ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA MECÁNICA

DESARROLLO DE CONTROLADORES POR SOFTWARE PARA UN MANIPULADOR USANDO OROCOS

“PROYECTOS DE INVESTIGACIÓN Y DESARROLLO”

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN DISEÑO PRODUCCIÓN Y AUTOMATIZACIÓN INDUSTRIAL

ING. CHRISTIAN JAVIER ACOSTA RUALES
christian.acosta02@epn.edu.ec

DIRECTOR: ING. ANA VERONICA RODAS BENALCÁZAR
ana.rodas@epn.edu.ec

CODIRECTOR: ING. OSCAR IVÁN ZAMBRANO OREJUELA, M.Sc.
ivan.zambrano@epn.edu.ec

Quito, abril 2019

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por **Christian Javier Acosta Ruales**, bajo mi supervisión.

Ing. Ana Rodas
DIRECTOR DE PROYECTO

M.Sc. Ing. Iván Zambrano
CO-DIRECTOR DE PROYECTO

DECLARACIÓN

Yo, Christian Javier Acosta Ruales, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Christian Javier Acosta R.

DEDICATORIA

A Dios.

Por todas sus bendiciones, permitiéndome llegar hasta este punto y lograr mis objetivos.

A Micaela y Danna.

Por ser mi motivación para seguir adelante y me impulsan cada día a superarme y tratar de ser un ejemplo a seguir.

A Mayra

Por su apoyo y paciencia en este proyecto.

A mis padres

Por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo en todas las etapas de mi vida.

AGRADECIMIENTO

Mi más grande y sincero agradecimiento a la Ing. Ana Rodas quien con su dirección, conocimiento, enseñanza y colaboración permitió el desarrollo de este trabajo, y a todos quienes con su apoyo me permitieron llegar a culminar con éxito este proyecto.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo es parte del proyecto de investigación PIMI 15-04 "Control adaptativo basado en inteligencia artificial aplicado a un sistema mecatrónico fundado en un robot paralelo para la diagnosis y rehabilitación", realizado por la facultad de Ingeniería Mecánica en colaboración con la Universidad Politécnica de Valencia, en el cual se busca implementar controladores para un robot paralelo mediante la programación en software libre basado en la utilización del software OROCOS. El controlador se lo implementó a un manipulador antropomórfico de tres grados de libertad, el cual traslada objetos metálicos de forma circular de color rojo o verde, que estarán ubicados en puntos específicos del área de trabajo, a su respectivo contenedor dependiendo del color del objeto. Para la implementación se utilizó una tarjeta Arduino como tarjeta de adquisición de datos, para controlar los motores, e ingresar la señal de un pulsador que da el inicio del proceso, Además se utilizó una cámara web para identificar los colores de los objetos.

Palabras clave: Manipulador, OROCOS, ROS, Software Libre

ABSTRACT

The present work is part of the research project PIMI 15-04 “Control adaptativo basado en inteligencia artificial aplicado a un sistema mecatrónico fundado en un robot paralelo para la diagnosis y rehabilitación”, carried out by the Facultad de Ingeniería Mecánica in collaboration with the Universidad Politécnica de Valencia, seeks to implement controllers for a manipulator based on open source software by using mainly OROCOS. The controller is applied on anthropomorphic manipulator of three degrees of freedom which transfers metallic red or green objects of circular shape. These objects are located in specific points of the work area, so the manipulator moves them to their respective container depending on the color of the object. For the implementation, we use an Arduino card, to acquire data control the motors, and read the signal of a button that gives the beginning of the process. Also, a web camera is used to identify the colors of the objects.

Keywords: Manipulator, OROCOS, ROS, free software.

“DESARROLLO DE CONTROLADORES POR SOFTWARE PARA UN MANIPULADOR USANDO OROCOS”

INTRODUCCIÓN

El software libre es una opción para la programación de robots, por lo que, en la comunidad de investigación de robótica, se ha convertido en un estándar, siendo una plataforma de desarrollo muy utilizada en el diseño y control de manipuladores industriales. Permite que tanto el hardware y software puedan ser modificados sin dificultad por el usuario, incrementando la capacidad y actualización del producto.

Cabe mencionar que el Ecuador, por decreto presidencial, tiene como política de Estado el uso de software libre para alcanzar soberanía y autonomía tecnológica.

Por estas razones en este proyecto se busca desarrollar los controladores para un manipulador mediante la programación en software libre, tanto en el sistema operativo (Linux) como los middlewares de control de robot desarrollado en OROCOS y ROS (Robotic Operatting System).

Al utilizar software libre se puede tener acceso a todo el código y en caso de querer adaptar el software para un propósito específico, no existen grandes restricciones, pudiendo ser utilizado y mejorado en proyectos futuros como el proyecto de investigación aprobado de la Escuela Politécnica Nacional que tiene por título “Control adaptativo basado en inteligencia artificial aplicado a un sistema mecatrónico fundado en un robot paralelo para la diagnosis y rehabilitación”

Objetivo general Desarrollar controladores en tiempo real por medio de componentes de software para integración del hardware de un manipulador con el middleware OROCOS.

Objetivos específicos

- Analizar el middleware OROCOS
- Determinar la cinemática directa del manipulador.
- Diseñar controladores en tiempo real basado en OROCOS
- Implementar el sistema completo: computador-tarjeta de adquisición de datos – manipulador.
- Realizar pruebas al sistema y compararlas con datos experimentales.

1. MARCO TEÓRICO

Hoy en día, los robots manipuladores están presentes en todas las instalaciones de producción moderna industriales. Por ello, a la robótica manipuladora se la considera una disciplina decisiva en el sector industrial. Además, en un futuro no muy lejano los robots manipuladores pasarán a ser también esenciales en la vida cotidiana de la sociedad. Así, existe una demanda creciente de aplicaciones con robots manipuladores con requisitos de software como son la reutilización, flexibilidad y adaptabilidad [1].

1.1. Robótica

La robótica es la ciencia que estudia el diseño y la implementación de robots, conjugando múltiples disciplinas, como mecánica, electrónica, informática, inteligencia artificial, ingeniería de control, entre otras [2].

Existen muchas definiciones de robot, cada una depende del enfoque del autor, pero según la Asociación Japonesa de Robótica Industrial (JIRA), los robots son dispositivos capaces de moverse de modo flexible, análogo al que poseen los organismos vivos, con o sin funciones intelectuales, lo que permite la realización de operaciones en respuesta a órdenes recibidas por humanos.

En la actualidad los robots han dado lugar a procesos de producción mucho más eficientes y a un alto grado de calidad en los productos, con una reducción significativa en los procesos donde exista desperdicio de material, debido al alto grado de precisión que pueden tener los robots, ya sea para el caso de ensamblaje, soldado o apilado de piezas, De tal manera que, la robótica es ya el principal motor de competitividad y flexibilidad en las industrias de fabricación a gran escala y es un sinónimo de progreso y desarrollo tecnológico. Los países y las empresas que cuentan con una fuerte presencia de robots consiguen altos niveles de competitividad y productividad, en los países más desarrollados, las inversiones en tecnologías robóticas han crecido de forma significativa y muy por encima de otros sectores [3].

1.2. Clasificación de robots

Existen varias maneras de clasificar a los robots sea por su aplicación, arquitectura, su generación, nivel de inteligencia, nivel de control, su nivel de lenguaje de programación, por su tipo, entre otras. En este caso se revisará la siguiente clasificación:

1.2.1. Robots móviles

Los robots móviles son dispositivos de transporte automático, es decir, una plataforma mecánica que contiene un sistema de locomoción capaz de navegar a través de un determinado ambiente de trabajo, dotado de cierto nivel de autonomía para su desplazamiento portando cargas. Sus aplicaciones pueden ser muy variadas y siempre están relacionadas con tareas que normalmente son riesgosas o nocivas para la salud humana, en áreas como la agricultura, en el transporte de cargas peligrosas o en tareas de exploración [4].

Los robots móviles a su vez pueden dividirse en terrestres, los cuales pueden tener ruedas, patas, o desplazarse arrastrándose mediante movimientos de contracción-expansión, acuáticos que pueden ser marinos o submarinos, y aéreos que pueden desplazarse por el aire.

1.2.2. Robots Industriales

Se los conoce también como manipuladores, según la Federación Internacional de Robótica bajo la norma ISO/TR 8373, un robot manipulador se define como: “Una máquina manipuladora con varios grados de libertad controlada automáticamente, reprogramable y de múltiples usos, pudiendo estar en un lugar fijo o móvil para su empleo en aplicaciones industriales”[5].

Los robots manipuladores pueden ser servo-controlados, reprogramables, polivalentes, capaces de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables, para la ejecución de tareas variadas. Normalmente tiene la forma de uno o varios brazos terminados en una muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Normalmente su uso es el de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material.

Los robots industriales son construidos de acuerdo con las tareas de manipulación de piezas o herramientas que se van a realizar, por esta razón a los robots industriales a su vez se los puede clasificar según la estructura de sus ejes principales que resultan de las distintas composiciones mecánicas que tienen los tres ejes principales del robot ya que los ejes secundarios o de muñeca siempre son similares.

- **Manipulador angular o antropomórfico:** Como se puede observar en la Figura1.1, estos robots, en su construcción, se asemejan al brazo humano; así, el primer eje principal correspondiente al tronco se encuentra sobre la placa base y produce un movimiento giratorio (cintura); el segundo y tercer eje forman el hombro y codo. Esta configuración proporciona gran accesibilidad y maniobrabilidad, ocupando poco espacio respecto al área de trabajo.

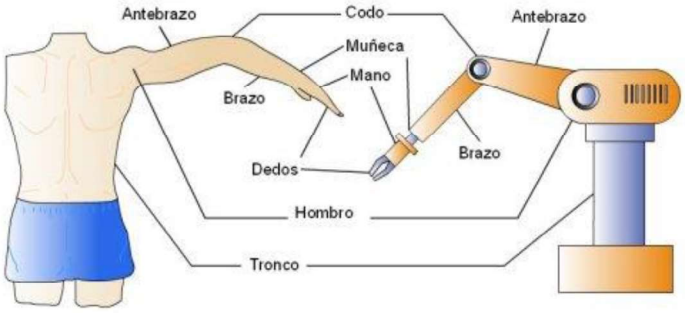


Figura 1.1. Robot angular.
(Fuente: https://sites.google.com/site/roboticclasshn/robots)

- **Manipulador paralelo:** son cadenas cinemáticas cerradas, en la que la herramienta terminal está conectado a la base mediante varias cadenas cinemáticas seriales independientes, como se puede observar en la Figura 1.2.

Figura 1.2. Robot paralelo.
(Fuente: http://new.abb.com/products/robotics/industrial-robots/irb-360)

- **Manipulador cartesiano:** Como se puede observar en la Figura1.3, posee tres articulaciones prismáticas, los ejes son ortogonales entre sí y los desplazamientos sobre ellos dan las coordenadas cartesianas X, Y, Z. Son rápidos, precisos, de fácil control, amplia zona de trabajo y elevada capacidad de carga.

Figura 1.3. Robot cartesiano.

(Fuente: https://www.logismarket.com.ar/general-automation-isgroup/robot-cartesiano-2/1978679069-1244808378-p.html)

- **Manipulador articulado horizontal (SCARA):** Se le denomina SCARA ya que es el acrónimo de Selective Compliant Assembly Robot Arm, que en español vendría a significar “brazo de robot acodado horizontal para montaje”. Como se puede observar en la Figura1.4, es un robot articulado de cuatro grados de libertad con posicionamiento horizontal. Posee dos ejes principales que proporcionan movimientos rotativos en un mismo plano para obtener un círculo, el tercero desplaza ese círculo en línea recta para engendrar un volumen que tendrá forma cilíndrica y el cuarto eje es vertical lineal que es la dirección más fácil para la inserción o montaje de componentes.

Figura 1.4. Robot scara.

(Fuente: http://www.toshiba-machine.co.jp/en/NEWS/product/20110523.html)

- **Manipulador esférico o polar:** Como se puede observar en la Figura1.5, posee dos ejes rotativos perpendiculares para poder realizar circunferencias, tanto horizontales como verticales, obteniéndose con ello una superficie esférica, la cual permite el desplazamiento del tercer eje en línea recta convirtiéndose este movimiento en el radio de dicha esfera.

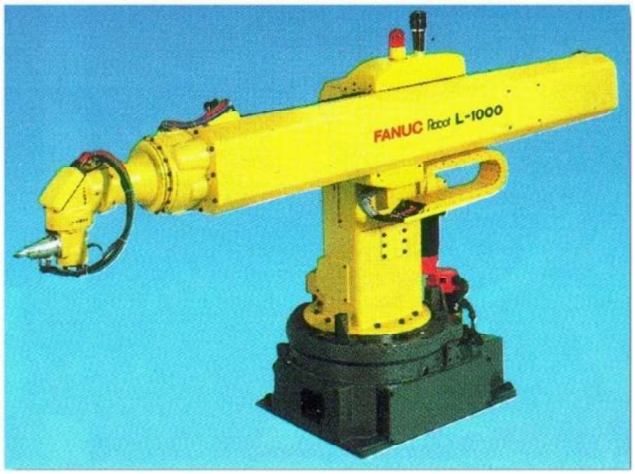


Figura 1.5. Robot esférico.

(Fuente: http://www.udesantiagovirtual.cl/moodle2/mod/book/view.php?id=24911&chapterid=226)

- **Manipulador Cilíndrico:** Como se puede observar en la Figura1.6, dispone de un giro en la base y dos desplazamientos perpendiculares entre sí, para determinar la posición de los puntos por medio de coordenadas cilíndricas.

Figura 1.6. Robot cilíndrico.

(Fuente: http://www.geocities.ws/opayan/TeoriaTipos.htm)

Los robots para su funcionamiento combinan tecnologías mecánicas, electrónicas y de programación logrando ejecutar tareas sin ayuda humana. El controlador es el componente del robot que procesa la información captada por los sensores, y según las instrucciones del programa que almacena, regula el movimiento de los motores u otros dispositivos de salida.

Actualmente existen muchos programas que permiten generar las instrucciones que controlan los movimientos y funciones de los robots como ROS, ORCA, MIRO, URBI, RT-MIDDLEWARE, entre otros. Este proyecto se basa en el uso del software ROS.

1.3. Robot Operating System (ROS)

ROS (Robot Operating System) es un framework para la escritura de software de robots. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear comportamientos para robots complejos y robustos a través de una amplia variedad de plataformas robóticas [6].

ROS es un meta sistema operativo de código abierto mantenido por la Open Source Robotics Foundation (OSRF). Proporciona los servicios como la abstracción de hardware, control de dispositivos a bajo nivel, implementación de utilidades comunes, paso de mensajes entre procesos, librerías, herramientas de visualización, comunicación por mensajes, administración y gestión de paquetes.

Las principales ventajas que ofrece ROS son:

Computación distribuida: Los sistemas de robótica modernos están basados en software que usan una infinidad de procesadores y ordenadores distintos, necesitando un sistema de comunicación entre ellos. ROS contiene mecanismos de comunicación que cubren estas necesidades [7].

Reutilización de Software: ROS promueve la reutilización de código con diferente hardware, proporcionando una gran cantidad de bibliotecas disponibles para la comunidad, como SLAM 2D basado en laser, SLAM visual, y Nubes de Puntos basadas en el reconocimiento de objetos 3D, así como herramientas para la visualización en 3D, entre otros. Como resultado, el tiempo total invertido en el desarrollo de software se

reduce en gran medida debido a la reutilización de código y de abstracción de hardware, y la integración de los robots y los sensores en ROS resulta beneficiosa[8].

Análisis/pruebas rápidas: Uno de los mayores retos que presenta el desarrollo de software para robots es la complejidad a la hora de realizar pruebas. La disponibilidad de robots a la hora de realizar pruebas no siempre es posible. ROS proporciona sistemas de simulación que sustituyen al hardware/software normalmente requerido y permite la reproducción de datos de sensores y otro tipo de mensajes [7].

Para poder entender la arquitectura de ROS, como se muestra en la Figura 1.7, se puede dividir en:

- Nivel del sistema de archivos (Filesystem Level)
- Nivel de computación gráfico (Computation Graph level)
- Nivel de comunidad (Comunity level)

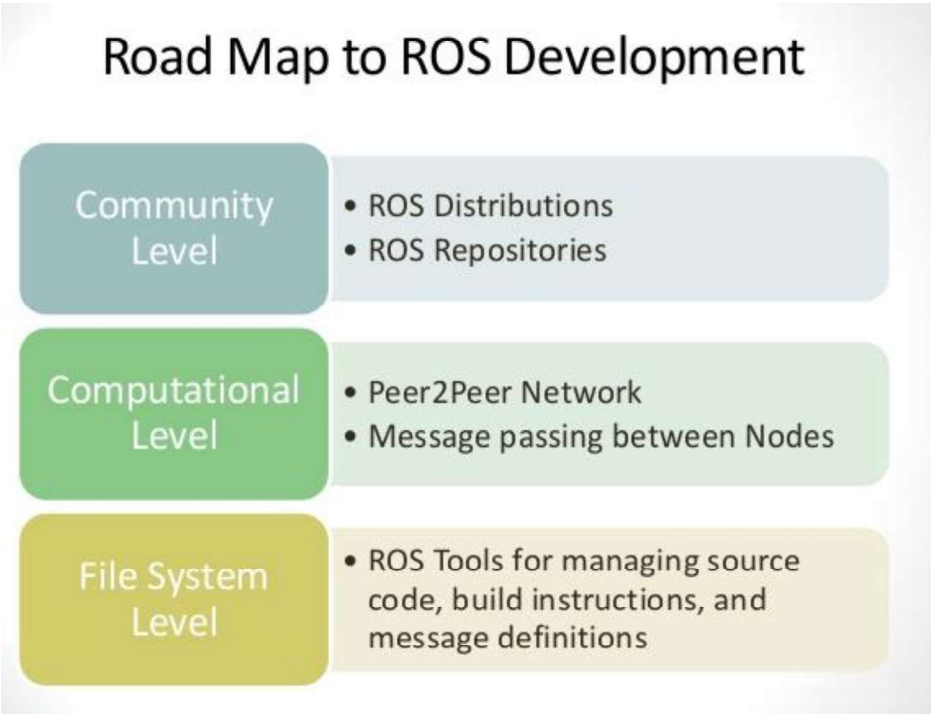


Figura 1.7. ROS.

(Fuente: An Introduction To Open-Source Robotic Framewok, VIT University - India, Narrendar R. C.)

1.3.1. Nivel del sistema de archivos (Filesystem Level)

Corresponden al grupo de conceptos usados para explicar cómo está formado ROS, estructura de carpetas y archivos necesarios para funcionar, como se puede ver en la Figura1.8.

ROS se encuentra dividido en carpetas, y cada una de estas carpetas tiene archivos que definen sus funcionalidades [9].

