ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ENERGÍA Y MECÁNICA

CARRERA DE INGENIERÍA MECATRÓNICA

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

AUTORES: TUMBACO MENDOZA DIANA CAROLINA
QUIMBITA ZAPATA WILMER ENRIQUE

TEMA: "DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO DE ROBOT DELTA CON IMPLEMENTACIÓN DE UN CORTADOR LÁSER CNC UTILIZANDO LA PLATAFORMA ROBOTIC OPERATING SYSTEM (ROS) PARA LA ELABORACIÓN DE ARTÍCULOS PUBLICITARIOS"

DIRECTOR: ING. DAVID RIVAS
CODIRECTOR: ING. EDDIE GALARZA

LATACUNGA, DICIEMBRE 2014

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA MECATRÓNICA

CERTIFICADO

ING. DAVID RIVAS (DIRECTOR)
ING. EDDIE GALARZA (CODIRECTOR)

CERTIFICAN:

Que el trabajo titulado **"DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO DE ROBOT DELTA CON IMPLEMENTACIÓN DE UN CORTADOR LÁSER CNC UTILIZANDO LA PLATAFORMA ROBOTIC OPERATING SYSTEM (ROS) PARA LA ELABORACIÓN DE ARTÍCULOS PUBLICITARIOS"**, realizado por: DIANA CAROLINA TUMBACO MENDOZA y WILMER ENRIQUE QUIMBITA ZAPATA ha sido guiado y revisado periódicamente y cumple normas estatuarias establecidas por la ESPE, en el Reglamento de Estudiantes de la Universidad de las Fuerzas Armadas - ESPE.

Debido a que constituye un trabajo con alto contenido científico, que aportará al desarrollo profesional y educativo si aprobamos su publicación.

Latacunga, Diciembre del 2014.

Ing. David Rivas	Ing. Eddie Galarza
DIRECTOR	**CODIRECTOR**

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA MECATRÓNICA

DECLARACIÓN DE RESPONSABILIDAD

DIANA CAROLINA TUMBACO MENDOZA
WILMER ENRIQUE QUIMBITA ZAPATA

DECLARAMOS QUE:

El proyecto de grado denominado **"DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO DE ROBOT DELTA CON IMPLEMENTACIÓN DE UN CORTADOR LÁSER CNC UTILIZANDO LA PLATAFORMA ROBOTIC OPERATING SYSTEM (ROS) PARA LA ELABORACIÓN DE ARTÍCULOS PUBLICITARIOS"**, ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme a las referencias que constan en las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

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

Latacunga, Diciembre del 2014.

Diana Carolina Tumbaco Mendoza.	Wilmer Enrique Quimbita Zapata.
050340026-9	050337317-7

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA MECATRÓNICA

AUTORIZACIÓN

NOSOTROS: DIANA CAROLINA TUMBACO MENDOZA
WILMER ENRIQUE QUIMBITA ZAPATA

Autorizamos a la Universidad de las Fuerzas Armadas - ESPE, la publicación, en la biblioteca virtual de la institución del trabajo denominado **"DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO DE ROBOT DELTA CON IMPLEMENTACIÓN DE UN CORTADOR LÁSER CNC UTILIZANDO LA PLATAFORMA ROBOTIC OPERATING SYSTEM (ROS) PARA LA ELABORACIÓN DE ARTÍCULOS PUBLICITARIOS"**, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Latacunga, Diciembre del 2014.

Diana Carolina Tumbaco Mendoza.	Wilmer Enrique Quimbita Zapata.
050340026-9	050337317-7

DEDICATORIA

El presente trabajo fruto de mi esfuerzo y dedicación se lo dedico:

A mis Padres Luis y Rosa, las personas más importantes de mi vida, por el gran esfuerzo que paralelamente conmigo han hecho.

Diana.

Dedico este trabajo a cada ser vivo que día a día lucha por salir adelante, en especial a mis padres Hugo y Aida quienes fueron un pilar importante en mi vida, también dedico a aquellas personas que partieron de mi lado, pero quedaron recuerdos grabados en mi corazón Rosa, Manuel, Delia y como no a mi ñaño Santiago, por todo ese cariño y amor les dedico este trabajo.

Wilmer

AGRADECIMIENTO

A DIOS Y A LA VIRGEN SANTISIMA

Por concederme bendiciones y permitirme concluir un ciclo más en mi vida.

"Dios perdona a quien obedece a su propia conciencia".

A MIS PADRES

Por su apoyo y consejo y porque sin ustedes no sería lo que soy.

A MIS HERMANOS

Por estar en mi vida y ser parte de este esfuerzo. Y como muestra de que las metas que uno se fija se pueden lograr.

A MIS TUTORES

Por ser piezas fundamentales en todo el desarrollo y culminación del proyecto, pero sobre todo por ser unas grandes personas.

A JOHN RAMBO (Mimes)

Por haber compartido conmigo innumerables afinidades y diferencias durante nuestra vida universitaria, por su trabajo y dedicación para culminar el presente proyecto y sobre todo por ser una gran persona

A MIS AMIGOS (SPSC-2)

Si a ellos con quienes he compartido laboratorios e incontables horas de trabajo y buenos ratos, lo cual no tiene precio, gracias por los buenos y malos momentos, eso sí más buenos que malos.

¿Quién puede pedir más? Nos hemos dado ánimos por el camino, y eso siempre ayuda.

Diana

Agradezco a Dios y la virgen por permitirme culminar una etapa más en mi vida junto a las personas que más quiero.

A mis padres Hugo y Aida por sus consejos y apoyo constante tanto moral como económicamente.

A mis queridas hermanas Naty y Lucy por ser las mejores hermanas, gracias por aquellos momentos inolvidables.

A mis tutores y docentes de la ESPEL por compartir sus conocimientos y permitir culminar la vida universitaria.

A Caroline (Carolis) por brindarme su amistad sincera y compartir momentos tantos de alegría como tristeza, gracias por escucharme y estar a mi lado cuando lo necesite.

A mis primos en especial a Chaval y Lili por ser como mis hermanos del alma.

A mis amigos Mary, Pame, Nancy, Ivone, Gina, Cristian y Ney por sus ocurrencias de cada día que han sacado de mí una sonrisa.

Finalmente agradezco a la S.O. por sentir esa adrenalina y pasión en cada encuentro.

Wilmer

ÍNDICE DE CONTENIDOS

PORTADA i
CERTIFICADO ii
DECLARACIÓN DE RESPONSABILIDAD iii
AUTORIZACIÓN iv
DEDICATORIA v
AGRADECIMIENTO vi
ÍNDICE DE CONTENIDOS viii
ÍNDICE DE TABLAS xii
ÍNDICE DE FIGURAS xiii
RESUMEN xvii
ABSTRACT xviii
PRESENTACIÓN xix
CAPÍTULO I 1
FUNDAMENTOS TEÓRICOS 1
1.1. OBJETIVOS 1
1.1.1. OBJETIVO GENERAL 1
1.1.2. OBJETIVOS ESPECÍFICOS 1
1.2. LA ROBÓTICA 2
1.2.1. INTRODUCCIÓN 2
1.2.2. DEFINICIÓN DE ROBOT. 3
1.2.3. ESTRUCTURA BÁSICA DE UN ROBOT 3
1.2.4. CONFIGURACIONES DE LOS ROBOTS 5
1.3. ROBOTS PARALELOS 5
1.3.1. INTRODUCCIÓN 5
1.3.2. ORÍGENES DE LOS ROBOTS PARALELOS 6
1.3.3. DEFINICIÓN DE ROBOT PARALELO 8
1.3.4. CLASIFICACIÓN DE LOS ROBOTS PARALELOS 9
1.4. ROBOT DELTA 12
1.5. CINEMÁTICA DEL ROBOT DELTA 13
1.5.1. CINEMÁTICA INVERSA 14

1.5.2. CINEMÁTICA DIRECTA 18
1.6. PLATAFORMAS DE DESARROLLO 20
1.6.1. FRAMEWORK DE DESARROLLO EN ROBÓTICA O MIDDLEWARE PARA ROBOTS 20
1.6.2. RSF (Robotic Software Framework) 22
1.7. ROS (Robotic Operating System) 25
1.7.2. ARQUITECTURA DE ROS 26
1.7.3. COMPONENTES DE ROS 30
1.8. ACTUADORES DE ALTO DESEMPEÑO 32
1.8.1. ACTUADORES DYNAMIXEL 34
1.8.2. USB2Dynamixel 36
1.9. LÁSER 37
1.9.1. CARACTERÍSTICAS DE LA LUZ LÁSER. 37
1.9.2. ELEMENTOS DE UN LÁSER. 38
1.9.3. DIODO LÁSER 40
1.10. ARTÍCULOS PUBLICITARIOS 42
1.10.1. GRABACIÓN POR LÁSER, CORTE POR LÁSER 42
CAPÍTULO II 44
DISEÑO Y CONSTRUCCIÓN DEL SISTEMA MECÁNICO 44
2.1. PARAMETROS DE DISEÑO 44
2.1.1. MATERIAL 44
2.1.2. DIMENSIONES 46
2.2. DISEÑO DEL SISTEMA MECÁNICO 47
2.2.1. DISEÑO DE LA ESTRUCTURA DEL ROBOT DELTA 47
2.2.2. ANÁLISIS ESTÁTICO 51
2.2.3. RESULTADOS OBTENIDOS 54
2.3. MONTAJE DEL SISTEMA MECÁNICO 56
2.3.1. COMPONENTES DEL PROTOTIPO 56
CAPÍTULO III 64
IMPLEMENTACIÓN Y DESARROLLO DEL SISTEMA DE CONTROL 64
3.1. TECNOLOGÍAS EMPLEADAS 64
3.1.1. SISTEMA OPERATIVO 64

3.1.2. PLATAFORMA DE DESARROLLO 64
3.1.3. BIBLIOTECA 65
3.1.4. LENGUAJES DE PROGRAMACIÓN 65
3.1.5. HERRAMIENTAS UTILIZADAS 65
3.2. INSTALACIÓN DEL SOFTWARE DE CONTROL 67
3.3. PREPARACIÓN DEL ENTORNO DE TRABAJO 67
3.3.1. CREACIÓN Y USO DEL ESPACIO DE TRABAJO ROS 67
3.4. CREACIÓN DE PAQUETES ROS 68
3.5. DISEÑO DE LA APLICACIÓN 69
3.5.1. SIMULACIÓN EN ROS 69
3.5.2. SOFTWARE PARA EL PROCESAMIENTO DE IMÁGENES 70
3.5.3. CREACIÓN DE LA APLICACIÓN. 74
3.5.4. ROBOT MODO PASIVO 75
3.6. INTEGRACIÓN ROS CON ACTUADORES DYNAMIXEL 75
3.6.1. CONFIGURACIÓN DE LOS ACTUADORES DYNAMIXEL Y ACCESO AL BUS 75
3.6.2. INSTRUCCIONES PARA CAMBIAR EL ID DE LOS SERVOS DYNAMIXEL. 76
3.6.4. ARCHIVO DE INICIO DE CONTROLADORES (MODO_DELTA_DIBUJADOR.LAUNCH & MODO_DELTA_PASIVO.LAUNCH). 81
3.7. IMPLEMENTACIÓN ELECTRÓNICA DEL SISTEMA 82
3.7.1. IMPLEMENTACIÓN LÁSER DE DIODO CNC 445NM 1W 83
3.7.2. Especificaciones 83
3.7.3. ENFOQUE DEL LÁSER 84
3.7.4. CONEXIÓN DEL MÓDULO AL LÁSER. 84
3.7.5. INTERACCIÓN ENTRE ARDUINO Y PYTHON 84
3.7.6. DIAGRAMA DEL SISTEMA DE CONEXIONES 86
CAPÍTULO IV 87
PRUEBAS Y RESULTADOS EXPERIMENTALES 87
4.1. PRUEBAS Y RESULTADOS DEL FUNCIONAMIENTO. 87
4.1.1. EJECUCIÓN GENERAL DE LA PLATAFORMA 87
4.2. PRUEBAS DE CORTE Y GRABADO EN DIFERENTES MATERIALES. 89

4.2.1. PROCESADO DE LA IMAGEN MODO VECTORIZADO. 89
4.2.2. PROCESADO DE LA IMAGEN MODO RASTERIZADO. 95
4.3. HERRAMIENTA RQT DE ROS 98
CAPÍTULO V 101
CONCLUSIONES Y RECOMENDACIONES 101
5.1. CONCLUSIONES 101
5.2. RECOMENDACIONES 103
REFERENCIAS BIBLIOGRAFICAS 104
ANEXOS 10909

ÍNDICE DE TABLAS

Tabla 1.1 Diferentes distribuciones de ROS. 31
Tabla 1.2 Ventajas y Desventajas de ROS 31
Tabla 1.3 Servomotor de radio control y servomotor Dynamixel. 33
Tabla 1.4 Características de los tipos de láser 38
Tabla 1.5 Longitud de onda del láser según el color 41
Tabla 2.1 Propiedades de los materiales 46
Tabla 2.2 Dimensiones del robot delta 46
Tabla 2.3 Material de los elementos del robot delta 52
Tabla 2.4 Componentes Construidos del Robot Delta 57
Tabla 2.5 Componentes necesarios Dynamixel 58
Tabla 2.6 Piezas Varias 58
Tabla 3.1 Características técnicas del láser 445nm 1W. 83
Tabla 4.1 Resultados de pruebas de corte del láser 91
Tabla 4.2 Resultados velocidad y tiempo de corte del cuadrado. 92
Tabla 4.3 Resultados velocidad y tiempo de Impresión de imágenes vectorizada. 93
Tabla 4.4 Impresión de diferente figuras 94
Tabla 4.5 Grabados en foamy 96
Tabla 4.6 Grabados en madera 97

ÍNDICE DE FIGURAS

Figura 1.1 Elementos estructurales de un robot 4
Figura 1.2 Tipos de articulaciones de un robot 4
Figura 1.3 Configuraciones más frecuente de los robots 5
Figura 1.4 (a) Robot tipo serie. (b) Robot o manipulador tipo paralelo..... 6
Figura 1.5 Posible primer mecanismo espacial paralelo patentado por J. E Gwinnett 6
Figura 1.6 Primer Robot Industrial Paralelo 7
Figura 1.7 Plataforma de Gough 7
Figura 1.8 Plataforma de Stewart 8
Figura 1.9 Simulador de Movimiento de Klaus Cappel. 8
Figura 1.10 Diseño Mecanismo de 5 barras 9
Figura 1.11 Robot Paralelo Planar 3RRR. 9
Figura 1.12 Plataforma Stewart..... 10
Figura 1.13 Robot paralelo Delta (Clavel, 1991) de 3 GDL traslacional. 10
Figura 1.14 Robot paralelo Orthoglide. 11
Figura 1.15 Robot Adept Quattro (Adept Technology, 2011) 11
Figura 1.16 Robot paralelo 6GDL, basado en mecanismo de 5 barras 12
Figura 1.17 Esquema del robot Delta (US Patent N º 4.976.582) 13
Figura 1.18 Parámetros Geométricos Robot Delta. 14
Figura 1.19 Articulación F1J1. 14
Figura 1.20 Vista lateral para análisis geométrico. 15
Figura 1.21 Rotación del sistema de referencia. 17
Figura 1.22 Nomenclatura de coordenadas $J1, J2, J3$ 18
Figura 1.23 Esferas..... 18
Figura 1.24 Cinemática directa. 19
Figura 1.25 Componentes Modulares. 20
Figura 1.26 BEAR Robot: Robot usado para búsqueda y rescate..... 21
Figura 1.27 Robot que asistirá a personas de la tercera edad..... 21
Figura 1.28 Robot quirurgico Da Vinci. 21
Figura 1.29 Logotipo de ROS. 25
Figura 1.30 Métodos de Comunicación entre Nodos. 27

Figura 1.31 Estructura basica de un paquete. 28
Figura 1.32 Estructura básica de un meta-paquete 29
Figura 1.33. Sistema de conexión en red serial para los actuadores del robot. 33
Figura 1.34 Actuadores Dynamixel AX-12A. 34
Figura 1.35. Aplicaciones Dynamixel. 34
Figura 1.36. Dimensiones del actuador Dynamixel AX-18A. 35
Figura 1.37 Motor Pin. 36
Figura 1.38. Conexión actuadores AX. 36
Figura 1.39 Control del USB2Dynamixel usando una PC. 37
Figura 1.40 Comparación entre la luz normal y la luz láser. 38
Figura 1.41 Elementos del diodo láser. 40
Figura 1.42 Estructura del encapsulado del dispositivo láser. 42
Figura 1.43 Grabado Láser. 43
Figura 2.1 Estructura Principal 45
Figura 2.2 Estructura Secundaria o Base 45
Figura 2.3 Dimensiones del robot delta 47
Figura 2.4 Plataforma Fija 48
Figura 2.5 Frame F3 48
Figura 2.6 Brazo 48
Figura 2.7 Plataforma Móvil 49
Figura 2.8 Junta esférica o rótula 49
Figura 2.9 Barra 49
Figura 2.10 Antebrazo 50
Figura 2.11 Pata 50
Figura 2.12 Tensor 50
Figura 2.13 Conector Delta 51
Figura 2.14 Prototipo de robot delta Ensamblado 51
Figura 2.15 Soporte Fijo del Robot Delta 53
Figura 2.16 Fuerza en la plataforma móvil 53
Figura 2.17 Robot Delta Mallado 54
Figura 2.18 Desplazamientos 54
Figura 2.19 Tensiones De Von Mises Del Robot Delta 55

Figura 2.20 Factor de Seguridad Robot Delta..... 55
Figura 2.21 Piezas Robot Delta: Plataforma Móvil y Pata 56
Figura 2.22 Piezas Robot Delta: Conector Delta Tensor 56
Figura 2.23 Piezas Robot Delta: Juntas Esféricas, Plataforma móvil Barra y Brazos..... 57
Figura 2.24 Conector Delta con Tensores..... 59
Figura 2.25 Colocación de las patas..... 59
Figura 2.26 Colocación de la plataforma fija..... 60
Figura 2.27 Ensamble Rotulas-Barras..... 60
Figura 2.28 Ensamble Brazo-Antebrazo 61
Figura 2.29 Ensamble Antebrazo-Plataforma móvil..... 61
Figura 2.30 Elementos Incluido en los Servomotores Dynamixel AX-12A..... 61
Figura 2.31 Disposición de tuercas en los servomotores ID3 ID6 y ID9 62
Figura 2.32 Ensamble Servomotor-Perno-Frame F3 62
Figura 2.33 Ensamble servomotor-brazo 63
Figura 2.34 Ensamble final del robot Delta 63
Figura 3.1 Pantalla del visor Rviz. 64
Figura 3.2 Intérprete de comandos Terminator. 66
Figura 3.3 Entorno de programación Gedit. 67
Figura 3.4 Salida gráfica del robot. 69
Figura 3.5. Slider de control. 70
Figura 3.6 a) Imagen Original. b) Escala de 120 pixeles y gamma de 2,2 (220/100). c) Imagen impresa por el robot. 71
Figura 3.7 Imagen rasterizada. a) Imagen más oscura. b) Imagen más brillante. 71
Figura 3.8. Imagen Vectorizada. 73
Figura 3.9 Proceso de la aplicación modo dibujo. 74
Figura 3.10 Proceso Aplicación modo pasivo. 75
Figura 3.11 Escaneo de Ids de los servos Dynamixel..... 76
Figura 3.12 Instrucciones para cambios de Id. 77
Figura 3.13 Parámetros de funcionamiento de los dynamixel con ROS..... 78
Figura 3.14 Comunicación de ROS con los servos dynamixel. 78
Figura 3.15 Parámetros con torque activado para modo_delta_dibujador.launch. ... 79

Figura 3.16. Parámetros de los servomotores con torque desactivado para modo_delta_pasivo.launch. 80
Figura 3.17 Archivo de inicio para modo_delta_dibujador.launch. 81
Figura 3.18 Archivo de inicio para modo_delta_pasivo.launch. 82
Figura 3.19 Esquema general del sistema. 82
Figura 3.20 Láser de diodo CNC 445nm 1W 83
Figura 3.21 Enfoque del lente del láser. 84
Figura 3.22 Modulo láser. 84
Figura 3.23 IDE de Arduino 85
Figura 4.1 Visualización de los movimientos del robot real en la simulación. 87
Figura 4.2 Movimiento de la trayectoria, realizado por el robot delta en Rviz. 88
Figura 4.3 Dibujando un cuadrado. 89
Figura 4.4 Cuadrado usando laser. 90
Figura 4.5 Enfocando el láser para el corte. 90
Figura 4.6 Impresión de un círculo a) Con lápiz b) Con láser 93
Figura 4.7 Procedimiento de grabado (Primeras pruebas). 95
Figura 4.8 Procesos ROS en ejecución. 98
Figura 4.9 Grafo de un solo nodo. 98
Figura 4.10 Grafos del sistema con sus topics. 99
Figura 4.11 Identificación del Grafo de un solo nodo. 99
Figura 4.12 Identificación de los Grafos del sistema con sus topics. 100
Figura 4.13 Tema publicados en el sistema. 100

RESUMEN

Los productos derivados de la mecatrónica son óptimos y comercialmente competitivos y la mayoría de las aplicaciones se refiere a la generación de modelos y prototipos robóticos didácticos, el campo de la robótica es muy usado para la enseñanza de la misma. Un robot didáctico, al igual que un industrial, es diseñado usando las leyes de la mecánica, y se usa la electrónica para controlar los movimientos por medio de los actuadores y finalmente, la computación para simular los movimientos en un ambiente virtual. Dentro del campo de la robótica, los prototipos de robots paralelos son muy ilustrativos y motivantes en la enseñanza de la Mecatrónica, pues su estructura mecánica está conformada por cadenas cinemáticas cerradas y los actuadores están colocados en cada cadena cinemática. El presente proyecto es el diseño y construcción de un prototipo de robot paralelo tipo delta, y como efector final un cortador láser capaz de grabar anuncios publicitarios en materiales suaves utilizando software libre. La estructura mecánica puede moverse en un espacio de trabajo limitado por las dimensiones de los eslabones del robot. El robot delta consta de 3 actuadores dispersos uno del otro cada 120 grados, cada actuador transmite el movimiento a cada brazo y estos trasmiten el movimiento a los antebrazos conectados a una plataforma móvil. El control de los movimientos de cada motor se lo realiza por computador utilizando, los actuadores inteligentes Dynamixel conectados en red TTL con IDs diferentes. El usuario debe ejecutar el código programado para la realización de la aplicación, se usara una PC con Ubuntu y Middleware ROS.

Palabras claves: MECATRÓNICA, ROBOT DELTA, SISTEMA OPERATIVO DE ROBOTS, UBUNTU, SERVOMOTORES DYNAMIXEL, PROGRAMA MIDDLEWARE.

ABSTRACT

The mechatronics products are optimized and commercially competitive and most applications relates to the generation of models and teaching robotic prototypes, the field of robotics is widely used for teaching it. An educational robot, as an industrialist, is designed using the laws of mechanics, and electronics are used to control the movements of the actuators and finally the computer to simulate the movements in a virtual environment. Within the field of robotics, the prototype parallel robots are very illustrative and motivating teaching mechatronics, because its mechanical structure consists of closed kinematic chains and actuators are placed at each powertrain. This project is the design and construction of a prototype parallel type delta robot and end effector as a laser cutter capable of recording commercials in soft materials using free software. The mechanical structure is movable in a working space limited by the dimensions of the links of the robot. Delta robot actuators comprises 3 dispersed each other every 120 degrees, each actuator transmits the movement to each arm and these transmit the movement to the forearms connected to a mobile platform. Controlling the movements of each motor is done by using computer, smart actuators Dynamixel TTL connected network with different IDs. The user must run the set for the completion of the implementation code, you used a PC with Ubuntu and ROS Middleware.

Keywords: MECHATRONICS, ROBOT DELTA, ROBOTIC OPERATING SYSTEM, UBUNTU, SERVOMOTOR DYNAMIXEL, MIDDLEWARE.

PRESENTACIÓN

En el presente proyecto se realiza el diseño y construcción de un prototipo de robot delta con implementación de un cortador láser CNC utilizando la plataforma robotic operating system (ROS) para la elaboración de artículos publicitarios.

En el Capítulo I se encuentra información referente a robots paralelos tipo delta, software ROS e información necesaria para el desarrollo del proyecto.

En el Capítulo II se detalla el diseño del sistema mecánico, con sus respectivos componentes y ensamble; donde se determinan algunos parámetros para su diseño y su construcción.

En el Capítulo III se procede con la implementación de ROS en el sistema y control del láser.

En el Capítulo IV se realizan las pruebas y análisis del funcionamiento, además se muestran los alcances, limitaciones y la factibilidad del proyecto.

En el Capítulo V se muestran las conclusiones y recomendaciones del proyecto.

Se incluye anexos y referencias bibliográficas para profundizar en el tema.

CAPÍTULO I

FUNDAMENTOS TEÓRICOS

1.1. OBJETIVOS

1.1.1. OBJETIVO GENERAL

Diseñar y construir un prototipo de robot delta con implementación de un cortador láser CNC utilizando la plataforma Robotic Operating System (ROS) para la elaboración de artículos publicitarios.

1.1.2 OBJETIVOS ESPECÍFICOS

- Diseñar la estructura, y arquitectura del robot delta utilizando un software CAD.
- Construir y ensamblar el robot de estructura paralela tipo delta.
- Analizar la geometría del robot para obtener un modelo que describa la cinemática directa e inversa y controlar el robot delta.
- Determinar las características y la funcionalidad de ROS (Robotic Operating System) para incorporar en el robot delta.
- Desarrollar un sistema de control utilizando ROS, el cual permita el control de los servos inteligentes Dynamixel y visualizar los movimientos del robot delta.
- Diseñar un simulador en ROS que permita ver los movimientos del robot delta en tiempo real.
- Diseñar un programa donde el robot delta dibuje una imagen seleccionada por el usuario.
- Implementar el cortador diodo láser CNC de baja potencia en la estructura mecánica.
- Verificar si la implementación láser logra cortar o grabar en materiales suaves para usarlos como artículos publicitarios.

1.2. LA ROBÓTICA

1.2.1. INTRODUCCIÓN

(Baturone, 2001) El término robot aparece en 1921, por Karel Capek en su obra teatral R.U.R. (Rossum's Universal Robot): Robota es una palabra checa que significa fuerza del trabajo o servidumbre. La introducción de la palabra robot a obras teatrales y películas, toma gran interés en las fábricas y empiezan a discutir el poder de las máquinas. Pronto se aplicaría a autómatas construidos en los años veinte y treinta, que trate de imitar movimientos de los seres vivos.

Los robots industriales, surgen de tecnologías del control automático. Mediante el control automático se pretende realizar procesos sin la presencia de agentes exteriores. En particular se presentan problemas en las señales de control tanto en los servosistemas, y los reguladores. El principio para mantener el control automático se basa en la realimentación. Las señales de consigna o referencia se comparan con las medidas de las variables del proceso para generar acciones de control.

La automatización industrial comienza también en el siglo XIX pero no hasta el siglo XX, después de la segunda guerra mundial. Se generalizan los sistemas automáticos en procesos industriales, en particular en control de posición y velocidad. Se emplea en barcos y aviones para seguir una determinada trayectoria (pilotos automáticos).

En la realización de sistemas de control automático se han empleado tecnologías como neumática, hidráulica, y eléctrica. En 1972 la aparición del microprocesador impulsa al control por computador, aplicando principalmente al control de robots. Los avances de la microelectrónica acentúan esta tendencia.

Las máquinas herramientas de control numérico se desarrollaron en EEUU en los años cuarenta y cincuenta. Se combinada el diseño de servosistemas con técnicas de computación digital. En 1953 "Massachussetts Institute of Technology" (MIT) presenta una máquina de estas características.

Los teleoperadores se desarrollaron en los años cuarenta para manejar materiales radioactivos. El primer teloperador accionado por servomecanismos eléctricos se presentó en 1947, en 1948 se introduce servosistemas con realimentación. En 1954 el Ingeniero George Devol patentó el considerado primer robot industrial.

1.2.2. DEFINICIÓN DE ROBOT.

(D'Inca, 2010) Existen varias definiciones de lo que es un robot, pero la que debería ser definitiva es la dada por la ISO (International Standards Organizatión) que dice:

> "Un robot industrial es una máquina manipulativa, automáticamente controlada, reprogramable, multi-propósito, con varios ejes reprogramables, los cuales pueden ser fijos o móviles, para uso en aplicaciones de automatización industrial".

Para los robots no industriales RIA (Robotics Industries Association o Asociación de Industrias de Robótica) prefiere la siguiente definición:

> "Un robot es una máquina multifuncional, reprogramable, diseñada para manipular materiales, partes, herramientas, o dispositivos especiales, mediante movimientos variables, programados para la realización de una variedad de tareas".

