuadritos
      ;
    cp5.addRadioButton("INFO2")        //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
      .setPosition(10,109)         //Posición inicial de la cadena de cuadritos
      .setItemWidth(20)            //Ancho de los cuadritos
      .setItemHeight(20)           //Largo de los cuadritos
      .addItem("UNAM reservados todos los derechos.", 0)
      .setColorLabel(color(255))       //Color de letra de los .addItem
      .moveTo(g5)                //Mover el botón al Grupo 4
      .toUpperCase(false)          //Cambia mayúsculas false
      ;
    cp5.addRadioButton("INFO3")        //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
      .setPosition(10,155)         //Posición inicial de la cadena de cuadritos
      .setItemWidth(20)            //Ancho de los cuadritos
      .setItemHeight(20)           //Largo de los cuadritos
      .addItem("Elaboró:", 0)
      .addItem("Jhovvany Hazzael Hernández Clemente", 1)
      .setColorLabel(color(255))       //Color de letra de los .addItem
      .moveTo(g5)                //Mover el botón al Grupo 4
      .toUpperCase(false)          //Cambia mayúsculas false
      .setSpacingRow(3)            //Espacio entre cuadritos
      ;
    cp5.addRadioButton("INFO4")        //Nombre de la subrutina para los botones Controlador-A (Scorbot-ER V+)
      .setPosition(10,221)         //Posición inicial de la cadena de cuadritos
      .setItemWidth(20)            //Ancho de los cuadritos
```

```
        .setItemHeight(20)              //Largo de los cuadritos
        .addItem("Revisó: M.I. Humberto Mancilla Alonso", 0)
        .setColorLabel(color(255))      //Color de letra de los .addItem
        .moveTo(g5)                     //Mover el botón al Grupo 4
        .toUpperCase(false)             //Cambia mayúsculas false
        ;

//***** Propiedad de Letra *****
PFont pfont = createFont("Eras Bold ITC",20,true);     //Creando un pfont con las características Arial 20 y se usa true para tener un smooth en la letra
ControlFont font = new ControlFont(pfont,241);         //ControlFont es un contenedor para una PFont se utilizara para personalizar la fuente de una etiqueta

// Grupo 1, Abrir archivo
 g1.captionLabel()              //Indica que se modificara texto en dicho grupo
   .align(CENTER,CENTER)        //Alineación del texto
   .toUpperCase(true)           //Cambia mayúsculas true
   .set("Abrir Archivo SVG")    //Carga el siguiente nombre
   .setColor(color(255))        //Color de letra
   .setFont(font)               //Propiedad de letra
   .setSize(14)                 //Tamaño de letra
   ;
 // Botón Grupo 1
 cp5.getController("Abrir")     //Indica que se modificara texto en el controlador
    .getCaptionLabel()          //Dar propiedad de texto
    .align(CENTER,CENTER)       //Alineación del texto
    .toUpperCase(false)         //Cambia mayúsculas false
    .set("Buscar")              //Carga el siguiente nombre
    .setColor(color(255))       //Color de letra
    .setFont(font)              //Propiedad de letra
    .setSize(13)                //Tamaño de letra
    ;

// Grupo 2, Controlador-A (Scorbot-ER V+)
 g2.captionLabel()                      //Indica que se modificara texto en dicho grupo
   .align(CENTER,CENTER)                //Alineación del texto
   .toUpperCase(true)                   //Cambia mayúsculas true
   .set("Controlador-A (Scorbot-ER V+)")    //Carga el siguiente nombre
   .setColor(color(255))                //Color de letra
   .setFont(font)                       //Propiedad de letra
   .setSize(14)                         //Tamaño de letra
   ;
 //Dimensión [cm] Slider Grupo 2
 cp5.getController("DCASERVP")      //Indica que se modificara texto en el controlador
    .getCaptionLabel()              //Dar propiedad de texto
    .align(CENTER,CENTER)           //Alineación del texto
    .toUpperCase(false)             //Cambia mayúsculas false
    .set("Dimensión [cm]")          //Carga el siguiente nombre
    .setColor(color(255))           //Color de letra
    .setFont(font)                  //Propiedad de letra
    .setSize(13)                    //Tamaño de letra
    ;
 // Rotación X [grad]
 cp5.getController("RXCASERVP")     //Indica que se modificara texto en el controlador
    .getCaptionLabel()              //Dar propiedad de texto
    .align(CENTER,CENTER)           //Alineación del texto
    .toUpperCase(false)             //Cambia mayúsculas false
    .set("[grad]")                  //Carga el siguiente nombre
    .setColor(color(255))           //Color de letra
    .setFont(font)                  //Propiedad de letra
    .setSize(9)                     //Tamaño de letra
    ;
 // Rotación Y [grad]
 cp5.getController("RYCASERVP")     //Indica que se modificara texto en el controlador
    .getCaptionLabel()              //Dar propiedad de texto
    .align(CENTER,CENTER)           //Alineación del texto
    .toUpperCase(false)             //Cambia mayúsculas false
    .set("[grad]")                  //Carga el siguiente nombre
    .setColor(color(255))           //Color de letra
    .setFont(font)                  //Propiedad de letra
    .setSize(9)                     //Tamaño de letra
    ;
 // Rotación Z [grad]
 cp5.getController("RZCASERVP")     //Indica que se modificara texto en el controlador
    .getCaptionLabel()              //Dar propiedad de texto
    .align(CENTER,CENTER)           //Alineación del texto
```

```
   .toUpperCase(false)           //Cambia mayúsculas false
   .set("[grad]")               //Carga el siguiente nombre
   .setColor(color(255))         //Color de letra
   .setFont(font)               //Propiedad de letra
   .setSize(9)                  //Tamaño de letra
   ;
// Doble Rotación X [grad]
cp5.getController("DRXCASERVP")     //Indica que se modificara texto en el controlador
  .getCaptionLabel()             //Dar propiedad de texto
  .align(CENTER,CENTER)          //Alineación del texto
  .toUpperCase(false)            //Cambia mayúsculas false
  .set("Eje X [grad]")           //Carga el siguiente nombre
  .setColor(color(255))          //Color de letra
  .setFont(font)                 //Propiedad de letra
  .setSize(8)                    //Tamaño de letra
  ;
// Doble Rotación Y [grad]
cp5.getController("DRYCASERVP")     //Indica que se modificara texto en el controlador
  .getCaptionLabel()             //Dar propiedad de texto
  .align(CENTER,CENTER)          //Alineación del texto
  .toUpperCase(false)            //Cambia mayúsculas false
  .set("Eje Y [grad]")           //Carga el siguiente nombre
  .setColor(color(255))          //Color de letra
  .setFont(font)                 //Propiedad de letra
  .setSize(8)                    //Tamaño de letra
  ;
//Propiedades de letra del menú de planos G2
cp5g2.getController("Plano XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g2.getController("Plano YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g2.getController("Plano XY Rotación Eje X").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g2.getController("Plano XY Rotación Eje Y").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g2.getController("Plano XY Rotación Eje Z").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g2.getController("Plano XY 2 Rotaciones").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g2.getController("Cilindro Parabólico Plano XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);

cp5g2.getController("Cilindro Parabólico Plano YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g2.getController("+ XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g2.getController("+ YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g2.getController("- XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g2.getController("- YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);


