ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO EN INGENIERÍA ELECTRÓNICA

AUTORES: CASA MONTALEZA, DOUGLAS ISRAEL
COQUE CISNEROS, DANILO XAVIER

TEMA: DESARROLLO DE SOFTWARE PARA LA PROGRAMACIÓN Y OPERACIÓN DEL MANIPULADOR ROBÓTICO CRS A255

DIECTOR: ING. IBARRA, ALEXANDER MSC.
CODIRECTOR: ING. ALULEMA, DARWIN MSC.

SANGOLQUÍ, MARZO 2014

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE
INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICADO

Ing. Alexander Ibarra Msc.

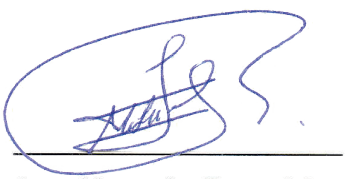
Ing. Darwin Alulema Msc.

CERTIFICAN

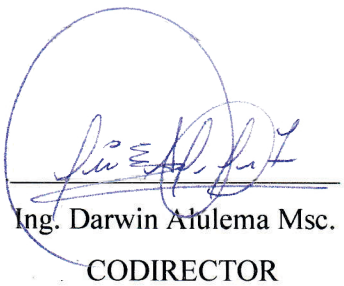
Que el trabajo titulado: "Desarrollo de software para la programación y operación del manipulador robótico CRS A255", realizado por el Sr. Douglas Israel Casa Montaleza con C.I. 1718984071 y el Sr. Danilo Xavier Coque Cisneros con C.I. 1718857103, ha sido guiado y revisado periódicamente y cumple con las normas estatutarias establecidas por la Universidad de las Fuerzas Armadas – ESPE, en el Reglamento de Estudiantes de la Escuela Politécnica del Ejército.

El mencionado trabajo consta de un documento empastado y un disco compacto el cual contiene los archivos en formato portátil de Acrobat (pdf). Se realiza la entrega de los documentos al Ingeniero Luis Orozco Msc. En su calidad de Coordinador de Carrera.

Sangolquí, 20 de Marzo del 2014



Ing. Alexander Ibarra Msc.
DIRECTOR



Ing. Darwin Alulema Msc.
CODIRECTOR

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE
INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

DECLARACIÓN DE RESPONSABILIDAD

DOUGLAS ISRAEL CASA MONTALEZA
DANILO XAVIER COQUE CISNEROS

DECLARAMOS QUE:

El proyecto de grado titulado: "Desarrollo de software para la programación y operación del manipulador robótico CRS A255", ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 20 de Marzo del 2014

Douglas Israel Casa Montaleza

Danilo Xavier Coque Cisneros

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE
INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Nosotros, Douglas Israel Casa Montaleza y Danilo Xavier Coque Cisneros

Autorizamos a la Universidad de la Fuerzas Armadas – ESPE la publicación, en la biblioteca virtual de la Institución, el proyecto de grado titulado "Desarrollo de software para la programación y operación del manipulador robótico CRS A255" cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Sangolquí, 20 de Marzo del 2014

Douglas Israel Casa Montaleza

Danilo Xavier Coque Cisneros

DEDICATORIA

Al creador del universo, por permitirme llegar a este momento tan importante de mi formación profesional, llenándome de fortaleza para aceptar las derrotas, coraje para derribar los miedos y perseverancia para alcanzar mis sueños.

A mi amada madre, el pilar fundamental de mi existencia, con su cariño y dedicación ha sabido formarme de buenos sentimientos, hábitos y valores. Llenándome cada día de su amor.

A mi amado padre, quien siempre ha velado por mi bienestar y mi educación. Ejemplo de entrega y lucha. Enseñándome que lo primordial en la vida es la familia.

A mi hermanas, quienes en todo momento han estado a mi lado consintiéndome, guiándome, corrigiéndome y sobre todo cuidándome. Juntos hemos salido adelante en los momentos más difíciles.

A mis sobrinas y sobrinos, son la razón de mis alegrías, motivación para no rendirme y continuar batallando cada día por mis metas.

A mis familiares y amigos, por acompañarme en todo momento, aconsejándome, brindándome su apoyo incondicional y por compartir buenos y malos momentos.

Con todo mi cariño y mi amor les dedico este logró de mi vida. Porque sin todos ustedes no lo hubiera alcanzado.

Douglas Israel Casa Montaleza

DEDICATORIA

A mi madre Fanicita

Por haberme apoyado con sus bendiciones y oraciones durante todo el desarrollo de este proyecto, por su presión, por su preocupación y la ayuda para que todo salga bien.

A mi padre Angelito

Por haberme inculcado valores como la responsabilidad, la dedicación al trabajo encomendado y realizar las cosas bien con ética y profesionalismo.

A mí querida esposa Patty

Por haber confiado en mí, siempre con sus palabras de ánimo, por su preocupación, por su apoyo y todo lo compartido.

A mi hermosa hija Camila

Por todo el cariño, el amor que le tengo y sea de ejemplo para su vida profesional y estudiantil.

A mis abuelitos

Por su preocupación, sus bendiciones, sus oraciones y la crianza que me han brindado.

A mis familiares

Por haberme siempre presionado para la culminación del proyecto.

Danilo Xavier Coque Cisneros

AGRADECIMIENTO

Agradezco primordialmente a mis padres por darme la vida y convertirme en un hombre de bien. Por todo el esfuerzo y sacrificio. Espero no fallarles en ningún momento de mi vida.

A mis hermanas Marjorie, Johanna y Elizabeth. Por ser como mis madres y estar pendientes en cada momento, son todo en mi vida y velaré por ustedes en cada día como la han hecho por mí.

A los Ingeniero Alexander Ibarra y Darwin Alulema, por confiar en mí y percibir el potencial profesional que puedo lograr, pero sobre todo por su valiosa amistad. Espero un día ser un excelente catedrático como lo son ustedes.

A mi amigo Danilo, por todo el arduo camino que transitamos para alcanzar este logro académico. Y su valiosa familia, por el apoyo brindado en cada momento.

A la Doctora Laura Chiriboga y la Licenciada Marianela Jaramillo. Por ayudarme en los momentos difíciles que pase, enseñándome a no rendirme y creer siempre en mí.

A la prestigiosa universidad ESPE y todos mis profesores, por sus valiosos conocimientos que día a día forjaron sabiduría y valores en mí, preparándome para la vida profesional.

A todos, que en algún momento han estado a mi lado, les agradezco por formar parte de mi vida.

Douglas Israel Casa Montaleza

AGRADECIMIENTO

En primer lugar agradezco a Dios por haberme permitido culminar una etapa más de mi vida, por toda la gracia que ha derramado sobre mí y toda mi familia.

En segundo lugar a todos mis profesores y maestros, por haber compartido todos sus conocimientos y motivación durante toda mi vida estudiantil.

Agradezco a todos mis familiares, por siempre compartir conmigo en momentos malos y buenos, siempre con sus consejos y palabras de bendición.

A los Ingenieros Alexander Ibarra y Darwin Alema, por haberme brindado su linda amistad y haber confiado en la capacidad mía y de Douglas para la realización de este proyecto.

A mi compañero de tesis Douglas Casa por haber dado todo de sí, para la culminación de este proyecto.

Danilo Xavier Coque Cisneros

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Actualmente los procesos industrializados a gran escala incorporan manipuladores robóticos con mayores prestaciones, razón por la cual, se han desarrollado aplicaciones en hardware y software para cumplir con todas las exigencias, permitiendo optimizar recursos, reduciendo tiempo de trabajo y mejorando la calidad del producto final. Por tal motivo, es importante el desarrollo de aplicaciones con tecnología abierta que puedan complementarse e innovarse con el transcurso del tiempo. El presente proyecto de investigación propone que los manipuladores robóticos CRS A255 vuelvan a estar operativos y no depender de proveedores que utilizan tecnología cerrada y software propietario. Una vez realizado un estudio y análisis previo, el proyecto se divide en dos etapas: la primera etapa es la optimización de las tarjetas de control y potencia que permiten la operación de las cinco articulaciones del manipulador robótico simultáneamente, esto se realizó mediante la implementación de algoritmos de movimientos a localizaciones en su área de trabajo; y una segunda etapa que comprende el diseño y desarrollo de un software de código abierto, seleccionando comandos que no impliquen matemática de robots, además se incorporó al software un compilador-interprete que permite la transmisión de datos desde el controlador al computador y viceversa; Dotando de varias funcionalidades al manipulador y así mejorar el aprendizaje de los estudiantes en la cátedra de robótica industrial.

PALABRAS CLAVES

- Manipulador Robótico CRS A255.
- Comunicación I2C.
- Comunicación serial.
- Compilador-intérprete de comandos.
- Algoritmo de movimiento simultaneo.

ABSTRACT

Nowadays industrial scale processes incorporate robotic manipulators with higher performance, whence, was developed applications in hardware and software to meet all requirements, allowing optimizing resources, reducing work time and improving end-product quality. For this reason, it is important applications development with open-technology that may complement and innovate by the passage of time. This research project proposes that CRS A255 robotic manipulators get back functional and not rely on provider who uses closed-technology and proprietary software. Once performed a previous analysis, this project is divided into two part: the first part is to optimize the control and power cards that allow the operation simultaneously for five robotic manipulator joints, this was done by implementing motion algorithms to locations in their work area , and a second part that includes the design and development of open-source software by selecting commands that don't involve math robots, also incorporated compiler-interpreter at the software that allows the transmission one data from the controller to the computer and manner contrary; by providing multiple functionalities at manipulator and improving the learning of students in the department of industrial robotics .

KEYWORDS

- CRS A255 Robotic Manipulator.
- I2C communication.
- Serial Communication.
- Program commands compiler-interpreter.
- Simultaneous motion algorithm.

CAPÍTULO 1

DESARROLLO DE SOFTWARE PARA LA PROGRAMACIÓN Y OPERACIÓN DEL MANIPULADOR ROBÓTICO CRS A255

1.1 ANTECEDENTES

El laboratorio de robótica del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armas "ESPE", se encuentra dotado de tres manipuladores robóticos en la configuración antropomórfica de la marca CRS Robotics en el modelo A255; actualmente, solo uno se encuentra en funcionamiento con el controlador (C500) y el software de control (RobComm) original (con el que vino de fábrica) y es el único utilizado para realizar prácticas de laboratorio para la cátedra de robótica industrial, con el transcurso del tiempo y el uso continuo de los manipuladores uno a uno se han deteriorado hasta el punto de quedar fuera de funcionamiento, al no existir ya la empresa representante de la marca de dicho manipulador no se puede realizar un mantenimiento o remplazo de partes o piezas

dañadas, para solventar este problema se inició hace un año un proyecto que comprende varias etapas basado en la habilitación del manipulador robótico con tecnología nacional creada y desarrollada por estudiantes de la carrera de Ingeniería Electrónica en Automatización y Control como proyecto de graduación, es así que se ha culminado con la primera etapa que comprendió el diseño e implementación del controlador CAD "Controlador Andrea Daniel" (operación manual mediante teach pendant) para el manipulador robótico CRS A255, para dar secuencia a este proyecto el presente perfil propone continuar con dicho trabajo implementando un software para la programación y operación del manipulador robótico CRS A255 con el fin de que los manipuladores vuelvan a estar operativos y no depender de proveedores o representantes propietarios de tecnología cerrada.

1.2 JUSTIFICACIÓN E IMPORTANCIA

Al implementar un nuevo software para la programación y operación del manipulador robótico, se puede reemplazar al programa actual el cual es el RobComm; además, al desarrollar una interfaz gráfica con software abierto permitirá a futuro que se pueda cambiar algunas características o añadir más prestaciones sin tener que pagar por licencias adicionales, teniendo de esta manera tecnología abierta al contar con códigos fuente de todo el sistema robótico, además se puede llevar un mantenimiento preventivo y correctivo de todos los componentes tanto en hardware como en software del controlador al tener toda la información pertinente para el caso.

La culminación integral del proyecto CRS A255 en todas sus etapas beneficiará relevantemente a la enseñanza de la cátedra de robótica industrial, ya que se podrá utilizar dicha tecnología para realizar las prácticas de laboratorio que la materia

requiere para complementar la fase de aprendizaje de tan importante cátedra, así mismo al culminarse este proyecto se pretende realizar una réplica a gran escala, en donde se reemplacen los controladores actuales (obsoletos) por la tecnología desarrollada.

1.3 ALCANCE DEL PROYECTO

El presente Proyecto de Tesis optimiza la operación que se realiza sobre el manipulador para su correcto funcionamiento, mediante la construcción de tarjetas electrónicas que se conectan al controlador actual (CAD) mediante comunicación I2C, adicionalmente se desarrolla una interfaz gráfica en un lenguaje de programación de alto nivel como es Java, por medio del uso del entorno de desarrollo de aplicaciones NetBeans, la misma que permite al usuario controlar al manipulador robótico mediante comandos, dicho software tiene similares características al RobComm tanto en programación online, offline y guiada/textual.

1.4 OBJETIVOS

1.4.1 Objetivo General

Desarrollar un software para la programación y operación del manipulador robótico CRS A255.

1.4.2 Objetivos Específicos

1. Optimizar el sistema de control para el movimiento del manipulador robótico en sus cinco grados de libertad.

2. Desarrollar una interfaz gráfica en lenguaje de programación Java mediante el entorno de desarrollo de aplicaciones NetBeans para controlar al manipulador robótico CRS A255.

3. Establecer una adecuada comunicación mediante el protocolo I2C para la transferencia de información entre el computador y el controlador CAD del manipulador robótico CRS A255.

4. Realizar las pruebas del correcto funcionamiento de todo el hardware y software implementado.

CAPÍTULO 2

MARCO TEÓRICO

2.1 INTRODUCCIÓN

A lo largo de la historia el ser humano ha tenido el interés de construir máquinas que imiten el movimiento de animales y del cuerpo humano, Los egipcios y griegos dotaron de partes mecánicas e hidráulicas a los brazos de las estatuas de sus dioses, los sacerdotes las operaban para simbolizar inspiración y divinidad, fascinando a los adoradores de los templos.

Siglos posteriores aparecieron precursores que diseñaron y desarrollaron muñecos mecánicos muy ingeniosos que ya incorporaban ciertas características de robots, desde entonces se amplía el concepto de robótica desarrollando mecanismos ya especializados con mayor complejidad.

"La robótica es una disciplina relativamente joven. Aunque el término robot se acuña en los años veinte del pasado siglo. La robótica industrial nace en los cincuenta y sólo en los setenta comienzan a impartirse cursos de robótica en un gran número de universidades" (Ollero, 2001, p.17).

"El uso del robot industrial junto con los sistemas de diseño asistido por computadora (CAD) y manufactura asistida por computadora (CAM), caracterizan las tendencias más recientes en la automatización del proceso de manufactura" (Craig, 2006,p.2).

2.1.1 Definiciones

"La robótica es una disciplina científica que aborda la investigación y desarrollo de una clase particular de sistemas mecánicos, denominados robots manipuladores, diseñados para realizar una amplia variedad de aplicaciones industriales, científicas, domésticas y comerciales" (Reyes, 2011, p.3).

Existen muchas definiciones de la palabra robot, destacando en cada concepto aspectos particulares que un autor resalta en su obra. Según el instituto de Robótica de Norteamérica (RIA, 1975) define a un robot industrial como un "manipulador multifuncional y reprogramable diseñado para manipular materiales, componentes, herramientas o dispositivos especializados mediante movimientos programados variables, con el fin de realizar diversas tareas".

Se puede comprender la definición de automatización o control automático a partir de la definición que el diccionario de la Real Academia de la Lengua Española (DRAE, 2001) realiza de Automática. “Disciplina que trata de los métodos y procedimientos cuya finalidad es la sustitución del operador humano por un operador artificial en la ejecución de una tarea física o mental previamente programada”.

El concepto de Cibernética proviene del griego que hace referencia a mecanismos precisos de gobierno y control, De igual forma que el concepto anterior el Diccionario de la Real Academia de la Lengua Española (DRAE, 2001) define cibernética como “El estudio de las analogías entre los sistemas de control y comunicación de los seres vivos y de las máquinas; y en particular, el de las aplicaciones de los mecanismos de regulación biológica a la tecnología.”

Conociendo estos conceptos se puede definir que un manipulador robótico es un sistema mecánico programable, con funciones similares a los miembros superiores del cuerpo humano interconectados a través de articulaciones que permiten realizar un movimiento hacia una posición deseada a otra, dentro del volumen de trabajo.

2.2 CLASIFICACIÓN

En la actualidad existe una gran variedad de robots definidos por su funcionalidad, aplicación, adaptación a un entorno. Sin embargo, se los puede clasificar de manera general. (Ver tabla. 1)

Tabla. 1 Tipos de robots.

CLASIFICACIÓN DE ROBOTS		
Móviles	Terrestres, Submarinos, aéreo-espaciales	
Humanoides	Diseño complejo	
Industriales	Brazos mecánicos	Robots manipuladores

Fuente: (Reyes, 2011, p.9).

2.2.1 Robots Móviles

Los robots móviles están provistos de diferente tipos de articulaciones, en el caso de los terrestres: de ruedas, patas u orugas; en los submarinos de equipos para navegación dentro del agua, en los aéreos ya que son aeronaves no tripuladas (UAV's) como helicópteros o pequeños aviones son operados a control remoto. Los robots móviles son utilizados para rastreo, evasión de obstáculos, traslado de objetos, entre otras aplicaciones. Se encuentran dotados de una gran variedad de sensores que proporcionan imágenes o señales electrónicas que sirven para el reconocimiento de un lugar o de un objeto específico. (Ver figura. 1)

Figura. 1 Robots móviles.
Fuente: (Platea & Gonzalez F, 2013)

2.2.2 Robots Humanoides

Conocidos también como androides, son máquinas antropomórficas, capaces de imitar algunas funciones básicas del ser humano como caminar, hablar, ver, recolectar, limpiar o transportar. Pero con limitantes en la implementación del equilibrio al desplazarse debido a su caminar bípedo. (Ver figura. 2)

Figura. 2 Robots humanoides
Fuente: (Gadgets, 2007)

2.2.3 Robots Industriales

Oficialmente la organización internacional de estandarización (ISO 8373, 1994) lo define como "un manipulador multipropósito, reprogramable y controlado automáticamente en tres o más ejes."

Los manipuladores robóticos tienen gran importancia en la industria como herramientas clave para la modernización, control y automatización de procesos destacando como ventajas: gran competitividad, productividad, eficiencia y rentabilidad de las empresas que realizan producción en masa.

"Los robots industriales trabajan sin descansar las 24 horas del día, todos los días del año, por lo que en aplicaciones industriales superan en desempeño a las personas, ya que los robots no se fatigan ni se cansan, y tienen la habilidad de repetir el proceso siempre con el mismo tiempo y la misma calidad (repetitividad)" (Reyes, 2011, p.12).

Entre las principales aplicaciones están: procesos de soldadura de punto o por arco, pintado y ensamblado de carrocerías automotrices y diversas piezas industriales; traslado de herramientas, estibado y empaquetado de materiales, a continuación se muestra un modelo actual de robot industrial. (Ver figura. 3)

Figura. 3 Manipulador robótico utilizado en la industria.
Fuente: (Kuka, 2013)

2.3 ESTRUCTURA DE LOS ROBOTS

Un manipulador industrial convencional es una cadena cinemática formada por un conjunto de eslabones o elementos interrelacionados mediante articulaciones o pares cinemáticos.

2.3.1 Articulaciones

Las articulaciones permiten la conexión y movimiento relativo (grado de libertad) entre dos eslabones consecutivos del robot. (Ver figura. 4)

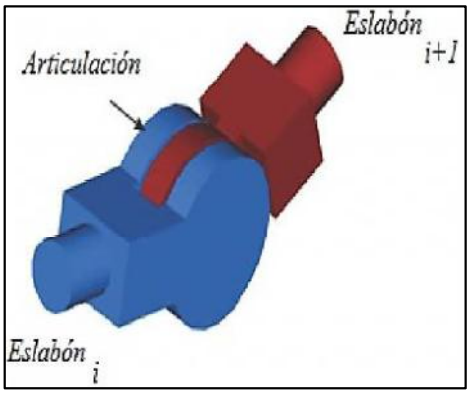
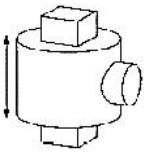
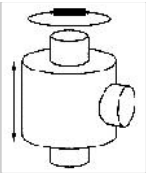
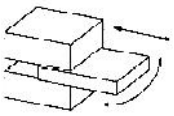


Figura. 4 Articulación de un manipulador robótico.
Fuente: (Mocencahua Mora , 2009)

Se detallan a continuación los tipos de articulaciones existentes: (Ver tabla. 2)

Tabla. 2 Tipos de articulaciones.

TIPO	CONCEPTO	GRÁFICO
Articulación Rotacional	Suministra un grado de libertad que consistente en una rotación alrededor del eje de la articulación.	
Articulación Prismática	El grado de libertad consiste en una traslación a lo largo del eje de la articulación.	
Articulación Cilíndrica	En esta articulación existen dos grados de libertad: una de rotación y una de traslación.	
Articulación Planar	Caracterizada por el movimiento de desplazamiento en un plano, tiene dos grados de libertad.	
Articulación Esférica	Con tres grados de libertad, combina tres giros en tres direcciones perpendiculares en el espacio.	

2.3.2 Estructuras Básicas

La estructura de un manipulador está compuesta por una secuencia de eslabones que tienen movimiento relativo entre sí materializados a través de las articulaciones que los unen.

El espacio de trabajo es el conjunto de puntos posibles donde puede ubicarse el efector final del manipulador. Corresponde al volumen encerrado por las superficies que determinan los puntos a los que accede el manipulador con su estructura totalmente extendida y totalmente plegada. (Ver figura. 5)

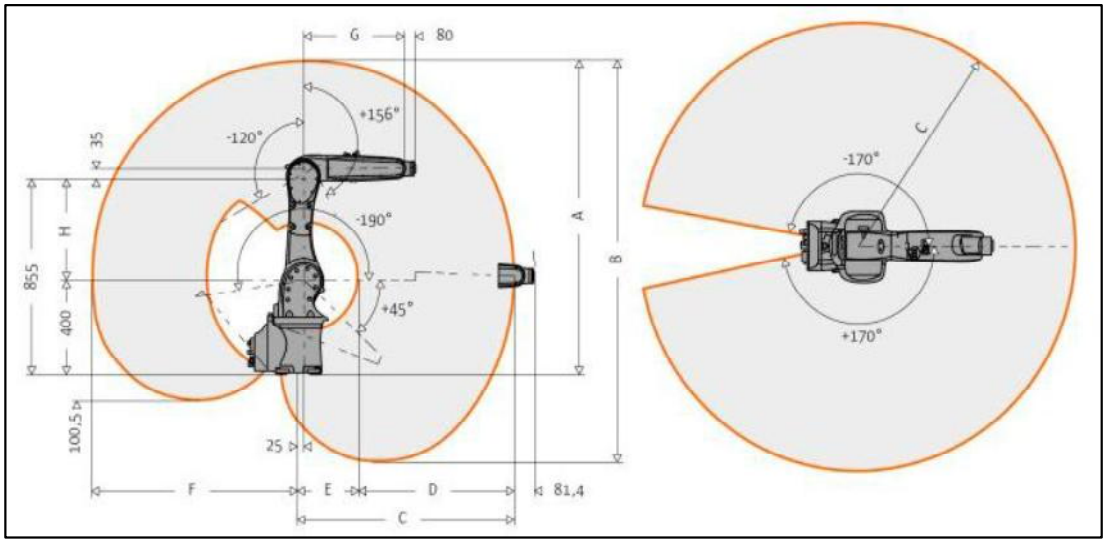


Figura. 5 Espacio de trabajo de robot KUKA Small Robots.
Fuente: (Kuka, 2013)

Todos los puntos del espacio de trabajo no tienen la misma accesibilidad. Puesto que los puntos de accesibilidad mínima son delimitados por la superficie del espacio de trabajo. A continuación se detallan las distintas configuraciones existentes:

2.3.2.1 Robot Cartesiano

- Constituida de 3 articulaciones prismáticas (PPP).
- Empleada para transporte de cargas voluminosas, con buena precisión y velocidad.
- La posición de un punto se efectúa mediante coordenadas cartesianas (x, y, z).
- Realiza una trayectoria en línea recta conocida como interpolación lineal.

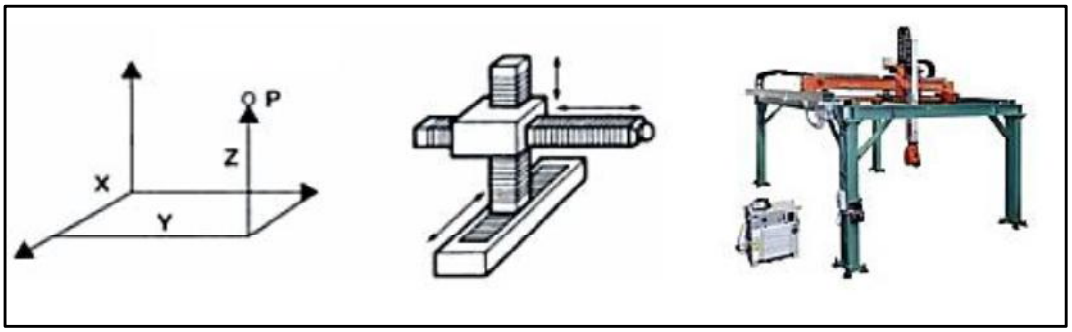


Figura. 6 Robot cartesiano.
Fuente: (Sass, 2013)

2.3.2.2 Robot Cilíndrico

- Constituida de 1 articulación rotacional y 2 prismáticas (RPP).
- Empleada para aplicaciones de tipo “Pick-and-Place”
- La posición de un punto se especifica en coordenadas cilíndricas.
- Este robot puede trabajar con varias máquinas ubicadas radialmente a su alrededor.

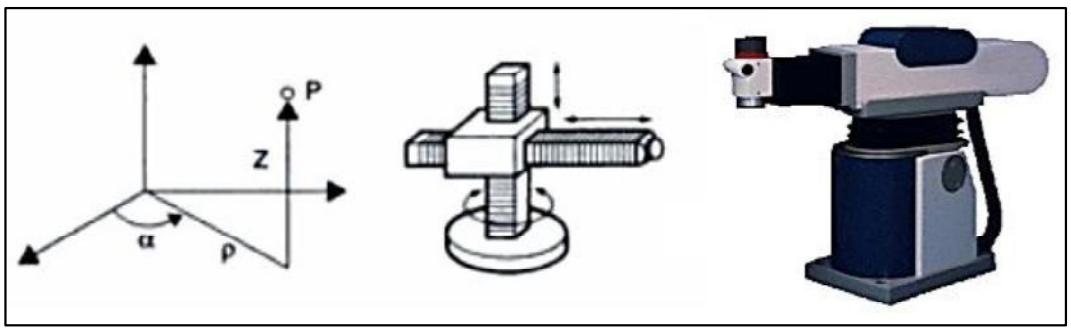


Figura. 7 Robot cilíndrico.
Fuente: (Sass, 2013)

2.3.2.3 Robot Polar o Esférico

- Consta de 2 articulaciones rotacionales y 1 prismática (RRP).
- Esta configuración permite un buen volumen de trabajo.
- La posición de un punto se especifica en coordenadas polares
- Utiliza la interpolación por articulación para moverse en sus dos primeras articulaciones y la interpolación lineal para la extensión y retracción.
- Su volumen de trabajo es el de una esfera hueca con un buen alcance.

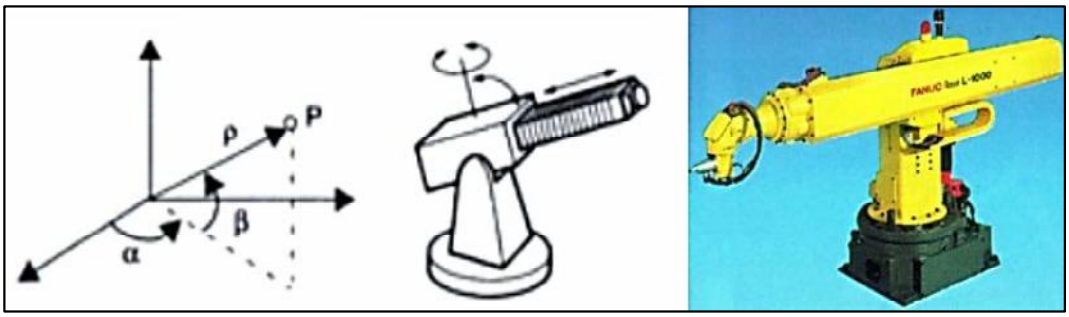


Figura. 8 Robot polar o esférico.
Fuente: (Sass, 2013)

2.3.2.4 Robot Scara

- Scara (Selected Compliance Assembly Robot Arm).
- Constituida de tres articulaciones rotaciones paralelas entre si y una de desplazamiento es sentido perpendicular al plano.
- Diseñada para realizar tareas de montaje en un plano, con gran rapidez.
- Empleada en aplicaciones de ensamblaje y de “Pick-and-Place”.

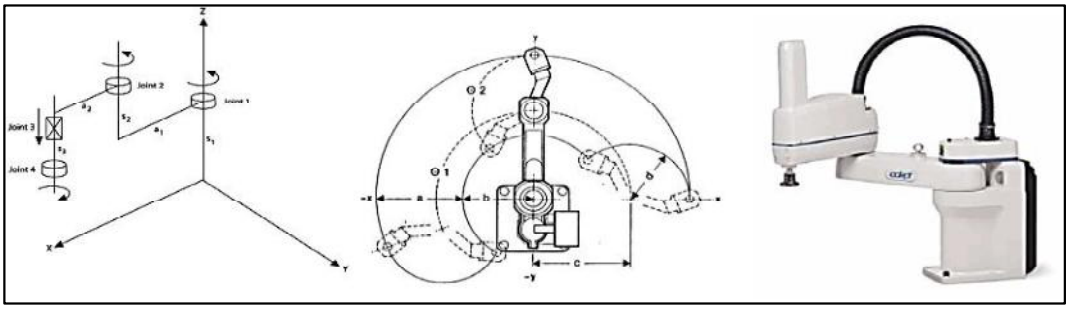


Figura. 9 Robot Scara.
Fuente: (Sass, 2013)

2.3.2.5 Robot Angular o Antropomórfico

- Constituida de 3 articulaciones rotacionales (RRR).
- Se caracteriza por la facilidad de realizar trayectorias complejas.
- La posición del extremo final se especifica en coordenadas angulares.
- Su estructura le da acceso a espacios cerrados, realiza trayectorias con alta maniobrabilidad y accesibilidad a zonas con obstáculos.
- Puede realizar movimientos de interpolación por articulación, tanto rotacional como angular.
- También puede realizar movimientos de interpolación lineal (para lo cual requiere mover simultáneamente dos tres de sus articulaciones).
- Mayormente empleado en industrias, laboratorios de investigación y desarrollo en robótica, donde es necesario el uso de máquinas para no poner en peligro al personal
- Este tipo de robots se encargan de tareas pesadas y repetitivas.

Figura. 10 Robot Angular o Antropomórfico.
Fuente: (Sass, 2013)

2.3.3 Efector Final

Es el elemento que se coloca en el extremo del último enlace del manipulador y que suministra la capacidad de agarre del objeto que se pretende manipular, tomando en cuenta: carga, fuerza de agarre, geometría y dimensiones del objeto a manejar,

tolerancia, tipos de movimientos a realizar, alimentación (neumática, eléctrica, hidráulica), tiempo de actuación del mecanismo de agarre y características de superficie de contacto. (Ollero, 2001, p.24)

Figura. 11 Tipos de efectores finales.
Fuente: (Williams, 2006)

El movimiento de un manipulador robótico provisto de una muñeca con un efector final se lo trata en dos pasos, primeramente el brazo se mueve para posicionar el extremo del último enlace y posteriormente se orienta la muñeca para que el efector final tenga la orientación adecuada.

Los tipos de pinzas más comunes son: de desplazamiento angular o pivotante y de desplazamiento lineal. En la primera los dedos de la pinza giran en relación con los puntos fijos del pivote, abriéndola o cerrándola entre sí; y la segunda, los dedos de la pinza abren o cierran ejecutando un movimiento paralelo. (Ver figura. 12)

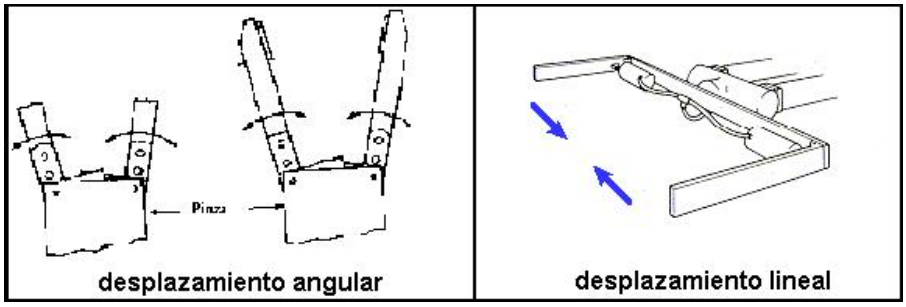


Figura. 12 Pinzas de desplazamiento angular y lineal.
Fuente: (Williams, 2006)

2.4 SENSORES

"Para conseguir que un robot realice su tarea con la adecuada precisión, velocidad e repetitividad, será preciso que tenga conocimiento tanto de su propio estado como del estado de su entorno." (Barrientos, 2007, p.36)

Para adquirir datos de la posición y velocidad de las articulaciones se utilizan sensores internos, mientras que los sensores externos son utilizados para adquirir información del entorno de un robot. A continuación se resumen los sensores más comúnmente empleados para obtener información de presencia, posición y velocidad en robots industriales. (Ver tabla. 3)

Tabla. 3 Tipos de sensores internos en robots.

SENSORES	CLASIFICACIÓN	
Presencia	Inductivo, Capacitivo, Efecto Hall, Célula Reed, Óptico, Ultrasónico, Contacto	
Posición	**Analógicos**	Potenciómetros, Resolver, Sincro, Inductosyn, LVDT
	Digitales	Encoders absolutos Encoders incrementales Regla óptica
Velocidad	Tacogeneratriz	

Fuente: (Barrientos, 2007, p.36)

2.4.1 Sensores de Posición

Para el control de posición se emplean fundamentalmente los denominados encoders y resolvers, ya que los distintos motores de los manipuladores robóticos se acoplan correctamente a este tipo de sensores.

2.4.2 Sensores de Presencia

Los sensores de presencia son capaces de detectar a un determinado objeto dentro de un radio de acción. Esta detección puede hacerse con o sin contacto con el objeto.

2.4.3 Sensores de Velocidad

> La información de la velocidad de movimiento de cada actuador se realimenta normalmente a un bucle de control análogo implementado en el propio accionador del elemento motor. No obstante, en ocasiones en las que el sistema de control del robot lo exija, la velocidad de giro de cada actuador es llevada hasta la unidad de control del robot. (Barrientos, 1997, p.42)

2.5 CONTROL DE ROBOTS

El propósito de un control de robots es el mantener un movimiento preestablecido del robot a lo largo de una trayectoria deseada, considerando las limitaciones físicas de los dispositivos actuadores. Se estudia el control de posiciones de las articulaciones, la velocidad y la posición, no solamente dependen de la señal de entrada sino también de la carga y de perturbaciones exteriores. (Sass, 2013)

2.5.1 Control Desacoplado de las Articulaciones

Considerando el desacoplamiento de las articulaciones del robot, permitiendo que un actuador únicamente tenga efecto sobre el movimiento de la articulación correspondiente.

De esta forma existe un controlador para cada articulación, el diseño del controlador para cada articulación puede hacerse utilizando las técnicas más frecuentes de diseño. (Ver figura. 13)

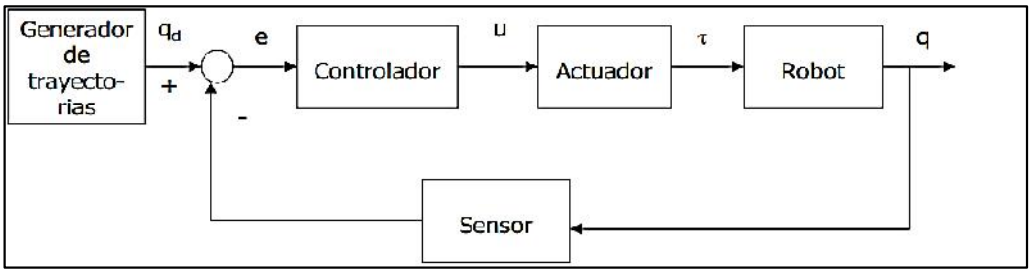


Figura. 13 Lazo de control de una sola articulación.
Fuente: (Sass, 2013)

El controlador depende de la estrategia de control aplicada ya sea un controlador P, PD, PI, PID comúnmente aplicados en robótica industrial.

Con la dificultad que la perturbación depende de cada instante de las posiciones y velocidades de las distintas articulaciones. Por consiguiente, sería necesario ajustar un valor diferente de las constantes Kp, Kd. y Ki, dependiendo del instante considerado.

2.5.2 Control Basado en el Modelo Dinámico

Requerida cuando el robot ejecuta movimientos a alta velocidad o cuando el robot tiene una estructura flexible.

Este modelo sirve para generar una señal de control que compensa las aceleraciones y los pares dinámicos “control por el par computado”. Permitiendo compensar gravedad, aceleraciones centrífugas y de efecto Coriolis.

Los métodos estudiados anteriormente tienen algunos problemas:

- En técnicas de desacoplamiento, los parámetros adecuados del controlador dependen mucho de la configuración.
- En técnicas basadas en el modelo dinámico, los problemas aparecen en la obtención de modelos precisos (fricción e inercia son parámetros difíciles de estimar).

- Una solución sería de cambiar los parámetros en función de la configuración del robot, llevando al control adaptativo.

- Otra solución consiste en mejorar el comportamiento del sistema en repeticiones sucesivas de las operaciones, llevando al control con aprendizaje.

2.5.3 Control Adaptativo

2.5.3.1 Alternativa 1

- Consiste en dividir el espacio de trabajo en regiones y calcular las ganancias (coeficientes) más apropiadas para cada una de ellas.
- Implementando una tabla teniendo como entradas, intervalos discretizados de las variables, que determinan la pertenencia a una región.
- Repetidamente el sistema de control busca la región la cual se encuentra el robot y asigna las ganancias correspondientes al controlador.
- Las regiones se definen por valores de variables articularias y por la carga.
- Requiere muchas regiones para poder tener una buena precisión.
- Cada configuración ajusta 3 parámetros (en caso de controlador PID) para cada articulación.

2.5.3.2 Alternativa 2: Modelo de Referencia

- Consiste en usar una ley de adaptación más para cambiar los parámetros del controlador. Esta ley de adaptación del modelo de referencia adapta los parámetros para que las variables articularias reales reproduzcan las señales generadas por un modelo de referencia.

2.5.3.3 Alternativa 3: Par Computado Adaptativo

Consiste en estimar la matriz de masa, los pares y aceleraciones centrífugas, de efecto Coriolis, de rozamiento y gravitacionales, para minimizar el error de seguimiento.

2.5.4 Control con Aprendizaje

Consiste en mejorar el comportamiento del sistema de control en repeticiones sucesivas de las operaciones. Si una parte del modelo es conocida se calcula un par de control, añadiendo otro par calculado con un modelo que se ajusta mediante una ley de aprendizaje en repeticiones sucesivas de la misma operación.

2.5.5 Control en el Espacio Cartesiano

En los anteriores casos, se considera que se conoce las trayectorias articularias deseadas. En la realidad, para ello requiere un generador de trayectoria, el cual permite obtener trayectorias articularias a partir de la trayectoria deseada en el espacio cartesiano. (Sass, 2013)

2.5.6 Control de Esfuerzos

- De gran interés en aplicaciones en las que el manipulador debe mantener contacto o aplicar fuerzas.
- Se requiere sensores de esfuerzos para medir diferentes fuerzas en el espacio de la tarea.
- Cuando el robot está aplicando una fuerza sobre el entorno, no hay movimientos y los motores deben proveer un torque para compensar la gravedad y aplicar las fuerzas requeridas.

2.6 PROGRAMACIÓN DE ROBOTS

La programación de robots industriales trata de enseñar al manipulador robótico su ciclo de trabajo.

Involucra los siguientes parámetros:

- Definir los movimientos que cumple el robot entre diferentes posiciones de trabajo, moviendo el robot en las posiciones deseadas y memorizándolas.
- Interpretar la información dada por los sensores.
- Actuar del efector final.
- Mandar señales de control a otros equipos.
- Comunicar con otros equipos y tomar decisiones con respecto al ciclo de trabajo, requiriendo un sistema de programación más avanzado.

Existen diferentes niveles de abstracción para definir la tarea de un robot:

- A nivel de actuadores, en términos de coordenadas articularias.
- A nivel del efector final, en términos de coordenadas operacionales.

- A nivel de los objetos, en términos de operaciones a realizar con los objetos (por ejemplo: ensamblar, alinear, entre otros.)
- A nivel de metas, en términos del objetivo de la tarea (por ejemplo: empaquetar varias botellas de agua).

2.6.1 Programación por Aprendizaje

Consiste en desplazar un sistema de referencia asociado efector final del robot de forma que se alcancen las configuraciones deseadas a la vez que se registran sus valores. (Sass, 2013)

Se clasifica en:

- **Guiado Pasivo:** Con sus manos, el programador posiciona el robot en la configuración deseada y se registran las coordenadas.

- **Guiado Activo:** Guiado mediante un puesto de mando ("teach pendant"), tal como un teclado, una botonera de programación, un joystick. Movimientos limitados a trayectorias sencillas (punto a punto, línea, rectas, arcos de círculo.)

- **Guiado con réplica del robot:** también pasivo, con la idea de mover una réplica del robot más ligero y más fácil de mover. La réplica puede tener una estructura diferente pero ello implica usar modelos cinemáticos del robot y de la réplica para realizar las transformaciones oportunas.

- **Guiado con robot virtual:** usando una computadora y un modelo del robot.

El control de la velocidad es fundamental, se usan velocidades lentas cuando el robot está cerca de obstáculos u objetos a manipular. Las velocidades más altas se usan en movimientos sin riesgo de colisión o "freeways".

Es difícil evaluar la velocidad lineal del efector final porque depende del número de articulación moviéndose, de la configuración del robot y de la carga. El modelo dinámico permite obtener esa velocidad pero al costo de muchos cálculos matemáticos.

2.6.2 Programación Textual

Se programan robots mezclando el uso de un guiado, para definir los puntos de trabajo, y un lenguaje de programación, para la lógica y la coordinación de la tarea.

Los lenguajes de programación permiten tratar señales de sensores analógicos y digitales, realizar cálculos complejos, comunicar con el entorno, controlar movimientos complejos. (Sass, 2013)

Ha existido una gran variedad de lenguajes de programación, pero no existe una normativa, la gran mayoría tiene su propio lenguaje de programación.

Primera generación (lenguajes primitivos)

- Con especificación de una secuencia de movimientos.
- Interpolación lineal entre puntos de trabajo.
- Tratamiento de señales de sensores digitales.
- Acción de apertura y cierre de pinza o efector final

- Ejemplo de instrucciones comunes: MOVE, WAIT, SIGNAL.
- Limitaciones: sin cálculos complejos, sin sensores analógicos, comunicación limitada con otros dispositivos, no puede ser extendido.

Segunda generación (lenguajes estructurados)

- Con movimientos más complejos: círculos, parábolas, trayectorias no lineales.
- Sensores analógicos y mandos analógicos.
- Uso de estructuras de datos más complejos.
- Uso de sistemas de referencia con transformaciones de coordenadas.
- Uso de estructura de programación: IF THEN ELSE, WHILE DO, FOR.
- Uso de subrutinas y de procesos paralelos o sincronizados
- Comunicación más avanzada con otros dispositivos. Por ejemplo, para memorizar información y controlar la actividad del robot. Permite extensión según los deseos del usuario.

2.6.3 Programación CAD

Este tipo de programación CAD (Diseño asistido por computador) usa un ambiente completamente virtual para diseñar una celda robótica, simularla y optimizarla. Tiene las siguientes características:

- Escoger el robot, tomando en cuenta aspectos ambientales.
- Concibe dispositivos de peri-robótica (efectores, alimentación y evacuación de productos, sensores).
- Programación de la celda de robótica

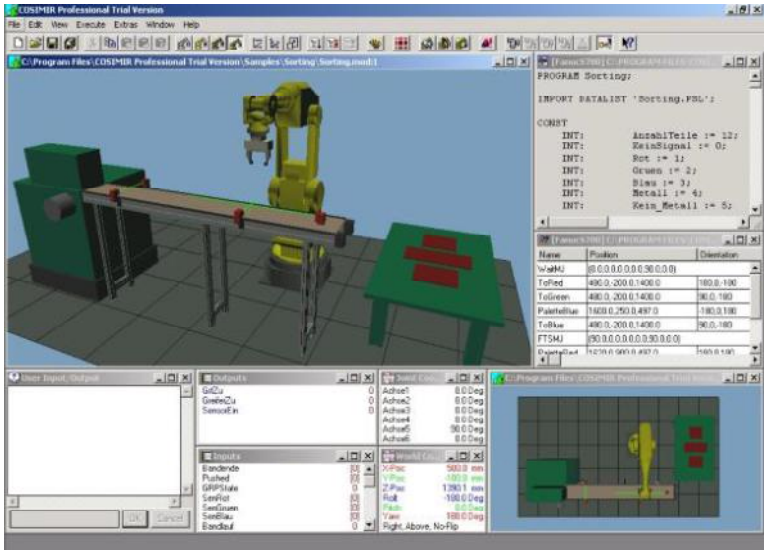


Figura. 14 Programación de diseño asistido por computador (CAD). Fuente: (Sass, 2013)

La mayoría de los programas actuales disponen de librerías de robots de diferentes marcas. Por ejemplo: COSIMIR ofrece robots KUKA, MITSUBISHI, ABB, FANUC, REIS, STAUBLI, ADEPT, MANUTEC, NIKO, VW. (Ver figura. 14)

A continuación se realiza una comparación de la programación CAD con otros tipos de programación. (Ver tabla. 4)

Tabla. 4 Comparación de la programación CAD con otros lenguajes.