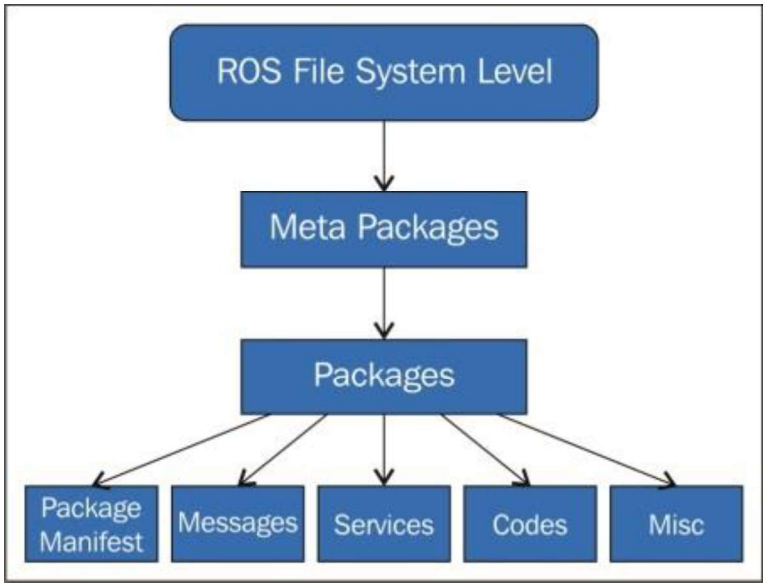


Figura 1.8. Nivel del sistema de archivos ROS.
(Fuente: Lentin Josep, Mastering ROS for robotics Programming, 2015)

Paquetes (Packages): Los paquetes ROS son la unidad más básica del software ROS. Contiene el proceso de ejecución de ROS (nodos), bibliotecas, archivos de configuración, etc., que se organizan juntos como una sola unidad [9], contiene una estructura formada por archivos y carpetas compuesta por:

bin/: Carpeta donde los programas son almacenados después de su compilación.

include/package_name/: Carpeta donde se encuentran las cabeceras (.h) de las librerías usadas.

msg/: Carpeta donde se encuentran los mensajes estándar y los creados.

src/package_name/: Carpeta donde se aloja el código fuente del programa.

srv/: Carpeta que contiene los tipos de servicios.

ROS dispone de herramientas para facilitar la creación, modificación y manejos de los paquetes:

rospack: Comando para buscar u obtener información de paquetes.
roscreate-pkg: Comando para crear nuevos paquetes.
rosmake: Comando para compilar un paquete.
rosdep: Comando para instalar dependencias necesitadas por el paquete.
rxdeps: Comando para mostrar dependencias usadas por el paquete.
roscd: Comando para cambiar de directorio.
rosed: Comando para editar archivos.
roscp: Comando para copiar archivos.
rosls: Comando para listar archivos de un paquete.

Meta Paquetes (Meta Packages): Los paquetes crean pequeñas colecciones de código para su reusabilidad, los paquetes se organizan en Meta-Paquetes. Estos están diseñados para mejorar el proceso de compartir código. Un Meta-Paquete es una estructura básica formada por carpetas y archivos que puede ser creados manualmente o con la ayuda de roscreate-stack [7].

Manifiesto de Paquetes (Package Manifest): Proporcionan información sobre un paquete, licencias, dependencias, marcas de compilación, etc. Un Package Manifest se gestiona con un archivo llamado package.xml.

Mensajes (Messages): Los mensajes en ROS son un tipo de información que un proceso envía a otro proceso [9], Un mensaje está compuesto por dos partes principales: **Campo** con el tipo de dato que va a ser transmitido en el mensaje, **constantes** que definen el nombre del campo.

Los tipos de datos en ROS podemos observar en la Figura 1.9.

Primitive type	Serialization	C++	Python
bool	Unsigned 8-bit int	uint8_t	bool
int8	Signed 8-bit int	int8_t	int
uint8	Unsigned 8-bit int	uint8_t	int
int16	Signed 16-bit int	int16_t	int
uint16	Unsigned 16-bit int	uint16_t	int
int32	Signed 32-bit int	int32_t	int
uint32	Unsigned 32-bit int	uint32_t	int
int64	Signed 64-bit int	int64_t	long
uint64	Unsigned 64-bit int	uint64_t	long
float32	32-bit IEEE float	float	float
float64	64-bit IEEE float	double	float
string	ASCII string (4-bit)	std::string	string
time	Secs/nsecs signed 32-bit ints	ros::Time	rospy.Time
duration	Secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration

Figura 1.9. Tipo de dato en ROS.
(Fuente: Introducción a ROS en Raspberry Pi, Universidad Oberta de Catalunya Javier Gutiérrez Pérez)

Servicios (Service): Las descripciones de servicio, almacenadas en my_package / srv / MyServiceType.srv, define las estructuras de petición y respuesta usadas por los servicios en ROS.

ROS utiliza un lenguaje de descripción de servicios simplificado para describir los tipos de servicios ofertados, permitiendo crear un nivel sobre los tipos de mensaje estándar para habilitar un sistema de comunicación basado en petición/respuesta entre los nodos. Dichos servicios se encuentran en los archivos .srv en la carpeta srv/ de cada paquete [10].

La herramienta **rossrv** proporciona información sobre los servicios, los paquetes que contienen dicho servicio y permite localizar archivos que usan ese servicio.

Código (Codes): Aquí es donde se encuentra el código fuente de los programas de ROS, src/package_name/.

Misceláneos (Misc): la mayoría de los paquetes de ROS se mantienen utilizando un Sistema de control de versiones (VCS) como Git, subversion (svn), mercurial (hg), etc. La colección de paquetes que comparten un VCS común puede denominarse

repositorios. El paquete en los repositorios puede ser lanzado usando una herramienta de automatización de lanzamiento llamada "bloom".

1.3.2. Nivel Computacional Gráfico (Computation Graph level)

ROS crea una red en la que conecta todos los procesos de tal forma que cualquier nodo de esta red puede interactuar con otros nodos, obtener la información respecto a la información que envían y transmitir información a dicha red [10]. La estructura se muestra en la Figura 1.10.

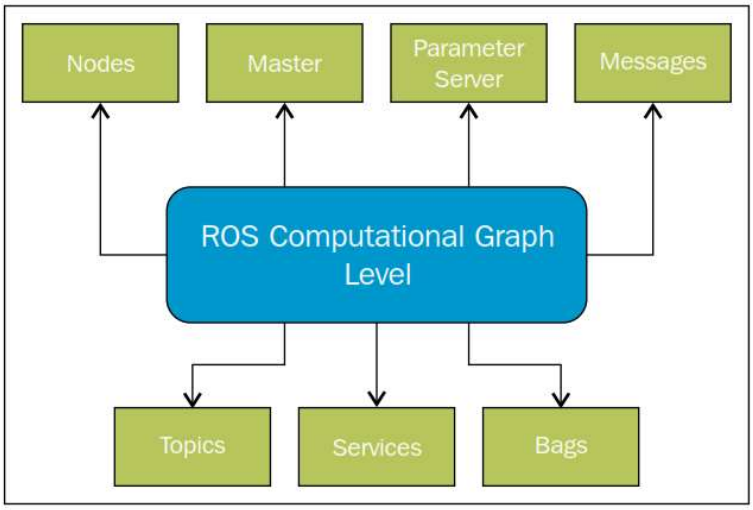


Figura 1.10. Niveles de computación gráfico en ROS.
(Fuente: Introducción a ROS en Raspberry Pi, Universidad Oberta de Catalunya
Javier Gutiérrez Pérez)

Nodos (Nodes): Son procesos ejecutables donde se realiza el trabajo deseado, que pueden comunicarse con otros procesos usando tópicos u otros servicios. Para que un proceso pueda comunicarse con otros nodos, es necesario la creación de un nodo con este proceso y luego se lo puede conectar a la red de ROS. Normalmente un sistema tendrá varios nodos para controlar distintas funciones. Para la implementar un nodo se puede utilizar librerías como rosccp para C++ o rospy para Phyton [10].

Las herramientas que proporciona Ros para el manejo de los nodos son :

rosnode info node: Proporciona información del nodo.
rosnode kill node: Termina la ejecución de un nodo.

rosnode list: Proporciona una lista de los nodos activos.
rosnode machine hostname: Proporciona una lista de los nodos activos en un ordenador en específico.
rosnode ping node: Prueba la conectividad con el nodo.
rosnode cleanup: Elimina la información relativa a nodos los cuales son inaccesibles.

Topicos (Topics): Los tópicos son buses de datos usados por los nodos para transmitir información. Cuando un nodo envía información, se dice que ese nodo está publicando un tópico. Un nodo puede estar suscrito a ese tópico para recibir un mensaje enviado por otro nodo. Cada mensaje debe de tener asignado un nombre para que pueda ser canalizado por ROS [7].

Un tópico puede tener varios suscriptores, sin la necesidad de que haya una conexión directa entre los nodos ya que el consumo y la producción de información está desvinculada [9].

La transmisión de información se realiza usando TCP/IP y UDP. Cuando la comunicación está basada en TCP/IP se conoce como TCPROS. En cambio, cuando la comunicación está basada en UDP se conoce como UDPROS.

ROS proporciona rostopic, una herramienta que ofrece información sobre los nodos o la información publicada en la red de ROS. Esta herramienta se puede usar con los siguientes parámetros [7]:

rostopic bw /topic: Muestra el ancho de banda usado por el tópico.
rostopic echo /topic: Muestra los mensajes de ese tópico.
rostopic find messag_type: Busca tópicos por tipo.
rostopic hz /topic: Muestra la frecuencia de publicación del tópico.
rostopic info /topic: Muestra información relativa al tópico.
Rostopic list: Muestra información de tópicos activos.
rostopic type /topic: Muestra información del tipo del nodo y del tipo de mensajes que publica.

Servicios (Services): Cuando se necesita comunicarse con los nodos y recibir una respuesta, no se puede hacer mediante temas, se necesita hacerlo con servicios. Los

servicios son desarrollados por los usuarios, ya que no existen servicios estándar. Dichos servicios deben asociarse a un tipo que es fijo para todos los servicios .svr y un nombre que identifica al servicio [10].

Las herramientas para trabajar con servicios son rossrv y rosservice, en el caso de rossrv, proporciona información sobre los servicios, exactamente como hace rosmsg. Mientras que rosservice puede mostrar y almacenar servicios, soportando los siguientes comandos [9]:

Rosservice call /service args: Invoca el servicio con los argumentos proporcionados.
Rosservice find msg-type: Encuentra servicios por tipo.
Rosservice info /service: Muestra información del servicio.
Rosservice list: Muestra lista de servicios activos.
Rosservice type /service: Muestra el tipo de servicio.

Master: Permite el registro y la búsqueda de nodos. Sin este elemento la comunicación con otros nodos, servicios y mensajes no es posible.

El ROS Master proporciona el nombre y registra los servicios del resto de los nodos del sistema, además de supervisar el tráfico de tópicos y servicios. Su función principal es la de habilitar los nodos y localizarse unos a otros hasta que exista una comunicación de pares [7].

Servido de parámetros (Parameter Server): Este parámetro da la posibilidad de tener información almacenada en una ubicación central a través de llaves. Además, permite modificar la configuración de los nodos sin la necesidad de compilar [9].

Mensajes (Messages): Un mensaje contiene la información que se quiere enviar a otros nodos, los cuales son publicados por tópicos. El mensaje tiene una estructura básica formada por tipos de dato estándar o tipos de datos creados por el usuario [11].

La herramienta rosmsg permite obtener información de los mensajes. Puede ser usada con los siguientes parámetros:

rosmsg show: Muestra información del mensaje.

rosmsg list: Muestra lista de mensajes.
rosmsg package: Muestra lista de mensajes del paquete.
rosmsg packages: Muestra paquetes que contienen mensajes.
rosmsg users: Muestra archivos que usan el tipo de dato message.

Bags: Un "bag" es un archivo creado por ROS en el que se pueden guardar mensajes, tópicos y servicios, entre otros. Esta información se puede visualizar y reproducir a gusto del usuario. Para usar "bags", ROS proporciona las siguientes herramientas [9]:

rosbab: Usado para reproducir, grabar u otras operaciones.
rxbag: Usado para visualizar información.
rostopic: Muestra los tópicos enviados a los nodos.

1.3.3. Nivel de Comunidad (Comunity level)

La comunidad de ROS está basada en distintos recursos que permiten a distintas comunidades el intercambio de software y conocimientos [11]. Estos recursos incluyen:

Distribuciones: Las distribuciones son una colección de paquetes de metadatos que se pueden instalar. Las distribuciones de ROS mantienen versiones estables en un conjunto de software [9].

Repositorios: ROS se basa en una red de repositorios de código, donde diferentes instituciones pueden desarrollar y lanzar sus propios componentes de software de robots [10].

ROS Wiki: ROS Wiki es el foro principal para documentación e información sobre ROS. Cualquier persona puede aportar nueva documentación, corregir errores o escribir tutoriales [9].

Lista de email: Es el principal canal de comunicación sobre actualizaciones de ROS, así como un método de debate sobre temas relacionados con ROS[10].

En este proyecto se usará ROS como programa base en donde instalaremos el Toolchain OROCOS.

1.4. Open Robot Control Software (OROCOS)

"Orocos", es un framework de software en código abierto y desarrollado en C++ para la construcción de aplicaciones basadas en componentes en tiempo real en automatización y robótica.

El objetivo principal de OROCOS es desarrollar una plataforma software de control bajo una licencia libre, manteniendo la independencia sobre arquitecturas concretas, contribuir al desarrollo de interfaces de programación de arquitecturas de control [12].

Además de sus raíces en el campo de la robótica, el entorno de desarrollo de aplicaciones en tiempo real de Orocos se ha centrado en el campo del control de maquinaria.

1.4.1. Proyecto OROCOS

El proyecto Orocos, presenta un entorno de desarrollo que ofrece funcionalidad genérica para los robots y sus aplicaciones. Este proyecto nació dentro de EURON (European Robotics Network) y fue patrocinado por la Comisión Europea (EC) en el año 2000. Actualmente el proyecto Orocos continua con fondos del Centro de Tecnología Flanders Mechatronics, desarrollando el entorno de tiempo real y coordinando la integración de Orocos en máquinas industriales[13].

OROCOS proporciona las funciones principales de un middleware que son: abstracción del sistema operativo y soporte de tiempo real, servicios middleware, infraestructura de comunicación, modelo basado en componentes [12]. OROCOS trabaja con las librerías: Kinematics and Dynamics (KDL), Bayesian Filtering (BFL) y Orocos Toolchain.

1.4.2. Biblioteca Cinemática y Dinámica (KDL)

La Biblioteca de Cinemática y Dinámica (KDL) desarrolla un marco independiente de la aplicación para el modelado y cálculo de cadenas cinemáticas, permitiendo reducir el modelado y la especificación del movimiento a un problema meramente geométrico con

cálculos matemáticos, pudiéndose resolver un movimiento en base a unas especificaciones en cada una de las articulaciones [14]. Junto a la cinemática, también se incluyen parámetros para la dinámica.

KDL contiene por el momento solo solucionadores genéricos para cadenas cinemáticas, esto puede ser usado con cuidado para cada KDL. La idea detrás de los solucionadores genéricos es tener una API (Interfaz de Programación de Aplicaciones) uniforme para cada tipo de solucionador, los solucionadores son:

ChainFkSolverPos
ChainFkSolverVel
ChainIkSolverVel
ChainIkSolverPos

Se debe crear un solucionador separada para cada cadena

Un tipo específico de solucionador puede agregar algunas funciones / parámetros específicos del mismo a la interfaz, pero aún tiene que usar la interfaz genérica para su propósito principal de resolución.

La cinemática directa usa la función JntToCart para calcular los valores del espacio cartesiano a partir de los valores del espacio conjunto. La cinemática inversa utiliza la función CartToJnt para calcular los valores espaciales conjuntos a partir de los valores espaciales cartesianos.

1.4.3. Biblioteca de Filtrado Bayesiano (BFL)

La Biblioteca de Filtrado Bayesiano de Orocos (BFL) proporciona un marco independiente de aplicaciones para la inferencia en redes bayesianas dinámicas, es decir, algoritmos de estimación y procesamiento recursivo de la información basados en la regla de Bayes [12].

1.4.4. Cadena de Herramientas Orocos (BFL)

La cadena de herramientas Orocos (OROCOS Toolchain) es la herramienta principal para crear aplicaciones de robótica en tiempo real utilizando componentes de software modulares y configurables en tiempo de ejecución.

Proporciona:

- Soporte multiplataforma: Linux, Windows (Visual Studio) y Mac OS-X
- Extensiones a otros marcos robóticos: ROS, Rock, Yarp
- Generadores de código para transferir datos definidos por el usuario entre componentes distribuidos
- Componentes programables en tiempo de ejecución y en tiempo real
- Registro e informe de eventos del sistema y datos comunicados.

En este proyecto se usa OROCOS como el middleware en donde se han desarrollado una serie de componentes para el control del manipulador.

1.5. Tarjetas de adquisición de datos

Las tarjetas de adquisición de datos son dispositivos diseñados para la adquisición y medición de señales que consiste en la toma de muestras de variables físicas del mundo real (sistema analógico por lo general), para generar datos que pueden ser manipulados por un programa diseñado en un ordenador [15]. Las tarjetas de adquisición de datos además de sus entradas también disponen salidas que permiten controlar actuadores para modificar las variables con las que se está trabajando.

Existen gran cantidad de marcas y modelos de tarjetas de adquisición de datos como Arduino, National Instruments, Measurement Computing, entre otras, cada una con características distintas como cantidad de entradas y salidas análogas, cantidad de entradas y salidas digitales, tipo de conexión, resolución del convertidor AD.

La tarjeta de adquisición de datos a usar permitirá el ingreso y salida de señales al y del computador. Se obtendrán del computador las señales para el control de la posición de

cada uno de los motores y permitirá ingresar al computador las señales de pulsantes que dan la señal de inicio.

2. METODOLOGÍA

El prototipo de manipulador que se va a implementar deberá trasladar objetos metálicos de forma circular, de color rojo o verde, que estarán ubicados en puntos específicos del área de trabajo. El traslado de los objetos a su respectivo contenedor depende del color del objeto.

El sistema completo contendrá la parte mecánica del manipulador, la tarjeta de adquisición de datos y el sistema de control del manipulador el cual se lo realizará en un computador como se muestra en la Figura 2.1.

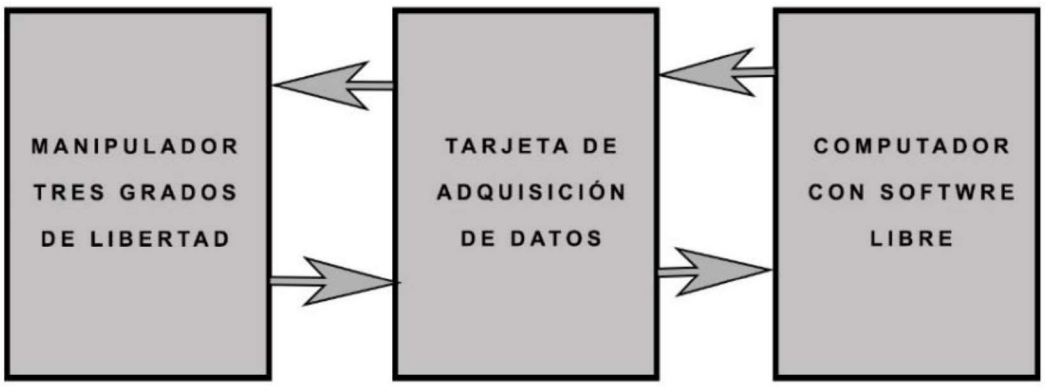


Figura 2.1.Diagrama en bloques del manipulador a implementar.
(Fuente: Javier Acosta, 2018)

Para ello se plantea construir un manipulador del tipo antropomórfico o robot angular de tres grados de libertad con la configuración típica del robot PUMA (Programmable Universal Machine for Assembly, or Programmable Universal Manipulation Arm), este tipo de manipulador es utilizado para cirugía robótica, aplicaciones coger – colocar, operación de ensamblado, manipulación de herramientas, manipulación de máquinas, por lo que es ideal para el trabajo que se pretende realizar con el mismo.

En la Figura 2.2 podemos observar el manipulador PUMA comercializado por la empresa AUTOMETION, siendo el modelo original diseñado para la línea de ensamble de GENERAL MOTORS.