// Grupo 3, Controlador-A (Scorbot-ER VII)
 g3.captionLabel()                  //Indica que se modificara texto en dicho grupo
  .align(CENTER,CENTER)             //Alineación del texto
  .toUpperCase(true)                //Cambia mayúsculas true
  .set("Controlador-A (Scorbot-ER VII)")     //Carga el siguiente nombre
  .setColor(color(255))             //Color de letra
  .setFont(font)                    //Propiedad de letra
  .setSize(14)                      //Tamaño de letra
  ;
 //Dimensión [cm] Slider Grupo 3
 cp5.getController("DCASERVII")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()             //Dar propiedad de texto
   .align(CENTER,CENTER)          //Alineación del texto
   .toUpperCase(false)            //Cambia mayúsculas false
   .set("Dimensión [cm]")         //Carga el siguiente nombre
   .setColor(color(255))          //Color de letra
   .setFont(font)                 //Propiedad de letra
   .setSize(13)                   //Tamaño de letra
   ;
 // Rotación X [grad]
 cp5.getController("RXCASERVII")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()             //Dar propiedad de texto
   .align(CENTER,CENTER)          //Alineación del texto
   .toUpperCase(false)            //Cambia mayúsculas false
   .set("[grad]")                 //Carga el siguiente nombre
   .setColor(color(255))          //Color de letra
   .setFont(font)                 //Propiedad de letra
   .setSize(9)                    //Tamaño de letra
   ;
 // Rotación Y [grad]
 cp5.getController("RYCASERVII")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()             //Dar propiedad de texto
```

```
   .align(CENTER,CENTER)          //Alineación del texto
   .toUpperCase(false)            //Cambia mayúsculas false
   .set("[grad]")                 //Carga el siguiente nombre
   .setColor(color(255))          //Color de letra
   .setFont(font)                 //Propiedad de letra
   .setSize(9)                    //Tamaño de letra
   ;
 // Rotación Z [grad]
 cp5.getController("RZCASERVII")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()             //Dar propiedad de texto
   .align(CENTER,CENTER)          //Alineación del texto
   .toUpperCase(false)            //Cambia mayúsculas false
   .set("[grad]")                 //Carga el siguiente nombre
   .setColor(color(255))          //Color de letra
   .setFont(font)                 //Propiedad de letra
   .setSize(9)                    //Tamaño de letra
   ;
// Doble Rotación X [grad]
cp5.getController("DRXCASERVII")     //Indica que se modificara texto en el controlador
  .getCaptionLabel()              //Dar propiedad de texto
  .align(CENTER,CENTER)           //Alineación del texto
  .toUpperCase(false)             //Cambia mayúsculas false
  .set("Eje X [grad]")            //Carga el siguiente nombre
  .setColor(color(255))           //Color de letra
  .setFont(font)                  //Propiedad de letra
  .setSize(8)                     //Tamaño de letra
  ;
// Doble Rotación Y [grad]
cp5.getController("DRYCASERVII")     //Indica que se modificara texto en el controlador
  .getCaptionLabel()              //Dar propiedad de texto
  .align(CENTER,CENTER)           //Alineación del texto
  .toUpperCase(false)             //Cambia mayúsculas false
  .set("Eje Y [grad]")            //Carga el siguiente nombre
  .setColor(color(255))           //Color de letra
  .setFont(font)                  //Propiedad de letra
  .setSize(8)                     //Tamaño de letra
  ;
//Propiedades de letra del menú de planos G3
cp5g3.getController("Plano XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Plano XZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Plano YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Plano XY Rotación Eje X").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Plano XY Rotación Eje Y").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Plano XY Rotación Eje Z").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Plano XY 2 Rotaciones").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Cilindro Parabólico Plano XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Cilindro Parabólico Plano XZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("Cilindro Parabólico Plano YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g3.getController("+ XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g3.getController("+ XZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g3.getController("+ YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g3.getController("- XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g3.getController("- XZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g3.getController("- YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);

// Grupo 4, Controlador-B (Scara-ER 14)
 g4.captionLabel()                 //Indica que se modificara texto en dicho grupo
   .align(CENTER,CENTER)           //Alineación del texto
   .toUpperCase(true)              //Cambia mayúsculas true
   .set("Controlador-B (Scara-ER 14)")     //Carga el siguiente nombre
   .setColor(color(255))           //Color de letra
   .setFont(font)                  //Propiedad de letra
   .setSize(14)                    //Tamaño de letra
   ;
 //Dimensión [cm] Slider Grupo 4
 cp5.getController("DCBSER14")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()              //Dar propiedad de texto
   .align(CENTER,CENTER)           //Alineación del texto
   .toUpperCase(false)             //Cambia mayúsculas false
   .set("Dimensión [cm]")          //Carga el siguiente nombre
   .setColor(color(255))           //Color de letra
   .setFont(font)                  //Propiedad de letra
   .setSize(13)                    //Tamaño de letra
   ;
```

```
 // Rotación X [grad]
 cp5.getController("RXCBSER14")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()            //Dar propiedad de texto
   .align(CENTER,CENTER)         //Alineación del texto
   .toUpperCase(false)           //Cambia mayúsculas false
   .set("[grad]")                //Carga el siguiente nombre
   .setColor(color(255))         //Color de letra
   .setFont(font)                //Propiedad de letra
   .setSize(9)                   //Tamaño de letra
   ;
 // Rotación Y [grad]
 cp5.getController("RYCBSER14")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()            //Dar propiedad de texto
   .align(CENTER,CENTER)         //Alineación del texto
   .toUpperCase(false)           //Cambia mayúsculas false
   .set("[grad]")                //Carga el siguiente nombre
   .setColor(color(255))         //Color de letra
   .setFont(font)                //Propiedad de letra
   .setSize(9)                   //Tamaño de letra
   ;
 // Rotación Z [grad]
 cp5.getController("RZCBSER14")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()            //Dar propiedad de texto
   .align(CENTER,CENTER)         //Alineación del texto
   .toUpperCase(false)           //Cambia mayúsculas false
   .set("[grad]")                //Carga el siguiente nombre
   .setColor(color(255))         //Color de letra
   .setFont(font)                //Propiedad de letra
   .setSize(9)                   //Tamaño de letra
   ;
// Doble Rotación X [grad]
cp5.getController("DRXCBSER14")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()            //Dar propiedad de texto
   .align(CENTER,CENTER)         //Alineación del texto
   .toUpperCase(false)           //Cambia mayúsculas false
   .set("Eje X [grad]")          //Carga el siguiente nombre
   .setColor(color(255))         //Color de letra
   .setFont(font)                //Propiedad de letra
   .setSize(8)                   //Tamaño de letra
   ;
// Doble Rotación Y [grad]
cp5.getController("DRYCBSER14")     //Indica que se modificara texto en el controlador
   .getCaptionLabel()            //Dar propiedad de texto
   .align(CENTER,CENTER)         //Alineación del texto
   .toUpperCase(false)           //Cambia mayúsculas false
   .set("Eje Y [grad]")          //Carga el siguiente nombre
   .setColor(color(255))         //Color de letra
   .setFont(font)                //Propiedad de letra
   .setSize(8)                   //Tamaño de letra
   ;
//Propiedades de letra del menú de planos G4
cp5g4.getController("Plano XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Plano XZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Plano YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Plano XY Rotación Eje X").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Plano XY Rotación Eje Y").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Plano XY Rotación Eje Z").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Plano XY 2 Rotaciones").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Cilindro Parabólico Plano XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Cilindro Parabólico Plano XZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("Cilindro Parabólico Plano YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
cp5g4.getController("+ XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g4.getController("+ XZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g4.getController("+ YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g4.getController("- XY").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g4.getController("- XZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);
cp5g4.getController("- YZ").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(11);

