



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**Teleoperación de un SCORBOT-ER 4u
empleando un PAMCA.**

T E S I S
Que para obtener el título de
INGENIERO MECATRÓNICO

P R E S E N T A

JUAN ANTONIO REMENTERIA LÓPEZ.

**DIRECTOR DE TESIS:
Dr. VÍCTOR JAVIER GONZÁLEZ VILLELA.**

**CODIRECTOR DE TESIS:
M.I. OCTAVIO DÍAZ HERNÁNDEZ.**



MÉXICO D.F.

NOVIEMBRE 2011



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

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

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

Agradezco en primer lugar a Dios por permitirme tener
La oportunidad de estudiar la carrera que yo amo.
Después agradezco a mi México querido por permitirme
A través de la Universidad Nacional Autónoma de México
Concluir mis estudios universitarios.
Agradezco a mi Amá por su apoyo incondicional todos
Estos años. Y también al Godfried por echarme la mano
Para seguir adelante.
Destaco también el apoyo total que siempre estuvo presente
En mis hermanas y hermanos, tanto de sangre como elegidos
Por mi mismo. Les aseguro que sin su comprensión y alegría
En los momentos difíciles esto hoy no sería posible.
Finalmente agradezco al Real Madrid por enseñarme que
Los Titanes también lloran. Al Toluca por enseñarme que hasta
Los más humildes pueden hacer y merecer grandes cosas.
Y a ella, que siempre ha sido mi guía en los momentos más
Oscuros y me ha dado una razón para esforzarme y ser
Parte de "Die Besten".
Para concluir agradezco a la DGAPA a través del proyecto
PAPIIT IN115811: "Investigación y desarrollo en sistemas
Mecatrónicos: robótica móvil, robótica paralela, robótica
Híbrida y teleoperación (2011-2013)" por el apoyo brindado
Para el desarrollo de mi tesis.

El hombre que ama lo que hace, está condenado al éxito.

Facundo Cabral.

El verdadero progreso es el que pone la tecnología al alcance de todos.

Henry Ford.

Para aquellos que luchan, la vida tiene un sabor que el temeroso jamás conocerá.

El Usurero. *Sucker Punch.*

Índice.

Resumen-----	3
1. Teleoperación en la robótica-----	5
1.1. La teleoperación-----	6
1.2. Teleoperación y robótica (Telerobótica)-----	15
1.3. Teleoperación de manipuladores antropomórficos-----	19
1.4. Trabajos Previos-----	22
1.5. Organización del trabajo-----	25
2. Descripción y análisis del proyecto-----	28
2.1. Método de Denavit-Hartenberg-----	29
2.1.1. Modelado cinemático por Denavit-Hartenberg para un dispositivo de cadena abierta de 5 grados de libertad-----	31
2.1.2. Ángulos de Euler-----	34
2.1.3. Cinemática inversa por Denavit-Hartenberg-----	37
2.2. Pantógrafo Maestro de Cadena Abierta (PAMCA)-----	41
2.2.1. Comunicación Serial-----	42
2.3. SCORBOT-ER 4u-----	45
2.3.1. Controller-A-----	49
2.4. Servo-Robot "Black Hawk"-----	52
2.4.1. Comunicación Serial RS232-----	53
2.5. Internet y sus protocolos de comunicación-----	54
2.5.1. TCP/IP-----	59
2.5.2. Protocolo TCP-----	61
2.6. Interfaz gráfica del sistema Maestro-Eslavo-----	66
2.6.1. Interfaz Maestro-----	69
2.6.2. Interfaz Esclavo-----	75
3. Implementación del proyecto-----	78
3.1. Implementación del maestro-----	79
3.2. Implementación del esclavo-----	87
3.3. Comunicación vía internet-----	100
4. Experimentación y resultados con el sistema teleoperado-----	109
4.1. Experimentación y resultados de la interfaz maestro-----	111
4.1.1. Ángulos y orientaciones adquiridas y ejecutadas por la interfaz maestro-esclavo-----	111
4.1.2. Adquisición del dispositivo instrumentado-----	116
4.1.3. Tiempo de retraso para adquisición de puntos-----	120
4.1.4. Retrasos en la imagen de la cámara en el maestro cuando proviene del esclavo empleando al internet en la tapa de la teleoperación-----	121
4.2. Experimentación y resultados del interfaz esclavo con el SCORBOT-ER 4u-----	122
4.3. Proceso funcional completo-----	126
4.3.1. Retrasos en la imagen en el maestro cuando proviene del esclavo empleando al internet en la tapa de la teleprogramación y el telemonitoreo-----	127

4.4. Aplicación de la interfaz en el robot " <i>Black Hawk</i> "-----	128
4.4.1. Trayectoria y orientaciones del recorrido por parte del pantógrafo para la interfaz del maestro-----	128
5. Discusión y Conclusiones-----	132
5.1. Discusión-----	133
5.2. Conclusiones-----	135
Bibliografía-----	138
Referencias de figuras-----	141
Anexos-----	144

Resumen.

La teleoperación es un conjunto de tecnologías que tienen por objetivo realizar la operación a distancia de algún dispositivo por parte de un operador humano. La importancia de estas técnicas radica en las aplicaciones posibles y la forma en que transforman los procesos y procedimientos.

El propósito principal del proyecto de esta tesis fue realizar la teleoperación de un SCORBOT-ER 4u vía internet empleando un Pantógrafo Maestro de Cadena Abierta (PAMCA) con la función de reproducir movimientos puntuales y trayectorias, y como objetivos secundarios realizar teleprogramación y telemonitoreo del proceso, además de mostrar la flexibilidad de la interfaz desarrollada en un robot impulsado por servomotores llamado "*Black Hawk*".

Para lograr los objetivos de esta tesis se realizaron dos interfaces en C# que conforman el binomio maestro-esclavo y con ello permiten la teleoperación de los manipuladores empleando internet, aplicando el protocolo de comunicación TCP/IP.

El maestro comprende de tres elementos; el primero es el PAMCA que permite la adquisición de puntos en el espacio. El segundo elemento es la interfaz en la computadora que tiene por responsabilidad generar la comunicación entre acciones y el operador humano, además de permitirle al humano generar los procedimientos deseados. El tercer elemento es la conexión con el internet que permite el flujo de información entre maestro y esclavo.

Por su parte el esclavo comprende de tres elementos; el primero es la interfaz en una PC que permite la interpretación de las instrucciones del maestro. El segundo elemento es el manipulador, que realizará las acciones provenientes de la interfaz. Y el tercer elemento es la conexión con el internet.

Los objetivos tanto principales como secundarios se cumplieron cabalmente, sin embargo se presentaron grandes limitantes en la operación. Estas limitantes en el caso del SCORBOT-ER 4u son los largos retardos presentes, debidos en gran medida al *hardware* utilizado. Lo que hace al proceso lento. En el caso de aplicar la interfaz a otro robot, se encontró que también las limitaciones provienen del *hardware* utilizado, en este caso el robot "*Black Hawk*" presentó problemas relacionados con la dinámica del mismo, no tanto con la velocidad del proceso.

La conclusión fundamental de este proyecto recae principalmente en los beneficios que se obtienen al generar conocimiento sobre los dispositivos teleoperados. Es decir, ahora tenemos un sistema capaz de manipular, programar y monitorear a distancia un robot de cinco grados de libertad empleando el internet. También este proyecto permitió ubicar que la efectividad en la velocidad de la ejecución para la teleoperación de un SCORBOT-ER 4u, está muy ligada al *hardware* de control utilizado para ejecutar y controlar los movimientos del mismo. Los retardos debidos a programación, procesamiento, retardos en la comunicación, *hardware* o la suma de estos son los elementos que más afectan en la ejecución de un procedimiento de teleoperación.

Teleoperación en la Robótica.

La tecnología no es en si el fin, sino el medio entre la sociedad del conocimiento y el desarrollo mundial.

Teleoperación.

La teleoperación es un conjunto de tecnologías que comprenden la operación o gobierno a distancia de un dispositivo por un ser humano. Por tanto, teleoperar es la acción que realiza un ser humano de operar o gobernar a distancia un dispositivo; mientras que un sistema de teleoperación será aquel que permita teleoperar un dispositivo, que se denominará dispositivo teleoperado. (Ejemplos de dichas aplicaciones Figura 1 y 2).

Al conjunto de tecnologías que comprenden la operación o gobierno a distancia por un ser humano de un manipulador se le llama telemanipulación. Telemanipular es la acción que realiza un ser humano de operar o gobernar a distancia un manipulador, mientras que un sistema de telemanipulación será aquel que permita teleoperar un manipulador. Otro atributo muy presente en los sistemas de teleoperación modernos es la telepresencia, que es la situación o circunstancia que se da cuando un ser humano tiene la sensación de encontrarse físicamente en un lugar remoto, dicho evento se consigue realimentando coherentemente al ser humano con la suficiente cantidad de información sobre el entorno remoto del dispositivo teleoperado.

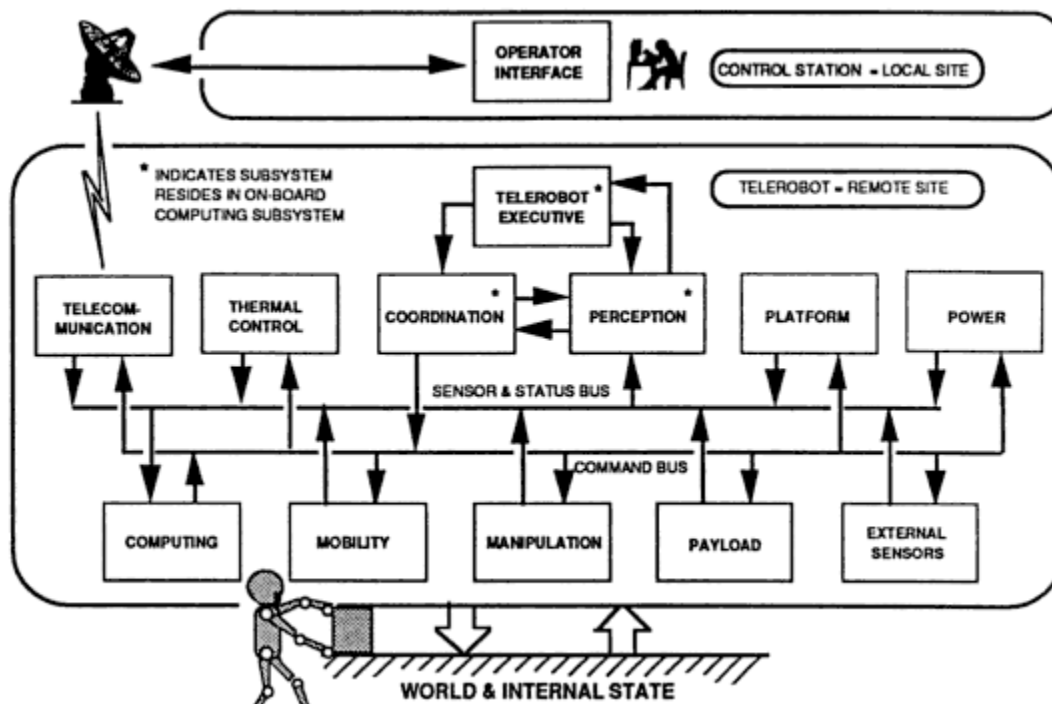


Figura 1. Sistema de Teleoperación de un Robot espacial

Adicionalmente, podemos identificar elementos sensoriales que tienen el objetivo de dar más información al ser humano con el fin de incrementar la percepción del sistema teleoperado. Cuando se genera una situación o circunstancia en la cual el ser humano tiene la sensación de encontrarse en un lugar distinto de donde físicamente está gracias a la información generada exclusivamente por la computadora. El entorno que se genera, y en el que el operador se encuentra inmerso se denomina entorno virtual, y la situación de estar en él, también se conoce como presencia virtual. Por otra parte la realidad aumentada, que consiste en una situación o circunstancia que percibe un operador cuando la información sensorial que es realimentada de un entorno es modificada previamente por una computadora con el objetivo de añadirle nueva información generada artificialmente y que es inaccesible directamente con los sentidos del operador, aunque este se encuentre en la zona remota. (Ver figura 4).

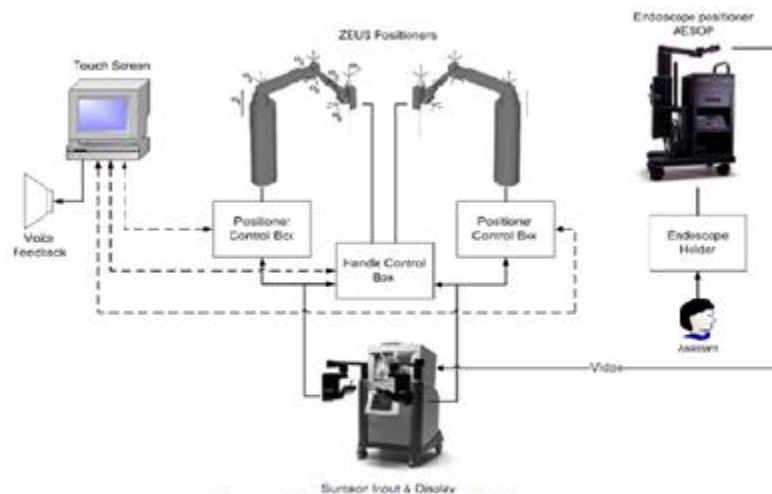


Figura 2. Sistema para cirugías Zeus

Aparte de las retroalimentaciones puramente visuales podemos encontrar la retroalimentación táctil (que consiste en la sensación de contacto aplicada a la piel), la retroalimentación cenestésica o de fuerzas (que consiste en una resistencia al avance o un peso que hace referencia a la excitación de los sensores corporales) y la retroalimentación háptica (que consiste en la retroalimentación de contacto tanto táctil como de fuerzas). Dependerá de la instrumentación y del tipo de aplicación a realizar la retroalimentación hacia el usuario. Más adelante se dará una explicación más completa de este tema.

Un elemento fundamental cuando hablamos de teleoperación se refiere a la supervisión y el control humano, que en sentido estricto significa que uno o más seres humanos están intermitentemente programando y se encuentran recibiendo información de manera continua de una computadora que interconecta los sensores y actuadores al proceso de control¹ o al ambiente de trabajo. En sentido menos estricto estamos hablando que la programación es continua al igual que la recepción.

Comúnmente se presentan cinco diagramas de control e interfaz hombre-máquina con un operador humano, y que también incluyen los sensores y actuadores que interactúan directamente con el proceso de control o el ambiente de trabajo.

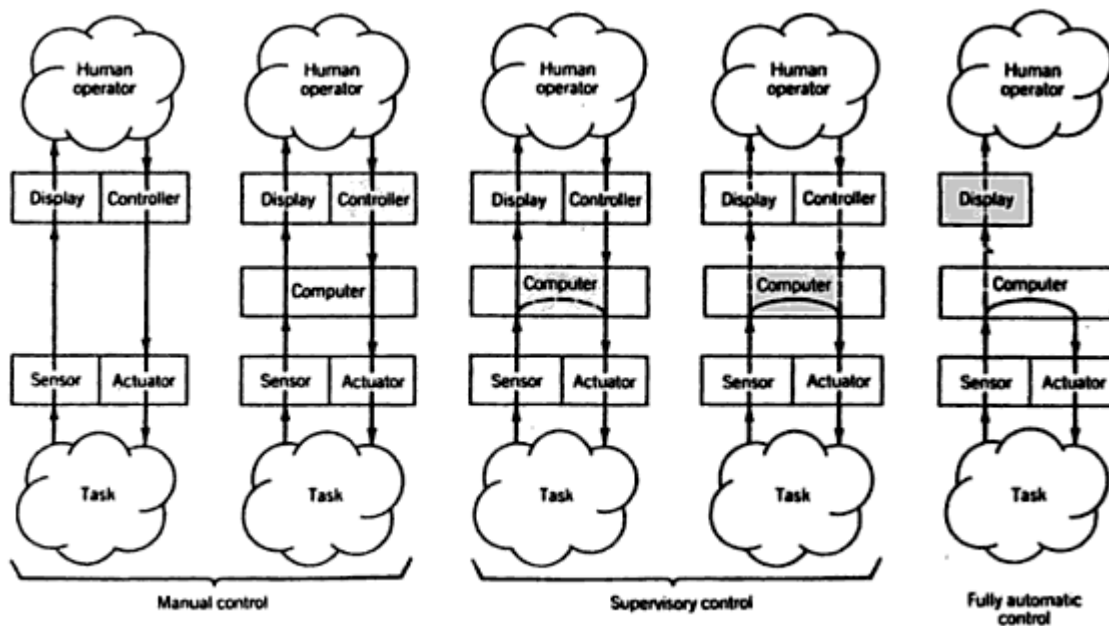


Figura 3. Modos de control

En la figura 3 podemos observar los diagramas de control e interfaz hombre-máquina anteriormente mencionados, en el diagrama 1 podemos observar un control manual total, inclusive sin computadora. En el sistema 2 encontramos una transformación significativa de procesamiento empleando la computadora, inclusive en este punto la computadora podría ayudar a las tareas de control tanto en el proceso de adquisición de datos como en los actuadores. Esta figura corresponde a la definición menos estricta de control de

¹ Sheridan, Thomas B. *Teleoperatorics, automation and human supervisory control*. (1992). USA: Massachusetts Institute of Technology Press. Página 1.

supervisión. Se denota, en ambas figuras que las decisiones dependen plenamente del operador humano. Si el humano se detiene en algún momento, el control también lo hace.

Cuando en una menor parte (figura 3) o mayor parte (figura 4) el control es acompañado de lazos de control cerrados empleando la computadora corresponde a la definición de control supervisado estrictamente. Y finalmente en la figura 5 encontramos implementado un sistema de control y esencialmente el sistema es automático (el operador humano puede observar pero no puede influenciar el proceso), ya no es control supervisado.

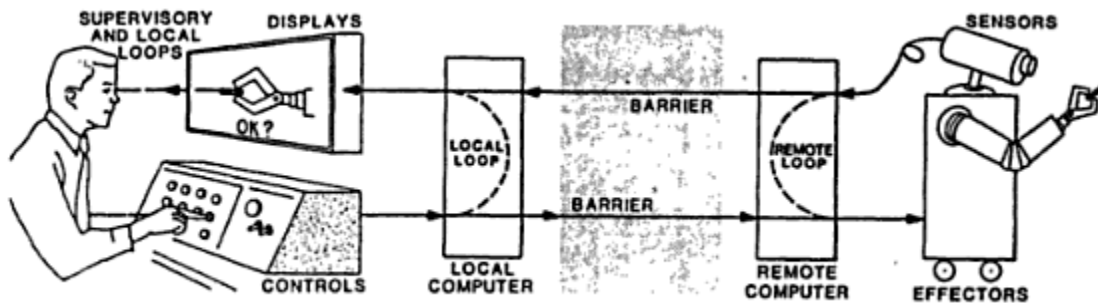


Figura 4. Concepto Básico de Teleoperación

Beneficios para el desarrollo del control supervisado:

- Alcanzar la precisión y fidelidad de la máquina sin sacrificar la capacidad cognitiva y adaptativa del humano.
- Hacer un control más rápido y sin las limitantes de las capacidades sensomotrices del ser humano.
- Hacer un control más fácil al hacer que el operador solamente provea de instrucciones en términos de mover objetos, objetivos a ser cumplidos, ligar instrumentos a utilizar o señales de control a enviar.
- Eliminar la necesidad de una atención humana continua y reducir la carga de trabajo para el operador.
- Hacer el control posible; inclusive ante retardos de comunicación entre el ser humano y el dispositivo teleoperado.

- Proveer de un sistema anti fallos ante una eventual falla catastrófica humana.
- Salvar vidas y reducir los costos eliminando la necesidad de un operador presente en ambientes hostiles, y el quipo de apoyo para la vida requerido para enviar al operador allí.



Figura 5. Primer Teleoperador Maestro-Eslavo Mecánico

En 1947 comenzaron las primeras investigaciones, lideradas por Raymond Goertz del *Argonne National Laboratory* en Estados Unidos, encaminadas al desarrollo de algún tipo de manipulador de fácil manejo a distancia mediante el uso por parte del operador de otro manipulador equivalente. En 1948 se desarrolló el primer manipulador teleoperado mecánico, denominado M1, antecesor de sistemas maestro-esclavo de telemanipulación existentes actualmente. El mecanismo del sistema permitía que la pinza situada en el extremo del manipulador esclavo reprodujera de forma fiel los movimientos hechos por el maestro. Ambos manipuladores eran prácticamente iguales, y los movimientos que producían eran eje a eje, de tal manera que describiesen la misma trayectoria.

A principio de los años cincuentas comenzaron desarrollos encaminados a motorizar ambos manipuladores (maestro y esclavo) de forma adecuada. Fue en el año de 1954 que Goertz presentó el primer manipulador maestro-esclavo con acondicionamiento eléctrico y servo control en ambos manipuladores llamado E1.

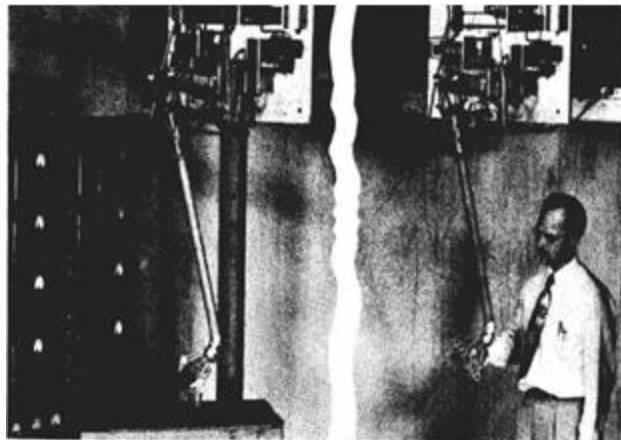


Figura 6. Manipulador Mestro-Eslavo E1

En los años sesenta se extendieron las investigaciones hasta el campo de las aplicaciones submarinas, siendo los sistemas cada vez más sofisticados, especialmente con la inclusión de cámaras y demás dispositivos para aumentar la telepresencia del operador.

A finales de los años sesenta y principio de los setenta, la tecnología de la teleoperación alcanzó su mayoría de edad con su utilización en aplicaciones espaciales. Aparecieron nuevos retos y problemas, siendo de especial relevancia la existencia de retrasos temporales en la comunicación entre la zona local y remota.

En paralelo con la evolución histórica de las técnicas de teleoperación ha habido una evolución tecnológica motivada por los desarrollos de control, la informática y la robótica.



Figura 7. Radio Control Camera

Ha habido, por una parte una evolución en los sistemas de comunicación, pasando de los sistemas mecánicos a los eléctricos, fibra óptica, radio e Internet, medio que suprime prácticamente las limitaciones de distancia.

La incorporación de las nuevas técnicas de desarrollo de la robótica y la tecnología multimedia han permitido incrementar las capacidades del sistema remoto, especialmente en lo que se refiere a autonomía y del puesto local de control, mejorando fundamentalmente las presentaciones de la interfaz hombre - máquina. (Ver figura 7, 8, 9 y 10).

Elementos de un sistema de teleoperación:²

1. Operador o teleoperador: Es un ser humano que realiza a distancia el control de la operación. Su acción puede ir desde un control continuo hasta una intervención

² Alencastre M., Muñoz L. Teleoperating Robots in Multiuser Virtual Environments.(2003).MEX:ENC. Página 314.

intermitente, con la que únicamente se ocupa de monitorizar y de indicar objetivos y planes cada cierto tiempo.

2. Dispositivo teleoperado: Puede ser un manipulador, un robot, un vehículo o dispositivo similar. Es la maquina que trabaja en la zona remota y que está siendo controlada por el operador.

3. Interfaz: Conjunto de dispositivos que permiten la interacción del operador con el sistema de teleoperación. Se considera al manipulador maestro como parte de la interfaz, así como a los monitores de video, o cualquier otro dispositivo que permita al operador mandar información al sistema y recibir información del mismo.



Figura 8. Robot con control Wireless

4. Control y canales de comunicación: Conjunto de dispositivos que modulan, transmiten y adaptan el conjunto de señales que se transmiten entre la zona remota y la local. Generalmente contará con uno o varias unidades de procesamiento.

5. Sensores: Conjunto de dispositivos que recogen la información, tanto de la zona local como de la zona remota, para ser utilizada por la interfaz y el control.



Figura 9. Robot Teleoperado de aplicaciones militares

Tipos de control:

1. Mecánicos:

Los comandos de controles se transmiten mecánicamente o hidráulicamente al dispositivo teleoperado. La retroalimentación es visual, y puede ser directa o mediante un monitor. Esta teleoperación es típica para manipular materiales peligrosos así como también micromanipulaciones.

2. Remotos:

El operador tiene la mayor parte del tiempo contacto visual con el dispositivo teleoperado. Los comandos de control son enviados por radio, cables o eléctricamente.

3. Normal o Estándar:

Control inalámbrico y retroalimentación visual empleando un sistema cámara con monitor.

4. Semi o totalmente autónomo: El robot es controlado cuando se le necesita. La retroalimentación es mediante la cámara del robot y el monitor del operador. El ser humano no requiere realizar todas las acciones, ya que solo requiere “control de supervisión”.

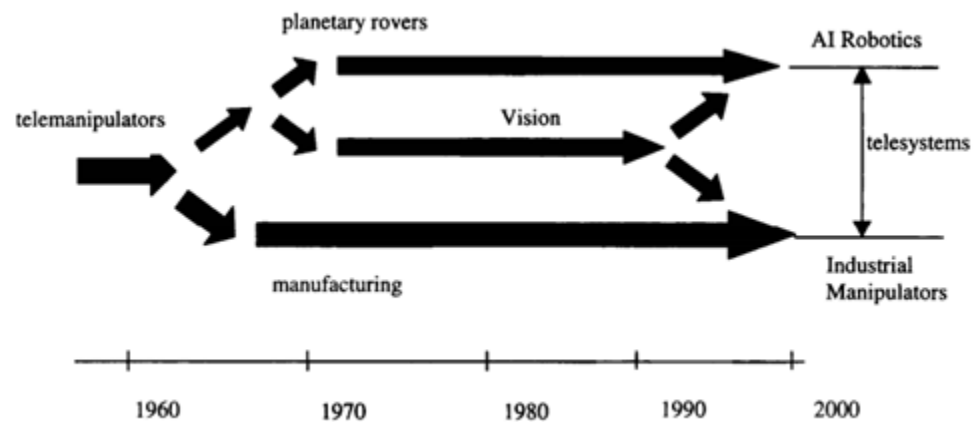


Figura 10. Tecnologías aplicadas a la Teleoperación

Tipos de teleoperación:

1. Control de lazo cerrado (Teleoperación directa): El operador controla los actuadores del dispositivo teleoperado con señales directas y obtiene una retroalimentación en tiempo real. Esto es solo posible cuando los retardos en el lazo de control son mínimos.
2. Coordinada (Teleoperación offline): El operador de nuevo controla los actuadores, pero ahora tenemos un control interno incluido, sin embargo este último no cuenta con autonomía. Los controles internos son usados solamente para cerrar los ciclos de control que el operador es incapaz de controlar debido a los retardos.
3. De control supervisado: El dispositivo teleoperado puede ahora formar parte de las decisiones más o menos autónomas, mientras que el operador monitorea principalmente los comandos de mayor rango.

Ventajas de la teleoperación:

- ✚ Sistemas más confiables y preparados para la solución de objetivos.
- ✚ Mejora en las condiciones de seguridad para el ser humano.
- ✚ Participación en el avance tecnológico.
- ✚ Más apreciación de la inteligencia humana en contraste con la inteligencia artificial.
- ✚ Reducción del trabajo humano.

Desventajas de la teleoperación:

- ✚ Aumento de la tasa de desempleo para los trabajadores menos preparados.
- ✚ Problemas para detectar fallas en sistemas automáticos (Fallas silenciosas).
- ✚ Insatisfacción y problemas ante crisis en el trabajo.
- ✚ Sensación de no contribución.
- ✚ Abandono de la responsabilidad.
- ✚ Centralización del manejo del control del trabajo y reducción del control del trabajador.

- ✚ Menos sociabilización.
- ✚ Pérdida de habilidades técnicas.
- ✚ Ignorancia tecnológica.
- ✚ Intimidación ante gran poder y gran responsabilidad.

Teleoperación y Robótica (Telerobótica).

Un telerobot es una forma avanzada de dispositivo de teleoperación, su comportamiento es supervisado por un operador humano empleando una computadora como intermediaria. (Imagen ejemplo, figura 11).



Figura 11. Robot de Exploración Espacial.

Esto significa que el operador humano se comunica intermitentemente con el dispositivo empleando la información recibida con la interfaz intermediaria para establecer metas, planes, contingencias, restricciones, asumpciones, sugerencias y ordenes relativas al ambiente de trabajo remoto, en repuesta

obtendremos información íntegra acerca de las metas logradas, dificultades, e información sensorial. El robot subordinado ejecuta las indicaciones recibidas del operador humano y en ocasiones contará con el plus de su inteligencia artificial y sensorial.

Aplicaciones principales de la telerobótica:

- ✚ Tareas sin estructuración y no repetitivas.
- ✚ Puntos clave de la tarea que requieran destreza en la manipulación, especialmente cuando hablamos en una coordinación visual motriz.

- ✚ Puntos clave de la tarea que requieran reconocimiento de objeto o extremo cuidado en la situación.
- ✚ Eliminar la necesidad de personal altamente entrenado.

Puntos críticos a considerar en la telerobótica:

- ✚ Siempre es necesaria la intervención humana en alguna parte del proceso.
- ✚ Siempre se automatiza lo que es fácil, lo demás se deja a los humanos. (Don Norman).
- ✚ La interfaz con el usuario es absolutamente crítica.

Aplicaciones de la telerobótica:

Desde los primeros desarrollos de la teleoperación, la industria nuclear ha sido el principal consumidor de sistemas de teleoperación. Sin embargo, con el paso de los años se fue viendo su aplicabilidad a otros sectores, especialmente a los relacionados con las industrias de servicio.

Aplicaciones en el Espacio:

Las aplicaciones en el espacio tienen buenas razones para usar la teleoperación como técnica de manipulación remota, algunas de estas razones son:

a) Seguridad.

Todas las operaciones espaciales son de alto riesgo, que pueden llegar a ocasionar la muerte de astronautas.

b) Costo.

El equipo necesario para los pasajeros humanos es mucho más caro y pesado que un sistema de teleoperación.

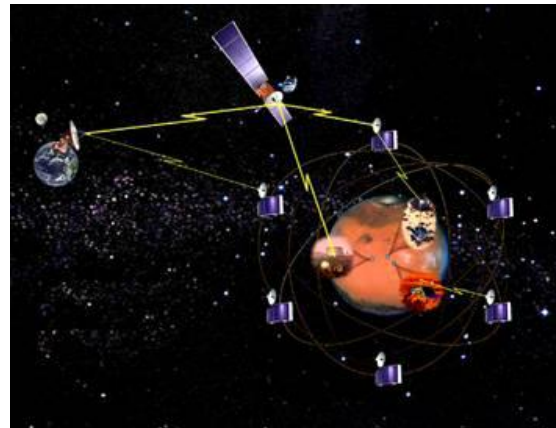


Figura 12. Comunicación con Robots en Marte

c) Tiempo.

Hay muchas misiones que duran muchos años para lograr los objetivos, obligando que sean sin tripulación humana.

Estas aplicaciones tienen un reto adicional, como lo es tener que trabajar con retardos temporales en las comunicaciones, lo que las hace especialmente problemáticas. Entre sus principales aplicaciones están: experimentación y exploración planetaria, mantenimiento y operación de satélites, construcción y mantenimiento de estaciones espaciales.

Aplicaciones en la industria nuclear:



Figura 13. Robot para aplicaciones nucleares.

Son las más numerosas. La utilidad del sistema de teleoperación radica en poder tratar de manipular sustancias radioactivas, así como moverse por entornos contaminados, sin peligro para el ser humano. Entre sus principales aplicaciones están: manipulación y experimentos con sustancias radioactivas, operación y mantenimiento de instalaciones nucleares (reactores, tuberías, instalaciones de elaboración de combustible nuclear, etc.), desmantelamiento, descontaminación de instalaciones y actuación en desastres nucleares.

Aplicaciones submarinas:

En este caso la mayoría de los manipuladores van montados en un vehículo submarino denominado R.O.V (*Remote Operated Vehicle*), que también va teleoperado. La utilidad de estos sistemas radica en poder acceder a ciertas zonas y profundidades donde le es imposible o peligroso a un buzo llegar. Entre las principales aplicaciones están: inspección, mantenimiento y construcción de instalaciones submarinas, minería submarina, e inspección de suelo marino.

La transmisión de los datos puede ser acústica, sin embargo la transmisión de los datos se ve afectada y llega a penas a los 10 Kbps. Ó por medio de un cable, sin embargo esto dificulta la dinámica del vehículo al aumentar el peso.

Aplicaciones militares:

Esta área provee muchas posibilidades para sistemas teleoperados, la mayoría de las tecnologías de teleoperación móvil han sido desarrolladas para aplicaciones militares, las tecnologías aquí usadas van desde sistemas de monitoreo remoto, hasta uso de los UAV (*Unmanned Air Vehicles*). Este tipo de vehículos tienen un campo de aplicación muy grande: vigilancia, adquisición de objetivos, detección de enemigos, reconocimientos de campo, etc. Los primeros sistemas de este tipo tenían lazo cerrado de control, el operador cerraba el lazo, hoy en día gracias a las nuevas tecnologías como el GPS (*Global Positioning System*) y el control supervisado los vehículos se vuelven más rápidos e inteligentes.



Figura 14. UAV Raven. Exploración militar.

Otra área de aplicación son los vehículos terrestres llamados UGV (*Unmanned Ground Vehicle*), dotados con tecnologías diferentes, incluidos sistemas de comunicación más amplios y rápidos gracias al uso de radiofrecuencias.

Aplicaciones médicas:

Recientemente se han fortalecido de forma importante la aplicación de tecnologías en el sector médico. Desde los desarrollos de prótesis o dispositivos de asistencia para discapacitados hasta la novedosa telecirugía. También encontramos la manipulación de microorganismos, microrobots y tecnologías que cada día evolucionan.

Otras aplicaciones:

La teleoperación también ha entrado con fuerza a la industria de la construcción, minería, mantenimiento de líneas de tensión, mantenimiento de instalaciones, intervención en desastres naturales y entretenimiento.

La teleprogramación, teleoperación y telemonitorización cada vez se aplica mas a la industria, en especial la automoción ya que la mayoría de sus procesos son líneas robotizadas, los sistemas de control cada día avanzan y con ello los nuevos paradigmas como la cooperación entre dichos sistemas y el ser humano, mostrando la amplia gama de aplicaciones en este ámbito de conocimiento.

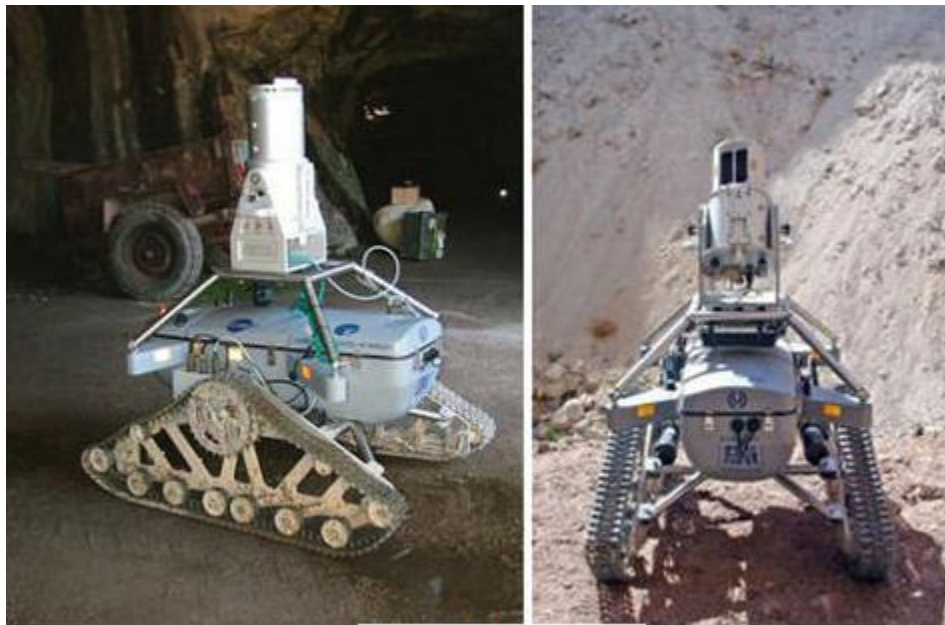


Figura 15. Robot para minas.

Teleoperación de manipuladores antropomórficos.

Un teleoperador o telerobot antropomórfico tiene una forma humana, que adquiere señales de su ambiente con algún dispositivo que asemeje a los ojos, manipule objetos mecánicamente con algún dispositivo que asemeje a las manos, los brazos y/o se mueva en muchas direcciones haciendo remembranza al cuerpo humano en sus habilidades motrices. El operador humano esta usualmente ubicado de una manera remota de él. Esto

significa que el teleoperador o telerobot antropomórfico debe proveer al operador humano una imagen remota del cuerpo o un alter ego físico. (Ver figura 16).

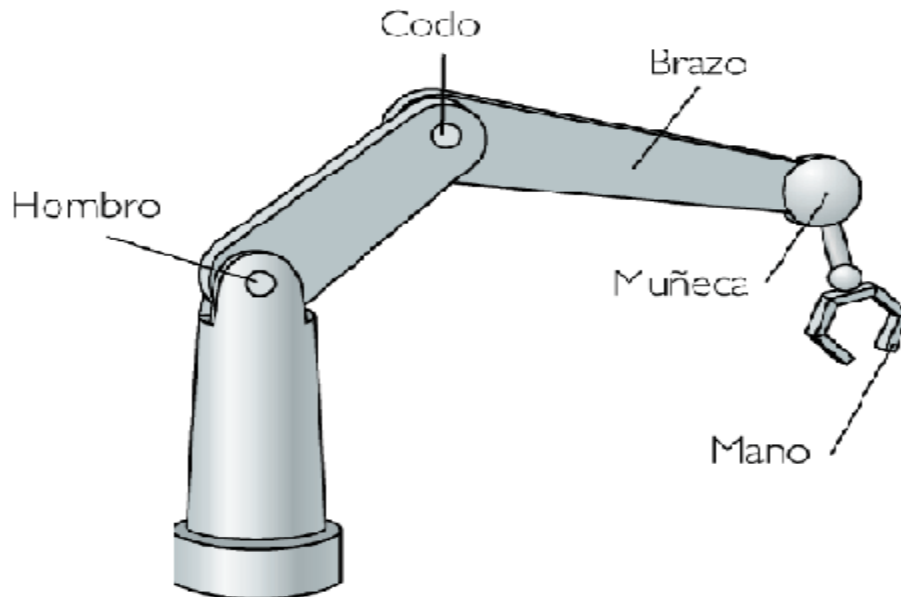


Figura 16. Manipulador Antropomórfico

Arquitecturas de teleoperación de manipuladores antropomórficos

Las diversas arquitecturas de control de teleoperación se diferencian básicamente por la información que se intercambia entre el maestro y el esclavo además del tipo de sensores involucrados que se requieren para el proceso. En función de la información intercambiada entre el maestro y el esclavo pueden clasificarse en las siguientes categorías:

1. Esquema Posición -Posición: La posición del esclavo se determina a partir de la del maestro y viceversa. No hay necesidad de sensores de fuerza.
2. Esquema Fuerza-Posición: La posición del esclavo se determina por seguimiento del robot maestro y las fuerzas que aparecen sobre el esclavo se miden y se generan en el maestro mediante sus motores. Solo se requiere medida de fuerzas en el esclavo.

3. Esquema Fuerza-Fuerza: Las trayectorias del maestro y el esclavo se determinan a partir de las lecturas de fuerza de ambos. También existe un control local de posición de ambos robots.
4. Esquema de Cuatro Canales: En este caso hay intercambio de información tanto en posición como en fuerza.

Las arquitecturas de teleoperación permiten diferentes grados de telepresencia que para el operador son fácilmente comparables y evaluables, afectando principalmente la estabilidad y transparencia del sistema.

La estabilidad es un concepto común y es una condición indispensable para este tipo de sistemas, ya que no sería aceptable que durante el manejo del sistema por parte del operador algún robot comenzara a sacudirse. El ruido eléctrico y los retrasos en señales de comunicación provocan inestabilidad, el ruido es un síntoma común en cualquier sistema de control real, pero los retrasos solo se encuentran cuando el robot se encuentra muy alejado del control.

El concepto transparencia es mucho más específico del concepto de la teleoperación. Para que las labores realizadas mediante el esclavo sean precisas no es suficiente un control de la posición del esclavo sino que además es necesario que el operador sea capaz de percibir las fuerzas que aparecen sobre el robot remoto durante la teleoperación, haciendo referencia a lo que se conoce como interfaces hápticas.



Figura 17. Dispositivos Hápticos

Es importante mencionar que los sistemas de manipulación teleoperados están formados por tres componentes básicos: el manipulador, el controlador y la fuente de energía.