LENGUAJES	CAD	OTROS LENGUAJES
Independencia Robot/programa	Totalmente	Parcialmente
Puesta en práctica	Transparencia con respecto al robot.	Se debe aprender tanto las características del robot y el lenguaje de programación utilizado
Errores de trayectorias	Debidas a errores en el modelo y errores de los lenguajes	Debidas a los cambios de bases y a los errores de repetición
Costo	Alto	Un costo racional

Fuente: (Reyes Cortés, 2011)

Existen algunas desventajas en este tipo de Programación:

- Imperfecciones del mundo real vs. el modelo perfecto (fricción, rigidez, tolerancias geométricas y de cambio de base.)
- Modelo del sistema de control diferente al del constructor: modelos del entorno, modelo de sensores y representación de informaciones no son geométricas.

2.7 MANIPULADOR ROBÓTICO CRS A255 Y CONTROLADOR CAD

El manipulador robótico CRS A255 posee una configuración antropomórfica es decir anatómicamente similar a la extremidad superior del cuerpo humano, está constituido por cinco grados de libertad o articulaciones que son cadera, hombro, codo, muñeca y su rotador.

Adicionalmente cuenta con un efector final o gripper, el cual le permite manipular diferentes tipos de objetos. (Ver figura. 15)

Figura. 15 Articulaciones del manipulador robótico CRS A255.
Fuente: (Maldonado & Sánchez, 2013)

A continuación se detallan las características principales del manipular robótico CRS A255. (Ver tabla. 5)

Tabla. 5 Principales características del manipulador robótico CRS A255.

PRINCIPALES CARACTERÍSTICAS	
Estructura	Cinco grados de libertad
Sistema de Accionamiento	Motores DC electromecánicos
Peso total	19 Kilogramos
Frenos	Un freno por articulación con excepción de la primera articulación.
Conexión efector final	Accionamiento eléctrico - neumático
Carga útil nominal	1 kilogramos

Fuente: (Maldonado & Sánchez, 2013)

2.7.1 Espacio de Trabajo del Manipulador Robótico CRS A255

Su espacio de trabajo está dado en función de su alcance y desplazamiento por el extremo de la muñeca descartando su efector final.

Puede desplazarse hasta 350° grados de forma radial y en forma vertical puede tener un desplazamiento hasta 150 grados aproximadamente. Como se aprecia a continuación. (Ver figura. 16)

El espacio de trabajo permite al operario tomar las precauciones necesarias para el acceso y manipulación del mismo sin exponerse a accidentes de cualquier tipo para el operario como del manipulador robótico.

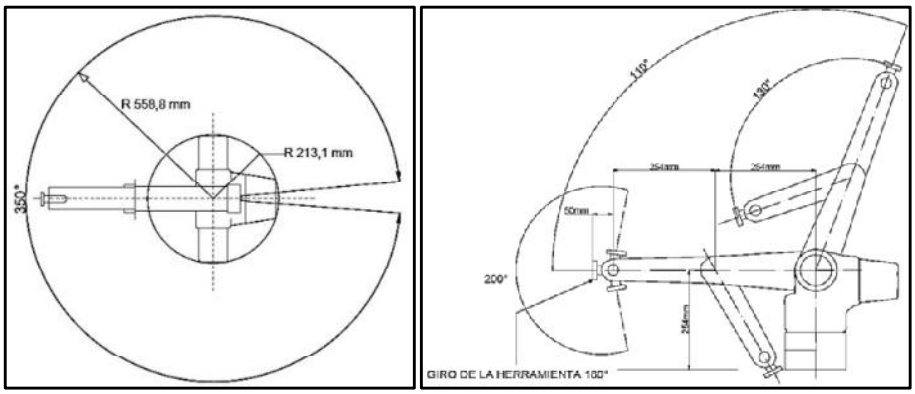
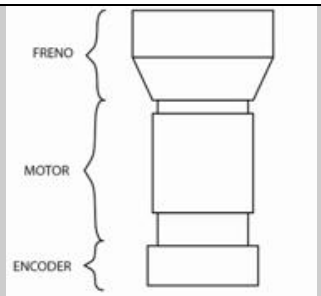
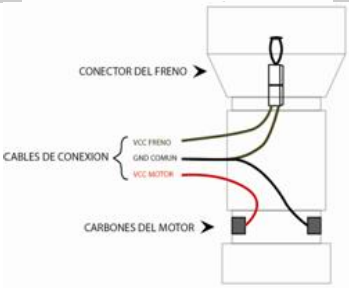
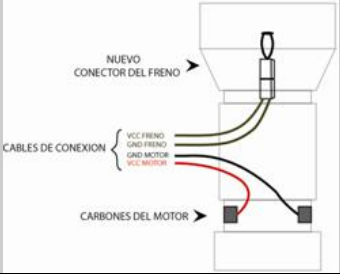


Figura. 16 Espacio de trabajo radial y vertical del manipulador robótico. Fuente: (Maldonado & Sánchez, 2013)

2.7.2 Activación de Frenos

El controlador CAD tiene una configuración diferente al controlador C500 para la desactivación de los frenos como se describe a continuación. (Ver tabla. 6)

Tabla. 6 Comparación del motor en los controladores C500 y CAD.

CARACTERÍSTICAS		GRÁFICO
Estructura de los motores del manipulador robótico CRS A255.	El motor está constituido por un encoder de tipo incremental y un solenoide para su freno.	
Configuración de cableado de motor y freno con el controlador C500.	Para activar el freno era necesario enviar el voltaje de activación debido a que la señal de referencia (GND) era la misma del motor.	
Configuración de cableado de motor y freno con el controlador CAD.	Existen dos señales de referencia (GND), una para alimentación del freno y otra para alimentación del motor.	

2.7.2.1 Encoders Incrementales

Los codificadores ópticos o encoders incrementales constan de un disco transparente con una serie de marcas colocadas radialmente y equidistantes entre sí; de un sistema de iluminación en el que la luz es colimada de forma correcta, y de un elemento fotoreceptor. El eje cuya posición se quiere medir va acoplado al disco trasparente. A medida que el eje gira se generan pulsos en el receptor, cada vez que la luz atraviese cada marca, y llevando una cuenta de estos pulsos es posible conocer la posición del eje.

El manipulador robótico CRS A255 cuenta con encoders por cada motor, los cuales permiten conocer la posición final del mismo al ser manipulado, esto se logra seleccionando una posición inicial de cada articulación e ir contando pulsos en sentido horario y anti horario.

Cuentan con tres canales de salida (A, B, Z), los canales A y B se encuentran desfasados en una porción de ciclo y el canal Z es el encargado de indicar cada vuelta que da el motor, entregando una señal TTL (Lógica de transistor a transistor), con resolución de 1000 pulsos por revolución.

2.7.3 Controlador del Manipulador Robótico CRS A255

El controlador que actualmente gobierna al manipulador robótico CRS A255 no es el de fábrica, sino que es la primera etapa de actualización del controlador. Se diseñó e implementó un nuevo controlador denominado CAD (Controlador Andrea Daniel). Su estudio y análisis está constituido por tres etapas en su diseño, que se detalla a continuación.

2.7.3.1 Etapa de Alimentación

Encargada de suministrar el voltaje y amperaje necesarios para el correcto funcionamiento del sistema, tomando la alimentación proveniente de la red eléctrica y acoplándola a los requerimientos del sistema. El sistema implementado cuenta con dos fuentes independientes. (Ver figura 17)

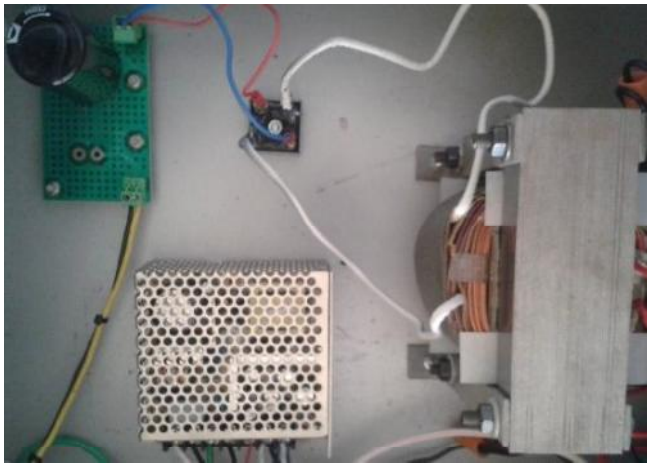


Figura. 17 Tarjeta de alimentación del controlador CAD.

En el siguiente diagrama se presenta funcionamiento de la tarjeta de alimentación del controlador CAD. (Ver figura. 18)

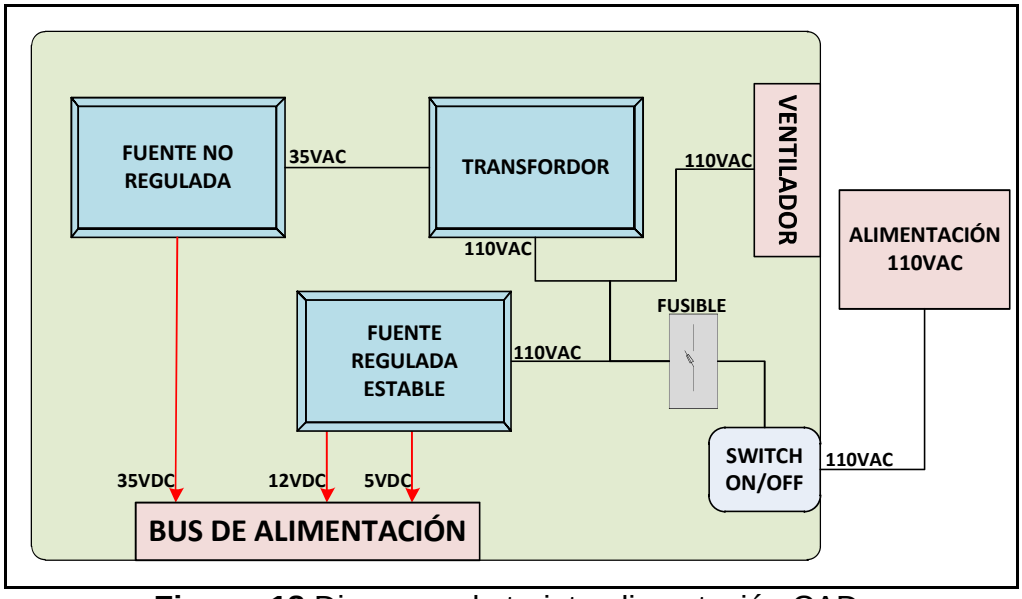


Figura. 18 Diagrama de tarjeta alimentación CAD.

La primera es una fuente no regulada que entrega un voltaje de +35VDC a 11 A, la cual es utilizada para alimentar los motores y frenos del manipulador robótico.

La segunda es una fuente regulada estable que entrega voltajes de +5VDC, +12VDC a 2 A cada una, encargada de alimentar a los circuitos integrados que disparan a los mosfet de la tarjeta de potencia y alimentar a la tarjeta de control.

A continuación se detalla el consumo de voltaje y corriente de todos los elementos electrónicos que conforma el controlador CAD. (Ver tabla. 7)

Tabla. 7 Consumo de voltaje y corriente de elementos electrónicos.

VOLTAJE	ELEMENTOS	CONSUMO DE CORRIENTE [mA]	CANTIDAD
5V	Microcontroladores (PIC16F877A)	250	4
	Compuertas AND (74HC08)	15	3
	Compuerta Inversora (74LS14)	15	3
	(MAX232)	15	1
	LCD 16x2	260	1
	LCD 16x4	260	1
	Driver para MOSFET (IR2110)	250	10
	Encoder	80	5
12V	Relé	250	2
	MOSFET (IRF530)	1	20
	Solenoide	130	1
35V	Motor	2000	5
	Freno	200	4

Fuente: (Maldonado & Sánchez, 2013)

2.7.3.2 Etapa de Potencia

Encargada de acondicionar las señales para poder accionar los motores y el solenoide desde el controlador. También de proveer la potencia necesaria para manejar un gripper neumático.

Se debe tomar en cuenta los requerimientos eléctricos de cada motor que son 35 VDC a 2 A. La tarjeta de potencia del controlador CAD se presenta a continuación. (Ver figura. 19)

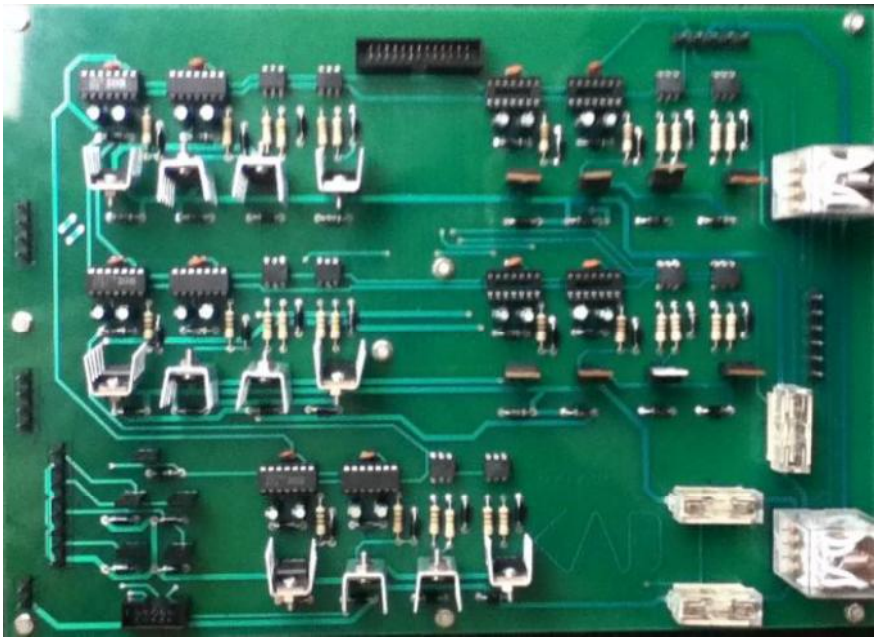


Figura. 19 Tarjeta de potencia del controlador CAD.
Fuente: (Maldonado & Sánchez, 2013)

La activación de los relés se realiza mediante el pulsador de ARM POWER y la desactivación del mismo se suscita al presionar el paro de emergencia. El circuito de enclavamiento de los relés se muestra a continuación. (Ver figura. 20)

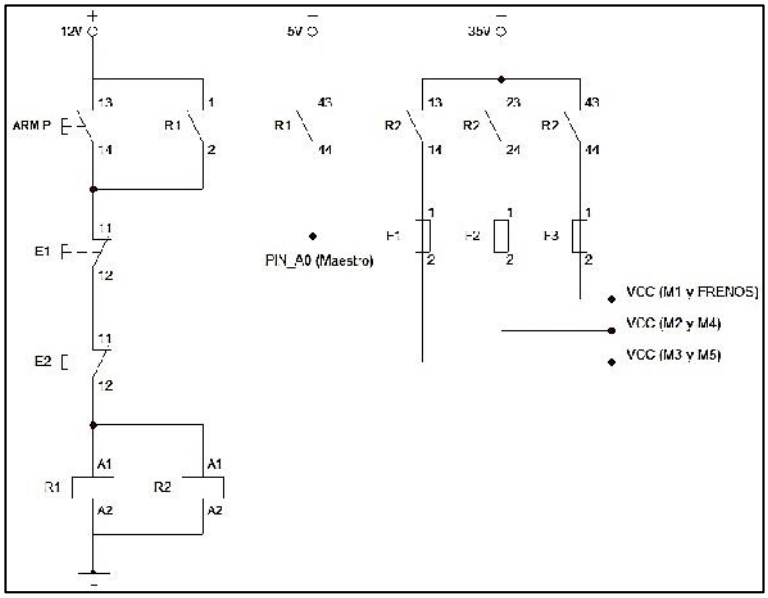


Figura. 20 Circuito de enclavamiento.

En el siguiente diagrama se presenta funcionamiento de la tarjeta de potencia del controlador CAD. (Ver figura. 21)

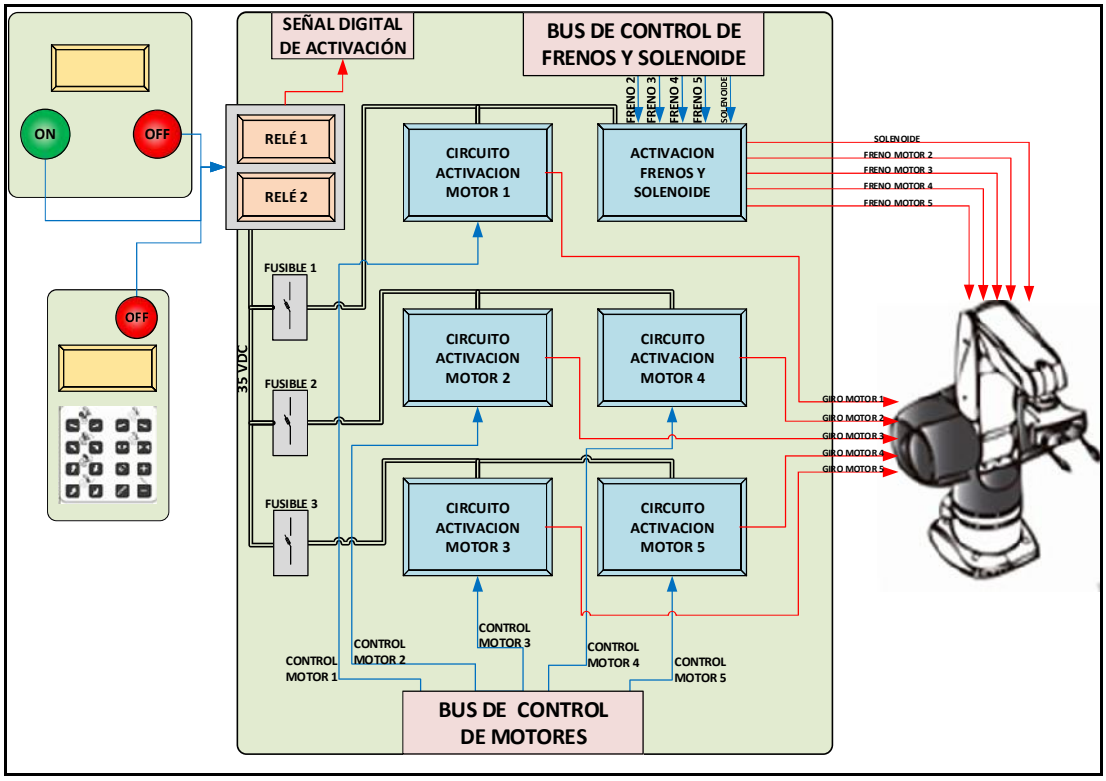


Figura. 21 Diagrama de funcionamiento de la tarjeta de potencia.

Los relés son los encargados de suministrar la alimentación a los fusibles y dar una señal lógica a la tarjeta de control.

Los tres fusibles sirven de protección contra posibles cortocircuitos y sobrecargas durante su operación, el “fusible 1” protege el circuito de activación del motor uno y frenos; el “fusible 2” protege el circuito de activación de motor dos y motor 4; el “fusible 3”protege el circuito de activación del motor tres y motor cinco.

El circuito de activación de motor se encarga de realizar el cambio giro y proporcionar el voltaje deseado para el movimiento del mismo. A continuación en la se muestra el diagrama del circuito de activación de un motor. (Ver figura. 22)

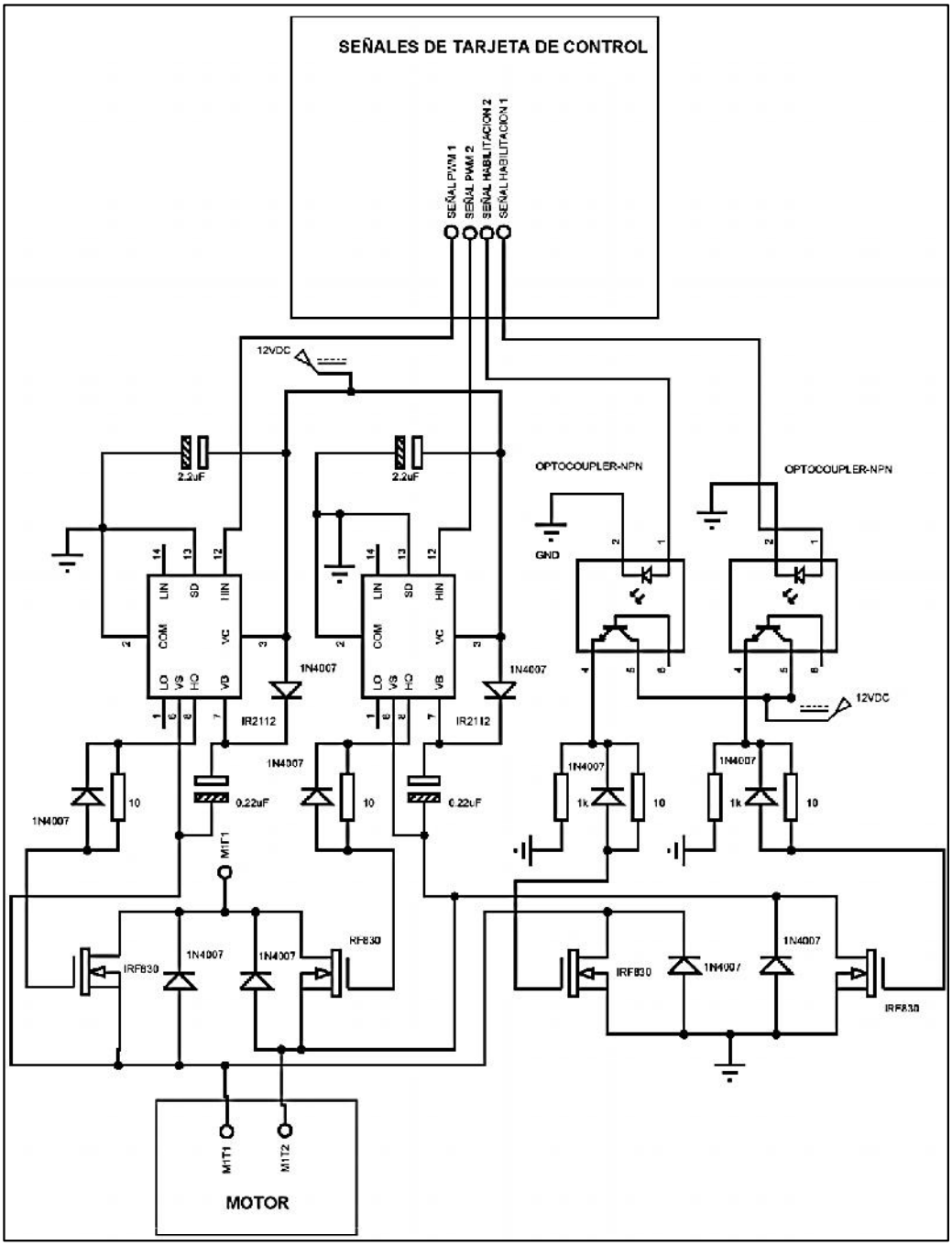


Figura. 22 Circuito de activación de un motor.
Fuente: (Maldonado & Sánchez, 2013)

Para el funcionamiento en un sentido se debe activar la SEÑAL DE PWM 1 con la SEÑAL DE ACTIVACIÓN 1, y para cambio de sentido estas señales anteriores se desactivan y se activan la SEÑAL DE PWM 2 con la SEÑAL DE ACTIVACIÓN 2.

Todos los motores se activan con el mismo funcionamiento de circuito; A diferencia del motor uno, todas los demás motores necesitan la desactivación del freno correspondiente, a continuación se muestra el circuito de uno de ellos. (Ver figura. 23)

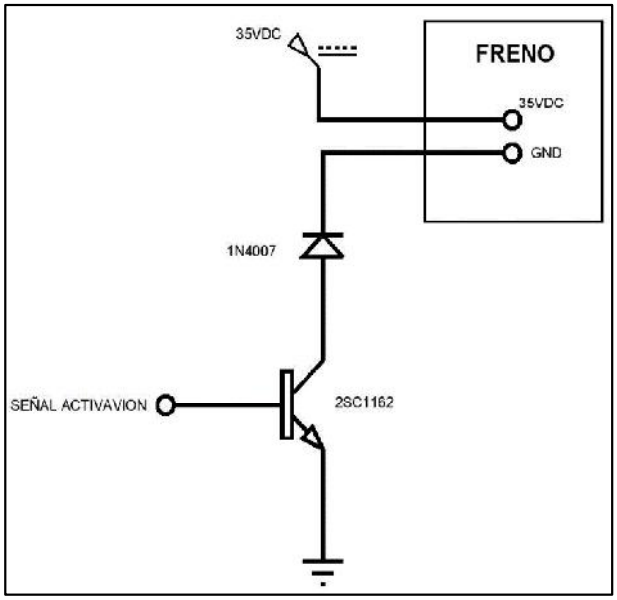


Figura. 23 Circuito de freno de un motor.
Fuente: (Maldonado & Sánchez, 2013)

El freno de un motor se inhabilita con un transistor NPN, que funciona en modo corto y saturación, al funcionar en saturación el transistor conduce entre los pines de colector y emisor, permitiendo polarizar a la bobina del freno para el movimiento del motor, en corte desconecta un terminal de la referencia de la bobina, lo cual bloquea el movimiento de un motor.

2.7.3.3 Etapa de Control

El controlador CAD controla a tres de los cinco grados de libertad que posee el manipulador robótico CRS A255, pero se encuentra implementada para poder controlar todas sus articulaciones y un gripper que funciona en base a un solenoide.

El sistema de control tiene una configuración “maestro-esclavo”. El maestro realiza una comunicación bidireccional con el computador, además gestiona la comunicación con los esclavos. Los esclavos son los encargados de proveer las señales necesarias para el circuito de accionamiento de cada motor en la tarjeta de potencia.

Los microcontroladores implementados son 4 PIC’s 16F877A, que por medio de una comunicación I2C (Interfaz de Comunicación entre Circuitos) permite interactuar al maestro como sus tres esclavos. Y por RS232 (Protocolo Estándar de Comunicación Serial) establece una comunicación con un ordenador. La tarjeta de control del controlador CAD se presenta a continuación. (Ver figura. 24)

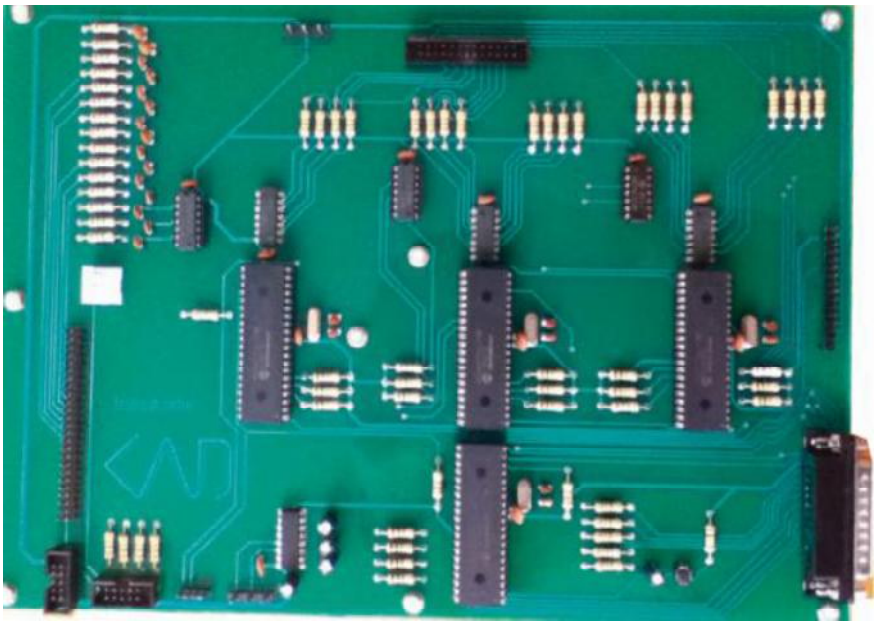


Figura. 24 Tarjeta de control del controlador CAD.
Fuente: (Maldonado & Sánchez, 2013)

En el siguiente diagrama se presenta el funcionamiento de la tarjeta de control del controlador CAD. (Ver figura. 25)

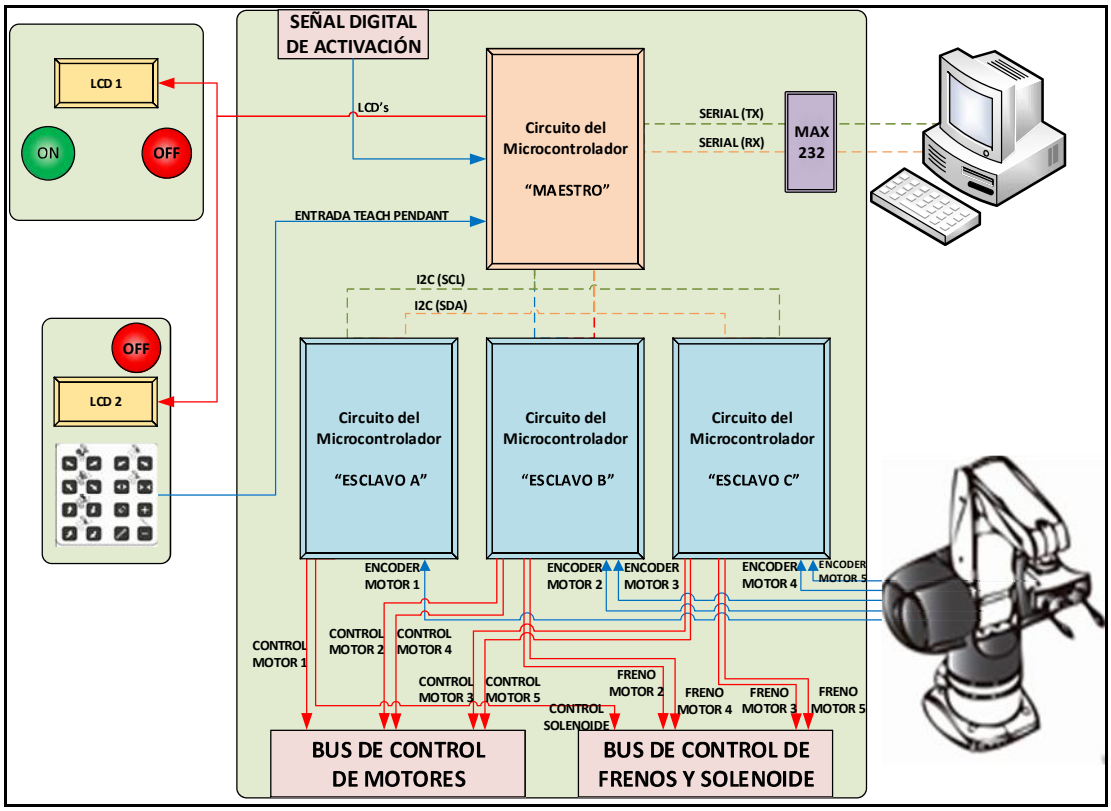


Figura. 25 Diagrama de funcionamiento de la tarjeta de potencia.

El microcontrolador "maestro" además de gestionar las comunicaciones entre los esclavos y PC; se encarga de manejar periféricos de entrada (TEACH PENDANT) y salida (LCD's). Para conocer el movimiento realizado de un motor cada microcontrolador ESCLAVO cuenta con señales de retroalimentación de sensores (encoder incremental).

A continuación se presenta el circuito un microcontrolador "esclavo" para el control de un motor. (Ver figura. 26)

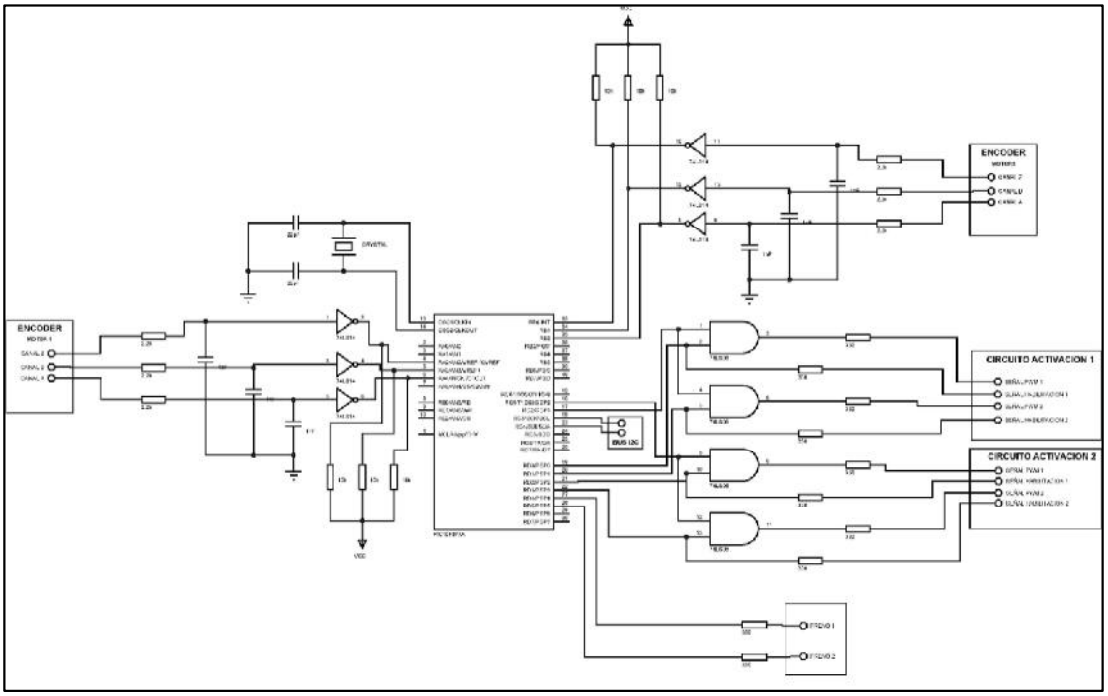


Figura. 26 Microcontrolador "esclavo" para el control de un motor.

El microcontrolador activa los circuitos de potencia de dos motores, Para el control de giro de cada motor utiliza tres señales (PWM, sentido CW y sentido CCW), por medio de dos compuertas AND se conmuta la señal de PWM para controlar el cambio de giro.

Las señales de entrada de encoders cuentan con un circuito de filtraje y de inversión de disparo por medio de un circuito integrado 74LS14.

2.7.3.4 Estructura Externa del Controlador CAD (Chasis)

El panel frontal del chasis se encuentra todos los elementos que permite el manejo del sistema como tal, es decir, el selector de modo de operación, pulsador de ARM POWER, botón de paro de emergencia, LCD indicador, y un conector necesario para la comunicación con el Teach Pendant.

En el panel posterior tiene el botón de encendido del controlador, los conectores para los cables de potencia y de alimentación, así como dos conectores DB9: uno para la conexión RS232 con el computador y el otro para señales de los frenos.

El chasis del Controlador CAD se presenta a continuación. (Ver figura. 27)

Figura. 27 Panel frontal y panel posterior del chasis del controlador CAD. Fuente: (Maldonado & Sánchez, 2013)

2.7.4 Teach Pendant

El teach pendant es el primer modo de trabajo con el que se puede manipular el brazo robótico manualmente. Contiene un conector DB-25 en la parte inferior para su conexión hacia el controlador. En su parte frontal cuenta con: LCD, un pulsador para paro de emergencia, un teclado matricial 4X4 para la manipulación de las 5 articulaciones.

El teach pendant del controlador CAD se presenta a continuación. (Ver figura. 28)

Figura. 28 Teach pendant del controlador CAD.
Fuente: (Maldonado & Sánchez, 2013)

2.7.5 Interfaz de Usuario

NetBeans IDE es un entorno de desarrollo integrado escrito en Java. Con un gran número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

A diferencia del software RobComm utilizado para el controlador C500; el controlador CAD implementa un programa en NetBeans, que permite comunicación con un puerto externo del computador. Con la ventaja de ser un software libre permitiendo que su programación sea manipulada y modificada para beneficio de nuevas innovaciones e implementaciones.

El programa actual en su primera versión tiene las características del teach pendant, con botones para el movimiento de tres de las articulaciones, y, apertura y cierre del gripper, así como también para el incremento y disminución de la velocidad, indicadores de la posición actual del robot, indicadores de modo de manejo (Teach Pendant o Computador) y la velocidad.

Realiza el envío de información desde el computador hacia el controlador para que se ejecute el movimiento de un motor a la vez mientras se mantenga presionado el botón correspondiente.

El programa también permite un monitoreo del modo de operación que está seleccionado en el controlador (computador o teach pendant). A continuación se muestra la interfaz gráfica. (Ver figura. 29)

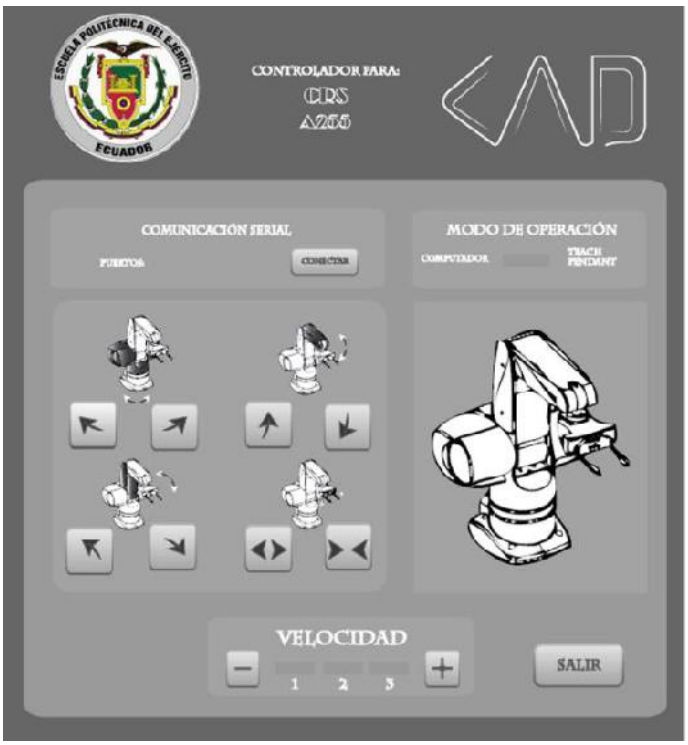


Figura. 29 HMI actual del controlador CAD V1.0.
Fuente: (Maldonado & Sánchez, 2013)

2.7.6 Software RobComm.

RobComm es un software propietario, (utilizado por el antiguo controlador C500, para maniobrar al manipulador robótico CRS A255). Es un programa completo en cuanto a todas las aplicaciones del manipulador robótico se refiere.

Sin embargo, no se tiene el código fuente del mismo. Lo que dificulta la posibilidad de adaptar el nuevo controlador a este entorno de trabajo y programación. A continuación se muestra los comandos pertenecientes a RobComm. (Ver tabla. 8). El funcionamiento de cada comando de RobComm se encuentra en el Anexo. 1.

Tabla. 8 Comandos de RobComm.

COMANDO DE ROBCOMM				
Control de Programa	**Localización**	**Movimiento**		**Operadores**
ABORT	HERE	APPRO	POSE	==
IF	NOTEACH	DEPART	MI	=!
IFSIG	TEACH	JOG	MA	>
IGNORE	MANUAL	JOINT	CAL	<
GOSUB	NOMANUAL	FINISH	CALRDY	>=
GOTO	BASE	MOTOR	Z	<=
PAUSE	DLOCN	MOVE	Y	GRIPPER
RETURN	POINT	XREADY	Z	OPEN
STOP	SET	READY	YAWX	CLOSE
DELAY	SHIFT	SPEED	PITCH	LIMP
IFSTART	SHIFTA	HOME	ROLL	LOCK
ONSIG	TOOL	CALZC	REACH	NOLIMP
ONSTART	ACTUAL	HOMEZC	ELBOW	UNLOCK
WAIT	WITH	XZERO	REACH	ALING
NEXT	ELBOW	CPATH	REMOTE	+
PROCCED	INVERT	CTPATH	CIRCLE	-
RETRY	REACH	GOPATH	ONLINE	++
RUN	POSE	ELBOW	INVERT	--
ONERR	@LOCATE			=
IFSTRING				
HALT				
IFPOWER				
ONPOWER				

Fuente: (Chacón, 2005)

La interfaz gráfica del software RobComm, tiene dos formas de programación: la primera es por medio de código secuencial el cual reconoce una sucesión de comandos, los compila y los ejecuta el manipulador robótico al correr el programa.

La segunda es por medio del modo terminal, el cual una vez realizada la comunicación con el manipulador robótico este ejecuta un comando a la vez. A continuación se muestra la interfaz gráfica del programa RobComm. (Ver figura. 30)

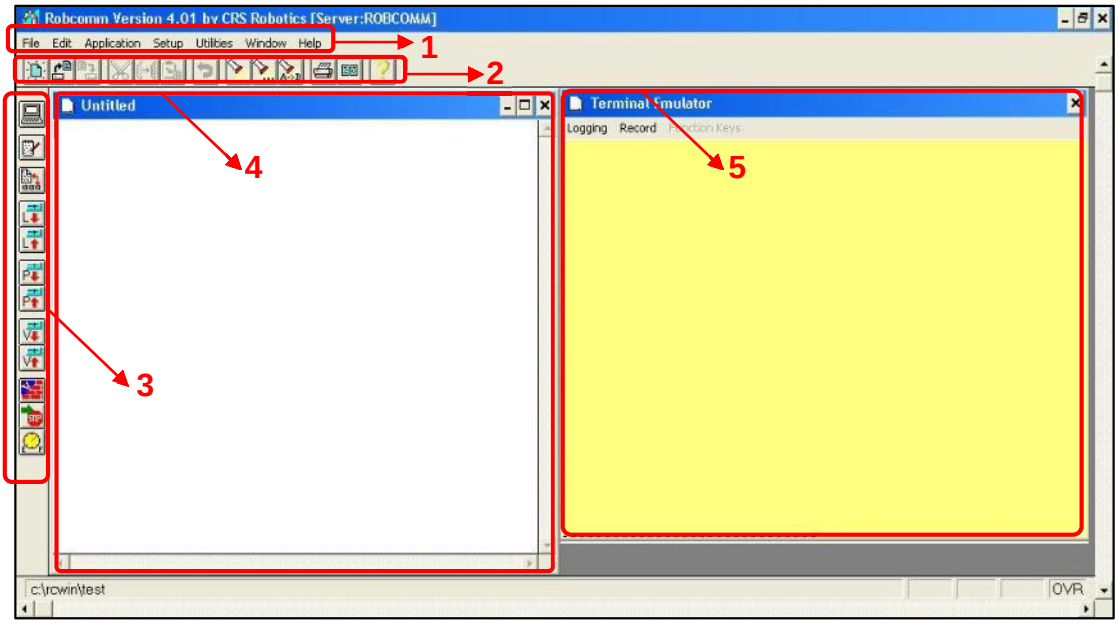


Figura. 30 Interfaz gráfica del software RobComm.

1. Barra de menús.
2. Barra de herramientas.
3. Barra de acceso rápido.
4. Ventana de escritura
5. Ventana terminal (línea de comandos)

2.8 COMPILADORES

2.8.1 Introducción

Un mecanismo de comunicación entre una persona y un ordenador viene dado por el envío y recepción de mensajes de tipo textual: el usuario escribe una orden y el ordenador la ejecuta devolviendo un resultado sobre las acciones llevadas a cabo. (Galvez & Mora, 2005)

Con la llegada de la computadora en los años 40 surge la necesidad de escribir secuencias de códigos, o programas que realizaran cálculos deseados. Dando origen al lenguaje máquina, que no era más que códigos numéricos que representaban las operaciones reales de la máquina que iban a efectuarse, con la desventaja que llevaba tiempo y era una forma tediosa de escribir programas.

Posteriormente fue reemplazada por el código ensamblador, el cual traduce los códigos simbólicos y las localidades de memoria a los códigos numéricos correspondientes del lenguaje máquina. Con el paso del tiempo se comienza a elevar el nivel de abstracción, alejándose cada vez más del lenguaje máquina, sustituyendo a las instrucciones simbólicas por códigos independientes parecidas al lenguaje humano.

Aunque el futuro se orienta al empleo de ergonómicas interfaces de usuario como pantallas táctiles, las tabletas gráficas, etc., se puede mencionar que casi todas las acciones que el usuario realiza se traducen antes o después a secuencias de comandos que son ejecutadas como si hubieran sido introducidas por teclado. (Galvez & Mora, 2005)

2.8.2 Traductores

Es un programa que convierte un código escrito en un lenguaje fuente hasta un código en un lenguaje destino. Los traductores intentan facilitar el trabajo en que las personas expresan sus necesidades y la forma en que un ordenador es capaz de interpretar instrucciones. Los traductores engloban tanto a los compiladores como a los intérpretes

Un compilador es un traductor que tiene como entrada un programa escrito en un lenguaje fuente y produce como salida en un programa equivalente escrito en otro lenguaje destino. También se entiende por compilador aquel programa que proporciona un fichero objeto en lugar del ejecutable final.

Figura. 31 Esquema de un Compilador.

Si el programa destino es ejecutable, puede ser invocado por el usuario para recibir entradas y generar salidas. Además cumple un rol importante, es que el de reportar errores durante el proceso de traducción.

Figura. 32 Esquema de un programa ejecutable.

Un intérprete es similar a un compilador, Solo que en lugar de producir un programa destino, aparenta ejecutar las instrucciones del programa origen, sobre la entrada del usuario.

Es decir el programa de entrada se reconoce y la salida se ejecuta a la vez. No se produce un resultado físico (código máquina) sino lógico (una ejecución).

Su principal ventaja es que permite una fácil depuración. Como inconveniente se encuentra, la lentitud de ejecución, ya que al ejecutar a la vez que se traduce no puede aplicarse un alto grado de optimización.

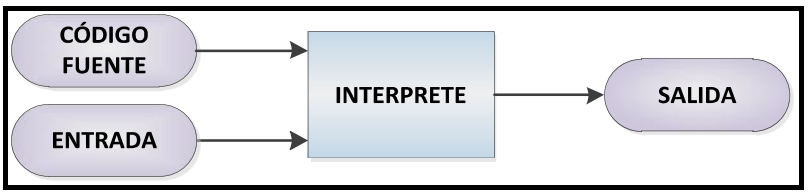


Figura. 33 Esquema de un intérprete.

