BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN INGENIERÍA ELECTRÓNICA,
OPCIÓN INSTRUMENTACIÓN ELECTRÓNICA

Tesis para obtener el grado de:

MAESTRO EN INGENIERÍA ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO MANIPULADOR CON CONTROL DE TELEOPERACIÓN COMO ROBOT DE ASISTENCIA PERSONAL

Presenta:

Jesús Enrique Pérez Ramírez*

Director de tesis:

M.C. Ricardo Álvarez González F.C.E.

Co-directores de tesis:

Dr. José Fermi Guerrero Castellanos F.C.E.

M.C. Alba Maribel Sánchez Gálvez F.C.C

*Becario Conacyt

Puebla, Pue. Enero 22 2021

Dedicatorias

A mis padres, a mis hermanas, a mi tía Elvira y a mi abuelita Josefina que sin importar la distancia ni lo efímero de nuestra existencia, siempre estaremos juntos.

Agradecimientos

A la Facultad de Ciencias de la Electrónica de la Benemérita Universidad Autónoma de Puebla, por darme la oportunidad de ingresar a la Maestría en Ingeniería Electrónica opción Instrumentación Electrónica.

Al CONACyT, por el apoyo otorgado en el proyecto de maestría: *"Desarrollo de un manipulador con telecontrol como robot de asistencia personal"*

Al M.C. Ricardo Álvarez González, al Dr. José Fermi Guerrero Castellanos y a la M.C. Alba Maribel Sánchez Gálvez, por su tiempo, sus asesorías y principalmente la confianza que me brindaron.

A mis sinodales, el Dr. Víctor Rodolfo González Díaz, el M.C. José Francisco Portillo Robledo y el M.C. Rodrigo Lucio Maya Ramírez, por los comentarios y consejos para la realización de este trabajo de tesis.

A los profesores y personal administrativo que forman parte de la Maestría en Ingeniería Electrónica, en especial a la M.C. Ana María Rodríguez Domínguez por su amabilidad y paciencia.

A mis amigos y compañeros; a Raúl Quechol Elías y a Ángel Juárez Palacios, amigos de vida. A mis compañeros del Laboratorio de Control Avanzado y Sistemas Ciber-Físicos; Mauricio Longinos Garrido y Néstor Antonio Tolentino Medrano que durante mi estancia en la maestría siempre me brindaron su apoyo.

A todos ellos, GRACIAS.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Introducción

La historia demuestra que a lo largo de la existencia del ser humano la población se encuentra en un constante crecimiento, entre muchas de las consecuencias de este suceso, los recursos naturales y las actividades humanas han estado experimentando cambios significativos.

La sobrepoblación mundial se ha convertido en un problema que cada vez más países deben enfrentar; principalmente la demanda de servicios, el aumento en la proporción de personas adultas (envejecimiento de sus habitantes), un creciente índice de esperanza de vida y un número cada vez mayor de hogares unipersonales [1], son algunos de los temas estudiados con el fin de conocer su origen y posibles consecuencias.

De manera particular, es posible concentrar un grupo formado de diferentes sectores de la población que debido a su condición requieren y demandan atenciones y servicios específicos, dentro de éste se encuentran personas que sufren deficiencias motoras o discapacidad en alguna de sus extremidades, debilitamiento físico y falta de fuerza muscular generalmente causados por degeneración de edad avanzada, problemas neurológicos [2], algunos causados por accidentes, entre otros. Bajo estas condiciones es posible observar que los miembros de este grupo se encuentran en un estado de dependencia y muchas veces requieren de asistencia para poder realizar sus actividades cotidianas.

Las causas son diversas, probablemente la principal sea la escasez de recursos económicos, sin embargo, es un hecho que no ha sido posible cubrir todas las necesidades de este sector de manera eficiente; este problema repercute directamente en la calidad de vida [3] y considerando que éste se encuentra en aumento [4]. Este trabajo de tesis propone un prototipo, que si bien no es una solución inmediata al problema, contribuye al desarrollo de la misma.

La capacidad tecnológica actual ofrece prometedoras soluciones a diversos aspectos de la actividad humana desde el área espacial, industrial, el hogar, automotriz, entre muchas otras. La finalidad de este trabajo consiste en realizar un aporte para que algún día, a través de la ciencia y la tecnología, la esperanza de vida más allá de ser una cifra en aumento tenga en su contenido un aumento en la calidad con la que se vive.

Los sistemas automáticos enfocados a la asistencia han sido ya objeto de estudio varios años atrás [2], en el mercado existen algunos en venta, sin embargo, el diseño que se considera implementar en esta tesis consiste en una arquitectura paralela no utilizada en los sistemas actuales, esta arquitectura robótica es ampliamente utilizada en el sector industrial principalmente en procesos de tipo *pick&place* por sus ventajas como precisión, eficiencia energética, mayor rigidez en su estructura, altas velocidades de operación, entre otras.

En este trabajo se describe el diseño, implementación y configuración de un prototipo cuya función es en esencia manipular objetos, es un sistema que pretende formar parte de otro mas complejo y con mayor funcionalidad, precisamente como un asistente personal, sin embargo, las posibilidades de ser útil para otros propósitos no se descartan.

Además del manipulador, el prototipo consta de un sistema de visón por computadora que complementa su funcionalidad, haciendo uso del lenguaje Python y librerías como *OpenCV* y *rospy*, éste proporciona la posición del objeto de interés que después de ser procesada por la cinemática inversa del robot, el efector final del mismo se coloca en la posición deseada. El fuljo de datos y la conexión de los sistemas es gestionada con ROS.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar e implementar el prototipo de un sistema con la capacidad de manipular objetos mediante una interfaz remota, para la asistencia de personas con movilidad limitada en sus extremidades inferiores.

1.2.2. Objetivos particulares

- Analizar herramientas y técnicas de visión por computadora y procesamiento de imágenes para implementar un sistema de detección de contornos.
- Analizar la arquitectura de manipulación robótica tipo Delta para así acoplar un prototipo al sistema
- Construir el prototipo de un sistema manipulador para mover objetos previamente identificados (botella). Mostrar su modelado matemático correspondiente.
- Analizar e implementar el algoritmo de cómputo necesario para controlar el sistema robótico manipulador. Se utilizarán librerías de OpenCV en el lenguaje de programación Python
- Analizar e implementar la conexión entre todos los subsistemas.

1.3. Justificación

Debido a diversas circunstancias algunas personas pierden la capacidad de ser autosuficientes y requieren de ser asistidas ya sea a corto plazo o de manera permanente, comúnmente se mantienen cuidados específicos, el de mayor prioridad es la demanda de atención y vigilancia; sin embargo, es una tarea demandante y no siempre es posible brindar los cuidados necesarios, es así como desde la perspectiva de la robótica, se plantea la posibilidad de contribuir a esta problemática a través del uso de sistemas asistentes robóticos enfocados al cuidado de personas. Los sistemas robóticos se incrementa gradualmente y se estima que al final de la próxima década los robots tendrán una presencia en muchas de las actividades que hoy realizan los seres humanos [5].

Ya es posible encontrar en el mercado varios modelos de sistemas dirigidos a la asistencia en el hogar, resultan ser de gran utilidad y con el avance de nuevas tecnologías serán capaces de realizar tareas cada vez más complejas. Este trabajo de tesis plantea implementar un prototipo que proporcione un aporte funcional y sea la plataforma de proyectos futuros utilizando las herramientas y recursos disponibles que, a diferencia de los sistemas actuales que utilizan estructuras de tipo serie, implemente un manipulador tipo paralelo conocido como manipulador paralelo *Delta*, las ventajas propias del robot paralelo son heredadas al prototipo implementado, además de que se abren nuevas posibilidades y capacidades.

Con un tamaño reducido, pensando en el interior de un hogar, la estructura paralela amplía el rango de objetos posibles a tomar ya que tiene la capacidad de manipular objetos de mayor peso respecto a una estructura tipo serie, mientras esta última requiere de actuadores mas potentes, que se traducen en mayor consumo energético, tamaño, peso y costo, el robot delta se apoya de tres cadenas cinemáticas con tan solo tres actuadores (igual o menos que los requeridos por un brazo robótico tipo serie), para manipular con precisión y rapidez eliminando la inercia de los eslabones de una estructura serie.

1.4. Descripción funcional del prototipo

El prototipo desarrollado en este trabajo de tesis es descrito de manera funcional a través del esquema mostrado en figura 1.1. Es importante mencionar que el prototipo no comprende la función de desplazamiento.

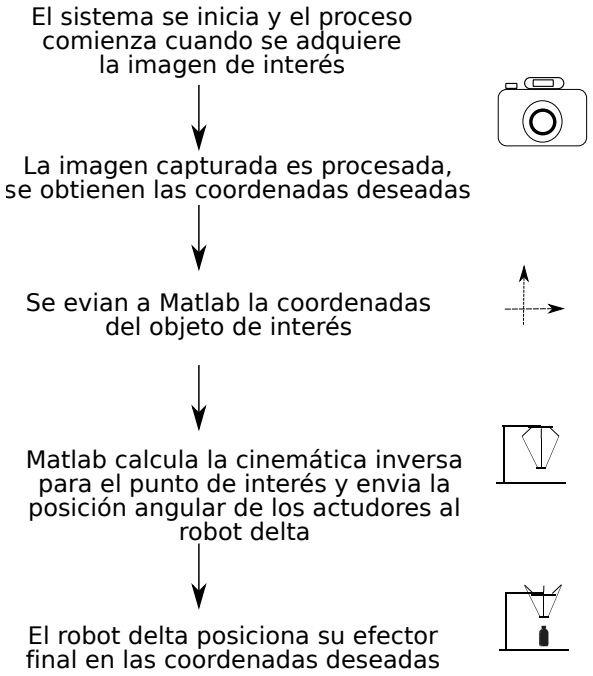


Figura 1.1: Proceso del sistema de asistencia

La implementación de este prototipo comprende diferentes etapas que serán resueltas por separado. La figura 1.2 muestra el bloque funcional del prototipo de manera más detallada. Es posible que en algunos bloques no se especifique su contenido ya que durante el desarrollo se decidirán las opciones que sean más adecuadas.

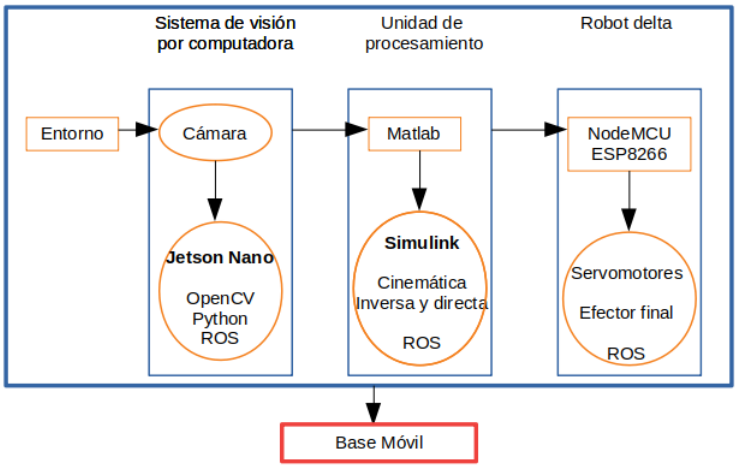


Figura 1.2: Diagrama de bloques funcionales

En la figura 1.2 se observa un bloque en color rojo que lleva por nombre **Base Móvil**, este bloque no se implementa en este trabajo de tesis.

1.5. Organización de la tesis

El resto del documento se encuentra estructurado de la siguiente manera:

- Capítulo 2: Se realiza un breve estudio de los hechos históricos y principios teóricos que dan fundamento al prototipo que se implementó. Debido a que se ocupan diversas áreas del conocimiento, está dividido en varias secciones.
- Capítulo 3: Aquí se describe el procedimiento realizado para la implementación y análisis del robot paralelo delta.
- Capítulo 4: Se realiza la descripción del desarrollo realizado para implementar el sistema de visión por computadora.
- Capítulo 5: Se presenta la forma en que se realizó la conexión de los sistemas presentados en los capítulos 3 y 4.
- Capítulo 6: El resultado de conectar los sistemas descritos en los capítulos anteriores se presentan aquí realizando las observaciones correspondientes.

- Capítulo 7: Para este capítulo se realizan conclusiones de manera puntual del prototipo realizado, además, se plantean posibles trabajos futuros para el mismo.
- Finalmente al termino del documento, se muestra la bibliografía utilizada y se anexan algunos apéndices.

Capítulo 2

Estado del arte, fundamentos y antecedentes

La primera sección de este capítulo presenta algunos sistemas de asistencia personal que se distribuyen en el mercado actualmente, desde asistentes de voz hasta complejos asistentes de telepresencia capaces de manipular una gran cantidad de objetos.

Posteriormente, se realiza una descripción de los antecedentes y fundamentos teóricos que dan sustento a los sistemas utilizados para llevar a cabo la implementación del prototipo planteado.

El trabajo requerido para alcanzar los objetivos establecidos está formado por tres áreas como se muestra en la figura 2.1, es así como se analizará cada una de estas tres partes por separado hasta el capítulo 6 donde se realiza la conexión de éstos.

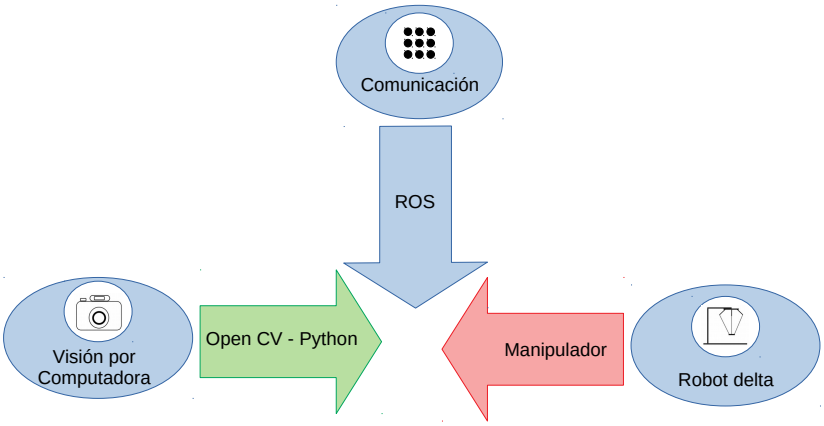


Figura 2.1: Sub sistemas

2.1. Sistemas de asistencia

Los robots asistentes son sistemas que han sido objeto de estudio desde hace ya varios años; los productos que se ofertan poseen una gran variedad de características y funciones con el fin de ampliar su mercado, éste se compone de personas que se encuentran limitadas parcialmente de movimiento; personas que por efecto de la edad se encuentran en una etapa de deterioro físico y cognitivo; o simplemente personas que desean tener una organización más eficiente de sus tareas y su tiempo. En el mercado se encuentran diversos productos interesantes y se ha realizado una clasificación de ellos de la siguiente manera:

- Asistentes de voz
  - Google Home
  - Amazon Echo
  - Google Assistant
  - Siri (HomePod)

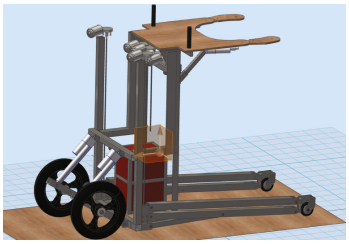
    - Cortana
    - Bixby
    - Mycroft Mark II
    - Movistar Home
- Asistentes de limpieza
    - Robots Aspiradores(iRobot Roomba 980,iRobot Braava, Samsung Powerbot R7065, Ecovacs Deebot M81 Pro, Eufy Robovac 11s, Neato Botvac D7)
- Asistentes para rehabilitación
    - ReWalk Personal 6.0 (Exoesqueleto)
    - ReStore Exo-Suit (Exoesqueleto)
    - Walking Assistant Robot [7]
    - A NAO robot [8]
    - ZORA [9]
    - Entre otros
- Asistentes para la enseñanza
    - Qbo Robot
    - Elinor [10]
    - RASA [11]
    - EASEL [12]
    - NAO [13]
    - ZORA [14]
    - Entre otros
- Asistentes en el área de la medicina
    - Da Vinci Surgical System

- Blood-drawing assistant
- RIO Robotic Arm Interactive Orthopedic System
- MiroSurge robotic system
- CorPath 200
- SPRINT (Single Port LapaRoscopy bImaNual roboT)

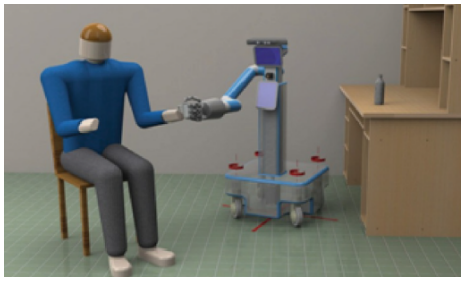
- Asistentes para personas con movimiento motriz limitado dentro del hogar
  - WAR [14]
  - Assistant Personal Robotic (APR) [15]
  - Rehabilitation Walker [16]
  - RAMCIP [17]
  - Care-o-Bot [18]

Siendo el último grupo el de mayor interés, se muestran en la figura 2.2 algunos ejemplos de éstos.

(a) Assistant Personal Robotic (APR) [15]



(b) Rehabilitation Walker [16]



(c) RAMCIP [17]

(d) Care o Bot[18]

Figura 2.2: Asistentes personales automatizados

Estos robots poseen la cualidad de estar diseñados bajo la sinergia de los diferentes sub-sistemas que los componen, las áreas que dan fundamento a estos complejos sistemas se muestran en la figura 2.3.

Figura 2.3: Integración de diferentes áreas

2.2. Robot paralelo delta

Los sistemas mecatrónicos surgen de la sinergia entre los sistemas mecánicos, electrónicos, informáticos y sistemas de control. Una adecuada integración de todos estos sistemas es lo que se conoce como Mecatrónica, aunque existen en la literatura diversas definiciones de este concepto, todas convergen en conseguir una óptima adaptación de todos estos sistemas trabajando en conjunto, ver figura 2.4.

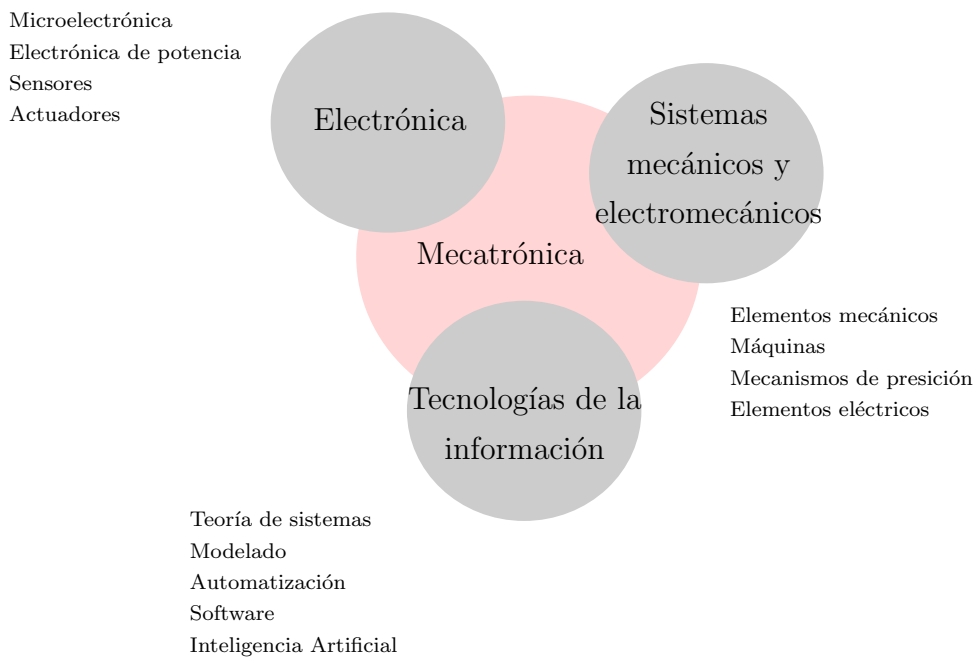


Figura 2.4: Sistemas mecatrónicos, un campo multidisciplinario

Existen dos maneras, las más comunes, de realizar la integración de un sistema mecatrónico:

- Integración de componentes (nivel Hardware): Consiste en diseñar e integrar de manera eficiente y óptima el sistema electrónico (sensores, actuadores y microcontroladores) al proceso mecánico.
- Integración por procesamiento de información (nivel Software): consiste en implementar estrategias de control avanzadas. Esto incluye la solución de tareas como supervisión con diagnóstico de fallas, optimización y gestión general de procesos.

Los sistemas mecatrónicos han mejorado el proceso de muchas tareas, sin embargo, también han sido capaces de realizar nuevas tareas que anteriormente parecían imposibles o simplemente eran desconocidas. El robot paralelo delta es un sistema mecatrónico ampliamente utilizado en áreas de tipo industrial, su arquitectura se compone de dos bases; la base más grande se encuentra fija mientras que la base de menor tamaño es móvil dándole soporte al efector final (el tipo de efector final

depende de la aplicación), ambas bases se encuentran en una relación paralela unidas a través de tres brazos articulados o cadena cinemáticas controladas cada uno por un actuador [24], ver figura 2.5.

