UNIVERSIDAD SAN FRANCISCO DE QUITO

Colegio de Ciencias e Ingeniería

Diseño, construcción y control de un brazo robótico

Freddy M. Alonzo,
Miguel E. Bravo

Luis Caiza, M.S., Director de Tesis

Tesis de grado presentada como requisito
para la obtención del título de Ingeniero Electrónico

Quito, Diciembre de 2014

Universidad San Francisco de Quito

Colegio de Ciencias e Ingeniería

HOJA DE APROBACIÓN DE TESIS

Diseño, construcción y control de un brazo robótico

Freddy Alonzo, Miguel Bravo

Luis Caiza, M. Sc.
Director de la tesis _____

Omar Aguirre, M. Sc.
Miembro del Comité de Tesis _____

Bernard Herrera, Ing.
Miembro del Comité de Tesis _____

Omar Aguirre, M. Sc
Director de la Carrera de Ing. Electrónica _____

Ximena Córdoba, Ph. D.
Decana de la Escuela de Ingenierías
Colegio de Ciencias e Ingenierías _____

Quito, Diciembre del 2014

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído la Política de Propiedad Intelectual de la Universidad San Francisco de Quito y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo de investigación quedan sujetos a lo dispuesto en la Política.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo de investigación en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma: _____

Nombre: Freddy Mosíah Alonzo Villegas

C. I.: 1714550702

Firma: _____

Nombre: Miguel Esteban Bravo Valencia

C. I.: 1715545420

Lugar: Quito, Ecuador Fecha: diciembre de 2014

DEDICATORIA

Yo Miguel Bravo dedico este proyecto a mis padres Miguel y Angeline, a mi novia Josy, a mi hermana Estefanía y mis hermanos Matheo y Benjamín, por haberme ayudado, apoyado, y siempre motivado para seguir adelante y jamás decaer.

Yo, Freddy Alonzo, dedico este proyecto a mis padres, Freddy y Norma y a mi hermana Raquel, quienes me han brindado su apoyo incondicional en cada aspecto de mi vida y también me han ayudado a alcanzar mis metas académicas.

AGRADECIMIENTOS

Agradecemos a Dios por haber hecho posible la culminación de este proyecto. A nuestros padres por el apoyo y la motivación brindada. Y a nuestros profesores Luis Caiza y Bernard Herrera por haber sabido esclarecer nuestras dudas y haber sabido transmitir sus conocimientos sobre la temática del proyecto.

RESUMEN

El presente proyecto consiste en el diseño, construcción y control de un brazo robótico automatizado con cuatro grados de libertad. Se utilizan principios de mecánica para realizar el diseño y simulación del sistema, además de un amplio conocimiento en electrónica y programación para lograr automatizar de manera óptima los movimientos de la estructura. Paso a paso se detalla la construcción y montaje del brazo, con sus actuadores y sensores necesarios. Se utiliza como controlador una tarjeta Arduino Mega, la cual permite comandos inalámbricos utilizando su módulo WiFi, además de un control sencillo con sus módulos tipo puente H. Se programa también una interfaz gráfica para realizar el control vía Web de 2 tipos, y se emplea también un circuito para el control manual con selección de sentido de rotación y velocidad de motores. Finalmente, se utilizan modelos cinemáticos y modelos geométricos para el cálculo de la cinemática inversa necesaria para traducir las coordenadas geométricas en ángulos de rotación de cada motor.

ABSTRACT

The current project consists in the design, construction and control of an automated robotic arm with four degrees of freedom. Principles of mechanics are used to construct the design and simulation of the system, along with an extensive knowledge in electronics and programming in order to optimally achieve the movements of the structure. The construction and mounting of the arm are described step by step, with its necessary actuators and sensors. An Arduino Mega card is being used as a controller, which allows wireless commands by using its WiFi module, and also a simple control with its H-bridge type modules. A graphical interface is also programmed for control via Web of two types, and a circuit is employed for manual control with selectable rotation direction and motor speeds. Finally, geometric and kinematic models are employed for the necessary inverse kinematics calculation to translate the geometric coordinates into rotation angles for each engine.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN Y DEFINICIÓN DEL PROYECTO

1.1 Introducción

El presente trabajo consiste en el diseño, construcción y control de un sistema de brazo robótico dedicado a la manipulación de objetos. Se abarcan temas relacionados con el diseño mecánico, electrónico y programación de controladores con el fin de automatizar el sistema.

1.2 Objetivo principal

Diseñar, construir y controlar un brazo robótico para manipulación de objetos, mediante el uso de software de simulación y elementos electromecánicos.

1.3 Objetivos específicos, metas y actividades

Objetivo Específico	Meta	Actividades
Realizar un diseño factible de la estructura del brazo robótico.	Conocer los elementos que están disponibles en el mercado.	Investigar el mercado global en busca de dispositivos existentes.
		Enlistar los materiales y herramientas que se necesitarán.
	Poder visualizar de la forma más acertada posible la estructura construida.	Realizar un diseño en papel (2D).
		Transmitir el diseño a un software de simulación (3D).
	Conseguir una idea clara de cuál sería el funcionamiento del sistema.	Usando software especializado simular el sistema para verificar su funcionamiento.
		Realizar un estudio de limitaciones del sistema según los elementos disponibles.
Construir el brazo robótico de acuerdo al diseño previo.	Utilizando los componentes y herramientas enlistadas, montar la estructura.	Utilizando como guías el diseño 2D y 3D realizar el montaje de la estructura de soporte del sistema.
		Realizar actividades de perforación, soldadura, corte, etc.
	Adaptar todos los elementos y actuadores en el sistema.	Montar en la estructura los motores.
		De ser necesario alterar el actuador para una mejor adaptación en el sistema.
Programar el sistema para que cumpla con un funcionamiento adecuado.	Obtener una idea clara de la estructura de la programación.	Realizar en papel un diagrama de las funciones, lazos, condicionamientos y otras estructuras que se requieren para el funcionamiento.
	Automatizar el sistema con la programación implementada.	Seleccionar el dispositivo que se usará para el control de la estructura.
		Programar el dispositivo y simular de ser posible.
		Realizar las conexiones entre el sistema de control y la estructura, y verificar su funcionamiento.

Tabla 1. Objetivos

1.4 Antecedentes

El uso de sistemas robóticos en la industria, para cumplir funciones que requieren extrema precisión ha ido en ascenso en las últimas décadas. El desarrollo de estos sistemas se ha enfocado en mejorar ciertos aspectos como resistencia para trabajar en diferentes condiciones, precisión con la que se realizan movimientos, multifuncionalidad (manipulación, corte, perforación, etc.), adaptabilidad en diferentes entornos de trabajo y la independencia en su funcionamiento, es decir que tenga la capacidad de tomar decisiones respecto a su actuación. En el área educativa, los prototipos o sistemas a escala, han sido de gran ayuda para adquirir conocimientos relacionados a la robótica, grados de libertad, sistemas de transmisión, ejes, movimiento, etc., de una forma más didáctica y palpable, pero no siempre es fácil obtener acceso a uno. Por lo tanto, dados todas estas utilidades, el diseño propio y construcción de prototipos de brazo robótico para manipulación, corte láser, escaneo o cualquier otra función, y que tenga un costo accesible tanto para la industria como para la educación, es un buen tema a considerar como proyectos de desarrollo, por estudiantes de ingeniería electrónica.

1.5 Justificación

El presente trabajo se realiza con un propósito didáctico para enseñanza en la carrera de Ingeniería Eléctrica y Electrónica en la Universidad San Francisco de Quito. Materias como control automático, control digital, automatismos industriales, introducción a la robótica, etc., podrían utilizar el sistema presentado para proyectos, clases y demostraciones. La elección del tema referente a diseño, construcción y programación de un sistema automatizado, tiene bases en los conocimientos y afinidades de quienes lo realizan. La problemática que se pretende satisfacer, es la necesidad de generar diseños propios u originales de sistemas de brazo robótico, que puedan ser empleados en la enseñanza y en la investigación.

CAPÍTULO 2: MARCO TEÓRICO

En este capítulo, se definen términos importantes que son utilizados a lo largo de todo el proyecto. También se muestra información técnica relevante respecto a los equipos y dispositivos elegidos para el desarrollo del mismo, así como la metodología implementada.

2.1 Marco teórico

2.1.1 Actuador

En todo sistema con capacidad de movimiento, el actuador es todo dispositivo que puede transformar la energía hidráulica, eólica, eléctrica, etc. en movimiento. (Mena, 2011). En este proyecto se utilizan motores de corriente continua.

2.1.2 Algoritmo

Es un conjunto de reglas que se encuentran en cierta secuencia, que sirven para cumplir con una función específica (Mena, 2011). En este caso la programación del sistema posee una serie de algoritmos que llevan a un funcionamiento completo del sistema.

2.1.3 Arduino

Arduino es una herramienta para crear computadoras que interactúan con el mundo exterior con mucha más facilidad que una computadora de escritorio. Es una plataforma tipo código abierto que se basa en un microcontrolador y un entorno de programación para escribir software. (Arduino – Introduction, s.f.). Las tarjetas Arduino tienen una gran versatilidad a la hora de emplearlas en un proyecto de este tipo, ya que el tipo de conexión es sencillo y permite tanto leer como escribir variables digitales y analógicas. Además, en lo concerniente a la programación, Arduino dispone de una gran cantidad de librerías que permiten controlar de manera más eficiente dispositivos como motores servo, motores a paso, etc.

2.1.4 Arduino Mega 2560

La plataforma Arduino se produce en diferentes tipos de tarjetas que varían en el microcontrolador utilizado, el tamaño y la cantidad de puertos de entrada y salida. Debido a la gran cantidad de puertos necesarios para controlar el sistema en conjunto, además de la cantidad de datos que se maneja, los cálculos que se requiere procesar y las librerías que se necesita implementar, se decide utilizar la tarjeta Arduino Mega 2560, mostrada en la Figura 1, la cual tiene las características requeridas (mostradas en la Tabla 2) para el proyecto.

Microcontrolador	ATmega2560
Voltaje de operación	5V
Voltaje de alimentación (recomendado)	7-12V
Voltaje de alimentación (límites)	6-20V
Pines de I/O digitales	54 (15 pueden producir salida PWM)
Pines de entradas analógicas	16
Corriente DC por cada pin I/O	40mA
Corriente DC para el pin 3.3V	50mA
Memoria Flash	256KB de los cuales 8 son usadas por el bootloader
SRAM	8KB
EEPROM	4KB
Velocidad del Reloj	16MHz

Tabla 2. Características de Arduino MEGA

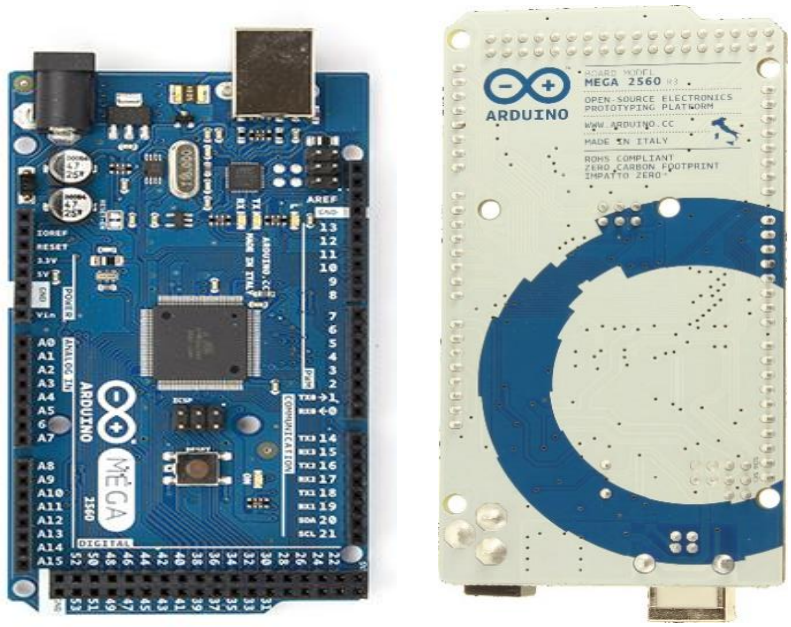

Figura 1. Arduino Mega 2160

2.1.5 Arduino Motor Shield

Para el control de la velocidad y dirección de los motores DC que van a mover la base y las articulaciones del brazo se utilizará los módulos Arduino Motor Shield, mostrados en la Figura 2, los cuales se adaptan con facilidad a la tarjeta Arduino. Algunas de las características de estos puentes h que se encuentran en la página Arduino Motor Shield R3 (s.f.) son:

- Voltaje de operación: 5-12V.
- Controlador del motor: L298P, (controla 2 motores DC o un motor paso a paso).
- Corriente máxima: 2A por canal o 4A máximo con una fuente de alimentación externa.
- Sensor de corriente: 1.65V/A.
- Funciones de frenado del motor.

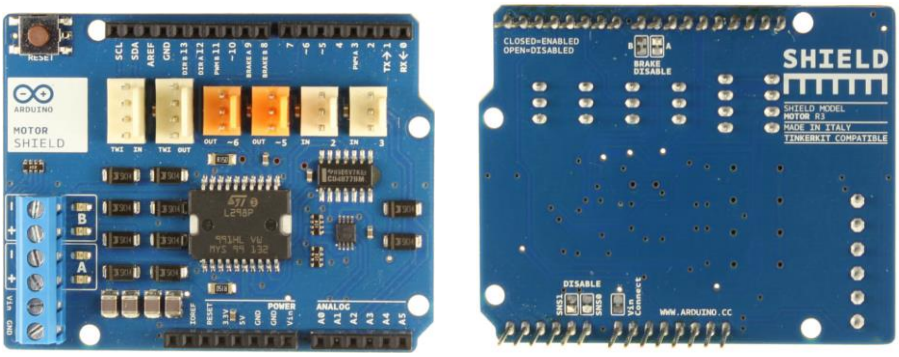

Figura 2. Arduino Motor Shield

2.1.6 Arduino WiFi Shield

El control de forma remota o inalámbrica del brazo se implementa utilizando comunicación WiFi (Fidelidad inalámbrica por sus siglas en inglés), debido al mayor alcance que tiene la señal comparada con la comunicación Bluetooth. Además una conexión WiFi podría permitir el control desde una red de área local o desde cualquier lugar con acceso a internet. El módulo Arduino WiFi Shield de la Figura 3 permite la comunicación directa con la tarjeta Arduino Mega 2560 usando librerías dedicadas de la interfaz de programación, evitando el proceso de programación de los protocolos de comunicación. Algunas de las características de este módulo que se presentan en la página Arduino WiFi Shield (s.f.) son:

- Voltaje de operación: 5V (provienen directamente de la tarjeta Arduino).
- Conexión a redes 802.11b/g.
- Tipos de encriptación soportados: WEP y WPA2 Personal.
- Conexión con Arduino mediante el puerto SPI.
- Ranura para tarjeta micro SD.
- Cabeceras ICSP.

- Conexión FTDI para debugging serial del módulo.
- Puerto mini-USB para actualizar el firmware del módulo WiFi.

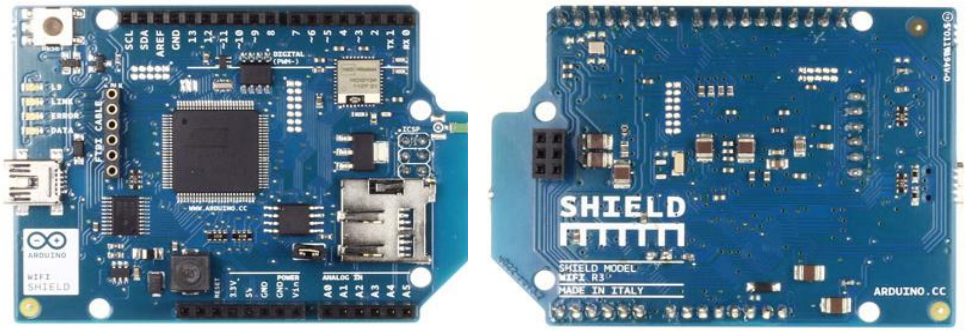

Figura 3. Arduino WiFi Shield

2.1.7 Articulación

Una articulación es la parte de la estructura del robot mediante los cuales se unen los eslabones y permiten un movimiento relativo entre los mismos. Por lo general cada articulación que se aumenta en el robot, incrementa también un grado de libertad en el mismo (Romero, s.f).

El agregar articulaciones puede aportar mayor maniobrabilidad en el robot, pero generalmente también dificulta el control del mismo, y la precisión se suele ver afectada por el error que se acumula. Por lo general los robots industriales modernos tienen seis o menos articulaciones para de este modo poder operar de una forma precisa (Ollero, 2001).

Existen dos tipos de articulaciones, las que se usan más comúnmente en robótica son las de rotación, que proveen al robot un grado de libertad rotacional alrededor del eje de la articulación. Las otras articulaciones usadas son las prismáticas, las cuales permiten realizar un desplazamiento a lo largo del eje de la articulación (Ollero, 2001). Estos dos tipos de articulación se muestran en la Figura 4.

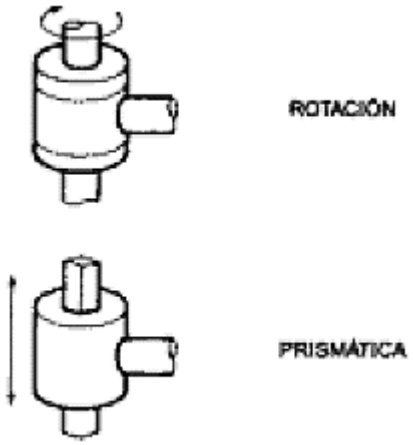

Figura 4. Articulaciones de rotación y prismáticas (Ollero, 2001)

2.1.8 Brazo Robótico

Se puede definir como el conjunto de elementos electromecánicos que propician el movimiento de un elemento terminal (gripper o herramienta), sea para cumplir una función o solo para manipular un objeto (Mena, 2011). La Figura 5 muestra una estructura simple de un brazo robótico.

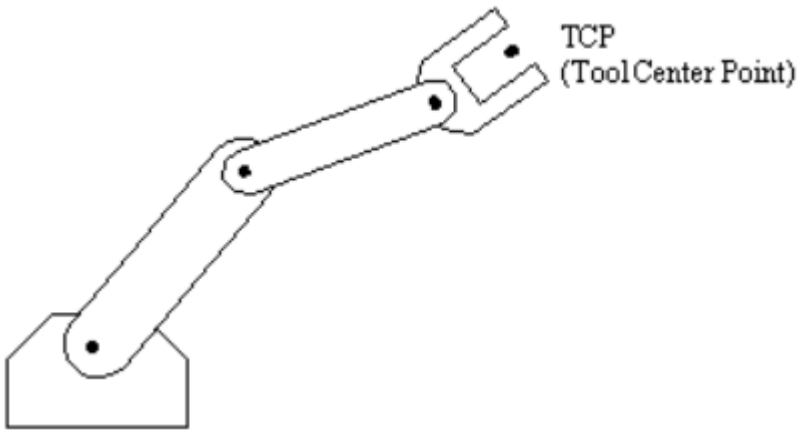

Figura 5. Brazo robótico

El sistema de un brazo robótico está compuesto por una estructura mecánica, transmisiones, actuadores, sensores, elementos finales y un controlador. Su constitución física es similar a la anatomía de los brazos de un ser humano y por lo general se hace referencia a los componentes del robot con los nombres de su parte correspondiente en la extremidad de una persona. Como por ejemplo hombro, codo, brazo, muñeca, etc. (Romero, s.f.).

Existen cuatro configuraciones básicas para un brazo robótico. La primera es la configuración cartesiana, que consta de tres articulaciones prismáticas. La segunda configuración es la cilíndrica, la cual tiene dos articulaciones prismáticas y una de rotación. La configuración polar o esférica está formada por dos articulaciones rotacionales y una prismática. Finalmente, la configuración angular tiene tres articulaciones rotacionales (Ollero, 2001).