Figura 2.2.Robot PUMA fabricado por la empresa UNIMATION.
(Fuente: http://www.roboticsurgerydoc.com/2016/07/22/a-brief-history-of-robotic-surgery/)

El prototipo de manipulador propuesto de tres grados de libertad, consiste en un mecanismo serial de tres eslabones rígidos conectados a través de tres articulaciones rotoides, las cuales corresponden a las articulaciones de cintura, hombro y codo.

Los eslabones que conforman el manipulador deberán ser construidos de un material liviano, pero de una buena resistencia mecánica para evitar la deformación del manipulador.

2.1. Diseño

Para el diseño y la simulación se utilizará un software de CAD (Diseño Asistido por Computador) el cual permite el diseño de objetos por computadora, y tiene varias ventajas como la facilidad para hacer modificaciones en el diseño y la posibilidad de simular el comportamiento del prototipo.

En este caso se optó por utilizar el software SolidWorks el cual permite modelar piezas y ensambles en 2D y 3D de los cuales se pueden generar tanto planos técnicos como otro tipo de información necesaria para la implementación.

El prototipo de manipulador será de uso didáctico, razón por la cual las dimensiones serán pequeñas, el área de trabajo en la que se utilizará el manipulador será una superficie rectangular de 20 cm x 30 cm, en la Figura 2.9 se muestra las partes del manipulador y dimensiones del área de trabajo.

Una de las partes más importantes son los motores, los cuales están ensamblados en los eslabones y permiten el movimiento rotacional entre eslabones, por lo que es necesario saber sus dimensiones, para poder determinar la forma y tamaño de los eslabones, los motores a ser utilizados en esta ocasión son los servomotores MG 946R.

Para la construcción de los eslabones se decidió utilizar planchas de aluminio por su bajo peso y su buena resistencia a la deformación tanto elástica como plástica. Las características mecánicas de se detallan en la Tabla 2.1.

Tabla 2.1. Resistencia mecanica de la plancha de aluminio.
Fuente:http://www.dipacmanta.com/aluminio/plancha-aluminio-lisa-astm-a1200.

Espesor mm	Resistencia Mecánica	% Elongación
0.7 - 4.00	127 – 135 kg/mm2	6.5 - 7.5

2.1.1. Segundo eslabón

Para el diseño se empezó por el segundo eslabón porque se ha decidido que en este eslabón estará ensamblados dos de los tres motores.

Este eslabón está formado por dos piezas iguales de 235 mm, separadas entre ellas 27 mm con perforación rectangulares para montar los dos motores. Tendrá otras perforaciones para reducir la cantidad de material y por consecuencia el peso.

El extremo que posee un agujero circular de 4 mm de diámetro, se lo utilizará para colocar un contrapeso que contrarreste el peso tanto de la parte delantera del segundo eslabón y el peso del tercer eslabón.

Las dos piezas que conforman el segundo eslabón estarán sujetas mediante dos tornillos y dos cilindros que evitaran que estás se separen y se muevan entre sí.

En la Figura 2.3 se muestra al segundo eslabón con todas las piezas que lo conforman.

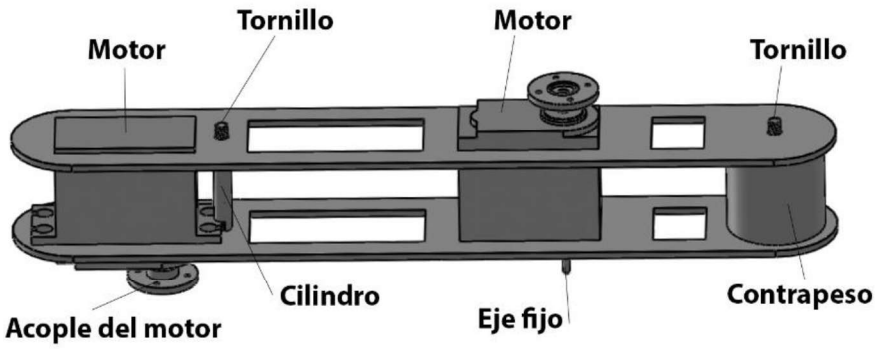


Figura 2.3. Vista en 3D del Segundo eslabón.
(Fuente: Javier Acosta, 2018)

En el Anexo 1, se puede observar el plano del segundo eslabón, con sus dimensiones.

2.1.2. Primer eslabón

Luego de tener el diseño del segundo eslabón y haber acoplado el motor de la segunda articulación en este, el primer eslabón deberá ajustarse al ancho del motor, motivo por el cual se lo hizo en forma de horquilla.

Debido a que localmente no se encontraron servomotores de doble eje, se acopló un eje fijo en la parte posterior del servomotor Mg 946R para mejorar el punto de apoyo entre el primer y segundo eslabón. En el primer eslabón, donde encaja con el eje fijo acoplado al motor, se colocó un rodamiento de bolas para reducir el rozamiento.

El primer eslabón tiene una longitud total de 200 mm para que el segundo eslabón no se encuentre muy cerca del área de trabajo. En él se realizaron dos perforaciones rectangulares para reducir el material y el peso del eslabón.

Este eslabón se unirá en la parte inferior con el motor de la primera articulación. Todos los motores se acoplan con los eslabones mediante un acople que viene en los servomotores, de los cuales se utilizó el acople circular.

A continuación, en la Figura 2.4 se indica los acoples que vienen con los servomotores.

Figura 2.4.Acople de los servomotores.
(Fuente: Javier Acosta, 2018)

En la Figura 2.5 se muestra al primer eslabón y el rodamiento acoplado.

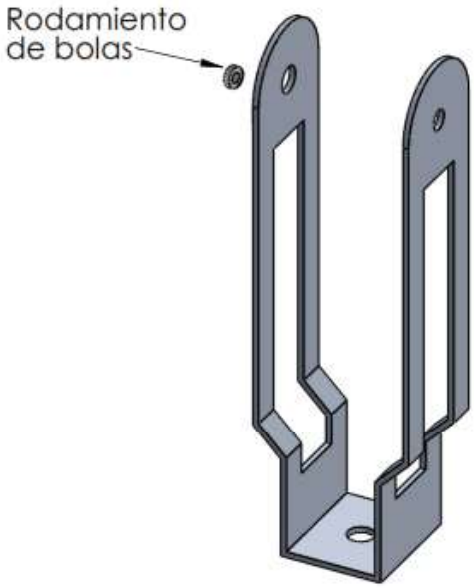


Figura 2.5. Vista en 3D del Primer eslabón.
(Fuente: Javier Acosta, 2018)

En el Anexo 2, se puede observar el plano del primer eslabón, con las partes que lo conforman y sus dimensiones

2.1.3. Tercer eslabón

El diseño del tercer eslabón se lo realizó de tal manera que el brazo pueda alcanzar objetos a una distancia de al menos 200 mm del manipulador, formándose un triángulo rectángulo en el que los lados son: el área de trabajo, el primer eslabón y la hipotenusa está formada por el segundo eslabón, tercer eslabón y el actuador final, dando como resultado el tercer eslabón con el actuador final una longitud de 165 mm, como se muestra en la Figura 2.6

.

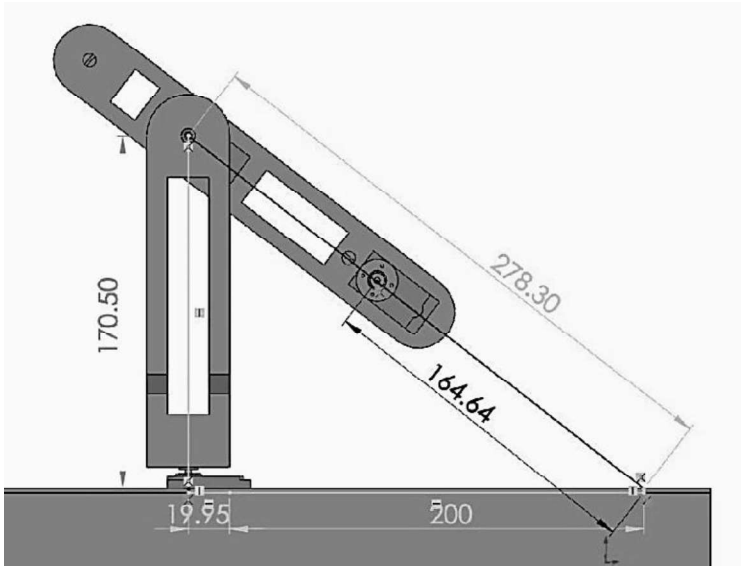


Figura 2.6.Calculo de la longuitud del tercer eslabón, las medidas se encuentran en milimetros.
(Fuente: Javier Acosta, 2018)

Se seleccionó como actuador final a un electroimán el cual tiene una longitud de 40 mm, por esta razón el tercer eslabón debe tener una longitud mínima de 125 mm, pero para garantizar que el manipulador alcance objetos hasta 20 cm de distancia, el tercer eslabón se lo hará de 130 mm.

Para este eslabón se optó por hacer un diseño en L para permitir que se una al motor de la tercera articulación mediante el acople del servomotor y en la parte inferir se une al actuador final.

En la Figura 2.7 se muestra al tercer eslabón y el electroimán acoplado.

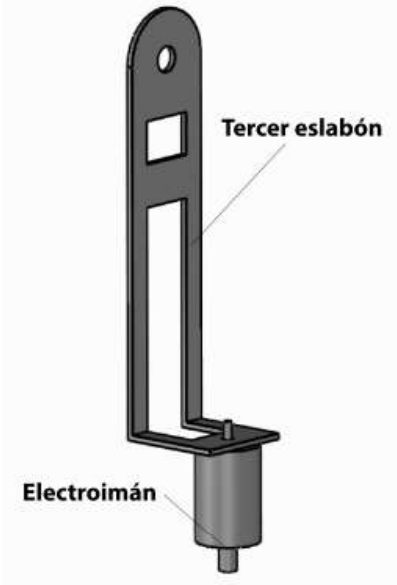


Figura 2.7. Vista en 3D del Tercer eslabón.
(Fuente: Javier Acosta, 2018)

Al igual que los otros eslabones se realizó perforaciones rectangulares para reducir material y reducir el peso. En el Anexo 3, se indica el plano correspondiente al tercer eslabón.

En la Figura 2.8, se puede observar la vista lateral del manipulador a implementar.

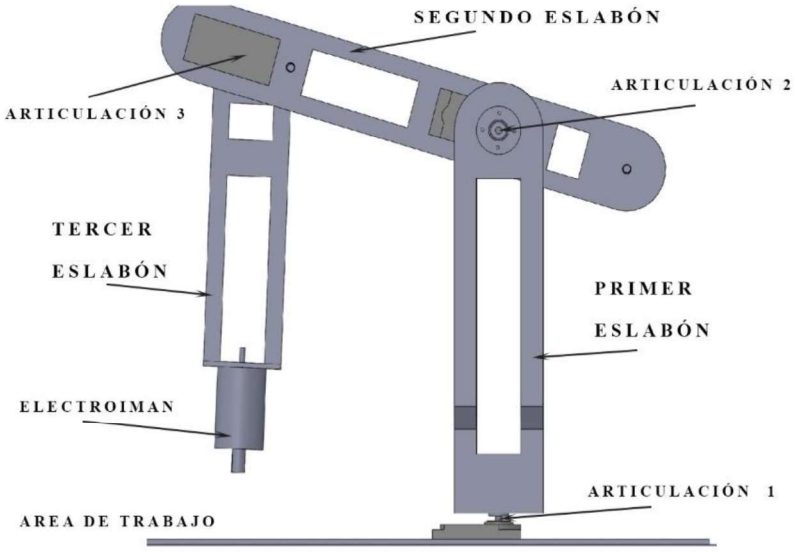


Figura 2.8.Vista lateral del manipulador.
(Fuente: Javier Acosta, 2018)

El diseño de los eslabones se lo realizó de tal manera que la construcción de estas piezas se la realice mediante el corte con una maquina CNC y luego la deformación mediante doblado de las piezas.

En la Figura 2.9 se muestra el diseño en 3D del prototipo de manipulador diseñado en Solid Works

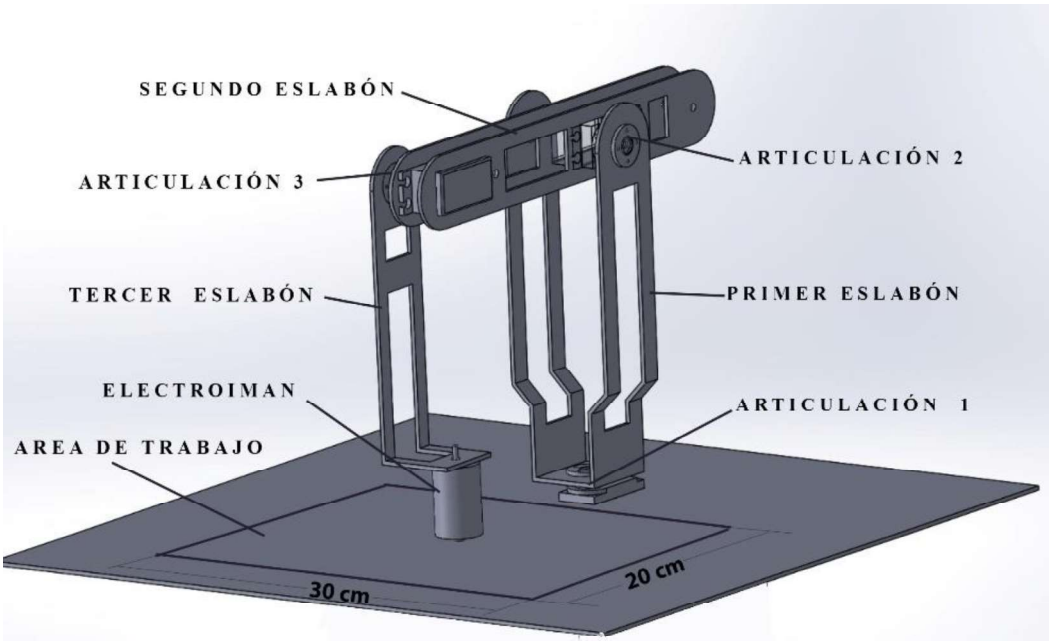


Figura 2.9.Partes de un manipulador.
(Fuente: Javier Acosta, 2018)

Para la simulación se realizó el diseño de cada una de las piezas y luego se creó un ensamble en el cual se montaron las partes y se establecieron las relaciones de posición entre cada una de las piezas.

Luego del diseño se procede a obtener la cinemática directa del manipulador.

2.2. Cinemática Directa

La cinemática directa de un manipulador trata que a partir de un conjunto de parámetros físicos, que definen la geometría del manipulador, y de los ángulos de las articulaciones, se halla la posición y orientación del efector final en el espacio tridimensional respecto

a un sistema de coordenadas de referencia, sin considerar las fuerzas que originan dicho movimiento [16].

El modelo cinemático directo de un robot manipulador se define como la matriz transformada homogénea (T) que vincula el sistema ubicado en la herramienta con el que se localiza en la base inmóvil del robot[17]. La matriz T está en función de los parámetros de las articulaciones el robot. Para un robot de 3 grados de libertad tenemos:

$$x = fx\ (q1, q2, q3)$$
$$y = fy\ (q1, q2, q3)$$
$$z = fz\ (q1, q2, q3)$$
$$\alpha = f\alpha\ (q1, q2, q3)$$
$$\beta = f\beta\ (q1, q2, q3)$$
$$\gamma = f\gamma\ (q1, q2, q3)$$

Para este caso todas las articulaciones son rotativas las variables son ángulos $q1$, $q2$, q3.

$$x, y, z \rightarrow Coordenadas\ de\ la\ posición\ del\ extremo\ del\ robot$$
$$\alpha, \beta, \gamma \rightarrow Ángulos\ de\ la\ orientacion\ del\ extremo\ del\ robot$$

Denavit y Hartenberg propusieron un método para describir y representar la geometría espacial de los elementos de un brazo con respecto a un sistema de coordenadas fijo a través de matrices de transformación de 4x4, y reducen el problema directo a encontrar la relación entre el desplazamiento espacial del sistema de coordenadas del efector final y el sistema de coordenadas de referencia [16].

El manipulador diseñado posee tres articulaciones sencillas, por esta razón su Grado De Libertad es 3 = n.

La representación Denavit Hartenberg (D-H) de un elemento rígido depende de 4 parámetros geométricos asociados con cada elemento que son.

Θ_i: Rotación entre X_{i-1} y X_i referidos a Zi, aplicando la Regla de la Mano Derecha (RMD)

d_i: Distancia entre Xi-1 y Xi medidos a lo largo de Z_i.

a_i: Distancia entre Z_i y Z_{i-1} medido a lo largo de X_i.

α_i: Rotación entre Z_i y Z_{i-1} referidos a X_i (RMD)

Para el cálculo correspondiente, se procedió de la siguiente manera:

- Se enumeraron los eslabones comenzando con 1 el primer eslabón móvil y 3 el último eslabón móvil como se observa en la Figura 2.9.

- Se enumeraron las articulaciones comenzando con 1 la del primer grado de libertad y terminando en n=3, como se observa en la Figura 2.9.

- Se localizó el eje de cada articulación, como las articulaciones son rotativas el eje será su propio eje de giro.

- Para i de 0 a n-1, se situó el eje Zi a lo largo de la articulación i, como se muestra en la Figura 2.10.

- Se fijó el sistema de coordenadas S0 en la base de tal manera que X0 debe ser normal a Z0 y Z1 como se muestra en la Figura 2.10.

- Para i de 1 a n, se situó el sistema de coordenadas S_i en la articulación i, de tal manera que Z se encuentre en el eje de la articulación, como se muestra en la Figura 2.10.

- Se estableció el eje Xi a lo largo de la normal común entre los ejes Zi-1 y Zi. como se muestra en la Figura 2.10.

- Se completó los sistemas de coordenadas con la regla de la mano derecha, en otras palabras, que Yi sea perpendicular al plano formado por los ejes Xi y Zi, como se muestra en la Figura 2.10.