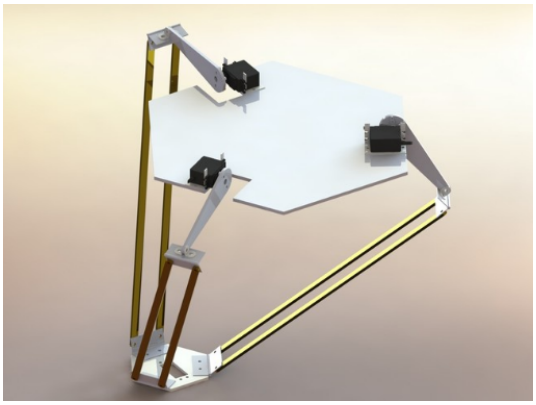
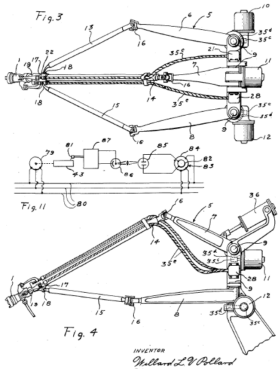


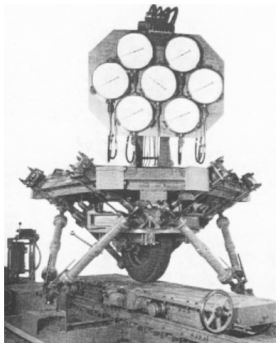
Figura 2.5: Robot paralelo delta

Los robots paralelos pueden ser clasificados de acuerdo con la cantidad de grados de libertad que poseen (de los 3 hasta los 6 grados de libertad). El Robot tipo paralelo Delta se encuentra dentro de los manipuladores de 3 grados de libertad, también conocido como de traslación. La arquitectura paralelo delta tiene la desventaja de contar con un espacio de trabajo reducido a comparación de otras estructuras, sin embargo, esta estructura proporciona mejoras en características como precisión, mayor capacidad de carga, mayor rigidez, mayor fuerza y velocidad. Algunas de las áreas de aplicación de estos sistemas se encuentran en el sector industrial realizando operaciones *pick and place*; en el sector aeroespacial como simuladores de vuelo y en el sector de investigación buscando nuevas y novedosas aplicaciones.
Esta arquitectura ha sido objeto de estudio desde hace ya varios años. La primera implementación documentada es trabajo del inventor de origen estadounidense Willard LV Pollard en los inicios de la década de 1940 que, con el objetivo de pintar superficies irregulares, crea un dispositivo para controlar el movimiento y posición de una pistola aerosol. Ver figura 2.6 (a). Posteriormente en el año de 1947 el Dr. Eric Gough diseñó un octaedro hexápodo con lados de longitud variable como se observa en la figura 2.6 (b). El objetivo de este dispositivo fue el de realizar un análisis del comportamiento de neumáticos para la empresa *Dunlop*; actualmente se utilizan

plataformas que trabajan bajo este mismo principio *MAST (Multi-Axis Simulation Table)* [30].



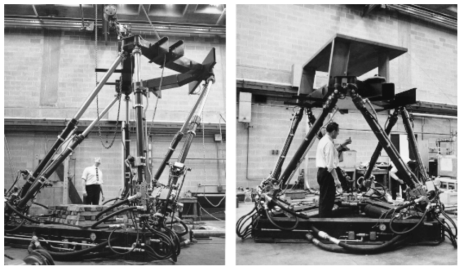
(a) Robot de 5 grados de libertad



(b) Primer hexápodo octaédrico en 1954

Figura 2.6: Primeros antecedentes de los robots paralelos

Fue hasta el año de 1965 cuando Mr. Stewart presenta un artículo donde describe una plataforma de 6 grados de libertad con el objetivo de servir como simulador de vuelo, este trabajo se ha convertido en una referencia dentro del estudio de plataformas paralelas, ver figura 2.7 (a). En el mismo contexto temporal, el ingeniero Klaus Cappel realiza varios estudios relacionados con plataformas paralelas construyendo muchas de ellas, ver figura 2.7 (b) [25].



(a) Primer simulador de vuelo

(b) Klaus Cappel y su hexápodo octaédrico

Figura 2.7: Primeros antecedentes de los robots paralelos

Sin embargo, la introducción de este diseño al sector industrial se realizó hasta 1979 cuando McCallion y Pham propusieron utilizar este sistema para una célula

de ensamblaje. Fue entonces cuando el Dr. Raymond Clavel desarrolla lo que hoy se conoce como el robot paralelo delta presentado en el año de 1985,ver figura 2.8.

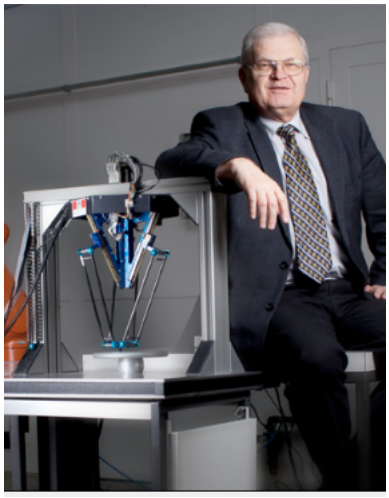
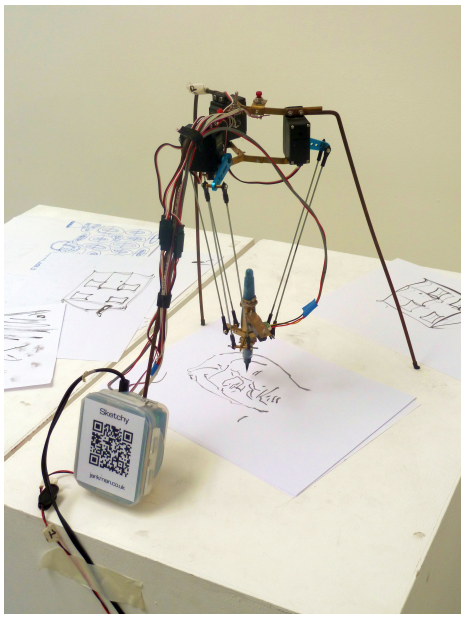
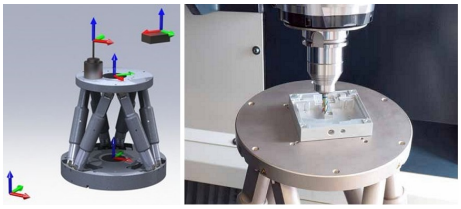


Figura 2.8: Raymond Clavel creador del Robot Paralelo Delta

Los robots paralelos tienen una arquitectura cinemática cerrada, en la cual se obtienen múltiples caminos que unen dos puntos del mecanismo, permitiendo mayores velocidades en el espacio de trabajo, una mayor rigidez estructural y altas velocidades frente a los robots de cadenas cinemáticas abiertas [19]. Gracias a las diferentes topologías de este tipo de sistemas es que existe una gran variedad de aplicaciones como micro robots posicionadores hasta grandes plataformas de gran capacidad de carga, desde aplicaciones médicas hasta simuladores de vuelo, desde dispositivos hápticos a robots manipuladores, desde robots experimentales a máquinas herramientas de control numérico y aplicaciones militares, en las figuras 2.9 y 2.10 se muestran algunas de estas aplicaciones.



(a) Desarrollo de nuevas aplicaciones

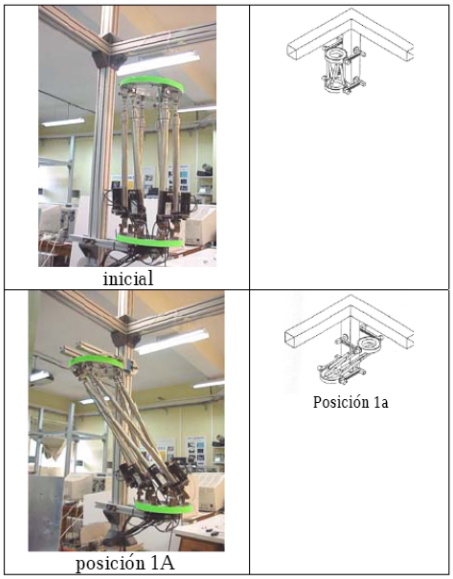


(b) Maquinado de piezas

(c) Proceso de "pick and place.[en] el sector industrial

(d) Simulador de vuelo

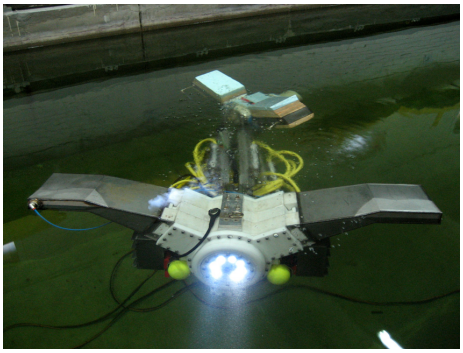
Figura 2.9: Aplicaciones de robots paralelos



(a) Robot paralelo trepador

(b) Posicionamiento de antenas parabólicas

(c) Simulador de movimiento (con fines didácticos y de entretenimiento)



(d) Prototipo REMO, robot submarino operado remotamente

Figura 2.10: Aplicaciones de robots tipo paralelo

2.3. Visión por computadora

La visión por computadora es un área de la informática la cuál permite que dispositivos vean, comprendan e interpreten aquello que, equipados con cámaras, se encuentren viendo.
Las técnicas utilizadas por los sistemas de visión por computadora permiten enfocarse en resolver problemas cada vez más complejos. Gracias a la inteligencia artificial, se han ampliado aplicaciones como autos de conducción autónoma, eliminar ruido de fondo de una imagen, resaltar alguna forma determinada dentro de una imagen, tomar fotografías de buena calidad en movimiento, estimar distancias, estimar texturas, reconocer patrones, reconocer estructuras y objetos dentro de una imagen.
En este trabajo de tesis, cuando el prototipo adquiere la imagen se procesa a través de Python haciendo uso del paquete *OpenCV*. En este trabajo de tesis se aplican tres diferentes técnicas: Detección de contornos, identificación de color y localización de centroides. Una vez que la imagen es adquirida, se procesa a través de Python haciendo uso del paquete *OpenCV*.

2.3.1. Detección de contornos

Un contorno se caracteriza y localiza por un cambio abrupto en el color, para hacer esto más sencillo y reducir la cantidad de información resulta útil representar una imagen de color en solo escala de grises, de esta manera es posible que se pueda detectar el cambio de tonalidad de un píxel oscuro a uno claro y viceversa. Los algoritmos que realizan esta tarea se encuentran fundamentados matemáticamente a través de derivadas, ver figura 2.11
La técnica de detección de contornos se realiza mediante el cálculo del gradiente de la imagen en dos direcciones ortogonales, éstas son las derivadas parciales en determinada dirección que caracterizan la tasa de variación de una función a lo largo de rectas paralelas a los ejes coordenados [20].

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \vdots \end{bmatrix}$$

Figura 2.11: Representación del Gradiente de una función de dos o más variables

En la ecuación 2.1 se muestra una expresión que representa la derivada parcial con una dirección específica, la ecuación 2.2 muestra la misma en su forma compacta. Se aprecia que, en la expresión 2.1 el termino después del signo igual consta de dos factores multiplicándose, el segundo representa el gradiente de la función.

$$D_u f(x,y) = (\cos(\theta i) + \sin(\theta i)) * [f_x(x,y)i + f_y(x,y)i] \tag{2.1}$$

$$D_u f(x,y) = U \cdot \nabla_f(x,y) \tag{2.2}$$

De la ecuación 2.2 se observa que la derivada direccional se puede obtener mediante el producto punto del vector gradiente con el vector unitario según la dirección que se requiera.
Entonces, la derivada de una función facilita las variaciones locales con respecto a la variable, de esta manera el valor de la derivada es mayor cuanto más rápidas son las variaciones, por lo que este concepto es utilizado para detectar las variaciones abruptas de los valores de cada píxel dentro de una imagen y así detectar contornos o bordes.

El proceso para la detección de bordes comúnmente comprende tres etapas principales:

- El pre-procesamiento: durante esta etapa la imagen se acondiciona de tal manera que las características deseadas a analizar sean prioritariamente expuestas. Regularmente se realizan métodos de tipo promedio, suavizado y filtrado de imágenes. Algunos de estos sólo se mencionan a continuación:

- Operaciones Puntuales: Función definida a intervalos, Negativo de una imagen, Extracción de bits, Procedimientos basados en histogramas.
- Operaciones espaciales: Filtrado espacial, Suavizado direccional, Filtrado de mediana, Ampliación de imágenes.
- Operaciones transformadas.
- Restauración de imágenes.
- Pseudocolor: Falso color, Pseudocolor.

- Detección de bordes: en esta etapa se realiza una búsqueda de elementos de borde que incluye la medición de su intensidad y orientación.
- Extracción: esta tercera etapa incluye marcar un borde y realizar el procesamiento del umbral con un elemento de borde fijo si su intensidad supera un determinado umbral.

Se pueden encontrar actualmente una gran cantidad de algoritmos para la detección de bordes o contornos dentro de una imagen, la mayoría de tipo lineal. Estos métodos lineales están basados en la primera derivada del brillo de la imagen. El funcionamiento se describe de la siguiente manera:

El gradiente de una imagen en cualquier punto se define como un vector bidimensional dado por la ecuación 2.3.

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\sigma}{\sigma x} f(x,y) \\ \frac{\sigma}{\sigma y} f(x,y) \end{bmatrix} \tag{2.3}$$

De la ecuación 2.3, el vector G apunta en la dirección de variación máxima de f en el punto (x, y) por unidad de distancia con la magnitud y dirección definidas por las ecuaciones 2.4 y 2.5, ver la figura 2.12.

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{2.4}$$

$$\phi(x,y) = \tan^{-1}\left[\frac{G_y}{G_x}\right] \tag{2.5}$$

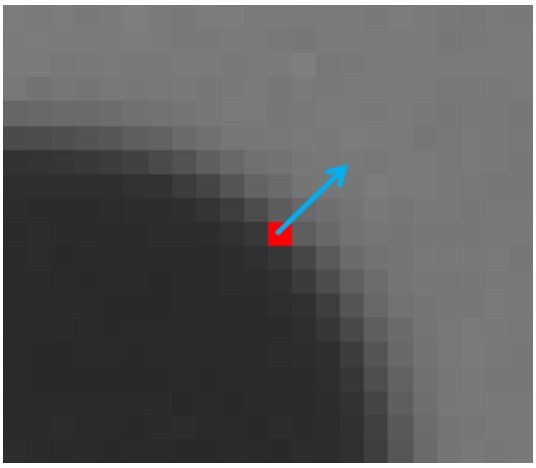


Figura 2.12: Cambio en la escala de grises a nivel de píxeles

Generalmente, en la práctica, se suele aproximar la magnitud del gradiente con valores absolutos como se ve en la ecuación 2.6, si el valor de la magnitud supera el umbral de este punto se considera como un borde.

$$|G| \approx |G_x| + |G_y| \tag{2.6}$$

Para encontrar la derivada de la ecuación (2.3), se utilizan las diferenciales de primer orden entre dos píxeles adyacentes, ver las ecuaciones 2.7 y 2.8. Esta es la forma básica de obtener el gradiente en un punto.

$$G_x = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \tag{2.7}$$

$$G_y = \frac{f(y + \Delta y) - f(y - \Delta y)}{2\Delta y} \tag{2.8}$$

El proceso descrito en la ecuación 2.7 y 2.8 se repite para todos los puntos de la imagen. El resultado de este procedimiento tiene solo dos posibles salidas, ver ecuación 2.9.

$$g(x, y) = \begin{bmatrix} 1 \; si \; G[f(x; y)] > T \\ 0 \; si \; G[f(x; y)] \leq T \end{bmatrix} \tag{2.9}$$

De la ecuación 2.9 se tiene que, T es un valor de umbral no negativo. Entonces, sólo los píxeles de borde que superen el valor de T se considerarán importantes [20], ver la figura 2.13.

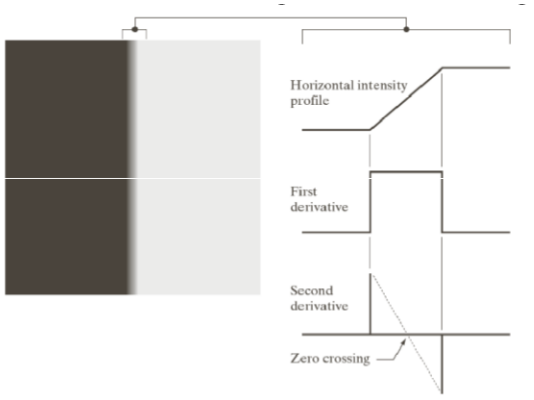


Figura 2.13: Primera y segunda derivada de un borde

Estos procesos se encuentran optimizados dentro de paquetes o bibliotecas de algoritmos que optimizan la programación, uno de los más populares es *OpenCV*, contiene muchas herramientas que facilitan el desarrollo de aplicaciones de visión por computadora, por sus siglas en inglés (*open source computer vision*) se define como una herramienta de libre acceso para tarea de visión por computadora; soporta lenguajes como C/C++, Python y Java la primera versión salió a la luz en el año de 2006, misma que estuvo disponible para sistema Linux, Mac y Windows. Actualmente para Android e iOS, también.
Esta tesis utiliza *OpenCV* en lenguaje *Python* ya que posee ventajas como: Flexibilidad, portabilidad, es de código abierto, existe una gran paquetería disponible, tiene un gran comunidad de programadores que respaldan su desarrollo, entre otras.

2.3.2. Identificación de color

Cada píxel en una imagen contiene en determinada medida los tres colores base del formato *RGB*: Rojo, verde y azul. En OpenCV cada píxel contiene estos tres colores en diferente proporción que se define de acuerdo a un valor que oscila entre 0 y 255 siendo el valor más alto una mayor intensidad de color. En este prototipo se eligió detectar el color azul [21].

Dentro de OpenCV cada píxel es un vector de tres componentes (rojo, verde y azul), para detectar un color en específico se requiere realizar una discriminación píxel por píxel donde se establece un rango deseado para cada componente. Sí los tres diferentes valores de cada componentes del píxel se encuentran dentro del rango, se puede determinar que ese píxel tiene un color en específico. La función es simple, realizar un filtro que divida en dos toda la imagen.
En la figura 2.14 se muestra el resultado de la selección. La imagen resultante es una máscara para el color azul, se han obtenido los objetos con el color deseado [22].

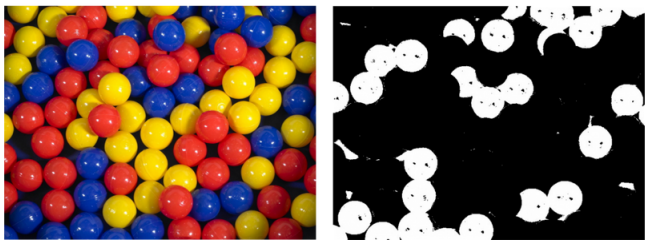


Figura 2.14: Máscara binaria

Posteriormente se aplican otros procedimientos con el fin de seguir con el procesamiento según sea el objetivo deseado.

2.3.3. Localización de centroides

En este trabajo, para encontrar el centro de un objeto se deberá primero encontrar un contorno y posteriormente calcular su centro. Se utiliza la teoría referente al *centro de gravedad de un objeto* [23].
OpenCV contiene paquetes que facilitan la implementación de esta función, se fundamenta en tres momentos (M_{00},M_{01} y M_{10}), éstos son una medida que indica la dispersión de una nube de puntos, se define como la ecuación 2.10.

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(xy) \tag{2.10}$$

De la ecuación 2.10, x y y son las coordenadas de un píxel y la función $I(xy)$ indica su intensidad, esta última solo puede tomar valores de 0 ó 1 ya que la imagen sobre la que se trabaja es de tipo binaria.

El primer momento a calcular es el de orden 0 y coincide con el área del objeto en píxeles, ver ecuación 2.11, ya que $I(x, y)$ solo tiene valores 0 ó 1, el resultado de $M00$ es equivalente al número de píxeles cuyo valor es 1.

$$M_{00} = \sum_x \sum_y x^0 y^0 I(xy) = \sum_x \sum_y I(xy) \tag{2.11}$$

El segundo momento a calcular es de orden 1 M_{10}, realizando la sustitución se tiene la ecuación 1.12. Entonces, M_{10} es igual a la suma de las coordenadas x de los píxeles cuya intensidad sea 1, por lo tanto si se divide M_{10} entre M_{00} se obtiene el centroide para la componente en x, ver ecuación 2.13. La misma lógica se utiliza para el tercer momento M_{01}, ver ecuaciones 2.14 y 2.15.

$$M_{10} = \sum_x \sum_y x^1 y^0 I(xy) = \sum_x \sum_y x I(xy) \tag{2.12}$$

$$\frac{M_{10}}{M_{00}} = \frac{\sum_x \sum_y xI(xy}{\sum_x \sum_y I(xy} = \frac{x_1 + x_2 ... x_n}{n} = \bar{x} \tag{2.13}$$

$$M_{01} = \sum_x \sum_y x^0 y^1 I(xy) = \sum_x \sum_y y I(xy) \tag{2.14}$$

$$\frac{M_{01}}{M_{00}} = \frac{\sum_x \sum_y yI(xy}{\sum_x \sum_y I(xy} = \frac{y_1 + 2_2 ... y_n}{n} = \bar{y} \tag{2.15}$$

Este cálculo viene integrado en Python en el comando *momentos*, las líneas de código que representan a la ecuación 2.13, por dar un ejemplo, serían:
cx = momentos['m10']/momentos['m00']

2.4. Comunicación

Como se mencionó en el inicio de este capítulo, el prototipo requiere de diferentes sistemas para su funcionamiento, incluso para conectar todos estos sistemas se hace uso de otro más.