2.1.9 Coordenadas generalizadas

Las coordenadas generalizadas son parámetros que describen la configuración de un sistema con respecto a otro usado como referencia (Mason, s.f.). En el estudio de los modelos cinemáticos directo e inverso de un brazo robótico, las coordenadas generalizadas son las coordenadas (x, y, z) con respecto al sistema de referencia por lo general ubicado en la base del robot.

2.1.10 Electromecánica

Es una característica que tienen ciertos componentes, la cual hace referencia a la combinación de la electrónica y la mecánica dentro de un mismo dispositivo (Mena, 2011).

2.1.11 Encoders rotacionales

Los encoders rotacionales son elementos electromecánicos que permiten obtener información de la rotación de los ejes de los motores. Consisten en una fuente de luz LED (diodo emisor de luz por sus siglas en inglés) y un sensor separados por una rueda que contiene un código de orificios a través de los cuales a medida que rota el motor deja que la luz llegue al sensor y permite así determinar cuánto ha rotado el eje del motor (Eitel, 2014).

2.1.12 Eslabón

Son elementos estructurales sólidos que forman parte del sistema mecánico del robot. Para juntar varios eslabones entre sí, es necesario tener articulaciones entre ellos. Los eslabones del robot junto con las articulaciones van a definir el alcance del robot y su espacio de trabajo, y también van a ser factores determinantes de la maniobrabilidad de los mismos (Romero, s.f.).

2.1.13 Espacio de trabajo

Se define el espacio de trabajo de un robot el volumen dentro del cual puede desplazarse su efector final. Este volumen está restringido por el tamaño de los eslabones del brazo robótico y los límites de giro de sus articulaciones. El espacio de trabajo es clasificado como regular o irregular dependiendo de la configuración del brazo robótico. Un espacio regular es característica de las configuraciones cartesiana y cilíndrica, mientras que para una configuración polar, el espacio de trabajo por lo general es irregular (Romero, s.f.).

2.1.14 Grados de libertad

Los grados de libertad son la cantidad de parámetros independientes que determinan la posición del elemento terminal del brazo robótico, el número de grados de libertad por lo general coincide con el número de eslabones de la cadena cinemática (Ollero, 2001).

Para posicionar y orientar un objeto en el espacio de cualquier manera deseada, se necesitan seis parámetros, tres para la posición y tres para la orientación. Es por esto que por lo general los brazos robóticos industriales en su mayoría tienen seis grados de libertad. Cuando el número de grados de libertad del robot excede los necesarios para que cumpla con su tarea se suele decir que un robot es redundante (Romero, s.f.).

2.1.15 Lenguaje de Programación

Es un tipo de lenguaje que sirve para configurar un dispositivo de control para que éste pueda cumplir con un funcionamiento requerido (Mena, 2011).

2.1.16 Matriz de transformación

Es una matriz de dimensión 4x4 que representa la transformación de un vector de un sistema de coordenadas a otro (Castejón, s.f.). En el cálculo del modelo cinemático directo de un brazo robótico, cada articulación tiene su propia matriz de transformación que relaciona su sistema de coordenadas con el sistema de coordenadas de la articulación anterior.

2.1.17 Modelo Cinemático

El modelo cinemático es una metodología que describe las relaciones estáticas entre las posiciones de las articulaciones de un robot, las coordenadas cartesianas y la posición de su actuador final (Stone, 1986).

El modelo cinemático de un robot es usado para su simulación y control. Los modelos tienen como base las transformaciones matriciales compuestas de sistemas de referencia y resultan más complejos a medida que se agregan grados de libertad al robot (Ollero, 2001).

El modelo cinemático directo de un robot es el que permite conocer la posición y orientación final del robot en función de las variables de las articulaciones. El modelo cinemático inverso es el que permite calcular las variables articulares del robot en función de la posición y orientación del efector final deseadas. El modelo cinemático inverso involucra la resolución de sistemas de ecuaciones no lineales que involucran senos y cosenos. Al momento de resolver este sistema de ecuaciones se debe primero determinar si existen soluciones dentro del espacio de trabajo del robot. Existen algunos casos en los que existe más de una solución aceptable (Ollero, 2001).

2.1.18 Motor DC

Es una máquina capaz de convertir energía eléctrica en energía mecánica provocando un movimiento de rotación gracias a la acción de un campo magnético. Debido a la facilidad con la que se pueden controlar son usados en aplicaciones que requieren diferentes velocidades y un control preciso. La alimentación de este motor es de corriente continua (Direct Current en inglés) y es por eso que lleva las siglas DC en su nombre (Fitzgerald, Kingsley & Umans, 2003).

2.1.19 Parámetros de Denavit-Hartenberg

Los parámetros de Denavit-Hartenberg son parámetros específicos de cada robot que dependen de las características geométricas de sus enlaces y articulaciones. Los parámetros de Denavit-Hartenberg son usados para encontrar las matrices de transformación de cada articulación y posteriormente obtener el modelo cinemático directo del brazo robótico (Ramírez, s.f.).

2.1.20 Sensor

Es un dispositivo electrónico que sirve para medir variables físicas en un entorno y convertirlas en señales eléctricas que puedan ser usados por el sistema (Mena, 2011).

2.1.21 Sistema de Control (SC)

Son equipos o dispositivos que en un sistema toman las decisiones a partir de una programación o configuración previamente realizada (Mena, 2011).

2.1.22 Sistemas de Transmisión

Son sistemas conformados por engranes, cremalleras, piñones, etc., y sirven para transmitir movimientos de un sistema a otro, o para convertir ciertos tipos de movimiento en otro (ej. movimiento rotacional en movimiento lineal) (Mena, 2011).

2.2 Revisión literaria

Como nos indica Gloria Martínez (2008), cuando hablamos de un robot con diseño propio se debe considerar que se pueden desarrollar con materiales reciclados y que éstos son diseñados, por lo general desde cero, por los propios programadores. Éstos tienen un bajo costo económico, tanto en adquisición, como en mantenimiento e instalación. Normalmente están diseñados para cumplir un solo tipo de aplicaciones; por lo que su arquitectura puede ser cerrada, es decir destinada a cumplir con cierto tipo específico de funciones y movimientos, por lo tanto no son adaptables. Son veloces para cumplir sus tareas, que como se dijo antes normalmente están definidas. También su ambiente de programación está limitado a ciertos movimientos y adquisición de datos de sus sensores.

Un brazo manipulador o brazo robótico se puede definir como el conjunto de elementos electromecánicos que propician el movimiento de un elemento terminal (gripper o herramienta), sea para cumplir una función o solo para manipular un objeto (Mendoza, 2004). Una especificación general de un brazo robótico comprende: sus grados de libertad, su configuración y su cinemática directa e inversa (Sandoval, 2007).

Marcela Aparicio (2004) indica que la estructura mecánica básica de un brazo robot está compuesta por eslabones, que cumplen una función similar a la de un hueso. También por actuadores, de funcionalidad parecida a la de un músculo. Además poseen sistemas de transmisión, con cierto parecido a los tendones. Y finalmente los cables para envío de la señal, que en cierto modo se los considera los nervios del robot. Los puntos de unión entre eslabones reciben el nombre de nodos y el elemento que permite dicha unión y un movimiento relativo entre ellos, al igual que en el caso de un brazo humano, es la articulación.

Tomando en cuenta que la mayoría del equipo robótico dentro de un área académica y de investigación son de tipo industrial o comercial, como afirman Tzvi (1989) y Schilling (1990), los cuales ya están especificados y caracterizados, y que además en la literatura revisada son pocos los modelos de robots de diseño propio, como el de Pernia (2002), en los cuales se basa más en su control que en su modelado, este trabajo presenta una metodología de diseño, construcción y control de un brazo robótico propio.

En el diseño de Sistemas Empotrados (sistema computacional diseñado para realizar funciones dedicadas), Laplante (1992) divide el proceso en dos elementos conceptuales. Por una parte, se tiene a la arquitectura o el hardware de control diseñado y construido en torno a un sistema basado en microprocesadores. Mientras que por otra parte se tiene a la programación o software que involucra los sistemas operativos, los lenguajes de programación, compiladores, herramientas de modelado, simulación y evaluación (Laplante, 1992; Halang & Sacha, 1992). La integración de la arquitectura hardware y software proporciona la arquitectura global del sistema empotrado (Halang & Sacha, 1992).

Los sistemas empotrados son utilizados en el control de buena parte de aplicaciones, como en los dispositivos electrónicos de consumo (videoconsolas, reproductores de audio/vídeo), en la automoción (control de airbag, climatizador), en la industria (control de procesos, robótica), en las comunicaciones (teléfonos móviles, modem), etc., por lo tanto, así no se aplique directamente en un brazo robótico, el estudio de sistemas empotrados, tiene muchas aplicaciones en la actualidad (Hassan, Soriano, Montagud & Domínguez, 2005).

Una implementación más avanzada y realizada mediante software en C, se desarrolla para analizar la cinemática de este manipulador como solución aproximada mediante gráficos

2D y 3D, para poder verificar el funcionamiento y los movimientos que el sistema puede realizar (Koyuncu & Güzel, 2007).

2.3 Metodología

La metodología a implementarse en la construcción de este sistema de brazo robótico automatizado, consiste en 4 etapas. En primer lugar se encuentra la etapa de diseño, donde utilizando conocimientos teóricos sobre robótica, y la relación entre movimiento y transmisión, se realizará un esquema de la estructura del sistema. En este esquema se definirán los grados de libertad, movimientos requeridos y dimensiones.

La segunda etapa consiste en la simulación del sistema en software especializado y destinado a esclarecer y determinar posibles fallas en el diseño, limitaciones, precisión y capacidad real de movimiento. En base a los resultados obtenidos en las simulaciones se modificará el diseño para que cumpla con los objetivos establecidos.

La tercera etapa consiste en la implementación física del sistema, es decir, el armado de la estructura, montaje de piezas, fijación de sistemas de generación y transmisión de movimiento y pruebas básicas de funcionamiento.

En la última etapa se realiza la automatización del sistema; al programar los comandos necesarios para que el brazo robótico tenga la capacidad de tomar ciertas decisiones de funcionamiento. Esto luego de una interacción con el medio en el que esté ubicado, la cual se realiza a través del uso de sensores, o a través de los comandos directos enviados de forma remota. Estas decisiones deben considerar la capacidad del sistema y así no pretender que éste realice movimientos que se encuentran fuera de su alcance o que sobrepasen sus limitaciones.

CAPÍTULO 3: DISEÑO MECÁNICO DEL SISTEMA

El diseño mecánico del sistema conlleva 3 etapas. Primero, en el diseño básico del sistema se definen características generales del brazo robótico, los grados de libertad, número de eslabones, etc. En la etapa 2 se tiene el diseño digital, tanto 2D como 3D, del sistema. Se realizan las piezas utilizando las medidas exactas para así visualizar la estructura del sistema y las posibles configuraciones. Y en la etapa 3, se realiza la simulación del sistema para visualizar las limitaciones físicas del movimiento de las articulaciones, además de posibles problemas de fricción que puedan existir y así realizar las correcciones necesarias para un correcto funcionamiento del sistema.

3.1 Diseño básico inicial

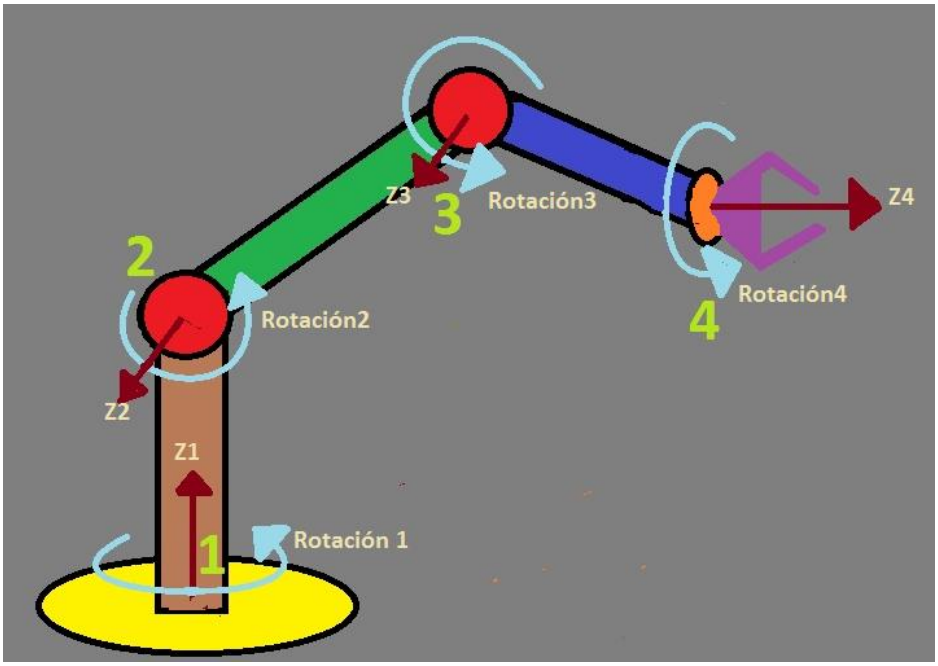

Figura 6. Modelo de un brazo con cuatro grados de libertad

El brazo se diseña para tener cuatro grados de libertad, como se muestra en la Figura 6, lo cual le permite abarcar un amplio espacio de trabajo. Los grados de libertad que dispone el robot son: uno en la base de rotación, otro en la articulación de rotación tipo hombro, un tercero

en la articulación de rotación tipo codo y finalmente una articulación de rotación tipo muñeca que sirve para darle orientación al actuador final, que para motivos de prueba es una pinza que permite manipular objetos.

3.2 Diseño digital del sistema

3.2.1 Presentación del software utilizado

El software escogido para realizar el diseño y la simulación del sistema es Autodesk Inventor, un software especializado en el diseño y simulación mecánica. Con la ayuda de este software se puede validar los diseños antes de que éstos sean creados para ahorrar tiempo y dinero en la construcción de los mismos (3D CAD Software | Inventor 3D CAD | Autodesk, s.f.).

Algunas de las funciones por las cuales se utiliza este software son:

- Diseño sencillo de piezas 2D y 3D.
- Ensamblaje sencillo de piezas en el diseño 3D.
- Facilidad para elegir los materiales de las piezas.
- Simulación del diseño completo en 3D.
- Licencia gratuita para estudiantes.

3.2.2 Diseño realizado

En la base del brazo robótico existen dos cilindros pegados en un extremo a una base plana. En el más grande de los cilindros, que tiene en su parte inferior un hueco de forma cónica,

se coloca el eje de rotación del brazo, y en el otro se coloca el motor que transmitirá el movimiento a través de una banda como se muestra en la Figura 7.

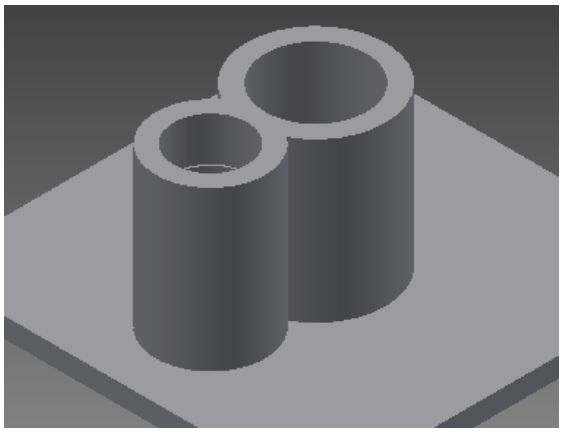

Figura 7. Diseño 3D de la base del brazo robótico

El eje de rotación es un cilindro que en un extremo tiene una punta de forma cónica y en la otra un hueco en forma cúbica el cual permite el movimiento del primer eslabón del brazo robótico y tiene huecos para insertar el eje que sostendrá este eslabón.

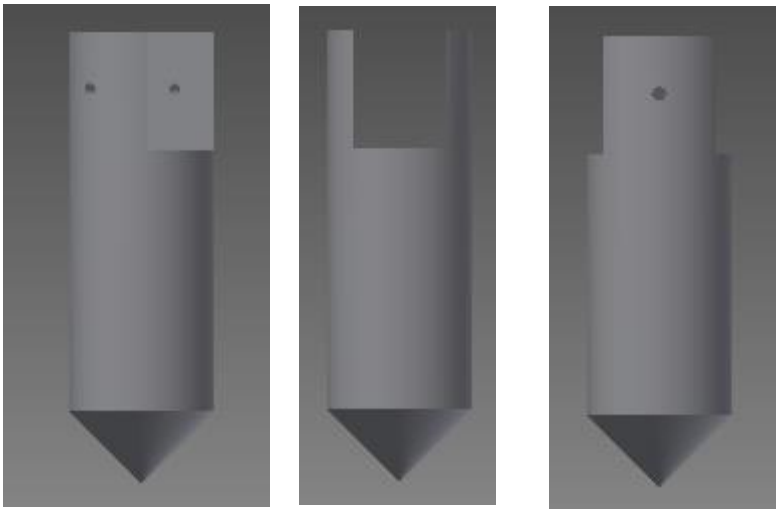

Figura 8. Diseño 3D del eje de rotación del brazo robótico

El primer eslabón del brazo robótico, mostrado en la Figura 9, en su parte central tiene 2 agujeros donde son posicionados los motores y en sus extremos una forma semicircular con

orificios por donde pasan los ejes de rotación. De esta forma utilizando correas para la transmision, se generan los movimientos tanto de este eslabón respecto a la base, como del siguiente eslabón respecto a éste.

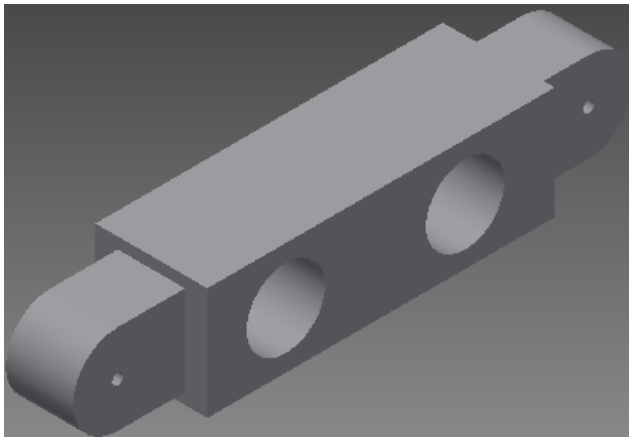

Figura 9. Diseño 3D del primer eslabón del brazo robótico

El segundo eslabón del brazo robótico, mostrado en la Figura 10, permite en sus extremos el ingreso de los otros eslabones centrados en el eje rotacional. La parte central tiene un agujero donde va colocado el motor que permite el movimiento del último eslabón mediante un sistema de transmisión.

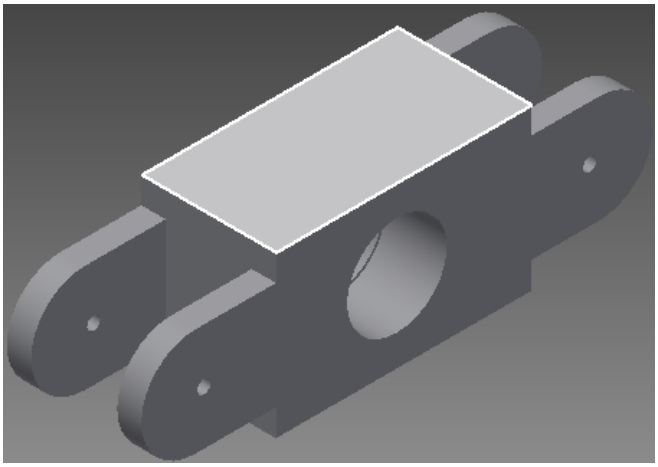

Figura 10. Diseño 3D del segundo eslabón del brazo robótico

El último eslabón tiene un extremo delgado para encajar en el eslabón anterior, centrando su eje y un agujero a través de la pieza, donde va colocado un motor que permite cambiar la orientación del actuador como se muestra en la Figura 11.