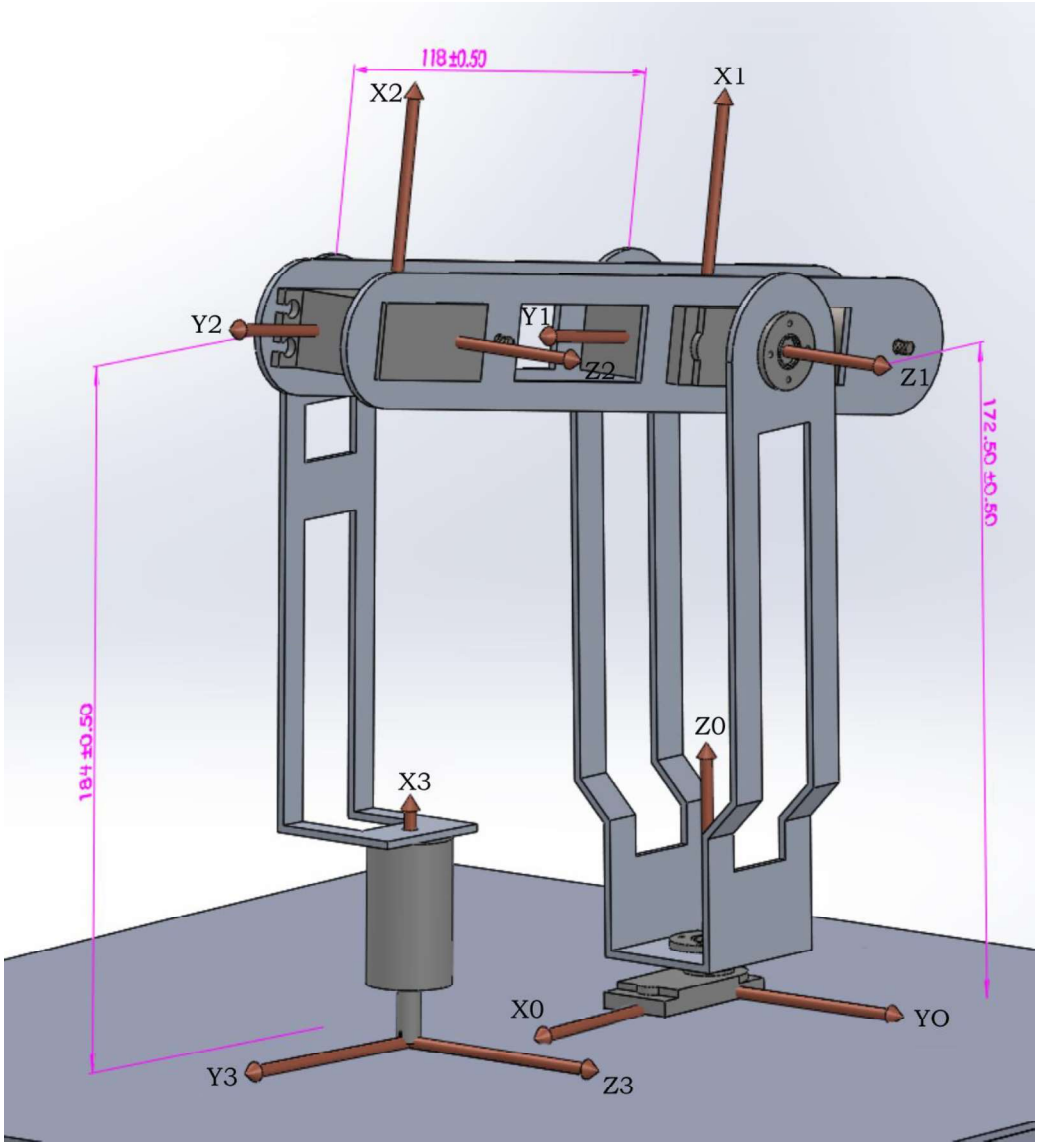


Figura 2.10. Sistema de coordenadas establecidos y dimenciones, las medidas se encuentran en mm. (Fuente: Javier Acosta, 2018)

Con los sistemas de referencia establecidos se determinan los valores de los 4 parámetros geométricos asociados con cada elemento (Θ_i, d_i, a_i, α_i), Para este caso los valores encontrados se los muestra en la tabla 2.2.

Tabla 2.2. Parámetros geométricos para cada articulación.

Fuente: Javier Acosta, 2018

Eslabón	Θ	*d*	*a*	*α*
1	Θ1	17.2 cm	0	90º
2	Θ2	0	11.8 cm	0
3	Θ3	0	18.4 cm	0

Entonces la transformación de coordenadas que describe la posición y orientación del sistema coordenado 3 con respecto al sistema coordenado 0 está dado por:

$$T_3^0(q) = A_1^0(q1)\ A_2^1(q2)\ A_3^2(q3),$$

donde

$$A_i^{i-1}(qi) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & di \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para nuestro caso:

$$A_1^0 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & 17.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^1 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 11.8\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & 11.8\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^2 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 18.4\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 18.4\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^0(q) = A_1^0\ A_2^1\ A_3^2$$

$$T_3^0(q) = \begin{bmatrix} \cos\theta_1\cos(\theta_2+\theta_3) & -\cos\theta_1\sin(\theta_2+\theta_3) & \sin\theta_1 & \cos\theta_1(11.8\cos\theta_2 + 18.4\cos(\theta_2+\theta_3)) \\ \sin\theta_1\cos(\theta_2+\theta_3) & -\sin\theta_1\sin(\theta_2+\theta_3) & -\cos\theta_1 & \sin\theta_1\,(11.8\cos\theta_2 + 18.4\cos(\theta_2+\theta_3)) \\ \sin(\theta_2+\theta_3) & \cos(\theta_2+\theta_3) & 0 & 11.8\sin\theta_2 + 18.4\sin(\theta_2+\theta_3) + 17.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3. Selección de dispositivos

A continuación, se detallan los dispositivos seleccionados para la implementación del manipulador:

2.3.1. Motor

Una de las partes más importantes en el manipulador son los motores que permitirán el movimiento del brazo. En este caso se seleccionó servomotores los cuales permiten un fácil control de posición, y tienen una buena relación torque-peso. Son diseñados para moverse determinada cantidad de grados y luego mantenerse fijo en una posición dependiendo del ancho de pulso que es enviado al terminal de señal.

El servomotor escogido es el MG-946R, indicado en la Figura 2.11

Figura 2.11. Servomotor MG-946R.

(Fuente: http://www.towerpro.com.tw/product/mg946r/)

Este servomotor tiene las siguientes características:

- Engranajes metálicos internos
- Torque: 10.5 Kg/cm (4.8V); 14 kg/cm (6.0 V)
- Velocidad de Funcionamiento: 0.20seg/60º (4.8 V sin carga); 0.17 seg/60º (6V)
- Ancho del pulso: entre 600 uS y 2400 uS
- Dimensiones: 40.7*19.7*42.9 mm
- Peso: 55 gramos

- Temperatura de Trabajo: 0 ºC hasta +55 ºC
- Ancho de banda muerta: 20 useg
- Voltaje de Operación: 4.8 - 7.2 Volts
- Cable de conexión de 300 mm

Estos motores poseen un cable de conexión con la siguiente distribución: Café = Tierra (GND), Rojo = VCC (5V), Naranja = Señal de control (PWM). Los cables café y rojo se conectan directo a la fuente de alimentación, mientras el cable naranja (señal) se conecta a la tarjeta de adquisición de datos a uno de los puertos por donde se está generando la señal PWM (terminales 8,9,10) que controla el movimiento de los tres motores del manipulador.

2.3.2. Actuador final

Debido a que los objetos a trasladar son metálicos magnéticos, se optó por un electroimán como actuador final, el cual será activado mediante una salida de la tarjeta de adquisición, que controla a un transistor en estado de corte y saturación, energizando o des energizando el electroimán con una fuente de 12V.

El electroimán a utilizar es el JF-0826B el cual es un actuador lineal magnético, cuyas características son las siguientes:

- Material: Material magnético duro
- Corriente: 2 A
- Voltaje: 12 V DC
- Aspiración inicial: 20 (N)
- Voltaje de trabajo: 24 V
- Recorrido nominal: 10 (mm)
- Longitud: 60mm

Al energizar el elemento, el campo magnético genera un desplazamiento en el vástago en dirección al electroimán, para poderlo utilizar solo como un electroimán se retiró el muelle que es el que permite que el vástago retorne cuando se des energiza la bobina;

con el vástago completamente introducido en la bobina se fijó el dispositivo al tercer eslabón del manipulador.

A continuación, en las Figuras 2.12 y 2.13, se presentan los gráficos del electroimán utilizado y su ubicación en el manipulador.

Figura 2.12. Electroiman utilizado.
(Fuente: https://www.banggood.com/es/JF-0826B-12V-DC-10mm-Hard-Magnetic-20N-Push-Pull-Electromagnet-Frame-Solenoid-p-1130239.html?cur_warehouse=CN)

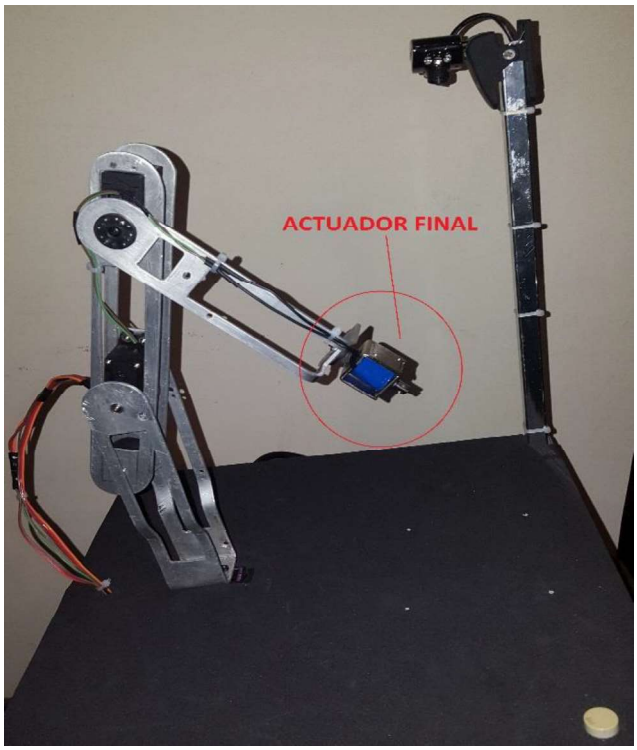


Figura 2.13. Ubicación del actuador final.
(Fuente: Javier Acosta, 2018)

2.3.3. Tarjeta de adquisición de datos

Para el ingreso y salida de señales desde el computador se requirió de una tarjeta de adquisición de datos la cual permite obtener desde el computador, las señales de control para cada motor y activar o desactivar el electroimán; además, permite ingresar señales de pulsantes que dan la señal de inicio. Se optó por trabajar con una tarjeta Arduino, debido a que se puede trabajar con Linux, sus buenas prestaciones y su bajo costo.

El modelo especifico con el que se trabajó con este proyecto es la tarjeta Arduino Mega 2560 que es una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede comunicarse a un PC a través del puerto USB utilizando lenguajes como Flash, Processing, MaxMSP, etc. [18].

En la Figura 2.14 se muestra la tarjeta Arduino utilizada.

La programación de esta tarjeta también es open-source y puede ser ejecutada en Mac, Windows y Linux, sus características son:

- 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM).
- 16 entradas análogas.
- UARTs (puertos serial por hardware).
- Cristal oscilador de 16 MHz.
- Conexión USB, jack de alimentación, conector ICSP.
- Botón de reset.
- Microcontrolador: ATmega2560
- Voltaje Operativo: 5 V
- Voltaje de Entrada: 7-12 V
- Voltaje de Entrada(límites): 6-20 V
- Corriente DC por cada Pin Entrada/Salida: 40 mA
- Corriente DC entregada en el Pin 3.3V: 50 mA
- Memoria Flash: 256 KB (8 KB usados por el bootloader)

- SRAM: 8 KB
- EEPROM: 4 KB

Arduino Mega puede ser alimentado mediante el puerto USB o con una fuente externa de poder, posee algunos pines para la alimentación del circuito aparte del adaptador para la alimentación:

- VIN: A través de este pin es posible proporcionar alimentación a la placa.
- 5 V: Se puede obtener un voltaje de 5 V y una corriente de 40 mA desde este pin.
- 3.3 V: Se puede obtener un voltaje de 3.3 V y una corriente de 50 mA desde este pin.
- GND: Ground (0 V) de la placa.

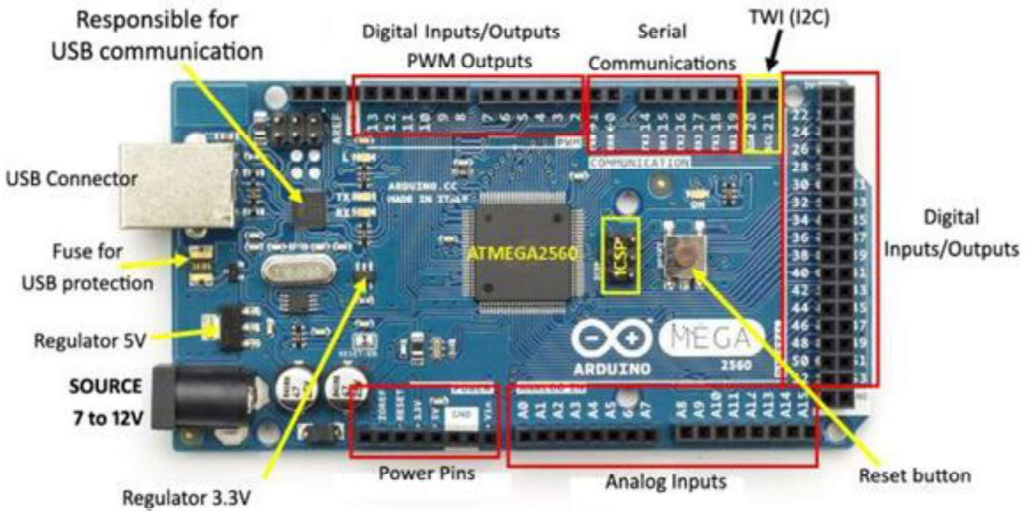


Figura 2.14.Tarjeta de adquisición de datos Arduino Mega 2560.
(Fuente: Javier Acosta, 2018)

Por los terminales 8, 9, 10 de la tarjeta de adquisición de datos se generan las señales PWM para cada motor, en el terminal 13 se genera la señal para activar y desactivar la bobina del electroimán y en el terminal 7 se conecta el pulsador que da el inicio al proceso del manipulador.

2.3.4. Cámara Web

Para poder reconocer los colores y objetos se utilizó una cámara web, ya que la única restricción de la cámara es que funcione con el sistema operativo Linux (Ubuntu 16.04),

se optó por utilizar una cámara web marca Altek. En la Figura 2.15 se muestra la cámara utilizada.

Figura 2.15.Cámara web Altek.
(Fuente: Javier Acosta, 2018)

Esta cámara se conecta al computador mediante el puerto USB y tiene las siguientes características:

- Interface USB 2.0.
- Balance de blancos automático.
- Control de exposición automático.
- Control de flash automático.
- VGA, 30 cuadros /segundo
- Resolución 300 k pixeles

2.3.5. Fuente de alimentación

Se requirió de una fuente de alimentación que proporcione los voltajes y corrientes necesarios para el correcto funcionamiento de los diferentes elementos del manipulador. En este caso es necesario que tenga una salida de 5 V con una corriente mínima de 4 A y otra salida de 12 V con una corriente mínima de 1 A, por lo que se optó por una fuente de computador la cual posee las siguientes características:

- 115 V / 230 V, 12 A / 6 A
- 50 / 60 Hz
- Salida de 3.3 V corriente máxima 25 A.
- Salida de 5 V corriente máxima 28 A.
- Salida de 5 VSB corriente máxima 2.5 A.
- Salida de 12 V_1 corriente máxima 18 A.
- Salida de 12 V_2 corriente máxima 18 A.
- Salida de -12 V corriente máxima 0.8 A.
- Potencia máxima de salida 450 W

En la Figura 2.16 se muestra la fuente utilizada.

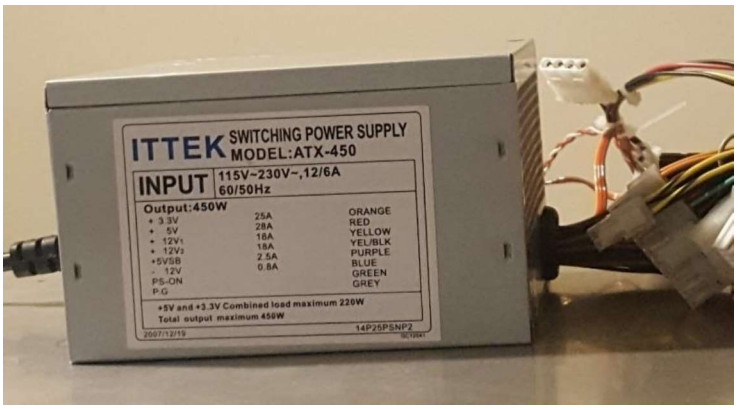


Figura 2.16.Fuente seleccionada.
(Fuente: Javier Acosta, 2018)

2.4. Circuito de Conexión Completo

A continuación, en la Figura 2.17 se muestra el circuito electrónico diseñado, en el que se detallan los pines de conexión en la tarjeta de adquisición de datos con los motores y demás componentes del proyecto.

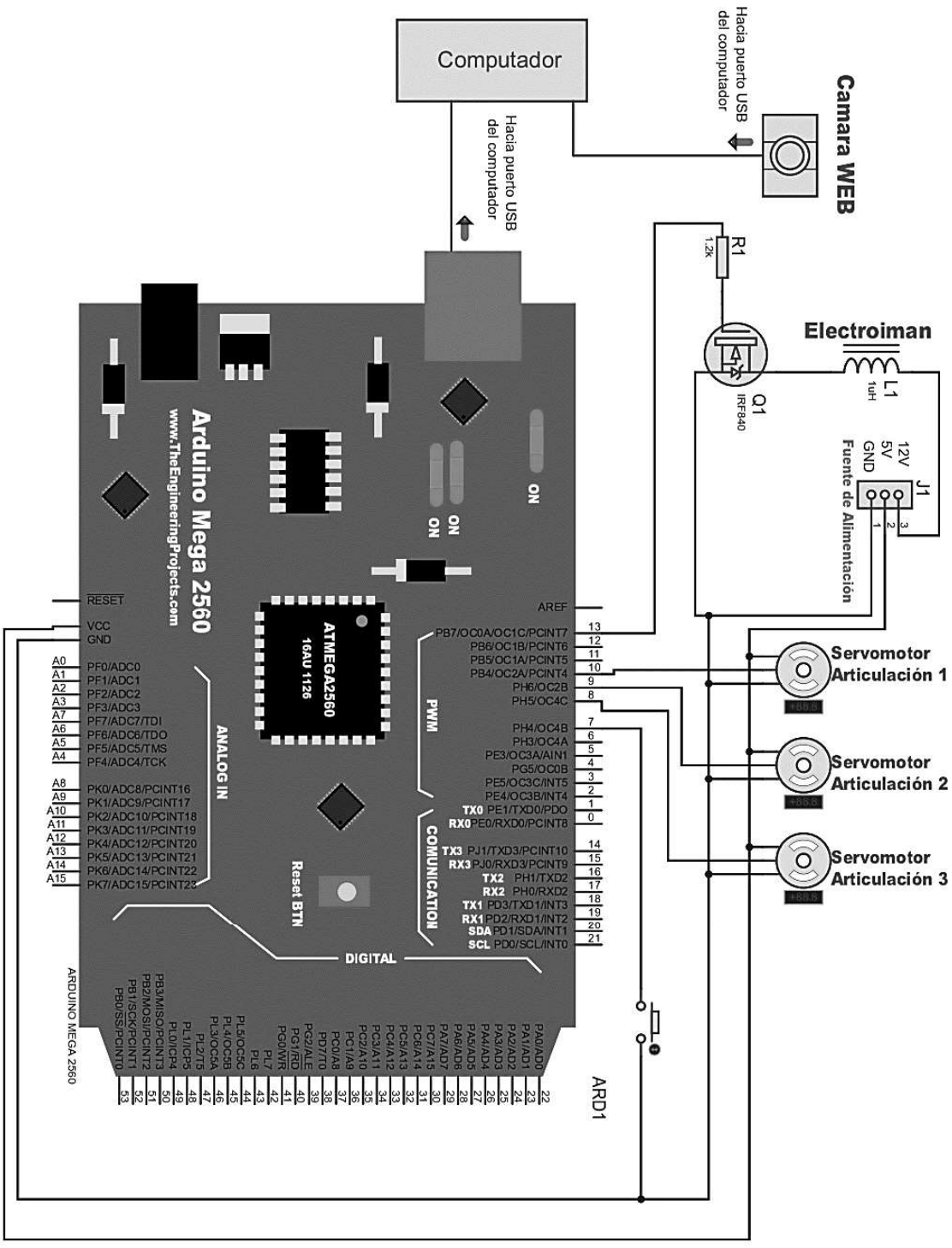


Figura 2.17. Circuito Electrónico diseñado.

(Fuente: Javier Acosta, 2018)

2.5. Fabricación del Manipulador.

El material seleccionado para la construcción de los eslabones es aluminio por sus prestaciones de bajo peso y resistencia mecánica. Para la fabricación de estos eslabones se procedió a obtener el código-G de las partes que conforman los eslabones, este código es un lenguaje de descripción de operaciones para máquinas de control numérico, Posteriormente se introdujo este código-G en la máquina CNC para proceder a cortar las piezas de una plancha de aluminio de 2 mm de espesor como se muestra en la Figura 2.18. En el Anexo 5, se puede observar el código-G generado para la máquina CNC.