1.2.3. ESTRUCTURA BÁSICA DE UN ROBOT

(Barrientos, Peñin, Balaguer, & Aracil, 2007) Un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. La mayoría de los robots deben su estructura a imitaciones de elementos de la naturaleza. Para hacer referencia a los distintos elementos que componen el robot, se usan términos como cuerpo, brazo, codo y muñeca como se muestra en la figura 1.1

Figura 1.1 Elementos estructurales de un robot

Fuente: http://afroner9.blogspot.com/2012/09/robotica-y-bionica.html

El movimiento de cada articulación puede ser de desplazamiento, de giro, o de una combinación de ambos. De este modo son posibles, diferentes tipos de articulaciones, que se muestran en la figura 1.2

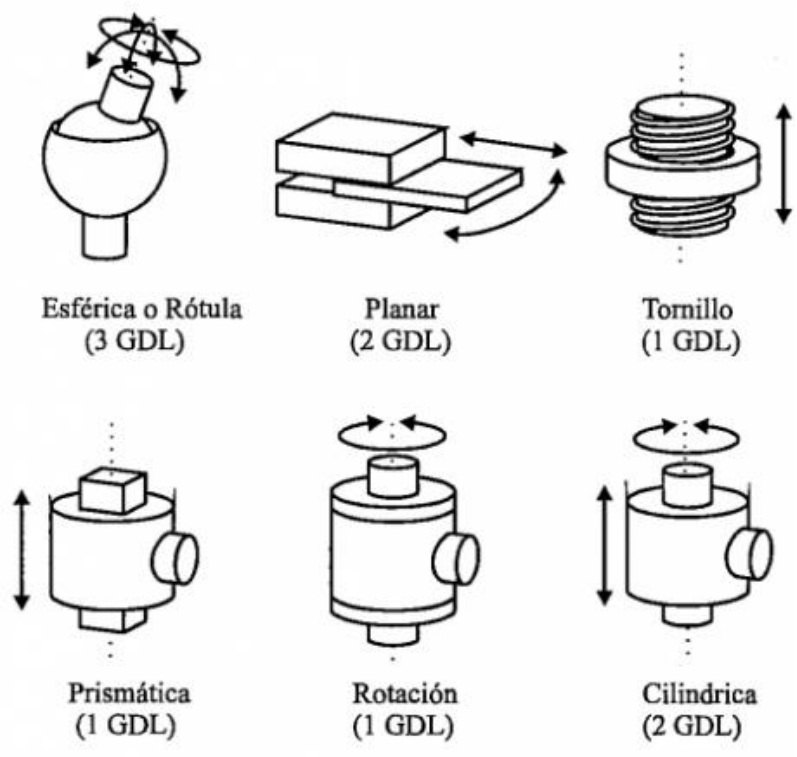


Figura 1.2 Tipos de articulaciones de un robot

Fuente: Barrientos, A., Peñin, L., Balaguer, C., & Aracil, R. (2007). Fundamentos de Robotica. Madrid: McGraw-Hill/Interamericana.

1.2.4. CONFIGURACIONES DE LOS ROBOTS

El empleo de diferentes combinaciones de articulaciones en un robot, da lugar a diferentes configuraciones con características a tener en cuenta tanto en el diseño y construcción del robot como en su aplicación. Las combinaciones más frecuentes son las representadas en la figura 1.3 donde se atiende únicamente a las tres principales articulaciones del robot, que son las más importantes a la hora de posicionar su extremo en un punto del espacio.

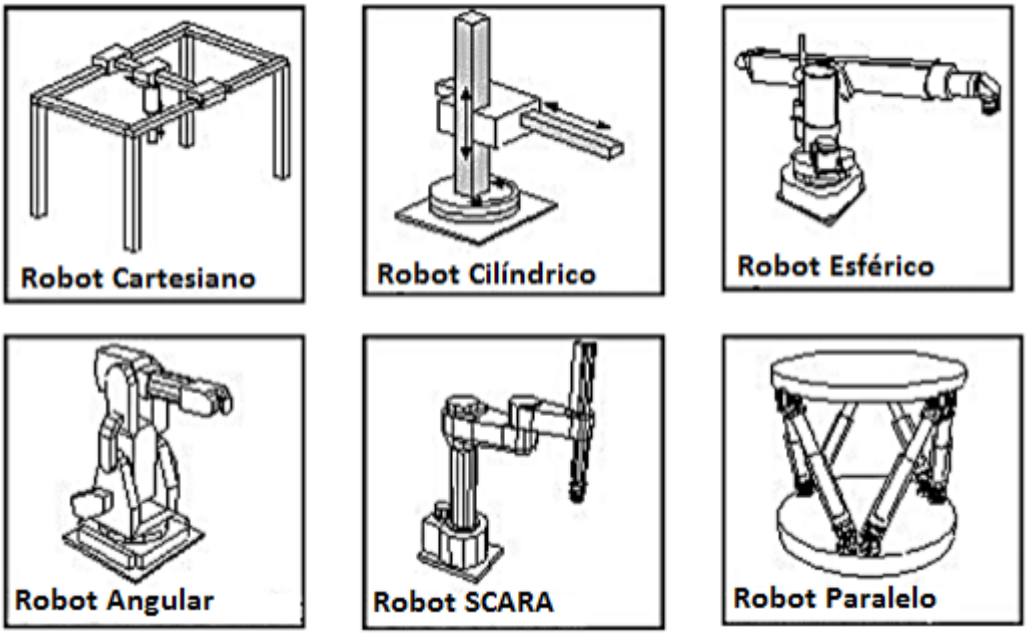


Figura 1.3 Configuraciones más frecuente de los robots

Fuente: http://repositorio.espe.edu.ec/bitstream/21000/5924/1/T-ESPEL-0940.pdf

1.3. ROBOTS PARALELOS

1.3.1. INTRODUCCIÓN

(Zabalza & Ros, 2007) Teniendo en cuenta su estructura, los robots se pueden clasificar en: Robots tipo serie y robots o manipuladores paralelos. Los robots tipo serie están formados por una cadena cinemática abierta, Figura. 1.4(a). En cambio, los robots paralelos están formados por dos plataformas, una fija y otra móvil, unidas por varias cadenas cinemáticas en paralelo, y por ello, formando cadenas cinemáticas cerradas, Figura. 1.4(b).

(a) (b)

Figura 1.4 (a) Robot tipo serie. (b) Robot o manipulador tipo paralelo.

Fuente: http://www.imem.unavarra.es/isidro/articles/Zabalza-Cuzco.pdf

Los robots paralelos frente a los robots tipo serie: Tienen como inconveniente, un menor espacio de trabajo. Como ventajas, la relación masa del robot frente la carga a soportar es mucho menor por lo que admiten mayores aceleraciones durante su movimiento, y por ello, mayores velocidades.

1.3.2. ORÍGENES DE LOS ROBOTS PARALELOS

(Aracil, 2006) El primer mecanismo paralelo se trataba de una plataforma de movimiento destinada a la industria del entretenimiento, Figura 1.5 esta plataforma fue patentada en 1931 (US Patent N° 1,789,680) por James E. Gwinnett. Este mecanismo nunca fue diseñado.

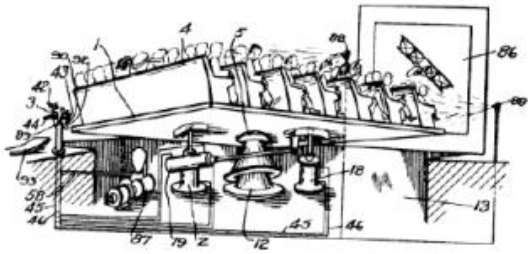


Figura 1.5 Posible primer mecanismo espacial paralelo patentado por J. E Gwinnett

Fuente: http://arvc.umh.es/documentos/articulos/RIAI%202006.pdf

En 1940 Willard L.V. Pollard presenta un robot paralelo (US Patent N° 2286571) con 5 grados de libertad destinado a operaciones de pintura con spray como se ve en la, Figura 1.6.

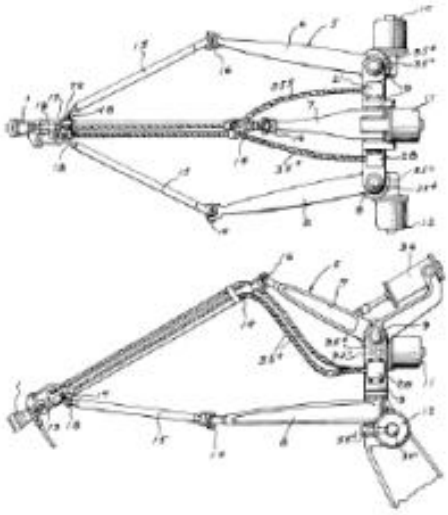


Figura 1.6 Primer Robot Industrial Paralelo

Fuente: http://arvc.umh.es/documentos/articulos/RIAI%202006.pdf

En 1947 el Dr. Eric Gough diseñó un octaedro hexápodo con lados de longitud variable, Figura 1.7, de esta manera simulaba el comportamiento de los neumáticos en el proceso de aterrizaje de un avión en Dunlop donde trabajaba.

Figura 1.7 Plataforma de Gough

Fuente: http://arvc.umh.es/documentos/articulos/RIAI%202006.pdf

Mr. Stewart en 1965 presentó un artículo que describía una plataforma de movimiento de 6 GDL, destinada a trabajar como simulador de vuelo, Figura 1.8, que es diferente al presentado por Gough.

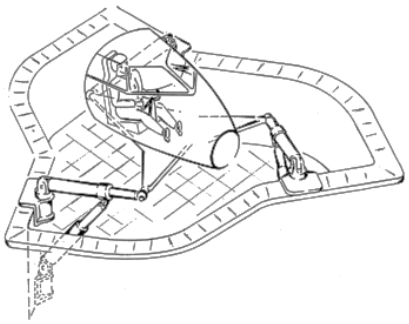


Figura 1.8 Plataforma de Stewart

Fuente: http://arvc.umh.es/documentos/articulos/RIAI%202006.pdf

Paralelamente, el ingeniero Klaus Cappel realizaba investigaciones con plataformas paralelas con 6 GDL, en 1967 patentaba un simulador de movimiento basado a un hexápodo, Figura 1.9

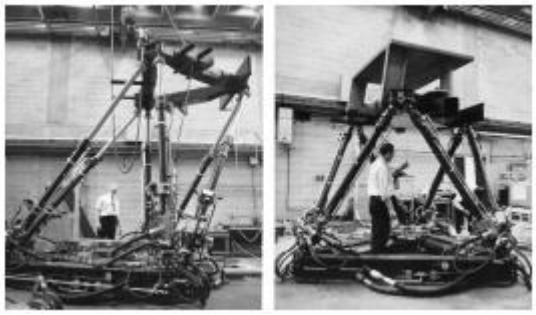


Figura 1.9 Simulador de Movimiento de Klaus Cappel.

Fuente: http://arvc.umh.es/documentos/articulos/RIAI%202006.pdf

La robótica paralela actualmente tiene mayor avance que épocas anteriores, ya que actualmente la capacidad de cómputo, de nuevos procesadores para resolver procesos numéricos iterativos, ha facilitado el trabajo para desarrollar los robots paralelos en diferentes aplicaciones.

1.3.3. DEFINICIÓN DE ROBOT PARALELO

"Un robot paralelo generalizado es un mecanismo de cadenas cinemáticas en lazo cerrado, cuyo efector final es unido a la base por varias cadenas cinemáticas independientes (Merlet, 2006)."

1.3.4. CLASIFICACIÓN DE LOS ROBOTS PARALELOS

(León, 2009) Los robots paralelos pueden ser clasificados de acuerdo a su movilidad, de acuerdo a esto pueden ser planares o espaciales.

A. **Robot Paralelos Planares.-** El mecanismo más simple es el de 5 barras, con 2 GDL y un solo lazo cerrado. Formado por 4 eslabones y una base unidos mediante 5 articulaciones rotacionales (5R) ver figura 1.10

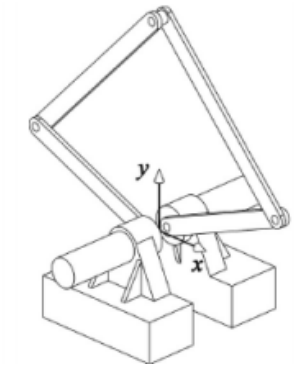


Figura 1.10 Diseño Mecanismo de 5 barras

Fuente: http://itzamna.bnct.ipn.mx/dspace/bitstream/123456789/4900/1/JABL.PDF

Otro tipo de robot paralelo planar es el 3RRR, figura 1.11, está formado por 3 brazos con 3 articulaciones. Este robot tiene 3 GDL, dos son traslaciones y una rotación.

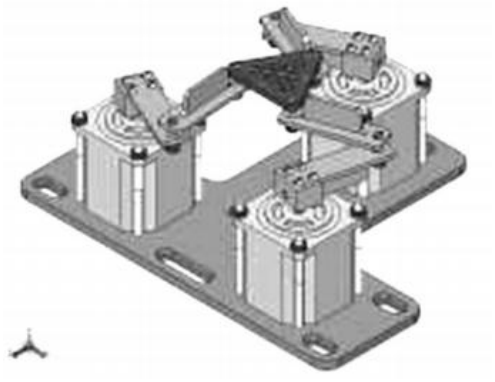


Figura 1.11 Robot Paralelo Planar 3RRR.

Fuente: http://itzamna.bnct.ipn.mx/dspace/bitstream/123456789/4900/1/JABL.PDF

B. Robots Paralelos Espaciales.- Un ejemplo típico es la "Plataforma Stewart" diseñada originalmente por (Stewart, 1965), figura 1.12 Inicialmente la plataforma fue pensada con 6 GDL, como simulador de vuelo. Actualmente

variantes de estos mecanismos son aplicados como máquinas herramienta, dispositivos apuntadores, etc.

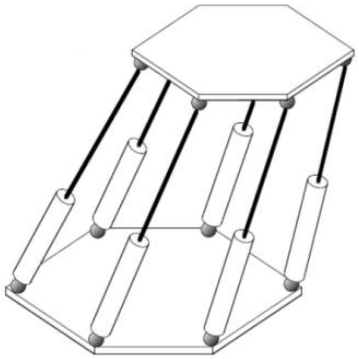


Figura 1.12 Plataforma Stewart

Fuente: http://itzamna.bnct.ipn.mx/dspace/bitstream/123456789/4900/1/JABL.PDF

Otro tipo de robot paralelo es el Diseñado por Clavel, el robot Delta es un manipulador paralelo traslacional, contiene articulaciones esféricas en los paralelogramos que lo conforman figura 1.13.

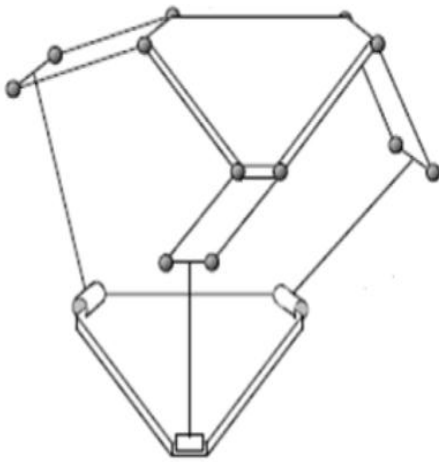


Figura 1.13 Robot paralelo Delta (Clavel, 1991) de 3 GDL traslacional.

Fuente: http://itzamna.bnct.ipn.mx/dspace/bitstream/123456789/4900/1/JABL.PDF

Existen varios mecanismos que son variantes o combinaciones de los mecanismos ya mencionados, que buscan optimizar el espacio de trabajo, la rigidez, las singularidades o simplemente facilitar su análisis.

El Orthoglide,(Chablat & Wenger,2003), es un manipulador traslacional desarrollado por Tsai, donde el espacio de trabajo es optimizado mediante una ubicación diferente de los brazos y actuadores, figura 1.14.

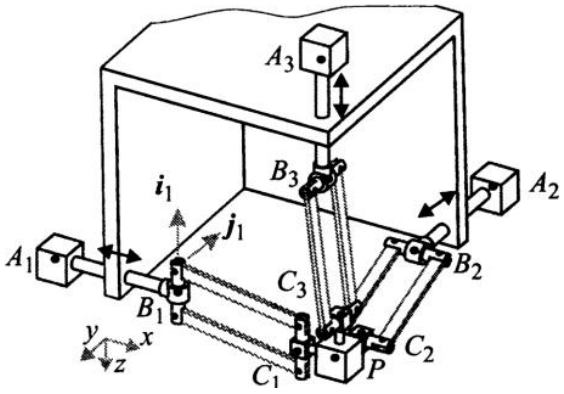


Figura 1.14 Robot paralelo Orthoglide.

Fuente: http://itzamna.bnct.ipn.mx/dspace/bitstream/123456789/4900/1/JABL.PDF

Otra variante es el H4 desarrollado por (Pierrot et al., 2003), este manipulador tiene un brazo más que le permite desarrollar un GDL de rotación, figura 1.15

Figura 1.15 Robot Adept Quattro (Adept Technology, 2011)

Fuente: http://dspace.ups.edu.ec/bitstream/123456789/1921/14/UPS-CT002354.pdf

En la figura 1.16 se muestra un mecanismo paralelo de 6 GDL diseñado a partir de dos mecanismos de 5 barras que se unen a la plataforma mediante 2 articulaciones esféricas.

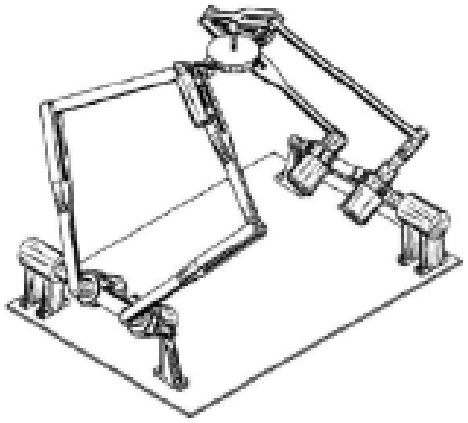


Figura 1.16 Robot paralelo 6GDL, basado en mecanismo de 5 barras

Fuente: http://itzamna.bnct.ipn.mx/dspace/bitstream/123456789/4900/1/JABL.PDF

1.4. ROBOT DELTA

(Yorobot, 2006) A principios de los años 80, Reymond Clavel (Profesor en EPFL – École la Politécnica Fédérale de Lausana) comienza con el uso de paralelogramos para construir un robot, no viene de un mecanismo paralelo patentado por la L Willard. Pollard en 1942. Lo llamó a su creación el robot Delta, esto se establecerá como uno de los diseños de robot paralelos más acertados, con varios cientos de robots activos por todo el mundo.

La idea básica del diseño de robot paralelo Delta es el empleo de paralelogramos, un paralelogramo permite a un eslabón de salida permanecer en orientación fija, con respecto a un eslabón de entrada. El empleo de tres paralelogramos restringe la orientación de la plataforma móvil, que permanece sólo con tres grados de libertad. Los eslabones de entrada de los tres paralelogramos son montados sobre palancas rotativas articuladas: con motores rotacionales o con actuadores lineales. Finalmente, una cuarta pierna es utiliza para transmitir el movimiento rotatorio desde la base a un “end-effector” montado sobre la plataforma móvil.

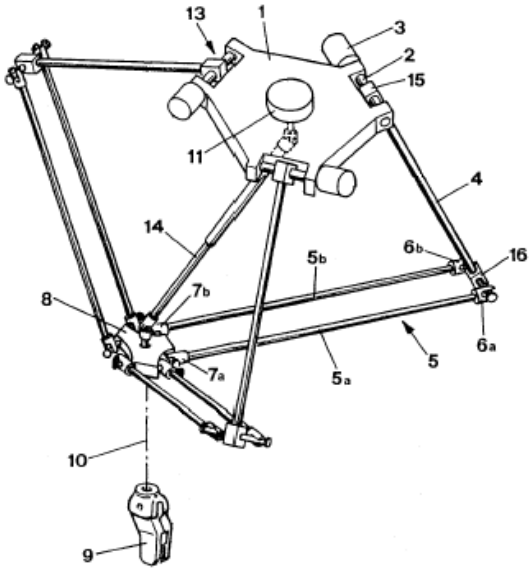


Figura 1.17 Esquema del robot Delta (US Patent N º 4.976.582)

Fuente: http://www.robotic-lab.com/blog/2006/11/26/robotica-paralela-delta/

La Patente (Bonev)**.-** El diseño del robot Delta está cubierto por una familia de 36 patentes, las más importantes son la patente de la OMPI emitido el 18 de junio de 1987 (WO 87/03528), la patente de EE.UU. emitió el 11 de diciembre, 1990 (EE.UU. 4.976.582), y la patente europea expedida el 17 de julio de 1991 (EP 0 250 470). En general, estas patentes protegen la invención en EE.UU., Canadá, Japón y países de Europa Occidental. Las patentes no especifican la manera en la que la estructura de Delta se acciona con el fin de incorporar el diseño básico, así como sus variantes (tales como la Triaglide o la Linapod).

1.5. CINEMÁTICA DEL ROBOT DELTA

(Peña, Oviedo, & Cárdenas, 2011) El problema cinemático en un robot es encontrar la relación de la posición del efector final y los ángulos, en el caso del Robot tipo Delta, el análisis cinemático inverso busca encontrar la relación entre la posición de la plataforma móvil, específicamente el punto $E(x_0, y_0, z_0)$ y los ángulos de los brazos θ1, θ2 y θ3 donde están colocados los actuadores como se muestra e la figura 1.18.

El objetivo es encontrar el ángulo de cada uno de los actuadores conociendo la posición del efector final, este problema se produce cuando se tienen las coordenadas

de un objeto que se quiere manipular y se desea saber el ángulo que debe suministrar el sistema de control.

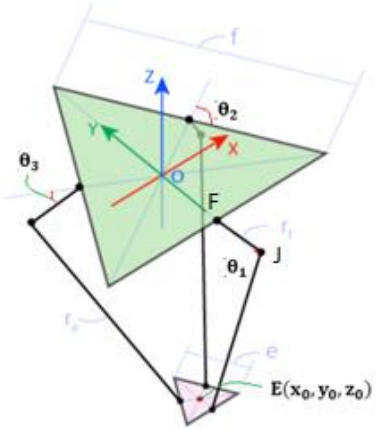


Figura 1.18 Parámetros Geométricos Robot Delta.

Fuente: http://awesomebytes.com/wp-content/uploads/2012/09/Informe_delta_spfeiffer_agabas_con_anexos.pdf

1.5.1 CINEMÁTICA INVERSA

(Pfeiffer & Gabas) Por diseño del robot, la articulación F_1J_1 ver figura 1.19 solo puede rotar en el plano YZ, formando una circunferencia con centro en el punto F_1 y con radio r_f.

Por el contrario, F_1, J_1 y E_1 son las llamadas articulaciones universales, lo que significa que E_1J_1 puede rotar libremente en relación al punto E_1, formando una esfera con centro en E_1 y radio r_e.

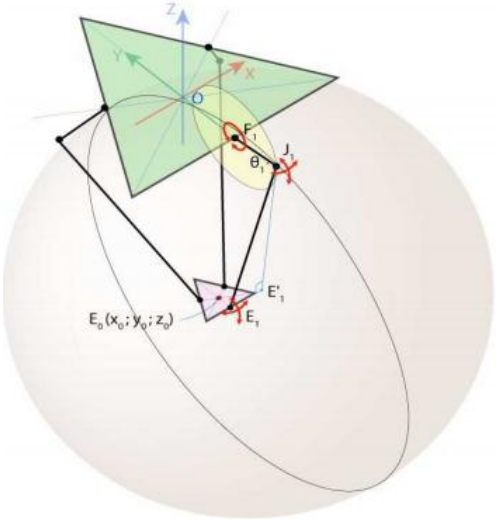


Figura 1.19 Articulación F1J1.

Fuente: http://awesomebytes.com/wp-content/uploads/2012/09/Informe_delta_spfeiffer_agabas_con_anexos.pdf

La intersección de la circunferencia y la esfera se produce en dos puntos, se toma como solución el punto con menor valor en la coordenada *Y*. Al determinar la posición del punto J1 se puede obtener el ángulo θ1 del actuador, En la Figura 1.20 se presentan los parámetros geométricos para el cálculo de las coordenadas

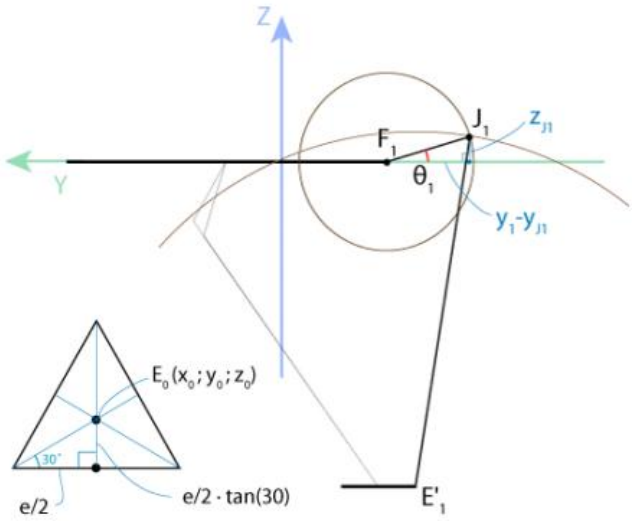


Figura 1.20 Vista lateral para análisis geométrico.

Fuente: http://awesomebytes.com/wp-content/uploads/2012/09/Informe_delta_spfeiffer_agabas_con_anexos.pdf

Coordenadas del Punto E_0,E_1, F_1 y E'_1.

Primero se realiza una sustitución donde:

$L_1 = y_{F1} = \frac{e}{2\sqrt{3}}$ $\qquad$ $L_2 = y_{E'1} = \frac{f}{2\sqrt{3}}$

Con los datos anteriores Se tiene las coordenadas de la siguiente manera:

$E_0(x_0, y_0, z_0)$; $E_1(x_0, y_0 - L_2, z_0)$; $F_1(0, -L_1, 0)$; $E'_1(0, y_0 - L_2, z_0)$

Con las coordenadas de los puntos, se plantea un sistema de dos ecuaciones no lineales que permiten encontrar la posición del punto J_1, con la cual se puede calcular el ángulo que forma el brazo con el plano horizontal, obteniendo así la solución esperada.

Sistema de ecuaciones:

$$(y_{J1} + \frac{f}{2\sqrt{3}})^2 + z_{J1}^2 = r_f^2 \quad \text{Ec 1.1}$$

$$(y_{J1} - y_0 + \frac{e}{2\sqrt{3}})^2 + (z_{J1} - z_0)^2 = r_e^2 - x_0^2 \quad \text{Ec 1.2}$$

Resolviendo este sistema de ecuaciones Ec 1.1y Ec 1.2 se llega la siguiente ecuación cuadrática Ec 1.3 que sirve para definir la solución.

$$ay_{J1}^2 + by_{J1} + c = 0 \quad \text{Ec 1.3}$$

Donde los valores de a, b y c son:

$$a = (1 + \frac{L_1 - y_0 + L_2{}^2}{z_0}) \quad \text{Ec 1.4}$$

$$b = (2\left(\frac{L_1 - y_0 + L_2}{z_0}\right)\left(\frac{r_e^2 - x_0^2 - z_0^2 - r_f^2 - L_1^2}{2z_0}\right) - 2L_1) \quad \text{Ec 1.5}$$

$$c = (\frac{r_e^2 - x_0^2 - z_0^2 - r_f^2 - L_1^2}{2z_0} - L_1^2 - r_f^2) \quad \text{Ec 1.6}$$

Cuya solución general es de la forma mostrada en la ecuación Ec 1.7.

$$y_{J1} = \frac{-b \overset{+}{-} \sqrt{b^2 - 4ac}}{2a} \quad \text{Ec 1.7}$$

Que tiene sentido solo cuando el argumento de la raíz cuadrada es positivo; de las dos posibles soluciones se toma la menor de las dos. El valor del ángulo del brazo 1 se calcula con la fórmula Ec 1.8

$$\theta_1 = \arctan\left(\frac{z_{J1}}{y_{F1} - y_{J1}}\right) \quad \text{Ec 1.8}$$

Para los otros brazos se usa la matriz de rotación con un ángulo de 120° para el brazo 2 y 240° para el 3. Esta matriz de rotación permite girar el sistema de coordenadas de manera que se pueda usar la solución descrita para el cálculo de los restantes ángulos.

Lo primero es rotar el sistema en el plano XY 120° alrededor del eje z en sentido anti-horario: Ver figura 1.21

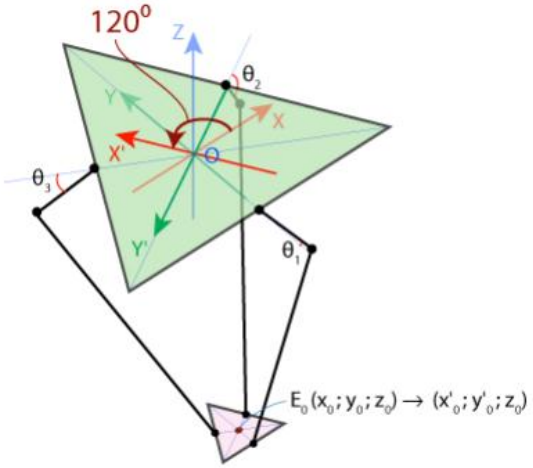


Figura 1.21 Rotación del sistema de referencia.