// Grupo 5, Acerca de Interfaz CADR
 g5.captionLabel()                //Indica que se modificara texto en dicho grupo
   .align(CENTER,CENTER)          //Alineación del texto
   .toUpperCase(true)             //Cambia mayúsculas true
   .set("Acerca de Interfaz CADR")     //Carga el siguiente nombre
   .setColor(color(255))          //Color de letra
```

```
    .setFont(font)                  //Propiedad de letra
    .setSize(14)                    //Tamaño de letra
    ;
  //Propiedades de letra del menú G5
  cp5.getController("Interfaz CADR.").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
  cp5.getController("Versión 1.0").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
  cp5.getController("14 - 01 - 2014").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
  cp5.getController("UNAM reservados todos los derechos.").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
  cp5.getController("Elaboró:").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
  cp5.getController("Jhovvany Hazzael Hernández Clemente").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);
  cp5.getController("Revisó: M.I. Humberto Mancilla Alonso").getCaptionLabel().toUpperCase(false).setColor(color(255)).setFont(font).setSize(12);


  // Creación del accordion
  accordion = cp5.addAccordion("acc")     //Nombre del accordion
            .setPosition(30,30)      //Posición inicial del accordion
            .setWidth(280)           //Ancho del accordion
            .addItem(g1)             //Agregar grupos
            .addItem(g2)
            .addItem(g3)
            .addItem(g4)
            .addItem(g5)
            ;
}

//***** Subrutina para cargar los archivos SVG *****
void file(){
  selectInput("Elegir archivo SVG...","fileSelected");     //Abre ventana de búsqueda de archivo y va a la siguiente subrutina
  cXY = color(255,0,50,80);                          //Asignación de color por default
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
}
void fileSelected(File selection){
  if (selection == null){
    println("Windows fue cerrado o el usuario cancelo.");     //Si no se selecciona nada imprime
  }
  if (selection != null){
    println("User selected " + selection.getAbsolutePath());     //Se asigna la cadena para localizar el archivo
    shp = RG.loadShape(selection.getAbsolutePath());            //Se carga el archivo a la variable shp para su procesamiento
    shp = RG.centerIn(shp, g, 100);
  }
}

//***** Subrutina de lo que se dibuja en el Sketch *****
void draw(){
  background(220);     //Color de fondo en ventana principal

  interfaz();         //Subrutina de ambiente de la interfaz
  planos();           //Subrutina de ubicación de planos
  dibujo();           //Llamado a la subrutina que encuentra los puntos de la imagen
  ejesXYZ();           //Subrutina de ejes
}

//***** Subrutina de dibujos (Decoración en la interfaz) *****
void interfaz(){
  pushMatrix();                //Guarda los valores siguientes
    fill(c);                  //Configura el color de relleno de lo que se dibuje posteriormente
    noStroke();               //Indica falta de borde de lo que se dibuje posteriormente
    ellipse(300,100,300,300);     //Dibujos
    ellipse(150,400,150,150);
    ellipse(380,260,50,50);
  popMatrix();                 //Fin de Guardar
}

//***** Subrutina de ubicación de planos *****
void planos(){
  pushMatrix();
    noStroke();             //Indica falta de borde de lo que se dibuje posteriormente
    translate(880, 100);     //Ubicación del origen para comenzar dibujo
    rotateX(angX);          //Ángulos de rotación para cada eje
    rotateY(angY);
    rotateZ(angZ);
    pushMatrix();
      rotateY(ang1);
```

```
    fill(cXZ);              //Color de relleno plano XZ
    rect(-60,-60,120,120);    //Dibujo de plano XZ
   popMatrix();
   pushMatrix();
    rotateX(ang2);
    fill(cXY);              //Color de relleno plano XY
    rect(-60,-60,120,120);    //Dibujo de plano XY
   popMatrix();
   pushMatrix();
    fill(cYZ);              //Color de relleno plano YZ
    rect(-60,-60,120,120);    //Dibujo de plano YZ
   popMatrix();
  popMatrix();
}

//***** Subrutina de ubicación de Ejes XYZ *****
void ejesXYZ(){
  pushMatrix();
  PFont ppfont = createFont("Eras Bold ITC",20,true);    //Creando un pfont con las características Arial 20 y se usa true para tener un smooth en la letra
  smooth();
  translate(800, 150);    //Ubicación del origen para comenzar dibujo
   pushMatrix();        //Z
    stroke(105,167,12);
    strokeWeight(4);
    line(100,100,100,50);
    line(100,50,92,58);
    line(100,50,108,58);
    fill(105,167,12);
    textFont(ppfont,20);
    text("Z",95,43);
   popMatrix();
   pushMatrix();        //Y
    stroke(0,0,255);
    strokeWeight(4);
    line(100,100,150,100);
    line(150,100,142,108);
    line(150,100,142,92);
    fill(0,0,255);
    textFont(ppfont,20);
    text("Y",160,105);
   popMatrix();
   pushMatrix();        //X
    stroke(255,0,0);
    strokeWeight(4);
    line(100,100,60,130);
    line(60,130,70,131);
    line(60,130,62,120);
    fill(255,0,0);
    textFont(ppfont,20);
    text("X",40,150);
   popMatrix();
  popMatrix();
}

//***** Subrutina que encuentra los puntos de la imagen *****
public void dibujo(){
  if (shp != null){
   pushMatrix ();
   translate(width/2, height/2);    //Se coloca la imagen al centro
   stroke(#2D4D90);                //Color del background

   // Split y escala del shape (forma)
   shp2 = new RShape(shp);           //RShape(points) --> Es el constructor
   shp2.insertHandleInPaths(0.5);    //.insertHandleInPaths(0.5) --> Inserta un punto de división en cada comando shape
   s1= dimension/RDMCASERVII;        //Variable de Escala de la Imagen
   scale(s1);                      //Escala de la Imagen