2.4.1. ROS

ROS, por sus siglas en inglés *Robot Operating System*, se define como un meta sistema operativo que contiene herramientas para ayudar a los desarrolladores de software a crear aplicaciones para robots, tiene la cualidad de enlazar sistemas y procesos con abstracción de hardware para el programador, es decir, sin importar el tipo de dispositivo o ambiente de trabajo es posible conectar éstos [26], ver figura 2.15.

Figura 2.15: Robotic Operating System

Una de las princípiales ventajas de éste es que es de código abierto. Algunas de sus características principales son las siguientes [27]:

- Comunicación entre procesos. Sin importar la naturaleza del dispositivo, dos o mas procesos pueden interactuar.
- Proporciona funcionalidades de un sistema operativo como: sub-procesamiento múltiple, control de dispositivos de bajo nivel, administración de paquetes y abstracción de hardware.
- Soporte y herramientas para diferentes lenguajes de programación.
- Disponibilidad de bibliotecas de terceros, por ejemplo, OpenCV.
- Contiene algoritmos implementados (PID, SLAM, entre otros).
- Algoritmos estándar. Es posible crear prototipos rápidamente utilizando repositorios de ROS.
- ROS es soportada por una gran comunidad de desarrolladores en todo el mundo.
- Contiene muchas herramientas de línea de comandos y GUI para depurar, visualizar y simular aplicaciones de robots.
- Está bajo licencia *open source*.

Debido a las ventajas y lo interesante que representa utilizar ROS, este proyecto hace uso de algunas de sus funcionalidades; temas como la instalación, configuración y puesta en marcha de ROS en los diferentes dispositivos utilizados en este trabajo, no se describen, sin embargo, todas las fuentes utilizadas se encuentran en la sección bibliográfica.

Capítulo 3

Diseño e implementación de la plataforma robótica

En este capítulo se describe el procedimiento realizado en la implementación del robot paralelo delta, desde el análisis de la cinemática hasta la construcción del mismo.
El robot paralelo Delta se puede definir como un sistema conformado por dos bases que guardan una relación paralela entre sí; estas dos bases se encuentran unidas a través de tres cadenas cinemáticas equidistantes, la base superior se encuentra fija, suspendida por un soporte externo mientras que la base inferior tiene la facultad de desplazarse por toda el área de trabajo siendo en ésta donde se coloca el efector final, ver figura 3.1.

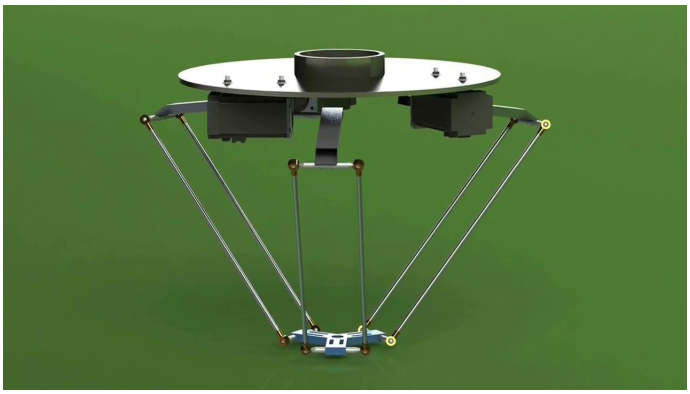


Figura 3.1: Robot paralelo delta

3.1. Cinemática

En esta tesis, el análisis del robot se basa en estudiar su movimiento, por lo tanto es necesario hablar de la cinemática del mismo, el método utilizado es de tipo *Geométrico*. Existen dos maneras de ver la cinemática, de manera directa e inversa, debido a la función deseada para este robot, se tiene mayor interés por la cinemática inversa, sin embargo, ambas se relacionan como se muestra en la figura 3.2.

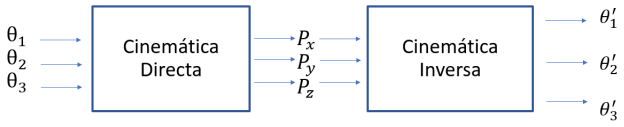


Figura 3.2: Cinemática Directa e Inversa.

Como se observa en la figura 3.2, la cinemática directa requiere conocer la posición angular de los tres actuadores para calcular la posición espacial del efector final y la cinemática inversa, por el contrario, requiere de las coordenadas espaciales del efector final para calcular la posición angular de los tres actuadores.
Ya que el sistema de visión por computadora será el que proporcione las coordenadas espaciales, existe un gran interés por calcular los valores de los ángulos $\theta_1, \quad \theta_2 \quad y \quad \theta_3$, de esta manera se describe la cinemática inversa para el robot paralelo delta.

3.1.1. Cinemática Inversa

En la figura 3.3 se presenta una vista de perfil de una cadena cinemática del robot, este análisis se realiza para una sola pierna y es reutilizado para las otras dos.

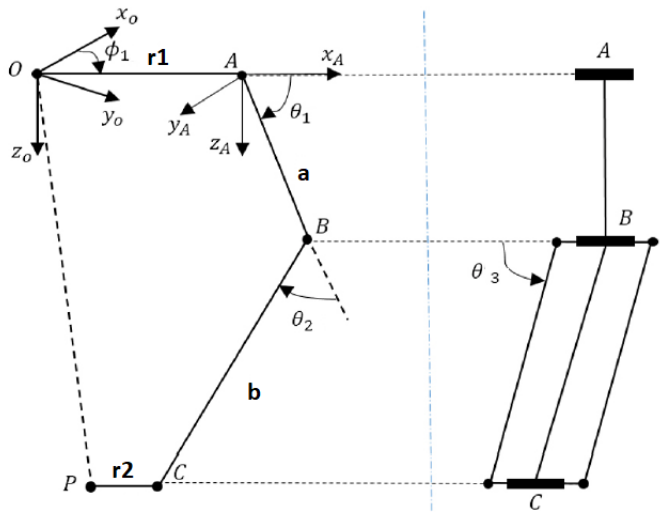


Figura 3.3: Vista frontal y lateral de una pierna

De la figura 3.3 se observan varios parámetros importantes que se definen a continuación:

- r1: Distancia desde el origen de la plataforma fija hasta la primera articulación (punto A).
- r2: Distancia desde el origen de la plataforma móvil hasta la tercera articulación (punto C).
- a: Longitud del brazo superior.
- b: Longitud del brazo inferior.
- O: Origen del sistema
- P: Posición del efector final
- A: Reubicación del origen
- θ_1: Ángulo actuado
- ϕ: Ángulo de cada pierna respecto al eje x

El análisis geométrico consiste en observar los puntos estratégicos, O, A, B, C y P; forman segmentos bien definidos ($\overline{OA}$, $\overline{AB}$, $\overline{BC}$, $\overline{CP}$, $\overline{OP}$) que observando las trayectorias se forman dos caminos diferentes del punto O al punto P, ver ecuación 3.1.

$$\overline{OA} \quad + \quad \overline{AB} \quad + \quad \overline{BC} \quad + \quad \overline{CP} \quad = \quad \overline{OP} \tag{3.1}$$

Para facilitar el análisis geométrico se traslada el origen del punto O al punto A, por lo tanto, la trayectoria que describe el recorrido del punto A al punto C se muestra en la ecuación 3.2.
El siguiente paso es encontrar el valor de C ya que conociendo éste es posible encontrar las expresiones matemáticas que dan solución a los valores de los ángulos θ, así, se estaría resolviendo la cinemática inversa para una cadena cinemática, el proceso es el mismo para las tres cadenas.

$$\overline{AB} \quad + \overline{BC} \quad = \quad \overline{OP} \quad - \quad \overline{OA} \quad + \quad \overline{CP} \tag{3.2}$$

La ecuación 3.2 se puede reescribir con los parámetros geométricos del mecanismo como se muestra en la ecuación 3.3 y 3.4. En este punto es importante mencionar que se debe añadir una matriz de rotación ya que el valor del ángulo ϕ es diferente para cada cadena cinemática (más adelante se explica por qué).

$$\begin{bmatrix} acos(\theta_1) \\ 0 \\ asin(\theta_1) \end{bmatrix} + \begin{bmatrix} bsin(\theta_3)cos(\theta_1+\theta_2 \\ bcos(\theta_3) \\ bsin(\theta_3)sin(\theta_1+\theta_2 \end{bmatrix} = \tag{3.3}$$

$$\begin{bmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - \begin{bmatrix} r1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} r2 \\ 0 \\ 0 \end{bmatrix} \tag{3.4}$$

De la ecuación 3.3, 3.4 y considerando el punto A como origen, es posible encontrar expresiones que calculen el valor para C, ver las ecuaciones 3.5, 3.6 y 3.7.

$$AB \quad + \quad BC = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = \begin{bmatrix} acos(\theta_1) + bsin(\theta_3)cos(\theta_1 + \theta_2) \\ bcos(\theta_3) \\ asin(\theta_1) + bsin(\theta_3)sin(\theta_1 + \theta_2) \end{bmatrix} = \tag{3.5}$$

$$\begin{bmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - \begin{bmatrix} r1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} r2 \\ 0 \\ 0 \end{bmatrix} \tag{3.6}$$

Por lo tanto:

$$\begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = \begin{bmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - \begin{bmatrix} r1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} r2 \\ 0 \\ 0 \end{bmatrix} \tag{3.7}$$

Con la ecuación 3.7 se tiene una expresión matemática con la cual es posible calcular el punto C ya que, ϕ, P, $r1$ y $r2$ son valores conocidos.
Las ecuaciones anteriores son el fundamento para finalmente deducir expresiones que calculen θ_1, θ_2 y θ_3 de la figura 3.3, cada una de éstas depende de algún valor del punto C, por ello la necesidad de obtener la ecuación 3.7.
De la ecuación 3.5 se observa que es posible obtener el valor de θ_3 de manera directa realizando un despeje del segundo renglón de la matriz, ver ecuación 3.8.

$$C_y = bcos(\theta_3) \quad \Longrightarrow \theta_3 = \arccos\left(\frac{C_y}{b}\right) \tag{3.8}$$

Mientras que para obtener θ_2 se realiza la suma de cuadrados de todas las filas de la matriz de la ecuación 3.5; después de realizar las operaciones matemáticas el resultado se expresa en la ecuación 3.9.

$$\theta_2 = \arccos\left(\frac{C_x + C_y + C_z + a^2 + b^2}{2ab\sin(\theta_3)}\right) \tag{3.9}$$

Finalmente para encontrar θ_1 se realiza la suma de $C_x, \quad C_y \quad y \quad C_z$ como se muestra en la ecuación 3.10, esta ecuación debe resolverse para θ_1, sin embargo, ya que las operaciones matemáticas son extensas se realizaron con la ayuda de *Matlab*.

$$C_x+C_y+C_z = acos(\theta_1)+bsin(\theta_3)cos(\theta_1+\theta_2)+bcos(\theta_3)+asin(\theta_1)+bsin(\theta_3)sin(\theta_1+\theta_2) \tag{3.10}$$

De esta manera se ha obtenido el ángulo actuado θ_1 para una cadena cinemática, los ángulos θ_2 y θ_3 no son actuados pero se utilizaron para llegar a la expresión de la ecuación 3.10. Este mismo procedimiento se repite para las dos cadenas cinemáticas restantes, sin embargo, el ángulo ϕ es diferente para cada cadena ya que su posición respecto al eje x es diferente, ver figura 3.4.

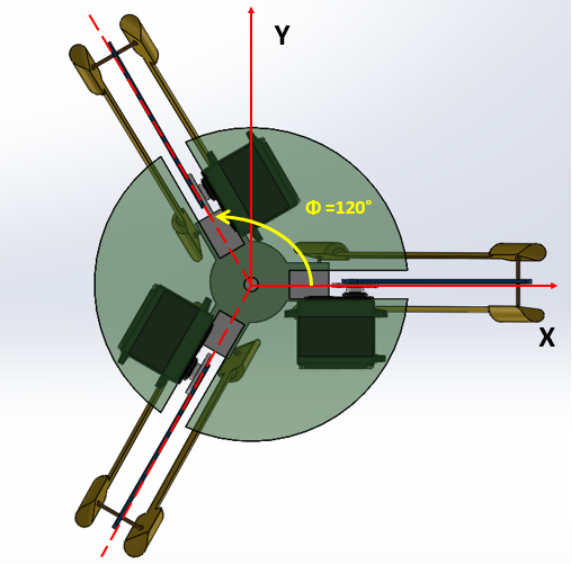


Figura 3.4: Vista superior

En la figura 3.4 se muestra la vista superior desde el plano xy del robot delta. Se observa que la cadena cinemática alineada con el eje x tiene un ángulo $\phi = 0$, siguiendo el recorrido antihorario se encuentra la segunda cadena cinemática con $\phi = 120$ grados respecto al eje x y finalmente la tercer cadena cinemática con $\phi = 240$ grados. Las tres cadenas conservan una posición equidistante de 120 grados sobre la base superior del robot delta.

3.1.2. Cinemática Directa

Para la cinemática directa se retoma la ecuación 3.1 y se reescribe como se muestra en la ecuación 3.11, esta expresión corresponde a una sola cadena cinemática.

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} r_1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} acos(\theta_1) \\ 0 \\ asin(\theta_1) \end{bmatrix} + \begin{bmatrix} bcos(\theta_1 + \theta_2)sin(\theta_3) \\ bcos(\theta_3) \\ bsin(\theta_1 + \theta_2)sin(\theta_3) \end{bmatrix} - \begin{bmatrix} r_2 \\ 0 \\ 0 \end{bmatrix} \tag{3.11}$$

De esta manera se ha analizado la cinemática del robot paralelo delta, las simulaciones realizadas en *Simulink* se presentan el la siguiente sección.

3.1.3. Cinemática en Simulink

Las expresiones matemáticas encontradas en el apartado anterior serán colocadas en Simulink para comprobar su funcionamiento, el primer bloque mostrado en la figura 3.5 es la comprobación que existe entre la cinemática directa y la cinemática inversa.

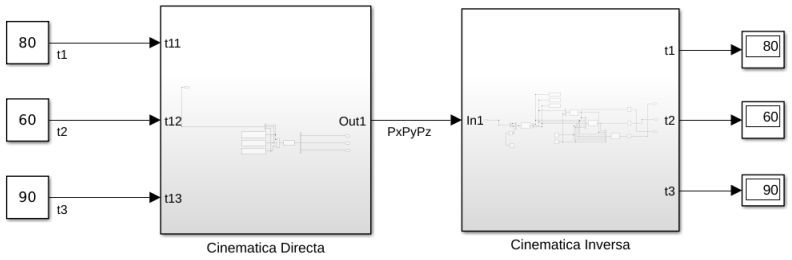


Figura 3.5: Comprobación de la cinemática

De la figura 3.5 se observa que los ángulos de entrada para la cinemática directa son los mismos calculados por la cinemática inversa. Esta comprobación está diseñada para solo una cadena cinemática.
La complejidad y el interés es mayor para la cinemática inversa así que en la figura 3.6 se muestra el interior de este bloque.

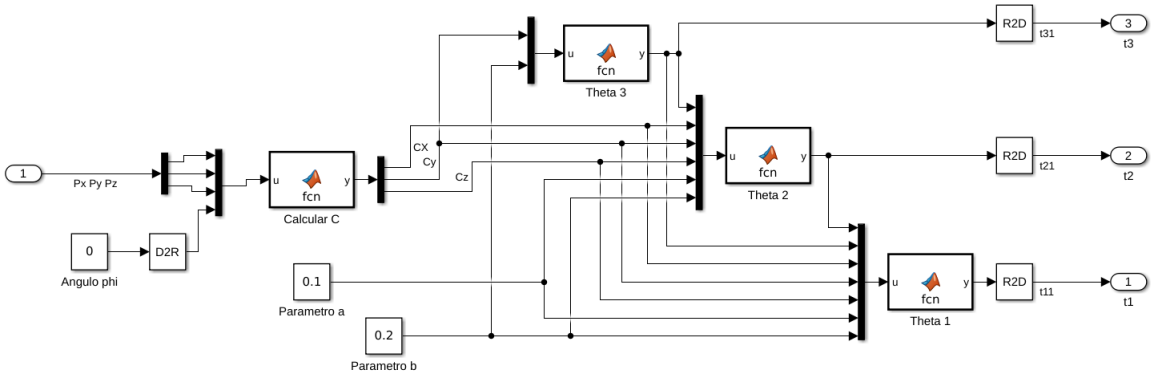


Figura 3.6: Cinemática Inversa

De la figura 3.6 se realizan las siguientes observaciones de izquierda a derecha:

- La entrada para la cinemática inversa siempre es una coordenada, en este caso, proporcionada por la cinemática directa, sin embargo, en la práctica esta coordenada es proporcionada por el sistema de visión por computadora.
- Esta comprobación se realizó para la cadena alineada con el eje x es por ello que $\phi = 0$. Los operaciones se realizan con los ángulos en radianes.
- Como se mencionó anteriormente, encontrar el valor de C es el primer paso.
- θ_3 y θ_2 no son ángulos actuados pero se requiere conocerlos para obtener el ángulo actuado, θ_1.
- Los parámetros tienen el valor mostrado en la tabla 3.1.

Parámetro	unidades(mm)
r1	800
r2	400
a	100
b	200

Tabla 3.1: Parámetros físicos

Ya que la cinemática inversa ha sido comprobada a través de la cinemática directa, entonces, es posible aplicar la cinemática inversa para cada una de las tres cadenas como se muestra en la figura 3.7, de esta manera se obtiene la cinemática inversa de todo el robot.

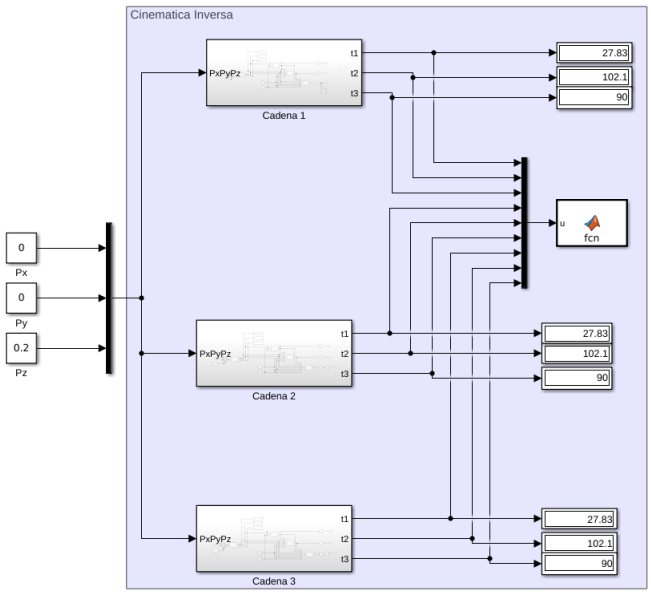


Figura 3.7: Cinemática inversa para el robot paralelo delta

De la figura 3.7 se observa que el bloque mostrado en la figura 2.13 se retoma de manera similar para cada una de las tres cadenas; algunas observaciones se enlistan a continuación:

- En la parte izquierda del diagrama se encuentra la coordenada deseada, la variable t1 correspondiente a cada cadena adopta una posición con el de que el efector final del robot alance ésta.

- La coordenada o punto deseado es $(0, 0, 0.1)$ esto significa que mientras en el eje x y y el efector final se coloca en 0, para el eje z se debe posicionar a 100 mm por debajo de la base superior.

- Los bloques para cada Cadena son casi idénticos, el ángulo *phi* es diferente para cada uno.

- Finalmente se agrega una función que, a través de líneas, dibuja el robot paralelo delta emulando la posición introducida, ver figura 3.8.

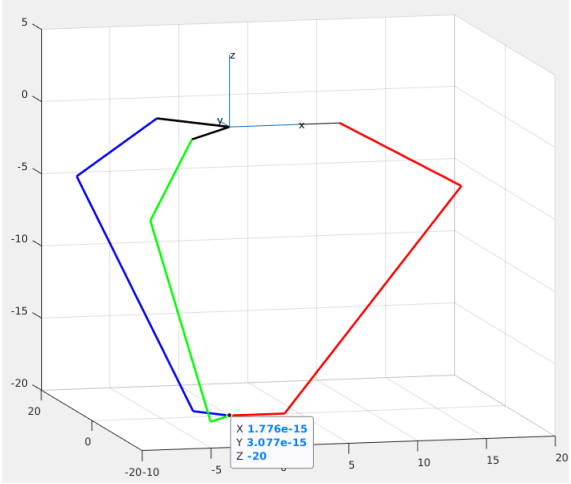


Figura 3.8: Animación 3D del robot Delta

De esta manera se comprobó la cinemática del robot paralelo delta, el código de esta simulación se encuentra en el apéndice A.