Los procesadores de lenguaje Java, combinan compilación e intérprete. Un código fuente Java se compila primero en una forma intermedia, llamada bycode ".class", luego los bycode son interpretados por una máquina virtual.

Esta combinación permite controlar en tiempo real, las características del programa que está ejecutándose; Hoy en día es muy utilizado por: Java Virtual Machine, Common Language Runtime (MS CLR .NET), Zend Engien (PHP), entre otros.

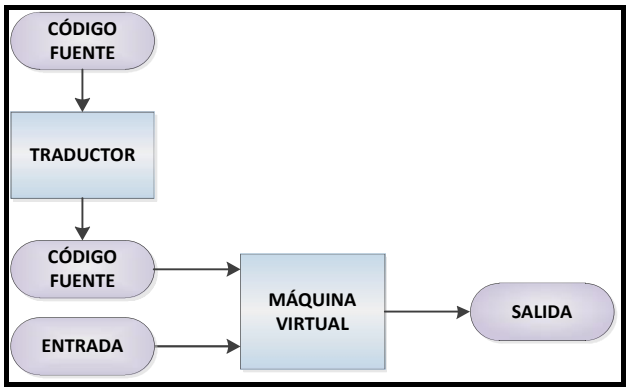


Figura. 34 Esquema de compilador-intérprete.

2.8.3 Estructura de un Traductor

Un traductor divide su labor en dos etapas: la primera de análisis, la cual controla que el texto fuente sea correcto, generando las estructuras necesarias para la generación de código, y la segunda de síntesis, donde se genera el código máquina equivalente semánticamente al programa fuente.

A continuación se muestra un diagrama que representa las etapas. (Ver figura. 35)

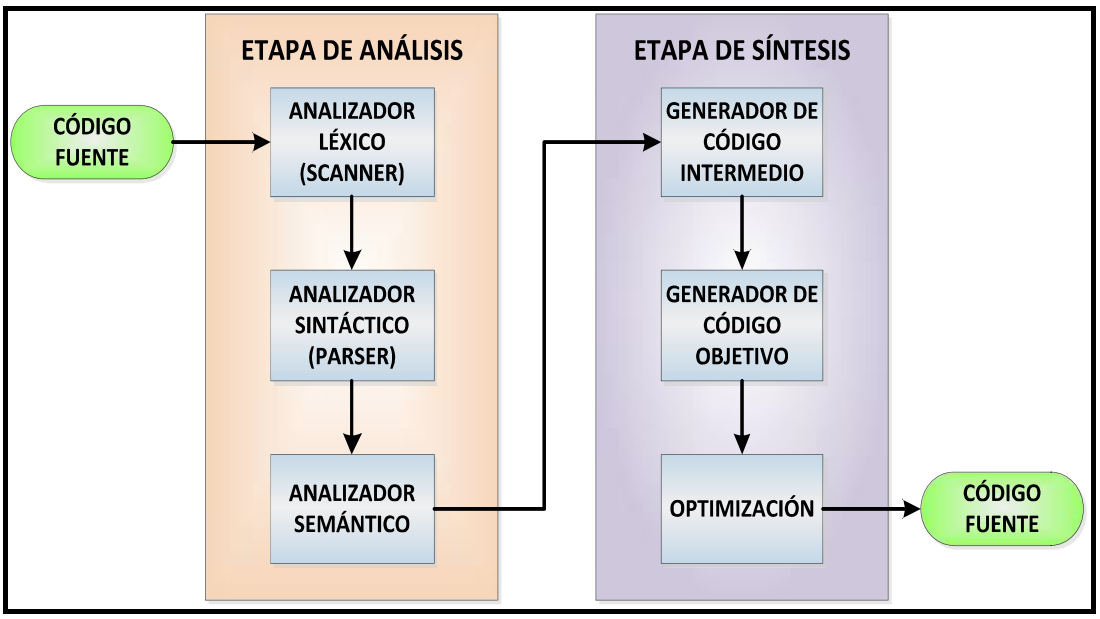


Figura. 35 Estructura de un traductor.

2.8.3.1 Analizador Léxico

Divide el programa fuente en los componentes básicos del lenguaje a compilar. Recolecta secuencias de caracteres en unidades significativas denominadas tokens.

Cada componente básico es una sub-secuencia de caracteres del programa fuente, y pertenece a una categoría gramatical: números, identificadores de usuario (variables, constantes), palabras reservadas, signos de puntuación, etc.

En el siguiente ejemplo se considera una línea de código:

a[indice] = 5 + 2

Este código contiene 12 caracteres diferentes pero sólo 8 tokens:

- **a** → identificador.
- **[** → corchete izquierdo.
- **indice** → identificador.
- **]** → corchete derecho.
- **=** → asignación.
- **5** → número.
- **+** → signo más.
- **2** → número.

Cada token se compone de uno o más caracteres que se reúnen en una unidad antes de que ocurra un procesamiento adicional.

2.8.3.2 Analizador Sintáctico

Recibe el código fuente en la forma de tokens proveniente del analizador léxico y realiza el análisis sintáctico, que determina la estructura del programa. Esto es semejante a un análisis gramatical sobre una frase en un idioma específico.

Determina los elementos estructurales del programa y sus relaciones. Los resultados del análisis sintáctico se representan como un árbol de análisis gramatical o un árbol sintáctico.

Utilizando el ejemplo anterior, la cual es una expresión de asignación compuesta de una expresión con subíndice a la izquierda y una expresión aritmética entera a la derecha. Esta estructura se puede representar como un árbol de análisis gramatical. (Ver figura. 36)

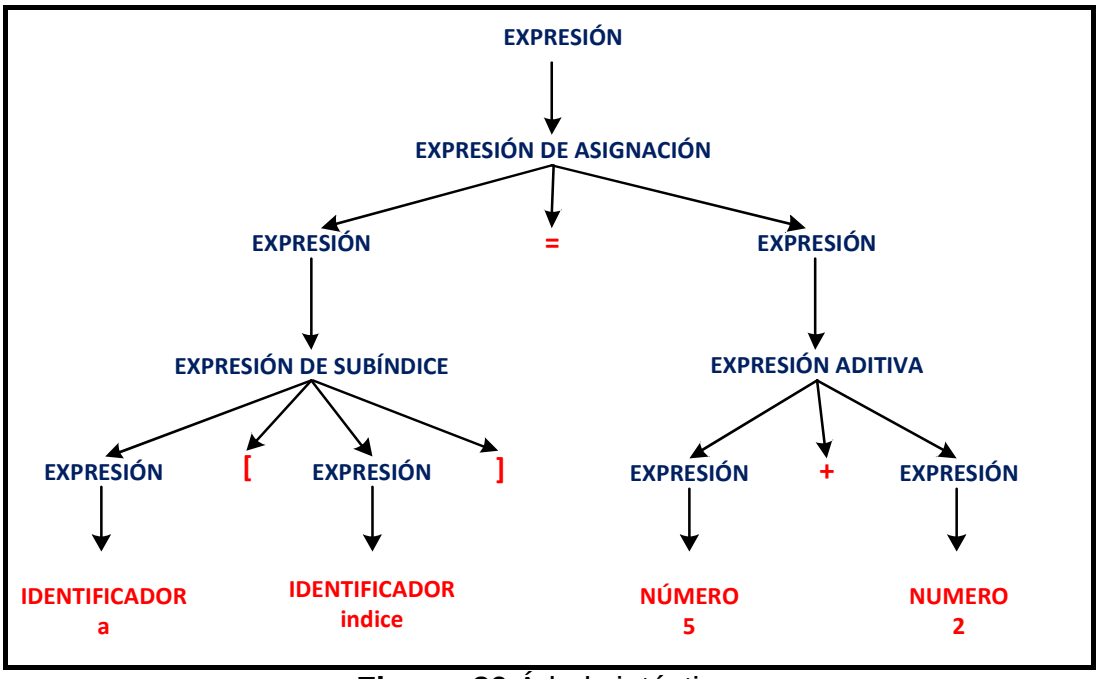


Figura. 36 Árbol sintáctico.

Se observa que los nodos internos del árbol sintáctico están etiquetados con los nombres de las estructuras que representan (color azul) y que las hojas del árbol representan la secuencia de tokens de la entrada (color rojo).

2.8.3.3 Analizador Semántico

Además de controlar que un programa cumpla con las reglas de la gramática del lenguaje, hay que comprobar que lo que se quiere hacer tiene sentido. El análisis semántico dota de un significado coherente a lo que hemos hecho en el análisis sintáctico.

El chequeo semántico se encarga de que los tipos que intervienen en las expresiones sean compatibles o que los parámetros reales de una función sean coherentes con los parámetros formales: p.ej. no suele tener mucho sentido el multiplicar una cadena de caracteres por un entero.

2.8.3.4 Generador de Código Intermedio

Después de la etapa de análisis, se suele generar una representación intermedia explícita del programa fuente.

Dicha representación intermedia se puede considerar como un programa para una máquina abstracta.

Cualquier representación intermedia debe tener dos propiedades importantes; debe ser fácil de generar y fácil de traducir al código máquina destino. Así, una representación intermedia puede tener diversas formas.

2.8.3.5 Generador de Código Objetivo

La fase final de un compilador es la generación de código objeto, que por lo general consiste en código ensamblador. Cada una de las variables usadas por el programa se traduce a una dirección de memoria.

Después, cada una de las instrucciones intermedias se traduce a una secuencia de instrucciones de máquina que ejecuta la misma tarea. Un aspecto decisivo es la asignación de variables a registros.

2.8.3.6 Optimización

En esta fase el compilador intenta mejorar el código objetivo generado por el generador de código. Dichas mejoras incluyen la selección de modos de direccionamiento para mejorar el rendimiento, reemplazando las instrucciones lentas por otras rápidas, y eliminando las operaciones redundantes o innecesarias.

2.8.4 Herramientas del Analizador Léxico y Sintáctico

Jlex y Java Cup son herramientas utilizadas para la programación de sistemas ya que permite generar compiladores a través de gramáticas independientes mediante analizadores, incorporándolas al gran potencial que brinda el lenguaje de programación Java.

- Jlex Es una herramienta desarrollada en Java que genera un analizador léxico, el cual toma una cadena de caracteres como entrada y los convierte en una secuencia de tokens.
- Java Cup Es un generador sintáctico el cual recibe de entrada un archivo con la estructura de la gramática y su salida es un árbol sintáctico (parser) escrito en Java.

2.8.4.1 Instalación de Jlex y Java Cup

- Primeramente es necesario configurar las variables de entorno antes de instalar los componentes necesarios para la instalación e integración de Jlex y Java Cup. La configuración de las variables de entorno se realiza vía consola:
- Nos situamos dentro de la carpeta **bin,** del JDK de Java, la que se encuentre en la ruta **C:\Archivos de programa\Java\jdk1.7.0_40\bin**.

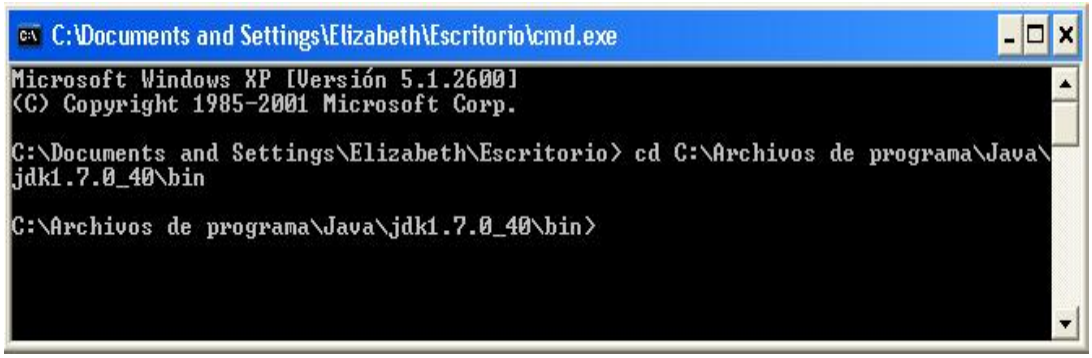


Figura. 37 Cambio de directorio a la carpeta JDK por consola.

- Ahora se escribe:

set CLASSPATH=C:\Archivosdeprograma\Java\jdk1.7.0_40\bin;%CLASSPATH%

set PATH=C:\Archivos de programa\Java\jdk1.7.0_40\bin;%PATH%

El PATH es la variable del sistema que utiliza el sistema operativo para buscar los ejecutables necesarios desde la línea de comandos o la ventana Terminal.

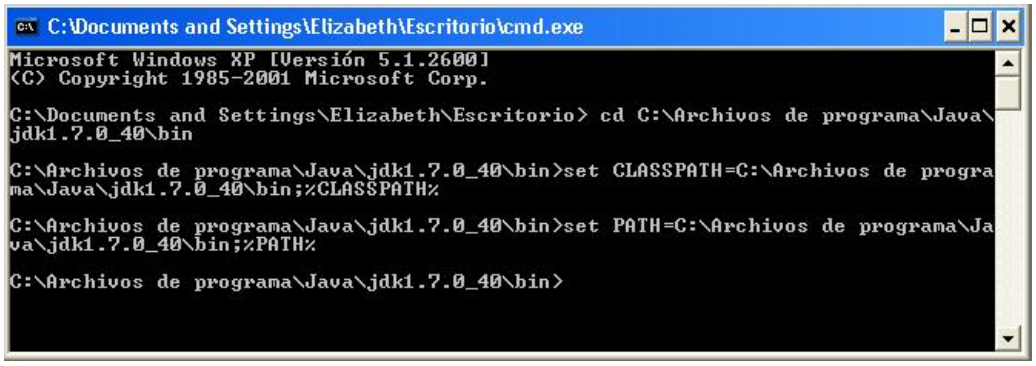


Figura. 38 Configuración de variables de entorno.

- Descargar la clase Main.java la cual servirá para instalar Jlex, la descarga puede realizar del siguiente enlace:

http://www.cs.princeton.edu/~appel/modern/java/JLex/

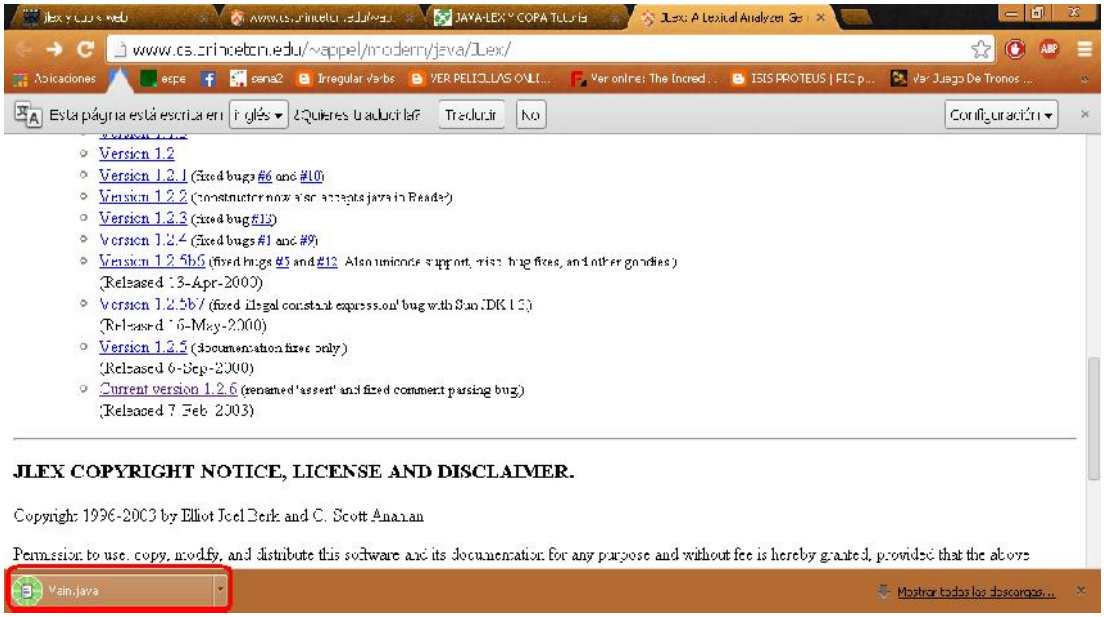


Figura. 39 Descarga de la clase Main.java para el JLex.

- Ubicarse en de la carpeta **bin**, dentro de la ruta de java **C:\Archivos de programa\Java\jdk1.7.0_40\bin**. Se crea una carpeta con el nombre **JLex**

- Dentro de esta carpeta copiamos la clase Main.java que descargamos anteriormente, ahora se compila la clase JLex por medio de consola, para que se generen los archivos ***.class*** necesarios para la completa integración de JLex a Netbeans.

- Se escribe: **Javac Jlex/Main.java**

Figura. 40 Compilación por consola de la clase Main.java.

- Se generan los .class dentro de la carpeta Jlex.

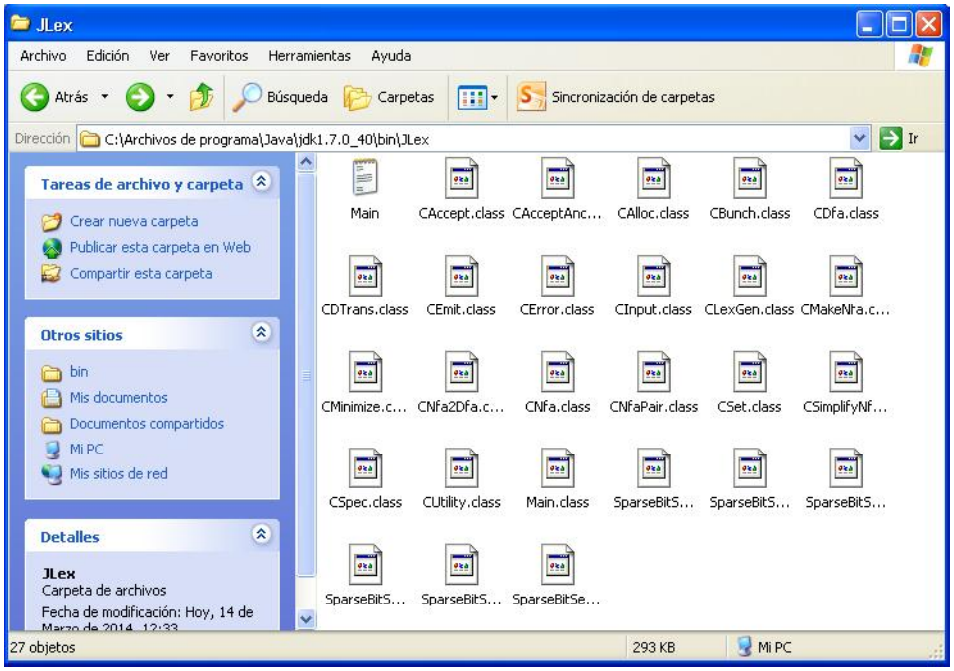


Figura. 41 Generación de los .class para JLex.

- Descargar los archivos para la integración de Java Cup con Netbeans, la descarga puede realizar del siguiente enlace:

http://www2.cs.tum.edu/projects/cup/

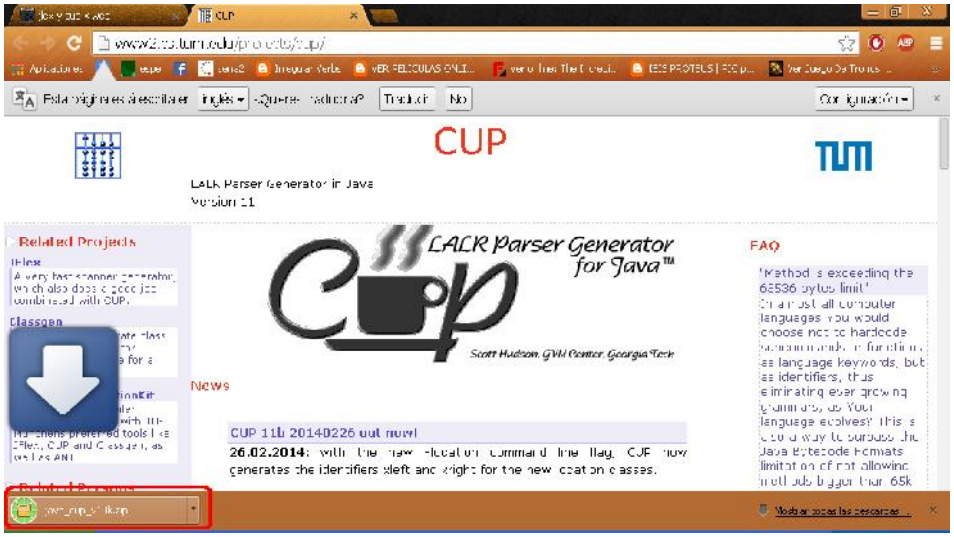


Figura. 42 Descarga de los componentes de Java Cup.

- Una vez descargada esta carpeta, descomprimirla, y deberá ser colocada en la carpeta **bin** del JDK.

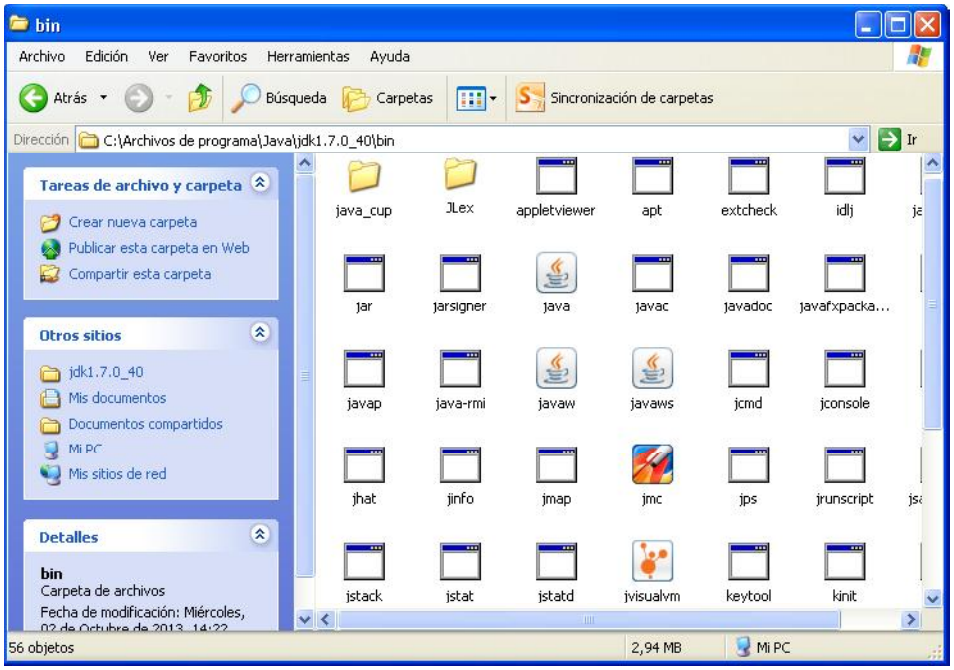


Figura. 43 Archivos descomprimidos de Cup en la carpeta bin.

CAPÍTULO 3

DISEÑO E IMPLEMENTACIÓN DE HARDWARE Y SOFTWARE

3.1 HARDWARE

Una vez conocidas las características técnicas, estructurales y funcionales del controlador CAD, se tiene una idea clara de todos los requerimientos que se desean incorporar en el desarrollo de este proyecto.

Para cumplir con todas las expectativas propuestas; es necesario realizar algunas modificaciones en la etapa de potencia y en la etapa de control, ya que el sistema requiere:

- El funcionamiento de las cinco articulaciones del manipulador robótico simultáneamente.
- Disponer con la capacidad necesaria para soportar el software que se desarrollará posteriormente.

- Dotar de puertos de entrada y salida para la interacción del manipulador con su entorno.

3.1.1 Análisis de Tarjeta de Potencia

Para que el manipulador robótico realice un correcto desempeño, se debe dejar en funcionamiento los dos circuitos de activación restantes de los motores que no se encontraban operativos. Tomando en cuenta que los cinco motores que controlan las articulaciones ahora deben funcionar simultáneamente.

En el actual controlador, existen tres fusibles: dos de ellos protegen a cuatro articulaciones (dos articulaciones por fusible), y un tercero que protege a la articulación restante junto al circuito de frenos y solenoide del gripper. Como se pudo ver anterior mente. (Ver figura. 21).

La primera dificultad se presenta, cuando el circuito de un motor llega a tener problemas, ocasionando que automáticamente quede fuera de funcionamiento dos articulaciones del manipulador o a su vez los frenos de todas las articulaciones

Una segunda dificultad se encuentra en que los porta-fusibles están soldados en la placa de potencia; como consecuencia de ello, se deberá abrir el chasis para el cambio de un fusible dañado, dejando fuera de funcionamiento al manipulador hasta que el mismo sea remplazado.

Por lo tanto debe proteger a cada motor con fusibles independientes, contra posibles variaciones en la corriente o daños que se presente durante la manipulación de los mismos. Adicionalmente se colocará porta-fusibles en la parte frontal exterior del chasis, ya que al ser un sistema de entrenamiento para estudiantes, el reemplazo de fusibles defectuosos no sea demoroso.

Para independizar con fusibles a cada motor y circuito de frenos, es necesario diseñar una nueva tarjeta de potencia, conservando las configuraciones de activación de cada motor de la anterior tarjeta.

3.1.2 Diseño e Implementación de Tarjeta de Potencia

Ya que se decidió independizar la protección por cada motor, se debe realizar cambios al circuito de enclavamiento como se muestra a continuación. (Ver figura 44)

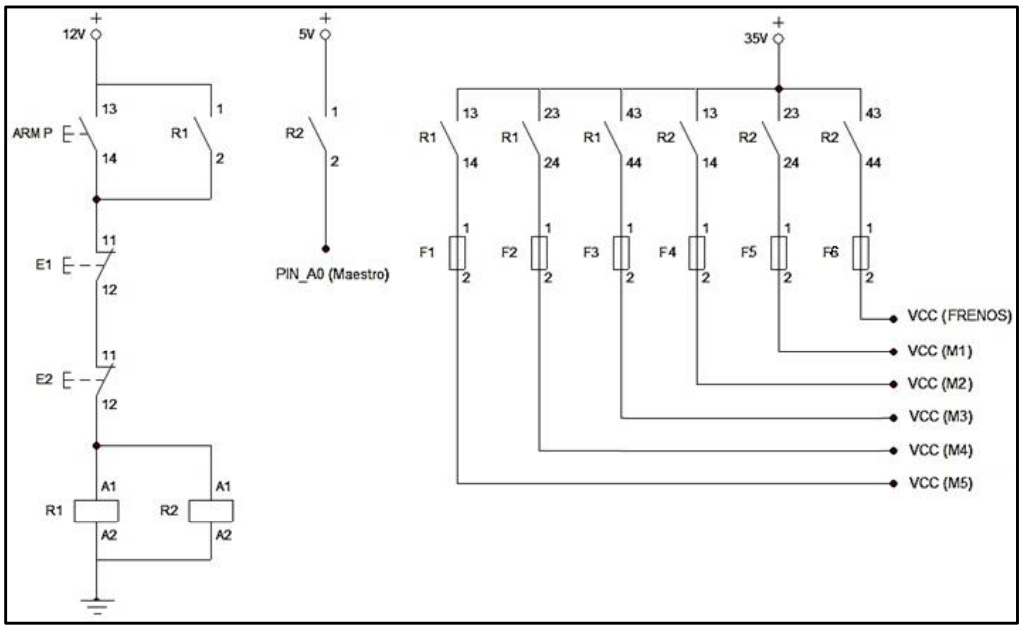


Figura. 44 Cambios realizados al circuito de enclavamiento.

Al ser seis circuitos en la placa (motores y frenos), no fue necesario añadir un relé adicional, ya que los ocho contactos auxiliares que se tienen en total son suficientes para realizar las conexiones necesarias.

En el siguiente diagrama de bloques se describe los cambios que se realizará a la tarjeta de potencia. (Ver figura. 45)

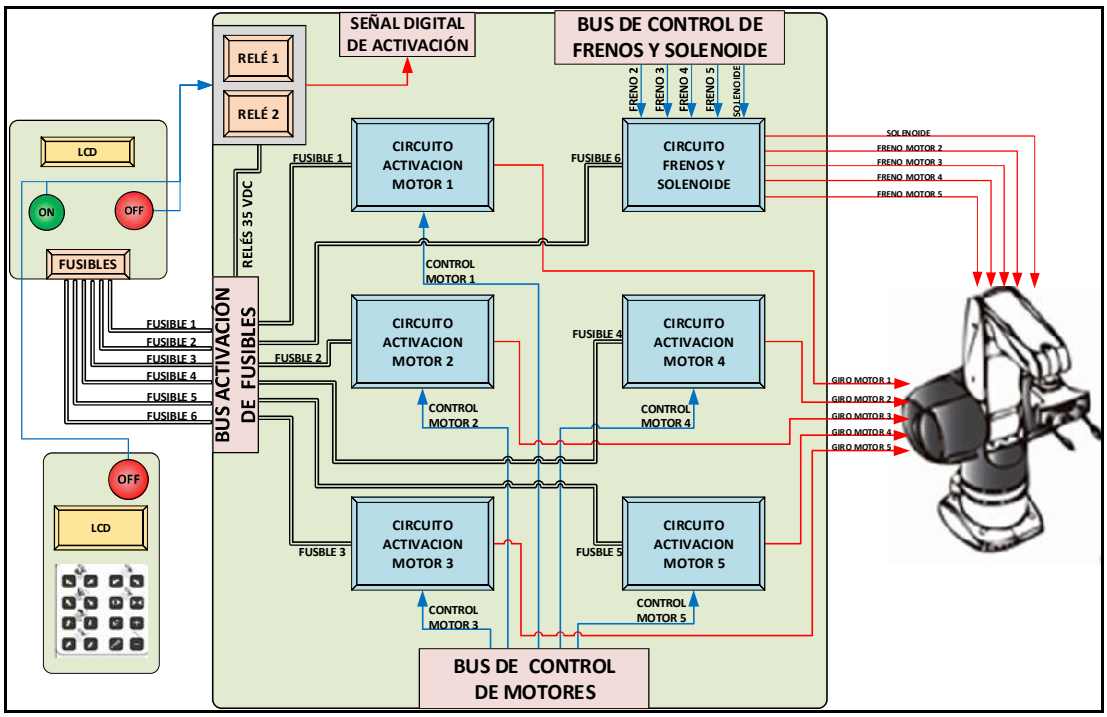


Figura. 45 Diagrama de funcionamiento de la nueva tarjeta de potencia.

Se puede observar que se adaptará un bus que permite la conexión entre los porta-fusibles ubicados en el panel exterior del case con la tarjeta de potencia.

En el Anexo. 2, se detalla el circuito de conexiones de los componentes y buses que conforman la nueva tarjeta de potencia. En adelante se la denominará con el nombre de tarjeta potencia CDC2.

A continuación se muestra el diseño de la tarjeta de potencia CDC2, observando la ubicación de los elementos y el diseño de las pistas en 2D y en 3D

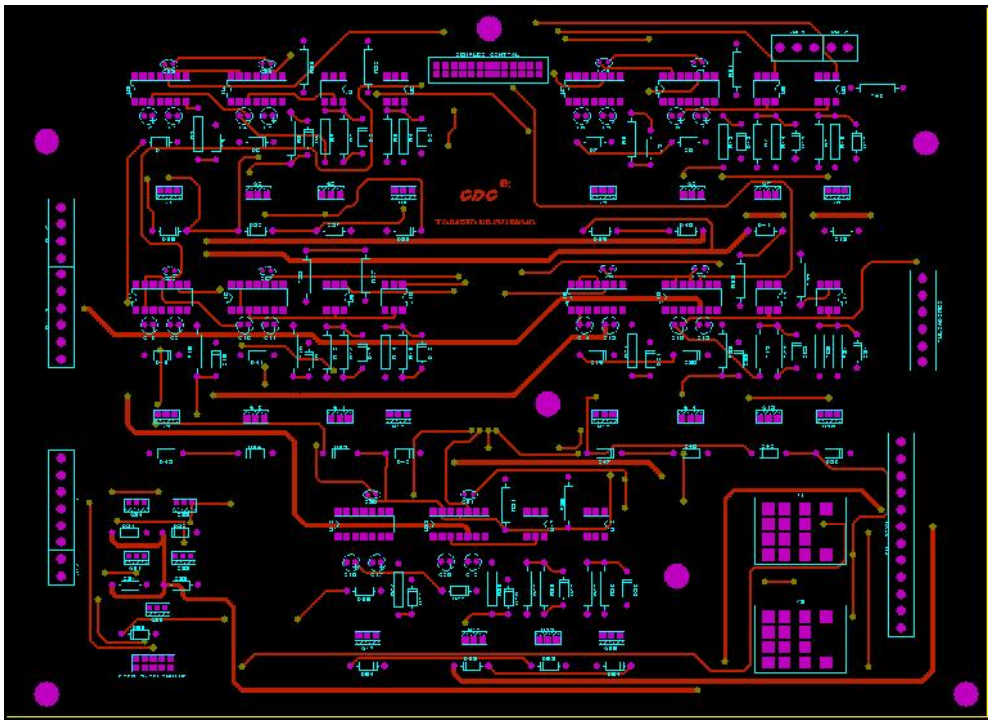


Figura. 46 Tarjeta de potencia CDC2, vista superior en 2D.

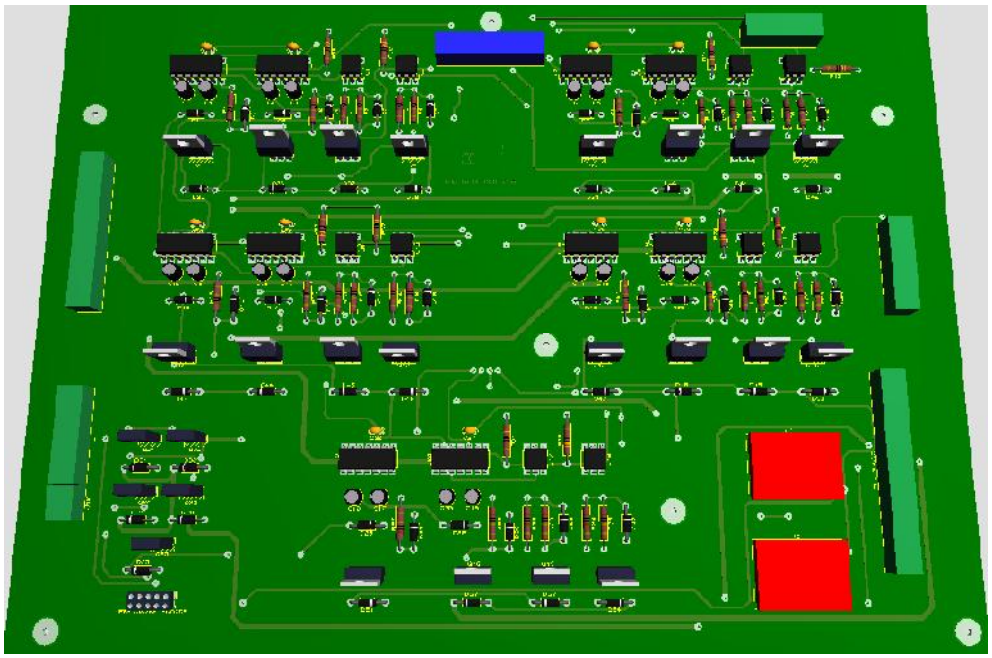


Figura. 47 Tarjeta de potencia CDC2, vista superior en 3D.

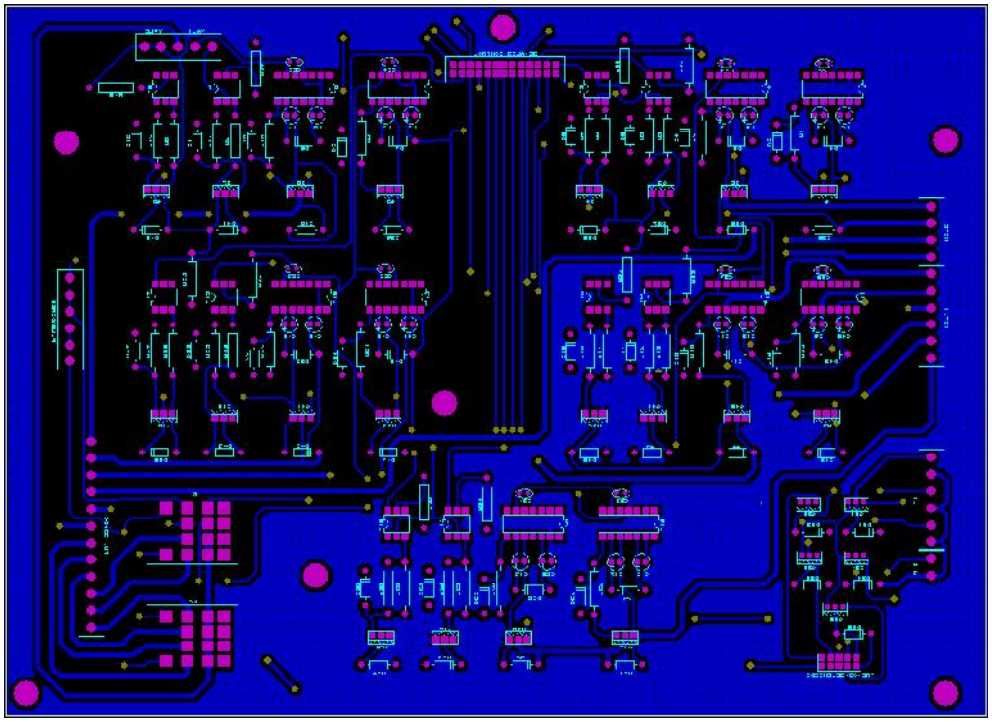


Figura. 48 Tarjeta de potencia CDC2, vista inferior en 2D.

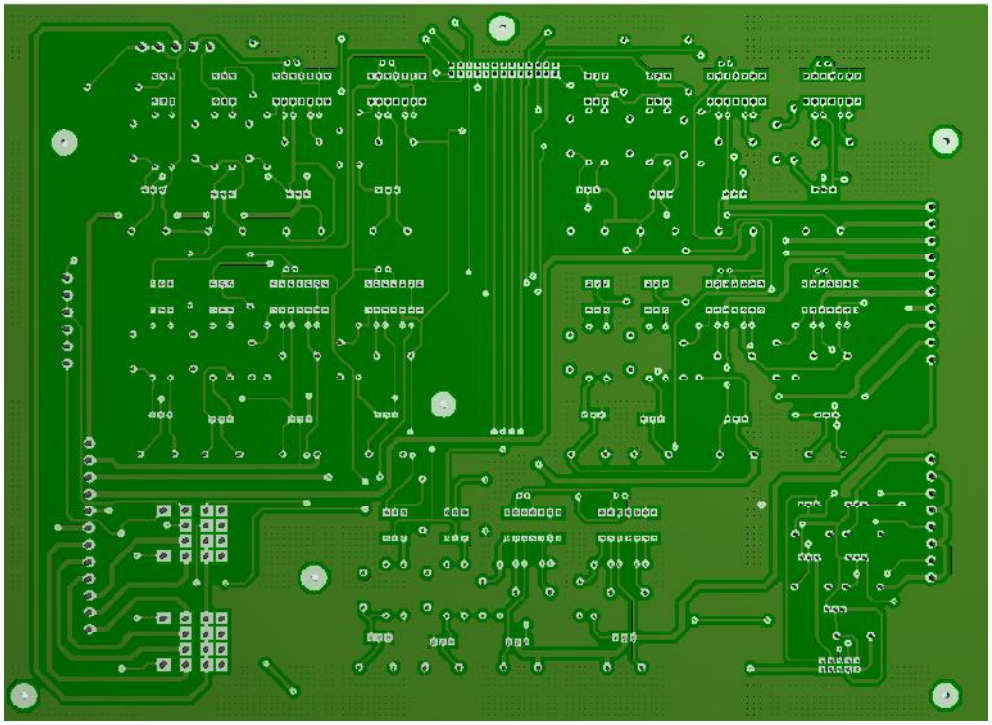


Figura. 49 Tarjeta de potencia CDC2, vista inferior en 3D.

En el Anexo. 3, se detalla la construcción de la PCB de la tarjeta, a continuación se muestra el acabado final de la tarjeta montada en el case. (Ver figura. 50)

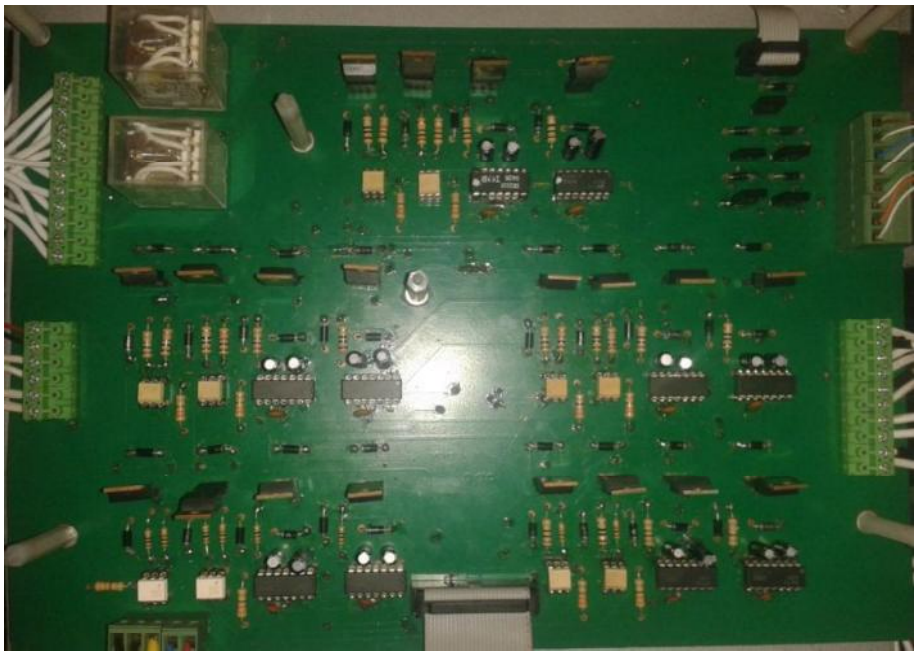


Figura. 50 Acabado final de tarjeta de potencia CDC2.

3.1.3 Análisis de Tarjeta de Control

Es necesario que los tipos de operación del sistema (modo manual y modo online) trabajen en conjunto, es decir que el sistema reciba órdenes del teach pendant y por comandos.

Por lo tanto el switch que permite seleccionar el modo de trabajo del manipulador robótico, dejó de ser necesario, y se lo excluirá para el desarrollo del proyecto.

Para la operación del manipulador robótico en modo online se va requerir una comunicación bidireccional que maneje cadena de caracteres para el envío y recepción de comandos.

Adicionalmente para contar con mayores prestaciones en el control de un motor, es necesario que el microcontrolador cuente con el mayor número de módulos independientes.

Todas las funciones que realiza actualmente el microcontrolador “maestro” son: gestionar las comunicaciones con los esclavos y el ordenador, además del manejo de teach pendant y LCD’s, dichas tareas dan como resultado que la memoria del mismo se encuentre ocupada casi en su totalidad, dejando sin capacidad de almacenamiento suficiente para continuar con el desarrollo de los cambios a implementar.

Una desventaja con el microcontrolador “esclavo” es que controla dos motores a la vez, implicando la división de recursos, saturación de la memoria de almacenamiento y disminución de la eficiencia en el control de cada motor.

Otra desventaja se encuentra en el circuito de señales de encoder, ya que al variar el ancho de pulso en un porcentaje mayor al 50% en el ciclo positivo, se pierde conteo de pulsos, esto se debe a que el circuito integrado de inversión de disparo (74LS14) no alcanza a descargarse (no llega a su valor lógico de cero), provocando que el microcontrolador asuma que no hubo conteo de pulsos.

3.1.4 Diseño e Implementación de Tarjeta de Control

El microcontrolador “maestro” actual será reemplazado por un PIC 18F452, a continuación se muestra un cuadro comparativo de las características principales entre los dos microcontroladores. (Ver tabla. 9)

Tabla. 9 Principales características entre PIC16F877A y PIC18F452.

CARACTERÍSTICAS	PIC 16F877A	PIC 18F452
Arquitectura	8 bit	8 bit
Frecuencia máx. CPU	20Mhz (5 MIPS)	40MHz (10MIPS)
Memoria de programa (flash)	8kB	32kB
RAM	368 byte	1536 byte
EEPROM	256 byte	256 byte
Número de pines	40 (Pin I/O:33)	40 (Pin I/O:34)
A/D conversor	1 (8 canales)	1 (8 canales)
Módulo CCP	2x CCP, resolución PWM 10 bit	2x CCP, resolución PWM 10 bit
Temporizadores	2x8 bits, 1x 16 bit	1x8 bits, 3x 16 bit
Periféricos de comunicaciones	1x A/E/USART, 1x MSSP (SPI/I2C)	1x A/E/USART, 1x MSSP (SPI/I2C)

La razón principal por la cual se decidió trabajar con este componente fue que su capacidad de memoria cuadruplica la capacidad del microcontrolador actual, permitiendo que a futuro se pueda implementar nuevas opciones y aplicar técnicas de control avanzadas.

Se trabajará con cinco microcontroladores “esclavos” PIC 16F88, a continuación se muestran las características principales del mismo. (Ver tabla. 10)

Tabla. 10 Características de PIC16F88.

CARACTERÍSTICAS	PIC 16F88
Arquitectura	8 bit
Frecuencia máx. CPU	20Mhz (5 MIPS)
Memoria de programa (flash)	7kB
RAM	368 byte
EEPROM	256 byte
Número de pines	18 (Pin I/O:16)
A/D conversor	1 (7 canales)
Módulo CCP	1x CCP, resolución PWM 10 bit
Temporizadores	2x8 bits, 1x 16 bit
Periféricos de comunicaciones	1x A/E/USART, 1x SSP (SPI/I2C)

Al trabajar con un microcontrolador por cada articulación, se puede aprovechar al máximo los recursos y capacidades que ofrece este, permitiendo que se pueda implementar nuevas alternativas en el control del motor.