   // Dibujo de las asas (handles) y las líneas que construye
   ps = shp2.getHandles();    //.getHandles() --> Devuelve los puntos en el camino de una serie de RPoint.
   beginShape();             //Comienza a escuchar vértices para armar una forma

   factdim = dimension/50;                //Factor de dimensión
   CU=(((1.25/35.43307)*3.05)*factdim)*10;    //Conversión de Unidades (Pixeles vs. cm)
```

```
   dataRS = new float[ps.length][2];     //Declaración de un arreglo de 2 dimensiones

   //Vector de Información
   arrayMI = new int[ps.length][5];

   for(int i= 0; i< ps.length; i++){     //Ciclo para hacer la localización de las coordenadas
    ellipse(ps[i].x, ps[i].y, 10, 10);

    dataRS[i][0] = ps[i].x*CU;
    dataRS[i][1] = ps[i].y*CU;

    noLoop();
   }
   endShape();     //Fin de la forma

   //Dibuja las particiones y escala de shape
   stroke(80,220,100);
   strokeWeight(3);
   shp2.draw();
   popMatrix ();
  }
}

//***** Subrutinas para hacer loops en la interfaz *****
public void mousePressed(){
 loop();
 ignoringStyles = !ignoringStyles;
 RG.ignoreStyles(ignoringStyles);
}
public void mouseMoved(){
 loop();
}

//*************************************************************************************************************************************************************
//***** LAS SIGUIENTES SUBRUTINAS COORESPONDEN A GENERAR EL ARCHIVO .CBU DE CADA CONTROLADOR ASÍ COMO EL ROBOT A UTILIZAR *****
//*************************************************************************************************************************************************************

//***** Subrutina Controlador-A (Scorbot-ER VII) para generar los archivos en los planos *****
void DDCASERVII(){
 dimension = cp5.getController("DCASERVII").getValue();     //Dimensión del dibujo
}
void POSCASERVII(int thePOSCASERVII){     //Subrutina para cargar al signo de la parábola
 switch(thePOSCASERVII){
   case(0): SigP = 0.1; break;
   case(1): SigP = 0.1; break;
   case(2): SigP = 0.1; break;
 }
}
void NEGCASERVII(int theNEGCASERVII){     //Subrutina para cargar al signo de la parábola
 switch(theNEGCASERVII){
   case(0): SigP = -0.1; break;
   case(1): SigP = -0.1; break;
   case(2): SigP = -0.1; break;
 }
}
void CASERVII(int theCASERVII){
 DDCASERVII();
 RSV();
 switch(theCASERVII){
   case(0):
    cXY = color(0,200);     //Cambio de color en el plano
    cXZ = color(0,255,0,80);
    cYZ = color(0,0,255,100);
    PHP = -850;
    PHR = -201;
    for(int i= 0; i< ps.length; i++){
     PPx = arrayMI[i][0];
     PPy = arrayMI[i][1];
     PPz = arrayMI[i][2];
     P = arrayMI[i][3];
     R = arrayMI[i][4];
     //Asignación del plano
     arrayMI[i][0] = PPx +200;
     arrayMI[i][1] = PPy;
```

```
    arrayMI[i][2] = PPz;
    arrayMI[i][3] = P + PHP;
    arrayMI[i][4] = R + PHR;
   }
   RSVII();
   break;
  case(1):
   cXY = color(255,0,50,80);
   cXZ = color(0,200);     //Cambio de color en el plano
   cYZ = color(0,0,255,100);
   PHP = -850;
   PHR = -1087;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    R = arrayMI[i][4];
    //Asignación del plano
    arrayMI[i][0] = -PPy;
    arrayMI[i][1] = PPz;
    arrayMI[i][2] = -PPx;
    arrayMI[i][3] = P + PHP;
    arrayMI[i][4] = R + PHR;
   }
   RSVII();
   break;
  case(2):
   cXY = color(255,0,50,80);
   cXZ = color(0,255,0,80);
   cYZ = color(0,200);     //Cambio de color en el plano
   PHP = 76;
   PHR = -201;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    R = arrayMI[i][4];
    //Asignación del plano
    arrayMI[i][0] = -PPz + 2000;
    arrayMI[i][1] = -PPy;
    arrayMI[i][2] = -PPx;
    arrayMI[i][3] = P + PHP;
    arrayMI[i][4] = R + PHR;
   }
   RSVII();
   break;
  case(3):
   cXY = color(0,200);     //Cambio de color en el plano
   cXZ = color(0,255,0,80);
   cYZ = color(0,0,255,100);
   PHP = -850;
   PHR = -201;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    R = arrayMI[i][4];
    //Asignando la Rotación en Y
    float deg = cp5.getController("RXCASERVII").getValue();     //Ángulo
    float CO = cos((deg*PI)/180);
    float SO = sin((deg*PI)/180);
    //Aquí va la matriz de rotación
    float PRx = PPx;
    float PRy = PPy*CO-PPz*SO;
    float PRz = PPy*SO+PPz*CO;
    //Redondeo
    PPx = round(PRx);
    PPy = round(PRy);
    PPz = round(PRz);
    //Asignación del plano
    arrayMI[i][0] = PPx + 200;
```

```
    arrayMI[i][1] = PPy;
    arrayMI[i][2] = PPz;
    arrayMI[i][3] = P + PHP;
    arrayMI[i][4] = R + PHR;
   }
   RSVII();
   break;
  case(4):
   cXY = color(0,200);     //Cambio de color en el plano
   cXZ = color(0,255,0,80);
   cYZ = color(0,0,255,100);
   PHP = -850;
   PHR = -201;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    R = arrayMI[i][4];
    //Asignando la Rotación en Y
    float deg = cp5.getController("RYCASERVII").getValue();     //Ángulo
    float CO = cos((deg*PI)/180);
    float SO = sin((deg*PI)/180);
    //Aquí va la matriz de rotación
    float PRx = PPx*CO+PPz*SO;
    float PRy = PPy;
    float PRz = -PPx*SO+PPz*CO;
    //Redondeo
    PPx = round(PRx);
    PPy = round(PRy);
    PPz = round(PRz);
    //Asignación del plano
    arrayMI[i][0] = PPx + 200;
    arrayMI[i][1] = PPy;
    arrayMI[i][2] = PPz;
    arrayMI[i][3] = P + PHP;
    arrayMI[i][4] = R + PHR;
   }
   RSVII();
   break;
  case(5):
   cXY = color(0,200);     //Cambio de color en el plano
   cXZ = color(0,255,0,80);
   cYZ = color(0,0,255,100);
   PHP = -850;
   PHR = -201;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    R = arrayMI[i][4];
    //Asignando la Rotación en Z
    float deg = cp5.getController("RZCASERVII").getValue();     //Ángulo
    float CO = cos((deg*PI)/180);
    float SO = sin((deg*PI)/180);
    //Aquí va la matriz de rotación
    float PRx = PPx*CO-PPy*SO;
    float PRy = PPx*SO+PPy*CO;
    float PRz = PPz;
    //Redondeo
    PPx = round(PRx);
    PPy = round(PRy);
    PPz = round(PRz);
    //Asignación del plano
    arrayMI[i][0] = PPx + 200;
    arrayMI[i][1] = PPy;
    arrayMI[i][2] = PPz;
    arrayMI[i][3] = P + PHP;
    arrayMI[i][4] = R + PHR;
   }
   RSVII();
   break;
  case(6):
```