El manipulador es quien realiza el trabajo físico, consiste en una secuencia de cuerpos rígidos llamados elementos, conectados mediante articulaciones prismáticas o de revolución, cada par articulación-elemento constituye un grado de libertad. El movimiento es proporcionado por los actuadores, los cuales permiten diversos ejes y alcances del manipulador. La energía con la que se alimentan los actuadores es convertida en energía mecánica.

Por su parte el controlador esta generalmente basado en un microprocesador siendo el corazón de la operación ya que permite el control de los movimientos del manipulador. Y la fuente de energía es la unidad que suministra energía al manipulador, esta puede ser neumática, eléctrica o hidráulica.

Teleoperación de un SCORBOT-ER 4u en el laboratorio de mecatrónica de la UNAM.

Trabajos previos.

Movimiento de un robot teleoperado a través de sistemas de libre distribución por internet.



Figura 18. SCORBOT-ER V Plus

Fue un proyecto realizado por el Ingeniero Carlos Enrique Contreras Vara con el fin de obtener su título como Ingeniero mecánico electricista, bajo la dirección del Dr. Víctor González Villela y con la codirección del Dr. Jesús Manuel Dorador González. En el año de 2002.

El objetivo principal de este trabajo de tesis era Mover un robot a través de Internet. Los objetivos particulares fueron utilizar plataformas libres para el desarrollo de los sistemas de control y utilizar un robot SCORBOT V (Ver figura 18) existente en el laboratorio de Sistemas de Manufactura Flexible en la Facultad de Ingeniería de la UNAM para el movimiento remoto por medio de conexiones de Internet, con un sistema desarrollado en plataforma y lenguajes de libre distribución.

El proyecto consistió en utilizar una computadora conectada a internet y auxiliada por los controladores del fabricante para ordenar las acciones que realizará el robot desde una computadora remota, empleando un programa en C para los comandos, mientras que el internet se puede manejar dentro del entorno de una página Web. Para dar información sobre el entorno de trabajo del robot se utilizó una cámara web que mostraba imágenes en tiempo real.

Los resultados del proyecto fueron satisfactorios al poder realizar programación y operaciones con el robot desde internet, sin embargo no se pudo resolver el problema del boqueo común del control ni tampoco los retardos que llegan a existir en el manejo de una cámara web.

Control de un brazo robot de cinco grados de libertad mediante un PLC.

Fue un proyecto realizado por los Ingenieros Gabriel Torres Miranda y Mario Alberto Matus Pérez con el fin de obtener su título como Ingenieros eléctricos electrónicos, bajo la dirección del M.F. Gabriel Hurtado Chong. En el año de 2010.

El proyecto consistió en crear una solución viable para el reemplazo de los controladores originales de los robots SCORBOT V, empleando un PLC y una interfaz HMI integrando LabVIEW y el módulo DSC. Además, hacer un comparativo de la implementación de las funciones básicas para la manipulación del Scrobot, entre un PLC y un PIC. Se creó una interfaz grafica la cual permitió al usuario manipular los movimientos del robot así como observar la simulación de los mismos. (Ver figura19).

El dispositivo funciono bajo los parámetros deseados, sin embargo las limitaciones son muchas y muy grandes. El control de lazo cerrado no fue el óptimo al utilizar solamente un control proporcional. Los movimientos se ven muy bruscos y faltos de capacidad para mover carga en el órgano terminal.

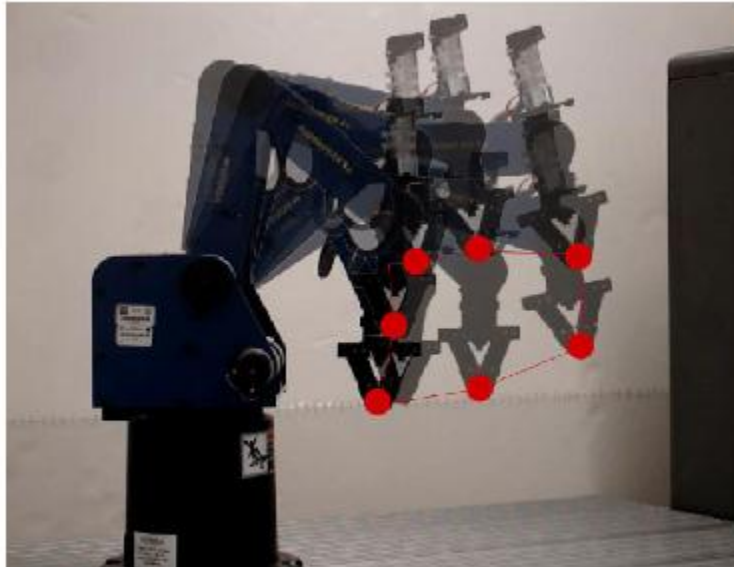


Figura 20. Trayectorias

Organización del trabajo:

Objetivo de proyecto:

Teleoperar un SCORBOT-ER 4u vía internet empleando un pantógrafo maestro de cadena abierta, capaz de reproducir los movimientos puntuales y trayectorias del maestro.

Alcances del proyecto:

- Desarrollar una interfaz entre el pantógrafo de cadena abierta y el SCORBOT-ER 4u que permita la reproducción geométrica de puntos.
- Realizar comunicación TCP/IP entre el dispositivo maestro y el esclavo, con el fin de enviar tanto comandos como video en internet.
- Desarrollar una interfaz que permita la teleprogramación y el telemonitoreo del dispositivo esclavo.
- Mostrar la flexibilidad de la interfaz al implementarla a otros manipuladores.

Metodología:

1. Investigar, aplicar y programar metodologías matemáticas para obtener la cinemática directa del pantógrafo e inversa del robot.
2. Empleando el controlador "*Controller-A*" generar el movimiento para llegar a una posición eliminando la mayor cantidad de problemas con la comunicación, velocidad, dinámica del robot y errores en la posición deseada. Primeramente en un solo eje, y después extrapolando la solución a los demás ejes.
3. Generar una interfaz para pruebas que nos permita ubicar de una manera intuitiva las posiciones de los ejes del robot.
4. Generar una interfaz que permita acoplar las ecuaciones cinemáticas con el movimiento de los ejes del robot.
5. Generar una interfaz que permita realizar la operación descrita en el punto cuatro pero ahora empleando internet para enviar los comandos e imágenes del proceso.
6. Generar una interfaz que permita la teleprogramación del robot.
7. Generar una interfaz que permita el telemonitoreo del robot.
8. Aplicar la interfaz anteriormente desarrollada en otro manipulador.

Justificación:

- ✚ Desarrollar tecnología de teleoperación para dispositivos que originalmente no están diseñados para este fin.
- ✚ Identificar las bondades y limitaciones al teleoperar el SCORBOT-ER 4u con el "*Controller-A*" del SCORBOT V.
- ✚ Desarrollar tecnología con fines tanto industriales como didácticos, de una forma más accesible al usuario.
- ✚ Utilizar en internet como un medio para generar la comunicación necesaria para la teleoperación en cualquier lugar del mundo.
- ✚ Mostrar la flexibilidad que se llega a presentar en las interfaces desarrolladas para problemas similares.
- ✚ Hacer investigación en el área de mecatrónica para la UNAM.

Impacto de proyecto:

- Generación de una interfaz que hace más amigable y preciso el trabajo a distancia con un SCORBOT-ER 4u para un ambiente didáctico dentro de la Facultad de Ingeniería de la UNAM.
- Desarrollo de sistemas teleoperados dentro de la Facultad de Ingeniería de la UNAM.
- Generar conocimiento sobre los sistemas teleoperados, teleprogramados y telemonitoreados.

En el siguiente capítulo de esta tesis se presentara la descripción y el análisis del proyecto. Se desglosaran todos los componentes de una manera comprensible para el lector. Además se anexaran diagramas para facilitar la comprensión más detallada del tema, y en caso dado se realizaran notas para ver los anexos.

Descripción y análisis del proyecto.

Parece ser que hoy en día, lo único que avanza es la tecnología.

Como se expuso en el capítulo anterior, el proyecto está enfocado en primer lugar a la teleoperación de un SCORBOT-ER 4u vía internet, y como objetivos secundarios la teleprogramación y el telemonitoreo de un proceso; además de implementar la aplicación diseñada a otro robot esclavo para mostrar la versatilidad del mismo.

Para realizar el proceso se diseñó una secuencia específica, en la cual tiene como inicio la ubicación de la posición del pantógrafo y termina en la reproducción de dicho movimiento por parte del esclavo.

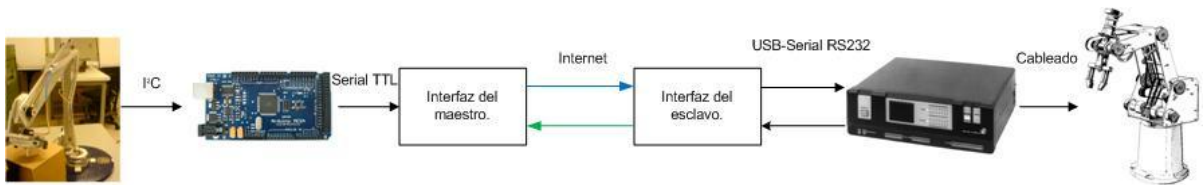


Figura1. Objetivo principal de la tesis.

Para generar la parte de teleoperación se diseñaron 2 interfaces independientes que son:

1. Interface del maestro:

Tiene como objetivo la comunicación directa con un pantógrafo instrumentado de cadena abierta

2. Interface del esclavo:

Tiene como objetivo la ejecución de las instrucciones recibidas desde el maestro.

Método de Denavit-Hartenberg.³

Matrices de transformación homogéneas por Denavit-Hartemberg:

Estableciendo un sistema coordinado en cada junta del manipulador, una matriz de transformación de 4x4 puede ser establecida para relacionar dos sistemas de coordenados sucesivos, empleando rotaciones y traslaciones de los mismos:

³ Lung-Wen Tsai. Robot analysis: the mechanics of serial and parallel manipulators. (1999). USA: John Wiley & Sons. Inc.

1. El sistema coordenado (i-1) es desplazado a través del eje Zi-1 una distancia di. Lo que permite que ambos orígenes coincidan.

$$T(z, d) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & di \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. El sistema de coordenadas (i-1) es rotado en el eje Zi-1 un ángulo Θ que provoca un alineamiento natural del eje Xi-1 con el eje Xi.

$$T(z, \theta) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. El sistema coordenado (i-1) es desplazado sobre el eje Xi una distancia ai.

$$T(x, a) = \begin{bmatrix} 1 & 0 & 0 & ai \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. El sistema de coordenadas (i-1) es rotado en el eje Xi un ángulo α_i que provoca la coincidencia completa de los dos ejes coordenados.

$$T(x, \alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Así al realizar la multiplicación de las cuatro transformaciones anteriormente citadas obtenemos:

$$T(z, d) T(z, \theta) T(x, a) T(x, \alpha) = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i) \sin(\theta_i) & \sin(\alpha_i) \sin(\theta_i) & a_i \cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i) \cos(\theta_i) & -\sin(\alpha_i) \cos(\theta_i) & a_i \sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz anteriormente citada es la matriz de transformación de Denavit-Hartenberg que relaciona un sistema de referencia final con uno inicial.

Modelado cinemático por Denavit-Hartenberg para un dispositivo de cadena abierta de 5 grados de libertad.

Para modelar el manipulador utilizando el modelo de Denavit-Hartenberg (en futuras referencias lo mencionaremos como D-H), necesitamos establecer los parámetros de los eslabonamientos y juntas cinemáticas para obtener las matrices de transformación. En el caso específico del pantógrafo maestro de cadena abierta (en futuras referencias lo llamaremos PAMCA), del robot SCORBOT-ER 4u y del robot "Black Hawk". Los parámetros serán los siguientes; las rotaciones que se realizan en las juntas cinemáticas las denotaremos con el símbolo Θ donde i puede ser del $[1, \dots, 5]$ y representa la junta. Por otra parte d_i representa desplazamientos con respecto al eje coordenado original y a_i representa el tamaño del eslabonamiento. Las figuras a continuación presentadas muestran la interpretación de lo anteriormente explicado:

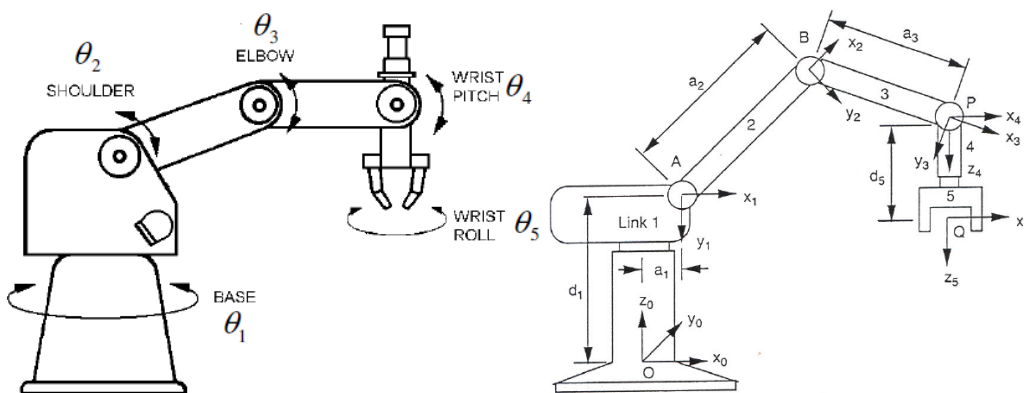


Figura 2-1. Rotaciones y dimensiones del Manipulador.

Junta	α_i	a_i	d_i	θ
1	$-\pi/2$	a_1	d_1	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3
4	$-\pi/2$	0	0	θ_4
5	0	0	d_5	θ_5

Tabla 2. Parámetros de los eslabonamientos para las ecuaciones de D-H.

Una vez identificados los parámetros procedemos a establecer las correspondientes matrices de transformación de D-H. Quedando como:

Transformación 0A_1

$$\begin{bmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & a_1 \cos(\theta_1) \\ \sin(\theta_1) & 1 & \cos(\theta_1) & a_1 \sin(\theta_1) \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformación 1A_2

$$\begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_2 \cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_2 \sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformación 2A_3

$$\begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & a_3 \cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & a_3 \sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformación 3A_4

$$\begin{bmatrix} \cos(\theta_4) & 0 & -\sin(\theta_4) & 0 \\ \sin(\theta_4) & 0 & \cos(\theta_4) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformación 4A_5

$$\begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ \sin(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finalmente al multiplicar las matrices anteriormente citadas obtenemos:

Transformación 0A_5

$$\begin{bmatrix} \cos(\theta_1) \cos(\theta_{234}) \cos(\theta_5) + \sin(\theta_1) \sin(\theta_5) & -\cos(\theta_1) \cos(\theta_{234}) \sin(\theta_5) + \sin(\theta_1) \cos(\theta_5) & -\cos(\theta_1) \sin(\theta_{234}) & c_{\theta_1}(a_1 + a_2 \cos \theta_2 + a_3 \cos \theta_{23} - d_5 \sin \theta_{234}) \\ \sin(\theta_1) \cos(\theta_{234}) \cos(\theta_5) - \cos(\theta_1) \sin(\theta_5) & -\sin(\theta_1) \cos(\theta_{234}) \sin(\theta_5) - \cos(\theta_1) \cos(\theta_5) & -\sin(\theta_1) \sin(\theta_{234}) & s_{\theta_1}(a_1 + a_2 \cos \theta_2 + a_3 \cos \theta_{23} - d_5 \sin \theta_{234}) \\ -\sin(\theta_{234}) \cos(\theta_5) & \sin(\theta_{234}) \sin(\theta_5) & -\cos(\theta_{234}) & d_1 - a_2 \sin(\theta_2) - a_3 \sin(\theta_{23}) - d_5 \cos(\theta_{234}) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde $\Theta_j = \Theta + \Theta_j$ y $\Theta_{jk} = \Theta + \Theta_j + \Theta_k$.

Esta matriz presenta la transformación desde el primer sistema de coordenadas hasta el final. Permitténdonos establecer la orientación y ubicación en el espacio del punto del efector final, estableciendo las siguientes ecuaciones:

Primera columna de la matriz, vector U, equivalente al eje x del sistema de referencia:

$$U_x = \cos(\theta_1) \cos(\theta_{234}) \cos(\theta_5) + \sin(\theta_1) \sin(\theta_5)$$

$$U_y = \sin(\theta_1) \cos(\theta_{234}) \cos(\theta_5) - \cos(\theta_1) \sin(\theta_5)$$

$$U_z = -\sin(\theta_{234}) \cos(\theta_5)$$

Segunda columna de la matriz, vector V, equivalente al eje y del sistema de referencia:

$$V_x = -\cos(\theta_1) \cos(\theta_{234}) \sin(\theta_5) + \sin(\theta_1) \cos(\theta_5)$$

$$V_y = -\sin(\theta_1) \cos(\theta_{234}) \sin(\theta_5) - \cos(\theta_1) \cos(\theta_5)$$

$$V_z = \sin(\theta_{234}) \sin(\theta_5)$$

Tercera columna de la matriz, vector W, equivalente al eje z del sistema de referencia:

$$W_x = -\cos(\theta_1) \sin(\theta_{234})$$

$$W_y = -\sin(\theta_1) \sin(\theta_{234})$$

$$W_z = -\cos(\theta_{234})$$

Cuarta columna de la matriz, vector q, ubicación espacial con respecto sistema de referencia:

$$q_x = \cos(\theta_1) (a_1 + a_2 \cos(\theta_2) + a_3 \cos(\theta_{23}) - d_5 \sin(\theta_{234}))$$

$$q_y = \sin(\theta_1) (a_1 + a_2 \cos(\theta_2) + a_3 \cos(\theta_{23}) - d_5 \sin(\theta_{234}))$$

$$q_z = d_1 - a_2 \sin(\theta_2) - a_3 \sin(\theta_{23}) - d_5 \cos(\theta_{234})$$

Ángulos de Euler.

Para la ubicación de la orientación de un cuerpo rígido los cosenos directores de una orientación contienen 9 parámetros. Entonces si definimos una rotación como un movimiento de 3 grados de libertad, entonces solamente 3 parámetros son suficientes para definir una orientación en el espacio.

Para la representación de Euler, tres rotaciones sucesivas son suficientes con respecto al eje de referencia o con respecto a un eje auxiliar, para determinar una orientación.

Por conveniencia utilizaremos tres matrices de rotación básicas, cuyo producto nos permitirá identificar la orientación final de cuerpo.

La primera matriz es cuando rotamos Θ grados solo sobre el eje Z.

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La segunda matriz es cuando rotamos Ψ grados solo sobre el eje X.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Psi) & -\sin(\Psi) \\ 0 & \sin(\Psi) & \cos(\Psi) \end{bmatrix}$$

La tercera matriz es cuando rotamos ϕ grados solo sobre el eje Y.

$$\begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix}$$

Las rotaciones anteriormente citadas las podemos visualizar en la figura que se muestra a continuación:

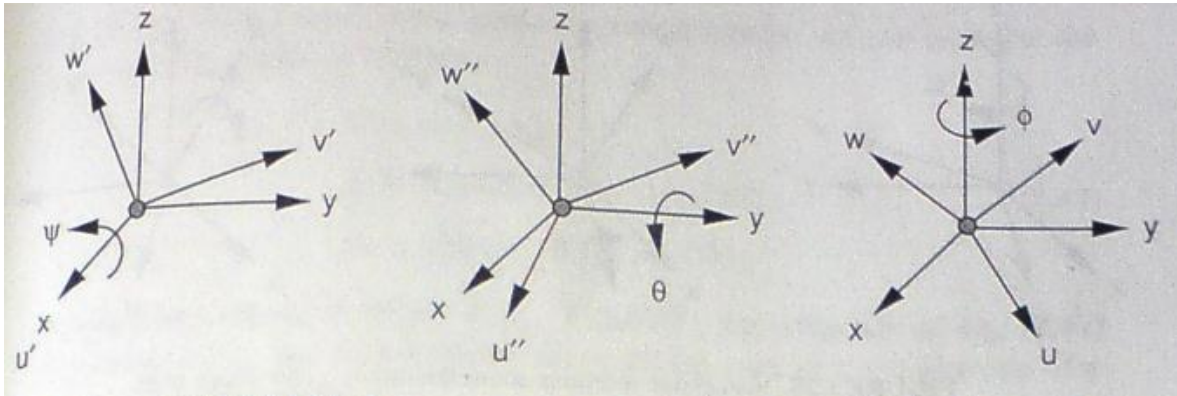


Figura3. Rotaciones con respecto a los ejes coordenados.

Representación de los ángulos de Euler w-v-w.

Una de las representaciones de los ángulos de Euler la podemos encontrar como una rotación ϕ en el eje w del sistema de referencia de apoyo (relacionado directamente con el eje z del sistema de coordenadas de referencia), seguido por una rotación Θ en el eje v (relacionado con el eje y del sistema de coordenadas de referencia), finalmente es seguido por una rotación Ψ de nuevo en el eje w.

Al realizar la correspondiente multiplicación de las matrices que representan las rotaciones anteriormente mencionadas, obtenemos la siguiente matriz con la información de las relaciones de rotación sucesivas:

$$A = \begin{bmatrix} \cos(\theta) \cos(\psi) \cos(\phi) - \sin(\phi) \sin(\Psi) & -\cos(\theta) \sin(\psi) \cos(\phi) - \sin(\phi) \cos(\Psi) & \cos(\phi) \sin(\theta) \\ \cos(\theta) \cos(\psi) \sin(\phi) + \cos(\phi) \sin(\Psi) & -\cos(\theta) \sin(\psi) \sin(\phi) + \cos(\phi) \cos(\Psi) & \sin(\phi) \sin(\theta) \\ -\sin(\theta) \cos(\Psi) & \sin(\theta) \sin(\Psi) & \cos(\theta) \end{bmatrix}$$

A partir de las rotaciones anteriormente citadas, ahora podemos obtener el valor de cada parámetro a partir de las siguientes ecuaciones:

$$\theta = \arccos(A_{33})$$

$$\phi = \arctan2\left(\frac{A_{23}}{\sin(\theta)}, \frac{A_{13}}{\sin(\theta)}\right)$$

$$\psi = \arctan\left(\frac{A_{32}}{\sin(\theta)}, -\frac{A_{31}}{\sin(\theta)}\right)$$

Con lo cual a partir de una matriz de orientación de 3 x 3 podemos inferir los ángulos de Euler correspondientes. Esta es la clave principal para generar la cinemática inversa del robot SCORBOT-ER 4u, ya que al enviar la posición en las coordenadas cartesianas ahora podemos anexar las orientaciones de la posición usando un robot de 5 grados de libertad.

Las ecuaciones anteriormente citadas nos permiten obtener la cinemática directa del manipulador de cinco grados de libertad empleando tanto Denavit-Hartenberg como ángulos de Euler. Con lo cual ya tenemos tanto la posición como la orientación del efector final.

Una vez teniendo todas las ecuaciones listas para realizar la cinemática directa se procedió a realizar una simulación numérica en la computadora empleando Matlab, para corroborar la funcionalidad del modelo propuesto.

```
function matrizZAF=cinematica_del_scorbot_5_DOI(a_1,a_2,a_3,d_1,d_5,th_1,th_2,th_3,th_4,th_5)
a_1=0.0;a_2=0.280;a_3=0.280;d_1=0.77;d_5=0.15;th_1=(0*(pi/180));th_2= 0.4734;th_3= 1.1873;th_4= 1.1051;th_5= -0.8912;%HOME DEL SCORBOT 4U
ZAU=[cos(th_1) 0 -sin(th_1) a_1*cos(th_1);sin(th_1) 0 cos(th_1) a_1*sin(th_1); 0 -1 0 d_1; 0 0 0 1];
UAD=[cos(th_2) -sin(th_2) 0 a_2*cos(th_2);sin(th_2) cos(th_2) 0 a_2*sin(th_2); 0 0 1 0; 0 0 0 1];
DAT=[cos(th_3) -sin(th_3) 0 a_3*cos(th_3);sin(th_3) cos(th_3) 0 a_3*sin(th_3); 0 0 1 0; 0 0 0 1];
TAC=[cos(th_4) 0 -sin(th_4) 0;sin(th_4) 0 cos(th_4) 0; 0 -1 0 0; 0 0 0 1];
CAF=[cos(th_5) -sin(th_5) 0 0;sin(th_5) cos(th_5) 0 0; 0 0 1 d_5; 0 0 0 1];
ZAF=ZAU*UAD*D*TAC*CAF;%Esta es la matriz de D-H
pitch=acos(ZAF(3,3));
rolla=atan2(ZAF(2,3)/sin(pitch),ZAF(1,3)/sin(pitch));
rollb=atan2(ZAF(3,2)/sin(pitch),-ZAF(3,1)/sin(pitch));
P_R=[rolla;rollb; pitch; 1];%Estos son los angulos de Euler
ZAFP_R=[ZAF,P_R];%Estas son las dos matrices en conjunto, dando la posicion con Denavit-Hartenberg y la orientacion con Euler.
matrizZAF=ZAFP_R;
```

Figura 4. Programa para simular el modelo en Matlab.

Una vez realizada la evaluación empleando Matlab se procedió a la programación de las ecuaciones en la interfaz del proyecto.

Cinemática Inversa por Denavit-Hartenberg:

Para resolver la cinemática inversa del manipulador es necesario tomar solo 5 de los 12 parámetros que obtenemos al generar la matriz de transformación ${}^0A^5$, estos cinco parámetros se deben a que el manipulador cuenta con solo 5 grados de libertad.

Para seleccionar los parámetros primero se descartan aquellos que no poseen la información necesaria para obtener todos los ángulos buscados, por ejemplo el vector W. Esto se debe a que el vector W es totalmente independiente del Roll. Una mejor selección

es el vector de posiciones (tomando los parámetros q_x y q_y) y el vector U . La cinemática inversa proviene directamente de la matriz de transformación ${}^0A_5 = {}^0A_1 {}^1A_2 {}^2A_3 {}^3A_4 {}^4A_5$.

Para que tengamos un proceso más directo, premultiplicamos por $({}^0A_1)^{-1} {}^0A_5 = {}^1A_2 {}^2A_3 {}^3A_4 {}^4A_5$.

Así en la primera columna de las matrices de transformación usadas obtendremos:

$$u_x c\theta_1 + u_y s\theta_1 = c\theta_{234} \theta_5$$

$$-u_z = s\theta_{234} c\theta_5$$

$$-u_x s\theta_1 + u_y c\theta_1 = -s\theta_5$$

De manera similar en la cuarta columna obtendremos:

$$q_x c\theta_1 + q_y s\theta_1 - a_1 = a_2 c\theta_2 + a_3 c\theta_{23} - d_5 s\theta_{234}$$

$$-q_2 + d_1 = a_2 s\theta_2 + a_3 s\theta_{23} + d_5 c\theta_{234}$$

$$-q_x s\theta_1 + q_y c\theta_1 = 0$$

Las ecuaciones anteriormente mostradas nos servirán para obtener el cálculo de los ángulos.

De la última ecuación salta a la vista de inmediato que:

$$\theta_1 = \tan^{-1} \frac{q_y}{q_x}$$

Donde $\theta_1 = \theta_1$ es una solución, pero también lo es $\theta_1 = \theta_1 + \pi$.

Una vez obteniendo el primer ángulo podemos obtener el quinto de la tercera ecuación de trabajo mostrada.

$$\theta_5 = \sin^{-1}(u_x s\theta_1 - u_y c\theta_1)$$

Donde $\Theta = \Theta$ es una solución, pero también lo es $\Theta = \Theta + \pi$.

Ahora con los ángulos 1 y 5, somos capaces de adquirir la suma de los ángulos 2,3 y 4. De la primera y segunda ecuación

$$\theta_{234} = \text{Atan } 2 \left[-u_z / c\theta_5, (u_x c\theta_1 + u_y s\theta_1) / c\theta_5 \right]$$

Ahora si reescribimos la ecuación cuarta y quinta de tal forma que:

$$a_2 c\theta_2 + a_3 c\theta_{23} = k_1$$

$$a_2 s\theta_2 + a_3 s\theta_{23} = k_2$$

Dónde

$$k_1 = q_x c\theta_1 + q_y s\theta_1 - a_1 + d_5 s\theta_{234}$$

$$k_2 = -q_2 + d_1 - d_5 \theta_{234}$$

Entonces podemos reescribir al utilizar las ecuaciones antes mencionadas como:

$$a_2^2 + a_3^2 + 2a_2 a_3 c\theta_3 = k_1^2 + k_2^2$$

De la cual despejamos Θ :

$$\theta_3 = \cos^{-1} \frac{k_1^2 + k_2^2 - a_2^2 - a_3^2}{2a_2 a_3}$$

Donde $\Theta = \Theta$ es una solución y también $\Theta = -\Theta$ es solución.

Ya con el ángulo 3, buscaremos obtener el ángulo 2 a partir de las ecuaciones que poseen las constantes:

$$(a_2 + a_3 c \theta_3) c \theta_2 - (a_3 s \theta_3) s \theta_2 = k_1.$$

$$(a_3 s \theta_3) c \theta_2 + (a_2 + a_3 c \theta_3) s \theta_2 = k_2$$

Reescribiéndolas como:

$$c \theta_2 = \frac{k_1 (a_2 + a_3 c \theta_3) + k_2 a_3 s \theta_3}{a_2^2 + a_3^2 + 2a_2 a_3 c \theta_3}$$

$$s \theta_2 = \frac{-k_1 a_3 s \theta_3 + k_2 (a_2 + a_3 c \theta_3)}{a_2^2 + a_3^2 + 2a_2 a_3 c \theta_3}$$

$$\theta_2 = A \tan 2(s \theta_2, c \theta_2)$$

Con lo cual obtenemos una única Θ , a partir de Θ , Θ y Θ .

Finalmente para obtener el cuarto ángulo basta hacer una simple resta:

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3.$$

Con esto concluimos que podemos obtener hasta 8 soluciones cinemáticas diferentes para un punto.

Al igual que en la cinemática directa, se procedió a evaluar el modelo en Matlab para ver los resultados en una simulación numérica. Y con ello tener una mejor perspectiva de los que se puede esperar en el momento de la ejecución de la interface.

```

function Thetas=cinematica_inversa_corregido()
q_x= 0.1582;q_y= 0.1582;q_z= -0.0103;pitch=2.5623;rolla=-2.3562;rollb=0.1808;a_1=0;a_2=0.280;a_3=0.280;d_5=0.15;d_1=0.077;
u_x=cos(rolla)*cos(pitch)*cos(rollb)-sin(rolla)*sin(rollb);%Reconstruimos los valores de el vector U
u_y=sin(rolla)*cos(pitch)*cos(rollb)+cos(rolla)*sin(rollb);
u_z=-sin(pitch)*cos(rollb);
u_xyz=[u_x;u_y;u_z;0;0];
Th_1=atan(q_y/q_x);
Th_5=asin(u_x*sin(Th_1)-u_y*cos(Th_1));
Th_234=atan2(-u_z/cos(Th_5), (u_x*cos(Th_1)+u_y*sin(Th_1))/cos(Th_5));
k1=q_x*cos(Th_1)+q_y*sin(Th_1)-a_1+d_5*sin(Th_234);
k2=-q_z+d_1-d_5*cos(Th_234);
Th_3=acos((k1^2+k2^2-a_2^2-a_3^2)/(2*a_2*a_3));
Th_2c=(k1*(a_2+a_3*cos(Th_3))+k2*a_3*sin(Th_3))/(a_2^2+a_3^2+2*a_2*a_3*cos(Th_3));
Th_2s=(-k1*a_3*sin(Th_3)+k2*(a_2+a_3*cos(Th_3)))/(a_2^2+a_3^2+2*a_2*a_3*cos(Th_3));
Th_2=atan2(Th_2s,Th_2c);
Th_4=Th_234-Th_2-Th_3;
Th_r=[Th_1 ;Th_2 ;Th_3 ;Th_4; Th_5];
Th_rg=Th_r*1;
Thetas=[Th_rg u_xyz];

```

Figura 5. Programa para simular el modelo en Matlab.

Pantógrafo Maestro de Cadena Abierta (PAMCA).

El pantógrafo maestro de cadena abierta es un dispositivo articulado antropomórfico de 5 grados de libertad que permite ubicar espacialmente puntos. En su órgano terminal nos permite visualizar la orientación y ubicación física de los puntos, e inclusive generando secuencias de los mismos obtener trayectorias bien definidas.

Tiene un parecido morfológicamente hablando con un brazo humano, al igual que los brazos manipuladores. Lo que permite tener una mejor visualización de las posibles posiciones que refleja el robot esclavo en un proceso de teleoperación.

Este dispositivo fue planeado, diseñado, fabricado y evaluado por el M.I Octavio Díaz Hernández con fines de investigación para la tesis doctoral del mismo.

Este dispositivo esta hecho de Aluminio comercial. En cada articulación cuenta con un *encoder* incremental que permite identificar la posición en la que se encuentre y los desplazamientos que se lleguen a presentar. Estos *encoders* leen con tarjetas de propósito



Figura 6. PAMCA.

especifico que a su vez se comunican empleando una comunicación vía *buffer* de datos del tipo I2C con un “Arduino Mega 2560”, que a su vez permite la comunicación con otros dispositivos utilizando comunicación serial TTL-USB. Dichas interfaces también fueron realizadas por el MI Octavio Díaz Hernández. Debido al carácter de investigación y desarrollo la información específica de estos dispositivos no será publicada en trabajos posteriores con el fin de proteger la propiedad intelectual del desarrollador.

Para realizar los cambios en la orientación del órgano terminal, basta con tomar la punta del pantógrafo y desplazarla de la manera deseada para obtener el resultado buscado. Lo único que se debe de tomar en cuenta es la ubicación física con respecto al eje de referencia, con el fin de tener una idea de los resultados que se tendrán en el esclavo.

Una vez establecido nuestra herramienta principal de la interfaz del maestro, establezcamos la teoría del análisis de las geometrías que representará, empleando las ecuaciones de Denavit-Hartenberg.

Una de las partes fundamentales del proyecto es la comunicación que se realiza entre la computadora donde se encuentra la interfaz del maestro y el pantógrafo. Esto se logra empleando comunicación serial con el “Arduino Mega 2560” donde se encuentra lista la información de los *encoders* para su procesamiento.

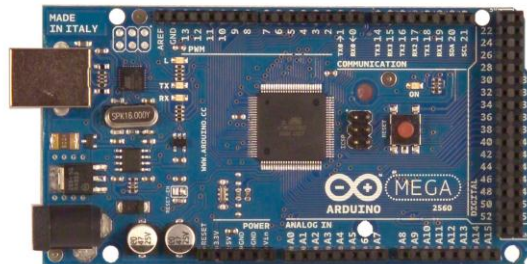


Figura7. Arduino Mega 2560 usado en el proyecto.

Comunicación Serial

La comunicación serial es un protocolo muy común para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. La mayoría

de las computadoras de escritorio incluyen dos puertos seriales RS-232. La comunicación serial es también un protocolo común utilizado por varios dispositivos para instrumentación; existen varios dispositivos compatibles con GPIB que incluyen un puerto RS-232. Además, la comunicación serial puede ser utilizada para adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo.

El concepto de comunicación serial es sencillo. El puerto serial envía y recibe bytes de información un bit a la vez. Aun y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. Por ejemplo, la especificación *IEEE 488* para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre cualesquier dos dispositivos; por el otro lado, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: (1) Tierra (o referencia), (2) Transmitir, (3) Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por un línea mientras se reciben datos por otra. Existen otras líneas disponibles para realizar *handshaking*, o intercambio de pulsos de sincronización, pero no son requeridas. Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

- a. Velocidad de transmisión (*baud rate*): Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bauds*). Por ejemplo, 300 baudios representa 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz. Con velocidades más altas se reduce la distancia máxima posible entre los

dispositivos. Las altas velocidades se utilizan cuando los dispositivos se encuentran uno junto al otro, como es el caso de dispositivos GPIB.

- b. Bits de datos: Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usará para referirse a todos los casos.
- c. Bits de parada: Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo la transmisión será más lenta.
- d. Paridad: Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para

tener 3 bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizado

Robot SCORBOT-ER 4u.

El robot SCORBOT-ER 4u fue diseñado y desarrollado para emular las condiciones de un robot industrial. Tiene fines didácticos y está pensado de manera que los estudiantes sean capaces de visualizar los mecanismos de movimiento internos.

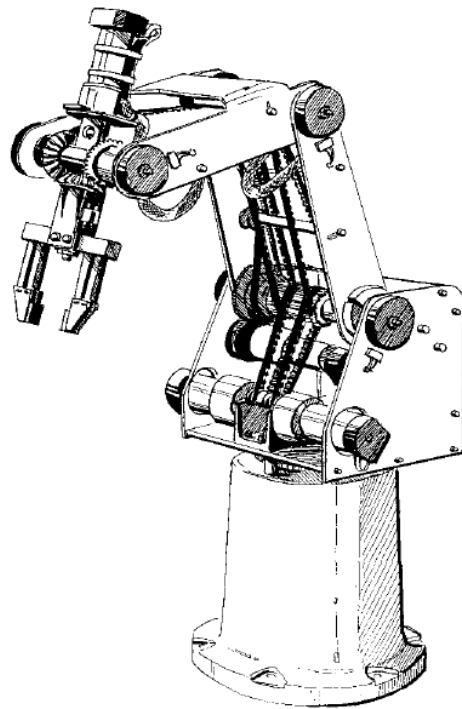


Figura8. SCORBOT-ER 4u

Este robot es del tipo verticalmente articulado, con cinco juntas de revolución. Con el *gripper* como órgano terminal tiene seis grados de libertad. Lo que permite tener un área de trabajo flexible para posicionar de diferentes formas el robot.

Las articulaciones son accionadas por motores, los cuales están acoplados al eje normalmente empleando bandas dentadas o engranes. La mayor parte de los motores están montados en la base del robot lo que reduce el peso en las partes superiores del mismo. La coordinación del movimiento de los ejes y por consiguiente de los motores está a cargo del "Controller A", que explicaremos mas a detalle mas adelante.

Numero de eje	Articulacion	Movimiento	Numero de motor
1	Base	Rotacion del cuerpo	1
2	Hombro	Sube y baja brazo	2
3	Codo	Sube y baja antebrazo	3
4	Pitch	Sube y baja pinza	4+5
5	Roll	Rota pinza	4+5
6	Pinza	Apertura/Cierre	6

Tabla9. Grados de libertad del Scorbot

Espacio de trabajo:

El tamaño de los eslabonamientos y el ángulo máximo que pueden realizar los ejes de cada motor determinan el espacio de trabajo.

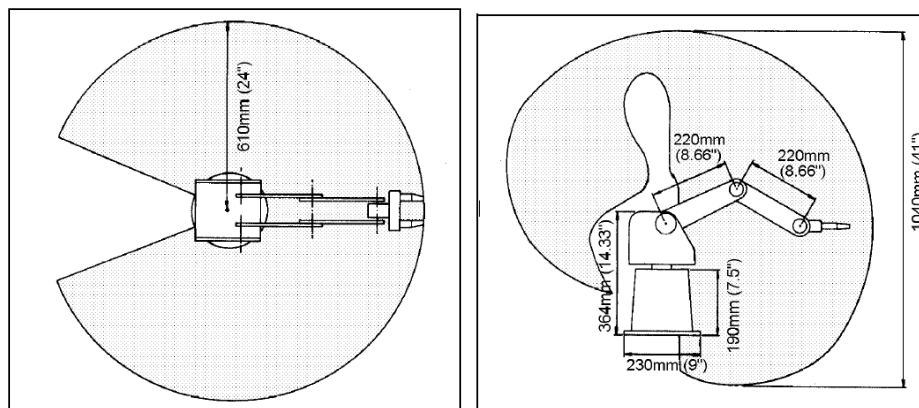


Figura10. Espacio de trabajo del SCORBOT-ER 4u

Motores:

Los motores que accionan el movimiento del robot son de corriente directa. Los cuales trabajan a 15V, y tiene un *encoder* de cuadratura en su parte posterior con la cual se retroalimenta el controlador y un reductor en la parte frontal.

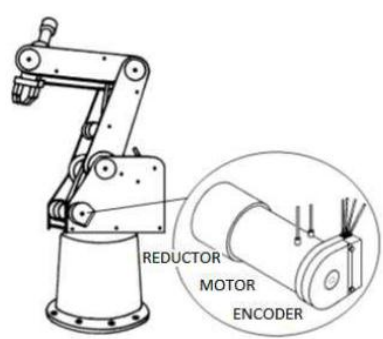


Figura11. Ejemplo de motor usado para accionar el movimiento del robot.

Transmisiones:

Diversas transmisiones son usadas para ejecutar el movimiento del robot:

1. La base y el hombro es movido por transmisiones de engranes dentados.
2. Bandas dentadas y poleas mueven el codo.
3. Bandas dentadas y poleas mueven el roll y el pitch con un diferencial.
4. Un tornillo sin fin abre o cierra el *gripper*.