En el siguiente diagrama de bloques se muestra el diseño de la nueva tarjeta de control. (Ver figura. 51)

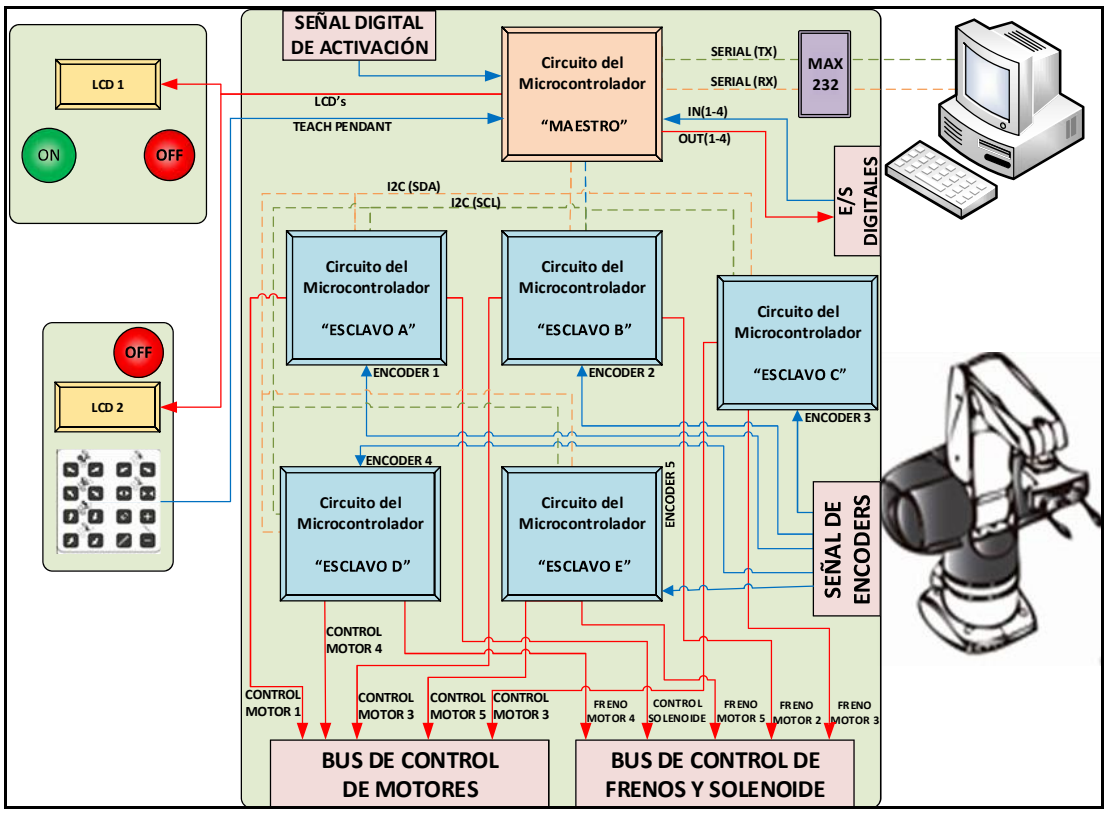


Figura. 51 Diagrama de funcionamiento de la nueva tarjeta de control.

A continuación se describen las conexiones que se realizaron a la placa de control.

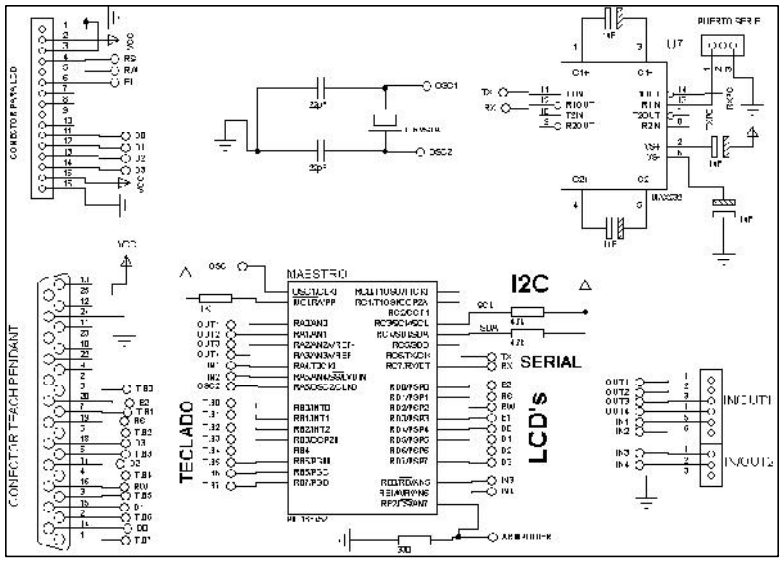


Figura. 52 Diagrama de conexión PIC "maestro".

Como se puede observar tiene la misma configuración de puertos que el microcontrolador “maestro” anterior, se descartó las señales provenientes del switch selector, además se adicionó entradas y salidas digitales por los puertos A y E.

A continuación se detalla la conexión de un microcontrolador esclavo para el control de un solo motor. (Ver figura. 53)

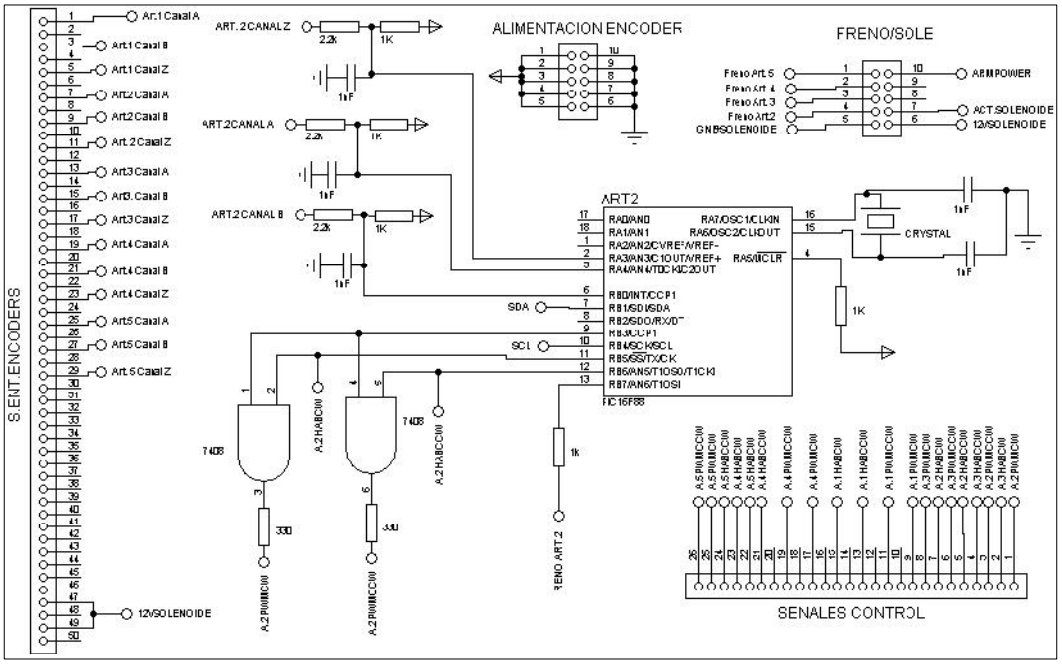


Figura. 53 Diagrama de conexión PIC "esclavo".

Al microcontrolador llegan tres señales de entrada de enconder, cabe recalcar que el circuito integrado 74LS14 se descartó para no tener pérdida de pulsos como sucedía en la antigua tarjeta.

Las tres señales de encoder (A, B, Z) se configuran de la siguiente forma:

- El canal A está conectado al pin RA4, este pin cuenta con el módulo contador Timer0, permitiendo que el conteo de pulsos se lleve a cabo de forma independiente al programa principal del microcontrolador.
- El canal B está conectado al pin RB0, este pin cuenta con la interrupción INT0, la cual es útil para conocer el sentido de desplazamiento de una articulación.
- El canal Z está conectado del al pin RA3, este pin se utiliza cuando se desee conocer cuántas revoluciones ha dado el motor.

Lo que concierne al circuito de activación de freno, se aumentó el valor de la resistencia de activación a 1Kohm, para tener una mayor protección del transistor.

Para el control de movimiento del motor no fue necesario añadir cambio por lo que se conserva la misma configuración de la tarjeta anterior.

En el Anexo. 4, se presenta el esquema del circuito completo de la tarjeta de control, en adelante se le denominará con el nombre de tarjeta control CDC2.

A continuación se detalla el diseño de la tarjeta observando la ubicación de los elementos y el diseño de las pistas en 2D y en 3D

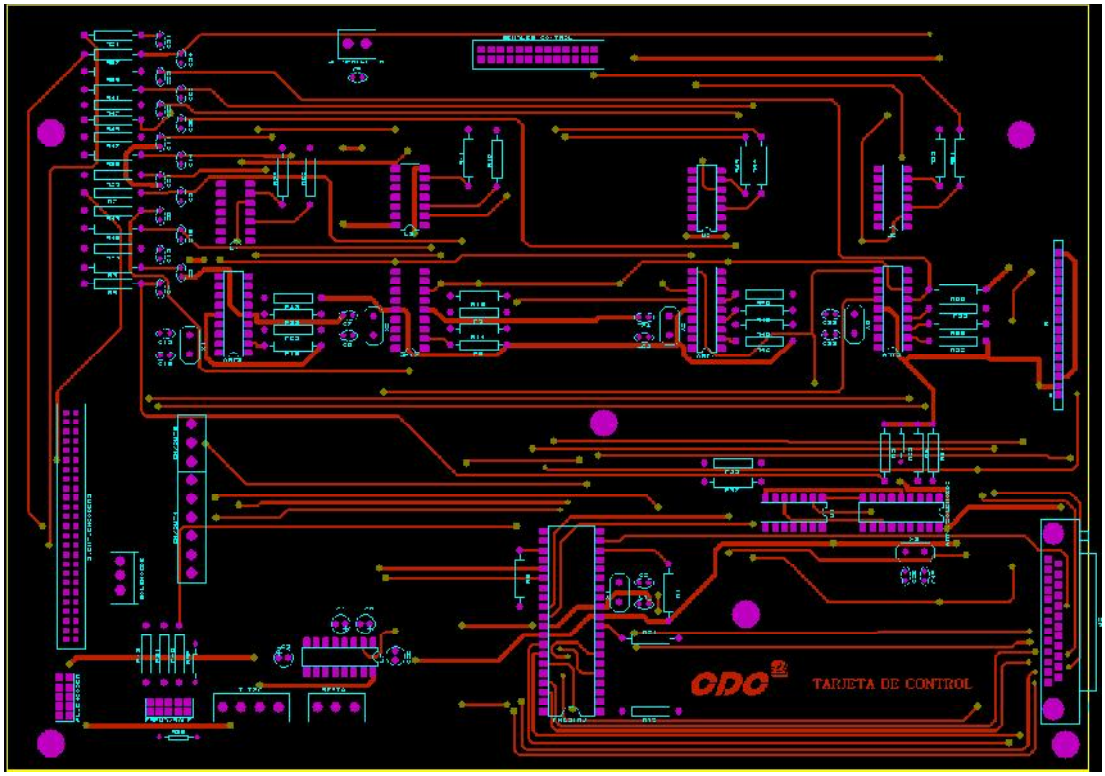


Figura. 54 Tarjeta de control CDC2, vista superior en 2D.

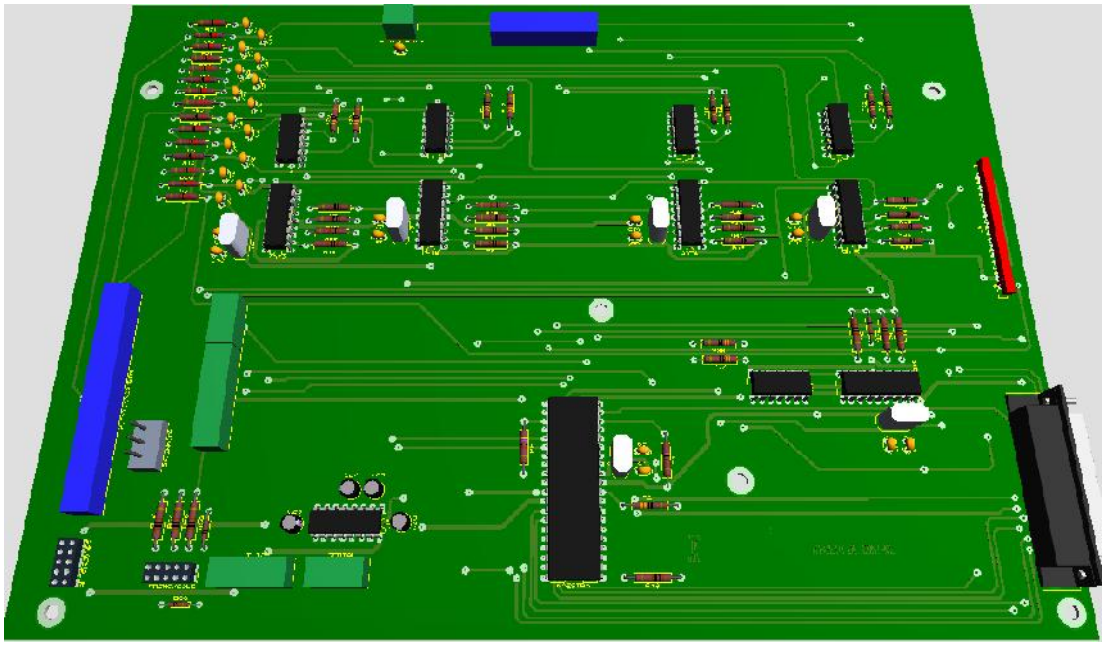


Figura. 55 Tarjeta de control CDC2, vista superior en 3D.

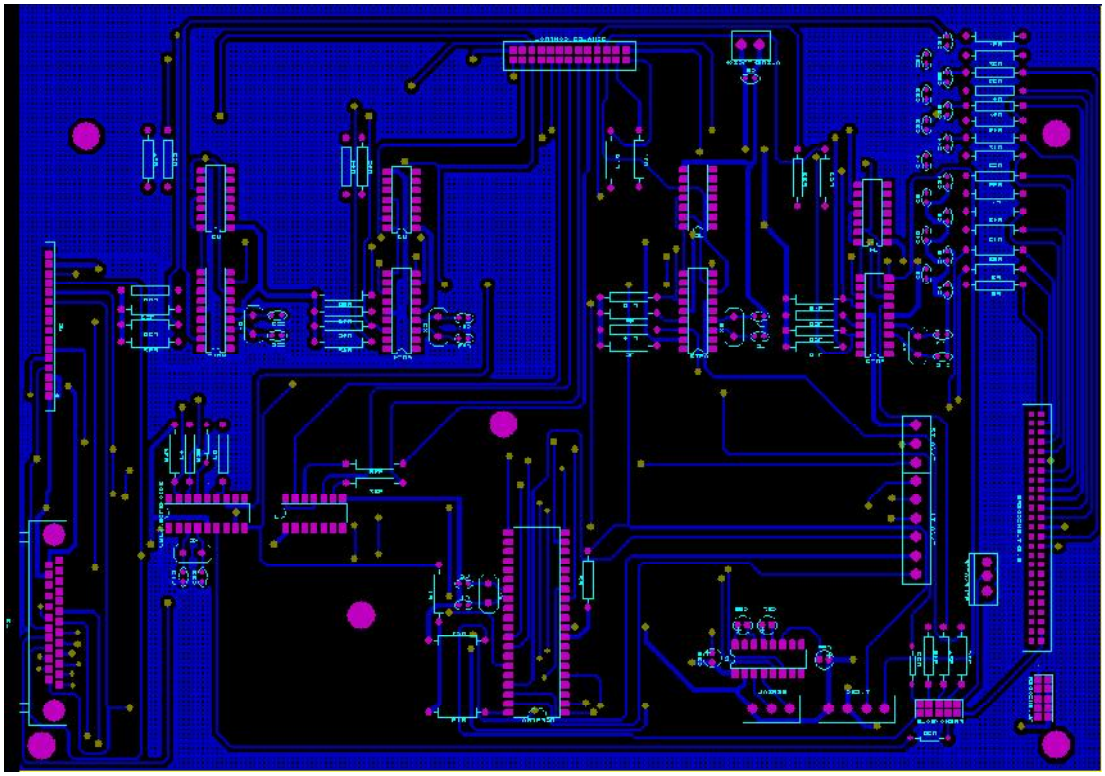


Figura. 56 Tarjeta de control CDC2, vista inferior en 2D.

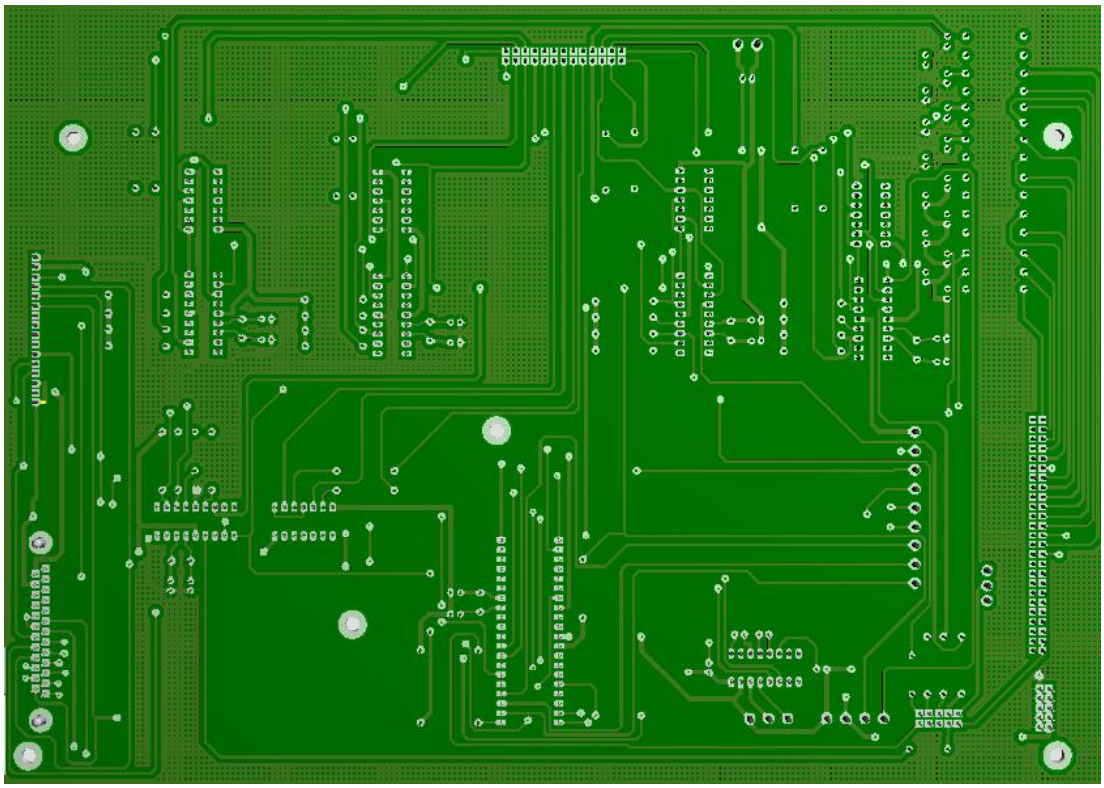


Figura. 57 Tarjeta de control CDC2, vista inferior en 3D.

En el Anexo. 5, se detalla la construcción de la PCB de la tarjeta, Se muestra a continuación el acabado final de la tarjeta montada en el case. (Ver figura. 58)

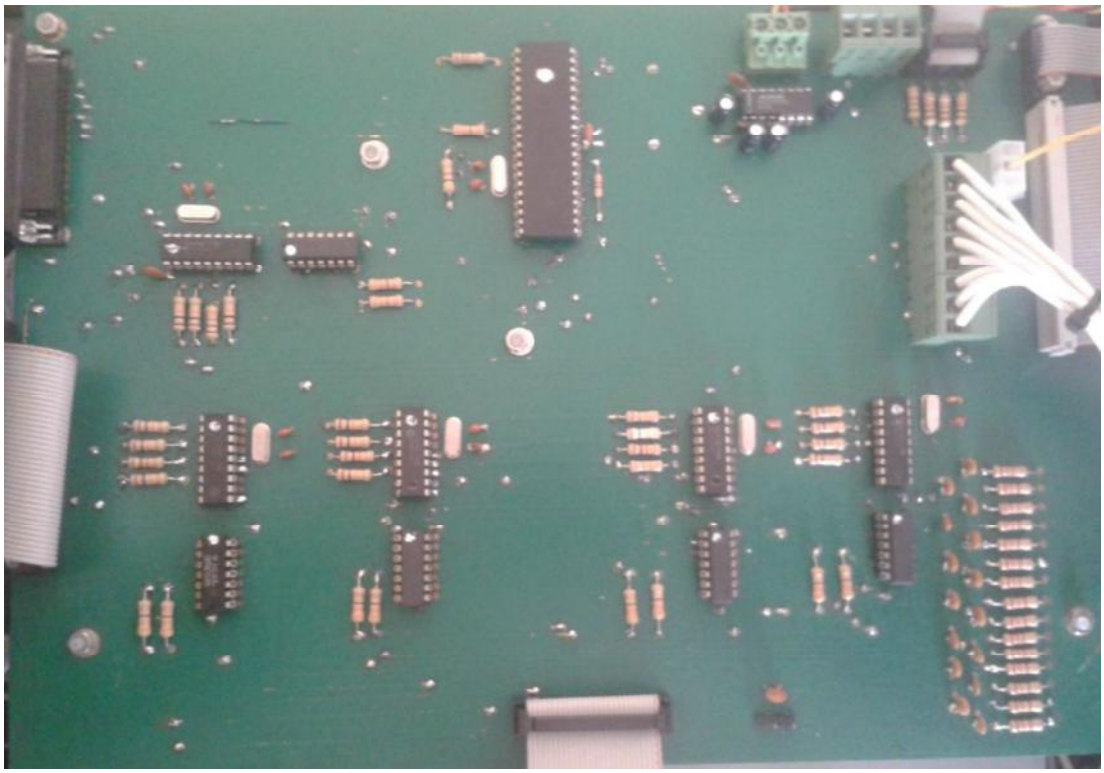


Figura. 58 Acabado final de tarjeta de control CDC2.

A continuación se muestra los cambios realizados al case del controlador, con los nuevos componentes que se implementaron. (Ver figura. 59)

Figura. 59 Cambios al case del controlador.

3.2 SOFTWARE

Se pretende desarrollar un software (interfaz gráfica de usuario), que emule el entorno de trabajo y ciertas funcionalidades del software RobComm. Para lo cual se requiere seleccionar los comandos que no impliquen cinemática o dinámica de robots, estructurar la sintaxis de los mismos para el envío y recepción de datos.

Además, al ser un software de programación se requiere un compilador para que los comandos sean ejecutados correctamente por el manipulador robótico.

En el siguiente diagrama se presenta los casos de uso general del sistema. (Ver figura. 60)

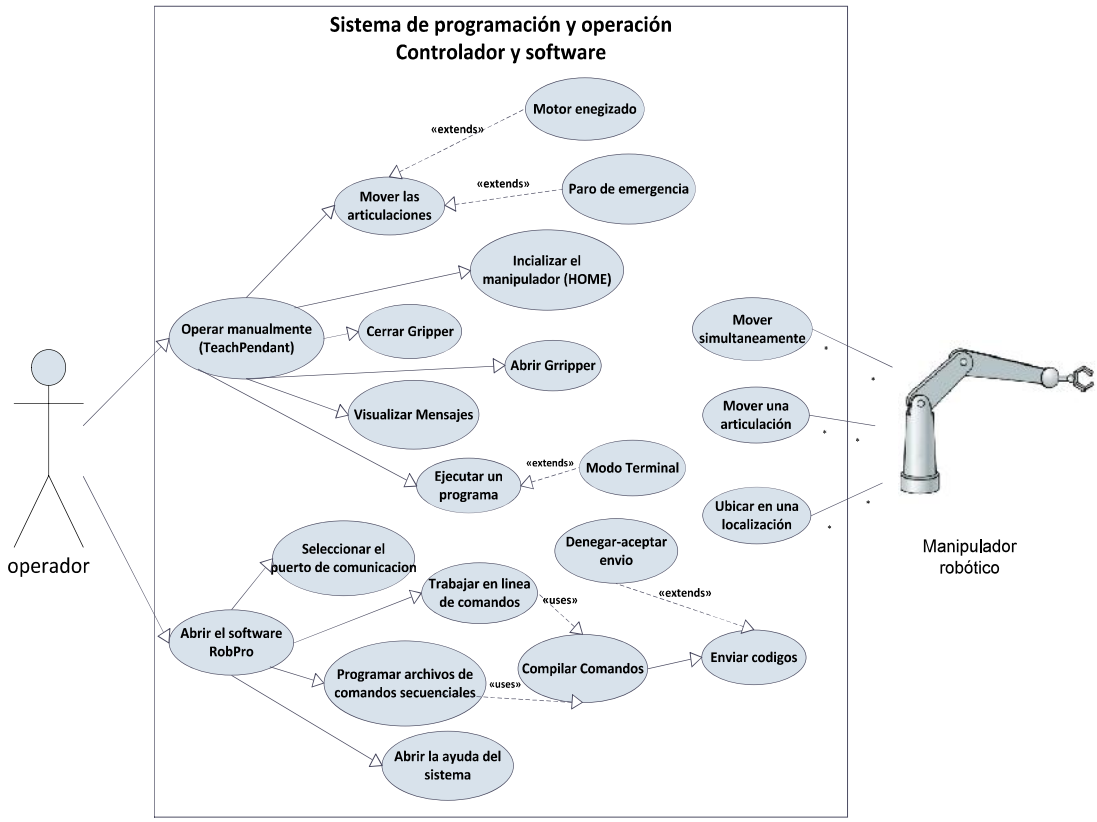


Figura. 60 Diagrama de casos de uso general del sistema.

3.2.1 Análisis de Comunicación

Para el desarrollo de la siguiente etapa, se plantea la estructura de comunicación e interacción del sistema incorporado con el software a desarrollarse, para dar cabida al movimiento simultaneo de todas las articulaciones.

Para lo cual se ha diseñado el siguiente diagrama por niveles (Ver figura. 61)

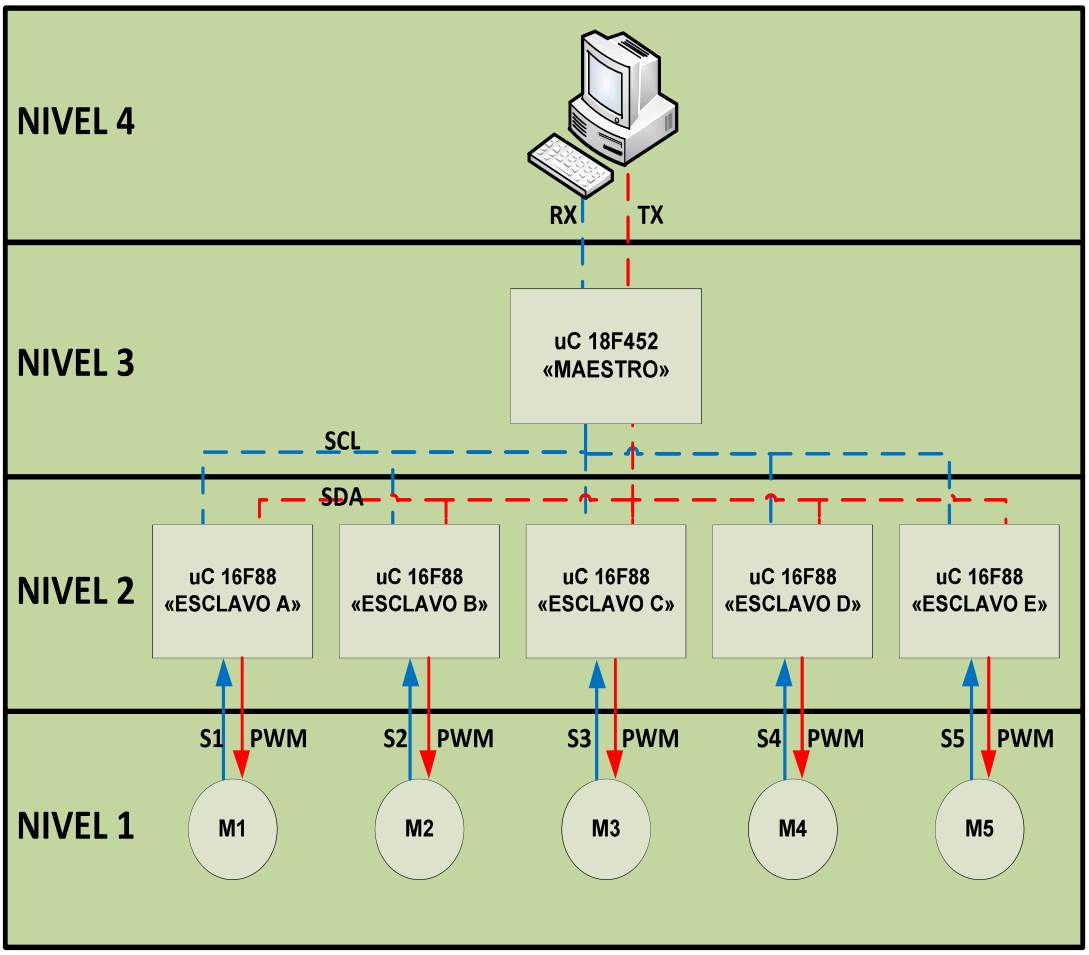


Figura. 61 Niveles de comunicación.

A continuación se detalla la función que desempeña cada nivel. (Ver tabla. 11)

Tabla. 11 Descripción de niveles de comunicación.

NIVELES	FUNCIÓN
Nivel 4	Recibe el comando del usuario, lo codifica para enviarlo al nivel tres, si existe un evento de recepción lo atiende, caso contrario continua con la ejecución del programa principal.
Nivel 3	Si existe un dato de recepción, este interrumpe su programa para atenderlo y codificarlo, posteriormente se coloca en modo escritura para establecer la comunicación y envío de dato con el "esclavo" requerido. Si requiere información de retorno se configura en modo lectura esperando una respuesta del mismo, caso contrario, continua con su programa principal.
Nivel 2	Si el "maestro" del nivel tres se encuentra en modo escritura, se establece la comunicación con un "esclavo" de este nivel. El mismo recibe un dato y decide que operación ejecutar sobre el motor conectado. A la vez almacena el sentido y número de pulsos que ha realizado el motor. Si el "maestro" requiere dicha información se actualiza las variables de envío con los valores solicitados.
Nivel 1	El motor recibe la información de activación o desactivación del "esclavo del nivel superior y envía la información de pulsos y sentido que este se ha desplazado.

Se realizó este análisis, ya que cada microcontrolador "esclavo" decide que operación realizar sobre un motor, por lo tanto, se determinó que para lograr simultaneidad de los motores, es necesario que la transmisión de datos del "maestro" a todos sus "esclavos" sea en el menor tiempo posible.

3.2.2 Análisis de Comandos

Para establecer los comandos que se van a implementar en el software, se consideraron aquellos que no requieren de un modelo matemático. Los comandos seleccionados de localización y movimiento son los siguientes. (Ver tabla. 12)

Tabla. 12 Comandos seleccionados con su sintaxis.

COMANDO	SINTAXIS	DETALLE U OBSERVACIÓN
CLOSE	CLOSE	Cierra gripper neumático
CPATCH	CPATH LOC1, LOC2…	LOC1: Nombre de una localización o variable que especifique el nombre de una localización.
HERE	HERE A1	A1: Es una localización creada.
JOINT	JOINT #ART, GRADOS	#ART: Número de articulación que se moverá.
LIMP	LIMP #EJE	#EJE: Valor o nombre de variable que especifica el número de articulación que deshabilita el freno.
LOCK	LOCK #ART	#ART: Número de articulación que se bloqueará.
MOTOR	MOTOR #MOTOR, PULSOS	#PULSOS: Valor o nombre de variable que especifica el número de articulación que se moverá cierto número de pulsos.
MOVE	MOVE LOC,S	LOC: Es el nombre de una localidad o variable que especifica una localidad.
NOLIMP	NOLIMP #EJE	#EJE: Valor o nombre de variable que especifica el número de articulación en el que se habilitará el freno.
OPEN	OPEN	Abrir gripper neumático.
READY	READY	Mueve al brazo a la posición ready (posición home).
INPUT	INPUT VAR,1	Es el nombre de una variable donde se almacenará información.
UNLOCK	UNLOCK #ART	#ART: Número de articulación que se desbloqueará.
OUTPUT	OUTPUT SIGNO #SALIDA	#SALIDA: Nombre de una salida que afectará una ejecución.
HOME	HOME	Alinea el brazo robótico.
PITCH	PITCH INCREMENTO	INCREMETO: Valor específica en Angulo que se moverá la muñeca.
ROLL	ROLL INCREMENTO	INCREMETO: Valor específica en Angulo que se rotará la muñeca.
WAIT	WAIT SIGNO #ENTRADA	#ENTREDA: especifica la entrada que será evaluada.
DLOCN	DLOCN #ART	#ART: Es el nombre de una variable específica a borrarse.
DELAY	DELAY TIEMPO	TIEMPO: Es el valor de tiempo en segundos.
FINISH	FINISH	Finalización de un programa.
PAUSE	PAUSE	Detiene el flujo de un programa y despliega un mensaje.
RUN	RUN	Carga, compila y corre a un programa.

La selección de estos comandos se ha realizado ya que no trabajan en coordenadas cartesianas, tampoco requieren la translación del sistema de coordinas al TCP (Punto central de la herramienta). Además al incorporar los comandos seleccionados no se requiere, de cálculos de complejos para realizar generación de trayectorias.

3.2.3 Diseño de Estructura de Comandos

Tomando en cuenta la estructura de comandos de RobComm se establece la nueva estructura sintáctica de intercambio de información de todo el sistema, desde la interfaz gráfica de usuario (nivel 4) hasta los microcontroladores "esclavos" (nivel 2).

3.2.3.1 Estructura de Envío de Comando de Usuario a Java

La nomenclatura usada es similar a la de RobComm, se escribe el nombre del comando, número de articulación (del 1 a 5), el sentido con las letras SA o SH (sentido horario o anti-horario), un número que especifica ángulo o pulsos (número entero), nombre de variable y para finalizar un comando, se añade punto y coma. La estructura de un comando es el siguiente. (Ver figura. 62)

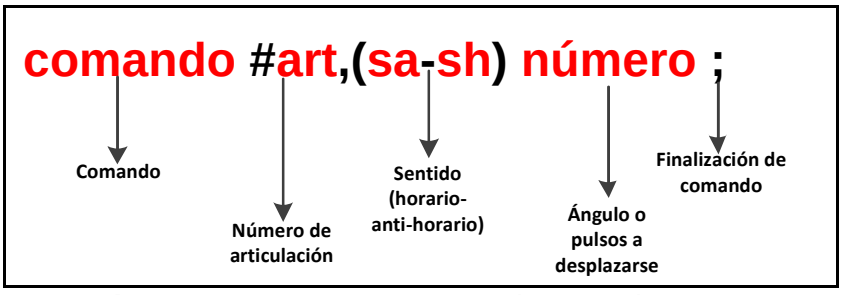


Figura. 62 Estructura comando Usuario-Java.

3.2.3.2 Estructura de Envío de Comando de Java a Maestro

El microcontrolador 18F452 recibirá una cadena de caracteres por lo tanto, la nomenclatura, para cada comando se designará una letra del abecedario, una vocal que identifique la articulación, el sentido se representa con las letras H o A (sentido horario o anti-horario), un número que especifica ángulo o pulsos (número entero) y para finalizar un comando, se añade un retorno de carro o Enter (\r). La estructura de un comando es el siguiente. (Ver figura. 63)

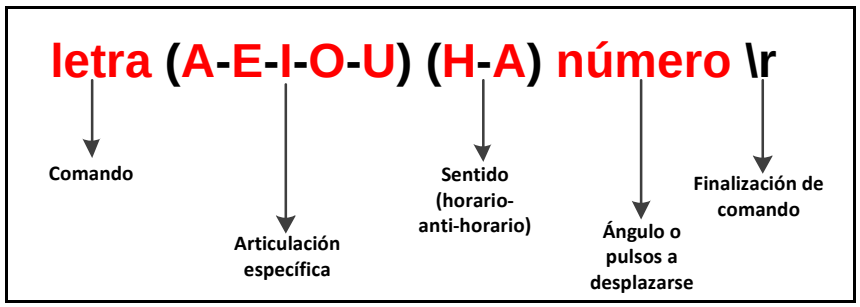


Figura. 63 Estructura comando Java-Maestro.

3.2.3.3 Estructura de Envío de Comando de Maestro a Esclavo

Un microcontrolador "esclavo" 16F88 recibirá una arreglo de bytes, por lo tanto la nomenclatura, para cada comando se designará un número de dos dígitos, el sentido se representa con 1 o 2 (sentido horario o anti-horario), un número que especifica ángulo o pulsos (número entero de 16 bits), no requiere un carácter de finalización de comando. La estructura de un comando es el siguiente. (Ver figura. 64)

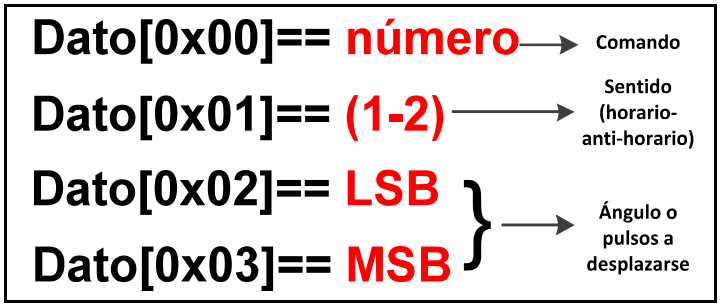


Figura. 64 Estructura comando Maestro-Esclavo.

Cabe recalcar que va depender del comando si se asigna valores como: número articulación, sentido, ángulo o pulsos; si este lo requiere para cumplir su función.

A continuación se detallan la sintaxis para el envío de un comando entre diferentes niveles (Ver tabla. 13)

Tabla. 13 Sintaxis de comandos para él envió de datos entre niveles.

COMANDO	NIVELES	SINTAXIS	OBSERVACIÓN
CLOSE	User-Java	CLOSE;	Comando enviado solo al primer esclavo.
	Java-Master	A\r	
	Master-Slave	10	
CPATH	User-Java	CPATH ID, ID,..;	Son comandos MOVE consecutivos.
	Java-Master	-	
	Master-Slave	-	
HERE	User-Java	HERE ID;	Comando enviado a todos los esclavos
	Java-Master	C\r	
	Master-Slave	12	
JOINT	User-Java	JOINT #ART,(SA-SH)ANG ;	Comando enviado a una articulación a la vez.
	Java-Master	D (A-E-I-O-U) (H-A) ANG \r	
	Master-Slave	13 (1-2) NUMERO	
LIMP	User-Java	LIMP #ART;	Comando enviado a una articulación a la vez.
	Java-Master	E (E-I-O-U) \r	
	Master-Slave	14	
LOCK	User-Java	LOCK #ART ;	Comando enviado a una articulación a la vez.
	Java-Master	F (A-E-I-O-U) \r	
	Master-Slave	-	
MOTOR	User-Java	MOTOR #ART,(SA-SH) PULSOS;	Comando enviado a una articulación a la vez.
	Java-Master	G (A-E-I-O-U)(H-A) PULSOS \r	
	Master-Slave	16 (1-2) NUMERO	
MOVE	User-Java	MOVE ID ;	Comando enviado a una articulación a la vez.
	Java-Master	H (A-E-I-O-U) (1-0), NUMERO \r	
	Master-Slave	17 (1-0) , NUMERO	
NOLIMP	User-Java	NOLIMP #ART ;	Comando enviado a una articulación a la vez.
	Java-Master	I (E-I-O-U) \r	
	Master-Slave	18	
OPEN	User-Java	OPEN;	Comando enviado solo al primer esclavo.
	Java-Master	J \r	
	Master-Slave	19	
READY	User-Java	READY;	Este comando se envía a todos los esclavos
	Java-Master	K \r	
	Master-Slave	20	
INPUT	User-Java	INPUT NUM,ID ;	Activa una entrada digital del pic “maestro”
	Java-Master	L(1-4)	
	Master-Slave	-	

Continúa en la siguiente página

COMANDO	NIVELES	SINTAXIS	OBSERVACIÓN
UNLOCK	User-Java	UNLOCK #ART ;	Comando enviado a una articulación a la vez.
	Java-Master	M (A-E-I-O-U) \r	
	Master-Slave	-	
OUTPUT	User-Java	OUTPUT (+/-) NUM ;	Activa una salida digital del pic "maestro"
	Java-Master	N (1-0)(1-4) \r	
	Master-Slave	-	
HOME	User-Java	HOME;	Este comando se envía a todos los esclavos
	Java-Master	O \r	
	Master-Slave	22	
ROLL	User-Java	ROLL (SA-SH) ANG;	Comando enviado solo al quito esclavo.
	Java-Master	Q (H-A) NUM \r	
	Master-Slave	24 NUM	
WAIT	User-Java	WAIT (+/-) (NUM);	Analiza el estado de una entrada digital del pic "maestro".
	Java-Master	R (A-B) NUM \r	
	Master-Slave	-	
DLOCN	User-Java	DLOCN ID ;	Comando interno de Java.
	Java-Master	-	
	Master-Slave	-	
DELAY	User-Java	DELAY NUM ;	Pausa momentánea en milisegundos pic "maestro".
	Java-Master	T NUM \r	
	Master-Slave	-	
FINISH	User-Java	FINISH;	Comando interno de Java.
	Java-Master	-	
	Master-Slave	-	
PAUSE	User-Java	PAUSE;	Comando interno de Java.
	Java-Master	-	
	Master-Slave	-	
RUN	User-Java	RUN;	Comando interno de Java.
	Java-Master	-	
	Master-Slave	-	
VAR	User-Java	VAR ID=NUM ;	Comando interno de Java.
	Java-Master	-	
	Master-Slave	-	
INT	User-Java	INT ID (INC) NUM;	Comando interno de Java.
	Java-Master	-	
	Master-Slave	-	

Descripción de la tabla:

- **ID**: Variable alfanumérica.
- **ART:** Articulación.
- **SA-SH:** Sentido horario u anti-horario.
- **ANG:** Ángulo (entero).

- **(A-E-I-O-U):** Asignación de una articulación.
- **(H-A):** Sentido horario u anti-horario.
- **NUM:** Número entero.
- **PULSOS:** Número de pulsos (entero).
- **INC:** Incremento o decremento (=, +=,-=).

En la tabla anterior se puede observar que se han añadido comandos extras a los conocidos de RobComm:

- **VAR:** Este comando servirá para la declaración de variables de tipo entero, en esta variable se guardará valores obtenidos por el comando INPUT o el comando WAIT.

- **INT:** Este comando sirve para actualizar un valor con un incremento o decremento numérico, dentro de una variable previamente declarada. Este comando servirá para poder realizar lazos de repetición dentro de una sentencia.

Adicionalmente se estructura una sentencia de condición y una sentencia de repetición:

- **IF:** Sentencia condicional, si la sentencia es verdadera cumplirá instrucciones, por falso puede realizar o no realizar otras instrucciones. Su estructura es la siguiente. (Ver figura. 65)

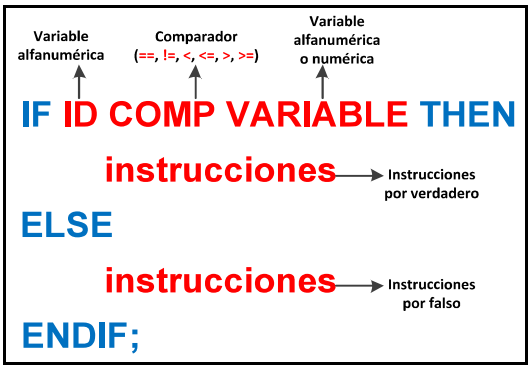


Figura. 65 Sentencia IF.

- **WHILE:** Sentencia de repetición: Si la condición se cumple entra en un lazo o bucle de repetición, cuando la condición deja de cumplirse sale mismo. Su estructura en la siguiente. (Ver figura. 66)

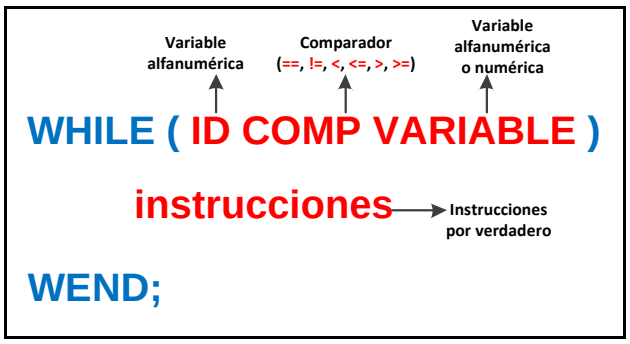


Figura. 66 Sentencia WHILE.

Las sentencias de condición y repetición se añadieron para reemplazar a las sentencias que existían para el control de programa como salto de etiqueta (GOTO) en RobComm. Ya que la estructuras diseñadas son más entendibles y manejables para el programador.

3.2.4 Diseño del Algoritmo de Posicionamiento de Motores

Se pretende establecer un algoritmo general para el desplazamiento simultáneo de los motores de un punto de localización (coordenadas angulares de todas las articulaciones) a otro, para lo cual se enumera los pasos a seguir:

```
Leer posición deseada: PosD
Leer posición actual: PosA
Calcular pulsos y sentido a moverse:
        SI sentidodeseado==sentidoactual ENTONCES
            SI PosD>PosA ENTONCES
                    Calcular: PosF=PosD-PosA
                    SI sentidodeactual==1 ENTONCES
                            Colocar : SenM=1
                    SI NO
                            Colocar: SenM=0
                    FINSI
            SI NO
                    SI PosA>PosD ENTONCES
                    Calcular: PosF=PosA-PosD
                            SI sentidoactual==1 ENTONCES
                                    Colocar: SenM=0
                            SI NO
                                    Colocar: SenM=1
                            FINSI
                    SI NO
                            PosF=0
                            Desactivar PWM
                            Habilitar freno
                    FINSI
        SINO
            PosF=PosD+PosA
            SI sentidodeseado==1 ENTONCES
                    SenM=1
            SI NO
                    SenM=0
            FINSI
        FINSI
Realizar: el movimiento de una articulación
```

3.2.5 Diseño de Programación de Microcontroladores

Una vez conocido el panorama general del funcionamiento del sistema, se realiza los programas necesarios en los microcontroladores para la ejecución de los comandos...