```
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = -850;
  PHR = -201;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   R = arrayMI[i][4];
   //Asignando la Rotación en X
   float degA = cp5.getController("DRXCASERVII").getValue();     //Ángulo
   float COA = cos((degA*PI)/180);
   float SOA = sin((degA*PI)/180);
   float PRAx = PPx*COA+PPz*SOA;
   float PRAy = PPy;
   float PRAz = -PPx*SOA+PPz*COA;
   //Asignando la Rotación en Y
   float degB = cp5.getController("DRYCASERVII").getValue();     //Ángulo
   float COB = cos((degB*PI)/180);
   float SOB = sin((degB*PI)/180);
   float PRBx = PRAx;
   float PRBy = PRAy*COB-PRAz*SOB;
   float PRBz = PRAy*SOB+PRAz*COB;
   //Redondeo
   PPx = round(PRBx);
   PPy = round(PRBy);
   PPz = round(PRBz);
   //Asignación del plano
   arrayMI[i][0] = PPx + 200;
   arrayMI[i][1] = PPy;
   arrayMI[i][2] = PPz;
   arrayMI[i][3] = P + PHP;
   arrayMI[i][4] = R + PHR;
  }
  RSVII();
  break;
 case(7):
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = -850;
  PHR = -201;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = round(SigP*pow(arrayMI[i][1]/10,2));
   P = arrayMI[i][3];
   R = arrayMI[i][4];
   //Asignación del plano
   arrayMI[i][0] = PPx + 200;
   arrayMI[i][1] = PPy;
   arrayMI[i][2] = PPz;
   arrayMI[i][3] = P + PHP;
   arrayMI[i][4] = R + PHR;
  }
  RSVII();
  break;
 case(8):
  cXY = color(255,0,50,80);
  cXZ = color(0,200);     //Cambio de color en el plano
  cYZ = color(0,0,255,100);
  PHP = -850;
  PHR = -1087;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = round(SigP*pow(arrayMI[i][1]/10,2));
   P = arrayMI[i][3];
   R = arrayMI[i][4];
   //Asignación del plano
   arrayMI[i][0] = -PPy;
   arrayMI[i][1] = PPz;
```

```
      arrayMI[i][2] = -PPx;
      arrayMI[i][3] = P + PHP;
      arrayMI[i][4] = R + PHR;
    }
    RSVII();
    break;
  case(9):
    cXY = color(255,0,50,80);
    cXZ = color(0,255,0,80);
    cYZ = color(0,200);     //Cambio de color en el plano
    PHP = 76;
    PHR = -201;
    for(int i= 0; i< ps.length; i++){
      PPx = arrayMI[i][0];
      PPy = arrayMI[i][1];
      PPz = round(SigP*pow(arrayMI[i][1]/10,2));
      P = arrayMI[i][3];
      R = arrayMI[i][4];
      //Asignación del plano
      arrayMI[i][0] = -PPz + 2000;
      arrayMI[i][1] = -PPy;
      arrayMI[i][2] = -PPx;
      arrayMI[i][3] = P + PHP;
      arrayMI[i][4] = R + PHR;
    }
    RSVII();
    break;
 }
}
void RSV(){
 for(int i= 0; i< ps.length; i++){
  float x = dataRS[i][0];
  float y = dataRS[i][1];
  //Puntos que corresponden al dibujo en el plano XYZ de acuerdo al punto origen
  PPx = round(y)*10;
  PPy = round(x)*10;
  PPz = 0;
  P = 0;
  R = 0;
  //Asignación del plano
  arrayMI[i][0] = PPx;
  arrayMI[i][1] = PPy;
  arrayMI[i][2] = PPz;
  arrayMI[i][3] = P;
  arrayMI[i][4] = R;
 }
}
void RSVII(){     //Subrutina para generar el tipo de archivo
 String[] marky = new String[13+2*(ps.length)];
 for(int i= 0; i< ps.length; i++){
  //Puntos HOME predefinidos X, Y, Z, P; y R se definen para cada caso de CASERVII
  PHX = 4000;
  PHY = -353;
  PHZ = 3000;
  PHP = PHP;
  PHR = PHR;
  //Asignación del plano para dibujar a partir del punto origen
  int Px = PHX + arrayMI[i][0];
  int Py = PHY + arrayMI[i][1];
  int Pz = PHZ + arrayMI[i][2];
  int P = arrayMI[i][3];
  int R = arrayMI[i][4];
  if(i==0){
   marky[0] = "";
   marky[1] = "$p 1 1 0 0 0 0 0 0 0";
   marky[2] = "$p 2 1 $%^&$ 465 -14771 -9463 -17 0 2";
   marky[3] = "$p 3 1 0 3018 -353 9290 76 -201 -32766";                    //Home del robot
   marky[4] = "$p 4 1 P1 "+PHX+" "+PHY+" "+PHZ+" "+PHP+" "+PHR+" -32766";     //Home predefinido
   i=0;
  }
  marky[5 + i] = "$p "+(5+i)+ " 1 P"+(2+i)+" "+Px+" "+Py+" "+Pz+" "+P+" "+R+" -32766";
 }
 marky[ps.length + 5] = "$pr 1 CADR";
 marky[ps.length + 6] = "";
```

```
 marky[ps.length + 7] = "            PROGRAM  CADR";
 marky[ps.length + 8] = "            **********************";
 marky[ps.length + 9] = "CLOSE";
 marky[ps.length + 10] = "MOVE     P1";
 for(int j= 0; j< ps.length; j++){
  marky[ps.length + 11 + j] = "MOVE     P"+(2+j);
 }
 marky[11+2*(ps.length)+0] = "END";
 marky[11+2*(ps.length)+1] = "(END)";
 saveStrings( "data/CADR.CBU",marky);
}

//*************************************************************************************************

