UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA MECATRÓNICA

SISTEMA DE SEGUIMIENTO INTELIGENTE PARA UN ROBOT MANIPULADOR

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO MECATRÓNICO

RODRIGO IVÁN ULLAURI GUARANGA

DIRECTOR: PhD. FAUSTO FREIRE

Quito, Octubre 2014

DECLARACIÓN

Yo **RODRIGO IVÁN ULLAURI GUARANGA**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Rodrigo Iván Ullauri Guaranga

C.I. 1721899696

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título "**Sistema de Seguimiento Inteligente para un Robot Manipulador**", que, para aspirar al título de **Ingeniero Mecatrónico** fue desarrollado por **Rodrigo Iván Ullauri Guaranga**, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 18 y 25.

_____

Fausto Freire Carrera

DIRECTOR DEL TRABAJO

C.I. 1802424737

INDICE DE CONTENIDOS

RESUMEN..... vi
ABSTRACT vii
1. INTRODUCCIÓN.....1
OBJETIVO GENERAL.....3
OBJETIVOS ESPECÍFICOS.....3
2. MARCO TEÓRICO.....5
2.1. ROBOT.....6
2.2. CLASIFICACIÓN DE LOS ROBOTS.....6
2.3. ROBOTS SEGÚN SU CONFIGURACIÓN MECÁNICA.....9
2.3.1. Robot Cartesiano.....9
2.3.2. Robot Cilíndrico.....10
2.3.3. Robot Esférico.....11
2.3.4. Robot SCARA.....12
2.3.5. Robot Articulado.....12
2.3.6. Robot Antropomórfico.....13
2.4. CINEMÁTICA DEL ROBOT.....14
2.4.1. Cinemática directa.....15
2.4.2. Cinemática inversa.....17
2.4.3. Relación entre cinemática directa e inversa.....18
2.5. MÉTODOS DE PROGRAMACIÓN DE ROBOTS.....19
2.5.1. Programación de robots “on-line”.....19
2.5.2. Programación de robots “off-line”.....19
2.5.3. Programación online vs offline.....20
2.5.4. Programación de robots híbrida (mixta).....21
2.5.5. Programación por aprendizaje inmediato.....21
2.5.6. Programación de robots punto a punto.....21
2.6. ROBOT MANIPULADOR.....22
2.6.1. Características mitsubishi melfa rv2aj.....23
2.7. INTELIGENCIA ARTIFICIAL (IA).....24
2.8. ANTECEDENTES DE LA INTELIGENCIA ARTIFICIAL.....25
2.9. RAMAS DE LA INTELIGENCIA ARTIFICIAL.....26

2.10. VISIÓN ARTIFICIAL 26
2.10.1. Etapas del Sistema De Visión Artificial 27
2.10.2. Elementos del sistema de Visión Artificial. 28
2.11. COMUNICACIÓN SERIAL, RS-232 30
2.11.1. Comunicación asíncrona 31
2.11.2. Comunicación síncrona 32
2.12. KINECT 33
2.13. COMPONENTES DEL SENSOR KINECT 34
2.13.1. Cámara rgb 34
2.13.2. Emisor de infrarrojo 35
2.13.3. Motor 36
2.13.4. Arreglo de micrófonos 37
2.13.5. Led 37
2.14. KINECT PARA WINDOWS VS KINECT PARA XBOX 360 38
2.15. CARACTERÍSTICAS DEL SDK 38
3. METODOLOGÍA 40
3.1. METODOLOGIA MECATRÓNICA 41
3.2. INVESTIGACION DEL PROBLEMA 43
3.3. REQUERIMIENTOS DEL SISTEMA 44
3.4. DISEÑO MECÁNICO 45
3.5. DISEÑO ELECTRÓNICO 45
3.6. DISEÑO DEL SISTEMA DE CONTROL 46
3.7. SIMULACIÓN 46
3.8. CONSTRUCCIÓN DEL PROTOTIPO 46
3.9. PRUEBAS 47
4. DISEÑO DEL SISTEMA 48
4.1. DISEÑO DEL SISTEMA 49
4.2. SUBSISTEMA DE VISIÓN ARTIFICIAL 51
4.2.1. Captura de imágenes 51
4.2.2. Flujo de Imágenes 52
4.2.3. Tipos de formatos en flujo de imagen 52
4.2.4. Transmisión de imágenes 55

4.2.5. Reconocimiento de articulaciones 57
4.2.6. Análisis de Posiciones de las extremidades del usuario 63
4.2.7. Cálculo de Vectores Posición 64
4.2.8. Análisis trigonometrico 67
4.2.9. Obtención de vectores 68
4.2.10. Vectores entre Articulaciones 69
4.3. SUBSISTEMA DE CONTROL 73
4.3.1. Descripción de la unidad controladora CR1-571 73
4.3.2. Comunicación entre Pc y controlador CR1-571 74
4.4. SUBSISTEMA MECÁNICO 78
4.4.1. Descripción del robot manipulador Mitsubishi Melfa RV2AJ 78
4.5. CINEMÁTICA ROBOT MANIPULADOR MELFA RV2AJ 79
4.5.1. Cinemática Directa 79
4.6. SISTEMA DE SEGUIMIENTO INTELIGENTE 84
4.6.1. Consolidación de los subsistemas 84
5. IMPLEMENTACION Y PRUEBAS 85
5.1. IMPLEMENTACION 86
5.1.1. Interfaz Aplicación 86
5.1.2. Comunicación Serial 87
5.1.3. Sistema de Visión Artificial 89
5.1.4. Sistema de seguimiento inteligento 90
5.2. PRUEBAS 91
5.2.1. Consideraciones generales 91
5.2.2. Inicializacion del sistema 92
5.2.3. Reconocimiento de Posiciones 93
6. CONCLUSIONES Y RECOMENDACIONES 96
6.1. CONCLUSIONES 97
6.2. RECOMENDACIONES 98
BIBLIOGRAFÍA 101

INDICE DE FIGURAS

Figura 2.1. Robot Cartesiano y Volumen de trabajo. 10
Figura 2.2. Robot Cilíndrico y Volumen de trabajo. 10
Figura 2.3. Robot Esférico y Volumen de trabajo. 11
Figura 2.4. Robot SCARA. 12
Figura 2.5. Robot Articulado (Mitsubishi RV2AJ). 13
Figura 2.6. Robot y Mano Antropomórfico (Robot HONDA-ASIMO). 14
Figura 2.7. Cinemática directa de brazo robótico. 15
Figura 2.8. Descripción Cinemática Inversa. 17
Figura 2.9. Relación entre análisis cinemático. 18
Figura 2.10. Morfología del brazo robótico. 22
Figura 2.11. Diagrama de control de posicionamiento en lazo cerrado. 23
Figura 2.12. Etapas de visión artificial. 28
Figura 2.13. Componentes del sensor Kinect. 34
Figura 2.14. Placa base sensor Kinect. 34
Figura 2.15. Angulo de Visibilidad 35
Figura 2.16. Emisor IR y Sensor Profundidad 36
Figura 2.17. Ángulos de Inclinación 37
Figura 2.18. Arreglo de Micrófonos 37
Figura 3.1. Disciplinas Involucradas en la Metodología Mecatrónica 42
Figura 3.2. Descripción diseño paralelo 42
Figura 4.1. Arquitectura del Sistema Mecatrónico 49
Figura 4.2. Requisitos del Sistema. 51
Figura 4.3. Arreglo RGB. 53
Figura 4.4. Filtro Bayer. 54
Figura 4.5. Método por Evento 55
Figura 4.6. Método por Demanda. 56
Figura 4.7. Proceso y código para captura de imágenes 57
Figura 4.8. Procesamiento del sensor de profundidad. 58
Figura 4.9. Proceso de reconocimiento de usuario y articulaciones. 59
Figura 4.10. Ubicación espacial del cuerpo humano respecto a Kinect 61
Figura 4.11. Reconocimiento y ubicación de articulaciones. 61
Figura 4.12. Jerarquía de articulaciones 62
Figura 4.13. Vectores posición respecto al origen (Kinect). 64
Figura 4.14. Vector posición entre articulaciones. 65
Figura 4.15. Suma de vectores método gráfico. 65
Figura 4.17. Adaptación de la morfología del robot 67
Figura 4.18. Vectores del sistema de visión artificial. 69
Figura 4.19 Relación de ángulos entre usuario y robot 70
Figura 4.20. Diagrama de flujo para el cálculo de ángulos 72
Figura 4.21. Diagrama de flujo Comunicación PC - Unidad Controladora 75

Figura 4.22. Diagrama de flujo envió de datos....77
Figura 4.23. Comunicación Pc - Unidad controladora robot....78
Figura 4.24. Robot Mitsubishi Melfa RV2AJ....79
Figura 4.25. Robot Mitsubishi Melfa RV2AJ aplicado Denavit-Hartenberg...80
Figura 4.26. Orientación de la pinza. 83
Figura 4.27. Consolidación de Subsistemas. 84
Figura 5.1. Interfaz Principal de la aplicación. 86
Figura 5.2. Adaptador USB a serial. 88
Figura 5.3. Puerto de comunicación de la unidad controladora. 88
Figura 5.4. Switch maestro. 89
Figura 5.5. Sistema de visión artificial. 90
Figura 5.6. Implementación del sistema mecatrónico. 90
Figura 5.7. Posición 1. 93
Figura 5.8. Posición 2. 93
Figura 5.9. Posición 3. 94
Figura 5.10. Posición 4. 94

INDICE DE TABLAS

Tabla 2.1. Clasificación de Robots por Generaciones....7
Tabla 2.2. Clasificación general de los robots. 8
Tabla 2.3. Clasificación de Robots Según la AFRI. 9
Tabla 4.1. Algoritmo de acceso a coordenadas espaciales. 64
Tabla 4.2. Código de acceso a coordenadas espaciales. 68
Tabla 4.4. Parámetros de comunicación. 74
Tabla 4.5. Comandos de control. 76
Tabla 4.7. Parámetros Denavit-Hartenberg para el brazo robótico. 81
Tabla 5.1. Comparación de Posiciones: Robot vs Operario. 95
Tabla 5.2. Tiempo de Respuesta. 95

INDICE DE ANEXOS

ANEXO I Características del robot Mitsubishi Melfa RV2AJ 105
ANEXO II Características de la unidad controladora CR1-571. 106

RESUMEN

El manejo y correcta programación de un robot industrial se puede convertir en una tarea muy compleja, a más de requerir mucho tiempo se necesita experiencia para lograr los movimientos necesarios para completar una secuencia de movimientos; el objetivo de este proyecto fue desarrollar un Sistema Inteligente de Seguimiento para el robot Mitsubishi Melfa RV2AJ, el mismo que analiza los movimientos de las extremidades superiores, para lo cual se utilizó un sistema de visión artificial.

El costo bajo moderado y la gran capacidad de análisis de imágenes, convierten a los sistemas basados en visión artificial en una herramienta eficaz para detectar o reconocer los movimientos del cuerpo humano.

Teniendo en cuenta lo anterior, se debió escoger un dispositivo capaz de captar imágenes de una manera idónea, en el proyecto se empleó el dispositivo Kinect, ya que presentó una gran flexibilidad para el análisis de imágenes.

El proyecto se desarrolló bajo la metodología mecatrónica en donde se investigó sobre el funcionamiento y programación del brazo robótico. Se desarrolló una aplicación para el control del robot utilizando el SDK de Kinect.

Por último, se realizaron varias pruebas para evaluar el sistema propuesto. Los resultados de las pruebas realizadas fueron satisfactorios cumpliendo con los objetivos.

ABSTRACT

The management and proper programming of an industrial robot can become a very complex task, more time consuming experience necessary to achieve the moves necessary to complete a sequence of movements; The objective of this project is to develop an Intelligent Monitoring System for Mitsubishi Melfa robot RV2AJ, which analyze the movements of the upper extremities, which is intended to use a machine vision system.

Moderate low cost and high capacity image analysis, make artificial vision based on an effective way to detect or recognize the tool movements of the human body systems.

Given the above, one must choose a device capable of capturing images in a convenient way, this project is the Kinect device was used, since it has great flexibility for image analysis.

The project was developed under the mechatronics methodology which investigated on the operation and programming of the robot arm. An application for robot control was developed using the Kinect SDK.

Finally, several tests were conducted to evaluate the proposed system. The results of the tests were satisfactory meeting the objectives.

1. INTRODUCCIÓN

Hoy en día la robótica y los sistemas inteligentes son muy importantes en la industria debido a las grandes ventajas que ofrecen frente a un operador humano como son: la capacidad de realizar tareas repetitivas con gran precisión y velocidad, trabajar por largas jornadas, el mejoramiento del control de calidad de productos mediante la implementación de sistemas inteligentes.

Estas ventajas han disminuido los costos de producción e incrementado la calidad de los productos.

La evolución hacia sistemas robóticos autónomos se debe al desarrollo de los sistemas inteligentes (Lógica Difusa, Redes Neuronales, Visión Artificial) los cuales permiten que los sistemas robóticos puedan responder a diferentes situaciones de una manera más eficaz, es decir que tienen la capacidad de tomar decisiones.

La principal aplicación de los robots tiene lugar en la industria, donde es habitual que se elija un brazo robótico, el cual se acople a las necesidades para realizar sobre todo tareas repetitivas como la fabricación de piezas en serie.

Pero a pesar de que un robot es un sistema con muchas ventajas, aun es necesario que el operario del robot cuente con un alto nivel de conocimiento en lenguajes de programación para poder controlarlo de una manera correcta.

Actualmente las empresas más renombradas en el diseño y fabricación de sistemas robóticos los cuales son:

- KUKA Robotics.
- ABB Robotics.
- Fanuc.
- Motoman.
- Yaskawa.
- Mitsubishi Robotics.

Las empresas mencionadas anteriormente diseñan interfaces de control mucho más amigables e intuitivas para el manejo de sus robots, entre los sistemas más notorios se encuentran los siguientes:

- Joysticks.
- Pantallas Táctiles.
- Interfaces Hápticas.
- Programación en lenguaje gráfico.

Estas interfaces permiten que el operario del robot no deba ser un experto en programación para poder manejarlo y resolver los diferentes problemas que puedan existir en el proceso.

De esta manera los sistemas robóticos se convierten en sistemas mucho más flexibles ya que permiten realizar cambios en su programa de control de una manera más rápida y fácil para el operario.

Por estos motivos, para el presente proyecto se plantearon los siguientes objetivos:

OBJETIVO GENERAL

Diseñar un sistema inteligente que controle los movimientos de un brazo robótico didáctico (Mitsubishi Melfa RV2AJ), a partir de la posición de las extremidades superiores del operario

OBJETIVOS ESPECÍFICOS

- Desarrollar un software de captación y reconocimiento de imágenes.
- Diseñar un sistema de control para el posicionamiento del robot.
- Desarrollar un algoritmo de comunicación entre el software de captación de imágenes y el robot.

El presente proyecto de titulación está enfocado hacia el desarrollo de sistemas robóticos tele operados mediante visión artificial, es decir que el robot pueda ubicarse en su espacio de trabajo sin la necesidad de tener una programación previa en su unidad controladora, y todo el control del robot dependa únicamente del software de análisis y reconocimiento de imágenes, el cual captura y analiza las posiciones espaciales de las articulaciones (hombro, codo, muñeca, mano) del brazo derecho del operario, para luego procesar todas estas posiciones y relacionarlas directamente con la morfología del robot, en este caso un robot de cinco grados de libertad (Mitsubishi Melfa RV2AJ), el cual se comunica directamente con el software de visión artificial y de esta manera reproduce todos los movimientos que realiza el operario con el brazo derecho, respetando las limitaciones mecánicas que tiene el brazo robótico, se puede apreciar en el Anexo 1.

Luego de implementar el sistema se procederá a realizar pruebas para determinar el tiempo y exactitud en los movimientos del robot.

2. MARCO TEÓRICO

Los robots a nivel tecnológico son los más utilizados en la industria debido a sus funcionalidades y practicidad en los procesos de manufactura, gracias al nivel de trabajo y tiempo de producción que alcanzan, se ha incrementado la implementación a nivel mundial, permitiendo al ser humano preocuparse de labores con más raciocinio como gestión de la producción o proyección de ventas, debido a estas razones se tienen una amplia gama de categorías y funcionalidades a nivel mundial (Ollero, 2001).

2.1. ROBOT

Es un sistema automático servo controlado, reprogramable, polivalente, capaz de orientar y posicionar piezas, útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables para la ejecución de tareas variadas. Normalmente tiene la forma de uno o varios brazos terminados en una muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción de entorno (Barrientos, 2007).

Un robot está formado por los siguientes elementos:

- Estructura mecánica
- Transmisores
- Sistema de Accionamiento
- Sistema Sensorial
- Sistema de Control
- Elementos Terminales

2.2. CLASIFICACIÓN DE LOS ROBOTS

A continuación se cataloga a los robots de acuerdo a:

- Generación.
- Tipo.
- Asociación Francesa de Robótica Industrial.

Tabla 2.1. Clasificación de Robots por Generaciones

GENERACIÓN	DEFINICIÓN
1G	Adquiere Información muy limitada de acuerdo a su entorno y acorde a esta actúa, repite tareas programadas y actúa en consecuencia a los sucesos.
2G	Adquieren también Información limitada de su entorno y el movimiento lo controla a través de una secuencia numérica almacenada en disco o cinta magnética; se utiliza en la industria automotriz y son de gran tamaño.
3G	Son reprogramables, estos incluyen todos los avances de las dos generaciones anteriores; utilizan las computadoras para su control y tienen cierta percepción de su entorno a través del uso de sensores; con esta generación se inicia la era de los robots inteligentes y aparecen los lenguajes de programación para escribir los programas de control que se le introducen a cada uno de ellos.
4G	Son robots altamente inteligentes con mejores sistemas sensoriales, para entender sus acciones y captar el mundo que los rodea o entorno además incorporan conceptos “modélicos” de conducta para ser capaces de actuar ante circunstancias determinadas.
5G	Actualmente se encuentran en desarrollo, pero dando pasos gigantescos en este tipo de generación de robots que serán los que nos acompañen en el futuro en todas nuestras actividades cotidianas y darán paso a una nueva era.

Fuente: (Barrientos, 2007)

Tabla 2.2. Clasificación general de los robots.

ROBOTS	DEFINICIÓN	TIPOS
Robot Industrial	Se comprende a una máquina de manipulación automática reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos especiales para la ejecución de trabajos diversos en las diferentes etapas de la producción industrial, ya sea en una posición fija o en movimiento.	• Robot Secuencial • Robot de Trayectoria Controlable • Robot Adaptivo • Robot Telemanipulado
Robot de Servicio	Dispositivos electromecánicos móviles o estacionarios dotados normalmente de uno o varios brazos mecánicos independientes controlados por un programa y que realizan tareas de servicio.	• Robots Médicos • Robots Domésticos
Robot Teleoperado	Dispositivos robóticos con brazos manipuladores, sensores y cierto grado de movilidad, controlados remotamente por un operador humano de manera directa o a través de un ordenador.	• Robot para Manejo de Materiales Peligrosos

Fuente: (Barrientos, 2007).

Tabla 2.3. Clasificación de Robots Según la AFRI.

TIPO	DEFINICIÓN
Tipo A	Manipulador con control manual o telemando.
Tipo B	Manipulador automático con ciclos pre ajustados, regulación mediante fines de carrera o topes; control por PLC; accionamiento neumático; eléctrico o hidráulico.
Tipo C	Robot programable con trayectoria continua o punto a punto. Carece de conocimientos sobre su entorno.
Tipo D	Robot capaz de adquirir datos de su entorno, readaptando su tarea en función de éstos.

Fuente: (Barrientos, 2007).

2.3. ROBOTS SEGÚN SU CONFIGURACIÓN MECÁNICA

2.3.1. ROBOT CARTESIANO

Es también llamado robot lineal, un robot cartesiano es un robot compuesto principalmente por tres ejes lineales, es decir únicamente cuentan con articulaciones prismáticas (únicamente se mueven en línea recta, no tienen rotaciones) y forman ángulos rectos uno respecto al otro. Además de otras características, esta configuración mecánica tiene un volumen de trabajo en forma cúbica y simplifica las ecuaciones en el control de los brazos robóticos.

Este tipo de configuración se utiliza sobre todo para tomar y ubicar objetos en el lugar de trabajo, también se utiliza en fresado y trefiladoras donde se maneja una herramienta para crear un diseño preciso. La forma habitual para controlar este tipo de brazo es una máquina de control numérico (Máquina CNC) (Clenet, 2012).

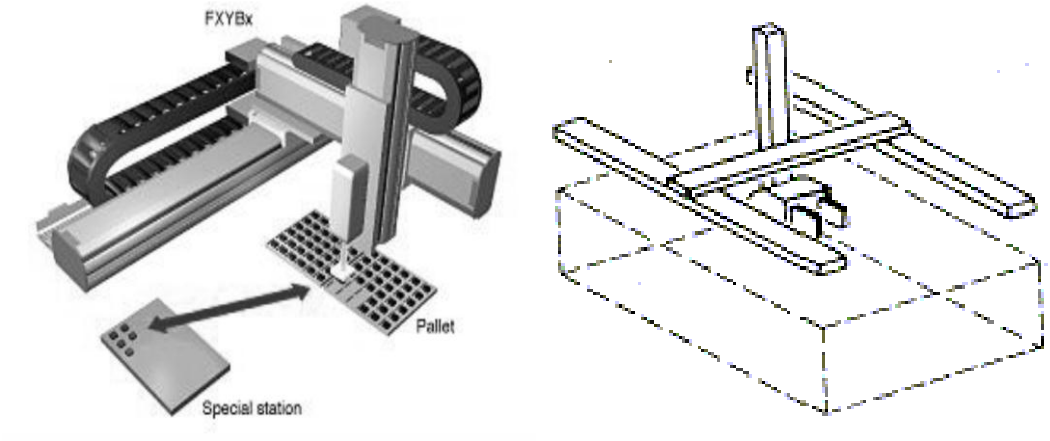

Figura 2.1. Robot Cartesiano y Volumen de trabajo.

Fuente: (Clenet, 2012).

2.3.2. ROBOT CILÍNDRICO

Este tipo de robot es muy similar al robot cartesiano. De hecho, el sistema se basa sólo en la rotación en lugar de la traslación. Es una combinación de revoluta y articulaciones prismáticas. En consecuencia, su volumen de trabajo es un cilindro. Se utiliza para las operaciones de montaje piezas y ubicación de herramientas para realizar tareas específicas, tiene casi las mismas aplicaciones que el robot cartesiano (Clenet, 2012).

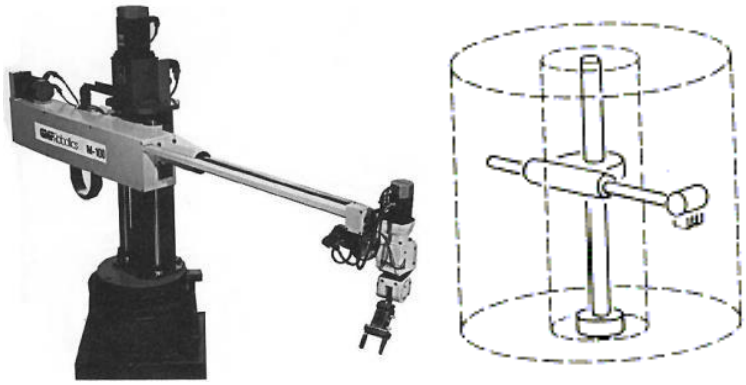

Figura 2.2. Robot Cilíndrico y Volumen de trabajo.

Fuente: (Clenet, 2012).

2.3.3. ROBOT ESFÉRICO

Mejor conocido como robot Polar, la ventaja de este tipo de brazo robótico es la configuración de sus juntas las que permiten que el robot pueda rotar y trasladar su efector final para así alcanzar su objetivo.

De hecho, los ejes del robot forman un sistema de coordenadas esféricas. El sistema básico de un robot polar está compuesto por al menos dos juntas rotativas y uno lineal con el fin de formar el volumen de trabajo (Esférico). Obviamente, se pueden añadir más articulaciones para más flexibilidad sin cambiar el área de trabajo. La mayoría de ellos tienen entre cuatro y seis juntas de revolución (Clenet, 2012).

Se utiliza para el manejo de máquinas herramientas, soldadura de gas o soldadura de arco.