3.1.4. Diseño en SolidWorks

Antes de construir el prototipo se realizó un diseño con el cual se pudiera visualizar el movimiento del mismo. No se ocuparon mas herramientas que la visual del software *SolidWorks*, sin embargo ayudaron a comprender algunas de las restricciones físicas del robot así como también se pudo mejorar la comprensión de la cinemática expuesta en el apartado anterior.
El diseño se muestra en las figuras 3.9 y 3.10 en diferentes perspectivas:

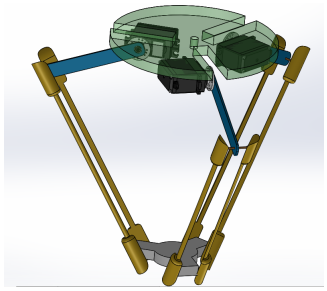
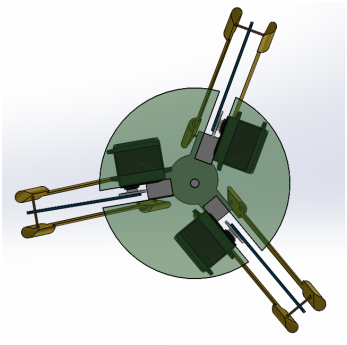
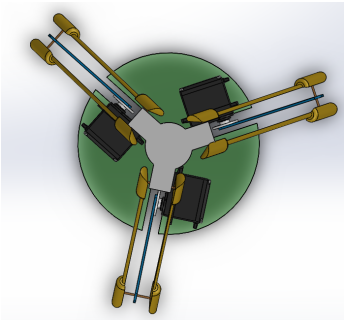


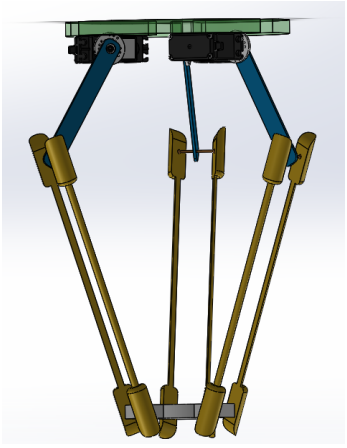
Figura 3.9: Diseño ilustrativo



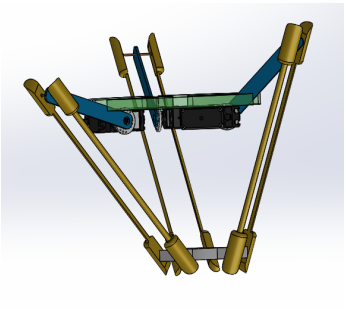
(a) Vista superior



(b) Vista inferior



(c) Cadenas extendidas



(d) Cadenas plegadas

Figura 3.10: Diseño ilustrativo en *SolidWorks*

Sin embargo, también se realizaron diseños de algunas piezas que se ocuparon en la construcción del mismo, a continuación se muestran en las figuras 3.11 y 3.12:

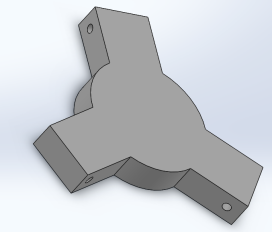
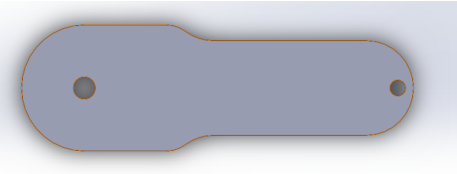
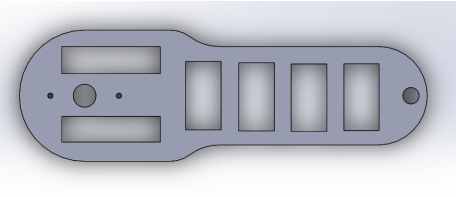


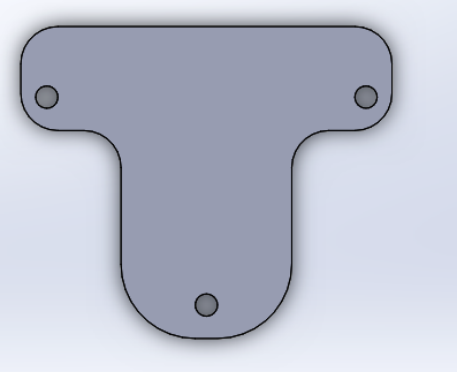
Figura 3.11: Base inferior



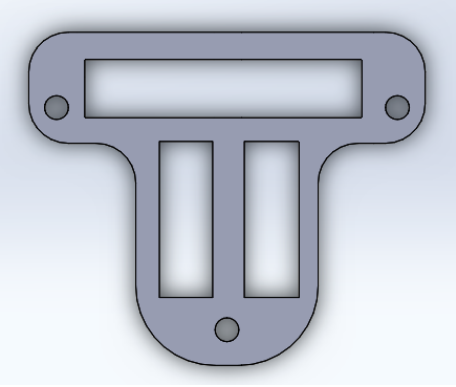
(a) Brazo superior versión 1.



(b) Brazo superior versión 2.



(c) Soporte para los motores versión 1.



(d) Soporte para los motores versión 2.

Figura 3.12: Piezas utilizadas para el ensamblaje del prototipo

3.1.5. Componentes físicos

En esta apartado se realiza la descripción de los elementos utilizados para la construcción del prototipo, durante el proceso de implementación física existieron eventos donde se realizaron cambios de componentes con fines de optimizar el sistema o por descompostura.
Las medidas y parámetros son coherentes con el apartado de simulaciones y se presentan en la tabla 3.2, posteriormente se enlistan los materiales utilizados y la descripción de cada uno de ellos.

Pieza	Parámetro	Símbolo	Unidades(mm)	Material
Base Superior	Radio base fija	r1	800	MDF
Base Inferior	Radio base móvil	r2	400	MDF
Brazo	Longuitud	a	100	MDF
Antebrazo	Longuitud	b	200	metal

Tabla 3.2: Parámetros físicos y materiales

Materiales:

- 1 Base para soportar el sistema, la altura debe ser igual o poco más de 300 mm, ver figura 3.13.

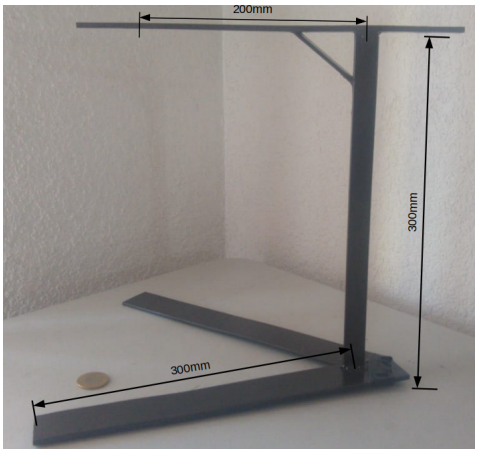
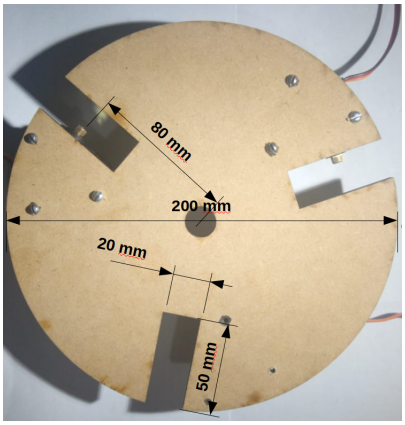


Figura 3.13: Soporte del prototipo

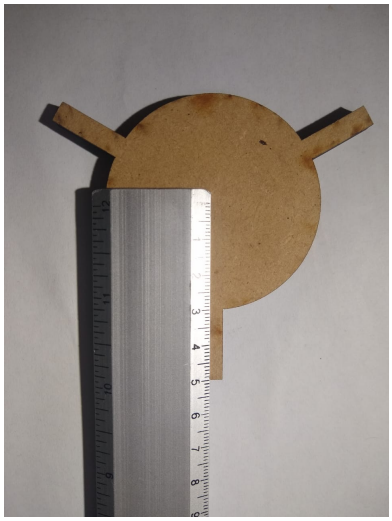
- Medio metro cuadrado de MDF de 5.5 mm de grosor. Se utilizó corte láser para obtener las piezas que se muestran en la figuras 3.14 y 3.15.



(a) Base fija (superior)

(b) Sujetador para el servomotor

(c) Base Móvil (inferior)



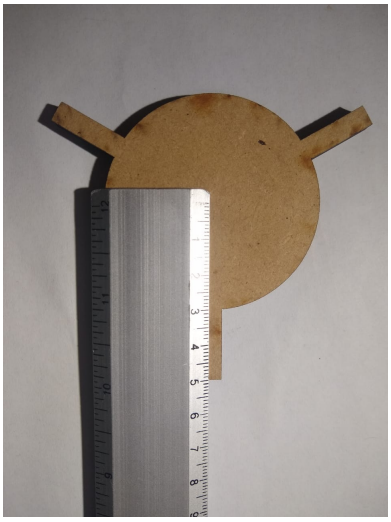
(d) Radio de la base móvil, 40 mm

Figura 3.14: Piezas modeladas en MDF

(a) 100 mm es la separación entre la centro de cada orificio

(b) Brazo, extremo conectado al antebrazo

(c) Brazo, extremo conectado actuador



(d) Radio de la base móvil, 40 mm

Figura 3.15: Piezas modeladas en MDF

- 3 Servomotores MG996R, éstos son una versión mejorada del MG995, ver figura 3.16.

Figura 3.16: Dimensiones aproximadas 47x19.7x42.9 mm

- 1 esparrago de 4 mm de diámetro como el mostrado en la figura 3.17.

Figura 3.17: Comercialmente sus medidas son 4 mm de diámetro

- 18 Tuercas para cuerda de 4 mm

- 12 Rotulas esféricas como las que se muestran en la figura 3.18.

(a) Largo de la pieza

(b) Ancho de la pieza

(c) Rosca de la pieza, comercialmente 4 mm

(d) Apertura de la rotula

Figura 3.18: Tipo de articulación esférica

- 3 Tornillos de 2.8 mm de diámetro como se muestra en la figura 3.19.

Figura 3.19: El diámetro de este elemento es de 2.8 mm y el largo de aproximadamente 32 mm

- 9 Tuercas para cuerda de 2.8 mm
- 9 Tornillos de 3 mm de diámetro y 1.5 pulgadas de largo como se muestra en la figura 3.20.

Figura 3.20: Tornillo de sujeción

- 9 Tuercas para cuerda de 3 mm

El procedimiento de la implementación se describe a continuación:

- Se recortaron 6 segmentos iguales del esparrago; a cada extremo del esparrago se coloca una articulación esférica; del centro de cada articulación existe una distancia de 200 mm. Ver figura 3.21.

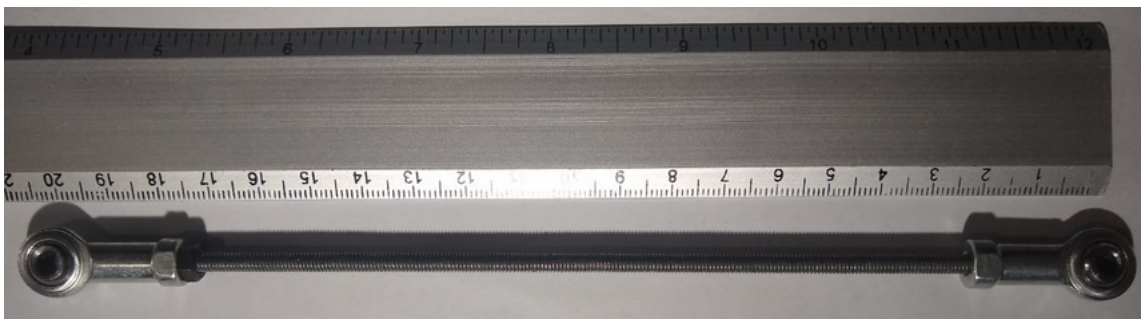
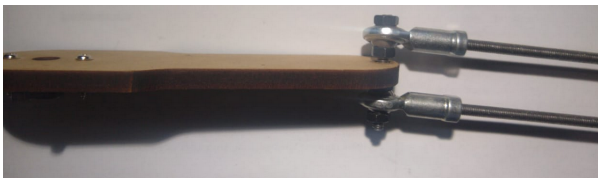


Figura 3.21: Antebrazo, se requieren dos de estas piezas para cada cadena cinemática

- Se recortan 3 segmentos iguales del esparrago; esta pieza une el antebrazo con el brazo, ver figura 3.22.

(a) Uniones



(b) El esparrago tiene movilidad dentro el orificio del Brazo

Figura 3.22: Ensable Brazo-Antebrazo

- Con los tornillos de 1.2 in se ensamblan los antebrazos a la mase móvil, se ocupan tres tuercas por cada ensamble, ver figura 3.23.

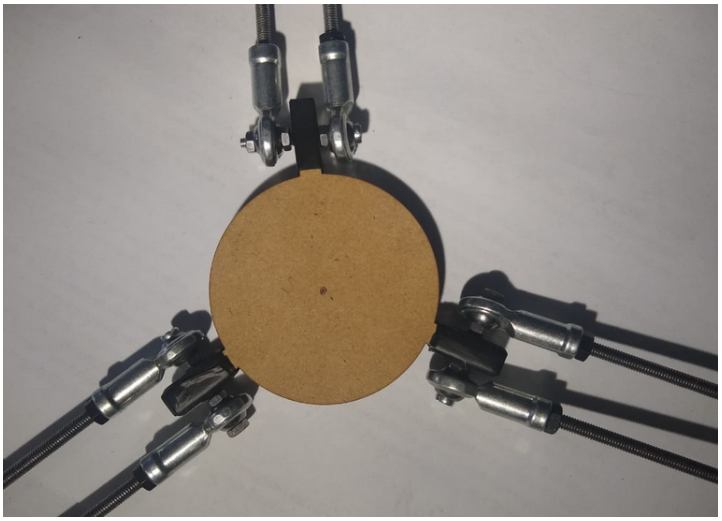


Figura 3.23: La distancia entre cada cadena cinemática respeta los 120 grados de equivalencia espacial

- La sujeción de los motores con la base superior se realiza por la parte inferior de la misma, se utilizan tres tornillos para sujetar cada motor. Se coloca el eje del motor a una distancia de 80 mm del centro de la base, ver figura 3.24.

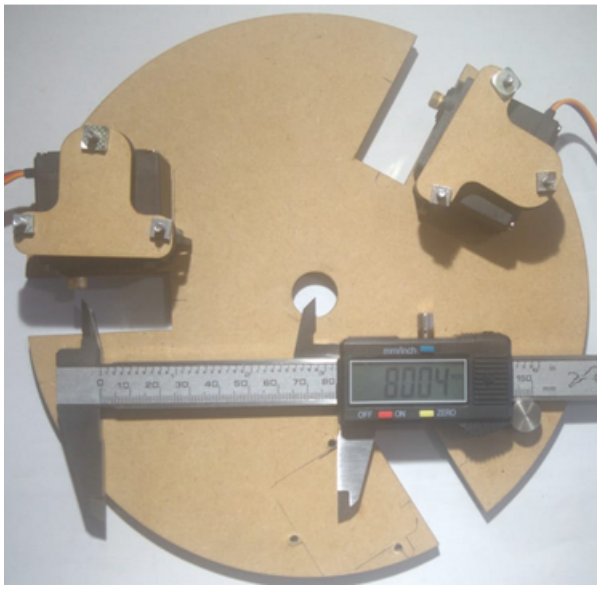


Figura 3.24: Servomotores ligados a la base fija

- Finalmente la base superior fija a 200 mm de la columna de soporte como se muestra en la figura 3.25. Posterior son colocados los brazos al eje del motor, de esta manera el prototipo ha sido ensamblado ver figura 3.26. La última parte es agregar la tarjeta de control y la cámara web.

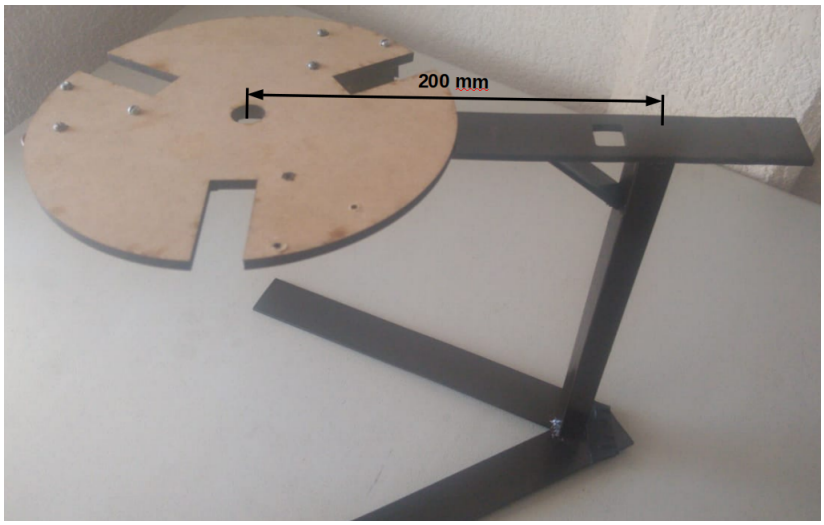


Figura 3.25: Colocación de la base fija

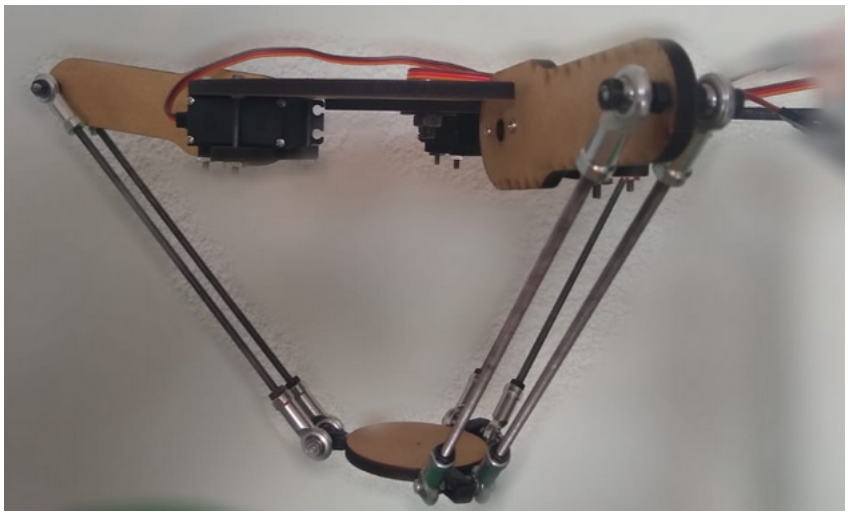


Figura 3.26: Prototipo del robot Delta

El control del prototipo está a cargo de Matlab, sin embargo, este robot será controlado por teleoperación, Matlab enviará a través de ROS las señales necesarias y la tarjeta NodeMCU ESP8266, conectada también a ROS, recibirá éstas (posición angular), después de una sencilla manipulación las envía como *PWM* a cada uno de los servomotores, la figura 3.27 ejemplifica ésto.

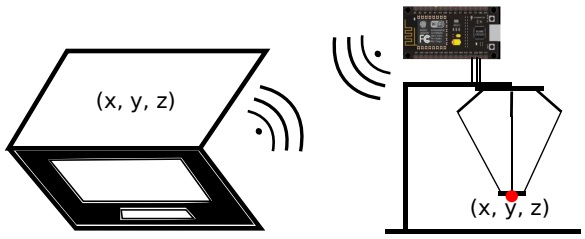


Figura 3.27: Sistema de teleoperación

NodeMCU ESP8266 es una pequeña tarjeta de bajo consumo energético y recursos limitados que es posible programar a través del IDE de *Arduino*, sin embargo, a diferencia de las placas de arduino, ésta contiene integrado un módulo *WiFi*, además su bajo costo la hace idónea para este trabajo; también es posible conectar con ROS con el software adecuado.

Rosserial es un protocolo de comunicación el cuál permite que tarjetas como Arduino o, en este caso, NodeMCU ESP8266 puedan acceder a ROS sin un sistema

operativo propio. Para acceder a esta herramienta se compilaron diversos paquetes en la computadora donde se encuentra Matlab y el nodo maestro, al terminar esta configuración es posible compilar y ejecutar nodos que comuniquen la tarjeta con ROS, el diagrama de la figura 3.28 muestra de manera gráfica la configuración de la red hasta este punto.