//***** Subrutinas Controlador-B (Scora-ER 14) para generar los archivos en los planos correspondientes
void DDCBSER14(){
 dimension = cp5.getController("DCBSER14").getValue();     //Dimensión del dibujo
}
void POSCBSER14(int thePOSCBSER14){     //Subrutina para cargar al signo de la parábola
 switch(thePOSCBSER14){
  case(0): SigP = 0.1; break;
  case(1): SigP = 0.1; break;
  case(2): SigP = 0.1; break;
 }
}
void NEGCBSER14(int theNEGCBSER14){     //Subrutina para cargar al signo de la parábola
 switch(theNEGCBSER14){
  case(0): SigP = -0.1; break;
  case(1): SigP = -0.1; break;
  case(2): SigP = -0.1; break;
 }
}
void CBSER14(int theCBSER14){     //Subrutina para generar el tipo de archivo
 DDCBSER14();
 RScaraP1();
 switch(theCBSER14){
  case(0):
   cXY = color(0,200);     //Cambio de color en el plano
   cXZ = color(0,255,0,80);
   cYZ = color(0,0,255,100);
   PHP = 0;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    //Asignación del plano
    arrayMI[i][0] = PPx;
    arrayMI[i][1] = PPy;
    arrayMI[i][2] = PPz;
    arrayMI[i][3] = P + PHP;
   }
   RScara();
   break;
  case(1):
   cXY = color(255,0,50,80);
   cXZ = color(0,200);     //Cambio de color en el plano
   cYZ = color(0,0,255,100);
   PHP = -100000;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    //Asignación del plano
    arrayMI[i][0] = -PPy;
    arrayMI[i][1] = PPz;
    arrayMI[i][2] = -PPx;
    arrayMI[i][3] = P + PHP;
   }
   RScara();
   break;
  case(2):
   cXY = color(255,0,50,80);
```

```
  cXZ = color(0,255,0,80);
  cYZ = color(0,200);     //Cambio de color en el plano
  PHP = 0;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   //Asignación del plano
   arrayMI[i][0] = -PPz;
   arrayMI[i][1] = -PPy;
   arrayMI[i][2] = -PPx;
   arrayMI[i][3] = P + PHP;
  }
  RScara();
  break;
 case(3):
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = 0;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   //Asignando la Rotación en Y
   float deg = cp5.getController("RXCBSER14").getValue();     //Ángulo
   float CO = cos((deg*PI)/180);
   float SO = sin((deg*PI)/180);
   //Aquí va la matriz de rotación
   float PRx = PPx;
   float PRy = PPy*CO-PPz*SO;
   float PRz = PPy*SO+PPz*CO;
   //Redondeo
   PPx = round(PRx);
   PPy = round(PRy);
   PPz = round(PRz);
   //Asignación del plano
   arrayMI[i][0] = PPx;
   arrayMI[i][1] = PPy;
   arrayMI[i][2] = PPz;
   arrayMI[i][3] = P + PHP;
  }
  RScara();
  break;
 case(4):
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = 0;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   //Asignando la Rotación en Y
   float deg = cp5.getController("RYCBSER14").getValue();     //Ángulo
   float CO = cos((deg*PI)/180);
   float SO = sin((deg*PI)/180);
   //Aquí va la matriz de rotación
   float PRx = PPx*CO+PPz*SO;
   float PRy = PPy;
   float PRz = -PPx*SO+PPz*CO;
   //Redondeo
   PPx = round(PRx);
   PPy = round(PRy);
   PPz = round(PRz);
   //Asignación del plano
   arrayMI[i][0] = PPx;
   arrayMI[i][1] = PPy;
   arrayMI[i][2] = PPz;
   arrayMI[i][3] = P + PHP;
  }
```

```
  RScara();
  break;
 case(5):
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = 0;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   //Asignando la Rotación en Y
   float deg = cp5.getController("RZCBSER14").getValue();     //Ángulo
   float CO = cos((deg*PI)/180);
   float SO = sin((deg*PI)/180);
   //Aquí va la matriz de rotación
   float PRx = PPx*CO-PPy*SO;
   float PRy = PPx*SO+PPy*CO;
   float PRz = PPz;
   //Redondeo
   PPx = round(PRx);
   PPy = round(PRy);
   PPz = round(PRz);
   //Asignación del plano
   arrayMI[i][0] = PPx;
   arrayMI[i][1] = PPy;
   arrayMI[i][2] = PPz;
   arrayMI[i][3] = P + PHP;
  }
  RScara();
  break;
 case(6):
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = 0;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   //Asignando la Rotación en Y
   float degA = cp5.getController("DRXCBSER14").getValue();     //Ángulo
   float COA = cos((degA*PI)/180);
   float SOA = sin((degA*PI)/180);
   float PRAx = PPx*COA+PPz*SOA;
   float PRAy = PPy;
   float PRAz = -PPx*SOA+PPz*COA;
   //Asignando la Rotación en X
   float degB = cp5.getController("DRYCBSER14").getValue();     //Ángulo
   float COB = cos((degB*PI)/180);
   float SOB = sin((degB*PI)/180);
   float PRBx = PRAx;
   float PRBy = PRAy*COB-PRAz*SOB;
   float PRBz = PRAy*SOB+PRAz*COB;
   //Redondeo
   PPx = round(PRBx);
   PPy = round(PRBy);
   PPz = round(PRBz);
   //Asignación del plano
   arrayMI[i][0] = PPx;
   arrayMI[i][1] = PPy;
   arrayMI[i][2] = PPz;
   arrayMI[i][3] = P + PHP;
  }
  RScara();
  break;
 case(7):
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = 0;
  for(int i= 0; i< ps.length; i++){
```

```
      PPx = arrayMI[i][0];
      PPy = arrayMI[i][1];
      PPz = round(SigP*pow(arrayMI[i][1]/1000,2))*100;
      P = arrayMI[i][3];
      //Asignación del plano
      arrayMI[i][0] = PPx;
      arrayMI[i][1] = PPy;
      arrayMI[i][2] = PPz;
      arrayMI[i][3] = P + PHP;
     }
     RScara();
     break;
    case(8):
     cXY = color(255,0,50,80);
     cXZ = color(0,200);     //Cambio de color en el plano
     cYZ = color(0,0,255,100);
     PHP = -100000;
     for(int i= 0; i< ps.length; i++){
      PPx = arrayMI[i][0];
      PPy = arrayMI[i][1];
      PPz = round(SigP*pow(arrayMI[i][1]/1000,2))*100;
      P = arrayMI[i][3];
      //Asignación del plano
      arrayMI[i][0] = -PPy;
      arrayMI[i][1] = PPz;
      arrayMI[i][2] = -PPx;
      arrayMI[i][3] = P + PHP;
     }
     RScara();
     break;
    case(9):
     cXY = color(255,0,50,80);
     cXZ = color(0,255,0,80);
     cYZ = color(0,200);     //Cambio de color en el plano
     PHP = 0;
     for(int i= 0; i< ps.length; i++){
      PPx = arrayMI[i][0];
      PPy = arrayMI[i][1];
      PPz = round(SigP*pow(arrayMI[i][1]/1000,2))*100;
      P = arrayMI[i][3];
      //Asignación del plano
      arrayMI[i][0] = -PPz;
      arrayMI[i][1] = -PPy;
      arrayMI[i][2] = -PPx;
      arrayMI[i][3] = P + PHP;
     }
     RScara();
     break;
  }
 }
 void RScaraP1(){     //Subrutina para el manejo de información
  for(int i= 0; i< ps.length; i++){
   float x = dataRS[i][0];
   float y = dataRS[i][1];
   //Puntos que corresponden al dibujo en el plano XYZ de acuerdo al punto origen
   PPx = round(y)*1000;
   PPy = round(x)*1000;
   PPz = 0;
   P = 0;
   //Asignación del plano
   arrayMI[i][0] = PPx;
   arrayMI[i][1] = PPy;
   arrayMI[i][2] = PPz;
   arrayMI[i][3] = P;
  }
 }
 void RScara(){     //Subrutina para generar el tipo de archivo
  String[] marky = new String[11+2*(ps.length)];
  for(int i= 0; i< ps.length; i++){
   //Puntos HOME predefinidos X, Y, Z, P; y R se definen para cada caso de CBSER14
   PHX = 395000;
   PHY = 0;
   PHZ = -100000;
   PHP = PHP;
```