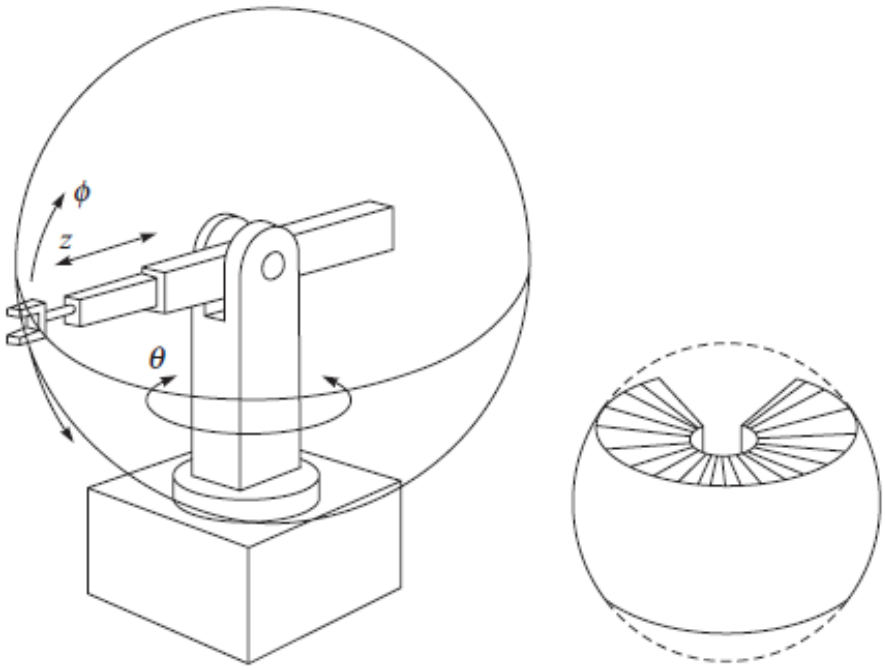

Figura 2.3. Robot Esférico y Volumen de trabajo.

Fuente: (Clenet, 2012).

2.3.4. ROBOT SCARA (SELECTIVE COMPLIANCE ASSEMBLY ROBOT ARM)

Es un robot de cuatro grados de libertad con posicionamiento horizontal. Los Robots SCARA se conocen por sus rápidos ciclos de trabajo, excelente repetitividad, gran capacidad de carga y su amplio campo de aplicación (Clenet, 2012).

De acuerdo con su nombre, el brazo de robot es rígido en el eje Z y flexible en los ejes X-Y. Esto es muy ventajoso para muchos tipos de montaje.

En general es más rápido y más exacto que los sistemas de robots cartesianos, aunque sigue siendo más costoso y requiere un software que controla el uso de la cinemática inversa para interpolación lineal de su movimiento.

Figura 2.4. Robot SCARA.

Fuente: (Clenet, 2012).

2.3.5. ROBOT ARTICULADO

Este tipo de brazo robótico se parece mucho a un brazo humano, debido a que sus articulaciones de giro le permiten acceder a su espacio de trabajo, hecho de al menos tres juntas rotativas, los robots articulados pueden estar

compuestos por diez o más articulaciones que interactúan entre sí para hacer el sistema más complejo. A medida que aumenta la cadena cinemática, cada articulación soporta otra articulación dando aún una mayor flexibilidad al movimiento del brazo (Clenet, 2012).

Los robots articulados son capaces de realizar muchas tareas diferentes por ejemplo trabajos de pintura, soldadura y hasta mecanizado según sus grados de libertad.

Figura 2.5. Robot Articulado (Mitsubishi RV2AJ).

Fuente: (Clenet, 2012).

2.3.6. ROBOT ANTROPOMÓRFICO

El robot antropomórfico no tiene propiedades típicas o características generales, excepto que su forma es semejante a un ser humano. Es decir tiene características relativas como una mano con cinco dedos o un cuerpo con dos piernas.

Los robots antropomórficos se utilizan principalmente para interactuar con humanos (Clenet, 2012).

Figura 2.6. Robot y Mano Antropomórfico (Robot HONDA-ASIMO).

Fuente: (Clenet, 2012).

2.4. CINEMÁTICA DEL ROBOT

La cinemática es la ciencia del movimiento que trata el tema sin considerar las fuerzas que lo ocasionan.

Teniendo en cuenta lo anterior se establece que la cinemática en un robot estudia el movimiento del mismo con respecto a un sistema de referencia.

Y se enfoca en la descripción analítica del movimiento espacial del robot como una función del tiempo y en particular por las relaciones entre la posición y la orientación del extremo final del robot respecto a las posiciones que toman sus articulaciones

En la cinemática del robot existen dos problemas fundamentales, el primero es la cinemática directa que consiste en determinar cuál es la posición y orientación del extremo final del robot y el segundo es la cinemática inversa que resuelve la configuración que debe de adoptar el robot para ubicarse en una posición (Barrientos, 2007).

2.4.1. CINEMÁTICA DIRECTA

El modelo cinemático directo, permite determinar la posición y orientación que adopta el extremo del robot, cuando cada una de las variables que fijan la posición u orientación de sus articulaciones toman valores determinados.

En 1955 Denavit y Hartenberg propusieron un método sistemático para describir y representar la geometría espacial de los elementos de una cadena cinemática (de un robot) con respecto a un sistema de referencia fijo.

Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciendo el problema cinemático directo al encontrar una matriz de transformación homogénea 4x4 que relacione la localización espacial del extremo del robot, con respecto a un sistema de coordenadas en su base (Barrientos, 2007).

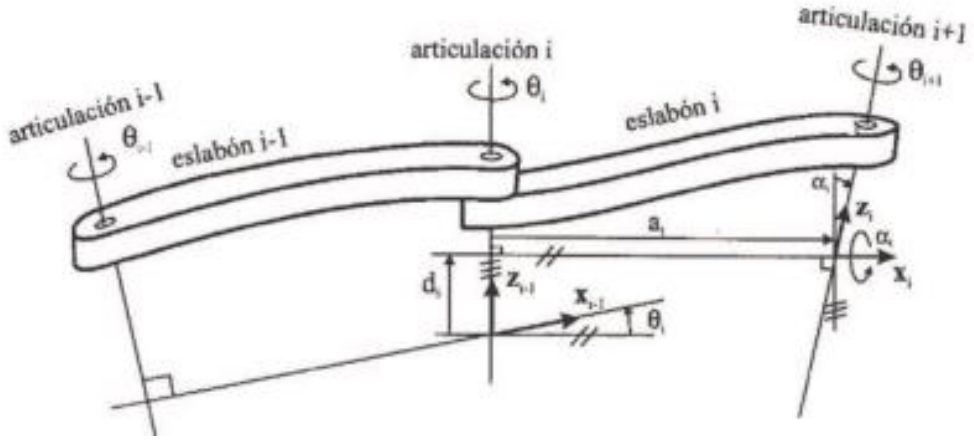

Figura 2.7. Cinemática directa de brazo robótico.

Fuente: (Barrientos, 2007).

Según la representación propuesta por Denavit-Hartenberg, será posible pasar de un eslabón al siguiente mediante cuatro transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento, i-1 con el sistema del elemento i.

Estas transformaciones son:

1. Rotaciones alrededor del eje Z_{i-1} un ángulo θ_i.
2. Traslación a lo largo de una Z_{i-1} distancia d_i, vector d_i (0, 0, d_i).
3. Traslación a lo largo de X_i una distancia a_i , vector a_i (0, 0, a_i).
4. Rotación alrededor del eje X_i un ángulo α_i .

Los cuatro parámetros de Denavit-Hartenberg dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que lo unen con el anterior y siguiente:

$\boldsymbol{\theta_i}$ Es el ángulo que forman los ejes x_{i-1} y x_i medido en un plano perpendicular al eje z_{i-1}, utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.

$\boldsymbol{d_i}$ Es la distancia a lo largo del eje z_{i-1} desde el origen del sistema de coordenadas (i-1) – ésimo hasta la intersección del eje z_{i-1} con el eje x_i. Se trata de un parámetro variable en articulaciones prismáticas.

$\boldsymbol{a_i}$ Es la distancia a lo largo del eje x_i que va desde la intersección del eje z_{i-1} con el eje x_i hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula la distancia más corta entre los ejes z_{i-1} y z_i

$\boldsymbol{\alpha_i}$ Es el ángulo de separación del eje z_{i-1} y el eje z_i, medido en un plano perpendicular al eje x_i, utilizando la regla de la mano derecha.

2.4.2. CINEMÁTICA INVERSA

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot.

$$q = [q1, q2, q3\ldots. qn]$$

Para que su extremo se posicione y oriente según una determinada localización espacial.

Así cómo es posible abordar el problema cinemático directo de una manera sistemática a partir de la utilización de matrices de transformación homogéneas, e independiente de la configuración del robot, no ocurre lo mismo con el problema cinemático inverso, siendo el procedimiento de obtención de las ecuaciones fuertemente dependiente de la configuración del robot (Barrientos, 2007).

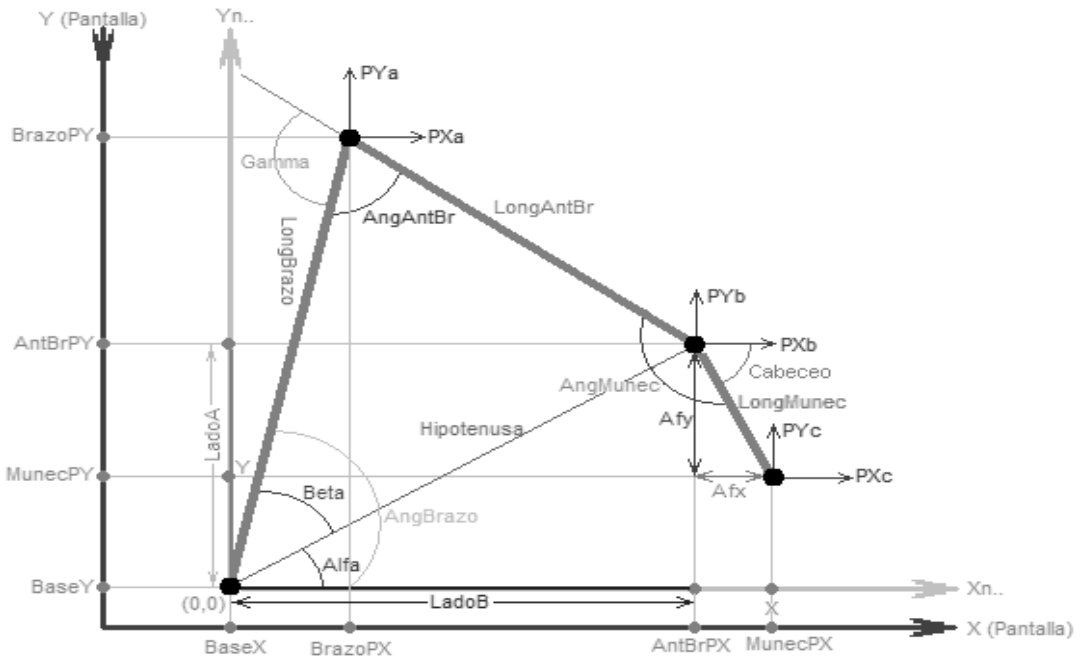

Figura 2.8. Descripción Cinemática Inversa.

Fuente: (Freire, 2012)

A la hora de resolver el problema cinemático inverso es mucho más adecuado encontrar una solución cerrada, esto es, encontrar una relación matemática explícita, este tipo de solución presenta las siguientes ventajas:

- En muchas aplicaciones, el problema cinemático inverso ha de resolverse en tiempo real. Una solución de tipo interactivo no garantiza tener la solución en el momento adecuado.

- Al contrario de lo que ocurría en el problema cinemático directo, con cierta frecuencia la solución del problema cinemático inverso no es única; existiendo diferentes soluciones que posicionan y orientan el extremo del robot del mismo modo. En estos casos una solución cerrada permite incluir determinadas reglas o restricciones que aseguren que la solución obtenida sea la más adecuada de entre las posibles.

Los métodos geométricos permiten obtener normalmente los valores de las primeras variables articulares, que son las que consiguen posicionar el brazo robótico. Para ello se usa relaciones trigonométricas y geométricas sobre los elementos del brazo, en ocasiones se suele recurrir a la resolución de triángulos formados por elementos y articulaciones del robot.

2.4.3. RELACIÓN ENTRE CINEMÁTICA DIRECTA E INVERSA

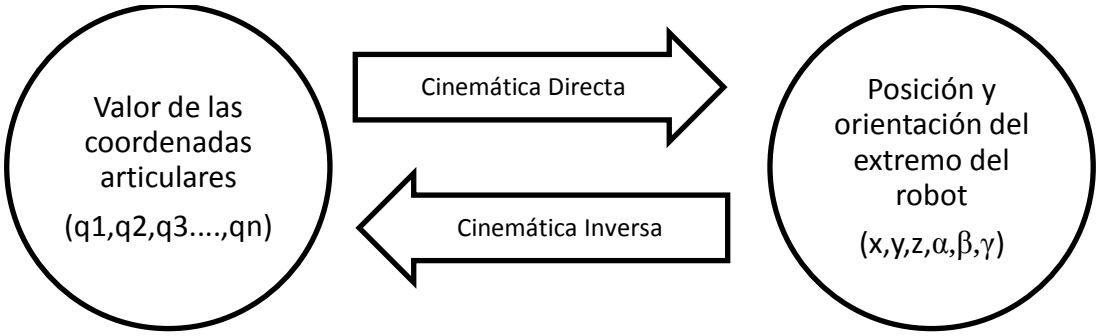

Figura 2.9. Relación entre análisis cinemático.

Fuente: (Barrientos, 2007)

2.5. MÉTODOS DE PROGRAMACIÓN DE ROBOTS

Para la programación de sistemas robóticos existen diferentes métodos los cuales son analizados a continuación:

2.5.1. PROGRAMACIÓN DE ROBOTS “ON-LINE” (EN LÍNEA)

La programación “on-line” tiene lugar en el propio lugar de producción e incluye a la célula de trabajo. El robot es programado mediante un elemento denominado “teach box”. La programación “on-line” tiene las siguientes ventajas y desventajas con respecto a la programación “off-line” (fuera de línea) (García, 2012).

Ventajas:

- Fácilmente accesible.
- El robot es programado de acuerdo a la situación real de los equipos y las piezas.

Desventajas:

- Ocupa un equipo de producción valioso y escaso.
- El lento movimiento del robot durante la programación.
- La lógica y cálculos del programa son arduos de programar.
- La parada de la producción durante la programación.
- El coste es equivalente al valor de la producción.

2.5.2. PROGRAMACIÓN DE ROBOTS “OFF-LINE” (FUERA DE LÍNEA)

La programación “off-line” tiene lugar en un ordenador y se utilizan modelos de la célula de trabajo robotizada, las piezas y los alrededores. Los programas del robot pueden ser creados, en la mayoría de los casos, mediante la utilización de datos CAD ya existentes por lo que la programación es rápida y eficaz. Los programas del robot son verificados

mediante una simulación y los errores detectados son corregidos (García, 2012).

Ventajas:

- No ocupa equipamiento de producción.
- Programación efectiva de la lógica y los cálculos por las facilidades de detección de errores existentes.
- Las localizaciones se construyen de acuerdo a modelos y esto puede significar que los programadores tendrán que ajustar bien los programas en línea o utilizar sensores.
- Programación eficaz de las localizaciones.
- Verificación del programa a través de simulación y visualización.
- Bien documentado a través del modelo de simulación con programas adecuados.
- Utilización de los datos existentes de CAD.
- El coste es independiente de la producción. La producción puede continuar durante la programación.
- Herramientas de apoyo a proceso, por ejemplo, la selección de parámetros de soldadura.

Desventajas:

- Exige invertir en un sistema de programación “off-line”.
- Exige tener personal experto en programación.

2.5.3. PROGRAMACIÓN ONLINE VS OFFLINE

Para que los robots puedan moverse hay dos técnicas de programación o entrenamiento ligeramente diferentes, los métodos de programación on-line y off-line.

La primera de ellas también llamado "edición en línea" requiere que el robot esté en funcionamiento para poder enseñarle algo. El método alternativo

consiste en escribir código en un PC con el fin de subirlo al robot. Ambos métodos tienen ventajas y desventajas

2.5.4. PROGRAMACIÓN DE ROBOTS HÍBRIDA (MIXTA)

Mediante la utilización de las ventajas de la programación “on-line” y “off-line” la técnica puede ser optimizada. A esta modalidad se le denomina generalmente como programación híbrida. Un programa de robot consiste principalmente en dos partes: localizaciones (posición y alineamiento) y lógica de programa (estructura de control, comunicación, cálculos). La lógica del programa y la mayor parte de los comandos (órdenes) de movimiento pueden ser desarrolladas de manera efectiva “off-line” con la utilización de datos CAD y la interacción del programador. Los comandos de movimiento para ubicar el emplazamiento de la pieza en la célula de trabajo del robot puede ser realizados “on-line” si fuera necesario. De esta manera pueden ser utilizadas las ventajas de ambos métodos (García, 2012).

2.5.5. PROGRAMACIÓN DE ROBOTS POR APRENDIZAJE INMEDIATO

En el modo “TEACH” (aprender), el actuador final (cabeza tecnológica) es guiado por el programador a lo largo del trayecto deseado, siendo este grabado en la memoria del control del sistema. Después de la activación del programa grabado, el robot repite la actividad aprendida en el modo “REPEAT” (repetir) una y otra vez. Este tipo de robot se utiliza principalmente en la soldadura continua a lo largo de un trayecto dado, o para la aplicación de pintura o recubrimiento de protección (García, 2012).

2.5.6. PROGRAMACIÓN DE ROBOTS PUNTO A PUNTO

El programador utiliza el panel de control para guiar el actuador final del robot al punto deseado, que es guardado en la memoria del control del sistema. El robot a partir de entonces realiza las operaciones de acuerdo a la actividad preestablecida entre los puntos individuales o en estos puntos.

Este tipo de robot es muy práctico para la soldadura por puntos de los cuerpos de los coches en las fábricas (García, 2012).

2.6. ROBOT MANIPULADOR

El diseño del brazo robótico está inspirado en la morfología de un brazo humano, aunque por cuestiones mecánicas el brazo robótico no puedo emular todos los movimientos de un brazo humano (González, 2007).

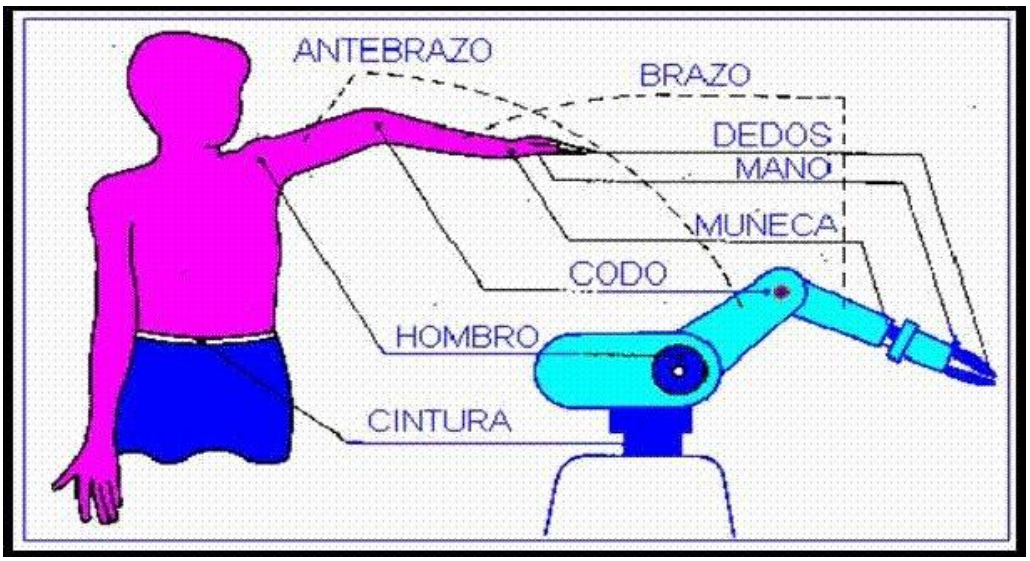

Figura 2.10. Morfología del brazo robótico.

Fuente: (Nuño, 2004).

Las articulaciones de un brazo robótico se mueven mediante motores eléctricos, aunque pueden utilizarse otras tecnologías (neumática, hidráulica). En la mayoría de los robots, la pinza se mueve de una posición a otra cambiando su orientación, todos los movimientos son comandados por la unidad controladora del robot la cual calcula los ángulos necesarios de cada articulación para llevar la pinza a las coordenadas deseadas (Barrientos, 2007).

La mayoría de los brazos articulados están equipados con servo controladores, o controladores por realimentación, que reciben datos de la

unidad controladora. Cada articulación del brazo tiene un dispositivo de medición conocido como encoder el cual es capaz de saber el ángulo exacto de cada articulación del robot y este envía datos a la unidad controladora.

Si el ángulo real del brazo no es igual al ángulo calculado para la posición deseada, la unidad controladora mueve la articulación hasta que el ángulo del brazo coincida con el ángulo calculado. Este tipo de control se denomina control de lazo cerrado (Barrientos, 2007).

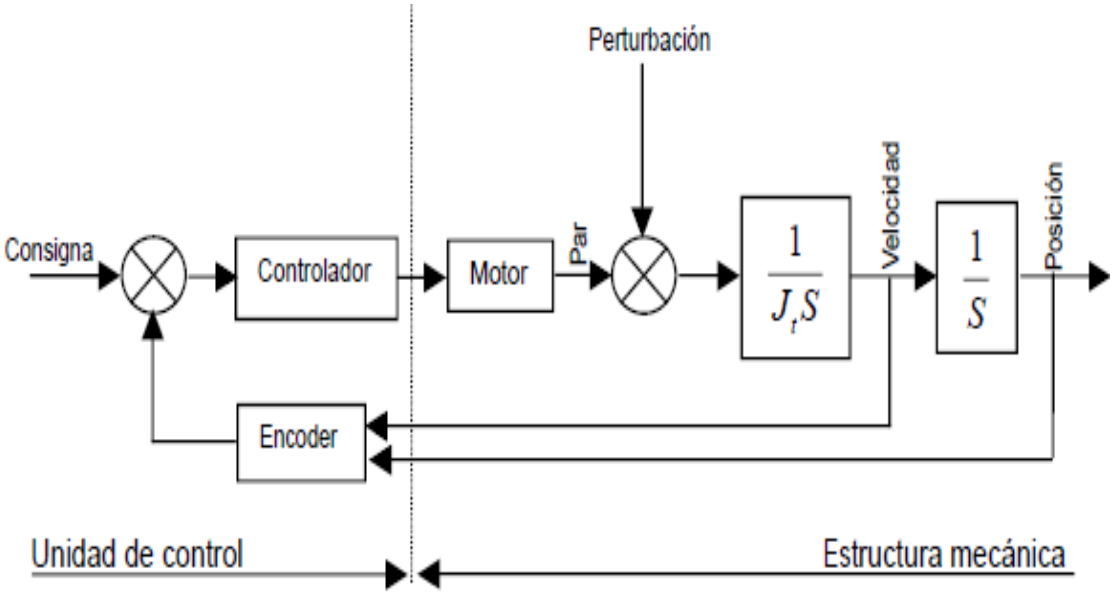

Figura 2.11. Diagrama de bloques de control de posicionamiento en lazo cerrado.

Fuente: (Barrientos, 2007).

2.6.1. CARACTERÍSTICAS DEL ROBOT ARTICULADO MITSUBISHI MELFA RV2AJ

A continuación se describen las características más significativas del robot articulado usado en este proyecto.

El robot Mitsubishi Melfa RV2AJ es un robot manipulador didáctico de 5 grados de libertad y un alcance máximo de 410 mm desde su base a su gripper, con la capacidad de levantar hasta 2 kg, a una velocidad total

máxima de 2100 mm/s y una repetitividad de posición de ±0.02 mm. (Ver Anexo 1).

El brazo cuenta con una unidad controladora CR1-571, a esta unidad se la denomina como el cerebro del robot, ya que esta es el que comanda todos los movimientos del robot.

La unidad controladora usa un CPU - RISC de 64 bits que permite la ejecución en paralelo de hasta 32 programas en modo multitarea. Un aspecto muy importante es el puerto RS-232 que se utiliza para descargar los programas hechos en la computadora y también se utiliza para comunicar el robot con dispositivos externos. Cabe destacar que por medio de un adaptador se puede comunicar el robot mediante red Ethernet (Protocolo TCP/IP), o bien con una red CC-Link de Mitsubishi que permite el intercambio rápido de datos, sobre todo entre el robot y un PLC (Robotics Equipment Corporation, 2014).