Fuente: http://awesomebytes.com/wp-content/uploads/2012/09/Informe_delta_spfeiffer_agabas_con_anexos.pdf

Obtener ecuaciones de rotación del sistema de referencia:

Matriz de rotación:

En dos dimensiones la matriz de rotación tiene la siguiente forma:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Para rotar vectores columna, se multiplica por la matriz de la siguiente forma:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Y entonces se tiene

$$x' = xcos(120) + ysin(120) \qquad \text{Ec 1.9}$$

$$y' = -xsin(120) + ycos(120) \qquad \text{Ec 1.10}$$

Esto da un nuevo sistema de referencia X'Y'Z' y en este sistema de referencia se encuentra $\theta 2$ utilizando el mismo algoritmo que para $\theta 1$. El único cambio que se necesita hacer es determinar las nuevas coordenadas x'o y y'o para el punto EO, esto se puede hacer fácilmente aplicando la correspondiente matriz de rotación.

Para encontrar $\theta 3$ hay que rotar el sistema de referencia en sentido horario y realizar el mismo proceso.

1.5.2. CINEMÁTICA DIRECTA

Los ángulos $\theta 1$, $\theta 2$ y θ 3 vienen dados y se necesita saber las coordenadas $(x_{0,}y_{0,}z_0)$ del punto E_0 del efector final.

Conociendo los ángulos θ, es fácil saber las coordenadas J_1,J_2 y J_3 ver figura1.22. Las articulaciones J_1E_1,J_2E_2 y J_3E_3 pueden rotar libremente alrededor de los puntos J_1,J_2,J_3 respectivamente, formando tres esferas con radio r_e.

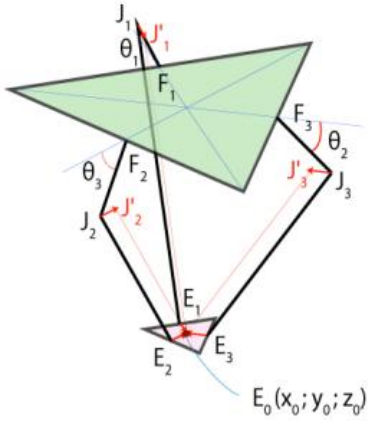


Figura 1.22 Nomenclatura de coordenadas J_1,J_2, J_3

Fuente: http://awesomebytes.com/wp-content/uploads/2012/09/Informe_delta_spfeiffer_agabas_con_anexos.pdf

Lo siguiente a realizar es mover los centros de las esferas de los puntos J_1, J_2 y J_3 a los puntos J'_1, J'_2 y J'_3 utilizando los vectores de transición E_1E_0,E_2E_0 y E_3E_0 respectivamente. Después de esta transición, las tres esferas se intersectarán en un punto: E_0 como se muestra en la siguiente figura 1.23.

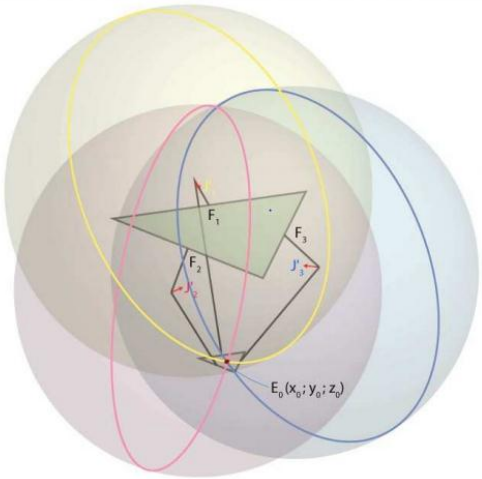
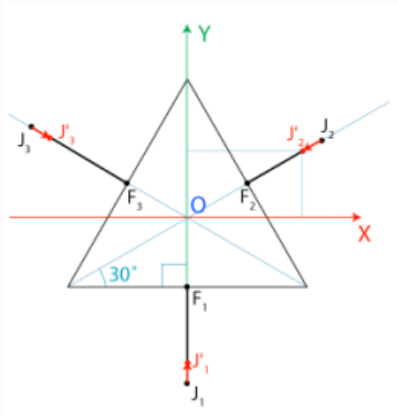


Figura 1.23 Esferas.

Fuente: http://awesomebytes.com/wp-content/uploads/2012/09/Informe_delta_spfeiffer_agabas_con_anexos.pdf

Para encontrar las coordenadas (x_0, y_0, z_0) del punto E_0, se necesita resolver el sistema de 3 ecuaciones de la forma $(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2 = r_e^2$ donde las coordenadas de los centros de las esferas (x_j, y_j, z_j) y los radios r_e son conocidos.



Primero hay que encontrar las coordenadas de los puntos J'_1, J'_2, J'_3:

Figura 1.24 Cinemática directa.

Fuente: http://awesomebytes.com/wp-content/uploads/2012/09/Informe_delta_spfeiffer_agabas_con_anexos.pdf

$$OF_1 = OF_2 = OF_3 = {}^{f}\!/_{2\tan(30)} = {}^{f}\!/_{2\sqrt{3}}$$

$$J_1J_1' = J_2J_2' = J_3J_3' = {}^{e}\!/_{2\tan(30)} = {}^{e}\!/_{2\sqrt{3}}$$

$$F_1J_1 = r_f\cos(\theta_1)$$

$$F_2J_2 = r_f\cos(\theta_2)$$

$$F_3J_3 = r_f\cos(\theta_3)$$

En las ecuaciones 11, 12, 13, 14, 15 y 16 de las tres esferas se designan las coordenadas de los puntos J_1, J_2, J_3 como (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3):

$$x^2 + y^2 + z^2 - 2y_1y - 2z_1z = r_e^2 - y_1^2 - z_1^2 \qquad \text{Ec 1.11}$$

$$x^2 + y^2 + z^2 - 2x_2x - 2y_2y - 2z_2z = r_e^2 - y_2^2 - y_2^2 - z_2^2 \qquad \text{Ec 1.12}$$

$$x^2 + y^2 + z^2 - 2x_3x - 2y_3y - 2z_3z = r_e^2 - y_3^2 - y_3^2 - z_3^2 \qquad \text{Ec 1.13}$$

$$x = a_1z + b_1 \qquad \text{Ec. 1.15}$$

$$x = a_1z + b_1 \qquad \text{Ec 1.16}$$

Resolviendo:

$(a_1^2 + a_2^2 + 1)z^2 + 2(a_1 + a_2(b_2 - y_1) - z_1)z + (b_1^2 + (b_2 - y_1)^2 + z_1^2 - r_e^2) = 0$ Ec1.17

Por último se necesita resolver esta ecuación cuadrática y encontrar z_0 (hay que elegir la raíz negativa más pequeña de la ecuación), después calcular x_0 y y_0 de las ecuaciones Ec 1.15 y Ec 1.16.

1.6. PLATAFORMAS DE DESARROLLO

1.6.1. FRAMEWORK DE DESARROLLO EN ROBÓTICA O MIDDLEWARE PARA ROBOTS

Los avances tecnológicos en informática, comunicación Wireless, mecatrónica, y tecnología de sensores son pioneros en un emergente campo de robots, ofreciendo una amplia gama de aplicaciones en tiempo real.

Las primeras generaciones de robots fueron diseñadas para realizar tareas específicas, y fabricados como una sola unidad, las nuevas generaciones de robots son considerados como complejos sistemas distribuidos y suelen ser ubicuos y autónomos esto se logra mediante la implementación del diseño modular de hardware y software donde actúan todos los componentes juntos (sensores, actuadores, y controladores) como se muestra en la **figura 1.25** obteniendo resultados eficientes, con fácil y rápida implementación, flexibles, personalizables, auto configurables, y capaz de interactuar con otros sistemas.

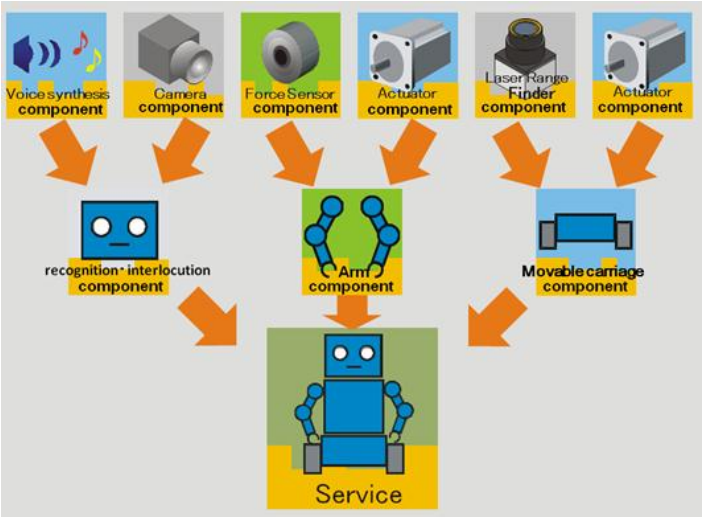


Figura 1.25 Componentes Modulares.

Fuente:http://geeksknowledge.blogspot.com/2012/09/rsf-o-middleware-para-robots.html

Ejemplos de estas aplicaciones son: búsqueda y rescate en misiones en entornos peligrosos, asistencia a las personas de la tercera edad, los robots quirúrgicos, etc.

Figura 1.26 BEAR Robot: Robot usado para búsqueda y rescate.

Fuente: http://4gwar.wordpress.com/2013/08/12/unmanned-systems-big-droid-drone-and-bot-show-opens-in-d-c/

Figura 1.27 Robot que asistirá a personas de la tercera edad.

Fuente: http://newsoaxaca.com/index.php/actualidad/16925-un-robot-asistira-a-personas-de-la-tercera-edad

Figura 1.28 Robot quirurgico Da Vinci.

Fuente: http://avancescientificosytecnologicos-cmm.blogspot.com/2010/11/da-vinci-robot-que-realiza-operaciones.html

1.6.2. RSF (Robotic Software Framework)

(Nader Mohamed, A Review of Middleware for Networked Robots , 2009) Se trata de una capa de abstracción software que sirve para que el programador se despreocupe de la implementación específica de un hardware y de todos sus detalles como drivers o APIs del fabricante. De esta forma se normaliza la programación para robots, y permite por ejemplo usar un láser dando igual la marca o modelo que sea, siempre que este tenga soporte dentro del middleware para robots. Sin embargo, el diseño y desarrollo de un middleware de éxito para los robots no es trivial.

Tiene que hacer frente a muchos desafíos dictados por robots, características por un lado y los requisitos de las aplicaciones por otra parte. Estos desafíos son los siguientes (Nader Mohamed, Middleware for Robotics: A Survey , 2008):

- **Simplificar el proceso de desarrollo:** el desarrollo de aplicaciones no es fácil para multiplex robots. El framework debería simplificar el proceso de desarrollo, proporcionando abstracciones de nivel superior con interfaces simplificadas que pueden ser utilizadas por los desarrolladores y el middleware también debe mejorar la integración de software y la reutilización.
- **Soporte de comunicaciones e interoperabilidad:** módulos robóticos puede ser diseñado e implementado por diferentes fabricantes. Comunicación eficiente y mecanismos de interoperabilidad simples se necesitan entre estos módulos. Por lo tanto, el framework robótico debería proporcionar estas funciones. Además, múltiples robots pueden estar dispuestos de manera ad hoc en la que no se pueden comunicar directamente uno con el otro. Este tipo de apoyo puede ser proporcionado por el middleware.
- **Proporcionar la utilización eficiente de los recursos disponibles:** Los robots en general tienen que ejecutar tareas de procesamiento y de comunicación intensiva en tiempo real. Ejemplos de ello son el procesamiento de la visión, la cartografía y la navegación. Por lo tanto, es necesaria la utilización eficiente de los componentes del robot y recursos. Los robots pueden tener microprocesadores individuales o múltiples, una o más redes de interconexión y varios otros recursos. Middleware debería ayudar en

la utilización eficiente de estos recursos para los diferentes requerimientos de la aplicación.

- **Proporcionar abstracciones heterogeneidad:** cualquier sistema robótico contiene muchos componentes de hardware y software heterogéneos. La comunicación y la cooperación entre estos componentes es un aspecto importante. Comúnmente la abstracción de este papel es desempeñado por un middleware que actúa como una capa de software de colaboración entre todos los módulos implicados, ocultando la complejidad de la comunicación de bajo nivel y la heterogeneidad de los módulos.
- **Apoyo a la integración con otros sistemas:** Los nuevos tipos de robots necesitan interactuar con otros sistemas, redes de sensores inalámbricos y servidores de gama alta. La mayoría de estas interacciones se deben hacer de una manera abstracta y en tiempo real. Por lo tanto, el middleware debe proporcionar servicios de interacción en tiempo real con otros sistemas.
- **Ofrecer servicios de robots a menudo necesarios:** Una gran cantidad de esfuerzo se gasta al escribir nuevas implementaciones de algoritmos existentes y servicios de control para los robots varias veces. El mismo algoritmo / servicio puede ser reescrito varias veces debido a los cambios en hardware del robot, el desarrollo de nuevas aplicaciones, los cambios en los sistemas operativos, los cambios de personal técnico, o simplemente para añadir nuevas funcionalidades. Estos servicios de robots a menudo necesarios deben ser proporcionados por el middleware robótico que permite la reutilización de los módulos que ofrecen estas funcionalidades.
- **Proporcionar descubrimiento recurso automático y configuración:** sistemas robóticos en red se consideran sistemas dinámicos debido a la movilidad de los robots y los cambios de su entorno. Por ejemplo, los dispositivos externos pueden estar disponibles dinámicamente / no disponible para el uso de un robot. Estos recursos externos pueden ser utilizados por los robots conectados en red para mejorar la potencia de procesamiento y la precisión cuando están disponibles. Por lo tanto, es necesaria la búsqueda de recursos automáticos, dinámicos y de configuración. Además, debe apoyar

los mecanismos de los robots para ser auto-adaptación, auto-configuración y auto-optimización.

- **Apoyo a los componentes embebidos y dispositivos de bajos recursos:** los robots en muchos casos utilizan o interactúan con los dispositivos integrados que puede tener varias limitaciones como un poder limitado, pequeña memoria, las funcionalidades del sistema operativo limitadas y limitada conectividad. El manejo de esos recursos suele ser de diferente manejo de otros recursos ordinarios; Por lo tanto, el middleware debe proporcionar funcionalidades especiales para gestionar los recursos como sea necesario.

Los middlewares más representativos diseñados específicamente para el desarrollo, programación, simulación y reutilización de sistemas robóticos, son:

- Microsoft Robotics Studio
- NXJ (Open source Java pmrg enviroment for the lego robot kit)
- ROS (Robotic Operating System)
- Player (robot framework)
- Orosco (C++ framework for components-base robot control software)
- Rock (the Robot Construction Kit)
- Orca (robot framework)
- MOOS (robot framework)
- CARMEN (robot simulator)
- Simbad robot simulator (robot simulator)
- Gazebo (multi-robot simulator)
- OpenJAUS (robot/unmanned system framework)
- Open RTM (Robotic technology middleware).

La elección o desarrollo de una arquitectura middleware para una aplicación robótica es crucial debido a la integración de todos los elementos y, sobre todo, para

alcanzar aspectos tan importantes como la eficiencia, el rendimiento, o la reusabilidad.

Pero también a la hora de hacer la elección del RSF que se quiere usar en el robot, debería ser un factor muy importante la existencia de un software de simulación de manera que el desarrollador no necesite tener el hardware real para programar la aplicación que maneja el robot.

Dados estos requisitos, parece obvio que lo mejor es usar ROS, que es un framework para el desarrollo de software específico para robótica mucho más robusto y soporta plataformas robóticas más complejas, a diferencia de otros que están más enfocado a plataformas robóticas simples. Es decir ROS ha logrado agrupar muchas de las mejores características de todos estos proyectos dando una solución integral al problema de desarrollo de sistemas robóticos.

Además, ROS proporciona también servicios estándares propios de un sistema operativo pero sin serlo, ya que se instala sobre otro, en general Linux y de manera recomendada Ubuntu, y por ello también recibe la denominación de Meta-Sistema Operativo.

1.7. ROS (Robotic Operating System)

(Berger, 2010) Fue creado por el instituto de investigación Willow Garage en 2007 bajo licencia BSD y todos sus programas son de código abierto, permitiéndose así su uso gratuito tanto para la investigación como para fines comerciales.

Es un software diseñado para hacer más fácil a las personas, controlar y programar robots. Desempeña el rol de un sistema de robot similar a un sistema operativo como: Windows o Linux a manera de un juego en un ordenador.

ROS proporciona interfaces estandarizadas para el hardware, herramientas para la creación, depuración, distribución y ejecución de programas y librerías para hacer una variedad de tareas comunes fáciles de lograr.

Figura 1.29 Logotipo de ROS.

Fuente: http://www.ros.org/

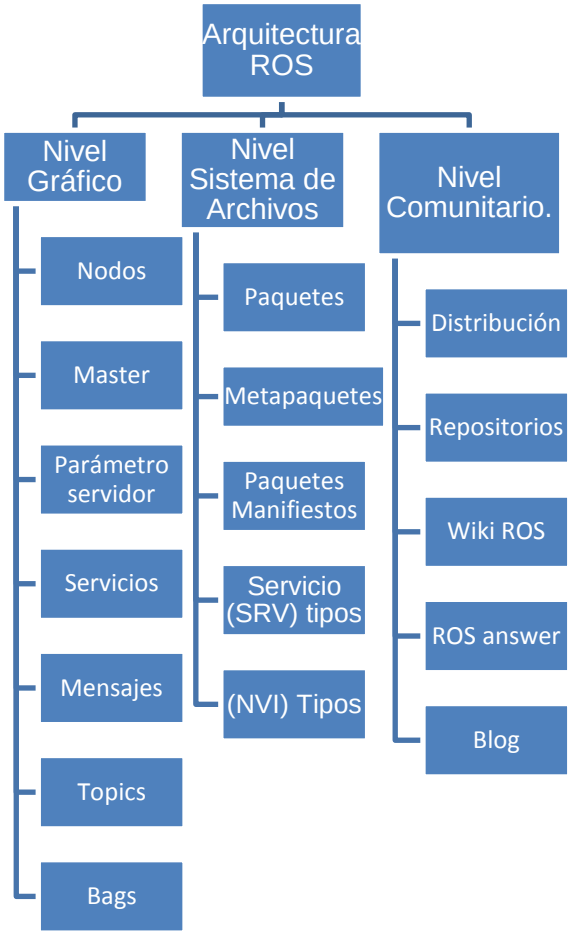
1.7.1 FILOSOFÍA DE ROS (Torre, 2012)

FILOSOFÍA

Todos los programas sean de uso libre, reutilizables, escalables, según la aplicación deseada e integrables con todo el software de robótica existente y futuro. Por ello el sistema se creó teniendo en cuenta otras plataformas software abiertas ya existentes como OpenCV, Player/Stage, Gazebo, Orosco/KDL y otros.

1.7.2 ARQUITECTURA DE ROS

(María Fernanda Utreras Abad, 2013) (Foundation, Qué es ROS , 2013) ROS está dividido en tres niveles: nivel gráfico, nivel sistema de archivos y nivel comunitario.



Organigrama 1.1. Arquitectura de ROS.

➢ **Nivel Gráfico**

El gráfico de la computación es la red peer-to-peer de los procesos de ROS que están procesando los datos conjuntamente. Se trata de un gráfico bastante útil en el desarrollo, ya que de una forma sencilla, se puede observar las conexiones internas en ROS.

- **Nodos:** Son los procesos ejecutables que realizan el cómputo. En un sistema de control robótico se espera que existan varios nodos, donde cada uno se encargue de ejecutar una misión particular.
- **Máster:** Es el proceso central que mantiene un registro de todo el grafo computacional. Es quien permite la comunicación entre los nodos. Además, cuenta con un servidor de parámetros que permite centralizar cierta información.
- **Parámetro Servidor** El servidor de parámetros permite que los datos se almacenen con llave en un lugar central. En la actualidad es parte del Máster.
- **Servicio:** Los servicios se definen mediante archivos SRV, que se compilan en el código fuente de una biblioteca de ROS. Los servicios están conformados por dos partes: un mensaje que es el que pregunta y otro mensaje que es el que responde.

El gráfico de la Figura 1.30 resume los dos métodos de comunicación entre nodos:

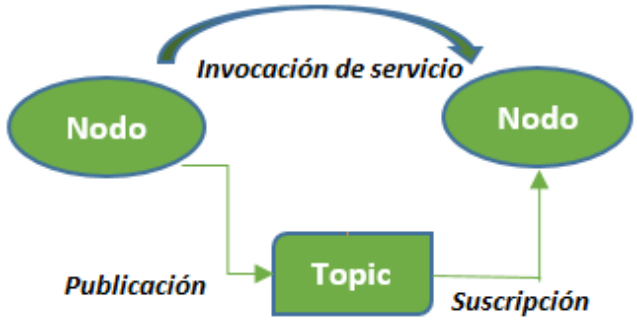


Figura 1.30 Métodos de Comunicación entre Nodos.

Fuente: http://wiki.ros.org/ROS/Concepts

De la figura 1.30 se puede observar las dos maneras de comunicarse entre los nodos: En primer lugar, a través de un tópico de comunicación, donde un nodo publica información en este tópico, y esta es recibida por todo nodo que esté suscrito al mismo tópico. Por otra parte, es posible que un nodo se comunique directamente con otro nodo a través de la invocación de un servicio.

- **Mensaje:** Los nodos se comunican entre sí enviándose mensajes. Un mensaje es una estructura de datos simple basada en campos de datos con tipo, que pueden ser tipos primitivos (integer, floating point, boolean, etc.) o arreglos de estos.
- **Tópico:** Los mensajes se envían generalmente en base a un sistema de publicación/suscripción.
- **Bags:** Las bolsas son un formato para guardar y reproducir datos de mensajes de ROS.

➢ **Nivel Sistema de Archivos**

Los conceptos de nivel de sistema de archivos son recursos ROS que se encuentran en el disco. Ejemplo (Location on my disk: /opt/ros/cturtle/common)

- **Paquetes:** Son la unidad principal de organización del software en ROS. Un paquete contiene procesos (nodos), librerías de desarrollo, sets de datos, ficheros de configuración.

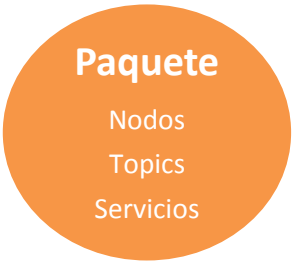


Figura 1.31 Estructura basica de un paquete.

Fuente: http://www.slideshare.net/vicegd/desarrollo-robtico-robot-operating-system-ros

- **Metapaquetes:** Son paquetes especializados que sólo sirven para representar un conjunto de paquetes relacionados.

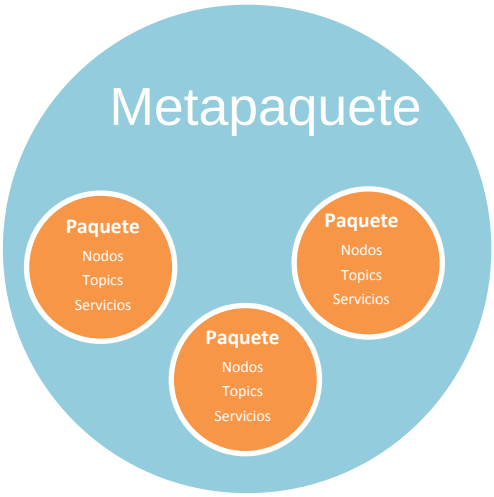


Figura 1.32 Estructura básica de un meta-paquete.

Fuente: http://www.slideshare.net/vicegd/desarrollo-robtico-robot-operating-system-ros

- **Paquete Manifiestos:** Manifiestos (Paquete.xml) proporcionan metadatos sobre un paquete, que incluye su nombre, versión, descripción, información sobre la licencia, las dependencias, y otra información de meta como paquetes exportados.
- **(NVI) Tipos:** Descripciones de los mensajes almacenados en my_paquete/msg/MyMessageType.msg, definen las estructuras de datos de los mensajes enviados en ROS.
- **Servicio (SRV) tipos:** las descripciones de servicio, almacenados en mi paquete define la solicitud y las estructuras de datos de respuesta para los servicios de ROS.

➢ **Nivel Comunitario**

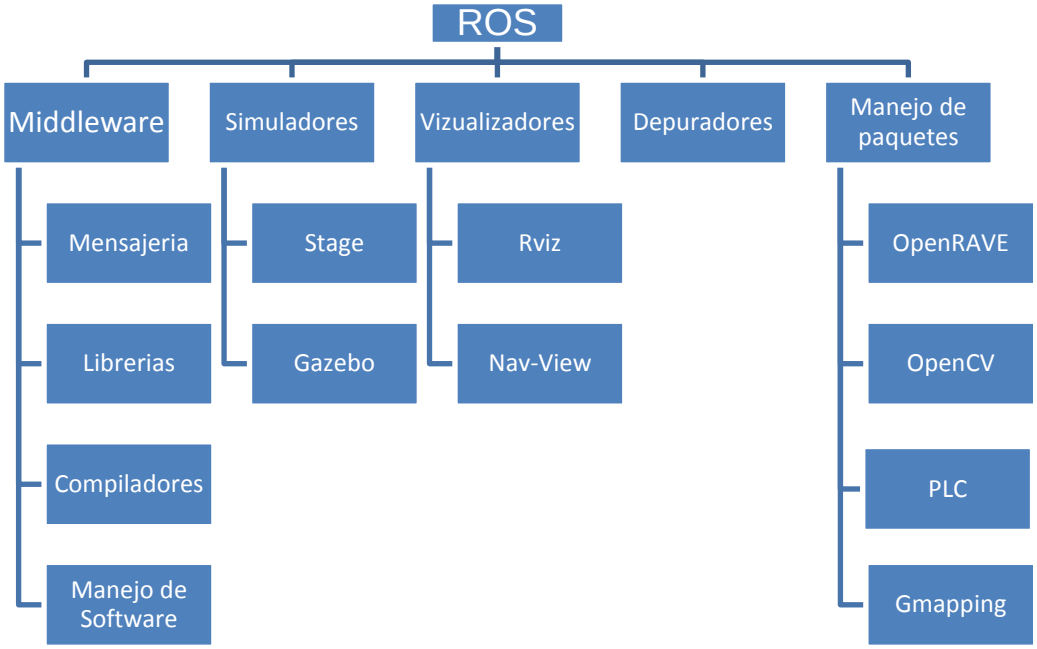
Son recursos que permiten a los usuarios intercambiar software y conocimiento. Estos recursos incluyen:

- **Distribuciones:** ROS Distribuciones son colecciones de paquetes que se pueden instalar. Las distribuciones juegan un papel similar al de las

distribuciones de Linux: hacen más fácil la instalación de una colección de software, y también mantienen versiones consistentes en un conjunto de software.

- **Repositorios:** ROS se basa en una red federada de repositorios de código, en los que diferentes instituciones pueden desarrollar y lanzar sus propios componentes de software del robot.
- **Wiki ROS:** Es el principal foro para la documentación de información sobre ROS. Cualquier persona puede inscribirse a una cuenta y contribuir con su propia documentación, proporcionar correcciones o actualizaciones, escribir tutoriales y más.
- **ROS Answer:** Un sitio Q & A (Question and Answer) para responder a sus preguntas relacionadas con ROS.
- **Blog**: El blog de Willow Garage[1] ofrece actualizaciones periódicas, incluyendo fotos y videos.

1.7.3 Componentes de ROS



Organigrama 1.2. Componentes de ROS.

[1] Willow Garage es un laboratorio de investigación robótica dedicada a la creación de software de código abierto para las aplicaciones para robots personales.

Distribuciones de ROS

En la tabla 1.1 se indica las diferentes versiones de ROS.

Tabla 1.1 Diferentes distribuciones de ROS.

DISTRIBUCIÓN	FECHA DE LANZAMIENTO
ROS Box Turtle	2 marzo 2010
ROS C Turtle	2 Agosto 2010
ROS Diamondback	2 Marzo 2011
ROS Electric Emys	30 Agosto 2011
ROS Fuerte	23 Abril 2012
ROS Groovy y Galapagos	31 Diciembre 2012
ROS Hydro Medusa	4 Septiembre 2013
ROS Indigo Igloo	Mayo 2014

Fuente: http://wiki.ros.org/Distributions

A continuación en la tabla 1.2 se indica las ventajas e inconvenientes de ROS

Tabla 1.2 Ventajas y Desventajas de ROS

VENTAJAS	DESVENTAJAS
Reduce el tiempo invertido en infraestructura y se centra en la investigación.	Sus objetivos son muy ambiciosos.
Aborda problemas de alto nivel.	Tratan con software muy variado (librerías hechas según criterio del autor).
Acelera el aprendizaje. • Viendo código de otros. • Viendo documentación de otros.	La integración de las aplicaciones no es inmediata. • Hay que leer suficiente documentación para comprender el funcionamiento. • Se debe revisar el código hecho, pues cada aplicación es distinta.

CONTINUA →

Fomenta el trabajo en equipo y establece convenios, procesos y metodologías para hacer	Es necesaria una etapa de depuración.
Software reusable.	Exponer una librería en ROS no es gratis.
	Continuos cambios y evolución, que pueden dar como resultado un software obsoleto.
	Por ahora no tiene soporte para Windows u otros sistemas empotrados.

Fuente: http://e-archivo.uc3m.es/handle/10016/16723

1.8. ACTUADORES DE ALTO DESEMPEÑO

(Bautista Quintero Ricardo, 2011)La tecnológica de estos actuadores inteligentes surge en las aplicaciones industriales que requerían desconcentrar el proceso de control en varios procesadores trabajando en paralelo. Este concepto se desarrolló en la industria con diversos protocolos, topologías y tipos de buses. Por ejemplo la tecnología ASI, bitbus, profibus, fieldbus y Ethernet. Todas estas tecnologías están enfocadas a comunicaciones a distancia y permiten manejar varias computadoras industriales o PLCs en conjunto.