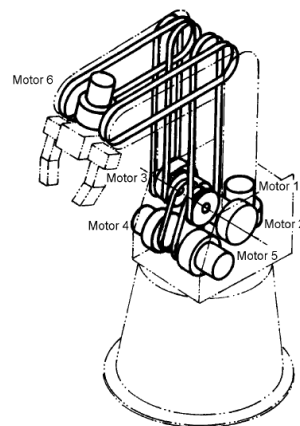


Figura12. Transmisiones para impulsar los ejes.

Diferencial en el pitch-roll:

Tanto el pitch como el roll son impulsados por los mismos motores, de la siguiente manera. Si se requiere realizar un movimiento únicamente con el pitch los motores 4 y 5 giraran en el mismo sentido. Pero si se requiere realizar un movimiento en roll, los motores giraran en sentido contrario, es decir mientras uno gira al lado positivo el otro girara al lado negativo. Con lo cual el arreglo de engranes permitirá el giro requerido.

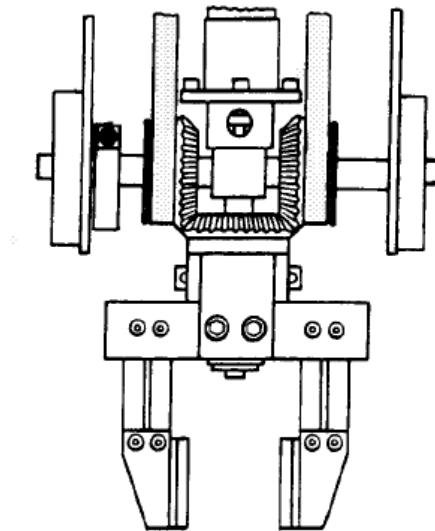


Figura13. Diferencial del pitch y roll.

Microswitches:

Son cinco los que contiene en su totalidad el robot. Tienen la responsabilidad de ubicar la carrera del robot y con ello establecer el HOME.

Datos técnicos relevantes:

Dentro de los elementos a considerar en el robot encontramos que es capaz de mover objetos con un peso máximo de 1kg. Sin ningún problema, gracias a las características de los motores y sus reducciones. A continuación se enumeran los datos técnicos más relevantes del robot:

Giro limite por articulación:

Eje	Angulo limite
Base	310°
Hombro	+130°/-35°
Codo	+/-130°
Pitch	+/-130°
Roll	+/-570°

Tabla14. Limites de las articulaciones.

Cuando se encuentra a máxima carga cada motor consume alrededor de 70 Watts. En conjunto el robot tiene una precisión de +/-0.18 mm y puede tener una velocidad máxima de 600 mm/seg por eje.⁴ (Para más información sobre el SCORBOT-ER 4u ver anexo 4).

Controller-A

Este es el controlador del Robot SCORBOT-ER 4u, básicamente cumple las funciones de conexión con la interfaz de la computadora, maneja la potencia del robot, previene sobrecargas, adquiere las señales de los sensores internos del robot, guarda programas, administra puntos grabados, genera controles para las trayectorias y procesamiento de comandos ACL. En general el *Controller-A* es una computadora de propósito específico con la finalidad de administrar todas las acciones del robot.



Figura15. Controller A

⁴ Manual de Usuario del SCORBOT-ER 4u, página 8.

Las características técnicas específicas del control son:⁵

✚ Tipo de control:

- PID (Proporcional, integral y diferencial)
- PWM (Modulación de ancho de pulso)

✚ Control sobre ejes:

- PWM de 20Khz.

✚ Control para trayectorias:

- Paraboloidal.
- Trapezoidal.
- Lazo abierto.
 - Estos controles tienen un ciclo de control cada 10ms. El controlador maneja los parámetros del PID y las aceleraciones.

✚ Control de velocidad:

- Por velocidad. (Manejado como porcentajes donde 100% son 600mm/s)
- Por tiempo de trayectoria.

✚ Parámetros de control:

- Servo control.
- Perfil de velocidad.
- Error de posición de eje.
- Operación del *gripper*.
- Protección ante carga térmica, impacto o límites alcanzados.
- Home.
- Interface de *encoder*.
- Cálculo cartesiano.

✚ Fuente de poder:

- 100/110/120 V AC. 50/60Hz. 500W máximo.

✚ Fuentes internas de poder:

⁵ Especificaciones del Controller-A, pagina 3-2 del manual de operación del SCORBOT V plus.

- Motores 12 VDC, 18 A.
- ✚ Temperatura ambiental de trabajo:
 - 2C°-40C°.
- ✚ CPU:
 - Motorola 58010.
- ✚ EPROM:
 - 384 Kb.
- ✚ RAM:
 - Sistema: 64 Kb.
 - Usuario: 128 Kb.
- ✚ Comunicación:
 - Serial RS232.
- ✚ Lenguaje de programación:
 - ACL. *Advanced Control Language*. (Para más información ver anexo 1).
- ✚ Grabar posiciones:
 - Juntas.
 - Cartesiano.
 - Absoluto.
 - Relativo.
- ✚ Sistemas coordinados:
 - Coordenadas XYZ.
 - Juntas coordinadas.

Para mayor información con respecto al *Controller-A*. Ver anexo 3.

Modos de programación para el controlador:

Para programar el controlador mediante el uso de comandos ACL tenemos dos opciones:

1. DIRECT: Son comandos que se ejecutan tan pronto como entran al controlador, son tecleados manualmente desde una terminal como un teclado.

2. EDIT: Son comandos que son ejecutados durante la programación o las rutinas en las que son usados.

Para cambiar entre un modo y otro es necesario enviar con el teclado el caracter “~”, y se realizara el relevo automático.

Servo-Robot “Black Hawk”.

Es un robot articulado verticalmente de 5 grados de libertad más el *gripper*. Tiene la característica que todos sus ejes son impulsados por servomotores de aeromodelismo. Lo que impacta directamente en el espacio de trabajo ya que se ve limitado a manejar tan solo de 0° a 180° grados por junta cinemática.

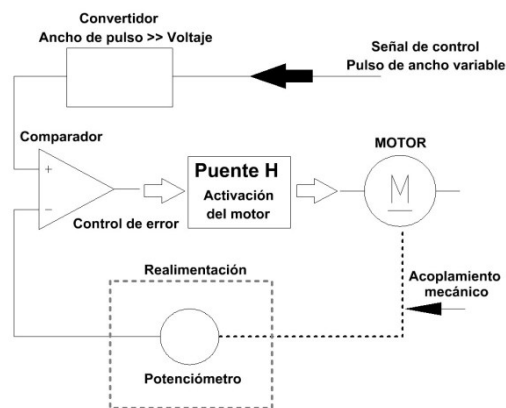


Figura16. Control interno de un servomotor de aeromodelismo.

La ventaja principal radica en su facilidad de control, ya que los servomotores cuentan con una etapa de control interno dependientes de una señal PWM que indica la posición que deben cumplir. Lo cual facilita ampliamente el control el lazo abierto de este dispositivo.

Sin embargo, el robot también cuenta con desventajas muy marcadas como es la configuración de la transmisión de potencia a los ejes. Esto se debe principalmente a que cada motor se encuentra unido a cada eje con tornillos. Es decir los motores pasan a ser parte de la estructura del robot, lo que



Figura 17. Black Hawk.

Es decir los motores pasan a ser parte de la estructura del robot, lo que

impacta directamente en la dinámica del mismo. Además otro elemento que también influye directamente en la dinámica del motor es la falta de cojinetes o valeros en las articulaciones. Lo que provoca fricciones directamente entre materiales.

Comunicación Serial RS232.

La norma RS232 resuelve tres aspectos en la comunicación que se establece entre el DTE, Equipo Terminal de Datos, como un PC y el DCE, equipo para la comunicación de datos, por ejemplo un ratón:

1. Características eléctricas de la señal: Se establece que la longitud máxima entre el DTE y el DCE no debe ser superior a los 15 metros y la velocidad máxima de transmisión es de 20.000 bps. Los niveles lógicos no son compatibles TTL, considerando:
 - a. 1 lógico entre -3V y -15V
 - b. 0 lógico entre +3V y +15V
2. Características mecánicas de los conectores: Se utiliza un conector 25 patillas, DB 25, o de 9 patillas, DB 9, donde el conector macho identifica al DTE y el conector hembra al DCE.
3. Descripción funcional de las señales usadas: Las señales están básicamente divididas en dos grupos:
 - a. Señales primarias, que son normalmente utilizadas para la transferencias de datos .
 - b. Señales secundarias, utilizadas para el control de la información que será transferida.

La norma RS232 está definida tanto para la transmisión síncrona como para la asíncrona, pero cuando se utiliza esta última, sólo un conjunto de terminales del DB25, es utilizado. La velocidad está estandarizada según la norma RS 232C en baudios: 75, 110, 150, 300, 600 , 1200 , 2400 ,4800 ,9600, y 19200.

Ver anexo 2 para más información sobre los conectores para comunicación RS232.

Internet y sus protocolos de comunicación

Internet

Durante los inicios de la informática se desarrollaron equipos. Una vez que éstos fueron capaces de funcionar solos, algunas personas tuvieron la idea de conectarlos para poder intercambiar datos; éste es el concepto de una red. Por lo tanto, no sólo se debían desarrollar conexiones físicas entre los equipos para que la información pudiera circular, sino que también se debía desarrollar un lenguaje de comunicación para que pudiera haber un verdadero intercambio. Se decidió que este lenguaje se denominara protocolo. En Internet se utilizan diversos protocolos, que son parte de una serie denominados TCP/IP. El TCP/IP se basa en la identificación de cada equipo con una dirección denominada dirección IP, que posibilita la transmisión de datos a la dirección correcta. Después estas direcciones se relacionaron con nombres de dominios para que pudieran recordarse con más facilidad.

Se desarrollaron redes de diferentes tipos en los cuatro rincones del planeta. Entonces, algunas personas decidieron conectar estas redes. Se desarrollaron protocolos para permitir que todas estas redes se pudieran comunicar y formar una red de redes. Poco a poco se fue formando una "telaraña" (*Web*) gigante, en la que la red más grande contenía a las demás redes. Esto se denominó Internet. Existen diferentes protocolos en Internet que permiten llevar a cabo diferentes acciones:

- IRC: chat en directo
- HTTP: navegar por páginas web
- FTP: transferir archivos

A cada acción se le asigna un número que se envía durante la comunicación (la transmisión se lleva a cabo mediante pequeños paquetes de información). Por lo tanto, es posible conocer con qué programa se relaciona cada pequeño paquete, por ejemplo:

- Los paquetes HTTP llegan al puerto 80 y se transmiten al navegador de Internet que solicitó la página.

- Los paquetes IRC llegan al puerto 6667 (generalmente ubicado cerca de 7000) y se transmiten a un cliente IRC.

Conexión a Internet

La tarjeta de interfaz de red es la parte del equipo que permite la conexión a una red a través de líneas especialmente proporcionadas para el envío de información digital. El módem permite la conexión a una red mediante líneas telefónicas, que originalmente no se proporcionaron para esto pero que todavía son el medio de comunicación más utilizado.

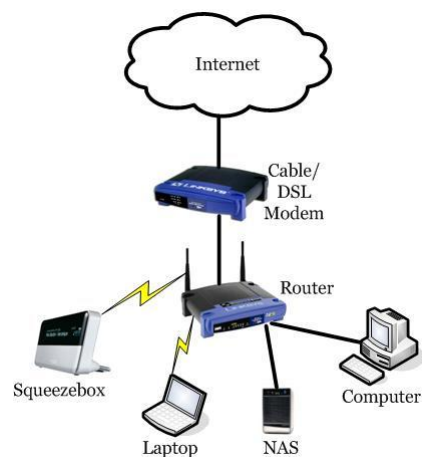


Figura 18. Conexión a Internet

Se asocia una dirección IP con la tarjeta de interfaz de red, lo cual permite identificar el equipo en la red.

La conexión con un módem es totalmente diferente. Éste permite establecer una comunicación entre dos equipos mediante una línea telefónica. Sin embargo, se puede acceder a una red mediante la conexión a un equipo conectado ("de un lado") a una o varias líneas telefónicas y a una red con una tarjeta de interfaz de red.

Este equipo generalmente pertenece al proveedor de servicios de Internet (ISP, *Internet Service Provider*). Cuando le conecta a través del intermediario, toma prestada la dirección IP que el equipo mantendrá durante la conexión. Cada vez que se conecta, arbitrariamente asigna una de estas direcciones IP libres que posee. Si puede brindar la misma dirección IP para cada conexión, entonces se denomina una "dirección IP fija".

Protocolo

Un protocolo es un método estándar que permite la comunicación entre procesos es decir, es un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red. Existen diversos protocolos de acuerdo a cómo se espera que sea la comunicación. Algunos protocolos, por ejemplo, se especializarán en el intercambio de archivos (FTP); otros pueden utilizarse simplemente para administrar el estado de la transmisión y los errores ,etc.

En Internet, los protocolos utilizados pertenecen a una sucesión de protocolos o a un conjunto de protocolos relacionados entre sí. Este conjunto de protocolos se denomina TCP/IP.

Entre otros, contiene los siguientes protocolos:

- HTTP:

Desde 1990, el protocolo HTTP (Protocolo de transferencia de hipertexto) es el protocolo más utilizado en Internet. El propósito del protocolo HTTP es permitir la transferencia de archivos (principalmente, en formato HTML) entre un navegador (el cliente) y un servidor web (denominado, entre otros, httpd en equipos UNIX) localizado mediante una cadena de caracteres denominada dirección URL.
- FTP:

El protocolo FTP (Protocolo de transferencia de archivos) es, como su nombre lo indica, un protocolo para transferir archivos.
- ARP:

El protocolo ARP tiene un papel clave entre los protocolos de capa de Internet relacionados con el protocolo TCP/IP, ya que permite que se conozca la dirección física de una tarjeta de interfaz de red correspondiente a una dirección IP. Por eso se llama Protocolo de Resolución de Dirección (en inglés ARP significa *Address Resolution Protocol*).
- ICMP
ICMP (Protocolo de mensajes de control de Internet) es un protocolo que permite administrar información relacionada con errores de los equipos en red.

- IP

Los equipos comunican a través de Internet mediante el protocolo IP (Protocolo de Internet). Este protocolo utiliza direcciones numéricas denominadas direcciones IP compuestas por cuatro números enteros (4 bytes) entre 0 y 255, y escritos en el formato xxx.xxx.xxx.xxx. Los equipos de una red utilizan estas direcciones para comunicarse, de manera que cada equipo de la red tiene una dirección IP exclusiva.

- TCP

TCP (que significa Protocolo de Control de Transmisión) en el nivel de aplicación, posibilita la administración de datos que vienen del nivel más bajo del modelo, o van hacia él.

- UDP

El protocolo UDP (Protocolo de datagrama de usuario) es un protocolo no orientado a conexión de la capa de transporte del modelo TCP/IP. Este protocolo es muy simple ya que no proporciona detección de errores.

- SMTP

El protocolo SMTP (Protocolo simple de transferencia de correo) es el protocolo estándar que permite la transferencia de correo de un servidor a otro mediante una conexión punto a punto.

- Telnet

El protocolo Telnet es un protocolo de Internet estándar que permite conectar terminales y aplicaciones en Internet. El protocolo proporciona reglas básicas que permiten vincular a un cliente (sistema compuesto de una pantalla y un teclado) con un intérprete de comandos (del lado del servidor).

- NNTP

El Protocolo de transferencia de noticias a través de la red (NNTP) es un protocolo TCP/IP basado en cadenas de texto que se envían de forma bidireccional a través de canales TCP ASCII de siete bits, se denomina habitualmente Protocolo de

noticias de Internet, porque contiene las reglas utilizadas para transportar artículos de noticias de un equipo a otro.

Generalmente los protocolos se clasifican en dos categorías según el nivel de control de datos requerido:

- Protocolos orientados a conexión: estos protocolos controlan la transmisión de datos durante una comunicación establecida entre dos máquinas. En tal esquema, el equipo receptor envía acuses de recepción durante la comunicación, por lo cual el equipo remitente es responsable de la validez de los datos que está enviando. Los datos se envían entonces como flujo de datos. TCP es un protocolo orientado a conexión.
- Protocolos no orientados a conexión: éste es un método de comunicación en el cual el equipo remitente envía datos sin avisarle al equipo receptor, y éste recibe los datos sin enviar una notificación de recepción al remitente. Los datos se envían entonces como bloques (datagramas). UDP es un protocolo no orientado a conexión.

Implementación de protocolos:

Un protocolo define únicamente cómo deben comunicar los equipos, es decir, el formato y la secuencia de datos que van a intercambiar. Por el contrario, un protocolo no define cómo se programa el software para que sea compatible con el protocolo. Esto se denomina implementación o la conversión de un protocolo a un lenguaje de programación.

Dirección IP

El organismo a cargo de asignar direcciones públicas de IP, es decir, direcciones IP para los equipos conectados directamente a la red pública de Internet, es el ICANN (*Internet Corporation for Assigned Names and Numbers*) que reemplaza el IANA desde 1998 (*Internet Assigned Numbers Agency*).

Una dirección IP es una dirección de 32 bits, escrita generalmente con el formato de 4 números enteros separados por puntos. Una dirección IP tiene dos partes diferenciadas:

- Los números de la izquierda indican la red y se les denomina netID (identificador de red).
- Los números de la derecha indican los equipos dentro de esta red y se les denomina host-ID(identificador de host).



Figura19.Ejemplo de dirección IP

TCP/IP

TCP/IP es un conjunto de protocolos. La sigla TCP/IP significa "Protocolo de control de transmisión/Protocolo de Internet". Proviene de los nombres de dos protocolos importantes del conjunto de protocolos, es decir, del protocolo TCP y del protocolo IP.

En algunos aspectos, TCP/IP representa todas las reglas de comunicación para Internet y se basa en la noción de dirección IP, es decir, en la idea de brindar una dirección IP a cada equipo de la red para poder enrutar paquetes de datos. Debido a que el conjunto de protocolos TCP/IP originalmente se creó con fines militares, está diseñado para cumplir con una cierta cantidad de criterios, entre ellos:

- Dividir mensajes en paquetes.
- Usar un sistema de direcciones.
- Enrutar datos por la red.
- Detectar errores en las transmisiones de datos.

El conocimiento del conjunto de protocolos TCP/IP no es esencial para un simple usuario, de la misma manera que el mismo usuario no necesita saber cómo funciona su red audiovisual o de televisión. Sin embargo, para las personas que desean administrar o brindar soporte técnico a una red TCP/IP, su conocimiento es fundamental.

En general, TCP/IP relaciona dos nociones:

- Estándar: TCP/IP representa la manera en la que se realizan las comunicaciones en una red.
- Implementación: la designación TCP/IP generalmente se extiende a software basado en el protocolo TCP/IP. En realidad, TCP/IP es un modelo cuya aplicación de red utilizan los desarrolladores. Las aplicaciones son, por lo tanto, implementaciones del protocolo TCP/IP.

Para poder aplicar el modelo TCP/IP en cualquier equipo, es decir, independientemente del sistema operativo, el sistema de protocolos TCP/IP se ha dividido en diversos módulos. Cada uno de éstos realiza una tarea específica. Además, estos módulos realizan sus tareas uno después del otro en un orden específico, es decir que existe un sistema estratificado. Ésta es la razón por la cual se habla de modelo de capas.

El término capa se utiliza para reflejar el hecho de que los datos que viajan por la red atraviesan distintos niveles de protocolos. Por lo tanto, cada capa procesa sucesivamente los datos (paquetes de información) que circulan por la red, les agrega un elemento de información (llamado encabezado) y los envía a la capa siguiente.

El objetivo de un sistema en capas es dividir el problema en diferentes partes, de acuerdo con su nivel de abstracción. Cada capa del modelo se comunica con un nivel adyacente (superior o inferior). Por lo tanto, cada capa utiliza los servicios de las capas inferiores y se los proporciona a la capa superior.

Las funciones de las diferentes capas son las siguientes:

- Capa de acceso a la red: especifica la forma en la que los datos deben enrutarse, sea cual sea el tipo de red utilizado.
- Capa de Internet: es responsable de proporcionar el paquete de datos (datagrama).

- Capa de transporte: brinda los datos de enrutamiento, junto con los mecanismos que permiten conocer el estado de la transmisión.
- Capa de aplicación: incorpora aplicaciones de red estándar (Telnet, SMTP, FTP, etc.).

Protocolo TCP

Con el uso del protocolo TCP, las aplicaciones pueden comunicarse en forma independientemente de las capas inferiores. Esto significa que los *routers* sólo tienen que enviar los datos en forma de datagramas, sin preocuparse con el monitoreo de datos porque esta función la cumple la capa de transporte.

Durante una comunicación usando el protocolo TCP, las dos máquinas deben establecer una conexión. La máquina emisora (la que solicita la conexión) se llama cliente, y la máquina receptora se llama servidor. Por eso es que decimos que estamos en un entorno Cliente-Servidor.

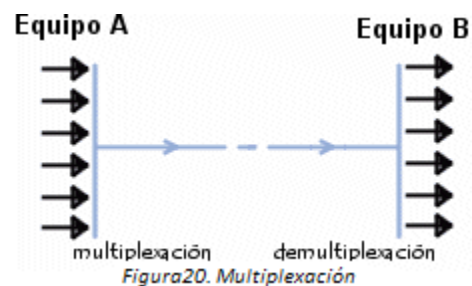
Las máquinas de dicho entorno se comunican en modo en línea, es decir, que la comunicación se realiza en ambas direcciones.

Para posibilitar la comunicación y que funcionen bien todos los controles que la acompañan, los datos se agrupan; es decir, que se agrega un encabezado a los paquetes de datos que permitirán sincronizar las transmisiones y garantizar su recepción.

Otra función del TCP es la capacidad de controlar la velocidad de los datos usando su capacidad para emitir mensajes de tamaño variable. Estos mensajes se llaman segmentos.

Multiplexación

TCP posibilita la realización de una tarea importante: multiplexar/demultiplexar; es decir transmitir datos desde diversas aplicaciones en la misma línea o, en otras palabras, ordenar la información que llega en paralelo.



Estas operaciones se realizan empleando el concepto de puertos (o conexiones), es decir, un número vinculado a un tipo de aplicación que, cuando se combina con una dirección de IP, permite determinar en forma exclusiva una aplicación que se ejecuta en una máquina determinada.

Formato del TCP

Un segmento TCP está formado de la siguiente manera:

URG ACK PSH RST SYN FIN																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Puerto de origen																Puerto de destino															
Número de secuencia																															
Número de acuse de recibo																															
Margen de datos				Reservado				Ventana																							
Suma de control																Puntero urgente															
Opciones																						Relleno									
Datos																															

Figura21. Formato del TCP

Significado de los diferentes campos:

- Puerto de origen (16 bits): Puerto relacionado con la aplicación en curso en la máquina origen.
- Puerto de destino (16 bits): Puerto relacionado con la aplicación en curso en la máquina destino.
- Número de secuencia (32 bits): Cuando el indicador SYN está fijado en 0, el número de secuencia es el de la primera palabra del segmento actual.
Cuando SYN está fijado en 1, el número de secuencia es igual al número de secuencia inicial utilizado para sincronizar los números de secuencia (ISN).
- Número de acuse de recibo (32 bits): El número de acuse de recibo, también llamado número de descarga se relaciona con el número (secuencia) del último segmento esperado y no el número del último segmento recibido.
- Margen de datos (4 bits): Esto permite ubicar el inicio de los datos en el paquete. Aquí, el margen es fundamental porque el campo opción, es de tamaño variable.

- Reservado (6 bits): Un campo que actualmente no está en uso pero se proporciona para el uso futuro.
- Indicadores (6x1 bit): Los indicadores representan información adicional:
 - URG: Si este indicador está fijado en 1, el paquete se debe procesar en forma urgente.
 - ACK: Si este indicador está fijado en 1, el paquete es un acuse de recibo.
 - PSH (PUSH): Si este indicador está fijado en 1, el paquete opera de acuerdo con el método PUSH.
 - RST: Si este indicador está fijado en 1, se restablece la conexión.
 - SYN: El indicador SYN de TCP indica un pedido para establecer una conexión.
 - FIN: Si este indicador está fijado en 1, se interrumpe la conexión.
- Ventana (16 bits): Campo que permite saber la cantidad de bytes que el receptor desea recibir sin acuse de recibo.
- Suma de control (CRC): La suma de control se realiza tomando la suma del campo de datos del encabezado para poder verificar la integridad del encabezado.
- Puntero urgente (16 bits): Indica el número de secuencia después del cual la información se torna urgente.
- Opciones (tamaño variable): Diversas opciones.
- Relleno: Espacio restante después de que las opciones se rellenan con ceros para tener una longitud que sea múltiplo de 32 bits.

Confiabilidad en la transferencia

El protocolo TCP permite garantizar la transferencia de datos confiable, a pesar de que usa el protocolo IP, que no incluye ningún monitoreo de la entrega de datagramas.

De hecho, el protocolo TCP tiene un sistema de acuse de recibo que permite al cliente y al servidor garantizar la recepción mutua de datos. Cuando se emite un segmento, se lo vincula a un número de secuencia. Con la recepción de un segmento de datos, la máquina receptora devolverá un segmento de datos donde el indicador ACK esté fijado en 1 (para poder indicar que es un acuse de recibo) acompañado por un número de acuse de recibo que equivale al número de secuencia anterior.

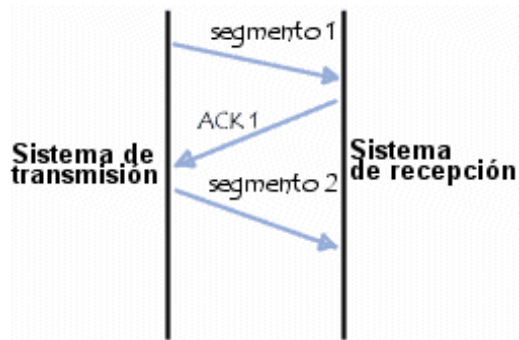


Figura22. Transmisión -recepción acuse de recibo

Además, usando un temporizador que comienza con la recepción del segmento en el nivel de la máquina originadora, el segmento se reenvía cuando ha transcurrido el tiempo permitido, ya que en este caso la máquina originadora considera que el segmento está perdido.



Figura23. Transmisión -recepción acuse de recibo para sincronización

Sin embargo, si el segmento no está perdido y llega a destino, la máquina receptora lo sabrá, gracias al número de secuencia, que es un duplicado, y sólo retendrá el último segmento que llegó a destino.

Establecimiento de conexiones.

Considerando que este proceso de comunicación, que se produce con la transmisión y el acuse de recibo de datos, se basa en un número de secuencia, las máquinas originadora y receptora (cliente y servidor) deben conocer el número de secuencia inicial de la otra máquina.

La conexión establecida entre las dos aplicaciones a menudo se realiza siguiendo el siguiente esquema:

- Los puertos TCP deben estar abiertos.

- La aplicación en el servidor es pasiva, es decir, que la aplicación escucha y espera una conexión.
- La aplicación del cliente realiza un pedido de conexión al servidor en el lugar donde la aplicación es abierta pasiva. La aplicación del cliente se considera "abierta activa".

Las dos máquinas deben sincronizar sus secuencias usando un mecanismo comúnmente llamado "negociación en tres" pasos que también se encuentra durante el cierre de la sesión.

Este diálogo posibilita el inicio de la comunicación porque se realiza en tres etapas, como su nombre lo indica:

- En la primera etapa, la máquina originadora (el cliente) transmite un segmento donde el indicador SYN está fijado en 1 (para indicar que es un segmento de sincronización), con número de secuencia N llamado número de secuencia inicial del cliente.
- En la segunda etapa, la máquina receptora (el servidor) recibe el segmento inicial que viene del cliente y luego le envía un acuse de recibo, que es un segmento en el que el indicador ACK está fijado en 1 y el indicador SYN está fijado en 1 (porque es nuevamente una sincronización). Este segmento incluye el número de secuencia de esta máquina (el servidor), que es el número de secuencia inicial para el cliente. El campo más importante en este segmento es el de acuse de recibo que contiene el número de secuencia inicial del cliente incrementado en 1.
- Por último, el cliente transmite un acuse de recibo, que es un segmento en el que el indicador ACK está fijado en 1 y el indicador SYN está fijado en 0 (ya no es un segmento de sincronización). Su número de secuencia está incrementado y el acuse de recibo representa el número de secuencia inicial del servidor incrementado en 1.

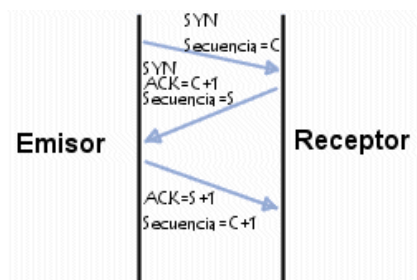


Figura24. Emisor – receptor Sincronización completa

Después de esta secuencia con tres intercambios, las dos máquinas están sincronizadas y la comunicación puede comenzar.

Existe una técnica de piratería llamada falsificación de IP, que permite corromper este enlace de aprobación con fines maliciosos.

Fin de Conexión

El cliente puede pedir que se termine una conexión del mismo modo que el servidor.

Para terminar una conexión se procede de la siguiente manera:

- Una de las máquinas envía un segmento con el indicador *FIN* fijado en 1, y la aplicación se autocoloca en estado de espera, es decir que deja de recibir el segmento actual e ignora los siguientes.
- Después de recibir este segmento, la otra máquina envía un acuse de recibo con el indicador *FIN* fijado en 1 y sigue enviando los segmentos en curso. Después de esto, la máquina informa a la aplicación que se ha recibido un segmento *FIN* y luego envía un segmento *FIN* a la otra máquina, que cierra la conexión.

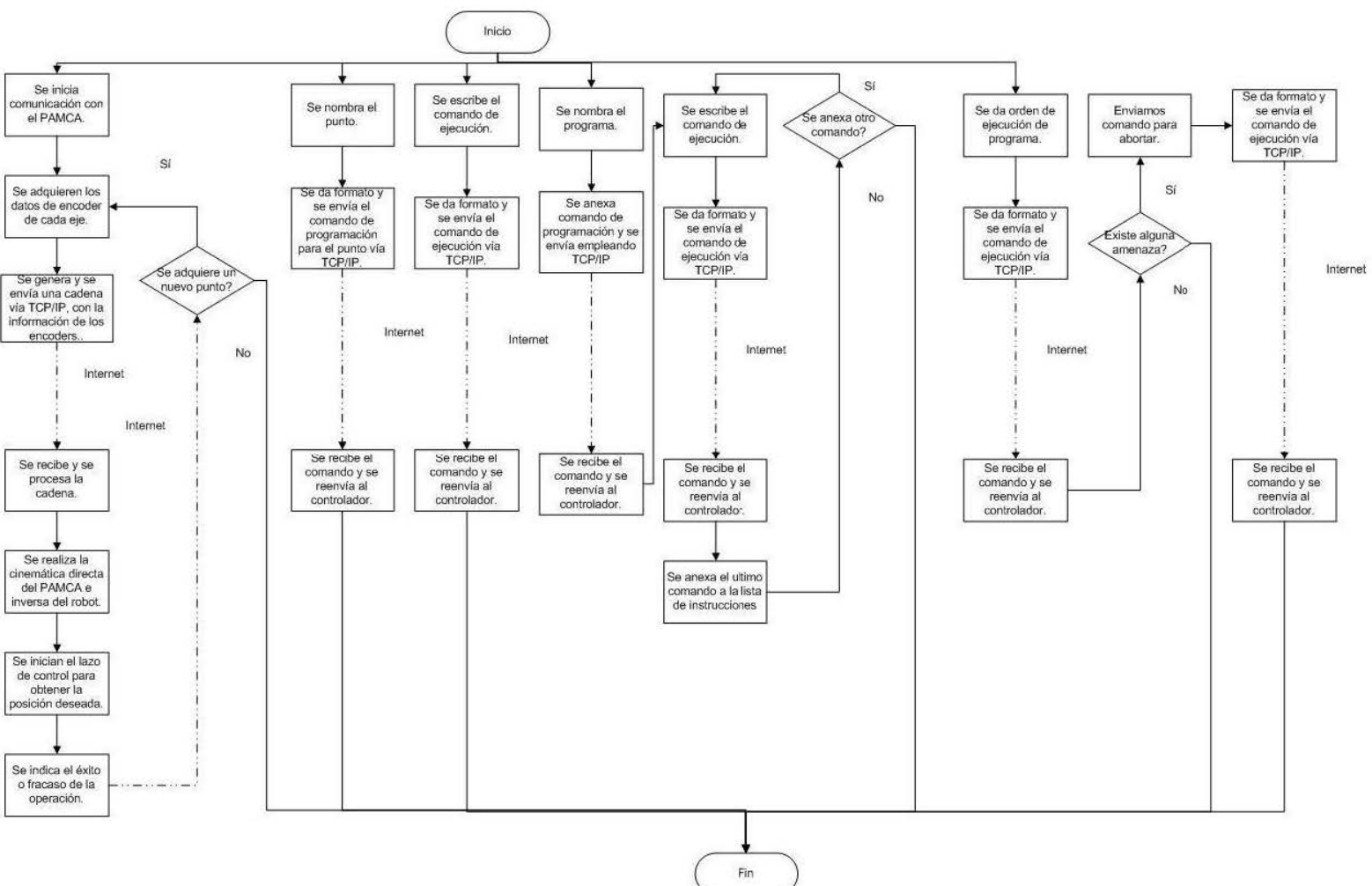
Interfaz Gráfica del Sistema Maestro – Esclavo.

La interfaz grafica tanto del programa Maestro como del Esclavo, tiene en objetivo de crear el puente de comunicación entre el usuario y la secuencia de eventos a realizar durante la ejecución del programa.

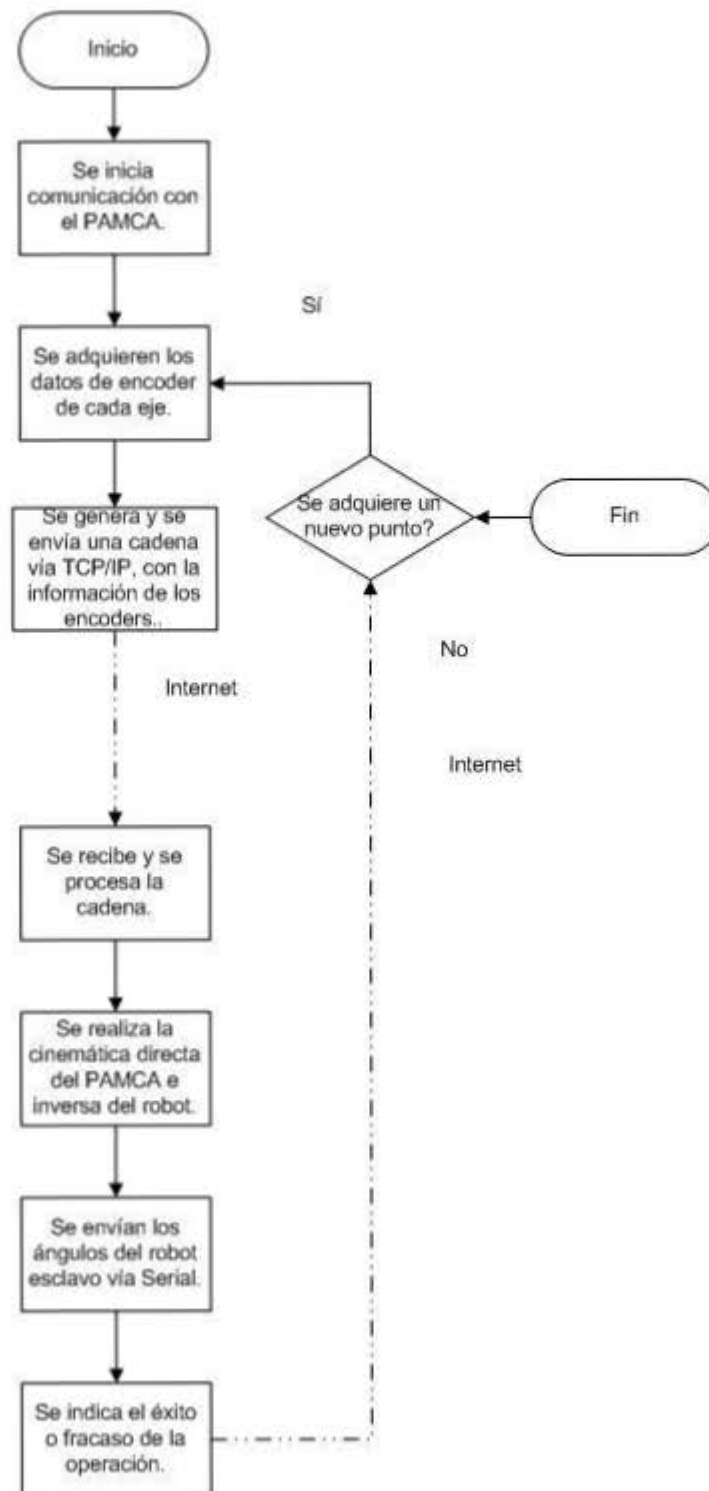
Básicamente, la interfaz grafica es el administrador directo de las acciones programadas, y permite al usuario seleccionar los parámetros y formas en las que se cumplen las órdenes. Otra de las funciones es de proveer al usuario de información sobre el estado en el que se encuentra el proceso. Desde datos numéricos hasta alarmas sobre posibles amenazas existentes. La interfaces diseñadas poseen los siguientes atributos:

1. Visualización de los datos de manera clara.
2. Agrupación de elementos semejantes.
3. Operación intuitiva del proceso.

Ahora bien, el programa maestro y el esclavo en conjunto generan una secuencia que tienen tan solo 4 funciones de trabajo que son las mostradas en la imagen inferior:



Y para la interfaz completa con el robot "Black Hawk", la operación es:



Interfaz del Maestro

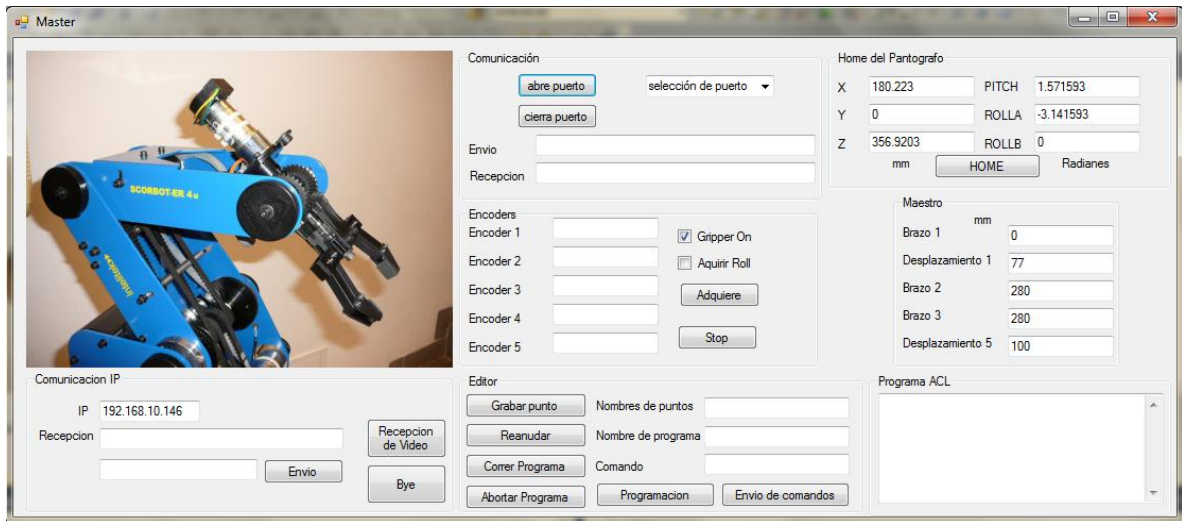


Figura25. Interfaz grafica del maestro.

La interfaz del maestro es la parte que se encarga de la operación, monitoreo y programación de la interfaz completa. Su función principal radica en enviar comandos, instrucciones, empezar y/o terminar procedimientos. En el caso específico del proyecto. La interfaz del maestro también servirá como servidor para la comunicación TCP/IP.



Figura26. Interfaz del maestro con PAMCA.