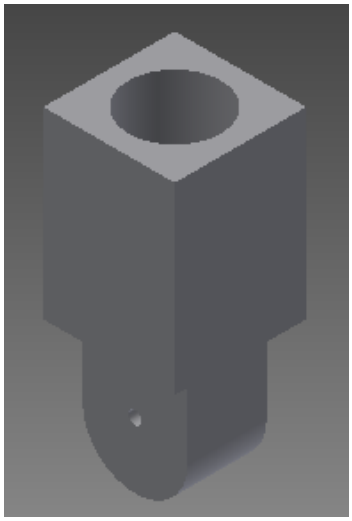

Figura 11. Diseño 3D del último eslabón del brazo robótico

Cabe recalcar que la transmisión está basada en el principio de que los ejes rotacionales se encuentran fijados a un eslabón pero son móviles respecto al siguiente.

3.3 Simulación del sistema

3.3.1 Presentación del simulador

El simulador usado es parte del software Autodesk Inventor presentado anteriormente. Para la simulación se especifica el material de las piezas del brazo, se agregan los ejes de movimiento y se definen las respectivas limitaciones de los movimientos de las piezas. Después de esto se definen los movimientos que se realizan durante la simulación.

3.3.2 Resultados de la simulación

En el ensamble de las piezas para la simulación, mostrado en la Figura 12, se puede observar que estas encajan perfectamente y permiten realizar los movimientos requeridos por el diseño.

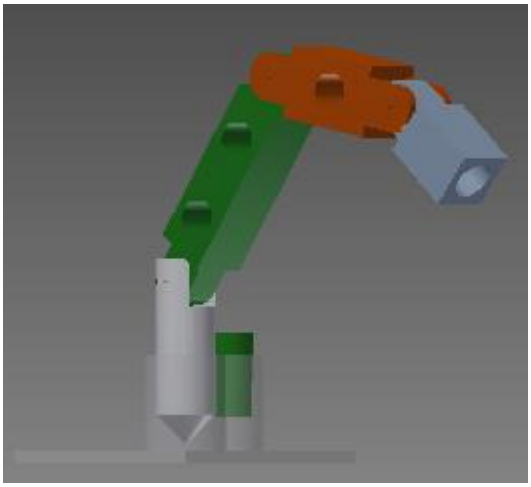

Figura 12. Diseño 3D del brazo robótico acoplado para simulación

Tras la simulación de movimiento se concluye que el diseño cumple con el objetivo requerido de evitar colisiones y permitir una rotación lo más cercana a 360°, sin embargo los cortes requeridos para obtener las piezas diseñadas son complicados, además debido al tamaño y forma de las mismas se podría tener conflicto entre el peso y la capacidad de los motores. El diseño del eje de la base presenta mucha fricción, lo cual generaría un desgaste innecesario en la pieza y problemas en el movimiento de la misma.

3.3.3 Arreglos realizados sobre el diseño

Con los resultados obtenidos en la simulación y posteriormente con las limitaciones de los materiales disponibles, se realizan cambios o correcciones en el diseño de la base del brazo

robótico. Debido a que el eje diseñado estaría expuesto a una gran cantidad de fricción, se cambia la estructura de la base por dos platos separados por ruedas locas que permitan el movimiento de rotación como se muestra en la Figura 13. Uno de los platos se encuentra asentado sobre 4 patas, dando la apariencia de una mesa y el motor que genera la rotación de todo el sistema se encuentra sujeto en medio de estas patas. Sobre esta "mesa" se asienta el resto del sistema, el cual no posee los motores en medio de los eslabones sino que se cambia por un diseño en el cual los motores sirvan a la vez como generadores de movimiento y como ejes de rotación. Estos cambios permiten tener la rotación requerida sin mucha fricción ni desgaste del sistema y también que los eslabones pierdan peso, al centrar el mismo en las articulaciones generando así menos torque. Además, se gana estabilidad y el espacio de trabajo aumenta por debajo de la base ya que el límite inferior no estaría dado por la base rotativa sino que se expande aproximadamente 15cm por las patas colocadas.

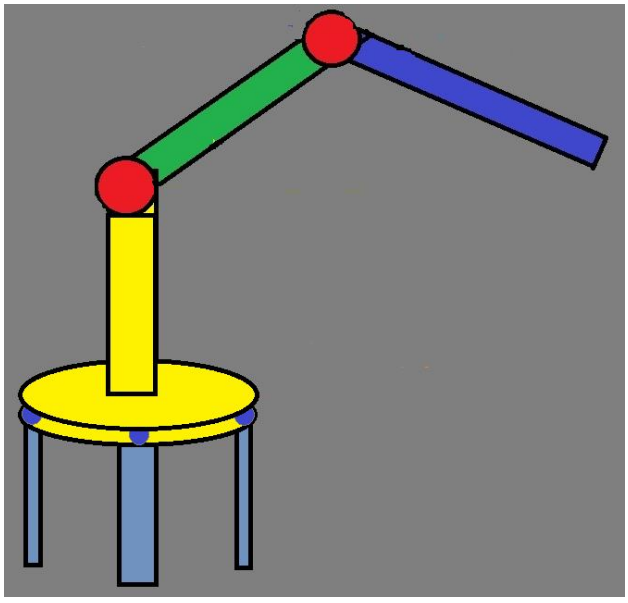

Figura 13. Resultado del cambio de diseño de la estructura

CAPÍTULO 4: HARDWARE

El Hardware se divide en 3 grupos de elementos los cuales se diferencian por la función que cumplen en el sistema, estos grupos son:

1. Material para estructura del sistema (eslabones, base y soporte).
2. Elementos electromecánicos (actuadores) y componentes adjuntos al sistema.
3. Elementos y componentes para el control.

Para elegir cada uno de los componentes se debe realizar un estudio de mercado que discrimina opciones basados en cualidades necesarias, requerimientos básicos de funcionamiento, limitaciones de presupuesto, facilidad de adquisición y limitaciones de existencia o no en el mercado.

4.1 Estudio de mercado

El material para la estructura, es decir lo que hace referencia a la base del sistema y los eslabones, debe ser de un material ligero, maleable y resistente. Las opciones iniciales son aluminio, duralón y grillón, y luego del estudio de mercado se opta por duralón, debido a que cumple de mejor manera los requerimientos ya mencionados.

En lo que se refiere a los componentes electromecánicos se debe elegir motores para generar todos los movimientos, los cuales no deben limitar el rango de actuación del sistema (girar 360º), además deben servir de ejes de rotación de los eslabones (ejes resistentes), ser fuertes para mover el peso necesario (alto torque) y tener un mecanismo para informar sobre la posición en la que se encuentran (mecanismo de control o encoders). Las opciones son motores paso a paso, motores servo y motores dc con encoders, siendo éstos últimos los elegidos. En este grupo también se encuentran los sensores para informar sobre las posiciones límites alcanzadas por el

sistema. En este caso se utilizan finales de carrera y un sensor magnético para la rotación de la base.

Entre las opciones contempladas para el control del brazo se considera la utilización de PIC (Controlador de Interfaz Periférico), FPGA (Arreglo de Compuertas Programables en Campo), y la plataforma Arduino. Los microcontroladores son descartados ya que a pesar de ser una de las alternativas más económicas es también una de las más delicadas para usar, debido a que para cada programa nuevo se debe desconectar el microcontrolador del circuito del brazo, lo cual puede ocasionar algún daño en sus conectores.

El caso de los FPGA es diferente, estos son versátiles y no es necesario desconectar los circuitos para poder programarlos, la desventaja con estos dispositivos es el costo ya que son dispositivos que se usan para aplicaciones más complejas y que requieran altas velocidades de procesamiento, lo cual no es tan esencial en este caso.

Arduino es la opción escogida debido a que es de bajo costo, tiene gran versatilidad y es de fácil uso. Arduino es una plataforma de computación del tipo Open-Source basada en un microcontrolador y una interfaz de desarrollo para escribir software para el dispositivo (Arduino – Introduction, s.f.). Arduino es seleccionada en este caso por sobre otras plataformas similares debido a que puede trabajar sobre cualquier sistema operativo de computación, su programación es basada en el lenguaje C++, y su hardware puede ser modificado o mejorado a través de módulos de bajo costo. Este último aspecto es relevante, ya que para el control de los motores elegidos se requiere de circuitos puente H, lo cual se facilita con esta opción de control que tiene Arduino.

4.1.1 Lista de materiales y descripción del funcionamiento en el sistema

MATERIALES	DESCRIPCIÓN
Arduino	Componente electrónico que controla todo el sistema.
WiFi shield	Accesorio de Arduino para conectar el controlador al internet.
Puente H	Componente que permite un fácil control de los motores.
Motores DC con encoders	Elemento electromecánico que genera todos los movimientos en el sistema, sus encoders incorporados permiten conocer su posición.
Planchas de duralón	Se cortan formas específicas para formar la estructura o esqueleto de todo el sistema: 2 platos para rotación, 4 patas para la base, 3 eslabones del brazo.
Pinza y servomotor	La pinza es el actuador final del sistema y se acciona con un servo.
Componentes eléctricos varios	Borneras, finales de carrera, sensor magnético de presencia, cables, leds y compuerta lógica NOT.

Tabla 3. Lista de materiales

La lista detallada de materiales y precios se encuentran en el Anexo A.

4.2 Construcción del sistema

Una vez elegidos todos los componentes, materiales y dispositivos a ser utilizados, se realiza la construcción del sistema siguiendo el procedimiento que se detalla a continuación.

1. Corte y construcción de piezas para la estructura.
2. Ensamblaje de estructura o esqueleto del sistema.
3. Adaptación de componentes varios.

4. Conexión entre sistema de control y estructura ensamblada.

4.2.1 Corte y Construcción de piezas para estructura

Como se indica en las secciones anteriores, para la estructura se trabaja con duralón, un material plástico resistente y maleable. El duralón se adquiere en planchas, en el taller con las herramientas adecuadas se procede a cortar piezas de acuerdo al diseño realizado: 4 piezas de 25cm x 5cm para los eslabones, 4 piezas de 12.5cm x 5cm para las patas y 2 platos de 15cm de diámetro como se muestra en la Figura 14. Además, con mucha precisión se taladran los huecos en los lugares donde van fijos los motores y los acoples; 2 y 3 orificios respectivamente.

Figura 14. Corte de piezas de la estructura

4.2.2 Ensamblaje de la estructura

Después de haber realizado los cortes y perforaciones necesarias se procede a unir todas las piezas utilizando los motores como ejes de rotación. Debido a la corta longitud del eje de los motores, se procede a limar el material para permitir el paso del motor para de esta forma tener una longitud adecuada de los ejes y en consecuencia mejorar el soporte en las articulaciones.

Figura 15. Ensamblaje de la estructura

Los acoples utilizados poseen perforaciones laterales por donde entran los prisioneros que vienen incluidos para generar una especie de agarre con el eje del motor. Debido a los movimientos y vibraciones estos prisioneros se desajustan generando movimientos no adecuados del sistema. Por lo tanto, como solución, se procede a colocar masilla epoxi en estas uniones generando una mejor fijación y estabilidad. La estructura una vez ensamblada se muestra en la Figura 15.

4.2.3 Conexión entre estructura y sistemas de control

Una vez que se tiene la estructura lista y los motores fijos al sistema, se procede a colocar en el sistema los elementos electrónicos que dan un cierto tipo de alarma o control, como son los finales de carrera. Éstos se colocan de tal forma que son activados cada vez que uno de los eslabones llega a un límite físico o a algún límite que por cuestiones de diseño no se quiera alcanzar; como es el caso del tercer eslabón, éste tiene la capacidad mecánica de girar cerca de 300 grados pero no se necesita que llegue más allá de una posición completamente vertical (90º respecto a la base).

Se coloca el sensor magnético entre los 2 platos de la base rotativa para así evitar que el sistema avance más de una vuelta, es decir que su rango de actuación sea entre 0º y 360º, ya que el hecho de girar más, llevaría a una desconexión del sistema o ruptura de cables. Los cuales para permitir la rotación del sistema van colocados entre los 2 platos pasando por perforaciones hechas en los mismos.

Una vez armado y cableado el sistema mecánico, se diseña un panel de control en el cual van colocados el controlador principal (Arduino), el módulo WiFi, los puentes H, el sistema de control manual y todas las borneras necesarias sobre un circuito sencillo para conectar este panel con el sistema mecánico.

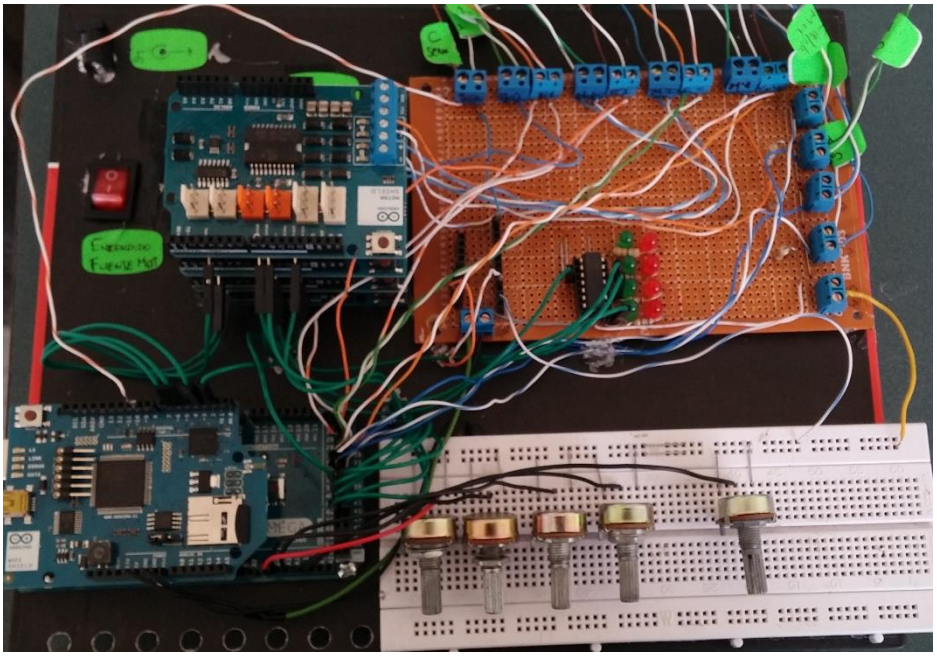

Figura 16. Circuitos y sistemas de control

Como se muestra en la Figura 16, el módulo WiFi se coloca sobre la placa del Arduino. Los cuatro puentes H se conectan sobrepuestos para así tener la misma alimentación y poder usar una sola fuente de 5V y un solo cable de tierra desde el Arduino. El control manual consiste

en 5 potenciómetros, uno para el control de apertura y cierre de la pinza, y los otros cuatro controlan cada uno de los motores. La lógica utilizada por estos potenciómetros utiliza su posición central como su etapa sin movimiento, y sus giros para uno u otro lado determinan la dirección de rotación del motor, además la cantidad que se gira el potenciómetro controla la señal PWM que se envía a cada motor controlando de esta manera su velocidad.

4.2.4 Primeras pruebas y rediseño de la estructura

Luego de varias pruebas realizadas con el sistema mostrado en la Figura 17, se obtiene como resultado que la estructura sólida de los eslabones diseñados es muy pesada para los motores disponibles, por lo tanto, en un comienzo se trata de modificar las piezas sólidas y reemplazarlas por dos placas paralelas.

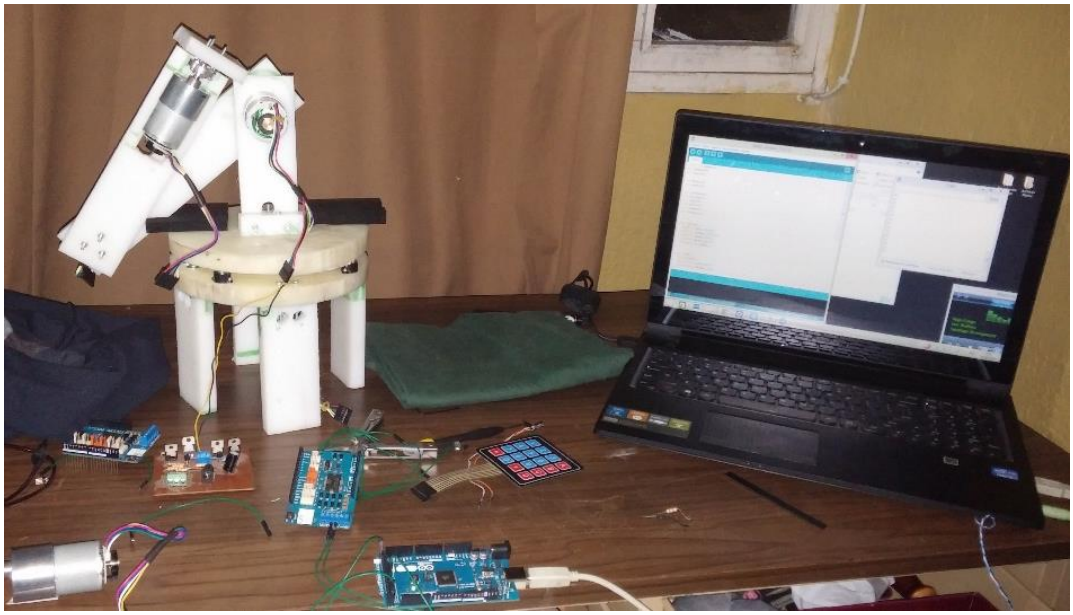

Figura 17. Pruebas del brazo

A pesar de la mejora en el diseño, las piezas siguen siendo pesadas y se trata de reducir a una sola placa, con lo cual los motores son capaces de realizar movimientos pero al costo de mucha potencia, se requiere alimentar con más del voltaje permitido por los motores lo cual es

un riesgo para todo el sistema. Se opta por utilizar un diseño basado en contra pesos, es decir, a los eslabones utilizados se los modifica de tal manera que los ejes de rotación o articulaciones queden en la parte central de los eslabones y se coloca pesos en los extremos opuestos a donde va fijado el siguiente eslabón como se muestra en la Figura 18. Con esta modificación se disminuye el consumo de energía de los motores y existen movimientos mucho más controlables y suaves.

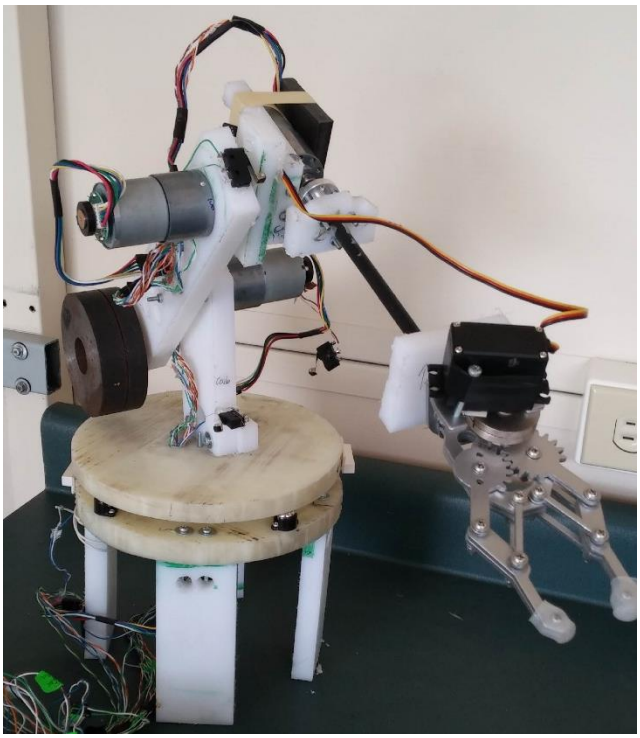

Figura 18. Versión final del brazo robótico

CAPÍTULO 5: SOFTWARE DEL SISTEMA

El capítulo de software abarca todo lo que hace referencia a la parte del control del sistema. Incluye una comparación entre las opciones de controladores considerados y el funcionamiento, características y comandos de programación utilizados en el elemento programable elegido.