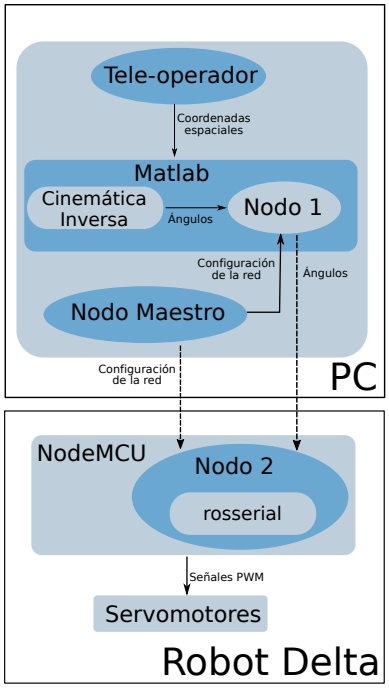


Figura 3.28: Configuración sin el sistema de visión

Sin embargo, la tarjeta también requiere de un programa que le diga como comunicarse con el nodo configurado anteriormente, la figura 3.29 muestra el código implementado.

```
 //   BILBIOTECAS Y PAQUETES**************
#include <SPI.h>
#include <Ethernet.h>
#include <Servo.h>
#define ROSSERIAL_ARDUINO_TCP
#include <ros.h>
#include <std_msgs/Float32.h>
ros::NodeHandle  nh;
//    ************************************
//  DIRECCIONES DE LA TARJETA
byte mac[] = { 0xA4, 0xCF, 0x12, 0xBF, 0x39, 0xF4 };
IPAddress ip(192, 168, 1, 72);

// DIRECCIONES DE LA COMPUTADORA DONDE SE
// EJECUTA EL NODO MAESTRO
IPAddress server(192, 168, 1, 75);
uint16_t serverPort = 11411;
// SE CREAN TRES OBEJTOS UNO PARA CADA SERVOMOTOR
Servo servo1, servo2, servo3;
// SE DEFINE EL TIPO DE MENSAJE Y LA SALIDA DEL
// PWM PARA CADA SERVOMOTOR
// 1
void messageCb1( const std_msgs::Float32& cmd_msg1){
  servo1.attach(2);
  servo1.write(cmd_msg1.data);
  }
  // 2
void messageCb2( const std_msgs::Float32& cmd_msg2){
  servo2.attach(4);
  servo2.write(cmd_msg2.data);

  }
```

(a) Primera parte

```
  // 3
  void messageCb3( const std_msgs::Float32& cmd_msg3){
  servo3.attach(15);
  servo3.write(cmd_msg3.data);

  }
// SE DEFINE QUE SON DATOS DE ENTRADA A TRAVÉS DE LOS
// TÓPICOS "servo1", "servo2" y "servo3"
ros::Subscriber<std_msgs::Float32> sub1("servo1", messageCb1 );
ros::Subscriber<std_msgs::Float32> sub2("servo2", messageCb2 );
ros::Subscriber<std_msgs::Float32> sub3("servo3", messageCb3 );
void setup(){
// SE CONCETA A LA RED Y LLAMA A LAS DIFERNTES FUNCIONES ANTES
//DESCRITAS
  Ethernet.begin(mac, ip);

  delay(1000);
  nh.getHardware()->setConnection(server, serverPort);
  nh.initNode();
  nh.subscribe(sub1);
  nh.subscribe(sub2);
  nh.subscribe(sub3);
}

void loop(){
  nh.spinOnce();
  delay(1);
```

(b) Segunda parte

Figura 3.29: Código cargado en NodeMCU ESP8266

De la figura 3.29 se realizan algunas observaciones:

- En la parte superior de la figura 3.29 (a) se observa que se agregan la cabeceras necesarias.
- Posteriormente se configuran las dirección de la tarjeta NodeMCU introduciendo la dirección IP y MAC, estos parámetros también se requieren especificar pero del nodo maestro.
- Con las funciones *messageCb1, messageCb2 y messageCb3* reciben el mensaje enviado por Matlab (el ángulo θ, respectivamente) y lo envían a determinado pin de salida.
- Finalmente se configura que el nodo es de tipo *Subscritpor*, es así como se determina que es un nodo que recibe datos de la red.

3.1.6. Espacio de trabajo

El diseño de este prototipo no comprende un espacio de trabajo específico, se sabe que este tipo de estructura tiene como desventaja tener un área reducida de

trabajo para el efector final, a pesar de que teóricamente es posible calcular este espacio, en la práctica el resultado es diferente.
Ya que para este trabajo de tesis optimizar el área de trabajo del prototipo no es una prioridad ni se encuentra en los objetivos deseados, los cálculos para esta sección se realizaron de manera experimental y los resultados fueron los siguientes, ver figura 3.30:

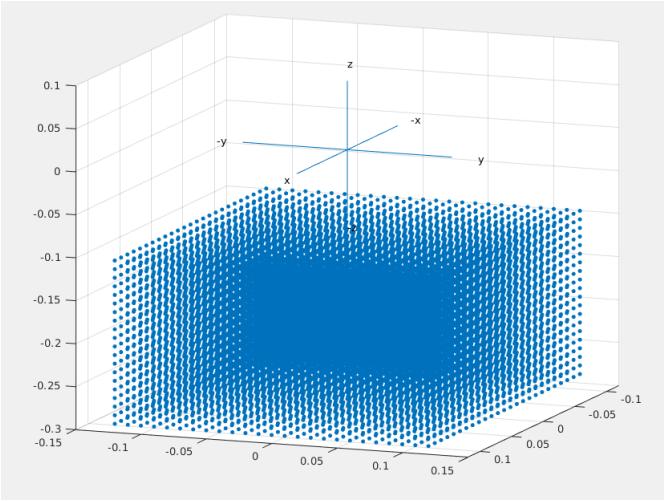


Figura 3.30: Primer filtro

A través del prototipo se delimitó el espacio de trabajo basado en el alcance del efector final en los tres ejes coordenados, el resultado es una nube cúbica de puntos como se muestra en la figura 3.30 que se encuentra solo en el eje z negativo. Las medidas se especifican en la tabla 3.3. Es importante mencionar que los puntos en las esquinas de esta nube, no son alcanzados por el efector final, se requiere de otro algoritmo se funcione como un segundo filtro y así, encontrar el espacio de trabajo de una manera mas específica.

Eje	x	y	z
Negativo	-120 mm	-120 mm	de -100 mm a -290 mm
Positivo	120 mm	120 mm	NA

Tabla 3.3: Alcance máximo físico

Capítulo 4

Sistema de visión

El sistema de visión tiene como propósito proporcionar las coordenada espaciales del objeto deseado al manipulador paralelo delta.
La secuencia de pasos requeridos por el sistema de visión para cumplir su meta se muestra en el diagrama de la figura 4.1.

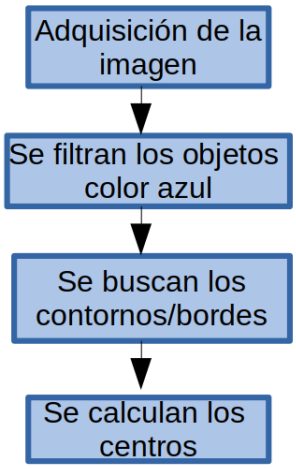


Figura 4.1: Función

El esquema mostrado en la figura 4.1 es, de manera muy general pero concisa, las etapas que se llevaron a cabo para la implementación en el lenguaje de programación Python, el código se muestra en las figuras 4.2, 4.4 y 4.5 donde se describe su función.

```
###### CAPTURA DE UNA FOTO  #############
        cap = cv2.VideoCapture(0)
        leido, frame = cap.read()
        if leido == True:
            sleep(3)
            cv2.imwrite("foto.png", frame)
            print("Foto tomada correctamente")
        else:
            print("Error al acceder a la camara")
        cap.release()
################################################
```

Figura 4.2: Adquisición de la imagen

De la figura 4.2, se inicializa la cámara y la lectura de ésta se asigna a *frame*, de tomar la foto de forma exitosa, la foto es guardada como *foto* en formato PNG. Es importante mencionar que se coloca un retardo de 3 segundos para dar tiempo que la cámara se inicie correctamente.

```
######## CALCULO DE BORDES/CONTORNOS ###########
### Y DEFINICION DEL UMBRAL DE COLOR (AZUL) ####
        fot = cv2.imread("foto.png")
        azulBajo = np.array([100,100,20],np.uint8)
        azulAlto = np.array([125,255,255],np.uint8)
###
        fot_HSV = cv2.cvtColor(fot,cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(fot_HSV,azulBajo,azulAlto)
        contornos, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
################################################
```

Figura 4.3: Contornos y filtro de color

Analizando el código de la figura 4.3, la imagen tomada en el paso anterior es colocada en *fot*, posterior es llevada a un espacio de color mas parecido a la percepción de ojo humano. Se define las características que filtran el color azul a través de *azulBajo* y *azulAlto*.
A través de *inRange* se filtran solo los objetos color azul, posterior *findContours* determina los bordes de dichos objetos. De esta manera se han obtenido los contornos de los objetos con color azul.

```
######## LOCALIZACION DE LOS CENTROS  ###########
        for c in contornos:
            area = cv2.contourArea(c)
            if area > 3000:
                M = cv2.moments(c)
                if (M["m00"]==0): M["m00"]=1
                x = int(M["m10"]/M["m00"])
                y = int(M['m01']/M['m00'])
                cv2.circle(fot, (x,y), 7, (0,255,0), -1)
                font = cv2.FONT_HERSHEY_SIMPLEX
                cv2.putText(fot, '{},{}'.format(x,y),(x+10,y), font, 0.75,(0,255,0)
                ,1,cv2.LINE_AA)
                nuevoContorno = cv2.convexHull(c)
                f = cv2.drawContours(fot, [nuevoContorno], 0, (255,0,0), 3)
```

Figura 4.4: Contornos y filtro de color

De la figura 4.4, para cada uno de los contornos encontrados se calcula su momento siempre y cuando el contorno tenga un tamaño significativo, para saber esto se calcula su área a través de *contourArea* esta cifra debe tener un tamaño mayor a 3000 píxeles. Finalmente se toma la imagen original y sobre ella se pintan los contornos y sus respectivos centros.

Es en esta última parte que ya se han obtenido las coordenadas x y y que como se mencionó, es el propósito de este sistema.

El dispositivo que se utiliza para la ejecución de este programa es la tarjeta de desarrollo *Jetson Nano Developer Kit*, en la figura 4.5 se muestra ésta y se enlistan algunas de sus características.

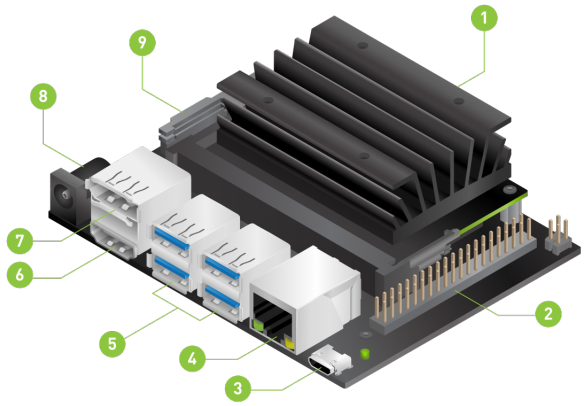


Figura 4.5: *Jetson Nano developer Kit de NVIDIA*

1. Ranura para tarjeta microSD para almacenamiento principal
2. Puerto de expansión de 40 pines
3. Puerto micro-USB para entrada de alimentación de 5V o para datos
4. Puerto Gigabit Ethernet
5. Puertos USB 3.0 (x4)
6. Puerto de salida HDMI
7. Conector DisplayPort
8. Conector DC Barrel para entrada de alimentación de 5V
9. Conector de cámara MIPI CSI

Esta tarjeta es una computadora que está dirigida y optimizada para el desarrollo de proyectos relacionados con inteligencia artificial, contiene procesadores GPU de 128 núcleos; memoria LPDDR4 de 4 GB; entre otras características lo cuál mejora su rendimiento, sin embargo, antes de poder hacer uso de ésta es necesario realizar su configuración.
Mostrar la configuración y puesta en marcha de esta tarjeta no es un objetivo de esta tesis, sin embargo, en el apéndice se agregan algunas notas.

Capítulo 5

Conexión de los sistemas

En este capítulo se muestra de manera más específica la configuración de la red, se determinan los nodos, los tópicos y, a través de flechas, la manera en que fluyen los datos. La conexión de los sistemas anteriores se realiza a través de *ROS*, gracias a esto se obtuvieron algunas ventajas que facilitaron la implementación gracias a que en ROS permite:

- Asegurar que todos los nodos se encuentran conectados
- Saber qué mensajes se están enviado
- Puede enlistar todas las características de cualquier nodo conectado a la red
- Puede enlistar todas las características de cualquier tópico activo a la red
- Es posible observar las características de un mensaje
- Ros posee una herramienta gráfica que muestra toda la red
- Los nodos en este trabajo de tesis se implementaron en Python, pero es posible implementarlo en otro lenguaje

El diagrama de conexión es como se muestra en la figura 5.1.

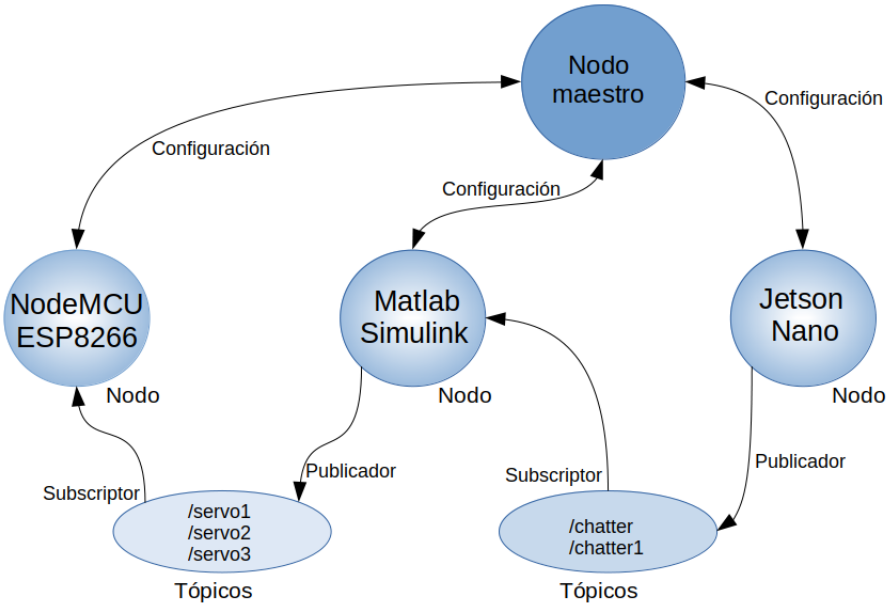


Figura 5.1: ROS

Como se observa en la figura 5.1, ROS trabaja con un sistema de *tópicos* y *nodos*, esta forma tiene la cualidad de abstraer las características físicas del dispositivo, por lo tanto únicamente con el software necesario y la paquetería de ROS, es posible conectar cualquier aparato a una red.

Un nodo es un hardware que tiene la capacidad de poder ser configurado a través de software para ejecutar las bibliotecas y paquetes de ROS, es posible conectarse a través del puerto serie pero en este trabajo se utiliza *Ethernet* con la ayuda de un módem convencional.

De la figura 5.1 se enumeran algunos puntos importantes en la secuencia en la que se inicia todo el sistema.

1. *ROS MASTER* o nodo maestro es el primer nodo que se inicia, se encuentra en la misma computadora donde está Matlab y los paquetes rosserial compilados. La secuencia de comandos utilizados se muestran en la figura 5.2.

```
roscore http://192.168.1.75:11311/
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
enrique@enrique-Inspiron-5548:~$ export ROS_MASTER_URI=http://192.168.1.75:11311
enrique@enrique-Inspiron-5548:~$ export ROS_IP="192.168.1.75"
enrique@enrique-Inspiron-5548:~$ roscore
```

Figura 5.2: Se *levanta* la red ROS

2. En la consola de Matlab se teclea el comando *rosinit(dirección IP del nodo maestro)*, de esta manera Matlab inicia automáticamente un nodo y al mismo tiempo se conecta a la de red ROS.

3. En la misma computadora se inicializa el nodo correspondiente a la tarjeta NodeMCU ESP8266, el comando requerido se muestra en a figura 5.3.

Figura 5.3: Se inicializa el tercer nodo

4. Se instaló un sistema operativo basado en linux en la tarjeta Jetson Nano Developer Kit, después de instalar ROS, es posible crear un nodo que sea capaz de tomar los valores x y y, calculados por el sistema de visión, y mandarlos a la red de manera inalámbrica, estos valores son la entrada del bloque de cinemática inversa programado en Simulink.
   Los comandos para inicializar este nodo se introducen en la terminal de la tarjeta Jetson Nano como se muestran en la figura 5.4.

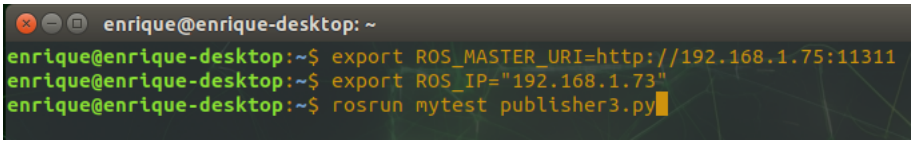


Figura 5.4: Se inicia un cuarto nodo

El código que se implementó en esta tarjeta se mostró en la sección de visión por computadora, es un nodo que *publica* las coordenadas x y y. Los resultados se muestran en el siguiente capítulo.

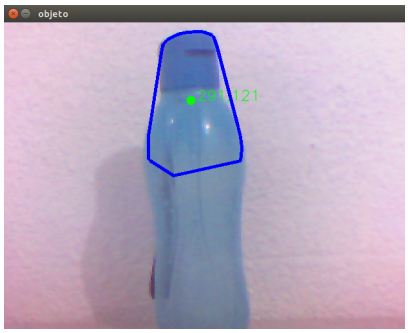
Capítulo 6

Resultados experimentales

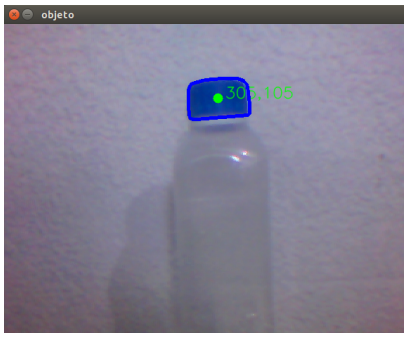
En este capítulo se muestran los resultados obtenidos del desarrollo, éste consistió en lo descrito en los capítulos 3, 4 y 5, cada uno de ellos se enfocó en una parte especifica de este trabajo de tesis, siguiendo el mismo método, aquí se presentan los resultados obtenidos de cada uno de éstos. Finalmente se presenta, de manera gráfica con la ayuda de *Node Graph*, todos los nodos conectados y los tópicos activos.
Implementar cada unos de los sistemas y hacer la conexión entre ellos consistió en varias etapas y cada una de ellas en diferentes pruebas, sin embargo, se presentan los resultados finales con los cuales se establece que se cumplieron los objetivos planteados.

6.1. Visión por computadora

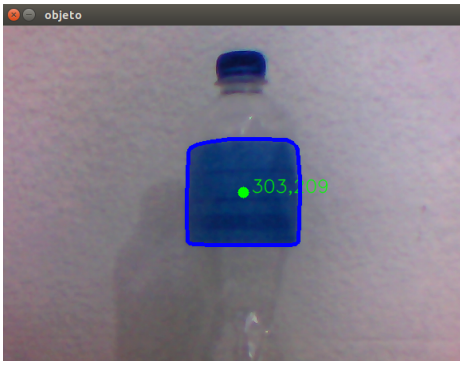
El objetivo que tiene el sistema de visión para este trabajo es calcular las coordenadas de un objeto, éste tiene la característica de ser color azul. De acuerdo a la figura 6.1 se observa que se cumple con lo deseado.