Básicamente las funciones que desempeñará son las siguientes:

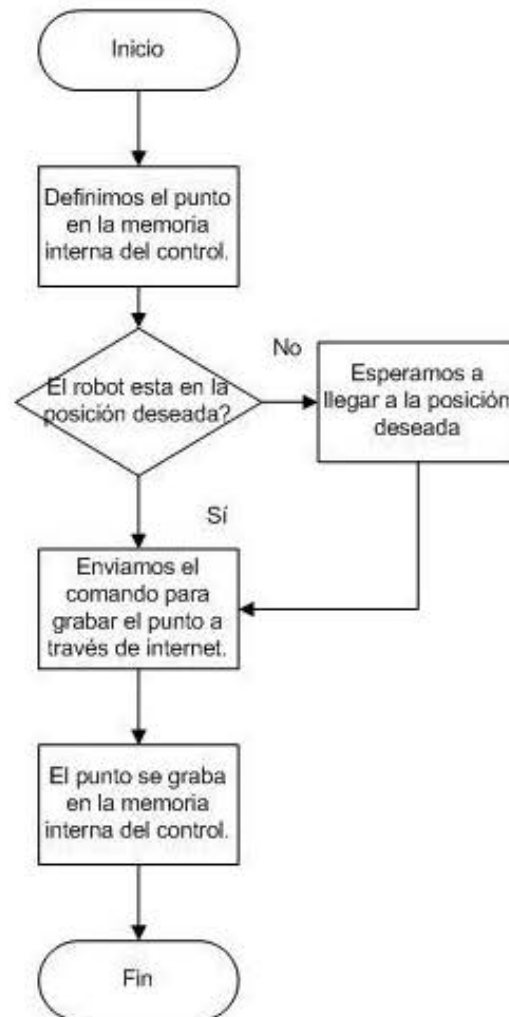
1. En primera instancia la interfaz se comunica con el PAMCA, empleando comunicación serial hacia y desde el Arduino, con el fin de identificar la posición inicial del procedimiento llamada "HOME". EL procedimiento descrito es:



- Para realizar la reproducción de geometrías por parte del manipulador que se esté controlando se realiza el envío de cadenas de comando que contendrán la información de los ángulos de las juntas del PAMCA. Con esta información el esclavo realizara las operaciones pertinentes para realizar el movimiento. La ejecución. El procedimiento sigue los siguientes pasos:



3. Una vez ubicado un punto en el espacio, basta con grabarlo en la memoria interna del controlador empleando un comando de control para utilizarlo cuantas veces sea necesario en alguna aplicación deseada. El procedimiento es el siguiente:



4. Para generar trayectorias basta con indicar el punto final y el movimiento deseado, e indicar el comando específico para su ejecución es decir:



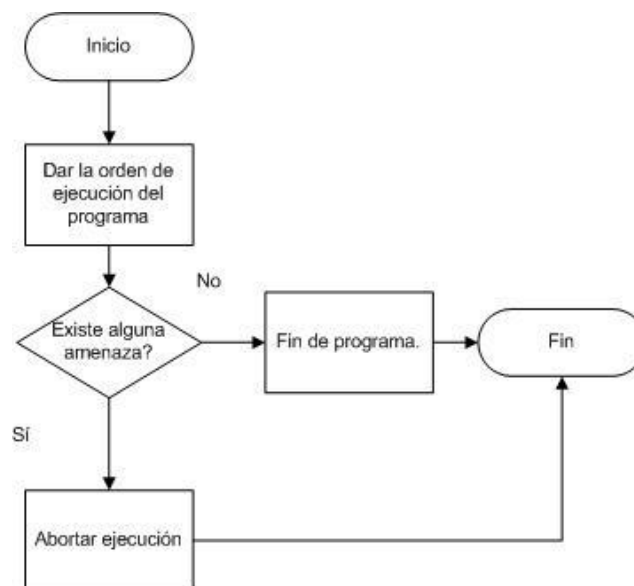
5. Para realizar la teleprogramación del dispositivo basta con nombrar el programa, e indicar las operaciones que va a realizar mediante comandos, como se ve a continuación:



6. Como parte del proceso, el maestro siempre se encuentra visualizando los movimientos del esclavo empleando una cámara para este fin. Por lo que la adquisición de dicha imagen desde el esclavo es:



7. La última de las acciones asociadas principalmente al maestro se encuentra en el telemonitoreo de los programas, o procedimientos que realice el esclavo. Este monitoreo se basa únicamente en la lógica binaria de “continuar” o “abortar”.



Interfaz del Esclavo

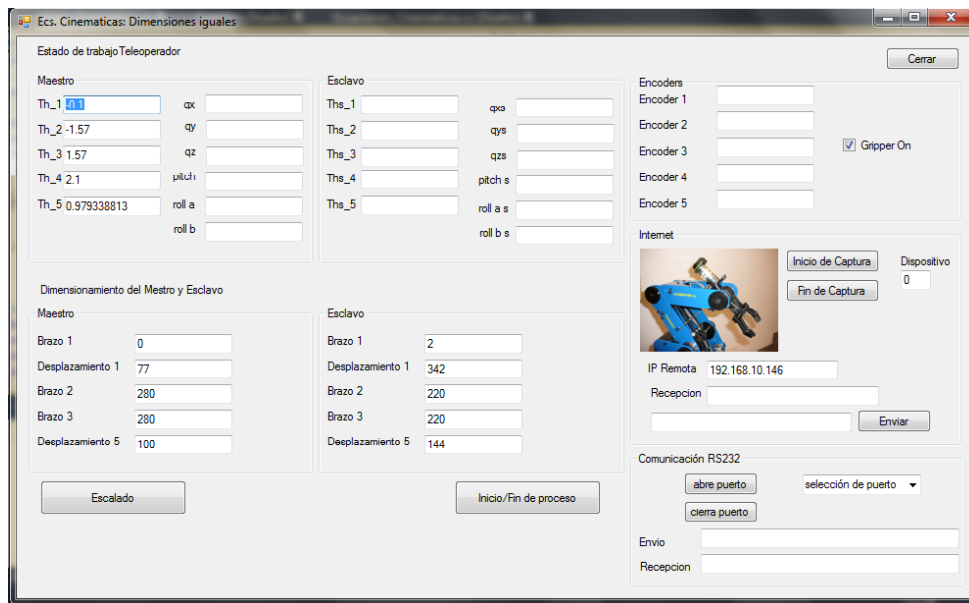


Figura27. Interfaz grafica del esclavo.

La interfaz del esclavo es la encargada de ejecutar todas las funciones relacionadas con el movimiento, programación y operación del robot. Por decirlo de alguna manera, es la parte ejecutora de las acciones. También se encarga en específico, para éste proyecto de realizar la adquisición de la imagen proveniente de la cámara conectada a la PC. En la interfaz de conexión TCP/IP, está programado para ser el cliente.

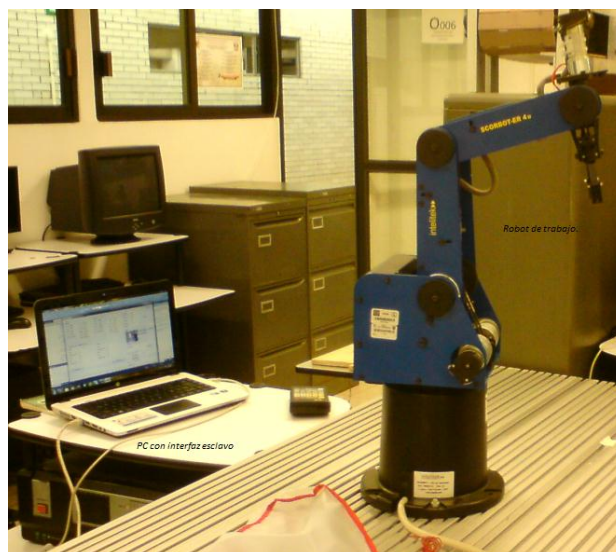
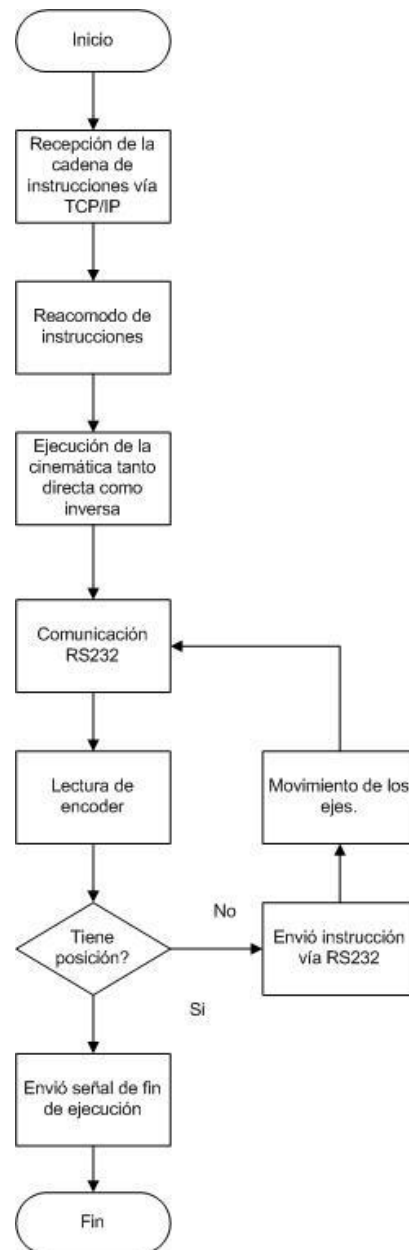


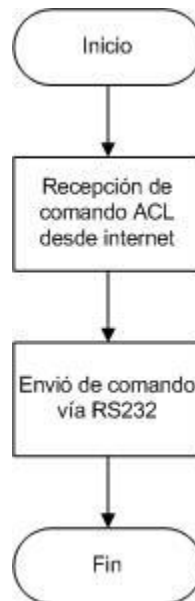
Figura28. Robot esclavo con Interfaz a PC.

Las funciones que tiene bajo su mando son las siguientes:

1. Realizar los movimientos ordenados por la interfaz del maestro. Es decir, tiene como primer objetivo reproducir las geometrías de los puntos ordenados por el maestro.



- Mandar los comandos provenientes del maestro al controlador, con el fin de realizar trayectorias, realizar/editar programas, grabar puntos y ejecutar/abortar programas.



- Finalmente, la interfaz también tiene el objetivo de realizar la adquisición de video de alguna fuente conectada a la PC.



Implementación del proyecto.

Si se puede imaginar, se puede programar.

Como se ha descrito en el capítulo anterior, los algoritmos mostrados deben de ser sucedidos por la parte ejecutable de un proyecto; es decir, se transformo de la idea a la aplicación en el campo virtual.

En este proyecto cada algoritmo tiene su propia interfaz gráfica, sus propios métodos y su propia secuencia de ejecución.

Implementación del maestro.

Comunicación de la interfaz con la computadora y el PAMCA.

Para lograr la comunicación entre la interfaz de la computadora y el PAMCA, fue necesario realizar comunicación serial empleando un cable entrada USB estándar “A” y salida estándar “B” con el Arduino MEGA 2560. Una vez conectado el hardware, se procedió a realizar la comunicación mediante el software. Para llevar a cabo este procedimiento en C#, creamos 3 métodos. El primero es `busca_puerto()`, `abre_puerto()` y `cierra_puerto()`. Básicamente desde que se inicia el programa se buscan los puertos disponibles, se hace una lista de los mismos, seleccionamos el puerto que se conecta con el Arduino Mega 2560, y lo abrimos para comunicarnos, una vez terminada la adquisición de datos del PAMCA, podemos cerrar el puerto. Los métodos anteriormente citados serán descritos a continuación:

```
private void Form1_Load(object sender, EventArgs e)//Se inicia la búsqueda del puerto
{
    busca_puertos(cmbPuerto);
}
private void busca_puertos(ComboBox cb)//Metodo para buscar los puertos conectados
{
    string[] ports = SerialPort.GetPortNames();//Indica los nombres de los puertos disponibles
    cb.Items.Clear();
    foreach (string port in ports)
    {
        cb.Items.Add(port);//Permite realizar una lista de puertos disponibles
    }
    cb.Text = "selección de puerto";
}
private void abre_puerto();//Metodo para abrir el puerto
{
    if (cmbPuerto.Text != "selección de puerto")
    {
        //Establecemos los parametros de la comunicacion serial, indicando que puerto se usara.
        puerto = new SerialPort(cmbPuerto.Text, 9600, Parity.None, 8, StopBits.One);
        if (puerto != null)
            puerto.Open();//abrimos el puerto
    }
}
private void cierra_puerto();//Metodo para cerrar el puerto
{
    if (puerto != null)
    {
        if (puerto.IsOpen == true)
            puerto.Close();//Se cierra el puerto que este en uso.
    }
}
```

Para realizar el envío y recepción de información con el puerto serial, se programaron los métodos `enviar(cadena)` y `recibir(cadena)`. Los cuales permiten realizar las operaciones antes mencionadas. Su código en C# es:

```
private void enviar(string mensaje)//Metodo para enviar informacion via serial
{
    if ((puerto != null) && (puerto.IsOpen == true))//Condicion de puerto disponible
        puerto.Write(mensaje);//Enviamos la cadena deseada
}
private string recibir()//Metodo para recibir informacion via serial
{
    if (puerto.IsOpen == true)//Condicion de puerto disponible
    {
        puerto.ReadTimeout = 1500;//Tiempo de espera para recibir respuesta
        return txtRecibir.Text = puerto.ReadExisting();//Recibe los datos del puerto serial
    }
    else return "";
}
```

Para ejecutar los métodos anteriormente descritos desde la interfaz del maestro, basta con ubicar el recuadro de “comunicación” y darle los parámetros que pida, como es el caso de ubicar el puerto al cual se conecta el PAMCA y si se inicia o se finaliza la comunicación. Además, la interfaz grafica permite la visualización de la información que transita a través del puerto serie.

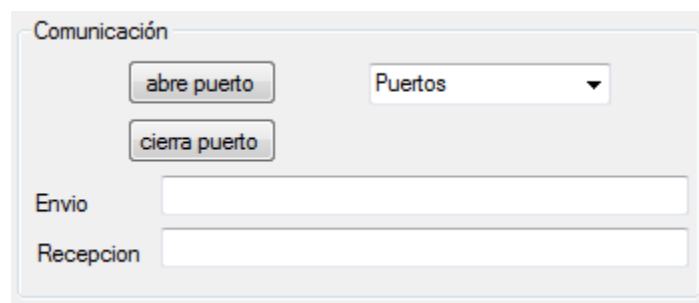


Figura1. Parte de la interfaz del Maestro, donde se ubica la comunicación serial.

Para realizar la ubicación de pantógrafo en el espacio por parte de la interfaz, y con ello ubicar las posiciones de los ángulos en cada junta. Ubicamos espacialmente al PAMCA en una posición donde determinamos su posición y orientación del efector final empleando los ángulos de Euler y coordenadas de cada eje. Dicha información se escribe directamente en la pantalla de la interfaz. Es necesario también considerar que se va a

realizar la cinemática inversa por lo que también se indican las dimensiones físicas en el recuadro que se titula como “Maestro”. Como se muestra en la siguiente imagen:

Figura2. Parte de la interfaz del Maestro, donde se ubica el HOME y se alimentan las dimensiones del PAMCA.

Una vez alimentada la interfaz con los valores de los parámetros pedidos se procede a presionar el botón “HOME”, de la interfaz que permite realizar la cinemática inversa y con ello ubicar los ángulos en los cuales la interfaz se encuentra. A partir de esto hacemos que los valores que se reciben por parte del PAMCA sean cero enviando el carácter “r” al Arduino Mega 2560 y se le agrega el ángulo calculado en la cinemática inversa en cada eje. Con ello la interfaz genera la ubicación precisa de los ángulos de cada eje.

```
//Calculamos la cinemática inversa del pantografo con las ecuaciones de D-H
//Vector U
ux = Convert.ToSingle(Math.Cos(rolla) * Math.Cos(pitch) * Math.Cos(rollb) - Math.Sin(rolla) * Math.Sin(rollb));
uy = Convert.ToSingle(Math.Sin(rolla) * Math.Cos(pitch) * Math.Cos(rollb) + Math.Cos(rolla) * Math.Sin(rollb));
uz = Convert.ToSingle(-Math.Sin(pitch) * Math.Cos(rollb));
//Ángulos del PAMCA
Th1 = Convert.ToSingle(Math.Atan(y / x));
Th5 = Convert.ToSingle(Math.Asin(x * Math.Sin(Th1) - uy * Math.Cos(Th1)));
Th234 = Convert.ToSingle(Math.Atan2(-uz / Math.Cos(Th5), (ux * Math.Cos(Th1) + uy * Math.Sin(Th1)) / Math.Cos(Th5)));
k1 = Convert.ToSingle((x * Math.Cos(Th1) + y * Math.Sin(Th1)) - braz1 + (desp5 * Math.Sin(Th234)));
k2 = Convert.ToSingle(-z + desp1 - (desp5 * Math.Cos(Th234)));
Th3 = Convert.ToSingle(Math.Acos((Math.Pow(k1, 2) + Math.Pow(k2, 2) - Math.Pow(braz2, 2) - Math.Pow(braz3, 2)) / (2 * braz2 * braz3)));
Th2c = Convert.ToSingle((k1 * (braz2 + braz3 * Math.Cos(Th3)) + k2 * braz3 * Math.Sin(Th3)) / (Math.Pow(braz2, 2) + Math.Pow(braz3, 2) + 2 * braz2 * braz3 * Math.Cos(Th3)));
Th2s = Convert.ToSingle((-k1 * braz3 * Math.Sin(Th3) + k2 * (braz2 + braz3 * Math.Cos(Th3))) / (Math.Pow(braz2, 2) + Math.Pow(braz3, 2) + 2 * braz2 * braz3 * Math.Cos(Th3)));
Th2 = Convert.ToSingle(Math.Atan2(Th2s, Th2c));
Th4 = Th234 - Th2 - Th3;
if (puerto.IsOpen == true)
{
    txtRecibir.Text = "";
    puerto.WriteLine("r\r");
    txt_envio.Text = "";
}
```

Una vez ubicados los ángulos de inicio del PAMCA podemos empezar a recoger la información proveniente del dispositivo. Y también se controla directamente en la interfaz gráfica si el *gripper* del SCORBOT-ER 4u se mantiene abierto o se cierra, o si se adquieren los cinco grados de libertad o solo 4.

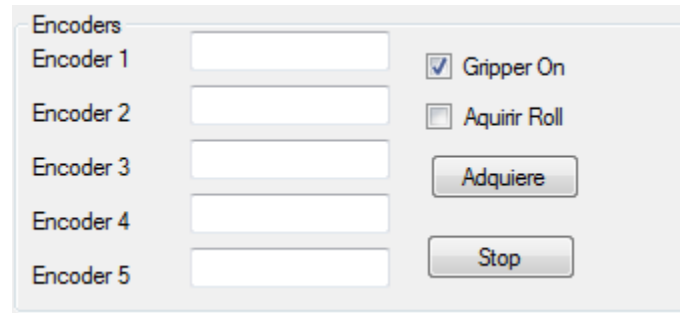


Figura3. Parte de la interfaz del Maestro, donde se ubica presentan los ángulos de cada junta del PAMCA.

Para inicial el procedimiento de la adquisición y envió de las cadenas con las instrucciones que realizará el maestro para la reproducción de geometrías se pulsa el botón “Adquire” y para finalizar de igual forma el envió de instrucciones se pulsa el botón “Stop”. A continuación mostramos el código que activa cada botón:

```
private void Adquiere_Click(object sender, EventArgs e)
{
    if (puerto.IsOpen == true)
    {
        puerto.WriteLine("i\r"); //Inicia el envio de datos de los angulos
        timer2.Enabled = true; //Inicia el timer que controla la recepcion de datos
    }
}

private void Stop_Click(object sender, EventArgs e)
{
    if (puerto.IsOpen == true)
    {
        timer2.Enabled = false;
        puerto.WriteLine("o\r"); //Finaliza el envio de datos de los angulos
    }
}
```

El *timer* que se activa cuenta con el siguiente proceso interno, que además se reinicia cada 100ms.

```

if (puerto.IsOpen == true)
{
    puerto.ReadTimeout = 65;//Tiempo para recibir la informacion del Arduino
    txtRecibir.Text = puerto.ReadLine();//Lectura de datos
}
if (txtRecibir.Text == "@@\r")//If que indica el inicio de una cadena correcta
{
    //Lectura de los 10 datos que el programa del Arduino envia
    enc1= puerto.ReadLine();
    enc2 = puerto.ReadLine();
    enc3 = puerto.ReadLine();
    enc4 = puerto.ReadLine();
    enc5 = puerto.ReadLine();
    txtRecibir.Text = puerto.ReadLine();
    txtRecibir.Text = puerto.ReadLine();
    txtRecibir.Text = puerto.ReadLine();
    txtRecibir.Text = puerto.ReadLine();
    txtRecibir.Text = puerto.ReadLine();
    //Transformacion de String a int, con el angulo inicial incluido del HOME
    enco1 = Convert.ToInt32(enc1) + Convert.ToInt32((Th1 * (180 / 3.1416F)));
    enco2 = Convert.ToInt32(enc2) + Convert.ToInt32((Th2 * (180 / 3.1416F)));
    enco3 = Convert.ToInt32(enc3) + Convert.ToInt32((Th3 * (180 / 3.1416F)));
    enco4 = Convert.ToInt32(enc4) + Convert.ToInt32((Th4 * (180 / 3.1416F)));
    enco5 = Convert.ToInt32(enc5) + Convert.ToInt32((Th5 * (180 / 3.1416F)));
    //Se reescribe los angulos en String
    txt_enc1.Text = Convert.ToString(enco1);
    txt_enc2.Text = Convert.ToString(enco2);
    txt_enc3.Text = Convert.ToString(enco3);
    txt_enc4.Text = Convert.ToString(enco4);
    txt_enc5.Text = Convert.ToString(enco5);

//Codigo para establecer la cadena de instrucciones a enviar
1) (adquisicion_rol1.Checked == true)//Condicion para adquirir el rol1 de RWCA
{
    1) (txt_gripper.Checked == true)//Condicion para abrir o cerrar el grripper
    {
        txt_envio.Text = String.Concat("X", "A", txt_enc1.Text, "B", txt_enc1.Text, "C", txt_enc1.Text, "D", txt_enc1.Text, "E", txt_enc1.Text, "F");
    }
    else
        txt_envio.Text = String.Concat("X", "A", txt_enc1.Text, "B", txt_enc2.Text, "C", txt_enc2.Text, "D", txt_enc2.Text, "E", txt_enc2.Text, "F");
}
else
{
    2) (txt_gripper.Checked == true)//Condicion para abrir o cerrar el grripper
    {
        txt_envio.Text = String.Concat("X", "A", txt_enc1.Text, "B", txt_enc1.Text, "C", txt_enc3.Text, "D", txt_enc3.Text, "E");
    }
    else
        txt_envio.Text = String.Concat("X", "A", txt_enc1.Text, "B", txt_enc1.Text, "C", txt_enc3.Text, "D", txt_enc4.Text, "E");
}
}

//si existe comunicacion se/ir enviar las instrucciones
if (txt_envio.Lines.Length > 1 && handshready == true && handshflirta==true)
{
    handshready = false;
    txt_envio = txt_envio.Text;
    Cls_Servicio.SendMessage(sender);
    txt_envio.Lines = null;
}
txt_envio.Text = "";
suSender = "";
txtRecibir.Text = "";

```

La interfaz de adquisición anteriormente citada fue utilizada en las pruebas para el SCORBOT-ER 4u, sin embargo el mismo método fue refinado para las pruebas con el

“Black Hawk”. Utilizando en lugar del *Timer* un *Thread*, que permite realizar operaciones en paralelo a las que se estén ejecutando en el momento. Lo cual impacta en el código de la interfaz de la siguiente manera:

```
private void Stop_Click(object sender, EventArgs e)
{
    thrinfo.Abort();
    //Detenemos el Thread
    if (puerto.IsOpen == true)
    {
        //Terminamos el flujo de señal
        puerto.WriteLine("o\r");
    }
}
//Metodo para llamar las acciones ante actualizacion
private void GetInformation()
{
    while (puerto.IsOpen == true)
    {
        //Se activa ante actualizacion de datos.
        this.Invoke(new EventHandler(DoUpdate));
    }
}
```

Ahora indicamos las acciones que se realizaran cuando se refresquen los datos provenientes de la comunicación serial:

```
private void DoUpdate(object s, EventArgs e)
{
    puerto.ReadTimeout = 65; //Tiempo para recibir la informacion del Arduino
    txtRecibir.Text = puerto.ReadLine(); //Lectura de datos
    if (txtRecibir.Text == "@@\r") //Condición que indica el inicio de una cadena correcta
    {
        //Lectura de los 10 datos que el programa del Arduino envia
        enc1 = puerto.ReadLine();
        enc2 = puerto.ReadLine();
        enc3 = puerto.ReadLine();
        enc4 = puerto.ReadLine();
        enc5 = puerto.ReadLine();
        txtRecibir.Text = puerto.ReadLine();
        txtRecibir.Text = puerto.ReadLine();
        txtRecibir.Text = puerto.ReadLine();
        txtRecibir.Text = puerto.ReadLine();
        txtRecibir.Text = puerto.ReadLine();
        //Transformacion de String a int, con el angulo inicial incluido del HOME
        enco1 = Convert.ToInt32(enc1) + Convert.ToInt32((Th1 * (180 / 3.1416F)));
        enco2 = Convert.ToInt32(enc2) + Convert.ToInt32((Th2 * (180 / 3.1416F)));
        enco3 = Convert.ToInt32(enc3) + Convert.ToInt32((Th3 * (180 / 3.1416F)));
        enco4 = Convert.ToInt32(enc4) + Convert.ToInt32((Th4 * (180 / 3.1416F)));
        enco5 = Convert.ToInt32(enc5) + Convert.ToInt32((Th5 * (180 / 3.1416F)));
        //Se reescribe los angulos en String
        txt_enc1.Text = Convert.ToString(enco1);
        txt_enc2.Text = Convert.ToString(enco2);
        txt_enc3.Text = Convert.ToString(enco3);
        txt_enc4.Text = Convert.ToString(enco4);
        txt_enc5.Text = Convert.ToString(enco5);
    }
}
```

En la primera parte del código se compensan los datos con el ángulo home, ahora se forma una cadena de comandos para su ejecución en el manipulador:

```
//Codigo para establecer la cadena de instrucciones a enviar
if (adquisicion_roll.Checked == true)//Condicion para adquirir el roll de PAMCA
{
    if (txt_gripper.Checked == true)//Condicion para abrir o cerrar el gripper
    {
        txt_envio.Text = String.Concat('X', 'A', txt_enc1.Text, 'B', txt_enc2.Text, 'C', txt_enc3.Text, 'D', txt_enc4.Text, 'E', txt_enc5.Text, 'O');
    }
    else
        txt_envio.Text = String.Concat('X', 'A', txt_enc1.Text, 'B', txt_enc2.Text, 'C', txt_enc3.Text, 'D', txt_enc4.Text, 'E', txt_enc5.Text, 'L');
}
else
{
    if (txt_gripper.Checked == true)//Condicion para abrir o cerrar el gripper
    {
        txt_envio.Text = String.Concat('X', 'A', txt_enc1.Text, 'B', txt_enc2.Text, 'C', txt_enc3.Text, 'D', txt_enc4.Text, 'O');
    }
    else
        txt_envio.Text = String.Concat('X', 'A', txt_enc1.Text, 'B', txt_enc2.Text, 'C', txt_enc3.Text, 'D', txt_enc4.Text, 'L');
}
}
//Si existe comunicacion TCP/IP enviar las instrucciones
if (txt_envio.Lines.Length >= 1 && banderaready == true && banderaflujo==true)
{
    banderaready = false;
    swSender = txt_envio.Text;
    ChatServer.SendAdminMessage(swSender);
    txt_envio.Lines = null;
}
txt_envio.Text = "";
swSender = "";
txtRecibir.Text = "";
```

Finalmente la parte final de la interfaz realiza la teleprogramación y tele-ejecución de los programas internos del controlador.

La interfaz de programación cuenta con 2 elementos distintivos, la primera parte que es el recuadro de Editor se refiere a los botones de las asignaciones a utilizar. Forman parte de la programación las acciones de grabar un punto, reanudar un programa/pausar, abortar el programa, correr el programa, iniciar la programación de un programa, y el envío de comandos tanto de programación como de ejecución directa para trayectorias. Todos estos comandos tienen que ser escritos en lenguaje ACL que es el lenguaje que maneja el controlador (Para mayor información consultar el apéndice 1 de la presente tesis).

También contiene tres cajas de texto donde podemos nombrar los puntos a grabar, dar nombre al programa a programar o ejecutar y escribir los comandos.

En el recuadro nombrado “Programa ACL” guarda todas las acciones relacionadas con la teleprogramación del controlador del robot. Con lo que nos permite revisar las acciones antes de ejecutarlas.

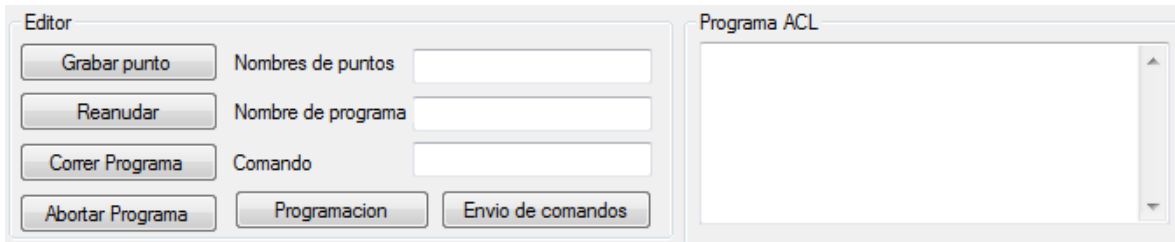


Figura4. Parte de la interfaz del Maestro, donde se ubica la teleprogramación y tele-ejecución de programas

El código de la parte de la interfaz anteriormente citada es:

Para grabar un punto:

```
private void gbr_pto_Click(object sender, EventArgs e)
{
    banderaflujo = false; //Bandera para parar la transmisión de datos hacia el Esclavo
    cadena_ptos = "Z"; //Caracter que indica que se graba un punto
    cadena_ptos = string.Concat(cadena_ptos, txt_ptos.Text); //Formación de la cadena
    txtLog.AppendText(txt_ptos.Text + "\r\n"); //Mostrar punto en lista de acciones
    swSender = cadena_ptos;
    ChatServer.SendAdminMessage(swSender); //Envio de cadena
    txt_envio.Lines = null;
    banderaflujo = true; //Reiniciar el envío de instrucciones
}
}
```

Método para programar un programa en el controlador de esclavo:

```
private void Programacion_Click(object sender, EventArgs e)
{
    editor = "Y"; //Caracter que indica que se graba un programa
    editor = string.Concat(editor, txt_nom_prog.Text); //Formación de la cadena
    txtLog.AppendText(txt_nom_prog.Text + "\r\n"); //Mostrar punto en lista de acciones
    swSender = editor;
    ChatServer.SendAdminMessage(swSender); //Envio de cadena
    txt_envio.Lines = null;
}
}
```

Método para enviar comandos ACL:

```
private void cmd_envio_Click(object sender, EventArgs e)
{
    editor = "Y"; //Caracter que indica que se envia un comando ACL
    editor = string.Concat(editor, txt_cmd.Text); //Formación de la cadena
    swSender = editor;
    txtLog.AppendText(txt_cmd.Text + "\r\n"); //Mostrar en lista de acciones
    ChatServer.SendAdminMessage(swSender); //Envio de cadena
    txt_envio.Lines = null;
}
```

Método para correr programas:

```
private void run_prog_Click(object sender, EventArgs e)
{
    editor = "YRUN "; //Caracter que indica que se corre el programa
    editor = string.Concat(editor, txt_nom_prog.Text); //Formación de la cadena
    txtLog.AppendText("RUN " + txt_nom_prog.Text + "\r\n"); //Mostrar en lista de acciones
    swSender = editor;
    ChatServer.SendAdminMessage(swSender); //Envio de cadena
    txt_envio.Lines = null;
}
```

Método para abortar la ejecución de programas.

```
private void Abort_Click(object sender, EventArgs e)
{
    editor = "YA "; //Caracter que indica que se aborta el programa
    editor = string.Concat(editor, txt_nom_prog.Text); //Formación de la cadena
    txtLog.AppendText("A " + txt_nom_prog.Text + "\r\n"); //Mostrar en lista de acciones
    swSender = editor;
    ChatServer.SendAdminMessage(swSender); //Envio de cadena
    txt_envio.Lines = null;
}
```

Implementación del esclavo.

Comunicación de la interfaz con el *Controller-A*

En primera instancia se realiza la comunicación vía serial con el controlador empleando un cable RS232 a la computadora. Para llevar a cabo la comunicación en el software de la interfaz se programan los parámetros de la comunicación. Con una velocidad de 9600 baudios, sin paridad, con 8 bits con un bit de paro. Como se muestra:

```

puerto = new SerialPort(cmbPuerto.Text, 9600, Parity.None, 8, StopBits.One);
if (puerto != null)
    puerto.Open();

```

Se utilizaron los mismos métodos de la interfaz del maestro para la conexión con el puerto serie en el esclavo.

Actualización ante la recepción de una nueva cadena de instrucción:

En primer lugar al igual que en el código del maestro existe el código del método UpdateLog(string), lo que nos permite refrescar la información apenas llegue la cadena desde el protocolo TCP/IP.

```

while (Connected)
{
    // Solicitamos que mientras este conectado, se refresque la información con el UpdateLog
    this.Invoke(new UpdateLogCallback(this.UpdateLog), new object[] { srReceiver.ReadLine() });
}

```

El algoritmo primero revisa el carácter inicial de la cadena entrante a la interfaz. Y existen solo tres posibilidades:

1. El carácter es X. Esto es el indicativo de que se inicia una nueva cadena de instrucciones que indican el inicio de una cadena para reproducir la geometría de los puntos. Inicia la programación para realizar la geometría.
2. El carácter es Z. Este es el indicativo de que se va a grabar un punto en la posición en la que se encuentre ubicado el SCORBOT-ER 4u. Se inicia la programación para grabar el punto en el controlador.
3. El carácter es Y. Este es el indicativo de que se va a enviar un comando en ACL. Se inicia la programación para enviar el comando al controlador.

Programación para generar la geometría del punto en el SCORBOT-ER 4u.

Una vez adquirida la cadena, y teniendo como inicio de la misma la X. Procedemos a separar la información en los diferentes ángulos requeridos (el ángulo 5 puede o no ser enviado, esto se debe a que podríamos evitar el roll para hacer al proceso más rápido de ser necesario).

```

//Caracter que indica inicio del proceso para reproducir puntos
if (cadena_ip[0] == 'X')
{
    for (v = 0; v < (cadena_ip.Length - 1); v++)
    {
        //Ubicacion de las separaciones de angulos en la cadena recibida
        if (cadena_ip[v] == 'A')
        {
            banderacaja1n = v;
        }
        else if (cadena_ip[v] == 'B')
        {
            banderacaja2n = v;
        }
        else if (cadena_ip[v] == 'C')
        {
            banderacaja3n = v;
        }
        else if (cadena_ip[v] == 'D')
        {
            banderacaja4n = v;
        }
        else if (cadena_ip[v] == 'E')
        {
            banderacaja5n = v;
        }
    }
}

```

También desde la cadena del maestro viene la información sobre la apertura o cierre del *gripper* en el efector final. Esta acción la manejamos simplemente con una bandera.

```

else if (cadena_ip[v] == 'O')
{
    banderagrippern = v;
    banderagripper = true;
}
else if (cadena_ip[v] == 'L')
{
    banderagrippern = v;
    banderagripper = false;
}

```

Ahora la información ya ordenada es procesada para convertir estas *strings* con ángulos en datos numéricos en radianes.

```
//Strings con la informacion del angulo deseado por eje, en grados
for (ers = banderacaja1n + 1; ers < banderacaja2n; ers++)
{
    caja1 = String.Concat(caja1, cadena_ip[ers]);
}
for (ers = banderacaja2n + 1; ers < banderacaja3n; ers++)
{
    caja2 = String.Concat(caja2, cadena_ip[ers]);
}
for (ers = banderacaja3n + 1; ers < banderacaja4n; ers++)
{
    caja3 = String.Concat(caja3, cadena_ip[ers]);
}
if (banderacaja5n > 0)
{
    for (ers = banderacaja4n + 1; ers < banderacaja5n; ers++)
    {
        caja4 = String.Concat(caja4, cadena_ip[ers]);
    }
    for (ers = banderacaja5n + 1; ers < banderagrrippern; ers++)
    {
        caja5 = String.Concat(caja5, cadena_ip[ers]);
    }
}
else
{
    for (ers = banderacaja4n + 1; ers < banderagrrippern; ers++)
    {
        caja4 = String.Concat(caja4, cadena_ip[ers]);
    }
}

//Conversion del string a numero flotante, en radianes
caja_1 = (Convert.ToSingle(caja1)) * (3.1416F / 180);
caja_2 = (Convert.ToSingle(caja2)) * (3.1416F / 180);
caja_3 = (Convert.ToSingle(caja3)) * (3.1416F / 180);
caja_4 = (Convert.ToSingle(caja4)) * (3.1416F / 180);
//Los angulos en radianes son desplegados en las cajas de texto de la interfaz
Th_1.Text = Convert.ToString(caja_1);
Th_2.Text = Convert.ToString(caja_2);
Th_3.Text = Convert.ToString(caja_3);
Th_4.Text = Convert.ToString(caja_4);
if (caja5 == "")
{
    Th_5.Text = Th_5.Text;

    if (caja_1 == caja_1r && caja_2 == caja_2r && caja_3 == caja_3r && caja_4 == caja_4r)
        punto_igual = true;
    else
    {
        punto_igual = false;
        caja_1r = caja_1;
        caja_2r = caja_2;
        caja_3r = caja_3;
        caja_4r = caja_4;
    }
}
}
```

```

else
{
    caja_5 = (Convert.ToSingle(caja5)) * (3.1416F / 180);
    Th_5.Text = Convert.ToString(caja_5);
    //Condicion para conocer si el punto ingresado es diferente al ultimo punto
    if (caja_1 == caja_1r && caja_2 == caja_2r && caja_3 == caja_3r && caja_4 == caja_4r && caja_5==caja_5r)
        punto_igual = true;
    else
    {
        //Guardamos el ultimo punto a reproducir
        punto_igual = false;
        caja_1r = caja_1;
        caja_2r = caja_2;
        caja_3r = caja_3;
        caja_4r = caja_4;
        caja_5r = caja_5;
    }
}
}

```

Ya teniendo los ángulos en radianes podemos mostrarlos en la caja de texto destinada para este fin:

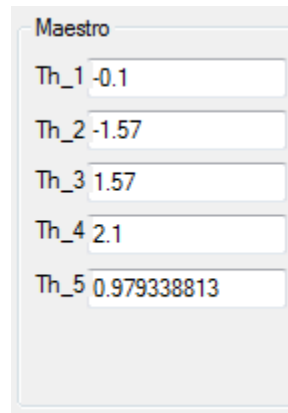


Figura5. Parte de la interfaz grafica del esclavo donde se desplegara la información de los ángulos convertidos en radianes.

Ahora ejecutaremos la cinemática directa para conocer los parámetros espaciales del PAMCA e inversa para conocer los ángulos que debe ejecutar el esclavo.

Para este fin, el esclavo requiere de la información sobre las dimensiones del PAMCA y del Esclavo. Las cuales son alimentadas en la interfaz grafica directamente:

Dimensionamiento del Maestro y Esclavo	
Maestro	
Brazo 1	<input type="text" value="0"/>
Desplazamiento 1	<input type="text" value="77"/>
Brazo 2	<input type="text" value="280"/>
Brazo 3	<input type="text" value="280"/>
Desplazamiento 5	<input type="text" value="100"/>
Esclavo	
Brazo 1	<input type="text" value="2"/>
Desplazamiento 1	<input type="text" value="342"/>
Brazo 2	<input type="text" value="220"/>
Brazo 3	<input type="text" value="220"/>
Desplazamiento 5	<input type="text" value="144"/>

Figura6. Parte de la interfaz grafica del esclavo donde se alimenta la información sobre las dimensiones.

Ahora la única diferencia real que existe entre la interface diseñada para realizar los ángulos tal cual y la cinemática inversa es el nombre de la clase a ejecutar.

```
//Llamamos a la clase Cinematicas para realizar la cinemática inversa escalada de la geometría a reproducir.
dir_inv analisis = new dir_inv(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15);
```

```
//Llamamos a la clase Cinematicas para realizar la cinemática inversa de la geometría a reproducir.
dir_inv_iguales analisis = new dir_inv_iguales(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15);
qx.Text = Convert.ToString(analisis.qx());
qy.Text = Convert.ToString(analisis.qy());
qz.Text = Convert.ToString(analisis.qz());
rolla.Text = Convert.ToString(analisis.rolla());
rollb.Text = Convert.ToString(analisis.rollb());
pitch.Text = Convert.ToString(analisis.pitch());
Ths_1.Text = Convert.ToString(analisis.Th1s());
Ths_2.Text = Convert.ToString(analisis.Th2s());
Ths_3.Text = Convert.ToString(analisis.Th3s());
Ths_4.Text = Convert.ToString(analisis.Th4s());
Ths_5.Text = Convert.ToString(analisis.Th5s());
qxs.Text = Convert.ToString(analisis.qxs());
qys.Text = Convert.ToString(analisis.qys());
qzs.Text = Convert.ToString(analisis.qzs());
rollas.Text = Convert.ToString(analisis.rollas());
rollbs.Text = Convert.ToString(analisis.rollbs());
pitchs.Text = Convert.ToString(analisis.pitchs());
```

Ahora que ya poseemos la información sobre los ángulos, posiciones y orientaciones tanto del maestro como del esclavo. Desplegamos la información en la parte grafica de la interfaz.

Figura7. Parte de la interfaz grafica del esclavo donde se desplegara la información de los ángulos, posiciones y orientaciones generadas.

A partir de este momento solo mostraremos el procedimiento para el eje 1 cuyo ángulo es también el 1, para evitar hacer más largo el análisis al tener información redundante de la ejecución.