5.1 Descripción de software de programación

Selección del elemento de control

Para programar la tarjeta de Arduino se utiliza el software Arduino IDE, siendo éste una herramienta para escribir y cargar a la tarjeta el software de forma sencilla. Arduino IDE es software escrito en Java, lo cual permite exportar más fácilmente a otras plataformas y es software tipo open-source (Arduino – Software, s.f.).

Arduino IDE ofrece la ventaja de utilizar un lenguaje de programación similar al C++, incorporando librerías dedicadas para los módulos y el control de la tarjeta Arduino utilizada. Además, incluye los drivers necesarios para la comunicación con la tarjeta, emulando un puerto serial virtual.

La programación de control remoto vía Web se implementa utilizando los lenguajes HTML5 para el contenido básico de las páginas, Javascript para validación de datos, AJAX para las funciones asíncronas de entrada y salida, y CSS3 para el estilo visual de la página.

En el desarrollo de estas páginas web se utiliza el software Notepad++, este software es de tipo open-source y es un editor de código que soporta varios lenguajes de programación (Notepad++ Home, s.f.), incluyendo los usados en el proyecto como HTML5, Javascript y CSS3.

5.1.1 Funciones utilizadas

WiFi.h

`WiFi.h` es una librería que permite a una tarjeta Arduino conectarse a internet a través de un módulo WiFi. Haciendo uso de esta librería, se puede usar a la tarjeta Arduino como servidor o como cliente dentro de una red (WiFi Library, s.f.).

Dentro de la clase WiFi de esta librería, las funciones usadas para la comunicación y control del brazo fueron:

- `status()`: permite verificar que el módulo WiFi se encuentre conectado.
- `begin(ssid,pass)`: establece la conexión entre Arduino y una red. Los parámetros `ssid` y `pass` son del tipo `char` y son el nombre de la red y la clave respectivamente.

  Dentro de la clase IP se hace uso de la función:
- `localIP()`: que devuelve la dirección IP asignada al Arduino por el router.

  De la clase Server, se utiliza:
- `WiFiServer(port)`: que inicia un servidor que escucha a conexiones entrantes por el puerto especificado.
- `available()`: establece la conexión entre el servidor creado y un cliente.

  De la clase Client:
- `WiFiClient()`: crea un cliente que puede conectarse a una IP y puerto definido.

SPI.h

La librería `SPI.h` permite acceder a la interfaz SPI (Serial Peripheral Interface), la cual se usa para comunicar un microprocesador con dispositivos periféricos en una corta distancia

(SPI Library, s.f.). El tipo de comunicación que se usa con esta interfaz es el de maestro-esclavo. El módulo Arduino WiFi Shield hace uso de la interfaz SPI.

SD.h

La librería `SD.h` de Arduino permite el leer y escribir archivos desde una tarjeta SD. La librería soporta tarjetas con el filesystem FAT16 y FAT32 y utiliza nombres del tipo 8.3 (8 caracteres para el nombre del archivo y 3 para la extensión). La librería también hace uso de la interfaz SPI (SD Library, s.f.).

Dentro de la clase SD, se usan las siguientes funciones:

- `begin(pin)`: inicializa la tarjeta SD e indica el pin dedicado para la comunicación, en el caso de Arduino Mega es el pin 4.
- `exists(filename)`: comprueba que exista en la tarjeta SD un archivo con el nombre especificado por `filename`.
- `open(filename)`: abre el archivo `filename` de la tarjeta SD.

Math.h

Dentro de la librería `math.h` se encuentran todas las funciones trigonométricas necesarias para los cálculos realizados con respecto al modelo cinemático directo e inverso.

Cabe recalcar que tanto para la programación en Arduino como en la simulación en Matlab, se requiere utilizar la función arco tangente 2 (`atan2`) en lugar del arco tangente simple, debido a que esta última no discrimina el cuadrante donde se ubica la coordenada, lo cual lleva a errores como por ejemplo si la coordenada Y es cero, el ángulo θ_1 siempre es cero sin importar si X es positivo ($\theta_1 = 0°$) o negativo ($\theta_1 = 180°$).

5.2 Tipos de control implementados

5.2.1 Control Manual

Como se mencionó en el capítulo anterior, el control manual definitivo es implementado mediante un teclado digital. Primero se utiliza un teclado matricial, pero debido al exceso de botones y de recursos utilizados para leer un teclado de este tipo, se procede a diseñar un teclado simple de 5 botones, uno por cada actuador (4 motores) y un switch de 2 posiciones para elegir la dirección de rotación. Además, se coloca un potenciómetro el cual está directamente relacionado con la señal PWM que se envía a los motores para controlar la velocidad de rotación, exceptuando el caso de la pinza, donde el potenciómetro determina la posición de apertura de la misma.

5.2.2 Control web de motores de forma independiente

El control de motores independientemente vía web se puede dividir en dos partes, la primera es la interfaz web mediante la cual el usuario envía instrucciones al brazo robótico y la segunda es la programación de la tarjeta Arduino que controla los actuadores.

La interfaz web asimismo puede ser dividida en tres segmentos diferentes. En primer lugar se encuentran las etiquetas HTML5 (lenguaje de marcado de hipertexto por sus siglas en inglés), mediante las cuales se define el contenido de la página web. Después se encuentra el Javascript que son funciones que realiza el navegador en el cual se muestre la página web. Finalmente se encuentra una parte de código CSS3, cuyo único objetivo es cambiar la apariencia visual de los elementos HTML5, razón por la cual no se entrará mucho en detalle con relación a este segmento.

En lo que respecta a las etiquetas HTML5, la parte relevante al control es un formulario que usa el método "`get`" con cinco elementos, uno por cada motor o servo del brazo. En el caso de los primeros cuatro motores, se usa etiquetas de entrada `<input>` del tipo "`range`", como se muestra en la Figura 19. Estas etiquetas son representadas como escalas móviles cuyo rango está definido en los atributos "min" o valor mínimo, "`max`" o valor máximo, y "`step`" o paso. Adicionalmente, al costado de cada entrada del tipo "`range`", se coloca un cuadro de texto en el cual se muestra el valor de la posición de la escala móvil. En el caso del último motor, que es el que controla la pinza, la etiqueta de entrada es del tipo "`checkbox`", representada por una casilla de verificación, la cual se activa por el usuario cuando se requiera que la pinza cambie de estado.

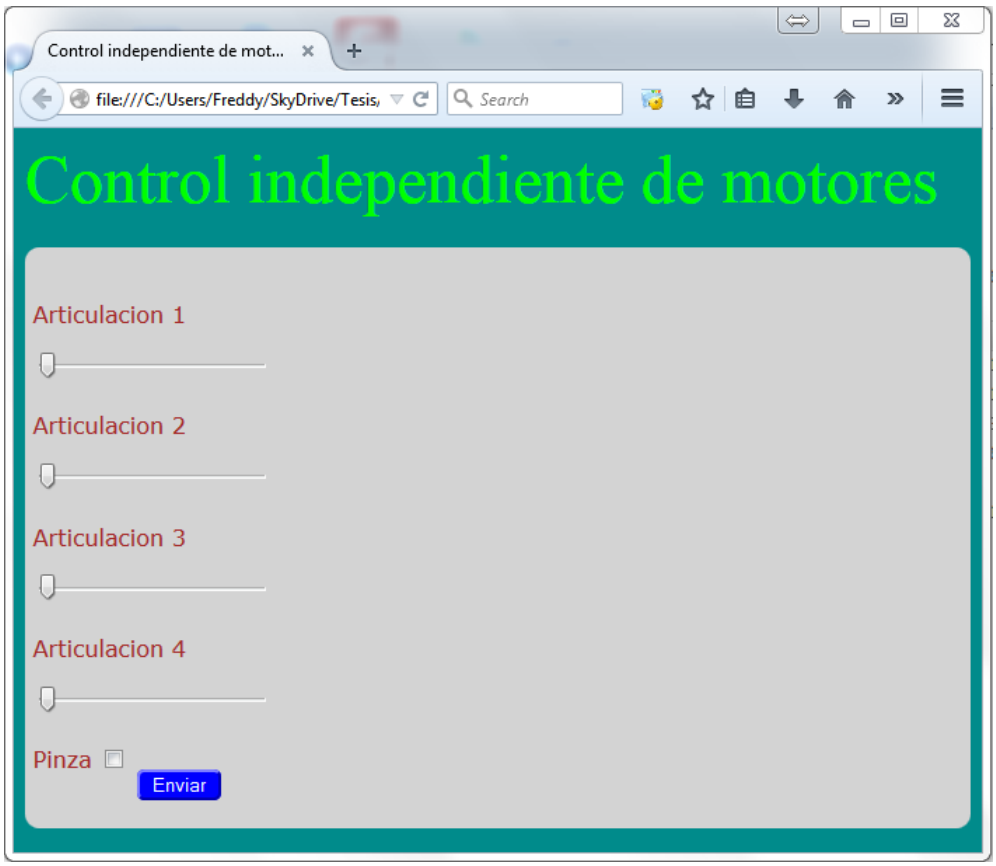

Figura 19. Interfaz web para control independiente de motores

En la parte de Javascript, se crea una función llamada `updateTextInput()`, cuyo propósito es mostrar los valores de las diferentes escalas móviles a medida que el usuario las manipula. La función es llamada dentro de las etiquetas `<input>`, mediante el atributo "`onchange`" el cual envía a la función dos elementos, el valor actual en la escala móvil y el identificador de a cuál motor corresponde. Con esta información se actualiza el contenido de los cuadros de texto que se encuentran al costado de cada escala.

En lo que respecta al programa de Arduino, lo primero a resaltar son las librerías `SPI.h`, `WiFi.h`, `SD.h` y `Servo.h` que se incluyen para tener acceso a los módulos adicionales a la tarjeta Arduino. Después de incluir las librerías mencionadas, la siguiente parte del programa es la declaración de variables que van a ser usadas, entre ellas las que contienen los parámetros de conexión a una red WiFi, las que permiten a la tarjeta Arduino funcionar como servidor, las que permiten abrir el archivo almacenado en la tarjeta SD, y las variables propias del programa que incluyen el control de los actuadores del brazo.

La inicialización del programa contiene la especificación de la comunicación serial que ayuda en la depuración de errores, la especificación de los pines que son usados como entradas y salidas, el establecimiento de la conexión a la red y la inicialización del servidor. Finalmente, en la inicialización se encuentra la calibración, que envía a cada actuador del brazo a una posición inicial y restablece los contadores usados con los encoders.

En el lazo principal de ejecución se encuentran los contadores asociados con cada encoder para conocer la posición de los actuadores del brazo, el envío de la información al cliente WiFi, y el llamado a una función que procesa los pedidos de movimiento del brazo.

Finalmente, se encuentran funciones específicas para el movimiento de cada motor que toman como parámetros de entrada la posición deseada mediante el procesamiento de los pedidos HTTP y se encargan de ubicar al motor haciendo uso de los encoders.

5.2.3 Control web con coordenadas generalizadas

Al igual que en el control de motores de forma independiente mediante la interfaz web, el control usando coordenadas generalizadas consta de dos secciones, la interfaz web y la programación de la tarjeta Arduino. La interfaz web a su vez tiene tres partes, el contenido HTML5, la programación Javascript y la parte visual mediante CSS3.

El contenido HTML5 de la página web es un formulario con el método "`get`" que, como se muestra en la Figura 20, consta de cuatro etiquetas `<input>` del tipo "`number`", las cuales se muestran como un cuadro de texto que únicamente permite ingresar números. Estos números son las coordenadas en X, Y, Z y la orientación del actuador final. Además al igual que en el caso anterior, hay otro elemento `<input>` del tipo "`checkbox`" que permite enviar la señal a la pinza.

La parte de Javascript consta de una función llamada `validateForm()`, cuyo propósito es asegurarse que el usuario envíe únicamente valores de coordenadas válidos para el movimiento del brazo. Al momento de enviar la información, se procede a validar los valores, esto se encuentra especificado en la etiqueta `<form>` en el atributo "`onsubmit`".

Figura 20. Interfaz web para control usando coordenadas xyz

En caso de que el valor no sea válido se mostrará un diálogo indicando el error, esto se logra mediante el comando `alert("")`. El programa del Arduino para este control es muy similar al anterior, la diferencia se encuentra únicamente en el procesamiento de los pedidos HTTP, ya que estos nos van a especificar las coordenadas generalizadas. Es por esto que se debe incluir el procesamiento matemático mediante el cual se implementa el modelo cinemático inverso y se calcula la posición deseada de cada motor, tras lo cual se llama a las funciones mencionadas en el control anterior.

CAPÍTULO 6: MODELO CINEMÁTICO

Una parte esencial para controlar una estructura robótica, es su modelo cinemático, el cual sirve para conocer la relación entre la rotación de los actuadores y la posición y orientación del actuador final. En este capítulo se determina tanto el modelo cinemático directo como el inverso, los cuales sirven para el control del sistema y para comprobar su correcto funcionamiento.

6.1 Modelo cinemático directo

Para calcular el modelo cinemático directo, que nos ayuda a determinar el espacio de trabajo del brazo, se obtienen los parámetros de Denavit-Hartenberg. Los parámetros de Denavit-Hartenberg son propios de cada brazo robótico y dependen de la geometría y construcción del mismo. En la Tabla 4 se presentan los parámetros para el brazo robótico presentado en este proyecto y cabe recalcar que existe una fila por cada grado de libertad del sistema.

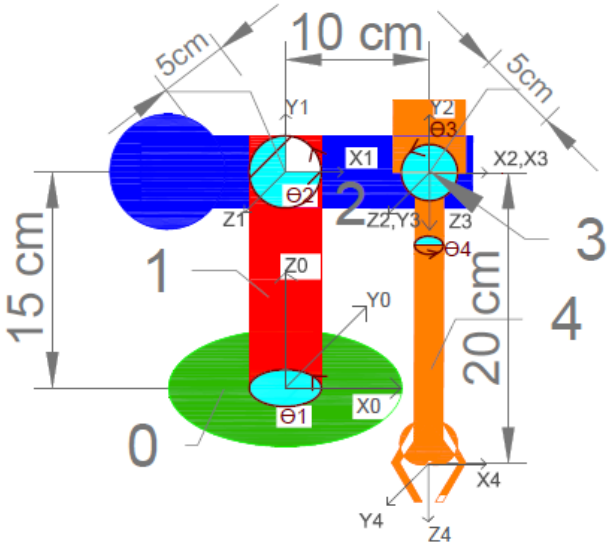

Figura 21. Asignación de ejes de referencia de acuerdo a la convención Denavit-Hartenberg

En la Figura 21 se muestran los ejes x_i, y_i y z_i asignados a cada articulación del brazo teniendo en cuenta la convención de Denavit-Hartenberg. El procedimiento para la asignación de estos ejes, de acuerdo a Paul (1981), se detalla a continuación.

1. Cada manipulador consiste en n eslabones unidos con n articulaciones. Se establece que la base del manipulador es el eslabón 0 y que el eslabón 1 se encuentra acoplado a la base a través de la articulación 1. El efector final no posee una articulación.
2. Se establece un sistema de coordenadas a cada eslabón del manipulador.
3. El origen del sistema de coordenadas para el eslabón n se fija en la intersección de la normal común entre los ejes de las articulaciones n y $n+1$ y el eje de la articulación $n+1$. En caso de que los ejes de las articulaciones se intersequen, el origen del sistema de coordenadas se fija en la intersección de los mismos.
4. El eje z del eslabón n se alinea con el eje de la articulación $n+1$.
5. El eje x se alinea con cualquier normal común que exista cuya dirección sea igual a la normal que va desde la articulación n hacia la articulación $n+1$. En caso de articulaciones que se intersecan, el eje x es paralelo o antiparalelo al producto vectorial $z_{n-1} \times z_n$.

Una vez asignados los ejes de rotación, se construye una tabla de parámetros de Denavit-Hartenberg, la forma de obtener estos parámetros de acuerdo con Paul (1981) es la siguiente:

- i, es el número de eslabón del brazo del cual se están obteniendo los parámetros.
- α_n es la rotación alrededor del eje x_n.
- a_n es la traslación a lo largo de $x_{n-1} = x_n$.

- $\theta_n(°)$ es la rotación alrededor del eje z_{n-1}.
- d_n es la traslación a lo largo del eje z_{n-1}.

Después de completar la tabla de Denavit-Hartenberg, se utilizan estos parámetros para encontrar las matrices de transformación por cada grado de libertad. El modelo cinemático completo se obtiene al multiplicar todas las matrices de transformación.

6.1.1 Parámetros de Denavit-Hartenberg

i	$\alpha_i(°)$	$a_i(cm)$	$\theta_i(°)$	$d_i(cm)$
1	90°	0	θ_1	15
2	0°	10	θ_2	5
3	90°	0	θ_3	-5
4	0°	0	θ_4	20

Tabla 4. Parámetros Denavit-Hartenberg

6.1.2 Matrices de transformación

En un brazo robótico, se puede considerar cada eslabón como una transformación de un sistema de coordenadas ubicado al comienzo del eslabón a otro sistema de coordenadas al final del mismo. La Ecuación 1, presentada por Paul (1981) muestra la matriz de transformación en función de los parámetros de Denavit-Hartenberg.

$$ {}^{0}_{1}\boldsymbol{T} = \begin{bmatrix} \boldsymbol{\cos\theta_1} & \boldsymbol{-\sin\theta_1\cos\alpha_1} & \boldsymbol{\sin\theta_1\sin\alpha_1} & \boldsymbol{a_1\cos\theta_1} \\ \boldsymbol{\sin\theta_1} & \boldsymbol{\cos\theta_1\cos\alpha_1} & \boldsymbol{-\cos\theta_1\sin\alpha_1} & \boldsymbol{a_1\sin\theta_1} \\ \boldsymbol{0} & \boldsymbol{\sin\alpha_1} & \boldsymbol{\cos\alpha_1} & \boldsymbol{d_n} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} $$

Ecuación 1

Esta matriz de transformación a su vez puede ser dividida como se muestra en la Ecuación 2, donde la matriz R (3x3) representa la rotación de la transformación, y la matriz T (3x1) muestra la traslación. Las Ecuaciones 3-6 son las matrices de transformación de cada eslabón en base a los parámetros de Denavit-Hartenberg mostrados en la Tabla 4.

$$ {}^{0}_{1}T = \begin{matrix} R & T \\ 0 & 1 \end{matrix} $$

Ecuación 2

$$ {}^{0}_{1}\boldsymbol{T} = \begin{bmatrix} \boldsymbol{\cos\theta_1} & \boldsymbol{0} & \boldsymbol{\sin\theta_1} & \boldsymbol{0} \\ \boldsymbol{\sin\theta_1} & \boldsymbol{0} & \boldsymbol{-\cos\theta_1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{15} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} $$

Ecuación 3

$$ {}^{1}_{2}\boldsymbol{T} = \begin{bmatrix} \boldsymbol{\cos\theta_2} & \boldsymbol{-\sin\theta_2} & \boldsymbol{0} & \boldsymbol{10\cos\theta_2} \\ \boldsymbol{\sin\theta_2} & \boldsymbol{\cos\theta_2} & \boldsymbol{0} & \boldsymbol{10\sin\theta_2} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{5} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} $$

Ecuación 4

$$ {}^{2}_{3}\boldsymbol{T} = \begin{bmatrix} \boldsymbol{\cos\theta_3} & \boldsymbol{0} & \boldsymbol{\sin\theta_3} & \boldsymbol{0} \\ \boldsymbol{\sin\theta_3} & \boldsymbol{0} & \boldsymbol{-\cos\theta_3} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{-5} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} $$

Ecuación 5

$$ {}^{3}_{4}\boldsymbol{T} = \begin{bmatrix} \boldsymbol{\cos\theta_4} & \boldsymbol{-\sin\theta_4} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\sin\theta_4} & \boldsymbol{\cos\theta_4} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{20} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} $$

Ecuación 6

Una vez obtenidas las matrices de transformación de cada eslabón, el modelo cinemático directo se obtiene al multiplicar todas las matrices ${}^{0}_{4}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T * {}^{3}_{4}T$. El resultado final se presenta en la Ecuación 7 y la relación directa entre los ángulos θ_1, θ_2, θ_3 y θ_4 con las coordenadas (x, y, z) se muestra en la Ecuación 8.

$$ {}^{0}_{4}T = \begin{bmatrix} a & d & g & j \\ b & e & h & k \\ c & f & i & l \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Ecuación 7

donde,

$$ a = \sin\theta_1 \sin\theta_4 + \cos\theta_1 \cos\theta_4 \cos(\theta_2 + \theta_3) $$

$$ b = -\cos\theta_1 \sin\theta_4 + \sin\theta_1 \cos\theta_4 \cos(\theta_2 + \theta_3) $$

$$ c = \cos\theta_4 \sin(\theta_2 + \theta_3) $$

$$ d = \sin\theta_1 \cos\theta_4 - \cos\theta_1 \sin\theta_4 \cos(\theta_2 + \theta_3) $$

$$ e = -\cos\theta_1 \cos\theta_4 - \sin\theta_1 \sin\theta_4 \cos(\theta_2 + \theta_3) $$

$$ f = -\sin\theta_4 \sin(\theta_2 + \theta_3) $$

$$ g = \cos\theta_1 \sin(\theta_2 + \theta_3) $$

$$ h = \sin\theta_1 \sin(\theta_2 + \theta_3) $$

$$ i = -\cos(\theta_2 + \theta_3) $$

$$ j = 10\cos\theta_1 [\cos\theta_2 + 2\sin(\theta_2 + \theta_3)] $$

$$ k = 10\sin\theta_1 [\cos\theta_2 + 2\sin(\theta_2 + \theta_3)] $$

$$ l = 10\sin\theta_2 - 20\cos(\theta_2 + \theta_3) + 15 $$

$$\begin{cases} x = 10\cos\theta_1\,[\cos\theta_2 + 2\sin(\theta_2 + \theta_3)] \\ y = 10\sin\theta_1\,[\cos\theta_2 + 2\sin(\theta_2 + \theta_3)] \\ z = 10\sin\theta_2 - 20\cos(\theta_2 + \theta_3) + 15 \end{cases}$$ *Ecuación 8*

6.2 Modelo cinemático inverso

Existen varios métodos para obtener el modelo cinemático inverso; se puede utilizar la representación anterior del modelo directo y partir de estas matrices para obtener las pseudo inversas y obtener el modelo cinemático inverso. Éste podría considerarse uno de los métodos más precisos pero requiere de un procesamiento matemático muy complejo, el cual posiblemente no soporta Arduino y en caso de que lo hiciera, requeriría de muchos recursos y de bastante tiempo para procesar. Por lo tanto, después de realizar ciertas simplificaciones se procede a implementar un método geométrico para el cálculo inverso requerido.