El controlador puede almacenar programas y posiciones en su memoria, 88 programas de 5000 líneas cada uno y puede guardar 2500 posiciones. (Ver Anexo 2)

2.7. INTELIGENCIA ARTIFICIAL (I.A.)

La inteligencia artificial tiene varias definiciones a continuación se citan algunas:

- Charniak y McDermott, (1985). “La inteligencia artificial es el estudio de las facultades mentales mediante el uso de modelos computacionales”.

- Winston, (1992). “La inteligencia artificial es el estudio de los cálculos que permiten percibir, razonar y actuar”.

- Luger y Sttubbledfield, (1993). “La inteligencia artificial es la rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente”.

Luego de analizar estos conceptos se puede decir que la inteligencia artificial es un proceso matemático e informático para simular el funcionamiento de la inteligencia humana.

2.8. ANTECEDENTES DE LA INTELIGENCIA ARTIFICIAL

Uno de los primeros pasos hacia la inteligencia artificial fue dado hace muchos años por Aristóteles (384-322 A.C), cuando se dispuso a codificar y tratar de explicar ciertos temas del razonamiento deductivo que el llamó silogismos. Otro intento de inteligencia fue ARSMAGNA un sistema capaz de responder cualquier pregunta, su diseño era de ruedas (Ponce, 2010).

La historia de la I.A. se remonta a los primeros años de la década de los 40, cuando los matemáticos Warren McCullock y Walter Pitts, desarrollan los algoritmos matemáticos necesarios para posibilitar el trabajo de clasificación, de una red neuronal. En 1949 Donald Hebb desarrolló un algoritmo de aprendizaje para dichas redes neuronales creando, conjuntamente con los trabajos de McCullock y Pitts, la escuela conexionista. Esta escuela se considera actualmente como el origen de lo que hoy se conoce como Inteligencia Artificial (Covarruias, 2008).

En la historia se reconoció a tres investigadores Herbert Simon, Allen Newell, J.C. Shaw, los cuales desarrollaron el primer software orientado a la resolución de problemas de la I.A. llamado IPL 11, un año más tarde estos tres investigadores desarrollan el primer programa de IA llamado "Logic Theorist", el cual era capaz de demostrar teoremas matemáticos. Este programa demostró 38 de los 52 teoremas, del segundo capítulo de "Principia Mathematica" de Russel y Whitehead.

En el mismo año surge la IA como disciplina de la Informática en la Conferencia de Computación de Dartmouth. En dicha conferencia se estableció como conclusión fundamental la posibilidad de simular inteligencia humana en una máquina.

En los primeros años de la década del 60 Frank Rosemblatt desarrolla, en la Universidad de Cornell, un modelo de la mente humana a través de una red neuronal y produce un primer resultado al cual llama Perceptron (Ponce, 2010).

Este sistema era una extensión del modelo matemático concebido por McCullock y Pitts para las neuronas, y funcionaba basándose en el principio de "disparar" o activar neuronas a partir de un valor de entrada el cual modifica un peso asociado a la neurona, si el peso resultante sobrepasa un cierto umbral la neurona se dispara y pasa la señal a aquellas con las que está conectada. Al final, en la última capa de neuronas, aquellas que se activen definirán un patrón el cual sirve para clasificar la entrada inicial. Este trabajo constituye la base de las redes neuronales de hoy en día.

2.9. RAMAS DE LA INTELIGENCIA ARTIFICIAL

Algunas de las ramas de la inteligencia artificial son:

- Lógica difusa
- Redes neurales artificiales
- Visión Artificial
- Algoritmos genéticos

2.10. VISIÓN ARTIFICIAL

Uno de los sentidos más importantes de los seres humanos es la visión. Ésta es empleada para obtener la información visual del entorno físico.

Un sistema de visión artificial trata de emular la función del sentido de la visión en un ser humano; y para ello utiliza variados algoritmos y métodos que permiten procesar la imagen adquirida.

En un sistema de visión artificial, un factor muy importante a considerar es la naturaleza de la luz, ya que los foto-receptores dentro de las cámaras captan la información de la imagen en función de la incidencia de luz sobre sí mismos (Platero, 2010).

2.10.1. ETAPAS DEL SISTEMA DE VISIÓN ARTIFICIAL

Un sistema de Visión Artificial actúa sobre una representación de una realidad, que le proporciona información sobre brillo, colores, formas, etc. Estas representaciones suelen estar en forma de imágenes estáticas, escenas tridimensionales o imágenes en movimiento (Covarruias, 2008).

Como todo sistema la visión artificial cuenta con varias etapas las cuales se describen a continuación:

Captura: Es una etapa puramente sensorial, la cual consiste en la captura o adquisición de las imágenes digitales mediante algún tipo de sensor (activo o pasivo).

Preproceso: La etapa consiste en el tratamiento digital de las imágenes, con el fin de facilitar las etapas posteriores. Es decir se aplican filtros y transformaciones geométricas, se eliminan partes indeseables de la imagen o se realzan partes interesantes de la misma.

Segmentación: Consiste en aislar los elementos que interesan de una escena para comprenderla.

Reconocimiento: Aquí se pretende distinguir los objetos, gracias al análisis de ciertas características que se establecen previamente para diferenciarlos.

Cabe mencionar que estas cuatro fases no se siguen siempre de manera secuencial, sino que en ocasiones deben realimentarse hacia atrás. Así, es normal volver a la etapa de segmentación si falla la etapa de reconocimiento, o a la de pre proceso, o incluso a la de captura, cuando falla alguna de las siguientes.

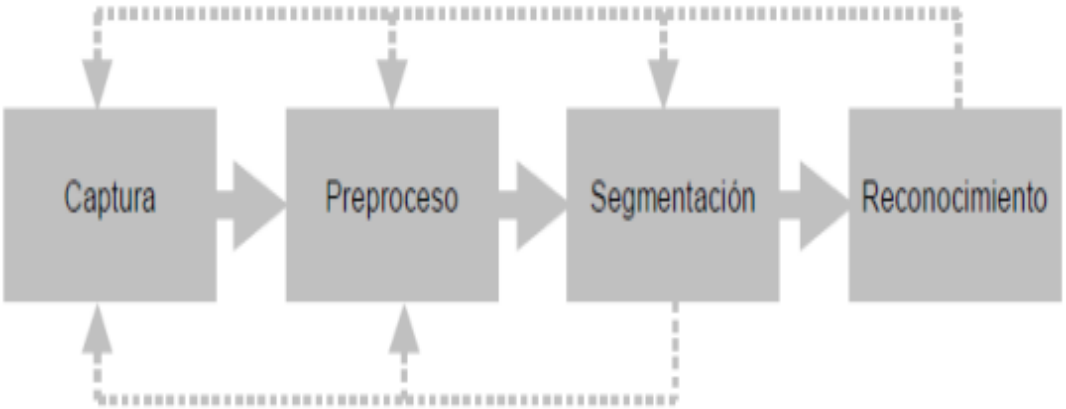

Figura 2.12. Etapas de visión artificial.

Fuente: (Alvarado, 2010)

2.10.2. ELEMENTOS DEL SISTEMA DE VISIÓN ARTIFICIAL.

Los elementos imprescindibles para los sistemas de visión artificial son:

- Un sensor óptico para captar la imagen: Una cámara de vídeo, una cámara fotográfica, una cámara digital, un escáner 3D.

- Un computador que almacene las imágenes y que ejecute los algoritmos de pre-procesado, segmentación y reconocimiento de la misma.

Existen diferentes maneras de obtener una imagen del entorno, gracias al desarrollo de la tecnología se han creado dispositivos especializados en la captura y tratamiento digital de las imágenes como son:

- Cámaras de alta velocidad
- Scanner de escenas 3D
- Cámaras de alta definición
- Cámaras lectoras de caracteres, etc.

Gracias a esto la etapa de adquisición de imágenes se ha vuelto más rápida y confiable ya que estos dispositivos son capaces de enfocarse en el objetivo deseado sin dejarse afectar por el ambiente que las rodea.

Entre los más nuevos dispositivos para la adquisición de imágenes se encuentran los que contienen múltiples sensores para analizar el entorno, este sería el caso del dispositivo “Kinect” desarrollado por Microsoft el cual cuenta con la capacidad de capturar figuras en 3D, utilizar visión estereoscópica para la ubicación espacial del usuario y además usar visión infrarroja para situaciones de poca luminosidad

Basándose en esto se han desarrollado otros dispositivos los cuales únicamente realizan una captura de imágenes con visión infrarroja este es el caso de Leap Motion un dispositivo capaz de reconocer las articulaciones de las manos del usuario con una precisión de 0.1 mm únicamente situándolas sobre el dispositivo.

Se han realizado varios proyectos utilizando ambos dispositivos para la tele operación de robots manipuladores o antropomórficos dando resultados satisfactorios ya que estos dispositivos además de ser baratos, pequeños y con una capacidad de respuesta muy rápida cada uno cuenta con sus respectivos kit de desarrollo de software o SDK (siglas en inglés de software development kit), los cuales cuentan con diferentes librerías para su programación, con esto se logra tener un mayor control sobre el dispositivo; todo esto permite concluir que estos dispositivos son interfaces mucho más sencillas de utilizar.

2.11. COMUNICACIÓN SERIAL, RS-232

La comunicación serial es un protocolo muy común para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. RS-232 (Recommended Standard 232) es una interfaz que designa una norma para el intercambio de una serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Equipo de Comunicación de datos).

Cuando se transmite información a través de una línea serie es necesario utilizar un sistema de codificación que permita resolver los siguientes problemas. (Freire, 2012)

- ***Sincronización de bits:*** El receptor necesita saber dónde comienza y donde termina cada bit de la señal recibida para efectuar el muestreo de la misma en el centro del intervalo de cada símbolo.

- ***Sincronización del caracter:*** La información serie se transmite por definición bit a bit, pero la misma tiene sentido en bytes.

- ***Sincronización del mensaje:*** Es necesario conocer el inicio y fin de una cadena de caracteres por parte del receptor para, para poder detectar algún error en la comunicación de un mensaje.

En una comunicación en general, se pueden establecer canales para la comunicación de acuerdo a tres técnicas.

- ***Simplex:*** es aquella que ocurre en una dirección solamente, deshabilitando al receptor de responder al transmisor. Normalmente la transmisión simplex no se utiliza donde se requiere interacción humano-máquina.

- ***Half duplex:*** La comunicación serie se establece a través de una sola línea, pero en ambos sentidos. En un momento el transmisor enviará información y en otro recibirá, por lo que no se puede transferir información en ambos sentidos de forma simultánea.

- ***Full duplex:*** Se utilizan dos líneas (una transmisora y otra receptora) y se transfiere información en ambos sentidos. La ventaja de este método es que se puede transmitir y recibir información de manera simultánea.

En cuanto a modos de trasmisión existen dos modos básicos para realizar la transmisión de datos y son:

- Transmisión asíncrona.
- Transmisión síncrona.

2.11.1. COMUNICACIÓN ASÍNCRONA

Cuando se opera en modo asíncrono no existe una línea de reloj común que establezca la duración de un bit y el caracter puede ser enviado en cualquier momento. Esto implica que cada dispositivo tenga su propio reloj y que previamente se haya acordado que ambos dispositivos transmitirán datos a la misma velocidad. En la transmisión asíncrona un caracter a transmitir es incluido con un indicador de inicio y fin de carácter, lo que se conoce como bit de inicio y un bit de parada. Durante el lapso en que no se transmite ningún byte, el canal debe estar en alto (1 lógico), de la misma forma al bit de parada se le asigna también un "1". Por otro lado, al bit de inicio del caracter a transmitir se le asigna un "0" lógico, por lo que un cambio de nivel de "1” lógico a "0" lógico indicará al receptor que un nuevo caracter será transmitido (Gentilini, 2011).

La transmisión asíncrona definida por la norma RS-232, se basa en las siguientes reglas (Gentilini, 2011).

- Cuando no se envían datos por la línea, ésta se mantiene en estado alto.
- Cuando se desea transmitir un caracter, se envía primero un bit de inicio que pone la línea en bajo durante el tiempo de un bit.
- Luego se envían todos los bits del mensaje a transmitir con los intervalos que marca el reloj de transmisión. Por convenio se transmiten entre cinco y ocho bits.
- Se envía primero el bit menos significativo y de último el bit más significativo.
- Con el último bit del mensaje se envía el bit (o los bits) de final, configurando la línea de comunicación a “1” lógico por lo menos durante el tiempo mínimo de un bit. Estos bits pueden ser un bit de paridad para detectar errores y el bit o bits de stop, que indican el fin de la transmisión de un caracter.

Los datos codificados por esta regla, pueden ser recibidos de acuerdo a los siguientes pasos:

- Esperar la transición 1 a 0 en la señal recibida.
- Activar el reloj con una frecuencia igual a la del transmisor.
- Muestrear la señal recibida al ritmo de ese reloj para formar el mensaje.
- Leer un bit más de la línea y comprobar si es 1 para confirmar que no ha habido error en la sincronización.

2.11.2. COMUNICACIÓN SÍNCRONA

Es un método más eficiente de comunicación en cuanto a velocidad de transmisión, esto porque no existe ningún tipo de información adicional entre los caracteres a ser transmitidos. Cuando se transmite de manera síncrona lo primero que se envía es un octeto de sincronismo, el cual realiza la misma función que el bit de inicio en la transmisión asíncrona, indicar al receptor

que se va enviar un mensaje. Este caracter utiliza la señal local de reloj para determinar cuándo y con qué frecuencia será muestreada la señal, es decir, permite sincronizar los relojes de los dispositivos transmisor y receptor (Gentilini, 2011).

2.12. KINECT

Kinect es un dispositivo desarrollado por Microsoft lanzado al mercado el 4 de noviembre del 2010, el cual originalmente fue conocido como “Project Natal”, este dispositivo permite al usuario interactuar y controlar videojuegos sin necesidad de tener contacto físico con un mando de videojuego tradicional, ya que este dispositivo es capaz de reconocer los movimientos del cuerpo humano de tal manera que logra interactuar directamente con la consola de juego, a todo esto se lo conoce como “NUI” o con su traducción al español “Interfaz natural de usuario”, en principio el dispositivo fue desarrollado únicamente para ser usado con la consola de juegos XBOX 360, pero debido a sus características y gran funcionalidad diferentes grupos de programadores encontraron la forma de utilizar este dispositivo en una PC, sin el consentimiento de Windows, esto lo consiguieron mediante la creación de un controlador el cual funcionaba bajo el sistema operativo Linux el cual es de código abierto, de esta manera los programadores eran capaces de captar video, y utilizar las diferentes ventajas de este dispositivo, esta es una de las razones por la cual Microsoft Research en Febrero del 2012 lanza “Kinect para Windows” y con este el primer SDK gratuito para programadores, el SDK es compatible tanto con el dispositivo de Kinect para XBOX 360 y Kinect para Windows, este SDK cuenta con todas las librerías necesarias para activar todas las características del sensor, con este SDK es posible realizar aplicaciones orientadas a pc las cuales pueden estar escritas en diferentes lenguajes como: C#, C, JAVA.

Actualmente el SDK se encuentra en la versión 1.8 la cual cuenta con un motor de reconocimiento de gestos, acciones, rostro humano, y voz.

2.13. COMPONENTES DEL SENSOR KINECT.

El dispositivo cuenta con una cámara RGB, un sensor de profundidad, un arreglo de micrófonos, un servo motor, un emisor de luz infrarroja y un LED (Light emitting diode) indicador de encendido como se puede apreciar en la Figura 2.13., en la Figura 2.14., se aprecia la placa base del sensor.

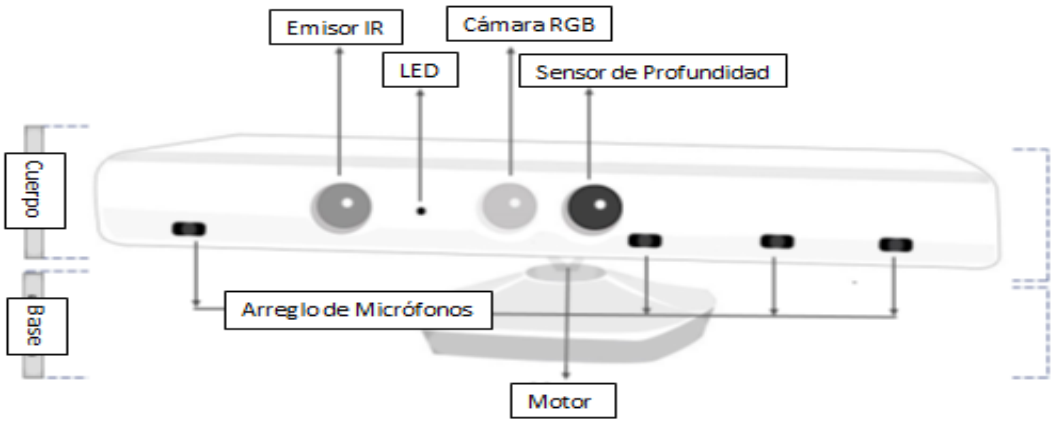

Figura 2.13. Componentes del sensor Kinect.

Fuente: (Jana, 2012)

Figura 2.14. Placa base sensor Kinect.

Fuente: (Jana, 2012).

2.13.1. CÁMARA RGB

Es la responsable de la capturar y transmitir los datos de video desde el entorno al computador, su principal función es detectar los tres colores

principales (Rojo, Verde, Azul), Los datos capturados por la cámara son devueltos como una sucesión de cuadros (Imágenes Fijas), las cuales luego de recopilarlas en el computador se convierten en un video continuo.

El sensor puede capturar datos de video en diferentes velocidades y resoluciones

- 640*480 pixeles - 30 cuadros por segundo (FPS)
- 1280*960 pixeles - 12 cuadros por segundo (FPS)

El ángulo de visibilidad del sensor es de 43° verticales y 57° horizontales, como se aprecia en la Figura 2.15.

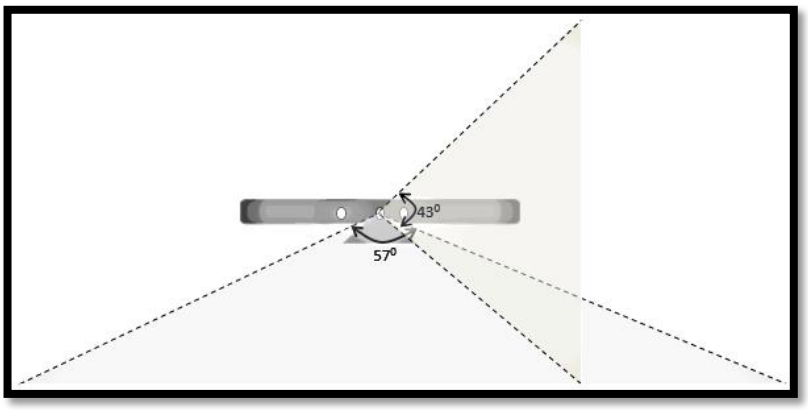

Figura 2.15. Ángulo de Visibilidad

Fuente: (Jana, 2012)

2.13.2. EMISOR DE INFRARROJO Y SENSOR DE PROFUNDIDAD

El sensor Kinect utiliza estos dos componentes para poder detectar la profundidad del entorno, el funcionamiento se basa en la cantidad de puntos infrarrojos que se encuentra en el ambiente, estos puntos son emitidos constantemente por el Emisor de Infrarrojo y a su vez estos son cuantificados por el sensor de profundidad como se muestra en la Figura 2.16.

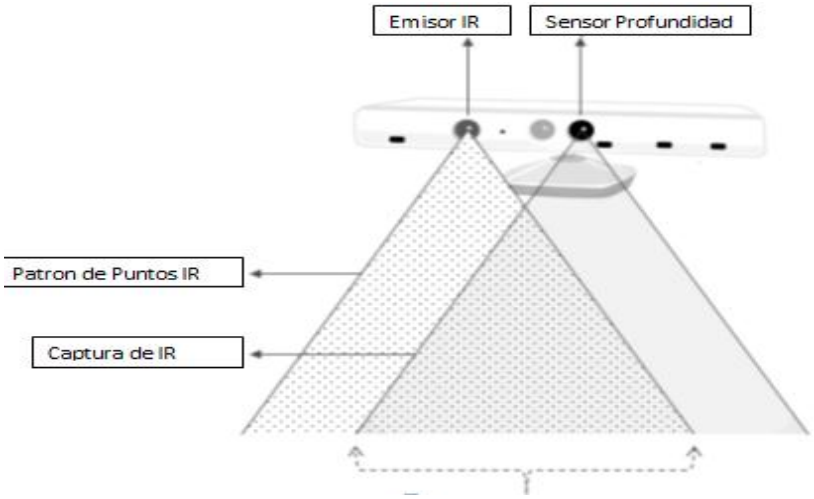

Figura 2.16. Emisor IR y Sensor Profundidad

Fuente: (Jana, 2012)

Cabe mencionar que el sensor de profundidad también transmite datos hacia el computador, pero éste lo hace en diferentes resoluciones y a una sola velocidad de captura la cual es 30 FPS

- 640 x 480 pixeles
- 320 x 240 pixeles
- 80 x 60 pixeles

Para poder realizar una correcta captura de datos de profundidad es necesario estar ubicado a una distancia mínima de un metro y máxima de tres metros, ya que en estas distancias tanto los puntos generados por el emisor infrarrojo (IR) y el ángulo de captación del sensor están en una misma región.

2.13.3. MOTOR

Es un elemento muy importante del dispositivo que permite ajustar el ángulo de inclinación del sensor para que este pueda capturar los datos de una forma correcta, los rangos de movimiento angular del Kinect se muestran en la Figura 2.17.

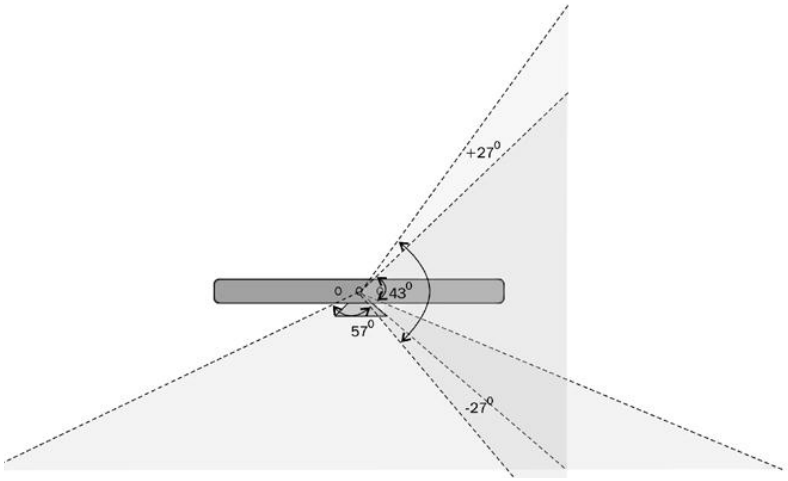

Figura 2.17. Ángulos de Inclinación

Fuente: (Jana, 2012)

2.13.4. ARREGLO DE MICRÓFONOS

Kinect cuenta con un arreglo de cuatro micrófonos ubicados a diferentes distancias, esto se debe a que el sensor es capaz de reconocer la ubicación del sonido y eliminar el ruido del exterior para poder realizar aplicaciones basadas en reconocimiento de voz, como se aprecia en la Figura 2.18.

Figura 2.18. Arreglo de Micrófonos

Fuente: (Jana, 2012)

2.13.5. LED (LIGHT EMITTING DIODE)

Es un indicador de color verde ubicado en la parte frontal del Kinect y nos permite verificar de una manera visual si el dispositivo se encentra encendido y activo, si este indicador no se enciende esto se puede deber a que el sensor no se encuentra conectado a una fuente de energía correcta.

2.14. KINECT PARA WINDOWS VS KINECT PARA XBOX 360

Realizar la comparación entre estos dos dispositivos podría resultar algo trivial ya que los dos cuentan con los mismos sistemas, aunque tienen muy pequeñas diferencias las cuales radican en la evolución de sus dispositivos internos, los cuales fueron mencionados anteriormente, la principal evolución se encuentra en su sensor de profundidad.

En la versión diseñada para Windows, el sensor de profundidad tiene un rango de captura entre 0,4 – 3 m, esto es conocido como Near Mode y en la versión de Xbox 360 el rango de captura es 0,8 – 4 m, adicional a esto no existen otras diferencias destacables entre los dispositivos.