Para el *gripper* es exactamente el mismo procedimiento, la única diferencia será que no se realizaran lecturas de *encoder*, solo se hará la ejecución en abierto o cerrado.

Comencemos con el procesamiento del ángulo 1 para transformarlo en datos *encoder* y darle una tolerancia al error de ubicación.

```
//Encoder 1; transformación a encoder del SCORBOT-ER 4u
enc1c = Convert.ToSingle(( analisis.Th1s()));
enc1c = Convert.ToSingle(0 + (((13092 - 0) * ((enc1c) - 0)) / (1.5708 - 0)));
enc1d = Convert.ToInt32(enc1c);
//Ponemos una tolerancia de %0.4
identificador1 = Convert.ToInt32(enc1d - 50);
identificador2 = Convert.ToInt32(enc1d + 50);
if (identificador1 < identificador2)
{
    enc1di = identificador1;
    enc1ds = identificador2;
}
else
{
    enc1di = identificador2;
    enc1ds = identificador1;
}
```


Una vez realizada la misma operación para los 5 ángulos restantes procedemos a ver si el punto se encuentra dentro de los límites del robot, de ser así continua el proceso en caso contrario mostrar una leyenda para indicárselo al usuario.

```
//Ubicamos si el angulo deseado esta dentro de los limites del robot.
if (enc1d < 23904 && enc1d > -18214 && enc2d < 15797 && enc2d > 723 && enc3d < 7200 && enc3d > -8083 && enc4d < 4335 && enc4d > -735 && enc5d < 5578 && enc5d > -4433)
{
    timer5.Enabled = true;
    puerto.DiscardInBuffer();
}
else
{
    txt_mensaje.Text = "Fuera de area de trabajo";
    SendMessage();
    txt_mensaje.Text = "";
}
}
```

El proceso continuará con el inicio del *Timer 5* que se ejecutara cada 500ms su código interno. En primer lugar dentro del *encoder* indicamos que eje se requiere mover empleando banderas. Es indispensable ya que indica si un eje ya llegó a la posición deseada o no, y también es responsable de cambiar el movimiento entre ejes una vez que el eje ha llegado a su destino.

```
private void timer5_Tick(object sender, EventArgs e)
{
    //Para mover un eje, se prende la bandera de movimiento en ese eje
    if (enc1 < enc1di || enc1 > enc1ds)
    {
        banderaenc1 = true;
        timer1.Enabled = true;
    }
}
```

Una vez encendida la bandera, se prende el *timer1*, el cual únicamente enviara un comando para adquirir el valor del *encoder* deseado, procesara la información y mostrara el valor del *encoder* en la interfaz grafica. Es importante resaltar que este proceso se realiza la transición entre modos del controlador, es decir de modo editor pasa a manual. Al final se apaga, y prende el *timer2*.

```
if (banderaenc1 == true)
{
    //Comando ACL para pedir el valor del encoder 1
    txtenviando.Text = "PRINT ENC[1]\r";
    enviar(txtenviando.Text);
}
}
```

```

//Recibimos la informacion sobre el encoder deseado
txtRecibir.Text = recibir();
printir = txtRecibir.Text;
//Ponemos un caracter que finaliza la cadena completa
printirt = String.Concat(printir, "\n");

printir = "";
for (a = 0; a < printirt.Length; a++)
{
    //Indica la ubicacion del inicio de la cadena
    if (printirt[a] == 'P')
    {
        b = a;
        banderaindice1 = true;
    }
}
//Indicamos las condiciones basicas de la cadena
if ((printirt.Length - b) > 12 && banderaindice1 == true)
{
    //Separamos el valor del encoder de lo demas
    for (a = b + 12; printirt[a] != '\n'; a++)
    {
        //Guardamos el valor del encoder en string
        print1 = string.Concat(print1, printirt[a]);
    }
    if (print1.Length > 1)
    {
        try
        {
            //Intentamos convertir el string en numero
            enc = Convert.ToInt32(print1);
            puerto.ReadExisting();
            //Entramos al modo "Manual"
            enviar("~\r");
            banderaindice2 = true;
        }
    }
}

if (banderaenc1 == true)
{
    //Mostramos el valor del encoder en la interfaz grafica
    enc1 = enc;
    txt_enc1.Text = print1;
    //Apagamos el Timer1 e iniciamos el 2
    timer2.Enabled = true;
    timer1.Enabled = false;
}

```

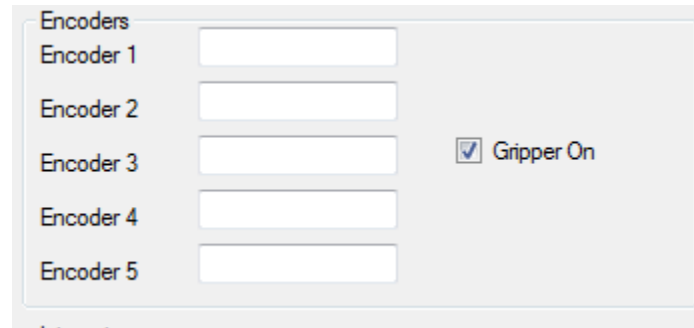


Figura8. Parte de la interfaz grafica del esclavo donde se desplegara la información de los encoders.

Al activarse el *timer2* solamente leerá basura bien definida y se apagará, antes prendiendo el *timer4*. Para continuar el proceso.

```
private void timer2_Tick(object sender, EventArgs e)
{
    //Este timer se encarga de leer una cadena especifica de basura
    if (puerto.IsOpen == true)
    {
        puerto.ReadTimeout = 1000;
        buffer1 = new char[1];
        puerto.Read(buffer1, j, 1);
        datos = new char[42];
        recepcion = Convert.ToString(puerto.Read(datos, j, k));
        txtRecibir.Text = recepcion;
        timer4.Enabled = true;
        timer2.Enabled = false;
    }
}
```

El *timer4* realiza la operación de enviar el comando ACL, para mover el eje en el sentido deseado. Y al final simplemente permite regresar al modo Editor del controlador. Y se apaga cuando prende el *timer3*.

```
private void timer4_Tick(object sender, EventArgs e)
{
    //Aqui se da el sentido a movimiento del eje
    if (enc1 < enc1di && banderaenc1 == true)
    {
        txtenviando.Text = "1\r";
        enviar(txtenviando.Text);
    }
    else if (enc1 > enc1ds && banderaenc1 == true)
    {
        txtenviando.Text = "Q\r";
        enviar(txtenviando.Text);
    }
}
```

```
//Leemos algun dato si existiese
puerto.ReadExisting();
enviar("~\r");
//Salimos del modo "Manual"
timer3.Enabled = true;
timer4.Enabled = false;
```

El *timer3* al igual que el *timer2* simplemente leerá basura bien definida en tamaño. Al final de la lectura se apaga.

Una vez que los cinco ejes han llegado a su posición deseada, se envían un mensaje vía TCP/IP que le indica al maestro el éxito de la operación y que se esperan instrucciones.

```
//Una vez listo para otro punto envia un caracter|
else
{
    txt_mensaje.Text = "R";
    SendMessage();
    txt_mensaje.Text = "";
}
```

Puntos en el controlador.

En primera instancia la cadena iniciara con Z, después se regenerará la cadena pero eliminando la Z del inicio. Se prendera el *timer10* que se encargará de mandar el comando ACL para grabar el punto y dará a conocer al usuario el éxito de la operación. Finalmente se apagará el *timer10*, y se activará el *timer11* que simplemente recogerá la basura que se genera de código inútil y finaliza el proceso.

```
else if (cadena_ip[0] == 'Z')//La cadena inicia con Z
{
    punto_grabado="";
    for (v = 1; v < (cadena_ip.Length - 1); v++)
        {//Este lazo se hace para obtener el nombre del punto.
            punto_grabado = String.Concat(punto_grabado, cadena_ip[v]);
        }
    if (puerto.IsOpen == true)
    {
        timer10.Enabled = true;
    }
}
```

```

private void timer10_Tick(object sender, EventArgs e)
{
    //Damos las instruccin en ACL para grabar el punto.
    ubicar="HERE ";
    puerto.ReadTimeout = 1000;
    ubicar=string.Concat(ubicar,punto_grabado,"\r");
    txtenviando.Text = ubicar;
    enviar(txtenviando.Text);
    puerto.ReadExisting();
    //Se indica el exito de la operacion.
    txt_mensaje.Text = "Punto grabado";
    SendMessage();
    txt_mensaje.Text = "";
    timer11.Enabled = true;
    timer10.Enabled = false;
}

private void timer11_Tick(object sender, EventArgs e)
{
    //Se recoge los datos basura.
    u = 5 + punto_grabado.Length + 9;
    datos2 = new char[u];
    recepcion = Convert.ToString(puerto.Read(datos2, j, u));
    txtRecibir.Text = recepcion;
    txt_mensaje.Text = "Punto grabado";
    SendMessage();
    txt_mensaje.Text = "";
    timer11.Enabled = false;
}

```

Trayectorias y programas (Comandos ACL).

En primera instancia la cadena iniciará con Y, después se regenerará la cadena pero eliminando la Y del inicio. Se mandará el comando ACL controlador y después se iniciará el *timer14*. Finalmente el *timer14* que simplemente recogerá la basura que se genera de código inútil, finaliza.

```

else if (cadena_ip[0] == 'Y')
{
    //La cadena empieza con Y
    instruccion = "";

    for (v = 1; v < (cadena_ip.Length - 1); v++)
    {
        //Se obtiene la instruccion
        instruccion = String.Concat(instruccion, cadena_ip[v]);
    }
}

```

```

else if (instruccion[0] == 'Y')
{ //Se envia la instruccion al controlador
    if (puerto.IsOpen == true)
    {

        instruccion = String.Concat(instruccion, "\r");
        txtenviando.Text = instruccion;
        enviar(txtenviando.Text);
        timer14.Enabled = true;
    }
}
}

private void timer14_Tick(object sender, EventArgs e)
{ //Se recoge la basura generada
    u = 82;
    datos2 = new char[u];
    recepcion = Convert.ToString(puerto.Read(datos2, j, u));
    txtRecibir.Text = recepcion;
    timer14.Enabled = false;
}
}

```

Programación para generar la geometría del punto en el robot “*Black Hawk*”.

La interfaz realiza exactamente las mismas conexiones que el anteriormente mostrado.

Se realizan exactamente las mismas operaciones iniciales del programa esclavo del SCORBOT-ER 4u. Sin embargo, una vez terminada la ubicación de la posición y la orientación tanto del maestro como del esclavo el código cambia de la siguiente forma:

```

//Se checa que los puntos sean diferentes al ultimo punto.
if (ang1r != ang1 || ang2r != ang2 || ang3r != ang3 || ang4r != ang4 || ang5r != ang5)
{
    //Se graba el nuevo angulo
    ang1r = ang1;
    ang2r = ang2;
    ang3r = ang3;
    ang4r = ang4;
    ang5r = ang5;
}

```

```

//Checamos si se encuentra dentro del area de trabajo
if (ang1 > 0 && ang1 < 3.142 && ang2 > -3.142 && ang2 < 0 && ang3 > -0.663 && ang3 < 2.478 && ang4 > -1.571 && ang4 < 1.571 && ang5 > -1.571 && ang5 < 1.571)
{
    //Arreglar la interseccion entre los angulos y y el modelo matematico.
    ang2 = -ang2;
    ang3 = -ang3 + 2.478F;
    ang4 = ang4 + 1.571F;
    ang5 = ang5 + 1.571F;

    //Aquí se arregla la posición del 0 con respecto al robot.
    ang1 = (0F + ang1) * 1000;
    ang2 = (3.142F - ang2) * 1000;
    ang3 = (0F + ang3) * 1000;
    ang4 = (0F + ang5) * 1000;
    ang5 = (0F + ang5) * 1000;
    //Redondeamos el angulo
    ang1e = Convert.ToInt32(ang1);
    ang2e = Convert.ToInt32(ang2);
    ang3e = Convert.ToInt32(ang3);
    ang4e = Convert.ToInt32(ang4);
    ang5e = Convert.ToInt32(ang5);
    //Convertimos el angulo en string
    ang1c = Convert.ToString(ang1e);
    ang2c = Convert.ToString(ang2e);
    ang3c = Convert.ToString(ang3e);
    ang4c = Convert.ToString(ang4e);
    ang5c = Convert.ToString(ang5e);
    //Enviamos la cadena
    cadena_mando = String.Concat(ang1c, ",", ang2c, ",", ang3c, ",", ang4c, ",", ang5c, ",");
    txtenviando.Text = cadena_mando;
    enviar(cadena_mando);
    recibir();
    //Enviamos caracter para indicar que la interfaz esta lista de nuevo.
    txt_mensaje.Text = "R";
    SendMessage();
}

```

Una vez enviada la cadena con la información de los ángulos que se desean reproducir en el Robot SCORBOT-ER 4u. La interfaz desarrollada por el estudiante de Ingeniería Mecatrónica Daniel Alberto Ortiz Pamanes para el proyecto “Interfaz 3D para teleoperación, visualización y comunicación” del Mechatronics Research Group para la tarjeta de desarrollo “Arduino Uno”. Será la encargada de enviar los PWM’s a los diferentes actuadores e impulsar los movimientos del robot.

Comunicación vía internet.

El programa C# contiene la librería System.Net.Sockets que permite el manejo e implementación de interfaces para desarrollar controles de acceso empleando redes.

Envío y recepción de video online.

Para realizar esta operación es fundamental desarrollar dos interfaces por separado, la primera será la interfaz del servidor, en este caso específico se anexará a la interfaz del maestro que se encargara de recibir el video, mientras que la interfaz del esclavo se encargará de enviar el video y será el cliente.

Interfaz del esclavo (Envío del video):

Para el envío de la imagen proveniente de la cámara web en primera instancia se realizará la transformación de la imagen en un arreglo de bits, para ser manipulados como un

Stream, en C# se utiliza la instrucción `MemoryStream()` para generar el *Stream* de él se transforma en el arreglo binario con la instrucción `GetBuffer()`;

```
ms = new MemoryStream();

byte[] arrImage = ms.GetBuffer();
```

Posteriormente serán mostrados en una pequeña ventana que permite visualizar la imagen adquirida directamente de la fuente, convertida en un *bitmap*.

```
data = Clipboard.GetDataObject();
if (data.GetDataPresent(typeof(System.Drawing.Bitmap)))
{
    bmap = ((Image)(data.GetData(typeof(System.Drawing.Bitmap))));
    bmap.Save(ms, ImageFormat.Bmp);
}
pantalla.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
```

Una vez terminado este proceso, se procede a realizar la comunicación con el servidor y se envía la cadena utilizando el protocolo TCP/IP. Esto se realiza con el método `Write()`, en combinación con el envío del *Stream* directamente a la red empleado un `NetworkStream`.

```
BinaryWriter mysw;

NetworkStream myns;

mysw = new BinaryWriter(myns);

mysw.Write(arrImage);
```

Lo anteriormente citado se muestra en el siguiente código utilizado para realizar la aplicación⁶:

⁶ Este código es una aplicación modificada del trabajo original de el libro "Professional Network, Distributed Systems & TCP/IP Programming In .NET Framework 1.1 & 2.0" de Fadi Abdelqader y su artículo publicado en <http://www.socketcoder.com/ArticlesRSS.aspx>


```

ms = new MemoryStream();// Almacenar la imagen en un arreglo
IDataObject data;
Image bmap;
// Copia de la imagen
SendMessage(hWnd, WM_CAP_EDIT_COPY, 0, 0);
// Conversión de la imagen en un bitmap
data = Clipboard.GetDataObject();
if (data.GetDataPresent(typeof(System.Drawing.Bitmap)))
{
    bmap = ((Image)(data.GetData(typeof(System.Drawing.Bitmap))));
    bmap.Save(ms, ImageFormat.Bmp);
}
pantalla.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
byte[] arrImage = ms.GetBuffer();
myclient = new TcpClient(remote_IP, port_number);//Conexion con el servidor
myns = myclient.GetStream();
mysw = new BinaryWriter(myns);
mysw.Write(arrImage);//Envio de la cadena al maestro
ms.Flush();
mysw.Flush();
myns.Flush();
ms.Close();
mysw.Close();
myns.Close();
myclient.Close();

```

Interfaz del maestro (Recepción del video):

En la recepción de la imagen se adquiere la cadena proveniente del protocolo TCP, en C# se utiliza el `NetworkStream()`, que permite adquirir la información del Stream una vez que se inicia el protocolo TCP.

```

ns = new NetworkStream(mysocket);

```

Esta cadena se transformará de nuevo en la imagen original y finalmente se mostrará en Picture Box. Para realizar esta operación, C# tiene una librería en `System.Drawing.Image()`, que permite el reacomodo de la imagen a partir de un arreglo binario.

```

imagen_recibida.Image = Image.FromStream(ns);

```

Aquí el código utilizado para la recepción del video en la interfaz del maestro.

```

// Abre el puerto
mytcp1 = new TcpListener(5000);
mytcp1.Start(); // Inicia comunicación
mysocket = mytcp1.AcceptSocket(); // Acepta al cliente
ns = new NetworkStream(mysocket); // Recibe el arreglo binario del cliente
imagen_recibida.Image = Image.FromStream(ns);
mytcp1.Stop(); // Cierra

if (mysocket.Connected == true) // Hace un loop hasta iniciar de nuevo el proceso
{
    while (true)
    {
        Start_Receiving_Video_Conference(); // Empieza de nuevo
    }
}
myns.Flush();

```

Envío y recepción de instrucciones vía TCP/IP:

Para realizar el envío y recepción de instrucciones a través del protocolo TCP/IP necesitamos al igual que en la operación de video desarrollar una interfaz servidor que en este caso será el maestro, y del otro lado tendremos una interfaz cliente que será el esclavo

Programación del Servidor (Maestro)⁷:

En primera instancia se ubica la interfaz grafica que se le presentara al usuario. Esta interfaz cuenta con tres botones. El primer botón "Recepcion de video" permite la conexión del cliente con el servidor, tambien permite la recepción del video. El segundo botón es el botón "Bye" que cierra el protocolo de comunicación TCP/IP. Y finalmente está el botón "Envío", que en caso de que se requiera hacer una prueba de conexión, se puede escribir una cadena en la caja de texto al lado del botón y enviarla al cliente.

La interfaz también muestra 2 cajas de texto aparte de la ya mencionada. La primera que indica "IP", es donde se escribe la IP de la computadora del servidor. Y la caja de texto "Recepcion" muestra el estado de toda la información proveniente de la conexión entre el cliente y el servidor.

En la parte superior se encuentra el picture box donde se visualiza la conexión de las imágenes provenientes del servidor.

Los métodos que describen cada acción que se realiza en el proceso se explicarán a detalle en el código más adelante.

⁷ Este código es una aplicación modificada del trabajo original de el artículo publicado en http://www.geekpedia.com/tutorial240_Csharp-Chat-Part-2---Building-the-Chat-Server.html

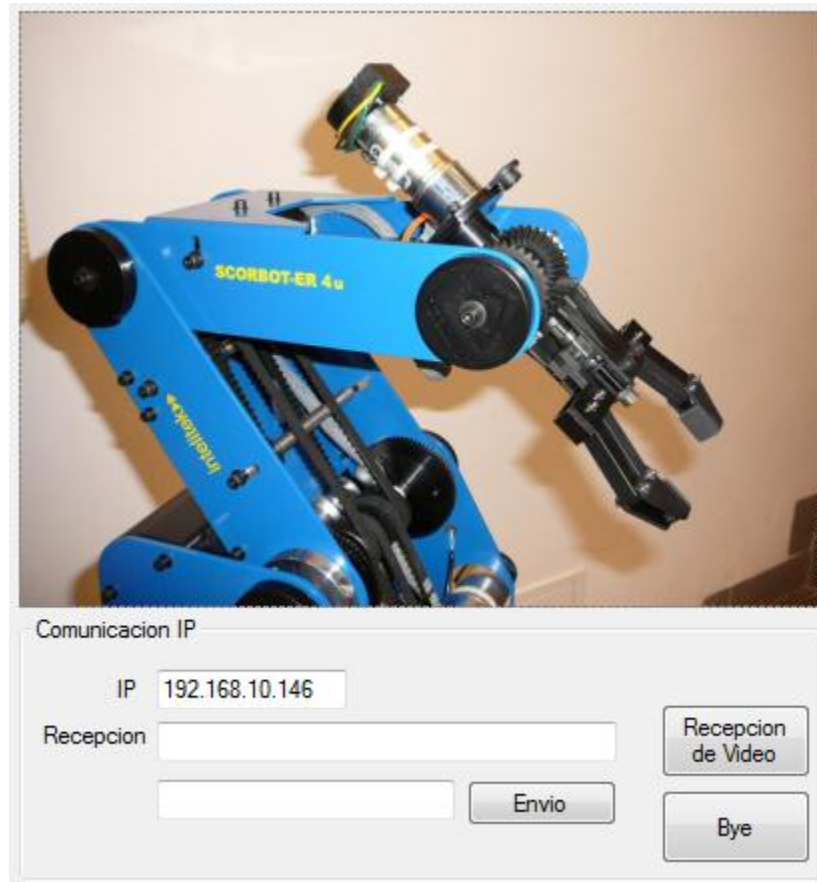


Figura9. Parte de la interfaz del maestro, encargada de seguir la comunicación TCP/IP.

En primera instancia establecemos el nombre de subprocesos que coadyuvaran durante la ejecución del programa. Estos subprocesos forman parte de las librerías `Systems.IO.Stream`, `System.Net.Sockets.TCP` y `System.Thread.Thread`.

```
TcpClient tcpClient;
// Nombres para envio y recepcion de informacion por parte del cliente
private Thread thrSender;
private StreamReader srReceiver;
private StreamWriter swSender;
private string currUser;
private string strResponse;
```

Establecemos la conexión con el cliente a través del método `Connection`, el cual aceptará el llamado del cliente y nos permitirá empezar a enviar y recibir información del mismo.

```

public Connection(TcpClient tcpCon)
{
    tcpClient = tcpCon;
    // Se acepta al cliente que espera conexion
    thrSender = new Thread(AcceptClient);
    // Llamamos al metodo de Aceptar al cliente
    thrSender.Start();
}

```

El método Connection, evalúa al cliente través del método AcceptClient(), que básicamente obtiene los datos del cliente para configurar su nombre de conexión, es decir, acepta o rechaza el nombre con el que será manejado el cliente. Una vez conectado guarda su nombre de cliente con la cual la información fluye del servidor al cliente y viceversa.

```

srReceiver = new System.IO.StreamReader(tcpClient.GetStream());
swSender = new System.IO.StreamWriter(tcpClient.GetStream());

// Lee la informacion del cliente
currUser = srReceiver.ReadLine();

// Obtenemos respuesta del cliente
if (currUser != "")

// Empezamos a recibir mensajes del cliente
ChatServer.AddUser(tcpClient, currUser);

// Se mantiene a la expectativa de mensajes
while ((strResponse = srReceiver.ReadLine()) != "")

// Ante algo invalido demueve al cliente
if (strResponse == null)
{
    ChatServer.RemoveUser(tcpClient);
}
else
{
    // Manda el mensaje
    ChatServer.SendMessage(currUser, strResponse);
}

```

Para recibir la información, se creó un método llamado StartListening() el cual tendrá la responsabilidad de iniciar y mantener las conexiones TCP. Generando conexiones con el método Connections() anteriormente citado. Básicamente obtiene la información de la IP local, y creando el puerto de comunicación que inicia el protocolo TCP. Y para que sigan

corriendo las demás funciones del programa se utiliza como una función de proceso en paralelo a través de un `System.Threading`.

```

public void StartListening()
{
    // Obtiene la IP
    IPAddress ipaLocal = ipAddress;

    // Crea una instancia TCP
    tlsClient = new TcpListener(1986);

    // Inicia protocolo TCP
    tlsClient.Start();

    // Corre servidor
    ServRunning = true;

    // Inicia proceso en paralelo
    thrListener = new Thread(KeepListening);
    thrListener.Start();
}

private void KeepListening()
{
    // El servidor esta corriendo
    while (ServRunning == true)
    {
        // Acepta conexiones
        tcpClient = tlsClient.AcceptTcpClient();
        // Crea las conexiones
        Connection newConnection = new Connection(tcpClient);
    }
}

```

Finalmente aparte de recibir información, el maestro requiere enviar las instrucciones al esclavo, a través de una operación simple. Al tener una conexión con el esclavo, el maestro simplemente se encarga de enviar una cadena de caracteres con el mismo protocolo utilizado desde el principio que es el TCP. Al igual que en el envío de video, la información la convertimos en un *Stream* empleando un `StreamWriter()`, que realizara una codificación en UTF-8. Y enviando la cadena con `swSenderSender` anterior mente mostrado en el método de conexión.

```

StreamWriter swSenderSender;//Codifica el mensaje para que se enviado via TCP

```

```
// Manda el mensaje al cliente, que es el unico elemento de la lista de conectados
swSenderSender = new StreamWriter(tcpClients[1].GetStream());
swSenderSender.WriteLine(Message);
swSenderSender.Flush();
swSenderSender = null;
```

Programación del cliente (esclavo):⁸

En primera instancia se ubica la interfaz grafica que se le presentará al usuario. Cuenta con una pequeña picture box, donde se muestran las imágenes que serán enviadas al servidor. También encontramos los botones “Inicio de Captura”, que permite la adquisición del video por parte de alguna fuente de video. Y “Fin de Captura” que termina con la adquisición de la imagen, además de ubicar en una lista el dispositivo al cual se conecta. Por otra parte se encuentra el botón “Enviar” , que en caso de que se requiera hacer una prueba de conexión, se puede escribir una cadena en la caja de texto al lado del botón y enviarla al cliente.

También se encuentran tres cajas de texto, la primera es “IP Remota” en donde se escribe la IP del servidor al cual se va a enviar la información. Y la caja de texto “Recepcion” muestra el estado de toda la información proveniente de la conexión entre el cliente y el servidor.



Figura10. Parte de la interfaz del esclavo, donde se da seguimiento a la comunicación TCP/IP.

En primera instancia se programó la conexión con el servidor (interfaz del maestro). Ubicamos en primer lugar nuestra IP objetivo, que es el servidor que nos administrará los movimientos, una vez realizado esto. Establecemos el puerto de conexión TCP y nos conectamos. Finalmente iniciamos el proceso paralelo para recibir mensajes. Para más información sobre la adquisición de la imagen ver anexo 6.

⁸ Este código es una aplicación modificada del trabajo original de el artículo publicado en http://www.geekpedia.com/tutorial240_Csharp-Chat-Part-1---Building-the-Chat-Server.html

```

//Declaracion de procesos
private StreamWriter swSender;
private StreamReader srReceiver;
private TcpClient tcpServer;
private delegate void UpdateLogCallback(string strMessage);
private delegate void CloseConnectionCallback(string strReason);
private Thread thrMessaging;
private IPAddress ipAddr;
private bool Connected;

// Ubicamos la ip objetivo del servidor
ipAddr = IPAddress.Parse(txt_ip.Text);
// iniciamos comunicacion TCP
tcpServer = new TcpClient();
tcpServer.Connect(ipAddr, 1986);

// Iniciamos el proceso paralelo para recibir mensajes
thrMessaging = new Thread(new ThreadStart(ReceiveMessages));
thrMessaging.Start();

```

Como parte del método `ReceiveMessages()`; lo primero que se tiene que programar es un proceso paralelo que permitirá mientras se encuentre conectado, recibir y actualizar la información que proviene del maestro. Y desplegarla en un textbox para posteriormente ejecutar las instrucciones.

Para decodificar la información proveniente del TCP, utilizamos el `srReceiver`, declarado anteriormente como el decodificador `StreamReader`.

```

while (Connected)
{
    // si estamos conectados se muestran mensajes en el text box
    this.Invoke(new UpdateLogCallback(this.UpdateLog), new object[] { srReceiver.ReadLine() });
}

// Al refrescar la información
private void UpdateLog(string strMessage)
{
    // Envía la información al text box
    txt_recepcion.Text = "";
    txt_recepcion.AppendText(strMessage + "\r\n");
}

```

Finalmente, para enviar mensajes desde el esclavo hacia el maestro se programó el método `SendMessage()`, que emplea el `swSender`, declarado anteriormente como una operación que codifica la información como un `Stream` y la envía vía TCP.

```

// Envío de mensajes
private void SendMessage()
{
    if (txt_mensaje.Lines.Length >= 1)
    {
        swSender.WriteLine(txt_mensaje.Text);
        swSender.Flush();
        txt_mensaje.Lines = null;
    }
    txt_mensaje.Text = "";
}

```

Experimentación y resultados con el sistema teleoperado.

El hardware es lo que puedes patear, y el software lo que puedes maldecir.

En esta sección se presentarán los datos obtenidos en las pruebas experimentales realizadas.

Para su mejor comprensión, se realizaron dos experimentos de teleoperación en el Robot SCORBOT-ER 4u y un experimento mas para mostrar la adaptabilidad de la interface desarrollada, el primero que solamente utilizó las lecturas de *encoders* con cinemática directa y el segundo que utilizó cinemática directa e inversa, en este ultimo también se realizó un ejercicio de teleprogramación y telemonitoreo. Mientras que en el experimento final se utilizo la interfaz anteriormente desarrollada para realizar la teleoperación de un robot controlado por servomotores de aeromodelismo, nombrado como "*Black Hawk*".

La prueba de Teleoperación utilizando únicamente las lecturas de los *encoders* con cinemática directa se dividió en 2 etapas:

1. Ángulos con orientaciones adquiridas y ejecutadas por la interfaz maestro-esclavo.
2. Ubicación espacial de los puntos tanto en el maestro como en el esclavo.

Para la prueba de Teleoperación usando ecuaciones cinemáticas tanto directa como inversa, se dividió en 6 etapas que consisten en:

1. Adquisición de datos del dispositivo instrumentado.
2. Retrasos en la imagen de la cámara en el maestro cuando proviene del esclavo empleando al internet en la tapa de la teleoperación.
3. Geometrías reproducidas por el esclavo.
4. Proceso funcional completo.
5. Retrasos en la imagen en el maestro cuando proviene del esclavo empleando al internet en la tapa de la teleprogramación y el telemonitoreo.

En tanto para la prueba en el robot "*Black Hawk*", se adquirió la posición y orientación de los puntos generados por el maestro y los ángulos reproducidos por el dispositivo esclavo.

En esta sección se muestra los resultados experimentales al adquirir simplemente los datos de los *encoders* del pantógrafo maestro instrumentado, con el fin de reproducción fielmente dichos ángulos en el esclavo SCORBOT-ER 4u.

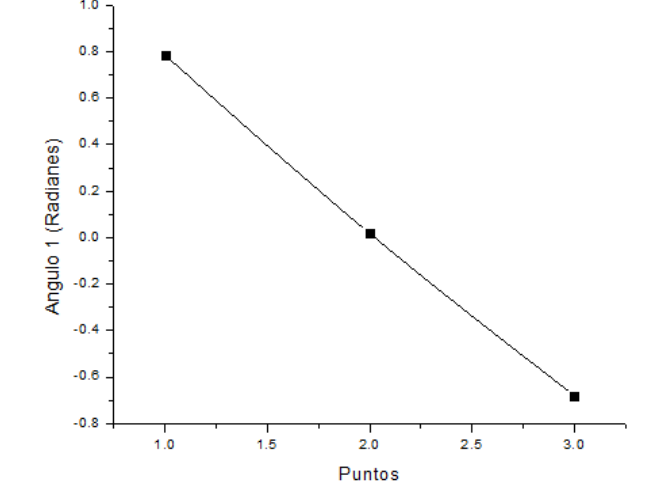
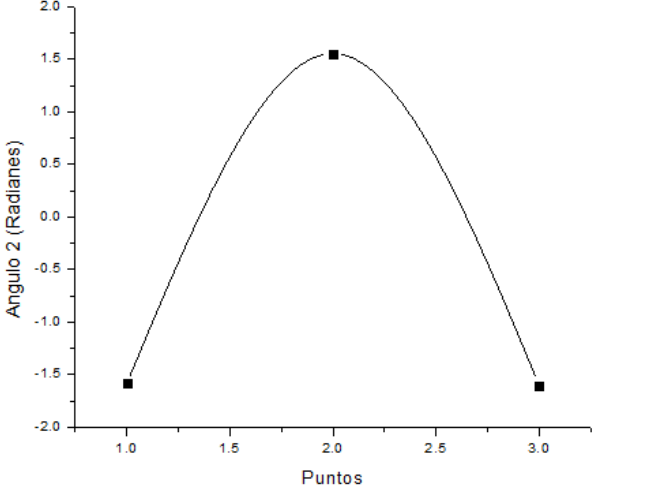
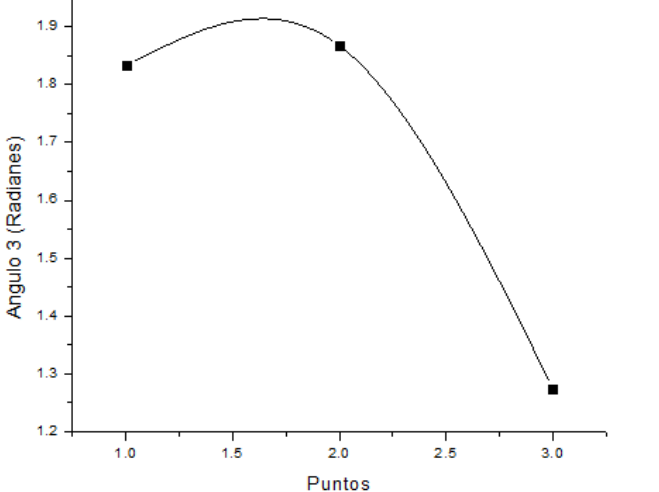
Condiciones iniciales de los experimentos:

- El *Controller A* se encontraba prendido, y conectado tanto al Robot SCORBOT-ER 4u como también a la interfaz esclavo empleando una laptop. También se encontraba por programación en modo EDIT.
- La interfaz esclavo utilizo la cámara web integrada a la laptop para adquisición de imágenes.
- La laptop donde se utilizo el programa esclavo tenía una conexión alámbrica Ethernet.
- El PAMCA estaba conectado a una PC. Que a su vez contenían la interfaz maestro y estaba conectada a internet mediante cable Ethernet.

Experimentación y resultados de la interfaz maestro.

Ángulos y orientaciones adquiridas y ejecutadas por la interfaz maestro-esclavo.

Los ángulos y las orientaciones de los ejes tanto del maestro como en el esclavo tienen que ser forzosamente los mismos debido a la paridad geométrica que presenta tanto el pantógrafo de 5 grados de libertad con respecto al SCORBOT-ER 4u que también tiene 5 grados de libertad. Mismo número de eslabones y mismo número de ejes, lo cual corrobora la afirmación anteriormente descrita. Los resultados obtenidos experimentalmente son los siguientes al ubicar tres puntos en el espacio:

Datos numéricos.	Representación Gráfica.																
<table border="1"> <thead> <tr> <th>Puntos</th> <th>Th1 (Rad)</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>0.7854</td> </tr> <tr> <td>2</td> <td>0.01745333</td> </tr> <tr> <td>3</td> <td>-0.68068</td> </tr> </tbody> </table>	Puntos	Th1 (Rad)	1	0.7854	2	0.01745333	3	-0.68068	 <p>Gráfico de Angulo 1 (Radianes) vs Puntos. El eje horizontal (Puntos) va de 1.0 a 3.0. El eje vertical (Angulo 1 (Radianes)) va de -0.8 a 1.0. Se muestran tres puntos conectados por una línea:</p> <table border="1"> <thead> <tr> <th>Puntos</th> <th>Angulo 1 (Radianes)</th> </tr> </thead> <tbody> <tr> <td>1.0</td> <td>0.7854</td> </tr> <tr> <td>2.0</td> <td>0.01745333</td> </tr> <tr> <td>3.0</td> <td>-0.68068</td> </tr> </tbody> </table>	Puntos	Angulo 1 (Radianes)	1.0	0.7854	2.0	0.01745333	3.0	-0.68068
Puntos	Th1 (Rad)																
1	0.7854																
2	0.01745333																
3	-0.68068																
Puntos	Angulo 1 (Radianes)																
1.0	0.7854																
2.0	0.01745333																
3.0	-0.68068																
<table border="1"> <thead> <tr> <th>Puntos</th> <th>Th2 (Rad)</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>-1.588253</td> </tr> <tr> <td>2</td> <td>1.553347</td> </tr> <tr> <td>3</td> <td>-1.605707</td> </tr> </tbody> </table>	Puntos	Th2 (Rad)	1	-1.588253	2	1.553347	3	-1.605707	 <p>Gráfico de Angulo 2 (Radianes) vs Puntos. El eje horizontal (Puntos) va de 1.0 a 3.0. El eje vertical (Angulo 2 (Radianes)) va de -2.0 a 2.0. Se muestran tres puntos conectados por una curva:</p> <table border="1"> <thead> <tr> <th>Puntos</th> <th>Angulo 2 (Radianes)</th> </tr> </thead> <tbody> <tr> <td>1.0</td> <td>-1.588253</td> </tr> <tr> <td>2.0</td> <td>1.553347</td> </tr> <tr> <td>3.0</td> <td>-1.605707</td> </tr> </tbody> </table>	Puntos	Angulo 2 (Radianes)	1.0	-1.588253	2.0	1.553347	3.0	-1.605707
Puntos	Th2 (Rad)																
1	-1.588253																
2	1.553347																
3	-1.605707																
Puntos	Angulo 2 (Radianes)																
1.0	-1.588253																
2.0	1.553347																
3.0	-1.605707																
<table border="1"> <thead> <tr> <th>Puntos</th> <th>Th3 (Rad)</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1.8326</td> </tr> <tr> <td>2</td> <td>1.867507</td> </tr> <tr> <td>3</td> <td>1.274093</td> </tr> </tbody> </table>	Puntos	Th3 (Rad)	1	1.8326	2	1.867507	3	1.274093	 <p>Gráfico de Angulo 3 (Radianes) vs Puntos. El eje horizontal (Puntos) va de 1.0 a 3.0. El eje vertical (Angulo 3 (Radianes)) va de 1.2 a 1.9. Se muestran tres puntos conectados por una curva:</p> <table border="1"> <thead> <tr> <th>Puntos</th> <th>Angulo 3 (Radianes)</th> </tr> </thead> <tbody> <tr> <td>1.0</td> <td>1.8326</td> </tr> <tr> <td>2.0</td> <td>1.867507</td> </tr> <tr> <td>3.0</td> <td>1.274093</td> </tr> </tbody> </table>	Puntos	Angulo 3 (Radianes)	1.0	1.8326	2.0	1.867507	3.0	1.274093
Puntos	Th3 (Rad)																
1	1.8326																
2	1.867507																
3	1.274093																
Puntos	Angulo 3 (Radianes)																
1.0	1.8326																
2.0	1.867507																
3.0	1.274093																

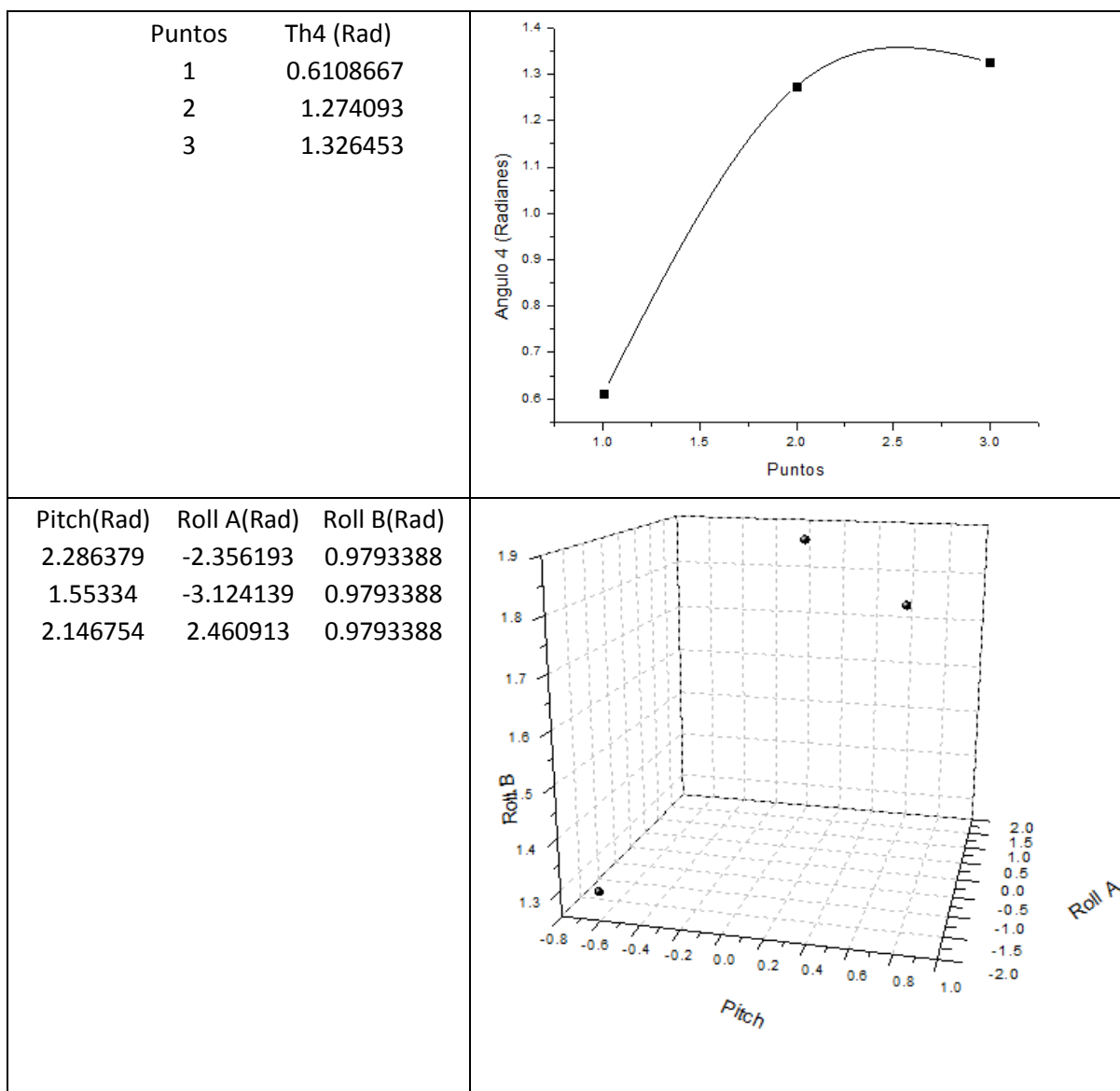


Figura 1. Recorrido del los ejes y orientación del punto.