Las simplificaciones consideradas son las siguientes:

- El motor 4, solo afecta a la orientación del efector final, por lo cual se lo puede obviar para el cálculo del modelo cinemático inverso, ya que en este modelo solo se trabaja con la posición del actuador final al procesar las coordenadas (x, y, z), mas no su orientación.
- El ángulo para el motor 1, puede obtenerse al sacar el arcotangente entre las coordenadas (x, y).

De esta forma se reduce el cálculo de 4 incógnitas a prácticamente 2 incógnitas, θ_2 y θ_3.

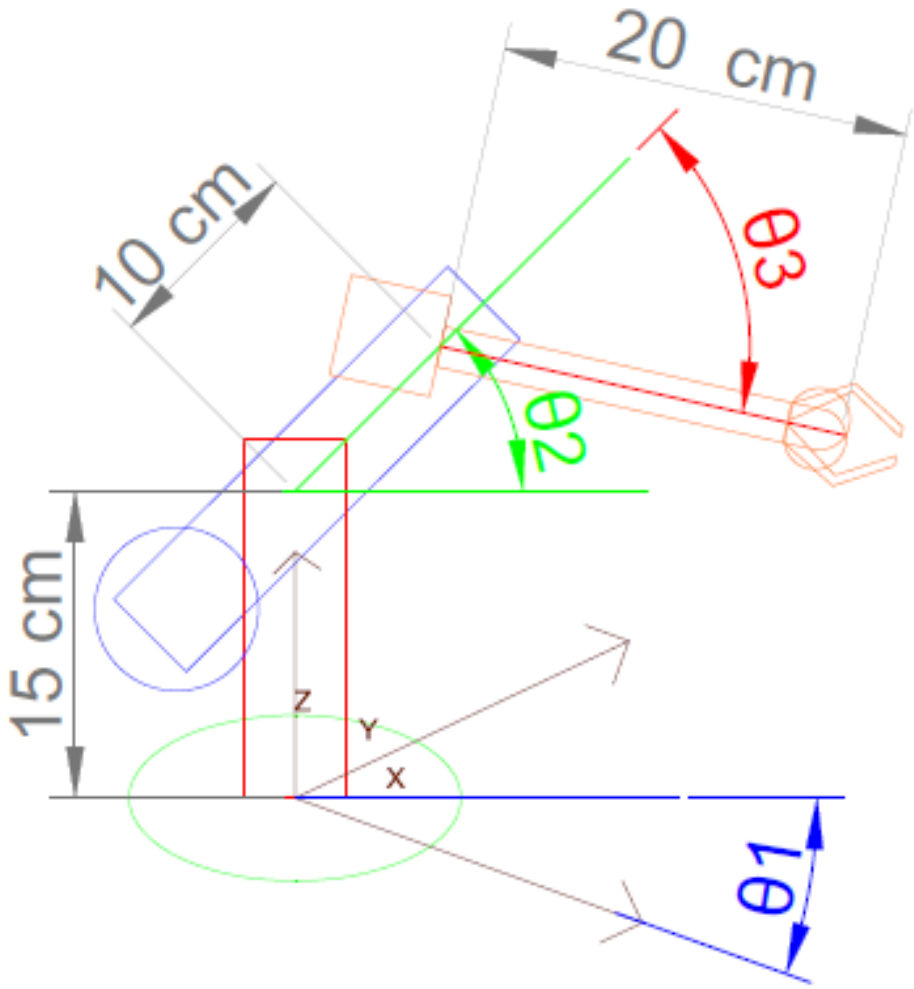

Figura 22. Variables para el cálculo de modelo inverso

Como se muestra en la Figura 22, las incógnitas que se deben hallar son θ_2, que es el ángulo entre el plano horizontal y el eslabón 2, y θ_3 que es el ángulo entre la recta a lo largo del eslabón 2 y el eslabón 3. El ángulo θ_1 se obtiene calculando el arco tangente de y sobre x.

Para el cálculo de los ángulos requeridos, se debe utilizar una serie de identidades trigonométricas, relaciones e igualdades entre ángulos y triángulos.

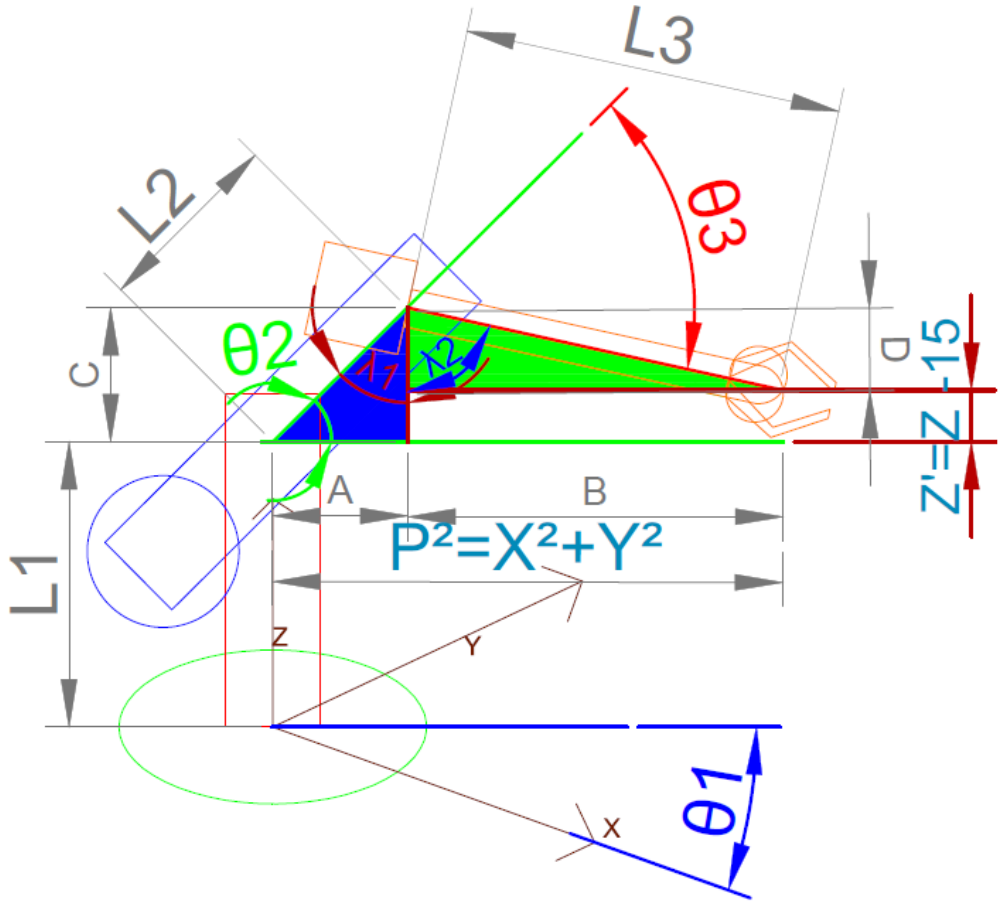

Figura 23. Triángulos utilizados para hallar θ_3.

Para hallar θ_3 se forman 2 triángulos principales como se muestra en la Figura 23. Partiendo de éstos, se plantean las siguientes identidades:

$$180° - (\lambda_1 + \lambda_2) = \theta_3$$

$$90° - \lambda_1 = \theta_2$$

$$P^2 = x^2 + y^2$$

$$P = A + B \rightarrow \begin{cases} A = L_2 * \cos(\theta_2) \\ B = L_3 * \sin(\lambda_2) \end{cases}$$

$$\boldsymbol{Z' = Z - L_1 = C - D \rightarrow \begin{cases} C = L_2 * \sin(\theta_2) \\ D = L_3 * \cos(\lambda_2) \end{cases}} \qquad \textit{Ecuación 9}$$

Al resolver la siguiente igualdad, utilizando las identidades antes mencionadas, se obtiene una ecuación que podría resolver θ_3:

$$P^2 + Z'^2 = x^2 + y^2 + Z'^2 = (A + B)^2 + (C - D)^2$$

$$A^2 + C^2 = {L_2}^2 \quad \& \quad B^2 + D^2 = {L_3}^2$$

$$2AB - 2CD = 2L_2L_3(\cos\theta_2 \sin\lambda_2 - \cos\lambda_2 \sin\theta_2) = 2L_2L_3 \sin(\lambda_2 - \theta_2)$$

$$\sin(\lambda_2 - \theta_2) = \sin(90 + \theta_2 - \theta_3 - \theta_2) = \cos\theta_3$$

$$\therefore \cos(\theta_3) = \frac{x^2 + y^2 + Z'^2 - L_2^2 - L_3^2}{2 * L_2 * L_3} \qquad \textit{Ecuación 10}$$

Se podría utilizar la Ecuación 10 para hallar θ_3, pero la función arcocoseno solo obtiene resultados positivos, lo cual no solo limita la actuación del sistema, sino que en este caso particular, elimina por completo los movimientos del eslabón 3, dado que por limitaciones físicas el sistema solo puede alcanzar posiciones con la configuración de codo arriba, es decir se debe cumplir siempre con la condición $\theta_3 \leq 0$. Por lo tanto para poder conseguir resultados negativos de θ_3, se convierte el arcocoseno de la Ecuación 10 en arcotangente con la identidad de la Ecuación 11, es decir el resultado es una ecuación de arcotangente en función del coseno de θ_3:

$$\tan(\theta_3) = \frac{\sin(\theta_3)}{\cos(\theta_3)} = \frac{\sqrt[2]{1 - \cos^2(\theta_3)}}{\cos(\theta_3)} \qquad \textit{Ecuación 11}$$

Debido a la limitación física antes mencionada, se toma solo el resultado negativo de la raíz, para así tener solo configuraciones de tipo codo arriba.

Para hallar θ_2 se utiliza un procedimiento similar al empleado para hallar θ_3. Se utilizan los triángulos mostrados en la Figura 24, los cuales permiten plantear las siguientes igualdades o identidades:

$$180° - \theta_3 = \gamma = 180° - \beta - \delta$$

$$\alpha + \beta = \theta_2 \quad \& \quad \delta = \theta_3 - \beta$$

$$L_2 * \sin(\beta) = U = L_3 * \sin(\delta) \qquad \textit{Ecuación 12}$$

El valor de α se obtiene analizando la Figura 24, mientras que utilizando las identidades anteriores para resolver la igualdad de la Ecuación 12, se obtiene el valor de β. Ambos valores se presentan en la Ecuación 13.

$$\frac{L_2}{L_3} = \frac{\sin(\theta_3 - \beta)}{\sin(\beta)} = \frac{\sin\theta_3 \cos\beta - \cos\theta_3 \sin\beta}{\sin(\beta)}$$

$$\begin{cases} \alpha = \operatorname{atan}\left(\frac{z}{p}\right) \\ \beta = atan\left(\frac{\sin(\theta_3)}{\frac{L_2}{L_3} + cos(\theta_3)}\right) \end{cases} \qquad \textit{Ecuación 13}$$

Se debe realizar un análisis de los posibles resultados de los ángulos para así evitar conflictos entre el ángulo calculado y el rango de ángulos posibles para el sistema. Según los límites físicos se debe cumplir con las siguientes condiciones:

- $0 \leq \theta_2 \leq 45°$
- $-90° \leq \theta_3 \leq 0$

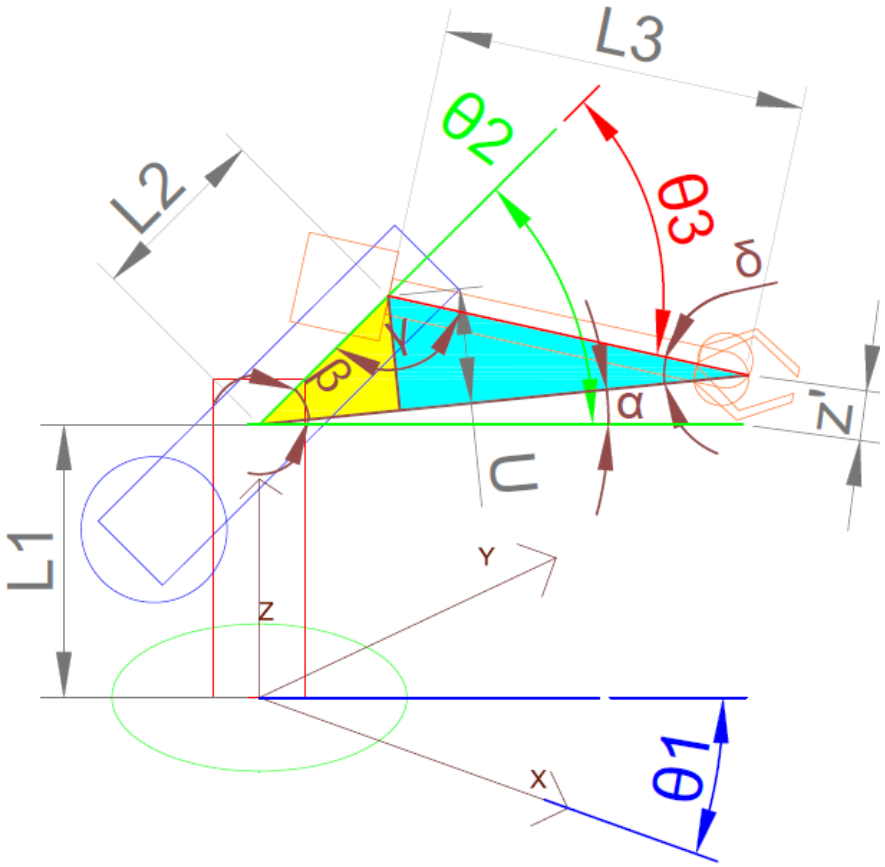

Figura 24. Triángulos utilizados para hallar θ_2.

Según la Ecuación 13, se aprecia que el signo de β depende del signo de θ_3, debido a las características de imparidad de las funciones seno y arcotangente. Por lo tanto al ser θ_3 siempre negativo, β también es siempre negativo. Este análisis lleva a replantear la ecuación de θ_2 (Ecuación 12), y cambiar el signo a β, teniendo como resultado:

$$\theta_2 = \alpha - \beta \qquad \textit{Ecuación 14}$$

Este cambio se realiza debido a que como se indica previamente, el sistema solo puede adoptar configuraciones de codo arriba, por lo tanto se debe cumplir para cualquier coordenada ingresada la condición $\alpha \leq \theta_2 \leq \alpha - \beta$, lo cual se cumple si y solo si β es negativo.

CAPÍTULO 7: RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

El resultado final del sistema es bastante satisfactorio, su espacio de trabajo es lo bastante amplio como para realizar manipulaciones estándar, y su error entre el posicionamiento teórico y el práctico es muy bajo. A continuación en el presente capítulo se presentan los resultados obtenidos tanto teóricos, que hace referencia a un buen diseño, como los resultados prácticos o experimentales.

7.1 Resultados

7.1.1 Espacio de trabajo

Para graficar el espacio de trabajo se obtuvieron todos los puntos a los cuales el efector final del brazo robótico puede llegar usando la función `fkine()` del toolbox de robótica de Matlab, tras lo cual mediante la función `mesh()` se graficaron en la Figura 25 y la Figura 26.

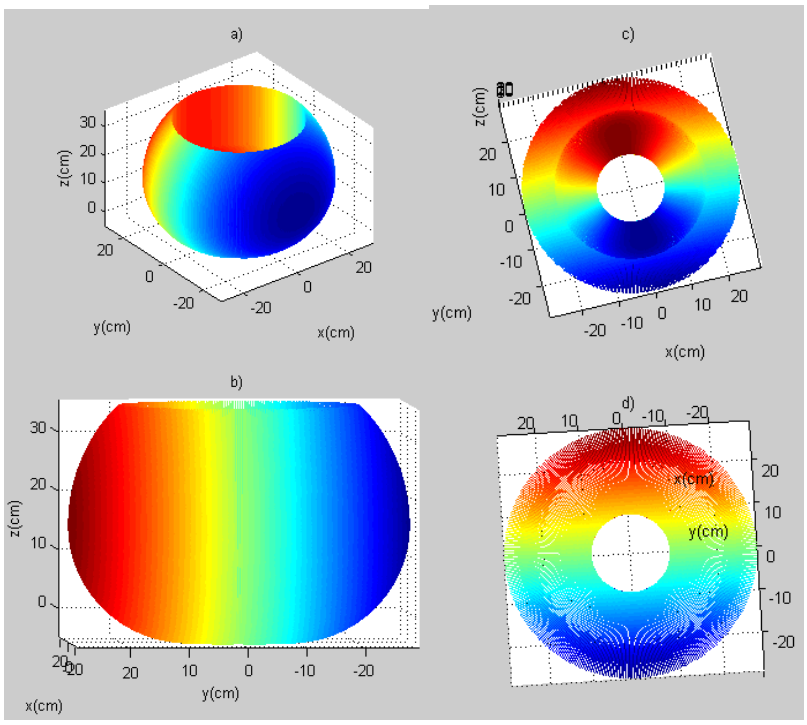

Figura 25. Espacio de trabajo del brazo

La Figura 25 a), nos muestra que el espacio de trabajo tiene una forma parecida a la de una esfera hueca a la cual se ha quitado una porción en la parte superior y una porción más pequeña en la parte inferior. En la Figura 25 b), se puede apreciar con mayor claridad que la apertura superior de la esfera es de mayor tamaño que la apertura inferior, y esto se debe a los límites establecidos en el tercer motor, cuyo rango de trabajo va de 0 a 90 grados.