De esta manera las redes industriales evolucionan y permiten en la actualidad el manejo práctico de servomotores con capacidades similares a las de una red industrial conectados en un bus. Por ejemplo, la **Tabla 1.3** muestra una comparación entre las características básicas de un servomotor RC (Radio Control) estándar y un servomotor de alto desempeño para conexión en red. Adicionalmente a lo descrito en la tabla los servomotores digitales AX-12+ cuentan con sensores de temperatura, posición y permite definir de forma programada el torque y velocidad máximos. Con los actuadores AX-12+ se pueden controlar hasta 255 servomotores por cada pin del microcontrolador utilizando el protocolo RS-232.

Comparativo de un servomotor de radio control estándar y un servomotor de alto desempeño.

Tabla 1.3 Servomotor de radio control y servomotor Dynamixel.

	Dynamixel AX-12+	Hitec HS-311
Voltaje de operación	7V~ 10V	4.8V~6V
Torque (kgf/cm)	12~16.5	3~3.5
Angulo de operación	300°	180°
Feedback	Si	No
Comunicación	Serial	PWM
Resolución	0.35°	*

*depende de la velocidad del procesador que lo habilite.

Fuente: http://www.mecamex.net/anterior/cong10/trabajos/art32.pdf

El sistema de conexión de red entre los servomotores está ilustrado en la Figura1.33

Figura 1.33. Sistema de conexión en red serial para los actuadores del robot.

Fuente: http://www.mecamex.net/anterior/cong10/trabajos/art32.pdf

El propósito de este tipo de topología está enfocado a distribuir las señales de control y alimentación eléctrica del servomotor.

Al guiarse en la **tabla 1.3** los servomotores adecuados son los servos Dynamixel y que fueron escogidos debido a que poseen características necesarias en la implementación del desarrollo del proyecto Robot delta. Características como la

posibilidad de ser controlados ya no solo en posición sino también en velocidad, su peso y tamaño reducido y sobre todo que se puede trabajar con ROS, fueron decisivas al momento de elegirlos.

1.8.1. ACTUADORES DYNAMIXEL

Los actuadores Dynamixel son utilizados por muchas universidades y centros de investigación de todo el mundo para diseñar su propio robot. Estos actuadores se distinguen por su precisión, alta calidad y las características de gran alcance que ofrece retroalimentación (mediciones de posición, velocidad, temperatura, entrada de tensión).

Figura 1.34 Actuadores Dynamixel AX-12A.

Fuente: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

Los servomotores Dynamixel se utilizan para realizar los brazos robóticos, robots móviles o incluso robots humanoides, ejemplos de aplicación se observan en la figura 1.35

Figura 1.35. Aplicaciones Dynamixel.

Fuente: http://www.robotica-personal.es/2009/11/nuevo-kit-robotis-bioloid-premium-en-ro.html

Existen numerosos modelos con características muy distintas. A continuación se describen los modelos utilizados en este proyecto.

Especificaciones del actuador AX-12A

- **Peso:** 55g
- **Dimensiones :** 32mm x 50mm x 40mm
- **Resolución:** 0.29°
- **Relación de reducción:** 1: 254
- **Torque :** 15.3 kg.cm
- **Velocidad de carga:** 59rpm (at 12V)
- **Angulo de operación:** 0° ~ 300°
- **Rango Temperatura:** -5℃ ~ +85℃
- **Voltaje:** 9 ~ 12V (Voltaje recomendado 11.1V)
- **Tipo de Comunicación:** Half duplex Asincrona, Comunicación serial (8bits, 1stop, No Parity)
- **ID:** 254 ID (0~253)
- **Velocidad de comunicación:** 7343bps ~ 1 Mbps
- **Feedback:** Posición, Temperatura, carga, entrada de voltaje.
- **Material :** Plástico

Dimensiones del Dynamixel AX-12A

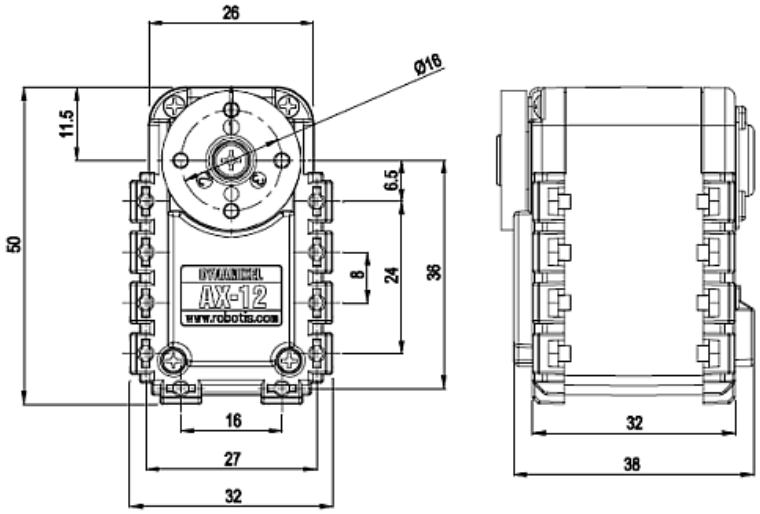


Figura 1.36. **Dimensiones del actuador Dynamixel AX-18A.**

Fuente: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

Dynamixel Pin de conexión

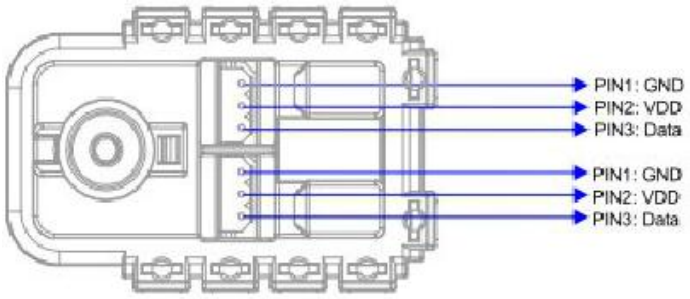


Figura 1.37 Motor Pin.

Fuente: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

1.8.2. USB2Dynamixel

(Czachórski, Tadeusz; Kozielski, Stanislaw ; Stanczyk, Urszula , 2011) El USB2Dynamixel es un dispositivo utilizado para accionar directamente el Dynamixel en una PC como se muestra en la **figura1.38**, es posible acceder a los actuadores directamente mediante un cable USB basado en el integrado FTDI. El cable dispone de conector DB9 u conector molex de 4 pines para RS-232 y RS-485 en los modelos de alta gama, y molex de 3 pines para los modelos TTL (modelos AX). A continuación se muestra el detalle de conexión para TTL:

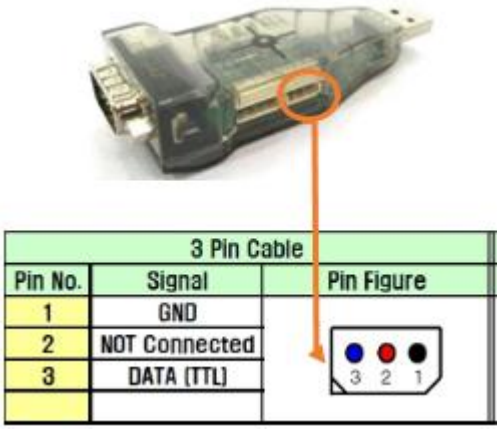


Figura 1.38. Conexión actuadores AX.

Fuente:http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm

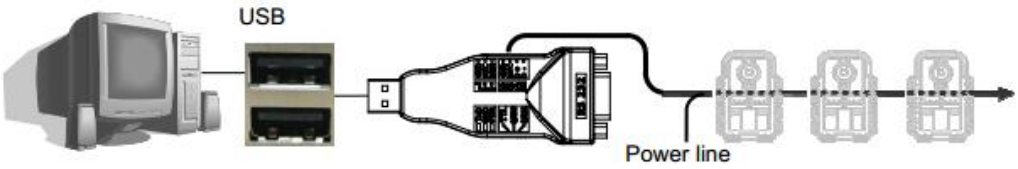


Figura 1.39 Control del USB2Dynamixel usando una PC.

Fuente:http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm

1.9. LÁSER

(Arranz, 2011) La palabra láser proviene de las iniciales L.A.S.E.R. cuyo significado es la frase inglesa "light amplification by stimulated emission of radiation" o sea amplificación de luz por emisión estimulada de radiación.

La salida de un láser puede ser un haz pulsado o continuo con diferente λ (longitud de onda); con potencias menores de 1 mili-vatio hasta potencias que alcanzan los millones de vatios. Pese a todas estas diferencias todos los láseres presentan características en común:

1.9.1. CARACTERÍSTICAS DE LA LUZ LÁSER.

(López, 2010) La luz láser presenta características bien definidas y específicas de ella como se muestra en la figura 1.40 y son:

- **Coherencia:** Esto significa que todas las ondas que conforman el haz láser, están en cierta fase relacionadas una con otra, tanto en tiempo como en espacio. Esto se debe a que cada fotón está en fase con el fotón entrante, característica que la luz led y la luz normal no poseen. Ésta característica hace que actúen en forma diferente sobre los tejidos.
- **Colimada:** Direccionalidad, en una sola dirección, ya que todas las ondas emitidas están casi paralelas y por tanto no hay divergencia del rayo de luz, por lo que permanece invariable aún después de largos recorridos.
- **Monocromaticidad:** Se da cuando una fuente luminosa establece su potencia de salida sobre una única longitud de onda, tal es el caso del láser; caso contrario se presenta por ejemplo con la luz del sol.

- **Intensidad (brillo):** Está definida como la cantidad de flujo luminoso que produce una fuente cualquiera.
- **Baja corriente, tensión y energía en general:** En su mayoría, los diodos láser exigen únicamente sólo algunos miliWatts de potencia, de 3 a 12 VDC, con una corriente relativamente pequeña.
- **Rayo de ángulo ancho:** El diodo láser produce un cono luminoso, pero este puede ser enfocado por medio de lentes convexos.

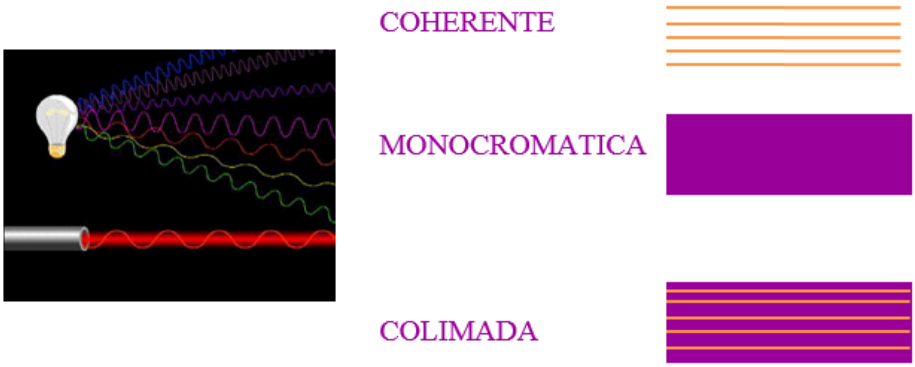


Figura 1.40 Comparación entre la luz normal y la luz láser.

Fuente: http://www.gnclaser.com/tecnicalaser.php?id=es

1.9.2. ELEMENTOS DE UN LÁSER.

(VIZUET, 2007) Para el correcto funcionamiento de un láser, es necesaria la inclusión de 3 fases:

- Elemento excitador (fenómeno de bombeo).
- Medio activo, emisor.
- Elemento amplificador (que es el resonador óptico en sí).

En la tabla 1.4 se describe las características de los tipos de láser.

Tabla 1.4 Características de los tipos de láser

Láser	Medio Activo	Rango de frecuencia de emisión	Régimen de emisión	Utilidades
Helio-Neón	Gas	Rojo	Continuo	→Metrología →Lectores de códigos de barras.
Ion de Ar	Gas	Verde-Azul	Continuo	→Bombeo →Espectáculos

CONTINUA →

Ion de Ar	**Gas**	**Verde-Azul**	**Continuo**	**→Bombeo →Espectáculos**
CO2	Gas	Infrarrojo	Continuo o pulsado	→Corte →Soldadura →Cirugía
Excímero	Gas	Ultravioleta	Pulsado	→Microprocesado →Cirugía
Químicos	Gas	Infrarrojo	Continuo	→Escudos antimisiles
Colorante	Líquido o Sólido	IR-Visible-UV	Continuo o pulsado	→Espectroscopía
Rubí	Sólido	Rojo	Pulsado	→Investigación
Neodimio: YAG	Sólido	Infrarrojo (*)	Continuo o pulsado	→Bombeo →Procesado de materiales →Cirugía
Titanio: Zafiro	Sólido	Infrarrojo	Continuo o pulsado	→Investigación →Pulsos ultracortos
Semiconductor	Sólido	Infrarrojo-Visible	Continuo	→Comunicaciones →Cd, DVD →Punteros →Bombeo
Fibra	Sólido	Infrarrojo-Visible	Continuo o pulsado	→Procesado de materiales →Comunicaciones →Espectroscopía
Electrones libres	(**)	Microondas - Rayos X	Pulsado	→Investigación

(*) La luz de estos láseres suele doblarse en frecuencia mediante un proceso llamado generación de segundo armónico, dando lugar a un haz de luz verde.

(**) Los láseres de electrones libres se basan en un mecanismo completamente distinto al del resto de láseres y no tienen un medio activo propiamente dicho.

Fuente:http://tesis.ipn.mx/xmlui/bitstream/handle/123456789/5120/1711_2007_ESIME-ZAC_MAESTRIA_hernandez_vizuet_mauricio.pdf?sequence=1

Elemento excitador: Este se produce por medio de la excitación del dispositivo y puede ser un sólido, líquido, gas, o algún material semiconductor que pueda ser bombeado a un estado de energía más alto.

Un medio activo: Que permita bombear energía en el medio excitable. Este medio puede ser óptico, químico, eléctrico, etc.

El elemento amplificador: Se da dentro del ya conocido resonador. En la mayoría de los casos la cavidad de resonancia es del tipo Fabry–Perot, un par de espejos, uno a cada lado del láser, los cuales permiten que la luz estimulada rebote de un lado al otro a través del medio excitable. Generalmente uno de los espejos es reflexivo mientras que el otro es parcialmente transparente lo que permite que el haz del láser escape. Los espejos pueden ser perfectamente planos, o uno de los dos o los dos pueden presentar una ligera curvatura.

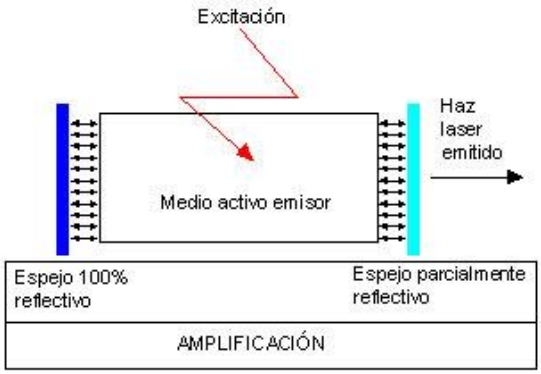


Figura 1.41 Elementos del diodo láser.

Fuente: http://www.gnclaser.com/tecnicalaser.php?id=es

1.9.3 DIODO LÁSER

(Arranz, 2011) Es un chip de material semiconductor controlado por una fuente de alimentación de bajo voltaje. Este tipo de láser tiene una retroalimentación óptica a través de un fotodiodo. El objetivo de tener este último elemento es el de regular la corriente del diodo láser.

Las longitudes de onda de estos dispositivos son descritos en la tabla 1.5:

Tabla 1.5 Longitud de onda del láser según el color.

Rojo	760-630 nanómetros
Naranja	630-600 nanómetros
Amarillo	600-570 nanómetros
Amarillo-verdoso	570-550 nanómetros
Verde	550-520 nanómetros
Verde-azulado	520-500 nanómetros
Azul	500-450 nanómetros
Violeta	450-380 nanómetros

Fuente: Arranz, A. C. (2011). *Tecnología Láser y sus aplicaciones Industriales.* (Vol. I). Barcelona, España: Marcombo, S.A. Recuperado el 5 de Junio de 2014

La calidad del haz láser para los diodos láser depende del diseño. El haz en forma pura presenta una forma elíptica o una forma de cuña, siendo a su vez muy astigmático. Su corrección requiere óptica adicional (interna o externa).

La longitud de su coherencia va desde unos cuantos milímetros hasta algunos metros. Las potencias de salida más comunes en el mercado van desde 0.1 mW hasta los 5 mW. Existen diodos láser con potencias arriba de 100 W. Para los diodos láser de muy altas potencias los sistemas no están hechos por un solo diodo sino por arreglos de diodos los cuales permiten alcanzar potencias de más de 100,000W.

Es posible encontrar diodos láser desde 1 dólar hasta más de 10,000 dólares. En general los láseres de diodos son de bajo costo, de baja potencia a la entrada, pero en la mayoría de los casos su controlador requiere de altas exigencias en su funcionamiento lo que eleva su costo.

Encapsulado de un diodo láser comercial

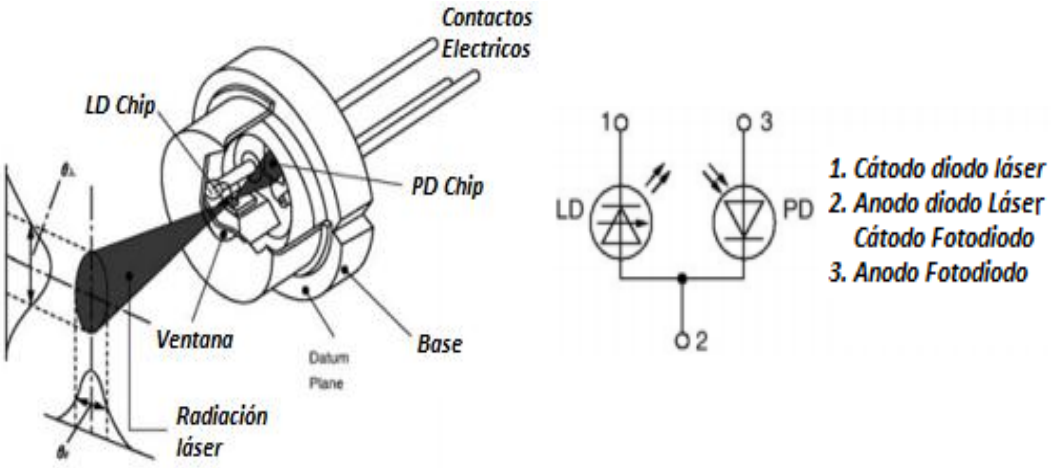


Figura 1.42 Estructura del encapsulado del dispositivo láser.

Fuente: http://kaven-fontes.blogspot.com/

1.10 ARTÍCULOS PUBLICITARIOS

1.10.1 GRABACIÓN POR LÁSER, CORTE POR LÁSER

(kairel)Técnica de impresión de artículos promocionales y de marketing utilizada principalmente sobre materiales resistentes y/o deslizantes que no permiten un buen agarre de la tinta de marcaje. Esta técnica permite marcar de forma permanente cualquier producto con la seguridad de que el logotipo o mensaje no será borrado nunca. El principal problema de esta técnica es que no permite seleccionar el color de marcaje, ya que se trata de una técnica basada en rasgar el producto creando un desnivel con el marcaje.

Cuando se graba por láser, el material de base es fundido o evaporado por la radiación láser. La intensidad de la radiación láser deberá por tanto superar un valor límite determinado, la denominada intensidad de valor umbral. La grabación por láser es el método más rápido de la mecanización por láser.

Distinguimos básicamente entre corte láser por fusión y corte láser por sublimación. En el corte láser por fusión se funde o se evapora el material, por ejemplo, un material acrílico. En el corte láser por sublimación se evapora el material, por ejemplo, la madera, saltándose la fase de licuefacción.

Su utilización va en constante aumento debido a su bajo costo y su elevada calidad. Y se puede realizar el marcaje en todo tipo de productos en diferentes materiales como:

- Maderas, MDF,
- Papel, cartulina, cartón, polietilenos.
- Acrílico, plásticos, pvc, estireno, gomas, foamy,
- Hule, espuma, esponja,
- Tela, cuero, cuero sintético, mezclilla, fieltro, etc.

Esta técnica ofrece una personalización elegante y permanente en los artículos en los que se aplica como se muestra en la figura 1.43.

Figura 1.43 Grabado Láser.

Fuente: http://kaven-fontes.blogspot.com/

CAPÍTULO II

DISEÑO Y CONSTRUCCIÓN DEL SISTEMA MECÁNICO

2.1. PARAMETROS DE DISEÑO

Para el diseño de cada una de las partes que conforman el sistema mecánico los principales parámetros que se debe tomar en cuenta son el material y dimensiones para el diseño.

2.1.1. MATERIAL

Para seleccionar el material del robot debe cumplir con ciertas restricciones, ya que el robot genera momentos de inercia por tener movimiento, para disminuir el momento de inercia al máximo el material debe ser ligero.

Además de ser ligero, el material debe ser resistente a la deformación que puede causar las fuerzas ejercidas por los momentos de inercia a los componentes del robot.

El último parámetro a tomar en cuenta es la maquinabilidad, ya que el material debe ser mecanizado por máquinas convencionales (torno y fresa)

En resumen el material debe ser:

- Bajo en peso
- Resistente
- Maquinabilidad

El material a tomar en cuenta y que cumple con los parámetros deseados son el Aluminio por su bajo peso para la estructura principal figura 2.1 donde se genera los movimientos y así evitar el movimiento de inercia. El Acero por su gran densidad la utilizamos en la estructura secundaria o parte fija ver figura 2.2

Figura 2.1 Estructura Principal

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

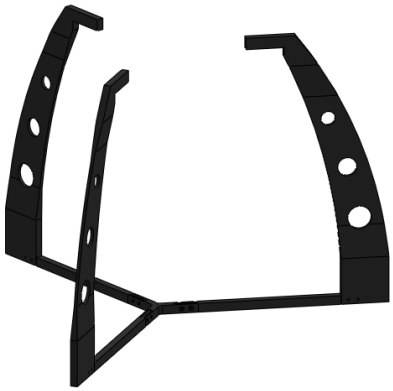


Figura 2.2 Estructura Secundaria o Base

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

A continuación analizamos los materiales a utilizar:

Aluminio: este material es un metal no ferroso, posee propiedades que lo hacen muy útil en la ingeniería mecánica, como su baja densidad y su alta resistencia a la corrosión y se mecaniza con facilidad es el metal que más se utiliza después del acero.

Acero: El acero es la aleación de hierro y carbono muy resistente, tiene su densidad casi 3 veces mayor al aluminio.

A continuación se presenta la tabla 2.1 de las características del material utilizado en el robot.

Tabla 2.1 Propiedades de los materiales

Material	Densidad (g/cm^3)	Módulo de elasticidad [GPa]
Acero	7.85	200
Aluminio	2.7	68.9

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Como se observa en la tabla 2.1 el aluminio es de bajo peso, por ende es el material a utilizar en la estructura principal del robot por su baja densidad. El acero tiene una gran densidad por ende el peso es alto, perfecto para la estructura secundaria del robot para que mantenga fija a la estructura principal y no se mueva en conjunto producto de los momentos o torque del servomotor al realizar los movimientos, también con esto evitamos movimientos del robot por fuerzas o perturbaciones extrañas al robot.

2.1.2 DIMENSIONES

Para las dimensiones de prototipo de robot delta se realizó una investigación respecto a trabajos similares, en la tabla 2.2 se indica las dimensiones de la estructura mecánica y se pueden apreciar en la figura 2.3

Tabla 2.2 Dimensiones del robot delta

Parámetros Geométricos	Valor
Longitud del brazo (La)	100 mm
Longitud del antebrazo (Lb)	419,73 mm
Radio del anillo móvil (r)	25,42 mm
Radio del anillo Fijo (R)	192 mm
Espacio entre brazos (e)	42,84 mm

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

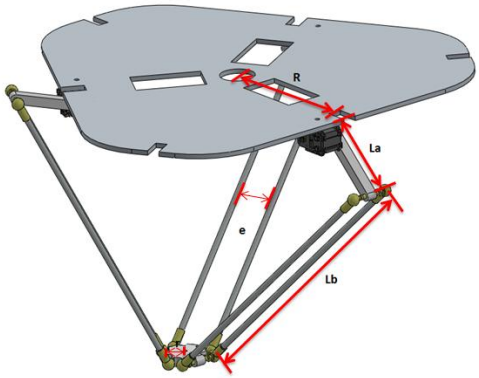


Figura 2.3 Dimensiones del robot delta

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

2.2. DISEÑO DEL SISTEMA MECÁNICO

A continuación se desarrolla el diseño de la estructura del robot en bases a los parámetros antes mencionado en el literal 2.1 para cada una de sus partes:

2.2.1. DISEÑO DE LA ESTRUCTURA DEL ROBOT DELTA

Estructura principal

La estructura principal es donde se encuentra todas las partes móviles del robot acopladas al servomotor, y el servomotor se mantiene estable a la plataforma fija.

El diseño de cada parte del robot son ideas a partir de robots ya realizados como el robot NUWAR desarrollado en la Universidad Western Australia, Robo Tenis desarrollado en universidad politécnica de Madrid y el robot Delta IRB 340 este robot es un robot industrial de estándar internacional fabricado por ABB.

Para este proyecto se utiliza los servomotores Dynamixel AX-12A que tienen por peso 55[gr], el torque de 15.3 [kg*cm] (Trossen Robotics, 2014)

El diseño de cada parte que conforma el robot se describe a continuación, las medidas reales de cada componente se encuentra en los Anexo B.

- **Plataforma fija:** Es la pieza de la figura2.4 donde se acoplan los servomotores y soporta el peso de los servomotores Dynamixel, brazos, antebrazos, la plataforma móvil y el efector final. Además esta parte se acopla a la base o estructura secundaria del robot para mantener estático al robot delta en funcionamiento.

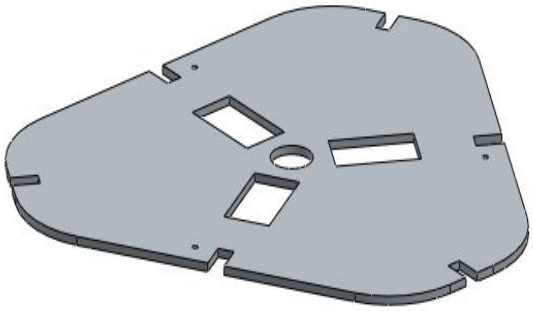


Figura 2.4 Plataforma Fija

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Frame F3** (Trossen Robotics, 2014): Estas piezas son propias del servomotor Dynamixel viene en cada servomotor en este proyecto sirve para adaptar los servos a la plataforma fija ver figura 2.5.

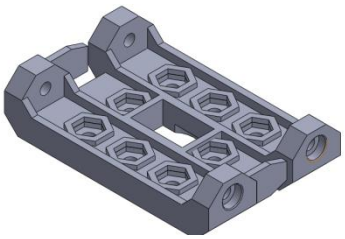


Figura 2.5 Frame F3

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Brazo:** Los brazos son las piezas que reciben los movimientos de los servomotores por ello deben tener un acoplamiento del brazo con los servomotores. Esta pieza debe ser ligera y debe tener resistencia a la deformación. Estas piezas también deben tener un acople para las juntas esféricas o ball joint que transmiten el movimiento al antebrazo. El brazo diseñado se muestra en la figura 2.6

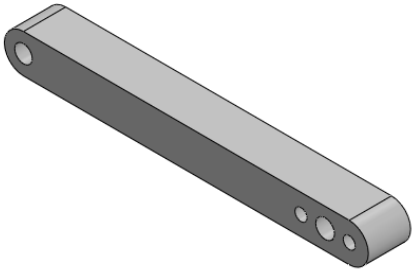


Figura 2.6 Brazo

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Plataforma móvil:** este componente es la figura 2.7 debe ser ligero tiene un hueco en el centro para el láser como efector final, es triangular para poder adaptarse a los antebrazos.

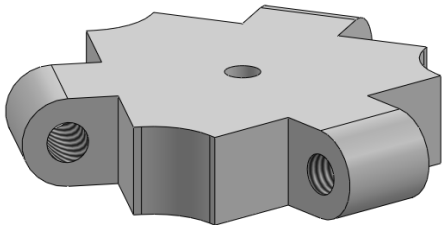


Figura 2.7 Plataforma Móvil

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Antebrazo:** Los antebrazos están conformados, 2 barras y 4 juntas esféricas o rotulas. Las juntas se unen a los brazos en la parte roscada al igual que a la plata forma móvil se unen en la parte roscada.

Juntas esféricas.- estas juntas no son fabricadas por la complicaciones que estas juntas, se encuentran en el mercado son fáciles de conseguir y existen de todo tamaño, igual son creadas en el software para el análisis de la estructura en la figura 2.8 se encuentra la junta esférica de medida 5 mm en plano se encuentra en el Anexo B.

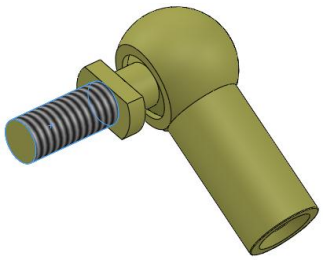


Figura 2.8 Junta esférica o rótula

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Barra: Las barras son las que se unen a las juntas esféricas deben ser ligeras y son las que se unen a la plataforma móvil ver figura 2.9.