Figura 2.18. Proceso de corte de las piezas en una maquina CNC.
(Fuente: Javier Acosta, 2018)

La Figura 2.19 muestra las partes del segundo eslabón mecanizadas con una CNC de corte por plasma.

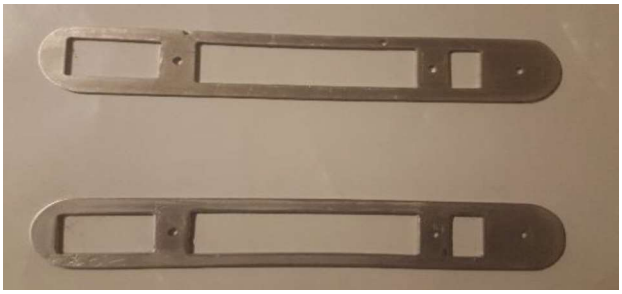


Figura 2.19. Partes del segundo eslabón cortado con una CNC de plasma.
(Fuente: Javier Acosta, 2018)

Se elimino la escoria generada en el proceso de corte y se mejoró el acabado, se procedió a realizar la deformación mediante doblado de las piezas para formar los eslabones 1 y 3, luego se pasó al ensamblaje del manipulador, en el Anexo 4, se puede apreciar el plano de despiece.

2.6. Diseño del controlador

El propósito de este proyecto es la creación e implementación del control para un manipulador mediante el uso de software libre. Para cumplir se utilizó un computador en el que se instaló el sistema operativo Ubuntu 16.04 LTS, la cual es una distribución de Linux, es una versión LTS esto quiere decir que es estable y tiene respaldo extendido de 5 años hasta el 2021.

Se utilizó el framework ROS KINETIC que es la décima versión oficial de ROS, en este framework se instaló OROCOS la cual viene como toolchain para las últimas versiones de ROS.

El Controlador está diseñado para que cumpla con las siguientes características:

Al pulsar el botón que comienza el proceso del manipulador, el control captura una imagen con la cámara WEB, se aplica filtros en la imagen para discriminar los colores que no pertenecen a los rangos a utilizar (rojos, verdes).

Con la imagen aplicada los filtros, determina el total de piezas rojas y verdes, determina la posición de cada pieza y discrimina a las piezas que se encuentran fuera de los puntos de trabajo del manipulador.

Con esto se determina en que puntos tengo piezas rojas y en cual verdes, si existe una pieza en la posición uno, ubica el brazo en esta posición, activa el electroimán, si la pieza es de color rojo mueve el brazo al depósito de rojos, si es de color verde mueve el brazo al depósito de verdes y desactiva el electroimán para soltar la pieza, esta operación la realiza por cada punto de trabajo si existe una pieza en dichos puntos.

Al terminar este proceso el controlador espera a que sea nuevamente pulsado el botón de inicio.

En el Anexo 5, se muestra el diagrama de flujo del control del prototipo del manipulador.

En el Anexo 6, se presenta el código generado para la tarjeta Arduino.

En la Figura 2.22 se muestra el estado final del manipulador:

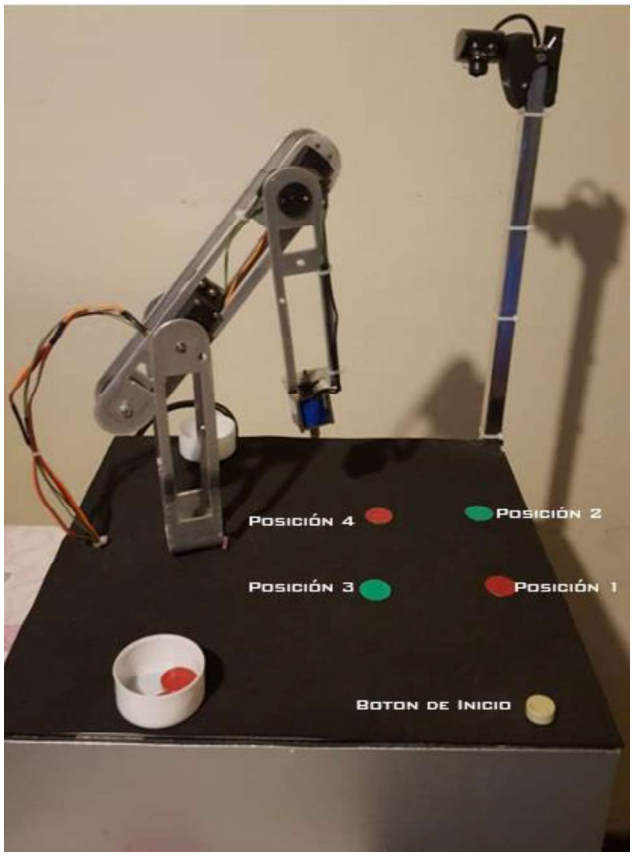


Figura 2.20.Estado final del manipulador.
(Fuente: Javier Acosta, 2018)

3. RESULTADOS Y DISCUSIÓN

Se realizaron varias pruebas de funcionamiento del equipo, las cuales detallamos a continuación:

3.1. Primera Prueba

Se realizó una prueba con una iluminación de un foco led 5W, con color de luz de 3000 K (kelvin), con 350 lux, ubicada a 1.5 m sobre el manipulador, en la Figura 3.1 se muestra al manipulador y el área de trabajo con la iluminación indicada.

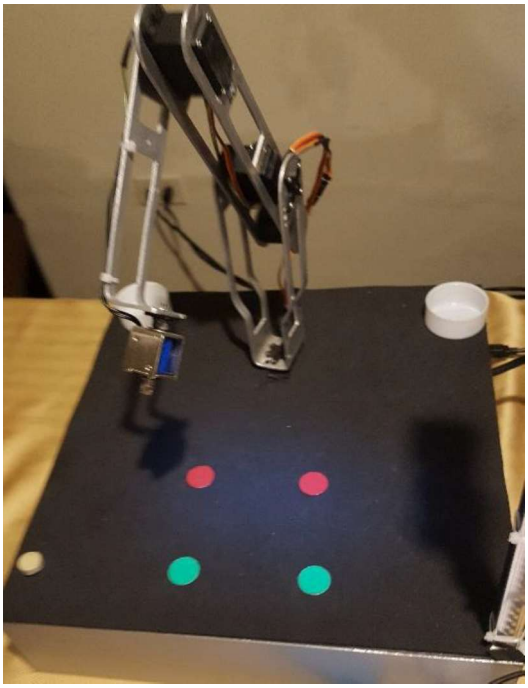


Figura 3.1.Manipulador en la primera prueba.
(Fuente: Javier Acosta, 2018)

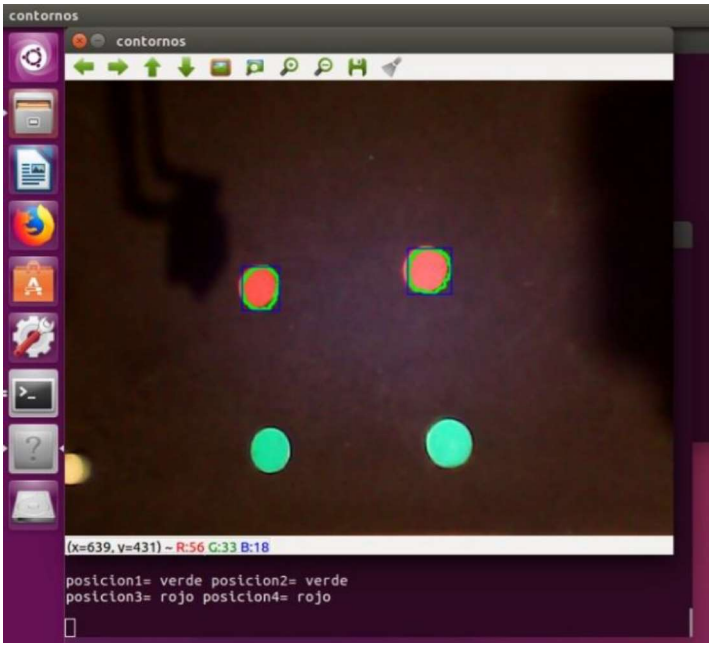


Figura 3.2.Imagen caturada por la camara y resultado de identificacion objetos en la primera prueba.
(Fuente: Javier Acosta, 2018)

En la Figura 3.2 se puede ver en la parte superior, la foto capturada por la cámara del manipulador y aplicada los filtros, y en la parte inferior los objetos que reconoce el control, como se puede observar el control reconoció los objetos rojos y verdes correctamente.

3.2. Segunda Prueba

Prueba con el manipulador expuesto directamente a la luz solar, en esta prueba el manipulador funciono correctamente detectando tanto los objetos rojos y verdes.

En la Figura 3.3 se muestra al manipulador y el área de trabajo expuestos directamente a la luz solar.

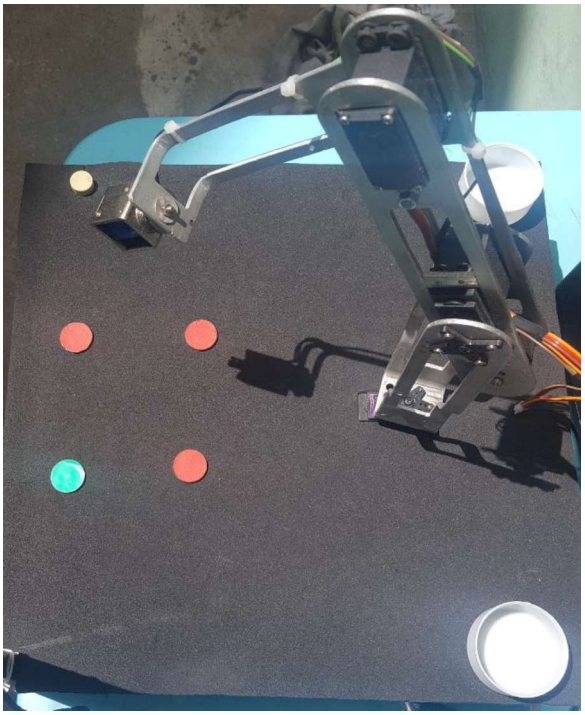


Figura 3.3.Prueba del manipulador expuesto a la luz solar.

(Fuente: Javier Acosta, 2018)

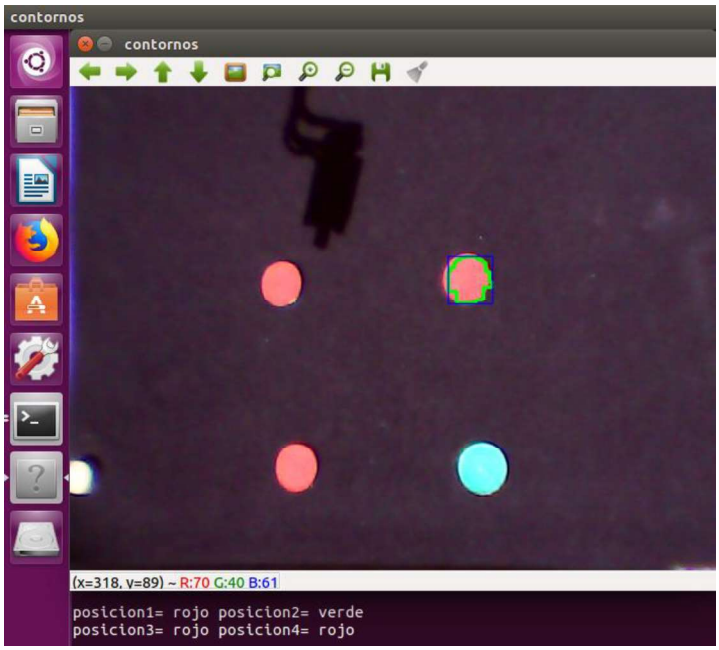


Figura 3.4. Imagen caturada por la camara y resultado de identificacion objetos en la segunda prueba.

(Fuente: Javier Acosta, 2018)

En la Figura 3.4 se puede ver en la parte superior, la foto capturada por la cámara del manipulador y aplicada los filtros, y en la parte inferior los objetos que reconoce el control, como se puede observar el control reconoció todos los objetos rojos y verdes correctamente.

3.3. Tercera Prueba

Prueba con el manipulador dentro de una habitación, pero junto a una ventana con luz solar, para esta prueba se ubicó al manipulador cerca de una ventana en un día soleado, el manipulador se encontraba completamente en la sombra, pero la luz solar generó un brillo en la imagen capturada, produciendo errores en el procesamiento de las imágenes, ya que no reconoció los objetos ni rojos ni verdes.

En la Figura 3.5 se muestra al manipulador y el are de trabajo junto a una luz intensa.

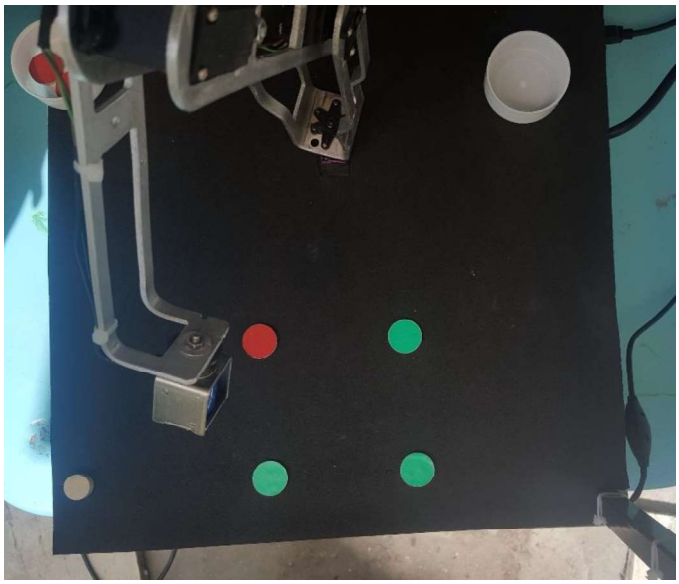


Figura 3.5. Prueba del manipulador junto a una vetana con luz solar.
(Fuente: Javier Acosta, 2018)

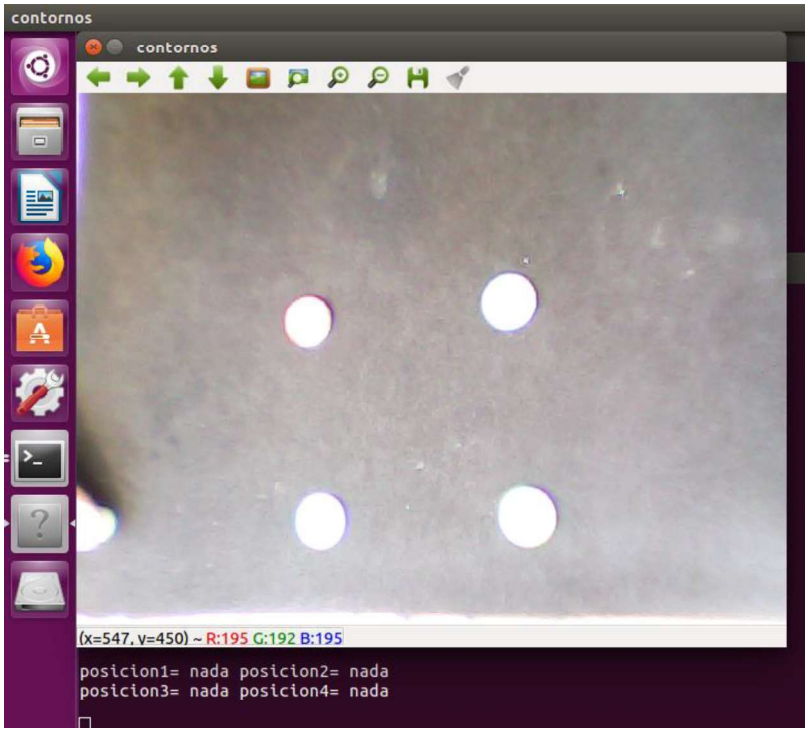


Figura 3.6. Imagen caturada por la camara y resultado de identificacion objetos en la tercera prueba. (Fuente: Javier Acosta, 2018)

En la Figura 3.6 podemos observar en la parte superior la imagen capturada y en la parte inferior podemos observar que los objetos no fueron reconocidos.

3.4. Cuarta Prueba

Se realizo una prueba con las cuatro fichas verdes para determinar el tiempo que se demora en mover todas estas fichas, para esta prueba se midió el tiempo desde que se oprime el botón de inicio del proceso hasta que luego de dejar la ultima ficha el brazo llega a la posición de reposo.

En la Tabla 3.1 podemos observar los resultados obtenidos de lasb20 pruebas realizadas.

Tabla 3.1. Pruebas realizadas con todas las fichas verdes.

Fuente: Javier Acosta, 2018

# de Prueba	4 fichas verdes (s)
1	26.8
2	27.3
3	27.2
4	27.1
5	28.4
6	27.9
7	26.6
8	27.4
9	26.9
10	26.0
11	25.7
12	27.4
13	26.5
14	27.0
15	26.7
16	27.3
17	27.2
18	27.5
19	26.8
20	27.1
Media (promedio)	**27.0**
Mediana	**27.1**
Desviación estándar	**0.6**

3.5. Quinta Prueba

Se realizo una prueba con las cuatro fichas rojas para determinar el tiempo que se demora en mover todas estas fichas, para esta prueba se midió el tiempo desde que se oprime el botón de inicio del proceso hasta que luego de dejar la última ficha el brazo llega a la posición de reposo.

En la Tabla 3.2 podemos observar los resultados obtenidos de las 20 pruebas realizadas.

Tabla 3.2. Pruebas realizadas con todas las fichas rojas.

Fuente: Javier Acosta, 2018

# de Prueba	4 fichas rojas (s)
1	27.0
2	27.2
3	28.2
4	27.8
5	26.9
6	26.8
7	27.4
8	27.6
9	27.3
10	27.6
11	28.0
12	27.3
13	27.8
14	27.2
15	26.6
16	25.8
17	26.5
18	26.8
19	28.1
20	27.6
Media (promedio)	27.3
Mediana	27.3
Desviación estándar	0.6

3.6. Sexta Prueba

Se realizo una prueba con dos fichas rojas y dos fichas verdes, en esta prueba se midió el tiempo desde que se oprime el botón de inicio del proceso hasta que luego de dejar la última ficha el brazo llega a la posición de reposo, en la cual las 5 primeras muestras

las dos fichas rojas se las coloco en la posición 1 y 2 en las cinco siguientes muestras en las posiciones 2 y 3, en las siguientes cinco muestras se ubicó en las posiciones 3 y 4 y en las últimas cinco muestras se las ubico en las posiciones 1 y 4.

En la Tabla 3.3 podemos observar los resultados obtenidos de las 20 pruebas realizadas.

Tabla 3.3. Pruebas realizadas con dos fichas rojas y dos fichas verdes.

Fuente: Javier Acosta, 2018

# de Prueba	2 fichas rojas y 2 verdes (s)
1	26.7
2	26.9
3	26.3
4	27.3
5	26.2
6	26.8
7	27.5
8	27.6
9	26.5
10	26.9
11	26.5
12	27.3
13	26.9
14	27.2
15	26.8
16	26.5
17	27.6
18	26.6
19	27.8
20	27.3
Media (promedio)	27.0
Mediana	26.9
Desviación estándar	0.5

3.7. Séptima Prueba

Se realizo una prueba con una sola ficha roja, en esta prueba se midió el tiempo desde que se oprime el botón de inicio del proceso hasta que luego de dejar la ficha el brazo llega a la posición de reposo, en la cual las 5 primeras muestras la ficha roja se la coloco en la posición 1, en las cinco siguientes muestras en la posición 2, en las siguientes cinco muestras se ubicó en la posición 3 y en las últimas cinco muestras se la ubico en la posición 4.