La Figura 25 c), muestra una vista desde arriba del plano xy del espacio de trabajo, se puede apreciar con mayor claridad que el espacio de trabajo no tiene forma de una cáscara delgada, sino que es de un grosor considerable. En la Figura 25 d), se muestra de forma aún más clara que en algunos puntos el grosor del contorno del espacio de trabajo tiene un poco menos de 10 centímetros. También se puede ver que el radio de la circunferencia de la parte inferior menor a 10 centímetros.

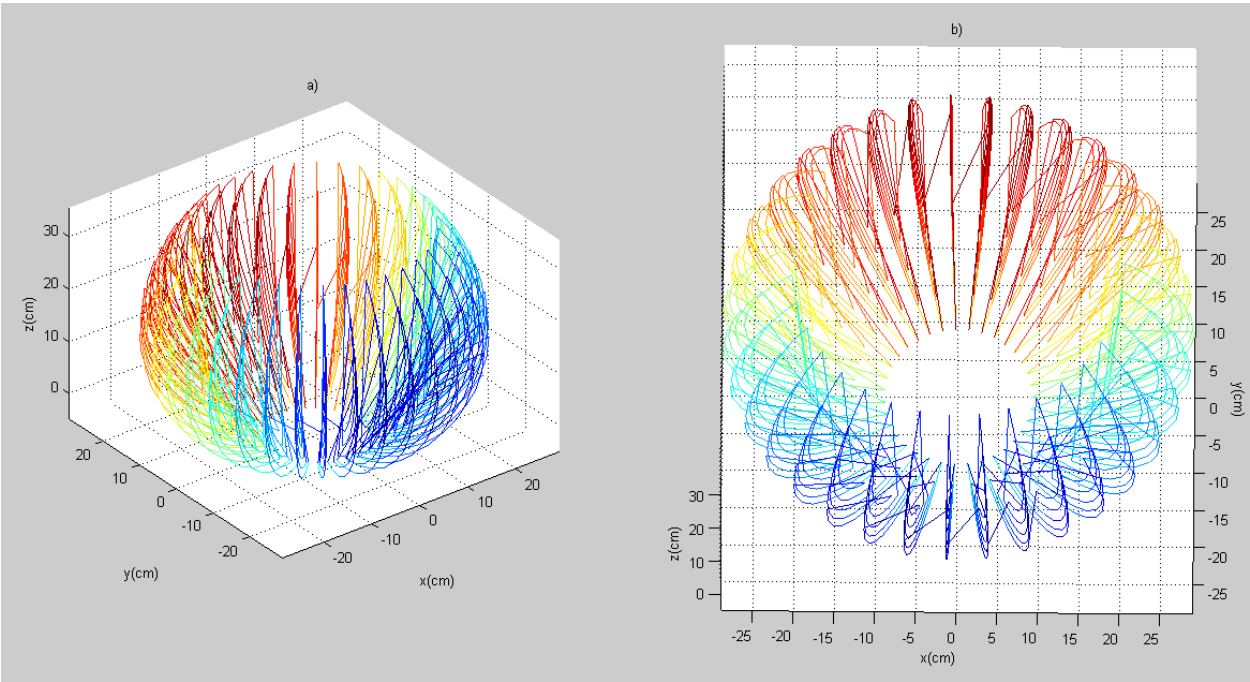

Figura 26. Espacio de trabajo graficado con menos puntos

La Figura 26 a), fue graficada usando menos puntos para ver en forma de malla el espacio de trabajo del brazo. En la Figura 26 b), se aprecia mejor el grosor del espacio de trabajo en cada posición del actuador final en el eje z.

7.1.2 Modelo cinemático directo

Para verificar los resultados del modelo cinemático directo, presentados en la Tabla 5, se situó al efector final en 10 posiciones diferentes mediante los ángulos θ_1, θ_2 y θ_3 y se compara los resultados para las coordenadas (x, y, z) calculadas mediante el modelo cinemático directo con los valores medidos experimentalmente.

Ángulos			Teórico			Real			%
$\theta_1(°)$	$\theta_2(°)$	$\theta_3(°)$	$x(cm)$	$y(cm)$	$z(cm)$	$x(cm)$	$y(cm)$	$z(cm)$	Error
0	0	0	10	0	-5	10	0	-5	0,00
0	30	0	18	0	2.7	18,5	0	3	2,88
45	0	0	7.1	7.1	-5	7,4	6,6	-5	-1,01
45	30	0	13.2	13.2	2.7	13,5	13	2,5	0,24
90	0	0	0	10	-5	2	9	-5	-6,60
90	0	45	0	24.1	0.8	2	23	0	-4,45
180	0	0	-10	0	-5	-9	1	-5	-8,08
180	0	45	-24.1	0	0.8	-23	1	0	-4,74
270	0	0	0	-10	-5	1,5	-9,5	-5	-3,14
270	0	45	0	-24.1	0.8	-1	-23	0	-4,74

Tabla 5. Resultados del modelo cinemático directo

El error calculado promedio es del -2.96%, éste se obtiene al comparar las coordenadas calculadas teóricamente mediante el modelo cinemático directo y las medidas experimentalmente halladas en el sistema. Este error se debe principalmente a que los encoders no miden grados sexagesimales, por lo que se debe hacer una conversión previa. Además,

dependiendo de la velocidad de rotación de los motores hay ocasiones en las que el Arduino a pesar de contar los pasos del encoder no procesa de manera instantánea la condición para detener los motores y se genera un error al pasar de la posición requerida.

7.1.3 Modelo cinemático inverso

La Tabla 6 muestra los resultados del modelo cinemático inverso, en la cual se relacionan las coordenadas teóricas requeridas y las coordenadas alcanzadas experimentalmente, para 5 posiciones o puntos aleatorios dados, que se encuentran dentro del espacio de trabajo.

Teóricas			Experimentales			ΔMagnitud (cm)	% ERROR
x_t(cm)	y_t(cm)	z_t(cm)	x_e(cm)	y_e(cm)	z_e(cm)		
20	-10	35	19.9	-10.1	35	0,86	-1,95
22	20	19	22	19.7	19.2	0,44	0,76
-18	19	29	-17.5	19.25	29	0,82	-0,94
0	10	-5	1	9.5	-5	0,54	-1,52
-30	0	15	-29.5	0	14	1,12	-2,72

Tabla 6. Resultados del modelo cinemático inverso

En la práctica, como se aprecia en la tabla anterior, se tiene una pequeña diferencia entre la posición deseada y la alcanzada. Esta variación posee un error en promedio de 1.57%, y dentro de las pruebas realizadas un máximo de 2,72%, lo cual es menor al 5% de error que es lo generalmente considerado como un máximo de tolerancia de error. La distancia entre la posición alcanzada y la posición deseada es de 0.75 cm en promedio, con un máximo de 1.12 cm. Todos estos factores demuestran que el sistema funciona de manera adecuada con un pequeño error casi despreciable, el cual se debe principalmente al siguiente factor:

- Los encoders trabajan solo con valores enteros, lo cual lleva a que los ángulos de movimiento que se envían como comando a cada motor, tengan una resolución de menor que los valores calculados en el modelo geométrico, el cual trabaja con

valores decimales. Es decir, se provoca un error a consecuencia de un redondeo forzado.

El sistema de control una vez finalizado se muestra con todos sus componentes en la Figura 27, mientras que el brazo robótico se presenta en diferentes configuraciones en la Figura 28.

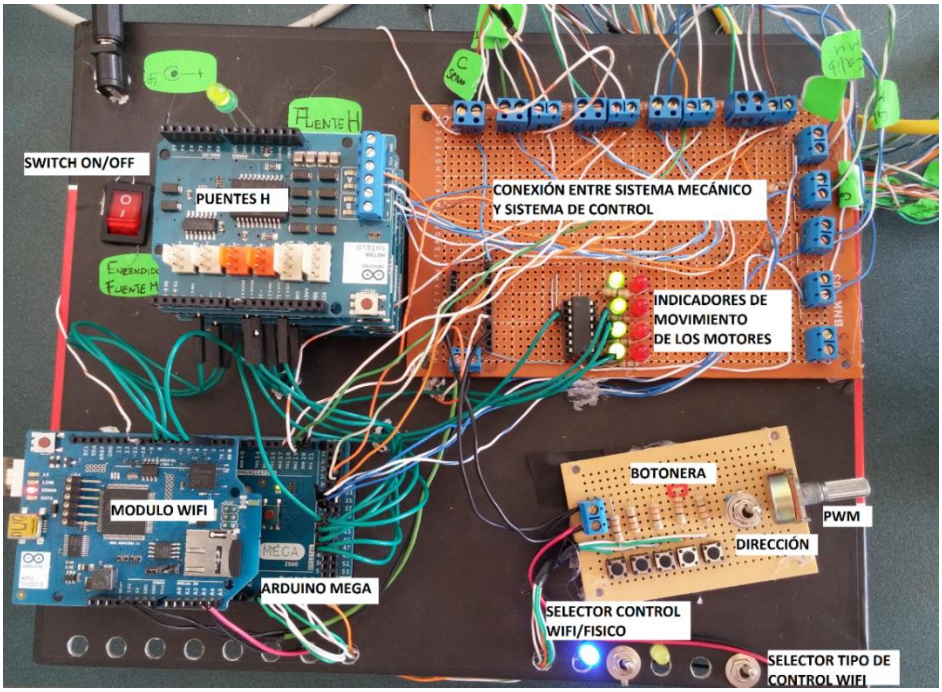

Figura 27. Panel de control finalizado

Como se aprecia en la figura, se tiene en la parte izquierda inferior al controlador o Arduino Mega, sobre éste se coloca el módulo WIFI y en la parte superior, se tiene la entrada de alimentación del sistema, el switch de encendido, y los puentes H colocados en cascada para así proporcionar una alimentación conjunta a esta sección. En la parte de la derecha superior se tiene la unión entre el sistema mecánico y el sistema de control, a través de borneras y un circuito sencillo de indicadores led para visualizar que motor está en funcionamiento. Y en la parte inferior se tiene el control físico o botonera detallada anteriormente, y 2 selectores, el

primero sirve para seleccionar si se quiere un control manual, lo cual activa la botonera y enciende el led azul, o si se quiere un control inalámbrico lo cual enciende el led amarillo. El otro selector hace la discriminación entre los 2 tipos de control inalámbrico, el control de articulaciones o el control de coordenadas generalizadas.

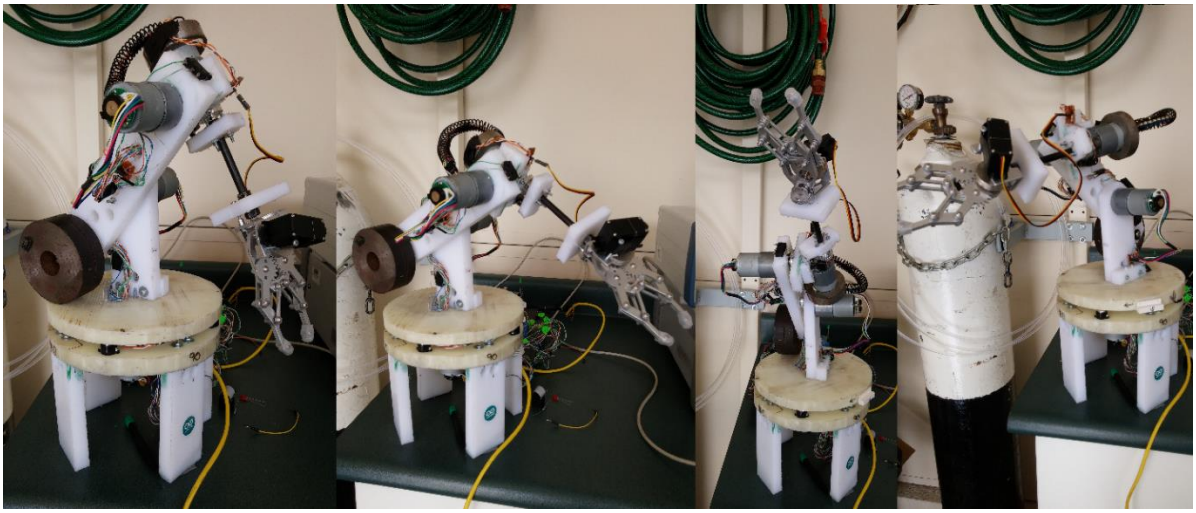

Figura 28. Brazo robótico en distintas posiciones

7.2 Conclusiones

Como se indica en la sección de resultados, el sistema funciona de manera satisfactoria, alcanza las posiciones solicitadas con un error menor al 5% tolerable, el cual se debe más a problemas mecánicos que a problemas de programación, por ejemplo a pesar de haber colocado macilla epoxy sigue existiendo un espacio minúsculo entre el eje y los acoples, y por ende el eslabón, lo que genera una rotación que no es exacta. Esta distancia en el eje puede ser de una magnitud despreciable, pero al final del eslabón representa algunos milímetros de error. Éste se va acumulando de cada articulación y así en el actuador final se tiene un error que ya es considerable.

En cuanto al sistema electrónico todo funciona de manera adecuada, el Arduino procesa de manera correcta tanto las señales inalámbricas como las enviadas manualmente, y trabaja muy bien con las alertas y los límites establecidos. Esto nos permite concluir que para esta clase de proyectos o sistemas el Arduino es una muy buena opción como sistema de control, cumple muy bien con lo solicitado y de manera eficaz.

Los objetivos planteados se cumplieron uno por uno, tenemos tanto el diseño, como construcción y control del sistema, a pesar de no haber utilizado al 100% el diseño original realizado, éste sirvió como un punto de partida para llegar al diseño final, la cual se desempeña de manera satisfactoria a pesar de los cambios y de no haber sido diseñada de manera muy detallada.

Se cumplió con la implementación del control manual, y de los 2 tipos de control inalámbrico, y además se los logró de cierta forma unificar los códigos al separar los comandos en funciones para así no tener mucha redundancia y no ocupar mucha de la memoria del Arduino que puede causar ciertos problemas.

La parte que se podría considerar como la más compleja del proyecto, es la del modelo cinemático, pero las simulaciones, las tablas de error, y las comparaciones entre cálculos y gráficos de Matlab con lo obtenido físicamente demuestran que el cálculo geométrico, a pesar de no ser de los más comunes, es muy acertado. Además, su planteamiento y deducción no representan un cálculo tan pesado como sería el tratar con las matrices de Denavit Hartenberg y sus inversas.

Algo que no se encuentra planteado de manera específica en los objetivos principales es el hecho de realizar la estructura del sistema con material reciclado en la medida de las

posibilidades. Esto fue muy tomado en cuenta, por esta razón se reutilizaron las placas de duralón en lugar de trabajar unas nuevas, en cuanto a los platos se adquirió los más económicos en la fábrica, es decir unos retazos que ya se encontraban cortados en material de grillón, reduciendo los costos de los platos a prácticamente la mitad. En cuanto a los pesos en lugar de realizar pesas de plomo bien diseñadas, se optó por buscar pesos entre los retazos que se encuentran en el taller de mecánica. Finalmente, para ahorrar lo que es mano de obra como tal, se optó por trabajar las piezas, sin el uso de mucho equipo especializado, sino al contrario con apenas una cierra, una lima, un taladro y varias brocas se logró realizar los cortes y las perforaciones necesarias para implementar la estructura.

7.3 Recomendaciones

Si se va a implementar un proyecto similar al brazo robótico presentado, se recomienda tomar en cuenta todos los pesos y torque generados por los eslabones del brazo robótico, y tomar en cuenta el escenario del peor caso, para así abarcar más allá de lo necesitado y no tener problemas de tipo mecánico.

Si no se tiene experiencia en el uso de programas de diseño mecánico, solicitar cualquier tipo de asesoría es de mucha ayuda ya que existen programas muy completos que pueden, con las simulaciones adecuadas, revelar problemas o inconvenientes que a simple vista no se observan.

Una recomendación en general para cualquier proyecto electromecánico es ir probando cada cable colocado a medida que se lo conecta, para evitar problemas y no tener que re-cablear el sistema. Siempre buscar las opciones más sencillas, la base de rotación pudo haber sido

mucho más completa y compleja, y se hubiera obtenido prácticamente los mismos resultados que se obtuvieron con apenas 4 ruedas locas un motor y los cortes adecuados.

En lo que respecta a la programación, es recomendable por más que se conozca una plataforma de programación, realizar una investigación que puede llevar a facilitar la programación e inclusive evitar líneas de código que pueden ser reemplazadas por un comando, como fue el caso de la función atan2, la cual se pudo haber programado pero hubiera llevado unas cuantas líneas de código y posiblemente generar una carga adicional en el procesamiento del Arduino.

Siempre es bueno explorar varias opciones. En este proyecto se consideraron varias opciones desde el material con el que se iba a realizar la estructura hasta la metodología con la cual se calcula el modelo cinemático directo. Esto no solo lleva a encontrar la solución más fácil, o económica, o eficaz, sino que también ayuda a enriquecer el conocimiento sobre algún tema en específico sobre el cual se puede tener poco o nada de información.

Una última recomendación es la de no fiarse al 100% por lo teórico o lo que ya está escrito. En éste caso, por ejemplo, se tuvo que obviar un poco la información presentada respecto al funcionamiento de los encoders de los motores, ya que la resolución que el datasheet presenta respecto al número de pasos por vuelta, en la práctica no es tan acertada. Además, como se ve desde un principio el diseño que se realizó tuvo que ser cambiado varias veces, la última inclusive cuando el sistema se encontraba ya armado, debido a que a pesar de lo completo que sea un software, sus simulaciones son hechas para casos específicos de materiales, actuadores, sensores, etc., los cuales en la práctica tal vez son inaccesibles, o simplemente muy complejos de implementar.

REFERENCIAS

3D CAD Softare | Inventor 3D CAD | Autodesk. (s.f.). Recuperado el 2014, de Autodesk: http://www.autodesk.com/products/inventor/overview

Aparicio, M. (2004). Control para un brazo robot colocado sobre plataforma móvil "Úrsula". Colombia. Obtenido de http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis93.pdf

Arduino - Introduction. (s.f.). Recuperado el 2014, de Arduino: http://arduino.cc/en/Guide/Introduction

Arduino - Software. (s.f.). Recuperado el 2014, de Arduino: http://arduino.cc/en/Main/Software

Arduino Mega 2560. (s.f.). Recuperado el 2014, de Arduino: http://arduino.cc/en/Main/arduinoBoardMega2560

Arduino Motor Shield R3. (s.f.). Recuperado el 2014, de Arduino: http://arduino.cc/en/Main/ArduinoMotorShieldR3

Arduino WiFi Shield. (s.f.). Recuperado el 2014, de Arduino: http://arduino.cc/en/Main/ArduinoWiFiShield

Castejón, C. (s.f.). Parámetros de Denavit-Hartenberg. Recuperado el 12 de Diciembre de 2014, de http://wainu.ii.uned.es/WAINU/ingenierias-tecnicas/optativas/robotica/otros/Parametros%20D-H.pdf

Eitel, E. (7 de Mayo de 2014). Basics of rotary encoders: Overview and new technologies. Recuperado el 12 de Diciembre de 2014, de http://machinedesign.com/sensors/basics-rotary-encoders-overview-and-new-technologies-0

Fitzgerald, A. E., Kingsley, C. J., & Umans, S. D. (2003). *Electric Machinery* (6th ed.). McGraw Hill.

Giancoli, D. C. (2008). *Physics for scientists and engineers with modern physics* (4th ed.). Pearson.

Hallang, W. A., & Sacha, K. M. (1992). Real-time systems: Implementation of industrial computerised process automation. (W. Scientific, Ed.)