```
   //Asignación del plano para dibujar a partir del punto origen
   int Px = PHX + arrayMI[i][0];
   int Py = PHY + arrayMI[i][1];
   int Pz = PHZ + arrayMI[i][2];
   int P = arrayMI[i][3];
   if(i==0){
    marky[0] = "";
    marky[1] = "$p 1 1 $%^&$ 465 0 -133296 0 0";
    marky[2] = "$p 2 1 0 395000 0 -168000 0 -32766";                //Home del robot
    marky[3] = "$p 3 1 P1 "+PHX+" "+PHY+" "+PHZ+" "+PHP+" -32766";     //Home predefinido
    i=0;
   }
   marky[4 + i] = "$p "+(4+i)+ " 1 P"+(2+i)+" "+Px+" "+Py+" "+Pz+" "+P+" -32766";
  }
  marky[ps.length + 4] = "$pr 1 CADR";
  marky[ps.length + 5] = "";
  marky[ps.length + 6] = "            PROGRAM   CADR";
  marky[ps.length + 7] = "            **********************";
  marky[ps.length + 8] = "MOVE     P1";
  for(int j= 0; j< ps.length; j++){
   marky[ps.length + 9 + j] = "MOVE     P"+(2+j);
  }
  marky[9+2*(ps.length)+0] = "END";
  marky[9+2*(ps.length)+1] = "(END)";
  saveStrings( "data/CADR.CBU",marky);
}

//*************************************************************************************************

//***** Subrutina Controlador-A (Scorbot-ER V+) para generar los archivos en los planos *****
void DDCASERVP(){
 dimension = cp5.getController("DCASERVP").getValue();     //Dimensión del dibujo
}
void POSCASERVP(int thePOSCASERVP){     //Subrutina para cargar al signo de la parábola
 switch(thePOSCASERVP){
   case(0): SigP = 0.1; break;
   case(1): SigP = 0.1; break;
 }
}
void NEGCASERVP(int theNEGCASERVP){     //Subrutina para cargar al signo de la parábola
 switch(theNEGCASERVP){
   case(0): SigP = -0.1; break;
   case(1): SigP = -0.1; break;
 }
}
void CASERVP(int theCASERVP){
 DDCASERVP();
 RSV();
 switch(theCASERVP) {
   case(0):
    cXY = color(0,200);     //Cambio de color en el plano
    cXZ = color(0,255,0,80);
    cYZ = color(0,0,255,100);
    PHP = -636;
    PHR = 0;
    for(int i= 0; i< ps.length; i++){
     PPx = arrayMI[i][0];
     PPy = arrayMI[i][1];
     PPz = arrayMI[i][2];
     P = arrayMI[i][3];
     R = arrayMI[i][4];
     //Asignación del plano
     arrayMI[i][0] = PPx;
     arrayMI[i][1] = PPy;
     arrayMI[i][2] = PPz;
     arrayMI[i][3] = P + PHP;
     arrayMI[i][4] = R + PHR;
    }
    RSVP();
    break;
   case(1):
    cXY = color(255,0,50,80);
    cXZ = color(0,255,0,80);
    cYZ = color(0,200);     //Cambio de color en el plano
```

```
  PHP = 264;
  PHR = 0;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   R = arrayMI[i][4];
   //Asignación del plano
   arrayMI[i][0] = -PPz + 600;
   arrayMI[i][1] = -PPy;
   arrayMI[i][2] = -PPx + 1500;
   arrayMI[i][3] = P + PHP;
   arrayMI[i][4] = R + PHR;
  }
  RSVP();
  break;
 case(2):
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = -636;
  PHR = 0;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   R = arrayMI[i][4];
   //Asignando la Rotación en Y
   float deg = cp5.getController("RXCASERVP").getValue();     //Ángulo
   float CO = cos((deg*PI)/180);
   float SO = sin((deg*PI)/180);
   //Aquí va la matriz de rotación
   float PRx = PPx;
   float PRy = PPy*CO-PPz*SO;
   float PRz = PPy*SO+PPz*CO;
   //Redondeo
   PPx = round(PRx);
   PPy = round(PRy);
   PPz = round(PRz);
   //Asignación del plano
   arrayMI[i][0] = PPx;
   arrayMI[i][1] = PPy;
   arrayMI[i][2] = PPz;
   arrayMI[i][3] = P + PHP;
   arrayMI[i][4] = R + PHR;
  }
  RSVP();
  break;
 case(3):
  cXY = color(0,200);     //Cambio de color en el plano
  cXZ = color(0,255,0,80);
  cYZ = color(0,0,255,100);
  PHP = -636;
  PHR = 0;
  for(int i= 0; i< ps.length; i++){
   PPx = arrayMI[i][0];
   PPy = arrayMI[i][1];
   PPz = arrayMI[i][2];
   P = arrayMI[i][3];
   R = arrayMI[i][4];
   //Asignando la Rotación en Y
   float deg = cp5.getController("RYCASERVP").getValue();     //Ángulo
   float CO = cos((deg*PI)/180);
   float SO = sin((deg*PI)/180);
   //Aquí va la matriz de rotación
   float PRx = PPx*CO+PPz*SO;
   float PRy = PPy;
   float PRz = -PPx*SO+PPz*CO;
   //Redondeo
   PPx = round(PRx);
   PPy = round(PRy);
   PPz = round(PRz);
```