En la Tabla 3.4 podemos observar los resultados obtenidos de las 20 pruebas realizadas.

Tabla 3.4. Pruebas realizadas con una sola ficha roja.

Fuente: Javier Acosta, 2018

# de Prueba	1 ficha (s)
1	8.9
2	8.5
3	6.8
4	7.2
5	8.2
6	8.0
7	8.9
8	7.9
9	8.4
10	7.3
11	7.4
12	8.2
13	8.0
14	8.3
15	8.6
16	8.6
17	8.4
18	7.6
19	8.7

20	6.8
Media (promedio)	8.0
Mediana	8.2
Desviación estándar	0.6

3.8. Octava Prueba

En esta prueba se registró todos errores de reconocimiento de objetos generados en las pruebas anteriores (cuarta, quinta, sexta, séptima)

Tabla 3.5. Errores generados en las pruebas.
Fuente: Javier Acosta, 2018

Prueba	1 ficha (s)
Cuarta prueba	No reconoció ficha 1
Quinta Prueba	No reconoció ficha 3
Sexta Prueba	Recoció ficha 2 que no había
Sexta Prueba	No reconoció ficha 2 y 3
Sexta Prueba	No reconoció ficha 1
Séptima Prueba	No reconoció ficha 2

De las 86 pruebas realizadas se obtuvieron 6 errores de reconocimiento de las imágenes, con estos datos se puede decir que el equipo tuvo un 93% de efectividad en el reconocimiento de las fichas.

4. CONCLUSIONES

- Es posible implementar controladores con el uso de software libre, sin la necesidad de programas licenciados. Esto permite desarrollar tecnología a más bajo costo.

- Tanto ROS como OROCOS son middlewares robóticos que permiten el desarrollo de software para control de robots en bajo nivel, lo que permite realizar acciones en tiempo real.

- Mediante el desarrollo de este proyecto se han sentado las bases para el control, mediante software libre, de robots. Si bien en este caso se realizó el control de un manipulador, es factible realizar el control de cualquier otro tipo de robot ya que los middlewares lo permiten.

- La programación de OROCOS bajo el ambiente de ROS permite crear esquemas de control funcionales mediante una programación basada en componentes. Tanto ROS como OROCOS fueron desarrollados para crear controladores para robots, por lo que su uso en este campo está creciendo.

- La validación del controlador desarrollado se lo hizo aplicándolo a un manipulador diseñado y construido localmente. Las pruebas indicaron que el robot ejecuta las acciones de control de manera adecuada.

- El procesamiento de imágenes requiere un ambiente de luz controlado para su correcto funcionamiento. Se puede mejorar el reconocimiento cambiando la cámara WEB por una de mejores características, pero se tiene el inconveniente que el sistema operativo Ubuntu no reconoce a todas las cámaras, a pesar de esto, definitivamente, no es posible trabajar en todo tipo de iluminación.

- La programación en ROS tiene un alto grado complejidad debido a que no existe la suficiente bibliografía sobre como programar en este middleware.

- El tiempo que el manipulador demora en organizar las cuatro fichas es en promedio 27 segundos y es similar si son las cuatro fichas rojas, o son las cuatro fichas verdes o si son rojas y verdes.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] E. Estévez, A. Sánchez García, J. Gámez García, y J. Gómez Ortega, «Aproximación Basada en UML para el Diseño y Codificación Automática de Plataformas Robóticas Manipuladoras», *RIAI - Rev. Iberoam. Autom. e Inform. Ind.*, vol. 14, n.º 1, pp. 82-93, 2017.

[2] M. Riccillo, «Robótica», *Conectados*, p. 36, 2012.

[3] V. I. A. Hernández Matías Juan Carlos, *Sistemas De Para Las Pymes Españolas Automatización Y Robótica*. 2015.

[4] I. Bambino, «Una Introducción a los Robots Móviles», pp. 1-86, 2008.

[5] R. Kelly y V. Santibáñez, *Control de movimiento de robots manipuladores*. 2003.

[6] GustavoVelasco, «es - ROS Wiki». 2014.

[7] J. G. Pérez, «Introducción a ROS en Raspberry Pi», 2017.

[8] A. Araújo, D. Portugal, M. S. Couceiro, J. Sales, y R. P. Rocha, «Desarrollo de un robot móvil compacto integrado en el middleware ROS», *RIAI - Rev. Iberoam. Autom. e Inform. Ind.*, vol. 11, n.º 3, pp. 315-326, 2014.

[9] L. Joseph, *Mastering ROS for Robotics Programming*, vol. 64, n.º 6. 2015.

[10] A. Martinez y E. Fernández, *Learning ROS for Robotics Programming A practical, instructive, and comprehensive guide to introduce yourself to ROS, the top-notch, leading robotics framework*, Primera ed. BIRMINGHAM - MUMBAI: PACKT PUBLISHING, 2013.

[11] Narrendar R. C., «An Intruction to Open-Source Robotic Framework.pdf», 2013.

[12] M. Vallés, J. I. Cazalilla, Á. Valera, V. Mata, y Á. Page, «Implementación basada en el middleware OROCOS de controladores dinámicos pasivos para un robot paralelo», *RIAI - Rev. Iberoam. Autom. e Inform. Ind.*, vol. 10, n.º 1, pp. 96-103, 2013.

[13] M. Matellán, Olivera, Vicente; Cañas Plaza, José, «Campeonato de Robótica Educativa», *Minist. Educ. y Cienc. España*, 2005.

[14] J. Macanas y J. Feilu, «Interacción entre webcam y brazo robot para el posicionamiento del efector final», *Esc. Técnica Super. Ing. Ind.*, 2014.

[15] F. J. Enríquez, E. Sifuentes, G. Bravo, y A. Castro, «Sistema embebido para validar el funcionamiento de la tarjeta de adquisición de datos USB-6009 de national instruments», *Inf. Tecnol.*, vol. 27, n.º 5, pp. 191-200, 2016.

[16] A. Jaramillo Botero, «Cinemática de manipuladores», p. 29, 2010.

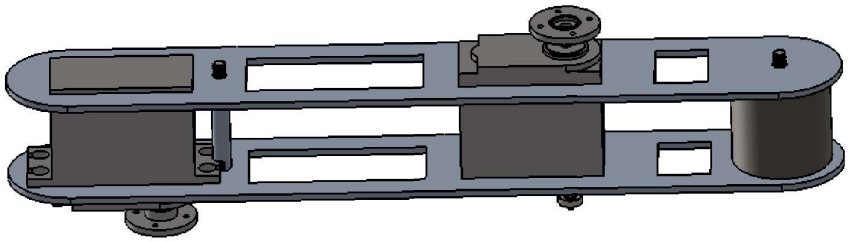
[17] M. (Universidad de M. Román, D. Domínguez, y E. Muñoz, «Simulador Cinemático de un Robot Maniplador Industrial», 2014.

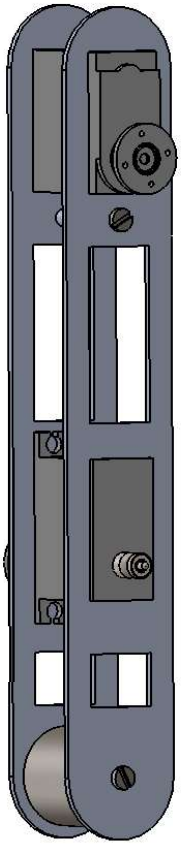
[18] D. Mellis, «Arduino Mega 2560», *Retrieved November*. p. 2560, 2011.

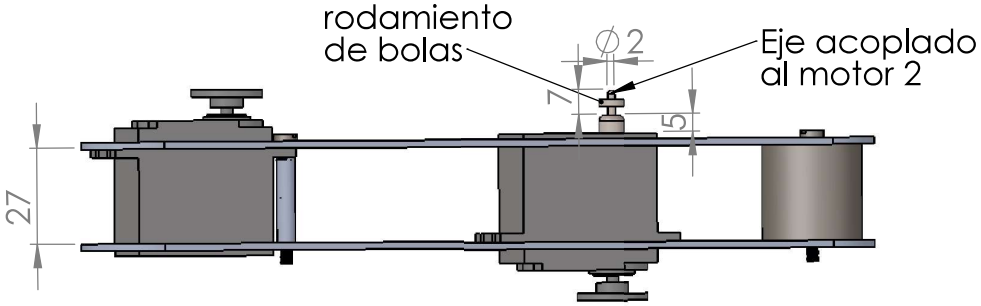
ANEXOS

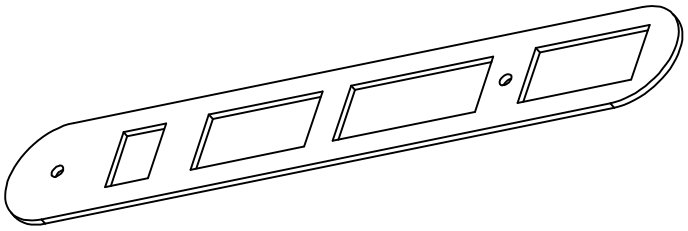
- ANEXO 1: PLANO DEL SEGUNDO ESLABÓN
- ANEXO 2: PLANO DEL PRIMER ESLABÓN
- ANEXO 3: PLANO DEL TERCER ESLABÓN
- ANEXO 4: PLANO DE DESPIECE DEL MANIPULADOR
- ANEXO 5: CÓDIGO-G GENERADO PARA CORTE DE PIEZAS CON MAQUINA CNC
- ANEXO 6: DIAGRAMA DE FLUJO DEL CONTROL DEL MANIPULADOR
- ANEXO 7: CÓDIGO GENERADO PARA LA TARJETA ARDUINO
- ANEXO 8: DIAGRAMA DE FLUJO DEL PROGRAMA EN LA TARJETA ARDUINO.
- ANEXO 9: MANUAL DE USUARIO.

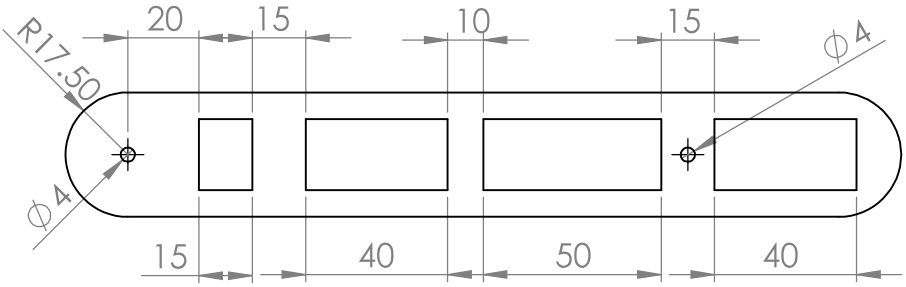
ANEXO 1: PLANO DEL SEGUNDO ESLABÓN

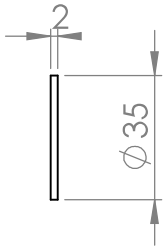






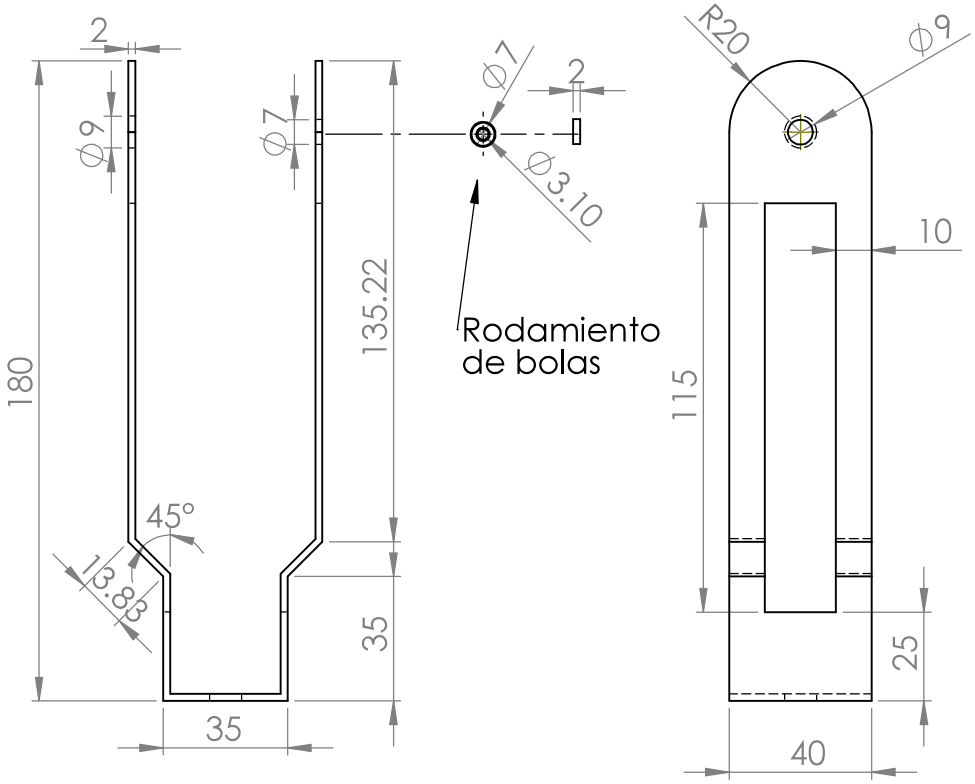


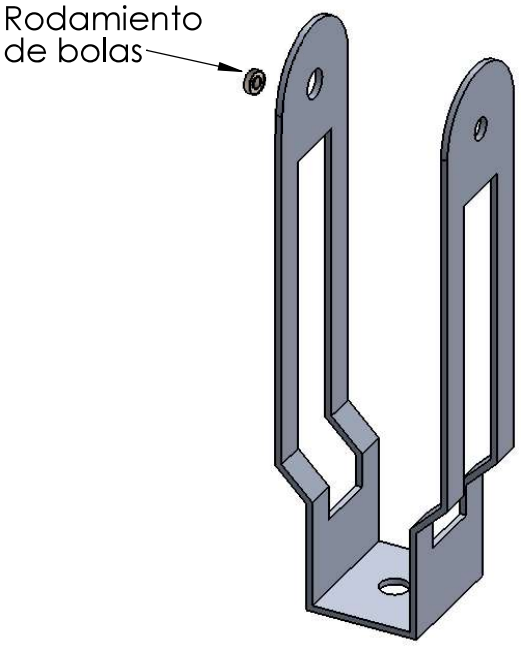


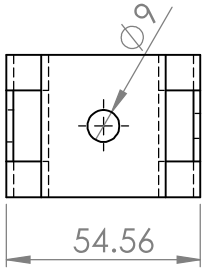


	NOMBRE	FIRMA	FECHA	TÍTULO:
DIBUJ.	Javier Acosta		31/07/2018	Manipulador
VERIF.				
APROB.				
FABR.				
CALID.			MATERIAL: Aluminio	N.º DE DIBUJO: Segundo Eslabón — A4
			PESO:	ESCALA:1:2 — HOJA 2 DE 4

ANEXO 2: PLANO DEL PRIMER ESLABÓN

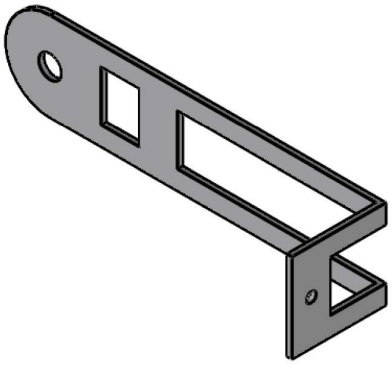


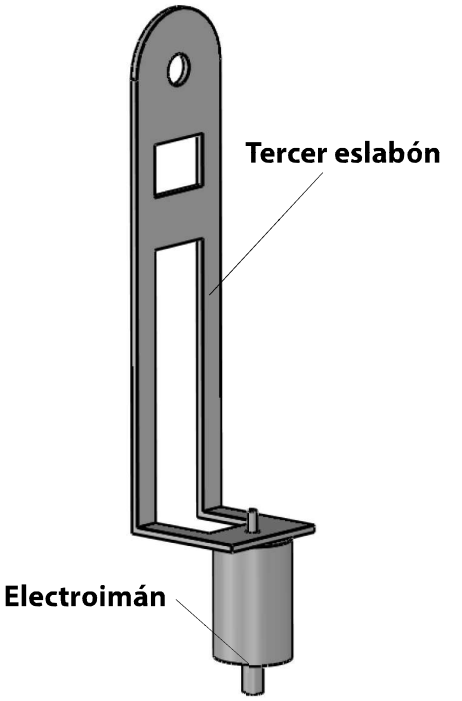


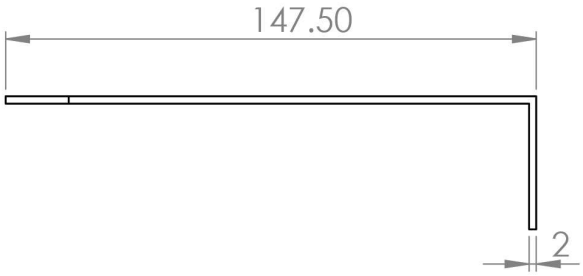


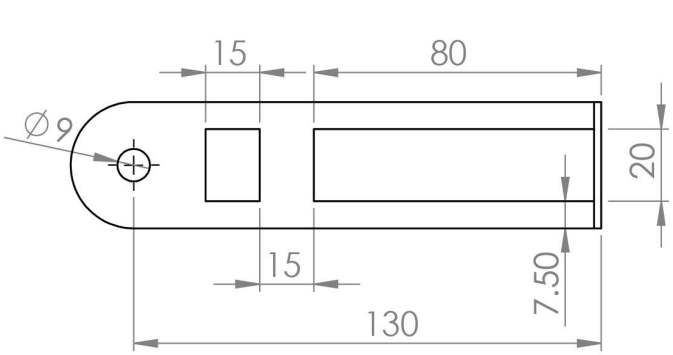
	NOMBRE	FIRMA	FECHA	TÍTULO:
DIBUJ.	Javier Acosta		31-07-2018	Manipulador
VERIF.				
APROB.				
FABR.				
CALID.			MATERIAL: Aluminio	N.º DE DIBUJO: Primer Eslabón — A4
			PESO:	ESCALA:1:2 — HOJA 1 DE 4