(a) Objeto 1



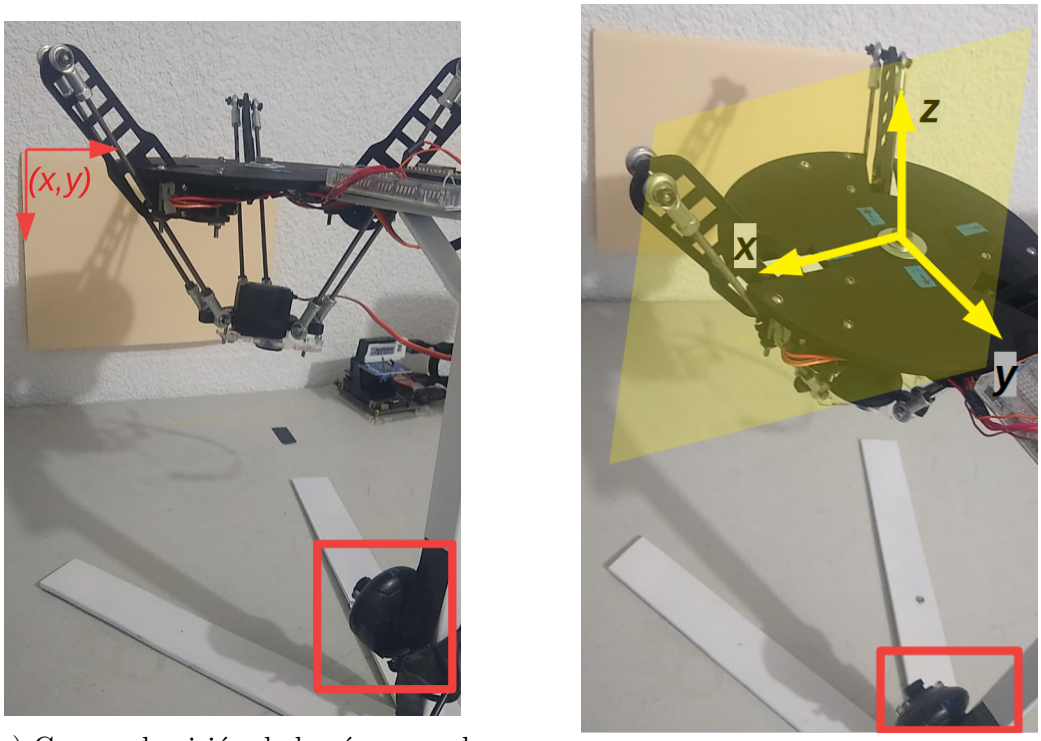
(b) Objeto 2



(c) Objeto 3

Figura 6.1: Coordenadas x y y del contorno en color azul

Para que el sistema funcione de manera adecuada se requiere de coordenadas espaciales, sin embargo, debido a la situación COVID-19 no fue posible acceder a una cámara que proporcionara éstas, así que se utilizó una cámara web ordinaria que trajo como consecuencia utilizar solo un plano del área de trabajo del robot, se muestra en la figura 6.2.



(a) Campo de visión de la cámara web

(b) Plano *(x,z) del robot delta*

Figura 6.2: Acondicionamiento de los sistemas coordenados

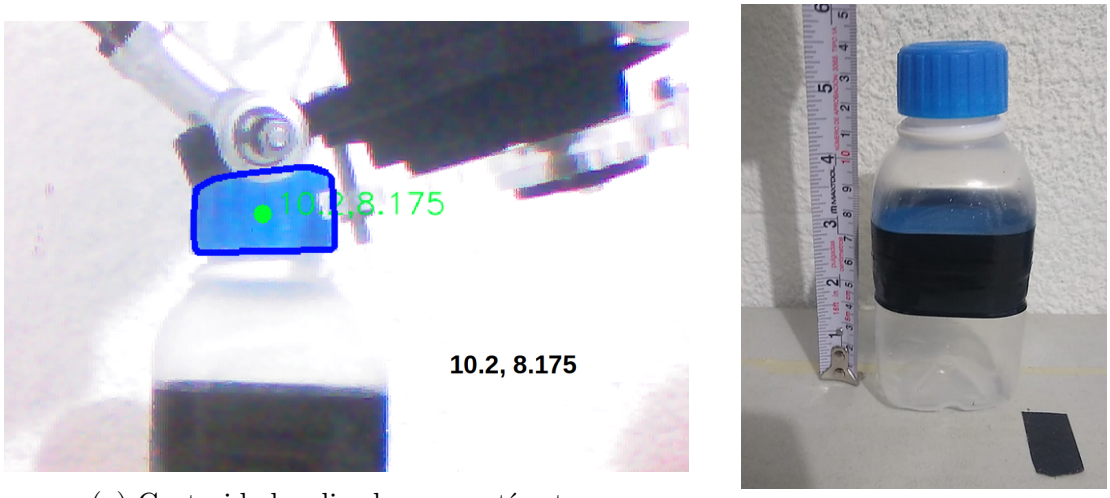
De la figura 6.2(a) se observa que la cámara, colocada al inferior del robot, tiene un campo de visión como se ilustra con la pantalla amarilla, que es un plano x,y perpendicular al eje y del robot, de esta manera se buscó una relación de tipo paralela entre el campo de visón de la cámara y alguno de los planos coordenados del robot. El plano x,z del robot cumple con los requisitos, ver figura 6.2(b), así que mientras el eje coordenado y del robot se mantiene inmóvil en 0, las coordenadas de ambos sistemas quedan de las siguiente manera: $x_{cámara} \Rightarrow x_{robot}$ y $y_{cámara} \Rightarrow z_{robot}$
El acoplamiento de estos sistemas consistió en lo siguiente:

1. Conocer la resolución de la cámara que es de (480,640), significa que en píxeles tienen 480 filas y 640 columnas en la imagen.

2. Se midió la amplitud en centímetros del campo de visión en los ejes x y y, esta medición proporcionó los valores de 24 cm en el x y 18 cm en el eje y, con estos

datos se encontró un factor que a diferencia de lo mostrado en la figura 6.1, muestra en centímetros la posición del objeto de interés, ver figura 6.3, de esta manera Matlab recibe las coordenadas en centímetros como se comprueba en la figura 6.4. El factor se obtuvo con la expresión de la ecuación 6.1.

$$\begin{aligned} Para \quad x &= 24cm/640píxeles = 0.0375 \\ Para \quad y &= 18cm/480píxeles = 0.0375 \end{aligned} \tag{6.1}$$



(a) Centroide localizados en centímetros

(b) Objeto de prueba

Figura 6.3: Conversión píxeles a centímetros

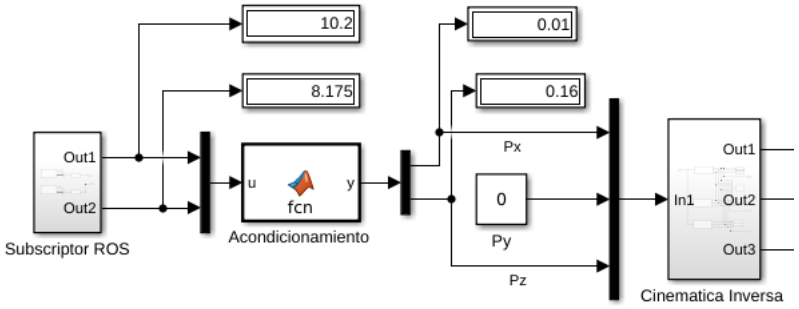


Figura 6.4: Simulink final

Como se observa en la figura 6.4 las salidas del bloque *Subscriptor ROS* son las

coordenadas *(x,y)* de la figura 6.3, en Matlab se terminan de acondicionar los datos para que sean interpretados por la cinemática inversa de manera adecuada, cabe mencionar que $\hat{x} \quad y \quad \hat{z}$ son las salidas del bloque *Acondicionamiento* de la figura 6.4.
Algunas de las observaciones encontradas en estas condiciones del prototipo son las siguientes:

- La cantidad de luz influye de manera determinante en el momento de calcular el centroide del objeto de interés, en repetidas ocasiones el exceso de luz impedía realizar la tarea.

- Es importante mencionar que debido a esta condición se dificulta que el manipulador tome el objeto de prueba de manera óptima.

- A pesar de que las coordenadas son enviadas en centímetros, Matlab no las reconoce de esta misma manera. Se realizó una acoplamiento entre el campo de visión en centímetros de la cámara y el *plano* de trabajo del robot, las expresiones para cada eje se muestran en la ecuación 6.2:

$$\begin{aligned} x_{cámara} + 12 &= x_{Matlab}/100 \\ y_{cámara} + 10 &= z_{Matlab}/100 \end{aligned} \tag{6.2}$$

  Se realiza la división entre 100 ya que para Matlab, el valor 1 representa un metro.

6.2. Manipulador paralelo delta

De acuerdo a lo descrito en el desarrollo, la plataforma móvil del robot alcanza el punto deseado, esto se ha determinado de manera visual y con apoyo de herramientas métricas, sin embargo, no se sabe con qué exactitud lo hace; se requiere de un sistema como *Optitrack* con el cual se pueda determinar tanto exactitud como precisión.
Respecto a la cantidad de carga que puede levantar: Cada cadena cinemática pesa 83 g, la base inferior 15 g y el efector final 73 g lo que suma 405 g de estructura colocada directo a los ejes de los tres motores; la carga máximo que soporta el sistema para trabajar de manera óptima sin ser forzado es de 1169 g contemplado la estructura

mencionada, por lo tanto, la carga máxima del objeto a levantar es de 764 g. El prototipo completo se muestra en la figura 6.5.

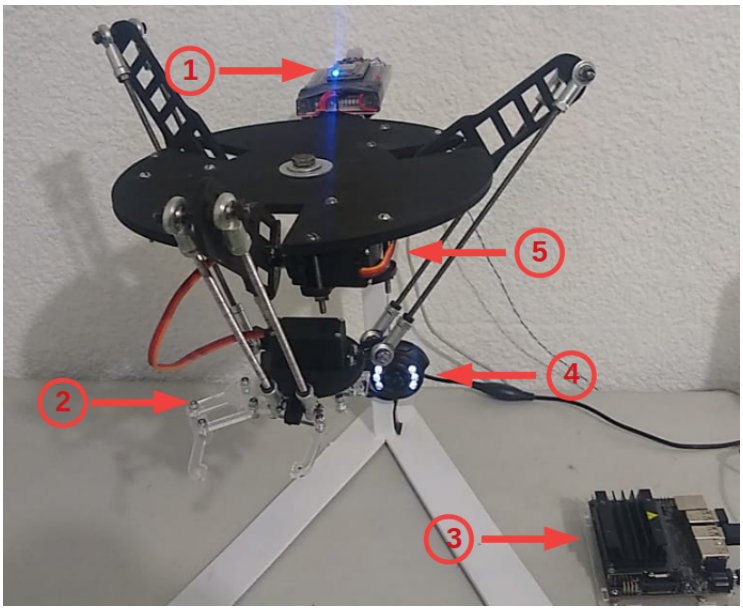


Figura 6.5: Prototipo implementado

A continuación se describen los puntos señalados en la figura 6.5 y algunas de las pruebas realizadas introduciendo las coordenadas desde Simulink se muestran en la figura 6.6.

1. En la parte superior se encuentra montada la tarjeta NodeMCU ESP8266.

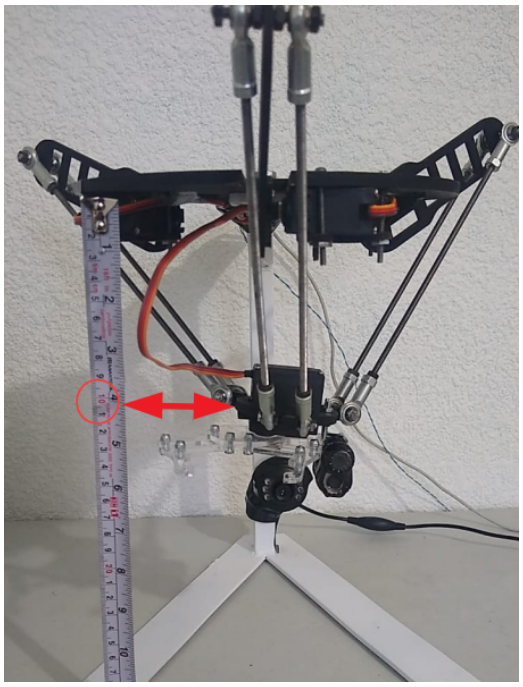
2. El efector final es una pinza accionada por un servomotor MG995.

3. Jetson Nano Developer Kit.

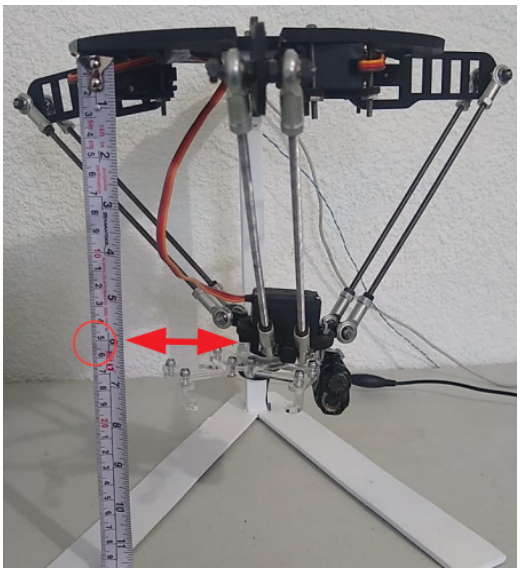
4. Cámara web

5. Servomotores MG996R uno para cada cadena cinemática.

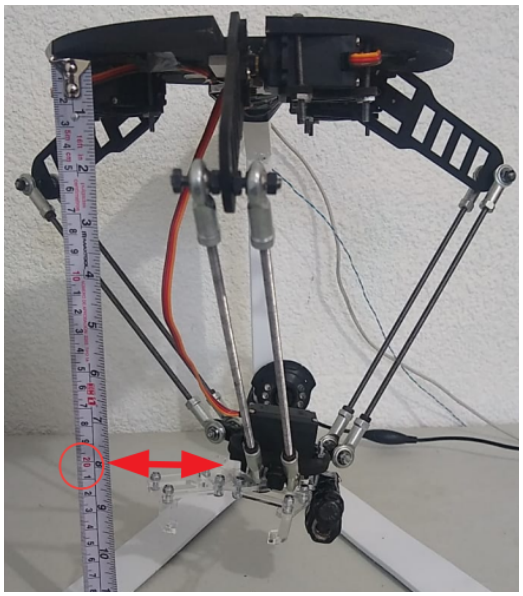
Se realizaron pruebas de posicionamiento del efector final, como se muestra en la figura 6.6, el sistema alcanza el punto deseado.



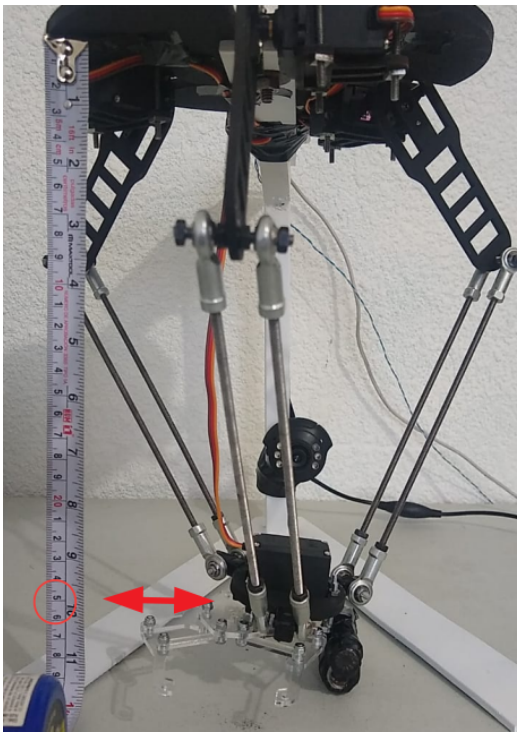
(a) $z = -10cm$



(b) $z = -15cm$



(c) $z = -20cm$



(d) $z = -25cm$

Figura 6.6: El efector final se posiciona en el punto deseado

La red ROS para las pruebas de la figura 6.6 se muestra en la figura 6.7.

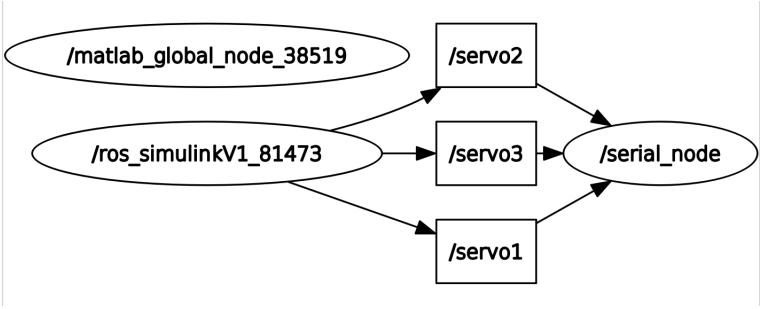


Figura 6.7: Red ROS

El esquema presentado en la figura 6.7 es una herramienta que pertenece a ROS y permite observar la red de manera esquemática, se observa que existen dos nodos conectados a través de tres tópicos, cada tópico pertenece a un servomotor y tal como lo muestran las flechas, los ángulos calculados desde el nodo de Simulink(*ros _ simulinkv1 _ 81473*) fluyen hacia el nodo de la tarjeta NodeMCU (*serial _ node*).

6.3. Conexión de sistemas

La conexión de todos los sistemas se observa a través de un esquema mostrado en la figura 6.8.

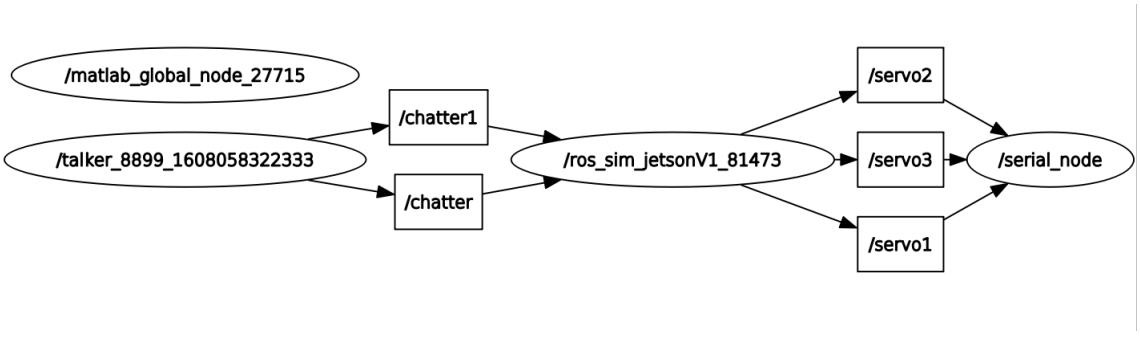


Figura 6.8: Red ROS completa

Como se planteó en el esquema de la figura 5.1 del capítulo 5, la figura 6.8 muestra el diagrama de la conexión final de ROS, se realizan las siguientes observaciones:

1. El nodo global de Matlab se muestra independiente.

2. El primer nodo pertenece a la tarjeta Jetson nano, ésta publica a través de los tópicos */chatter y /chatter1* las coordenada x y y respectivamente.

3. El nodo de Simulink recibe las coordenadas subscribiéndose a los tópicos */chatter y /chatter1*.

4. Después de calcular la cinemática inversa, el nodo de Simulink publica los ángulos para cada motor a través de los tópicos */servo1, /servo2 y /servo3*.

5. La terjeta NodeMCU recibe los ángulos subscribiéndose a los tópicos */servo1, /servo2 y /servo3*, posteriormente cada ángulo es enviado a los servomotores a través de señales PWM.

Finalmente en la figura 6.9 se muestran los mismo tópicos y nodos de la figura 6.8 pero en la terminal de *Ubuntu*.

```
enrique@enrique-Inspiron-5548:~$ rosnode list
/matlab_global_node_27715
/rosout
/rqt_gui_py_node_7999
/serial_node
/talker_8899_1608058322333
enrique@enrique-Inspiron-5548:~$ rostopic list
/chatter
/chatter1
/diagnostics
/rosout
/rosout_agg
/servo1
/servo2
/servo3
/statistics
```

Figura 6.9: Se corroboran los nodos y tópicos de la figura 6.8

Capítulo 7

Conclusiones

A continuación se presentan algunas conclusiones de este trabajo de manera puntual.

- Los resultados obtenidos del sistema de visión se pueden ver modificados debido a la cantidad de luz, de tener mayor recursos se podría conseguir una cámara con mejores características.
- Se comprueba en la práctica que este diseño de robot tiene como desventaja un reducido espacio de trabajo, el análisis de éste puede ser más minucioso y detallado para una posible optimización.
- Los materiales utilizados en la implementación pueden ser fácilmente modificados pero también pueden ser mejorados.
- En un inicio la instalación y configuración de ROS representó un desafío, fue complicado y llevó mucho tiempo para lograr que funcionara, sin embargo, una vez configurado y entendido su funcionamiento básico facilitó la conexión de los sistemas aquí descritos y aún más, es posible agregar más dispositivos con relativa facilidad.
- El costo del sistema puede incrementarse o mantenerse relativamente bajo en función de dos componentes principalmente: los servomotores y la tarjeta de de visión (Jetson Nano), ambos pueden ser reemplazados poniendo en juego la calidad de todo el sistema.

- Del punto anterior, la tarjeta encargada del sistema de visión tiene la capacidad de ejecutar diferentes algoritmos de inteligencia artificial (IA).Es interesante probar otras técnicas de procesamiento de imagen conectado al manipulador delta.

- El resultado de este trabajo de tesis se convirtió en un prototipo de pruebas que si bien, se requiere de más trabajo para ser un asistente personal, es un plataforma que da pie a nuevos proyectos.

7.1. Trabajo a futuro

Este prototipo se encuentra lejos de colocarse en una cadena de producción, sin embargo, es una base sobre la cual se pueden trabajar varias posibilidades y descubrir nuevas aplicaciones. El posible trabajo futuro para este proyecto puede ser:

- Implementar algoritmos más complejos para el sistema de visión por computadora, de esta manera es posible explotar las capacidades de la tarjeta Jetson Nano.

- Colocar una cámara de mejor calidad.

- Optimizar los algoritmos utilizados en la cinemática inversa.

- Mudar el procesamiento de Matlab a un lenguaje de fuente abierta como Python.