3.2.5.1 Programación de Comandos en el "Maestro"

- **Programa principal**

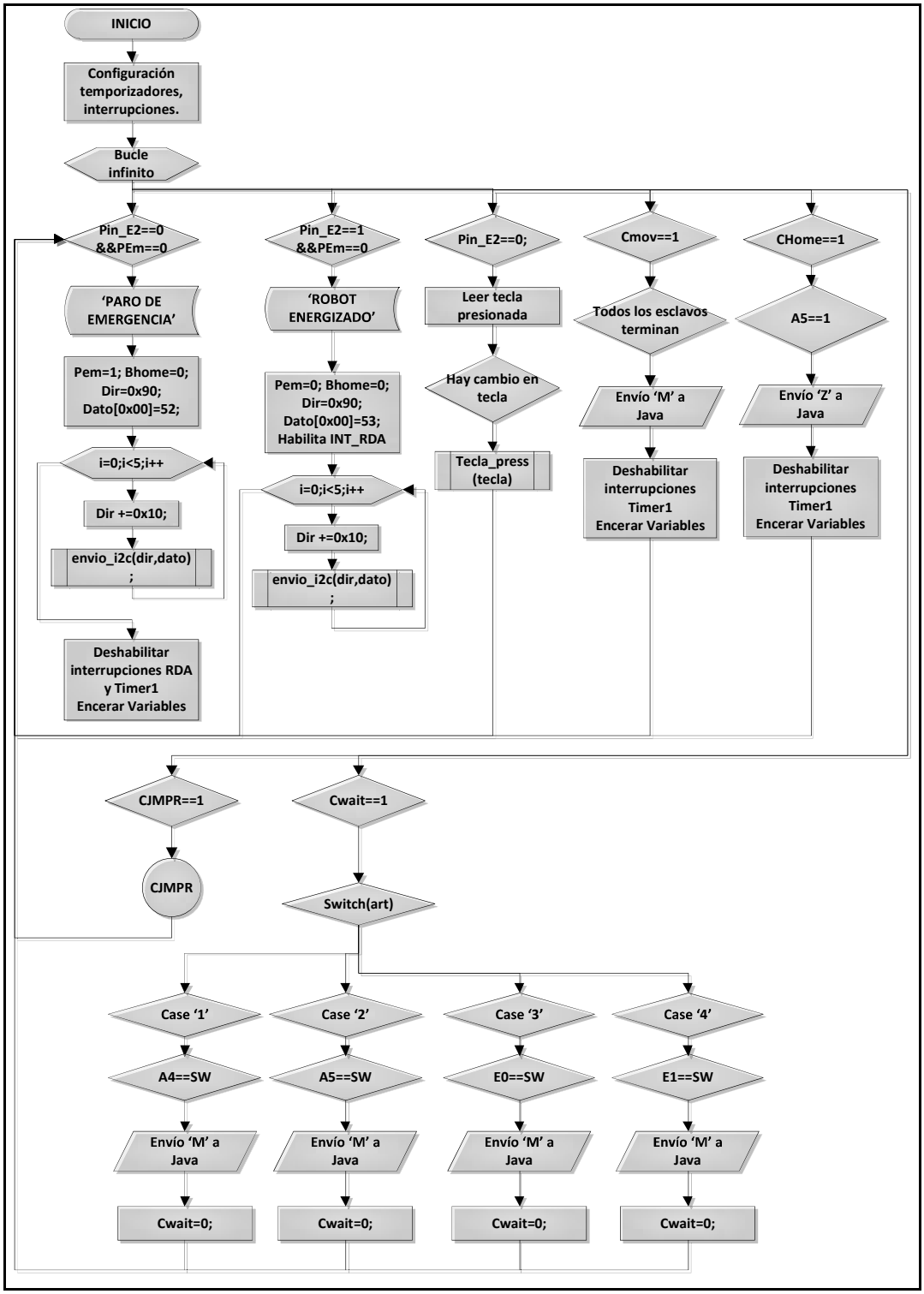


Figura. 67 Programa principal del pic "maestro" parte 1.

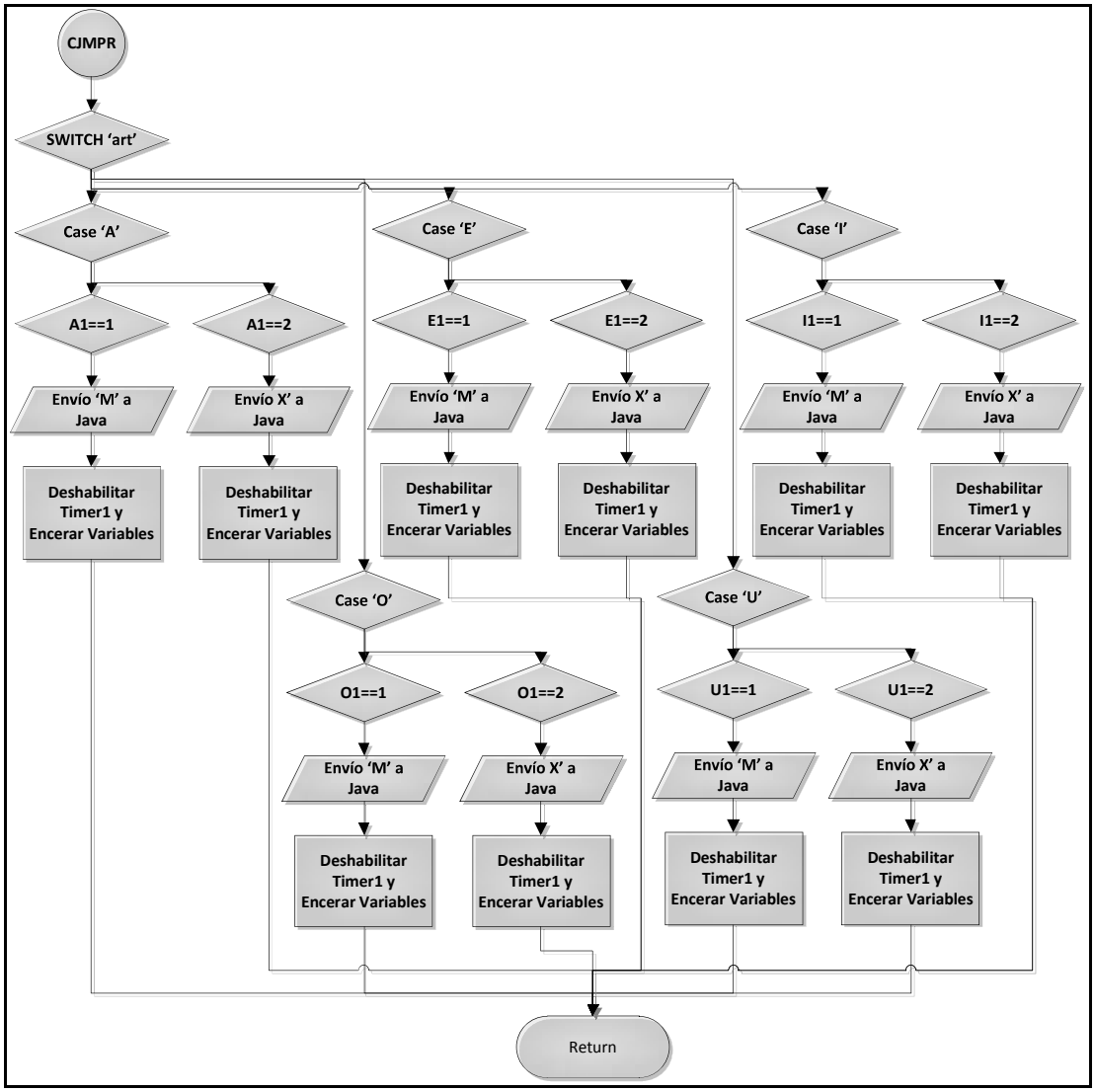


Figura. 68 Programa principal del pic "maestro" parte 2.

Tabla. 14 Variables implementadas en el pic "maestro".

VARIABLE	DESCRIPCIÓN
char l[40];	Almacena un string enviado desde la PC
char * ptr=0;	Puntero de Arreglo l (Datos recibidos)
char * ptr1;	Puntero del contenido numérico de la Variable SPEED
int16 Num=0;	Almacena el número de cualquier comando
char c;	Primer dato del String recibido
char art;	almacena el número de articulación
char sen;	almacena el sentido del motor a mover

Continúa en la siguiente página

VARIABLE	DESCRIPCIÓN
int8 recibo[0x05];	Arreglo de byte que almacena los datos en modo lectura
int8 dato[0x04], dir=0, dir1=0;	Arreglo de byte que almacena los datos en modo escritura
int1 La=0, Le=0, Li=0, Lo=0, Lu=0;	Banderas del comando LOCK
int16 PosF=0;	Almacena la posición final de cada articulación
int8 tecAc=0, in=0;	Bandera para recibir el dato por teclado
char tecAn;	Bandera que almacena la tecla presionada anteriormente.
int8 A1=0, A2=0, A3=0, A4=0, A5=0;	Banderas para saber cuándo acabo todos los movimientos
int1 SW=0;	Almacena 1 o 0 para el comando wait
int1 CMov=0, CHome=0, CJMPR=0, CWait=0;	Habilita el control de acabado de un movimiento
int1 PEm=0;	Bandera de paro de emergencia
int1 BHome=0;	Bandera de homeo

- **Funciones de recibo y envío por I2C**

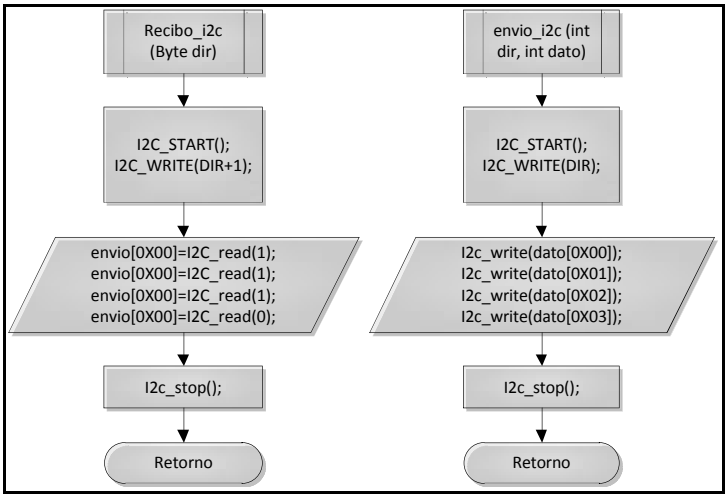


Figura. 69 Funciones de en recibo y envío por I2C.

Estas funciones permiten ser llamadas en cualquier punto del programa cuando sean necesarias.

- **Interrupción por I2C**

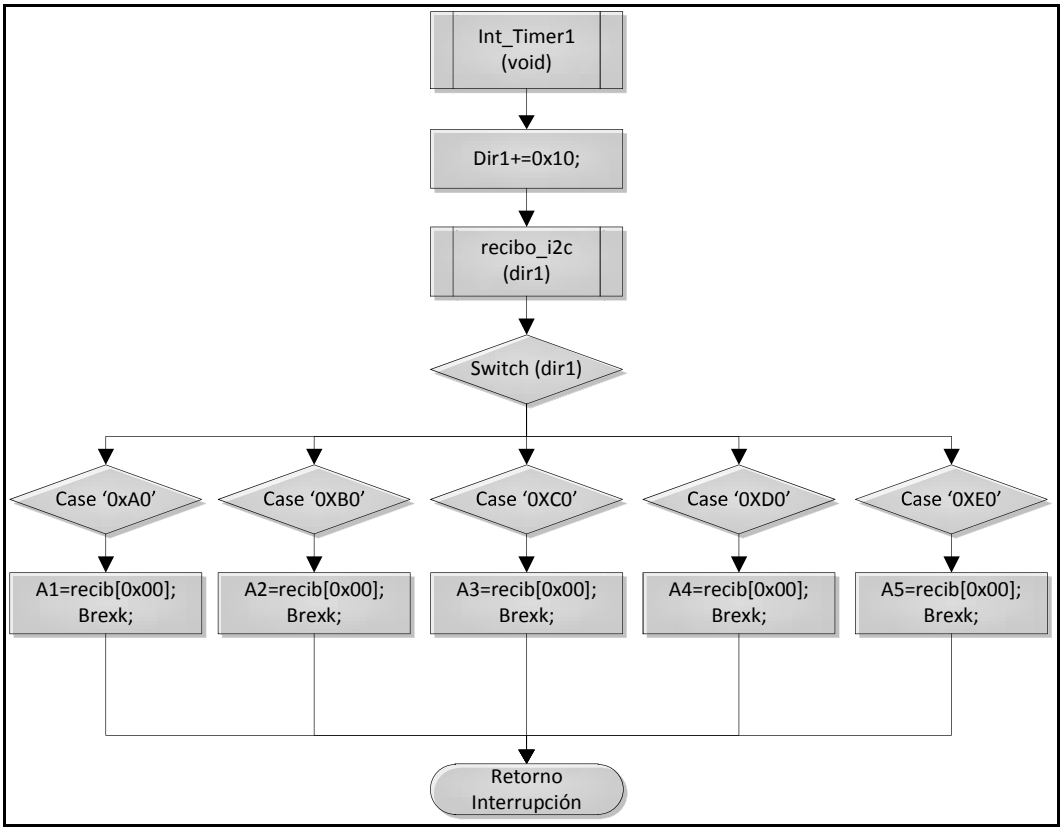


Figura. 70 Interrupción por I2C.

Esta interrupción permite conocer si el esclavo ha acabado la ejecución de una instrucción.

- **Función para detección de tecla presionada**

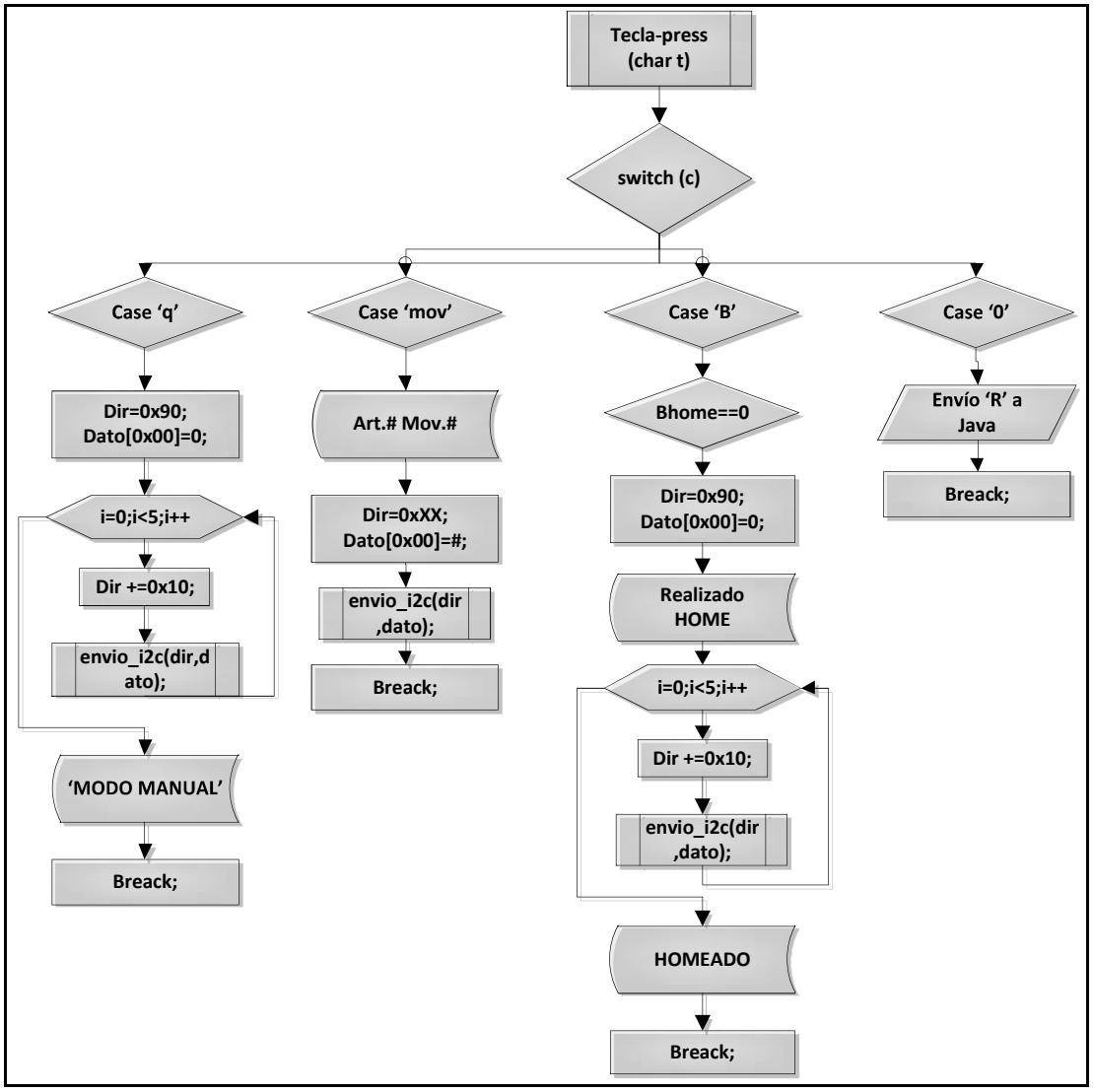


Figura. 71 Función Tecla presionada.

Esta función permite conocer la operación que el usuario ha presionado desde el teach pendant.

- **Interrupción Serial**

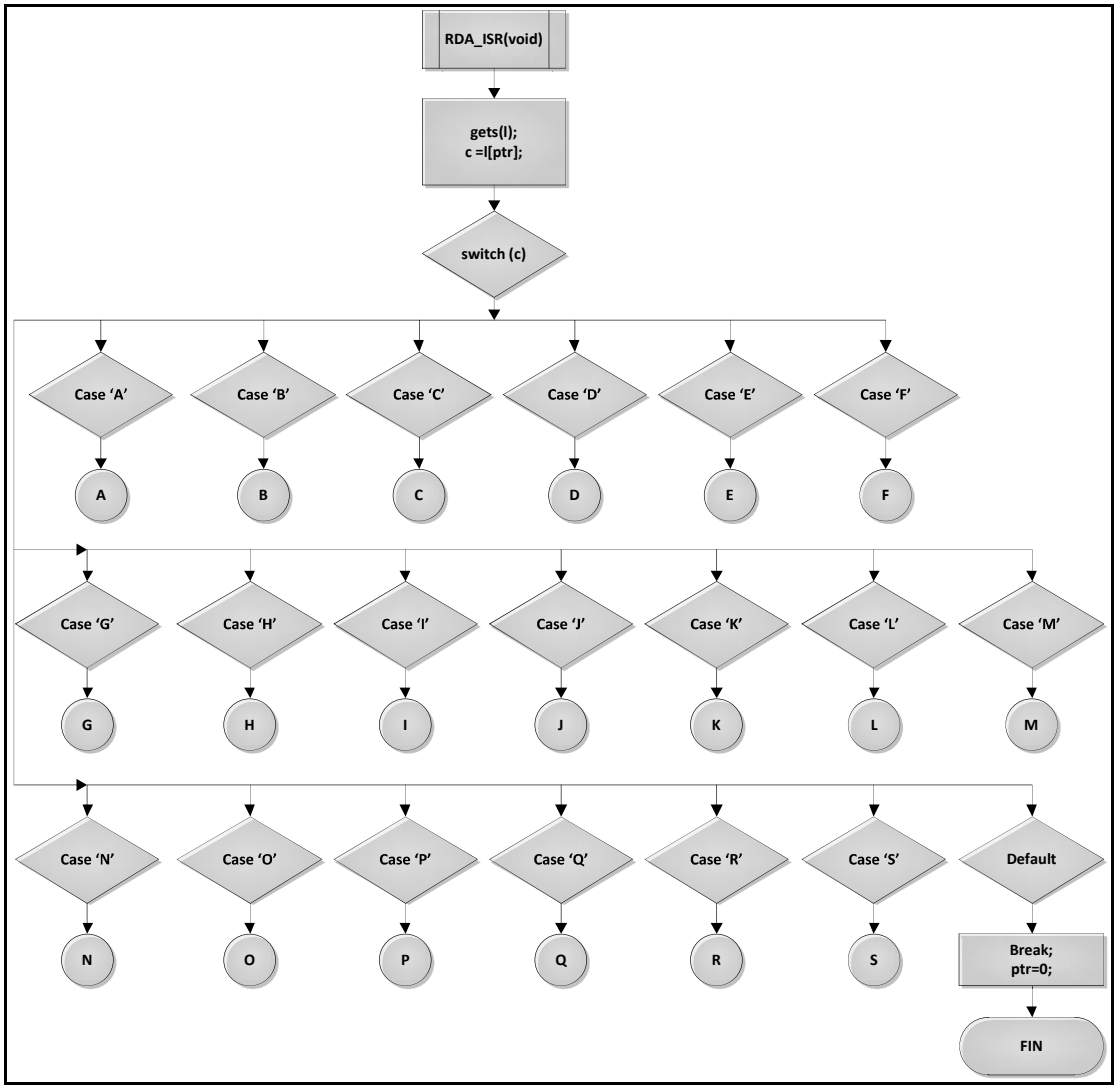


Figura. 72 Interrupción Serial.

El diagrama representa la operación que se debe realizar, analizando el primer carácter del string que se recibe por parte del programa.

- **Comandos para gripper**

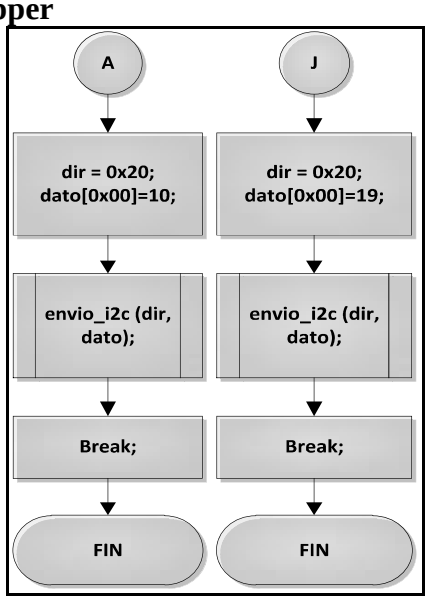


Figura. 73 Comandos para gripper.

Envía al primer esclavo la orden de abrir o cerrar el gripper del manipulador.

- **Comando Here**

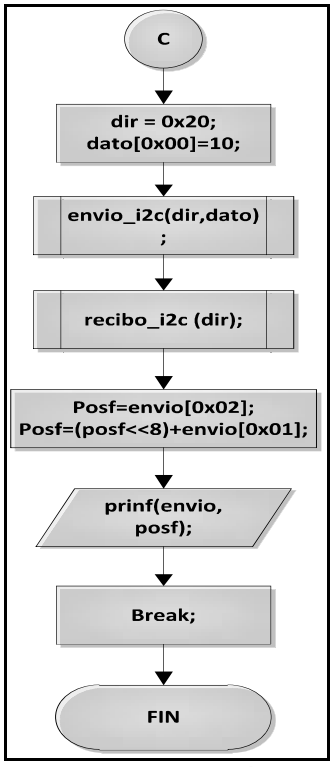


Figura. 74 Comando Here.

Este comando permite conocer la posición actual de todas las articulaciones, esto lo realiza solicitando la información a cada esclavo.

- **Comandos Limp y NoLimp**

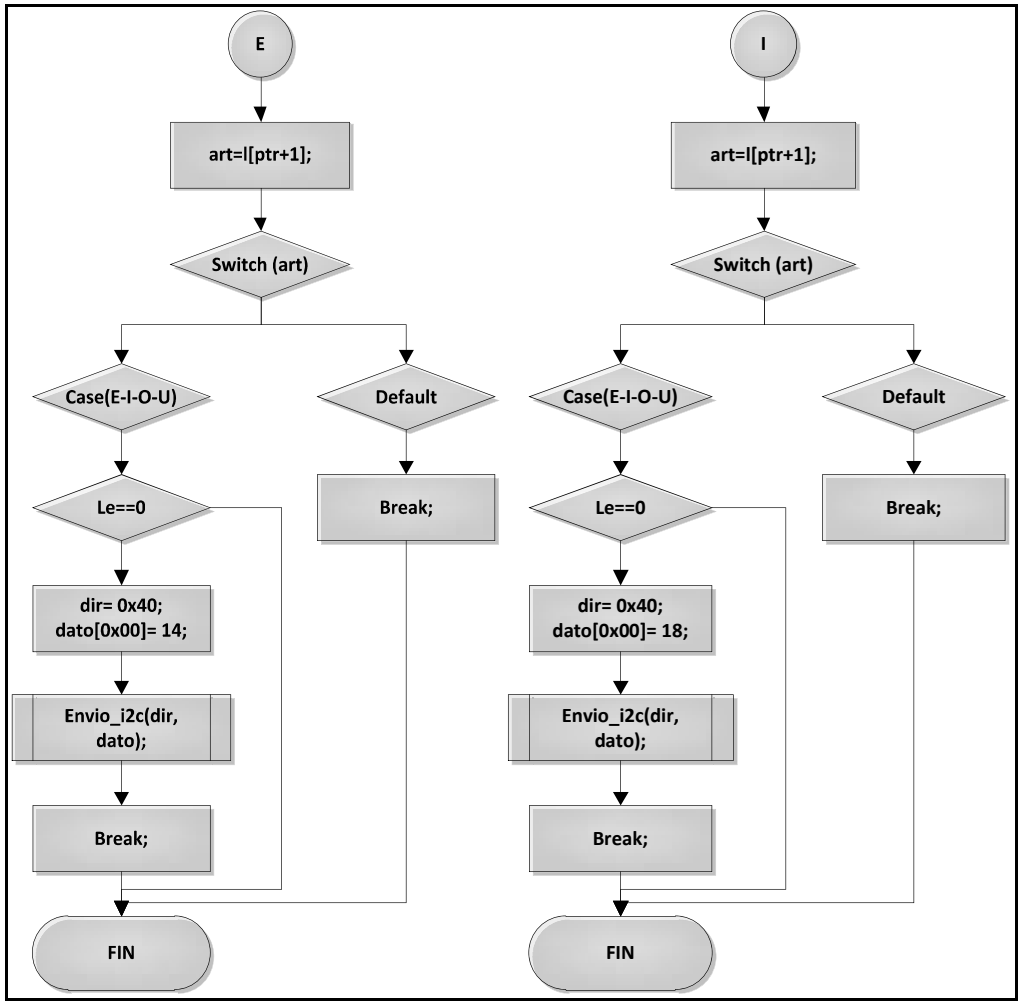


Figura. 75 Comandos Limp y NoLimp.

Se envía una orden a un “esclavo” específico para que habilite o deshabilite el bit de activación de freno en una articulación.

- **Comandos Ready y Home**

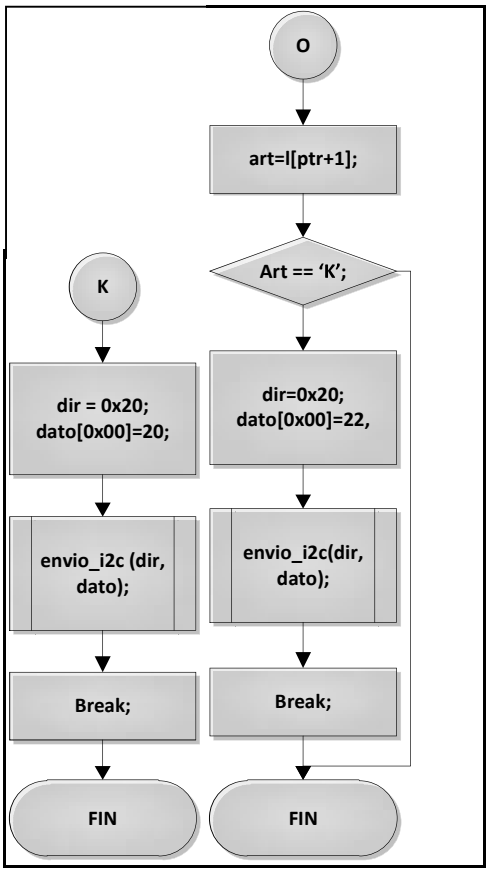


Figura. 76 Comandos Ready y Home.

El comando ready envía a todos los esclavos que regresen a la posición inicial, mientras que el comando home envía la orden a todos los esclavos que mover 5 grados a cada sentido para realizar el movimiento home, este comando solo realiza una vez, al comenzar la manipulación del robot.

- **Comandos Lock y Unlock**

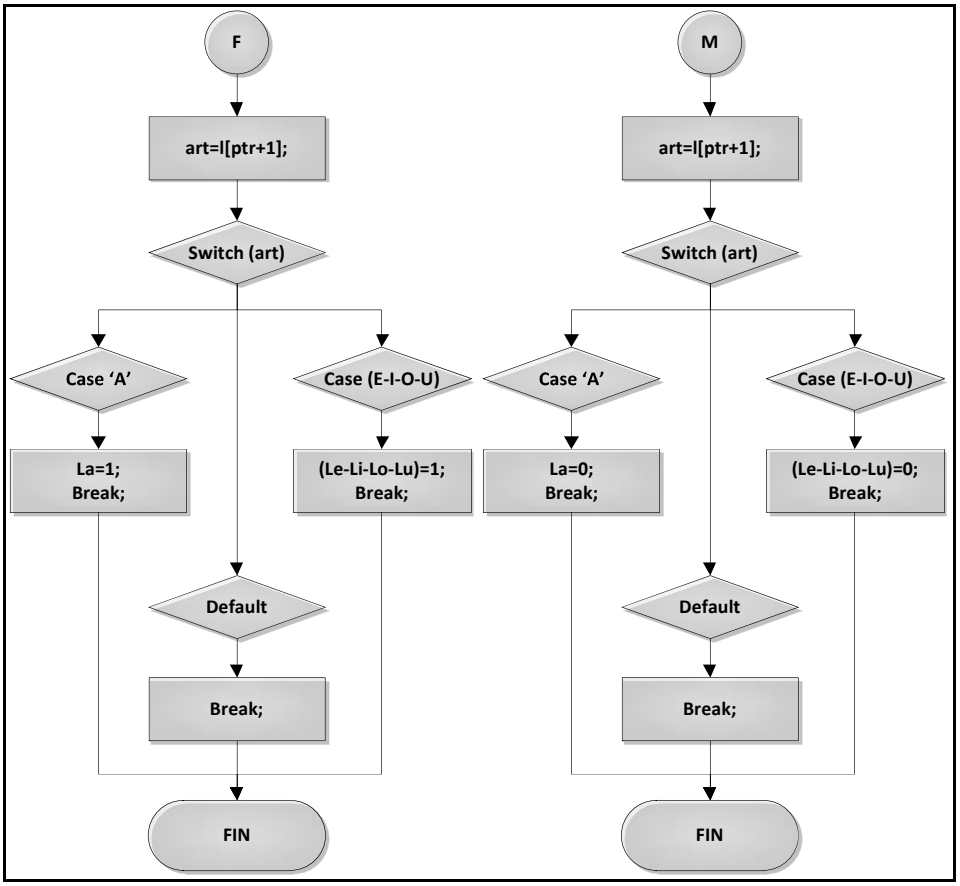


Figura. 77 Comandos Lock y Unlock.

Este comando se encarga de activar o desactivar las banderas correspondientes para que una articulación no realice instrucciones de operación.

- **Comando Input**

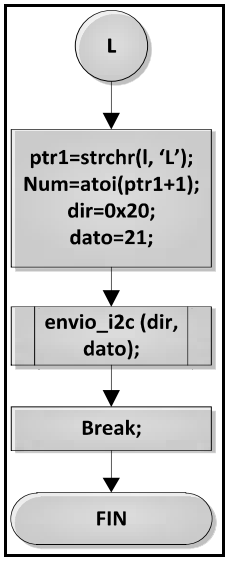


Figura. 78 Comando Input.

Evalúa el estado actual de una entrada digital específica.

- **Comando Output**

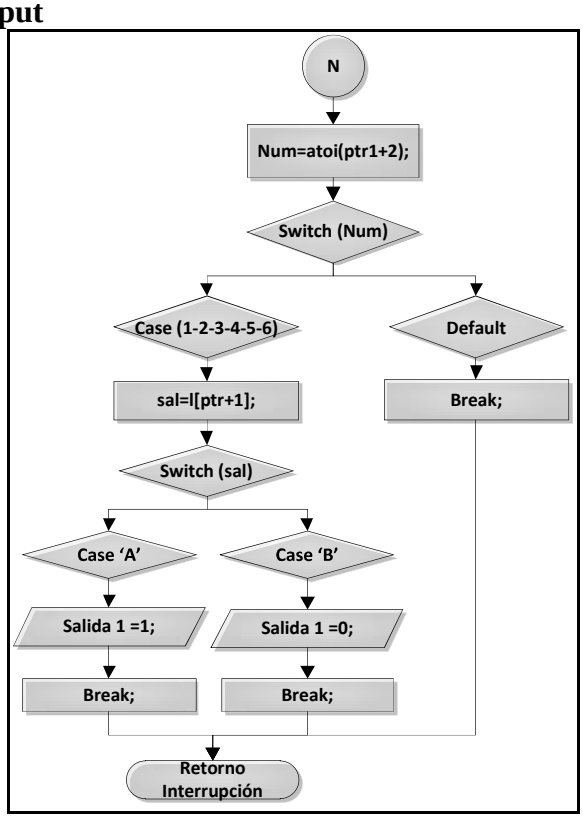


Figura. 79 Comando Output.

Activa o desactiva la salida digital seleccionada.

- **Comando Wait**

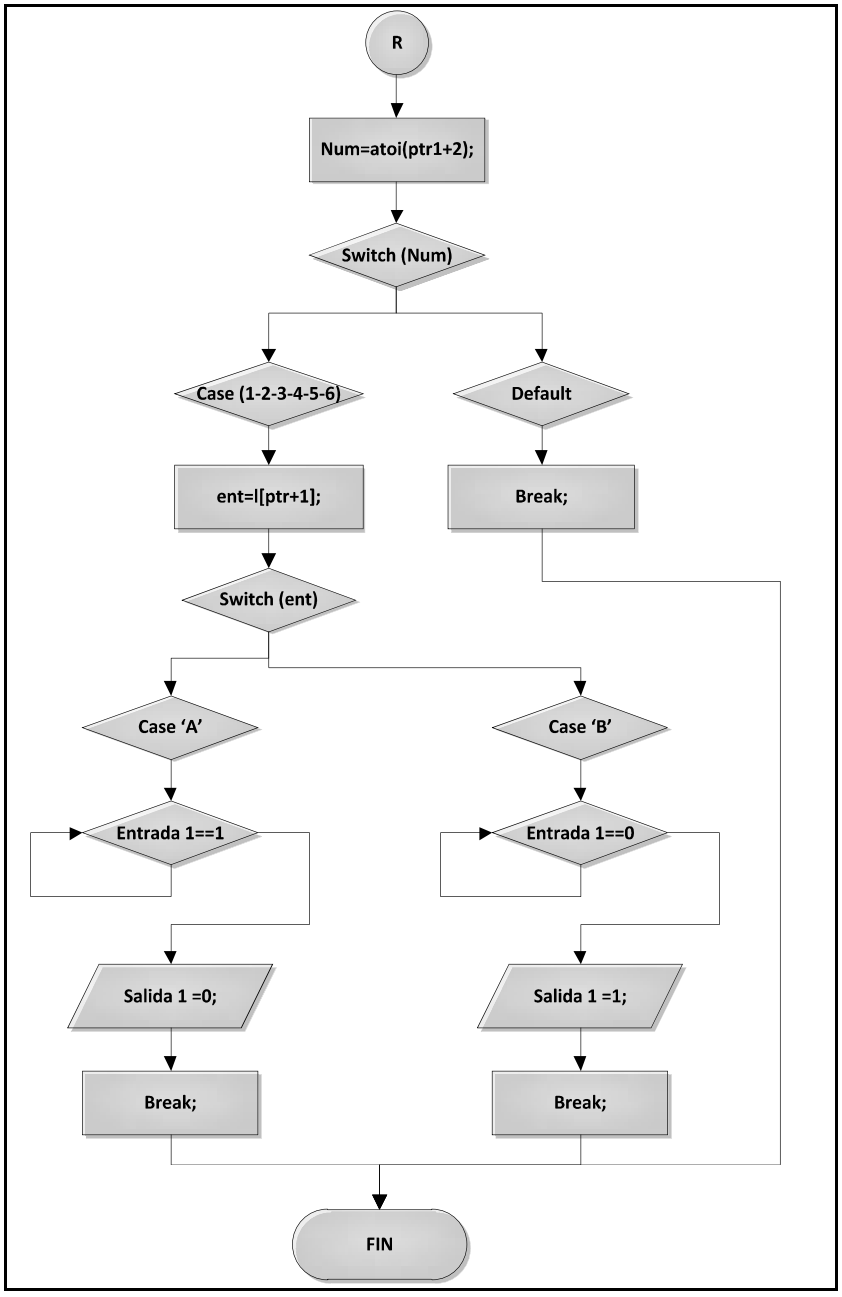


Figura. 80 Comando Wait.

Espera hasta que la condición de encendido o apagado de una entrada digital se cumpla para enviar una respuesta a Java.

- **Comando Joint, Motor, Pitch, Roll**

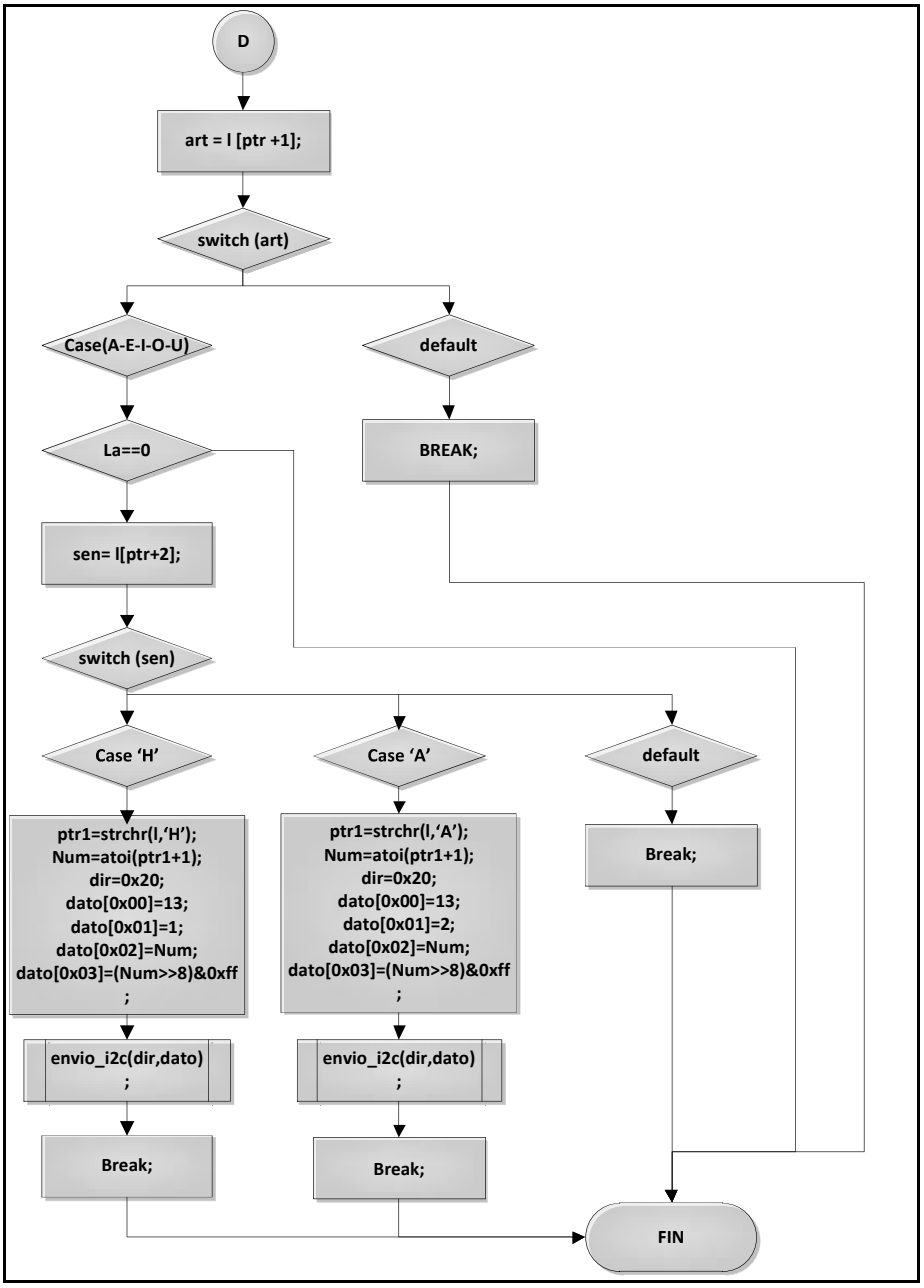


Figura. 81 Comando Joint.

Este diagrama funciona de forma similar para los cuatro comandos, envía a un esclavo específico los datos de posición y sentido al cual se debe mover un motor.

- **Comando Move**

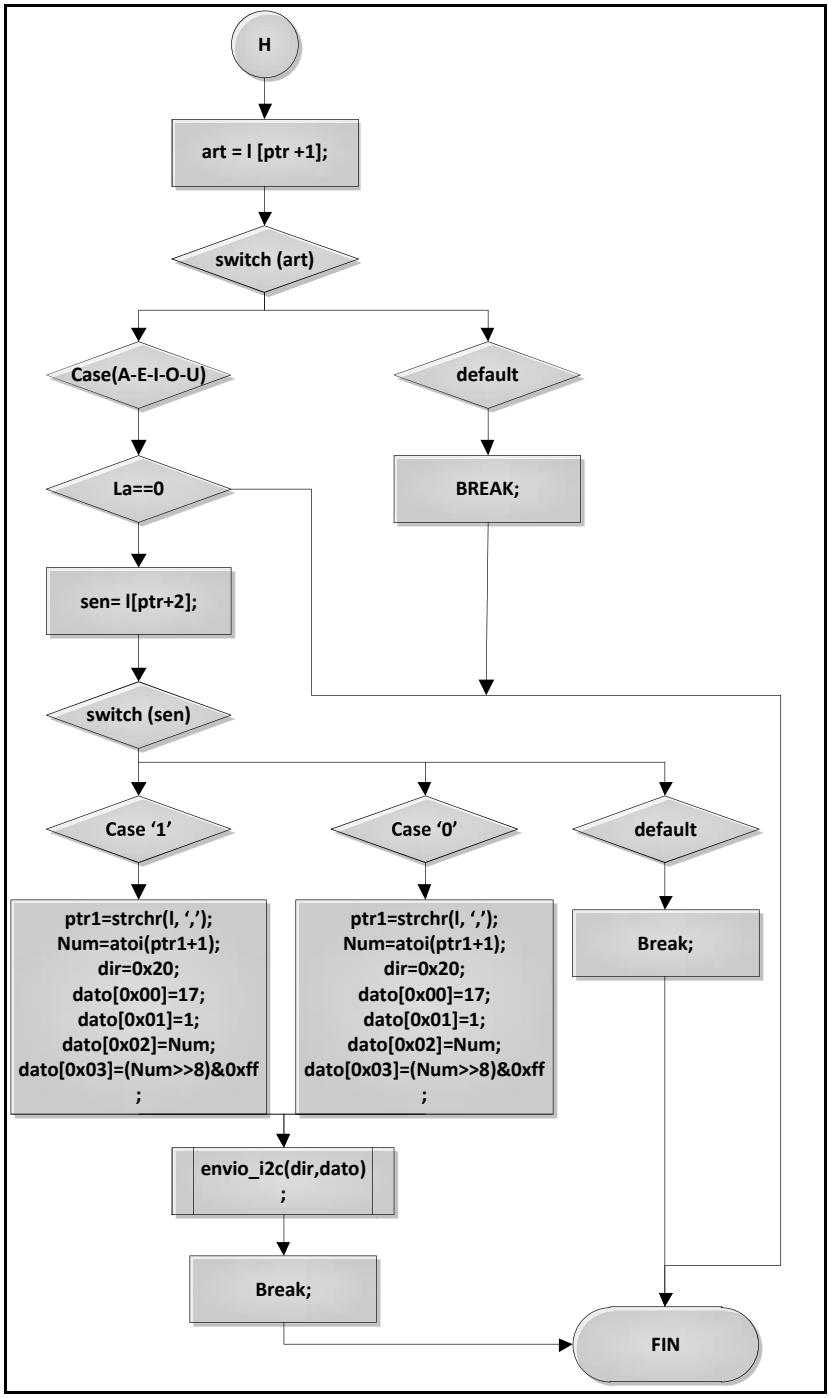


Figura. 82 Comando Move.

Este comando es similar a Joint, pero envía la información a todos los esclavos en un tiempo determinado.