Así que antes de realizar una selección entre estos dispositivos se debe tener en cuenta que tipo de aplicación se piensa desarrollar, la recomendación de Windows es usar el Kinect diseñado para esta plataforma ya que con este dispositivo se puede sacar el máximo provecho del Kit de Desarrollo de Software (SDK).

Cabe mencionar que existe una diferencia más entre los dispositivos la cual es su precio, el precio del Kinect para Xbox 360 aproximadamente es la mitad del costo del Kinect desarrollado para Windows.

2.15. CARACTERÍSTICAS DEL KIT DE DESARROLLO DE SOFTWARE PARA KINECT (SDK)

Luego de analizar la composición externa e interna del dispositivo, se continúa con la descripción del software de desarrollo de aplicaciones.

Para poder utilizar el SDK es necesario tener conocimiento en lenguaje C, además de esto, es necesario contar con un Entorno de Desarrollo Integrado (IDE); el IDE recomendado por Windows es Visual Studio.

Visual Studio es de gran ayuda debido a su alta flexibilidad para el desarrollo de aplicaciones con interfaz visual, las aplicaciones desarrolladas para Kinect necesariamente deben contar con una interfaz gráfica para poder visualizar los datos capturados por el dispositivo.

El SDK del Kinect provee múltiples librerías para la directa interacción entre el software y los elementos externos del sensor como son:

- Cámara RGB
- Sensor de profundidad
- Emisor infrarrojo
- Arreglo de micrófonos
- Motor de inclinación.

Las principales características del SDK se mencionan a continuación:

- Captura y procesamiento de imagen
- Procesamiento de profundidad de objetos
- Seguimiento de las articulaciones del cuerpo humano
- Reconocimiento de gestos
- Captura y procesamiento de audio

Estas características permiten desarrollar aplicaciones muy intuitivas para el usuario ya que solo debe realizar movimientos para ejecutar acciones.

3. METODOLOGÍA

3.1. METODOLOGIA MECATRÓNICA

“La mecatrónica está compuesta de “meca” de mecanismo y por “trónica” de electrónica” (Yasakawa, 2008)

“Mecatrónica se refiere al diseño integrado de los sistemas buscando un menor costo, una mayor eficiencia y mayor confiabilidad y flexibilidad desde el punto de vista mecánico, eléctrico, electrónico, de programación y de control” (Steren, 1998).

“El término mecatrónica se usa para describir la integración de sistemas de control basados en microprocesadores, sistemas eléctricos y sistemas mecánicos. Un sistema mecatrónico no es simplemente la unión de sistemas eléctricos y mecánicos, y es más que un simple sistema de control: es la integración completa de todo lo anterior”. (Boltón, 2006)

Después de analizar las definiciones citadas anteriormente se puede concluir que mecatrónica es una rama multidisciplinaria de la ingeniería, la cual integra diferentes áreas técnicas como son: Mecánica, Electrónica y Control e Informática con el objetivo de desarrollar sistemas flexibles, confiables e inteligentes para la industria.

Para aplicar esta metodología se requiere hacer un análisis integrado de todas las áreas técnicas involucradas para que el sistema pueda tener las principales características de un sistema mecatrónico los cuales son: Flexibilidad, Rentabilidad y Confiabilidad, además de esto la metodología se basa en el diseño paralelo el cual trata de integrar todas las etapas del diseño en una etapa sinérgica en la cual el diseño electrónico, mecánico, software, hardware, control no se analicen de una forma individual sino en una manera conjunta para poder optimizar el tiempo de diseño y mejorar el prototipo.

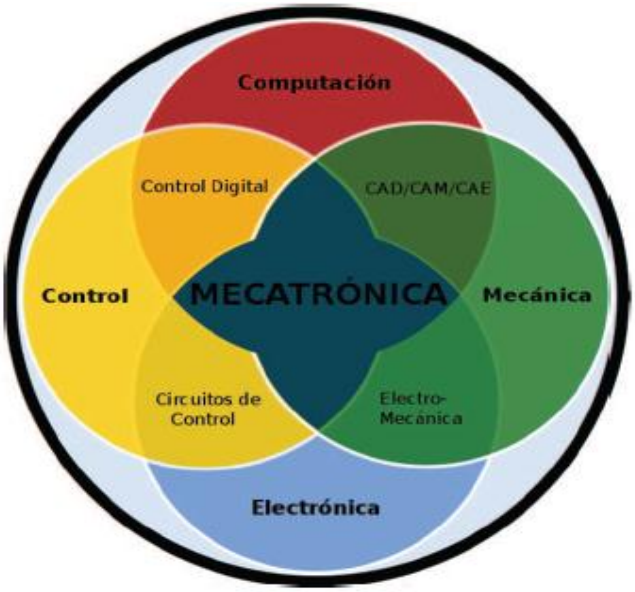

Figura 3.1. Disciplinas Involucradas en la Metodología Mecatrónica

Fuente: (Gentilini, 2011)

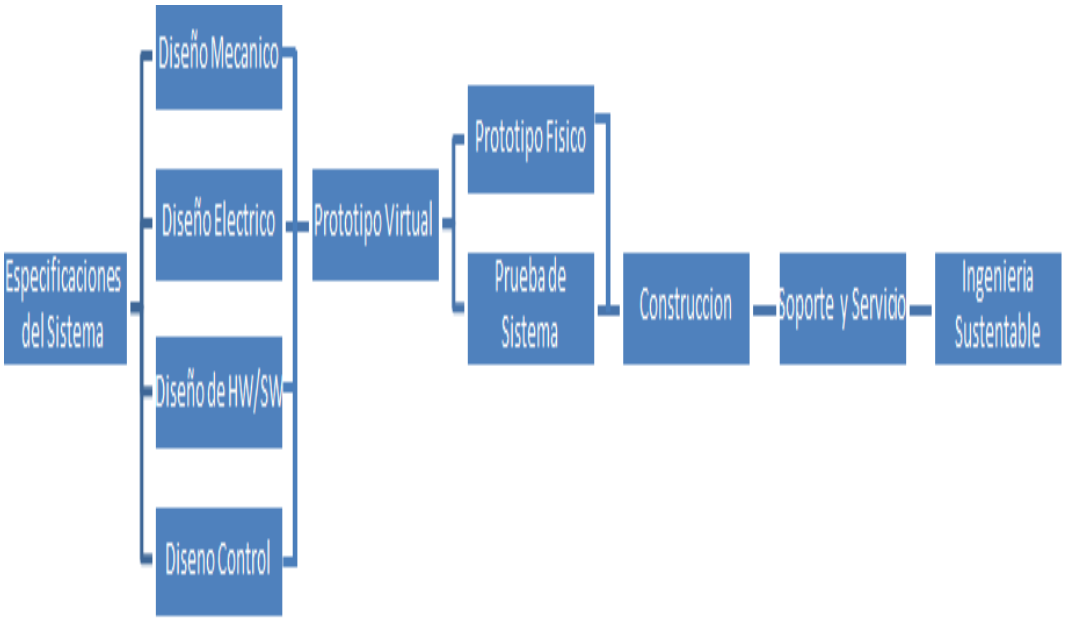

Figura 3.2. Descripción diseño paralelo

Fuente: (Gentilini, 2011)

Como se aprecia en la Figura 3.2., el diseño paralelo primero analiza todas las especificaciones necesarias para definir el sistema, luego de haber comprendido todas estas especificaciones se pasa al bloque de diseño en donde de forma sinérgica los diferentes diseños se acoplan correctamente, para después pasar hacia la creación de un prototipo virtual en el cual se analiza su funcionamiento únicamente de manera virtual, si en este punto todo está correcto se pasa al diseño de un prototipo físico el cual es testeado de diferentes maneras.

3.2. INVESTIGACIÓN DEL PROBLEMA

Para los robots no existe un lenguaje de programación generalizado. De hecho, en la actualidad existen multitud de lenguajes destinados a la programación de robots, puesto que en la mayoría de los casos los propios fabricantes desarrollan el lenguaje destinado a su robot concreto.

El proceso de programación de un robot consiste en introducir en su unidad de control las instrucciones necesarias para desempeñar tareas que le han sido asignadas, esto requiere que el operario tenga un alto conocimiento en lenguajes de programación, cinemática de robots y el proceso que debe realizar el autómata.

Partiendo de lo anterior se ha visto la necesidad de diseñar e implementar un sistema de seguimiento inteligente para un robot manipulador Mitsubishi Melfa RV2AJ, el cual pueda ser de fácil aprendizaje para el operario, y no requiera altos conocimientos en lenguajes de programación.

De esta manera lograr que el operario pueda controlar el robot Mitsubishi Melfa RV2AJ sin la necesidad de precargar ninguna instrucción en la unidad controladora del robot y así maximizar la flexibilidad del robot manipulador, sin que esto represente algún peligro para el operario o la integridad del brazo manipulador Mitsubishi Melfa RV2AJ.

3.3. REQUERIMIENTOS DEL SISTEMA

Para cumplir con lo planteado en el proyecto se requiere lo siguiente:

- Implementar un sistema de Teleoperación basado en visión artificial

- Implementar un algoritmo de reconocimiento de movimientos y posiciones del operario.

- Diseñar un software de interfaz entre el sistema de visión artificial y la unidad controladora del robot.

- Implementar el sistema mecatrónico que responda aproximadamente en tiempo real sobre un robot articulado Mitsubishi Melfa RV2AJ

En este sistema de teleoperación robótico el operador toma el control del robot remotamente respetando todas las restricciones mecánicas del robot (Ver Anexo 1) y con esto el operario se encuentra seguro y aislado del proceso de producción, para este objetivo se pueden utilizar diferentes tecnologías como pueden ser:

- Joysticks
- Mouse
- Servidores remotos
- Visión artificial, etc.

Para poder cumplir con los requerimientos del proyecto, el sistema mecatrónico utilizará un sistema de visión artificial para reconocer el movimiento y posición de las articulaciones de las extremidades superiores del operario y transmitirlas al robot intentando lograr una respuesta en tiempo real.

Continuando con la metodología Mecatrónica se debe analizar el bloque de diseño en el cual está involucrado el diseño mecánico, eléctrico/electrónico, software y hardware, control.

3.4. DISEÑO MECÁNICO

Para lograr que el diseño mecánico cumpla con las necesidades del sistema se procederá de la siguiente manera:

1. Analizar los requerimientos del sistema, en pequeños subsistemas, lo que facilita el análisis y permita mejor manejo de la información.
2. Definir los elementos que intervienen en el sistema mecánico.
3. Simular el sistema mecánico en un software.
4. Implementar para realizar pruebas de funcionamiento.

3.5. DISEÑO ELECTRÓNICO

Para lograr un diseño electrónico que cumpla con las necesidades del sistema se procederá de la siguiente manera:

1. Analizar los requerimientos del sistema, en pequeños subsistemas, lo que facilita el análisis y permite mejor manejo de la información.

2. El siguiente paso es establecer un diagrama de flujo del proceso para conocer exactamente cada una de los procesos que se realizarán, para de esta manera diseñar los subsistemas electrónicos y lograr una sincronización de los mismos.

3. Analizar el sistema de comunicación que se utilizará así como sus velocidades.

4. Realizar pruebas, corregir errores e implementar.

3.6. DISEÑO DEL SISTEMA DE CONTROL

1. Analizar los requerimientos del sistema, para luego diseñar el control de cada subsistema.

2. Elegir un controlador principal que definirá los movimientos y el funcionamiento de los demás subsistemas.

3. Programar al controlador principal para que realice las operaciones necesarias y se comunique con los subsistemas.

3.7. SIMULACIÓN

Para analizar y estudiar el comportamiento el modelo de solución propuesto se realizará una simulación digital a través de un IDE (Entorno de Desarrollo Integrado) de programación en donde se verificará si la interfaz es capaz de reconocer y enviar las posiciones del usuario hacia el sistema controlador del robot.

3.8. CONSTRUCCIÓN DEL PROTOTIPO

Una vez desarrollado el software de reconocimiento de posiciones basado en visión artificial y comprobado que este no tenga errores ni problemas para reconocer las posiciones angulares de las extremidades superiores del usuario, se procederá a implementar la comunicación entre el software de reconocimiento y el sistema controlador del robot manipulador Mitsubishi Melfa RV2AJ y de esta manera continuar con el siguiente punto de la metodología.

3.9. PRUEBAS

Luego de haber implementado todo el prototipo se procederá a realizar las siguientes pruebas:

1. Capacidad de imitación de movimientos del robot con respecto al brazo humano
2. Apertura y cierre de pinza mediante gestos
3. Advertencia de posiciones no seguras para el robot.
4. Tiempo de respuesta entre el cambio de movimientos
5. Ubicación del robot en una posición de seguridad cuando no se encuentre un usuario frente al sistema de visión artificial

4. DISEÑO DEL SISTEMA

4.1. DISEÑO DEL SISTEMA

El sistema mecatrónico estará basado en visión artificial y será implementado en un brazo articulado Mitsubishi Melfa RV2AJ el cual se comunicará por puerto serial con la aplicación programada en Visual Studio utilizando el SDK de Kinect.

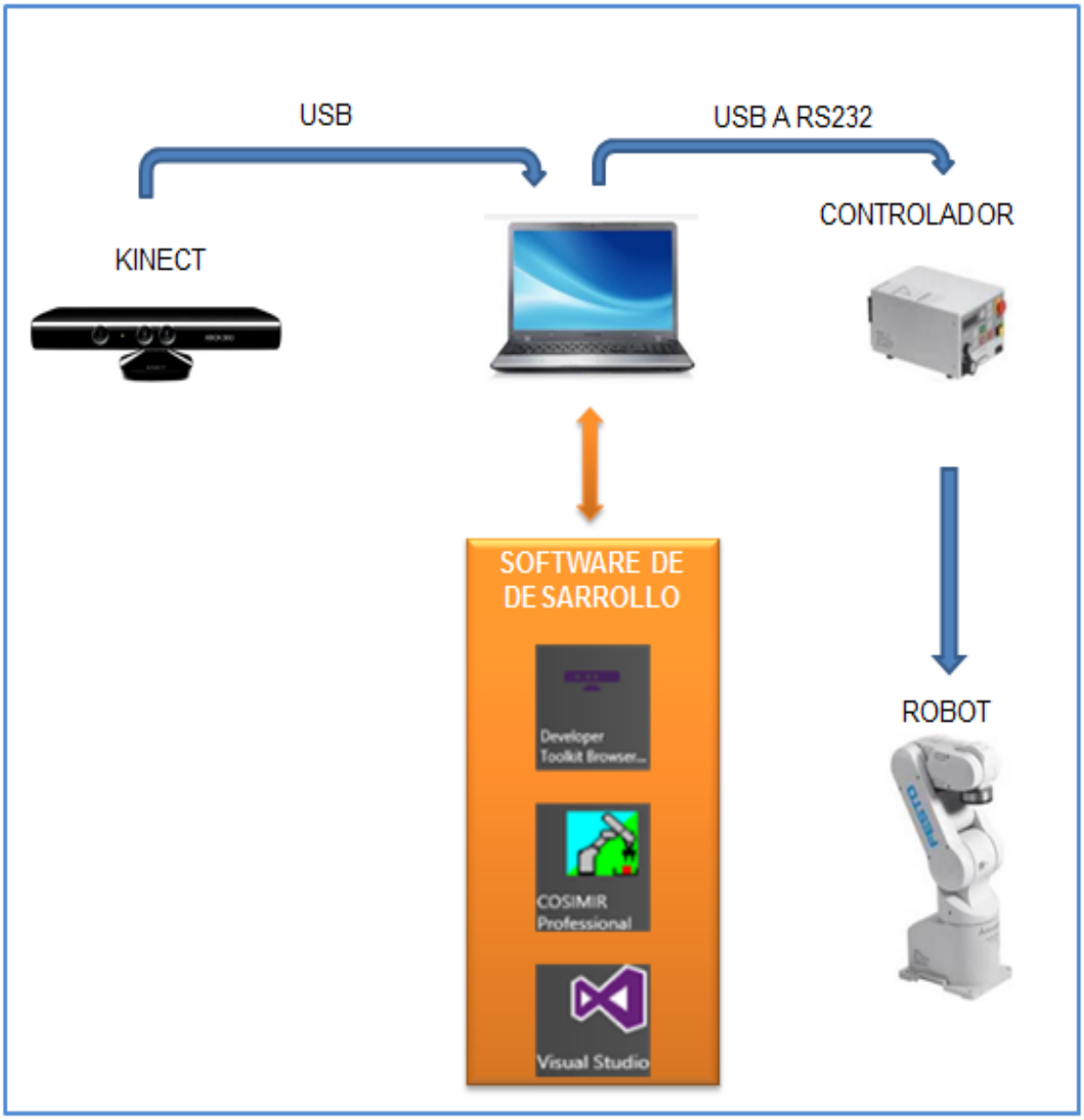

Figura 4.1. Arquitectura del Sistema Mecatrónico.

Fuente: El autor.

Luego de apreciar la arquitectura del sistema mecatrónico, se procederá con la segmentación de los subsistemas:

- Subsistema de Visión Artificial
    - Captura de imágenes
    - Reconocimiento de articulaciones de las extremidades superiores
    - Análisis de posiciones
- Subsistema de Control
    - Descripción de la unidad controladora CR1-571
    - Comunicación entre PC y controlador CR1-571
- Subsistema Mecánico
    - Descripción del robot manipulador Mitsubishi Melfa RV2AJ

Luego de haber analizado y descrito todos los subsistemas se procede a la consolidación de todos éstos y se lo denominará como el Sistema de Seguimiento Inteligente en donde se podrá verificar el funcionamiento del sistema de posicionamiento del robot articulado, utilizando un software diferente al establecido para este tipo de robot.

- Sistema de Seguimiento Inteligente
    - Consolidación de los subsistemas y verificación de funcionamiento

4.2. SUBSISTEMA DE VISIÓN ARTIFICIAL

El subsistema de visión artificial es una parte vital del proyecto ya que aquí es donde se va analizar las posiciones de las extremidades superiores del operario, todo esto se va a lograr utilizando el IDE Visual Studio y el SDK de Kinect, así que a continuación se procede a describir todos los subsistemas necesarios para el diseño del sistema de visión artificial

4.2.1. CAPTURA DE IMÁGENES

Este punto describirá el uso del Kinect como un dispositivo para la captura y reconocimiento de imágenes, ya que como se menciona en el Capítulo 2 este dispositivo cuenta con una cámara RGB la cual es capaz de transmitir imágenes del entorno hacia el pc.

Antes de continuar con el desarrollo del software para capturar imágenes se procede a describir los requisitos para utilizar Kinect en una PC.

- Sistema operativo Windows 7 , Windows 8
- Kinect para Xbox 360
- Kinect SDK para Windows
- Kinect herramientas de desarrollo
- IDE Visual Studio
- Fuente externa de poder

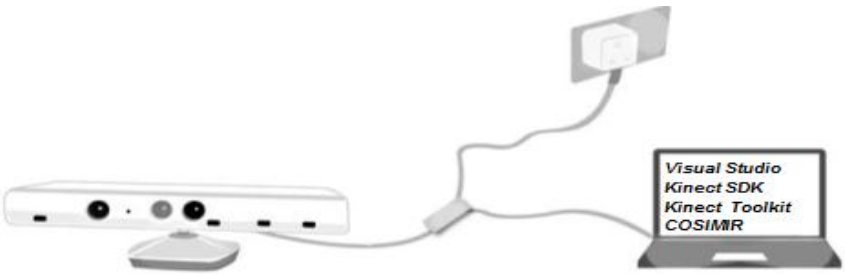

Figura 4.2. Requisitos del Sistema.

Fuente: El autor.

La aplicación de captura de imágenes debe ser capaz de capturar las imágenes del entorno y procesarlas, todo esto se desarrolla utilizando las librerías del SDK de Kinect.

La captura de imágenes en el dispositivo Kinect comprende los siguientes puntos:

1. Activar y parametrizar la cámara RGB del Kinect.
2. Capturar el flujo de imágenes RGB cuadro por cuadro en Visual Studio.
3. Procesar las imágenes.

Este es un proceso general, el cual debe ser ejecutado de esta forma ya que si al momento de programar el Kinect se omite uno de estos puntos, el sensor no podrá capturar las imágenes y mucho menos procesarlas.

Antes de continuar con el desarrollo de los puntos para la captura de imágenes se explicará algunos conceptos necesarios para entender el proceso de captura.

4.2.2. FLUJO DE IMÁGENES

El flujo de imágenes no es nada más que una sucesión de imágenes estáticas. Kinect puede capturar y enviar imágenes o cuadros estáticos en dos velocidades: 12 cuadros por segundo o a 30 cuadros por segundo, estas velocidades dependen de la resolución a la cual se parametrice el Kinect.

4.2.3. TIPOS DE FORMATOS EN FLUJO DE IMAGEN

EL dispositivo Kinect soporta los siguientes tipos de color en el flujo de imagen.

- RGB
- YUV
- BAYER

RGB, este nombre es una referencia a las iniciales Red, Green, Blue. Cada pixel RGB de una imagen capturada por el dispositivo Kinect es un arreglo de 4x4 ordenado de la siguiente manera: Los tres primeros valores son rojo, verde y azul, El color Alpha es el cuarto valor en el arreglo, este define el nivel de transparencia de la imagen.

Azul	**Verde**	**Rojo**	Alpha
Azul	**Verde**	**Rojo**	Alpha
Azul	**Verde**	**Rojo**	Alpha
Azul	**Verde**	**Rojo**	Alpha

Figura 4.3. Arreglo RGB.

Fuente: El autor.

YUV, el parámetro Y representa la luminancia (información en blanco y negro), mientras que U y V representan la crominancia (información con respecto al color).

A continuación se muestran las relaciones entre YUB y RGB:

$$Y = 0.299R + 0.587\ G + 0.114\ B$$

$$U = -0.147R - 0.289\ G + 0.436B = 0.492(B - Y)$$

$$V = 0.615R - 0.515G - 0.100B = 0.877(R-Y)$$

Ambos datos YUV y RGB representan la misma captura de imagen ya que usan la misma cámara, la única diferencia es la calidad de los colores.

Como se mencionó anteriormente el dispositivo Kinect puede capturar imágenes a distintas velocidades y todo esto depende de la resolución a la que se preestablezca el sensor.

- 640 x 480 a 30 cuadros por segundo (RGB- 32 bits)
- 1280 x 960 a 30 cuadros por segundo (RGB- 32 bits)
- 640 x 480 a 15 cuadros por segundo (YUV – 16 bits)

Analizando los datos anteriores se puede afirmar que al usar el formato YUV la imagen ocupa menos memoria en comparación a una imagen en RGB, debido a que una imagen RGB necesita 32 bits por pixel y una imagen YUV únicamente necesita 16 bits por pixel, estos bits se almacenan en un buffer que afecta directamente a la velocidad de procesamiento de la imagen, pero antes de tomar una decisión en la selección de un tipo de imagen también se debe considerar que la calidad de la imagen RGB es mayor que una imagen YUV.

BAYER, este tipo de formato de imagen tiene una combinación de colores: rojo, verde, azul, pero este formato define un patrón que usa 50% de verde, 25% de rojo y 25% de azul. Este patrón es llamado Filtro Bayer.

Este filtro se diseñó debido a la sensibilidad del ojo humano hacia el color verde y de esta manera puede asimilar más cambios en las imágenes saturadas por el color verde.

Verde	Rojo	Verde	Rojo
Azul	Verde	Azul	Verde
Verde	Rojo	Verde	Rojo
Azul	Verde	Azul	Verde

Figura 4.4. Filtro Bayer.

Fuente: El autor.

Kinect soporta las siguientes resoluciones en formato Bayer

- 640 x 480 a 30 cuadros por segundo
- 1280 x 960 a 12 cuadros por segundo

Además de capturar imágenes en color, Kinect es capaz de capturar imágenes infrarrojas a 640 x 480 a 30 cuadros por segundo, eso lo logra ya que el emisor infrarrojo dispara un patrón hacia el entorno y este rebota hacia la cámara del dispositivo Kinect y lo captura.