Los datos anteriormente mostrados fueron obtenidos mediante la cinemática directa realizada tanto al pantógrafo como al SCORBOT-ER 4u. Y corroborados físicamente con el uso de un flexometro. Los errores entre el punto medido físicamente y el obtenido con la interface son prácticamente iguales, la máxima variación fue en el orden de 2 mm.

Ubicación espacial de los puntos tanto en el maestro como en el esclavo.

El verdadero cambio que se percibe en este primer experimento se presentó en la ubicación de las coordenadas cartesianas en los ejes X, Y y Z. Que van a ser totalmente diferentes en el esclavo como en el maestro, debido a la diferencia del tamaño de los

eslabones en ambos dispositivos. Es decir, si fueran cinemáticamente iguales, las dimensiones también lo serían, sin embargo debido a las diferencias del tamaño en sus eslabonamientos; la ubicación del efector final será diferente. Los resultados obtenidos experimentalmente fueron los siguientes:

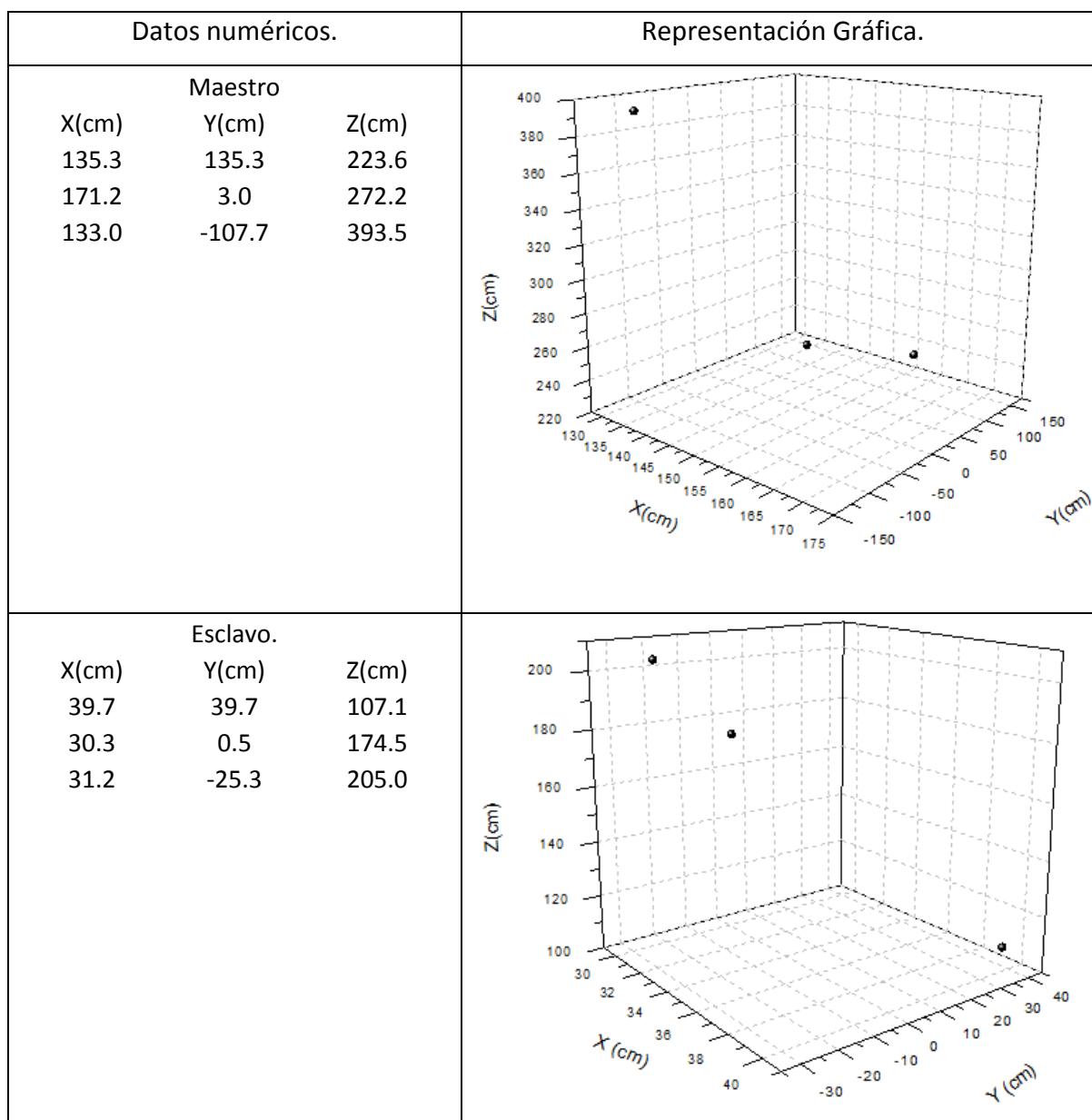


Figura 2. Ubicación dentro del sistema de referencia.

La información presentada corrobora la afirmación anteriormente mencionada. Los datos fueron adquiridos primeramente con las ecuaciones cinemáticas directas tanto del maestro como del esclavo. Y al igual que el ejercicio anterior fue corroborada físicamente con un flexometro.



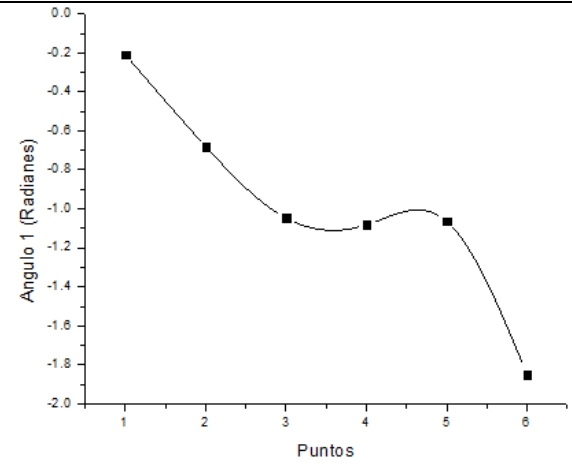
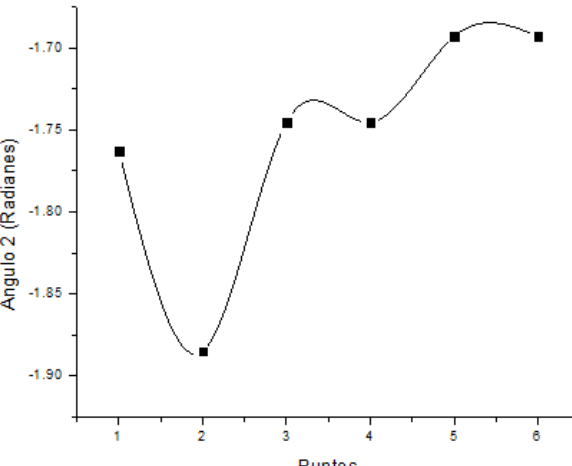
Figura 3. Relación espacial entre el maestro y el esclavo.

Una vez realizado el experimento donde se puede ubicar visualmente la relación física existente entre ambos dispositivos que conforman el ambiente de trabajo, y corroboramos que se encuentra la relación cinemática entre ambos dispositivos. Podemos pasar a la siguiente etapa de experimentación que consistirá básicamente en cargar las ecuaciones cinemáticas directas en el maestro mientras que le cargaremos las ecuaciones cinemáticas inversas al esclavo. Para hacer que ambos dispositivos conserven tanto la misma orientación como la misma posición en el sistema de referencia usado. Se realizó un análisis físico con una inspección visual, y después de corroborar nuestros experimentos con un flexometro, se garantizó la veracidad del análisis teórico con el análisis físico.

Adquisición del dispositivo instrumentado.

Como primera parte del análisis de los resultados del experimento realizaremos un seguimiento de las posiciones de los puntos en el sistema de referencia utilizado por el dispositivo maestro. Analizaremos uno a uno los ejes y sus recorridos para llegar a los puntos deseados.

Los resultados obtenidos fueron los siguientes:

Datos numéricos.		Representación Gráfica.
Puntos	Th1(Rad)	
P1	-0.20944	
P2	-0.68068	
P3	-1.0472	
P4	-1.082107	
P5	-1.064653	
P6	-1.850053	
Puntos	Th2 (Rad)	
P1	-1.762787	
P2	-1.88496	
P3	-1.745333	
P4	-1.745333	
P5	-1.692973	
P6	-1.692973	

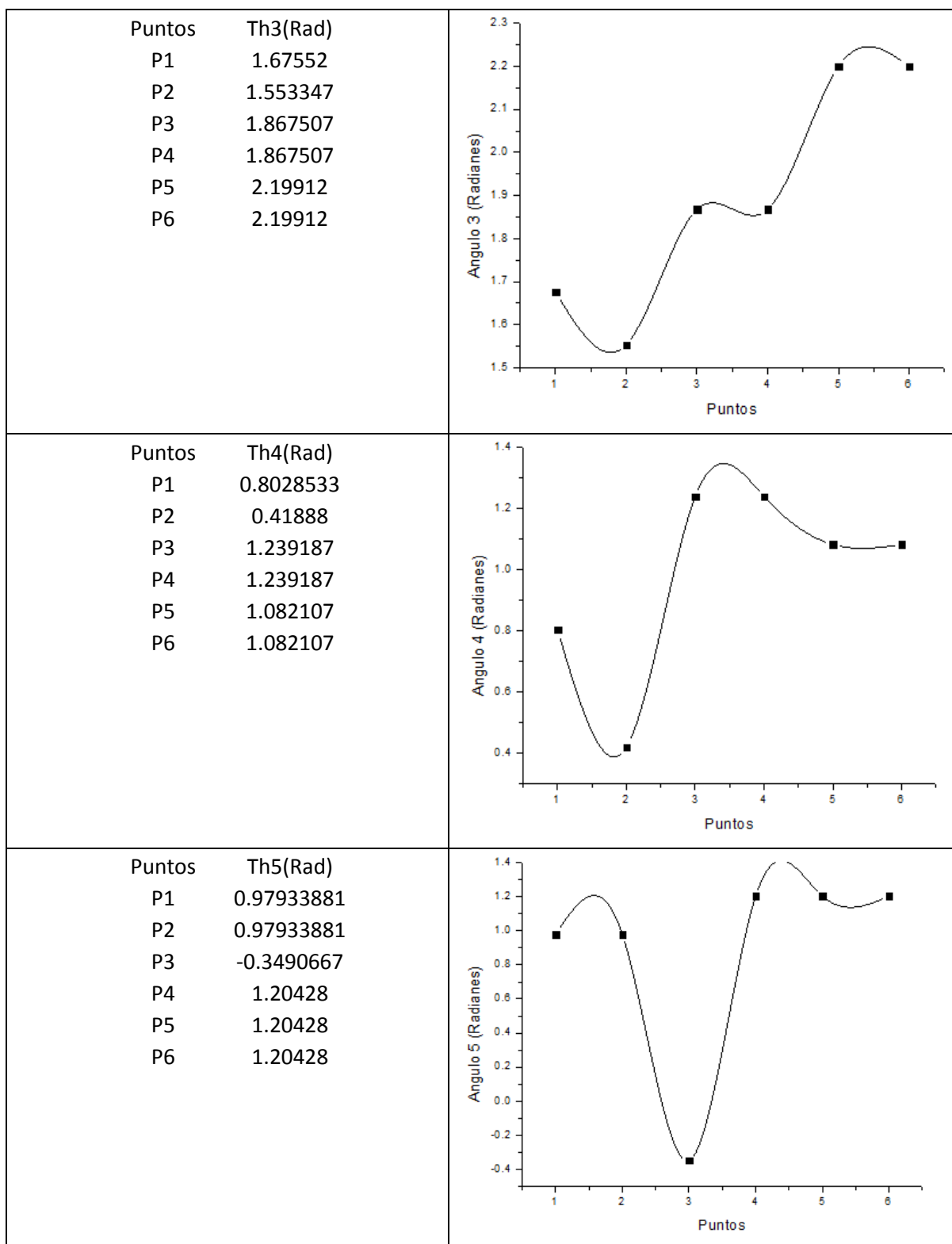


Figura 4. Puntos y ángulos del máster

Ahora se analiza la posición del punto en el espacio, con respecto al eje de referencia central, ubicado en el origen:

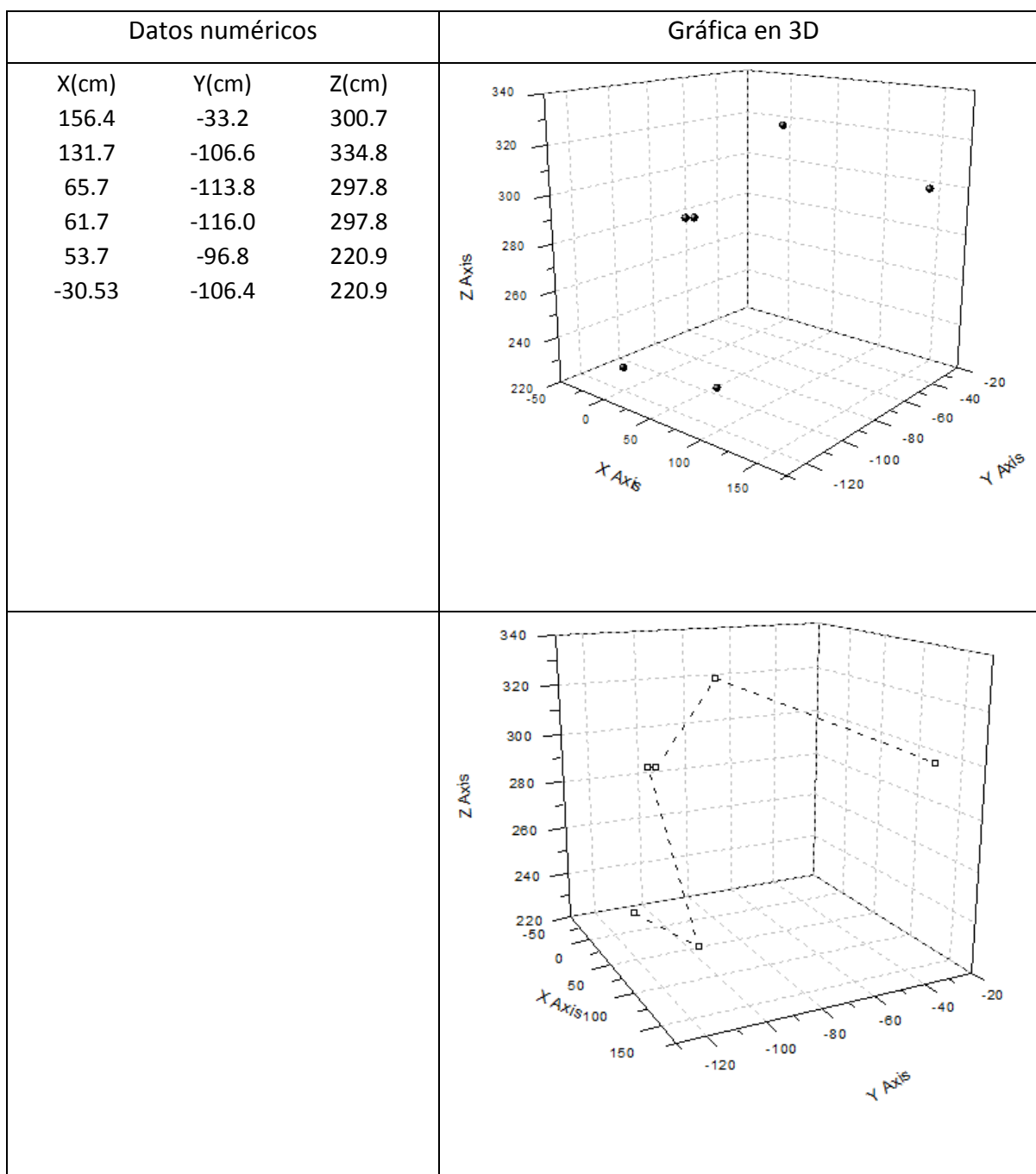


Figura 5. Ubicación de puntos en el espacio

Los datos anteriormente mostrados fueron de primera mano generados por el análisis teórico utilizando las ecuaciones de cinemática directa en el maestro. Fueron

corroboradas con el uso de un flexometro, y no se detectaron grandes diferencias entre el resultado teórico y el real.

Para terminar con la presentación grafica de los datos obtenidos con el dispositivo pantógrafo maestro de cadena abierta, ubicaremos la orientación de los puntos:

Pitch(Rad)	Roll A(Rad)	Roll B(Rad)
2.426006	2.932153	2.162254
3.054326	2.460913	2.162254
1.780232	2.094393	-3.106686
1.780232	2.059486	1.937313
1.553339	2.07694	1.937313
1.553339	1.29154	1.937313

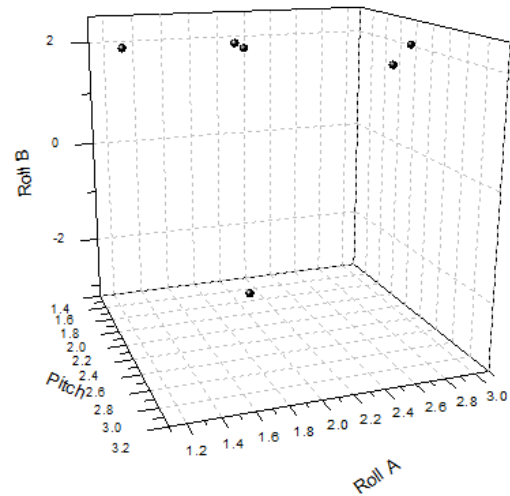


Figura 6. Orientaciones.

La gráfica anteriormente mostrada señala los cambios en la orientación del punto, medidas en radianes, con respecto al origen del sistema de referencia.



Figura 7. Mediciones espaciales.

A los elementos que tienen que ver con la cinemática y la geometría de los puntos en el espacio de trabajo se le tiene que agregar un elemento plenamente relacionado con la interface desarrollada, este elemento es el tiempo de atraso presente en la adquisición de la lectura de los *encoders* procedente del ArduinoMega empleando la comunicación serial RS232. Dicho retraso fue medido con un cronometro y los resultados fueron los siguientes.

Tiempo de retraso para adquisición de puntos.

Puntos	Tiempo de adquisición por punto (s)
1	15
2	132
3	197
4	236
5	105
6	188

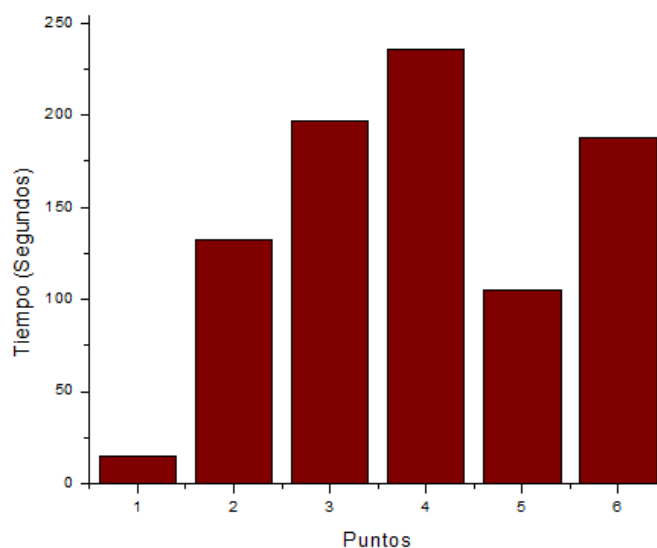


Figura 8. Tiempos de retardo en adquisición de los puntos.

El siguiente elemento a analizar fue el tiempo de retraso presente en la adquisición de la imagen procedente de la cámara web fija en el esclavo, y la recepción de la información en el maestro.

Dicha información fue medida con un cronometro durante diez minutos mientras se realizaban movimientos entre los puntos presentados anteriormente.

Retrasos en la imagen de la cámara en el maestro cuando proviene del esclavo empleando al internet en la tapa de la teleoperación.

No. de cambio registrado en el Picture Box.	Tiempo entre cambio de frames. (s)	No. de cambio registrado en el Picture Box.	Tiempo entre cambio de frames. (s)	No. de cambio registrado en el Picture Box.	Tiempo entre cambio de frames. (s)
1	6	14	25	27	33
2	5	15	19	28	31
3	7	16	26		
4	23	17	24		
5	16	18	35		
6	22	19	27		
7	32	20	22		
8	6	21	16		
9	4	22	12		
10	15	23	19		
11	23	24	26		
12	24	25	29		
13	21	26	31		

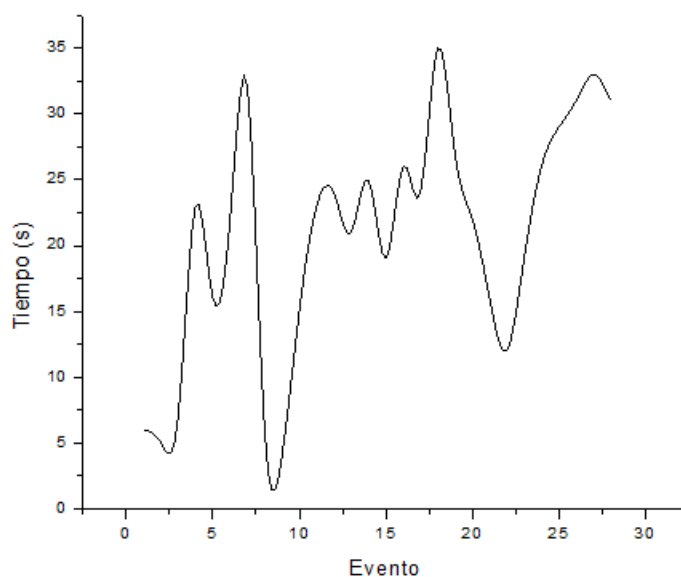


Figura 9. Tiempo de retraso entre cambio de frames.

Experimentación y resultados de la interfaz esclavo con el SCORBOT-ER 4u.

En esta etapa, la presentación gráfica se mostrará el recorrido en los ejes del esclavo para cumplir con la relación cinemática inversa proveniente de la interfaz con el maestro.

Datos numéricos.		Representación Gráfica.
Datos	Th1 Esclavo	
P1	-0.20944	
P2	-0.68068	
P3	-1.0472	
P4	-1.082107	
P5	-1.064653	
P6	1.29154	
Datos	Th2 Esclavo	
P1	-1.196345	
P2	-1.679779	
P3	-0.8555228	
P4	-0.8555228	
P5	-0.4221787	
P6	0.02515091	

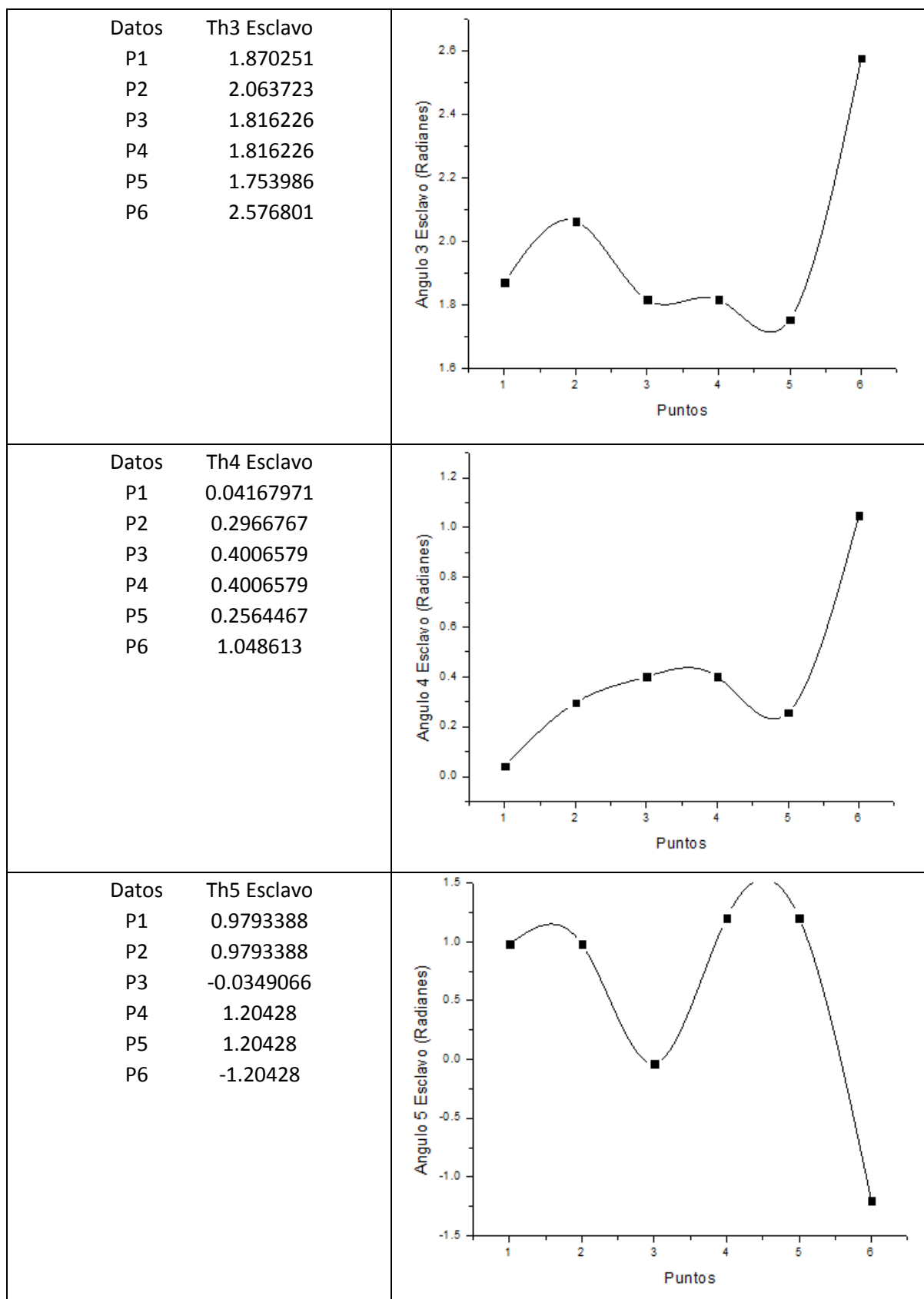


Figura 10. Ángulos del dispositivo esclavo en cada ejecución.

En la siguiente figura se mostrará la ubicación de los puntos realizados por el dispositivo esclavo con respecto al sistema de referencia usado en la interfaz misma.

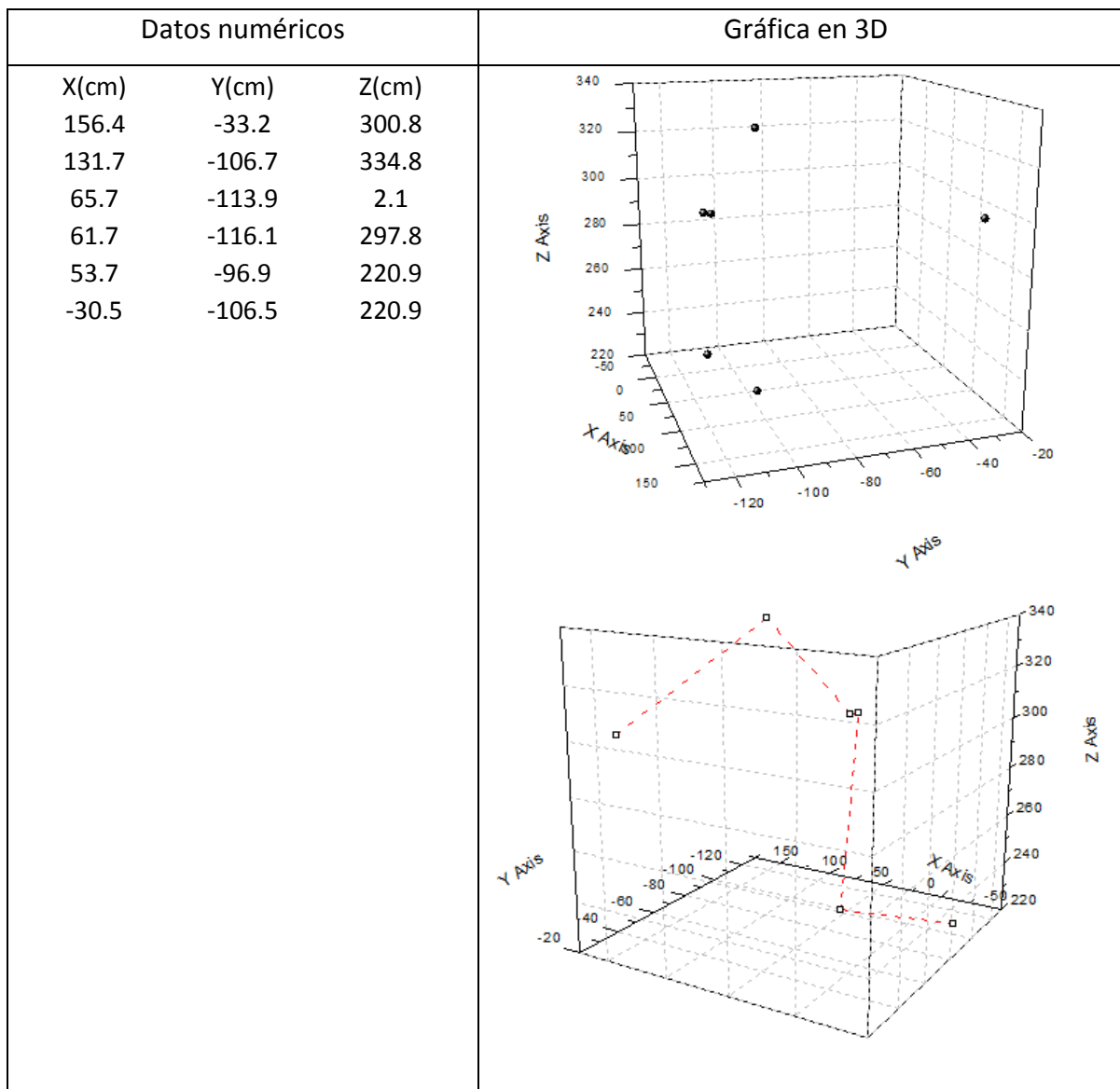


Figura 11. Ubicación de los puntos en el espacio por el esclavo.

Los datos anteriormente obtenidos teóricamente fueron validados físicamente al medirlos con un flexometro. Y no se mostraron diferencias muy marcadas entre ambos elementos.

Para completar el análisis de la geometría de los puntos solo falta analizar la orientación, que se muestra en la siguiente figura:

Pitch(Rad)	Roll A (Rad)	Roll B(Rad)
2.426006	2.932153	2.162254
3.054326	2.460913	2.162254
1.780232	2.094393	-3.106686
1.780232	2.059486	1.937313
1.553339	2.07694	1.937313
1.588254	-1.850053	1.937313

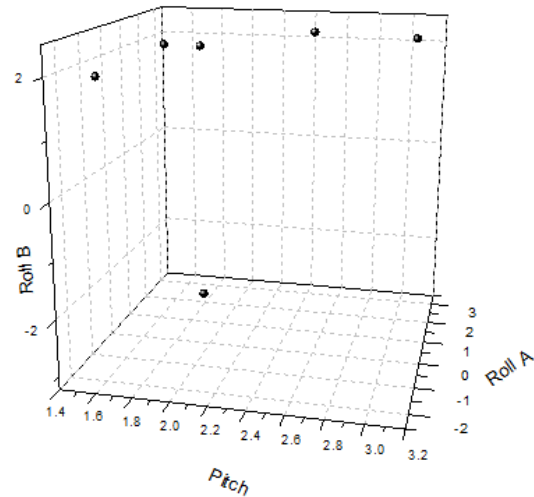


Figura 12. Orientación del punto en el esclavo.

Finalmente para concluir el análisis de los resultados obtenidos durante la experimentación de la teleoperación del SCORBOT-ER 4u, mostraremos el análisis de los tiempos de retraso generados entre movimiento y movimiento durante el proceso de cambio entre el punto 3 y 4 anteriormente mostrados. Este análisis es plenamente perteneciente a la interfaz del esclavo.

Estos resultados fueron adquiridos con el uso de un cronometro y los resultados fueron los siguientes:

Movimiento registrado en algún eje.	Tiempo (s).	Movimiento registrado en algún eje.	Tiempo (s).
1	1	11	3
2	1	12	1
3	3	13	1
4	5	14	1
5	3	15	3
6	2	16	4
7	1	17	3
8	1	18	1
9	4	19	2
10	2		

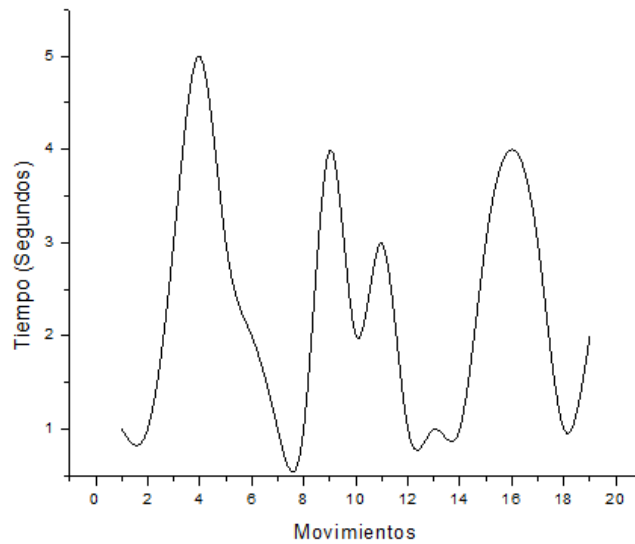


Figura 13. Tiempo de espera entre movimientos.

Proceso funcional completo.

El tiempo total entre movimientos partiendo del punto "HOME" tanto del esclavo como del maestro, dio como resultado la siguiente tabla generada a partir de un cronometro.

Proceso funcional completo	Tiempo (Mins)
P1	12:36.3
P2	18:24.8
P3	16:34.5
P4	00:42.9
P5	08:16.4
P6	10:12.1

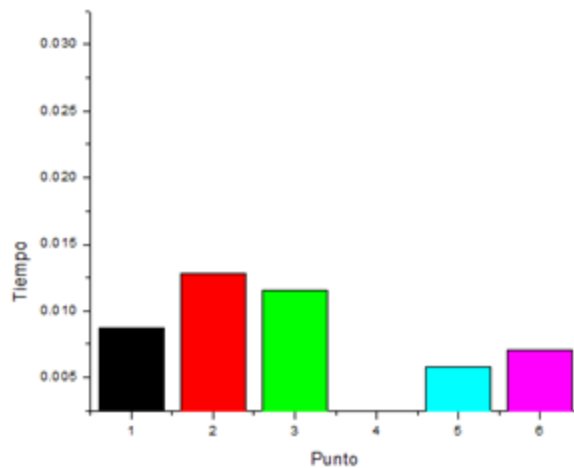


Figura 14. Tiempo de espera por ejecución.

Una vez terminada la presentación de los resultados provenientes de la etapa de Teleoperación. Ahora presentaremos los resultados provenientes del retraso del video generado durante el proceso de teleprogramación y telemonitoreo.

Estos resultados se obtuvieron al realizar la teleprogramación y la posterior ejecución de un programa para el SCORBOT-ER 4u, empleando un contador de eventos y un cronometro. Los resultados son los siguientes:

Retrasos en la imagen en el maestro cuando proviene del esclavo empleando al internet en la tapa de la teleprogramación y telemonitoreo.

Minuto	Cambios registrados en el Picture Box.
1	8
2	12
3	18
4	14
5	12
6	10
7	18
8	14
9	9
10	18

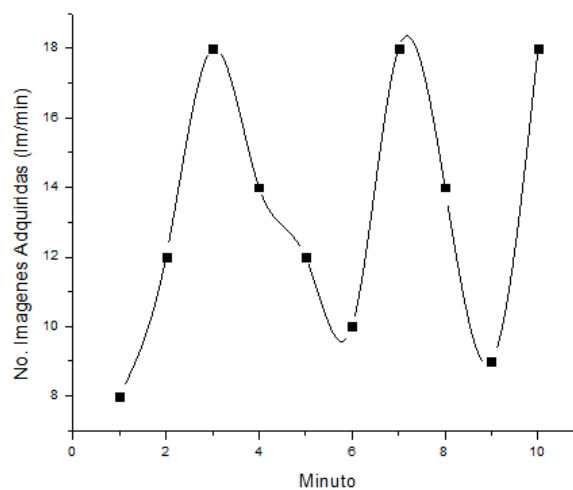


Figura 15. Relación imágenes por minuto durante la teleprogramación y telemonitoreo.

Finalmente, las figuras que a continuación se muestran presentan los resultados obtenidos a partir de la aplicación de la interface desarrollada a un robot con un control totalmente diferente, basado en la comunicación Serial con un "Arduino Uno" que ejecuta la interfaz con los servomotores que realizan los movimientos de las articulaciones de este dispositivo.

Aplicación de la interfaz en el robot “*Black Hawk*”.

Trayectoria y orientaciones del recorrido por parte del pantógrafo para la interfaz del maestro.

En esta etapa de la presentación gráfica, se mostrará el recorrido del efector final para el pantógrafo de la interfaz del maestro para cumplir con la generación de una trayectoria aleatoria de puntos.

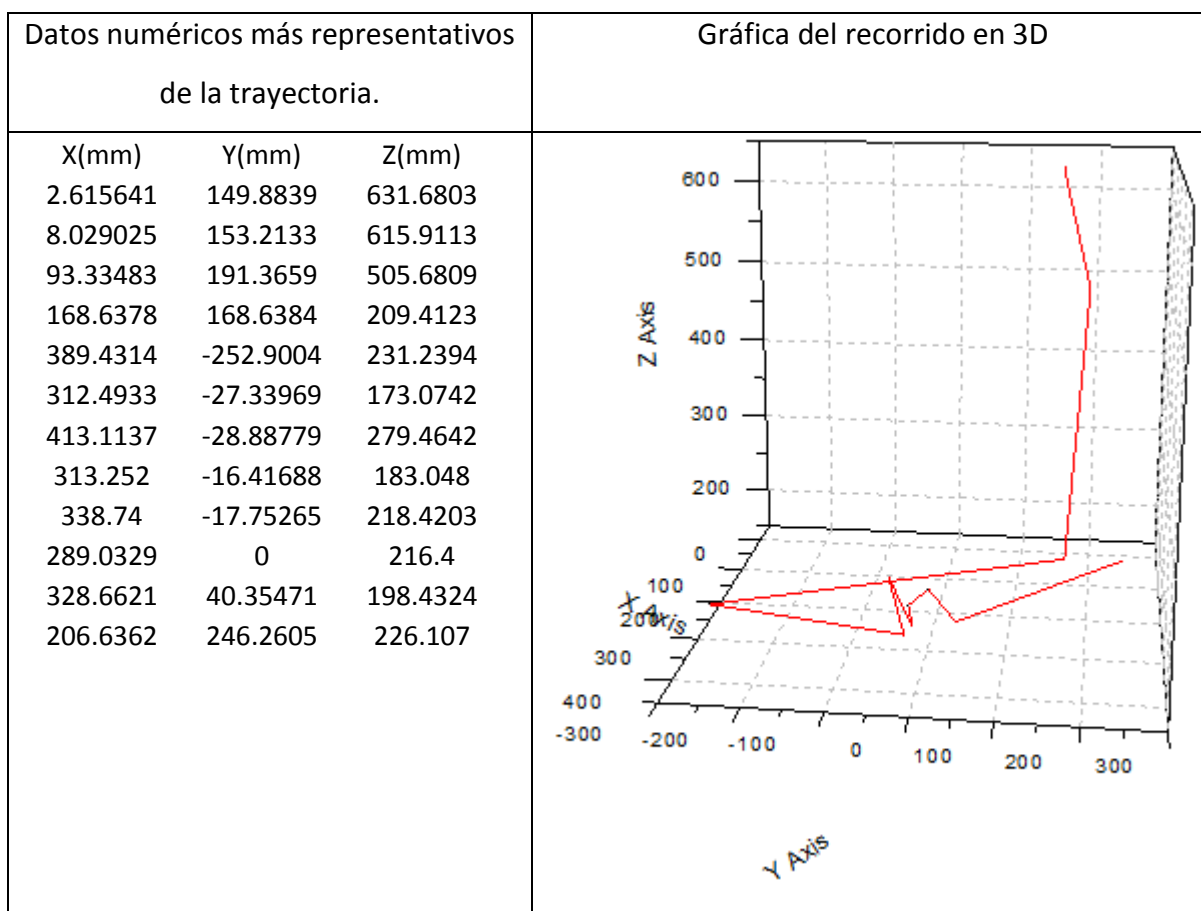


Figura16.Recorrido del PAMCA.