3.2.5.2 Programación de Comandos en “Esclavos”

- **Programa Principal**

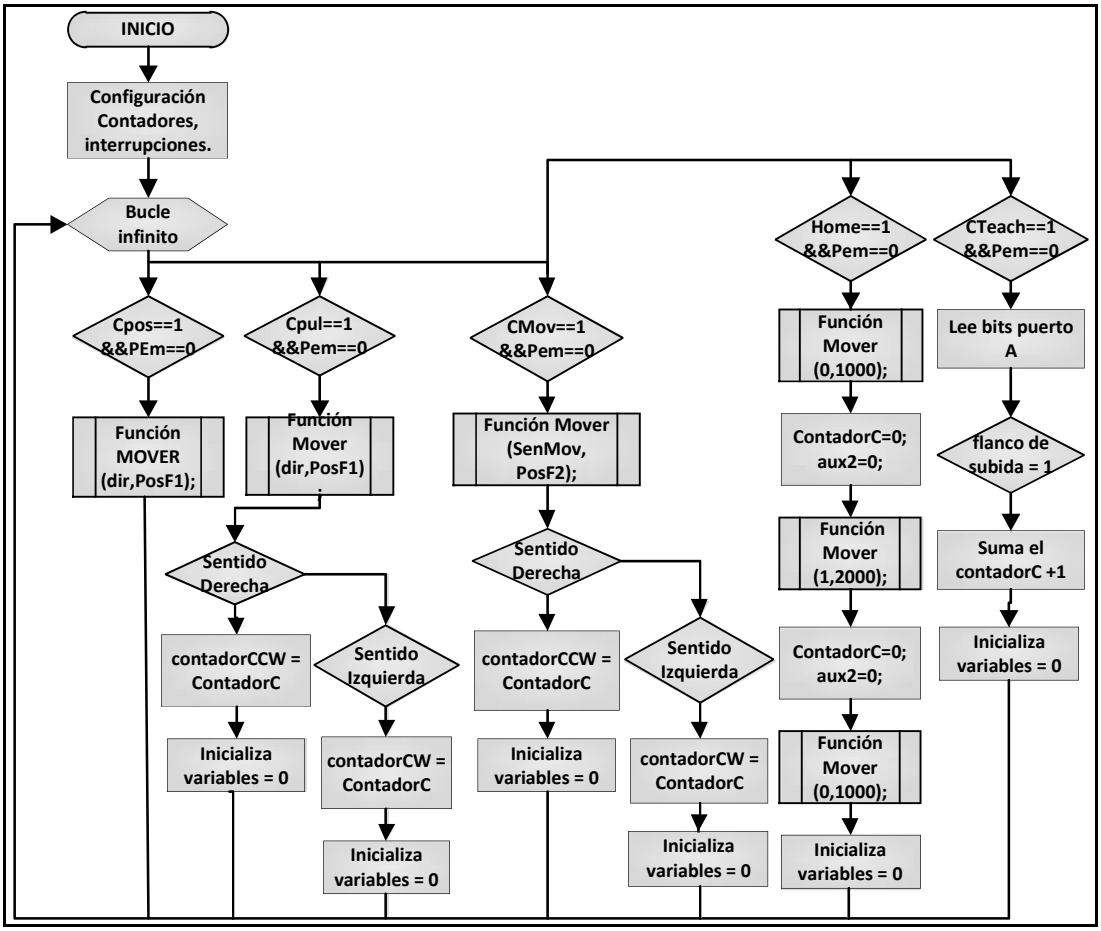


Figura. 83 Diagrama del programa principal.

En el programa principal se encuentran todos los lazos de control de los comandos: Here, Move, Motor, Joint, Pitch, Roll, Home, cuando se activan sus correspondientes banderas estos comandos ejecutan el movimiento del motor con el numeró de pulsos, y sentido respectivos, además lleva una cuenta de pulsos y reinicia el estado de las variables.

- **Función Mover**

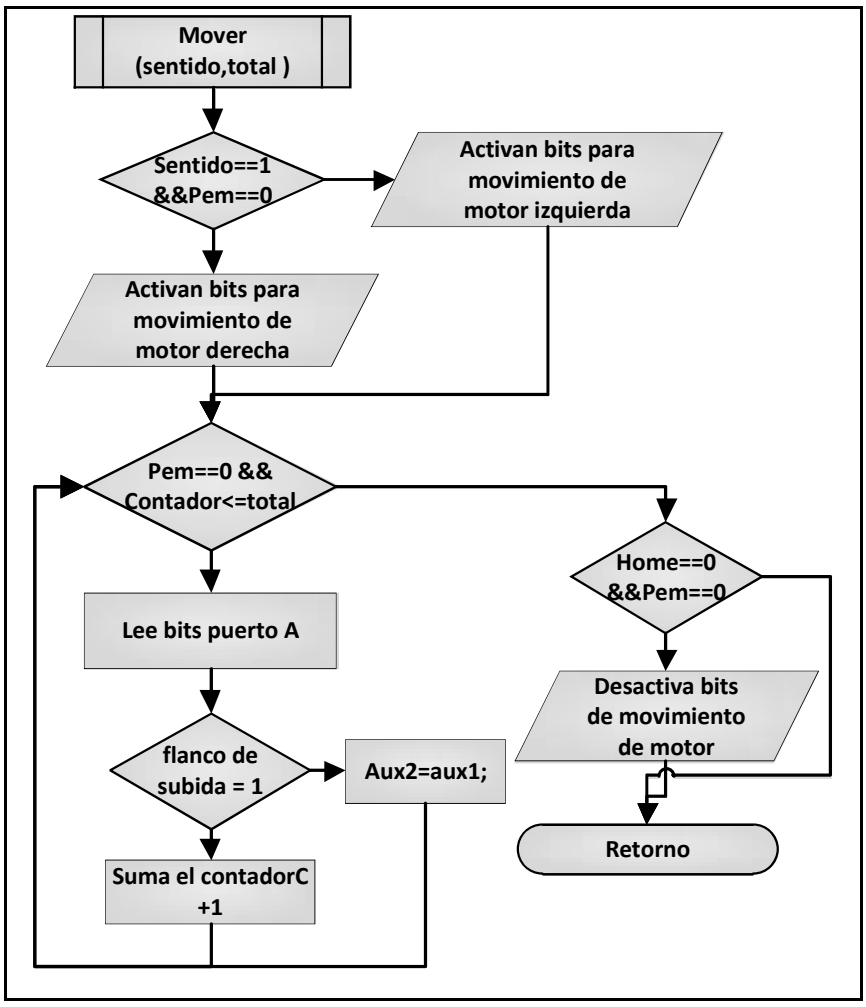


Figura. 84 Diagrama de función Mover.

Esta función es llamada por los comandos de movimiento, tiene como parámetros el sentido y números de pulsos a mover, realiza el movimiento del motor y una vez finalizado lo detiene llevando una cuenta del movimiento realizado.

- **Interrupción I2C**

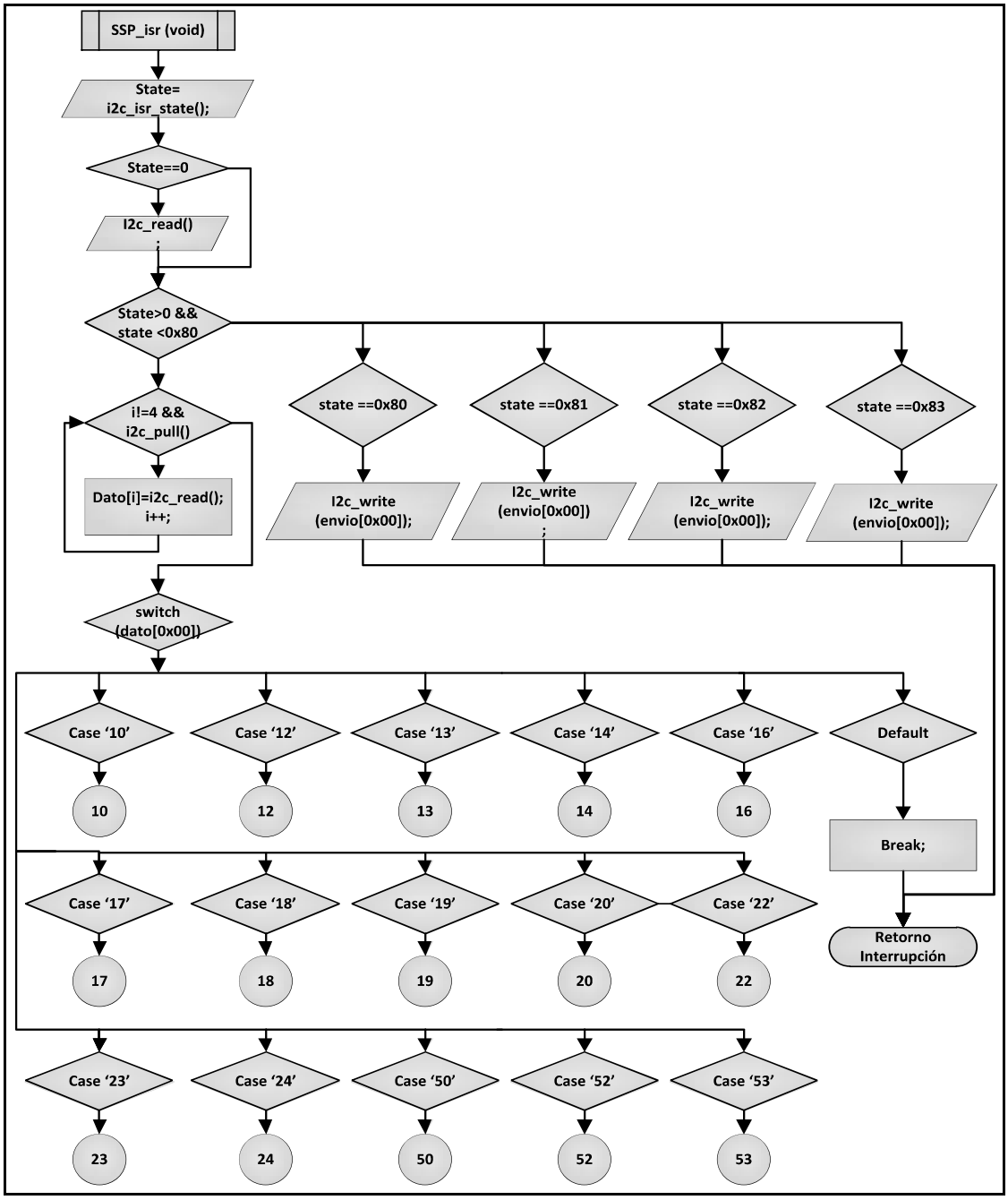


Figura. 85 Diagrama interrupción I2C.

Es la interrupción por el bus I2C, esta interrupción se activa cuando el pic maestro envía datos, esta los recibe, conociendo el comando a realizar además recibe parámetros como sentido y números de pulsos si los comandos lo requieren para su ejecución.

- **Interrupción Externa**

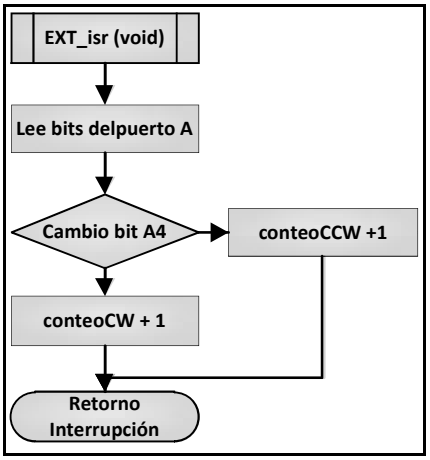


Figura. 86 Diagrama de interrupción externa.

Esta interrupción es habilitada por el comando Limp, para el conocer el sentido del motor cuando el movimiento es libre, es decir, sin alimentación del motor.

- **Comandos de Gripper**

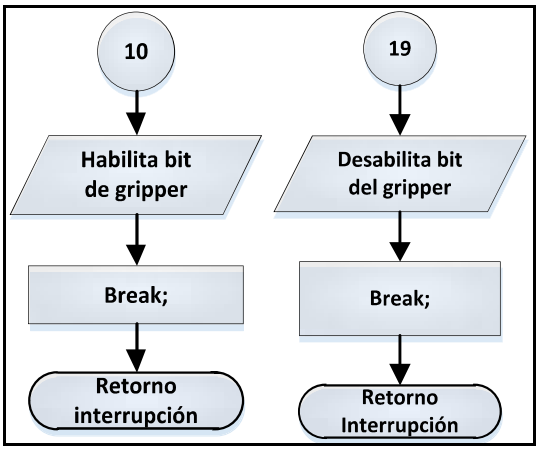


Figura. 87 Diagrama de comandos de gripper.

Estos comandos activan o desactivan el bit que controla el solenoide del manipulador, este comando solo es controlado por el primer microcontrolador maestro.

- **Comando Here**

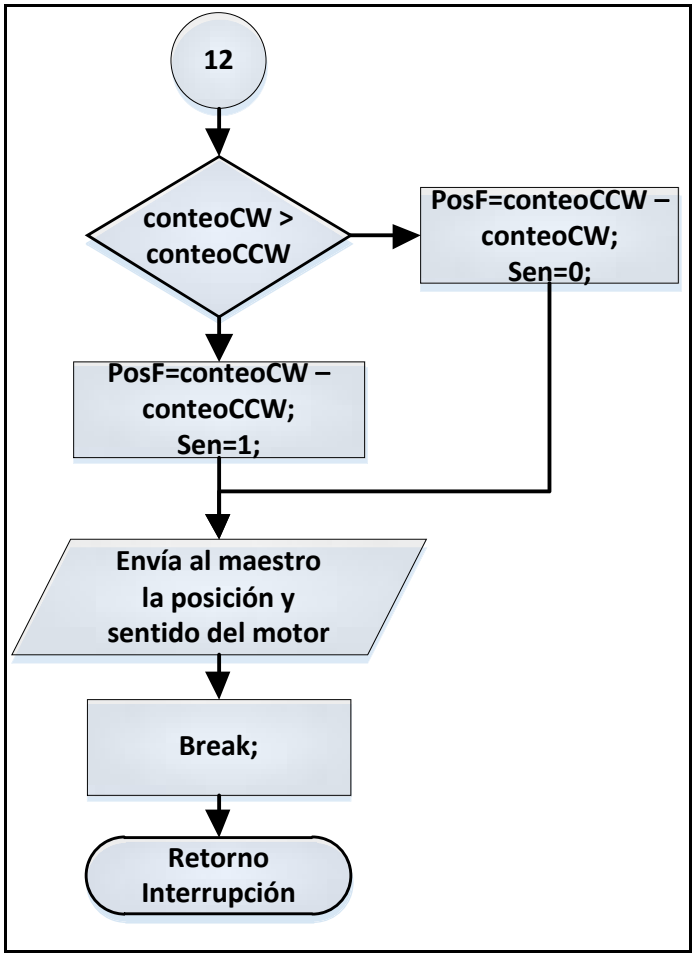


Figura. 88 Diagrama de comando Here.

El comando Here envía al microcontrolador maestro el movimiento que ha realizado una articulación, enviándole la dirección en la que se encuentra y el número de pulsos que ha dado el motor.

- **Comandos Limp y Nolimp**

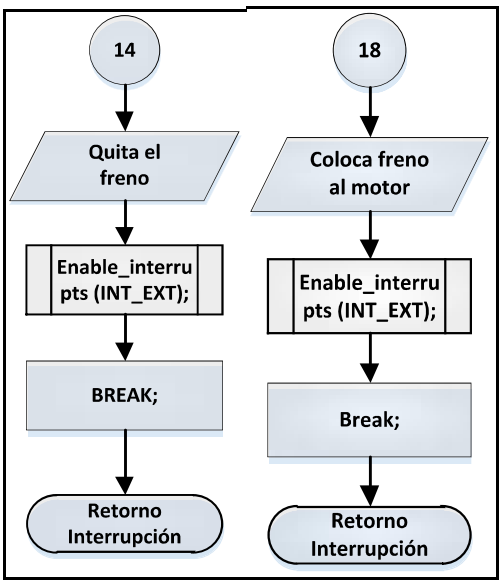


Figura. 89 Diagrama de comandos Limp y Nolimp.

Estos comandos activan o desactivan los bits que controlan el freno del motor, además activan la interrupción externa para conocer el sentido en el cual el motor será movido. Una vez colocado el freno al motor esta interrupción vuelve a desactivarse.

- **Comandos Ready y Home**

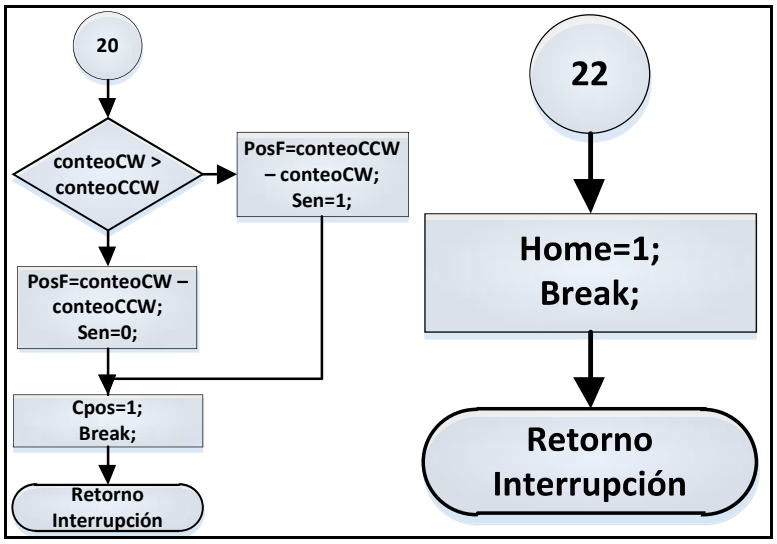


Figura. 90 Diagramas de comandos Ready y Home.

El comando ready, regresa a la posición home, es decir mueve al motor para que su conteo de pulsos vuelva a cero y se ubique a la posición inicial. El comando home hace que el motor realiza un movimiento de 5 grados y que regrese a su posición inicial, este comando solo se permite realizar una vez para ubicar al manipular robótico en las muescas.

- **Comandos Joint, Motor, Pitch, Roll**

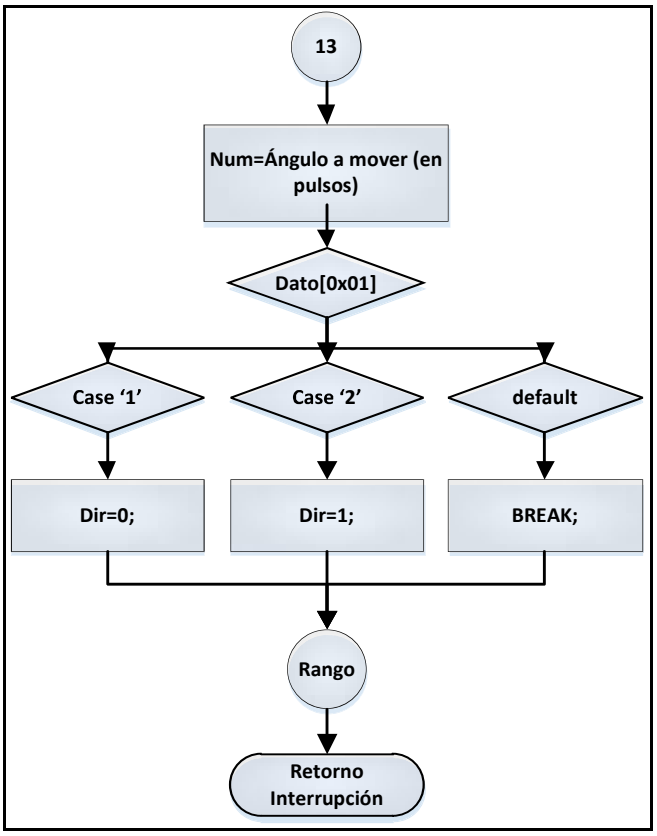


Figura. 91 Diagrama de comando Joint, Motor, Pitch, Roll.

Estos cuatro comandos operan similarmente recibiendo sentido y posición a mover, Joint, Pitch y Roll reciben la posición en ángulos la cual luego es transformada en número de pulsos.

Estos comando realizan un algoritmo para conocer si el movimiento a realizar se encuentra dentro del rango permitido esto lo realiza caso contrario envía una señal al microcontrolador maestro que el movimiento a realizar esta fuera de rango.

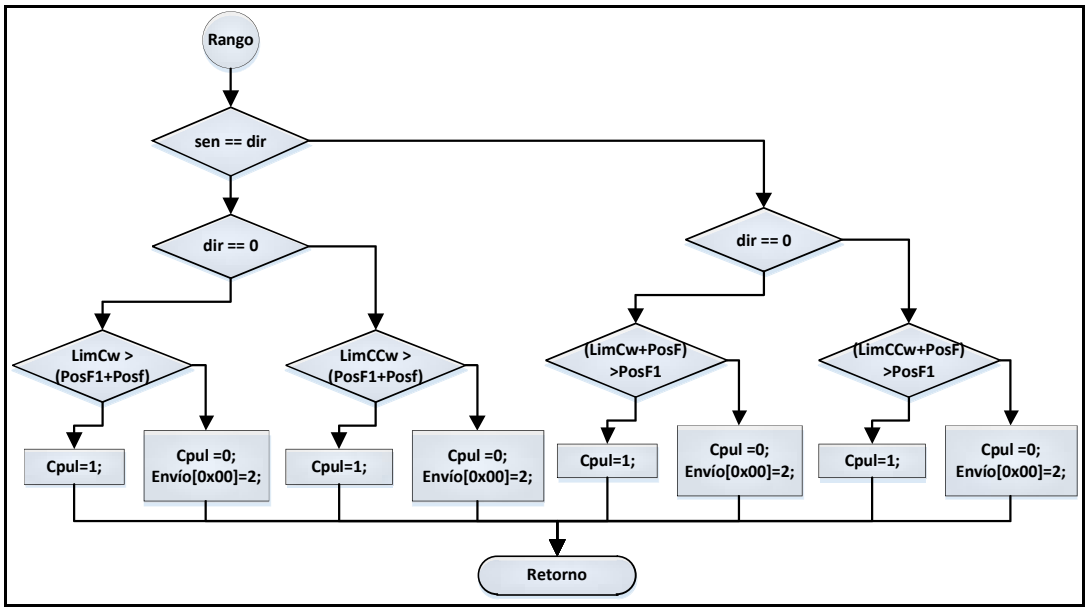


Figura. 92 Diagrama para conocer rango de un motor.

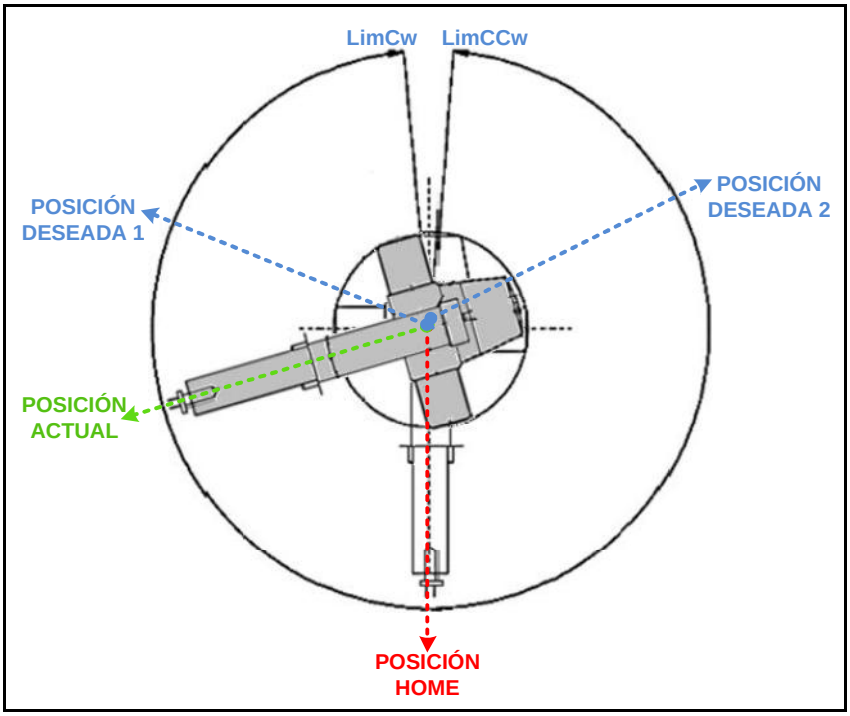


Figura. 93 Representación de límites de movimiento.

En la figura. 93, se conoce la POSICIÓN ACTUAL (PosF) y su sentido (Sen). En (PosF1) se almacena la POSICIÓN DESEADA y en (dir) el sentido al que se desea desplazar.

En el primer caso: La POSICIÓN DESEADA 1 se encuentra en el mismo cuadrante que POSICIÓN ACTUAL, por lo tanto, (sen) es igual a (dir). Ahora si el movimiento es hacia CW comprobamos que el rango CW (LimCW) tiene que ser mayor a la suma entre **PosF +PosF1,** si el mismo cumple el movimiento es permitido ya que se encuentra en el rango, caso contrario enviará una señal al pic "maestro".

En el segundo caso: La POSICIÓN DESEADA 2 se encuentra en el mismo cuadrante opuesto al de POSICIÓN ACTUAL, por lo tanto, (sen) es diferente a (dir). Ahora si el movimiento es hacia CCW comprobamos que la suma entre rango CWW **LimCWW + PosF** tiene que ser mayor **PosF1,** si el mismo cumple el movimiento es permitido ya que se encuentra en el rango, caso contrario enviará una señal al pic "maestro".

- **Comando Move**

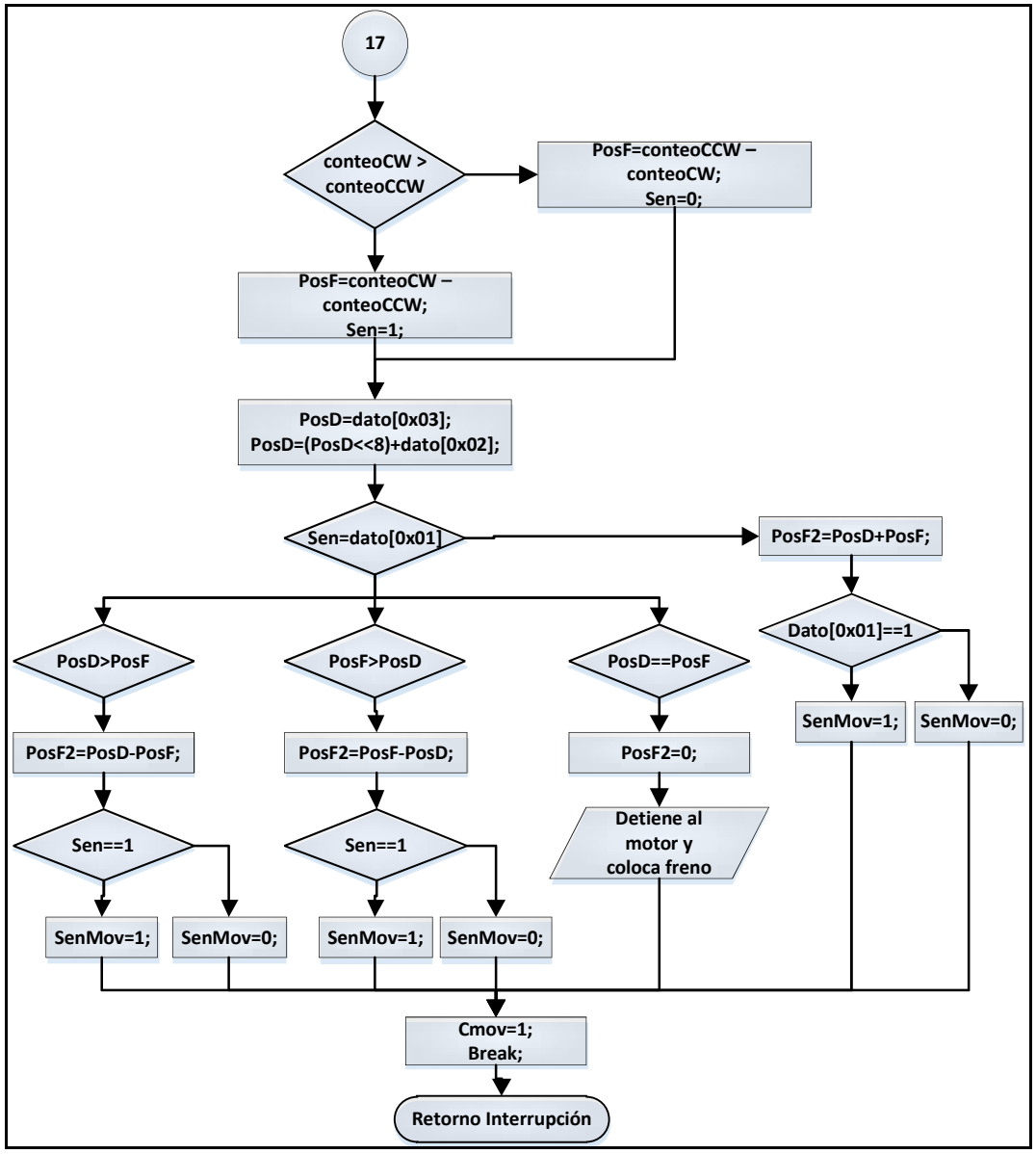


Figura. 94 Diagrama de comando Move.

Para mejor entendimiento del diagrama de flujo del Comando Move, a continuación describe el funcionamiento del algoritmo de posicionamiento del motor de un esclavo, para lo cual se analizan los siguientes casos:

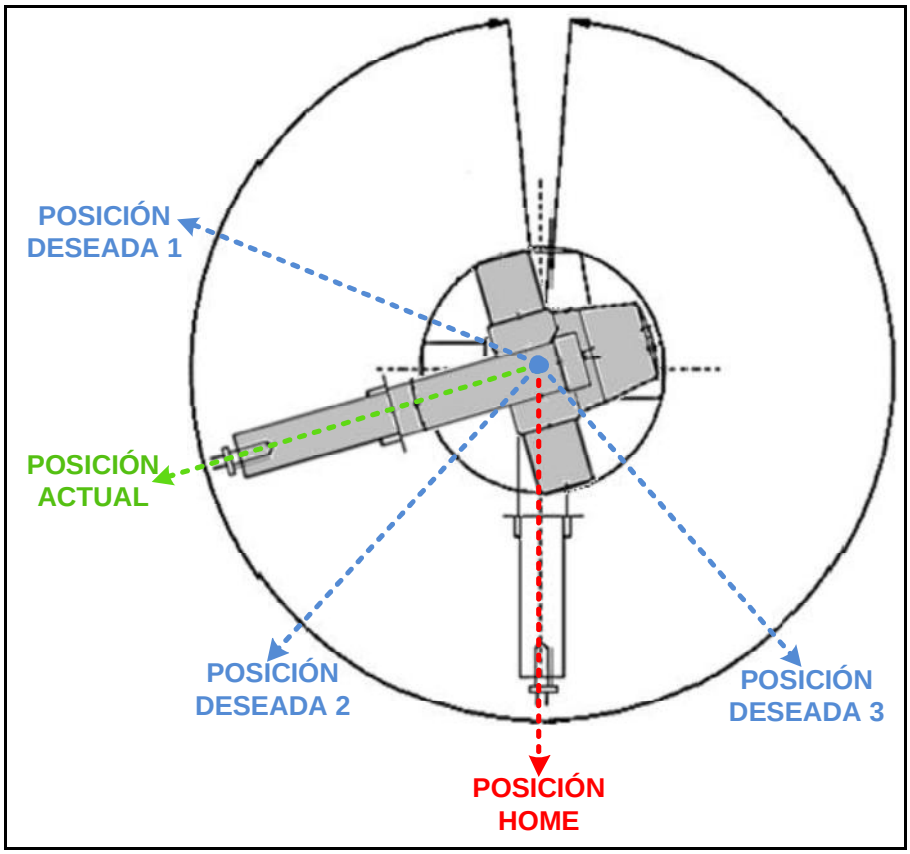


Figura. 95 Representación de casos del comando Move.

En la figura. 95, se conoce cuantos pulsos se ha movido desde POSICIÓN HOME hasta POSICIÓN ACTUAL (PosF) y su sentido (Sen). En (PosD) se almacena la POSICIÓN DESEADA y en (dato [0x01]) el sentido al que se desea desplazar.

En el primer caso: La POSICIÓN DESEADA 1 se encuentra en el mismo cuadrante que POSICIÓN ACTUAL, por lo tanto, el sentido es el mismo. El desplazamiento de la POSICIÓN DESEADA es mayor a POSICIÓN ACTUAL, entonces el número de pulsos a mover (PosF2) será el resultado de la resta de: **PosD-PosF** y el sentido se mantiene.

En el segundo caso: La POSICIÓN DESEADA 2 se encuentra en el mismo cuadrante que POSICIÓN ACTUAL, por lo tanto, el sentido es el mismo. El desplazamiento de la POSICIÓN DESEADA es menor a POSICIÓN ACTUAL, entonces el número de pulsos a mover (PosF2) será el resultado de la resta de: **PosD-PosF** y en sentido será contrario al actual.

En el tercer caso: La POSICIÓN DESEADA 3 se encuentra en cuadrante opuesto al de POSICIÓN ACTUAL, entonces el sentido de desplazamiento cambia, entonces el número de pulsos a mover (PosF2) será el resultado de la suma de **PosD+PosF** y el sentido será el de POSICIÓN DESEADA.

- **Comando Movimiento Teach**

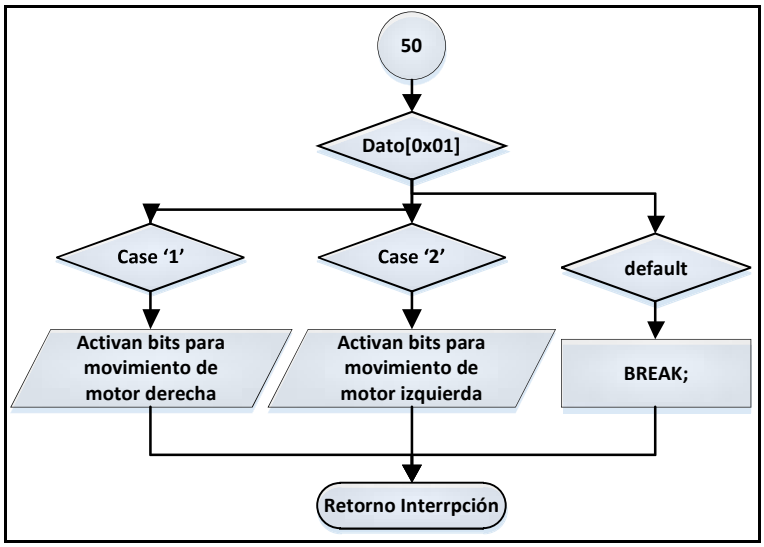


Figura. 96 Diagrama de movimiento de Teach.

El comando del teach activa el movimiento del motor y dependiendo de Dato[0x01] decide a que dirección debe realizar dicho movimiento.

- **Comando de Paro de Emergencia**

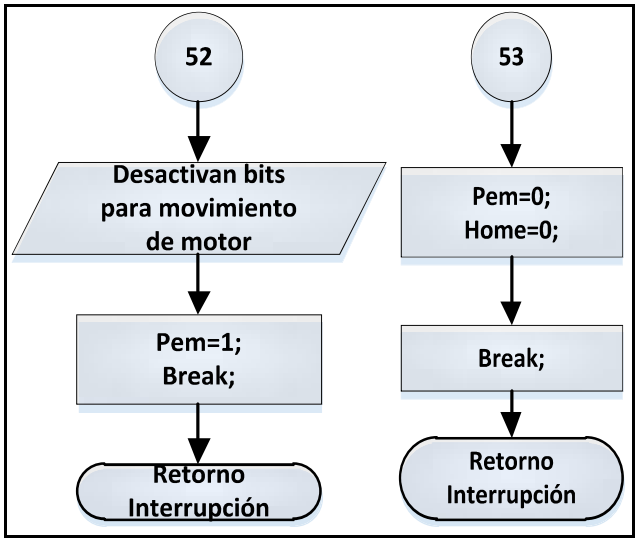


Figura. 97 Diagrama de activación y desactivación de paro de emergencia.

Al recibir la instrucción de paro de emergencia automáticamente se detiene el movimiento del motor, sin importar donde se encuentre. Posteriormente encera las variables por default. Al desactivarse el paro de emergencia permite que el manipulador vuelva a realizar posición home, ya sea desde el teach pendant o desde el modo online.

3.2.6 Diseño de Interfaz Gráfica de Usuario

Antes de empezar con el diseño de la interfaz, es necesario visualizar el siguiente diagrama, que representa todas las tareas que pueden realizar el usuario u operador en todo el sistema desde el punto de vista del mismo.

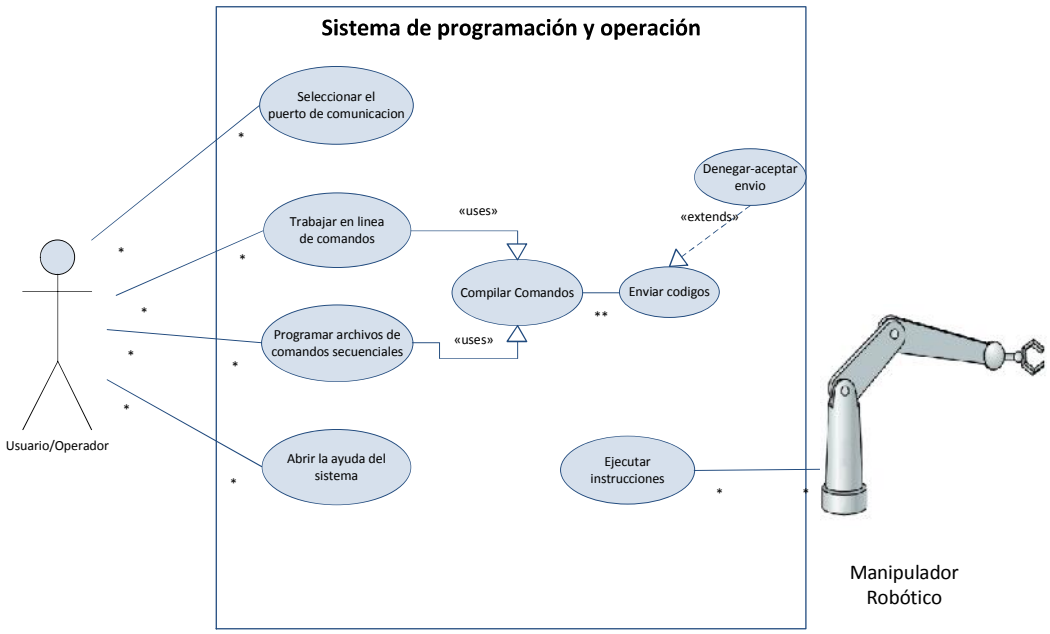


Figura. 98 Diagrama de casos de uso.

Como se observa en la figura.98 los actores involucrados en el sistema son el usuario y el manipulador robótico, las tareas que ellos realizan se encuentra limitadas por el sistema

Para el diseño de la interfaz gráfica de usuario (HMI), se planteó desarrollar un HMI moderno, más llamativo a la vista, que contemple similares características y estructuras que el software propietario RobComm. El diseño de cada ventana a implementar se detallara en el siguiente orden:

- Ventana de Presentación
- Ventana Principal
- Ventana de Comunicación
- Ventana en modo Terminal
- Ventana Acerca (descripción del software)
- Ventana de Ayuda

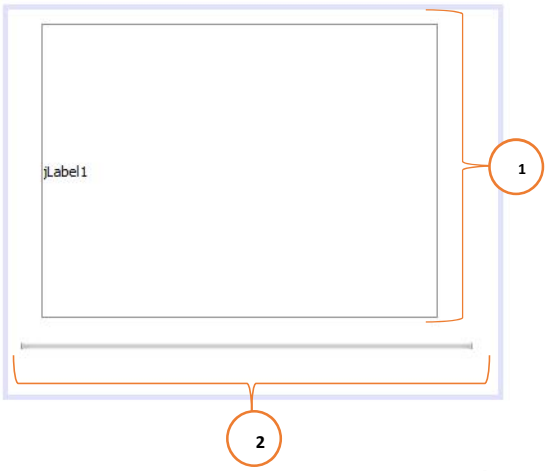


Figura. 99 Ventana de presentación.

1. **Etiqueta (JLabel):** Permite visualizar una imagen texto o animación.
2. **Barra de progreso (JProgressBar):** Representa el progreso de carga de todos los componentes para que funcione correctamente el software.

La ventana de presentación permite visualizar una animación que destaca el nombre y logo del software durante un tiempo corto, esto es importante para que el usuario identifique y se familiarice con la marca.

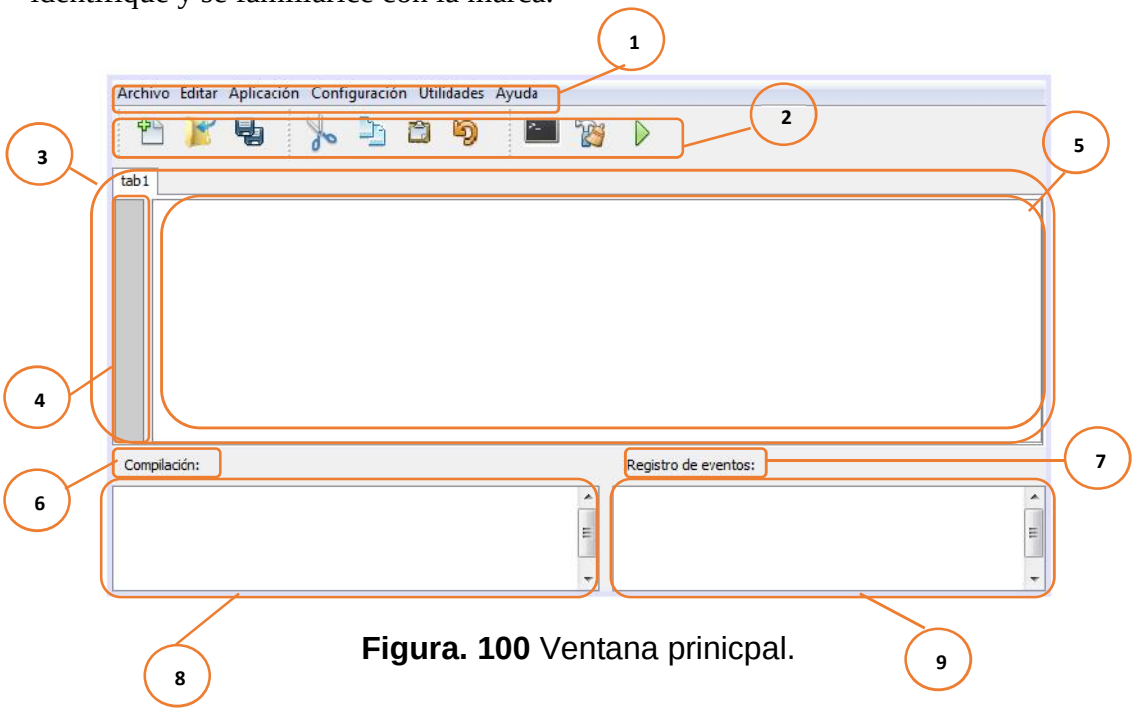


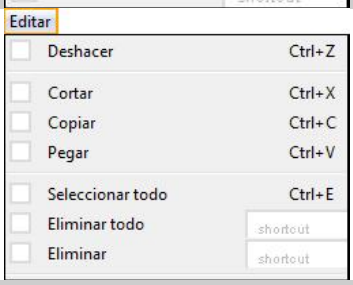
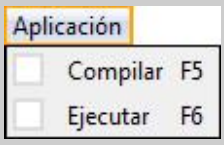
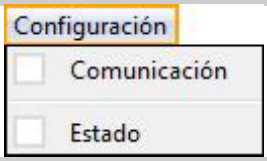
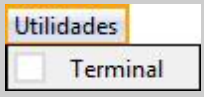
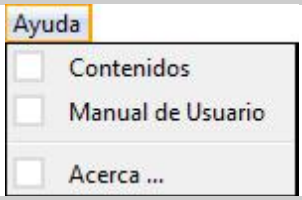
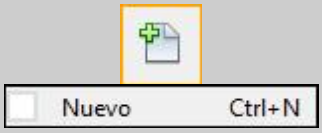
Figura. 100 Ventana prinicpal.

1. **Barra de menús (JMenuBar):** Contiene los diferentes menús para realizar archivos, edición y acceder a diferentes ventana de trabajo y configuración.
2. **Barra de herramientas (JToolbar):** Permite acceder a diferentes ventanas o realizar operaciones sobre el texto de manera rápida.
3. **Pestaña panel (JTabbedPanel):** Es un contenedor tipo pestaña, el cual contiene dos áreas de texto (JTextArea).
4. **Área de texto 1 (JTextArea):** Permite visualizar el número de instrucción.
5. **Área de texto 2 (JTextArea):** Donde se realiza la escritura del código.
6. **Etiqueta (JLabel):** Nombre indicativo Compilar.
7. **Etiqueta (JLabel): N**ombre indicativo Registro de eventos.
8. **Área de texto (JTextArea):** Permite visualizar el resultado de compilación si existe errores o esta correcto todo el código.
9. **Área de texto (JTextArea):** Indica los eventos que se realiza durante la ejecución del código.

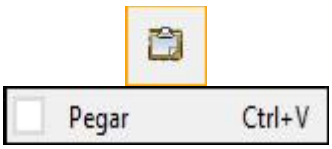
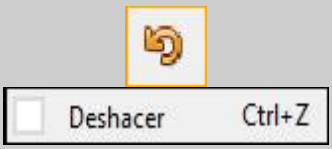
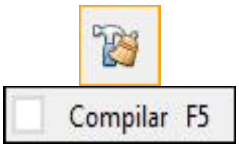
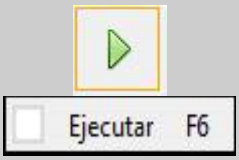
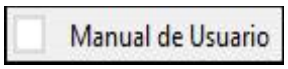
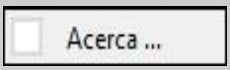
La ventana principal es el punto de partida para el usuario, desde esta ventana se puede trabajar en modo de comandos secuenciales, permitiendo manipular archivos (creados internamente en la interfaz o guardados en un directorio), editarlos y abrirlos.

Aparte desde la misma se puede acceder a otras ventanas que permiten trabajar en otro modo (terminal, línea de comandos), realizar configuraciones sobre el controlador y visualizar la ayuda. A continuación de describe cada uno de los componentes incorporados en la barra de menús y herramientas de la Ventana Principal (JFrame).