4.2.4. TRANSMISIÓN DE IMÁGENES DESDE KINECT HACIA WINDOWS

Existen dos maneras de transmitir las imágenes desde el sensor hacia la aplicación estos métodos son los siguientes:

- Método por Evento
- Método por Demanda

4.2.4.1. Método por Evento

Usando este método, el dispositivo Kinect envía las imágenes capturadas a la aplicación siempre y cuando una nueva imagen sea capturada por el Kinect. Para lograr esto es necesario suscribirse a un Evento usando código de programación, este código va a procesar las imágenes entrantes a la aplicación. Antes de suscribirse al evento, es necesario inicializar el SDK con los valores de resolución y color de captura de la imagen, una vez realizado esto, Kinect es capaz de transmitir imágenes hacia la aplicación continuamente hasta que se le ordene que deje de transmitir o se detenga la aplicación o el Kinect

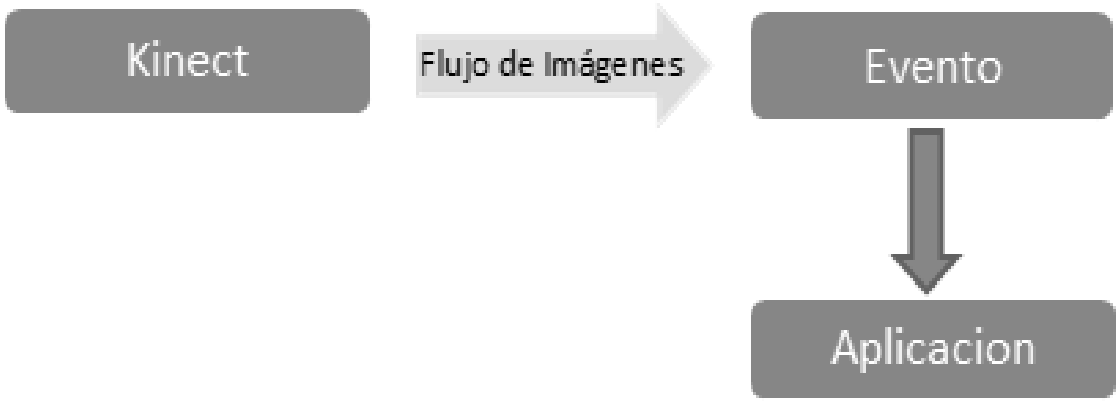

Figura 4.5. Método por Evento

Fuente: El autor.

4.2.4.2. Método por Demanda

En este método, la aplicación debe pedir o requerir que Kinect capture y envié una imagen hacia la aplicación caso contrario el Kinect no realizará ninguna acción, la captura de las imágenes va a depender totalmente de la aplicación más que de la inicialización del SDK.

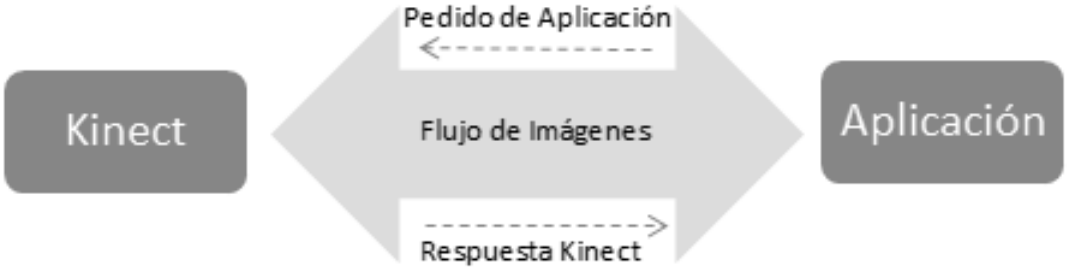

Figura 4.6. Método por Demanda.

Fuente: El autor.

Finalmente luego de haber expuesto los conceptos básicos del SDK de Kinect para la captura de imágenes se procederá a describir el proceso completo para poder transmitir imágenes del entorno hacia la aplicación usando Kinect.

Para lograr la captura de imágenes previamente se debe decidir el formato del flujo de imagen, la resolución y velocidad de captura, el método de captura. Para el desarrollo de este proyecto se definirán los siguientes parámetros.

Formato de flujo de imagen: RGB

Resolución y velocidad de captura: 640 x 480 a 30 cuadros por segundo

Método de Captura: Método por evento

Estos parámetros deben ser configurados mediante código en la aplicación, a continuación en la Figura 4.7., se muestra el proceso de captura con su respectivo código.

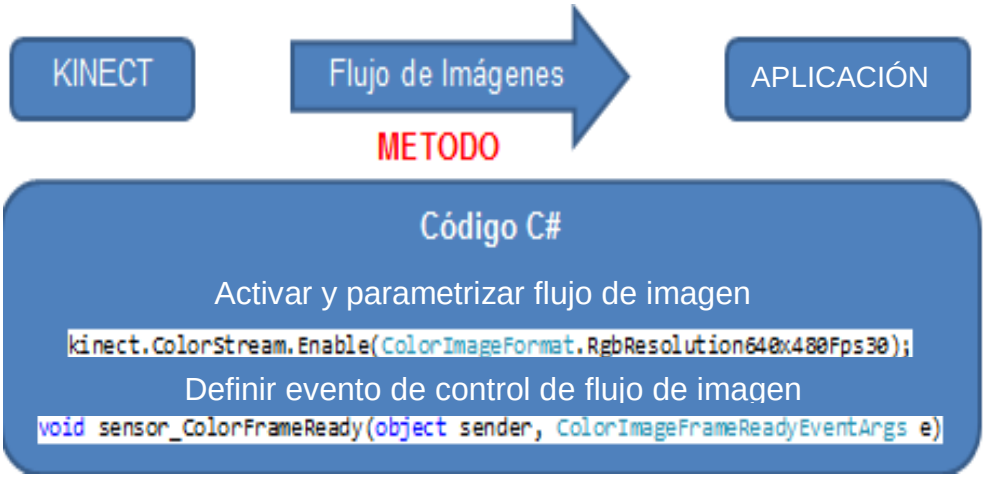

Figura 4.7. Proceso y código para captura de imágenes

Fuente: El autor.

Cabe mencionar que se definen estos parámetros debido que el proyecto necesita una buena calidad de imagen y para poder procesar correctamente éstas es necesario tener el mayor número de muestras que el Kinect pueda capturar.

4.2.5. RECONOCIMIENTO DE ARTICULACIONES DE LAS EXTREMIDADES SUPERIORES

Otra característica que posee el SDK de Kinect es la segmentación y reconocimiento del esqueleto humano, pero antes de llegar a este punto se analizará los principios que usa el dispositivo Kinect para lograr este objetivo.

El dispositivo Kinect devuelve datos (pixeles) de profundidad sin procesar, donde cada pixel contiene un valor que representa la distancia entre el dispositivo y los objetos.

Al utilizar el emisor infrarrojo y el sensor de profundidad estos dos elementos puede simular el efecto de visión estereoscópica y calcular la profundidad de los pixeles de una imagen.

Visión estereoscópica es la que emplea más de una imagen para obtener una representación tridimensionalidad de un entorno.

A continuación se muestra la manera que el dispositivo Kinect simula la visión estereoscópica.

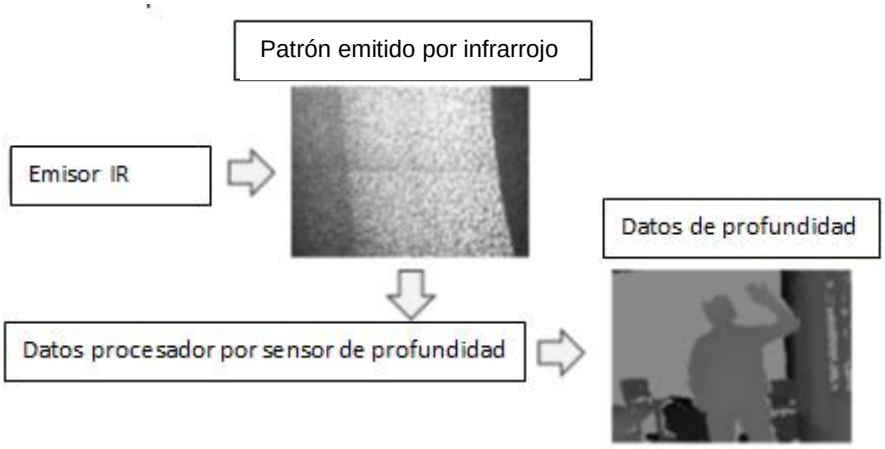

Figura 4.8. Procesamiento del sensor de profundidad.

Fuente: El autor.

Como se puede apreciar en la Figura 4.8. el dispositivo Kinect activa el emisor de infrarrojo el cual dispara un patrón de puntos hacia el entorno, este patrón es capturado por el sensor de profundidad y luego estos datos son procesados de tal manera que el dispositivo puede calcular la distancia de un pixel del entorno

Luego de exponer cómo el dispositivo puede capturar la profundidad del entorno se empezará con la explicación de otra característica del SDK y del dispositivo Kinect, la cual es conocida como SkeletalTracking.

La característica más notoria del dispositivo Kinect es el Skeletal tracking. Skeletal tracking significa seguimiento de esqueleto y se basa en un algoritmo que logra identificar partes del cuerpo de las personas que están en el campo de visión del sensor.

Por medio de este algoritmo se puede obtener puntos que hacen referencia a las partes del cuerpo de una persona y hacer un seguimiento de éstos identificando movimientos y/o posturas.

El proceso de identificación del esqueleto tiene varias fases, la primera es obtener los datos del mapa de profundidad para separar los distintos usuarios que podrían encontrarse en el campo de visión del Kinect basándose en el fondo o entorno.

Una vez identificados los usuarios del software se procede a clasificar las distintas partes del cuerpo, para más tarde obtener los Joints o articulaciones.

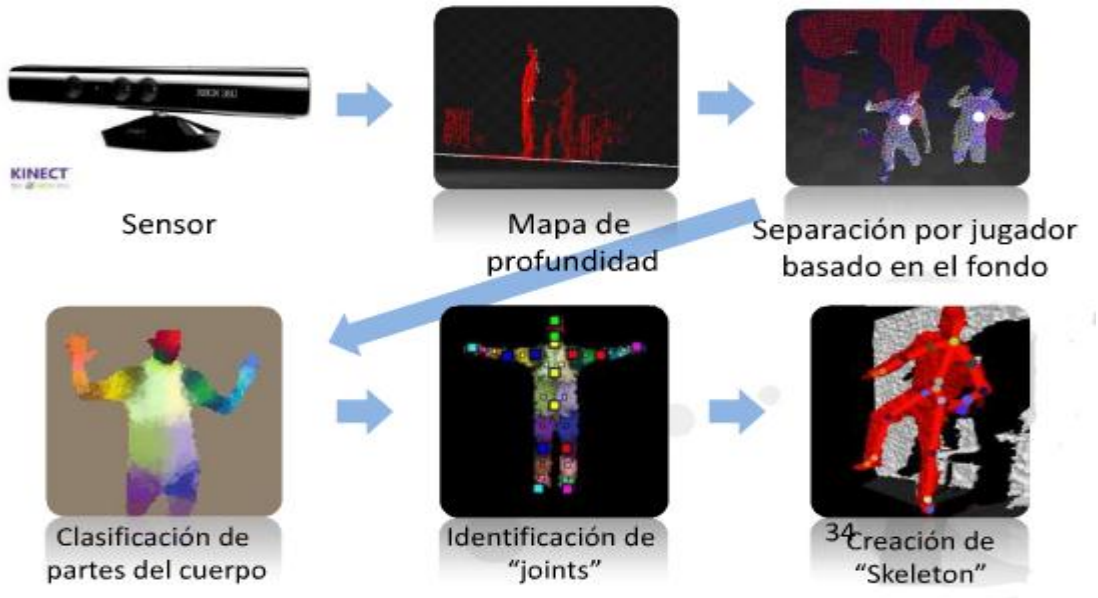

Figura 4.9. Proceso de reconocimiento de usuario y articulaciones.

Fuente: (edmsn, 2011)

El algoritmo de seguimiento de esqueleto está albergado en el SDK de Kinect, este algoritmo soporta el seguimiento de hasta 20 puntos los cuales representan las articulaciones del usuario que detecta el dispositivo Kinect. Cada punto es identificado por un nombre según su ubicación:

- Cabeza
- Hombros
- Codos
- Muñecas
- Brazos
- Espalda
- Cintura
- Rodillas
- Tobillos, así sucesivamente

Además de realizar el reconocimiento de las articulaciones el algoritmo consta de la función denominada Skeleton-Traking State, esta función define la forma de procesamiento de los datos de cada articulación, los cuales son:

- **Tracked:** En este estado el algoritmo es capaz de calcular las posiciones especiales de cada articulación

- **Not Tracked:** En este estado el algoritmo no es capaz de ubicar las posiciones espaciales de cada articulación

- **Inferred:** En este estado el algoritmo no tiene completa visibilidad de las articulaciones pero es capaz de calcular la ubicación espacial de estas basándose en otra articulación.

A continuación se muestra la captura de las posiciones basadas en el algoritmo de Skeletal Traking

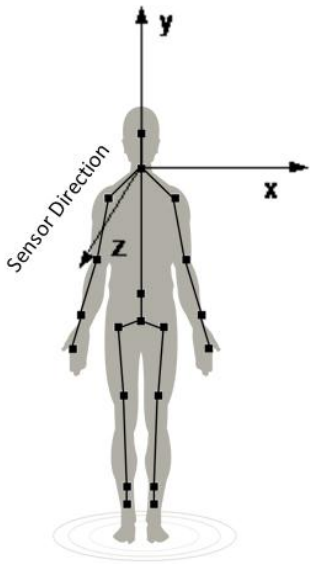

Figura 4.10. Ubicación espacial del cuerpo humano respecto a Kinect.

Fuente: (edmsn, 2011)

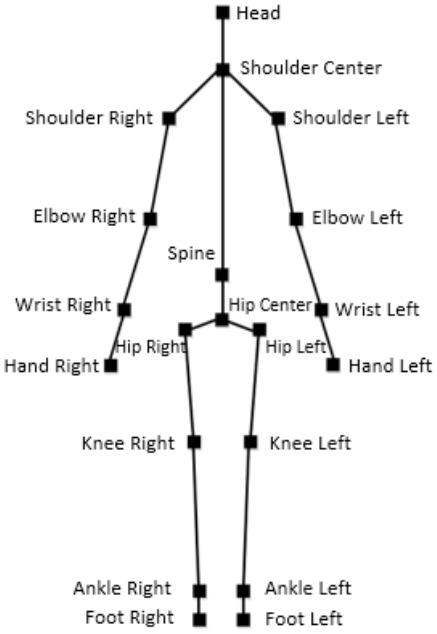

Figura 4.11. Reconocimiento y ubicación de articulaciones.

Fuente: (edmsn, 2011)

Para lograr formar una representación visual de un esqueleto humano como el de la Figura 4.11. es necesario acceder a otra librería del SDK la cual se denomina Bones, esta librería permite dibujar una unión entre dos articulaciones, esta librería no es nada más que un algoritmo de interfaz de usuario y no es tan trascendente en el desarrollo del proyecto, únicamente se la hace referencia para tener conocimiento de cómo se puede generar una representación visual del algoritmo

A continuación se muestra la forma de realizar esta representación visual

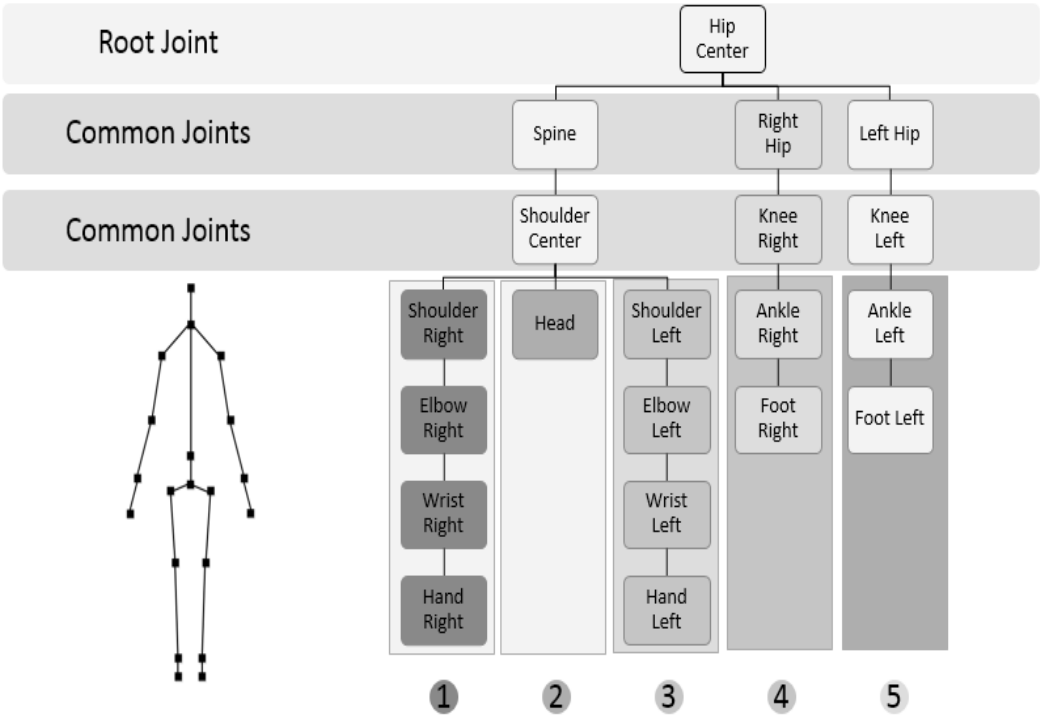

Figura 4.12. Jerarquía de articulaciones.

Fuente: (edmsn, 2011)

Como se puede apreciar en la Figura 4.12. para poder realizar la interfaz visual es necesario tener en cuenta que las articulaciones se definen por jerarquías, estas jerarquías permiten conectar las articulaciones de tal manera que cuando se unan estas puedan representar un esqueleto humano.

Luego de conocer las características del dispositivo Kinect para la realización de este proyecto, se continuará con el desarrollo del algoritmo para encontrar las posiciones angulares entre dos articulaciones las cuales son vitales para el proyecto, ya que estos datos serán codificados y enviados al robot para que este sea capaz de posicionarse y copiar el movimiento del usuario.

4.2.6. ANÁLISIS DE POSICIONES DE LAS EXTREMIDADES DEL USUARIO

El desarrollo del algoritmo capaz de calcular los ángulos que existen entre dos articulaciones será programado en C# y luego será anidado en el programa final, el algoritmo utilizará todas las características del dispositivo Kinect anteriormente analizadas, de esta manera la aplicación está totalmente basada en el SDK y es compatible tanto con Kinect para Windows como para Kinect para Xbox360.

Para poder calcular el ángulo entre las articulaciones del usuario primero se debe ubicar las coordenadas espaciales ***X,Y,Z*** de cada articulación, luego de tener las coordenadas de cada articulación se procede a formar vectores de posición de cada articulación, estos vectores tendrán como origen el centro del Kinect, se decide usar como centro de coordenadas al dispositivo Kinect debido a que si se usa otro punto el algoritmo para la ubicación espacial de las coordenadas debería realizar un autoajuste de coordenadas continuamente y esto provocaría que los datos no sean los reales y por consiguiente el control del robot tenga problemas, luego de obtener los vectores de posición relativos al dispositivo Kinect, se continua con el cálculo de los vectores entre cada articulación y de igual manera todos estos serán relativos al dispositivo Kinect, de esta forma se asegura que todo el sistema esté referenciado hacia el Kinect y además que los valores calculados sean reales, finalmente luego de conocer todos los vectores que definen al sistema, se utilizará los conceptos de cosenos directores y producto punto para calcular los ángulos necesarios para tener el control de los eslabones del robot manipulador.

4.2.7. CÁLCULO DE VECTORES POSICIÓN

El cálculo de los vectores posición de cada una de las articulaciones depende principalmente de la obtención de las coordenadas espaciales de las articulaciones, a estas coordenadas se puede acceder desde el SDK de Kinect, el cual devuelve las coordenadas XYZ de cada una de las articulaciones, estos datos están expresados en metros, para poder utilizar estos datos en una aplicación es necesario utilizar el siguiente código:

Tabla 4.1. Algoritmo de acceso a coordenadas espaciales.

Coordenada	Código C#	Articulación
X	skeleton.Joints[JointType.HandRight].Position.X;	Mano derecha
Y	skeleton.Joints[JointType.HandRight].Position.Y;	Mano derecha
Z	skeleton.Joints[JointType.HandRight].Position.Z;	Mano derecha

Fuente: El autor.

Como se puede apreciar en la Tabla 4.1, el acceso a las coordenadas espaciales de las articulaciones es simple, se debe tener en consideración la articulación a la cual se accede y que el valor que retorna el SDK está en metros.

Luego de acceder a las coordenadas espaciales de las articulaciones se procede a describir el proceso de análisis para definir los vectores posición de cada articulación y los vectores posición entre articulaciones

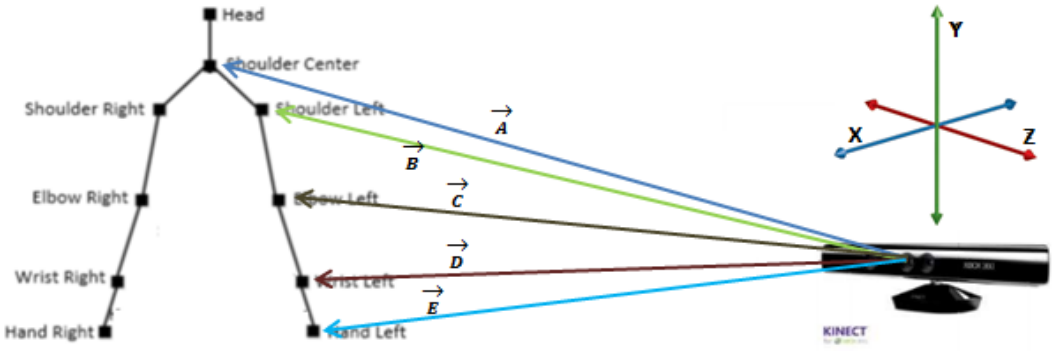

Figura 4.13. Vectores posición respecto al origen (Kinect).

Fuente: El autor.

Con ayuda de la Figura 4.13 se puede describir de una manera más fácil el vector posición de cada una de las articulaciones.

Estos vectores cuentan con tres coordenadas, conociendo esto se continúa con el cálculo de los vectores entre las articulaciones, para esto únicamente se usa la suma y resta de vectores

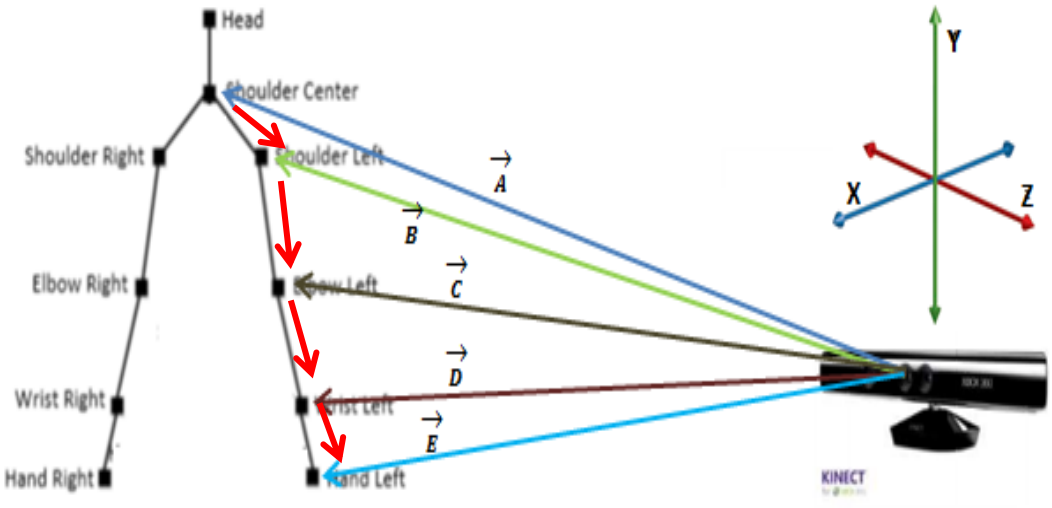

Figura 4.14. Vector posición entre articulaciones.

Fuente: El autor.