Figura 2.9 Barra

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

El antebrazo quedaría como se ve en la figura 2.10 la parte roscada es la que se unen a la plataforma móvil y al brazo

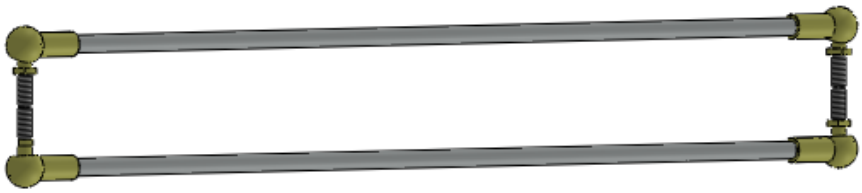


Figura 2.10 Antebrazo

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Estructura secundaria o Base

En los parámetros se explica que el robot debe ser firme es por eso que se lo ha hecho en acero por densidad mantiene al robot firme.

A continuación se describe las partes de la base:

- **Pata:** Ya que el robot puede describir trayectorias mayores a la plataforma fija no se las realizo en forma recta. Las patas son las que soportan a todo el robot ver figura 2.11

Figura 2.11 Pata

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Tensor:** Los tensores sirven para mantener el robot uniforme ya que estos están sometidos al peso del robot ver figura 2.12

Figura 2.12 Tensor

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Conector Delta**: sirve para fijar a los tensores y así mantener la uniformidad del robot ver figura 2.13

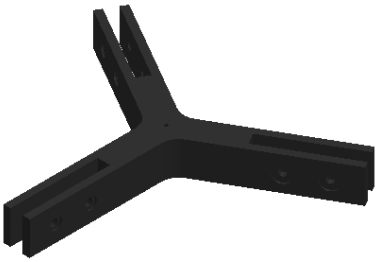


Figura 2.13 Conector Delta

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Ensamblaje final

Al tener todos los componentes ya podemos ensamblarlo el prototipo, como se muestra en la figura 2.14

Figura 2.14 Prototipo de robot delta Ensamblado

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

2.2.2. ANÁLISIS ESTÁTICO

En este trabajo se realiza el análisis del robot delta utilizando software ANSYS, con el fin de analizar el comportamiento estructural del robot Delta.

Para realizar el diseño y análisis se seguirá los siguientes pasos:

- **Realizar la Estructura del robot**

  La estructura y ensamblado se la realizó en el literal 2.2.1 con la ayuda del software solidwork.

- **Ejecutar Software Ansys e Importar La Estructura**

Una vez creado el diseño procedemos a ejecutar Workbench de ANSYS para el análisis estructural, e importamos la estructura del robot.

- **Definición Del Material**

El material a utilizar son aluminio para la estructura principal, acero para la base o estructura secundaria y plástico para los servomotores y el soporte del servomotor propio del servomotor.

En la tabla 2.3 se observa la lista de componentes con su respectivo material.

Tabla 2.3 Material de los elementos del robot delta

Elemento	**Cantidad**	**Material**
Plataforma Fija	1	Aluminio
Brazo	3	Aluminio
Rótulas	12	Aluminio
Barra	6	Aluminio
Plataforma Móvil	1	Aluminio
Pata	3	Acero
Tensor	3	Acero
Conector Delta	1	Acero
Frame F3	3	Plástico
Servomotores	3	Plástico

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Especificación De Restricciones Y Cargas**

Las cargas y restricciones son necesarias para definir el entorno de servicio del modelo.

Primero tenemos que definir el soporte fijo del robot delta como se ve figura 2.15 es la parte que soporta las partes móviles del robot

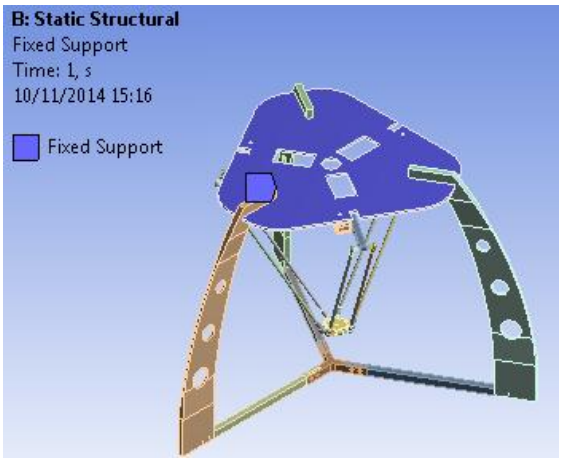


Figura 2.15 Soporte Fijo del Robot Delta

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Luego aplicamos la carga, en el robot la carga se pone en la plataforma móvil quien es el que lleva el efector final se le aplica una Fuerza de 15 [N], se aplica este valor ya que a futuro se puede cambiar de efector final, también porque puede existir fuerzas extrañas o perturbaciones al robot. Ver figura 2.16

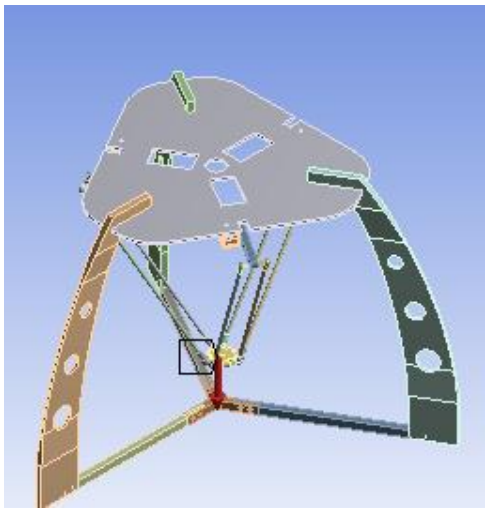


Figura 2.16 Fuerza en la plataforma móvil

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Ejecución del Estudio**

Antes de ejecutarse se realiza el mallado que es útil para el análisis del diseño ya que entre más fina es el mallado mejores resultados se obtendrá. Al tratarse de un prototipo se considera el mallado que nos da el software, en la figura 2.17 se ve el robot mallado.

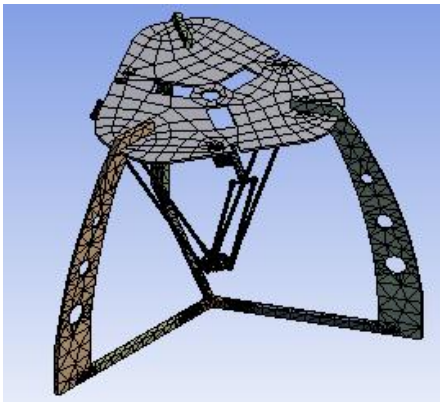


Figura 2.17 Robot Delta Mallado

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Finalmente se ejecuta el análisis, los resultados que se tomara en cuenta son:

- Deformación Total
- Tensión Equivalente (Von-Mises)
- Factor de Seguridad

2.2.3. RESULTADOS OBTENIDOS

- **Deformación Total:** En la figura 2.18 se indica donde tendrá desplazamientos, con un valor máximo de 0.15mm

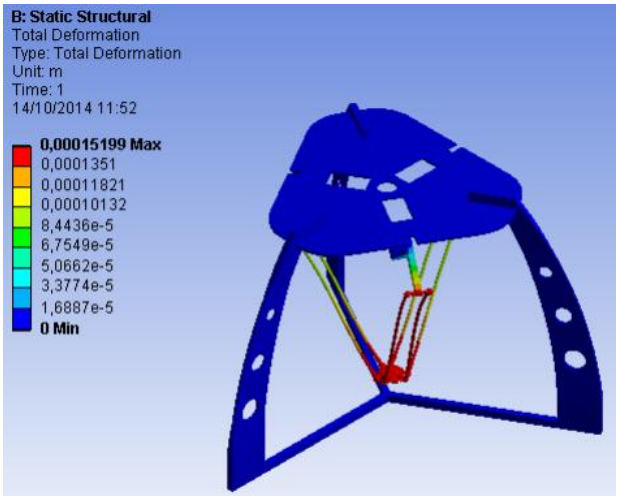


Figura 2.18 Desplazamientos

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Tensión Equivalente:** En la figura 2.19 nos indica donde fallaría la estructura la parte color roja, en este caso el valor máximo de análisis es de 4,25 [Pa].

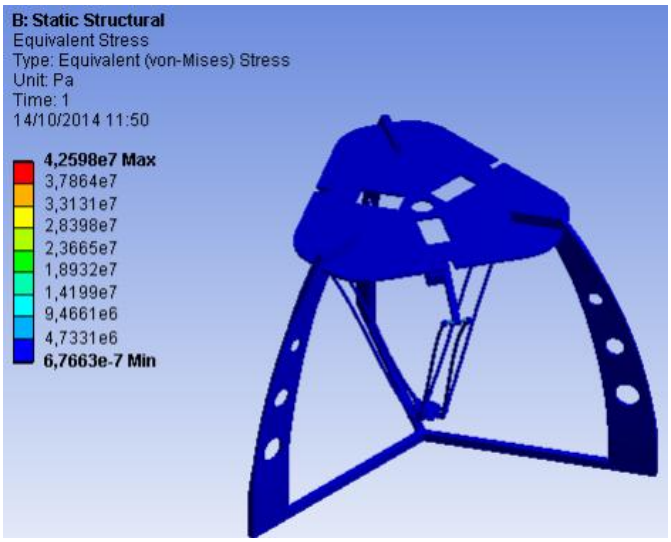


Figura 2.19 Tensiones De Von Mises Del Robot Delta

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

- **Factor de Seguridad:** Por últi m o el la figura 2.20 se indica una escala de cómo se reparte el factor de seguridad. En la siguiente figura nos indica un valor mínimo de factor de seguridad de 2,04 este valor es óptimo para nuestro diseño, finalmente se concluye que la estructura resiste y no tiene fallos de seguridad estructural con el material asignado.

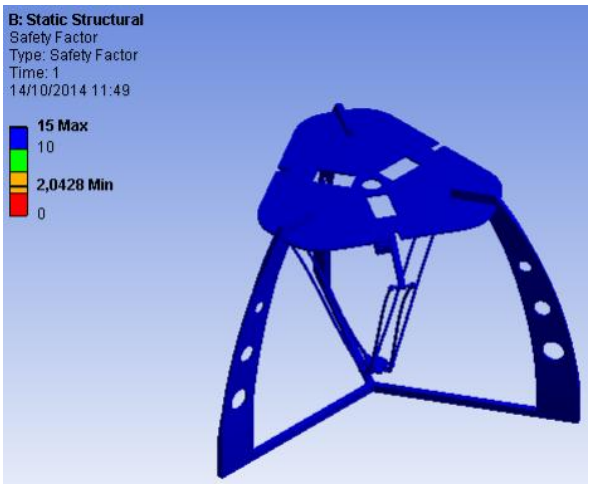


Figura 2.20 Factor de Seguridad Robot Delta

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

2.3. MONTAJE DEL SISTEMA MECÁNICO

Una vez concluido con el diseño y análisis se procede a la construcción de los componentes del robot delta.

2.3.1. COMPONENTES DEL PROTOTIPO

Todas las piezas como tales fueron fabricadas en máquinas convencionales. En las figura 2.21, figura 2.22 y figura 2.23, se muestra los componentes maquínanos del prototipo de robot delta con su respectivo material.

Figura 2.21 Piezas Robot Delta: Plataforma Móvil y Pata

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

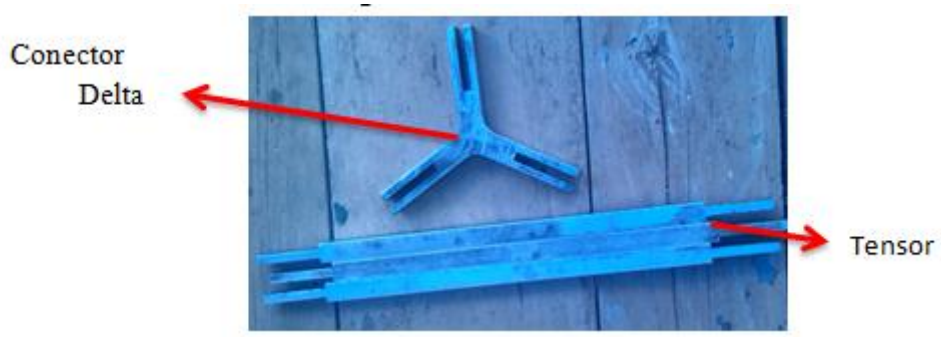


Figura 2.22 Piezas Robot Delta: Conector Delta Tensor

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

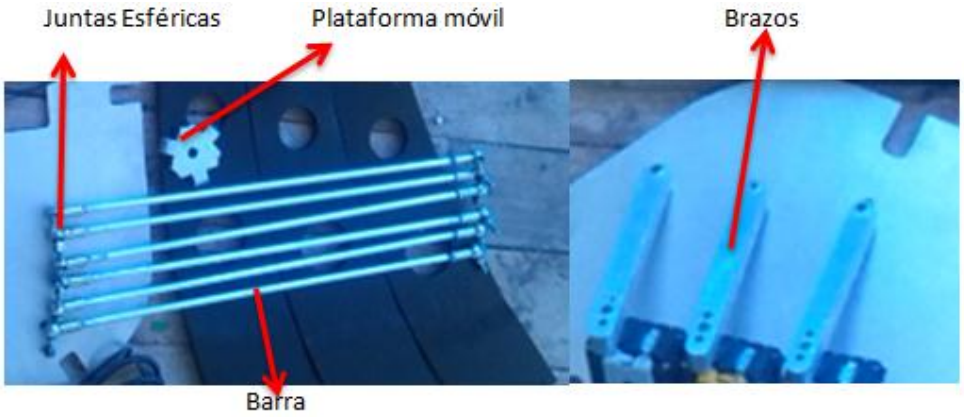


Figura 2.23 Piezas Robot Delta: Juntas Esféricas, Plataforma móvil Barra y Brazos

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Para mayor información sobre los componentes en el Anexo B se especifica las medidas a escala de cada pieza del robot delta.

2.3.2. Ensamble del Robot Delta

Ya construidas todos los componentes del robot y analizado su calidad procedemos a ensamblar, las piezas pueden ser montadas y desmontadas.

En la tabla 2.4 se indica la lista de componentes construidas por maquinas convencionales, y el número de piezas necesarias.

Tabla 2.4 Componentes Construidos del Robot Delta

Pieza	Cantidad
Plataforma Fija	1
Plataforma Móvil	1
Brazo	3
Barra	6
Pata	3
Tensor	3
Conector Delta	1

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

En la tabla 2.5 se indica los componentes utilizados para el ensamblaje para el acople de los servomotores Dynamixel y los componentes necesarios que viene en conjunto con estos servomotores.

Tabla 2.5 Componentes necesarios Dynamixel

Pieza	Cantidad
Servomotores Dynamixel	3
Tornillos S1	18
Frame F3	3

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Por último en la tabla 2.6 se indica los componentes necesarios para la sujeción de los diferentes componentes que pueden ser comprados en cualquier ferretería.

Tabla 2.6 Piezas Varias

Pieza	Cantidad
Tornillos con cabeza ranurada de 5/32”x1”	12
Perno 5/16”x1” rosca milimétrica	3
Perno ¼”x 1”	3
Tuerca 5/32”	12
Tuercas ¼”	3
Rotulas con Norma DIN 71802	12

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

A continuación describimos el ensamble de cada una sus partes:

1. En la figura 2.24 se presenta el ensamble de la base del robot primero tenemos que montar el conector delta con los tensores, para ellos utilizamos el conector delta, los tres tensores que se unen por medio de 6 tornillos con cabeza ranurada de 5/32”x1” y 6 tuercas 5/32”.

Figura 2.24 Conector Delta con Tensores

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

2. A continuación colocamos las patas, las patas tiene una ranura en la parte inferior la cual se monta en los tensores. Se necesita 3 patas y 6 tornillos con cabeza ranurada de 5/32”x1” y 6 tuercas 5/32” para asegurar las patas a los tensores, ver figura 2.25.

Figura 2.25 Colocación de las patas

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3. La siguiente pieza a colocar es la plataforma fija que va en la parte superior de la base, se asegura con pernos 5/16”x1”, como recomendación se puede colocar rodelas de presión 5/16” para ejercer mayor presión sobre la tuerca o cabeza del perno el momento de ser ajustada. Se necesita la Plataforma fija 3 pernos y 3 rodelas de presión, No se necesita tuercas ya que la patas en la parte superior el agujero se le paso el machuelo para abrir rosca y conectarla al perno sin necesidad de tuercas ver figura 2.26.

Figura 2.26 Colocación de la plataforma fija

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

4. El paso siguiente es ensamblar el brazo y el antebrazo para cadena cinemática. Es importante este ensamble ya que es la parte que trasmite al movimiento al efector final del robot.

   Primero tenemos que ensamblar las rotulas o juntas esféricas con las barras, se necesita 12 juntas esféricas o rotulas las cuales deben ser engrasadas adecuadamente para que la fricción sea mínima y no restringa el movimiento generando cargas innecesarias al robot, y las 6 barras a ser unidas. Las barras tienen a sus extremos rosca la cual permite la unión a las juntas esféricas, el ensamble de estos componentes es importante se debe fijar que queden todos simétricos como se ve en la figura 2.27

Figura 2.27 Ensamble Rotulas-Barras

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

5. El siguiente paso es ensamblar al brazo, el brazo tiene un orifico machuelado que permite que la rosca macho de la junta esférica se una con el brazo se necesita 3 brazos a cada brazo se une 2 juntas formando así el brazo y antebrazo del robot como se ve en la figura 2.28

Figura 2.28 Ensamble Brazo-Antebrazo

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

6. Ahora se ensambla el antebrazo a la plataforma móvil. La plataforma móvil al igual que los brazos tiene tres orificios machuelados para la unión de los antebrazos, en la figura 2.29 se indica el ensamble a la plataforma móvil.

Figura 2.29 Ensamble Antebrazo-Plataforma móvil

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Para acoplar y fijar el servomotor tanto al brazo y a la plataforma fija respectivamente se utiliza componentes que vienen junto con los servomotores. La figura 2.30 muestra las piezas que viene con cada servomotor.

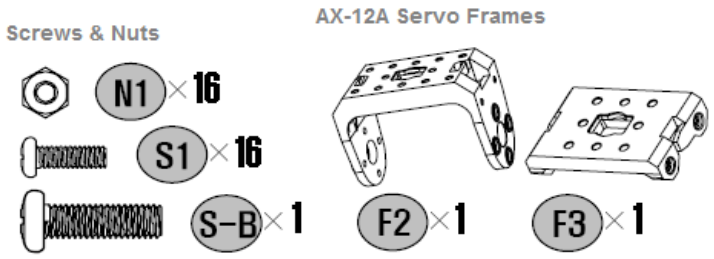


Figura 2.30 Elementos Incluido en los Servomotores Dynamixel AX-12A

Fuente: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

7. El procedimiento previo al acople del servomotor y al brazo se colocan en los tres servomotores las tuercas N1 que sujeta al servomotor a la plataforma fija en la figura 2.31 muestra cómo va ubicadas las tuercas en cada uno de los servomotores

Figura 2.31 Disposición de tuercas en los servomotores ID3 ID6 y ID9

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

8. Ya los servomotores con las tuercas introducidas colocamos el frame F3. En el frame F3 debe estar introducido un perno de1/4”x1”, y lo ajustamos con los tornillos s1 para fijarlo al servomotor en la figura 2.33 se muestra el ensamble.

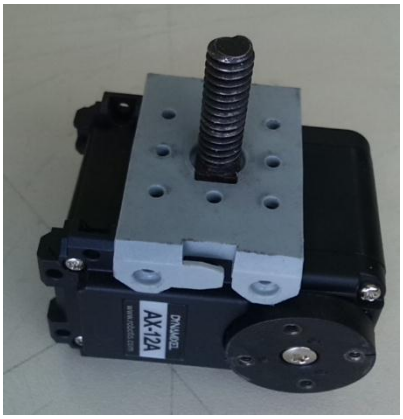


Figura 2.32 Ensamble Servomotor-Perno-Frame F3

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

9. Como siguiente paso tenemos que acoplar el servomotor al brazo para esta fijación se utiliza los tornillos S1, para la correcta posición del brazo con el servomotor debemos tomar en cuenta que el servomotor tiene ranuras en la rueda que trasmite el movimiento una ranura en la parte superior y dos ranuras en la parte inferior, para la correcta colocación

tomamos en cuenta la parte de la rueda donde este solo una ranura y colocar el extremo del brazo al servomotor como se ve en la figura 2.33.

Figura 2.33 Ensamble servomotor-brazo

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

10. Finalmente fijamos los servomotores a la plataforma fija, introducimos los pernos que ya están acoplados a los servomotores, en los orificios de la plataforma fija y lo aseguramos con las tuercas de 1/4" uno por uno ver figura 2.34.

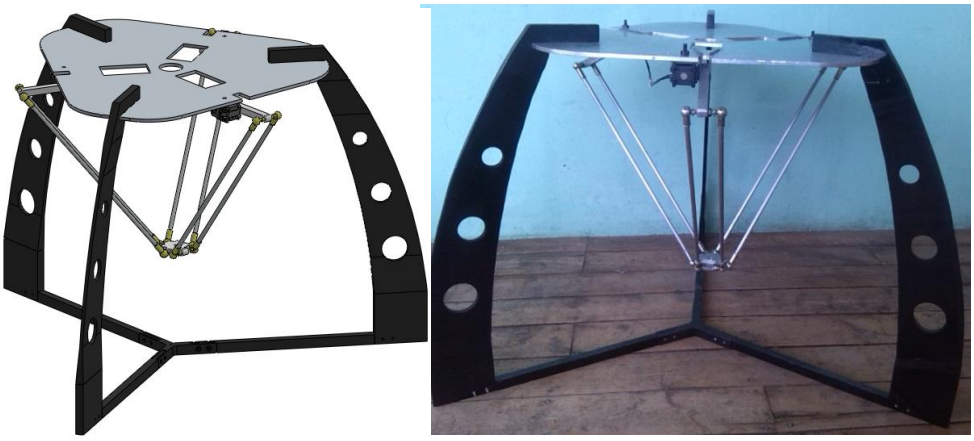


Figura 2.34 Ensamble final del robot Delta

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

CAPÍTULO III

IMPLEMENTACIÓN Y DESARROLLO DEL SISTEMA DE CONTROL

3.1 TECNOLOGÍAS EMPLEADAS

3.1.1 SISTEMA OPERATIVO

En la actualidad ROS funciona sobre plataformas tipo Unix, está mayoritariamente probado en sistemas Linux y Mac OS X y oficialmente soportado para determinadas versiones de Ubuntu.

Por su popularidad y compatibilidad con el sistema ROS, todo el desarrollo del proyecto se realiza sobre el sistema operativo Ubuntu en su versión 14.04 LST (Trusty Tahr).

3.1.2 PLATAFORMA DE DESARROLLO

Robot Operating System (ROS), propone la utilización del framework ROS como herramienta software principal, ya que ofrece diferentes herramientas y librerías para el desarrollo de sistemas robóticos.

La idea principal de ROS es la comunicación de diferentes programas, también conocidos como nodos, entre sí para generar estructuras de nodos más complejas que ejecuten numerosas tareas. ROS consta de múltiples herramientas que asisten al programador en su tarea, tales como la herramienta de visualización y simulación Rviz. Ver la figura 3.1, que muestra modelos virtuales de robots.

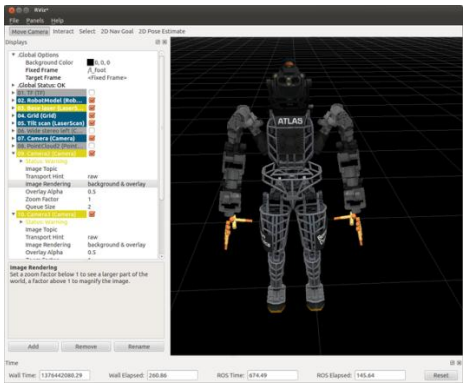


Figura 3.1 Pantalla del visor Rviz.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.1.3 BIBLIOTECA

OpenCV (Open Source Computer Vision Library) es una biblioteca de libre uso para aplicaciones en tiempo real de visión por computador. Fue creado por un grupo de investigación de Intel en desarrollo de un proyecto para la creación de aplicaciones de uso intensivo de la CPU.

La biblioteca implementa casi todos los algoritmos de visión por ordenador que se pueda imaginar, además trabaja perfectamente con ROS.

3.1.4 LENGUAJES DE PROGRAMACIÓN

Para el desarrollo del proyecto se ha utilizado el lenguaje de programación Python muy común en el campo de la robótica. Este tipo de programación se ha optado por diversos motivos: Se trata de un lenguaje interpretado o de script, multiplataforma, y orientado a objetos.

Python incorpora un entorno de ejecución que detecta con exactitud errores en el código. De manera, que la depuración del código se realiza en el mismo entorno de programación. La sintaxis es muy limpia y favorece a la legibilidad del código. Se ahorra mucho tiempo a la hora de ejecutar el código ya que no hace la acción de compilar.

3.1.5 HERRAMIENTAS UTILIZADAS

a) Interprete de comandos

Al trabajar sobre Ubuntu y ROS es necesario familiarizarse con la terminal del sistema ya que todo se maneja mediante uso de comandos. Esto facilita el movimiento entre directorios, ejecución de programas, cambio de permisos, mostrar variables del sistema, etc.

Debido a que es necesario utilizar diferentes comandos Linux y/o ROS simultáneamente y observar los numerosos resultados se ha optado por utilizar un terminal más sofisticado que el que trae por defecto la distribución de Ubuntu.

Este intérprete de comandos tiene el nombre de Terminator y se encuentra en los repositorios de Ubuntu.

Como se puede observar Terminator permite dividir una ventana de comandos en numerosas sub-ventanas de diferentes tamaños y posiciones, ver figura 3.2.

Figura 3.2 Intérprete de comandos Terminator.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Esto hace que sea mucho más sencillo contrastar datos, lanzar varias aplicaciones a la vez y capturar resultados. Por todo ello se ha convertido en una herramienta indispensable el manejo de comandos en ROS/Linux.

b) Entorno de desarrollo integrado

Existe gran cantidad de entornos de desarrollo (IDE) compatibles con Ubuntu Genéricos, dedicados a lenguajes específicos, más o menos pesados, con una interfaz amigable, etc.

Dentro de este gran abanico de posibilidades se ha elegido uno adaptado a nuestras necesidades, Gedit Text Editor ver Figura 3.3.

Con mayor similitud a un editor de texto que a un entorno de desarrollo, gedit ha sido la decisión última ya que es un entorno ágil, simple y dinámico.

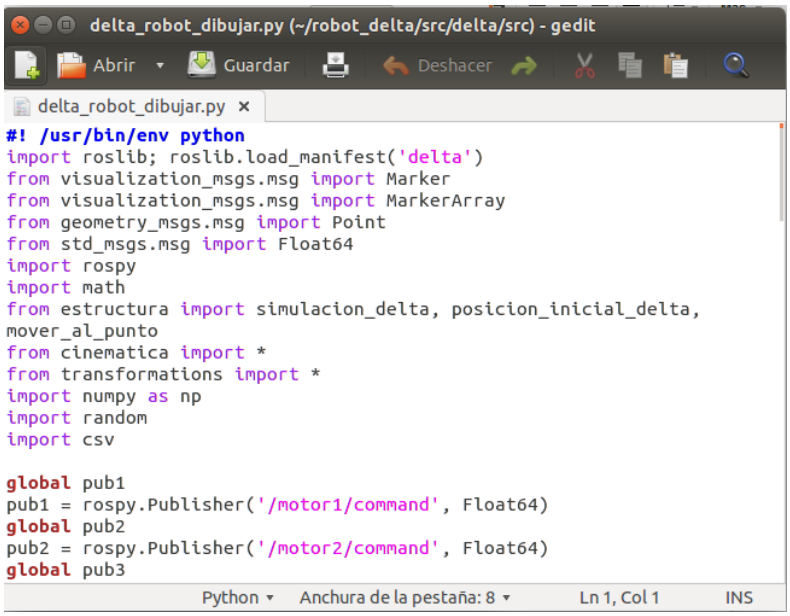


Figura 3.3 Entorno de programación Gedit.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.2 INSTALACIÓN DEL SOFTWARE DE CONTROL

La instalación de ROS, creación del espacio de trabajo y sus paquetes, disponibles en el Manual de Usuario.

3.3 PREPARACIÓN DEL ENTORNO DE TRABAJO

3.3.1 CREACIÓN Y USO DEL ESPACIO DE TRABAJO ROS

Es conveniente y para mayor comodidad crear un espacio personal o Overlay donde experimentar con paquetes ROS o crear paquetes propios, ya que ROS tiene por defecto como directorio para paquetes /opt/ros/indigo/stacks. Se pueden crear los nuevos stacks o packages en este directorio por defecto, si es que se lo gestiona adecuadamente. En diagrama 3.1 se encuentra el proceso de creación de un Overlay.