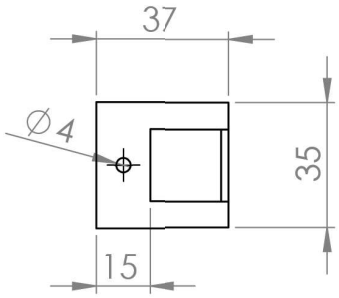
ANEXO 3: PLANO DEL TERCER ESLABÓN





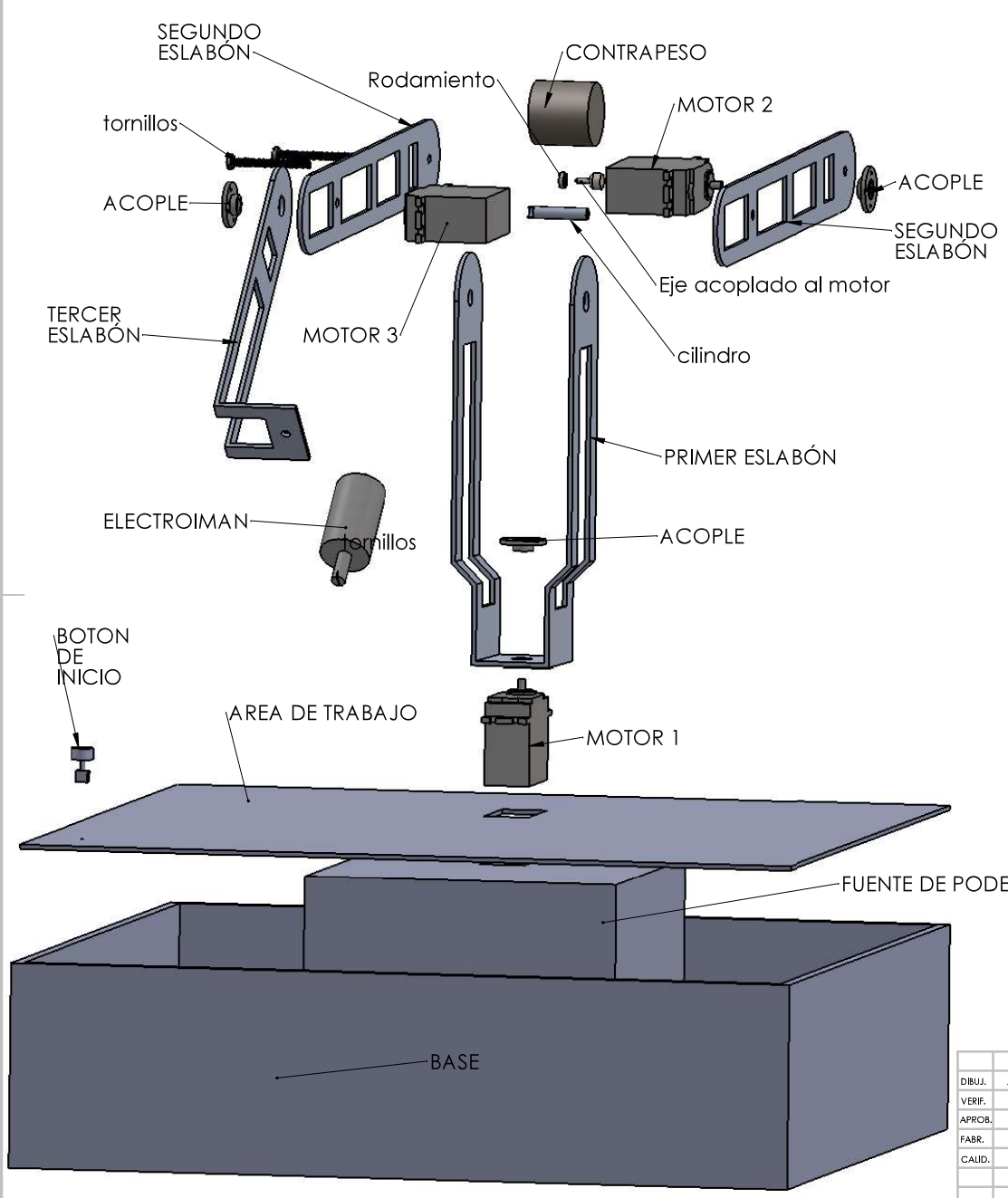


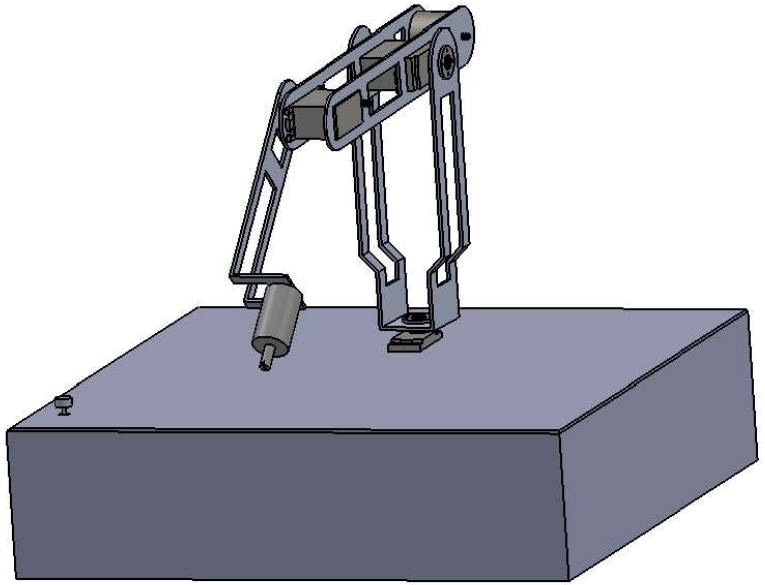


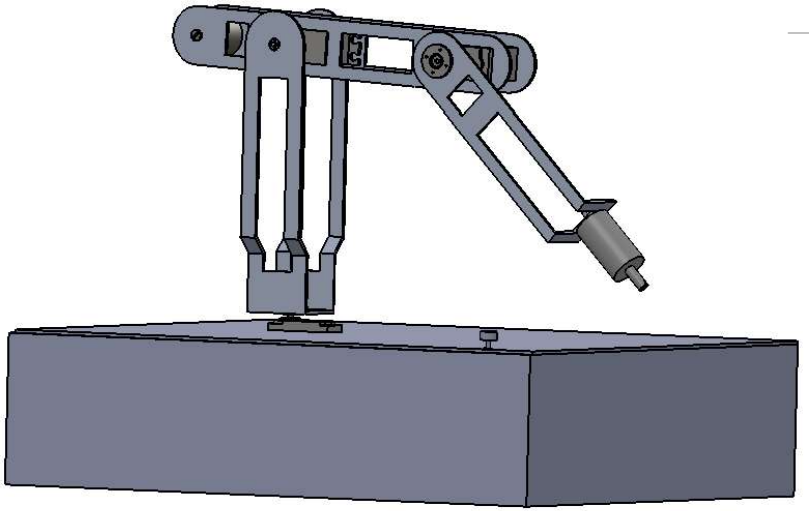


	NOMBRE	FIRMA	FECHA	TÍTULO:
DIBUJ.	Javier Acosta		31-07-2018	Manipulador
VERIF.				
APROB.				
FABR.				
CALID.			MATERIAL: Aluminio	N.º DE DIBUJO Tercer Eslabón — A4
			PESO:	ESCALA:1:2 — HOJA 1 DE 4

ANEXO 4: PLANO DE DESPIECE DEL MANIPULADOR







	NOMBRE	FIRMA	FECHA	TÍTULO:
DIBUJ.	Javier Acosta		20/02/2019	Manipulador
VERIF.				
APROB.				
FABR.				
CALID.			MATERIAL: Aluminio	N.º DE DIBUJO: Despiece — A3
			PESO:	ESCALA:1:2 — HOJA 1 DE 1

ANEXO 5: CÓDIGO-G GENERADO PARA CORTE DE PIEZAS CON MAQUINA CNC

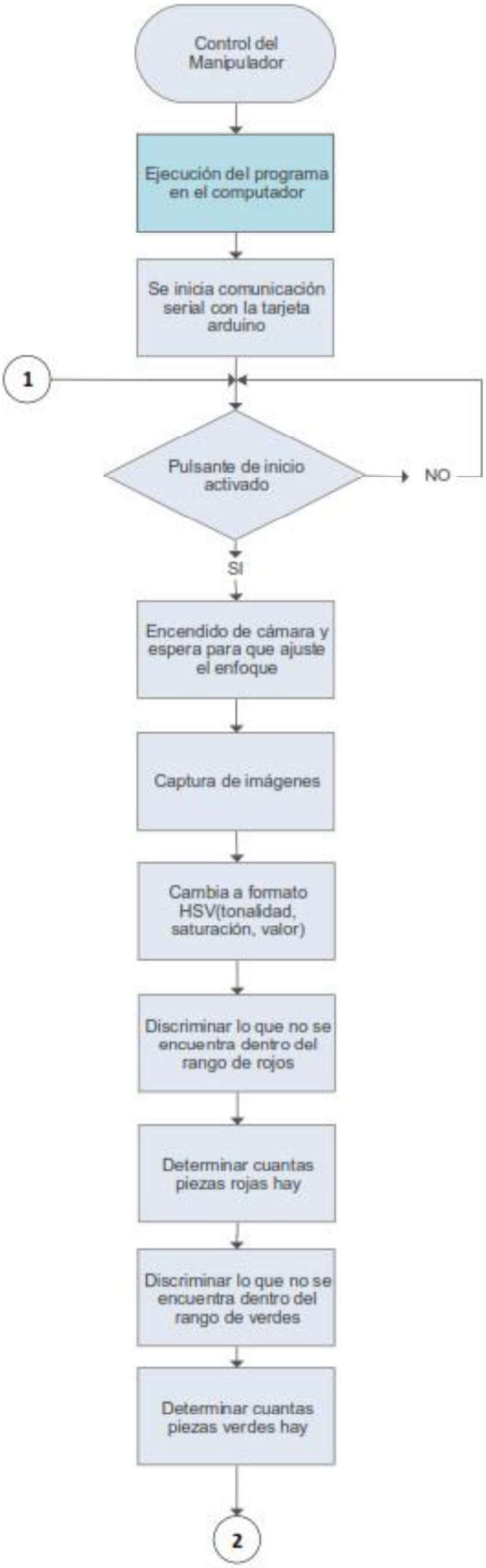
```
%
G00 G90 G17 G21
M3 S3000

T1 M6
X2222.883 Y1592.032
G01 Z-2. F150
G03 X2222.883 Y1592.032 I-2.285 F450
G00 Z5.

X2260.593 Y1580.604
G01 Z-2. F150
X2243.452 Y1580.604 F450
X2243.452 Y1603.459
X2260.593 Y1603.459
X2260.593 Y1580.604
G00 Z5.

X2239.985 Y1545.912
G01 Z-2. F150
X2239.985 Y1525.912 F450
X2357.985 Y1525.912
X2357.985 Y1545.912
X2239.985 Y1545.912
G00 Z5.

X2223.485 Y1535.827
G01 Z-2. F150
G03 X2223.485 Y1535.827 I-3.5 F450
G00 Z5.

X2260.593 Y1635.769
G01 Z-2. F150
X2243.452 Y1635.769 F450
X2243.452 Y1658.623
X2260.593 Y1658.623
X2260.593 Y1635.769
G00 Z5.

X2222.883 Y1647.196
G01 Z-2. F150
G03 X2222.883 Y1647.196 I-2.285 F450
G00 Z5.

X2241.347 Y1692.667
G01 Z-2. F150
X2241.347 Y1712.667 F450
X2256.347 Y1712.667
X2256.347 Y1692.667
X2241.347 Y1692.667
G00 Z5.

X2225.847 Y1702.667
G01 Z-2. F150
```

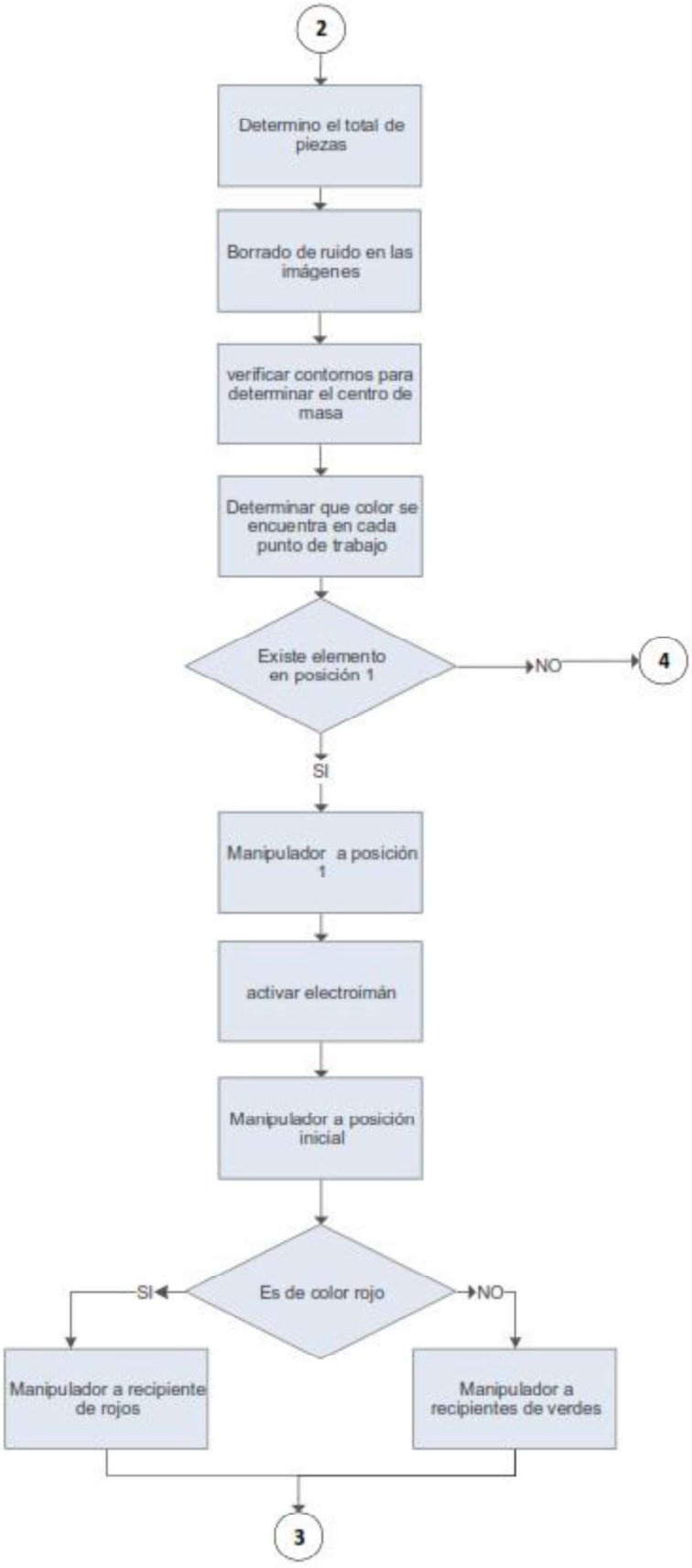
```
G03 X2225.847 Y1702.667 I-4.5 F450
G00 Z5.

X2323.444 Y1635.769
G01 Z-2. F150
X2277.734 Y1635.769 F450
X2277.734 Y1658.623
X2323.444 Y1658.623
X2323.444 Y1635.769
G00 Z5.

X2323.444 Y1580.604
G01 Z-2. F150
X2277.734 Y1580.604 F450
X2277.734 Y1603.459
X2323.444 Y1603.459
X2323.444 Y1580.604
G00 Z5.

X2392.008 Y1580.604
G01 Z-2. F150
X2334.871 Y1580.604 F450
X2334.871 Y1603.459
X2392.008 Y1603.459
X2392.008 Y1580.604
G00 Z5.

X2402.864 Y1592.032
G01 Z-2. F150

X2409.149 Y1603.459
G01 Z-2. F150
X2454.858 Y1603.459 F450
X2454.858 Y1580.604
X2409.149 Y1580.604
X2409.149 Y1603.459
G00 Z5.

X2392.008 Y1635.769
G01 Z-2. F150
X2334.871 Y1635.769 F450
X2334.871 Y1658.623
X2392.008 Y1658.623
X2392.008 Y1635.769
G00 Z5.

X2402.864 Y1647.196
G01 Z-2. F150
G03 X2402.864 Y1647.196 I-2.285 F450
G00 Z5.

X2409.149 Y1658.623
G01 Z-2. F150
X2454.858 Y1658.623 F450
```

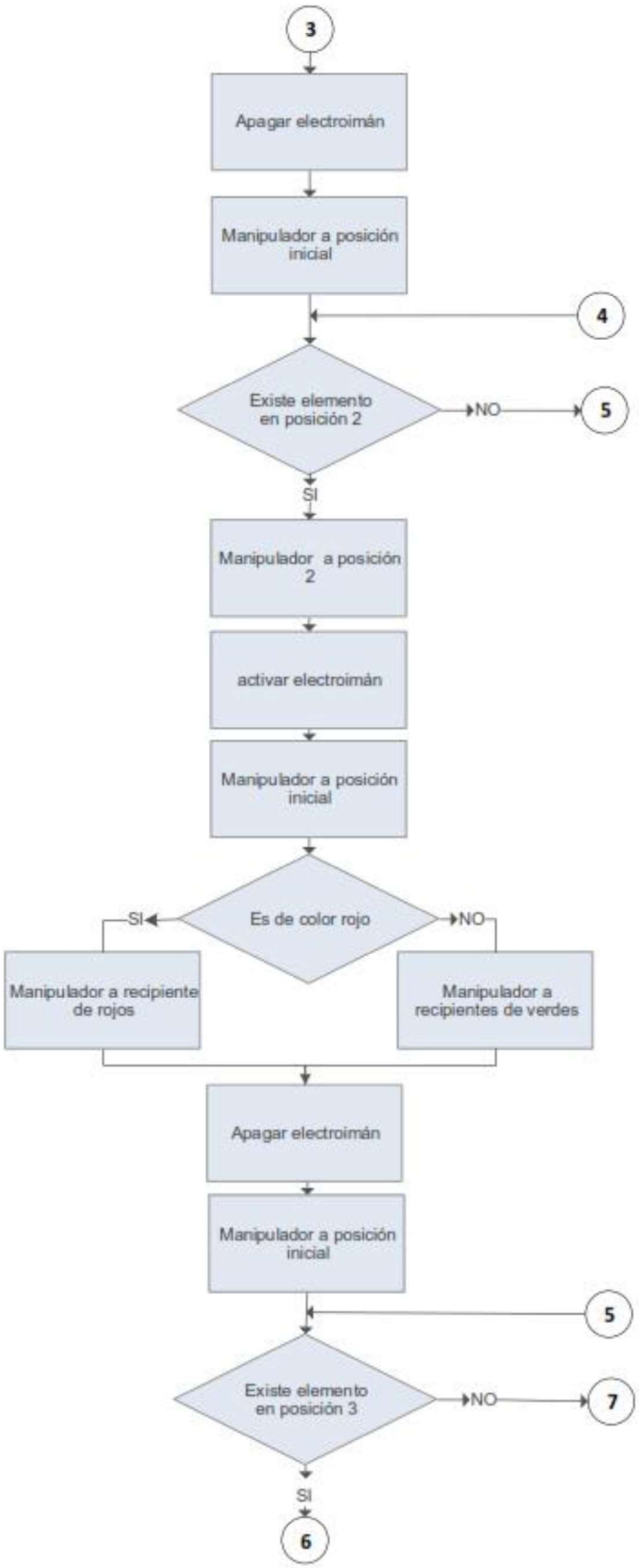
```
X2454.858 Y1635.769
X2409.149 Y1635.769
X2409.149 Y1658.623
G00 Z5.

X2449.144 Y1627.198
G01 Z-2. F150
G03 X2449.144 Y1667.194 J19.998 F450
G01 X2220.597 Y1667.194
G03 X2220.597 Y1627.198 J-19.998
G01 X2449.144 Y1627.198
G00 Z5.

X2449.144 Y1572.034
G01 Z-2. F150

X2441.985 Y1545.827
G01 Z-2. F150
X2441.985 Y1525.827 F450
X2559.985 Y1525.827
X2559.985 Y1545.827
X2441.985 Y1545.827
G00 Z5.

X2404.485 Y1535.827
G01 Z-2. F150
G03 X2404.485 Y1535.827 I-4.5 F450
G00 Z5.

X2382.847 Y1702.667
G01 Z-2. F150
G03 X2382.847 Y1702.667 I-2. F450
G00 Z5.

X2373.347 Y1712.667
G01 Z-2. F150
X2271.347 Y1712.667 F450
X2271.347 Y1692.667
X2373.347 Y1692.667
X2373.347 Y1712.667
G00 Z5.

X2388.347 Y1722.667
G01 Z-2. F150
X2388.347 Y1682.667 F450
X2221.347 Y1682.667
G02 X2221.347 Y1722.667 J20.
G01 X2388.347 Y1722.667
G00 Z5.

X2579.985 Y1555.827
G01 Z-2. F150
X2219.985 Y1555.827 F450
G03 X2219.985 Y1515.827 J-20.
```