Tabla. 15 Componentes de barra de herramientas y menús.

ICONO	NOMBRE –DE COMPONENTE	DESCRIPCIÓN
	M_Archivo (JMenu)	Menú contenido en la barra de menús que permite generar un archivo nuevo, abrir un archivo, guardar y salir.
	M_Editar (JMenu)	Menú Editar permite realizar cambios en el texto del archivo como deshacer, cortar, pegar, seleccionar todo el texto, eliminar todo el texto previamente selecciona y eliminar.
	M_Aplicacion (JMenu)	Menú Aplicación permite desplegar un submenú donde se puede compilar y ejecutar el código secuencial escrito.
	M_Configuración (JMenu)	Menú Configuración permite visualizar un submenú para configurar la comunicación y verificar el estado del mismo.
	M_Utilidades (JMenu)	Menú Utilidades permite desplegar un submenú para trabajar en modo terminal, es decir en línea de comandos.
	M_Ayuda (JMenu)	Menú Ayuda permite visualizar un submenú de contenidos, manual de usuario y una descripción general del software.
	Btn_Nuevo (JButton) MItem_Nuevo (JMenuItem)	Los dos componentes permiten crear archivos nuevos para el desarrollo de código secuencial.
	Btn_Abrir (JButton) MItem_Abrir (JMenuItem)	Los dos componentes permiten abrir archivos desde un directorio de almacenamiento.
	Btn_Guardar (JButton) MItem_Guardar (JManuItem)	Los dos componentes permiten guardar un archivo desarrollado en la interfaz.

Continúa en la siguiente página

ICONO	NOMBRE –DE COMPONENTE	DESCRIPCIÓN
	MItem_Salir (JMenuItem)	Permite salir y finalizar el programa-
	Btn_Cortar (JButton) MItem_Cortar (JMenuItem)	Permiten cortar el texto previamente seleccionado.
	Btn_Copiar (JButton) MItem_Copiar (JMenuItem)	Permiten copiar el texto seleccionado.
	Btn_Pegar (JButton) MItem_Pegar (JMenuItem)	Permiten pegar el texto que se seleccionó previamente al copiar o cortar.
	Btn_Atrás (JButton) MItem_Atrás (JMenuItem)	Permiten retornar a un estado anterior cada vez que es presionado.
	Btn_Compilar (JButton) MItem_Compilar (JMenuItem)	Permiten compilar el texto desarrollado para posteriormente codificarlo.
	Btn_Ejecutar (JButton) MItem_Ejecutar (JMenuItem)	Permiten Ejecutar el código desarrollado secuencialmente.
	MItem_Comunicación (JMenuItem)	Permite desplegar una ventana (pop-up).
	MItem_Contenidos (JManuItem)	Permite visualizar la ayuda del sistema para editar y escribir programas.
	MItem_ManUsuario (JMenuItem)	Permite que el usuario conozca el funcionamiento del software.
	MItem_Acerca (JMenuItem)	Desplegara una ventana donde se muestra información de la versión, el nombre del software.

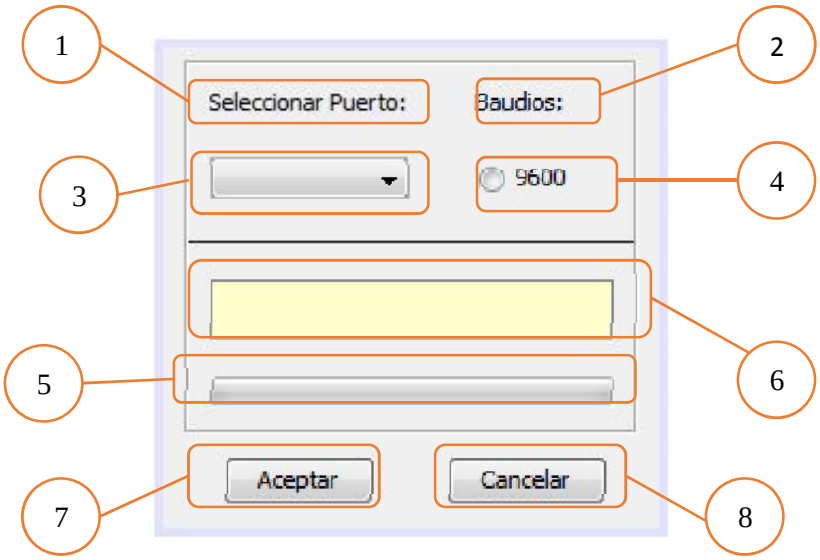


Figura. 101 Ventana comunicación.

1. **Etiqueta:** Representa la acción que debe realizar el usuario sobre la caja combinada (Combo Box) componente 3.
2. **Etiqueta:** Representa la unidad para la transmisión de datos
3. **Caja combinada (Combo box):** Al seleccionar la flecha, se desplegara los puertos que se encuentran conectados al ordenador (COM1, COM2….)
4. **Botón de opción (Radio Button):** Se debe seleccionar para indicar la velocidad de transmisión
5. **Barra de progreso:** Permite visualizar el progreso de la configuración de comunicación
6. **Caja de texto:** Permite visualizar mensajes de comunicación es exitosa o fallida
7. **Botón:** Al presionar dicho botón empieza a realizarse todos los pasos para establecer la comunicación
8. **Botón:** Al presionar este botón la ventana se cerrara sin efectuar alguna operación de configuración

La ventana de comunicación se podrá visualizar al momento de seleccionar de la barra de menú “configuración” el menú ítem “Comunicación”, desde la misma el usuario podrá seleccionar el puerto y velocidad de transmisión, además se añade una barra de progreso que indica el avance de la configuración.

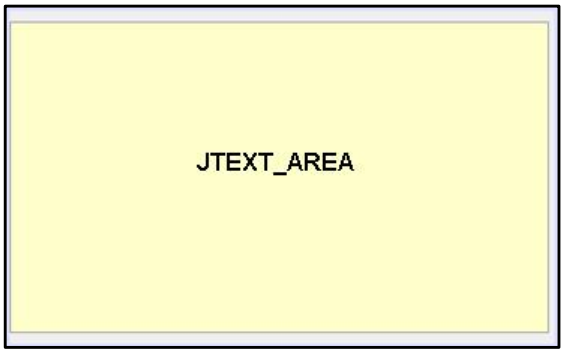


Figura. 102 Ventana terminal.

La ventana terminal se presenta cuando se selecciona el botón  o desde la barra de menú la opción utilidades el ítem terminal, se recuerda que el usuario trabaja en modo terminal (Línea de comandos), es decir se ejecuta un comando a la vez. La ventana cuenta con un solo componente que es un área de texto (JTextArea)

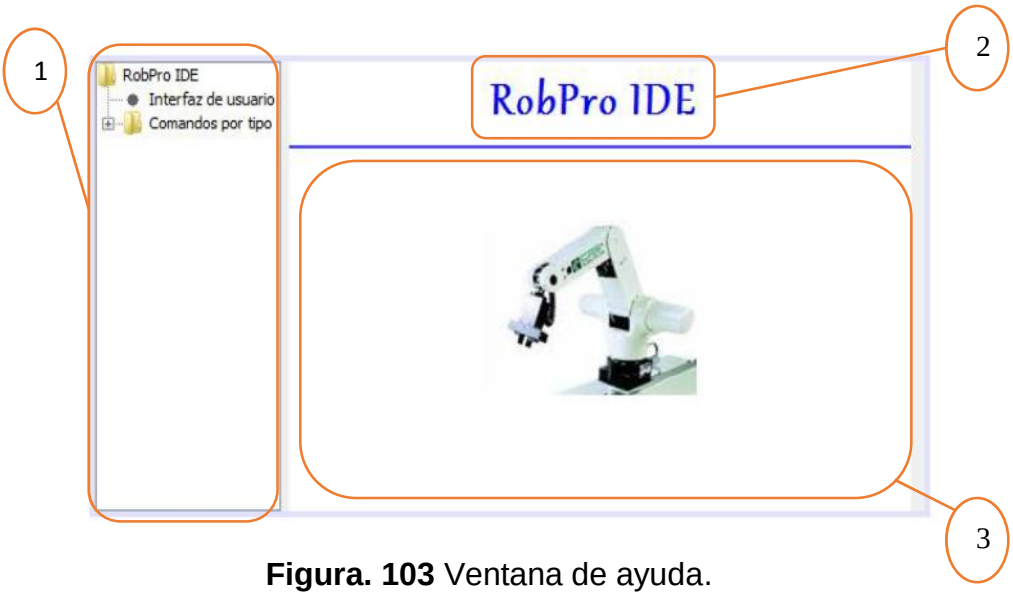


Figura. 103 Ventana de ayuda.

1. **Árbol de carpetas y archivos:** Permite al usuario acceder de manera rápida a la información que se ha seleccionado.

2. **Etiqueta:** Etiqueta (JLabel), permite cambiar el título del comando que se ha selecciona del árbol del archivos.

3. **Panel:** Panel (JPanel) que sirve de contenedor, que permite presentar información e imágenes que el usuario ha seleccionado.

La figura. 103, representa la ventana de ayuda que permite al usuario revisar la sintaxis de los comandos, parámetros de envío de los mismos, y conocer como es la estructura de sentencias para la creación de programas para el manipulador robótico.

3.2.7 Implementación de Software

Una vez realizado el análisis y diseño del funcionamiento del sistema; se comienza a desarrollar e implementar los programas internos de la interfaz gráfica, lo cual permitirá interactuar al usuario con el manipulador robótico. El algoritmo para realizar el compilador-interprete del software es siguiente.

```
Seleccionar las 3 primeras letras del comando
SI serror==1 ENTONCES
SI las 3 primeros caracteres == HOM ENTONCES
   SI home==1 ENTONCES
         envío serial “O\r”
         Home-1
   SI NO
         envío serial “K\r”
   FINSI
```

```
            FINSI
            FinSi
            SI serror==1 Y home==1 ENTONCES
               SEGÚN Las 3 primeros caracteres HACER
                        CASO Comando directo
                                envío código
                        CASO Comando con parámetros
                                ang=vacío
                                SEGÚN art HACER
                                        CASO 1
                                                Determinar sentido
                                                Ang=seleccionar dato hasta ";"
                                                Envío código
                                        CASO 2
                                                Similar Caso 1
                                        CASO 3
                                                Similar Caso 1
                                        CASO 4
                                                Similar Caso 1
                                        CASO 5
                                                Similar Caso 1
                                        FINSEGUN
                                FINSEGUN
                        CASO Comando con retorno y parámetros
                                Crear un espacio de memoria para la recepción
                                Envío código
                        CASO Comando interno
                                Seleccionar la operación
                                Evaluar si existe el elemento
                                Realizar acción
                        FIN SEGÚN
            SI NO
               SI serror==0 Y paroE==falso ENTONCES
                        Presentar "Error de compilación"
               FINSI
               SI home==0 Y serror ==1 ENTONCES
                        Presentar "Indique posición HOME"
               FINSI
FINSI
```

En el mediante el siguiente diagrama de bloques se esquematiza el funcionamiento interno del software.

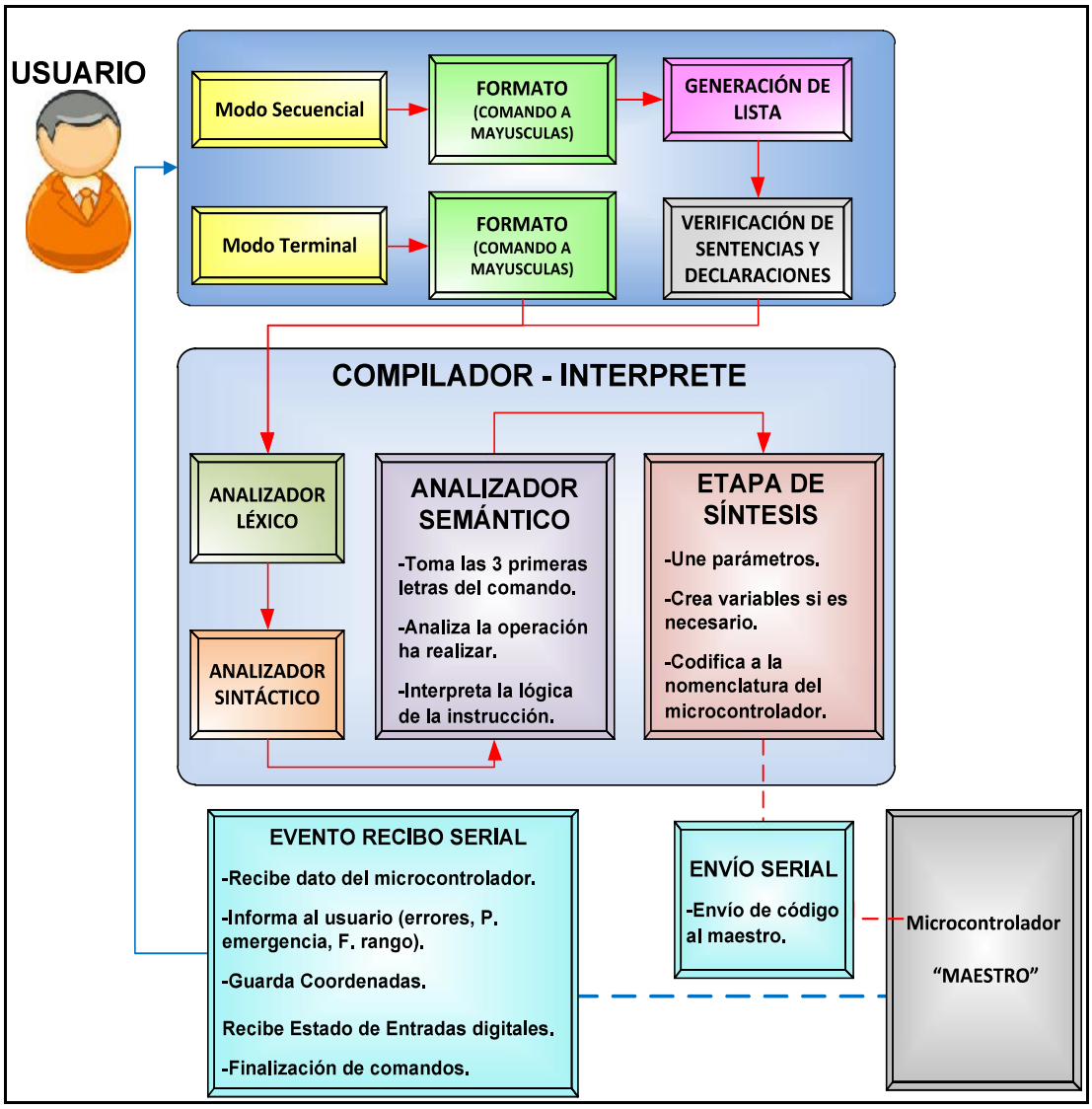


Figura. 104 Funcionamiento interno del software.

Cuando el usuario interactúa sobre la interfaz y realiza acciones como: abrir, crear editar, borrar, los comandos ingresados pasan a una etapa de formato donde se los transforma a mayúsculas, con el objetivo de normalizar la información y evitar errores.

Si se encuentra en el modo de trabajo terminal la información pasa directo a una etapa de tratamiento del mismo; caso contrario, si se encuentra en modo secuencial,

antes de ir a la etapa de tratamiento, los comandos son ingresados a una lista, donde se asigna un índice a cada uno de ellos.

Una vez en la lista pasan al reconocimiento de sentencias de control de programa como: IF, WHILE, VAR, INT, PAUSE; y luego cada comando pasa a la etapa de tratamiento.

Siguiendo el algoritmo de diseño de un traductor, se ha decidido no diseñar un compilador compacto; sino, la unión de un compilar-interprete, el mismo que analizará la estructura de cada comando, verificando que haya concordancia en el mismo y generar instrucciones codificadas para que el microcontrolador "maestro" realice acciones sobre el mismo.

Para lo cual el compilador-interprete tiene la siguiente estructura:

- Analizador léxico.
- Analizador sintáctico.
- Analizador semántico.
- Etapa de síntesis.

3.2.7.1 Implementación del Analizador Léxico

Jlex genera el analizador léxico de nuestro compilador, para lo cual se seguirá la siguiente estructura:

- **Código del Usuario:** Es esta sección se coloca el código java que deseamos usar en la clase que será generada, deben ir las importas de librerías. Posteriormente se coloca **(%%).**

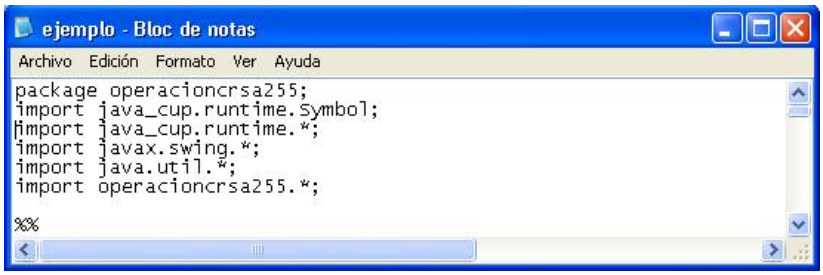


Figura. 105 Código de usuario de JLex.

- **Directivas Jlex:** En esta sección irán las directivas, o especificaciones para que opere JLEX, para obtener la salida deseada.

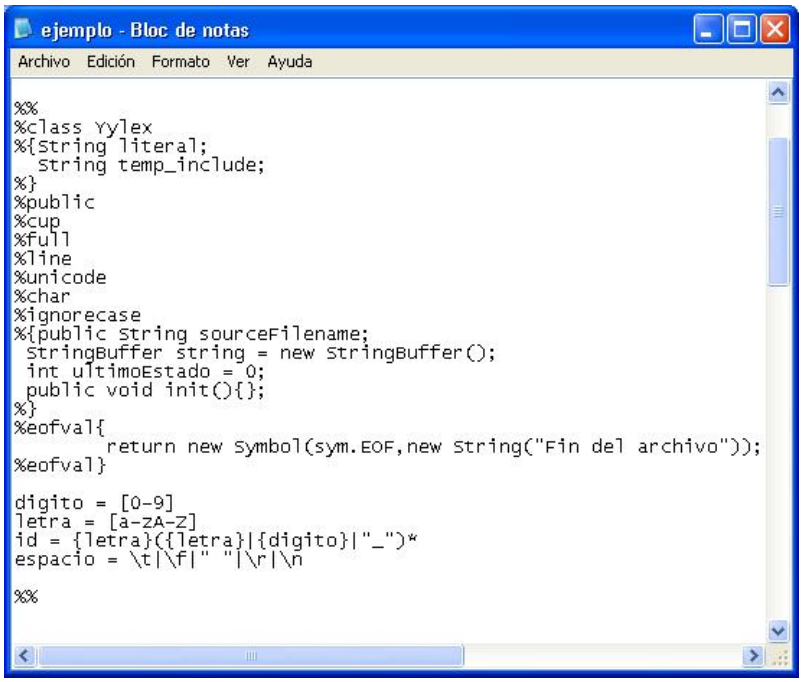


Figura. 106 Directivas de JLex.

- **Reglas para las Expresiones Regulares:** En esta sección del archivo Jlex, se definen las reglas para obtener los tokens de una cadena leída.

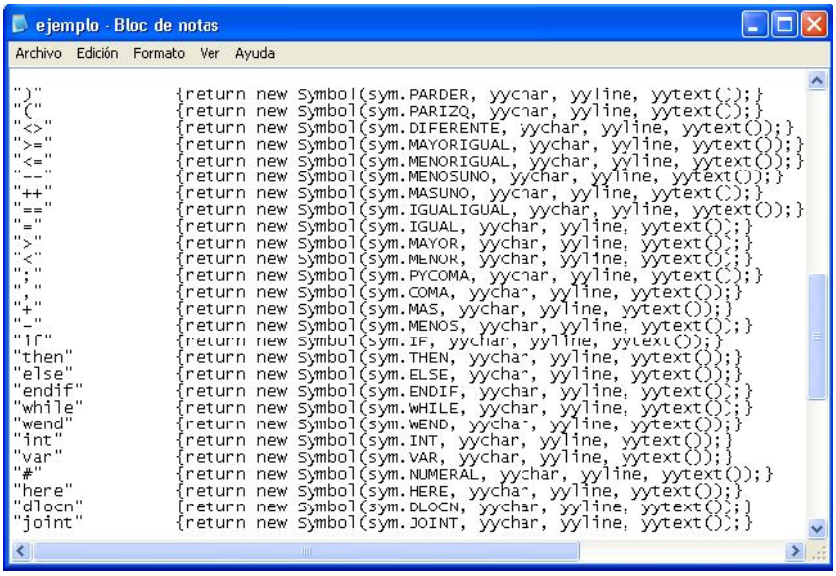


Figura. 107 Reglas para las Expresiones del JLex.

La declaración de las palabras reservadas son las siguientes:

Tabla. 16 Palabras reservadas del léxico.

PALABRA RESERVADA	DECLARACIÓN DE RETORNO DEL LÉXICO
")"	{return new Symbol(sym.PARDER, yychar, yyline, yytext());}
"("	{return new Symbol(sym.PARIZQ, yychar, yyline, yytext());}
"<>"	{return new Symbol(sym.DIFERENTE, yychar, yyline, yytext());}
">="	{return new Symbol(sym.MAYORIGUAL, yychar, yyline, yytext());}
"<="	{return new Symbol(sym.MENORIGUAL, yychar, yyline, yytext());}
"--"	{return new Symbol(sym.MENOSUNO, yychar, yyline, yytext());}
"++"	{return new Symbol(sym.MASUNO, yychar, yyline, yytext());}
"=="	{return new Symbol(sym.IGUALIGUAL, yychar, yyline, yytext());}
"="	{return new Symbol(sym.IGUAL, yychar, yyline, yytext());}
">"	{return new Symbol(sym.MAYOR, yychar, yyline, yytext());}
"<"	{return new Symbol(sym.MENOR, yychar, yyline, yytext());}
";"	{return new Symbol(sym.PYCOMA, yychar, yyline, yytext());}
PALABRA RESERVADA	DECLARACIÓN DE RETORNO DEL LÉXICO
","	{return new Symbol(sym.COMA, yychar, yyline, yytext());}
"+"	{return new Symbol(sym.MAS, yychar, yyline, yytext());}

Continúa en la siguiente página

PALABRA RESERVADA	DECLARACIÓN DE RETORNO DEL LÉXICO
"-"	{return new Symbol(sym.MENOS, yychar, yyline, yytext());}
"IF"	{return new Symbol(sym.IF, yychar, yyline, yytext());}
"THEN"	{return new Symbol(sym.THEN, yychar, yyline, yytext());}
"ELSE"	{return new Symbol(sym.ELSE, yychar, yyline, yytext());}
"ENDIF"	{return new Symbol(sym.ENDIF, yychar, yyline, yytext());}
"WHILE"	{return new Symbol(sym.WHILE, yychar, yyline, yytext());}
"WEND"	{return new Symbol(sym.WEND, yychar, yyline, yytext());}
"INT"	{return new Symbol(sym.INT, yychar, yyline, yytext());}
"VAR"	{return new Symbol(sym.VAR, yychar, yyline, yytext());}
"#"	{return new Symbol(sym.NUMERAL, yychar, yyline, yytext());}
"HERE"	{return new Symbol(sym.HERE, yychar, yyline, yytext());}
"DLOCN"	{return new Symbol(sym.DLOCN, yychar, yyline, yytext());}
"JOINT"	{return new Symbol(sym.JOINT, yychar, yyline, yytext());}
"MOTOR"	{return new Symbol(sym.MOTOR, yychar, yyline, yytext());}
"MOVE"	{return new Symbol(sym.MOVE, yychar, yyline, yytext());}
"CPATH"	{return new Symbol(sym.CPATH, yychar, yyline, yytext());}
"LIMP"	{return new Symbol(sym.LIMP, yychar, yyline, yytext());}
"LOCK"	{return new Symbol(sym.LOCK, yychar, yyline, yytext());}
"NOLIMP"	{return new Symbol(sym.NOLIMP, yychar, yyline, yytext());}
"UNLOCK"	{return new Symbol(sym.UNLOCK, yychar, yyline, yytext());}
"OPEN"	{return new Symbol(sym.OPEN, yychar, yyline, yytext());}
"CLOSE"	{return new Symbol(sym.CLOSE, yychar, yyline, yytext());}
"READY"	{return new Symbol(sym.READY, yychar, yyline, yytext());}
"INPUT"	{return new Symbol(sym.INPUT, yychar, yyline, yytext());}
"OUTPUT"	{return new Symbol(sym.OUTPUT, yychar, yyline, yytext());}
"PITCH"	{return new Symbol(sym.PITCH, yychar, yyline, yytext());}
"ROLL"	{return new Symbol(sym.ROLL, yychar, yyline, yytext());}
PALABRA RESERVADA	DECLARACIÓN DE RETORNO DEL LÉXICO
"DELAY"	{return new Symbol(sym.DELAY, yychar, yyline, yytext());}
"WAIT"	{return new Symbol(sym.WAIT, yychar, yyline, yytext());}
"RUN"	{return new Symbol(sym.RUN, yychar, yyline, yytext());}

Continúa en la siguiente página

PALABRA RESERVADA	DECLARACIÓN DE RETORNO DEL LÉXICO
"PAUSE"	{return new Symbol(sym.PAUSE, yychar, yyline, yytext());}
"FINISH"	{return new Symbol(sym.FINISH, yychar, yyline, yytext());}
"HOME"	{return new Symbol(sym.HOME, yychar, yyline, yytext());}
{ID}	{return new Symbol(sym.ID, yychar, yyline, yytext());}
{DIGITO}	{return new Symbol(sym.NUMERO, yychar, yyline, new Integer(yytext()));}

En el Anexo. 6, se encuentra el código del analizador léxico completo. Realizando estos pasos se generará un archivo con extensión .lex en este caso se llamará ejemplo.lex.

3.2.7.2 Implementación del Analizador Sintáctico

Java Cup genera el analizador sintáctico de nuestro compilador, A continuación detallaré como se estructura un archivo de entrada para Cup:

- **Imports Java:** Se declaran todas las librerías de Java, que se usarán para el desarrollo del software.

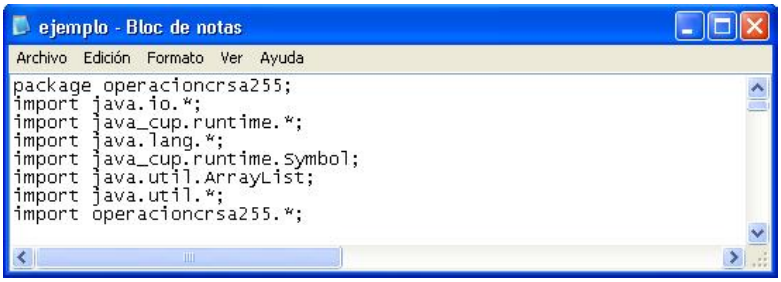


Figura. 108 Importaciones para Java.

- **Código del Usuario para el Parser:** Se declaran métodos y variables que se usarán en la clase resultante. Si se declaran variables o métodos públicos en esta sección estos podrán ser accedidos por otras clases. Ejemplo:

parser code {: /* Código del parser*/:}

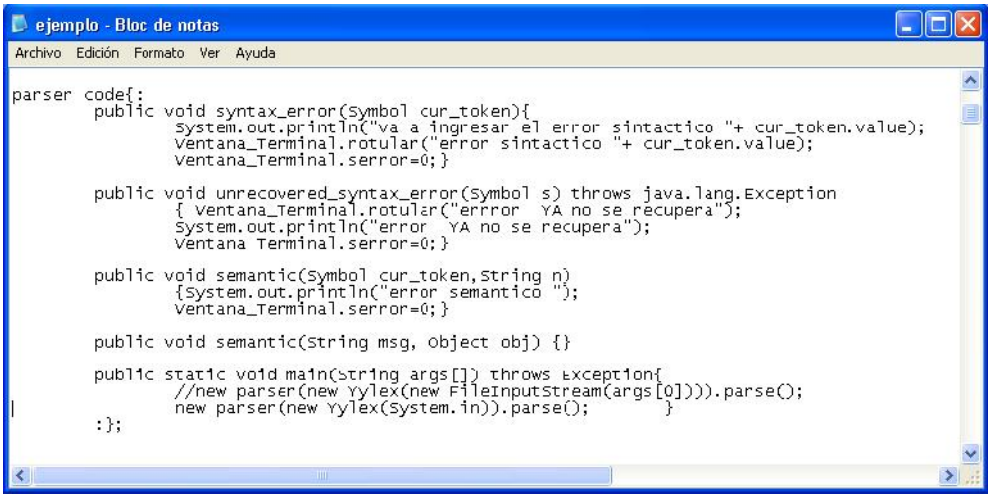


Figura. 109 Código del usuario para el Parser.

- **Código del Usuario para las Acciones de la Gramática:** Genera una salida ya sea para errores semánticos, sintácticos o de traducción a un código equivalente. Se Puede declarar funciones en esta sección y mandarlas a llamar en cada acción gramatical. Ejemplo:

action code {:/*Código para las acciones*/:}

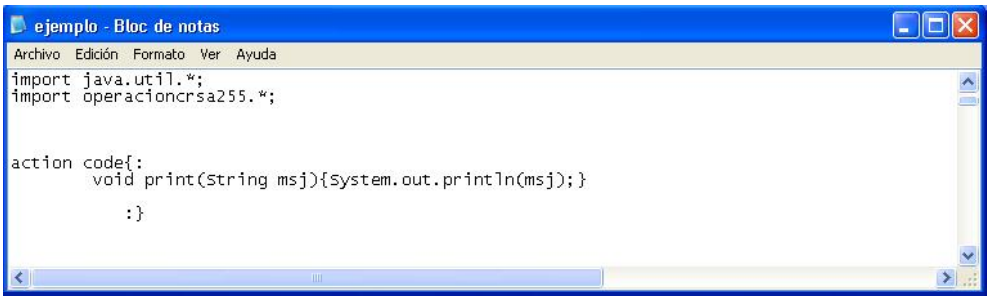


Figura. 110 Código del Usuario para las Acciones de la Gramática.

- **Declaración de Variables para la Gramática:** Se declaran variables Terminales <terminal> y variables no terminales <non terminal>. Las variables terminales serán todos los símbolos terminales de la gramática y las variables no terminales serán todas las variables que representan producciones.

La sintaxis para la declaración es la siguiente:

<Tipo de variable> < Tipo de dato > < Id de la variable >

<Tipo de variable > puede ser Terminal o No terminal.

<Tipo de dato> puede ser cualquier tipo de dato primitivo de Java o uno creado por nosotros.

<Id de la variable > aquí se especifica el id de la variable.

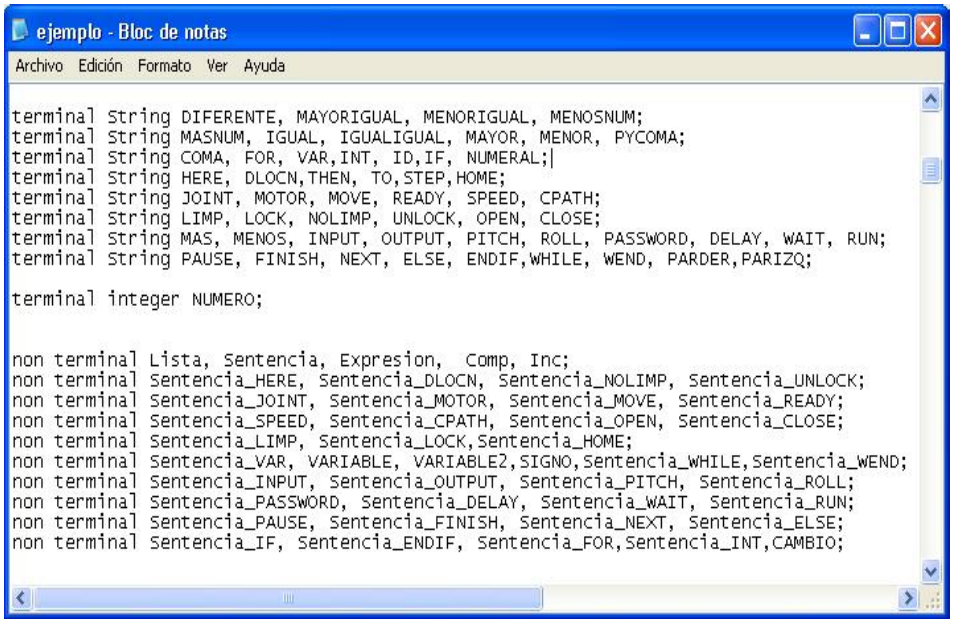


Figura. 111 Declaración de variables para la gramática.

- **Gramática:** En esta sección se escribe la gramática. Con la siguiente sintaxis:

<non terminal > ::= < terminales o No terminales > ;

Como un no-terminal puede tener más de un lado derecho en Cup se utiliza el símbolo "|"

<non terminal > ::= < terminales o No terminales >

|<terminales o No terminales> ;

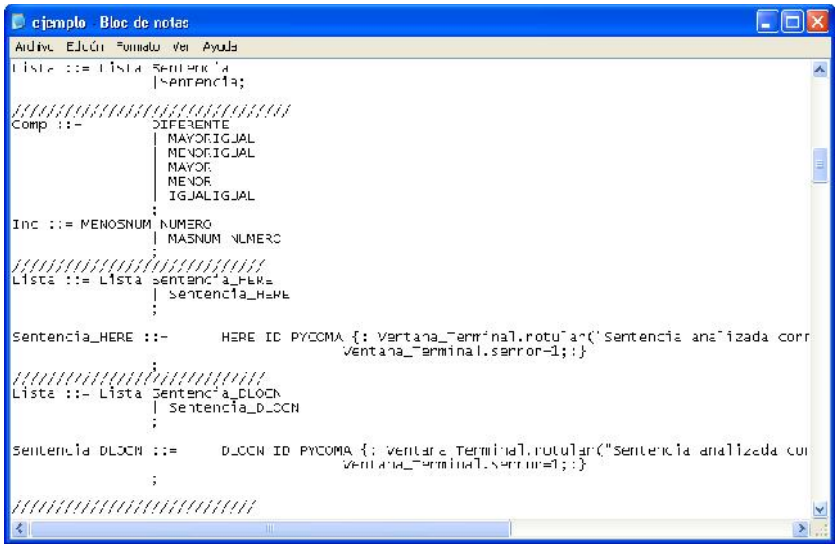


Figura. 112 Estructura de la gramática del parser.

En el Anexo 7, se encuentra el código del analizador sintáctico completo. Realizando estos pasos se generará un archivo con extensión .cup en este caso se llamará ejemplo.cup.

3.2.7.3 Implementación del Analizador Léxico y Sintáctico en Java

Creamos un proyecto en Netbeans con el nombre OperaciónCRSA255, la cual contendrá todas las clases con las que se trabajará. (Ver figura. 113)

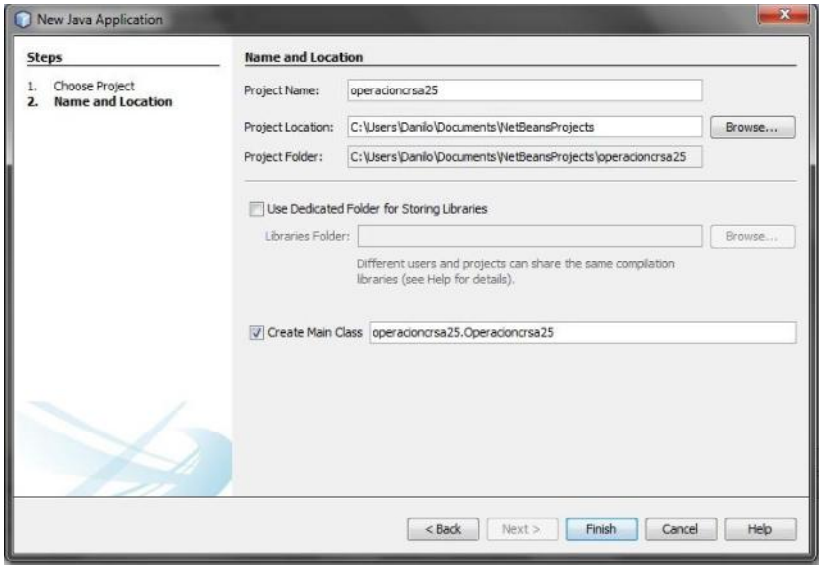


Figura. 113 Creación de un nuevo proyecto en Netbeans.

Ahora los dos archivos: ejemplo.lex y ejemplo.cup, los colocamos en el directorio del proyecto en Netbeans:

D:\Compilador\operacioncrsa255\src\operacioncrsa255. También en este directorio añadimos las carpetas Jlex y java_cup que anteriormente descargamos.

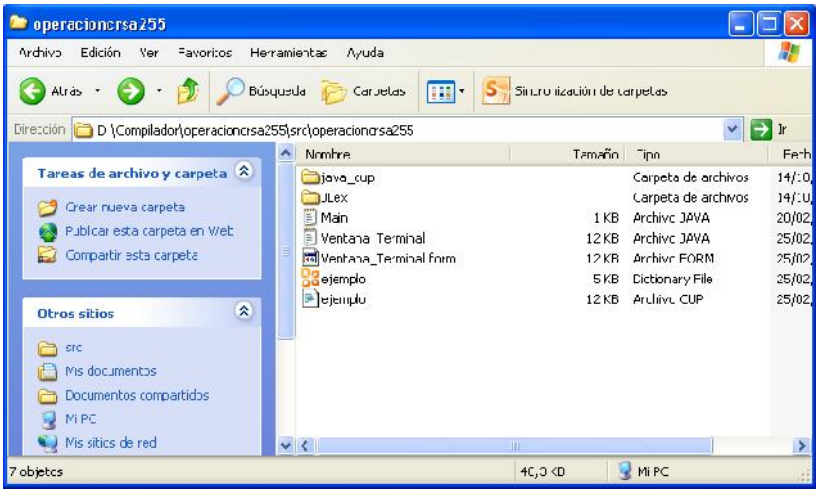


Figura. 114 Directorio con .Lex y .Cup.

Ahora se compila para generar las clases **Yylex.java sym.java y parser.java** que va a usar Netbeans, desde modo consola nos ubicamos en la carpeta del proyecto y se ingresa las siguientes líneas de código:

Java JLex.Main ejemplo.lex

Java java_cup.Main ejemplo.cup

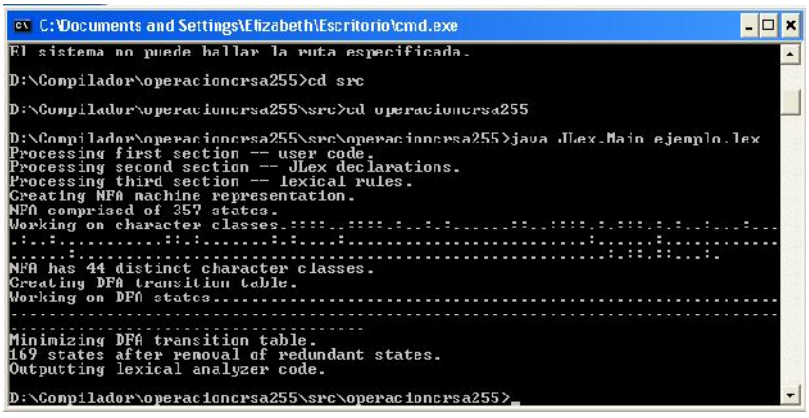


Figura. 115 Compilación por consola de .JLex.

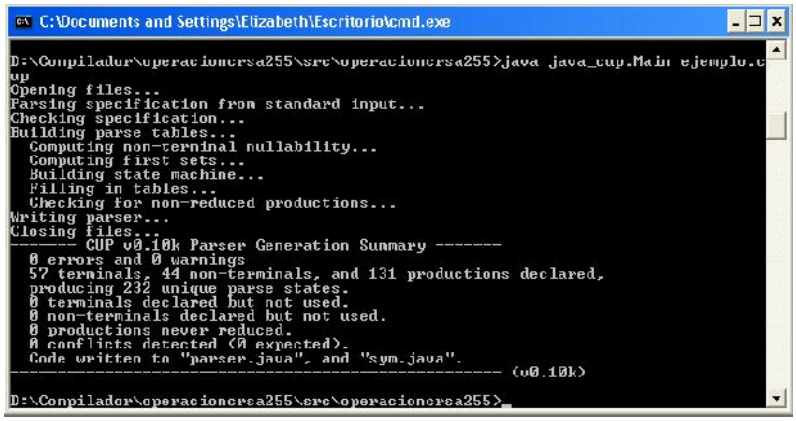


Figura. 116 Compilación por consola de .Cup.

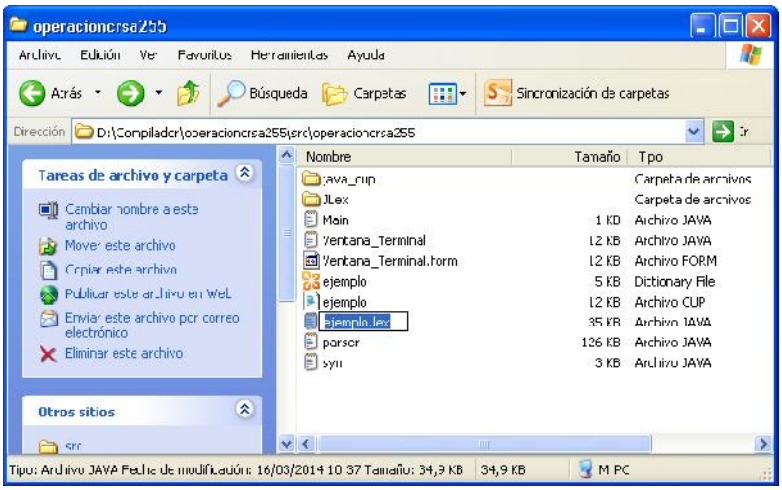


Figura. 117 Generación de archivos ejemplo.lex, parser.cup, sym.java.

Una vez generado los archivos se necesitan al ejemplo.lex.java renombrarlo como Yylex.java

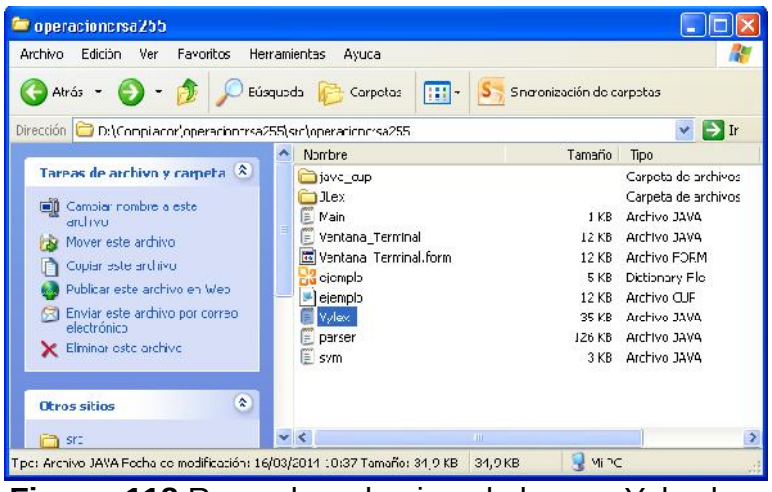


Figura. 118 Renombrando ejemplo.lex por Yylex.lex.

A continuación se requiere una clase llamada Compilador .java.

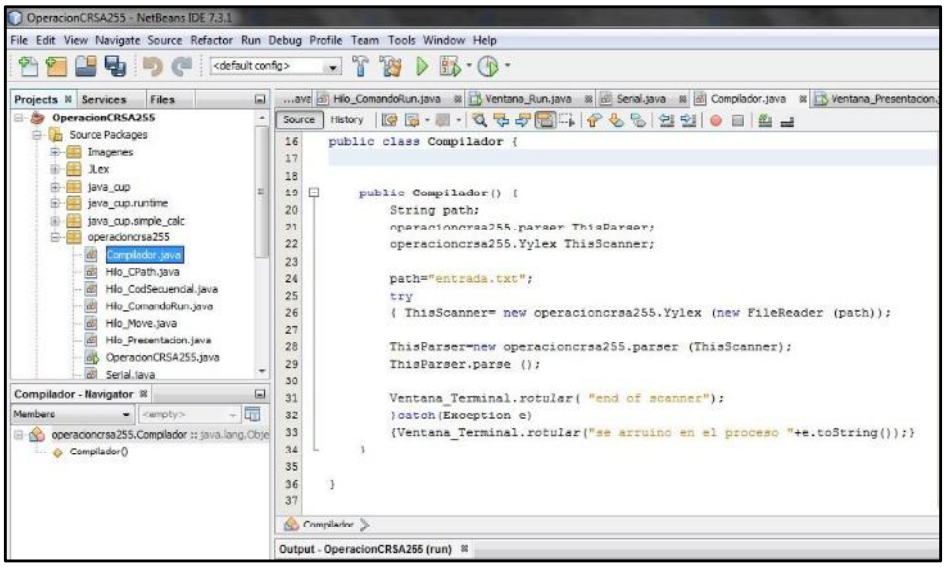


Figura. 119 Clase Compilador.java.

Esta clase permite crear dos objetos: el primero tipo parser (ThisParser) y el segundo tipo Yylex (ThisScanner).

El objeto Yylex se carga un archivo .txt de los comandos, para realizar el análisis léxico, postiormente se realiza el análisis sintáctico, lo cual permite conocer si existe errores en el código escrito en el usuario.