```
    //Asignación del plano
    arrayMI[i][0] = PPx;
    arrayMI[i][1] = PPy;
    arrayMI[i][2] = PPz;
    arrayMI[i][3] = P + PHP;
    arrayMI[i][4] = R + PHR;
   }
   RSVP();
   break;
  case(4):
   cXY = color(0,200);     //Cambio de color en el plano
   cXZ = color(0,255,0,80);
   cYZ = color(0,0,255,100);
   PHP = -636;
   PHR = 0;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    R = arrayMI[i][4];
    //Asignando la Rotación en Z
    float deg = cp5.getController("RZCASERVP").getValue();     //Ángulo
    float CO = cos((deg*PI)/180);
    float SO = sin((deg*PI)/180);
    //Aquí va la matriz de rotación
    float PRx = PPx*CO-PPy*SO;
    float PRy = PPx*SO+PPy*CO;
    float PRz = PPz;
    //Redondeo
    PPx = round(PRx);
    PPy = round(PRy);
    PPz = round(PRz);
    //Asignación del plano
    arrayMI[i][0] = PPx;
    arrayMI[i][1] = PPy;
    arrayMI[i][2] = PPz;
    arrayMI[i][3] = P + PHP;
    arrayMI[i][4] = R + PHR;
   }
   RSVP();
   break;
  case(5):
   cXY = color(0,200);     //Cambio de color en el plano
   cXZ = color(0,255,0,80);
   cYZ = color(0,0,255,100);
   PHP = -636;
   PHR = 0;
   for(int i= 0; i< ps.length; i++){
    PPx = arrayMI[i][0];
    PPy = arrayMI[i][1];
    PPz = arrayMI[i][2];
    P = arrayMI[i][3];
    R = arrayMI[i][4];
    //Asignando la Rotación en X
    float degA = cp5.getController("DRXCASERVP").getValue();     //Ángulo
    float COA = cos((degA*PI)/180);
    float SOA = sin((degA*PI)/180);
    float PRAx = PPx*COA+PPz*SOA;
    float PRAy = PPy;
    float PRAz = -PPx*SOA+PPz*COA;
    //Asignando la Rotación en Y
    float degB = cp5.getController("DRYCASERVP").getValue();     //Ángulo
    float COB = cos((degB*PI)/180);
    float SOB = sin((degB*PI)/180);
    float PRBx = PRAx;
    float PRBy = PRAy*COB-PRAz*SOB;
    float PRBz = PRAy*SOB+PRAz*COB;
    //Redondeo
    PPx = round(PRBx);
    PPy = round(PRBy);
    PPz = round(PRBz);
    //Asignación del plano
    arrayMI[i][0] = PPx;
```

```
      arrayMI[i][1] = PPy;
      arrayMI[i][2] = PPz;
      arrayMI[i][3] = P + PHP;
      arrayMI[i][4] = R + PHR;
    }
    RSVP();
    break;
   case(6):
    cXY = color(0,200);     //Cambio de color en el plano
    cXZ = color(0,255,0,80);
    cYZ = color(0,0,255,100);
    PHP = -636;
    PHR = 0;
    for(int i= 0; i< ps.length; i++){
     PPx = arrayMI[i][0];
     PPy = arrayMI[i][1];
     PPz = round(SigP*pow(arrayMI[i][1]/10,2));
     P = arrayMI[i][3];
     R = arrayMI[i][4];
     //Asignación del plano
     arrayMI[i][0] = PPx;
     arrayMI[i][1] = PPy;
     arrayMI[i][2] = PPz;
     arrayMI[i][3] = P + PHP;
     arrayMI[i][4] = R + PHR;
    }
    RSVP();
    break;
   case(7):
    cXY = color(255,0,50,80);
    cXZ = color(0,255,0,80);
    cYZ = color(0,200);     //Cambio de color en el plano
    PHP = 264;
    PHR = 0;
    for(int i= 0; i< ps.length; i++){
     PPx = arrayMI[i][0];
     PPy = arrayMI[i][1];
     PPz = round(SigP*pow(arrayMI[i][1]/10,2));
     P = arrayMI[i][3];
     R = arrayMI[i][4];
     //Asignación del plano
     arrayMI[i][0] = -PPz + 600;
     arrayMI[i][1] = -PPy;
     arrayMI[i][2] = -PPx + 1500;
     arrayMI[i][3] = P + PHP;
     arrayMI[i][4] = R + PHR;
    }
    RSVP();
    break;
  }
}
void RSVP(){     //Subrutina para generar el tipo de archivo
 String[] marky = new String[13+2*(ps.length)];
 for(int i= 0; i< ps.length; i++){
  //Puntos HOME predefinidos X, Y, Z, P; y R se definen para cada caso de CASERVP
  PHX = 4000;
  PHY = 0;
  PHZ = 3000;
  PHP = PHP;
  PHR = PHR;
  //Asignación del plano para dibujar a partir del punto origen
  int Px = PHX + arrayMI[i][0];
  int Py = PHY + arrayMI[i][1];
  int Pz = PHZ + arrayMI[i][2];
  int P = arrayMI[i][3];
  int R = arrayMI[i][4];
  if(i==0){
   marky[0] = "";
   marky[1] = "$p 1 1 0 0 0 0 0 0 0";
   marky[2] = "$p 2 1 $%^&$ 465 -3 -1 -1 -3 0";
   marky[3] = "$p 3 1 0 1690 0 5042 -636 -1 -32766";                    //Home del robot
   marky[4] = "$p 4 1 P1 "+PHX+" "+PHY+" "+PHZ+" "+PHP+" "+PHR+" -32766";     //Home predefinido
   i=0;
  }
```

```
  marky[5 + i] = "$p "+(5+i)+ " 1 P"+(2+i)+" "+Px+" "+Py+" "+Pz+" "+P+" "+R+" -32766";
 }
 marky[ps.length + 5] = "$pr 1 CADR";
 marky[ps.length + 6] = "";
 marky[ps.length + 7] = "              PROGRAM   CADR";
 marky[ps.length + 8] = "              **********************";
 marky[ps.length + 9] = "CLOSE";
 marky[ps.length + 10] = "MOVE     P1";
 for(int j= 0; j< ps.length; j++){
  marky[ps.length + 11 + j] = "MOVE     P"+(2+j);
 }
 marky[11+2*(ps.length)+0] = "END";
 marky[11+2*(ps.length)+1] = "(END)";
 saveStrings( "data/CADR.CBU",marky);
}
```

Bibliografía.

[1] SANCHEZ LEON, José Antonio, *Diseño de la interfaz grafica del sitio círculos de aprendizaje,* 2005. 40p. Tesis de licenciado en diseño y comunicación visual. UNAM, Escuela nacional de artes plásticas. [En línea] http://dgb.unam.mx/ [Consulta: 20/01/2014].

[2] ANDONI, Alonso, y GALÁN Carmen, *La tecnociencia y su divulgación: un enfoque transdisciplinar*, Anthropos, 2004, 287 p.

[3] Processing 2 [En línea] http://www.processing.org/ [Consulta: 20/01/2014].

[4] Online-Convert.com [En línea] http://imagen.online-convert.com/es/convertir-a-svg [Consulta: 20/01/2014].

[5] Gráficos vectoriales de mapa de bits. [En línea] http://ocw.innova.uned.es/mm2/tcm/contenidos/pdf/tema1.pdf [Consulta: 15/02/2014].

[6] Processing Cheatsheet v002. [En línea] http://surattack.com/Processing%20Cheatsheet%20v002%20without%20insult.pdf [Consulta: 15/11/2013].

[7] Ramón, Verónica, *Inauguran laboratorio de ingeniería mecánica. El LIMAC opera con tecnología de frontera; es líder en su campo,* GACETA UNAM, Registro No.057990, 01/04/04, Pag. 1,3.