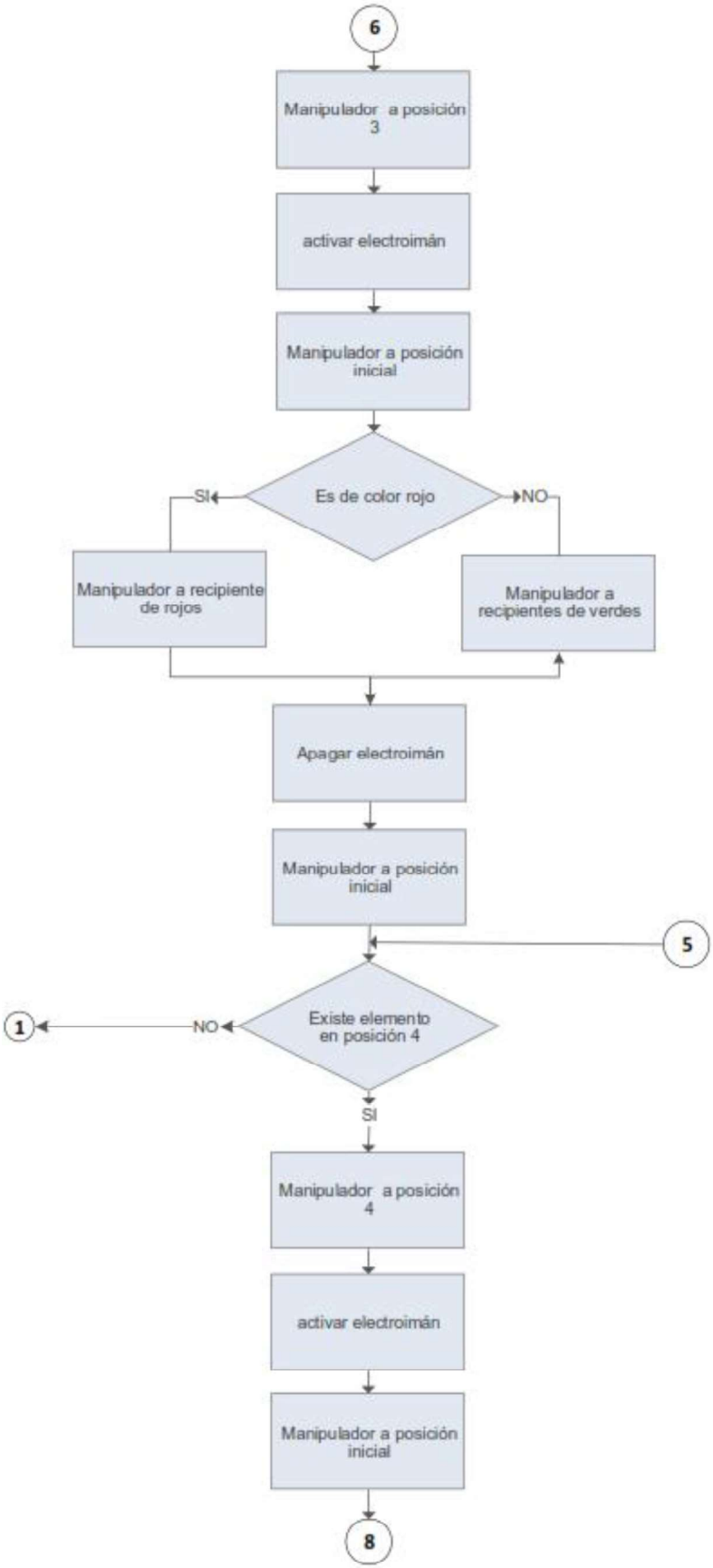
```
G01 X2579.985 Y1515.827
G03 X2579.985 Y1555.827 J20.
G00 Z5.

X2584.485 Y1535.827
G01 Z-2. F150
G03 X2584.485 Y1535.827 I-4.5 F450
G00 Z5.
X0. Y0.
M5
M30
%
```

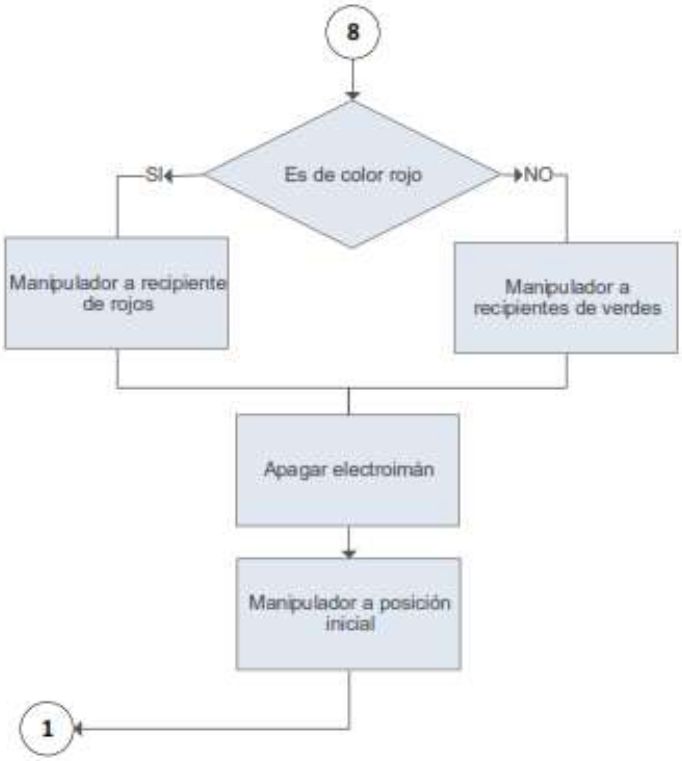
ANEXO 6: DIAGRAMA DE FLUJO DEL CONTROL DEL MANIPULADOR


Control del Manipulador
Ejecución del programa en el computador
Se inicia comunicación serial con la tarjeta arduino
1
Pulsante de inicio activado
NO
SI
Encendido de cámara y espera para que ajuste el enfoque
Captura de imágenes
Cambia a formato HSV(tonalidad, saturación, valor)
Discriminar lo que no se encuentra dentro del rango de rojos
Determinar cuantas piezas rojas hay
Discriminar lo que no se encuentra dentro del rango de verdes
Determinar cuantas piezas verdes hay
2


2
Determino el total de piezas
Borrado de ruido en las imágenes
verificar contornos para determinar el centro de masa
Determinar que color se encuentra en cada punto de trabajo
Existe elemento en posición 1
NO
4
SI
Manipulador a posición 1
activar electroimán
Manipulador a posición inicial
Es de color rojo
SI
NO
Manipulador a recipiente de rojos
Manipulador a recipientes de verdes
3


3
Apagar electroimán
Manipulador a posición inicial
4
Existe elemento en posición 2
NO
5
SI
Manipulador a posición 2
activar electroimán
Manipulador a posición inicial
Es de color rojo
SI
NO
Manipulador a recipiente de rojos
Manipulador a recipientes de verdes
Apagar electroimán
Manipulador a posición inicial
5
Existe elemento en posición 3
NO
7
SI
6


6
Manipulador a posición 3
activar electroimán
Manipulador a posición inicial
Es de color rojo
SI
NO
Manipulador a recipiente de rojos
Manipulador a recipientes de verdes
Apagar electroimán
Manipulador a posición inicial
5
Existe elemento en posición 4
1
NO
SI
Manipulador a posición 4
activar electroimán
Manipulador a posición inicial
8

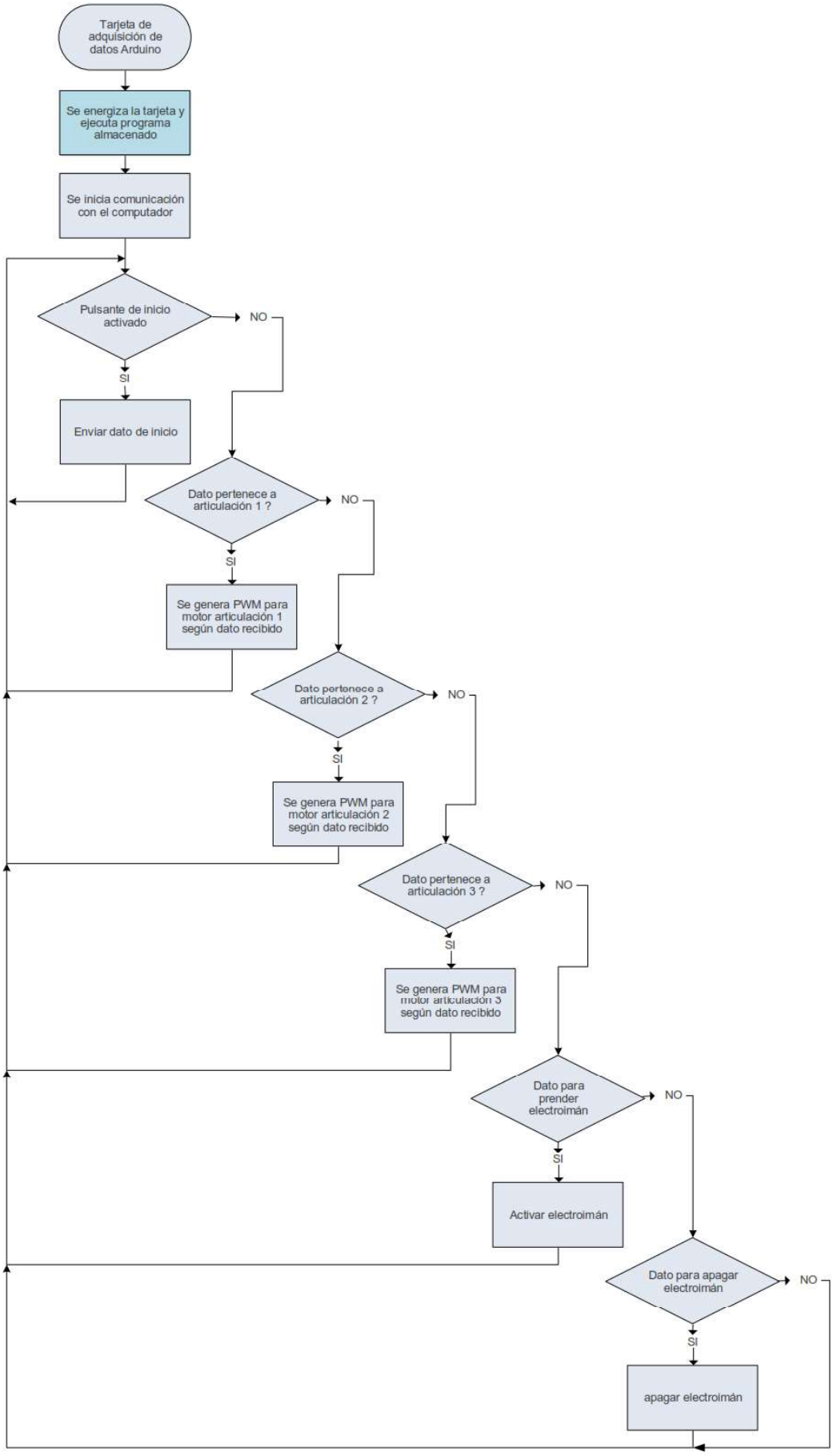

8
Es de color rojo
SI
NO
Manipulador a recipiente de rojos
Manipulador a recipientes de verdes
Apagar electroimán
Manipulador a posición inicial
1

ANEXO 7: CÓDIGO GENERADO PARA LA TARJETA ARDUINO

```
#include <Servo.h>
Servo s1,s2,s3;
int cnt=0,val1,val2,val3;
int x,y,z,estado=0;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  s1.attach(8);
  s2.attach(9);
  s3.attach(10);
  pinMode(7, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);
  pinMode(5, INPUT_PULLUP);
  s1.write(201);
  s1.write(145);
  s1.write(2);
  delay(1000);
}

void loop() {
  int p1 = digitalRead(7);
  int p2 = digitalRead(6);
  int p3 = digitalRead(5);
  if(p1==0 && p2==0 && p3==0 && estado==0) {cnt=0;  envia();estado=1;delay(200);}
  if(p1==1 && p2==0 && p3==0 && estado==0) {cnt=1;  envia();estado=1;delay(200);}
  if(p1==0 && p2==1 && p3==0 && estado==0) {cnt=2;  envia();estado=1;delay(200);}
  if(p1==1 && p2==1 && p3==0 && estado==0) {cnt=3;  envia();estado=1;delay(200);}
  if(p1==0 && p2==0 && p3==1 && estado==0) {cnt=4;  envia();estado=1;delay(200);}
  if(p1==1 && p2==0 && p3==1 && estado==0) {cnt=5;  envia();estado=1;delay(200);}
  if(p1==0 && p2==1 && p3==1 && estado==0) {cnt=6;  envia();estado=1;delay(200);}
  if(p1==1 && p2==1 && p3==1) {cnt=7; envia(); estado=0;delay(100);}

  if (Serial.available() > 0) {
    int dato=Serial.read();
```

```
    if (dato<=85)
      x=map(dato,0,85,0,180);
      s1.write(x);
    if (dato>85 && dato<=170)
      y=map(dato,86,170,0,180);
      s2.write(y);
    if (dato>170 && dato<=253)
      z=map(dato,171,253,0,180);
      s3.write(z);

    if(dato==254)
       digitalWrite(13,HIGH);
    if(dato==255)
       digitalWrite(13,LOW);
  }
}
void envia(){
   Serial.print("A");Serial.print(cnt);Serial.println("B");delay(200);
  }
```

ANEXO 8: DIAGRAMA DE FLUJO DEL PROGRAMA EN LA TARJETA ARDUINO.


Tarjeta de adquisición de datos Arduino
Se energiza la tarjeta y ejecuta programa almacenado
Se inicia comunicación con el computador
Pulsante de inicio activado
NO
SI
Enviar dato de inicio
Dato pertenece a articulación 1 ?
NO
SI
Se genera PWM para motor articulación 1 según dato recibido
Dato pertenece a articulación 2 ?
NO
SI
Se genera PWM para motor articulación 2 según dato recibido
Dato pertenece a articulación 3 ?
NO
SI
Se genera PWM para motor articulación 3 según dato recibido
Dato para prender electroimán
NO
SI
Activar electroimán
Dato para apagar electroimán
NO
SI
apagar electroimán

ANEXO 9: MANUAL DE USUARIO.

Descripción del sistema

El Controlador está diseñado para que cumpla con las siguientes características:

Al pulsar el botón de inicio del proceso del manipulador, el control captura una imagen con la cámara WEB, determina el total de piezas rojas y verdes, determina la posición de cada pieza y discrimina a las piezas que se encuentran fuera de los puntos de trabajo del manipulador.

Con esto se determina en que puntos tengo piezas rojas y en cual verdes, si existe una pieza en la posición uno, ubica el brazo en esta posición, activa el electroimán, si la pieza es de color rojo mueve el brazo al depósito de rojos, si es de color verde mueve el brazo al depósito de verdes y desactiva el electroimán para soltar la pieza, esta operación la realiza por cada punto de trabajo si existe una pieza en dichos puntos, al terminar este proceso el controlador espera a que sea nuevamente pulsado el botón de inicio.

En la Figura 5.1 se muestra el manipulador con su partes.

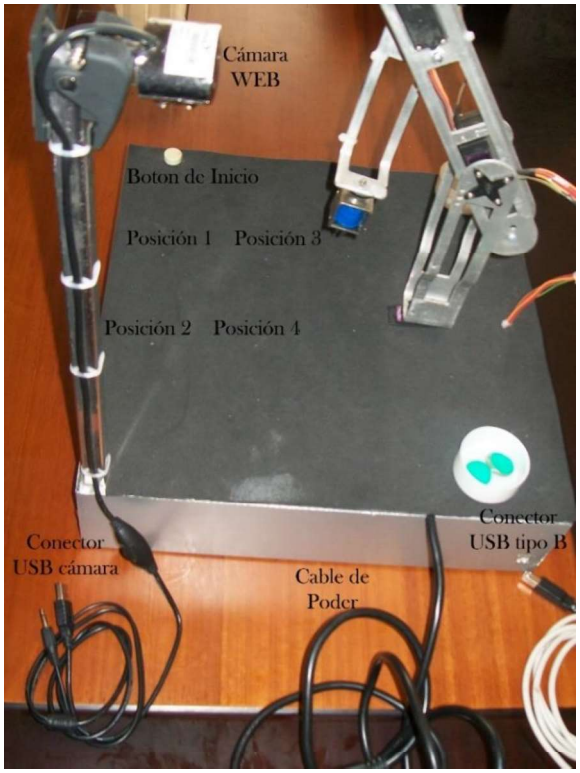


Figura 5.1. Partes del manipulador
(Fuente: Javier Acosta, 2018)

Puesta en Marcha

Para poner en funcionamiento el manipulador se deben seguir los siguientes pasos:

1. Conectar el manipulador a la energía eléctrica mediante el cable de poder a una toma de 110 VAC.
2. Encienda el computador en donde se encuentran instalados los programas de control del manipulador.
3. Conecte el cable USB de la cámara WEB del manipulador a un puerto USB el computador.
4. Conecte la tarjeta de adquisición de datos del manipulador mediante el cable USB tipo B de un extremo y tipo A del otro extremo, el conector Tipo B del Cable conéctelo al puerto USB del manipulador y el terminal Tipo A del cable a un puerto USB del computador.
5. Abra una terminal, acceda a la carpeta catkin_ws con el comando: ***cd catkin_ws***
   Luego escribimos el comando:
   rosrun rtt_ros deployer/home/javier/catkin_ws/ops.ops
   Se ejecuta el Deployer: ***contador.start***
   En la Figura 5.2 se muestra la terminal con los comandos para ejecutar el programa.

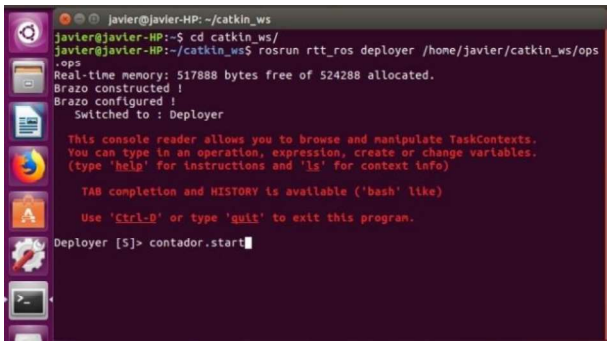


Figura 5.2. Ejecución de programa
(Fuente: Javier Acosta, 2018)

6. Abrir un terminal nuevo en la cual se acceda a la carpeta catkin_ws con el comando: **cd catkin_ws**
   Luego se ejecuta el programa en Phyton con el comando:
   python enlacePython.py
   En la Figura 5.3 se muestra los comandos para ejecutar el programa en Phyton.

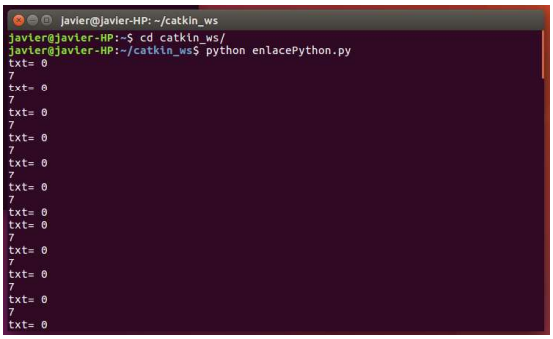


Figura 5.3. Ejecución de programa en Phyton
(Fuente: Javier Acosta, 2018)

Luego de estos pasos el manipulador está listo para operar.