Los vectores de color rojo representan los vectores entre las articulaciones, para calcular estos vectores se usan conceptos básicos de vectores

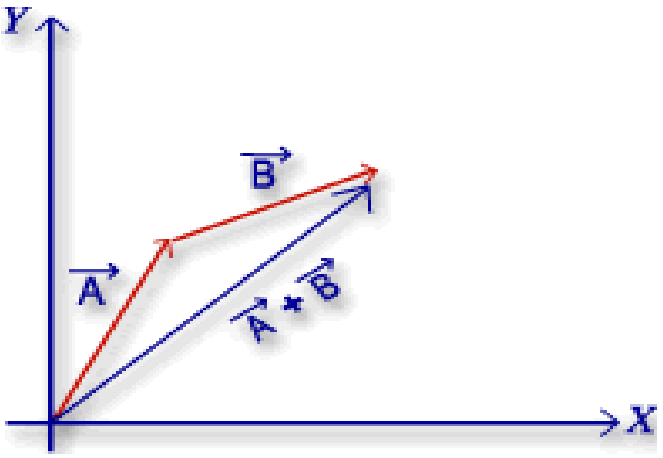

Figura 4.15. Suma de vectores método gráfico.

Fuente: El autor.

En la Figura 4.15 se muestra la manera en la cual se procede a realizar el análisis para encontrar el vector que representa la unión de dos articulaciones, extrapolando la Figura 4.15 al sistema de vectores, del presente trabajo; se puede decir que el origen está representando por el dispositivo Kinect, Vector A y Vector A+B son los vectores posición de cada articulación, Vector B representa la unión entre dos articulaciones, con esto en mente se tiene que el Vector B estaría representado por la suma algebraica del Vector A y el vector A+B, de esta manera se definen los vectores que unen todas las articulaciones.

Luego de haber ubicado todos los vectores que intervienen en el sistema se procede a estudiar la morfología de un robot manipulador para definir los ángulos necesarios para la correcta teleoperación del robot.

Como se puede apreciar en la Figura 2.10 cada articulación del usuario está ligada a una articulación del robot pero para el proyecto se realiza una variación y radica en hacer coincidir la base del robot con la cintura del operador , ya que a pesar de que Kinect es un dispositivo muy potente para adquirir datos del cuerpo humano, no es capaz de adquirir rotaciones de ciertas articulaciones en este caso de la cintura del usuario, para poder controlar la rotación de la base del robot se emplea el ángulo que forma el brazo con el pecho, ya que este ángulo puede simular fácilmente la rotación de la cintura del usuario; a continuación se muestra la modificación de la morfología.

Como se denota en la Figura 4.16 cada eslabón del robot está asociado directamente a un vector el cual será el responsable de generar los ángulos que controlaran el movimiento de cada articulación del robot.

Después de haber analizado de manera teórica como se procederá a generar los ángulos necesarios para el control de movimiento del robot, se continuará con la descripción analítica de este proceso.

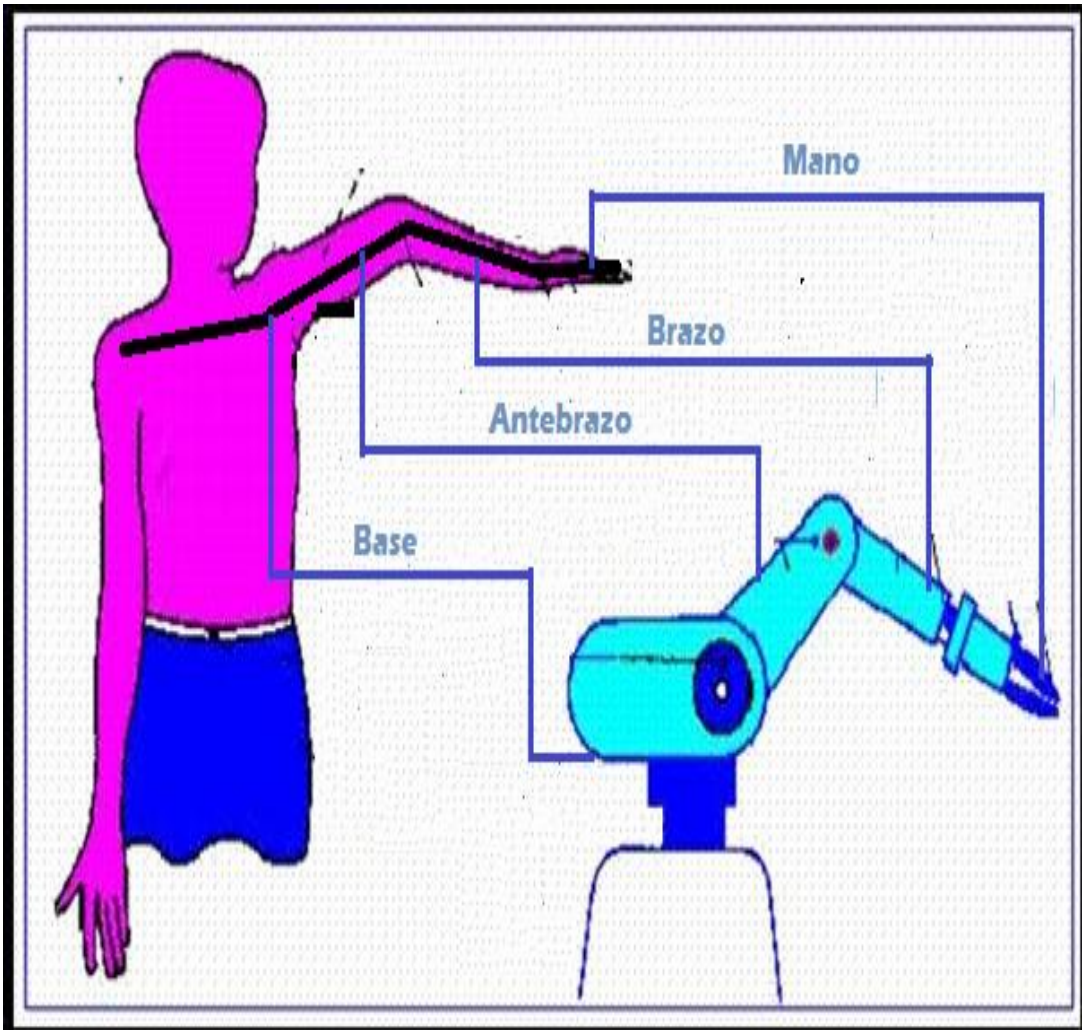

Figura 4.16. Adaptación de la morfología del robot

Fuente: El autor.

4.2.8. ANÁLISIS TRIGONOMÉTRICO

Los únicos datos que se conocen para este análisis son los vectores obtenidos mediante el SDK los cuales van a representar las coordenadas espaciales de cada articulación, a continuación se muestra el código y análisis matemático para la obtención de los vectores de articulación y los vectores que unen las articulaciones

4.2.9. OBTENCIÓN DE VECTORES POSICIÓN DEL BRAZO DERECHO DEL USUARIO UTILIZANDO EL SDK DE KINECT

Tabla 4.2. Código de acceso a coordenadas espaciales.

Articulación	Código C#
Mano derecha	skeleton.Joints[JointType.HandRight].Position.X;
	skeleton.Joints[JointType.HandRight].Position.Y;
	skeleton.Joints[JointType.HandRight].Position.Z;
Muñeca derecha	skeleton.Joints[JointType.WristRight].Position.X;
	skeleton.Joints[JointType.WristRight].Position.X;
	skeleton.Joints[JointType.WristRight].Position.X;
Codo derecho	skeleton.Joints[JointType.ElbowRight].Position.X;
	skeleton.Joints[JointType.ElbowRight].Position.Y;
	skeleton.Joints[JointType.ElbowRight].Position.Z;
Hombro derecho	skeleton.Joints[JointType.ShoulderRight].Position.X;
	skeleton.Joints[JointType.ShoulderRight].Position.Y;
	skeleton.Joints[JointType.ShoulderRight].Position.Z;

Fuente: El autor.

Una vez obtenidas las coordenadas espaciales de cada articulación se calcula el vector que une cada una de estas articulaciones.

4.2.10. VECTORES ENTRE ARTICULACIONES

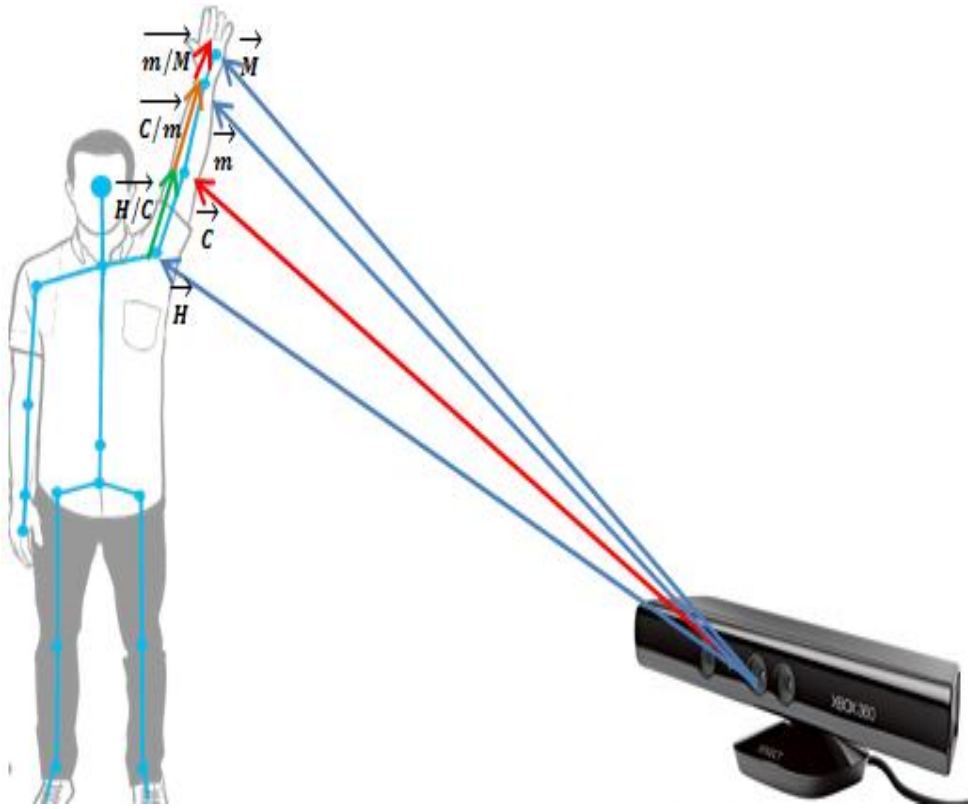

Figura 4.17. Vectores del sistema de visión artificial.

Fuente: El autor.

Vector Hombro/Codo

- $\vec{H} + \overrightarrow{H/C} = \overrightarrow{C}$

$$\overrightarrow{\boldsymbol{H/C}} = \overrightarrow{\boldsymbol{C}} - \overrightarrow{\boldsymbol{H}} \qquad \textbf{[4.1]}$$

Vector Codo/Muñeca

- $\vec{C} + \overrightarrow{C/m} = \overrightarrow{m}$

$$\overrightarrow{C/m} = \overrightarrow{m} - \overrightarrow{C} \quad \textbf{[4.2]}$$

Vector Muñeca/Mano

- $\vec{m} + \overrightarrow{m/M} = \overrightarrow{M}$

$$\overrightarrow{m/M} = \overrightarrow{M} - \overrightarrow{m} \quad \textbf{[4.3]}$$

Después de realizar el análisis de los vectores posición de todo el sistema se realiza el análisis para obtener el ángulo que se forma entre los distintos vectores, para esto se utiliza el concepto de producto punto.

El producto punto de dos vectores es un número real que resulta al multiplicar el producto de sus módulos por el coseno del ángulo que forman.

$$\vec{A}.\vec{B} = |\vec{A}|.|\vec{B}|.\cos \propto \quad \textbf{[4.4]}$$

Luego de conocer el concepto de producto punto se continuará con la aplicación del producto punto en el proyecto.

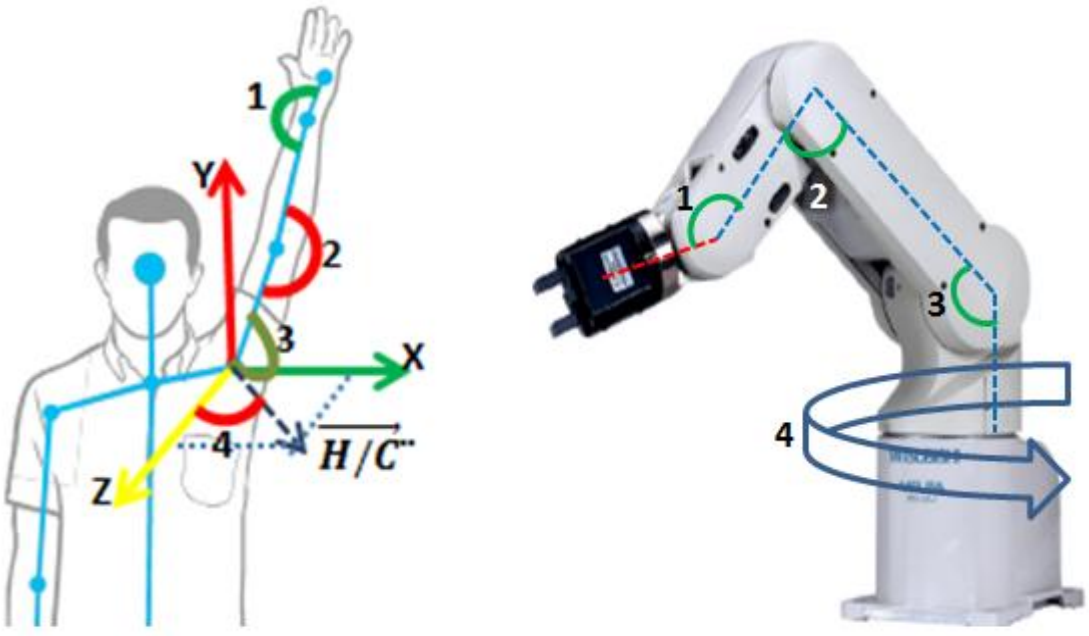

Figura 4.18. Relación de ángulos entre usuario y robot

Fuente: El autor.

La Figura 4.18 denota los ángulos que se utilizarán para la teleoperación del robot, como se puede apreciar existen cuatro ángulos los cuales estarán ligados a cada una de las articulaciones del robot.

Calculo de ángulos

$$\measuredangle \mathbf{1} = \mathbf{cos^{-1}}\left(\frac{\overrightarrow{m/M}.\overrightarrow{C/m}}{|\overrightarrow{m/M}|.|\overrightarrow{C/m}|}\right) \quad \textbf{[4.5]}$$

$$\measuredangle \mathbf{1} = \mathbf{cos^{-1}}\left(\frac{\left(\overrightarrow{m/M}\, i\, .\ \overrightarrow{C/m}\, i\right) + \left(\overrightarrow{m/M}\, j\, .\ \overrightarrow{C/m}\, j\right) + \left(\overrightarrow{m/M}\, k\, .\ \overrightarrow{C/m}\, k\right)}{\left|\overrightarrow{m/M}\right|.\left|\overrightarrow{C/m}\right|}\right)$$

$$\measuredangle \mathbf{2} = \mathbf{cos^{-1}}\left(\frac{\overrightarrow{C/m}.\overrightarrow{H/C}}{|\overrightarrow{C/m}|.|\overrightarrow{H/C}|}\right) \quad \textbf{[4.6]}$$

$$\measuredangle \mathbf{2} = \mathbf{cos^{-1}}\left(\frac{\left(\overrightarrow{C/m}\, i\, .\ \overrightarrow{H/C}\, i\right) + \left(\overrightarrow{C/m}\, j\, .\ \overrightarrow{H/C}\, j\right) + \left(\overrightarrow{C/m}\, k\, .\ \overrightarrow{H/C}\, k\right)}{\left|\overrightarrow{C/m}\right|.\left|\overrightarrow{H/C}\right|}\right)$$

$$\measuredangle \mathbf{3} = \mathbf{cos^{-1}}\left(\frac{\overrightarrow{H/C}.\overrightarrow{H/C"}}{|\overrightarrow{H/C}|.|\overrightarrow{H/C"}|}\right) \quad \textbf{[4.7]}$$

$$\measuredangle \mathbf{3} = \mathbf{cos^{-1}}\left(\frac{\left(\overrightarrow{H/C}\, i\, .\ \overrightarrow{H/C"}\, i\right) + \left(\overrightarrow{H/C}\, j\, .\ \overrightarrow{H/C"}\, j\right) + \left(\overrightarrow{H/C}\, k\, .\ \overrightarrow{H/C"}\, k\right)}{\left|\overrightarrow{H/C}\right|.\left|\overrightarrow{H/C"}\right|}\right)$$

Para el cálculo del ángulo 4 que controla la base es necesario definir un vector adicional, el cual estará ubicado en el hombro del usuario, de esta manera se asegura el cálculo de la rotación, a este vector se lo denomina Vector H/C1

$$\measuredangle \mathbf{4} = \mathbf{cos^{-1}}\left(\frac{\overrightarrow{H/C}.\overrightarrow{H/C1}}{|\overrightarrow{H/C}|.|\overrightarrow{H/C1}|}\right) \quad \textbf{[4.8]}$$

$$\measuredangle 4 = \cos^{-1}\left(\frac{\left(\overrightarrow{H/C}\,i\,.\,\overrightarrow{H/C1}\,i\right)+\left(\overrightarrow{H/C}\,j\,.\,\overrightarrow{H/C1}\,j\right)+\left(\overrightarrow{H/C}\,k\,.\,\overrightarrow{H/C1}\,k\right)}{\left|\overrightarrow{H/C}\right|.\left|\overrightarrow{H/C1}\right|}\right)$$

Con esto quedan definidas de manera analítica las ecuaciones necesarias para definir los ángulos que controlan los movimientos de las articulaciones del robot manipulador.

Diagrama de flujo para el cálculo de ángulos basado en las ecuaciones.

A continuación se muestra el diagrama de flujo para el cálculo de ángulos entre eslabones

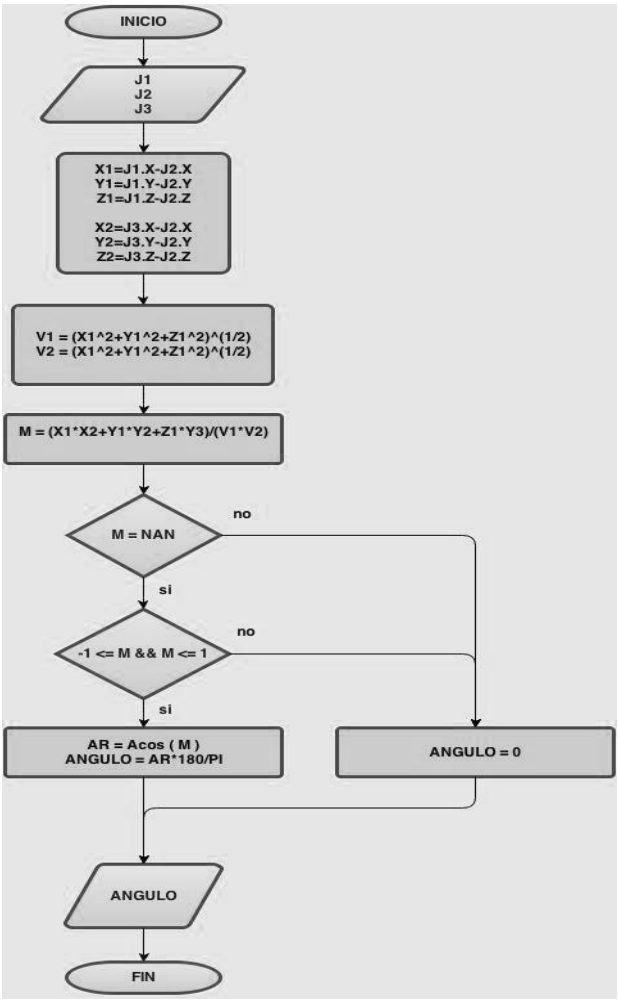

Figura 4.16. Diagrama de flujo para el cálculo de ángulos

Fuente: El autor.

4.3. SUBSISTEMA DE CONTROL

4.3.1. DESCRIPCIÓN DE LA UNIDAD CONTROLADORA CR1-571

Este controlador es el cerebro del robot y es donde se encuentra el sistema de control del Robot. El CR1-571 se basa en la misma arquitectura que utilizan los robots de mayores dimensiones de la marca Mitsubishi esta arquitectura es RISC, trabajando con las mismas posibilidades y lo más importante, el mismo lenguaje de programación. El controlador del robot permite la ejecución en paralelo de hasta 32 programas en modo multitarea. Un aspecto muy importante es el puerto RS-232 que se utiliza para descargar los programas diseñados en la computadora y también se utiliza para comunicar el robot con dispositivos externos. Cabe destacar que por medio de un adaptador se puede comunicar el robot mediante red Ethernet (Protocolo TCP/IP), o bien con una red CC-Link de Mitsubishi que permite el intercambio rápido de datos, sobre todo entre el robot y un PLC (Controlador Lógico Programable). Otro punto importante del controlador es que permite almacenar los programas en memoria, de esta forma se pueden guardar 88 programas de 5000 líneas cada uno y también se puede guardar 2500 posiciones.

En el Anexo 2 se muestran más características del controlador.

Luego de revisar las principales características del controlador se continuará con la descripción de la aplicación necesaria para realizar la comunicación entre la unidad controladora y el PC sin utilizar el software de fábrica que utiliza la unidad controladora

4.3.2. COMUNICACIÓN ENTRE PC Y CONTROLADOR CR1-571

Para realizar la comunicación entre una PC y la unidad controladora CR1-571 es necesario tener en cuenta los siguientes parámetros.

Tabla 4.3. Parámetros de comunicación.

Bits por segundo:	9600
Bits de datos:	8
Paridad:	Par
Bits de parada:	2
Control de flujo:	Ninguno

Fuente: El autor.

Estos parámetros fueron extraídos desde Cosimir, que es el programa recomendado por Mitsubishi para la programación del controlador CR1-571

Estos parámetros son fundamentales para realizar una correcta comunicación, ya que si no se respeta esto la comunicación jamás se llevará a cabo, además de todo esto es necesario un requisito de hardware en la PC el cual es un puerto RS232, ya que la PC utilizada en este proyecto no cuenta con este tipo de conexión se decide usar un conversor de USB a RS232.

Luego de conocer estos parámetros básicos para la comunicación, se realiza un algoritmo en C#, el cual debe controlar el puerto USB de tal manera que pueda enviar y recibir datos según sea la necesidad de la aplicación.

A continuación se describe el diagrama de flujo para la comunicación entre la PC y la unidad controladora.

Como se puede apreciar el algoritmo está parametrizado con los valores requeridos por la unidad controladora.

Luego de parametrizar todo el algoritmo de comunicación, se diseñan los algoritmos de escritura de datos, estos algoritmos se van a encargar de

enviar las posiciones procesadas por el algoritmo de cálculo de ángulos entre articulaciones.