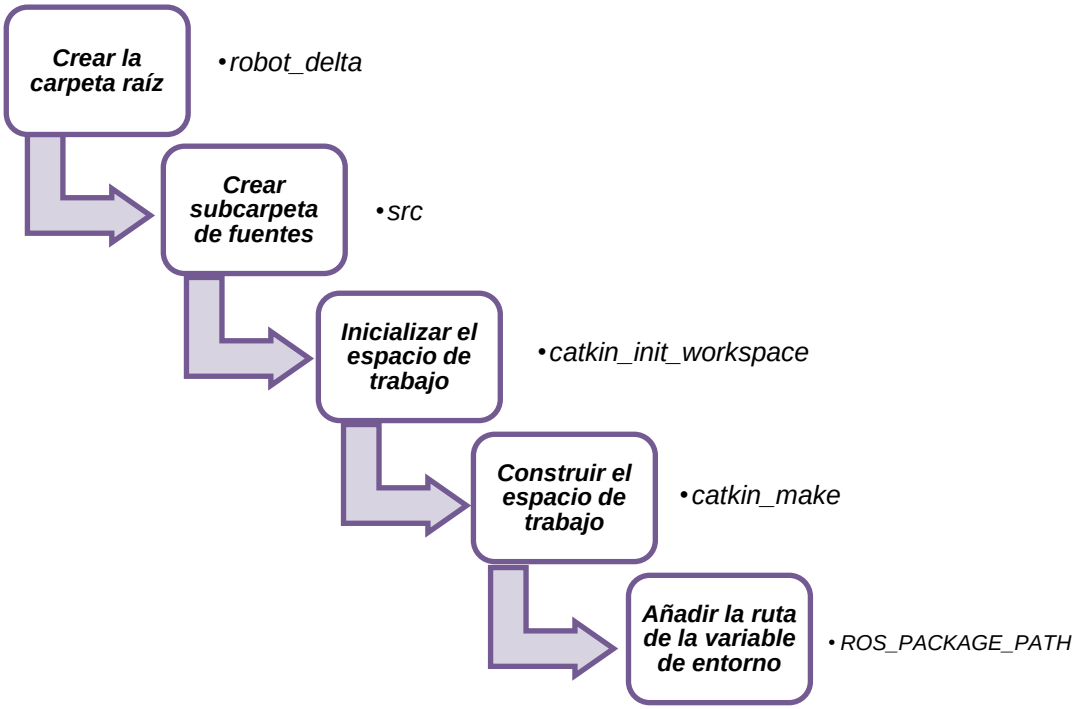


Diagrama 3.1. Proceso para crear un espacio de trabajo con ROS.

3.4 CREACIÓN DE PAQUETES ROS

Para crear un paquete catkin se realiza el siguiente proceso, ver diagrama 3.2.

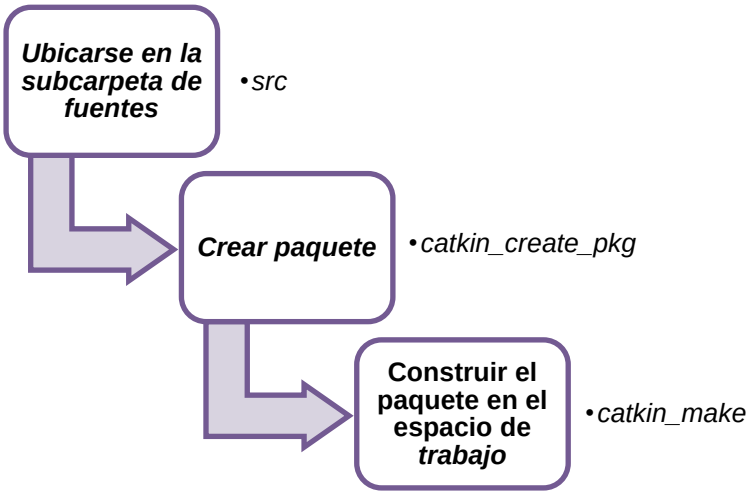


Diagrama 3.2. Proceso para crear un paquete ROS.

3.5 DISEÑO DE LA APLICACIÓN

La aplicación corresponde a un simulador para un robot Delta dado. En forma paralela se incluye el sistema que permita hacer un dibujo y que el robot reproduzca este dibujo con diodo láser instalado en el efector final del robot siendo capaz de cortar o grabar en materiales suaves. También se incluye el sistema que reproduce e l simulador de los movimientos del robot real.

3.5.1 SIMULACIÓN EN ROS

Para controlar el robot utilizamos las ecuaciones de la cinemática directa e inversa y que eestán desarrolladas en el capítulo 1, en cuanto a la simulación se utiliza la herramienta de visualización 3D Rviz de ROS.

Se ha optado por utilizar Markers, estos son formas simples (flechas, cubos, bolas, texto, etc.) ver Figura 3.4 representan los componentes del robot.

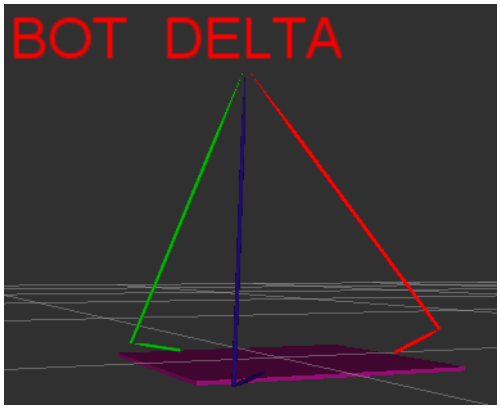


Figura 3.4 Salida gráfica del robot.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Para trabajar en ROS se construyó un espacio de trabajo donde se crea un paquete y se programa los scripts, el primer script contiene la cinemática del robot delta “cinematica.py”, el siguiente script se llama “construccion_delta.py” archivo que consta de 3 funciones: crear_simulacion() la cual se encarga de crear el modelo del robot, colocar_pisicion_inicial() poner una posición inicial al modelo y para poder mover el modelo a las posiciones deseadas, está también la función mover_simulacion_al_punto (x, y, z, simulationMarkerArray) la cual se usa para actualizar la posición de todo el modelo dejando el elemento terminal donde se cruzan los 3 brazos en el punto deseado.

En este mismo script también importamos la librería ya creada “cinematica.py”, con esto es posible aplicar cinemática inversa para conocer la configuración total del robot de acuerdo a la posición a alcanzar.

3.5.2 SOFTWARE PARA EL PROCESAMIENTO DE IMÁGENES

Para la creación de la aplicación se utiliza el software para procesamiento de imágenes OpenCV.

a) Software para rasterizar.

Un control deslizante *width* permite al usuario cambiar la resolución de salida hasta que el aliasing[2] se vea bien, y un segundo control deslizante permite al usuario cambiar el valor *gamma* del valor predeterminado de 2,2 para cambiar el brillo de una imagen.

Figura 3.5. Slider de control.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

En la pantalla puede mostrar los 256 valores de escala de grises figura 3.6 b), el robot sólo puede imprimir un rango mucho menor de escala de grises, que varía en función del tamaño de píxel, como se indica en la figura 3.6 c)

Esto puede producir variaciones en la visualización de las regiones de color más claro, y para las regiones de color más oscuro, donde las regiones aparentemente tienen diferentes colores son a la vez el mismo sombreado.

Para solucionar esto, los valores de los píxeles de salida se agrupan en cubos basados en el número de garabatos[3] en ese píxel. De esta manera, la imagen de salida tiene el mismo número de niveles de escala de grises como la imagen impresa. Con

[2] ***Aliasing***: Es un fallo en la representación de los gráficos, efecto que consiste en la presencia de dientes de sierra en los bordes de los polígonos. El efecto de aliasing da a las imágenes una apariencia tosca.

[3] **Garabatos:** Al no poder generar niveles de escala de grises con un lápiz debido a que se crea una línea de constate oscuridad se opta por crear garabatos, es decir se dibujan funciones triangulares en los pixeles, creando niveles enteros de ondas triangulares con diferentes fases.

un poco de ajuste del control deslizante los colores que aparecen en pantalla están más cerca de los colores que se mostraran en la impresión.

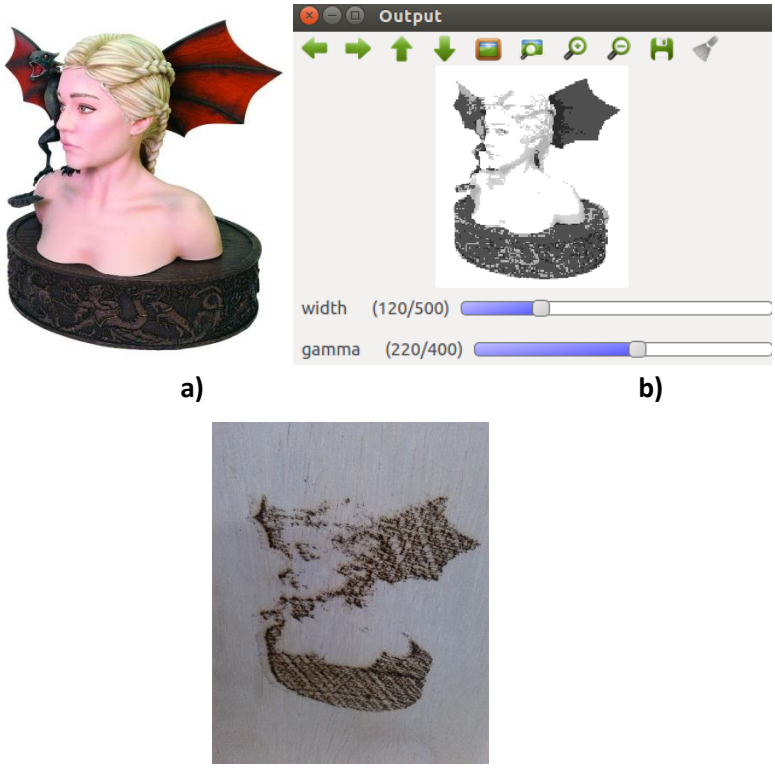


Figura 3.6 a) Imagen Original. b) Escala de 120 pixeles y gamma de 2,2 (220/100). c) Imagen impresa por el robot.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Los siguientes son capturas de pantalla del programa en acción. En la figura 3.7 se puede ver lo que hay significativamente en menos de 256 niveles de gris.

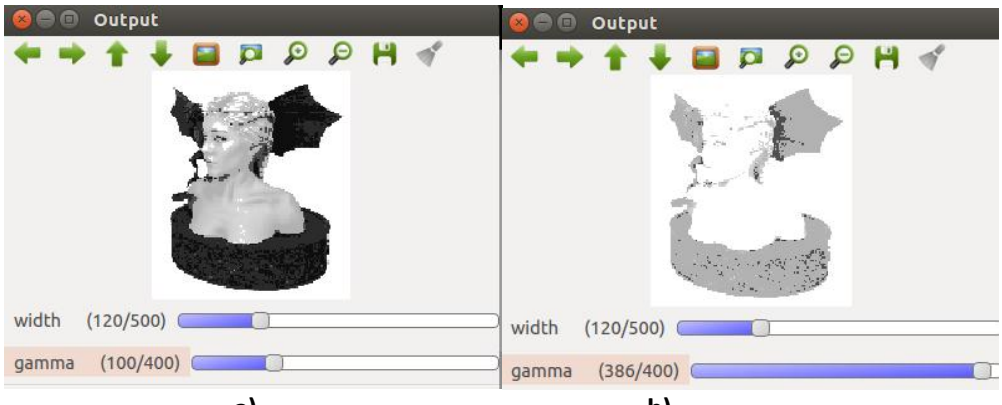


Figura 3.7 Imagen rasterizada. a) Imagen más oscura. b) Imagen más brillante.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

b) Software para vectorizar.

Para el vectorizador de imágenes, se utiliza: detección de bordes de Canny, detección contornos, y la aproximación poligonal de curvas.

- **Adquirir imagen y cambiar el tamaño.**

Al escoger la imagen de origen puede resultar demasiado grande, por ejemplo mayor de 1200 píxeles de ancho, una imagen así no tiene demasiados detalles. Dado que los algoritmos de procesamiento de imágenes se ejecutan sólo una vez, aunque en la velocidad de procesamiento no hay ningún problema, el robot delta no puede trazar tan rápido.

Y si la imagen original es demasiado pequeña, presenta baja resolución. Las posiciones del manipulador se cuantifican sobre la base de la resolución de la imagen de entrada, así que se ha cambiado el tamaño de la imagen, debe ser alrededor de 800 a 1200 pixeles. Así que incluso si la imagen de entrada es pequeña, cambiar el tamaño a una imagen más grande da mejores resultados.

Por último, pasar la imagen a escala de grises, mediante el uso de cv2.cvtColor

- **Filtro y detección de bordes.**

Para obtener una imagen binaria, sin ruido y con todos los bordes y líneas detectados se utiliza cv2.Gaussianblur como filtro y cv2.Canny para detectar los bordes.

Al usar Canny hay que tomar en cuenta las variables para ajustar la histéresis para el umbral alto y bajo, (lowThreshold, highThreshold)

- **Encontrar los contornos y el polígono de aproximación.**

La imagen obtenida presenta todos los bordes, son píxeles sólo blancos en una imagen en negro. Ellos necesitan ser agrupados en las líneas. Aquí es donde la función cv2.FindContours es de utilidad, esta función encuentra todos los grupos de píxeles que pueden convertirse en contornos y los convierte en contornos unificados estos contornos son almacenado como en una jerarquía, que muestran que los contornos se encuentran dentro de otros contornos.

En este caso, queremos que todos los contornos en el orden que sea, pasarlos a cv2.CV_RETR_LIST para obtener una lista de los contornos, ahora estos contornos no son líneas rectas agradables que se pueden aproximar linealmente, el último paso es entonces aplicar cv2.ApproxPoly, que es, aproximación poligonal, darle la lista de los contornos, y le va a devolver una lista de polígonos y una lista de puntos. Exactamente lo que necesita el robot.

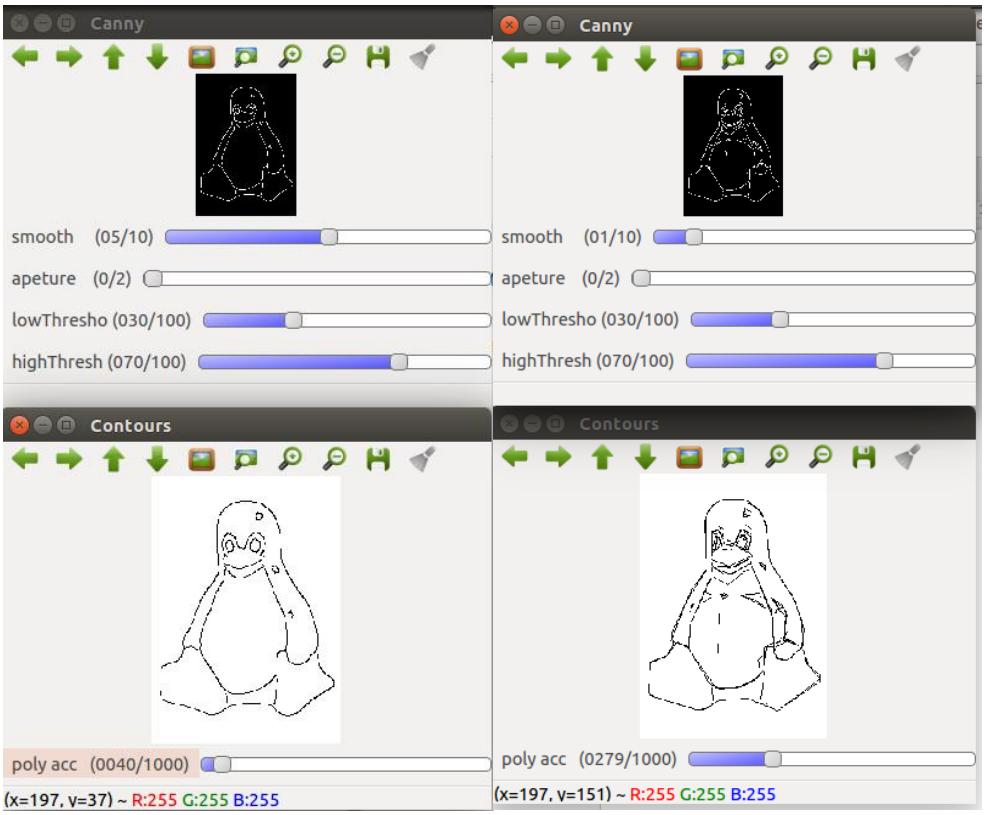


Figura 3.8. Imagen Vectorizada.
Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

La interfaz gráfica del software para rasterizar y vectorizar está construida para procesar automáticamente la imagen seleccionada cada vez que el usuario modifica un parámetro de procesamiento.

Los parámetros width, gamma, smooth, apeture, lowThreshold, highThreshold inciden directamente en la velocidad de la simulación del robot al momento de dibujar; por ejemplo, ajustando los parámetros de tal forma de obtener más calidad, se trataran más segmentos y tomara más tiempo dibujarlos.

3.5.3. CREACIÓN DE LA APLICACIÓN.

Para crear esta aplicación utilizamos el paquete de interfaz de ROS con OpenCV, dentro del paquete delta también hay tres scripts llamado "vectorizar.py", "rasterizar.py" en el script "drawing.py" se importa los archivos vectorizar y rasterizar y se programa para que el usuario elija una imagen y pueda procesarla, los puntos de la imagen ya procesada se guardan en un archivo de texto llamado "Ptos_Dibujo.txt". El otro script es "robot_delta_dibujador.py" se encarga de colocar cada uno de los puntos del archivo "Ptos_Dibujo.txt" en la posición deseada dando como resultado una imagen, en este mismo script se puede variar la escala "scale" de estos puntos que facilita la calidad de impresión de la imagen, además dentro de este archivo se puede variar el tamaño de la impresión, el proceso de la aplicación se puede visualizar en la figura 3.9.

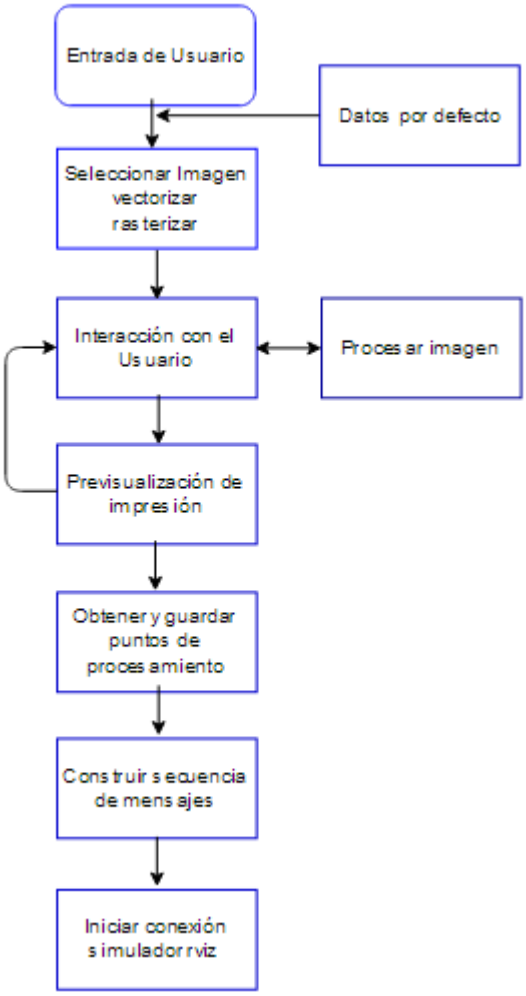


Figura 3.9 Proceso de la aplicación modo dibujo.
Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.5.4 ROBOT MODO PASIVO

El siguiente script es para que el robot trabaje en modo pasivo es decir que permite visualizar los movimientos del robot real en la simulación.

En la Figura 3.10 se encuentra el proceso de aplicación modo pasivo.

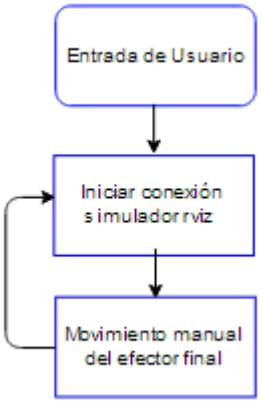


Figura 3.10 Proceso Aplicación modo pasivo.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.6 INTEGRACIÓN ROS CON ACTUADORES DYNAMIXEL

Existe soporte ROS para estos actuadores mediante el paquete dynamixel_motor.

Para instalar este paquete pasar al Manual de Usuario, ya con el paquete instalado utilizamos: tres actuadores AX-12A, un SMPS2Dynamixel Adapter y el controlador USB2Dynamixel.

Para comprobar la integración con ROS hay que seguir el siguiente procedimiento:

3.6.1 CONFIGURACIÓN DE LOS ACTUADORES DYNAMIXEL Y ACCESO AL BUS

Asegurarse de que los actuadores tengan identificadores distintos, ya que por defecto tienen asignado el ID 1. Se puede modificar el ID utilizando la utilidad Dynamixel Wizard, incluido en el software RoboPlus (windows) o bien utilizando la librería de código abierto de software Robotis servo, este paquete incluye dos

bibliotecas independientes (lib_robotis.py & usbscan.py) para consultar y controlar Robotis Dynamixel Servos. En este caso se ha optado por el segundo método. Y la única dependencia es pyserial para las comunicaciones.

3.6.2 INSTRUCCIONES PARA CAMBIAR EL ID DE LOS SERVOS DYNAMIXEL.

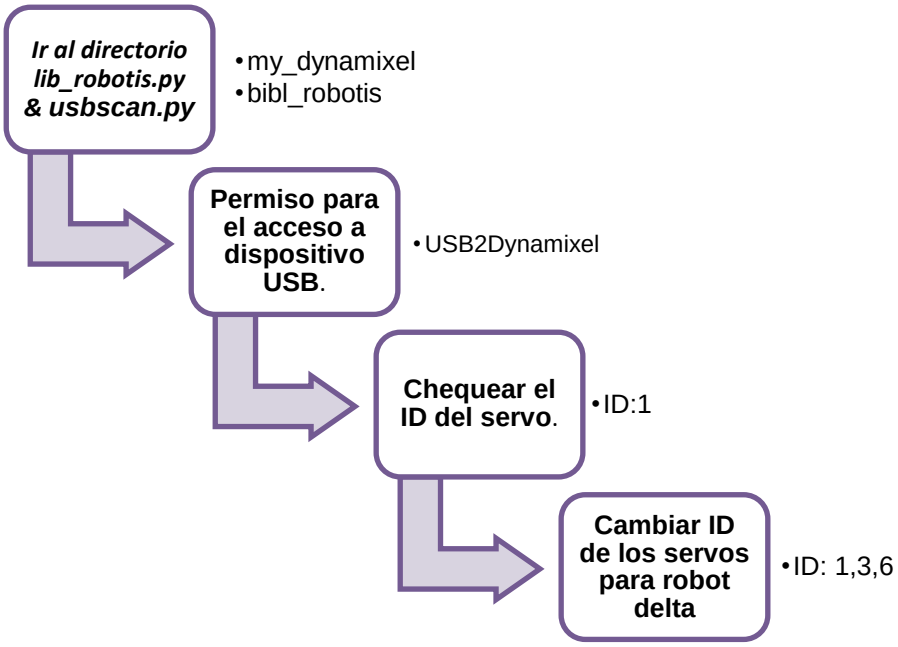


Diagrama 3.3. Proceso para cambiar el ID de los servos.

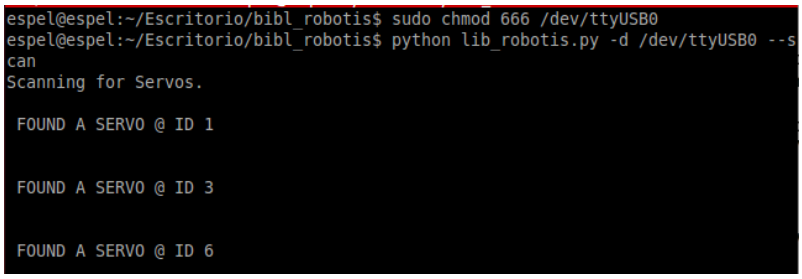


Figura 3.11 Escaneo de Ids de los servos Dynamixel

Elaborado por Tumbaco, Diana y Quimbita, Wilmer

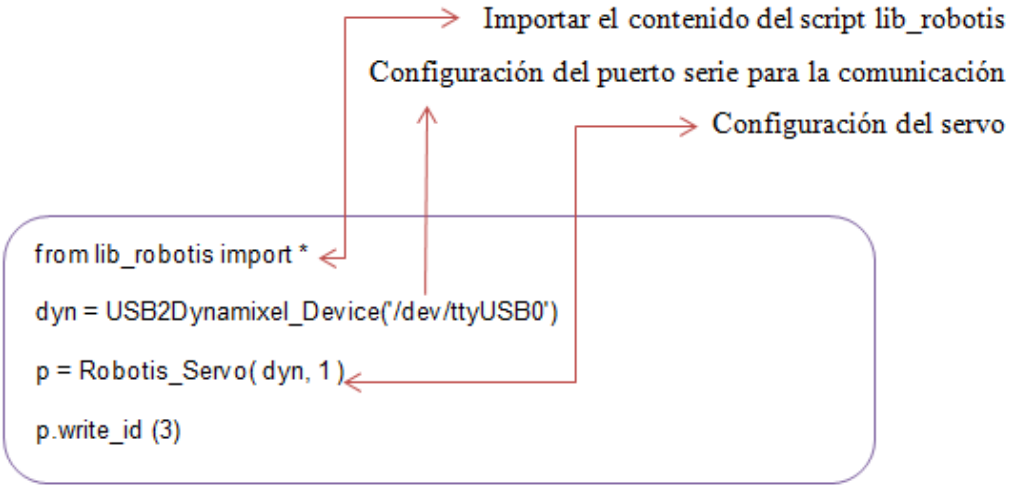


Figura 3.12 Instrucciones para cambios de Id.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.6.3 ACCESO AL BUS Y VISUALIZACIÓN DE DATOS

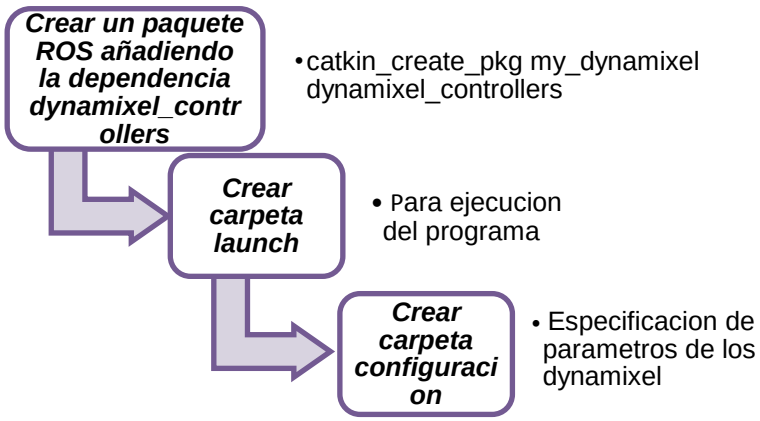


Diagrama 3.4. Proceso para acceder a los dynamixel.

– **Directorio launch.**

a) controller_manager.launch, ver Figura 3.13, es un archivo de ejecución que contiene todos los parámetros necesarios de los 3 actuadores AX-12A con identificaciones 1 al 6, los cuales están conectados a /dev/ttyUSB0 puerto serie.

```
<launch>
    <!-- Start the low-level driver manager with parameters -->
    <node name="ttyUSB0_manager" pkg="dynamixel_controllers" type="controller_manager.py"
output="screen">
        <rosparam>
            namespace: dxl_manager
            serial_ports:
                delta_port:
                    port_name: "/dev/ttyUSB0"
                    baud_rate: 1000000
                    min_motor_id: 1
                    max_motor_id: 6
                    update_rate: 20
        </rosparam>
    </node>
</launch>
```

Paquete ROS Indigo Dynamixel motor

Puerto del controlador USB2Dyamixel

ID de los servos van 1 al 6

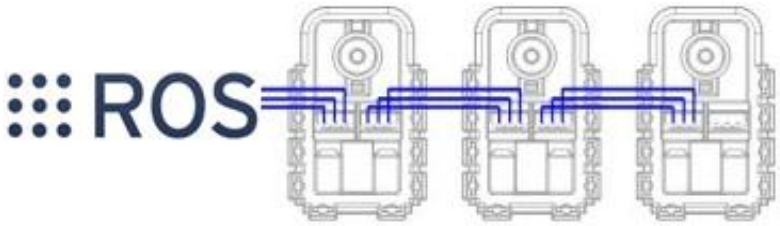
Por defecto 20 Hz

Figura 3.13 Parámetros de funcionamiento de los dynamixel con ROS.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

b) Ejecutar el archivo launch recién creado.

```
$ roslaunch my_dynamixel controller_manager.launch
```




```
[INFO] [WallTime: 1412176818.408901] delta_port: Pinging motor IDs 1 through 6..
.
[INFO] [WallTime: 1412176818.805261] delta_port: Found 3 motors - 3 AX-12 [1, 3,
 6], initialization complete.
```

Figura 3.14 Comunicación de ROS con los servos dynamixel.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

– **Directorio Configuración.**

Aquí se va especificar los parámetros de los servos dynamixel, los archivos "motors.yaml" y "motors _torque_off.yaml" que al ser creado, lee la configuración de los 3 servomotores que para las dos aplicaciones, solo se diferencia en que el caso del modo pasivo los servomotores no tienen par activado y en modo dibujador sí.

a) *motors.yaml*

En el archivo motors.yaml ver figura 3.15 primero definimos los parámetros motor1, motor2, motor3, este para enumerar los nombres de

los servos. A continuación se especifica el tipo de controlador que controlara cada servo así como su identificación de hardware, el valor de la posición inicial y sus valores mínimos y máximos de posición.