- Colocar este prototipo sobre una base móvil.

- Optimizar el funcionamiento del efector final.

- Reducir el hardware colocando todo el procesamiento en una sola tarjeta posiblemente provoque que el desempeño se vea afectado de manera negativa, sin embargo, si la aplicación no es exigente, se convierte en una posibilidad con beneficios económicos significativos.

Bibliografía

[1] K. Yamazaki et al., *"Home-assistant robot for an aging society"* Proc. IEEE, vol. 100, no. 8, pp. 2429–2441, 2012.

[2] United Nation, *World Population Prospects the 2010 Revision*, Department of Economic and Social Affairs, Population Division, vol. 1, 2011.

[3] Bistrain Coronado C., *Cambios recientes en la esperanza de vida en México, análisis por medio de su descomposición*, Realidad, datos y espacio. Revista internacional de estadística y geografía, Vol. 6, Núm. 3, pp. 78–97, 2015

[4] INEGI, *Cuéntame, Población. Discapacidad en México*, INEGI, 2010. [En línea]. Disponible: http://cuentame.inegi.org.mx/poblacion/discapacidad.aspx?tema=P. [Último acceso: 25 Febrero 2019].

[5] L. O. S. T. D. E. L. Futuro, *"Los trabajos del futuro"*, 2017.

[6] IFR, *"Welcome to the IFR Press Conference 18 th September 2019 Shanghai"*,IFR Press Conference, September 2019. [En línea] Disponible: https://www.coursehero.com/file/50351895/IFR-World-Robotics-Presentation-18-Sept-2019pdf/

[7] A. L. Benabid et al., *"An exoskeleton controlled by an epidural wireless brain–machine interface in a tetraplegic patient: a proof-of-concept demonstration"* Lancet Neurol., vol. 4422, no. 19, pp. 1–11, 2019.

[8] M. Alemi, A. Meghdari, A. Ghanbarzadeh, L. J. Moghadam, and A. Ghanbarzadeh, *"Impact of a social humanoid robot as a therapy assistant in children cancer*

treatment" Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8755, pp. 11–22, 2014.

[9] R. J. F. van den Heuvel, M. A. S. Lexis, and L. P. de Witte, *"ZORA Robot Based Interventions to Achieve Therapeutic and Educational Goals in Children with Severe Physical Disabilities"* Int. J. Soc. Robot., no. 0123456789, 2019.

[10] M. Chang, *Learning by Playing. 4th International Conference on E-Learning and Games, Edutainment. 2009.*

[11] A. Agah, J. J. Cabibihan, A. M. Howard, M. A. Salichs, and H. He, *"Preface,"* *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9979 LNAI, pp. V–VI, 2016.

[12] N. F. Lepora, A. Mura, M. Mangan, P. F. M. J. Verschure, M. Desmulliez, and T. J. Prescott, *"Preface,"* *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9793, pp. V–VIII, 2016.

[13] U. Qidwai, S. B. A. Kashem, and O. Conor, *"Humanoid Robot as a Teacher's Assistant: Helping Children with Autism to Learn Social and Academic Skills"*, J. Intell. Robot. Syst., 2019.

[14] H. M. Shim, J. G. Ryu, O. S. Kwon, E. H. Lee, H. K. Min, and S. H. Hong, *"Development of the walking assistant robot for the elderly"*, IFMBE Proc., vol. 14, no. 1, pp. 3003–3006, 2007.

[15] A. Mohamed, P. Novais, A. Pereira, G. V. González, and A. Fernández-Caballero, *"Ambient Intelligence - Software and Applications: 6th International Symposium on Ambient Intelligence (ISAmI 2015)"*, Adv. Intell. Syst. Comput., vol. 376, pp. 65–73, 2015.

[16] T. Březina, *"Mechatronics 2017 - Preface"*, Adv. Intell. Syst. Comput., vol. 644, p. v, 2018.

[17] S. Serino, A. Matic, D. Giakoumis, G. Lopez, and P. Cipresso, *"Pervasive computing paradigms for mental health: 5th international conference*, mindcare 2015

Milan, Italy, september 24–25, 2015 revised selected papers," Commun. Comput. Inf. Sci., vol. 604, pp. 186–195, 2016.

[18] Fraunhofer Institute for Manufacturing Engineering and Automation, *"Fraunhofer IPA"*, 2019. [En línea]. Disponible: https://www.care-o-bot.de/en/care-o-%bot-4.html. [Último acceso: 2019].

[19] Lung-Wen Tsai, *ROBOT ANALYSIS, The Mechanics of Serial ans Parallel Manipulators*, Department of Mechanical Engineering and Institute for Systems Research University of Meryland, A Wiley Interscience Publication, 1999.

[20] Nelli F. *(2018) Image Analysis and Computer Vision with OpenCV. In: Python Data Analytics*. Apress, Berkeley, CA. [En línea]. Disponible: https://doi-org.proxydgb.buap.mx/10.1007/978-1-4842-3913-1_14

[21] Wainschenker R., Massa J. M., *Procesamiento Digital de Imágenes*, Clase Teórico Práctica No. 1, Optativa Área Procesamiento de señales.

[22] Brahmbhatt S. *(2013) Basic Machine Learning and Object Detection Based on Keypoints. In: Practical OpenCV.* Apress, Berkeley, CA. [En línea]. Disponible: https://doi-org.proxydgb.buap.mx/10.1007/978-1-4302-6080-6_8

[23] Haralick, Shapiro, Gonzalez, Woods, *Imágenes binarias*, Computer and Robot Vision and Digital Image Processing. [En línea]. Disponible: http://www.sc.ehu.es/ccwgrrom/transparencias/pdf-vision-1-transparencias/capitulo-3.pdf

[24] Alvarez Sánchez, J. G., *Fabricación de un manipulador paralelo*, tesis de posgrado, Facultad de ingeniería, Universidad Nacional Autónoma de México, Ciudad de México, 2005.

[25] Aracil R., Saltarén R:J., et al, *Robots Paralelos: Máquinas con un pasado para una roótica del futuro*, Universidad Politécnica de Madrid, Madrid. Universidad Miguel Hernández, Elche. e-mail aracil@etsii.upm.es

[26] Koubaa A., *Robot Operating System (ROS)*: The Complete Reference(Volume 1), pp. 3-5, Springer, 2016

[27] Schkolnik Muller D. A., *Desarrollo de una herramienta gráfica de exploración de robots con ros*, trabajo de fin de grado, Facultad de ciencias físicas y matemáticas, Universidad de Chile, Santiago de Chile, 2015.

[28] ROS.org, *Documentation, Browse Software, News, Download*, Open Source Robotics Foundation, [En línea] Disponible: http://wiki.ros.org/es

[29] NVIDIA DEVELOPER, *Getting Started with Jetson Nano Developer Kit*, [En línea] Disponible: https://www.google.com/search?client=ubuntu&channel=fs&q=linea&ie=utf-8&oe=utf-8

[30] ROS.org, *rosserial*, Open Source Robotics Foundation, [En línea] Disponible: http://wiki.ros.org/rosserial

[31] Vaingast S., *Beginning Python Visualization: Crafting Visual Transformation Scripts*, segunda edición, Apress, 2014.

[32] Kalb I., *Learn to Program with Python*, Apress, California, USA. 2016.

Apéndice A

Códigos de la cinemática inversa implementados en Matlab

1. function y = foundC(u)
   Px = u(1);
   Py = u(2);
   Pz = u(3);
   phi = u(4);
   r1 =0.08;
   r2 =0.04;
   op1 = [cos(phi), sin(phi), 0; -sin(phi), cos(phi), 0; 0, 0, 1];
   op2 = [Px; Py; Pz];
   op3 = [r2-r1; 0; 0];
   C = op1*op2 + op3;

   y = C;

2. function y = theta1(u)
   t2 = u(1);
   t3 = u(2);
   Cx = u(3);
   Cy = u(4);
   Cz = u(5);
   a = u(6);
   b = u(7);
   A= -(a + b*sin(t3)*cos(t2) + b*sin(t3)*sin(t2) + b*cos(t3) + Cx + Cy + Cz);

B= 2*(a - b*sin(t3)*sin(t2) + b*sin(t3)*cos(t2));
C= a + b*sin(t3)*cos(t2) + b*sin(t3)*sin(t2) + b*cos(t3);
t1p = 2*atan((-B + sqrt(B^2 - 4*A*C))/(2*A));
t1n = 2*atan((-B - sqrt(B^2 - 4*A*C))/(2*A));
y=t1n;

3. function y = theta2(u)
   t3 = u(1);
   Cx = u(2);
   Cy = u(3);
   Cz = u(4);
   a = u(5);
   b = u(6);
   k = (Cx^2 + Cy^2 + Cz^2- a^2 - b^2)/(sin(t3)*2*a*b);
   t2 = acos(k);
   y = t2;

4. function y = theta3(u)
   Cy = u(1);
   b = u(2);
   t3 = acos(Cy/b);
   y = t3;

5. Animación
   function fcn(u)
   figure(1)
   clf
   t1p1=-u(1);
   t2p1=-u(2);
   t3p1=u(3);

   t1p2=-u(4);
   t2p2=-u(5);

```
t3p2=u(6);


t1p3=-u(7);
t2p3=-u(8);
t3p3=u(9);
rbspi1=[0,0,0];
rbspf1=[8,0,0];
rb1pi1=rbspf1;
rb1pf1=rb1pi1 +[10,0,0];
rb2pi1=rb1pf1;
rb2pf1=rb1pf1+[20,0,0];
rbfpi1=rb2pf1;
rbfpf1=rbfpi1-[4,0,0];
bspi1=rbspi1;
bspf1=rbspf1;
yrot=roty(t1p1);
b1pi1 = bspf1;
temp=rb1pf1-rb1pi1;
temp=temp*yrot;
b1pf1=rb1pi1+temp;
diff=b1pf1-rb2pi1;
b2pi1=rb2pi1+diff;
b2pf1=rb2pf1+diff;
yrot=roty(t2p1+t1p1);
xrot=rotz(-90+t3p1);
temp=b2pf1-b2pi1;
temp=temp*xrot*yrot;
temp=temp+b2pi1;
b2pf1=temp;
bfpi1=b2pf1;
bfpf1=(b2pf1-[4,0,0]);
bspi2=rbspi1;
```

```
bspf2=rbspf1;
yrot=roty(t1p2);
b1pi2 = bspf2; temp=rb1pf1-rb1pi1;
temp=temp*yrot;
b1pf2=rb1pi1+temp;
diff=b1pf2-rb2pi1;
b2pi2=rb2pi1+diff;
b2pf2=rb2pf1+diff;
yrot=roty(t2p2+t1p2);
xrot=rotz(-90+t3p2);
temp=b2pf2-b2pi2;
temp=temp*xrot*yrot;
temp=temp+b2pi2;
b2pf2=temp;
bfpi2=b2pf2;
bfpf2=(b2pf2-[4,0,0]);
phi2= 120;
zrot2=rotz(phi2);
bspi2=bspi2*zrot2;
bspf2=bspf2*zrot2;
b1pi2=b1pi2*zrot2;
b1pf2=b1pf2*zrot2;
b2pi2=b2pi2*zrot2;
b2pf2=b2pf2*zrot2;
bfpi2=bfpi2*zrot2;
bfpf2=bfpf2*zrot2;
bspi3=rbspi1;
bspf3=rbspf1;
yrot=roty(t1p3);
b1pi3 = bspf3;
temp=rb1pf1-rb1pi1;
temp=temp*yrot;
```

```
b1pf3=rb1pi1+temp;
diff=b1pf3-rb2pi1;
b2pi3=rb2pi1+diff;
b2pf3=rb2pf1+diff;
yrot=roty(t2p3+t1p3);
xrot=rotz(-90+t3p3);
temp=b2pf3-b2pi3;
temp=temp*xrot*yrot;
temp=temp+b2pi3;
b2pf3=temp;
bfpi3=b2pf3;
bfpf3=(b2pf3-[4,0,0]);
phi3=240;
zrot3=rotz(phi3);
bspi3=bspi3*zrot3;
bspf3=bspf3*zrot3;
b1pi3=b1pi3*zrot3;
b1pf3=b1pf3*zrot3;
b2pi3=b2pi3*zrot3;
b2pf3=b2pf3*zrot3;
bfpi3=bfpi3*zrot3;
bfpf3=bfpf3*zrot3;
plot3([bspi1(1),bspf1(1)],[bspi1(2),bspf1(2)],[bspi1(3),bspf1(3)],'k')
hold on
grid on
plot3([b1pi1(1),b1pf1(1)],[b1pi1(2),b1pf1(2)],[b1pi1(3),b1pf1(3)],'r')
plot3([b2pi1(1),b2pf1(1)],[b2pi1(2),b2pf1(2)],[b2pi1(3),b2pf1(3)],'r')
plot3([bfpi1(1),bfpf1(1)],[bfpi1(2),bfpf1(2)],[bfpi1(3),bfpf1(3)],'r')
plot3([bspi2(1),bspf2(1)],[bspi2(2),bspf2(2)],[bspi2(3),bspf2(3)],'k')
plot3([b1pi2(1),b1pf2(1)],[b1pi2(2),b1pf2(2)],[b1pi2(3),b1pf2(3)],'g')
plot3([b2pi2(1),b2pf2(1)],[b2pi2(2),b2pf2(2)],[b2pi2(3),b2pf2(3)],'g')
plot3([bfpi2(1),bfpf2(1)],[bfpi2(2),bfpf2(2)],[bfpi2(3),bfpf2(3)],'g')
```

```
plot3([bspi3(1),bspf3(1)],[bspi3(2),bspf3(2)],[bspi3(3),bspf3(3)],'k')
plot3([b1pi3(1),b1pf3(1)],[b1pi3(2),b1pf3(2)],[b1pi3(3),b1pf3(3)],'b')
plot3([b2pi3(1),b2pf3(1)],[b2pi3(2),b2pf3(2)],[b2pi3(3),b2pf3(3)],'b')
plot3([bfpi3(1),bfpf3(1)],[bfpi3(2),bfpf3(2)],[bfpi3(3),bfpf3(3)],'b')
origen = [0,0,0];
P1 = [5,0,0];
P2 = [0,5,0];
P3 = [0,0,5];
line([origen(1),P1(1)],[origen(2),P1(2)],[origen(3),P1(3)]);
line([origen(1),P2(1)],[origen(2),P2(2)],[origen(3),P2(3)]);
line([origen(1),P3(1)],[origen(2),P3(2)],[origen(3),P3(3)]);
text(5,0,0,'x');
text(0,5,0,'y');
text(0,0,5,'z');
set(gca,'CameraPosition',[1 1 1]); view(45,45)
```

Apéndice B

Publicación presentada en el congreso COMROB

Esta publicación se presentó en el día 29 de octubre de año 2020 en el XXII Congreso Mexicano de Robótica 2020 en su modalidad como I Congreso Virtual COMROB 2020.
El XXII COMRob 2020 es un evento organizado por la Asociación Mexicana de Robótica e Industria A.C. (AMRob), teniendo como objetivo el difundir y promover la robótica. Este evento atrae investigadores, profesionales, industriales y estudiantes de los diversos niveles académicos con los objetivos de presentar los avances y resultados de sus proyectos educativos, de investigación, desarrollo tecnológico y aplicaciones industriales.
La participación se realizó en tiempo y forma según lo establecido por el comité organizador. En la figura B.1 se muestra e horario de participación, posteriormente se anexa la publicación.

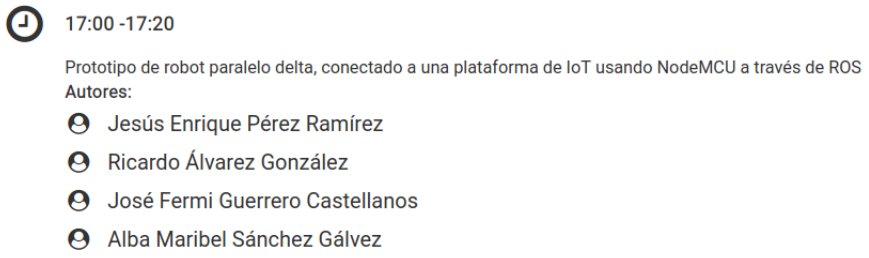


Figura B.1: Horario de participación

Prototipo de robot paralelo delta, conectado a una plataforma de IoT usando NodeMCU a través de ROS

J. E. Pérez-Ramírez[1] R. Álvarez-González[1] J. F. Guerrero-Castellanos [1] A. M. Sánchez-Gálvez[2]
[1] Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Electrónica,
18 sur y Av. San Claudio, Ciudad Universitaria, Puebla, Pue., 72570. México
jesus.perezra@alumno.buap.mx, ricardo.alvarez@correo.buap.mx, fermi.guerrero@correo.buap.mx
[2] Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación,
14 sur y Av. San Claudio, Ciudad Universitaria, Puebla, Pue., 72570. México
alba.sanchez@correo.buap.mx

***Resumen*— Se presenta un sistema cuya principal característica consiste en que el procesamiento que rige su control se encuentra conectado a él de manera remota. Es un prototipo de robot paralelo tipo Delta que se comunica de manera inalámbrica con un sistema que procesa su cinemática inversa, éste le proporciona los ángulos requeridos para alcanzar un punto específico dentro de su espacio de trabajo.**

I. INTRODUCCIÓN

De manera general, los sistemas robóticos automatizados se ven involucrados cada vez mas en diversos sectores de la actividad humana; se han convertido en parte indispensable del mismo [1]; a consecuencia de esto, han surgido corrientes ideológicas y plataformas tecnológicas como el Internet de las cosas [2], la industria 4.0 [3] y las ciudades inteligentes [4] que contribuyen al desarrollo y ampliación de éstos. Además, fenómenos sociales y naturales son escenarios que han abierto la posibilidad de ampliar y crear nuevas aplicaciones donde la interacción humana sea reducida. Los motivos para asistir o sustituir las tareas humanas son extensos; desde hacer mas fácil y cómoda la vida humana hasta realizar trabajos que ponen en riesgo la salud o la vida; muchos de tipo industrial y también, como asistentes dentro del hogar.

De manera particular, los robots Delta han ganado terreno en el sector industrial desde la última década del siglo pasado, enfocados principalmente en tareas de tipo "pick and place", tienen capacidades como: Alta precisión, mover cargas pesadas, rigidez estructural y rapidez [5], estas características han abierto nuevos campos de aplicación estando presente en diversos proyectos de actualidad dentro y fuera del sector industrial como parte de sistemas más complejos. Esto ha motivado a que este trabajo no solo consista en la implementación de un prototipo robótico, sino también, en añadir la capacidad de poder conectarlo a una red inalámbrica con el fin de ser operado a distancia.

De esta manera, se obtiene la capacidad de trabajar en colaboración con otros sistemas (robots móviles, sistemas de inteligencia artificial, sistemas de visión por computadora, drones, entre otros) que complementándose, realicen tareas con alto grado de dificultad. Éstos pueden ser incorporados en diversas áreas de automatización, saliendo incluso del sector industrial de fabricación; almacenes y asistencia [6] son algunos ejemplos recientes donde la inclusión de los robots complementan la actividad humana.

II. DESCRIPCIÓN DEL SISTEMA

Tomando parte de un sistema más complejo, el prototipo presentado es un sistema controlado por teleoperación. Se desea que la posición del efector final del robot Delta, alcance un punto específico dentro de su espacio de trabajo, este punto es una coordenada que el usuario proporciona a través de una conexión inalámbrica como se observa en la figura 1. La descripción del funcionamiento de este

Fig. 1. Sistema con teleoperación

sistema se observa en el diagrama de la figura 2, se inicia cuando el usuario o teleoperador toma una coordenada del espacio de trabajo del robot Delta y la introduce en Matlab, posteriormente, la cinemática inversa procesa la coordenada deseada y como resultado se obtiene la posición angular de cada uno de los tres actuadores del robot Delta. Estos datos son transimitidos inalámbricamente desde el Nodo 1 hasta el Nodo 2, de esta manera los tres actuadores del robot Delta reciben la posición angular correspondiente colocando el efector final en la posición deseada.

Es importante mencionar que la conectividad se encuentra configurada con ROS ya que proporciona diversas herramientas para gestionar las conexiones tanto de los dispositivos conectados como la información y su flujo. Además, gracias a que todo elemento conectado es visualizado dentro de la red como un nodo, es posible conectar sistemas, sensores, programas de simulación, actuadores y dispositvos de diferente naturaleza, únicamente agregando las bibliotecas de ROS necesarias.

Finalmente, se resalta que este sistema realiza los procesamientos más complejos (calcular la cinemática inversa) de manera remota respecto al propio robot Delta, esto es, no se requiere que la unidad de procesamiento se encuentre en la misma ubicación que el prototipo. Esta cualidad resulta una ventaja cuando las condiciones requieran una alta eficiencia en consumo energético.