Hassan, H., Soriano, J., Montagud, J., & Domínguez, C. (2005). Instrucción en el diseño de sistemas empotrados. Aplicación al control de un brazo robótico. España. Obtenido de http://www.ceautomatica.es/old/actividades/jornadas/XXIV/documentos/econ/90.pdf

Koyuncu, B., & Güzel, M. (2007). Software Development for the Kinematic Analysis of a Lynx 6 Robot Arm. *International Journal of Applied Science, Engineering and Technology, 4*.

Laplante, P. A. (1992). Real-time systems design and analysis. *An engineer's handbook*. IEEE Press.

Martinez, G. (Junio de 2008). Diseño propio y construcción de un brazo robótico con 5 GDL. Obtenido de http://antiguo.itson.mx/rieeandc/vol4p1_archivos/Art2Junio08.pdf

Mason, G. W. (s.f.). Generalized Coordinates Systems. Provo, Utah, United States of America. Obtenido de http://einstein1.byu.edu/~masong/emsite/downloads/pdffiles/generalizedcoords.pdf

Mena, D. (2011). Términos de Glosario de IA y Robótica. Retrieved from http://www.cursosporinternet.info/index.php/the-news/38-iar/69-terminos-del-glosario-de-ia-y-robotica.html

Mendoza, E. (2004). Control de un Robot Manipulador. (Tesis Profesional). México. Obtenido de http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/mendoza_s_ea/capitulo2.pdf

Notepad++ Home. (s.f.). Recuperado el 2014, de Notepad++: http://notepad-plus-plus.org

Ollero, A. (2001). *Robótica, maniuladores y robots móviles.* Barcelona, España: Marcombo.

Paul, R. P. (1981). *Robot manipulators: mathematics, programming, and control : the computer control of robot manipulators.* Massachusetts, United States of America.

Pernia, R. (2002). Diseño y Construcción del Prototipo Mecánico de un Robot de Limpieza Doméstico. (Proyecto de Especialidad Mecánica).

Ramírez, K. (s.f.). Cinemática directa del robot. Recuperado el 12 de Diciembre de 2014, de http://www.kramirez.net/Robotica/Material/Presentaciones/CinematicaDirectaRobot.pdf

Raz, T. (1989). Graphics Robot Simulator for Teaching Introductory Robotics. *IEEE Transactions on Education, 32(2)*.

Romero, A. (s.f.). Estructura de un robot industrial. Obtenido de http://www.juntadeandalucia.es/averroes/iesalfonso_romero_barcojo/departamentos/tecnologia/unidades_didacticas/ud_controlroboticav1/morfologia%20de%20un%20robot.pdf

Sandoval, R. (2007). Apuntes de Fundamentos de Robótica.

Schilling, R. (1990). Fundamentals of Robotics: Analysis and Control. Englewood Cliffs, N.J.: Prentice Hall.

SD Library. (s.f.). Recuperado el 2014, de Arduino: http://arduino.cc/en/Reference/SD

SPI Library. (s.f.). Recuperado el 2014, de Arduino: http://arduino.cc/en/Reference/SPI

Stone, H. W. (1986). *Kinematic modeling, identification, and control of robotic manipulators.* Pittsburgh, Pennsylvania, Estados Unidos de América.

WiFi Library. (s.f.). Recuperado el 2014, de Arduino: http://arduino.cc/en/Reference/WiFi

ANEXOS

Anexo A

A.1. Proforma nacional

La proforma nacional está realizada en base a 3 proformas de 3 tiendas de electrónica que son líderes en el mercado quiteño. La Tabla 7 resume los elementos más costosos y más importantes del sistema, de las 3 proformas obtenidas:

Parte	**Cantidad**	**P. Unitario**	**Total**
Motor DC 100:1	5	$45	$225
Encoders externos	5	$20.25	$101.25
Arduino Mega 2560 R3	1	$52.50	$52.50
Arduino Wifi Shield	No hay		
Puente H	5	$16	$80
Total			**$458.75**

Tabla 7. Proforma nacional

En cuanto al resto de materiales que se utilizan para las conexiones, construcción de la estructura y ensamblaje de la misma se presenta la Tabla 8.

Como se aprecia la compra representa un valor alto, sin considerar el resto de materiales, teniendo que adaptar muchos circuitos por cada motor aparte de la ubicación de los encoders, y considerando que un componente esencial como es el módulo Wifi para Arduino no se lo puede conseguir en el mercado nacional y se tendría que importar o adaptar otra solución para la comunicación inalámbrica.

Parte	**Cantidad**	**P. Unitario**	**Total**
Plancha Duralón 25x30x2cm	1	$30	$30
Cilindros Duralón 25dx2cm	2	$10.50	$21
Varios electrónicos	Paquete	$10	$10
Tornillos, rodelas y tuercas diferente tamaño	50	$.15	$7.5
Brocas diferente tamaño x3	1	$8	$8
Finales de Carrera	3	1.2$	3.6$
Sensor magnético	1	4.2$	4.2$
Pinza	1	$28	$28
Acople (motor – Sistema) x 2	2	$15	$30
Ruedas locas	4	$5.50	$22
Teclado matricial	1	$5	$5
Total			**$169.30**

Tabla 8. Costos de materiales

A.2. Proforma internacional

De acuerdo a lo mencionado anteriormente, algunos de los elementos usados en la construcción del brazo no se pudieron encontrar en el mercado local, por este motivo se decidió realizar las compras de estos elementos en Estados Unidos.

Los materiales comprados en Estados Unidos fueron los elementos de control y los actuadores principales del brazo, es decir la tarjeta Arduino con todos sus módulos, y todos los

motores que harán funcionar el equipo. Los motores fueron comprados directamente de la página del fabricante (http://www.pololu.com) y la tarjeta Arduino junto con los módulos en Amazon (http://www.amazon.com).

Los costos unitarios y totales de los materiales comprados en el extranjero se detallan en la Tabla 9.

Parte	**Cantidad**	**P. Unitario**	**Total**
Motor con encoder 131:1	2	$39.95	$79.90
Motor con encoder 100:1	3	$39.95	$119.85
Arduino Mega 2560 R3	1	$45.00	$45.00
Arduino Wi-Fi Shield	1	$89.95	$89.95
Arduino Motor Controller	5	$25.95	$129.75
Total			**$464.45**