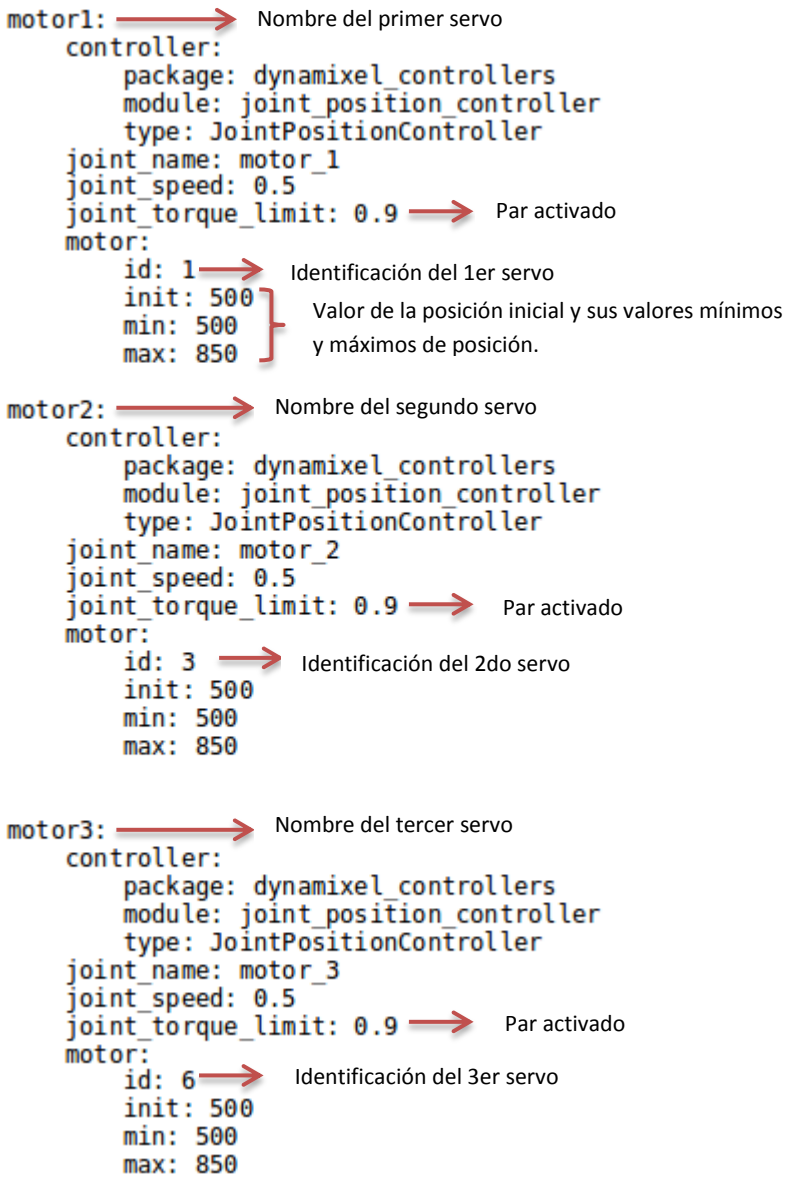


Figura 3.15 Parámetros con torque activado para modo_delta_dibujador.launch.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

b) *motors_torque_off.yaml*

El archivo motors_torque_off.yaml ver en la figura 3.16, es igual que archivo motors.yaml constan de los mismos parámetros pero se diferencia que el par esta desactivado y permite controlar manualmente los servomotores.

```
motor1:
    controller:
        package: dynamixel_controllers
        module: joint_position_controller
        type: JointPositionController
    joint_name: motor_1
    joint_speed: 2
    joint_torque_limit: 0.0 ──→ Par Desactivado
    motor:
        id: 1
        init: 500
        min: 500
        max: 850

motor2:
    controller:
        package: dynamixel_controllers
        module: joint_position_controller
        type: JointPositionController
    joint_name: motor_2
    joint_speed: 2
    joint_torque_limit: 0.0 ──→ Par Desactivado
    motor:
        id: 3
        init: 500
        min: 500
        max: 850


motor3:
    controller:
        package: dynamixel_controllers
        module: joint_position_controller
        type: JointPositionController
    joint_name: motor_3
    joint_speed: 2
    joint_torque_limit: 0.0 ──→ Par Desactivado
    motor:
        id: 6
        init: 500
        min: 500
        max: 850
```

Figura 3.16. Parámetros de los servomotores con torque desactivado para modo_delta_pasivo.launch.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Con esta configuración, los programas pueden leer el estado de los servomotores suscribiéndose al tópico creado o modificarlo publicando en él.

3.6.4 ARCHIVO DE INICIO DE CONTROLADORES (MODO_DELTA_DIBUJADOR.LAUNCH & MODO_DELTA_PASIVO.LAUNCH).

Lo siguiente que se necesita es crear un archivo que ponga en marcha el robot en donde se cargan los parámetros del controlador para el servidor de parámetros y la puesta en marcha del controlador en la figura 3.17 y 3.18 están los archivo de lanzamiento para ejecución del programa tanto para el modo dibujador y modo pasivo.

```
<launch>
    <!-- Start the low-level driver manager with parameters -->
    <node name="ttyUSB0_manager" pkg="dynamixel_controllers" type="controller_manager.py"
output="screen">
        <rosparam>
            namespace: dxl_manager
            serial_ports:
                delta_port:
                    port_name: "/dev/ttyUSB0"
                    baud_rate: 1000000
                    min_motor_id: 1
                    max_motor_id: 6
                    update_rate: 20
        </rosparam>
    </node>
    <!-- Start motor1 controller -->
    <rosparam file="$(find my_dynamixel)/config/motors.yaml" command="load"/>
    <node name="motor_spawner" pkg="dynamixel_controllers" type="controller_spawner.py"
          args="--manager=dxl_manager
                --port=delta_port
                --type=simple
                motor1
                motor2
                motor3"
          output="screen"/>

     <node name="Delta_Dibujando" pkg="delta1" type="robot_delta_dibujador.py" />

</launch>
```

Nombre de cada uno de los servos

Script de programa de aplicación modo dibujo

Figura 3.17 Archivo de inicio para modo_delta_dibujador.launch.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

```
<launch>
    <!-- Start the low-level driver manager with parameters -->
    <node name="ttyUSB0_manager" pkg="dynamixel_controllers" type="controller_manager.py"
output="screen">
        <rosparam>
            namespace: dxl_manager
            serial_ports:
                delta_port:
                    port_name: "/dev/ttyUSB0"
                    baud_rate: 1000000
                    min_motor_id: 1
                    max_motor_id: 6
                    update_rate: 20
        </rosparam>
    </node>
    <!-- Start motor1 controller -->
    <rosparam file="$(find my_dynamixel)/config/motors_torque_off.yaml" command="load"/>
    <node name="motor_spawner" pkg="dynamixel_controllers" type="controller_spawner.py"
          args="--manager=dxl_manager
                --port=delta_port
                --type=simple
                motor1
                motor2
                motor3"
          output="screen"/>

     <node name="Delta_pasivo" pkg="delta1" type="delta_pasivo.py" />

</launch>
```

Script de programa de aplicación modo pasivo

Figura 3.18 Archivo de inicio para modo_delta_pasivo.launch.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.7 IMPLEMENTACIÓN ELECTRÓNICA DEL SISTEMA

En el middleware ROS con lenguaje Python se programan las órdenes enviadas al sistema de comunicación (dynamixel USB) y el control en sí de la posición se realiza automáticamente mediante el PID de los servos dynamixel y para el control del láser se usa una tarjeta Arduino Uno. La figura 3.19 muestra la comunicación del sistema.

Figura 3.19 Esquema general del sistema.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.7.1 IMPLEMENTACIÓN LÁSER DE DIODO CNC 445NM 1W

Antes de poner en funcionamiento del láser ver figura 3.20, es necesario conocer algunos datos característicos del mismo para su mejor uso.

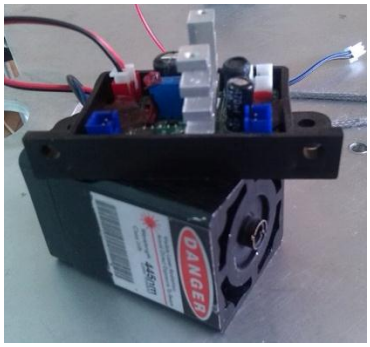


Figura 3.20 Láser de diodo CNC 445nm 1W

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.7.2 Especificaciones

En la tabla 3.1 se encuentra las especificaciones del módulo láser azul.

Tabla 3.1 Características técnicas del láser 445nm 1W.

Longitud de onda (nm)	**445**
Potencia de salida (mw)	1000
Estabilidad de la energía	< 4%
Tiempo de calentamiento (min)	< 5
Haz de divergencia, ángulo completo (mrad)	< 2.0
Diámetro de apertura	~ 3*6
Tiempo de vida(horas)	1000
Temperatura °C	10~ 35
Modulación Externa	5v TTL/5v Analógica
Frecuencia	0.30 KHZ
Posición de Apertura (mm)	20*40

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.7.3 ENFOQUE DEL LÁSER

Para enfocar el láser necesita seguir el siguiente paso ver figura 3.21;

1. Con el desarmador plano o con los dedos se puede mover el lente:

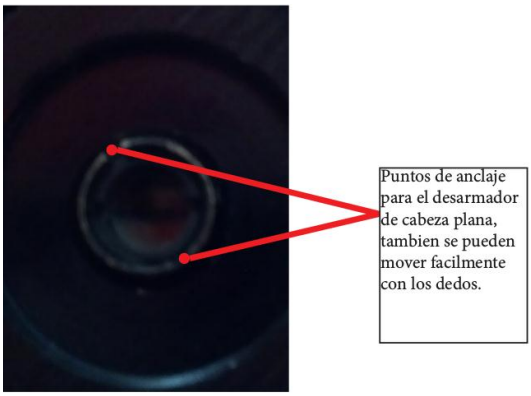


Figura 3.21 Enfoque del lente del láser.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.7.4 CONEXIÓN DEL MÓDULO AL LÁSER.

La conexión es la siguiente:

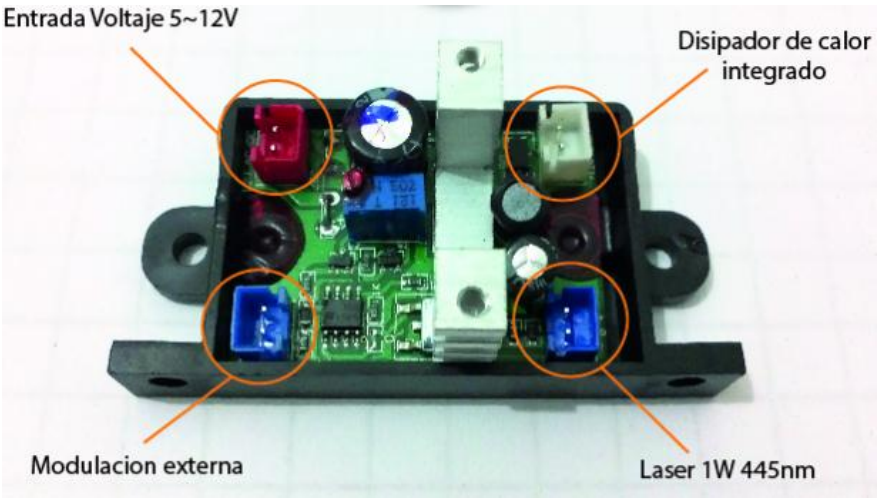


Figura 3.22 Modulo láser.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

3.7.5 INTERACCIÓN ENTRE ARDUINO Y PYTHON

Para el control del láser utilizaremos la tarjeta Arduino. El encendido y apagado del láser se hace mediante comandos que enviaremos desde el ordenador a Arduino utilizando un script en Python.

A. Instalación del entorno de desarrollo para arduino

Instalación arduino con todas sus dependencias.

```
$ sudo apt-get install arduino
```

Por ultimo hay que darle permisos al puerto para poder usarlo. Por defecto se usa el puerto **ttyACM0**.

```
$ sudo chmod a+rw /dev/ttyACM0
```

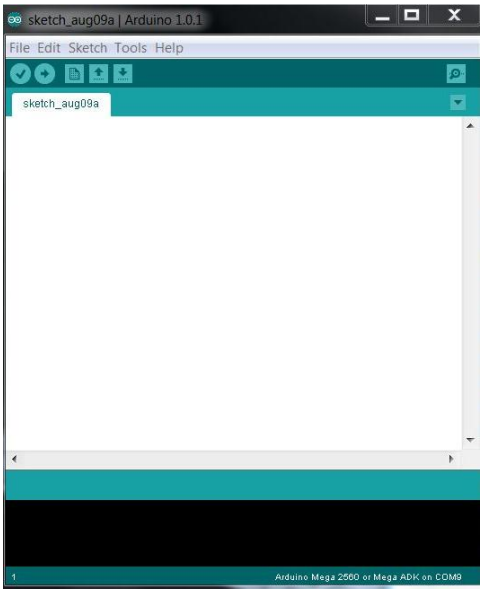


Figura 3.23 IDE de Arduino

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Una vez con el IDE instalado como se muestra en la figura 3.23 tenemos un código en python para que se ejecute dentro del ordenador que envía mensajes (carácteres) a la placa Arduino. Ésta estará previamente cargada con un código para recibir mensajes por serial y activar el diodo láser.

B. Control del diodo láser

Primero se escribe el código en python que se debe ejecutar en el ordenador. Éste código lee un carácter introducido por el usuario y lo envía por Serial a Arduino. Si se escribe una ‘s’, salimos del programa.

Guardar el código en un fichero con extensión '.py' control.py. Ahora se escribir el código para Arduino, que encenderá el diodo laser al recibir 'e' por Serial y lo apagará al recibir 'a'.Ver código de programación en el Anexo D.

3.7.6 DIAGRAMA DEL SISTEMA DE CONEXIONES

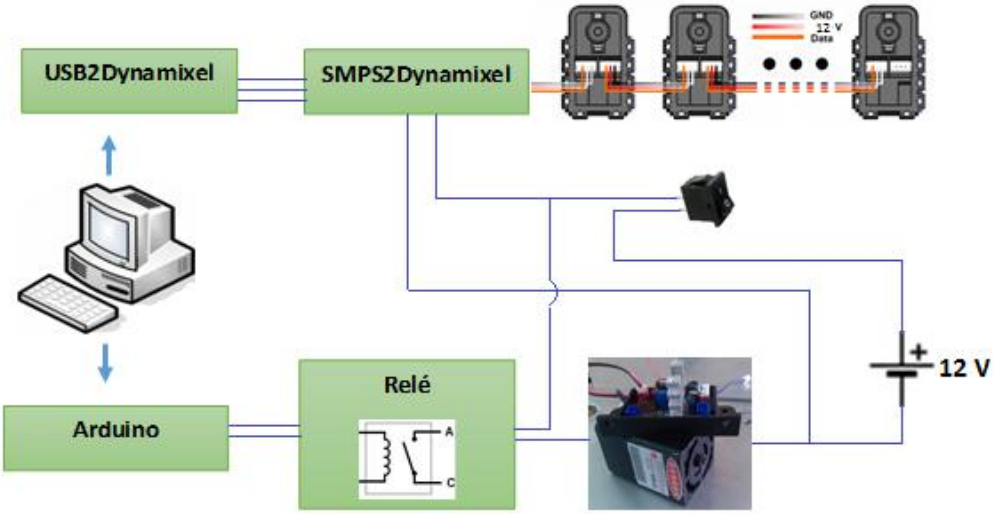


Figura 3.24 Diagrama de conexiones del Robot Delta.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

CAPÍTULO IV

PRUEBAS Y RESULTADOS EXPERIMENTALES

4.1 PRUEBAS Y RESULTADOS DEL FUNCIONAMIENTO.

4.1.1 EJECUCIÓN GENERAL DE LA PLATAFORMA

a) Modo Pasivo

Para comenzar la ejecución debemos ejecutar el comando:

```
$ rosrun rviz rviz
$ roslaunch my_dynamixel modo_delta_pasivo.launch
```

Este comando iniciará el controlador de los motores con la configuración“motors _torque_off.yaml”. Una vez iniciado este controlador, se pasa a ejecutar “delta_pasivo.py”, ver los resultados en la figura 4.1.

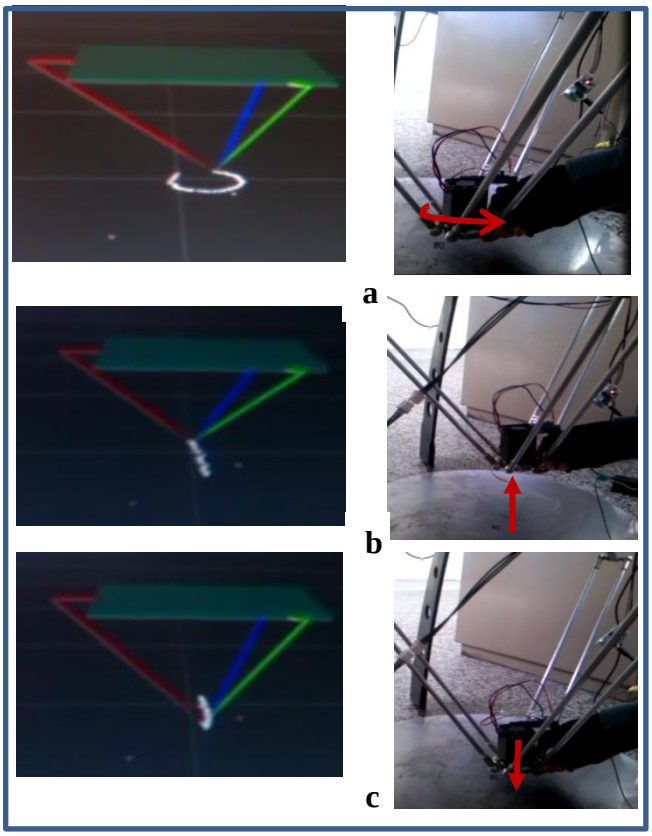


Figura 4.1 Visualización de los movimientos del robot real en la simulación.

a. Movimiento en círculo b. Movimiento hacia arriba c. movimiento hacia abajo.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

b) **Modo movimiento delta**

Para iniciar este modo debemos ejecutar el comando:

```
$ rosrun rviz rviz
$ roslaunch my_dynamixel modo_movimiento_delta.launch
```

Este comando iniciará el controlador de los servomotores con la configuración "motors.yaml". Una vez iniciado este controlador, se pasa a ejecutar "simul_demo_robot_delta_dynamixel.py".

Ahora Rviz nos permite ver el movimiento completo del robot además del movimiento real ejecutado, de esta manera podemos verificar si el movimiento se está cumpliendo o no. En la Figura 4.2 se muestra una secuencia de movimientos producidos por el robot, en este se observa las diferentes posiciones que adopta las estructura mecánica del robot a lo largo de la trayectoria ejecutada.

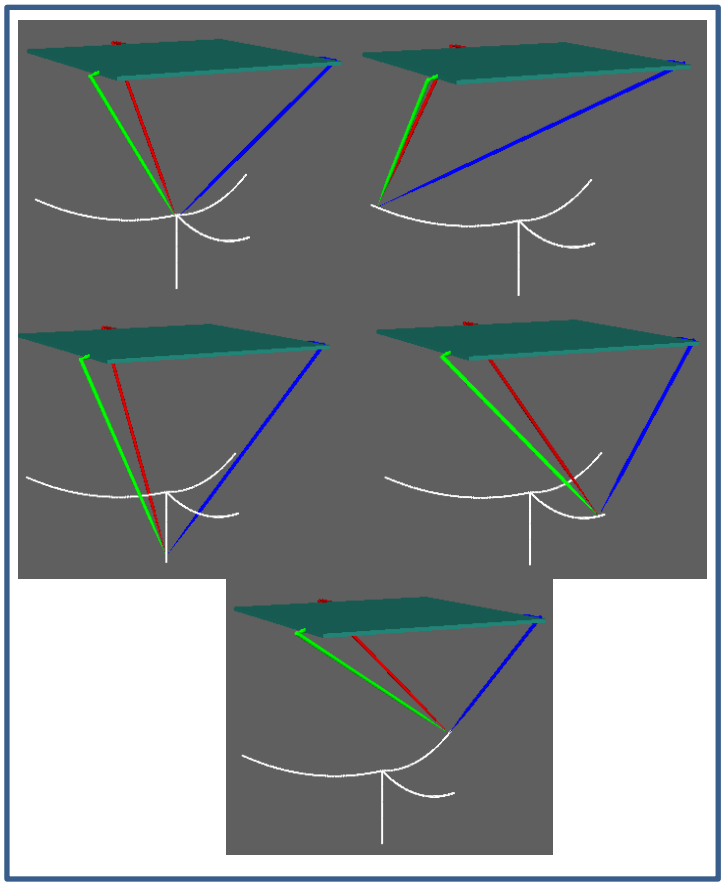


Figura 4.2 Movimiento de la trayectoria, realizado por el robot delta en Rviz.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

c) Modo dibujador

Antes de realizar esta prueba tenemos que escoger una imagen clara y procesarla a través del archivo “drawing.py” como ya se explicó en capítulos anteriores.

La prueba que se realizó empieza una vez encendido el switch, los servomotores y el láser están listos para recibir el launch y el script de arranque, el láser será el primero en ejecutarse, pues necesita unos 30 segundos para calentarse, después transcurrido los 30 segundos lanzamos el nodo que hace posible que el robot se coloque en cada punto obtenido de la imagen anteriormente procesada, imprimiendo dicha imagen.

Para iniciar este modo debemos ejecutar el comando:

```
$ roslaunch my_dynamixel modo_delta_dibujador.launch
$ rosrun rviz rviz
```

Este comando iniciará el controlador de los servomotores con la configuración “motors.yaml”. Una vez iniciado este controlador, se pasa a ejecutar “robot_delta_dibujador.py”.

4.2. PRUEBAS DE CORTE Y GRABADO EN DIFERENTES MATERIALES.

4.2.1 PROCESADO DE LA IMAGEN MODO VECTORIZADO.

El modo vectorizado nos facilita obteniendo los contornos de una imagen el mismo que puede servir para cortar.

Para verificar si el robot sigue la trayectoria deseada se colocó como efector final un lápiz. En la figura 4.3, se observa que la trayectoria es la deseada pero no tiene la precisión deseada.

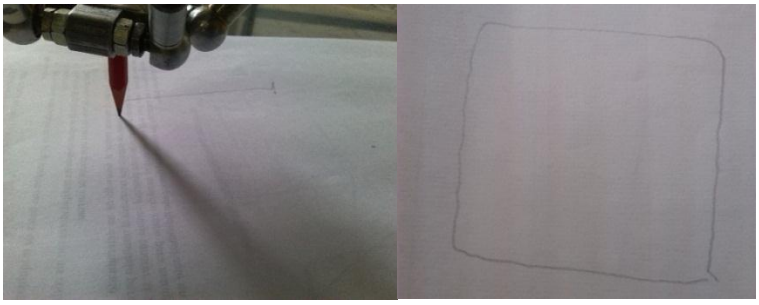


Figura 4.3 Dibujando un cuadrado.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Al sustituir el lapiz por el diodo laser se observa lo siguiente, ver figura 4.4.

Figura 4.4 Cuadrado usando laser.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Los resultados no son los esperados al comparar el lapiz con el laser, entonces se tendra que calibrar el láser para obtener el enfoque perfecto y tomar en cuenta la velocidad para conseguir un buen corte.

Pueda que se tenga que jugar un poco con la lente y la distancia del foamy negro ver figura 4.5. Para hacer el ajuste fino se tomó el tiempo que tardo el láser para quemar un agujero a través del foamy.

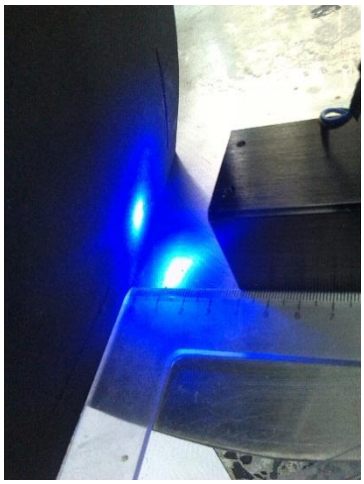
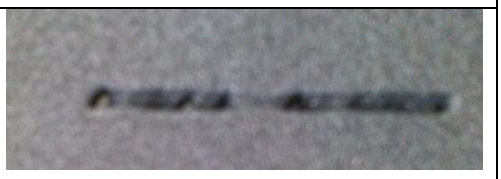
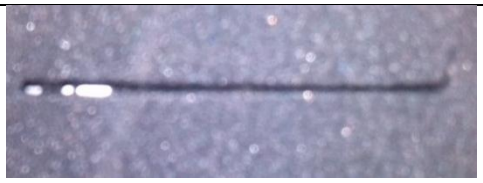


Figura 4.5 Enfocando el láser para el corte.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

En la tabla 4.1 se muestran las pruebas y resultados con el enfoque final del láser.

Tabla 4.1 Resultados de pruebas de corte del láser

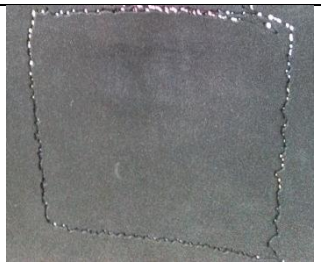
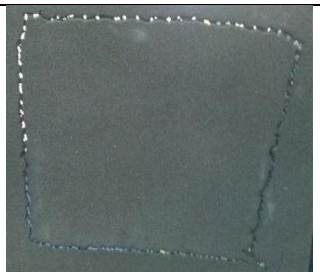
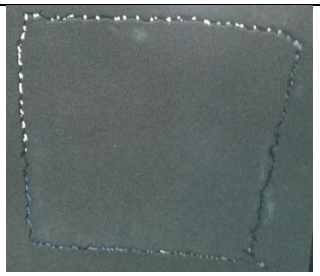
Distancia de corte (cm)	Tiempo de calentamiento (segundos)	Resultados
8	12	
6	10	
5	4	
2	3	
4	2	
3.5	2	

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

La mejor distancia para cortar es de 3.5 cm en cuanto el tiempo depende del tiempo de encendido en menos de 30 segundos el laser esta listo para quemar pero tambien se toma en cuenta el tipo de material y el color del mismo en colores obscuros sucede mas rapido el proceso que corte.

En la tabla 4.2 se muestra pruebas hechas a diferentes velocidades y tiempos con el enfoque final.

Tabla 4.2 Resultados velocidad y tiempo de corte del cuadrado.

Velocidad (cm/s)	Pasada	Tiempo	Resultados
1	3	38	
0.5	2	38	
0.07	1	37.5	
0.05	1	37.5	
0.05	1	38	

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

En la figura 4.6 se encuetra pruebas hechas con un circulo.

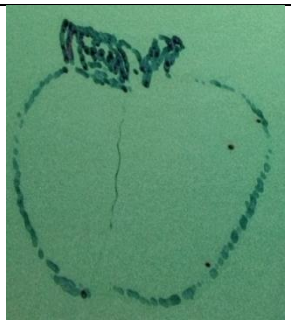
a) **b)**

Figura 4.6 Impresión de un círculo a) Con lápiz b) Con láser

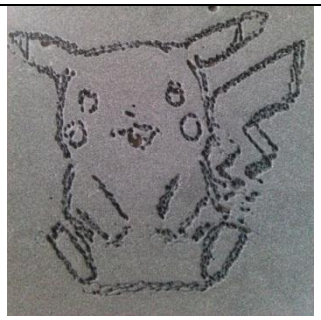
Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Los resultados obtenidos con la aplicación modo dibujo se muestran en la tabla 4.3 y tabla 4.4 imágenes vectorizadas.

Tabla 4.3 Resultados velocidad y tiempo de Impresión de imágenes vectorizada.

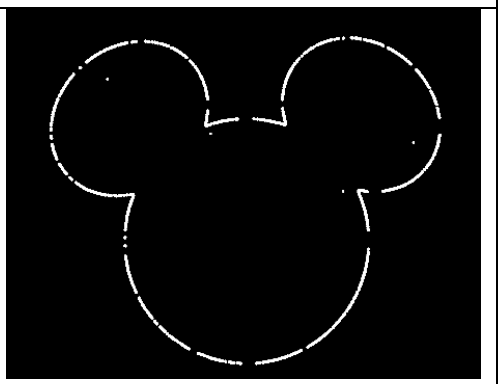
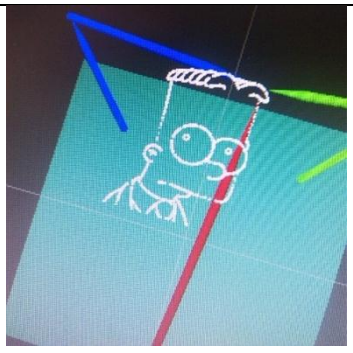
Tiempo(min)	Impresión Vectorizada Según Número de Puntos
3	
4	

CONTINUA →

8	
12	

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Tabla 4.4 Impresión de diferente figuras

Tiempo (min)	Imagen Impresa	Imagen Simulada
2		
7		

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

La velocidad de los servomotores no influye en proceso de imprimir la imagen con láser, al procesar la imagen se genera puntos todo depende como el usuario procese la imagen si se quiere mayor calidad más puntos se requiere y por lo mismo mayor tiempo para imprimir dicha imagen.