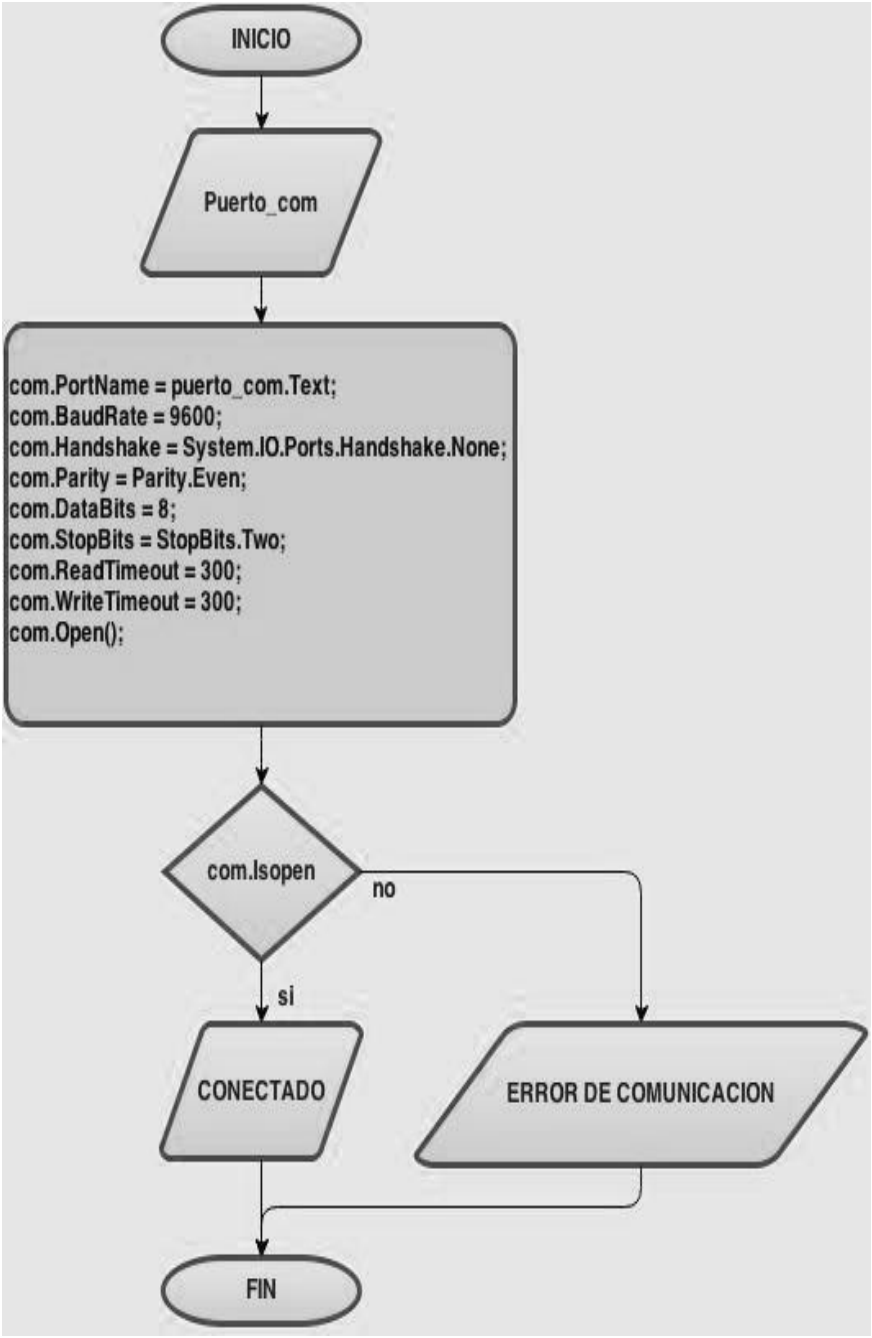

Figura 4.17. Diagrama de flujo Comunicación PC - Unidad Controladora

Fuente: El autor.

Para poder enviar y escribir datos en la unidad controladora es necesario conocer las palabras reservadas que utiliza la unidad, a continuación se muestra una lista de las principales palabras que se utilizan:

Tabla 4.4. Comandos de control.

Cadena de control	***Propósito***
1;1;RSTALRM	Reiniciar el controlador
1;1;STATE	Verificar el estado del controlador
1;1;SRVON	Arrancar los servomotores del robot
1;1;EXECJOVRD	Reescribir las coordenadas
1;1;EXECJCOSIROP	Almacenar nuevas coordenadas
1;1;SAVE	Guardar la rutina actual
1;1;STOP	Parar la ejecución de una rutina
1;1;RUN	Ejecutar el programa almacenado
1;1;EXECHOPEN 1	Abrir el actuador
1;1;EXECHCLOSE 1	Cerrar el actuador
1;1;RSTPRG	Reinicia un programa para borrarlo
1;1;FDEL	Borra un programa
1;1;NEW	Crea un nuevo programa
1;1;LOAD	Asigna el nombre a un programa
1;1;PRTVERLISTL	Crea una secuencia de puntos
1;1;PRTVEREMDAT	Envía una secuencia de datos
1;9;LISTL	Inicia una lista de puntos
1;9;EMDAT	Envía los puntos
1;1;SLOTINIT	Pone al controlador en el slot 1

Todas estas sentencias fueron capturadas del programa Cosimir, ya que sin estas sentencias sería imposible programar los movimientos en la unidad controladora.

A continuación se describe el diagrama de flujo para el envío y escritura de datos en la unidad controladora.

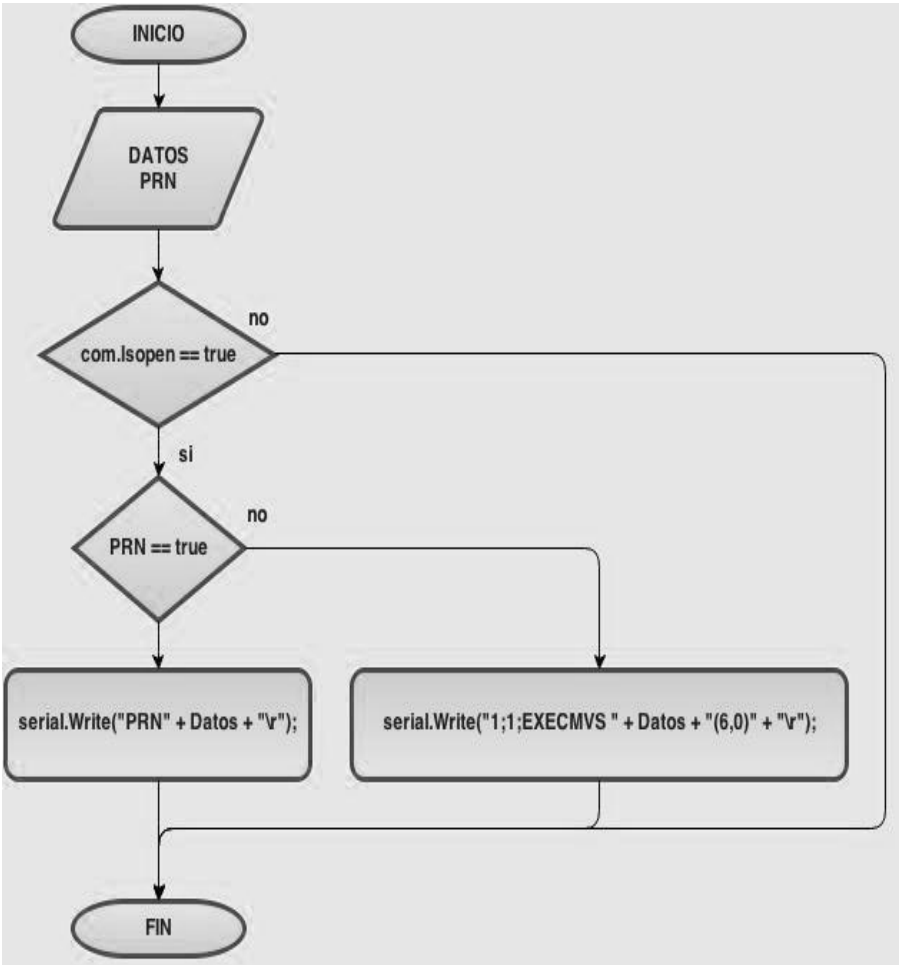

Figura 4.18. Diagrama de flujo para envio de datos

Fuente: El autor.

El algoritmo recibe dos variables, las cuales son: los datos a escribir, y un booleano que confirma si el método debe escribir hacia la controladora, además de esto cuenta con una sentencia de control la cual confirma si el puerto de comunicación se encuentra operativo.

Finalmente todos estos algoritmos, deberán ser anidados en la aplicación final.

A continuación se muestra un diagrama de bloques del funcionamiento del algoritmo comunicación del PC hacia la unidad controladora.

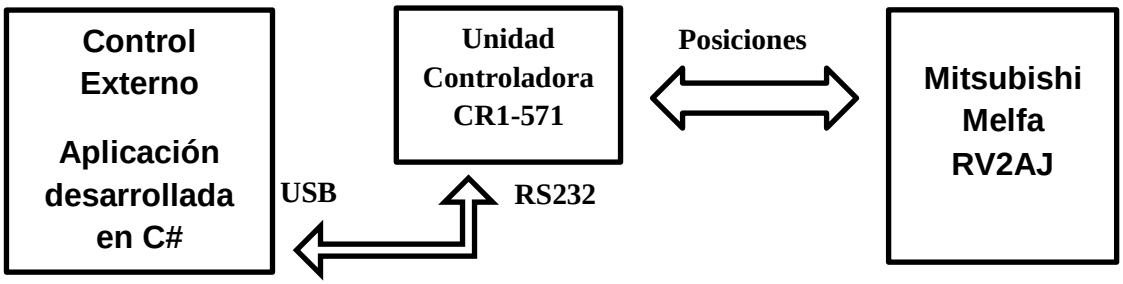

Figura 4.19. Comunicación Pc - Unidad controladora robot

Fuente: El autor.

4.4. SUBSISTEMA MECÁNICO

4.4.1. DESCRIPCIÓN DEL ROBOT MANIPULADOR MITSUBISHI MELFA RV2AJ

El brazo robótico RV-2AJ de Mitsubishi se puede ver en la Figura 4.21 Se trata de un robot de brazo articulado vertical con 5 grados de libertad, que se utiliza para el transporte de piezas. El diseño del RV-2AJ lo hace ideal para aplicaciones donde no sobra el espacio y con movimiento de cargas de hasta 2 Kg de peso. Este robot tiene un alcance de 410mm, y combina una velocidad máxima de 2,100mm/s con una precisión de ±0.02mm.

Los servomotores de corriente alterna, unidos a codificadores de posición absolutos, garantizan fiabilidad y bajo mantenimiento.

Estos codificadores permiten apagar el robot en cualquier momento y al conectar de nuevo la alimentación, el robot podrá continuar trabajando

desde la posición actual. El brazo tiene integradas en su interior una serie de conductos que permiten la conexión de pinzas y ventosas neumáticas.

También tiene integrado un conector para cuatro sensores, y la posibilidad de utilizar pinzas de accionamiento eléctrico. En el Anexo 1 se puede ver las características del brazo robótico.

Figura 4.20. Robot Mitsubishi Melfa RV2AJ

Fuente: (Robotics Equipment Corporation, 2014)

4.5. CINEMÁTICA ROBOT MANIPULADOR MELFA RV2AJ

4.5.1. CINEMÁTICA DIRECTA

El modelo cinemático directo, permite determinar la posición y orientación que adopta el extremo del robot, cuando cada una de las variables que fijan la posición u orientación de sus articulaciones toman valores determinados.

El modelo cinemático directo utiliza el algoritmo de Denavit-Hartenberg, el mismo que emplea matrices para representar la rotación y traslación de cada

eslabón, todo el algoritmo se encuentra detallado en el Capítulo 3 de este proyecto.

A continuación se procede con el desarrollo del algoritmo de D-H sobre la geometría del robot manipulador y de esta manera se definirá el modelo matemático para el movimiento del robot.

La Figura 4.25 muestra las medidas y los ejes de referencia de cada eslabón basado en el algoritmo de Denavit-Hartenberg.

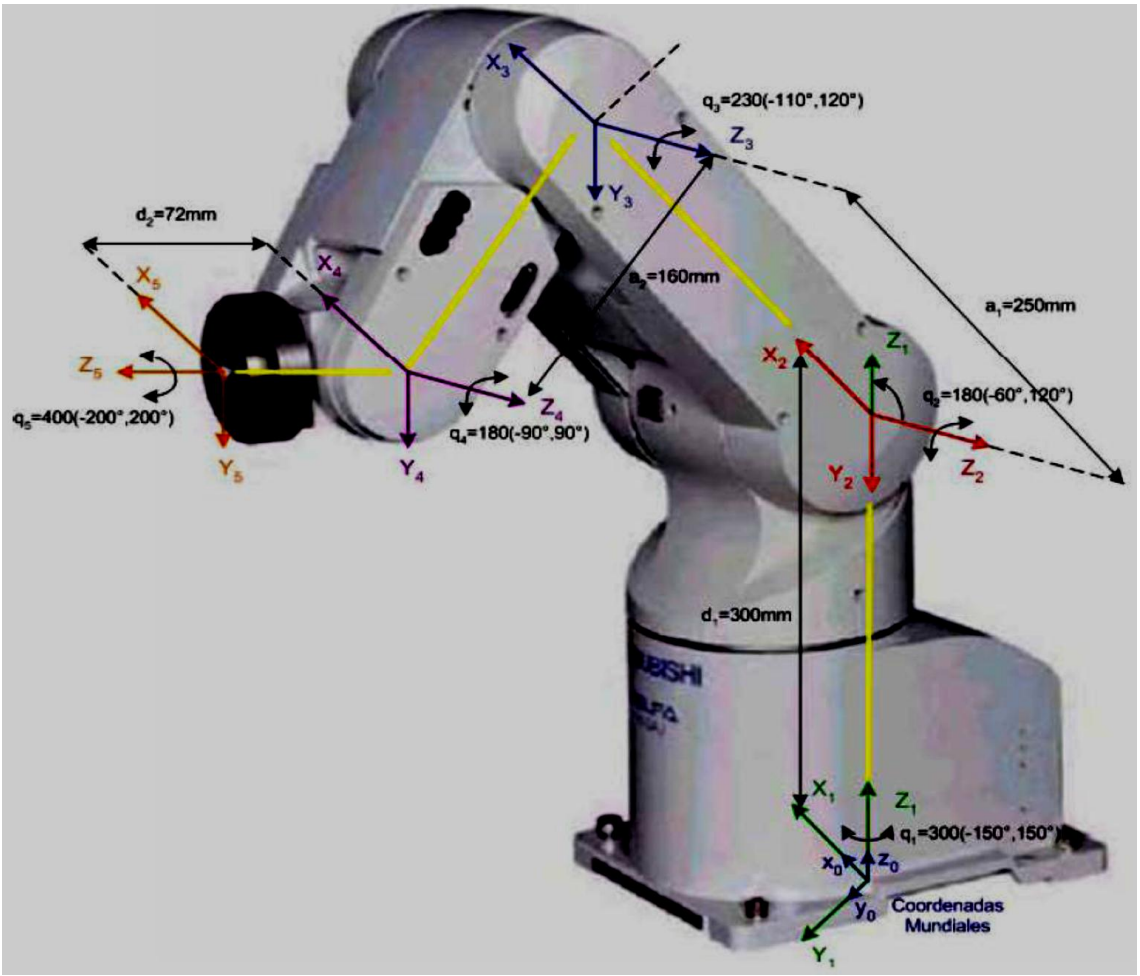

Figura 4.21. Robot Mitsubishi Melfa RV2AJ aplicado Denavit-Hartenberg

Fuente: (Gentilini, 2011)

Tabla 4.6. Parámetros Denavit-Hartenberg para el brazo robótico.

Articulación	θi (rad)	α1 (rad)	ai (mm)	di (mm)
1 (J1)	θ1	-90	0	0,3
2 (J2)	θ2-90º	0	0,25	0
3 (J3)	θ3	0	0,16	0
4 (J5)	θ4+90º	90	0	0
5 (J6)	θ5	0	0	0,72
6 (Pinza)	0	0	0	0,123

Una vez obtenidos los datos se procede con el desarrollo de las matrices que controlan el sistema robótico

$$ {}^{0}A_{1} = \begin{bmatrix} C1 & 0 & -S1 & 0 \\ S1 & 0 & C1 & 0 \\ 0 & -1 & 0 & di \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}A_{2} = \begin{bmatrix} C2 & -S2 & 0 & \propto_{2}.C2 \\ S2 & C2 & 0 & \propto_{2}.C2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}A_{3} = \begin{bmatrix} C3 & -S3 & 0 & \propto_{3}.C3 \\ S3 & C3 & 0 & \propto_{3}.S3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}A_{4} = \begin{bmatrix} C4 & 0 & S4 & 0 \\ S4 & 0 & -C4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}A_{5} = \begin{bmatrix} C5 & -S5 & 0 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 1 & d5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}A_{6} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & d6 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Dónde:

$$C \propto_i = \cos \propto_i$$

$$C\theta_i = \cos \theta_i = C_i$$

$$S \propto_i = \sin \propto_i$$

$$S\theta_i = \sin \theta_i = S_i$$

Luego para obtener la matriz de transformación homogénea **T** se aplica la fórmula correspondiente al modelo de Denavit-Hartenberg.

$$\boldsymbol{T = {}^{0}A_{1} \,.\, {}^{1}A_{2} \,.\, {}^{2}A_{3} \,.\, {}^{3}A_{4} \,.\, {}^{4}A_{5} \,.\, {}^{5}A_{6}} \qquad \textbf{[4.9]}$$

$$T = \begin{bmatrix} n_x & o_x & \propto_x & p_x \\ n_y & o_y & \propto_y & p_y \\ n_z & o_z & \propto_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

nx = ((C1*C2*C3-C1*S2*S3)*C4+(-C1*C2*S3-C1*S2*C3)*S4)*C5-S1*S5

ny = ((S1*C2*C3-S1*S2*S3)*C4+(-S1*C2*S3-S1*S2*C3)*S4)*C5+C1*S5

nz = [((-S2*C3-C2*S3)*C4+(S2*S3-C2*C3)*S4)*C5

ox = -((C1*C2*C3-C1*S2*S3)*C4+(-C1*C2*S3-C1*S2*C3)*S4)*S5-S1*C5

oy = -((S1*C2*C3-S1*S2*S3)*C4+(-S1*C2*S3-S1*S2*C3)*S4)*S5+C1*C5

oz = -((-S2*C3-C2*S3)*C4+(S2*S3-C2*C3)*S4)*S5

ax = (C1*C2*C3-C1*S2*S3)*S4-(-C1*C2*S3-C1*S2*C3)*C4

ay = (S1*C2*C3-S1*S2*S3)*S4-(-S1*C2*S3-S1*S2*C3)*C4

az = (-S2*C3-C2*S3)*S4-(S2*S3-C2*C3)*C4

px=((C1*C2*C3-C1*S2*S3)*S4-(-C1*C2*S3-
C1*S2*C3)*C4)*D6+((C1*C2*C3-C1*S2*S3)*S4-(-C1*C2*S3-
C1*S2*C3)*C4)*D5+C1*C2*A3*C3-C1*S2*A3*S3+C1*A2*C2

py=((S1*C2*C3-S1*S2*S3)*S4-(-S1*C2*S3-S1*S2*C3)*C4)*D6+((S1*C2*C3-S1*S2*S3)*S4-(-S1*C2*S3-S1*S2*C3)*C4)*D5+S1*C2*A3*C3-S1*S2*A3*S3+S1*A2*C2

pz=((-S2*C3-C2*S3)*S4-(S2*S3-C2*C3)*C4)*D6+((-S2*C3-C2*S3)*S4-(S2*S3 C2*C3)*C4)*D5-S2*A3*C3-C2*A3*S3-A2*S2+D1

Como se aprecia en las ecuaciones anteriores, queda reflejado el valor de la posición **(px, py, pz)** y orientación **(n, o, a)** del extremo del robot en función de las coordenadas articulares **(θ1, θ2, θ3, θ4, θ5, θ6).**

En cuanto a la orientación, fue necesario calcular el ángulo conocido como pitch (cabeceo) que corresponde al ángulo que posee la pinza de acuerdo al sistema de referencia establecido en la base del brazo. Este ángulo de cabeceo se obtuvo con la siguiente fórmula:

$$\boldsymbol{pitch = \theta = \cos^{-1} \alpha_z} \qquad \textbf{[4.10]}$$

Las demás componentes de los vectores (n, o, a), sirven para obtener otros datos que para la finalidad de este proyecto no fue necesario calcularlos, tal es el caso de los ángulos conocidos como: roll (alabeo) y yaw (guiñada). Las ecuaciones de orientación **(n, o, a)** son utilizadas en caso de querer resolver el problema de la cinemática inversa del brazo robótico.

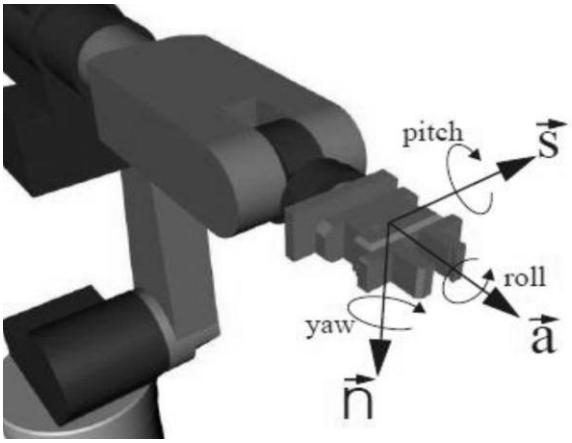

Figura 4.22. Orientación de la pinza de acuerdo al sistema de referencia en la base del brazo.

Fuente: (Gentilini, 2011)

4.6. SISTEMA DE SEGUIMIENTO INTELIGENTE

En este punto se implementó todos los subsistemas como uno solo, esto es posible porque el diseño se basó en la metodología mecatrónica y logrando realizar un diseño paralelo, en donde todos los subsistemas funcionan individualmente de manera correcta y al unir como un solo sistema, podrán interactuar eficientemente.

4.6.1. CONSOLIDACIÓN DE LOS SUBSISTEMAS Y VERIFICACIÓN DE FUNCIONAMIENTO

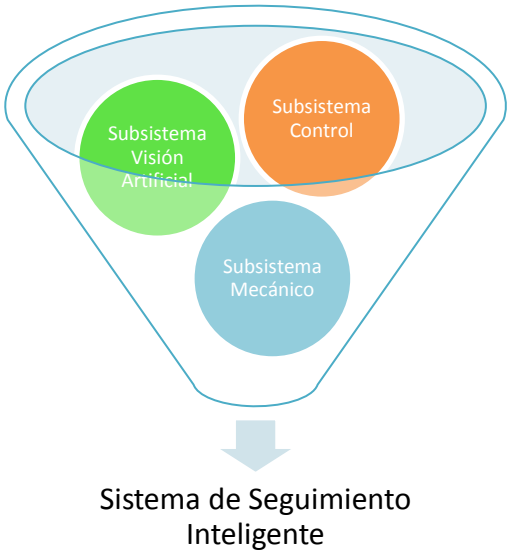

Figura 4.23. Consolidación de Subsistemas.

Fuente: El autor.

Luego de haber diseñado todos los subsistemas de una manera correcta, se podrá afirmar que el sistema mecatrónico funcionará de una manera eficaz y para poder verificar el funcionamiento de todos los subsistemas en conjunto, se realiza un conjunto de pruebas y estos resultados serán analizados y mostrados en el Capítulo 5.

5. IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se muestra la implementación de los sistemas, para luego realizar pruebas de funcionamiento.

5.1. IMPLEMENTACIÓN

En este punto se mostrará todos los subsistemas del proyecto de manera individual.

5.1.1. INTERFAZ APLICACIÓN

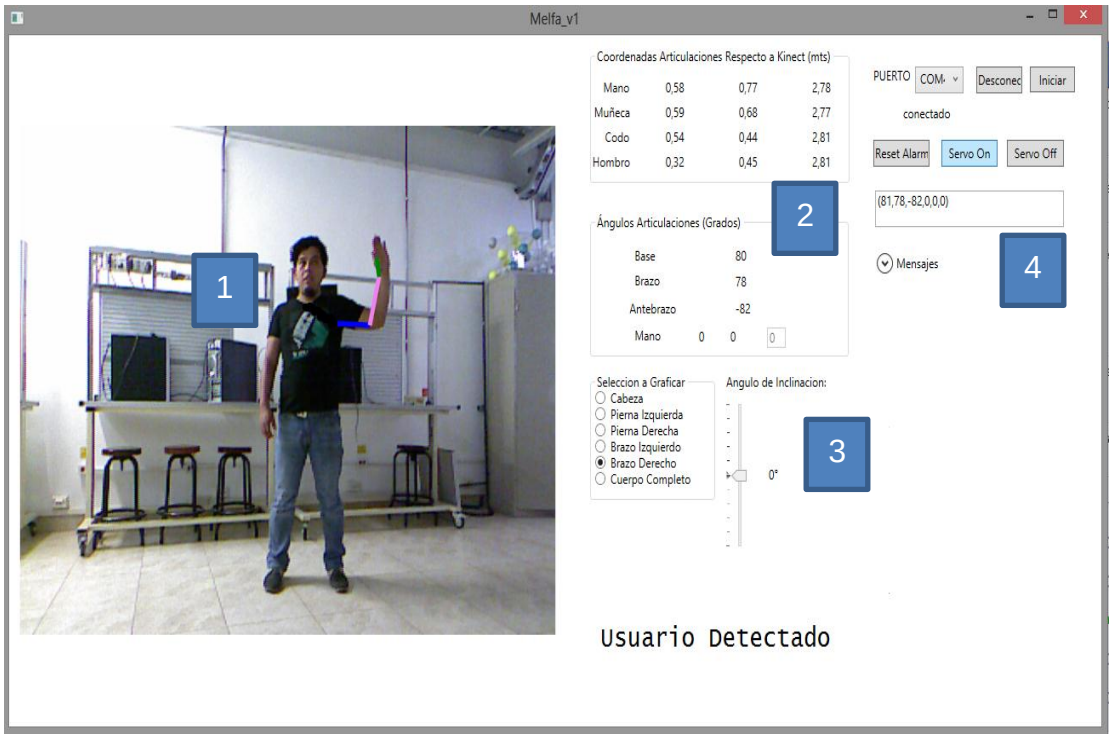

Figura 5.1. Interfaz Principal de la aplicación.