Tabla 9. Proforma internacional

Anexo B

B.1. Código del controlador (Arduino)

```
#include <SPI.h>
#include <WiFi.h>
#include <SD.h>
#include <Servo.h>
#include <math.h>

//WiFi
char ssid[] = "RedBrazo1"; //SSID
char pass[] = "cinematica123"; //Password
int status = WL_IDLE_STATUS; //WiFi radio's status
WiFiServer server(80); //Crea un servidor en el puerto 80

String HTTP_req; //Almacena el request HTTP
File webFile;

//Inicialización de Pines
//Motor 1
const int pwmMotor1 = 2;
//const int encoderMotor1 = 21;//24
const int calibMotor1 = 32;
```

```
const int brakeMotor1 = 38;
const int dirMotor1 = 39;
volatile int posMotor1 = 0;
int ultEncoder1 = LOW;
int Encoder1 = LOW;
int dir1 = LOW;
int pos1 = 0;
//Motor 2
const int pwmMotor2 = 3;
//const int encoderMotor2 = 20;//26
const int calibMotor2 = 33;
const int brakeMotor2 = 40;
const int dirMotor2 = 41;
volatile int posMotor2 = 0;
int ultEncoder2 = LOW;
int Encoder2 = LOW;
int dir2 = LOW;
int pos2 = 0;
//Motor 3
const int pwmMotor3 = 5;
//const int encoderMotor3 = 19;//28
const int calibMotor3 = 34;
const int brakeMotor3 = 42;
const int dirMotor3 = 43;
volatile int posMotor3 = 0;
int ultEncoder3 = LOW;
int Encoder3 = LOW;
int dir3 = LOW;
int pos3 = 0;
//Motor 4
const int pwmMotor4 = 6;
//const int encoderMotor4 = 18;//30
const int calibMotor4 = 35;
const int brakeMotor4 = 44;
const int dirMotor4 = 45;
volatile int posMotor4 = 0;
int ultEncoder4 = LOW;
int Encoder4 = LOW;
int dir4 = LOW;
int pos4 =0;
//Servo
Servo pinza; //Usa pin 9 (PWM)
int abierto = 1;

//pulsadores control manual
int botmanual=A5;
int botdir=A14;
int bot1=A8;
int bot2=A9;
int bot3=A10;
int bot4=A11;
int bot5=A12;
//pulsador selección de página web
int controlweb = A4;

int pwmin=A15;
int varpwm=0;
// Variables para modelo inverso
//constantes longitudes links
double l1=15;
double l2=10;
double l3=20;
//datos a leer desde pag web
```

```
int xcor=0;//-30 -> 30
int ycor=0;
int zcor=0;//datos en z restado offset
int zcor1=0;// dato leido
//variables para calculos varios
float cosq3=0;
float q3=0;
float q2=0;
float alpha=0;
float beta=0;
//angulos para motores
int theta1=0;
int theta2=0;
int theta3=0;
//constantes pi
const float pi=3.141592654;
//control serial
int cont=1;
String inString="";
int ant1 = 0;
int ant2 = 0;
int ant3 = 0;
int ant4 = 0;

void setup()
{
  server.begin(); //Inicia el servidor web
  Serial.begin(9600); //Para debugging

  //Pines
  pinMode(pwmMotor1,OUTPUT);
  //pinMode(encoderMotor1,INPUT);
  pinMode(calibMotor1,INPUT);
  pinMode(brakeMotor1,OUTPUT);
  pinMode(dirMotor1,OUTPUT);
  attachInterrupt(2,con1,RISING);

  pinMode(pwmMotor2,OUTPUT);
  //pinMode(encoderMotor2,INPUT);
  pinMode(calibMotor2,INPUT);
  pinMode(brakeMotor2,OUTPUT);
  pinMode(dirMotor2,OUTPUT);
  attachInterrupt(3,con2,RISING);

  pinMode(pwmMotor3,OUTPUT);
  //pinMode(encoderMotor3,INPUT);
  pinMode(calibMotor3,INPUT);
  pinMode(brakeMotor3,OUTPUT);
  pinMode(dirMotor3,OUTPUT);
  attachInterrupt(4,con3,RISING);

  pinMode(pwmMotor4,OUTPUT);
  //pinMode(encoderMotor4,INPUT);
  pinMode(calibMotor4,INPUT);
  pinMode(brakeMotor4,OUTPUT);
  pinMode(dirMotor4,OUTPUT);
  attachInterrupt(5,con4,RISING);

  pinza.attach(9);

  //Revisar presencia del módulo WiFi
  if (WiFi.status() == WL_NO_SHIELD){
    Serial.println("Módulo WiFi no está presente");
```

```
    //No continuar ejecutando el prorgama
    while(true);
  }

  //Conectarse a la red WiFi:
  while (status != WL_CONNECTED){
    Serial.print("Estableciendo conexión a la red: ");
    Serial.println(ssid);
    //Conectarse a la red
    status = WiFi.begin(ssid,pass);
    //Esperar 10 segundos hasta conectarse
    delay(10000);
  }

  //Información de la conexión:
  Serial.println("Conectado a la red");
  Serial.println(WiFi.localIP());

  //Inicialización de tarjeta SD
  Serial.println("Inicializando tarjeta SD...");
  if (!SD.begin(4)){
    Serial.println("ERROR - Falló la inicialización de tarjeta SD!");
    return; // inicialización fallida
  }
  Serial.println("Tarjeta SD inicilizada exitosamente.");
  //Verificar el archivo CtrlInde.htm
  if(!SD.exists("CtrlInde.htm")){
    Serial.println("ERROR - El archivo CtrlInde.htm no existe!");
    return; //No se encontró el archivo
  }
  Serial.println("Archivo Coordena.htm encontrado!");
  if(!SD.exists("Coordena.htm")){
    Serial.println("ERROR - El archivo Coordena.htm no existe!");
    return; //No se encontró el archivo
  }
  Serial.println("Archivo Coordena.htm encontrado!");
  Serial.println("Calibrando");

  calibrar();

  pinMode(bot1,INPUT);
  pinMode(bot2,INPUT);
  pinMode(bot3,INPUT);
  pinMode(bot4,INPUT);
  pinMode(bot5,INPUT);
  pinMode(botdir,INPUT);
  pinMode(botmanual,INPUT);
}

void loop()
{
  if(digitalRead(botmanual)==LOW)
  {contmanual();}
  else {
  //contadores();
  //Freno de fin de carrera
  if (digitalRead(calibMotor1)==HIGH && dirMotor1 == LOW){
    digitalWrite(brakeMotor1,HIGH);
    analogWrite(pwmMotor1,0);
    posMotor1 = 0;
  }

  if (digitalRead(calibMotor2)==HIGH){
```

```
    digitalWrite(brakeMotor2,HIGH);
    analogWrite(pwmMotor2,0);
    posMotor2 = 0;
  }

  if (digitalRead(calibMotor3)==HIGH){
    digitalWrite(brakeMotor3,HIGH);
    analogWrite(pwmMotor3,0);
    posMotor3 = 0;
  }

  if (digitalRead(calibMotor4)==HIGH){
    digitalWrite(brakeMotor4,HIGH);
    analogWrite(pwmMotor4,0);
    posMotor4 = 0;
  }

  WiFiClient client = server.available();  // try to get client

    if (client) {  // got client?
        boolean currentLineIsBlank = true;
        while (client.connected()) {
            if (client.available()) {   // client data available to read
                char c = client.read(); // read 1 byte (character) from
client
                HTTP_req += c;  // save the HTTP request 1 char at a time
                // last line of client request is blank and ends with \n
                // respond to client only after last line received
                if (c == '\n' && currentLineIsBlank) {
                    // send a standard http response header
                    client.println("HTTP/1.1 200 OK");
                    client.println("Content-Type: text/html");
                    client.println("Connection: close");
                    client.println();
                    // send web page
                    if(digitalRead(controlweb) == HIGH){
                      webFile = SD.open("CtrlInde.htm");
                    } else {
                      webFile = SD.open("Coordena.htm");
                    }
                    // open web page file
                    if (webFile) {
                      byte clientBuf[64];
                      int clientCount = 0;

                        while(webFile.available()) {
                            clientBuf[clientCount] = webFile.read();
                            clientCount++;
                            if(clientCount>63){
                            client.write(clientBuf,64); // send web page to
client
                            clientCount = 0;
                            }
                        }
                        if(clientCount>0)
                          client.write(clientBuf,clientCount);
                        webFile.close();
                    }
                    Serial.print(HTTP_req);
                    moverMotores(client);
                    HTTP_req = "";
                    break;
                }
```

```
                // every line of text received from the client ends with
\r\n
                if (c == '\n') {
                    // last character on line of received text
                    // starting new line with next character read
                    currentLineIsBlank = true;
                }
                else if (c != '\r') {
                    // a text character was received from client
                    currentLineIsBlank = false;
                }
            } // end if (client.available())
        } // end while (client.connected())
        delay(1);      // give the web browser time to receive the data
        client.stop(); // close the connection
    } // end if (client)
  }
}

void moverMotor1(int pos1){
  if (pos1 == 15*131)
    pos1 =0;

  Serial.print("Motor 1 a:");
  Serial.println(pos1);

  if (pos1 == ant1)
    return;
  ant1 = pos1;

  while((pos1 != posMotor1)){
    //Verificar dirección
    if(pos1 > posMotor1){
      digitalWrite(dirMotor1,HIGH);
      dir1 = HIGH;
    } else {
      if(digitalRead(calibMotor1) == HIGH && dirMotor1 == LOW){
        digitalWrite(brakeMotor1,HIGH);
        analogWrite(pwmMotor1,0);
        posMotor1 = 0;
        break;
      }
      digitalWrite(dirMotor1,LOW);
      dir1 = LOW;
    }
    //Mover motor
    digitalWrite(brakeMotor1,LOW);
    analogWrite(pwmMotor1,170);
  }
  Serial.print("Motor 1: ");
  Serial.println(posMotor1);
  digitalWrite(brakeMotor1,HIGH);
  delay(500);
  analogWrite(pwmMotor1,0);
}

void moverMotor2(int pos2){
  Serial.print("Motor 2 a:");
  Serial.println(pos2);

  if (pos2 == ant2)
    return;
  ant2 = pos2;
```

```
  if (pos2<100 && pos2<posMotor2 ){
  while (digitalRead(calibMotor2)==LOW)
  {digitalWrite(dirMotor2,LOW);
  digitalWrite(brakeMotor2,LOW);
  dir2 = HIGH;
  analogWrite(pwmMotor2,200);
//Serial.println("pos2");}
  }
  delay(700);
  analogWrite(pwmMotor2,0);
  delay(300);
  posMotor2=0;}



  while((pos2 != posMotor2)){
    //Verificar dirección
    if(pos2 > posMotor2){
      digitalWrite(dirMotor2,HIGH);
      dir2 = HIGH;
    } else {
      if(digitalRead(calibMotor2) == HIGH){
        //digitalWrite(brakeMotor2,HIGH);
        delay(500);
        analogWrite(pwmMotor2,0);
        posMotor2 = 0;
        break;
      }
      digitalWrite(dirMotor2,LOW);
     dir2 = LOW;
    }
    //Mover motor
    digitalWrite(brakeMotor2,LOW);
    if(dir2 == HIGH)
      analogWrite(pwmMotor2,130);
      else
      analogWrite(pwmMotor2,150);
    //analogWrite(pwmMotor2,175);
  }


  digitalWrite(brakeMotor2,HIGH);
  delay(500);
  analogWrite(pwmMotor2,0);
   delay(200);
   int er2=abs(pos2-posMotor2);
   Serial.println(er2);
   while (er2>5){
  while((pos2 != posMotor2)){
    //Verificar dirección
    if(pos2 > posMotor2){
      digitalWrite(dirMotor2,HIGH);
      dir2 = HIGH;
    } else {
      if(digitalRead(calibMotor2) == HIGH){
        digitalWrite(brakeMotor2,HIGH);
        delay(500);
        analogWrite(pwmMotor2,0);
        posMotor2 = 0;
        break;
      }
      digitalWrite(dirMotor2,LOW);
     dir2 = LOW;
```

```
    }
    //Mover motor
    digitalWrite(brakeMotor2,LOW);
    if(dir2 == HIGH)
      analogWrite(pwmMotor2,100);
      else
      analogWrite(pwmMotor2,100);
    //analogWrite(pwmMotor2,175);
  }

  digitalWrite(brakeMotor2,HIGH);
  delay(500);
  analogWrite(pwmMotor2,0);
 er2=abs(pos2-posMotor2);
 }
  Serial.print("Motor 2: ");
  Serial.println(posMotor2);
}

void moverMotor3(int pos3){
  Serial.print("Motor 3 a:");
  Serial.println(pos3);

  if (pos3 == ant3)
    return;
  ant3 = pos3;

  while((pos3 != posMotor3)){
    //Verificar dirección
    if(pos3 > posMotor3){
      digitalWrite(dirMotor3,HIGH);
      dir3 = HIGH;
    } else {
      if(digitalRead(calibMotor3) == HIGH){
        digitalWrite(brakeMotor3,HIGH);
        delay(500);
        analogWrite(pwmMotor3,0);
        posMotor3 = 0;
        break;
      }
      digitalWrite(dirMotor3,LOW);
     dir3 = LOW;
    }
    //Mover motor
    digitalWrite(brakeMotor3,LOW);
    if(dir3 == HIGH)
      analogWrite(pwmMotor3,150);
      else
      analogWrite(pwmMotor3,60);
  }

  digitalWrite(brakeMotor3,HIGH);
  delay(500);
  analogWrite(pwmMotor3,0);
  delay(200);
   int er3=abs(pos3-posMotor3);
   Serial.println(er3);
   while (er3>7){

     while((pos3 != posMotor3)){
    //Verificar dirección
    if(pos3 > posMotor3){
      digitalWrite(dirMotor3,HIGH);
```

```
      dir3 = HIGH;
    } else {
      if(digitalRead(calibMotor3) == HIGH){
        digitalWrite(brakeMotor3,HIGH);
        delay(500);
        analogWrite(pwmMotor3,0);
        posMotor3 = 0;
        break;
      }
      digitalWrite(dirMotor3,LOW);
     dir3 = LOW;
    }
    //Mover motor
    digitalWrite(brakeMotor3,LOW);
    if(dir3 == HIGH)
      analogWrite(pwmMotor3,150);
      else
      analogWrite(pwmMotor3,60);
  }

  digitalWrite(brakeMotor3,HIGH);
  delay(500);
  analogWrite(pwmMotor3,0);
  delay(200);

   er3 = abs(pos3-posMotor3);
   }
  Serial.print("Motor 3: ");
  Serial.println(posMotor3);
}

void moverMotor4(int pos4){
  Serial.print("Motor 4 a:");
  Serial.println(pos4);

  if (pos4 == ant4)
    return;
  ant4 = pos4;

  while((pos4 != posMotor4)){
    //Verificar dirección
    if(pos4 > posMotor4){
      digitalWrite(dirMotor4,HIGH);
      dir4 = HIGH;
    } else {
      if(digitalRead(calibMotor4) == HIGH){
        digitalWrite(brakeMotor4,HIGH);
        delay(500);
        analogWrite(pwmMotor4,0);
        posMotor4 = 0;
        break;
      }
      digitalWrite(dirMotor4,LOW);
     dir4 = LOW;
    }
    //Mover motor
    digitalWrite(brakeMotor4,LOW);
    analogWrite(pwmMotor4,60);
  }
  Serial.print("Motor 4: ");
  Serial.println(posMotor4);
  digitalWrite(brakeMotor4,HIGH);
  delay(500);
```

```
  analogWrite(pwmMotor4,0);
}

void calibrar(){
  Serial.print("posMotor1: ");
  Serial.println(posMotor1);
  while(digitalRead(calibMotor1) != HIGH){
    digitalWrite(dirMotor1,LOW);
    digitalWrite(brakeMotor1,LOW);
    analogWrite(pwmMotor1,175);
  }
  digitalWrite(brakeMotor1,HIGH);
  delay(500);
  analogWrite(pwmMotor1,0);
  Serial.println("Motor 1 Calibrado");
  Serial.print("posMotor1: ");
  Serial.println(posMotor1);
  posMotor1 = 0;
  Serial.println(posMotor1);

  while(digitalRead(calibMotor2) != HIGH){
    digitalWrite(dirMotor2,LOW);
    digitalWrite(brakeMotor2,LOW);
    analogWrite(pwmMotor2,150);
  }
  delay(700);
  digitalWrite(brakeMotor2,HIGH);
  analogWrite(pwmMotor2,0);
  Serial.println("Motor 2 Calibrado");
  Serial.print("posMotor2: ");
  Serial.println(posMotor2);
  posMotor2 = 0;
  Serial.println(posMotor2);

  while(digitalRead(calibMotor3) != HIGH){
    digitalWrite(dirMotor3,LOW);
    digitalWrite(brakeMotor3,LOW);
    analogWrite(pwmMotor3,150);
  }
  digitalWrite(brakeMotor3,HIGH);
  delay(500);
  analogWrite(pwmMotor3,0);
  Serial.println("Motor 3 Calibrado");
  Serial.print("posMotor3: ");
  Serial.println(posMotor3);
  posMotor3 = 0;
  Serial.println(posMotor3);

  while(digitalRead(calibMotor4) != HIGH){
    digitalWrite(dirMotor4,LOW);
    digitalWrite(brakeMotor4,LOW);
    analogWrite(pwmMotor4,43);
  }
  digitalWrite(brakeMotor4,HIGH);
  delay(500);
  analogWrite(pwmMotor4,0);
  Serial.println("Motor 4 Calibrado");
  Serial.print("posMotor4: ");
  Serial.println(posMotor4);
  posMotor4 = 0;
  Serial.println(posMotor4);
}
```

```
// Obtener valores de la página web
void moverMotores(WiFiClient cl){
  if(HTTP_req.indexOf("base") > -1){
    Serial.print("Base = ");

Serial.println(HTTP_req.substring(HTTP_req.indexOf("base=")+5,HTTP_req.inde
xOf("&link1")));
    int a = 0;
    a =
HTTP_req.substring(HTTP_req.indexOf("base=")+5,HTTP_req.indexOf("&link1")).
toInt();
    pos1 = map(360-a,0,360,0,15*131);
    moverMotor1(pos1);
  }

  if(HTTP_req.indexOf("link2") > -1){
    Serial.print("Link2 = ");

Serial.println(HTTP_req.substring(HTTP_req.indexOf("link2=")+6,HTTP_req.ind
exOf("&link3")));
    int a = 0;
    a =
HTTP_req.substring(HTTP_req.indexOf("link2=")+6,HTTP_req.indexOf("&link3"))
.toInt();
    pos3 = map(a,0,90,0,440);//4*100);
    moverMotor3(pos3);
  }

  if(HTTP_req.indexOf("link1") > -1){
    Serial.print("Link1 = ");

Serial.println(HTTP_req.substring(HTTP_req.indexOf("link1=")+6,HTTP_req.ind
exOf("&link2")));
    int a = 0;
    a =
HTTP_req.substring(HTTP_req.indexOf("link1=")+6,HTTP_req.indexOf("&link2"))
.toInt();
    pos2 = map(a,0,45,0,330);//230);
    moverMotor2(pos2);
  }

  if(HTTP_req.indexOf("link3") > -1){
    Serial.print("Link3 = ");
    if(HTTP_req.indexOf("pinza") > -1){

Serial.println(HTTP_req.substring(HTTP_req.indexOf("link3=")+6,HTTP_req.ind
exOf("&pinza")));
      int a = 0;
      a =
HTTP_req.substring(HTTP_req.indexOf("link3=")+6,HTTP_req.indexOf("&pinza"))
.toInt();
      pos4 = map(a,0,90,0,350);//200);//4*100);
      moverMotor4(pos4);
    } else {

Serial.println(HTTP_req.substring(HTTP_req.indexOf("link3=")+6,HTTP_req.ind
exOf("HTTP")-1));
      int a = 0;
      a =
HTTP_req.substring(HTTP_req.indexOf("link3=")+6,HTTP_req.indexOf("HTTP")-
1).toInt();
      pos4 = map(abs(a),0,90,0,350);//200);//4*100);
      moverMotor4(pos4);
```

```
    }
  }

  if(HTTP_req.indexOf("xcor") > -1){
    Serial.print("X = ");

Serial.println(HTTP_req.substring(HTTP_req.indexOf("xcor=")+5,HTTP_req.inde
xOf("&ycor")));
    xcor =
HTTP_req.substring(HTTP_req.indexOf("xcor=")+5,HTTP_req.indexOf("&ycor")).t
oInt();
    ycor =
HTTP_req.substring(HTTP_req.indexOf("ycor=")+5,HTTP_req.indexOf("&zcor")).t
oInt();
    zcor1 =
HTTP_req.substring(HTTP_req.indexOf("zcor=")+5,HTTP_req.indexOf("&orie")).t
oInt();
    calcang(xcor,ycor,zcor1);
  }

  if(HTTP_req.indexOf("orie") > -1){
    Serial.print("Orientacion = ");
    if(HTTP_req.indexOf("pinza") > -1){

Serial.println(HTTP_req.substring(HTTP_req.indexOf("orie=")+5,HTTP_req.inde
xOf("&pinza")));
      int a = 0;
      a =
HTTP_req.substring(HTTP_req.indexOf("orie=")+5,HTTP_req.indexOf("&pinza")).
toInt();
      pos4 = map(a,0,90,0,350);//200);//4*100);
      moverMotor4(pos4);
    } else {

Serial.println(HTTP_req.substring(HTTP_req.indexOf("orie=")+5,HTTP_req.inde
xOf("HTTP")-1));
      int a = 0;
      a =
HTTP_req.substring(HTTP_req.indexOf("orie=")+5,HTTP_req.indexOf("HTTP")-
1).toInt();
      pos4 = map(abs(a),0,90,0,350);//200);//4*100);
      moverMotor4(pos4);
    }
  }

  if(HTTP_req.indexOf("pinza") > -1){
    Serial.print("Pinza = ");

Serial.println(HTTP_req.substring(HTTP_req.indexOf("pinza=")+5,HTTP_req.ind
exOf("HTTP")-1));
    if (abierto == 1){
      pinza.write(255);
      abierto = 0;
    } else{
      pinza.write(75);
      abierto = 1;
    }
  }
}

void con1(){
  if(dir1==HIGH)
    posMotor1++;
```

```
    else
    posMotor1--;
   if(posMotor1==pos1 || calibMotor1 == HIGH){ //|| posMotor1 <=0 ||
posMotor1 >= 1900 ){
     digitalWrite(brakeMotor1,HIGH);
     delay(500);
     analogWrite(pwmMotor1,0);
   }
}

void con2(){
  if(dir2==HIGH)
    posMotor2++;
    else
    posMotor2--;
    if(posMotor2==pos2 || calibMotor2 == HIGH){
     digitalWrite(brakeMotor2,HIGH);
     analogWrite(pwmMotor2,0);
    }
}


void con3(){
  if(dir3==HIGH)
    posMotor3++;
    else
    posMotor3--;
    if(posMotor3==pos3 || calibMotor3 == HIGH){
     digitalWrite(brakeMotor3,HIGH);
     analogWrite(pwmMotor3,0);
    }
}

void con4(){
  if(dir4==HIGH)
    posMotor4++;
    else
    posMotor4--;
    if(posMotor4==pos4 || calibMotor4 == HIGH){
     digitalWrite(brakeMotor4,HIGH);
     analogWrite(pwmMotor4,0);
    }
}

void contmanual()
{
  if (digitalRead(controlweb)==HIGH)
  {
  varpwm=round(analogRead(pwmin)/4);
 if (digitalRead(bot1)==HIGH){
   digitalWrite(dirMotor1,digitalRead(botdir));
   dir1= digitalRead(botdir);
   if((dir1 == LOW && digitalRead(calibMotor1) == LOW) || (dir1 == HIGH &&
posMotor1 < 1750)) {
     digitalWrite(brakeMotor1,LOW);
     analogWrite(pwmMotor1,varpwm);
     if (digitalRead(calibMotor1) == HIGH && dir1 == LOW)
     {posMotor1=0;}
   }
    Serial.print("bot1 = ");
    Serial.println(digitalRead(bot1));
    Serial.print("direccion = ");
    Serial.println(digitalRead(botdir));
```

```
    Serial.print("pwmentrada = ");
    Serial.println(varpwm);
  Serial.print("posicion 1= ");
  Serial.println(posMotor1);

} else if (digitalRead(bot2)==HIGH){
  digitalWrite(dirMotor2,digitalRead(botdir));
  dir2= digitalRead(botdir);
  if((dir2 == LOW && digitalRead(calibMotor2) == LOW) || (dir2 == HIGH &&
posMotor2 < 2*131)) {
     digitalWrite(brakeMotor2,LOW);
     analogWrite(pwmMotor2,varpwm);
     if (digitalRead(calibMotor2) == HIGH)
     {posMotor2=0;}
   }
  Serial.print("bot2 = ");
  Serial.println(digitalRead(bot1));
  Serial.print("direccion = ");
  Serial.println(digitalRead(botdir));
  Serial.print("pwmentrada = ");
  Serial.println(varpwm);
  Serial.print("posicion 2= ");
  Serial.println(posMotor2);

}  else if (digitalRead(bot3)==HIGH){
  digitalWrite(dirMotor3,digitalRead(botdir));
  dir3= digitalRead(botdir);
  if((dir3 == LOW && digitalRead(calibMotor3) == LOW) || (dir3 == HIGH &&
posMotor3 < 4*100)) {
     digitalWrite(brakeMotor3,LOW);
     analogWrite(pwmMotor3,varpwm);
     if (digitalRead(calibMotor3) == HIGH)
     {posMotor3=0;}
   }
  Serial.print("bot3 = ");
  Serial.println(digitalRead(bot3));
  Serial.print("direccion = ");
  Serial.println(digitalRead(botdir));
  Serial.print("pwmentrada = ");
  Serial.println(varpwm);
  Serial.print("posicion 3= ");
  Serial.println(posMotor3);
}  else if (digitalRead(bot4)==HIGH){
  digitalWrite(dirMotor4,digitalRead(botdir));
  dir4= digitalRead(botdir);
  int cm4;
  //if (digitalRead(calibMotor4) == HIGH))
  //cm4=HIGH;

  if((dir4 == LOW && digitalRead(calibMotor4) == LOW && posMotor4>0) ||
(dir4 == HIGH && posMotor4 < 4*100)) {
     digitalWrite(brakeMotor4,LOW);
     analogWrite(pwmMotor4,(varpwm));
     if (digitalRead(calibMotor4) == HIGH)
     {posMotor4=0;}
   }
  Serial.print("bot4 = ");
  Serial.println(posMotor4);//digitalRead(bot1));
  Serial.print("direccion = ");
  Serial.println(digitalRead(botdir));
  Serial.print("pwmentrada = ");
  Serial.print("posicion 4= ");
  Serial.println(posMotor4);
```

```
  Serial.println(digitalRead(calibMotor4));}
  else if(digitalRead(bot5)==HIGH){
    pinza.write(map(varpwm,0,255,75,255));
    Serial.print("bot5 = ");
    Serial.println(digitalRead(bot5));
    Serial.print("pinza = ");
    Serial.println(varpwm);}
  else
  {stopmotores();}
  }
  else
  {
  if (Serial.available() > 0)
  {
    while (Serial.available() > 0) {
      inString +=(char) Serial.read();
      delay(250);
    }
      Serial.print("Ingrese Coordenada");
//      Serial.println(inString.toInt());

      if (cont==1)
    {
      Serial.print("Coordenada X:");
      xcor=inString.toInt();
      Serial.println(xcor);
      cont++;
    } else if(cont==2)
    {
      Serial.print("Coordenada Y:");
      ycor=inString.toInt();
      Serial.println(ycor);
      cont++;
    }else if(cont==3)
    {
      Serial.print("Coordenada Z:");
      zcor1=inString.toInt();
      Serial.println(zcor);
      cont=1;
      calcang(xcor,ycor,zcor1);
    }
  }
      inString = "";

  }
}

void stopmotores()
{
digitalWrite(brakeMotor1,HIGH);
analogWrite(pwmMotor1,LOW);
digitalWrite(brakeMotor2,HIGH);
analogWrite(pwmMotor2,LOW);
digitalWrite(brakeMotor3,HIGH);
analogWrite(pwmMotor3,LOW);
digitalWrite(brakeMotor4,HIGH);
analogWrite(pwmMotor4,LOW);
}

void calcang(int xcor, int ycor, int zcor1){
  zcor=zcor1-l1;  //resto offset en z
  Serial.println("Coordenadas:");
    Serial.println(xcor);
```

```
    Serial.println(ycor);
    Serial.println(zcor1);

  if (pow(xcor,2) + pow(ycor,2) + pow(zcor,2) <= 900 && pow(xcor,2) +
pow(ycor,2) + pow(zcor,2) >= 100){//limite punto lejano max
    cosq3=(pow(xcor,2) + pow(ycor,2) + pow(zcor,2) - pow(l2,2) -
pow(l3,2))/(2*l2*l3);//calculo de ang theta3
    q3=atan2(-sqrt(1-pow(cosq3,2)),cosq3);//procesamiento para incluir
valores negativos
    if(q3<=0 && q3>=-pi/2){//restriccion theta3 solo entre 0 y -90
      beta=atan(sin(q3)/((l2/l3)+cos(q3)));//calculo theta2=alpha-beta
      if (beta<=0){//beta solo negativo posicion solo codo arriba
        alpha=atan(zcor/sqrt(pow(xcor,2)+pow(ycor,2)));//componente alpha
para theta2
        q2=alpha-beta;//calculo theta2
        if (q2<=pi/4 && q2>=0){//limite theta2 solo entre 0 y 45
        //asignacion de valores a variables finales
          float theta1deg, theta2deg, theta3deg;
          theta1deg = round(atan2(ycor,xcor)*180/pi);
          theta2deg = round(q2*180/pi);
          theta3deg = round(q3*180/pi);

          Serial.println("angulos");
          Serial.print("theta1: ");
          Serial.println(theta1deg);
          Serial.print("theta2: ");
          Serial.println(theta2deg);
          Serial.print("theta3: ");
          Serial.println(theta3deg);
          if (theta1deg<=0)
          theta1deg=-theta1deg;
          else
          theta1deg=360-theta1deg;


          if ((int)theta1deg != theta1){
          theta1 = (int)theta1deg;
          moverMotor1(map(theta1,0,360,0,15*131));
          }
          theta3 = (int)theta3deg;
          moverMotor3(map(90-theta3,0,90,0,440));//2*100));
          theta2 = (int)theta2deg;
          moverMotor2(map(45-theta2,0,45,0,330));//3*100));
        }
      }
    }
  }
}
```

B.2. Código del control web mediante coordenadas generalizadas

```
<!DOCTYPE html>
<html>
<title>
      Control del brazo coordenadas rectangulares
</title>
<head>
      <font color="#00FF00" size="20">Control del brazo coordenadas
rectangulares</font>
      <!--Javascript para validación-->
      <script type="text/javascript">
            function updateTextInput(val,id) {
                  document.getElementById(id+'Text').value=val;
```

```
		}
		function validateForm() {
			var x = document.forms["Coordenadas"]["xcor"].value;
			if(x<-30 || x>30 || x=="" || x==null) {
				alert("El valor ingresado no es válido para X");
				return false;
			}
			var y = document.forms["Coordenadas"]["ycor"].value;
			if(y<-30 || y>30 || y=="" || y==null) {
				alert("El valor ingresado no es válido para Y");
				return false;
			}
			var z = document.forms["Coordenadas"]["zcor"].value;
			if(z<-5 || z>36 || z=="" || z==null) {
				alert("El valor ingresado no es válido para Z");
				return false;
			}
			if(x*x + y*y + (z-15)*(z-15) < 100 || x*x + y*y + (z-
15)*(z-15) >900){
				alert("La combinación de puntos XYZ no es válida
x^2+y^2+z^2 debe estar entre 100 y 900");
				return false;
			}
		}
	</script>
	<style type="text/css">
		form {
			background:#00AAAA;
			position:relative;
			border-radius:10px;
			font-family:verdana;
			font-size:15;
			color:brown;
		}
		input[type=number] {
			position:absolute;
			left:20px;
			border-radius:5px;
		}
		input[type=checkbox] {
			position:relative;
			left:0px;
		}
		p:before {
			content: "\00a0 ";
			}
		body {
			background:black;
		}
		input[type=submit] {
			background:blue;
			color:white;
			border-radius:5px;
			border-color:blue;
			position:relative;
			left:80px;
			bottom:20px;
		}
		input[type=submit]:hover {
			background:lime;
			color:white;
			border-radius:5px;
			border-color:lime;
```

```
            }
            input[type=text] {
                  position:relative;
                  left:20px;
                  border:none;
                  color:brown;
                  background:transparent;
                  cursor:none;
                  font-size:15px;
            }
      </style>
</head>
<body>
      <form name="Coordenadas" onsubmit="return validateForm()"
method="get">
            <br>
            <p>X</p> <input id="XCor" name="xcor" type="number" value="" />
            <br>
            <p>Y</p> <input id="YCor" name="ycor" type="number" value="" />
            <br>
            <p>Z</p> <input id="ZCor" name="zcor" type="number" value="" />
            <br>
            <p>Orientacion</p> <input id="Orie" name="orie" type="range"
min="0" max="90" step="1" value="0"
onchange="updateTextInput(this.value,this.id);" /> <input id="OrieText"
type="text" value="" />
            <br>
            <p>Pinza <input id="Pinza" name="pinza" type="checkbox" /> </p>
            <br>
            <input type="submit" value="Enviar" />
      </form>
</body>
</HTML>
```