Ahora se presentaran las orientaciones de los puntos más representativos anteriormente citados:

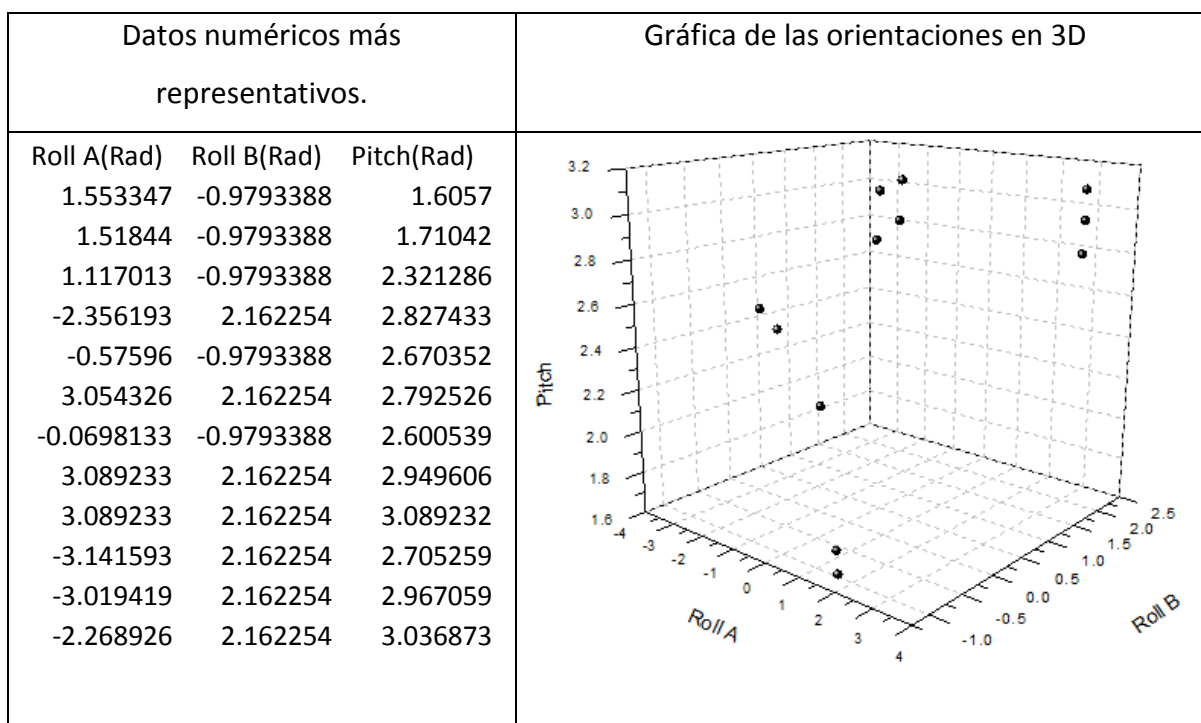
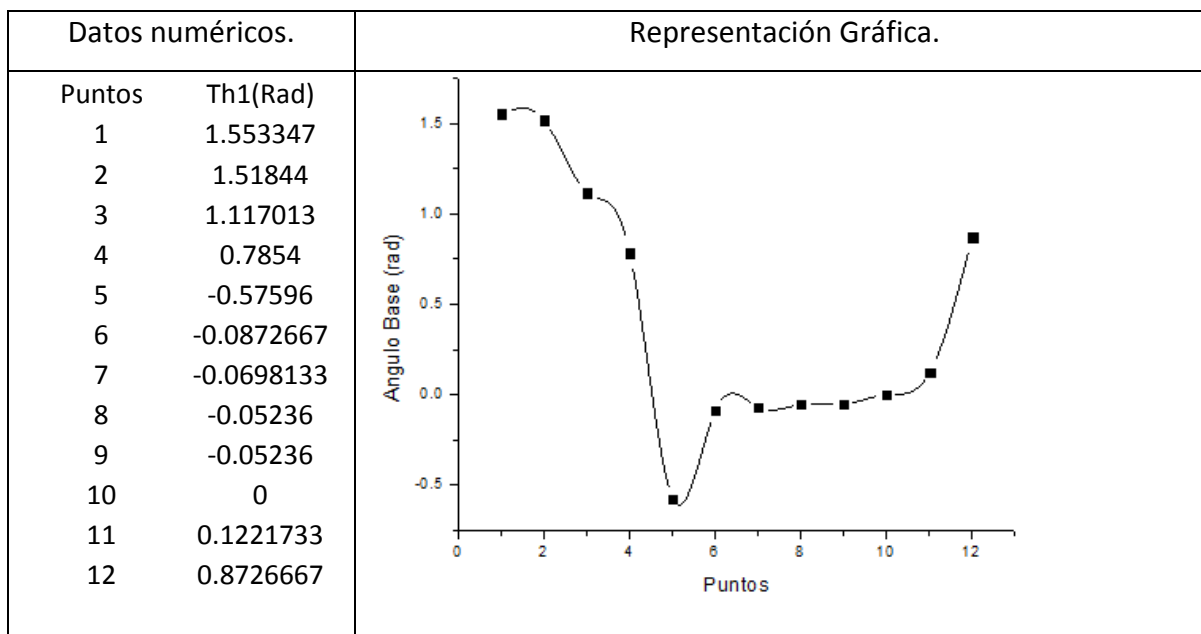
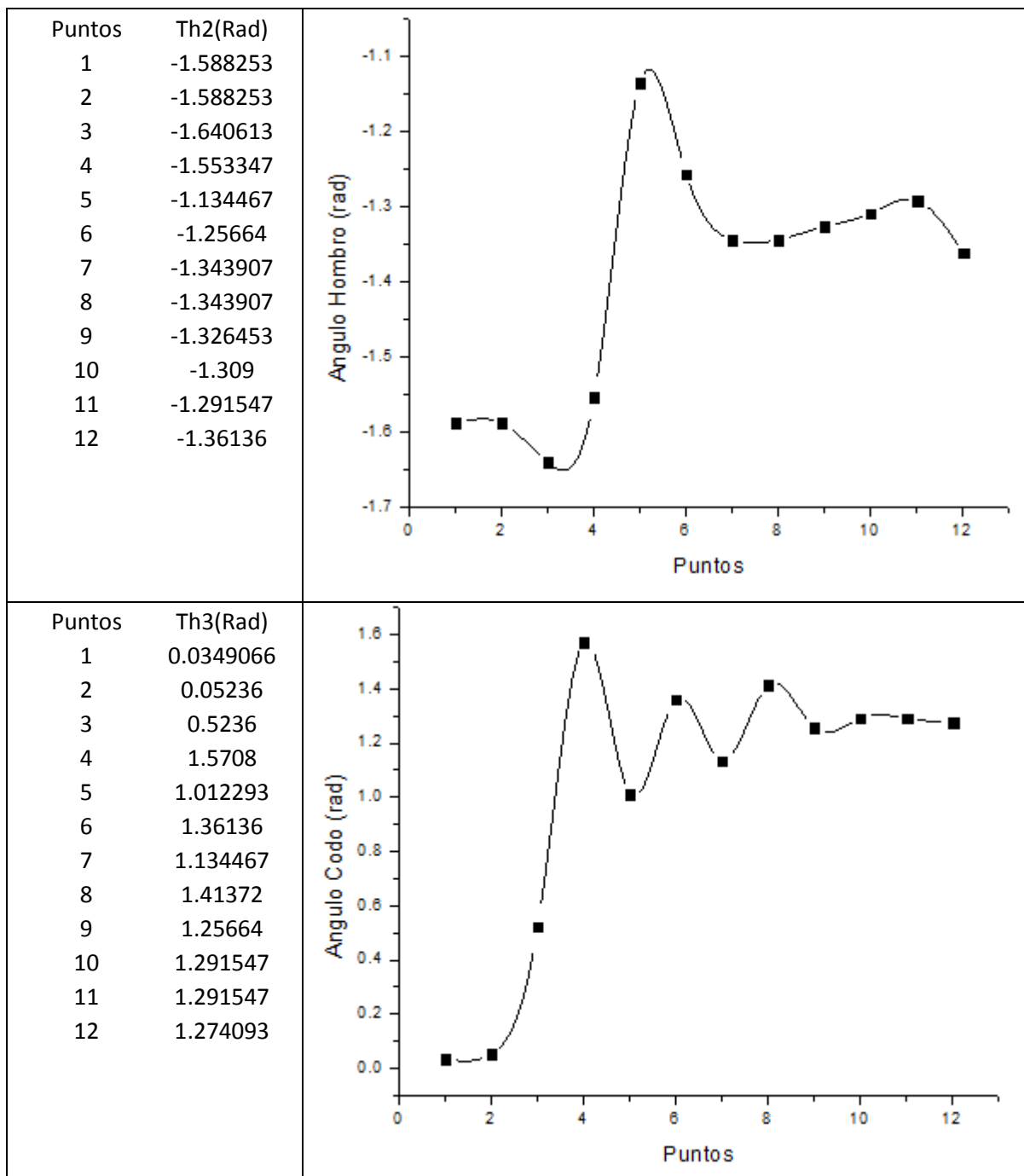


Figura17. Orientaciones.

A partir de los puntos y las orientaciones generadas con el pantógrafo maestro, se reprodujeron los correspondientes ángulos en el dispositivo esclavo:





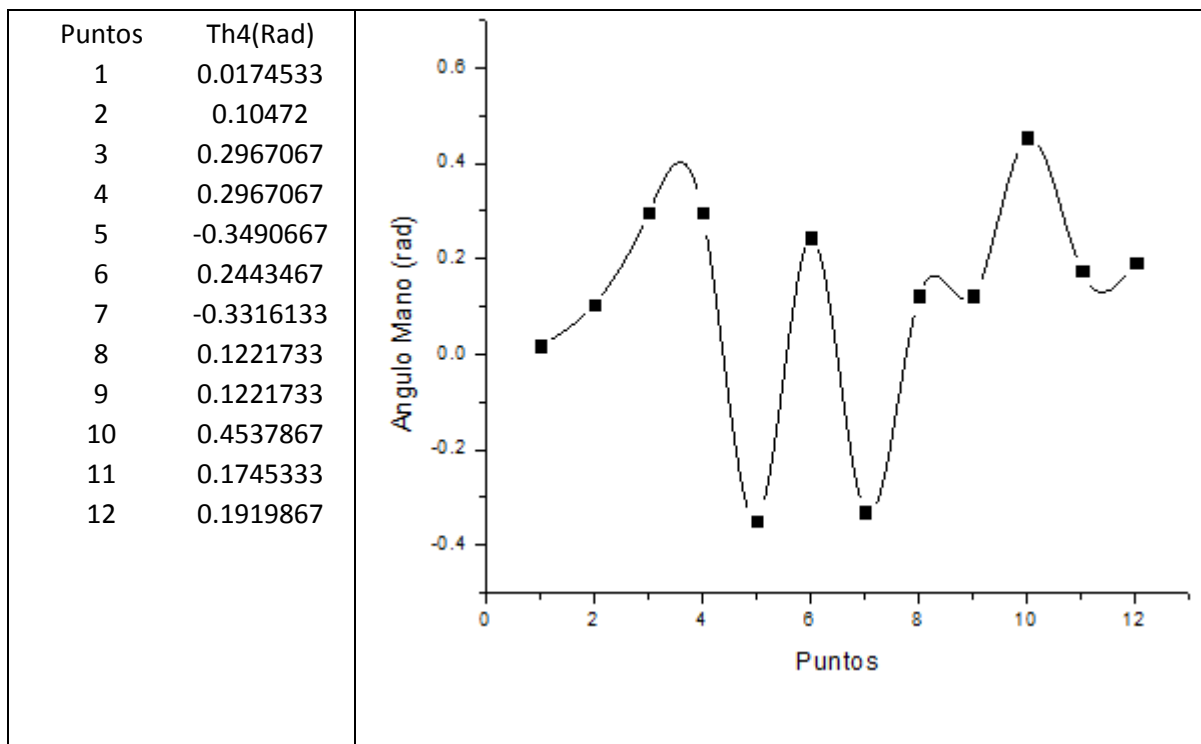


Figura18.Recorrido del "Black Hawk".

Discusión y conclusiones.

Las tecnologías podrán cambiar, los principios nunca.

Discusión

A partir de las pruebas y los resultados arrojados de las mismas. Englobamos los factores que impactaron positivamente, los eventos que perjudicaron el desempeño del proceso y los desafíos que se podrían corregir refinando cada vez más el procedimiento expuesto anteriormente.

Factores positivos:

- La selección de C# para realizar la programación de las interfaces fue correcta. La facilidad para escribir código, depurar errores del código, la velocidad a la que corre el programa, el diseño de ambientes virtuales y controlar errores lo hace ideal para desarrollar proyectos de teleoperación. Especialmente al utilizar la plataforma .Net de Visual Studio, la creación de conexiones y control de dispositivos del tipo I.O. fue más sencillo gracias a las librerías ya preparadas para este fin. Una situación que también influye es el costo del programa a comparación de otros programas de características similares. Para alumnos universitarios la licencia es gratuita gracias a los convenios con Microsoft y la UNAM.
- El método de Denavit-Hartenberg utilizado para realizar la cinemática directa e inversa fue totalmente confiable. Específicamente esto fue demostrado al realizar la ejecución de las pruebas en los robots y obtener que los datos teóricos son muy semejantes a los reales presentados en el robot.
- El controlador utilizado para impulsar el SCORBOT-ER 4u. El control para mantener al robot en la posición deseada, la repetitividad y la precisión fueron elementos bastante robustos, lo que permitió que la ejecución corriera sin problema mecánico alguno. Además los elementos de seguridad como protección a la sobrecarga, protección térmica y protección al impacto evitaron daños estructurales en el robot. Otro factor utilizado en el proyecto basado en el controlador, fue la posibilidad de programar trayectorias en lenguaje ACL.

- La comunicación empleando el protocolo TCP/IP con el puerto de comunicación TCP, fue bastante adecuado para el desarrollo académico del proyecto. Tanto en la recepción como en el envío de datos, no se pierde información en el camino.
- El dispositivo PAMCA, permito adquirir y visualizar las posiciones a reproducir en el esclavo. La facilidad para la manipulación del dispositivo y la adquisición de los datos en la interfaz maestro hacen del PAMCA un dispositivo ideal para generar trayectorias y adquirir información espacial.
- La plataforma Arduino tanto en el maestro como en el esclavo, demostró ser eficaz, al permitir aplicar y depurar códigos a gran velocidad. Además demostró la robustez de las tarjetas de trabajo para aplicaciones académicas.
- El robot impulsado por servomotores "*Black Hawk*", permitió ejecutar las instrucciones a gran velocidad al no necesitar involucrar un código de control de posición y con ello generar trayectorias en directo.

Factores negativos:

- El controlador del robot SCORBOT-ER 4u está muy limitado para hacer procesos de teleoperación en directo debido a la lentitud para procesar la información. Es decir, ante una petición de la interfaz para obtener los datos de los *encoders* del robot, el controlador tarda 650ms en responder la información requerida de un solo eje. Otro factor que también afecta al desempeño es el movimiento de los ejes del robot. En otras palabras, para generar la geometría de la trayectoria es necesario mover eje por eje el robot debido a que solo puede recibir y ejecutar un comando a la vez haciendo inviable el movimiento multiejes y los procesos en paralelo.
- La comunicación TCP/IP, usando el puerto TCP limitó bastante el funcionamiento de la recepción de la imagen proveniente de la fuente de video. Es decir, para que la operación fuera exitosa la cadena de video se tardaba demasiado tiempo en llegar del esclavo al maestro, con lo que no se tenía una visualización en tiempo real.

- El robot “*Black Hawk*” no presenta la misma precisión que el SCORBOT-ER 4u, especialmente por las limitantes en el diseño. Además de carecer de elementos de seguridad o de la capacidad para montarle herramientas en el efector final. Por lo cual sus aplicaciones recaen en el ámbito puramente académico.

A partir de las pruebas realizadas, se definieron sugerencias para continuar refinando el proyecto:

- Realizar un algoritmo para aumentar el número de soluciones cinemáticas posibles, y con ello no estar limitados a los cálculos realizados a la primera iteración en la interface.
- Debido al tamaño del código, se recomendaría optimizar y reducir el número de banderas, excepciones y procesos. Algunos de ellos redundantes, de tal manera que el código se redujera considerablemente utilizando técnicas de diseño para la programación. Además de intentar aplicar otro tipo de métodos para evitar el uso de *timers* y remplazarlos con procesamientos en paralelo.
- Como se explicó anteriormente utilizar puertos TCP, para generar la comunicación TCP/IP en el video es muy ineficaz. Por ende, se propone cambiar de TCP a UDP, que es un protocolo más flexible especialmente para tener más velocidad en el proceso, aunque se sacrifique la calidad del mismo.
- Como aplicación futura, se propone utilizar al maestro como centro de una operación para controlar varios esclavos. Es decir, a partir de un solo maestro poder programar varios esclavos, utilizando TCP/IP.

Conclusiones

El objetivo del proyecto fue teleoperar un SCORBOT-ER 4u empleando un Pantógrafo Maestro de Cadena Abierta (PAMCA). El resultado fue positivo, se logró reproducir la geometría de los puntos adquiridos por el PAMCA, e inclusive también se logró reproducir los ángulos del dispositivo antes mencionado. Todo esto se obtuvo empleando teleoperación off-line generando una cadena interna de control en la interface del

esclavo, se logró obtener la posición deseada de cada eslabonamiento. Sin embargo, las limitaciones fueron muchas, como lo demostraron las pruebas realizadas a la interface completa, el tiempo para lograr reproducir la geometría de un solo punto es muy grande en comparación con un robot como el "*Black Hawk*". Esto se debe principalmente a los retrasos del hardware que controla al robot. Por otra parte, una vez adquiridos y almacenados ciertos puntos en el controlador; la generación de trayectorias resulta muy fácil, al poder decidir el tipo de movimiento que debe seguir el robot para recorrer los puntos. Entonces, tenemos una aplicación lenta para la adquisición de puntos pero sencilla para programar trayectorias, y robusta al momento de ejecutarlas.

Ahora, con la parte de adquisición de puntos de la interfaz aplicada al movimiento del robot "*Black Hawk*", se logro reproducir una trayectoria aplicando una teleoperación online. Con un retraso apenas y perceptible entre el movimiento del PAMCA y el robot esclavo. Demostrando la versatilidad de la aplicación.

En la comunicación TCP/IP se observó claramente la capacidad de procesamiento de las interfaces, especialmente en la recepción de los datos de video. Donde, si existen procesos paralelos como los *timers*, el procesamiento de la cadena *Stream* se ve fuertemente castigada. Dando como consecuencia, que el video no se muestre en tiempo real.

Para hacer el proceso de trabajo más eficiente, es decir lograr mover el Robot SCORBOT-ER 4u a la posición final en un tiempo menor a los 2 segundos, lo más indicado seria desarrollar o comprar un nuevo controlador capaz de realizar movimientos en los ejes en paralelo y que tenga etapas de control más rápidas y efectivas. Además dicho controlador debe poseer la capacidad para generar una conexión efectiva con la interfaz esclavo.

Otro trabajo a futuro que queda pendiente es cambiar la retroalimentación del robot de tal manera que se pueda evitar la etapa de retroalimentación de los *encoders* utilizando el controlador, y más bien utilizar un sistema de visión obtener las posiciones lo cual permitiría eliminar una etapa muy tardada del proceso.

A partir de la investigación y el desarrollo realizado el paso natural siguiente sería aplicar técnicas de teleoperación más sofisticadas, que incluyan generación de ambientes en realidad aumentada y también en realidad virtual, que asemejen ambientes de trabajo reales, aplicaciones de supervisión humana e incluso inteligencia artificial. Todo esto con fines que podrían ir desde aplicaciones académicas y educativas, que permitan a los estudiantes tener un contacto de primera mano con tecnologías enfocadas a generar procesos a distancia. Hasta aplicaciones industriales donde los procesos sean controlados desde un centro de mando donde se concentre toda la información de los procesos en tiempo real y se generen decisiones para mejorar la productividad y el desempeño de las plantas de trabajo.

Bibliografía.

- Sheridan, Thomas B. (1992). ***Telerobotics, automation and human supervisory control***. USA: Massachusetts Institute of Technology Press.
- Jean Vertut, Philippe Coiffet. (1986). ***Teleoperation and robotics: evolution and development***. USA: Kogan Page.
- Tavokili M, Patel R.V. (2008). ***Haptic for Teleoperated Surgical Robotic Systems***. USA: World Scientific.
- Sakaar S, Ruoff C. (1994). ***Teleoperation and robotics in space***. USA: AIAA.
- Alencastre M., Muñoz L. (2003). ***Teleoperating Robots in Multiuser Virtual Environments***. MEX:ENC.
- Padilla M. (2008). ***Manipulador teleoperado inalámbicamente***. MEX:UDLAP.Capitulo 2.
- Lung-Wen Tsai. (1999). ***Robot analysis: the mechanics of serial and parallel manipulators***. USA: John Wiley & Sons.Inc.
- Chen Q., Fei S., Song A. (2003). ***Control robusto con observador basado en teleoperación por internet***. USA:ACC.
- Fong T., Thorpe C. (2001). ***Vehicle Teleoperation Interfaces***. USA: Kluwer Academic Publishers.
- Ollero B.A. (2001). ***Robótica, manipuladores y robots móviles***. España:Alfaomega-Marcombo.
- Schilling K.J., Vernet M.P.(2002). ***Remotely controlled experiments with mobile robots***. Proceedings of the Thirty Four Southeastern Symposium on System Theory.
- Sheridan T.B. (2003). ***Space teleoperation through time delay: Review and prognosis***, Transactions on Robotics and Automation. USA:IEEE.
- Comunicación RS232:
 - <http://rddatos.tripod.com/rs232.htm>
 - <http://www.tscm.com/rs-232.pdf>
 - <http://www.electronicafacil.net/tutoriales/INTERFAZ-RS232C.php>

- <http://www.tuelectronica.es/tutoriales/telecomunicaciones/comunicacion-a-traves-del-puerto-rs232.html>
- Internet y sus protocolos:
 - <http://es.kioskea.net/contents/internet/internet.php3>
 - <http://es.kioskea.net/contents/internet/protocol.php3>
 - <http://es.kioskea.net/contents/internet/http.php3>
 - <http://es.kioskea.net/contents/internet/ftp.php3>
 - <http://es.kioskea.net/contents/internet/arp.php3>
 - <http://es.kioskea.net/contents/internet/icmp.php3>
 - <http://es.kioskea.net/contents/internet/ip.php3>
 - <http://es.kioskea.net/contents/internet/tcp.php3>
 - <http://es.kioskea.net/contents/internet/udp.php3>
 - <http://es.kioskea.net/contents/internet/smtp.php3>
 - <http://es.kioskea.net/contents/internet/telnet.php3>
 - <http://es.kioskea.net/contents/internet/ip.php3>
 - <http://es.kioskea.net/contents/internet/dns.php3>
 - <http://es.kioskea.net/contents/internet/port.php3>
 - <http://es.kioskea.net/contents/internet/tcpip.php3#q=tcp%2Fip&cur=1&url=%2F>
 - <http://es.kioskea.net/contents/internet/tcp.php3>
- Servomotores:
 - <http://www.neoteo.com/servomotores-el-primer-paso-hacia-tu-robot>
- Métodos para programación en C#
 - <http://msdn.microsoft.com/en-us/library/sb27wehh.aspx>
 - <http://support.microsoft.com/kb/205087/es>
 - [http://technet.microsoft.com/es-es/library/bb125093\(EXCHG.65\).aspx](http://technet.microsoft.com/es-es/library/bb125093(EXCHG.65).aspx)
 - <http://msdn.microsoft.com/es-es/library/system.io.ports.serialport.aspx#Y3235>
 - [http://msdn.microsoft.com/es-es/library/system.net.sockets\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/system.net.sockets(VS.80).aspx)

- Ejemplos en C#
 - http://www.geekpedia.com/tutorial239_Csharp-Chat-Part-1---Building-the-Chat-Client.html
 - http://www.geekpedia.com/tutorial240_Csharp-Chat-Part-2---Building-the-Chat-Server.html
 - <http://elvex.ugr.es/decsai/csharp/>
 - client/server application
 - <http://www.csharp-examples.net/socket-send-receive/>
 - http://foro.elhacker.net/net/tutorial_sockets_en_c-t165986.0.html
 - <http://www.ucontrol.com.ar/forosmf/problemas-con-mis-proyectos/controlar-pc-desde-otro-pc/5/?wap2>
 - <http://www.switchonthecode.com/tutorials/csharp-tutorial-simple-threaded-tcp-server>
 - <http://www.dreamincode.net/forums/topic/114939-streaming-webcam-from-client-to-server/>
 - <http://processing.org/discourse/yabb2/YaBB.pl?num=1192330628>
 - http://www.codeproject.com/KB/IP/Video_Voice_Conferencing.aspx

Referencias de figuras.

Capítulo 1.

- Figura 1---Sakaar S, Ruoff C. Teleoperation and robotis in space(1994).USA: AIAA. Página 7.
- Figura 2---Tavokili M, Patel R.V, Haptic for Teleoperated Surgical Robotic Systems(2008).USA: World Scientific. Página 6.
- Figura 3--- Sheridan,Thomas B.Telerobotics, automation and human supervisory control.(1992). USA: Masacchussets Institute of Tchnology Press. Página 2.
- Figura 4--- Sheridan,Thomas B.Telerobotics, automation and human supervisory control.(1992). USA: Masacchussets Institute of Tchnology Press. Página 14.
- Figura 5--- Jean Vertut, Philippe Coiffet. Teleoperation and robotics: evolution and development.(1986).USA: Kogan Page. Página 85.
- Figura 6----- Jean Vertut, Philippe Coiffet. Teleoperation and robotics: evolution and development.(1986).USA: Kogan Page. Página 86.
- Figura 7--- <http://www.instructables.com/id/Camara-de-Video-en-Carro-de-Radio-Control-Video-/>
- Figura 8---
<http://www.projectsjugaad.com/electronicprojectslinks/Electronicsprojects57.asp>
- Figura 9--- <http://autsys.tkk.fi/fsr/attach/Material/Teleoperation.pdf>
- Figura 10--- <http://www.gizmag.com/military/>
- Figura 11---http://en.wikipedia.org/wiki/File:NASA_Mars_Rover.jpg
- Figura 12---<http://www.spacetoday.org/SolSys/Mars/MarsExploration/MTO.html>
- Figura 13---http://jnn-digital.blogspot.com/2011_03_30_archive.html
- Figura 14---<http://www.robotikka.com/>
- Figura 15---<http://tecomagazine.net/2008/04/18/robot-minero-ayuda-a-salvar-vidas-y-trazar-mapas-de-tuneles/>
- Figura 16---http://www.kalipedia.com/tecnologia/tema/robotica/graficos-brazo-mecanico.html?x1=20070821klpinginf_54.Ees&x=20070821klpinginf_96.Kes

- Figura 17--- Nuño O. Teleoperación: Técnicas, aplicaciones, entorno sensorial y teleoperación inteligente. UPC. 2004.
- Figura 18--- Manual SCORBOT V Plus.
- Figura 19---Gabriel Torres, Mario Matus. Control de un brazo robot de cinco grados de libertad mediante un PLC. UNAM. 2010.
- Figura 20--- Helleman Olguin, Daniel Sanchez. Control de un brazo robotico teleoperado mediante un acelerómetro. UNAM.2011.

Capitulo 2.

- Figura 2-1--- Manual Scorbot Er-4u Lung-Wen Tsai. Robot analysis: the mechanics of serial and parallel manipulators.(1999).USA: John Wiley & Sons.Inc. Página 124.
- Figura 3--- Lung-Wen Tsai. Robot analysis: the mechanics of serial and parallel manipulators.(1999).USA: John Wiley & Sons.Inc. Página 40.
- Figura7---Pagina de Arduino.
- Figura8---Manual Scorbot ER4U
- Tabla9---Manual Scorbot ER4u
- Figura 10---Manual Scorbot ER4u
- Figura 11---Manual Scorbot ER4u
- Figura 12---Manual Scorbot ER4u
- Figura 13---Manual Scorbot ER4u
- Tabla14---Manual Scorbot ER4u
- Figura 15---Manual Scorbot ER4u
- Figura16---<http://www.neoteo.com/servomotores-el-primer-paso-hacia-tu-robot>
- Figura18---http://wiki.slimdevices.com/index.php/Network_Design
- Figura19---<http://www.comusoft.com/como-conocer-tu-ip-publica-o-privada>
- Figura20--- <http://es.kioskea.net/contents/internet/tcp.php3>
- Figura21--- <http://es.kioskea.net/contents/internet/tcp.php3>

- Figura22--- <http://es.kioskea.net/contents/internet/tcp.php3>
- Figura23--- <http://es.kioskea.net/contents/internet/tcp.php3>
- Figura24--- <http://es.kioskea.net/contents/internet/tcp.php3>

Anexos.

Anexo 1. *Advanced Control Language.*

ACL has two types of commands:

- **DIRECT** commands, which are executed as soon as they are entered at the terminal/computer keyboard.
- Indirect, or **EDIT** commands, which are executed during the running of the programs and routines in which they are used.

Some commands are available in both the DIRECT mode and the EDIT mode.

DIRECT Mode

When DIRECT mode is active, all commands entered from the keyboard are immediately executed by the controller.

Whenever the DIRECT mode is active, the screen shows the following cursor prompt:

>_

DIRECT mode commands can be included in programs for execution from a running program by prefacing them with the character @. The @ signals to the controller that the string be read as a DIRECT mode command, and activated from a running program.

Once the @ command has been transmitted, and its execution has begun, the program continues running regardless of the @ command's status. Use the DELAY command to ensure completion of a @ command.

EDIT mode commands can be executed in the DIRECT mode when preceded by the command DO.

Refer to the command descriptions for @, DELAY, and DO in Chapter 3.

Manual Keyboard Control

When in DIRECT mode, you can assume direct control of the robot and peripheral axes from the keyboard by activating Manual mode. This mode is useful when a teach pendant is not available.

To activate the Manual mode, type either of the following:

<Alt>+m	(when using ATS)
~	(usually by pressing <Shift>+`)

The commands which can be executed in Manual mode are comparable to those available from the teach pendant.

Refer to the command ~ (Manual Keyboard Mode) in Chapter 3 for a complete description of the functions available in Manual mode.

Comandos utilizados en modo directo (D) v en modo edit (E)

3.1 Definición v grabado de posiciones

Defp: este comando sirve para definir una posición, reservando un espacio en la memoria del controlador, el nombre puede ser numérico o alfanumérico de hasta 5 caracteres.

(D,E) Defp <i>pos</i>	define una posición llamada <i>pos</i> en el grupo A
(D,E) Defpb <i>pos</i>	define una posición llamada <i>pos</i> en el grupo B
(D,E) Defpc <i>pos</i>	define una posición llamada <i>pos</i> en el grupo C

Dimp: este comando permite definir un vector de una cantidad determinada de posiciones,

(D,E) Dimp <i>pvec</i> [n]	define un vector de n posiciones llamado <i>pvec</i> en el grupo A
(D,E) Dimpb <i>pvec</i> [n]	define un vector de n posiciones llamado <i>pvec</i> en el grupo B
(D,E) Dimpc <i>pvec</i> [n]	define un vector de n posiciones llamado <i>pvec</i> en el grupo A

Cada posición se llamará *pvec* [1], *pvec* [2], ... *pvec* [n]

Delp: sirve para borrar una posición ya definida, libera el espacio reservado en memoria.

(D,E) Delp <i>pos</i>	borra la posición existente llamada <i>pos</i>
(D,E) Delp <i>pvec</i>	borra el vector de posiciones <i>pvec</i>

Undef: borra los valores de las coordenadas grabadas pero la posición sigue definida. Puede usarse en posiciones individuales o en vectores.

(D,E) Undef <i>pos</i>	borra el contenido de la posición existente llamada <i>pos</i>
(D,E) Undef <i>pvec</i>	borra el contenido del vector de posiciones <i>pvec</i>

Here: graba la posición en la que se encuentra el robot en coordenadas joint

(D,E) Here <i>pos</i>	graba en la posición <i>pos</i> definida anteriormente las coordenadas joint
-----------------------	--

Herec: graba la posición en la que se encuentra el robot en el momento de ejecutar la instrucción, en coordenadas cartesianas.

(D,E) Herec <i>pos</i>	graba en la posición <i>pos</i> definida anteriormente, las coordenadas cartesianas
------------------------	---

Herer: graba las coordenadas en las que se encuentra el robot relativas a otra posición

- (D) **Herer *pos*** graba en la posición *pos* las coordenadas joint referidas a la posición actual. Se deberán ingresar los valores de corrimiento luego de ejecutar la instrucción.
- (D,E) **Herer *pos2 pos1*** graba en la posición *pos2* definida anteriormente las coordenadas joint de corrimiento relativas a la posición *pos1* grabada previamente.

Teach: graba coordenadas cartesianas de una posición del robot de acuerdo a valores ingresados por el usuario

- (D) **Teach *pos*** graba en la posición *pos* las coordenadas cartesianas absolutas. Se deberán ingresar los valores de las coordenadas para definir la posición deseada.

```
>teach pos
  X -- [.] >
  Y -- [.] >
  Z -- [.] >
  P -- [.] >
  R -- [.] >
```

Teachr: graba coordenadas cartesianas de una posición con respecto a otra ya grabada

- (D) **Teachr *pos*** graba en la posición *pos* las coordenadas cartesianas relativas a la posición actual. Se deberán ingresar los valores de corrimiento para definir la posición deseada.
- (D) **Teachr *pos2 pos1*** graba en la posición *pos2* las coordenadas cartesianas relativas a la posición *pos1* grabada con anterioridad. Se deberán ingresar los valores de corrimiento para definir la posición deseada.
- (D) **Attach *pvec*** asigna las posiciones contenidas en el vector *pvec* a las posiciones del teach pendant.
- (D) **Setpv *pos*** graba la posición en *pos* en coordenadas joint.
- (D,E) **Setpv *pos axis var*** cambia una de las coordenadas joint de la posición *pos* predeterminada.
- (D,E) **Setpvc *pos coord var*** cambia una de las coordenadas cartesianas de la posición *pos* predeterminada.

3.2 Movimiento a posiciones grabadas

- (D,E) **Move** *pos* el robot se mueve a la posición grabada en *pos*.
- (D,E) **MoveL** *pos* la pinza se mueve desde la posición actual hasta la posición *pos*. en línea recta, siempre que esto sea posible.
- (D,E) **MoveC** *pos1 pos2* mueve la pinza desde la posición en que se encuentra hasta la posición *pos1* siguiendo una trayectoria circular a través de la posición *pos2* .
- (E) **Moved** *pos* el robot se mueve a la posición grabada en *pos* y hasta que no llega hasta dicha posición no continúa con el programa .
- (E) **MoveLd** *pos* la pinza se mueve desde la posición actual hasta la posición *pos*. en línea recta, siempre que esto sea posible y no continúa con el programa mientras no haya completado el .
- (E) **MoveCd** *pos1 pos2* mueve la pinza desde la posición en que se encuentra hasta la posición *pos1* siguiendo una trayectoria circular a través de la posición *pos2* y no pasa a la siguiente línea de programa hasta que no se haya completado el movimiento.

3.3 Comandos de control de ejes

- (D,E) **Open** abre la pinza
- (D,E) **Close** cierra la pinza
- (D) **Con** habilita el servo control de todos los ejes, o de algún grupo en particular
- (D) **Con axis** habilita el servo control de un eje especificado en *axis*
- (D) **Coff** deshabilita el servo control de todos los ejes, o de algún grupo en particular
- (D) **Coff axis** deshabilita el servo control de un eje especificado en *axis*
- (D,E) **Set anout[n] = DAC** deshabilita el control del eje n y setea el valor de la tensión

- (D,E) **Speed var** setea la velocidad de los ejes del grupo A para las instrucciones MOVE y MOVES , la variable var será un porcentaje del valor de velocidad máxima.
- (D,E) **Speed1 var** setea la velocidad de los ejes del grupo A para las instrucciones MOVE y MOVES , la variable var será un porcentaje del valor de velocidad máxima.

3.4 Comandos de control de programas

Run: permite ejecutar programas ya editados.

- (D,E) **Run prog** ejecuta el programa *prog*.

A: aborta la ejecución de programas.

- (D) **A prog** aborta la ejecución del programa *prog*.
- (D) **<ctrl> + A** aborta la ejecución de todos los programas que estén corriendo

Stop: aborta la ejecución de programas.

- (E) **Stop prog** aborta la ejecución del programa *prog*.
- (E) **Stop** aborta la ejecución de todos los programas que estén corriendo

Suspend y Continue: sirven para interrumpir y continuar ejecutando un programa.

- (D,E) **Suspend prog** interrumpe la ejecución del programa.
- (D,E) **Continue prog** continua la ejecución del programa interrumpido
- (E) **Delay var** suspende la ejecución del programa durante el tiempo(en centésimas de segundo) especificado en *var*.
- (E) **Wait var1 oper var2** suspende la ejecución del programa hasta que la condición se satisfaga.
- (E) **Trigger prog by {in/out} n {0/1}** ejecuta un programa condicional al cambio en el estado de la entrada o salida *n*.

Referencias:

ACL. *Advanced Control Language*.(1995) Cuarta Edicion.Eshered Robotech.

Anexo 2. Conexiones del cable *RS232*.

Listado de las patillas

Las patillas físicas en los pines son las siguientes:

Segnal	Patilla en DB9	Patilla en DB25

GND:.....	patilla 5.....	patilla 7
RX:.....	patilla 2.....	patilla 3
TX:.....	patilla 3.....	patilla 2
RTS:.....	patilla 7.....	patilla 4
CTS:.....	patilla 8.....	patilla 5
DSR:.....	patilla 6.....	patilla 6
DTR:.....	patilla 4.....	patilla 20

Localización física de las patillas en un DB9

Los conectores suelen llevar una chuleta numerando los pines en el plástico que rodea a los susodichos, ese plástico que a veces es azul, o a veces en negro o a veces es blanco. Vista desde el LADO DE FUERA DEL PC, que también concuerda con la vista desde la CARA DE LAS SOLDADURAS de un DB9 Hembra usado para construir el cable:

```

-----
\ 1 2 3 4 5 /
\ 6 7 8 9 /
-----

```

Vista desde el lado de fuera de un DB9 Hembra usado para construir el cable:

```

-----
\ 5 4 3 2 1 /
\ 9 8 7 6 /
-----

```

Localización física de las patillas de un DB25

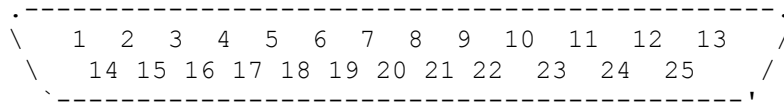
Vista desde el **exterior del PC**, que también concuerda con la vista desde la **cara de las soldaduras** de un DB25 hembra usado para construir el cable:

```

-----
\ 13 12 11 10 9 8 7 6 5 4 3 2 1 /
\ 25 24 23 22 21 20 19 18 17 16 15 14 /
-----

```

Vista desde el lado de fuera de un DB25 hembra usado para construir el cable:



Referencias:

<http://es.tldp.org/COMO-INSFLUG/COMOs/Terminales-Como/Terminales-Como-3.html>

Anexo 3. Controller A.

Controller-A Control Process

The basic steps of the Controller-A control loop are described below. Refer to Figure A-5. The entire control cycle takes 10ms.

The processor calculates the command position and speed once per cycle. It outputs a digital value to the DAC unit in the range of ± 5000 .

The analog unit creates a series of pulses, resulting in an average voltage value proportional to the DAC input.

The power unit drives the motor by switching $\pm 24V$ to it at 20KHz, according to the input pulse. The motor cannot react to this high frequency of switching and is therefore affected by only the average value of the voltage.

This method of controlling the time during which current flows through the motor, rather than controlling the value of the current, is known as PWM (Pulse Width Modulation) control. Refer to Figure A-6.

Once per cycle the processor reads the encoder's count and calculates the motor's position and speed (rate of encoder counts). The processor then compares the actual (output) position and speed values with the desired (input) ones, determines the error values and takes the necessary action to cancel them.