Además, como se puede observar en la figura 4.6 nos da varias fotos donde se aprecia la calidad de impresión de la imagen, y hay que tomar en cuenta que son puntos y el láser logra detectar esos puntos que son evidentes pues la programación está hecha para simular puntos, esto se debe a que ROS no tiene ninguna conexión con algún programa de CNC que nos facilitaría para generar trayectorias perfectas que solo los códigos G nos proporciona, razón por la cual no logra un corte perfecto, como se ve en las fotos solo corta en los puntos marcados y pasar con el láser 2 o 3 veces no es factible porque siempre pasan por los mismos puntos, talvez si se usa un láser de mayor potencia con una longitud de onda mayor logre cortar completamente.

4.2.2 PROCESADO DE LA IMAGEN MODO RASTERIZADO.

El modo rasterizado se utiliza para realizar grabados

A continuación se indican varias fotografías donde se aprecia como las imágenes quedan grabadas en diferentes tipos de materiales ver figura 4.7.

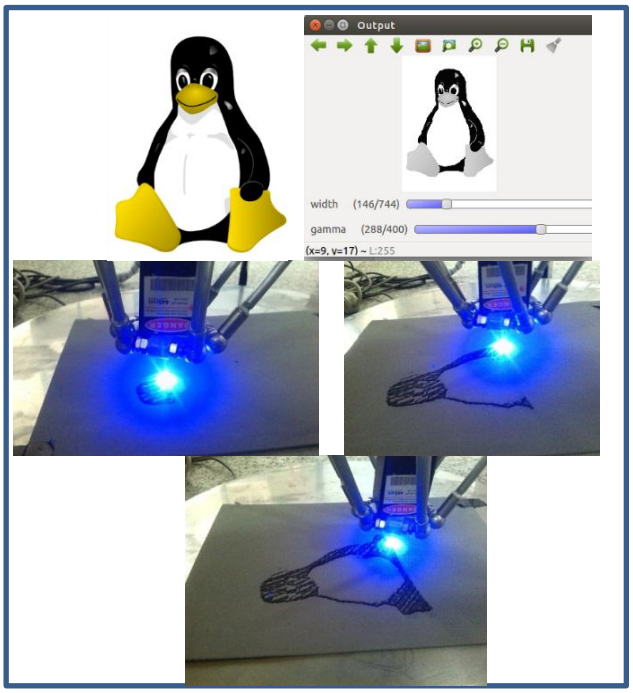


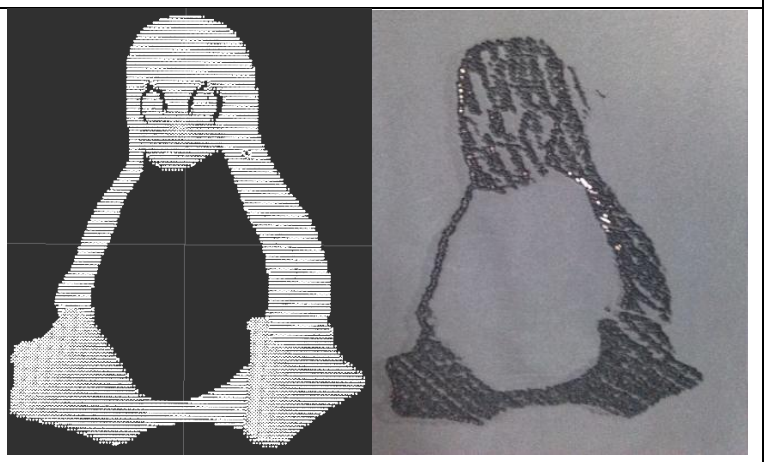
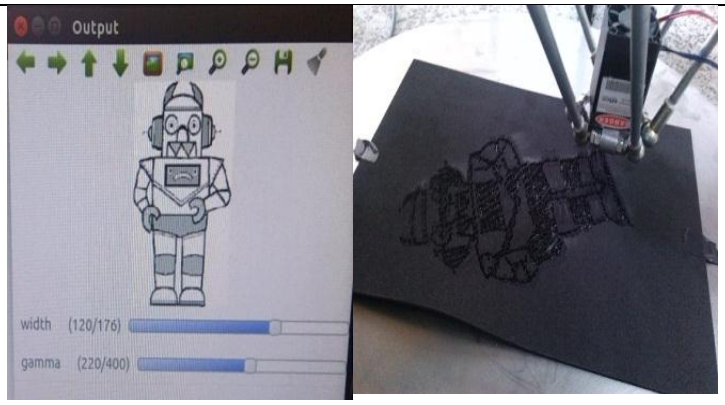
Figura 4.7 Procedimiento de grabado (Primeras pruebas).

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Resultados de grabado

En la tabla 4.5 se indica figuras grabadas en foamy.

Tabla 4.5 Grabados en foamy

Tiempo (min)	Comparación de Impresión en Foamy
12	
30	
90	

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

En la tabla 4.6 se indica figuras grabadas en Madera.

Tabla 4.6 Grabados en madera

Tiempo (min)	Impresión en Madera(aglomerado, Triplex, balza)
120	
15	
30	 

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

En cuanto a los resultados obtenidos en grabado se ha ido mejorando, en madera los resultados se aprecian mejor, es obvio que el láser tuvo mejores resultados en grabado tomando en cuenta que, al ser un prototipo puede tener fallas.

4.3 HERRAMIENTA RQT DE ROS

ROS ofrece rqt , un marco basado en Qt para el desarrollo de interfaces gráficas para el robot.

rqt_top es para vigilar los procesos de ROS. En la figura 4.8 se indica los procesos en Ejecución.

```
$ rosrun rqt_top rqt_top
```

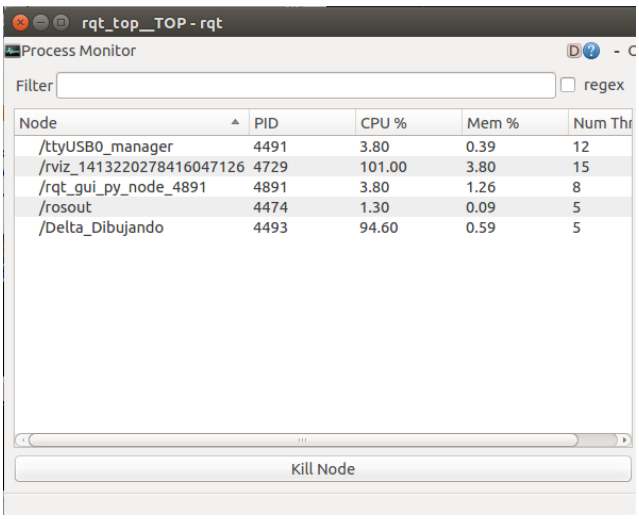


Figura 4.8 Procesos ROS en ejecución.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

rqt_graph crea un gráfico dinámico de lo que está pasando en el sistema, es decir presenta los nodos en forma gráfica.

```
$ rosrun rqt_graph rqt_graph
```

Verá algo similar a la figura 4.9 y figura 4.10:

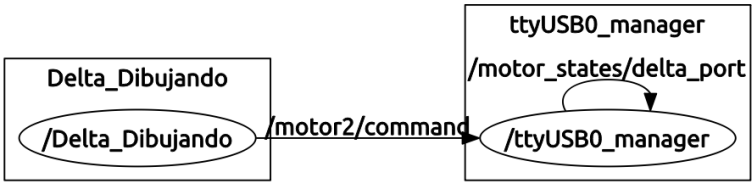


Figura 4.9 Grafo de un solo nodo.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

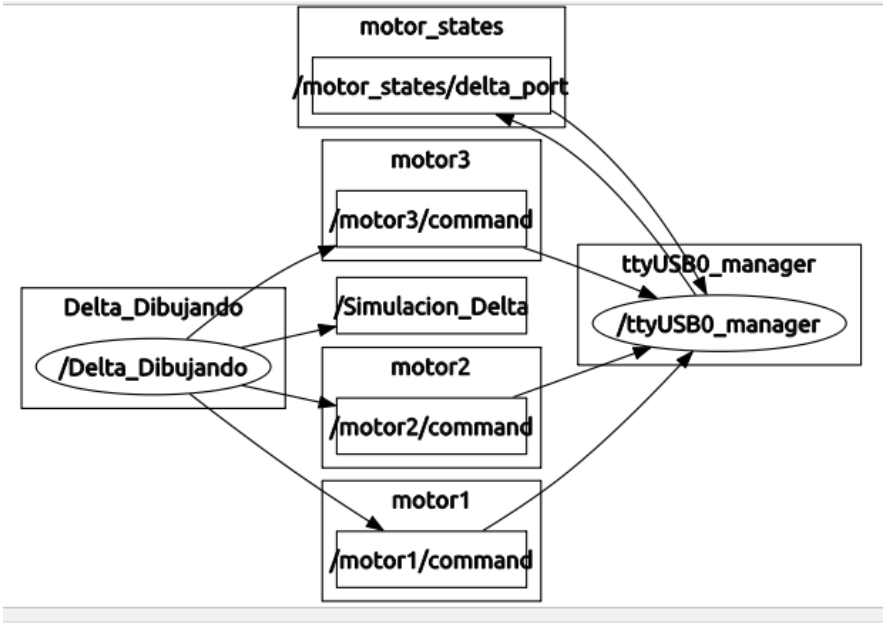


Figura 4.10 Grafos del sistema con sus topics.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

Si se coloca el puntero del ratón sobre / motor2 / command se pondrán de relieve los nodos ROS (rojo y verde) y temas (flecha verde). Como se puede ver, Delta_Dibujando y el ttyUSB0_manager estos nodos se comunican sobre el tema llamado / motor2 / command como se ve en la figura 4.11 y la figura 4.12.

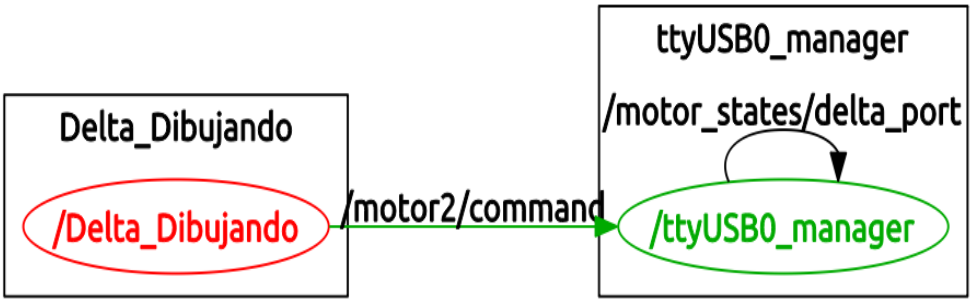


Figura 4.11 Identificación del Grafo de un solo nodo.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

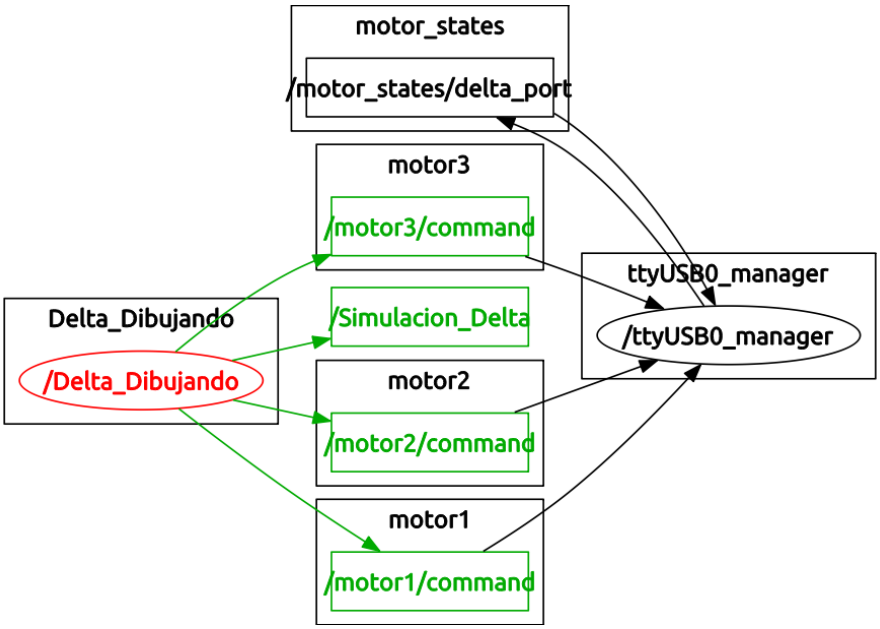


Figura 4.12 Identificación de los Grafos del sistema con sus topics.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

En la figura 4.13 se indica el **rqt_topic** es el que permite supervisar cualquier número de temas que se publicó dentro del sistema.

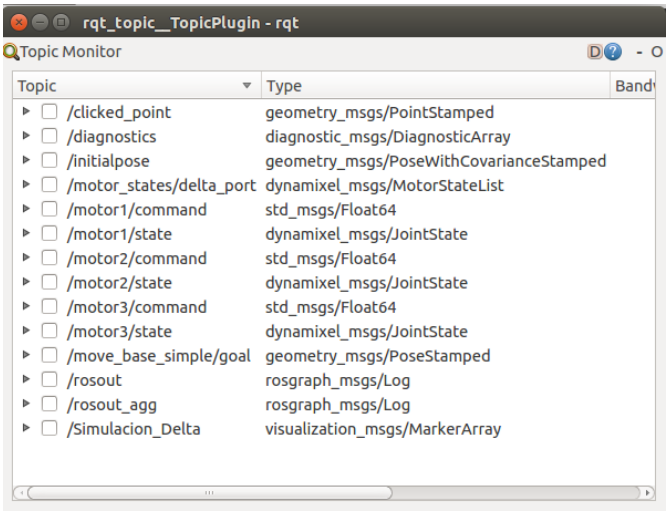


Figura 4.13 Tema publicados en el sistema.

Elaborado por: Tumbaco, Diana y Quimbita, Wilmer

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Para el diseño robot delta se utilizó software CAD solidworks y para el análisis de la estructura utilizamos Ansys.
- Las dimensiones fueron tomadas de múltiples proyectos y modificadas para un mejor desempeño.
- El material utilizado fue aluminio por sus características mecánicas en especial su bajo peso, resistencia a la deformación y facilidad de maquinabilidad.
- En el análisis de la estructura tenemos que el robot es resistente ya que se obtiene un factor de seguridad mayor que 1, y su deformación es mínima.
- Las juntas deben tener la menor fricción posible para mayor precisión.
- Robotic Operating System facilita un desarrollo organizado de código reusable y abierto, con herramientas necesarias como Rviz y obtener una visualización del estado real del robot, así como los valores obtenidos por el análisis cinemático, de esta forma se puede hacer la comparación y corroborar que todo esté funcionando correctamente.
- Los Dynamixel al ser servos inteligentes y al interactuar directamente con ROS nos facilita su programación pero no se obtiene la precisión buscada, al momento de imprimir la imagen, la solución es utilizar la gama de los Dynamixel PRO con la diferencia que son de alto costo.
- La velocidad mínima para trabajar con los servomotores es de 0,07 cm/s menor a esto se desestabiliza y es notoria porque no logra llegar a la posición Z indicada.
- La velocidad tanto en el corte como en el grabado disminuye cuando trabaja el simulador y el robot en tiempo real.

- El láser no corta colores claros porque se reflejan muy bien los colores claros lo contrario sucede con el color negro que absorbe mucha energía, por lo que quema rápidamente.
- El láser quema o corta mejor en materiales tipo foamy especialmente si son de color negro, pero al enviar puntos el láser solo quema dichos puntos, por lo que a futuro se debería trabajar con trayectorias.
- El láser de 1 watt para grabar resultó una buena opción sobre todo para grabado en balsa y en aglomerado.
- La velocidad de grabado o corte depende del número de puntos enviados, entre más puntos sea el dibujo mayor será el tiempo.
- El robot puede cambiar según su efector final para ser multipropósitos.

5.2 RECOMENDACIONES

- Al ensamblar el robot tomar en cuenta la posición de los servomotores ya que deben estar separados 120° cada uno.
- Para la colocación de los brazos al servomotor correr el programa en modo pasivo para verificar su posición de trabajo.
- Usar el USB2Dynamixel ya que es un dispositivo para operar Dynamixel AX-12A directamente de la PC ahorrándose tiempo de trabajo.
- Al ser un láser de clase 3B: La radiación láser accesible es peligroso para los ojos y, en casos especiales también para la piel. Por favor, tomar precaución, no usar el láser sin gafas de seguridad de protección adecuadas.
- Realizar una investigación exhaustiva para lograr una conexión de ROS con algún software CNC ayudaría mucho para las mejoras del proyecto propuesto.
- Para mejorar los prototipos de robots paralelo se debe realizar investigaciones previas en su defecto incentivar a los desarrolladores de ROS que se involucren en el estudio de la robótica paralela debido a que cada vez se hace más popular a nivel académico e industrial.
- Esta plataforma robótica presenta un prototipo que puede ser mejorado en muchos aspectos por lo mismo se recomienda continuar con el estudio y desarrollo de prototipos de robots tipo paralelo.
- Realizar el estudio del movimiento dinámico del robot delta para mejoras del prototipo en cuanto a velocidades y aceleraciones.

REFERENCIAS BIBLIOGRAFICAS

NETGRAFÍA

(s.f.). Recuperado el 23 de Mayo de 2014, de http://www.kairel.com/tecnicas-de-marcaje.html

(12 de Mayo de 2014). Recuperado el 14 de Abril de 2012, de http://kaven-fontes.blogspot.com/

4GWar. (12 de Agosto de 2013). Recuperado el 20 de Mayo de 2014, de 4GWar: http://4gwar.wordpress.com/2013/08/12/unmanned-systems-big-droid-drone-and-bot-show-opens-in-d-c/

4Gwar. (2013). UNMANNED SYSTEMS: Big Droid, Drone and 'bot Show Opens in D.C. UPDATE. *4Gwar*.

Aracil, R. (enero de 2006). *arvc.umh.es.* Obtenido de http://arvc.umh.es/documentos/articulos/RIAI%202006.pdf

Bautista Quintero Ricardo, A. C. (4 de Noviembre de 2011). *http://www.mecamex.org/.* Recuperado el 8 de Octubre de 2013, de http://www.mecamex.org/: http://www.mecamex.net/anterior/cong10/trabajos/art32.pdf

Bonev, I. (s.f.). *http://www.parallemic.org/*. Obtenido de http://www.parallemic.org/Reviews/Review002p.html

Constante, P. (Agosto de 2012). *repositorio.espe.edu.ec*. Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/5924/1/T-ESPEL-0940.pdf

D'Inca, C. (22 de Agosto de 2010). *http://www.um.edu.ar/.* Obtenido de http://www.um.edu.ar/catedras/claroline/document/document.php?cmd=exChDir&file=L0FQVU5URVM%3D&cidReset=true&cidReq=IE030

Díaz, V. G. (26 de Enero de 2013). Obtenido de http://www.slideshare.net/vicegd/desarrollo-robtico-robot-operating-system-ros

Foundation, O. S. (17 de Diciembre de 2013). *http://www.ros.org/*. (P. Rey, Editor) Recuperado el 10 de Junio de 2014, de http://www.ros.org/: http://wiki.ros.org/

Foundation, O. S. (23 de Diciembre de 2013). *http://www.ros.org/*. (T. Foote, Editor) Recuperado el 12 de Enero de 2014, de http://www.ros.org/: http://wiki.ros.org/ROS/Concepts

Gonzáles, N., & Reinoso, E. (Marzo de 2011). *dspace.ups.edu.ec*. Obtenido de http://dspace.ups.edu.ec/bitstream/123456789/1921/14/UPS-CT002354.pdf

Guerrero, G. N. (30 de Septiembre de 2012). *http://geeksknowledge.blogspot.com/*. Obtenido de http://geeksknowledge.blogspot.com/: http://geeksknowledge.blogspot.com/2012/09/rsf-o-middleware-para-robots.html

http://awesomebytes.com/. (13 de Septiembre de 2012). Recuperado el 4 de Noviembre de 2013, de http://awesomebytes.com/: http://awesomebytes.com/2012/09/13/computer-aplications-for-a-delta-robot-via-ros/

http://forums.trossenrobotics.com/. (27 de Julio de 2007). Recuperado el 12 de Septiembre de 2013, de http://forums.trossenrobotics.com/tutorials/introduction-129/delta-robot-kinematics-3276/

http://opencv.org/. (s.f.).

http://robosavvy.com/. (24 de Junio de 2014). Obtenido de http://robosavvy.com/: http://robosavvy.com/store/product_info.php/manufacturers_id/15/products_id/1725

http://support.robotis.com/. (2010). Recuperado el 12 de Junio de 2014, de http://support.robotis.com/: http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm

http://wiki.ros.org/ROS/Tutorials/BuildingPackages. (s.f.).

http://wiki.ros.org/ROS/Tutorials/CreatingPackage. (s.f.).

http://wiki.ros.org/rviz. (s.f.).

http://www.robotica-personal.es. (27 de Noviembre de 2009). Obtenido de http://www.robotica-personal.es: http://www.robotica-personal.es/2009/11/nuevo-kit-robotis-bioloid-premium-en-ro.html

http://www.ros.org/. (22 de Mayo de 2014). Recuperado el 23 de Junio de 2014, de http://www.ros.org/: http://wiki.ros.org/ROS/Introduction

Isabel Rodrigo Martín, R. C. (s.f.). *UF1523: Preparación de la impresión en serigrafía.* Innovación y Cualificacion S.L. Recuperado el 12 de Febrero de 2014, de http://books.google.com.ec/books?id=vnGSAgAAQBAJ&pg=PT4&dq=Isabe

l+Rodrigo+Mart%C3%ADn,+Roc%C3%ADo+Collado+Alonso+UF1523:+Preparaci%C3%B3n+de+la+impresi%C3%B3n+en+serigraf%C3%ADa.&hl=es-419&sa=X&ei=-BSeU_P-GMu3yATO6YGoDA&ved=0CB0Q6AEwAA#v=onepage&q=Isabel%

León, J. A. (28 de Mayo de 2009). *itzamna.bnct.ipn.mx*. Obtenido de http://itzamna.bnct.ipn.mx/dspace/bitstream/123456789/4900/1/JABL.PDF

López, C. A. (Diciembre de 2010). *http://eie.ucr.ac.cr/.* Recuperado el 13 de Mayo de 2014, de http://eie.ucr.ac.cr/: http://eie.ucr.ac.cr/uploads/file/proybach/pb2010/pb2010_041.pdf

Lucas Mateo Urrea Mantilla, S. A. (2012). *http://repository.unimilitar.edu.co/.* Obtenido de http://repository.unimilitar.edu.co/: http://repository.unimilitar.edu.co/bitstream/10654/9953/1/UrreaMantillaLucasMateo2012.pdf

María Fernanda Utreras Abad, D. D. (17 de Octubre de 2013). *http://www.dspace.espol.edu.ec/.* Recuperado el 12 de Mayo de 2014, de http://www.dspace.espol.edu.ec/: http://www.dspace.espol.edu.ec/bitstream/123456789/25358/1/Resumen%20de%20tesis%20MUtreras%20y%20DDecimavilla%2c%20director%20de%20tesis%20Ph.D.%20Daniel%20Ochoa%20D.%2017%20octubre%202013.pdf

MARTINEZ, D. M. (19 de Noviembre de 2012). *http://e-archivo.uc3m.es/.* Recuperado el 19 de Enero de 2014, de http://e-archivo.uc3m.es/: http://e-archivo.uc3m.es/handle/10016/16723

Nader Mohamed, J. A.-J. (Septiembre de 2008). *In Proc. of The IEEE Intl. Conf. on Robotics, Automation, and Mechatronics.* Recuperado el 27 de Mayo de 2014, de In Proc. of The IEEE Intl. Conf. on Robotics, Automation, and Mechatronics: http://faculty.uaeu.ac.ae/Nader_M/papers/RAM2008.pdf

Nader Mohamed, J. A.-J. (Mayo de 2009). *IJCSNS International Journal of Computer Science and Network Security.* Recuperado el 23 de Mayo de 2014, de IJCSNS International Journal of Computer Science and Network Security: http://faculty.uaeu.ac.ae/ijawhar/publications/review%20for%20mid%20rob%20journ.pdf

Ner, A. (27 de Septiembre de 1012). *http://afroner9.blogspot.com/*. Obtenido de http://afroner9.blogspot.com/2012/09/robotica-y-bionica.html

Newsoaxaca. (26 de Julio de 2013). Recuperado el 20 de Mayo de 2014, de Newsoaxaca: http://newsoaxaca.com/index.php/actualidad/16925-un-robot-asistira-a-personas-de-la-tercera-edad

Peña, C., Oviedo, E., & Cárdenas, P. (Junio de 2011). *www.scielo.org.co*. Obtenido de http://www.scielo.org.co/pdf/cein/v21n1/v21n1a05.pdf

Pfeiffer, S., & Gabas, A. (s.f.). *awesomebytes.com*. Obtenido de http://awesomebytes.com/wp-content/uploads/2012/09/Informe_delta_spfeiffer_agabas_con_anexos.pdf

Robotnik. (5 de Febrero de 2013). *Robotnik* . Obtenido de http://www.robotnik.es/blog/interfaces-de-usuario-en-ros/#comments

Romero, A. M. (21 de Junio de 2014). *http://wiki.ros.org/*. Recuperado el 12 de Mayo de 2014, de http://wiki.ros.org/: http://wiki.ros.org/ROS/Concepts

Sanjuán, J. C. (15 de Noviembre de 2010). *cmm.blogspot.com*. Obtenido de cmm.blogspot.com: http://avancescientificosytecnologicos-cmm.blogspot.com/2010/11/da-vinci-robot-que-realiza-operaciones.html

Source, O. S. (1 de Mayo de 2014). *http://www.ros.org/*. (D. Thomas, Editor) Recuperado el 15 de Mayo de 2014, de http://www.ros.org/: http://wiki.ros.org/Distributions

Torre, R. R. (17 de septiembre de 2012). *http://bibing.us.es/*. Obtenido de http://bibing.us.es/: http://bibing.us.es/proyectos/abreproy/5139/fichero/PFC-+Por+cap%EDtulos%252F3-Entorno+de+simulaci%F3n.pdf

Trossen Robotics. (6 de 11 de 2014). Obtenido de http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

VIZUET, M. H. (28 de Marzo de 2007). *http://tesis.ipn.mx/*. Recuperado el 9 de Mayo de 2014, de http://tesis.ipn.mx/: http://tesis.ipn.mx/xmlui/bitstream/handle/123456789/5120/1711_2007_ESIME-ZAC_MAESTRIA_hernandez_vizuet_mauricio.pdf?sequence=1

Yorobot. (26 de noviembre de 2006). *www.robotic-lab.com*. Obtenido de http://www.robotic-lab.com/blog/2006/11/26/robotica-paralela-delta/

Zabalza, I., & Ros, J. (23-25 de Octubre de 2007). *www.imem.unavarra.es*. Obtenido de http://www.imem.unavarra.es/isidro/articles/Zabalza-Cuzco.pdf

Zavatsky, M. (7 de Diciembre de 2010). *http://www.legomindstormsrobots.com*. Obtenido de http://www.legomindstormsrobots.com/lego-minstorms/delta-kinematics-robot/

BIBLIOGRAFÍA

Arranz, A. C. (2011). *Tecnología Láser y sus aplicaciones Industriales.* (Vol. I). Barcelona, España: Marcombo, S.A. Recuperado el 5 de Junio de 2014

Barrientos, A., Peñin, L., Balaguer, C., & Aracil, R. (2007). *Fundamentos de Robotica.* Madrid: McGraw-Hill/Interamericana.

Baturone, A. O. (2001). Robótica Manipuladores y robots móviles. Barcelona: Marcombo S.A.

Berger, E. (18 de Agosto de 2010). Interview with Eric Berger (Co-Director, Personal Robotics Program, Willow Garage). (A. K. Michael Beetz, Entrevistador) Springer-verlag 2010. doi:10.1007/s13218-010-0058-7

Gómez, C. R. (s.f.). *MF1349_1: Impresión de productos en Tampografía.* Innovación y Cualificacion S.L.

Merlet, J. (2006). *Parallel Robots.* Springer.

Xianwen Kong, C. G. (2007). *Type Synthesis of Parallel Mechanisms* (Vol. 33). (S. B. Heidelberg, Ed.)

ANEXOS

ANEXO A: PROPIEDADES DE LOS MATERIALES.

ANEXO B: PLANOS MECÁNICOS.

ANEXO C: INDICACIONES USO DEL LÁSER

ANEXO D: CODIGO DE PROGRAMACION.

ANEXO E: MANUAL DE USUARIO

Elaborado por:

_____	_____
Diana Carolina Tumbaco Mendoza.	Wilmer Enrique Quimbita Zapata.

Aprobado por:

_____

Ing. Vicente Hallo

DIRECTOR DE LA CARRERA DE INGENIERÍA MECATRÓNICA

Certificado por:

_____

Dr. Freddy Jaramillo Checa

SECRETARIO ACADÉMICO