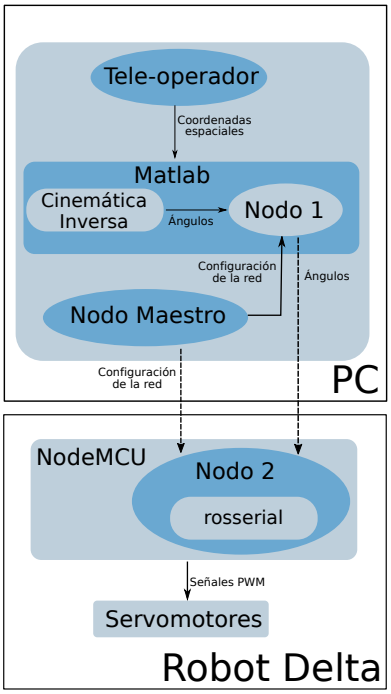


Fig. 2. Diagrama del sistema

III. Robot Paralelo Delta

Los robots con arquitectura paralela han sido objeto de estudio desde hace varios años atrás como plataformas para la simulación de vuelos en el área aereoespacial e incluso como equipo para supervisar estándares de calidad [7]. Un diseño particular que se desprende de éstos es el robot paralelo tipo Delta que es una opción interesante en un contexto en el que los sistemas robóticos automátizados ganan terreno (en gran medida gracias a la inteligencia artificial y la conectividad) en un sector que hace años la mayoría era mano obrera.

Como se observa en la figura 3, el robot Delta se compone de dos bases; la base de mayor tamaño se encuentra fija, mientras que la menor es móvil dándole soporte al efector final (la función del efector final depende de la aplicación). Ambas bases conservan una relación paralela unidas por tres brazos articulados o cadenas cinemáticas equidistantes controladas cada una por un actuador [7].

A. Notación

Las figuras 3 y 4 muestran el nombre de cada parámetro, así como sus dimensiones en relación con la tabla 1. Los dos eslabones de cada cadena cinemática utilizan los ángulos θ para describir su posición en el espacio, mientras que el ángulo ϕ posiciona cada cadena cinemática de manera equidistante respecto a la base superior del robot Delta.

Tabla 1: Parámetros y especificaciones físicas

Pieza	Parámetro	Símbolo	Unidades (mm)
Base Superior	Radio base fija	r1	800
Base Inferior	Radio base móvil	r2	400
Brazo	Longitud	a	100
Antebrazo	Longitud	b	200

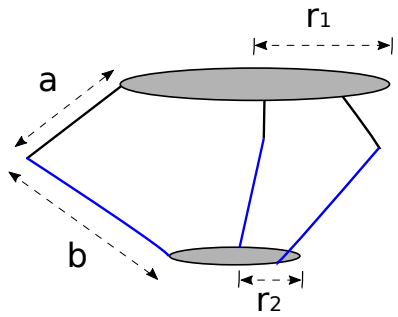


Fig. 3. Notación para el Robot Delta

B. Cinemática Inversa

El enfoque del modelo matemático para este robot se realizó de manera geométrica como se expone en otras publicaciones [8]. En la figura 4 se muestra una cadena cinemática, en ésta se observan puntos clave; A, B, C, O y P. Estos puntos forman segmentos bien definidos ($\overline{OA}$, $\overline{AB}$, $\overline{BC}$, $\overline{CP}$, $\overline{OP}$) que, siguiendo la trayectoria que forman es posible observar que se crea un ciclo cerrado. Es posible notar que del punto O al punto P existen dos caminos a seguir, ver ecuación 1.

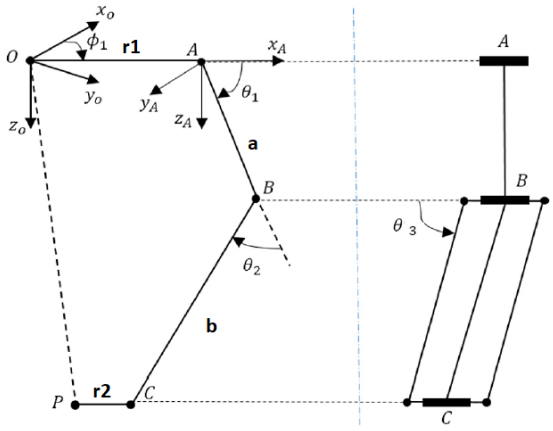


Fig. 4. Cadena cinemática vista frotal

$$\overline{OA} + \overline{AB} + \overline{BC} + \overline{CP} = \overline{OP} \quad (1)$$

De la figura 4 se observan dos sistemas de referencia. El sistema de coordenadas con origen en O es el sistema

principal del cual se rige todo el sistema, sin embargo, únicamente para facilitar el análisis geométrico se recurre a recorrer dicho origen al punto A. Por lo tanto, la trayectoria que describe el recorrido del punto A al punto C se muestra en la ecuación 2. La importancia de encontrar el punto C, radica en que conociendo este punto ($Cx \quad Cy \quad Cz$), es posible encontrar las expresiones matemáticas que dan solución a los valores de los ángulos θ_1, θ_2, y θ_3. De esta manera se estaría resolviendo la cinemática inversa para una cadena cinemática1, este proceso se repite para cada una.

$$\overline{AB} \quad + \quad \overline{BC} \quad = \quad \overline{OP} \quad - \quad \overline{OA} \quad + \quad \overline{CP} \tag{2}$$

La ecuación 2 se puede reescribir con los parámetros geométricos del mecanismo como se muestra en la ecuaciones 3 y 4. Es importante mencionar que se debe añadir una matriz de rotación como se observa en la ecuación 5. Sin embargo, el valor de ϕ es diferente para cada una de las tres cadenas cinemáticas, puede tomar valores de $\phi = 0$, $\phi = 120$ y $\phi = 240$ grados respectivamente.

$$\begin{bmatrix} acos(\theta_1) \\ 0 \\ asin(\theta_1) \end{bmatrix} + \begin{bmatrix} bsin(\theta_3)cos(\theta_1+\theta_2) \\ bcos(\theta_3) \\ bsin(\theta_3)sin(\theta_1+\theta_2) \end{bmatrix} = \tag{3}$$

$$\begin{bmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - \begin{bmatrix} r1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} r2 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

Basándose de las ecuaciones 3 y 4, y además considerando el punto A como origen, es posible realizar las siguientes afirmaciones con el fin de encontrar una expresión con la cual sea posible calcular el punto C.

$$AB \quad + \quad BC = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = \begin{bmatrix} acos(\theta_1) + bsin(\theta_3)cos(\theta_1+\theta_2) \\ bcos(\theta_3) \\ asin(\theta_1) + bsin(\theta_3)sin(\theta_1+\theta_2) \end{bmatrix} = \tag{5}$$

$$\begin{bmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - \begin{bmatrix} r1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} r2 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

Por lo tanto:

$$\begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = \begin{bmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - \begin{bmatrix} r1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} r2 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

De esta manera, de la ecuación 7 se tiene una expresión matemática con la cual es posible encontrar las coordenadas del punto C ya que, ϕ, P, $r1$ y $r2$ son valores conocidos. Las siguientes ecuaciones describen la expresión matemática para encontrar los valores de θ_1, θ_2 y θ_3. Como se verá, cada una de éstas depende de algún valor del punto C; así se justifica la necesidad de obtener este punto como primer paso.

De la ecuación 5 se observa que es posible obtener el valor de θ_3 de manera directa realizando un despeje del segundo renglón de la matriz, ver ecuación 8.

$$C_y = bcos(\theta_3) \quad \Longrightarrow \theta_3 = \arccos\left(\frac{C_y}{b}\right) \tag{8}$$

Mientras que para obtener θ_2 se realiza la suma de cuadrados de todas las filas de la matriz de la ecuación 5; después de algunas operaciones matemáticas el resultado se expresa en la ecuación 9.

$$\theta_2 = \arccos\left(\frac{C_x + C_y + C_z + a^2 + b^2}{2ab\sin(\theta_3)}\right) \tag{9}$$

Para encontrar θ_1 se utiliza el mismo procedimiento que con θ_2, se realiza la suma de $C_x, \quad C_y \quad y \quad C_z$ como se muestra en la ecuación 10. Posteriormente, esta ecuación se resuelve para θ_1. Cabe mencionar que el procedimiento matemático se realizó con la ayuda de Matlab.

$$C_x + C_y + C_z = acos(\theta_1) + bsin(\theta_3)cos(\theta_1 + \theta_2) + bcos(\theta_3) + asin(\theta_1) + bsin(\theta_3)sin(\theta_1+\theta_2) \tag{10}$$

De esta manera se tiene la cinemática inversa del robot Delta. Las tres ecuaciones para encontrar θ_1, θ_2 y θ_3 son introducidas a Simulink donde se procesa el punto deseado para cada cadena cinemática obteniendo la posición angular de cada actuador como se muestra en la figura 5, los bloques nombrados "Cadena Cinemática" 1, 2 y 3 tienen a su salida t1, t2 y t3, estas variables representan a θ_1, θ_2 y θ_3 respectivamente.
Es importante mencionar que el único ángulo actuado para cada cadena cinemática es t1, calcular t2 y t3 ayudaron a encontrar las expresiones matemáticas pero, en este trabajo, no se utilizan más.

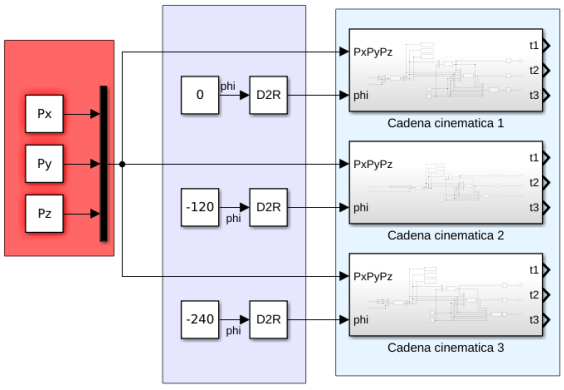


Fig. 5. Cinemática inversa en Simulink

IV. Configuración de la red ROS

ROS se define como un meta sistema operativo que proporciona herramiemtas para gestionar proyectos de robótica dentro de una red [9], permite tener control y administración de la misma y sus elementos, además gracias a que trabaja con nodos, es posible sumar o restar módulos de manera sencilla. El nodo principal o nodo maestro es el primero en ser iniciado y él se encarga de configurar los nodos secundarios. Como se observa en la figura 2, el Nodo Maestro se encuentra en la misma computadora que Matlab y por consecuencia el Nodo 1.

Para conectar el robot Delta a ROS fue necesario elegir una interfaz que proporcionará su integración a la red, la opción elegida fue la tarjeta NodeMCU ESP8266 la cuál presenta cualidades como: bajo consumo energético, bajo costo, contiene un módulo wifi integrado; reducido tamaño físico; algunos de sus puertos de propósito general admiten señales PWM (idóneas para el accionamiento de servomotores); su programación es posible realizarla a través del IDE Arduino; ya que es un sistema basado en código abierto existe una gran comunidad que da soporte a la documentación del mismo; es posible correr paqueterias de ROS dentro de ésta; entre otras [10].

La red que se implementó se ilustra en la figura 6. ROS trabaja con nodos y tópicos, los nodos son los elementos que emiten o reciben información mientras que los tópicos son los canales por los cuales viajan los datos. Se observa que el Nodo 1 se encuentra publicando (emitiendo) los ángulos θ_1 a los tópicos /servo1, /servo2 y /servo3 respectivamente, el Nodo 2 se suscribe a cada uno de los tópicos para poder acceder a los ángulos.

Es así como los ángulos calulados por la cinemática inversa llegan de manera remota hasta los servomotores del robot Delta.

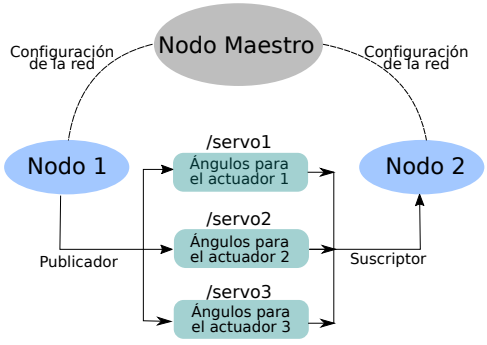


Fig. 6. Red ROS

A. ROSSERIAL

En esta parte del proyecto el objetivo es configurar un nodo que sea capaz de tomar datos de la red (suscriptor) para un posterior procesamiento de los mismos, como se mostró en la figura 6, se trata del *Nodo 2*.

La tarjeta de NodeMCU ESP8266 no tienen la capacidad de soportar un sistema opertativo por el cuál sea posible hacer una instalación completa de ROS, sin embargo existe la posibilidad de acceder de manera límitada haciendo uso de paquetes y bibliotecas a través de un protocolo de comunicación conocido como *rosserial*.

Rosserial es un protocolo de comunicación de tipo serial cuyo objetivo es conectarse a una red ROS a través de un puerto serie o una toma de red [11]. Disponible para algunas tarjetas de desarrollo como Arduino, NodeMCU ESP8266, entre otras permite la conectividad de manera limitada, es decir, con un número reducido de nodos y un reducido tamaño del buffer de datos.

Para ocupar esta alternativa, se realizó la descarga, instalación y compilación de los paquetes *rosserial* [11], cabe mencionar que la versión de ROS utilizada es *ROS Melodic Morenia* en Ubuntu 18.04.4 LTS. Después de esto debe estar un directorio con el nombre *rosserial* dentro de la carpeta que guarda el espacio de trabajo de ROS, *catkin_ws*. De esta manera se encuentran los algoritmos configurados para generar un nodo publicador, suscriptor o ambos.

Posteriomente, la rutina para la tarjeta NodeMCU se describe con la ayuda del IDE de Arduino, en este algoritmo se colocan las cabeceras y funciones que complementan la configuración del Nodo 2, se determina que éste se suscribe a los tópicos /servo1, /servo2 y /servo3; se realiza la configuración de la conexión ethernet y además, se asignan las salidas de las señales PWM.

V. Resultados

El sistema tiene un determinado protocolo de encendido y operación, en la figura 7 se muestra un diagrama que lo describe. De acuerdo a lo planteado en las secciones

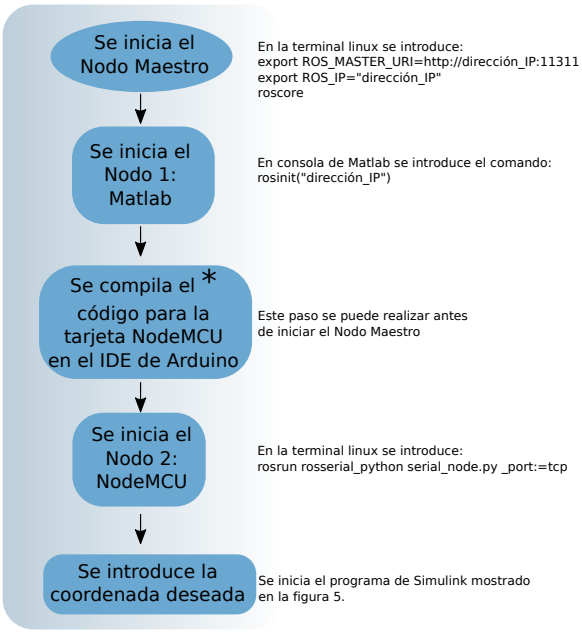


Fig. 7. Secuencia de inicio del sistema

anteriores y el esquema de la figura 7, se presenta el resultado del funcionamiento del sistema. En la figura 8 se observa que, con la ayuda de *Node Graph*, es posible

visualizar de manera gráfica la conexión de los nodos a través de los tópicos.

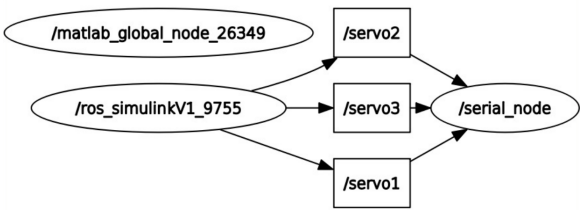


Fig. 8. Red ROS

Una característica importante es que el procesamiento mas demandante, el cálculo de la cinemática inversa, se está realizando en una localidad remota al prototipo, la operación del robot Delta esta a cargo de una tarjeta de reducido tamaño y bajo consumo energético, esta condición libera al prototipo de soportar un peso adicional y una carga energética que para algunas aplicaciones puede ser determinante. La implementación del robot Delta se muestra en la figura 9.

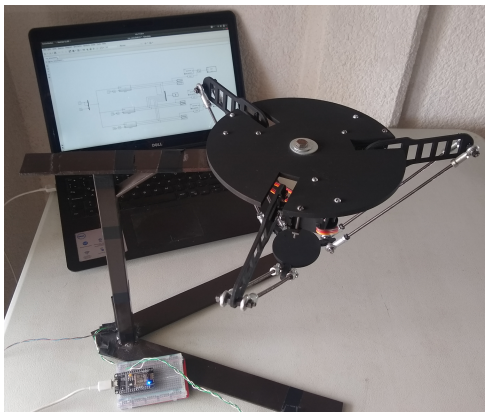


Fig. 9. Prototipo del robot Delta

Los servomotores utilizados para la construcción del prototipo tiene la matrícula MG995 con 15Kg-cm. El primer eslabón y ambas bases son de MDF con el fin de disminuir el peso. Además, ambos extremos del segundo eslabón están sujetos mediante rotulas esféricas.

VI. CONCLUSIONES

- Realizar el control y procesamiento de manera remota representa una gran ventaja si la aplicación requiere que el robot delta se desplace en diferentes escenarios como parte de un sistema móvil.
- La tarjeta NodeMCU ESP8266 se encuentra limitada en dos aspectos importantes, en la cantidad de nodos y en el tamaño del buffer de datos, sin embargo, a pesar de que los tres ángulos transmitidos son de tipo Float, no presentó problema.
- La implementación se pudo haber desarrollado con cualquiera de las dos tarjetas de desarrollo, Arduino o NodeMCU ESP8266. Sin embargo, el costo marcó la diferencia ya que conseguir un módulo wifi para arduino resulta un gasto doble que comprar una tarjeta NodeMCU ESP8266.
- Se considera que el efector final del robot Delta se encuentra situado en el centro de la base móvil. Dependiendo de la tarea que se proponga realizar, es posible adaptar el adecuado.
- Si bien ROS es un sistema que no es recomendable para determinadas aplicaciones, la adaptación de prototipos se ve beneficiada con él, ya que permite conocer información variada tanto de los nodos como de los tópicos y sus mensajes.
- Es importante configurar el tipo de mensaje que se desea transmitir ya que para ambos nodos, publicador y suscriptor, debe ser el mismo. De lo contrario, no se realiza la comunicación.

REFERENCIAS

[1] M. HAHELE *"Robots Conquer the World [Turning Point]"*, in IEEE Robotics & Automation Magazine, vol. 23, no. 1, pp. 120-118, March 2016, doi: 10.1109/MRA.2015.2512741.

[2] S. K. VISHWAKARMA, P. UPADHYAYA, B. KUMARI AND A. K. MISHRA, "SMART ENERGY EFFICIENT *Home Automation System Using IoT," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Ghaziabad, India, 2019, pp. 1-4, doi: 10.1109/IoT-SIU.2019.8777607.

[3] S. SHAMIM, S. CANG, H. YU, Y. LI, L. Y. CHEN AND X. YAO, *"How firms in emerging economies can learn industry 4.0 by extracting knowledge from their foreign partners? A view point from strategic management perspective"*, 2019 International Conference on Advanced Mechatronic Systems (ICAMechS), Kusatsu, Shiga, Japan, 2019, pp. 390-395, doi: 10.1109/ICAMechS.2019.8861622.

[4] A. FOUNOUN AND A. HAYAR, *"Evaluation of the concept of the smart city through local regulation and the importance of local initiative"*, 2018 IEEE International Smart Cities Conference (ISC2), Kansas City, MO, USA, 2018, pp. 1-6, doi: 10.1109/ISC2.2018.8656933.

[5] S. STAICU AND D. C. CARP-CIOCARDIA, *"Dynamic analysis of Clavel's Delta parallel robot"*, 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 2003, pp. 4116-4121 vol.3, doi: 10.1109/ROBOT.2003.1242230.

[6] INTERNATIONAL FEDERATION OF ROBOTICS *The Impact of Robots on Productivity, Employment and Jobs*, published by International Federation of RoboticsFrankfurt, Germany. April 2017 updated April 2018

[7] LUNG-WEN TSAI, *ROBOT ANALYSIS, The Mechanics of Serial and Parallel Manipulator*, Department of Mechanical Engineering and Institute for Systems Research, University or Maryland, A Wiley-Interscience Publication 1999, pp.116-117.

[8] LUNG-WEN TSAI, *"ROBOT ANALYSIS, The Mechanics of Serial and Parallel Manipulator"*, Department of Mechanical Engineering and Institute for Systems Research, University or Maryland, A Wiley-Interscience Publication 1999, pp.134-142.

[9] LENTIN JOSEPH, *"Robot Operating System (ROS) for Absolute Beginners: Robotics Programming Made Easy"*, Apress, corrected publication 2018. pp. 132-135

[10] GITHUB, "ESP8226/ARDUINO", 2020 GitHub, Inc. Available: https://github.com/esp8266/Arduino

[11] LENTIN JOSEPH, *"Robot Operating System (ROS) for Absolute Beginners: Robotics Programming Made Easy"*, Apress, corrected publication 2018. pp. 228-233