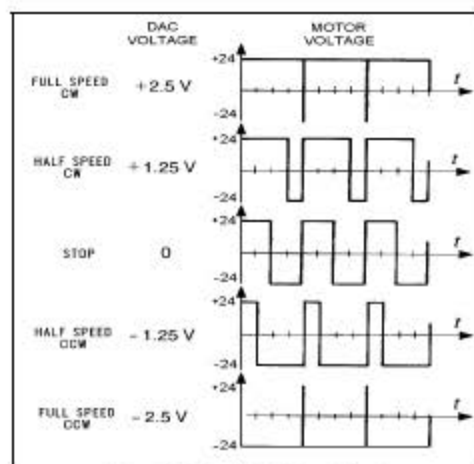


Figure A-6: Controller-A Control Signals

Trajectory Control

For better path performance (that is, to accurately reach the desired state and avoid overshoots), trajectory control profiles, may be programmed into the control system. **Controller-A** offers two profiles: paraboloid and trapezoid. Refer to Figure A-7.

Paraboloid

The paraboloid profile causes the motors to accelerate slowly until maximum speed is reached, then decelerate at the same rate.

Trapezoid

The trapezoid profile causes the motors to accelerate and decelerate quickly at the start and end of movement, with a constant speed along the path.

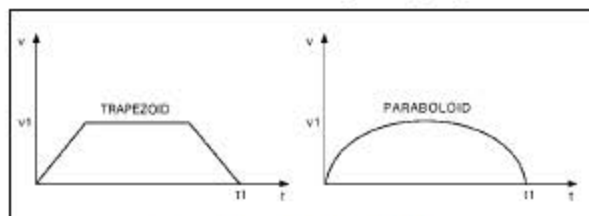


Figure A-7: Trajectory Control Profiles

Path Control

It is desirable that the path and speed of a robot between taught points be predictable. Ideally, the path between consecutive points is traversed at a constant velocity with defined acceleration and deceleration segments.

Along the path, motion of all joints should be proportional, so that all the joints start and finish moving at the same time. The method of coordinating the movement of the joints so that all joints reach the desired location simultaneously is termed joint interpolation.

Point-to-Point Control

Point-to-point control (PTP) involves the positioning of the robot's end effector at given points, without defining the exact path of the end effector between any two points.

Point-to-point control is suitable for applications which require an exact and static position of the end effector at the points where operations will be performed.

In principle, point-to-point control can be used to guide the robot through a large array of positions, thus resulting in a complex path. In order to obtain such a path, points must be defined and recorded in a very close sequence. The number of positions will be limited, however, by the capacity of the control system to maintain positions in memory.

Continuous Path Control

Continuous path control (CP) involves the movement of the end effector between two points along a path defined by a mathematical formula. This method of control is suitable for applications in which the end effector executes operations along a precise trajectory.

During program execution, the control system calculates and plans the path, and instructs the robot motors to move accordingly.

When continuous path control is required, the processor divides the path into short segments, and interpolates the motion of the joints as frequently as possible.

Three type of CP control are possible.

- **Joint Control:** Each axis moves according to the trajectory profile. The gripper path is not defined; only the start and end points are defined. All axes start and stop movement at same time.
- **Linear Path Control:** The axes are coordinated in order to move the TCP (tool center point; tip of the gripper) in a straight line according to the trajectory profile.
- **Circular Path Control:** The axes are coordinated in order to move the TCP along a circular path according to the trajectory profile.

The Control Parameters

In the robotic system controller by **Controller-A**, as in common in closed-loop systems, the controlled value (C) is measured by an optical encoder. The encoder signals serve as feedback to the controller, enabling it to correct any deviations from the desired value.

Since control systems cannot react immediately to the input signal, there will always be a lag between the generation of an error signal and the actual correction of the controlled value.

The PID (proportional, integral, differential) control parameters allow the controller to adapt to various conditions of operation, such as overcoming nonlinear functions in the system.

Proportional Control

The proportional parameter is the gain of the control system. Its value determines the reaction time to position errors.

When a position error exists (that is, the actual motor position is off by a certain amount of encoder counts), the processor multiplies the error by the proportional parameter and adds the product to the DAC value, thereby reducing the error.

The proportional parameter is the parameter in the PID control system which acts most quickly in reducing the position error, especially during motion. It is also the first parameter to respond to position errors when the robot has stopped at a target position.

The greater the proportional parameter, the faster the system responds and reduces the error. But, using too great a value for the proportional parameter will cause the axis to oscillate.

The main disadvantage of proportional control is that it cannot completely cancel the error, because once it has reduced the error it cannot generate enough power to overcome friction in the system and propel the axis to its target position.

Even in steady state, under load, the controlled value (output signal) will always be different from the desired value (input signal). The steady state error can be reduced by increasing the gain, but this will increase the oscillation and reduce stability.

Differential Control

In differential control, the controller output (C) is a function of the rate at which the error (U_e) changes. The faster the rate of change of the error, the greater the controller output (C). In other words, the controller is sensitive to the slope of the error signal.

The differential parameter is responsible for reducing the speed error. The control system calculates the actual speed once per cycle and compares it to the desired value. While the robot is accelerating (during the first part of path) the differential acts as a driving factor.

While the robot is decelerating (during the second, and last, part of path), the differential acts as a braking factor. A good differential setting will result in a clean and smooth motion along the entire path. Lack of the differential will cause overshoot at the end of path. High differential values will cause small vibrations along the path.

In this control method, the controller predicts the value of the error in accordance with the error signal slope, and causes the correction to take place in advance. However, if the error is constant and unchanging, differential control will not be able to reduce the error to zero.

Integral Control

In integral control, all the state errors which have been recorded each cycle are totalled and their sum is multiplied by the integral parameter value.

In integral control, the controller output (C) reduces the error signal (U_e) to zero at a rate proportional to the size and duration of the error. In other words, the greater the error, the greater the controller output; and, the longer the duration of the error, the greater the controller output.

The main advantage of integral control is that the steady state error is always reduced to zero since its value increases each cycle, thus strengthening the control system's ability to react and reduce the error. However, using too great a value for the integral parameter may cause overshoots, while too small a value may prevent the cancellation of a steady state error.

Unlike the proportional parameter, the integral parameter takes effect more slowly and is less noticeable during motion. However, when the axis comes to a complete stop and the proportional parameter can no longer reduce the steady state error, the integral parameter takes over and can cancel the error completely.

Proportional–Integral–Differential Control

The PID control method enables optimal exploitation of all three types of control—proportion, integral and differential. In this manner, it creates an output response which follows the input signal closely, without gaps or lags, in both slow and rapid processes, including those in which the load is in a constant state of change. In summary, the PID control parameters serve the following functions:

- **Proportional Parameter:** Enables fast and powerful reactions of the arm to movement commands. Responsible for the repeatability of the motion.
- **Integral Parameter:** Assists the proportional parameter in eliminating steady state errors.
- **Differential Parameter:** Provides the required damping.

Offset

Control theories often assume complete linearity; that is, the speed is proportional to the power supplied to the motor.

However, at low levels of power, the motor will not move, mainly due to friction; that is, the static friction is higher than the dynamic friction. This is a non-linearity. Figure A-8 shows linearity and non-linearity.

The offset is a threshold level of the DAC. Above this DAC value the control system acts as a linear system. Below this value, the control system acts as an on/off system. Figure A-9 shows the offset.

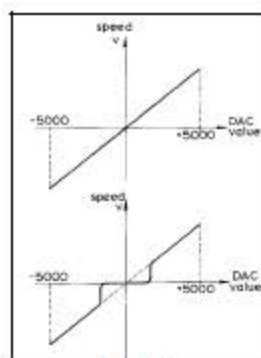


Figure A-8:
Linearity and Non-Linearity

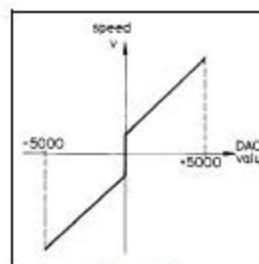


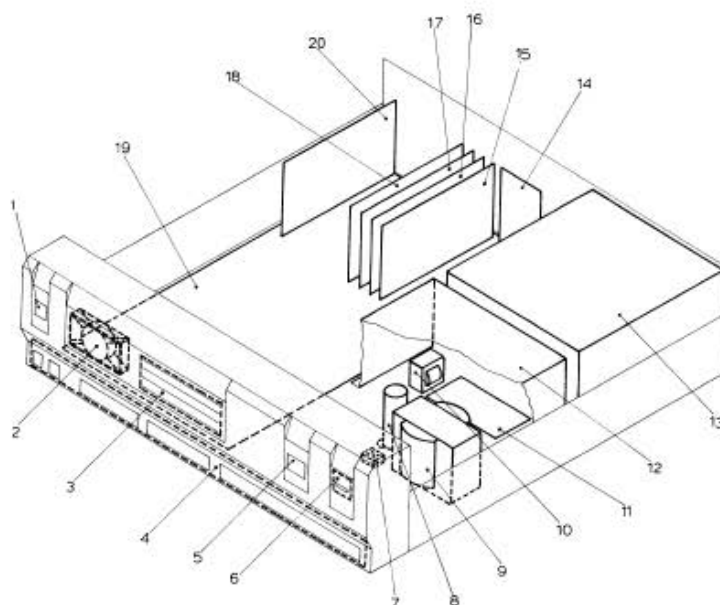
Figure A-9:
Control System Offset

Changing Parameter Values

The control system parameters of **Controller-A** are factory-set, and are suitable for most robotic applications.

Although parameter values can be manipulated by user commands, only experienced users should attempt to do so.

For more details refer to the *ACL Reference Guide*.



Controller

Dwg #	Cat #	Description
	110715	Metal case - lower part
	110717	Metal case - upper part
	113002	Controller front panel
	110719	Metal case - rear panel
	110723	Coil fastener
	102501	Lexan tags for front panel
	110725	Long bracket - driver card support
	107204	Blank brackets
12	110721	Transformer cover
13	35008	Logic power supply (220/110VAC)
9	35006	24V/12V Transformer (100VAC)
10	35003	Gripper coil assembly
2	35001	Fan plus cabling and connector
19	450541	Main board
15-17	45018	Driver cards for robot
18	45019	Driver cards for accessories
3	45011	Display card
4	45013	I/O card
11	45023	User power supply card
14	45009	Communication card
5-6	45003	Power LED card plus motors switch plus cabling
1	40004	Emergency switch plus cabling
	40018	+24VDC feed cable (from J12 to capacitor J12)
	40007	Diode bridge cabling
	40005	Switching cable (from J20 to user power supply)
	411807	Flat cable (from J10 to I/O card)
	411806	Flat cable (from J13 to communication card)
	411808	Flat cable (from J11 to display card)
	411805	Flat cable (from J8 to communication card)
	40017	Gripper cable (jumper)
	40010	Grounding cable for capacitor
	40009	Grounding cable for transformer metal cover
	40006	Resistors cable for capacitor
7	408102	Diode bridge
8	404501	10,000 μ F/63V capacitor
	45024	Teach pendant card [inside teach pendant]

Referencias:

SCORBOT-ER V Plus.USER MANUAL.(1996) .Intelitek.

Anexo 4. *SCORBOT-ER 4u.*

Maintenance

The maintenance and inspection procedures detailed below will ensure continued optimum performance of the SCORBOT-ER 4u system.

Daily Operation

Perform a routine inspection of your system at the start of every working session, in the following order:

1. Before you power on the system, check the following items:
 - The installation meets all safety standards.
 - The robot is properly bolted to the work surface.
 - All cables are properly and securely connected. Cable connector screws are fastened.
 - No output is connected directly to a power supply.
 - No people are within the robot's working range.
2. After you have switched on the PC and the controller, check the following items:
 - The power and motor LEDs on the controller light up.
 - No unusual noises are heard.
 - No unusual vibrations are observed in any of the robot axes.
 - There are no obstacles in the robot's working range.
3. Bring the robot to a position near home, and activate the homing procedure. Check the following items:
 - Robot movement is normal.
 - No unusual noise is heard when robot arm moves.
 - Robot reaches home position in every axis.

Periodic Inspection

The following inspections should be performed regularly:

1. Visually check leads, cables and rubber components. Replace any cables which show signs of abrasion or wear.
2. Check all bolts and screws in the robot arm using a wrench and screwdriver. Retighten as needed.
3. Check all the tension of robot arm belts. When you press on a belt, the slack should be no greater than 2mm (0.08"). Refer to Figure 14.

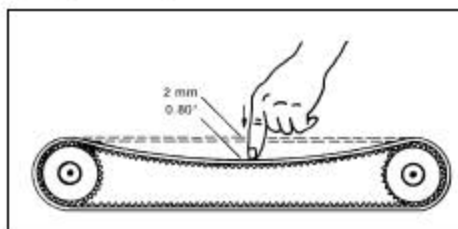


Figure 14: Belt Tension

Qualified Technician Only: Tighten the belts only if you are absolutely certain they are slipping or retarding the motors. For complete information, refer to the section, "Adjustments and Repairs," later in this chapter.

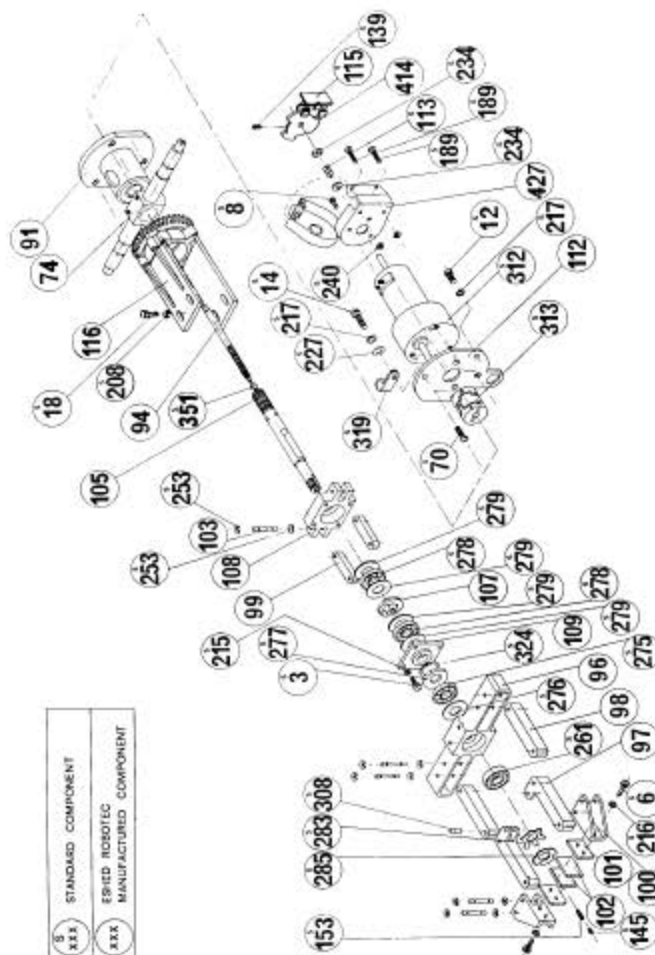
4. **Qualified Technician Only:** Check for excessive backlash in the base axis. For complete information, refer to the section, "Adjustments and Repairs," later in this chapter.

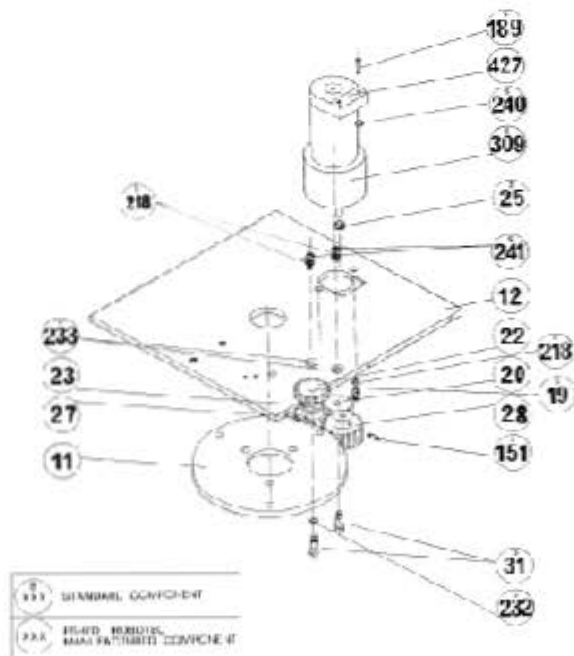
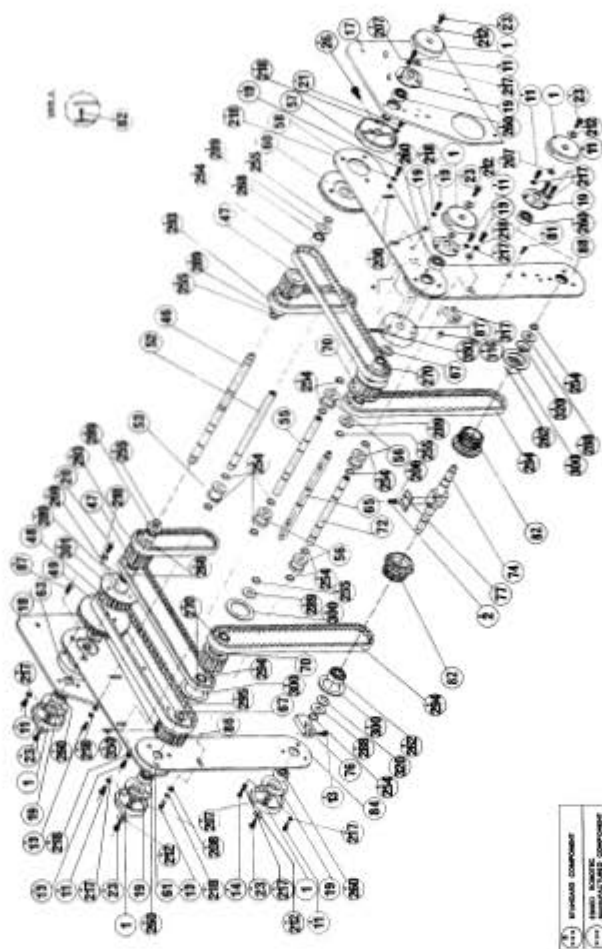
Dwg #	Cat #	Description
1	113012	Bearing housing cover (plastic)
2	111401	Main shaft base
S 2	306003	Socket head cap screw #4-40 X 1/4
S 3	306004	Socket head cap screw #4-40 X 3/8
4	113004	Base plate
5	113001	Base
S 6	306201	Socket head cap screw #6-32 X 1/4
S 8	306002	Socket head cap screw #2-56 x 3/8
11	111906	Spur gear (120 teeth)
S 11	306204	Socket head cap screw #8-32 x 1/4
12	112103	Bottom Plate - shoulder
S 12	301205	Socket head cap screw #8-32 x 3/8
S 13	306206	Socket head cap screw #8-32 x 1/2
S 14	306207	Socket head cap screw #8-32 x 5/8
15	112401	Support base - motors 4+5
16	112403	Support clamp - motors 4+5
17	110205	Right side plate - shoulder
18	110210	Left side plate - shoulder
S 18	306401	Socket head cap screw #10-32 x 3/8
S 19	306402	Socket head cap screw #10-32 x 1/2
20	111901	Anti-backlash spur gear (transfer)
S 20	306404	Socket head cap screw #10-32 x 3/4
S 21	306405	Socket head cap screw #10-32 x 7/8
22	111902	Anti-backlash spur gear (upper)
S 22	306407	Socket head cap screw #10-32 x 1/4
23	113501	Anti-backlash spring
S 23	306403	Socket head cap screw #10-32 x 5/8
24	107003	Washer
S 24	306408	Socket head cap screw #10-32 x 1 1/2
S 25	321001	Ball bearing (motor 1 gear)
S 26	306602	Socket head cap screw #1/4-20 x 1
27	111903	Anti-backlash spur gear (base)
S 27	306602	Socket head cap screw #1/4-20 x 5/8
28	111907	Spur gear (base motor)
S 31	306414	Socket head cap screw #10-32 x 3/4 x 1/4 shoulder
32	319404	Spur gear (motors 2+3)
34	112412	Motor support (motor 2)
35	112412	Motor support (motor 3)
37	112402	Motor support (motors 4+5)
38	319406	Timing belt pulley (motors 4+5)
40		Rear cross bar [not used in ER 4u]
46	111402	Main shoulder shaft
47	111909	Timing belt pulley
48	111911	Timing belt pulley

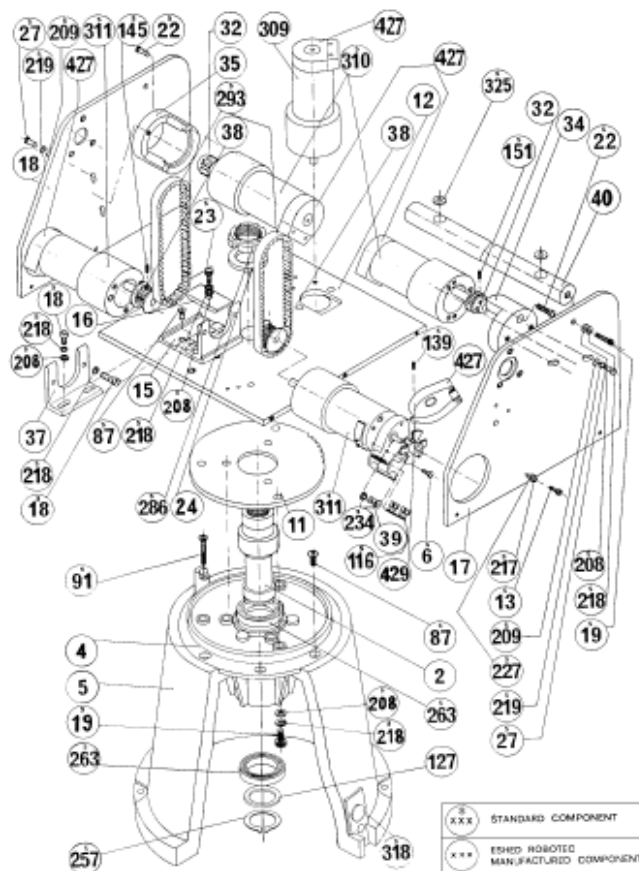
Dwg #	Cat #	Description
49	111905	Spur gear (72 teeth)
52	111405	First tension shaft
53	113013	Tension wheel
55	111406	Second tension shaft
56	113014	Tension pulley
57	112406	Clamp - lower arm - left side plate
58	110215	Upper arm - right side plate
60	111904	spur gear (right - 72 teeth)
61	110220	Upper arm - left side plate
63	112407	Clamp - lower arm - left side plate
64	111403	Middle shaft
67	107001	Aluminum spacer
70	111910	Timing belt pulley
S 70	306007	Flat head socket screw #4-40 x 1/4
72	111407	Third tension shaft
74	111404	Gripper axis
76	112439	Stopper (motors 4+5)
77	110705	Base plate limit switch
S 81	306201	Flat head socket screw #8-32 x 3/8
82	113008	Timing belt pulley - miter gear
S 82	306211	Flat head socket screw #8-32 x 1/2
84	110228	Forearm left side plate
86	111912	Timing belt pulley
87	112114	Flange
S 87	306410	Flat head socket screw #10-32 x 1/2
88	110223	Forearm - right side plate
91	112408	Gripper gear motor support
S 91	306412	Flat head socket screw #10-32 x 1/4
94	113801	Lead screw
96	112117	Gripper bridge
97	112118	Gripper finger (inner)
98	112119	Gripper finger (outer)
99	112120	Gripper finger (short)
100	112113	Gripper clamp
101	110703	Mounting plate - gripper
102	113201	Rubber pad - gripper
103	111409	Pivot pin
105	111408	Main shaft - gripper
107	113802	Lead nut - gripper
108	112115	Bearing housing
109	112116	Bearing housing cover
112	110229	Gripper motor base plate
113		Spring [not used in ER 4u]
S 115	45008	Encoder circuitry (20 slots)

Dwg #	Cat #	Description
116	113009	Miter gear (bottom)
S 116	45008	Encoder circuitry (20 slots)
127	107009	Spacer washer (for base bearing)
S 139	306008	Socket head set screw #4-40 x 1/8
S 145	306213	Socket head set screw #8-32 x 3/16
S 151	306413	Socket head set screw #10-32 x 3/16
S 153	306214	Socket head set screw #8-32 x 1/4 (without head)
S 187	302002	Socket binding head screw M2 x 10 (limit switch)
S 188	302001	Slotted binding head screw M2 x 8 (limit switch)
S 189	302006	Slotted binding head screw M2x20 (encoder housing)
S 206	313001	Washer (for screw #4-40)
S 207	107012	Washer (black); internal; for plastic cover $\varnothing 12.5 \times \varnothing 5.5 \times 0.6$
S 208	313004	Washer for screw #10-32
S 209	313005	Washer for screw $\varnothing 1/4$
S 212	314508	Washer lock; black; external $\varnothing 5$
S 215	314002	Spring washer (for screw #4-40)
S 216	314003	Spring washer (for screw #6-32)
S 217	314004	Spring washer (for screw #8-32)
S 218	314005	Spring washer (for screw #10-32)
S 219	314006	Spring washer (for screw $\varnothing 1/4$)
S 225	314503	Lock washer M2
S 227	313003	Washer (for screw #8-32)
S 232	107008	Teflon washer $\varnothing 1/4" \times \varnothing 3/8" \times 0.6\text{mm}$
S 233	107007	Teflon washer $\varnothing 1/4" \times \varnothing 1/2" \times 0.6\text{mm}$
S 234	113016	Nylon washer $\varnothing 11 \times 4$ [not used in ER 4u]
S 240	310001	Hexagonal nut M2
S 253	316006	E-Ring $\varnothing 1/8$ DIN 6799
S 254	316003	Retaining ring $\varnothing 10$ DIN 471
S 255	316004	Retaining ring $\varnothing 12$ DIN 471
S 257	316302	Retaining ring $\varnothing 25$ DIN 471
S 260	320005	Ball bearing $\varnothing 8 \times \varnothing 22 \times 7$
S 261	320004	Ball bearing $\varnothing 10 \times \varnothing 19 \times 5$
S 262	320006	Ball bearing $\varnothing 10 \times \varnothing 26 \times 8$
S 263	320203	Ball bearing $\varnothing 25 \times \varnothing 47 \times 8$
S 268	320701	Needle bearing $\varnothing 12 \times \varnothing 16 \times 10$
S 269	320702	Needle bearing $\varnothing 12 \times \varnothing 19 \times 16$
S 270	320704	Needle bearing $\varnothing 15 \times \varnothing 21 \times 12$
S 270	320705	Bushing for #320704
S 275	320501	Thrust bearing $\varnothing 10 \times \varnothing 24 \times 2$
S 276	320502	Thrust washer $\varnothing 10 \times \varnothing 24 \times 1$
S 277	320503	Thrust washer $\varnothing 10 \times \varnothing 24 \times 2.5$
S 278	320504	Thrust bearing $\varnothing 12 \times \varnothing 26 \times 2$
S 279	320505	Thrust washer $\varnothing 12 \times \varnothing 26 \times 1$
S 283	314501	Lock washer

Dwg #	Cat #	Description
S 285	310401	Lock nut - gripper
S 286	310402	Lock nut - base KM 5
S 288	100706	Washer \varnothing 10.5 x \varnothing 20 x 0.5
S 289	100705	Washer \varnothing 12.5 x \varnothing 22 x 0.5
S 293	319201	Timing belt
S 294	319202	Timing belt
S 295	319203	Timing belt
S 300	315202	Flange - timing belt pulley
S 301	315201	Flange - timing belt pulley
S 308	317501	Pivot pin \varnothing 1/8" x 3/8"
S 309	430901	Motor Gear - base; 127.7:1
S 310	430901	Motor Gear - shoulder/elbow; 127.7:1
S 311	430902	Motor Gear - pitch/wrist 65.5:1
S 312	430903	Motor Gear - gripper
S 313	319001	Coupling
S 315	410802	Limit switch
S 316	310802	Nut for harness
S 317	300006	Harness clamp
S 318	113006	Rubber plug (base)
S 319	300007	Harness clamp
S 320	314007	Conical washer
S 322	113203	Rubber grommet
S 324	113202	O-ring (rubber)
S 325	113204	Rubber stopper
S 350	317801	Roll pin \varnothing 1/8 x 1 1/4
S 351	317502	Ball bearing \varnothing - 3.5 mm
414	105003	Encoder disk (20 slots) - gripper
427	113005	Encoder housing (plastic)
429	105003	Encoder disk (20 slots)







Referencias:

SCORBOT-ER 4U.USER MANUAL.(2001).Intelitek.

Anexo 5. Clase Cinemáticas.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Teleoperacion
{
    abstract class ecuaciones
    {
        private float th1;
        public float Th1
        {
            set
            {
                if (value < 0 || value >=0)
                    th1 = value;
                else
                    th1 = 0;
            }
            get
            {
                return th1;
            }
        }
        private float th2;
        public float Th2
        {
            set
            {
                if (value < 0 || value >= 0)
                    th2 = value;
                else
                    th2 = 0;
            }
            get
            {
                return th2;
            }
        }
        private float th3;
        public float Th3
        {
            set
            {
                if (value < 0 || value >= 0)
                    th3 = value;
                else
                    th3 = 0;
            }
            get
            {
                return th3;
            }
        }
        private float th4;
        public float Th4
        {
            set
            {
                if (value < 0 || value >= 0)
                    th4 = value;
                else
                    th4 = 0;
            }
            get
            {
```

```
        return th4;
    }
}
private float th5;
public float Th5
{
    set
    {
        if (value < 0 || value >= 0)
            th5 = value;
        else
            th5 = 0;
    }
    get
    {
        return th5;
    }
}
private float a_1;
public float a1
{
    set
    {
        if (value < 0)
            a_1 = 0;
        else
            a_1 = value;
    }
    get
    {
        return a_1;
    }
}
private float d_1;
public float d1
{
    set
    {
        if (value < 0)
            d_1 = 0;
        else
            d_1 = value;
    }
    get
    {
        return d_1;
    }
}
private float a_2;
public float a2
{
    set
    {
        if (value < 0)
            a_2 = 0;
        else
            a_2 = value;
    }
    get
    {
        return a_2;
    }
}
private float a_3;
public float a3
{
    set
    {
        if (value < 0)
```

```
        a_3 = 0;
    else
        a_3 = value;
    }
    get
    {
        return a_3;
    }
}
private float d_5;
public float d5
{
    set
    {
        if (value < 0)
            d_5 = 0;
        else
            d_5 = value;
    }
    get
    {
        return d_5;
    }
}
private float a_1s;
public float a1s
{
    set
    {
        if (value < 0)
            a_1s = 0;
        else
            a_1s = value;
    }
    get
    {
        return a_1s;
    }
}
private float d_1s;
public float d1s
{
    set
    {
        if (value < 0)
            d_1s = 0;
        else
            d_1s = value;
    }
    get
    {
        return d_1s;
    }
}
private float a_2s;
public float a2s
{
    set
    {
        if (value < 0)
            a_2s = 0;
        else
            a_2s = value;
    }
    get
    {
        return a_2s;
    }
}
}
```

```

private float a_3s;
public float a3s
{
    set
    {
        if (value < 0)
            a_3s = 0;
        else
            a_3s = value;
    }
    get
    {
        return a_3s;
    }
}
private float d_5s;
public float d5s
{
    set
    {
        if (value < 0)
            d_5s = 0;
        else
            d_5s = value;
    }
    get
    {
        return d_5s;
    }
}
public abstract float rollb();
public abstract float qx();
public abstract float qy();
public abstract float qz();
public abstract float pitch();
public abstract float rolla();
public abstract float Th1s();
public abstract float Th2s();
public abstract float Th3s();
public abstract float Th4s();
public abstract float Th5s();
public abstract float rollbs();
public abstract float qxs();
public abstract float qys();
public abstract float qzs();
public abstract float pitches();
public abstract float rollas();
public double AN1s;
public double AN5s;
public double AN3s;
public double AN2s;
public double AN4s;
public double ux;
public double uy;
public double uz;
public double AN234;
public double k1;
public double k2;
}
class dir_inv : ecuaciones
{
    public dir_inv(float ang1, float ang2, float ang3, float ang4, float ang5, float brz1,
float desp1, float brz2, float brz3, float desp5, float brz1s, float desp1s, float brz2s, float
brz3s, float desp5s )
    {
        Th1 = ang1;
        Th2 = ang2;
        Th3 = ang3;
        Th4 = ang4;
    }
}

```



```

Th5 = ang5;
a1 = brz1;
d1 = desp1;
a2 = brz2;
a3 = brz3;
d5 = desp5;
a1s = brz1s;
d1s = desp1s;
a2s = brz2s;
a3s = brz3s;
d5s = desp5s;
{
    AN1s = Math.Atan((Math.Sin(Th1) * (a1 + (a2 * Math.Cos(Th2)) + (a3 *
Math.Cos(Th2 + Th3)) - (d5 * Math.Sin(Th2 + Th3 + Th4)))) / (Math.Cos(Th1) * (a1 +
(a2 * Math.Cos(Th2)) + (a3 * Math.Cos(Th2 + Th3)) - (d5 * Math.Sin(Th2 + Th3 +
Th4))));
    ux = Math.Cos((Math.Atan2((-Math.Sin(Th1) * Math.Sin(Th2 + Th3 +
Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))), ((-Math.Cos(Th1) *
Math.Sin(Th2 + Th3 + Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4)))))) *
Math.Cos((Math.Acos(-Math.Cos(Th2 + Th3 + Th4)))) *
Math.Cos((Math.Atan2(((Math.Sin(Th5) * Math.Sin(Th2 + Th3 + Th4)) /
(Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))), ((Math.Cos(Th5) * Math.Sin(Th2 +
Th3 + Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4)))))) -
Math.Sin((Math.Atan2((-Math.Sin(Th1) * Math.Sin(Th2 + Th3 + Th4)) /
(Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))), ((-Math.Cos(Th1) * Math.Sin(Th2
+ Th3 + Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4)))))) *
Math.Sin((Math.Atan2(((Math.Sin(Th5) * Math.Sin(Th2 + Th3 + Th4)) /
(Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))), ((Math.Cos(Th5) * Math.Sin(Th2 +
Th3 + Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))))));
    uy = Math.Sin((Math.Atan2((-Math.Sin(Th1) * Math.Sin(Th2 + Th3 +
Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))), ((-Math.Cos(Th1) *
Math.Sin(Th2 + Th3 + Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4)))))) *
Math.Cos(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))) *
Math.Cos((Math.Atan2(((Math.Sin(Th5) * Math.Sin(Th2 + Th3 + Th4)) /
(Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))), ((Math.Cos(Th5) * Math.Sin(Th2 +
Th3 + Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4)))))) +
Math.Cos((Math.Atan2((-Math.Sin(Th1) * Math.Sin(Th2 + Th3 + Th4)) /
(Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))), ((-Math.Cos(Th1) * Math.Sin(Th2
+ Th3 + Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4)))))) *
Math.Sin((Math.Atan2(((Math.Sin(Th5) * Math.Sin(Th2 + Th3 + Th4)) /
(Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))), ((Math.Cos(Th5) * Math.Sin(Th2 +
Th3 + Th4)) / (Math.Sin(Math.Acos(-Math.Cos(Th2 + Th3 + Th4))))))));
    AN5s = Math.Asin(ux * Math.Sin(AN1s) - uy * Math.Cos(AN1s));
    uz = -Math.Sin(Th2 + Th3 + Th4) * Math.Cos(Th5);
    AN234 = Math.Atan2(-uz / Math.Cos(AN5s), (ux * Math.Cos(AN1s) + uy *
Math.Sin(AN1s)) / Math.Cos(AN5s));
    k1 = ((Math.Cos(Th1) * (a1 + (a2 * Math.Cos(Th2)) + (a3 *
Math.Cos(Th2 + Th3)) - (d5 * Math.Sin(Th2 + Th3 + Th4))) * Math.Cos(AN1s)) +
((Math.Sin(Th1) * (a1 + (a2 * Math.Cos(Th2)) + (a3 * Math.Cos(Th2 + Th3)) - (d5 *
Math.Sin(Th2 + Th3 + Th4)))) * Math.Sin(AN1s)) - a1s + (d5s * Math.Sin(AN234));
    k2 = (-(d1 - (a2 * Math.Sin(Th2)) - (a3 * Math.Sin(Th2 + Th3)) - (d5
* Math.Cos(Th2 + Th3 + Th4))) + d1s - (d5s * Math.Cos(AN234));
    AN3s = (Math.Acos((Math.Pow(k1, 2) + Math.Pow(k2, 2) - Math.Pow(a2s,
2) - Math.Pow(a3s, 2)) / (2 * a2s * a3s)));
    AN2s = (Math.Atan2((-k1 * a3s * Math.Sin(AN3s) + k2 * (a2s + a3s *
Math.Cos(AN3s))) / (Math.Pow(a2s, 2) + Math.Pow(a3s, 2) + 2 * a2s * a3s *
Math.Cos(AN3s)), (k1 * (a2s + a3s * Math.Cos(AN3s)) + k2 * a3s * Math.Sin(AN3s)) /
(Math.Pow(a2s, 2) + Math.Pow(a3s, 2) + 2 * a2s * a3s * Math.Cos(AN3s))));
    AN4s = (AN234 - AN2s - AN3s);

```

```
        return Convert.ToSingle(Math.Atan2(((Math.Sin(AN5s) * Math.Sin(AN2s
+ AN3s + AN4s)) / (Math.Sin(Math.Acos(-Math.Cos(AN2s + AN3s + AN4s))))),
((Math.Cos(AN5s) * Math.Sin(AN2s + AN3s + AN4s)) / (Math.Sin(Math.Acos(-
Math.Cos(AN2s + AN3s + AN4s)))))));
    }
```

Anexo 6. Adquisición de video.

```

#region WebCam API
    const short WM_CAP = 1024;
    const int WM_CAP_DRIVER_CONNECT = WM_CAP + 10;
    const int WM_CAP_DRIVER_DISCONNECT = WM_CAP + 11;
    const int WM_CAP_EDIT_COPY = WM_CAP + 30;
    const int WM_CAP_SET_PREVIEW = WM_CAP + 50;
    const int WM_CAP_SET_PREVIEWRATE = WM_CAP + 52;
    const int WM_CAP_SET_SCALE = WM_CAP + 53;
    const int WS_CHILD = 1073741824;
    const int WS_VISIBLE = 268435456;
    const short SWP_NOMOVE = 2;
    const short SWP_NOSIZE = 1;
    const short SWP_NOZORDER = 4;
    const short HWND_BOTTOM = 1;
    int iDevice = 0;
    int hHwnd;
    [System.Runtime.InteropServices.DllImport("user32", EntryPoint =
"SendMessageA")]
    static extern int SendMessage(int hwnd, int wMsg, int wParam,
[MarshalAs(UnmanagedType.AsAny)]
    object lParam);
    [System.Runtime.InteropServices.DllImport("user32", EntryPoint =
"SetWindowPos")]
    static extern int SetWindowPos(int hwnd, int hWndInsertAfter, int x, int y,
int cx, int cy, int wFlags);
    [System.Runtime.InteropServices.DllImport("user32")]
    static extern bool DestroyWindow(int hwnd);
    [System.Runtime.InteropServices.DllImport("avicap32.dll")]
    static extern int capCreateCaptureWindowA(string lpszWindowName, int
dwStyle, int x, int y, int nWidth, short nHeight, int hWndParent, int nID);
    [System.Runtime.InteropServices.DllImport("avicap32.dll")]
    static extern bool capGetDriverDescriptionA(short wDriver, string lpszName,
int cbName, string lpszVer, int cbVer);
    private void OpenPreviewWindow()
    {
        int iHeight = 320;
        int iWidth = 200;
        //
        // Open Preview window in picturebox
        //
        hHwnd = capCreateCaptureWindowA(iDevice.ToString(), (WS_VISIBLE |
WS_CHILD), 0, 0, 640, 480, pantalla.Handle.ToInt32(), 0);
        //
        // Connect to device
        //
        if (SendMessage(hHwnd, WM_CAP_DRIVER_CONNECT, iDevice, 0) == 1)
        {
            //
            // Set the preview scale
            //
            SendMessage(hHwnd, WM_CAP_SET_SCALE, 1, 0);
            //
            // Set the preview rate in milliseconds
            //
            SendMessage(hHwnd, WM_CAP_SET_PREVIEWRATE, 66, 0);
            //

```

```

        // Start previewing the image from the camera
        //
        SendMessage(hHwnd, WM_CAP_SET_PREVIEW, 1, 0);
        //
        // Resize window to fit in picturebox
        //
        SetWindowPos(hHwnd, HWND_BOTTOM, 0, 0, iWidth, iHeight, (SWP_NOMOVE
| SWP_NOZORDER));
    }
    else
    {
        //
        // Error connecting to device close window
        //
        DestroyWindow(hHwnd);
    }
}
private void ClosePreviewWindow()
{
    //
    // Disconnect from device
    //
    SendMessage(hHwnd, WM_CAP_DRIVER_DISCONNECT, iDevice, 0);
    //
    // close window
    //
    DestroyWindow(hHwnd);
}

```

Referencias:

http://www.codeproject.com/KB/IP/Video_Voice_Conferencing.aspx