Fuente: El autor.

La pantalla principal de la aplicación contiene cuatro secciones:

1. **Visualizador de flujo de imagen:** Aquí se muestra la imagen del entorno que obtiene el Kinect, y además el procesamiento del esqueleto que realiza el algoritmo del SDK.

2. **Visualizador de cálculo de coordenadas y ángulos:** Muestra las posiciones espaciales de cada articulación del brazo derecho, y se procesan los ángulos entre cada articulación adaptándose a la morfología del brazo robótico.

3. **Slider ajuste ángulo de inclinación del Kinect:** Este slider permite el manejo del ángulo de captura en el eje vertical respecto al sensor.

4. **Sección de comunicación con el robot:** Esta sección es la que permite la comunicación desde la aplicación hacia el robot, esta sección cuenta con un selector de puerto y cinco botones los cuales controlan diferentes acciones de la unidad controladora:

    - **Conectar:** Abre y configura el puerto serial de la PC.
    - **Iniciar:** Configura la unidad controladora para que pueda recibir datos de manera externa.
    - **Reset Alarm:** Resetea posibles alarmas debido a posiciones inadecuadas y ubica al robot en una posición de seguridad.
    - **Servo On:** Activa los servomotores del robot.
    - **Servo Off:** Desactiva los servomotores del robot.

Gracias al diseño de esta aplicación ya no es necesario utilizar el software de fábrica del robot, por lo tanto para poder realizar el movimiento del robot no es necesario cargar ningún programa en la memoria de la unidad controladora.

5.1.2. COMUNICACIÓN SERIAL

En este punto únicamente describirá la implementación del sistema de comunicación, ya que en capítulos anteriores se ha definido las configuraciones necesarias para lograr este objetivo.

Figura 5.2. Adaptador USB a serial.

Fuente: El autor.

Figura 5.3. Puerto de comunicación de la unidad controladora.

Fuente: El autor.

Utilizando el adaptador de USB a serial es posible conectarse directamente hacia la unidad controladora sin necesidad de utilizar algún tipo de electrónica extra, cabe mencionar que para poder utilizar la unidad controladora en modo externo, es necesario ubicar el switch maestro en posición AUTO EXT.

Figura 5.4. Switch maestro.

Fuente: El autor.

5.1.3. SISTEMA DE VISIÓN ARTIFICIAL

El sistema de visión artificial utilizará el dispositivo Kinect para capturar imágenes y enviarlas a la aplicación para procesarlas, este dispositivo se conecta mediante USB a la PC, cabe recalcar que para que el dispositivo Kinect funcione correctamente es necesario conectar una fuente externa la cual provee la energía necesaria para que el dispositivo active todos sus elementos.

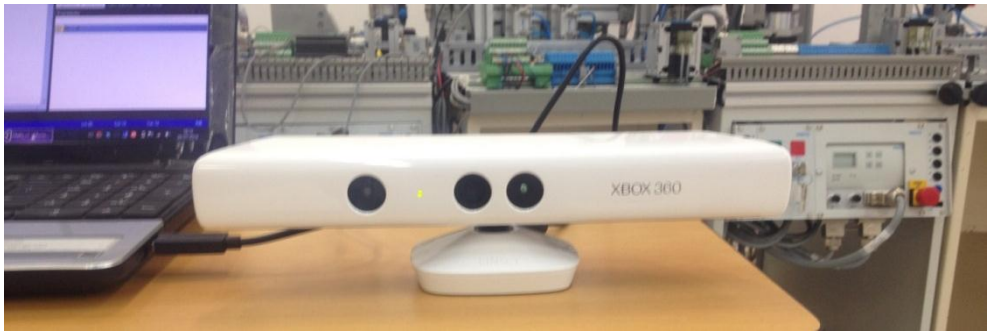

Figura 5.5. Sistema de visión artificial.

Fuente: El autor.

5.1.4. SISTEMA DE SEGUIMIENTO INTELIGENTE

Finalmente luego de implementar el sistema de control, visión artificial y mecánico, se procede a definir completamente el sistema mecatrónico basado en visión artificial para la teleoperación de un brazo robótico Melfa RV2AJ.

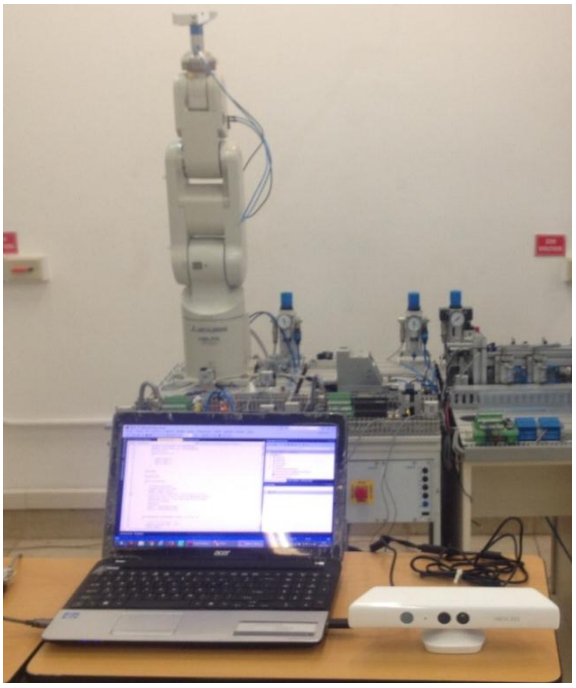

Figura 5.6. Implementación del sistema mecatrónico.

Fuente: El autor.

Como se observa en la Figura 5.6 el sistema de seguimiento inteligente se encuentra implementado y listo para poder realizar pruebas.

Siguiendo estos pasos se puede poner en marcha y ejecutar movimientos en el robot, sin necesidad de cargar ningún programa en la unidad controladora y sin realizar ningún programa para los movimientos del robot

5.2. PRUEBAS

Con el fin de verificar que el robot cumple los objetivos y delimitar sus acciones es sometido a una serie de pruebas, con el fin de observar el desempeño en su área de trabajo y de esta manera mostrar sus resultados.

5.2.1. CONSIDERACIONES GENERALES

El brazo robótico RV2AJ que se encuentra en el laboratorio de mecatrónica de la Universidad Tecnológica Equinoccial debe ser utilizado con fines exclusivamente didácticos y en condiciones absolutamente seguras.

Por lo tanto, el software de control cuenta con una posición de seguridad, la cual se activará cuando el software no reconozca a ningún operario en su campo de visión o cuando el usuario pulse el botón Reset en la aplicación.

El área de trabajo del robot debe encontrarse despejada para evitar colisiones no deseadas.

Únicamente un operario debe ubicarse frente del Kinect, si llegase a detectar más operarios el software no permitirá que la aplicación se comunique con el robot.

Se debe tomar en cuenta que el Kinect utiliza luz infrarroja para segmentar el cuerpo humano del entorno, por esto es importante eliminar fuentes de luz infrarroja innecesarias.

Al finalizar una sesión de trabajo es necesario cerrar la comunicación de una manera correcta, es decir pulsando el botón desconectar.

5.2.2. INICIALIZACIÓN DEL SISTEMA

Se definirán los pasos necesarios para poder activar el sistema de teleoperación:

1. Conectar Kinect a la PC.
2. Iniciar aplicación.
3. Ajustar ángulo de inclinación del Kinect.
4. Conectar adaptador USB a serial a la PC y a la unidad controladora del robot.
5. Encender unidad controladora, y esperar a que la unidad inicie (aproximadamente 10 segundos).
6. Ubicar switch principal en la posición AUTO EXT.
7. En la aplicación seleccionar puerto de comunicación.
8. Clic en CONECTAR.
9. Clic en INICIAR, esperar tres segundos para que la unidad controladora se configure para el control externo.
10. Clic en SERVO ON para activar servomotores del robot.
11. Ubicar a usuario frente a Kinect y empezar con los movimientos.
12. Al terminar retirarse del campo de visión del Kinect, y esperar a que el robot se ubique automáticamente en la posición de seguridad.
13. Clic en DESCONECTAR.
14. Cerrar la aplicación.
15. Desconectar tanto el Kinect y el adaptador USB a serial de la PC.

5.2.3. RECONOCIMIENTO DE POSICIONES

Para determinar el correcto funcionamiento de la aplicación se realizó varios movimientos con la extremidad superior derecha del operario. Se observó y calculó el error absoluto de cada posición comparando la posición del operario con la del robot.

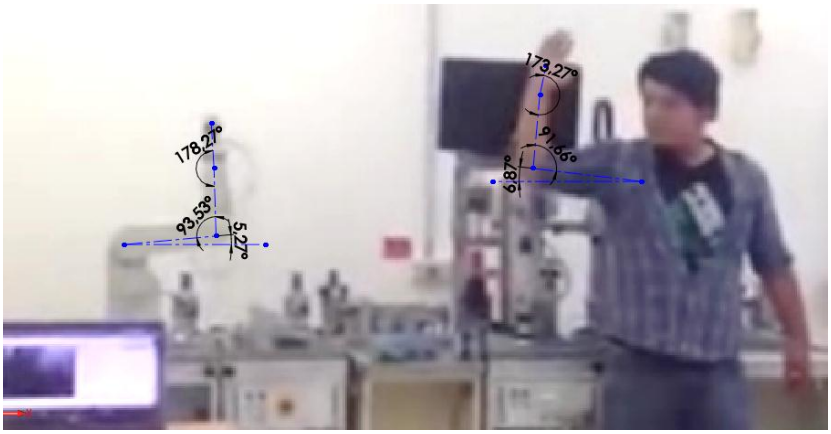

Figura 5.7. Posición 1.

Fuente: El autor.

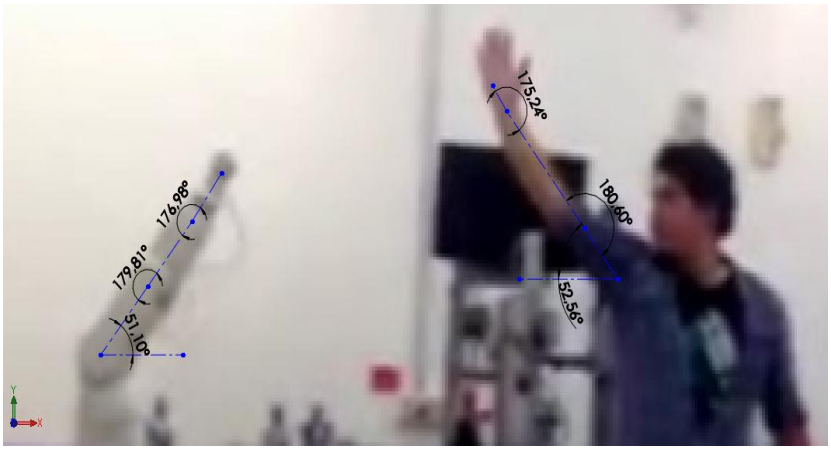

Figura 5.8. Posición 2.

Fuente: El autor.

Figura 5.9. Posición 3.

Fuente: El autor.

Figura 5.10. Posición 4.

Fuente: El autor.

A continuación en la Tabla 5.1 se muestran todos los datos obtenidos del análisis de posición con sus respectivos errores absolutos, cabe mencionar que para obtener estos datos la cámara que capturó las imágenes, siempre se mantuvo en una sola ubicación.

Tabla 5.1. Comparación de Posiciones: Robot vs Operario.

Comparación de Posiciones: Robot vs Operario									
Posición	Brazo		Error	Antebrazo		Error	Mano		Error
1	5,27°	6,87°	1,6°	93,53°	91,66°	1,87°	173,27°	178,22°	4,95°
2	51,10°	52,56°	1,46°	179,81°	180,60°	0,79°	179,98°	180,60°	0,62°
3	6,55°	6,08°	0,47°	171,67°	176,15°	4,48°	175,22°	169,55°	5,67°
4	49,41°	48,93°	0,48°	90,98°	90,91°	0,07°	169,39°	166,33°	3,06°
	Error Promedio		1,00°	Error Promedio		1,80°	Error Promedio		3,57°

Fuente: El autor.

Luego de obtener las posiciones se utilizó la misma secuencia de captura de imágenes para poder determinar el tiempo de respuesta del robot, dando como resultado la Tabla 5.2

Tabla 5.2. Tiempo de Respuesta.

Tiempo de Respuesta				
Posición	1 - 2	2 - 3	3 - 4	Promedio
Tiempo	650 ms	700 ms	680 ms	517 ms

Fuente: El autor.

Con estos resultados se puede determinar que el control no se realiza completamente en tiempo real y esto se debe a la cantidad de iteraciones que debe realizar el software antes de definir los ángulos entre cada articulación.

Pero a pesar del número de iteraciones que realiza el tiempo de respuesta se lo puede considerar aceptable, debido a que durante las pruebas se pudo apreciar que es casi imperceptible al ojo humano en movimientos continuos.

Al igual que la prueba anterior la cámara que grabó la secuencia siempre se mantuvo en una sola posición y para poder obtener los tiempos, se disminuyó la velocidad de reproducción.

6. CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- Después de una extensa investigación, se logró controlar el funcionamiento del robot Mitsubishi RV-2AJ, conociendo sus limitaciones, medidas de seguridad, configuración óptima y lenguaje de programación.

- El software realizado para el control del robot demuestra que es posible manejar el robot Mitsubishi Melfa RV2AJ sin la necesidad de un software de fábrica.

- El sistema implementado demostró que es posible utilizar interfaces diferentes al TEACHBOX proporcionado por Mitsubishi para el control del robot Mitsubishi Melfa RV2AJ

- El software únicamente logró controlar cuatro grados de libertad del robot de los cinco grados de libertad que se aspiraba, el último grado de libertad que tiene el robot denominado “roll” no se pudo controlar debido a las limitaciones del sensor Kinect para calcular rotaciones sobre una extremidad

- Para el desarrollo del sistema de teleoperación se aplicaron muchos conocimientos de ingeniería Mecatrónica en las áreas de diseño, software, y electrónica lo que demuestra que el trabajo debe ser multidisciplinario y está acorde con la definición de mecatrónica en donde se afirma que esta área del conocimiento es multidisciplinaria

- Durante el desarrollo del proyecto se ve la importancia de utilizar herramientas computacionales para realizar programación, pruebas y la posterior validación para comprobar la funcionalidad de lo que se está desarrollando, esto permite ahorrar tiempo y dinero, mejorando así todo el proceso de diseño.

6.2. RECOMENDACIONES

- El robot Melfa RV2AJ debe ser utilizado exclusivamente con fines didácticos y en condiciones absolutamente seguras.

- Mientras el robot Melfa RV-2AJ se encuentra en movimiento es altamente riesgoso mantenerse cerca del área de trabajo, es siempre recomendable avisar a las personas que se encuentran cerca del robot, que tengan la precaución de alejarse hasta una posición segura, para que nadie salga lastimado y el equipo no se estropee.

- Antes de iniciar una sesión de trabajo con el robot es recomendable verificar que todas condiciones normales de funcionamiento estén dadas:
    - Las fuentes de alimentación estén habilitadas y funcionando
    - Las botones de paro de emergencia estén desenclavados
    - Los cables de comunicación estén correctamente conectados a la controladora, al robot y la computadora
    - Los drivers de comunicación PC - robot estén instalados correctamente en la computadora
    - Cuando suene alguna alarma por algún movimiento indebido, se la puede desactivar desde el programa, pulsando el botón de RESET en la aplicación.

- Realizar correctamente el proceso de Puesta en Marcha para que la aplicación no genere ningún tipo de error al momento de comunicarse con el robot.

- Disminuir las fuentes de luz infrarroja innecesarias en el ambiente, estas pueden generar que la aplicación no calcule correctamente los ángulos entre las articulaciones.

GLOSARIO DE TÉRMINOS

AC: Corriente Alterna.

AFRI: Siglas de Asociación Francesa de Robótica Industrial.

CAD: Siglas en ingles de Computer-Aided Design, es el uso de un amplio rango de herramientas computacionales que asisten a ingenieros, arquitectos y diseñadores.

CNC: Siglas de control numérico computarizado, es un sistema de automatización de máquinas herramienta que son operadas mediante comandos programados en un medio de almacenamiento, en comparación con el mando manual mediante volantes o palancas.

CR1-571: Modelo asignado por Mitsubishi a la unidad controladora del robot MELFA RV2AJ.

FPS: Siglas en inglés frames per second, es español cuadros por segundo, es la medida de la frecuencia a la cual un dispositivo reproduce o capta fotogramas.

IDE: Entorno de desarrollo integrado. Programa informático compuesto por un conjunto de herramientas de programación.

IR: Es un tipo de radiación electromagnética y térmica, de mayor longitud de onda que la luz visible, pero menor que la de las microondas.

KUKA: Principales fabricantes mundiales de robots industriales y sistemas de soluciones automatizadas de fabricación**.**

MELFA – BASIC IV: Lenguaje de programación para robots Mitsubishi.

MELFA RV2AJ: Modelo asignado por Mitsubishi al robot manipulador

PLC: Siglas en ingles de Programmable Logic Controller, es una computadora utilizada en la ingeniería automática o automatización

industrial, para automatizar procesos electromecánicos, tales como el control de la maquinaria de la fábrica en líneas de montaje o atracciones mecánicas.

RGB: Siglas en inglés Red, Green, Blue, en español rojo, verde y azul, es la composición del color en términos de la intensidad de los colores primarios de la luz.

RISC: Son las siglas en inglés de, Reduced Instruction Set Computer, que en español es Computador con Conjunto de Instrucciones Reducidas, el cual es un tipo de diseño de CPU generalmente utilizado en microprocesadores o micro controladores

SDK: Siglas en ingles de software development kit, es un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.

TCP/IP: Siglas de Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP).

TEACHBOX: Interfaz de control para robots, la cual envía instrucciones directamente a la unidad de control del robot.

YU : Es un espacio de color típicamente usado como parte de un sistema de procesamiento de imagen en color.

BIBLIOGRAFÍA

- Alvarado, D. (2010). *Teleoperación Háptica de Brazo Manipulador.* Mexico: Universidad Nacional Autónoma De México.
- Barrientos, A. (2007). *Fundamentos de la Robotica.* España: McGraw-Hill.
- Clenet, B. (2012). *Anthropomorphic Robotic Arms: An Overview and Tele-Operation using Kinect Sensor.* Plymouth: University of Plymouth.
- Covarruias, R. F. (2008). *Introduccion a la Inteligencia Artificial.* Mexico: Universidad de Mexico.
- edmsn. (09 de Agosto de 2011). *MSDN España*. Recuperado el 20 de Febrero de 2014, de http://blogs.msdn.com/b/esmsdn/archive/2011/08/09/reto-sdk-de-kinect-detectar-poses-con-skeletal-tracking.aspx
- Freire, A. (2012). *Diseño de una API para el manejo del robot RV2AJ.* QUITO: UPS.
- García, H. L. (2012). *Programación de robots industriales.* UNIVERSIDAD DE OBVIEDO.
- Gentilini, E. F. (2011). *Sistema Teleoperado de un brazo robótico Mitsubishi RV-2AJ.* Costa Rica.
- González, A. (2007). *Telerobótica de Asistencia - Aplicaciones en el Ámbito.* España: Universidad Politécnica de Cataluña.
- Ing. Ana Karen Sedano Flores, D. J. (2012). *Laboratorio Remoto Basado en Web Para el Control de un Brazo.* Mexico: CONGRESO INTERNACIONAL DE INVESTIGACION ISSN 1946-5351 Online.
- Jana, A. (2012). *Kinect for Windows SDK.* Livery Place: Packt Publishing Ltd.
- Nuño, E. B. (2004). *Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente.* Barcelona, España: Universidad De Cataluña.
- Ollero, A. (2001). *Robótica: Manipuladores u robots móviles.* España: Marcombo Alfaomega.

- Padilla, O. (2008.). *Manipulador Teleoperado inalámbricamente.* Mexico: Universidad de las Américas Puebla Escuela de Ingeniería y Ciencias.
- Petr, B. Z. (4 de Mayo de 2014). *tu-chemnitz.* Obtenido de Mecatrónica Modulo Robotica: https://www.tu-chemnitz.de/mb/WerkzMasch/minos/es/download/Webseite/Modul%2010/loesungsbuch.pdf
- Platero, C. (2010). *Apuntes de Visión Artificial.* España.
- Ponce, P. (2010). *INTELIGENCIA ARTICIAL CON APLICACION A LA INGENIERIA.* MEXICO: ALFAOMEGA.
- *Robotics Equipment Corporation*. (4 de Mayo de 2014). Obtenido de http://servicerobotics.eu/en/industrial-image-processing/viscam-extensions/
- Bishop, R. (2008). The Mechatronics Handbook: Mechatronic Systems.
- Boltón, W. (2006). Mecatrónica SISTEMAS DE CONTROL Alfaomega.
- Bolton, W. (2006). Programmable Logic Controllers. Reino Unido: Newnes.
- Braunl, T. (2006). Embedded Robotics. Australia: Springe.
- Bryan, L. A., & Bryan, E. A. (1997). Programmable Controllers: Theory Estados Unidos: Industrial Text Company.
- Buitrago, P. (2010). Gestión de Proyectos. Recuperado el 2013, de http://paobuitrago.edublogs.org/.
- Robotec. (2013). Tecnología Robótica. Recuperado el 2013, de http://robotec11.tripod.com/id4.html.
- ROBOTEC Tecnología robótica. (2012). Clasificación de los Robots. Recuperado el 3 de Enero de 2013, de http://robotec11.tripod.com/id4.html.

- Robots Industriales. (2011). Clasificación del robot Industrial. Recuperado el 04 de Enero de 2013, de http://cfievalladolid2.net/tecno/cyr_01/robotica/industrial.html.

- Robots Industriales. (2011). Definición del robot Industrial. Recuperado el 5 de Enero de 2013, de http://cfievalladolid2.net/tecno/cyr_01/robotica/industrial.htm

- Rocatek. (2011). Que es un PLC (Avanzado). Recuperado el 5 de Enero de 2013, de http://www.rocatek.com/forum_plc2.php

- Roldan, J. (2006). Tecnología Eléctrica Aplicada. España: Thomson.

- Siemens. (2012). LOGO!Soft Comfort. Recuperado el 5 de Enero de 2013, de http://nclem.net/gradrgi/Logo/Logiciel/LogoSC/V1.0/Tools/LOGO Soft%20Comfort/prog/bin/LogoComfort_ES.pdf

- Yasakawa. (2008). Mecatronica. Bishop.

ANEXOS

ANEXO I

Características del robot Mitsubishi Melfa RV2AJ

Características		**Unidad**	**Especificación**
Modelo		-	RV-2AJ
Grados de libertad		-	5
Instalación		-	Sobre el suelo
Estructura		-	Vertical
Sistema guiado		-	Servo motor AC
Método detección de posición		-	J1, J2 y J3: 50W con freno; J4 y J6: 15W sin freno; J5: 15W con freno
Tipo de detector		-	Encoder Absoluto
Longitud del brazo	Hombro	mm	0
	Antebrazo		250
	Brazo		160
	Codo		0
	Muñeca		72
Rango de movimiento	J1	Grados [º]	300 (-150, 150)
	J2		180 (-60, +120)
	J3		230 (-110, +120)
	J4		180 (-90, +90)
	J5		400 (-200, +200)
Velocidad de movimiento	J1	Grados [º]/s	180
	J2		90
	J3		135
	J4		180
	J5		210
Velocidad total máxima		mm/s	Aproximado 2100
Capacidad de carga	Máxima	Kg	2
	Rating		1.5
Repetitividad de posición		mm	±0.02
Peso		Kg	17

Fuente: (Robotics Equipment Corporation, 2014).

ANEXO II

Características de la unidad controladora CR1-571.

Características	**Especificación**
Número de ejes controlables	6
Tipo de procesador	CPU principal: 64 bit RISC
Capacidad de memoria	**No. Programas:** 88 de 5000 líneas máximo cada uno **No. Posiciones:** 2500
Lenguaje de programación	MELFA – BASIC IV
Entradas/Salidas	16/16 pero se puede ampliar a 240/240
Parada de emergencia	1
Conexión RS-232	1 para conexión a PC
Conexión RS-432	1 para Teaching Box
Interfaces de extensión	3
Pinza	4/0 (con opciones)

Fuente: (Robotics Equipment Corporation, 2014).