3.2.7.4 Implementación del Analizador Semántico y Etapa de Síntesis

Esta etapa permite conocer la operación que el usuario quiere realizar sobre el manipulador. Además determina si se debe enviar parámetros o reservar espacios de memoria para los datos de retornos por parte del microcontrolador. Y finalmente codifica el comando y lo envía. A continuación se detalla el diagrama de flujo del programa implementado. (Ver figura. 120)

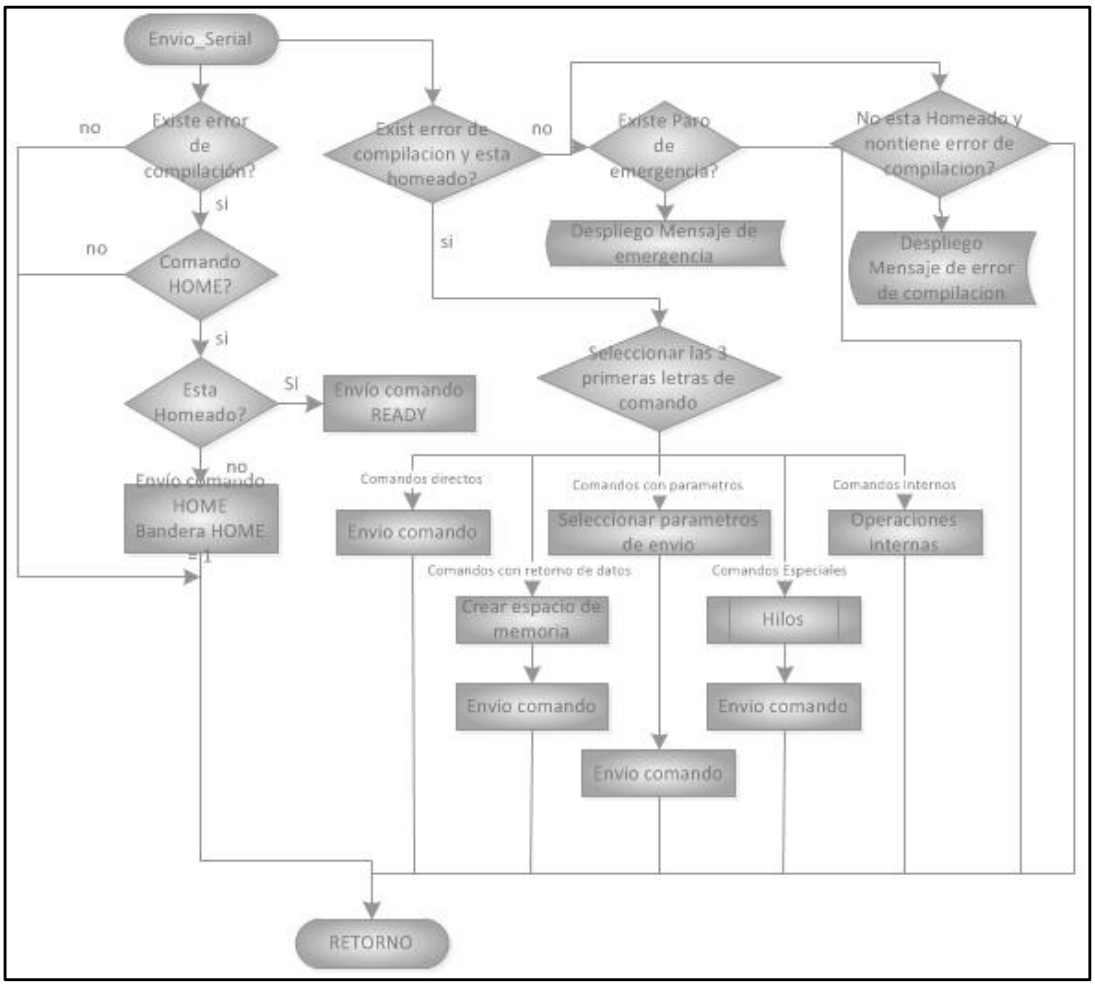


Figura. 120 Análisis Semántico y Etapa de síntesis.

La figura. 120, representa el algoritmo implementado para realizar el análisis semántico y etapa de síntesis, se ha decidido simplificar el diagrama para un mejor entendimiento, por lo tanto se clasifica a los comandos de la siguiente manera.

- **Comandos directos:** Son comandos que no requieren enviar parámetros adicionales (sentido y número de pulsos), por tal razón dichos comandos se enviar directo el código que representa el comando a realizarse entre ellos están **CLOSE, OPEN**. **READY, HOME, DELAY**.

- **Comandos con retorno de datos:** Son comandos que requieren crear variables internas, es decir reservar un espacio de memoria para que cuando los datos sean retornados por parte del controlador se guarden en dichas posiciones de memoria, los comandos involucrados son **HERE, INPUT, WAIT.**

- **Comandos con parámetros:** Son comandos que adicional de enviar la operación a realizarse necesitan enviar información adicional como que numero de articulación mover, sentido y pulsos a moverse entre ellos están **JOINT, MOTOR, LIMP, NOLIMP, LOCK, UNLOCK**. **PITCH**, **ROLL, OUTPUT.**

- **Comandos especiales:** Son comandos que envían parámetros y esperan el retorno de datos para poder continuar con su operación, para satisfacer con esta necesidad se ha incorporado el concepto de multitarea (Realizar varias operaciones al mismo tiempo) , esto se lo realiza creando clases del tipo hilo (conocido Thread en Java) estas permiten separar las tareas para atender a otra de mayor prioridad los comandos son **CPATH, MOVE**

- **Comandos Internos:** Estos comandos permiten tener control sobre el programa desarrollado por el usuario por lo tanto no es necesario codificarlos ya que estos realizan las operaciones solo sobre el software entre ellos están **IF, WHILE, VAR, DLOCN, PAUSE, FINISH, RUN, INT**

3.2.7.5 Estructura de Programación Interna

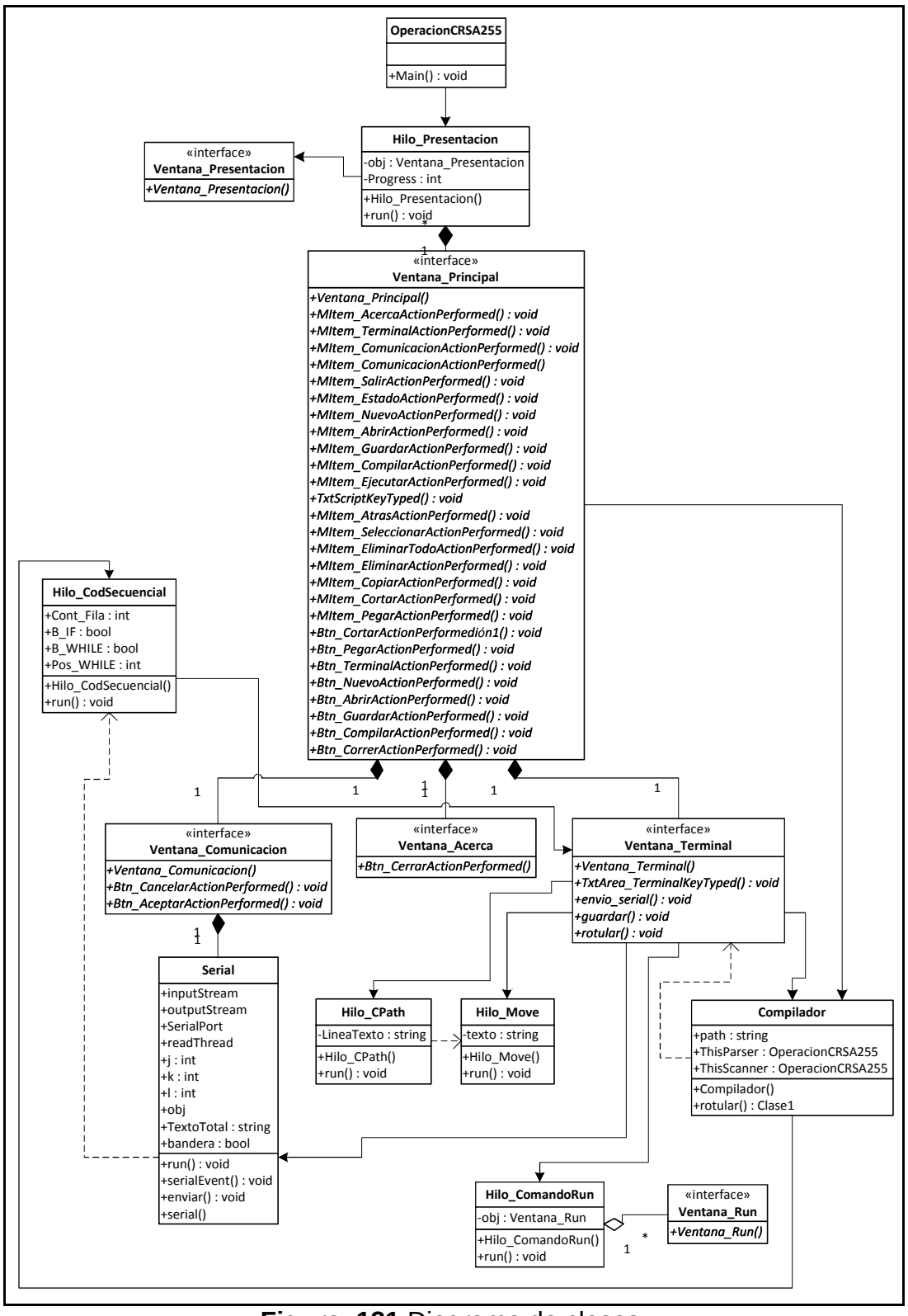


Figura. 121 Diagrama de clases.

Una vez visto el diagrama de clases, donde se muestra la interacción y dependencia de los mismos, a continuación se realiza los diagramas de flujo simplificado y sintetizado de los modos de trabajo que el usuario puede realizar en la interfaz gráfica (Modo terminal, modo secuencial).

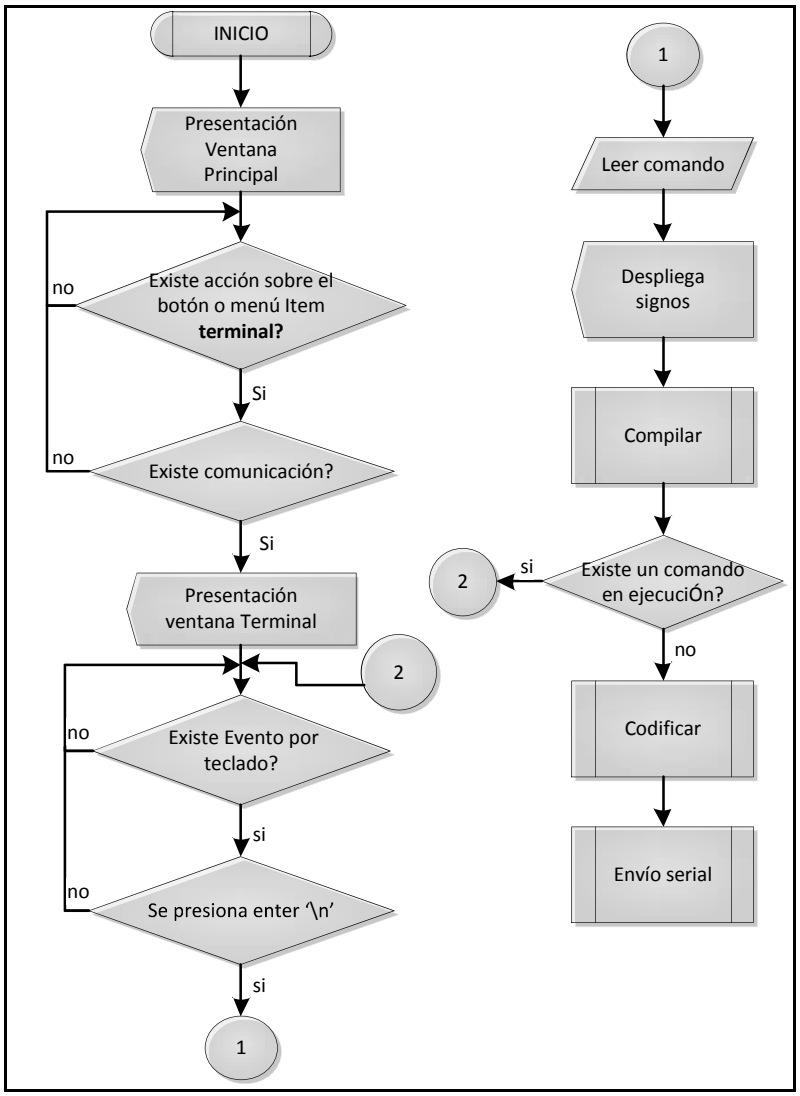


Figura. 122 Diagrama de modo de terminal.

La figura. 122, representa un diagrama simplificado y general de todas las operaciones que se realiza en la programación para que el usuario trabaje en este modo (Línea de comandos).

A continuación se muestra el diagrama simplificado del modo secuencial.

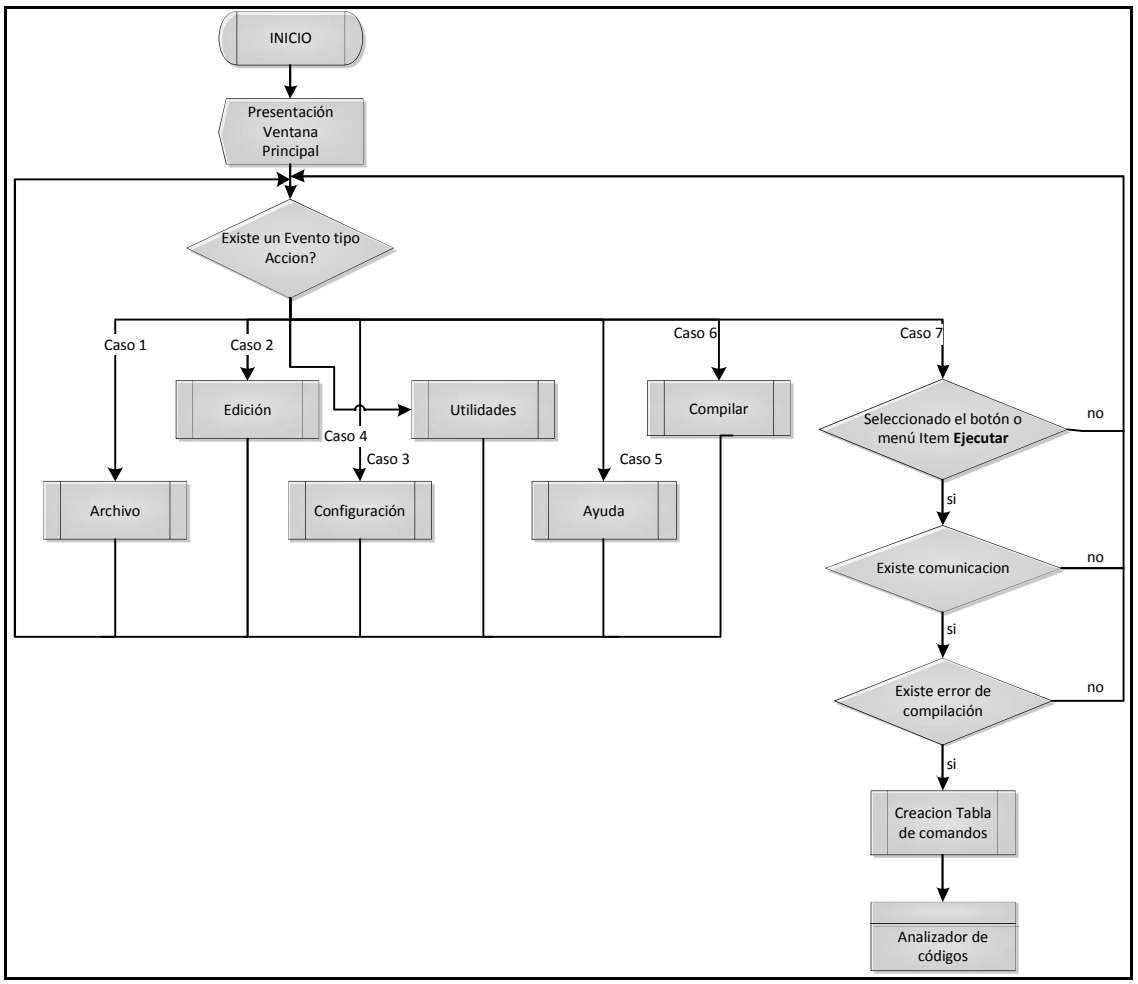


Figura. 123 Diagrama de modo secuencial.

La figura. 123, representa el diagrama de programación simplificado, no se ha realizado el diagrama de flujo completo ya que sería más confuso su entendimiento, por tal motivo los casos 1 al 6 representan operaciones que el usuario realiza sobre el texto como son archivo (Crear nuevo, Abrir, guardar y salir), edición (Cortar, pegar, copiar, rehace, eliminar, seleccionar todo, etc), Configuración (comunicación y estado del puerto), utilidades (Modo terminal) y ayuda.

El recuadro final (Analizador de códigos) representa el llamado a una clase del tipo hilo (Thread) que permite realizar otras operaciones al mismo tiempo (multitarea), a continuación se muestra el diagrama de flujo de programa implementado en el hilo.

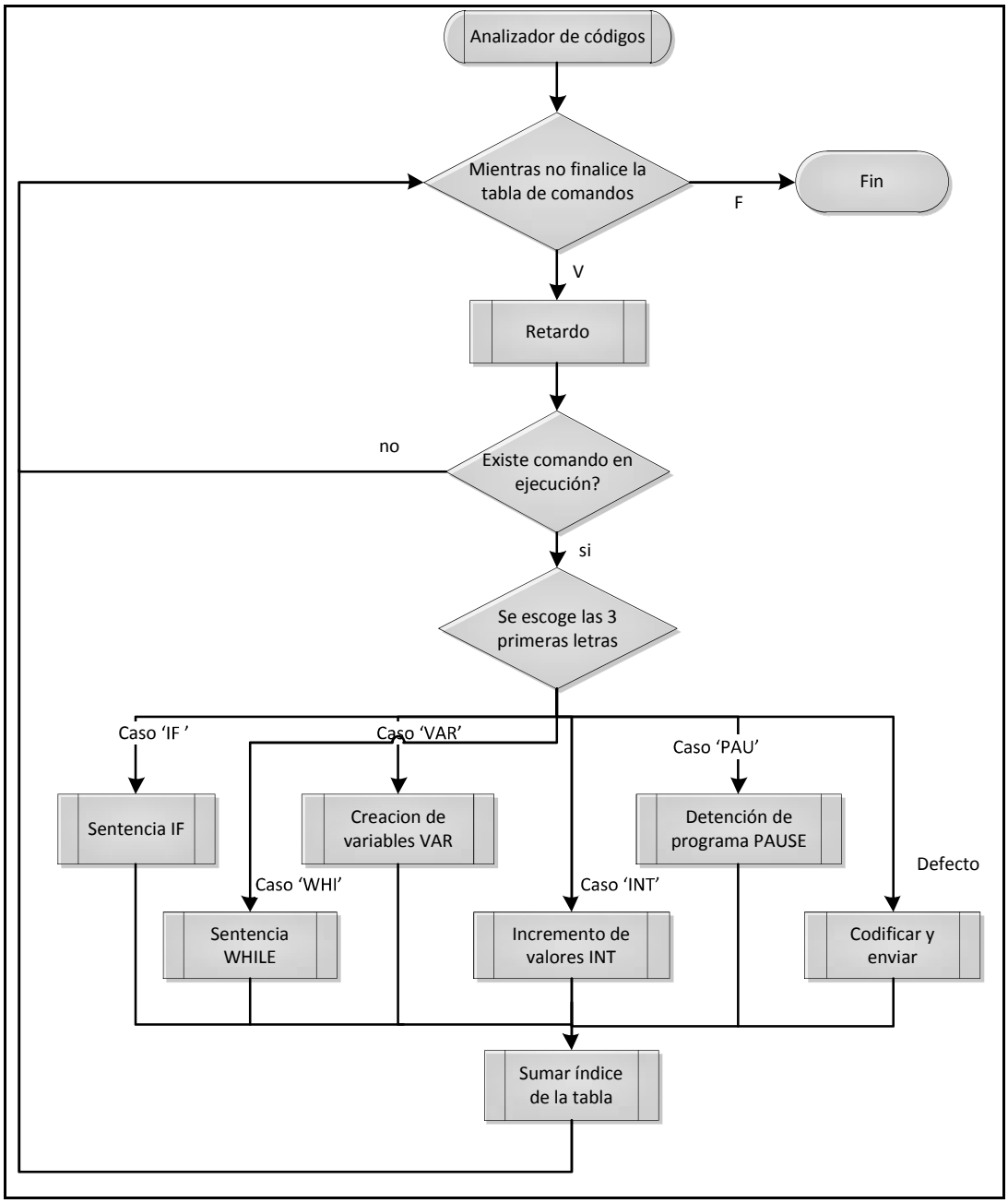


Figura. 124 Diagrama de análisis de sentencias de control de programa.

La figura. 124 se puede observar como el sistema discrimina estructuras de programa y comandos, esto se hace necesario ya que en modo secuencial el usuario tiene la capacidad de colocar lazos de sentencia condicional y de repetición permitiendo así que el manipulador realice tareas repetitivas

CAPÍTULO 4

PRUEBAS Y RESULTADOS

4.1 INTRODUCCIÓN

En este apartado se realizará las pruebas de funcionamiento de hardware, donde se analizará conceptos de precisión, exactitud, repetitividad, repetibilidad; pruebas de velocidad a valores fijos de set-point, conteo de pulsos en rango de trabajos de cada articulación. Con lo cual se espera conocer el comportamiento del sistema.

En cuanto lo concerniente a software se realizará la ejecución de la interfaz gráfica tanto en modo secuencial y terminal, se evaluará tiempos de transmisión y recepción de datos, ejecución de los comandos, interacción del teach pendant en simultaneidad con el software, y finalmente se pondrá a prueba el sistema con posibles casos de funcionamiento erróneos para comprobar la respuesta y seguridades del sistema.

4.2 PRUEBAS Y RESULTADOS DE HARDWARE

Se realizó pruebas de conteo de pulsos con las tarjetas anteriores del controlador CAD, para verificar el cumplimiento de los valores teóricos con los experimentales.

4.2.1 Pruebas Preliminares

Como se mencionó en el capítulo 3, las razones por la cual se decidió rediseñar la tarjeta de control, se debió a pérdidas de pulsos en el contador del microcontrolador, por motivos del inversor por disparo (74LS14). Las pruebas para determinar estos resultados se realizaron con el motor de la primera articulación a una frecuencia constante, variando el porcentaje de duty, en sentido horario y anti-horario.

Tabla. 17 Conteo de pulsos con tarjetas CAD.

Freq: 20.2 KHz	Teórico	Experimental CW (subida)	Experimental CCW (bajada)
Duty 25%	150 pulsos/10ms	144 pulsos/10ms	166 pulsos/10ms
Duty 50%	300 pulsos/10ms	207 Pulsos/10ms	222 Pulsos/10ms
Duty 75%	450 pulsos/10ms	327 pulsos/10 ms	339 pulsos/10 ms
Duty 100%	600 pulsos/10ms	413 pulsos/10ms	440 pulsos/10ms

En la tabla. 17, se puede observar que a un mayor porcentaje del duty, la pérdida de pulsos se vuelve considerables, evitando que se pueda determinar la posición correctamente del manipulador.

A continuación se realizan las pruebas anteriores con las nuevas tarjetas CDC2 de control y potencia. Los datos obtenidos son los siguientes. (Ver tabla. 18).

Tabla. 18 Conteo de pulsos con tarjetas CDC2.

Freq: 20.2 KHz	Teórico	Experimental CW (subida)	Experimental CCW (bajada)
Duty 25%	150 pulsos/10ms	148 pulsos/10ms	147 pulsos/10ms
Duty 50%	300 pulsos/10ms	298 Pulsos/10ms	297 Pulsos/10ms
Duty 75%	450 pulsos/10ms	445 pulsos/10 ms	443 pulsos/10 ms
Duty 100%	600 pulsos/10ms	592 pulsos/10ms	589 pulsos/10ms

Se puede observar que la pérdida de pulsos a porcentajes de duty alto no son tan considerables como ocurría con las anteriores tarjetas, lo que nos permite llevar un correcto conteo de pulsos cuando se analice velocidad angular.

Se realiza una comparación de las características principales entre la tarjeta de control CAD Y CDC2 de potencia.

Tabla. 19 Comparación entre tarjeta CAD y CDC2.

Características	Tarjeta CAD	Tarjeta CDC2
Número de microcontroladores	4 uC 16F877A	1 uC 18F452 5 uC 16F88
Memoria de programa utilizada.	Maestro: ROM55% RAM37% Esclavos:ROM20% RAM 15%	Maestro: ROM40% RAM9% Esclavos:ROM34% RAM 13
Componentes de filtraje	Filtro pasa bajo con inversor por disparo.	Filtro pasa bajo sin inversor por disparo.
Puertos de Entrada y Salida	No cuenta con puertos	4 estradas y 4 salidas digitales.

Se tiene que considerar que los microcontroladores esclavos de la tarjeta CAD, estaban programados el control de una articulación, si se deseaba implementar el control de dos articulaciones por microcontrolador, el uso de la memoria se hubiera duplicado. Además que el código de los microcontroladores esclavos del CDC2 son más extensos ya que controlan alrededor de quince comandos cada uno de ellos.

4.2.2 Pruebas de Conteo de Pulsos en cada Articulación

Para realizar pruebas de presión se utilizarán los comandos JOINT, MOTOR, PITCH Y ROLL. Los cual nos proporcionará conocer el porcentaje de error en el movimiento realizado por cada motor.

Tabla. 20 Conteo de pulsos y ángulo desplazado de Art. 1.

Movimiento Articulación 1				
	Conteo de pulsos	**% error**	**Angulo desplazado**	**% error**
Teóricos	18000	-	90º	-
Sentido CW	18001	0.005	91º	1.11
Sentido CCW	18001	0.005	91º	1.11

Tabla. 21 Conteo de pulsos y ángulo desplazado de Art. 2.

Movimiento Articulación 2				
	Conteo de pulsos	**% error**	**Angulo desplazado**	**% error**
Teóricos	9000	-	45º	-
Sentido CW	9001	0.005	46º	2.22
Sentido CCW	9001	0.005	40º	11.11

Tabla. 22 Conteo de pulsos y ángulo desplazado de Art. 3.

Movimiento Articulación 3				
	Conteo de pulsos	**% error**	**Angulo desplazado**	**% error**
Teóricos	18000	-	90º	-
Sentido CW	18001	0.005	93º	3.33
Sentido CCW	18001	0.005	91º	1.11

Tabla. 23 Conteo de pulsos y ángulo desplazado de Art. 4.

Movimiento Articulación 4				
	Conteo de pulsos	**% error**	**Angulo desplazado**	**% error**
Teóricos	2430	-	45º	-
Sentido CW	2431	0.005	46º	2.22
Sentido CCW	2431	0.005	40º	11.11

Tabla. 24 Conteo de pulsos y ángulo desplazado de Art. 5.

Movimiento Articulación 5				
	Conteo de pulsos	**% error**	**Angulo desplazado**	**% error**
Teóricos	18000	-	90º	-
Sentido CW	18001	0.005	91	1.11
Sentido CCW	18001	0.005	91	1.11

Obtenido los resultados de movimiento angular de cada uno de los motores, se comprueba que la primera y la quinta articulación realizan movimientos precisos, con un error relativamente pequeño, la tercera articulación tiene un porcentaje no mayor al 5% pero es considerable al momento de aplicar el concepto de repetibilidad.

La segunda y cuarta articulaciones tienen un porcentaje de error muy alto, dichos errores se hacen presentes ya que se encuentran perturbados por factores como: el peso que ejercen las articulaciones sobre una específica, la gravedad presente en movimientos de bajada, inercia por el frenado instantáneo que se realiza sobre ellos.

4.2.3 Pruebas de Repetibilidad

Para realizar estas pruebas se utilizaran los comandos MOVE, READY y CPATH, ya que estos permiten volver a localizaciones programadas las veces que sean necesarias desde un punto en cualquier localización y trasladarse otro punto indistintamente.

Se realizaron pruebas donde se todas las articulaciones se desplazaron con un ángulo específico, y que regresen a la posición inicial. Se detectó que al realizar estos movimientos repetitivos simultáneamente, el manipulador robótico cada vez pierde su orientación de posición inicial Home. Esto se ve afectado por las perturbaciones que se añaden a las articulaciones dos, tres y cuatro.

Como se ven los resultados el robot cuenta con la capacidad de repetibilidad, pero al tener errores significativos, en ciertas articulaciones hace que cada vez que regresa al punto programada aumenta cada el porcentaje de error, ya que el movimiento simultaneo es decir un error de un motor afecta a la otra por acción de las cadenas.

4.3 PRUEBAS Y RESULTADOS EN SOFTWARE

En este apartado se realizan las pruebas de funcionamiento para determinar las características del sistema, si existen posibles errores posteriormente serán corregidos.

4.3.1 Pruebas de Transmisión de Datos desde Pc a Controlador y de Controlador a Pc

Al realizar el envío de datos para el comando Move hacia el controlador, se detectó que al transmitir datos consecutivamente sin tiempo de retardo, estos se perdían lo que conlleva a que el controlador realice operaciones erróneas.

De igual manera cuando se intentaba enviar datos del comando Here desde el controlador hacia la PC sucedía el mismo efecto anterior se perdían localizaciones de algunos motores.

Para dar solución a dicho problema se determinó enviar los datos con un retardo adicional tanto desde el controlador a la PC y viceversa.

Por lo cual el envío secuencial de datos desde la PC se debe considerar un tiempo no menor a 10 milisegundos, adicionalmente el tiempo mínimo de envío desde el controlador debe ser no menor a 250 milisegundos.

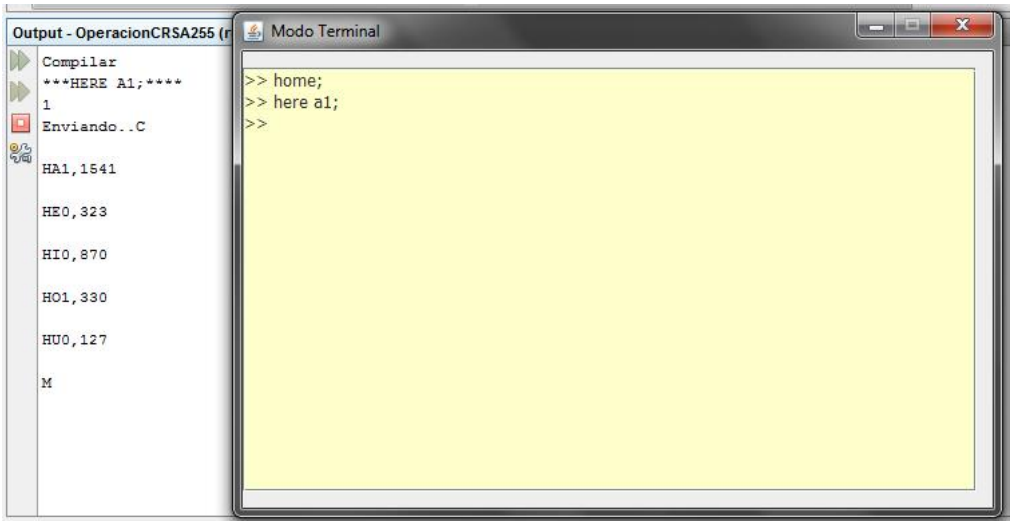


Figura. 125 Recepción de datos con el comando Here.

Como se observa en la figura.125, se recibió toda la información por parte del controlador al realizar el comando HERE desde la interfaz terminal.

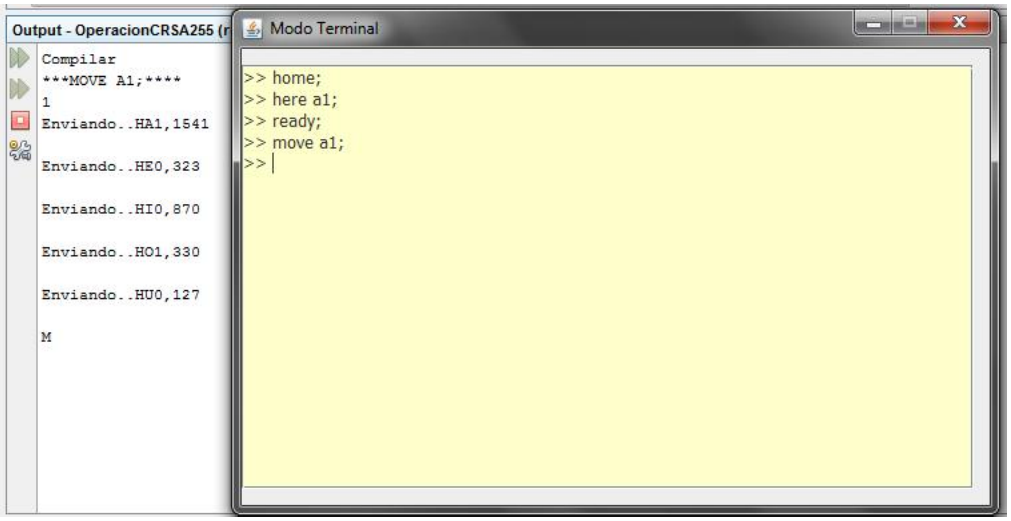


Figura. 126 Envío de datos con el comando Move.

Como se observa en la figura. 126, al realizar el comando MOVE el software envía la información necesaria para que el controlador la ejecute.

4.3.2 PRUEBAS DE SEGURIDADES DEL SISTEMA

Se realizó pruebas de respuestas del sistema cuando se acciona el paro de emergencia y posteriormente se hizo pruebas de seguridad de movimientos dentro del rango.

A continuación se muestra una figura de la activación de paro de emergencia en modo terminal.

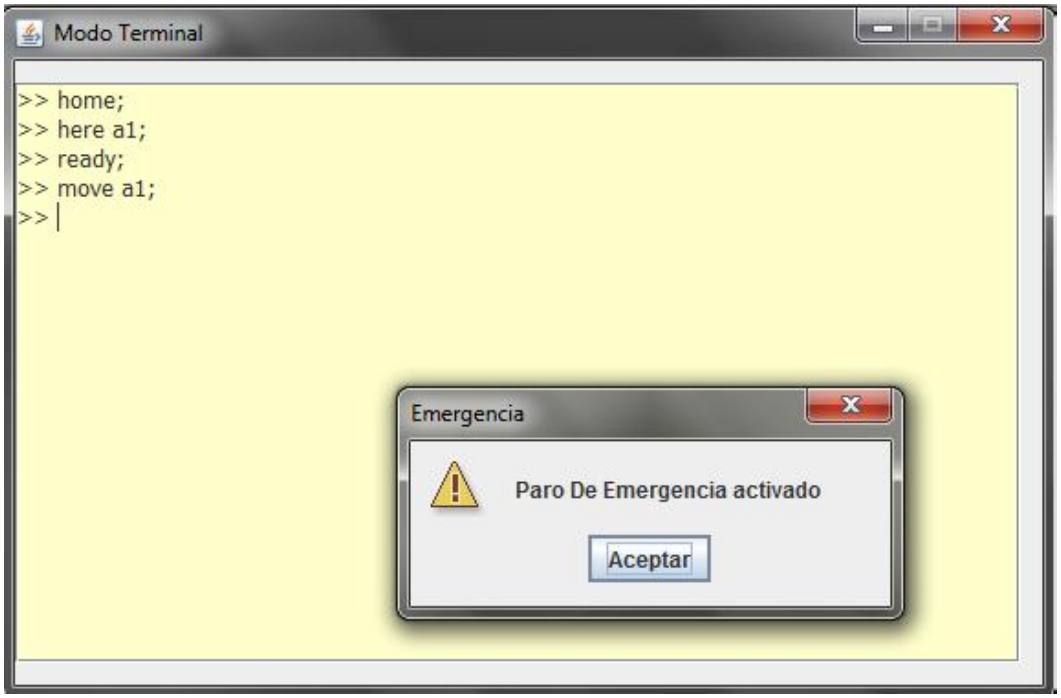


Figura. 127 Activación de paro de emergencia en modo terminal.

A continuación se realiza la prueba de paro de emergencia en modo secuencial, para lo cual se abre un programa ya realizado y se lo ejecuta

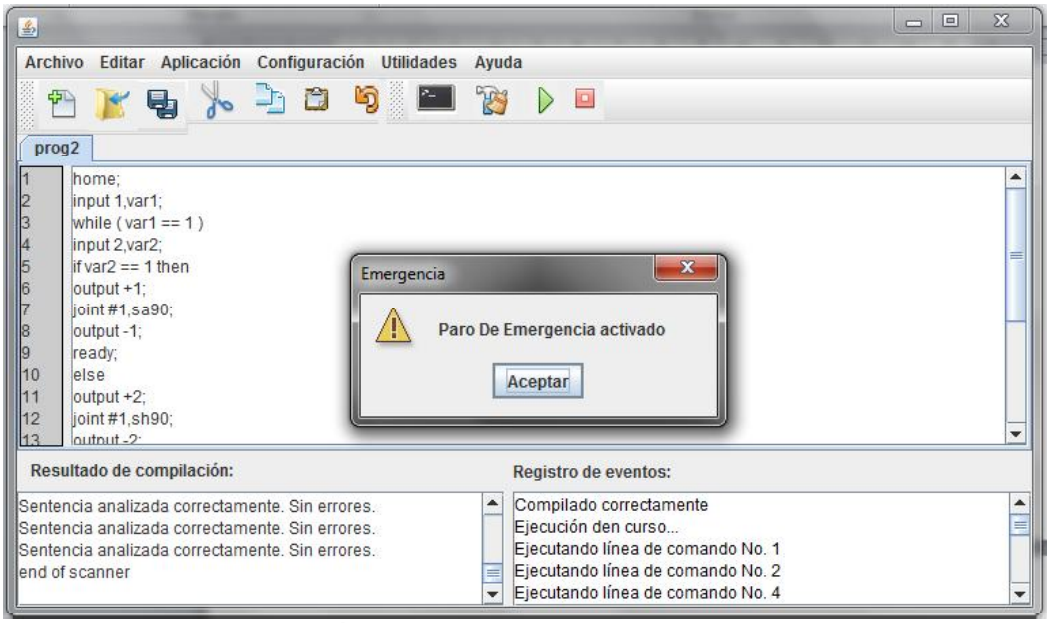


Figura. 128 Activación de paro de emergencia en modo secuencial.

De igual forma se despliega un mensaje de paro de emergencia por el LCD del Teach Pendant.

Figura. 129 Mensaje de paro de emergencia en Teach Pendant.

Adicional otra seguridad que es necesaria es delimitar los movimientos de ciertos comandos ya sea que estos se desplacen cierta cantidad de ángulos o por número de pulsos. A continuación se realiza pruebas sobre el modo terminal y posteriormente sobre el modo secuencial.

Figura. 130 Ventana de fuera de rango en modo terminal.

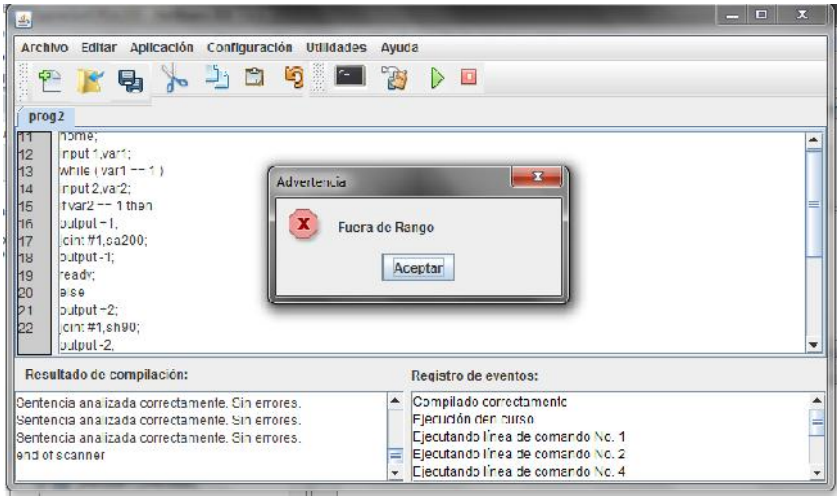


Figura. 131 Ventana de fuera de rango en modo secuencial.

En el Anexo. 8 se encuentra el manual de usuario del funcionamiento del sistema.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Al comandar con un solo microcontrolador el movimiento de un motor, este dota de mayores recursos y capacidades para realizar un mejor desempeño del sistema, ya que no se comparte: memoria, módulos temporizadores, módulos contadores y puertos con otro motor.

- Para el movimiento simultáneo de las cinco articulaciones del manipulador robótico, se establece que cada microcontrolador esclavo tome las decisiones para el correcto funcionamiento de un motor, realizando cálculos para el movimiento y solo depender del maestro para recibir y enviar datos. Sin importar las acciones que estén realizando los demás esclavos.

- Para realizar el movimiento del manipulador a localizaciones previamente guardadas, se implementó un algoritmo de movimiento simultáneo de las cinco articulaciones, sin que involucre un desarrollo matemático complejo, dotando al sistema de la capacidad de trasladarse de una posición a otra sin tener que pasar por la posición de HOME, teniendo en cuenta que no genera una trayectoria especifica.

- El uso de herramientas y librerías que ofrece el lenguaje java, como JLex y Java Cup, han permitido adaptar la estructura de los comandos seleccionados a un compilador básico, que permite de manera independiente del programa fuente, conocer si existe un error en la gramática del comando, así poder prevenir de posibles errores, antes que se envíe información al controlador y provoque un funcionamiento erróneo.

- Para la implementación del analizador semántico, se optó por desarrollar un algoritmo que seleccione las tres primeras letras del comando escrito por el usuario, esto permite que se optimice los recursos de procesamiento de información por parte de la PC, logrando que la codificación del mismo sea realice en un menor tiempo.

- Al implementar la segunda etapa del proyecto de rehabilitación de los controladores del manipulador robótico CRS A255, se ha logrado que la cátedra de robótica cuente con prototipos funcionales para realizar prácticas de laboratorio y así contribuir para un mejor aprendizaje de los estudiantes.

- El uso de clases tipo hilo en la programación del software, es de gran importancia ya que permiten realizar subprocesos que no tienen mayor relevancia en la ejecución del programa principal, es decir esto permite que mientras el usuario este interactuando con la interfaz, otra operación puede estar ejecutándose, sin importar que este haya finalizado la operación para comenzar una nueva acción.

5.2 RECOMENDACIONES

- Antes de utilizar el software implementado es recomendable, primero revisar la sintaxis de las instrucciones, y como se deben escribir las mismas, además de conocer la operación que realiza cada comando sobre el manipulador robótico.

- Es recomendable familiarizarse con el manipulador en modo manual con el Teach Pendant, antes de trabajar con el sistema en modo online, ya que se puede conocer las características de movimiento de cada articulación.

- Hay que tomar precauciones en la programación de microcontroladores cuando se utiliza diferentes familias como los PIC’s 18f y 16f, ya que no mantienen las mismas estructura dentro de un compilador, esto provoca incorporar librerías adicionales para el correcto funcionamiento.

- Se recomienda para la realización de trabajos futuros dotar al sistema de una técnica de control avanzada, lo cual permitirá que el mismo pueda trabajar a velocidades diferentes, dependiendo de las tareas a realizar.

- Un aporte adicional para el mejoramiento del software, es la optimización del compilador implementado, por uno más avanzado que permita adicionar mayores prestaciones, mejorando la estructura de las sentencias, auto competición de instrucciones, opciones de corrección, entre otros.

- Es recomendable el estudio de técnicas y normas para el diseño de interfaces graficas ya que así el usuario puede interactuar amigablemente con el software, además el mismo debe contar con herramientas que faciliten el trabajo como ayudas, manual de usuario, botones de acceso rápido y avisos emergentes.

- Se recomienda el desarrollo de la técnica de control sobre el modulo temporizador TIMER1 ya que se dejó este módulo activado para no dañar la programación ya realizada.

- La realización de prácticas de laboratorio, simulando que el manipulador robótico se encuentra en un ambiente industrial. Como son: procesos de selección, control de calidad, desarrollo de producción, entre otros.

- Dotar al software de comunicación DDE, esto permite que el mismo pueda comunicarse con otros programas, y así poder complementarlo con un sistema SCADA.

BIBLIOGRAFÍA

- Barrientos, A., Peñín, L., Balaguer, C., & Aracil, R. (2007). *Fundamentos de Robótica* (Primera ed.). Madrid, España: McGRAW-HILL.
- Chacón, A. (2005). *Manual de Mantenimiento.* Sangolquí: Espe - Sangolquí.
- Craig, J. (2006). *Robótica* (Tercera ed.). México Df: Pearson educación.
- Gadgets, H. (13 de 12 de 2007). *Hacked Gadgets*. Obtenido de Toyota robotic musical: http://hackedgadgets.com/2007/12/13/toyota-robotics-musical-robots/#section3
- Galvez, S., & Mora, M. (2005). *Compiladores.* Málaga: Universidad de Málaga.
- Kuka. (2013). *Kuka-robotics*. Obtenido de Products: http://pdf.directindustry.es/pdf/kuka-roboter/nuevos-rapidos-precisos-kuka-small-robots/17587-435871.html
- Maldonado, D., & Sánchez, A. (2013). *Estudio, Diseño e implementación de un controlador de tres grados de libertad para el manipulador robótico crs a255.* Sangolquí: Escuela Politécnica del Ejército.
- Mocencahua Mora , D. (2009). *Facultad de ciencias electrónicas*. Obtenido de Ftp Descargas: http://www.ece.buap.mx/ini/des.html
- Ollero Baturone, A. (2001). *Robótica Manipuladores y Robots Móviles.* Barcelona, España: Marcombo.
- Platea, & Gonzalez F, V. (2013). *Control y Robótica*. Obtenido de Clase de robots: http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.2.htm

- Reyes Cortés, F. (2011). *Robótica Control de Robots Manipuladores.* México Df: Alfaomega.
- Sass, L. (25 de 07 de 2013). *Introduccción a la Robótica.* Obtenido de Universidad de San Francisco de Quito: http://profesores.usfq.edu.ec/laurents/IME440/IME440_RobotManip.pdf.
- Williams, K. (2006). *Thinkbotics*. Obtenido de Products: http://www.thinkbotics.com/products.html

ACTA DE ENTREGA

El proyecto fue entregado al Departamento de Eléctrica y Electrónica y reposa en la Universidad de las Fuerzas Armadas – ESPE, desde:

Sangolquí, 2 DE ABRIL de 2014

ELABORADO POR:

DOUGLAS ISRAEL
CASA MONTALEZA

171898407-1

DANILO XAVIER
COQUE CISNEROS

171885710-3

AUTORIDAD

Ing. Luis Orozco Msc.

DIRECTOR DE LA CARRERA DE ELÉCTRICA Y ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL