DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TEMA: REPOTENCIACIÓN DEL SISTEMA DE CONTROL Y POTENCIA DE 2 BRAZOS ROBÓTICOS CRS A255

AUTOR: CAZA NEGRETE, STALIN XAVIER

DIRECTOR: ING. ERAZO SOSA, ANDRÉS SEBASTIÁN, M.Sc.

SANGOLQUÍ

2018

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, ¨***REPOTENCIACIÓN DEL SISTEMA DE CONTROL Y POTENCIA DE 2 BRAZOS ROBÓTICOS CRS A255***¨, fue realizado por el señor ***CAZA NEGRETE, STALIN XAVIER***, el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 22 Agosto del 2018

Ing. Andrés Erazo, M.Sc.

Director

C.C. 1720400082

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Yo, ***CAZA NEGRETE, STALIN XAVIER***, declaro que el contenido, ideas y criterios del trabajo de titulación: ¨***REPOTENCIACIÓN DEL SISTEMA DE CONTROL Y POTENCIA DE 2 BRAZOS ROBÓTICOS CRS A255***¨, es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armada ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 22 Agosto del 2018

Stalin Xavier Caza Negrete

C.C. 1716089246

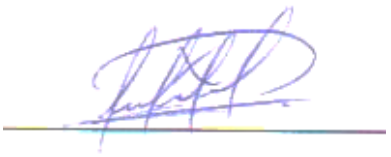
DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Yo, ***CAZA NEGRETE, STALIN XAVIER***, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación ¨*REPOTENCIACIÓN DEL SISTEMA DE CONTROL Y POTENCIA DE 2 BRAZOS ROBÓTICOS CRS A255*¨ en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 22 de Agosto del 2018



Stalin Xavier Caza Negrete

C.C. 1716089246

DEDICATORIA

La culminación de este proyecto se lo dedico a las personas más importantes en mi vida, Marco y Liria, mis padres, que siempre han estado al pendiente de mí, brindándome su apoyo incondicional, en todo momento, aun en esas largas desveladas de tareas universitarias, ya que sin ellos, no hubiese sido posible alcanzar este logro en mi vida.

Stalin Caza

AGRADECIMIENTO

Un agradecimiento enorme a mis padres por darme la vida y convertirme en un hombre de bien, por todo el esfuerzo y sacrificio, demostrándome día a día que los sueños se pueden lograr siempre que te lo propongas y te esfuerces de todo corazón.

A Marly e Irina mis dos hermanitas, que con cada una de sus locuras hacían que aun mis días más agobiantes o estresados en la universidad, se volvieran felices, siempre sacándome una carcajada son sus ocurrencias.

A mi hermano Erick, que a pesar de estar ocupado, nunca me ha negado su ayuda, y me ha enseñado a ponerle un buen animo a la vida, aunque parezca que las cosas van mal.

A Dayana, mi compañera y amiga incondicional quien me ha brindado todo su amor, consejos y paciencia durante los momentos difíciles del proyecto, y quien me motiva a seguir adelante en esos momentos en que ni yo creo poder seguir.

A toda mi familia en general, abuelos, tíos y primos por sus palabras de aliento en todo este trayecto, pero en especial a Brayan quien desde el cielo me bendice, ha sido mi guía en momentos difíciles y sé que estaría muy orgulloso de mí.

A mis amigos, con quienes compartí momentos buenos y experiencias únicas, a lado de quienes he crecido personal y profesionalmente, gracias por estar ahí en todo momento.

Stalin Caza

ÍNDICE DE CONTENIDOS

CERTIFICACIÓN DEL DIRECTOR…..i
AUTORÍA DE RESPONSABILIDAD.....ii
AUTORIZACIÓN.....iii
DEDICATORIA.....iv
AGRADECIMIENTO.....v
ÍNDICE DE CONTENIDOS.....vi
ÍNDICE DE TABLAS.....ix
ÍNDICE DE FIGURAS.....xi
RESUMEN.....xiv
ABSTRACT.....xv
CAPÍTULO I.....1
INTRODUCCIÓN 1
1.1 Antecedentes 1
1.2 Justificación e importancia.....5
1.3 Alcance del proyecto.....6
1.4 Objetivos 8
1.4.1 Objetivo General 8
1.4.2 Objetivos Específicos.....8
CAPÍTULO II 10
ESTADO INICIAL 10
2.1. Hardware 10
2.1.1. Estructura 13
2.1.2. Motores y frenos.....13
2.1.3. Sensores 14
2.2. Interfaz de conversión 15
2.3. Sistema de control 16
2.4. Conexiones eléctricas.....17
CAPÍTULO III 19

OPTIMIZACION DEL HARDWARE 19
3.1. Optimización en robots 19
3.1.1. Opciones de mejora 19
3.1.2. Mejoras realizadas 20
3.1.2.1. Montaje y prueba de motores 20
3.1.2.2. Activación de frenos independientes 21
3.1.2.3. Acoplar freno a la articulación de la cintura 21
3.1.2.4. Diseño de nuevo encoder 22
3.2. Interfaz de conversión 26
3.2.1. Requerimientos 26
3.2.2. Selección del microcontrolador 28
3.2.3. Diseño de la etapa de potencia 29
3.2.3.1. Motores 29
3.2.3.2. Frenos 30
3.2.4. Diseño de la etapa de control 31
3.2.5. Implementación de la interfaz de conversión 32
3.2.6. Conexión y montaje de la interfaz de conversión 33
CAPÍTULO IV 36
OPTIMIZACIÓN DEL SISTEMA DE CONTROL 36
4.1. Características e instalación de software 37
4.1.1. ROS. 37
4.1.1.1. Infraestructura de comunicación. 37
4.1.1.2. Partes fundamentales de la comunicación ROS 37
4.1.1.3. Instalación ROS 39
4.1.1.4. Creación del espacio de trabajo para ROS 41
4.1.2. Interfaz de conversión con ROS 41
4.1.2.1. Instalación IDE Arduino 42
4.1.2.2. Instalación Teensyduino 42
4.1.2.3. Instalación Rosserial 43
4.1.3. Controlador con ROS 44

4.1.3.1. Creación de un nodo ROS C++ en Ubuntu 44
4.1.4. Entorno de simulación V-REP con ROS 46
4.1.4.1. Instalación De V-REP 46
4.1.4.2. Instalación de Rosinterface 46
4.2. Aplicación 48
4.3. Algoritmos de control 49
4.3.1. Algoritmo para el entono de simulación V-REP 49
4.3.1.1. Tópicos ROS en V-REP 52
4.3.2. Algoritmo para el controlador difuso C++ 53
4.3.2.1. Lógica difusa 53
4.3.2.2. Diseño del controlador difuso 55
4.3.2.3. Tópicos ROS en controladores difusos 59
4.3.3. Algoritmo para la interfaz de conversión 62
4.3.3.1. Tópicos ROS para la interfaz de conversión 63
CAPÍTULO V 66
PRUEBAS Y RESULTADOS 66
5.1. Resultados en la optimización de hardware 66
5.2. Resultados de la creación de interfaz de conversión 67
5.3. Resultados del sistema de control de posición angular de motores 70
5.3.1. Limitaciones 71
5.4. Pruebas de error en el movimiento de articulaciones 72
5.5. Prueba de tiempo de procesamiento en tarjetas 75
5.6. Prueba de tiempo de procesamiento en controladores difusos 77
5.7. Prueba de tiempo en replicar movimiento 78
CAPÍTULO VI 80
CONCLUSIONES Y RECOMENDACIONES 80
6.1. Conclusiones 80
6.2. Recomendaciones 81
REFERENCIAS BIBLIOGRÁFICAS 83

ÍNDICE DE TABLAS

Tabla 1. *Resumen de condiciones de hardware iniciales*.....*15*

Tabla 2. *Características eléctricas de los motores* *20*

Tabla 3. *Resumen de requerimientos para el microcontrolador*.....*27*

Tabla 4. *Características más relevantes de la Teensy 3.6* *28*

Tabla 5. *Tabla de verdad del módulo L298N*.....*30*

Tabla 6. *Límites angulares de articulaciones* *51*

Tabla 7. *Editores y suscriptores del robot virtual_2*.....*52*

Tabla 8. *Editores y suscriptores del robot virtual_1*.....*52*

Tabla 9. *Variables lingüísticas para el error*.....*55*

Tabla 10. *Variables lingüísticas para el PWM* *57*

Tabla 11. *Reglas de control para el controlador difuso* *58*

Tabla 12. *Compensación de PWM* *58*

Tabla 13. *Editores y suscriptores del controlador difuso del robot_2*.....*59*

Tabla 14. *Editores y suscriptores del controlador difuso del robot_1*.....*60*

Tabla 15. *Editores y suscriptores de la interfaz de conversión*.....*63*

Tabla 16. *Editores y suscriptores de la interfaz de conversión*.....*64*

Tabla 17. *Consumo de corriente dispositivos de control*.....*69*

Tabla 18. *Prueba de error en articulación de la cintura.* *72*

Tabla 19. *Prueba de error en articulación del hombro.* *73*

Tabla 20. *Prueba de error en articulación del codo*.....*73*

Tabla 21. *Prueba de error en articulación del pitch de la muñeca.* *74*

Tabla 22. *Prueba de error en articulación del roll de la muñeca.....74*

Tabla 23. *Tiempo de procesamiento Teensy sin consignas.....75*

Tabla 24. *Tiempo de procesamiento Teensy con consignas.....76*

Tabla 25. *Tiempo de procesamiento controlador sin consignas.....77*

Tabla 26. Tiempo de procesamiento controlador con consignas. 78

Tabla 27. *Prueba replicar movimiento.....78*

ÍNDICE DE FIGURAS

Figura 1. Servicio de robots para el uso profesional.....1

Figura 2. Etapas de repotenciación del robot manipulador CRS A255 8

Figura 3. Articulaciones.....11

Figura 4. Ubicación de sensor, motor y freno.....11

Figura 5. Estado inicial de los manipuladores 12

Figura 7. Estado inicial de los manipuladores CRS A255, vista trasera.....12

Figura 8. Problemas en la estructura.....13

Figura 9. Motor del manipulador CRS A255.....14

Figura 10. Encoders faltantes.....14

Figura 11. Interfaz de conversión inicial.....16

Figura 12. Interfaz de conversión inicial inutilizable.....16

Figura 13. Conectores traseros del Robot CRS A255.....17

Figura 14. Descripción de pines de conectores del Robot CRS A255.....17

Figura 15. Descripción de pines de conectores del Robot CRS A255.1.....18

Figura 16. Conexión Original (IZQ) Conexión Actual (DER). 21

Figura 17. Partes de un encoder Fuente: (Herrera, 2011) 22

Figura 18. Encoder diseñado: Base de acople (IZQ), disco ranurado (DER). 23

Figura 19. Encoder adquirido.....23

Figura 20. Base diseñada para encoder adquiridos 24

Figura 21. Base impresa para encoder adquiridos 24

Figura 22. Montaje de encoder nuevos, vista superior (IZQ), vista lateral (DER). 25

Figura 23. Diseño 3D de la nueva carcasa 25

Figura 24. Nueva carcasa sobre el manipulador CRS A255 26

Figura 25. Conexión del módulo L298N 29

Figura 26. Circuito de elevación de potencia para los frenos 30

Figura 27. Circuito del regulador de voltaje LM317 32

Figura 28. Diseño del circuito para la interfaz de conversión. 32

Figura 29. Diseño del circuito para la interfaz de conversión. PCB 33

Figura 30. Interfaz de conversión. Vista inferior (IZQ), Vista superior (DER) 33

Figura 31. Diseño de conectores en AutoCAD 24 pines (IZQ), 57 pines (DER) 34

Figura 32. Instalación de conectores y fusibles nuevos 34

Figura 33. Conexión de cables a los pines de conectores SPEC-MIL. 35

Figura 34. Montaje de ambos circuitos sobre carcasa 35

Figura 35. Implementación completa dentro de la carcasa 35

Figura 36. Esquema de operación del control de posición de motores 36

Figura 37. Elementos de ROS. 38

Figura 38. Modificación del archivo .bashrc. 40

Figura 39. Selección de tarjeta Teensy 3.6 en IDE Arduino 43

Figura 40. Instalación de Rosserial 44

Figura 41. Carga del complemento RosInterface. 48

Figura 42. Aplicación: Reproducción de movimiento a modo de espejo 49

Figura 43. Robot Virtual semejante a CRS A255 (IZQ), Control de mando de posición de las articulaciones (DER) 50

Figura 44. Escenario en entorno de simulación V-REP 50

Figura 45. Grado de pertenencia a un suceso utilizando lenguaje común 54

Figura 46. Partes de un controlador difuso Fuente: (Kouro samir, 2010).....54

Figura 47. Lazo de control para la posición de motores del manipulador CRS A 255.....55

Figura 48. Funciones de pertenencia correspondientes al error.....56

Figura 49. Funciones de pertenencia correspondientes al PWM 57

Figura 50. Lógica en interfaz de conversión.....63

Figura 51. Estado inicial (IZQ) y estado actual (DER) del hardware de los manipuladores CRS A255.....66

Figura 52. Estado inicial (IZQ) y estado actual (DER) de los encoders de los manipuladores CRS A255.....67

Figura 53. Estado inicial (ARRIBA) y estado actual (DEBAJO) de los manipuladores CRS A255. 67

Figura 54. Estado inicial (IZQ) y estado actual (DER) de carcasa internamente. 68

Figura 55. Estado inicial (IZQ) y estado actual (DER) de conectores en carcasa. 68

Figura 56. Estado inicial (IZQ) y estado actual (DER) de interfaz de conversión. 68

Figura 57. Estado inicial (ARRIBA) y estado actual (DEBAJO) de la interfaz de conversión montadas en su carcasa.....69

Figura 58. Disipador para evitar el sobrecalentamiento en regulador de 3.3Vdc 70

Figura 59. Montaje de manipuladores para la aplicación 70

Figura 60. Monitorización en V-REP 71

Figura 61. Toma de tiempos de procesamiento de tarjetas Teensy 3.6.....75

Figura 62. Toma de tiempos de procesamiento en los controladores 77

RESUMEN

En el presente proyecto de investigación se realiza la repotenciación de dos manipuladores CRS A255, tanto en su sistema de control como el sistema de potencia, además de dotarlos de un entorno de simulación donde se podrá monitorizar su funcionamiento. Para ello se ha previsto una aplicación en la cual al mover un robot desde su entorno de simulación el segundo robot debe copiar los movimientos pero a modo de espejo. El desarrollo se ha enfocado en tres etapas: la primera etapa es la optimización de hardware donde se hacen todas las correcciones necesarias para que los manipuladores queden completamente íntegros y operativos, incluyendo la creación de la interfaz de conversión, encargada de la transmisión de datos entre el computador y el manipulador, debido a que en un inicio no se contaba con la misma; la segunda es dotar de un sistema de control de posición angular para cada articulación de los manipuladores robóticos haciendo uso de lógica difusa; y la tercera etapa abarca la adaptación de un robot dentro del software de simulación para con su ayuda controlar y monitorizar el manipulador real. Para llevar a cabo este proyecto se utilizó una estructura de comunicación ROS que funciona sobre el sistema operativo Ubuntu, permitiendo la transferencia de datos entre los manipuladores robóticos CRS A255 mediante la tarjeta Teensy 3.6, su controlador de lógica difusa y el entorno de simulación V-REP.

PALABRAS CLAVE:

- **MANIPULADOR ROBOTICO CRS A255**
- **REPOTENCIACION**
- **SISTEMA OPERATIVO ROBOT (ROS)**
- **V-REP**

ABSTRACT

In the present research project the repowering of two CRS A255 manipulators is carried out, both in its control system and the power system, in addition to providing them with a simulation environment where their operation can be monitored. For this, an application has been foreseen in which when moving a robot from its simulation environment, a second robot must copy the movements but in a mirror mode. The development has focused on three stages the first stage is the optimization of the hardware where all the necessary corrections are made so that the manipulators are completely intact and operative, including the creation of the conversion interface, responsible for the transmission of data between the computer and the manipulator, because in the beginning it did not have the same; the second is to provide an angular position control system for each articulation of robotic manipulators using fuzzy logic; and the third stage involves the adaptation of a robot within the simulation software to help control and monitor the real manipulator. To carry out this project we used a ROS communication structure that works on the Ubuntu operating system, allowing the transfer of data between the robotic manipulators CRS A255 through the Teensy 3.6 card, its fuzzy logic controller and the V-REP simulation environment.

KEYWORDS:

- **CRS A255 ROBOTIC MANIPULATOR**
- **REPOTENTIATION**
- **ROBOT OPERATING SYSTEM (ROS)**
- **V-REP**

CAPÍTULO I

INTRODUCCIÓN

1.1 Antecedentes

En el mundo globalizado en el cual vivimos, se puede evidenciar claramente que la competencia va creciendo en los sistemas de producción. Una parte fundamental para la creación de productos a gran escala, con calidad y eficiencia, es mediante la implementación de robots industriales flexibles. De acuerdo a datos estadísticos tomados de la Federación Internacional de Robótica (IFR); se estima un fuerte aumento del uso de robots industriales en distintas aplicaciones de manufactura. La Figura 1 muestra el número de unidades robóticas que se han utilizado en la industria los años 2014 y 2015, estimando un aumento de dichas unidades para el periodo 2016-2019. (Haegele, 2016).

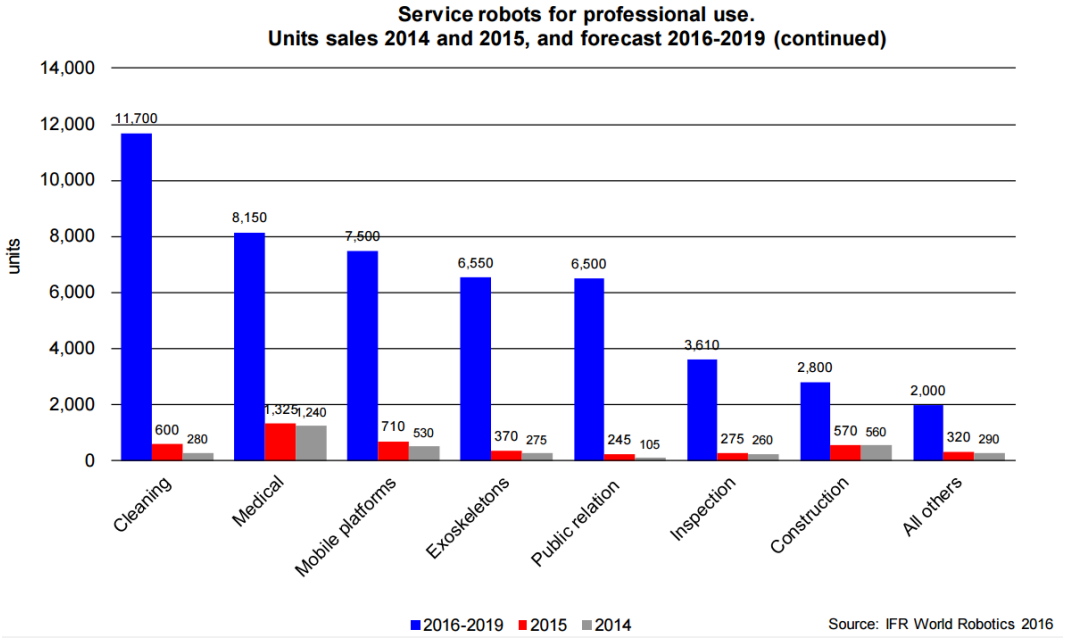


Figura 1. Servicio de robots para el uso profesional
Fuente: (Haegele, 2016)

La rapidez, precisión y versatilidad de los robots manipuladores, los hacen adecuados para varias labores de manufactura como ensamble, pintura, soldadura, transporte de materiales, entre otras; donde las habilidades de operadores humanos no son suficientes. Sin embargo, adaptar este tipo de elemento a las líneas de producción no es una tarea sencilla, ya que se debe tener un alto grado de conocimiento en cuanto a su funcionamiento e instalación para que no existan inconvenientes al momento de su puesta en funcionamiento.

Según Ochoa (2016), los robots deben ser aceptados y poder trabajar en varios campos como el industrial por lo que se desarrollan nuevos métodos para mejorar su funcionalidad como lo es el Sistema Operativo Robótico (ROS). Este representa un gran avance en el campo de la robótica debido a que su código es abierto y cuenta con gran cantidad de estructuras y algoritmos ya diseñados, lo cual evita iniciar desde cero cualquier proyecto. (Ochoa, Aguiar, & Erazo, 2016).

Gracias a las bondades de ROS ha sido posible el desarrollo de múltiples proyectos aplicados a distintas clases de robots, en diversas áreas, como se detalla en los siguientes ejemplos:

- La integración de dispositivos en un robot quirúrgico teleoperado, en el cual el principal problema era la comunicación e interacción entre tres brazos robóticos y los componentes que conformaban dicho robot, lo cual se solucionó por medio de la jerarquización UML de dos etapas; una que se encargaba de la gestión de entradas y salidas, y la otra gestión netamente de la comunicaciones entre ellas, de esta forma se logró el correcto funcionamiento del robot. (Bauzano, Fernandez/Iribar, López, Renteria, & Muñoz, 2015).
- Otro trabajo en el cual ROS ha sido de ayuda es al implementar interfaces gráficas para el mapeado de entornos y navegación en la WEB, permitiendo al usuario controlar de forma remota el robot móvil al desplazarse por un entorno y paralelamente realizar el mapeado de todo

su alrededor, gracias a herramientas, paquetes y algoritmos de las bibliotecas propias de ROS. (Rapado, 2016).

- También se está utilizando en proyectos de la NASA como se puede ver en ROS Hexápodo (2016), en el cual se utilizó el sistema operativo robótico para dar a un robot hexápodo la posibilidad ser controlado de manera funcional mediante un mando de PlayStation3. Se logró adaptar toda la programación ya realizada en C++ a ROS. Además se integró al sistema el software de código abierto Arduino, para con ello manipular el robot desde un tablero con pulsadores y potenciómetros. (Davis & Bankieris, 2016).

Como se ha detallado anteriormente, ROS es muy versátil y funcional en toda clase de robots. Centrándose más en el tema de este proyecto de titulación, podemos ver su uso en manipuladores robóticos industriales:

- En el trabajo de García (2016) se realiza el diseño, la construcción y control de un manipulador construido a partir de impresiones en 3D, donde se utilizan dispositivos como Arduino, Raspberry Pi y un computador, para lo cual se crearon tres programas para ROS que permiten integrar todos los componentes y así controlar el brazo robótico, utilizando el protocolo de comunicación Rosserial. (Garcia, 2016).
- Dentro del campo de los manipuladores robóticos hay que destacar también el trabajo realizado por Samper (2016), el cual utilizó algoritmos de ROS para controlar y monitorizar el funcionamiento de un manipulador robótico de 6 grados de libertad, encargado de tareas de inspección y mantenimiento en el tren de inspección remota en el acelerador de Hadrones, de la Organización Europea para la Investigación Nuclear (CERN), debido a que en dicho lugar

existen altas concentraciones de radiación y el trabajo no puede ser realizado directamente por humanos. (Samper, 2016).

Dentro de la Universidad de las Fuerzas Armadas "ESPE" también se han realizado varios proyectos haciendo uso del Sistema Operativo Robótico, dentro de ellos se encuentran los siguientes:

- Un robot social desarrollado por Mejía (2016), el cual utiliza ROS para integrar todos sus componentes y comunicaciones; además gracias a sus librerías permite el reconocimiento de rostros e interacción con las personas. (Mejía Silva & Núñez Arroba, 2016).
- Otro proyecto interesante es el realizado por Garzón (2016), que cuenta con un robot móvil para el reconocimiento, exploración y mapeo de distintos entornos. Esto se logró utilizando SLAM dentro de ROS, con lo cual se puede notar la versatilidad de este sistema operativo para distintos tipos de aplicaciones. (Garzón Jaramillo & Obando Maldonado, 2016).
- En los trabajos realizados por Chimarro (2015) y Tumbaco (2014), referentes al diseño y construcción de robots, circular y delta respectivamente; ambos utilizan ROS para el control y monitorización de los mismos. (Chimarro Amaguaña & Enríquez Herrera, 2015) (Tumbaco Mendoza & Quimbita Zapata, 2014).

En la universidad no se ha realizado ningún trabajo que conjugue ROS y los manipuladores robóticos industriales como el CRS A255, pero se cuenta con varios proyectos que involucran dichos robots como son:

- En el proyecto de titulación desarrollado por Rivera (2015), se plantea el diseño e implementación de técnicas de control con lógica difusa para los actuadores del brazo robótico CRS A255, rediseñando la tarjeta de control y potencia; lo cual permitió un control bastante

preciso de los movimientos del robot. Uno de los puntos favorables de su trabajo es que se da una solución en cuanto a la tarjeta para realizar el control (STM32F4 Discovery), la cual es muy versátil y de fácil programación; esto sirve para realizar la repotenciación del sistema de control de una manera más sencilla. (Rivera, 2015).

- El proyecto de titulación realizado por Casa (2014), detalla la realización del software de programación y operación del brazo robótico CRS A255, remplazando el programa por defecto entregado por el proveedor del robot. Para ello se utilizó el lenguaje de programación JAVA, que envía una cadena de caracteres a los microcontroladores esclavos MICROCHIP 16F88, que decodifican la orden y ejecutan distintos movimientos del brazo robótico, cabe enfatizar que según Casa y Coque, los comandos "no implican cinemática o dinámica de robots", por lo cual el algoritmo general establece el desplazamiento simultaneo de los motores desde un punto hasta otro, mediante la localización de las coordenadas rectangulares de todas las articulaciones. (Coque & Casa, 2014).

Un punto importante de este proyecto es la integración de los manipuladores robóticos al computador sin la necesidad de contar con el software de defecto, contemplando así la idea de controlar este tipo de robot industrial desde un entorno de simulación ajeno al recomendado por el fabricante, en este caso mediante el uso de ROS.

1.2 Justificación e importancia

Actualmente el laboratorio de investigación del departamento de Eléctrica, Electrónica y Telecomunicaciones de la Universidad De Las Fuerzas Armadas "ESPE", tiene a su disposición dos brazos robóticos de la marca CRS Robotics modelo A255, que poseen una configuración del tipo antropomórfica, los cuales no se encuentran cien por ciento funcionales debido a que su

hardware o software no responde de manera adecuada, ya sea por su mal uso o por su desgaste con el pasar de los años.

Es por esta razón que el desarrollo de este proyecto de titulación es de gran importancia, ya que al finalizarlo se desea dejar ambos robots manipuladores con capacidades completamente funcionales. Esto ayudará a que las nuevas generaciones de estudiantes de la carrera Ingeniería Electrónica, Automatización y Control tengan un mejor sustento teórico y práctico en cuanto a la manipulación y control de robots industriales; además de aportar con una plataforma para el desarrollo de proyectos de investigación referente a esta temática.

Una de las mejoras significativas a los manipuladores robóticos CRS A255 actuales es la repotenciación del hardware, no realizada en ninguno de los trabajos anteriores. Esto permitirá tener un mejor funcionamiento, precisión y control de los mismos. Además se desarrollará un entorno de simulación en V-REP, con el cual se podrá controlar la posición angular de cada articulación y monitorizar el funcionamiento de dichos robots industriales.

Según José Rapado, "ROS es independiente del lenguaje de programación, pero da soporte nativo a lenguajes como C++, Python, Octave/Matlab, LISP o JAVA." (Rapado, 2016); Esto permite que en un futuro se pueda hacer cambios o mejoras sin necesidad de que los estudiantes tengan que aprender un lenguaje de programación especifico, sino que lo realicen en el que sea de su agrado, motivando de esta manera el aprendizaje en ellos.

1.3 Alcance del proyecto

Según a lo mencionado en las secciones anteriores, se necesita repotenciar los robots industriales CRS A255, teniendo en cuenta que la repotenciación es el conjunto de actividades que se realiza a un sistema para aumentar su capacidad de producción o eficiencia. Este proceso implica cambios,

aumento o reemplazo en alguno de los componentes operativos de dicho sistema, sin alterar el resto de componentes. (Calvo, 2006).

Teniendo esto como premisa, en el presente proyecto de titulación se realizará la restructuración, del hardware, sistema de potencia y sistema de control enfocado a la posición angular de cada motor ubicado en las articulaciones del manipulador, además de dotarlo con un entorno de simulación, para con ello lograr el funcionamiento de los brazos robóticos CRS A255.

Se realizará un análisis de la situación inicial de cada robot CRS A255 y de acuerdo a ello se llevará a cabo las debidas correcciones para su funcionamiento, destacando entre ellas las siguientes etapas:

- **Potencia.-** Comprende el acondicionamiento de motores, ya que varios de los mismos no funcionan o están deteriorados, lo cual provoca un funcionamiento erróneo en cada articulación del robot. Hay que tener en cuenta que estos manipuladores no disponen de un sistema de frenado por lo cual se debe realizar el diseño, creación o adaptación de este sistema, para tener movimientos más precisos y controlados.
- **Adquisición de señales.-** Para tener una mayor precisión en cada movimiento se debe calibrar, ajustar o cambiar los sensores (encoder, giroscopio, acelerómetro, etc.) en cada articulación, se requiere verificar el estado de las tarjetas de microcontroladores, para conocer sus características, funcionamiento, programación, comunicación e instalación en el manipulador, y de acuerdo a ello realizar la optimización de las mismas en el caso de ser necesario, lo cual servirá de retroalimentación al sistema de control y también al entorno de simulación. Uno de los puntos de interés de este proyecto es lograr una precisión alta en los trabajos que deba llevar a cabo el brazo robótico

- **Control.-** Se desarrollará un controlador de posición angular para cada uno de los motores correspondientes a las articulaciones del manipulador, sin tener en cuenta el modelo dinámico del sistema.
- **Simulación.-** En esta etapa se utilizará un robot con características similares a las del manipulador CRS A255 como modelo CAD dinámico en el entorno de simulación V-REP, donde se verificará el funcionamiento del manipulador del entorno físico.

  En la Figura 2 se puede observar las etapas a llevar a cabo en el desarrollo del presente proyecto.

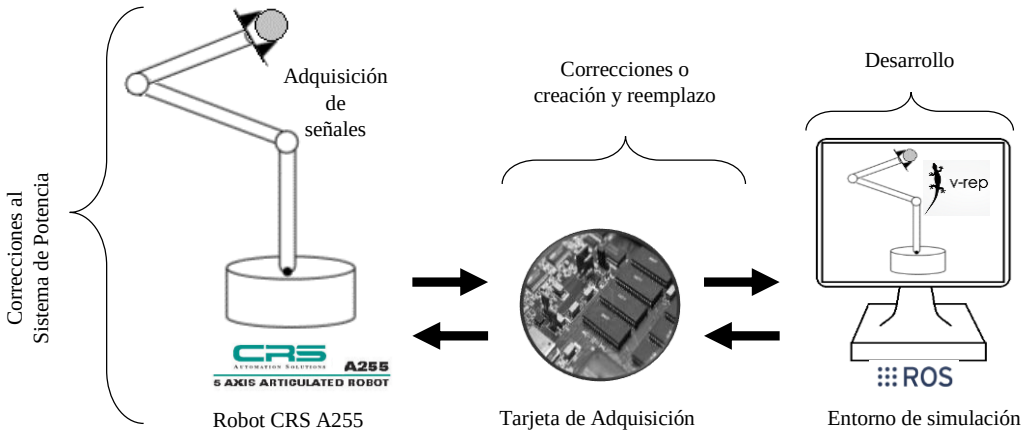


Figura 2. Etapas de repotenciación del robot manipulador CRS A255

1.4 Objetivos

1.4.1 Objetivo General

- Repotenciar el sistema de control de posición angular de motores y sistema de potencia de 2 brazos robóticos CRS A255, para ser controlados desde un entorno de simulación.

1.4.2 Objetivos Específicos

- Realizar un análisis y reparación del Hardware del manipulador robótico industrial de la marca CRS Robotics modelo A255.

- Verificar y optimizar el funcionamiento de las tarjetas de control y potencia para el brazo robótico CRS A255, a manera de tarjetas de adquisición de datos y potencia.
- Implementar un algoritmo de control basado en ROS en un computador con Sistema Operativo Linux y con comunicación hacia el entorno de simulación.
- Adaptar un robot con similares características al manipulador CRS A255 como modelo CAD dinámico en el entorno de simulación, para interactuar con el mismo.
- Integrar el funcionamiento del robot CRS A255 con el robot del entorno de simulación, a través de la lógica generada en ROS.
- Verificar el funcionamiento del robot CRS A255 del entorno físico y su modo y calidad de operación de acuerdo al robot del entorno simulado.

CAPÍTULO II

ESTADO INICIAL

Para llevar a cabo la repotenciación de los manipuladores robóticos CRS A255 de los cuales se dispone en las instalaciones de la Universidad de las Fuerzas Armadas “ESPE”, es necesario conocer las condiciones en las cuales se encuentran actualmente los mismos y a partir de ellas tomar acciones correctivas para mejorar su funcionamiento. Por ende se ha decidido realizar un análisis de las siguientes etapas:

- Hardware
- Interfaz de conversión
- Sistema de control
- Conexión eléctrica

Esta división permite una mejor organización conforme se avance en el proyecto, y a su vez ayuda a tener una visión general del sector en el cual se debe realizar mejoras o reemplazos en caso de ser necesarios.

2.1. Hardware

Gracias al manual de usuario del manipulador robótico CRSA255 se sabe la ubicación de sensores, motores, frenos y se definen cinco articulaciones que le dan 5 grados de libertad de movimiento al manipulador. Como se indica en la Figura 3 y Figura 4.

.

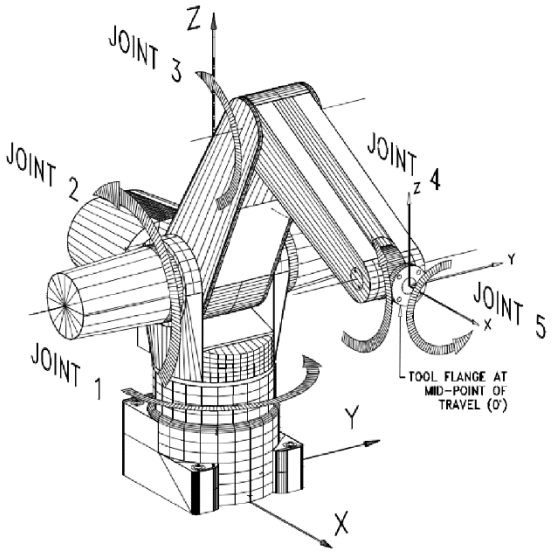


Figura 3. Articulaciones: JOINT 1: cintura, JOINT 2: hombro, JOINT3: codo, JOINT 4: muñeca pitch, JOINT 5: muñeca roll.
Fuente: (Corporation, 2000)

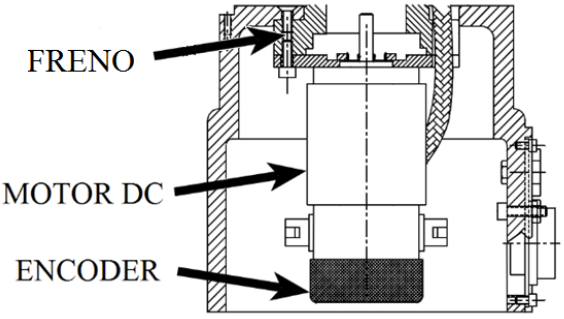


Figura 4. Ubicación de sensor, motor y freno
Fuente: (Corporation, 2000)

Cabe destacar que estas especificaciones se encuentran en cuatro de los cinco motores del manipulador, es decir en las articulaciones de hombro, codo, roll de muñeca y pitch de muñeca. La articulación de la cintura no cuenta con el freno, ya que los mismos solo se utilizan en caso de emergencia para detener el robot o cuando se desconecta su energía; además que el efecto de la gravedad no induce un cambio en el eje de movimiento de esta articulación.

El estado en el que fueron entregados los manipuladores robóticos al inicio de este proyecto de titulación se puede observar en la Figura 5 y Figura 6, se aprecia la ausencia de varios de sus componentes como sensores, motores, carcasa entre otros.

Figura 5. Estado inicial de los manipuladores

Figura 6. Estado inicial de los manipuladores CRS A255, vista trasera.

La falta de componentes o deterioro de los mismos únicamente se encontraron en uno de los manipuladores, ya que el otro nunca antes había sido expuesto a malas condiciones de uso que afecten su parte de hardware.

2.1.1. Estructura

El estado de todos los componentes de la estructura como son cadenas, engranes y piñones se encontraron en buenas condiciones, aunque con impurezas y en mala posición; lo cual no permite el movimiento adecuado de varias de sus articulaciones. La Figura 7 es un ejemplo de dichos problemas.

Figura 7. Problemas en la estructura.

2.1.2. Motores y frenos

Se encontraron todos los motores y frenos en condiciones operables, aunque los correspondientes a las articulaciones de la muñeca no estaban en sus ubicaciones originales. Como se observa en la Figura 8, se encontraban desmontados de la estructura del robot.

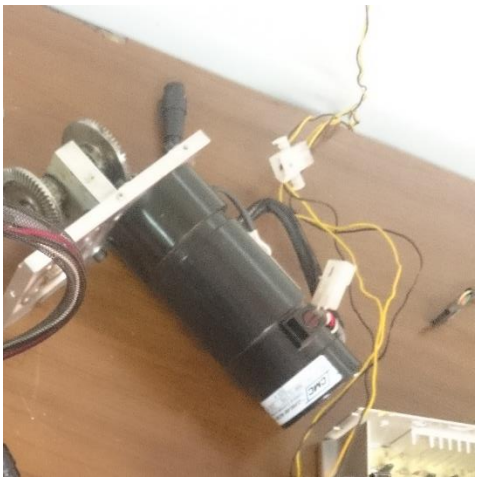


Figura 8. Motor del manipulador CRS A255

2.1.3. Sensores

El manipulador requiere de encoders incrementales para medir el movimiento rotacional de cada articulación y son de vital importancia para el control de los movimientos, pero estos no se encontraban en las articulaciones del hombro, codo, y muñeca roll. Esto se evidencia en la Figura 9.

Figura 9. Encoders faltantes

En la Tabla 1 se puede ver un resumen del estado de hardware del manipulador robótico CRS A255.

Tabla 1.
Resumen de condiciones de hardware iniciales

	MOTOR		ENCODER		FRENOS	
ARTICULACIÓN	**Montado**	**Funcional**	**Montado**	**Funcional**	**Montado**	**Funcional**
Cintura	si	si	si	si	no	no
Hombro	si	si	no	no	si	si
Codo	si	si	no	no	si	si
Pitch muñeca	no	si	si	si	si	si
Roll muñeca	no	si	no	no	si	si

2.2. Interfaz de conversión

La interfaz de conversión es el circuito electrónico que permite la transferencia de datos entre el sistema a ser controlado y el controlador que indica las acciones a realizar, en este caso el manipulador CRS A255 y un computador, respectivamente.

Inicialmente se contó con el circuito electrónico para transmisión de datos y potencia, del proyecto de titulación desarrollado anteriormente con uno de estos manipuladores, llamado: "*Sistema Replicador De Movimiento Articular De Extremidad Superior Derecha En Brazo Robótico Industrial Por Estudio Electromiográfico Y Uso De Kinect*". Desarrollada por Sandoval Daniela y Trujillo Andrea en el 2016. (Sandoval Daniela, 2016).

Por desgracia dichas tarjetas no encuentran en el mejor estado, no tienen un diseño profesional y se encuentran destruidas casi por completo, por lo cual no se cuenta con una interfaz de conversión funcional para ninguno de los dos manipuladores robóticos; esto se puede apreciar en la Figura 10 y Figura 11.

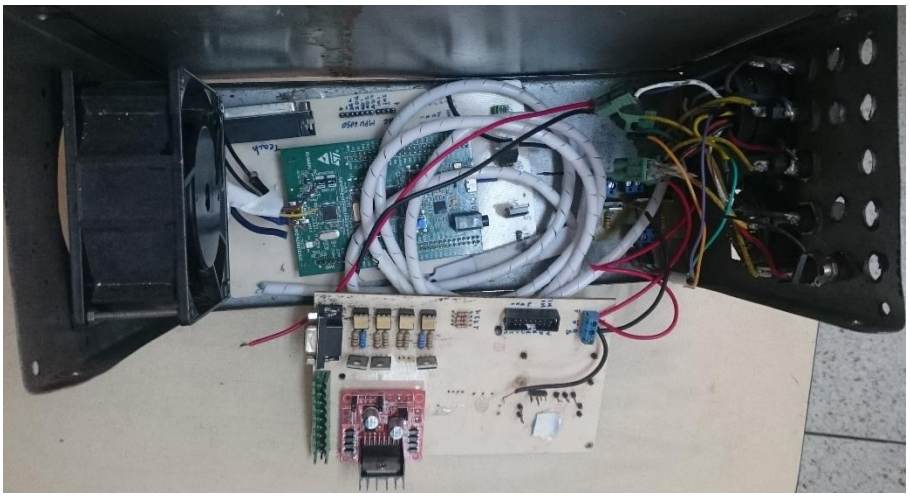


Figura 10. Interfaz de conversión inicial

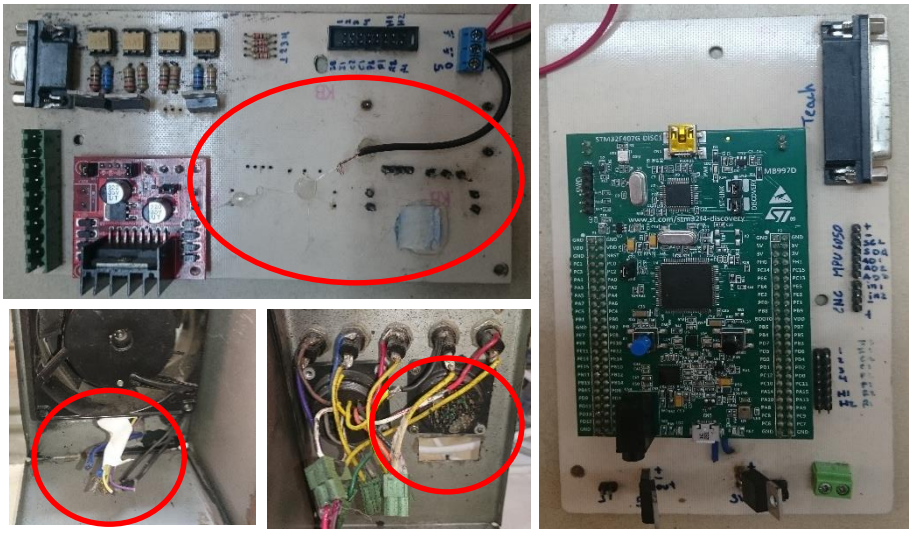


Figura 11. Interfaz de conversión inicial inutilizable

2.3. Sistema de control

Debido a que la interfaz de comunicación es inutilizable, no se pudo comprobar el funcionamiento del sistema de control implementado anteriormente. Además que el mismo no contaba con un entorno de simulación específicamente desarrollado para el monitoreo del manipulador robótico CRSA255. Sino únicamente para la visualización de las señales electromiografías. (Sandoval Daniela, 2016)

2.4. Conexiones eléctricas

Cada manipulador cuenta en su base con varios conectores como se muestra en la Figura 12. Para comprobar su funcionamiento, se realizó el chequeo de continuidad con un multímetro, comprobando así que todo el cableado interno esta en óptimas condiciones. En la Figura 13 y Figura 14 se puede ver la descripción de cada pin.

Figura 12. Conectores traseros del Robot CRS A255

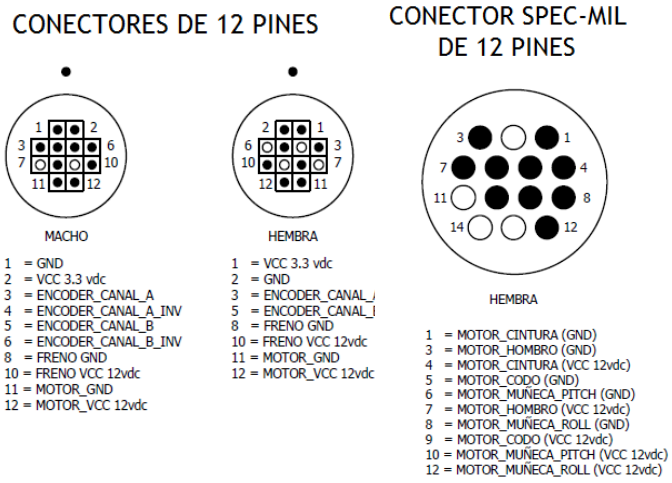
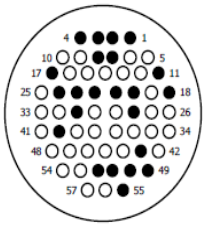


Figura 13. Descripción de pines de conectores del Robot CRS A255

CONECTOR SPEC-MIL DE 55 PINES



HEMBRA

1 = ENCODER_CANAL_A (CINTURA)
2 = ENCODER_CANAL_B (CINTURA)
3 = ENCODER_CANAL_A (HOMBRO)
4 = ENCODER_CANAL_B (HOMBRO)
7 = ENCODER_CANAL_Z (CINTURA)
8 = FRENO GND (HOMBRO)
11 = FRENO VCC 12vdc (HOMBRO)
17 = VCC 3.3 vdc (CINTURA)
18 = FRENO VCC 12vdc (CODO)
20 = ENCODER_CANAL_A (CODO)

21 = ENCODER_CANAL_B (CODO)
22 = ENCODER_CANAL_A (MUÑECA_PITCH)
23 = ENCODER_CANAL_B (MUÑECA_PITCH)
24 = GND (CINTURA)
28 = FRENO GND (CODO)
31 = FRENO GND (MUÑECA_PITCH)
40 = GND (CODO)(MUÑECA_PITCH)
43 = ENCODER_CANAL_A (MUÑECA_ROLL)
49 = ENCODER_CANAL_B (MUÑECA_ROLL)
50 = FRENO GND (MUÑECA_ROLL)

51 = VCC 3.3 vdc (HOMBRO) (MUÑECA_PITCH)
52 = VCC 3.3 vdc (CODO) (MUÑECA_ROLL)
55 = GND (HOMBRO) (MUÑECA_PITCH)

Figura 14. Descripción de pines de conectores del Robot CRS A255.1

CAPÍTULO III

OPTIMIZACION DEL HARDWARE

3.1. Optimización en robots

La optimización de hardware de los manipuladores CRS A255 es el paso principal para poder controlar la posición angular de cada articulación, ya que si esta etapa no funciona de manera adecuada el implementar un controlador será imposible. En este capítulo se indican todas las mejoras de hardware que se plantearon como una opción, cuáles de ellas se implementaron satisfactoriamente en los manipuladores CRS A255, y las razones por la cuales algunas de ellas no se pudieron llevar a cabo. Todo esto con el fin de no tener problemas al momento de realizar el movimiento y lectura de posiciones de cada una de las articulaciones en la futura etapa de control.

3.1.1. Opciones de mejora

De acuerdo al análisis inicial de los manipuladores robóticos CRS A225 tratado en el capítulo anterior se llegó a la conclusión de que se requieren hacer las siguientes mejoras para su correcto funcionamiento:

- Colocar los motores faltantes en sus ubicaciones originales y verificar que trabajen de una manera adecuada con el peso de cada una de las articulaciones y en el caso de ser necesario realizar el bobinado del rotor.
- Activar frenos independientemente para cada una de las articulaciones, así se logrará un mejor control al momento de mover cada articulación a la posición deseada y detenerla, ya que actualmente solo se activan de dos en dos (hombro y roll de muñeca, codo y pitch de la muñeca). Otra de las razones es debido a que con la configuración original del manipulador los frenos únicamente sirven como sistema de protección; es decir, en caso de una desconexión de energía

o paro de emergencia el manipulador activa sus frenos para evitar golpes a los operarios, materias de producción o a sus mismas articulaciones, al caer de manera abrupta.

- Acoplar freno para la articulación de la cintura, con el fin de detener el giro cuando se llegue a la posición deseada evitando rebotes y oscilaciones.
- Diseñar nuevos encoders en las articulaciones faltantes que se asemejen en su funcionamiento y tamaño a los originales, esto para tener la misma precisión al momento de realizar mediciones y además así evitar el cambio de la carcasa original del manipulador.

3.1.2. Mejoras realizadas

3.1.2.1. Montaje y prueba de motores

Se realizó exitosamente el montaje de los motores faltantes que correspondían a las articulaciones del roll y pitch de la muñeca, y su funcionamiento fue exitoso al mover las articulaciones correspondientes. Adicional a ello, se realizó el reemplazo de carbones en cada uno de los motores del manipulador para evitar posibles errores en su funcionamiento. Mejorando el contacto y conducción de energía a través de ellos.

En la Tabla 2 se evidencian las características eléctricas de los motores de todas las articulaciones una vez realizadas las correcciones necesarias. Las mediciones de intensidad de corriente eléctrica en cada motor se realizaron, permitiendo su libre movimiento sin obstrucciones y deteniendo su movimiento en un punto con una fuerza externa.

Tabla 2.
Características eléctricas de los motores

ARTICULACIÓN	Tensión	I max sin obstrucción	I max con obstrucción
CINTURA	12v	1.2 A	3.8 A
HOMBRO	12v	2.1 A	3.6 A
CODO	12v	1.3 A	3.6 A

CONTINÚA

PITCH MUÑECA	12v	0.7 A	3.4 A
ROLL MUÑECA	12v	0.75 A	3.7 A

3.1.2.2. Activación de frenos independientes

Esta etapa de repotenciación se realizó para ambos brazos robóticos y para ello se requirió modificar el cableado interno de los manipuladores, ya que en su configuración original no se permite la desactivación de frenos de manera independiente, porque su conexión a la señal de referencia es la misma que la del motor de su articulación correspondiente, Figura 15, lo cual imposibilita activar los motores con una señal PWM para el controlador actual, debido a que también afectaría a los frenos impidiendo su desactivación durante el movimiento.

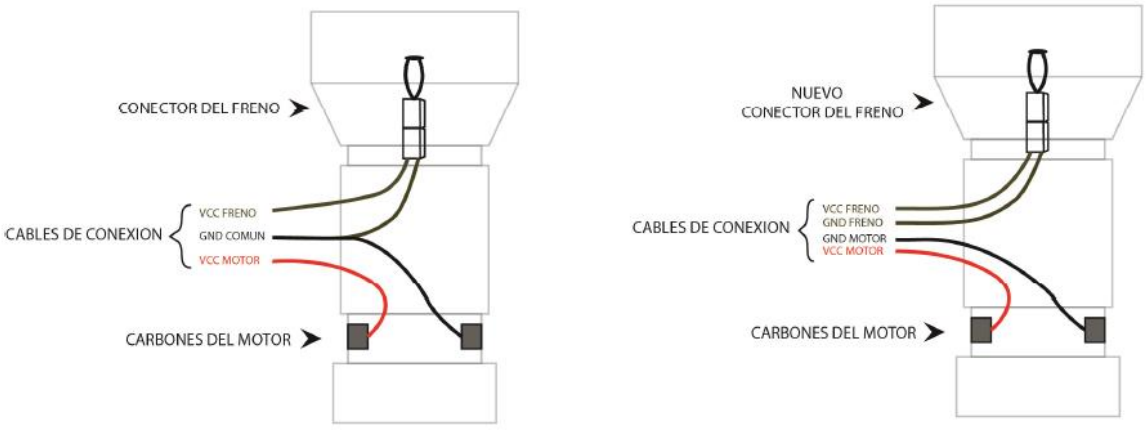


Figura 15. Conexión Original (IZQ) Conexión Actual (DER)
Fuente: (Maldonado Daniel, 2013).

Una vez realizada la conexión para cada articulación, la desactivación de los frenos se puede realizar de forma independiente, logrando así la repotenciación planteada.

3.1.2.3. Acoplar freno a la articulación de la cintura

Es una de las repotenciaciones planeadas que no se pudo llevar a cabo debido a que el motor de esta articulación se encuentra debajo del robot y su espacio de trabajo es muy limitado, por lo que

no se pudo encontrar un freno de tipo solenoide que cumpliera con las condiciones de tamaño para montarlo en ese lugar. Se realizan recomendaciones para trabajos futuros sobre este punto.

3.1.2.4. Diseño de nuevo encoder

Un encoder incremental entrega un tren de pulsos con valores TTL en función del giro de su eje, esto se obtiene haciendo cruzar un haz de luz por medio de un disco ranurado y cada vez que la luz se corte se enviará un pulso. Las partes de un encoder se pueden ver en la Figura 16.

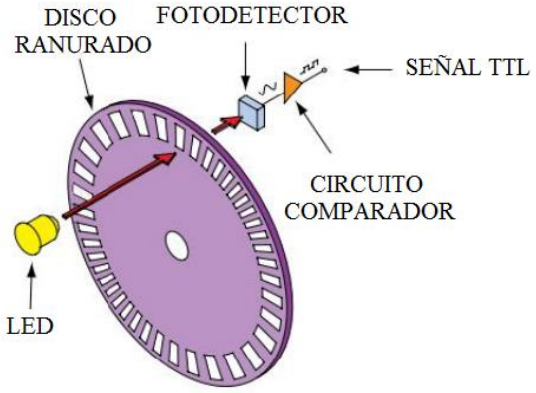


Figura 16. Partes de un encoder
Fuente: (Herrera, 2011)

Se intentó diseñar un encoder de tipo incremental para cada una de las articulaciones donde hacía falta. Pero debido a la necesidad, el obtener 1000 PPR en un disco ranurado de forma no industrial fue imposible, lo máximo que se pudo lograr fue 500 líneas sobre un acetato transparente, que puede utilizarse como un disco de encoder. Lo cual no cumplía de igual manera con estándares industriales. Además un sensor capaz de leer dichas líneas sobre el acetato, no se puede encontrar en nuestro país, y fuera del mismo existe a costos elevados con un tiempo de llegada mayor a dos meses.

Debido a esto el diseño inicial del disco y la base de acople para el motor que se puede ver en la Figura 17, no pudo llegar a ser implementado.

Figura 17. Encoder diseñado: Base de acople (IZQ), disco ranurado (DER).

Dado que no se logró realizar el diseño completo de un encoder con las características requeridas, la solución fue comprar encoders fuera del país a un precio más económico que diseñarlo por completo. Los encoders adquiridos son iguales al de la Figura 18 y entre sus características se encuentran las siguientes:

- PPR: 600.
- Vin: 3.3VDC a 5 VDC.
- Tamaño: 38mm x 35.5mm.
- Longitud del cable: 1.5m.
- Dos fases de salida.

Figura 18. Encoder adquirido

Para los nuevos encoders se diseñó una nueva base de acople entre estos y los ejes de los motores. Para ello se utilizó el software AutoCAD, obteniéndose el resultado que se aprecia en la Figura 19.

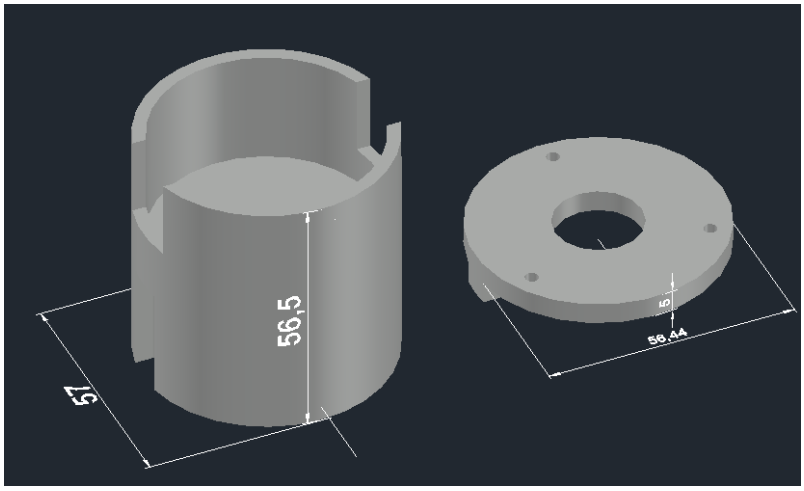


Figura 19. Base diseñada para encoder adquiridos

Se los creo haciendo uso de una impresora 3D, obteniendo la pieza que se ve en la Figura 20.

Figura 20. Base impresa para encoder adquiridos

En la Figura 21, se puede apreciar el resultado de acoplar los nuevos encoders al manipulador robótico.

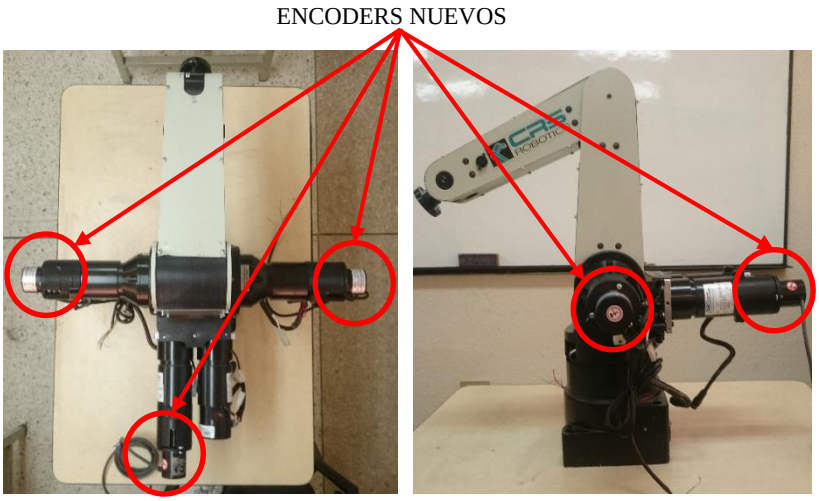


Figura 21. Montaje de encoder nuevos, vista superior (IZQ), vista lateral (DER).

Como se observa en las imágenes anteriores al colocar los nuevos encoders, las dimensiones del manipulador CRS A255 aumentan considerablemente razón por la cual no fue factible colocar su carcasa original nuevamente. La solución para mantener una buena estética del manipulador y cumplir así con la repotenciación del hardware, fue crear una nueva carcasa y colocarla sobre el robot. Este diseño se llevó a cabo en AutoCAD como se puede ver en la Figura 22, y su implementación y montaje como se observa en la Figura 23.

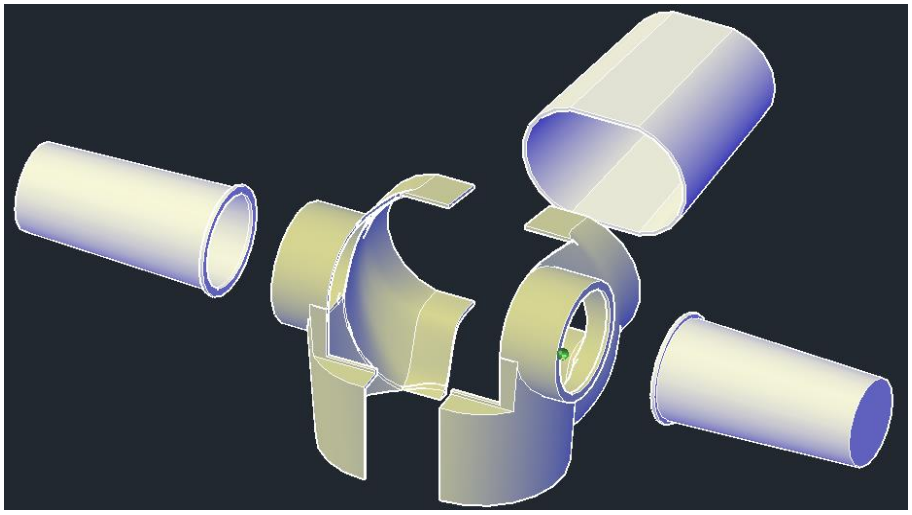


Figura 22. Diseño 3D de la nueva carcasa

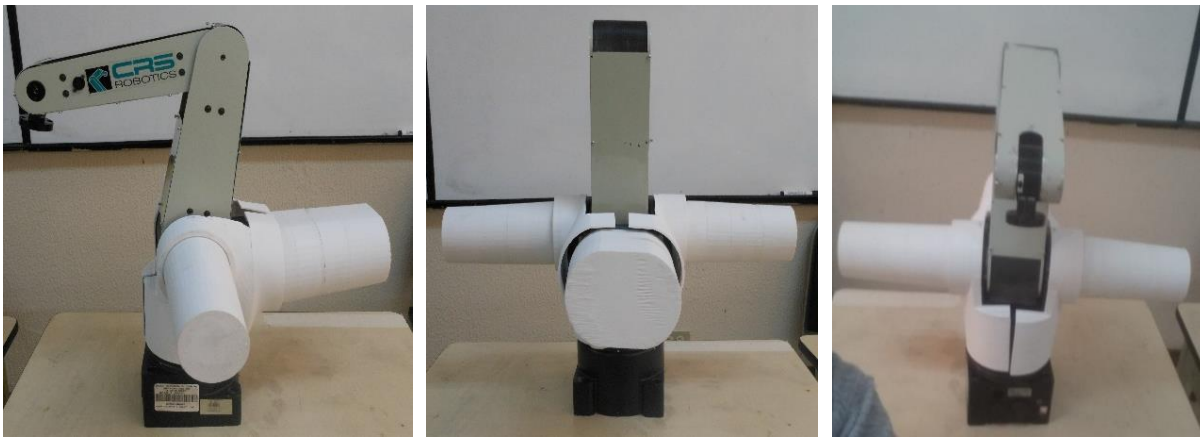


Figura 23. Nueva carcasa sobre el manipulador CRS A255

3.2. Interfaz de conversión

La creación de este circuito es un factor muy importante en la etapa de repotenciación del sistema de control de posición angular del manipulador CRS A255 ya que será el encargado de transmitir los datos generados desde los encoders hacia el medio de control que en este caso será un computador, quien de igual forma deberá indicar a los motores de las articulaciones el movimiento que estos deben realizar.

Como se detalló en el capítulo anterior, las tarjetas que servían para la comunicación y envió de energía a la parte de potencia se encuentran inservibles, razón por la cual se decidió diseñar y crear dos circuitos de microcontroladores, uno para cada manipulador robótico, para que así en el caso que se necesite utilizar un solo robot se lo pueda hacer de forma independiente.

3.2.1. Requerimientos

Para empezar con el diseño del circuito electrónico se debe tomar en cuenta las características del sistema al cual se quiere controlar, ya que con esa información se puede elegir el microcontrolador que mejor se adapte a los requerimientos planteados. Las características que se desea para el sistema de control de posición angular de los manipuladores son las siguientes:

- Se debe contar con un microcontrolador capaz de manejar más de diez interrupciones, ya que se requiere leer dos fases de cada encoder en todo momento para saber en qué posición se encuentra cada articulación.
- El microprocesador debe ser capaz de intercambiar información con el computador que será encargado de realizar el control.
- Al contar con motores de corriente continua, el control se realizará por medio de la modulación de ancho de pulso (PWM), por su facilidad al momento de implementarse, pero teniendo en cuenta que dichos motores deben girar en ambos sentidos; por lo cual se debe tener previsto el uso de un puente H. Para ello se requiere por lo menos cuatro pines del microcontrolador por cada motor.
- Se cuenta con cuatro frenos, uno por cada articulación, lo cual implica cuatro pines adicionales del microcontrolador.
- Como se pensó en un inicio, este sistema lo van a utilizar en el área académica y de investigación, por lo tanto debe contemplarse que en cualquier momento se sobrepasarán los límites de corrientes y tensión recomendadas, dando como resultado quema de elementos o módulos dentro de las tarjetas, por lo cual las mismas deben ser modulares con partes fácilmente reemplazables además de contar con un sistema de protección como fusibles.

Un resumen de las características que el microcontrolador elegido debe cumplir se presenta en la Tabla 3.

Tabla 3.
Resumen de requerimientos para el microcontrolador

	CANTIDAD	**USO**
PINES DE ENTRADA	10	Encoders/interrupciones

CONTINÚA

PINES DE SALIDA	20	Motores/PWM
	4	Frenos
CONEXIÓN A PC	1	Comunicación con el controlador

3.2.2. Selección del microcontrolador

Una vez analizada las características del sistema, se optó por utilizar dos tarjetas Teensy 3.6 una para cada manipulador, debido a que todos sus pines tienen la opción de ser utilizados como interrupciones externas lo cual es conveniente para la lectura de los encoders, y además la cantidad de pines con que estos microcontroladores cuentan permiten abarcar todas las señales necesarias para realizar el control. En la Tabla 4 se presentan sus características más relevantes.

Tabla 4.
Características más relevantes de la Teensy 3.6

Microcontrolador	**MK66FX1M0VMD18**
Núcleo	ARM Cortex 180 MHz
Memoria	1 M Flash, 256 K RAM, 4 K EEPROM
Velocidad puerto USB	(12M bit/seg)
Pines PWM	22
Puertos I2C	4
Pines de E/S	42/ todos configurables con interrupciones
Entradas analógicas	25 , resolución 13 bits
Salidas analógicas	2 , resolución 12 bits
Puerto seriales	6
Puertos SPI	3

Fuente: (Stoffregen Paul, 2015).

Cabe recalcar que al realizar una comparación con la tarjeta utilizada por Rivera, la STM32f4 Discovery, se definen características de velocidad de procesamiento y I/Os similares; por lo que de acuerdo a los resultados presentados por este trabajo previo se cumplen los requisitos necesarios para el correcto manejo de la información. (Rivera, 2015)

3.2.3. Diseño de la etapa de potencia

Se hace referencia a todas las señales eléctricas que no se encuentran entre el rango de 0 a 5V, con la cual opera la tarjeta microcontroladora, razón por la que se debe hacer una etapa de elevación de tensiones y corrientes con el fin de controlar los motores y frenos de los manipuladores CRS A255.

Hay que tener en cuenta que se utiliza un conversor de voltaje AC/DC de 110VAC/12VDC a la entrada del circuito, lo cual permite conectarlo directamente al tomacorriente del laboratorio.

3.2.3.1. Motores

Como se indicó anteriormente, los manipuladores robóticos cuentan con motores de corriente continua, por lo cual el control de su velocidad se realizará por medio de PWM, pero además debe ser capaz de controlar el sentido de giro de los mismos, para lo cual se requiere una configuración de puente H. Debido a estas características se optó por utilizar el módulo L298N que es un puente H con la capacidad de controlar la velocidad de los motores por medio de PWM, con una corriente máxima de operación de 3 amperios, lo cual es más que suficiente para los valores de corriente que requieren los motores del manipulador para su funcionamiento, según lo visto en la Tabla 2. Su conexión y tabla de verdad de funcionamiento se observan en la Figura 24 y la Tabla 5 respectivamente.

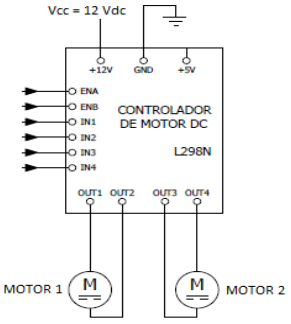


Figura 24. Conexión del módulo L298N

Tabla 5.
Tabla de verdad del módulo L298N

ENA	IN1	IN2	MOTOR1	ENB	IN3	IN4	MOTOR2
0	1	0	No gira	0	1	0	No gira
0	1	1	No gira	0	1	1	No gira
0	0	0	No gira	0	0	0	No gira
0	0	1	No gira	0	0	1	No gira
1	1	0	Horario	1	1	0	Horario
1	1	1	No gira	1	1	1	No gira
1	0	0	No gira	1	0	0	No gira
1	0	1	Anti horario	1	0	1	Anti horario

3.2.3.2. Frenos

Los frenos de cada articulación de los manipuladores robóticos se activan con una tensión de 12VDC entre sus polos, por lo cual se diseñó un circuito de elevación de voltaje para que puedan ser controlados desde la salida de la tarjeta Teensy 3.6, la cual opera a 3.3V en sus pines. Este circuito se puede observar en la Figura 25.

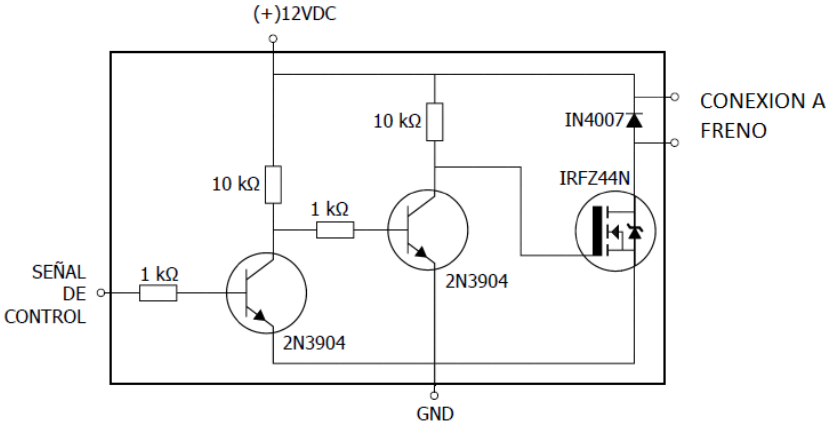


Figura 25. Circuito de elevación de potencia para los frenos

El cálculo de la resistencia de base necesaria para saturar los transistores 2N3904 se muestra a continuación:

$$V_b = I_b . R_b$$

$$R_b = \frac{V_b}{\frac{I_{CE}}{h_{FE}}}$$

$$R_b = \frac{3.3 - 0{,}7}{\frac{0.12}{30}} = 650\ \Omega = 1k$$

Como se conoce los transistores de tipo MOSFET se activan con tensión y en el caso particular del IRFZ44N, según su hoja de datos para alcanzar su saturación se requieren 10VDC como mínimo. En la configuración de la Figura 25, al saturarse el segundo transistor 2N3904 se tiene 12VDC en el gate del MOSFET entrando este en saturación y activando el freno.

3.2.4. Diseño de la etapa de control

Para alimentar el circuito encargado de la parte del control, se requiere una tensión de alimentación de 3,3Vdc, con la cual funcionan las entradas de control de los módulos L398N y los encoders del manipulador robótico. Se utilizó el circuito integrado LM317 que es un regulador de voltaje variable que puede abarcar tensiones desde los 1.5VDC hasta 24VDC, dependiendo de la configuración de resistencias entre sus pines.

El circuito a montar es mostrado en la Figura 26. De acuerdo a su hoja de datos la ecuación para calcular el valor de resistencias para un valor de tensión deseado a la salida es la siguiente:

$$V_{out} = 1.25(1 + \frac{R_2}{R_1})$$

$$\left(\frac{V_{out}}{1.25} - 1\right) R_1 \ = R_2$$

Asumiendo

$$V_{out} = 3.3$$

$$R_1 = 1000\ \Omega$$

$$R_2 = 1540 = 1500\Omega$$

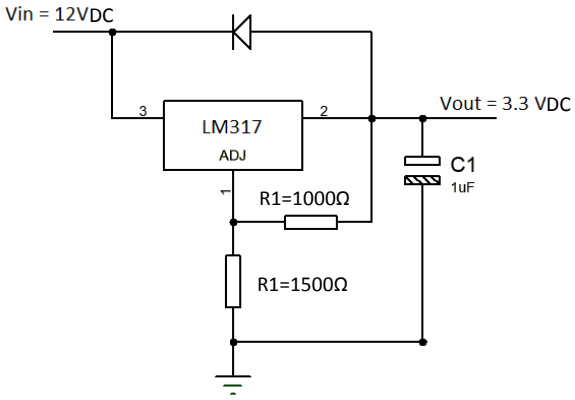


Figura 26. Circuito del regulador de voltaje LM317

3.2.5. Implementación de la interfaz de conversión

Una vez determinados todos los componentes que conformarán la interfaz de conversión, se realiza el diseño del circuito eléctrico en el programa Proteus. Para colocar la tarjeta Teensy 3.6 y los módulos L298N se tuvo que tomar las medidas de los mismos y representarlos con espadines dentro del diseño como se muestra en la Figura 27 y en la Figura 28.

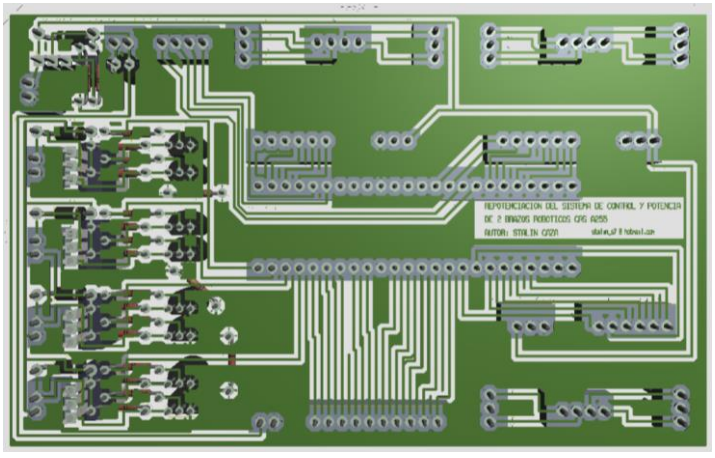


Figura 27. Diseño del circuito para la interfaz de conversión.

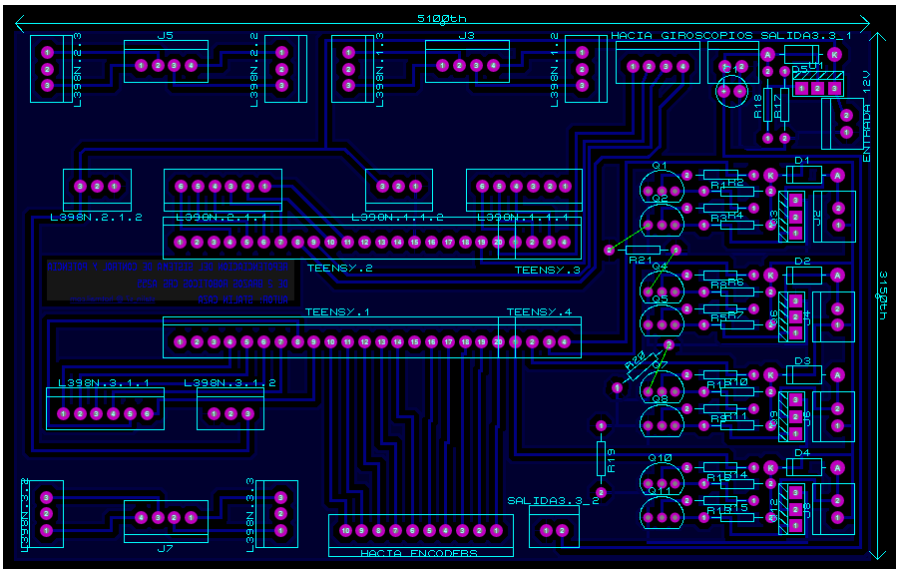


Figura 28. Diseño del circuito para la interfaz de conversión. PCB

La creación de la placa PCB se realizó utilizando métodos caseros como son el planchado y revelado en acido, obteniendo el resultado que se muestra en la Figura 29, cabe mencionar que las medidas de la placa son: 80mm x 129.5mm.

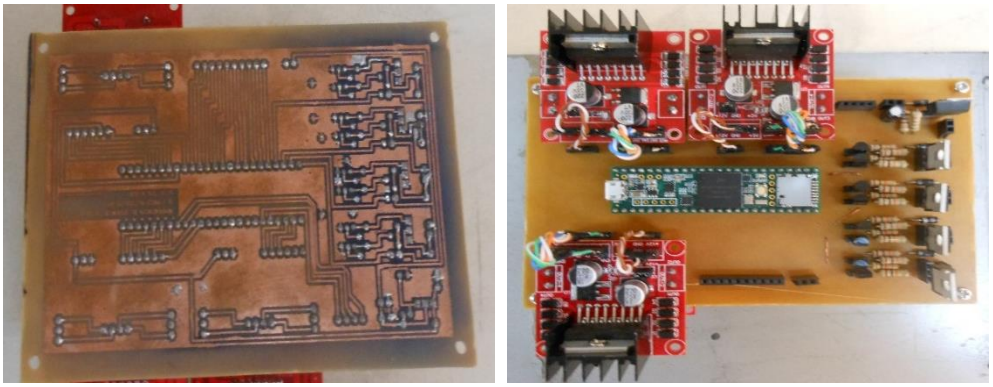


Figura 29. Interfaz de conversión.
Vista inferior (IZQ), Vista superior (DER)

3.2.6. Conexión y montaje de la interfaz de conversión

Para el correcto montaje de los dos circuitos de conversión, se modificó a la carcasa existente, realizando dos orificios en la parte delantera de la misma. Además al no conseguir los conectores

SPEC-MIL necesarios para la conexión de los cables originales de los manipuladores robóticos, se llevó a cabo el diseño de los mismos en AutoCAD, se los generó en una impresora 3D, y se los colocó en la carcasa de metal, tal como se ve en la Figura 30 y Figura 31.

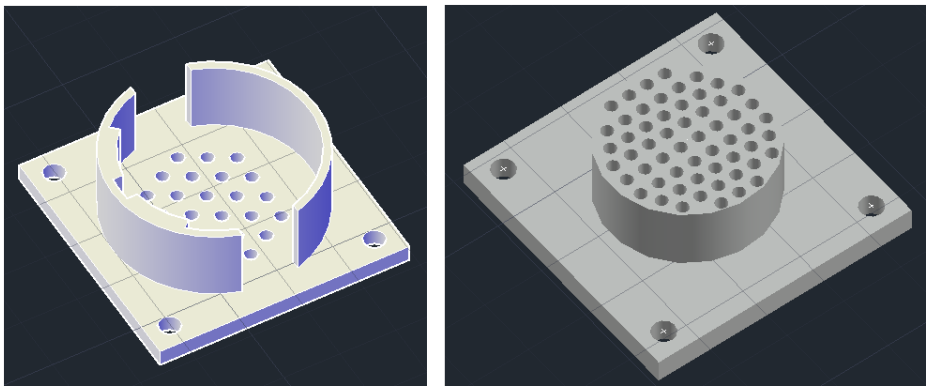


Figura 30. Diseño de conectores en AutoCAD
24 pines (IZQ), 57 pines (DER)

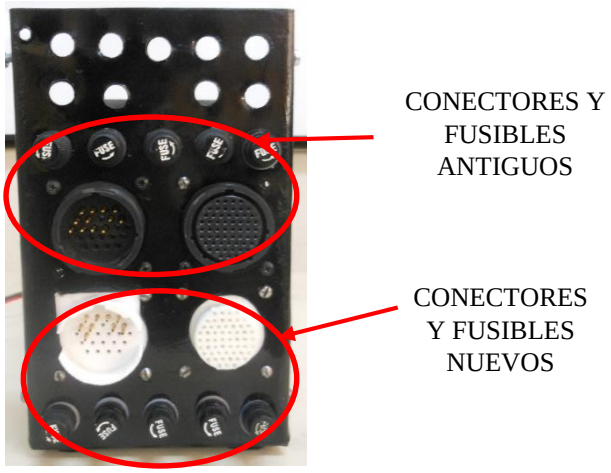


Figura 31. Instalación de conectores y
fusibles nuevos

El cableado interno se lo desarrolló en función a los planos eléctricos que se pueden ver en el Anexo 5, el cual fue realizado en el software EPLAN Electric con el fin de evitar conexiones erróneas en el momento del montaje del circuito; además de tener una documentación para futuras

aplicaciones, mejoras o remplazos en el circuito diseñado. En la Figura 32, Figura 33 y Figura 34, se puede observar el montaje final del circuito dentro de su carcasa.

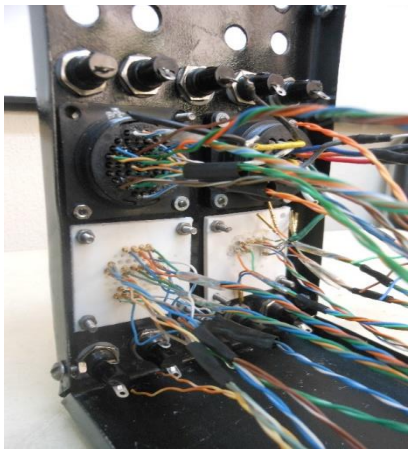


Figura 32. Conexión de cables a los pines de conectores SPEC-MIL.

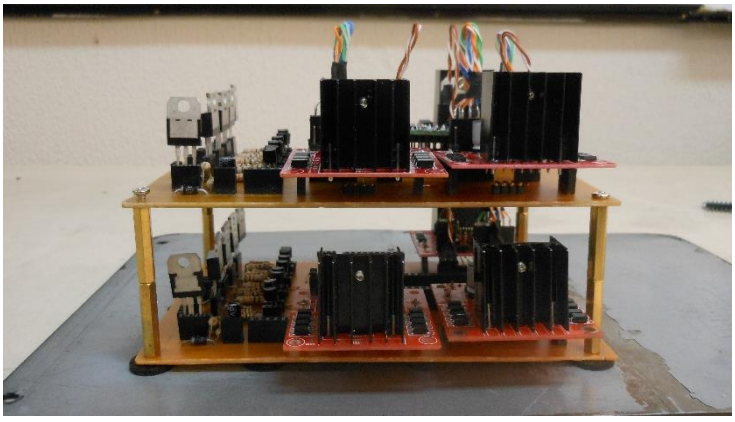


Figura 33. Montaje de ambos circuitos sobre carcasa

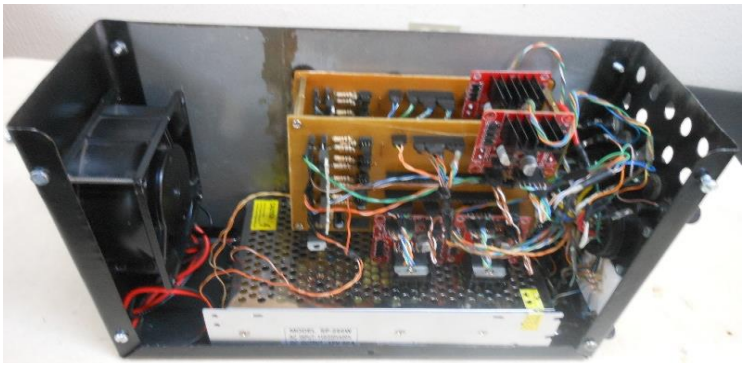


Figura 34. Implementación completa dentro de la carcasa

CAPÍTULO IV

OPTIMIZACIÓN DEL SISTEMA DE CONTROL

Como se detalló en el análisis inicial de los manipuladores CRS A 255, existe una completa carencia de un sistema de control para ambos robots, debido al mal uso de los controladores originales o al descuido de los controladores diseñados en proyectos anteriores. Razón por la cual se ha decidido dotar de un sistema automático de control de posición angular de motores, para cada articulación de los manipuladores, sin tomar en cuenta el modelo dinámico del mismo, el cual recibirá su consigna de forma manual desde el entorno de simulación, permitiendo la monitorización del funcionamiento de los mismos durante su trabajo.

Como se planteó en los objetivos de este proyecto, el sistema de control debe estar basado en una estructura ROS, la cual se encarga de llevar a cabo la comunicación entre los componentes (manipuladores, interfaz de conversión, controladores difusos, entorno de simulación), corriendo sobre un computador con Sistema operativo Ubuntu. Lo mencionado anteriormente se puede apreciar de mejor manera en la Figura 35.

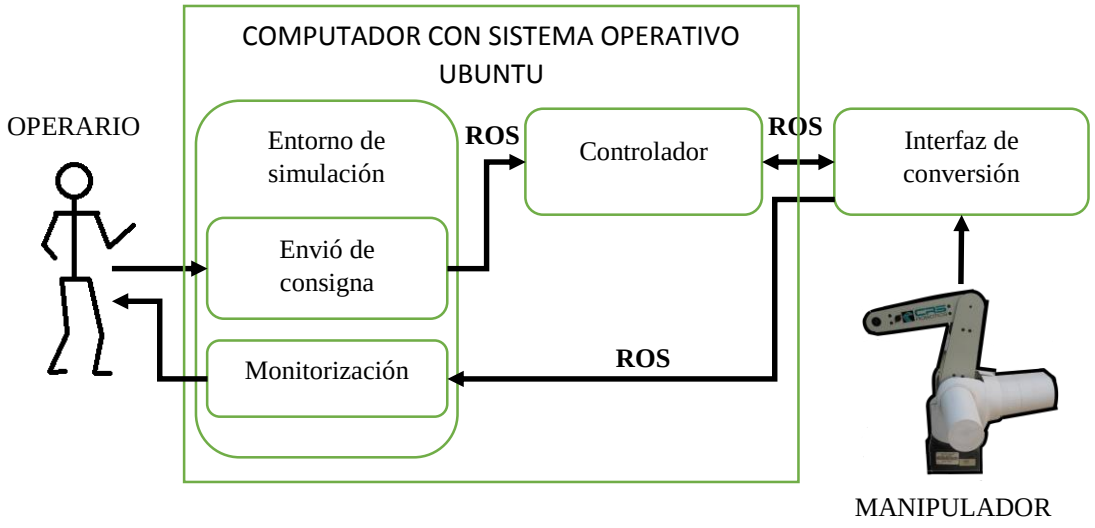


Figura 35. Esquema de operación del control de posición de motores

En este capítulo se detallará todo el software necesario, aplicación y algoritmos, para llevar a cabo el control de posición angular de los motores del manipulador robótico CRS A255 cumpliendo las características inicialmente planteadas.

4.1. Características e instalación de software

4.1.1. ROS.

El Sistema Operativo Robótico, (ROS por sus siglas en ingles), cuenta con un diseño modular y distribuido, lo que quiere decir que el usuario puede elegir los paquetes ROS que más se adapten a sus proyectos y hacer uso de ellos, debido a que es software libre, permitiendo su uso a nivel de investigación y comercial. Un punto importante en el uso de ROS es su independencia del lenguaje de programación, ya que el mismo se puede hacer en C++, Python, Lisp, Octave Java, entre otros. Y al final todos se lograran comunicar de forma correcta. (Ferguson, 2015).

4.1.1.1. Infraestructura de comunicación.

ROS brinda una interfaz de transmisión de mensajes entre procesos conocida como middleware, proporcionando los servicios:

- Publicar y suscribirse a mensajes
- Grabar y reproducir mensajes
- Llamadas a procedimientos remotos

4.1.1.2. Partes fundamentales de la comunicación ROS

Para llevar a cabo la transmisión de información dentro de ROS se requieren del maestro, nodos, mensajes, tópicos y servicios:

- El maestro es el encargado de registrar todos los nodos existentes y de esa manera ayudar a que los mismos puedan encontrase entres sí; es decir, es el responsable de que toda la comunicación ROS funcione.
- Un nodo es un proceso ejecutable en ROS, el cual funcionará de manera independiente de otros, es el encargado de publicar o subscribirse a mensajes y así comunicarse con otros nodos pertenecientes a un sistema.
- Los mensajes son una estructura de datos que se caracterizan de acuerdo a su tipo por ejemplo: entero, booleano, coma flotante, cadena de caracteres, etc. Los nodos son los encargados de enviar estos mensajes y lo hacen mediante tópicos o temas.
- Un tópico es un lugar donde siempre llegan mensajes de un mismo tipo, por lo tanto si un nodo está interesado en los valores que están llegando a cierto tópico, únicamente se suscribe a él y así podrá usar dicha información.
- Por su parte los servicios están a cargo de realizar un tarea por ello siempre necesitan enviar una solicitud y se obtendrá una respuesta, todo lo contrario a los mensajes, en los cuales únicamente se lee los datos que se están transmitiendo.

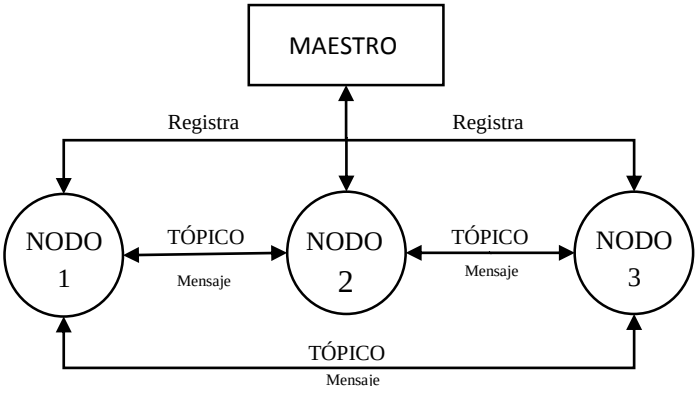


Figura 36. Elementos de ROS.

4.1.1.3. Instalación ROS

Antes de iniciar con la instalación de ROS es necesario conocer las características del computador a utilizar, y de acuerdo a estas, seleccionar la versión que más se adapte a sus capacidades. Para el caso de este proyecto se cuenta con un computador con las siguientes características:

- Sistema Operativo: Ubuntu 16.04 (XENIAL)
- Procesador: Intel Core i7
- Arquitectura: 64 bits
- Capacidad de disco duro: 100 Gb
- Memoria RAM: 4Gb

De acuerdo a estas características la mejor opción en el momento de realizar este proyecto es la versión ROS Kinetic LTS, lanzada el 23 de Mayo de 2016. Cabe destacar que no es la versión más actual pero cuenta con soporte a largo plazo, por ello las siglas LTS (Long Term Support), lo cual garantiza que durante los próximos tres años, es decir hasta el 2021, se seguirán realizando actualizaciones y correcciones para su correcto funcionamiento. Una vez decidida la versión se procede a instalar siguiendo los pasos indicados a continuación:

- Configurar las fuentes de software ROS

  $ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

- Configurar las llaves de acceso a los servidores

  $ sudo apt-key adv --keyserver hkp: //ha.pool.sks-keyservers.net: 80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

- Actualizar el índice de paquetes Debian

  $ sudo apt-get update

- Instalamos la versión Kinetic

  $ sudo apt-get install ros-kinetic-desktop-full

- Inicializar rosdep para poder instalar dependencias y componentes de ROS

  $ sudo rosdep init

  $ rosdep update

- Añadir las variables de entorno ROS a la sesión *.bashrc* esto ayuda a que cada vez que se ejecute una aplicación bajo esta estructura, no se requiera su configuración de forma manual. Para realizar esto se necesita abrir el archivo *. /bashrc* y copiar la siguientes líneas en la parte inferior del mismo, donde ¨*PRUEBAS_ROSINTERFACE*¨ es el espacio de trabajo, como se indica en la Figura 37.

  $ gedit ~ / .bashrc

  source /opt/ros/kinetic/setup.bash

  source ~/PRUEBAS_ROSINTERFACE/devel/setup.bash

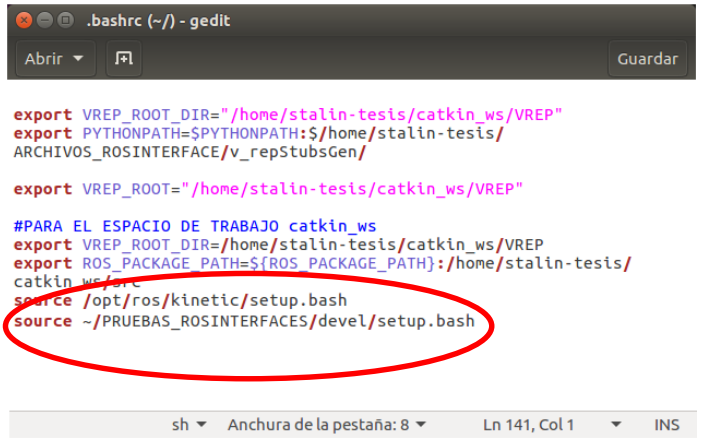


Figura 37. Modificación del archivo .bashrc.

- Descargar dependencias para en el futuro poder crear paquetes ROS

  $ sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential

4.1.1.4. Creación del espacio de trabajo para ROS

El espacio de trabajo será el directorio donde se crearán todos los paquetes y programas ROS necesarios para llevar a cabo el control de los manipuladores robóticos por lo tanto es un paso esencial en el proceso de desarrollo de este proyecto. Para crear el espacio de trabajo se ejecutan los siguientes comandos:

$ mkdir -p ~/PRUEBAS_ROSINTERFACE/src

$ cd ~/ PRUEBAS_ROSINTERFACE /

$ catkin_make

- El nombre PRUEBAS_ROSINTERFACE es el nombre con el cual se creara el directorio.
- Para verificar que el espacio de trabajo se haya creado correctamente se debe ejecutar:

  $ echo $ROS_PACKAGE_PATH

- La respuesta en el terminal debe ser la dirección del directorio donde creo el espacio de trabajo, algo similar a lo siguiente: /home/stalin_tesis/PRUEBAS_ROSINTERFACE/src: /opt/ros/ kinetic/share

4.1.2. Interfaz de conversión con ROS

Una vez que se ha instalado ROS en el computador, es necesario hacer que este intercambie información con la interfaz de conversión, desarrollada en la etapa de repotenciación de hardware. Esto es posible gracias al complemento del IDE de Arduino, Rosserial, el cual “Es un protocolo para envolver los mensajes serializados ROS estándar y multiplexar múltiples temas y servicios a

través de un dispositivo de caracteres, como un puerto en serie o un socket de red". (Ferguson, 2015).

Gracias a esto la interfaz de conversión se comportará como un nodo ROS, publicando y suscribiéndose a mensajes directamente. Para ello se requiere la instalación del IDE de Arduino, donde se realizará la programación, y su complemento Teensyduino que permite el reconocimiento de la Tarjeta Teensy 3.6 dentro del IDE.

4.1.2.1. Instalación IDE Arduino

Una de las ventajas de haber seleccionado la tarjeta microcontroladora Teensy 3,6, es la facilidad en cuanto a su programación, ya que la misma se puede realizar desde el IDE de Arduino.

Para realizar la descargar e instalación de la versión más actual, se ejecuta las siguientes líneas de comandos en el terminal de Ubuntu.

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

4.1.2.2. Instalación Teensyduino

Descargar Teensyduino desde la página oficial de Teensy e instalar en el mismo directorio del IDE de Arduino con la ejecución en el terminal del comando mostrado a continuación:

```
$ chmod +x TeensyduinoInstall.linux64
```

Se requiere modificar las reglas udev de Ubuntu, permitiendo así el uso de dispositivos Teensy a usuarios que no tengan permisos root.

```
$sudo cp 49-teensy.rules /etc/udev/rules.d/
```

Con estos pasos se puede utilizar el IDE Arduino para programar la tarjeta Teensy 3.6, con los mismos comandos que se utilizan para cualquier modelo de Arduino. Siempre es necesario

seleccionar la tarjeta Teensy 3.6 en el menú de herramientas como se muestra en la Figura 38, para poder enlazarse y cargar librerías para su programación.

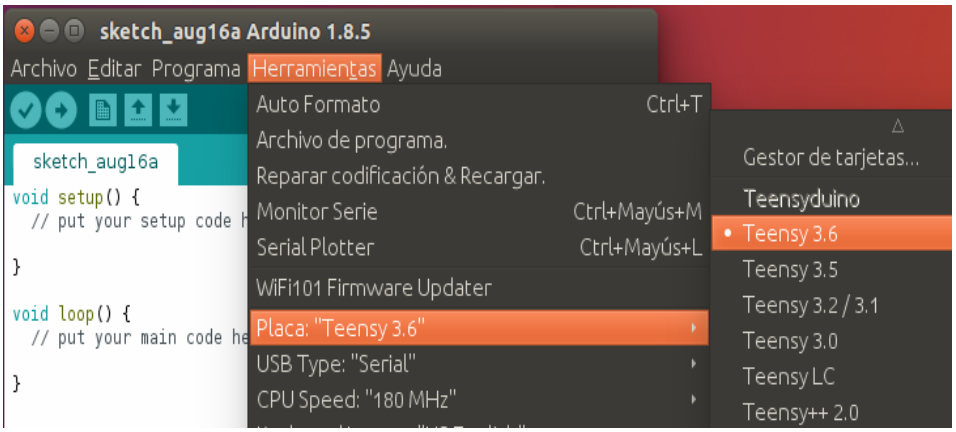


Figura 38. Selección de tarjeta Teensy 3.6 en IDE Arduino

4.1.2.3. Instalación Rosserial

La instalación del complemento Rosserial para el IDE de Arduino es un proceso simple el cual se logra llevando a cabo los pasos que se describen a continuación:

- Descargar Rosserial desde el terminal y crear la librería *ros_lib*. Para esto es necesario acceder a la carpeta */scr* dentro del espacio de trabajo *PRUEBAS_ROSINTERFACE* creado anteriormente, y ejecutar los siguientes comandos:

  $ git clone https://github.com/ros-drivers/rosserial.git

  $ catkin_make

- Copiar la librería *ros_lib* generada, en el directorio donde se instaló el IDE de Arduino. Se puede verificar la correcta instalación del complemento, si en los ejemplos se encuentra *ros_lib* como se muestra en la Figura 39, lo cual es indicativo de que la tarjeta Teensy 3.6 ya se puede comunicar con ROS.

Figura 39. Instalación de Rosserial

4.1.3. Controlador con ROS

Para realizar el control de posición angular de los motores de los manipuladores Robóticos CRS A255 se utilizará el procesamiento del computador en el cual se instaló ROS, por lo cual es necesario conocer cómo crear un ejecutable que sea capaz de llevar a cabo tareas secuenciales de un archivo programado en C++; y de acuerdo a ellas publique o se suscriba a un tema en específico ROS. Esto implica que el archivo fuente generado cuente con varios suscriptores y editores que permitan la comunicación tanto con la tarjeta de la interfaz de conversión, como también con el entorno de simulación.

4.1.3.1. Creación de un nodo ROS C++ en Ubuntu

Para la creación de un nodo ROS se requieren seguir los pasos descritos a continuación:

- Ingresar al espacio de trabajo y a la carpeta src,

  $ cd PRUEBAS_ROSINTERFACE

  $ cd src

- Dentro de este directorio crear un paquete con el nombre que se desee, el cual contendrá los archivos fuente

  $ catkin_create_pkg CONTROLADORES std_msgs rospy roscpp

  $ catkin_make

- Crear un nuevo directorio */scr* dentro del paquete

  $ mkdir –p scr

- Ingresar a la carpeta */scr* y generar el archivo fuente teniendo en cuenta te que tenga la extensión *.cpp* para que Ubuntu lo reconozca como un archivo de código fuente C++

  $ touch control_fuzzy_1.cpp

- En este archivo fuente se escribe todo el programa que se desee y se lo guarda.
- Abrir el archivo *CMakeLists.txt* que se encuentra dentro del directorio del paquete.

  $ gedit CMakeLists.txt

- Al final del archivo colocar las siguientes líneas de código, con el nombre del paquete y el nombre archivo que contiene el código fuente y guardar.

  add_executable(control_fuzzy_1 src/ control_fuzzy_1.cpp)

  target_link_libraries(control_fuzzy_1 ${catkin_LIBRARIES})

  add_dependencies(control_fuzzy_1CONTROLADORES_generate_ messages_cpp)

- Ir al directorio del espacio de trabajo y ejecutar

  $catkin_make

- Con eso se compila y se crea el ejecutable del archivo C++ que se haya escrito, en caso de existir algún error se mostrará en el terminal.
- Para la ejecución del mismo y ver su funcionamiento se utiliza el siguiente comando

$rosrun CONTROLADOR control_fuzzy_1

4.1.4. Entorno de simulación V-REP con ROS

V-REP es un simulador de robots, con una arquitectura de control por objetos, los cuales se pueden controlar mediante un script. Dicho script puede utilizar distintas interfaces de programación, permitiendo que gracias a ello sus controladores usen lenguajes como C++, Phyton, Java, Lua, entre otros.

Es un programa muy utilizado para desarrollo de algoritmos de control para robots industriales, permitiendo así la verificación y creación de prototipos, de una manera rápida. Además cuenta con modelos robots de todo tipo y de marcas reconocida a nivel mundial como KUKA, y ABB.

Se optó por usar este simulador de robots en su versión libre para estudiantes, debido a que gracias al complemento *RosInterface* , puede tratarse como un nodo ROS, al suscribirse y publicar mensajes de forma directa desde la simulación.

4.1.4.1. Instalación De V-REP

La instalación de V-REP en Ubuntu es muy sencilla, únicamente se requiere realizar la descarga desde la página oficial, descomprimir el archivo y copiar todo su contenido dentro del espacio de trabajo que se desee.

4.1.4.2. Instalación de Rosinterface

Para instalar el complemento RosInterface se requiere crear la librería de forma manual ya que las existentes no funcionan o no se han desarrollado para ROS Kinetic y Ubuntu Xenial. Crear las librerías de este complemento requiere seguir los siguientes pasos:

- Crear una carpeta para construir la librería fuera del espacio de trabajo ejemplo: *lib_build*
- Ingresar al directorio de *V-rep / programming / ros_packages* y copiar las carpetas siguientes carpetas a *lib_build*

vrep_common

vrep_joy

vrep_plugin

- Editar el archivo *package.xml* dentro de *vrep_common*, eliminando la línea que se muestra a continuación.

  <build_depend>opencv2</ build_depend >

- Abrir el terminal desde esta ubicación y ejecutar los siguientes comandos

  $ mkdir build

  $ cd build

  $ cmake ..

  $ make –j

- Ir a la carpeta *vrep_plugin* y ejecutar el comando

  $ source ../vrep_common/build/devel/setup.bash

- Luego hacer el mismo procedimiento anterior, editar el archive *package.xml* dentro de *vrep_plugin*, eliminando la siguiente línea

  <build_depend>opencv2</ build_depend >

- Abrir el terminal en esta ubicación y ejecutar el comando

  $ mkdir build

  $ cd build

  $ cmake ..

  $ make –j

- Ir a la carpeta *build/devel/lib* creada dentro de *vrep_plugin*, copiar el archivo *lib_repExtRos.so*, y pegarlo en la carpeta de instalación de V-REP
- Una vez realizado estos pasos, al iniciar V-REP se puede ver en el terminal que aparece "Plugin 'RosInterface': load succeeded" como se observa en la Figura 40. Lo cual indica que el complemento está funcionando de forma correcta y ya se puede hacer uso de V-REP como un nodo ROS.

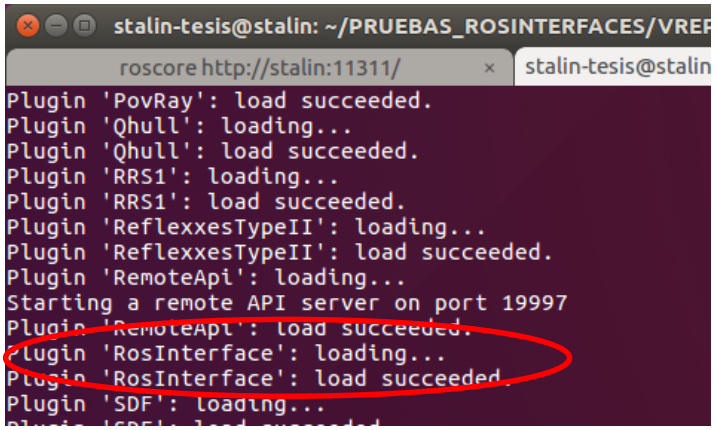


Figura 40. Carga del complemento RosInterface.

4.2. Aplicación

Para continuar con el desarrollo del proyecto y la implementación del sistema de control de posición angular de las articulaciones, se dispuso realizar una aplicación con los manipuladores robóticos, en la cual se pueda verificar el funcionamiento de las partes repotenciadas. Para ello se decidió realizar la reproducción de movimiento de los manipuladores CRS A255 a modo de espejo.

El sistema estará encargado de controlar el manipulador (Robot_2), mediante el movimiento del robot (Virtual_2) en el software de simulación V-REP, y el manipulador (Robot_1) deberá realizar los mismos movimientos pero a modo de espejo y esto debe reflejarse nuevamente en el software de simulación V-REP pero en el robot (Virtual_1). Para esto se requiere que ambas tarjetas de la interfaz de conversión estén funcionando al mismo tiempo, al igual que los

controladores difusos ROS para cada uno de los manipuladores. En la Figura 41 se puede observar de mejor manera el funcionamiento de la aplicación.

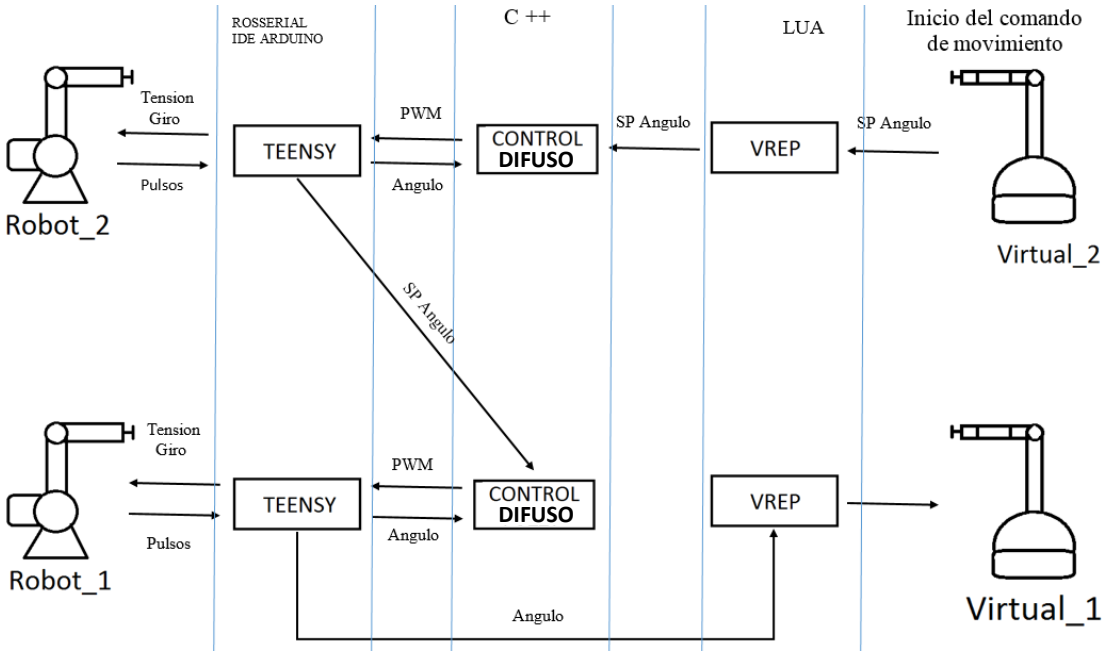


Figura 41. Aplicación: Reproducción de movimiento a modo de espejo

4.3. Algoritmos de control

En esta sección se explicará a detalle los algoritmos implementados para cada una de las partes involucradas en el control de posición angular de los motores de los manipuladores robóticos CRS A255. Tomando en cuenta que el lenguaje de programación es distinto en cada una, pero al conectarlas todas con la estructura ROS se puede realizar la comunicación entre ellas.

4.3.1. Algoritmo para el entono de simulación V-REP

Para iniciar con la aplicación propuesta se requiere él envió de datos desde el entorno de simulación, por lo cual es necesario escoger un robot virtual que asemeje en características a los manipuladores CRS A255, y este sea el encargado de indicar la posición que deberá adoptar cada una de las articulación de los manipuladores reales.

De entre todos los modelos que ofrece V-REP se seleccionó el brazo robótico ABB IRB 140 que se puede observar en la Figura 42, debido a su gran similitud en características al manipulador CRS A255, su única diferencia es que cuenta con un grado de libertad extra entre la muñeca y el codo, el mismo que no se tomara en cuenta al momento de enviar los datos.

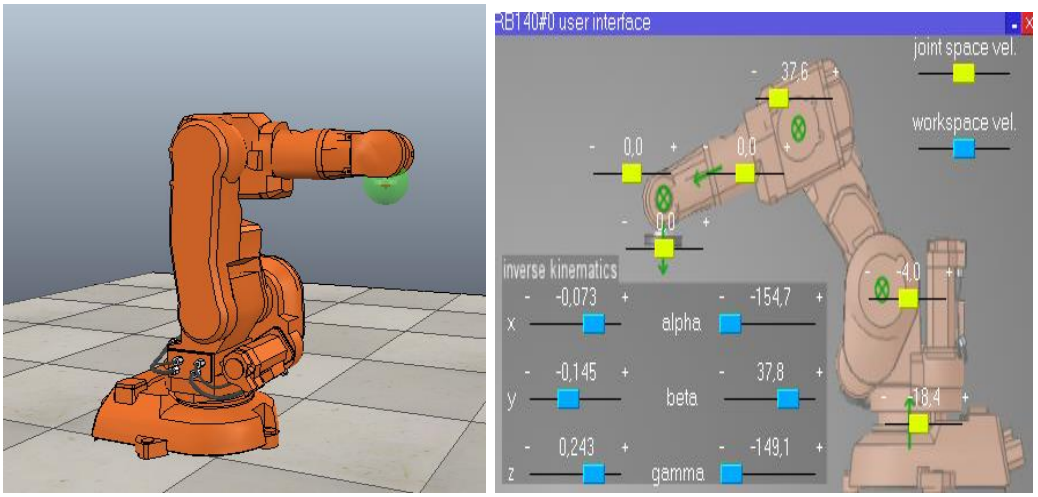


Figura 42. Robot Virtual semejante a CRS A255 (IZQ),
Control de mando de posición de las articulaciones (DER)

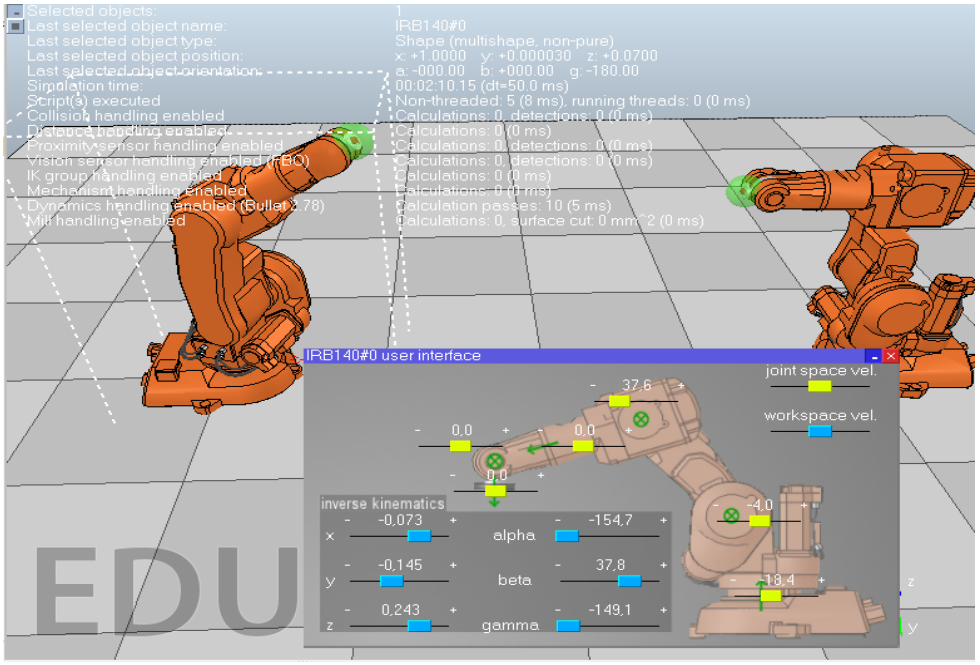


Figura 43. Escenario en entorno de simulación V-REP

El escenario completo del entorno de simulación se observa en la Figura 43. Donde el robot que se encuentra en el lado izquierdo (Virtual_2) es el encargado de enviar la consigna mediante los deslizadores de su interfaz de usuario y el de la derecha se encarga del monitoreo (Virtual_1).

Los límites de los ángulos para el manipulador se detallan en la Tabla 6.

Tabla 6.
Límites angulares de articulaciones

ARTICULACIÓN	ÁNGULOS
Cintura	-170º a 170º
Hombro	-90º a 20º
Codo	-40º a 90º
Pitch muñeca	-100º a 100º
Roll muñeca	-170º a 170º

Desde este punto en adelante se utilizarán los siguientes nombres para hacer referencia a los robots y de esta manera no existan confusiones:

- CRS A255 Con repotenciación de su estructura = Robot_1
- CRS A255 Sin repotenciación de su estructura = Robot_2
- ABB IRB 140 Encargado de recibir datos = Virtual_1
- ABB IRB 140 Encargado de enviar datos = Virtual_2

Una de las ventajas que ofrece V-REP es que muchos de los robots cuentan con el script de funcionamiento, permitiendo su modificación de acuerdo a las necesidades de cada caso en particular. Para ello hay que tener en cuenta que el script que utilizan los robots en V-REP para su simulación está escrito en lenguaje de programación LUA y consta de tres partes esenciales para el uso de ROS:

- Inicialización, donde se definen los tópicos ROS, con su nombre y tipo de dato.

- Ejecución, esta sección se encarga de realizar la simulación del robot, procesa toda la información y hace uso de los datos obtenidos gracias a los tópicos ROS.
- Limpieza, donde se cierran todas los tópicos ROS abiertos, una vez el simulador deja su ejecución.

4.3.1.1. Tópicos ROS en V-REP

Como se necesita que el robot Virtual_2 envié los valores de los ángulos de cada una de sus articulaciones, se debe implementar los tópicos ROS que se muestran en la Tabla 7.

Tabla 7.
Editores y suscriptores del robot virtual_2

	TÓPICO	TIPO DE DATO	FUNCIÓN
EDITORES	virtual_2_angulo_cintura	Float32	Publica el valor del ángulo en que se encuentra la articulación de la cintura
	virtual_2_angulo_hombro	Float32	Publica el valor del ángulo en que se encuentra la articulación del hombro
	virtual_2_angulo_codo	Float32	Publica el valor del ángulo en que se encuentra la articulación de codo
	virtual_2_angulo_pitch	Float32	Publica el valor del ángulo en que se encuentra la articulación del pitch de la muñeca
	virtual_2_angulo_roll	Float32	Publica el valor del ángulo en que se encuentra la articulación del roll de la muñeca

Además se requiere agregar el robot Virtual_1, este será el encargado de monitorear el movimiento realizado por el Robot_1, y en función de esas lecturas de datos realizar el movimiento adecuado. Se debe implementar los tópicos ROS que se muestran en la Tabla 8.

Tabla 8.
Editores y suscriptores del robot virtual_1

	TÓPICO	TIPO DE DATO	FUNCIÓN
SUSCRIPTOR	robot_1_angulo_cintura	Float32	Recibe el valor del ángulo en que se encuentra la articulación de la cintura desde el robot_1
ES	robot_1_angulo_hombro	Float32	Recibe el valor del ángulo en que se encuentra la

CONTINÚA

		articulación del hombro desde el robot_1
robot_1_angulo_codo	Float32	Recibe el valor del ángulo en que se encuentra la articulación de codo desde el robot_1
robot_1_angulo_pitch	Float32	Recibe el valor del ángulo en que se encuentra la articulación del pitch de la muñeca desde el robot_1
robot_1_angulo_roll	Float32	Recibe el valor del ángulo en que se encuentra la articulación del roll de la muñeca desde el robot_1

Todas las modificaciones en los Script de los robots Virtual_1 y Virtual_2 de V-REP se pueden ver en el Anexo 1.

4.3.2. Algoritmo para el controlador difuso C++

4.3.2.1. Lógica difusa

Para entender la lógica difusa se debe recordar la lógica clásica en la cual un hecho solamente puede tener dos posibles valores de veracidad, pudiendo ser una afirmación verdadera o falsa en la lógica difusa un hecho ya no tiene únicamente dos posibles valores de verdad sino que puede ser parcialmente verdadera o parcialmente falsa, es decir se le asigna un grado de pertenencia al hecho pudiendo este tomar valores de verdad entre 0 y 1.

Por lo tanto se puede describir a la lógica difusa como un sistema interpretativo donde a los elementos de un conjunto se les asigna grados de pertenencia relativa, entre dos valores o fronteras. (Kouro samir, 2010).

Esto permite incorporar sentencias del lenguaje común para asignar un grado de pertenencia a un suceso por ejemplo en la velocidad de un auto, se puede decir: el auto va muy rápido, el auto va rápido, el auto va despacio y el auto va muy despacio y su grado de pertenencia seria el que se observa en la Figura 44.

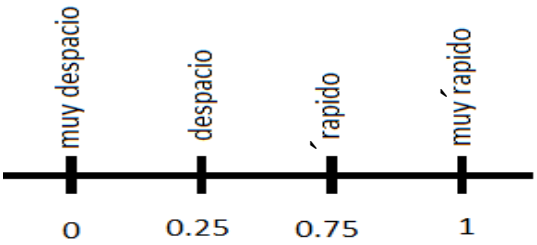


Figura 44. Grado de pertenencia a un suceso utilizando lenguaje común

Para cada conjunto difuso existe una función de pertenencia que indica en qué medida un elemento forma parte de ese conjunto difuso. Estas funciones de pertenencia pueden ser del tipo trapezoidal, triangular, curvas, entre otras. El conjunto difuso que se utiliza con más frecuencia es el de triángulos simétricos debido a que ofrece buenos resultados y es fácil de implementar, razón por la cual se puede utilizar para el desarrollo de controladores. (Rivera, 2015). Un controlador difuso está compuesto de tres partes importantes que son:

- Fusificador, Encargado de convertir variables numéricas en expresiones lingüísticas de acuerdo a su función de pertenencia.
- Controlador difuso, basa su funcionamiento en reglas heurísticas del tipo: ¨Si¨ - sucede tal evento - ¨entonces¨ - realizar tal acción, estas reglas deben ser sencillas y eficientes.
- Defusificador, Convierte en variables numéricas el resultado obtenido del controlador.

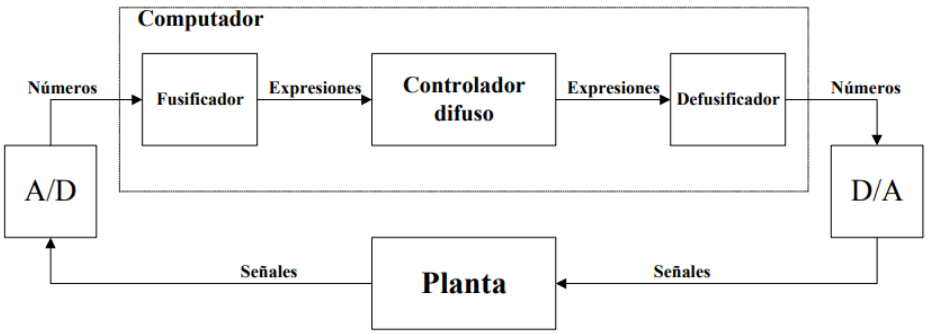


Figura 45. Partes de un controlador difuso
Fuente: (Kouro samir, 2010)

4.3.2.2. Diseño del controlador difuso

Se realizará el control de posición angular de cada una de las articulaciones del manipulador robótico CRS A255, mediante el uso de lógica difusa, para lo cual se ha definido el lazo de control que se puede ver en la Figura 46.

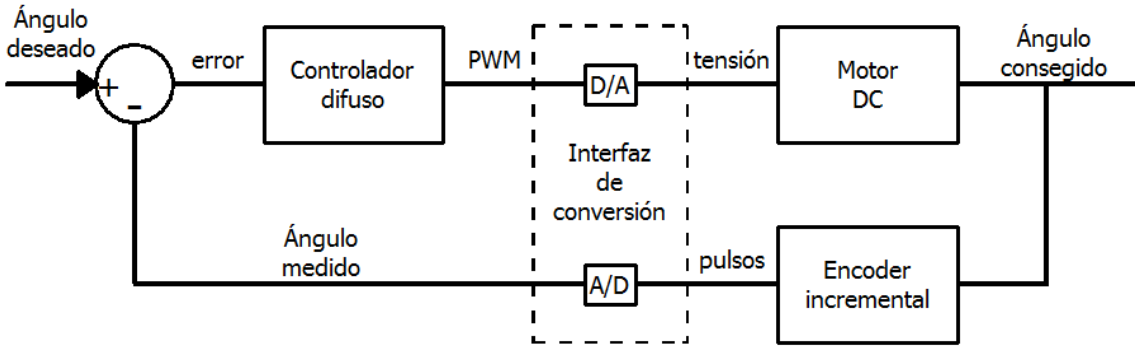


Figura 46. Lazo de control para la posición de motores del manipulador CRS A 255

Como el controlador difuso cuenta en su entrada con el error entre el ángulo deseado y el ángulo medido se definen las variables lingüísticas que se muestran en la Tabla 9.

Tabla 9.
Variables lingüísticas para el error

VARIABLE LINGÜÍSTICA	SIGNIFICADO
ENG	Error negativo grande
ENM	Error negativo medio
ENP	Error negativo pequeño
EC	Error cero
EPP	Error positivo pequeño
EPM	Error positivo medio
EPG	Error positivo grande

Sus valores se observan en la Figura 47. Se normalizó al valor de -1 a 1 debido a que se utilizará el mismo controlador para cada una de las articulaciones, por lo cual se deberá hacer una multiplicación por el número de grados que pueda girar cada articulación.

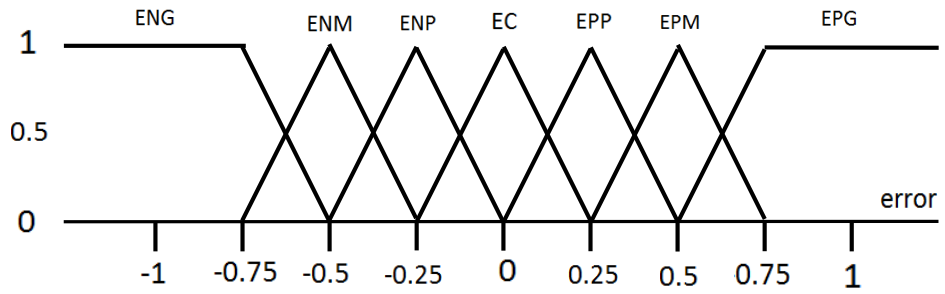


Figura 47. Funciones de pertenencia correspondientes al error

Sus funciones de pertenencia son las siguientes:

- $ENG = 1\ ;\ error\ \leq -0.75$
- $ENG = (-4 * error) - 2\ ;\ -0.75 <\ error\ \leq -0.5$
- $ENM = (4 * error) + 3\ ;\ -0.75 <\ error\ \leq -0.5$
- $ENM = (-4 * error) - 1\ ;\ -0.5 <\ error\ \leq -0.25$
- $ENP = (4 * error) + 2\ ;\ -0.5 <\ error\ \leq -0.25$
- $ENP = (-4 * error)\ ;\ -0.25 <\ error\ \leq 0$
- $EC = (4 * error) + 1\ ;\ -0.25 <\ error\ \leq 0$
- $EC = (-4 * error) + 1\ ;\ 0 <\ error\ \leq 0.25$
- $EPP = (4 * error)\ ;\ 0 <\ error\ \leq 0.25$
- $EPP = (-4 * error) + 2\ ;\ 0.25 <\ error\ \leq 0.5$
- $EPM = (4 * error) - 1;\ 0.25 <\ error\ \leq 0.5$
- $EPM = (-4 * error) + 3\ ;\ 0.5 <\ error\ \leq 0.75$
- $EPG = (4 * error) - 2\ ;\ 0.5 <\ error\ \leq 0.75$
- $EPG = 1;\ \geq 0.75$

A la salida del controlador se debe obtener un valor de PWM por lo cual los términos lingüísticos definidos son los indicados en la Tabla 10.

Tabla 10.
Variables lingüísticas para el PWM

VARIABLE LINGÜÍSTICA	SIGNIFICADO
NG	PWM negativo grande
NM	PWM negativo medio
NP	PWM negativo pequeño
C	PWM cero
PP	PWM positivo pequeño
PM	PWM positivo medio
PG	PWM positivo grande

Sus valores se observan en la Figura 48, se normalizó al valor de -1 a 1 debido a que se puede utilizar para distintos valores de salida PWM para lo cual solo bastaría con multiplicar el valor obtenido por el valor PWM requerido como máximo.

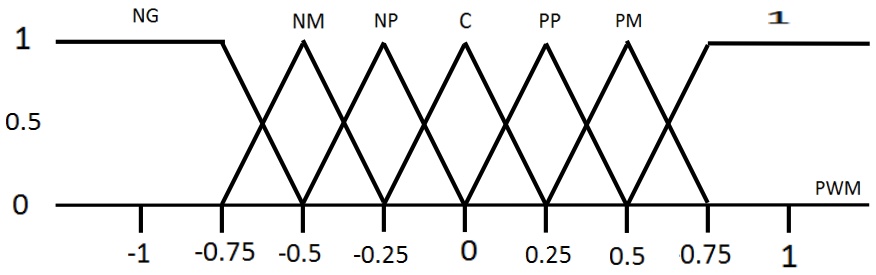


Figura 48. Funciones de pertenencia correspondientes al PWM

Una vez definidos los términos lingüísticos para la entrada y salida del controlador difuso, además de las funciones de pertenencia que convierten las variables numéricas en lingüísticas, se requiere que el controlador entienda que acción va a realizar con esos valores. Para ello se definen las reglas de control, las cuales se resumen en la Tabla 11.

Tabla 11.
Reglas de control para el controlador difuso

SI	ENTONCES
ENG	NG
ENM	NM
ENP	NP
ECO	C
EPP	PP
EPM	PM
EPG	PG

Para realizar la defusificación se necesita hacer el cálculo del centroide de las áreas que estén implicadas en las reglas de control. Se detectaron 12 casos posibles para estas reglas de control, que se explican en el Anexo 2.

Una vez se cuenta con el valor del PWM a la salida del controlador se realiza una etapa de compensación, debido a que el controlador difuso desarrollado no toma en cuenta la dinámica del robot, por lo que en ciertos ángulos de cada articulación el movimiento es realizado con ayuda de la gravedad, lo cual vuelve al control inestable.

Para realizar esta etapa de compensación se utilizaron las condiciones detalladas en la Tabla 12, de acuerdo a cada articulación. En la cintura y en el roll de la muñeca no se requiere compensación debido a que su movimiento no se ve afectado por la acción de la gravedad.

Tabla 12.
Compensación de PWM

ARTICULACIÓN	ÁNGULO	GIRO	PWM FINAL
hombro	Θ>0	Horario	(PWM*0.69)+70
hombro	Θ>0	Anti horario	(PWM*0.29)-180
hombro	Θ<0	Horario	255
hombro	Θ<0	Anti horario	144
codo	Θ<60	Horario	(PWM*0.13)+220
codo	Θ<60	Anti horario	120
codo	Θ>60	Horario	(PWM*0.15)+215
codo	Θ>60	Anti horario	-160
pitch	Θ>0	Horario	(PWM*0.15)+215

CONTINÚA

pitch	Θ>0	Anti horario	-150
pitch	Θ<0	Horario	(PWM*0.15)+215
pitch	Θ<0	Anti horario	110

El código fuente para los controladores del Robot_1 y Robot_2 se puede observar completamente en el Anexo 3.

4.3.2.3. Tópicos ROS en controladores difusos

Para que el controlador del robot_2 pueda funcionar se requiere que tome los valores de los ángulos desde el entorno de simulación del robot Virtual_2 y envié los valores PWM hacia el Robot_2, por lo cual se debe implementar los tópicos ROS que se muestran en la Tabla 13.

Tabla 13.
Editores y suscriptores del controlador difuso del robot_2

	TÓPICO	TIPO DE DATO	FUNCIÓN
EDITORES	robot_2_PWM_cintura	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación de la cintura
	robot_2_PWM_hombro	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación del hombro
	robot_2_PWM_codo	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación del codo
	robot_2_PWM_pitch	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación del pitch de la muñeca
	robot_2_PWM_roll	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación del roll de la muñeca
SUSCRIPTORES	robot_2_angulo_cintura	Float32	Recibe el valor del ángulo en que se encuentra la articulación de la cintura para calcular el error
	robot_2_angulo_hombro	Float32	Recibe el valor del ángulo en que se encuentra la articulación del hombro para calcular el error

CONTINÚA

	robot_2_angulo_codo	Float32	Recibe el valor del ángulo en que se encuentra la articulación de codo para calcular el error
	robot_2_angulo_pitch	Float32	Recibe el valor del ángulo en que se encuentra la articulación del pitch de la muñeca para calcular el error
	robot_2_angulo_roll	Float32	Recibe el valor del ángulo en que se encuentra la articulación del roll de la muñeca para calcular el error
	virtual_2_angulo_cintura	Float32	Recibe el valor del ángulo en que se encuentra la articulación de la cintura como set point
	virtual_2_angulo_hombro	Float32	Recibe el valor del ángulo en que se encuentra la articulación del hombro como set point
	virtual_2_angulo_codo	Float32	Recibe el valor del ángulo en que se encuentra la articulación de codo como set point
	virtual_2_angulo_pitch	Float32	Recibe el valor del ángulo en que se encuentra la articulación del pitch de la muñeca como set point

Además para que el controlador del robot_1 también pueda funcionar se requiere que tome los valores de los ángulos desde el robot_2 y envié los valores PWM hacia el Robot_1, para esta tarea se requiere utilizar los tópicos ROS de la Tabla 14.

Tabla 14.
Editores y suscriptores del controlador difuso del robot_1

	TÓPICO	TIPO DE DATO	FUNCIÓN
EDITOR	robot_1_PWM_cintura	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación de la cintura
	robot_1_PWM_hombro	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación del hombro
	robot_1_PWM_codo	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación del codo

CONTINÚA

	robot_1_PWM_pitch	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación del pitch de la muñeca
	robot_1_PWM_roll	Float32	Publica el valor del PWM con que se debe realizar el movimiento de la articulación del roll de la muñeca
SUSCRIPTORES	robot_1_angulo_cintura	Float32	Recibe el valor del ángulo en que se encuentra la articulación de la cintura para calcular el error
	robot_1_angulo_hombro	Float32	Recibe el valor del ángulo en que se encuentra la articulación del hombro para calcular el error
	robot_1_angulo_codo	Float32	Recibe el valor del ángulo en que se encuentra la articulación de codo para calcular el error
	robot_1_angulo_pitch	Float32	Recibe el valor del ángulo en que se encuentra la articulación del pitch de la muñeca para calcular el error
	robot_1_angulo_roll	Float32	Recibe el valor del ángulo en que se encuentra la articulación del roll de la muñeca para calcular el error
	robot_2_angulo_cintura	Float32	Recibe el valor del ángulo en que se encuentra la articulación de la cintura como set point
	robot_2_angulo_hombro	Float32	Recibe el valor del ángulo en que se encuentra la articulación del hombro como set point
	robot_2_angulo_codo	Float32	Recibe el valor del ángulo en que se encuentra la articulación de codo como set point
	robot_2_angulo_pitch	Float32	Recibe el valor del ángulo en que se encuentra la articulación del pitch de la muñeca como set point
	robot_2_angulo_roll	Float32	Recibe el valor del ángulo en que se encuentra la articulación del roll de la muñeca como set point

4.3.3. Algoritmo para la interfaz de conversión

La interfaz de conversión es el encargado de leer los pulsos enviados por los encoders de cada una de las articulaciones y a partir de ellos generar el valor de los ángulos que serán enviados a los controladores difusos, como se observa en la Figura 46.

También se encarga de recibir el valor PWM desde los controladores difusos y en función del signo que estos tengan, decidir el sentido de giro de cada uno de los motores y la activación o desactivación de los frenos. Mientras va sumando o restando el valor del ángulo dentro de su memoria. Para lo cual se utiliza la siguiente lógica:

- Si el valor de PWM recibido es mayor a cero, esto indica a la articulación que debe moverse en sentido horario, pero además el valor actual del ángulo debe irse sumando en función de los datos recibidos por el encoder y ese ángulo será enviado hacia el controlador en cada nueva lectura del encoder.
- En su contraparte si el PWM recibido es menor a cero, esto indica un giro anti horario en la articulación, por lo cual el valor del ángulo debe irse restando con cada lectura del encoder y se enviará hacia el controlador para su siguiente proceso.

Cabe mencionar que en cuanto el valor del PWM es muy próximo a cero, se detiene el motor y se activan los frenos correspondientes a esa articulación. Esto se hace con el fin de evitar oscilaciones bruscas al llegar a la posición deseada.

Todo el script de programación de las tarjetas Teensy 3.6 correspondientes al robot_1 y robot_2 se pueden ver en el Anexo 4.

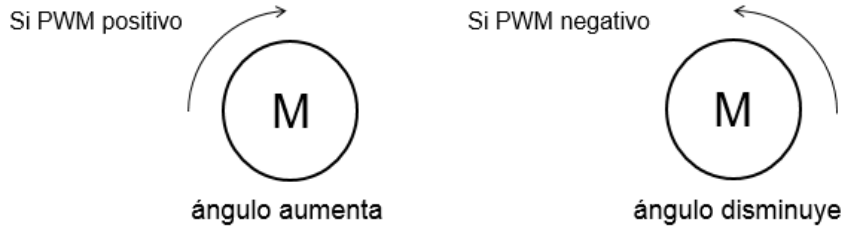


Figura 49. Lógica en interfaz de conversión

4.3.3.1. Tópicos ROS para la interfaz de conversión

El circuito de control y potencia del Robot_1 requiere los valores de PWM enviados desde el controlador del Robot_1 para su funcionamiento y además envía la información sobre el valor de ángulos del Robot_1 de cada articulación, por lo cual se debe implementar los tópicos ROS que se muestran en la Tabla 15, para su correcto funcionamiento.

Tabla 15.
Editores y suscriptores de la interfaz de conversión

	TÓPICO	TIPO DE DATO	FUNCIÓN
EDITORES	robot_1_angulo_cintura	Float32	Publica el valor del ángulo en que se encuentra la articulación de la cintura
	robot_1_angulo_hombro	Float32	Publica el valor del ángulo en que se encuentra la articulación del hombro
	robot_1_angulo_codo	Float32	Publica el valor del ángulo en que se encuentra la articulación de codo
	robot_1_angulo_pitch	Float32	Publica el valor del ángulo en que se encuentra la articulación del pitch de la muñeca
	robot_1_angulo_roll	Float32	Publica el valor del ángulo en que se encuentra la articulación del roll de la muñeca
SUSCRIPTORES	robot_1_PWM_cintura	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación de la cintura
	robot_1_PWM_hombro	Float32	Recibe el valor del PWM con que se debe realizar

CONTINÚA

			el movimiento de la articulación del hombro
	robot_1_PWM_codo	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación del codo
	robot_1_PWM_pitch	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación del pitch de la muñeca
	robot_1_PWM_roll	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación del roll de la muñeca

De igual manera el circuito de control y potencia del Robot_2 requiere los valores de PWM enviados desde el controlador del Robot_2 para su funcionamiento y además debe enviar la información sobre el valor de ángulos de cada articulación del Robot_2, los tópicos ROS a implementar son los mostrados en la Tabla 16.

Tabla 16.
Editores y suscriptores de la interfaz de conversión

	TÓPICO	TIPO DE DATO	FUNCIÓN
EDITORES	robot_2_angulo_cintura	Float32	Publica el valor del ángulo en que se encuentra la articulación de la cintura
	robot_2_angulo_hombro	Float32	Publica el valor del ángulo en que se encuentra la articulación del hombro
	robot_2_angulo_codo	Float32	Publica el valor del ángulo en que se encuentra la articulación de codo
	robot_2_angulo_pitch	Float32	Publica el valor del ángulo en que se encuentra la articulación del pitch de la muñeca
	robot_2_angulo_roll	Float32	Publica el valor del ángulo en que se encuentra la articulación del roll de la muñeca
SUSCRIPTORES	robot_2_PWM_cintura	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación de la cintura

CONTINÚA

robot_2_PWM_hombro	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación del hombro
robot_2_PWM_codo	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación del codo
robot_2_PWM_pitch	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación del pitch de la muñeca
robot_2_PWM_roll	Float32	Recibe el valor del PWM con que se debe realizar el movimiento de la articulación del roll de la muñeca

CAPÍTULO V

PRUEBAS Y RESULTADOS

5.1. Resultados en la optimización de hardware

Una vez realizadas las adecuaciones necesarias en cada uno de los manipuladores robóticos se observan los siguientes resultados. En cuanto a hardware se puede decir que se mejoró los manipuladores robóticos CRS A255 en los siguientes aspectos

- La activación de frenos ahora es independiente de cada articulación, teniendo en cuenta que al activarse un paro de emergencia o desconectarse de la energía, los mismos se activarán garantizando seguridad para los operarios y la integridad del manipulador.
- Se puede medir la posición de cada una de las articulaciones gracias a que se logró acoplar nuevos encoders, lo cual era imposible antes de la repotenciación por la falta de los mismos.
- Ahora el manipulador CRS A255 es íntegro en cuanto a sus partes ya que anteriormente se encontraba desarmado e inutilizable, estos cambios se pueden notar en la Figura 50, Figura 51 y Figura 52.

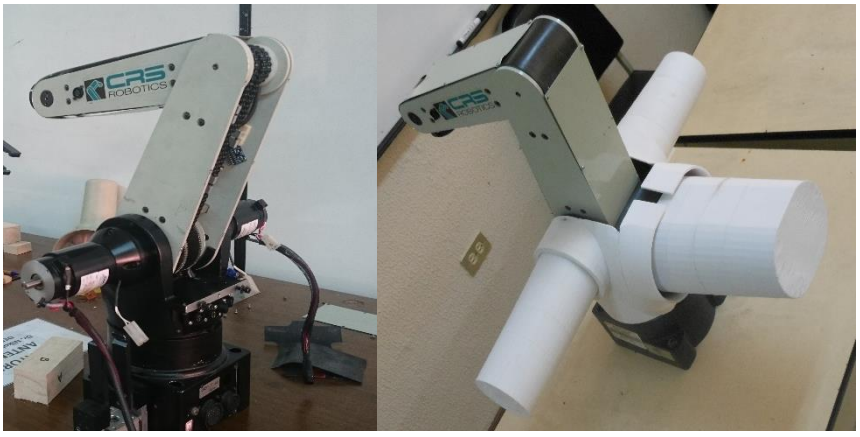


Figura 50. Estado inicial (IZQ) y estado actual (DER) del hardware de los manipuladores CRS A255.

Figura 51. Estado inicial (IZQ) y estado actual (DER)
de los encoders de los manipuladores CRS A255.

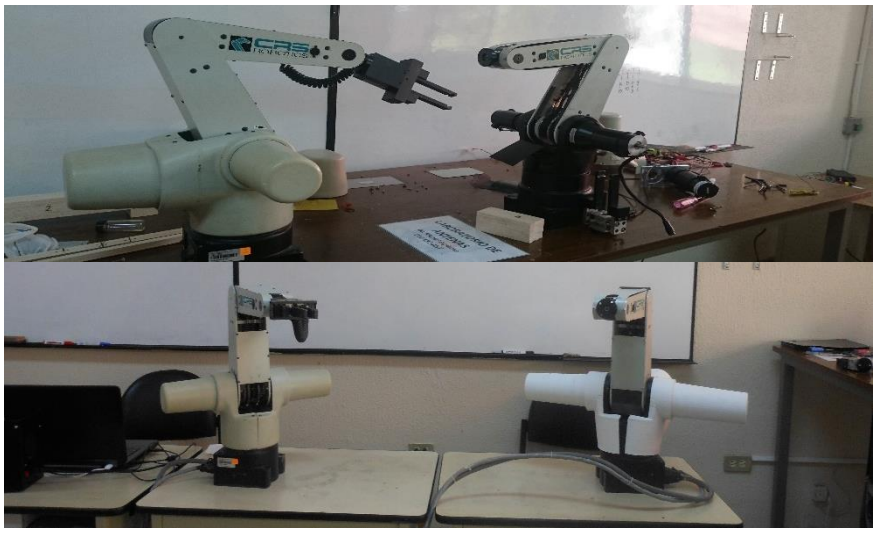


Figura 52. Estado inicial (ARRIBA) y estado actual (DEBAJO)
de los manipuladores CRS A255.

5.2. Resultados de la creación de interfaz de conversión

Se creó nuevas tarjetas para la interfaz de conversión, obteniendo los resultados que se pueden ver la Figura 53, Figura 54, Figura 55 y Figura 56 , las cuales están funcionales, dotando así de un medio de transmisión de datos entre el manipulador y su controlador, superando el problema inicial en el cual sus circuitos de control y potencia se encontraban inutilizables.

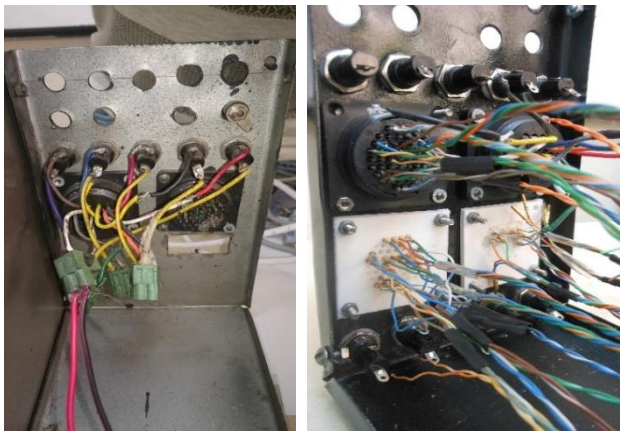


Figura 53. Estado inicial (IZQ) y estado actual (DER) de carcasa internamente.

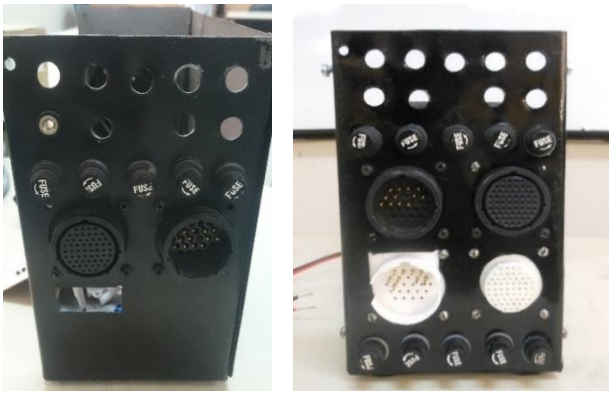


Figura 54. Estado inicial (IZQ) y estado actual (DER) de conectores en carcasa.

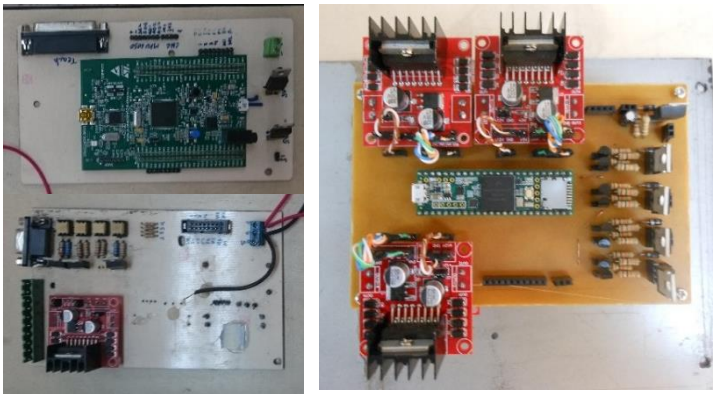


Figura 55. Estado inicial (IZQ) y estado actual (DER) de interfaz de conversión.

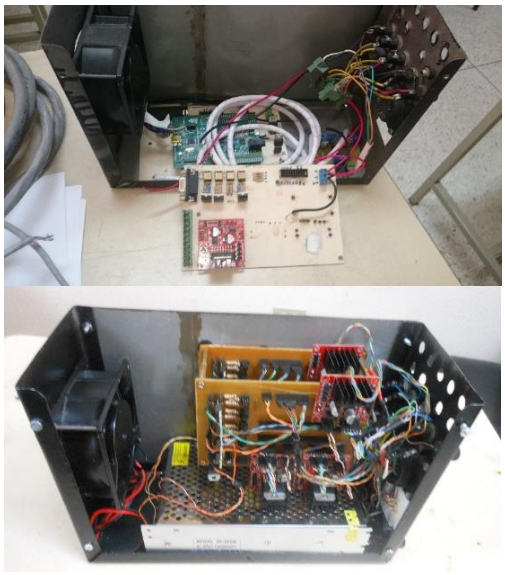


Figura 56. Estado inicial (ARRIBA) y estado actual (DEBAJO) de la interfaz de conversión montadas en su carcasa.

Al momento de probar su funcionamiento con los manipuladores CRS A255 se notó un exceso en la temperatura de los reguladores de voltaje, esto debido al consumo de corriente en los dispositivos que requerían la tensión de 3.3VDC, lo cual incluye a los encoder y módulos L298N en su parte de control. Como se muestra en la Tabla 17, la corriente consumida es de 950mA, teniendo en cuenta que el regulador LM317 según su hoja técnica puede suministrar un máximo de 1.5 A antes de quemarse; para evitar fallos en esta parte del circuito, se decidió colocar un disipador de calor el cual resolvió el problema; este se aprecia en la Figura 57.

Tabla 17.
Consumo de corriente dispositivos de control

DISPOSITIVO	CANTIDAD	CONSUMO	TOTAL
Encoder	5	100mA	500mA
Módulo L298N	3	150mA	450mA
			950mA

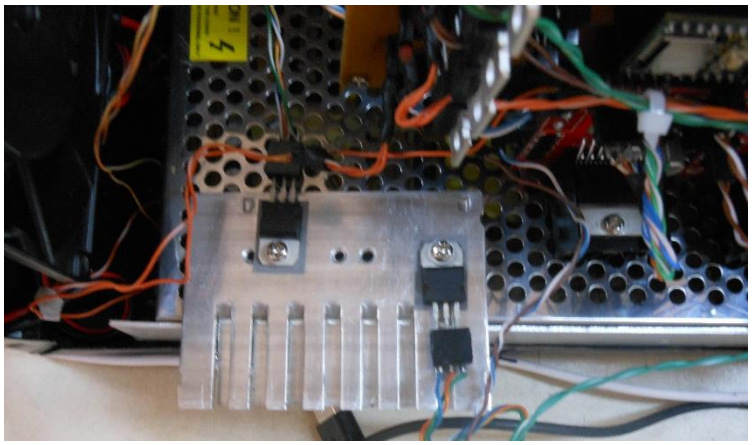


Figura 57. Disipador para evitar el sobrecalentamiento en regulador de 3.3Vdc

5.3. Resultados del sistema de control de posición angular de motores

Para realizar la prueba del sistema de control de posición angular de las articulaciones de los manipuladores, se realizó las conexiones necearías para el funcionamiento de la aplicación descrita en los capítulos anteriores, el montaje se puede verificar en la Figura 58.

Figura 58. Montaje de manipuladores para la aplicación

El sistema es capaz de replicar los movimientos de robot Virtual_2 en el Robot_2 y a su vez el Robot_1 realiza el movimiento a modo espejo el cual se ve reflejado en el entorno de simulación

en el robot Virtual_1. En la Figura 59, se puede observar el funcionamiento del entorno de simulación en conjunto con los manipuladores.

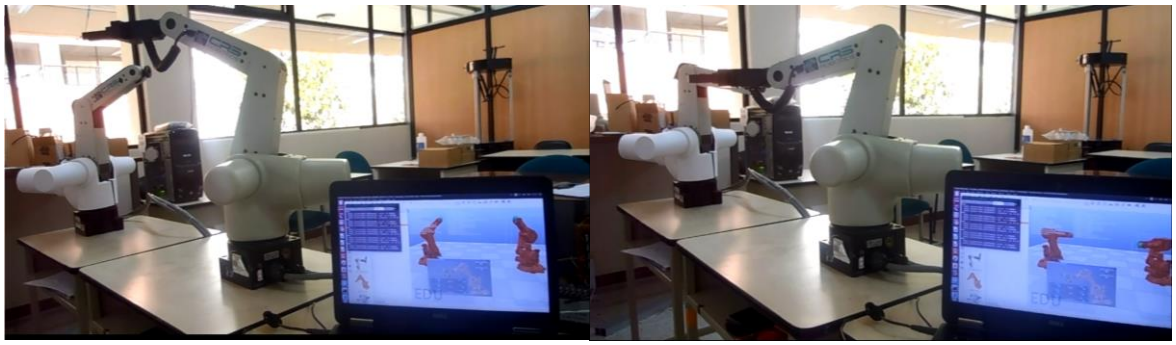


Figura 59. Monitorización en V-REP

5.3.1. Limitaciones

Después de probar el sistema completo por varias ocasiones, se observó que su funcionamiento es correcto pero teniendo en cuenta las siguientes limitaciones:

- El movimiento se restringe a una articulación a la vez, esto debido a que al momento de intentar mover todas las articulaciones en conjunto se pierden datos en la lectura de los encoders. Mediante un análisis se definió que cada señal de los encoder requiere ser manejada mediante una interrupción. El microcontrolador al realizar su trabajo de manera secuencial, detiene la ejecución de cualquier otra tarea para abarcar esa interrupción, dejando de lado las otras interrupciones que se den al mismo instante de tiempo.
- Velocidad reducida al momento de realizar movimientos. Al realizar movimientos en las articulaciones los encoders envían en promedio 500 pulsos por grado, debiendo procesarse cada uno de esos pulso para poder realizar el control, lo que implica realizar el procesamiento en el circuito de control para enviar el valor como grado al controlador, para que este haga su trabajo y devuelva un valor PWM, que nuevamente el circuito de control debe analizar mediante su

lógica para enviar la orden hacia el motor. Por lo que al realizar movimientos muy rápidos se pierden muchos de esos datos ocasionando que el sistema no responda adecuadamente.

- El robot simulado excede los valores de ángulos del robot real, por lo cual hay que tener cierto cuidado al momento de mover al robot Virtual_2 encargado de iniciar el proceso y no sobrepasar los límites del manipulador real. Pudiendo limitarse este comportamiento mediante código.

Una vez realizada la repotenciación total de los manipuladores CRS A225 se realizaron una serie de pruebas para comprobar su funcionamiento acorde a la aplicación creada.

5.4. Pruebas de error en el movimiento de articulaciones

Para esta prueba se realizó el movimiento de cada articulación para diez valores de posición angulares distintos, con el fin de obtener el error absoluto promedio de cada articulación. Los resultados de esta prueba están expresados en grados y se pueden ver en la Tabla 18, Tabla 19, Tabla 20, Tabla 21 y Tabla 22.

Tabla 18.
Prueba de error en articulación de la cintura.

	Virtual_2	Robot_2	Error (Robot_2 y Virtual_2)	Robot_1	Error (Robot_1 y Virtual_2)	Error (Robot_1 y Robot_2)
CINTURA	-46	-42	4	-39	7	3
	1	-3	4	-5	6	2
	87	83	4	85	2	2
	61	57	4	60	1	3
	-90	-86	4	-85	5	1
	47	44	3	41	6	3
	-5	-2	3	-1	4	1
	-33	-29	4	-25	8	4
	90	86	4	83	7	3
	-55	-51	4	-52	3	1
	0	-4	4	-7	7	3
PROMEDIO			3,82		5,09	2,36

Tabla 19.
Prueba de error en articulación del hombro.

	Virtual_2	Robot_2	Error (Robot_2 y Virtual_2)	Robot_1	Error (Robot_1 y Virtual_2)	Error (Robot_1 y Robot_2)
HOMBRO	20	25	5	22	2	3
	-2	0	2	4	6	4
	-12	-17	5	-18	6	1
	-6	-4	2	-7	1	3
	-22	-23	1	-26	4	3
	-13	-9	4	-15	2	6
	5	1	4	3	2	2
	17	22	5	24	7	2
	10	15	5	17	7	2
	7	8	1	4	3	4
	-4	-9	5	-10	6	1
PROMEDIO			3,55		4,18	2,82

Tabla 20.
Prueba de error en articulación del codo

	Virtual_2	Robot_2	Error (Robot_2 y Virtual_2)	Robot_1	Error (Robot_1 y Virtual_2)	Error (Robot_1 y Robot_2)
CODO	10	16	6	13	3	3
	53	52	1	48	5	4
	22	26	4	22	0	4
	9	13	4	11	2	2
	0	-1	1	0	0	1
	-8	-4	4	-16	8	12
	64	63	1	64	0	1
	50	55	5	58	8	3
	19	16	3	11	8	5
	1	-3	4	-2	3	1
	-12	-12	0	-14	2	2
PROMEDIO			3,00		3,55	3,45

Tabla 21.
Prueba de error en articulación del pitch de la muñeca.

	Virtual_2	Robot_2	Error (Robot_2 y Virtual_2)	Robot_1	Error (Robot_1 y Virtual_2)	Error (Robot_1 y Robot_2)
PITCH	10	1	9	10	0	9
	43	42	1	24	19	18
	50	38	12	46	4	8
	-21	-19	2	1	22	20
	71	69	2	66	5	3
	88	82	6	78	10	4
	-73	-67	6	-58	15	9
	-48	-34	14	-37	11	3
	6	3	3	2	4	1
	36	32	4	35	1	3
	-40	-35	5	-37	3	2
PROMEDIO			5,82		8,55	7,27

Tabla 22.
Prueba de error en articulación del roll de la muñeca.

	Virtual_2	Robot_2	Error (Robot_2 y Virtual_2)	Robot_1	Error (Robot_1 y Virtual_2)	Error (Robot_1 y Robot_2)
ROLL	95	94	1	71	24	23
	-38	-31	7	-28	10	3
	127	117	10	123	4	6
	-133	-120	13	-125	8	5
	57	47	10	37	20	10
	111	100	11	86	25	14
	32	26	6	25	7	1
	103	96	7	101	2	5
	-28	-20	8	-12	16	8
	73	67	6	71	2	4
	55	43	12	30	25	13
PROMEDIO			8,27		13,00	8,36

Luego de analizar los resultados obtenidos, se puede decir que el error generado al momento de realizar un movimiento en las articulaciones del manipulador robótico, se encuentra dentro de un rango de los ± 5 grados.

5.5. Prueba de tiempo de procesamiento en tarjetas

Esta prueba se realizó para saber cuál es el tiempo que se toman las tarjetas Teensy 3.6 al ejecutar el código programado cuando se envían y no se envían consignas para su procesamiento, con el fin de determinar porque existe un retraso en los manipuladores al momento de replicar el movimiento. Para ello se utilizó los mensajes de información de ROS, como se muestra en la Figura 60.

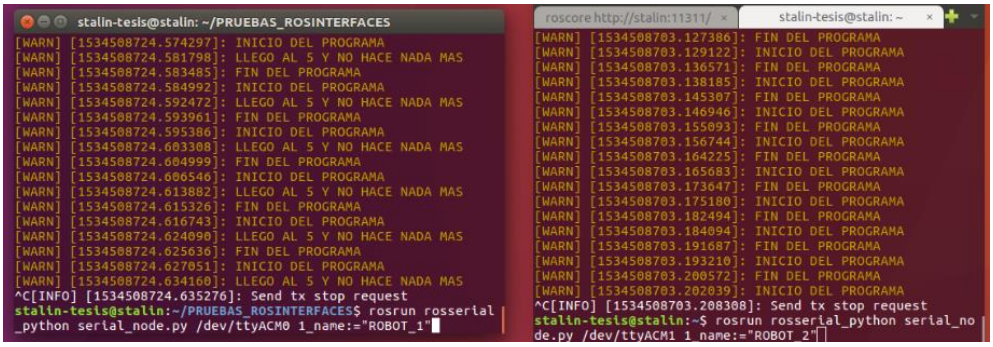


Figura 60. Toma de tiempos de procesamiento de tarjetas Teensy 3.6.

El valor se obtiene restando el tiempo mostrando en el mansaje ¨FIN DEL PROGRAMA¨ con el tiempo mostrado en el mensaje ¨INICIO DEL PROGRAMA¨, se tomaron 10 valores y se realizó el promedio, los resultados se pueden ver en la Tabla 23 y Tabla 24.Las unidades en las que se trabaja son segundos.

Tabla 23.
Tiempo de procesamiento Teensy sin consignas

TIEMPO DE PROCESAMIENTO TEENSY SIN CONSIGNAS					
Robot_1			**Robot_2**		
ti	**tf**	**t proceso**	**ti**	**tf**	**t proceso**
0,633859	0,643911	0,010052	0,829834	0,838075	0,008241
0,645488	0,654646	0,009158	0,83979	0,84714	0,00735

CONTINÚA

0,65621	0,66546	0,00925	0,848565	0,85623	0,007665
0,667057	0,676761	0,009704	0,857722	0,865941	0,008219
0,678428	0,689064	0,010636	0,867453	0,87523	0,007777
0,691179	0,700621	0,009442	0,876891	0,885384	0,008493
0,289531	0,298402	0,008871	0,887278	0,895928	0,00865
0,29981	0,308514	0,008704	0,897774	0,905941	0,008167
0,30996	0,31867	0,00871	0,907662	0,916162	0,0085
0,320194	0,329767	0,009573	0,146768	0,15407	0,007302
t proceso promedio		0,00941	**t proceso promedio**		0,0080364

Tabla 24.

Tiempo de procesamiento Teensy con consignas

TIEMPO DE PROCESAMIENTO TEENSY CON CONSIGNAS					
Robot_1			**Robot_2**		
ti	**tf**	**t proceso**	**ti**	**tf**	**t proceso**
0,101108	0,110563	0,009455	0,887917	0,897192	0,009275
0,112031	0,121378	0,009347	0,898779	0,908062	0,009283
0,12297	0,132612	0,009642	0,909579	0,92012	0,010541
0,134207	0,144255	0,010048	0,921776	0,931091	0,009315
0,145811	0,156573	0,010762	0,932788	0,942545	0,009757
0,158398	0,167907	0,009509	0,944107	0,95351	0,009403
0,169604	0,179316	0,009712	0,955619	0,965475	0,009856
0,180858	0,191173	0,010315	0,967521	0,977437	0,009916
0,192773	0,202053	0,00928	0,978921	0,991163	0,012242
0,203642	0,213313	0,009671	0,99298	1,008269	0,015289
t proceso promedio		0,0097741	**t proceso promedio**		0,0104877

Al analizar los tiempos obtenidos, se puede observar que el tiempo de procesamiento del robot_1 es de 9.77 cuando existen consignas a procesar, mientras que para el robot_2 el tiempo de procesamiento con consignas asciende a 10.4 milisegundo. Estos valores nos indican que existe un retraso de 0,63 milisegundos en la transmisión de datos entre las tarjetas Teensy de ambos manipuladores.

5.6. Prueba de tiempo de procesamiento en controladores difusos

Esta prueba se realizó para saber cuál es el tiempo que se toman los controladores de cada robot al ejecutar el código programado, cuando se envían y no se envían consignas para su procesamiento, con el fin de determinar porque existe un retraso en los manipuladores al momento de replicar el movimiento, para ello se utilizó los mensajes de información de ROS, como se muestra en la Figura 61.

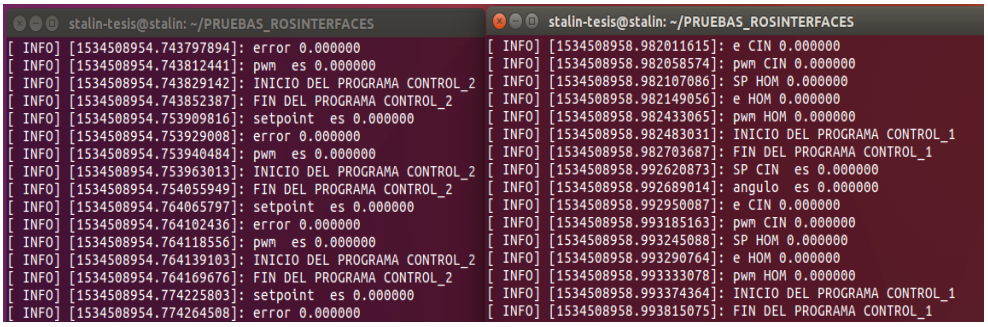


Figura 61. Toma de tiempos de procesamiento en los controladores

El valor se obtiene restando el tiempo mostrando en el mansaje ¨FIN DEL PROGRAMA¨ con el tiempo mostrado en el mensaje ¨INICIO DEL PROGRAMA¨, se tomaron 4 valores y se realizó el promedio, los resultados se pueden ver en la Tabla 25 y Tabla 26. Las unidades en las que se trabaja son segundos.

Tabla 25.
Tiempo de procesamiento controlador sin consignas.

TIEMPO DE PROCESAMIENTO EN CONTROLADORES SIN CONSIGNAS					
Robot_1			**Robot_2**		
ti	**tf**	**t proceso**	**ti**	**tf**	**t proceso**
0,982483	0,9827037	0,000221	0,7438291	0,7438524	2,3245E-05
0,9933744	0,9938151	0,000441	0,753963	0,7540559	9,2936E-05
0,0038885	0,0039603	7,19E-05	0,7641391	0,7641697	3,0573E-05
0,0144633	0,0145395	7,62E-05	0,7743022	0,7743442	4,2021E-05
t proceso promedio		0,000202	**t proceso promedio**		0,000047194

Tabla 26.
Tiempo de procesamiento controlador con consignas.

TIEMPO DE PROCESAMIENTO EN CONTROLADORES CON CONSIGNAS					
Robot_1			**Robot_2**		
ti	**tf**	**t proceso**	**ti**	**tf**	**t proceso**
0,2366993	0,2368365	0,0001371	0,2423554	0,2425912	0,0002358
0,2471509	0,2473042	0,0001534	0,2526487	0,2528782	0,0002294
0,2575153	0,2576243	0,000109	0,2629235	0,2630521	0,0001286
0,268017	0,2686235	0,0006066	0,2732576	0,2735912	0,0003335
t proceso promedio		0,0002515	**t proceso promedio**		0,0002318

El procesamiento del computador en el controlador del robot_1 y robot_2 es de 0.2515 milisegundos y 0.2318 milisegundos, respectivamente. En relación al procesamiento de la Teensy nos indica que existe un retraso de 9.51 milisegundos para el robot_1, y 10.16 milisegundos para el robot_2.

5.7. Prueba de tiempo en replicar movimiento

El tiempo que se demora el Robot_1 en copiar el movimiento del Robot_2, para ello se tomaron 5 mediciones con un cronómetro externo, las mismas que se expresan en segundos y estas se muestran en la Tabla 27.

Tabla 27.
Prueba replicar movimiento

TIEMPO EN REPLICAR MOVIMIENTO	
PRUEBA	**TIEMPO**
1	1,2
2	1,4
3	0,9
4	1,1
5	0,6
PROMEDIO	1,04

Los resultados indican que el tiempo que demora el Robot_1 en copiar el movimiento del Robot_2, es en promedio 1,04 segundos, lo cual muestra que existe un retraso en cuanto a la réplica del movimiento; principalmente generado por temas de manejo de información entre el distinto software.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- Para la construcción de la interfaz de coversión, se utilizó como elemento principal la tarjeta Teensy 3.6 cuyo microprocesador es de tipo secuencial, lo cual se notó no es eficiente, debido a que se deben realizar lecturas simultaneas en los encoders, perdiéndose datos con el manejo de interrupciones en el modelo secuencial, por lo que se requiere una tarjeta con procesamiento en paralelo para un mejor control de las articulaciones; o en su caso la implementación de operación en paralelo para microcontroladores.
- La lógica difusa es una buena opción para realizar control de posición de las articulaciones del manipulador robótico CRS A255, debido a que la falta de información sobre al peso de cada uno de sus componentes, imposibilita un análisis dinámico.
- Al implementar el controlador difuso las articulaciones no llegaban a estabilizarse en la posición deseada, dando oscilaciones en torno a la misma, esto se resolvió, gracias a la optimización de los frenos, haciendo que los mismos se activen al llegar al valor requerido, obteniendo un error de ± 5 grados aproximadamente.
- El retraso que se notó en el procesamiento de las tarjetas Teensy 3.6 con respecto al computador encargado del control es de 9,5 milisegundos en promedio en cada ciclo, esto se ve reflejado en el segundo de retraso que le toma al robot_1 imitar al robot_2.
- No es factible la creación de encoders incrementales que cumplan con las características requeridas para este proyecto, debido a que dentro del país no se cuenta la tecnología necesaria a costos accesibles que lleve a cabo esta tarea.

- El plano eléctrico creado en este proyecto, es una herramienta útil al momento de realizar evaluación de estado, verificación de fallos y reemplazo de componentes eléctricos y electrónicos en los manipuladores robóticos CRS A255 y en las tarjetas de control y potencia.
- La utilización de una estructura ROS permite que la comunicación entre el circuito de control y potencia, el controlador y el entorno de simulación, se pueda implementar de forma fácil y comprensible.
- Haciendo uso del software V-REP, se logró integrar el funcionamiento de los manipuladores robóticos CRS A255 reales, con los robots ABB IRB 140 del entorno de simulación, haciendo modificaciones en su script y adaptándolo a las condiciones específicas de este proyecto, eliminando así la necesidad de modelar desde cero un modelo CAD robótico.
- El análisis de hardware se llevó a cabo únicamente de las partes eléctricas y electrónicas, mientras que las partes mecánicas como cadenas, engranes, piñones entre otras, deben ser evaluadas por personal calificado para ello.
- El uso de software libre como Ubuntu, Arduino, ROS y V-REP, permitió el desarrollo de este proyecto, ya que admiten el acceso a su código fuente, para modificarlo de acuerdo a las necesidades específicas de cada aplicación, sin requerir la comprar de licencias.

6.2. Recomendaciones

- En proyectos futuros se recomienda utilizar como parte central del circuito de control y potencia, una matriz de puertas programables (FPGA por sus siglas en inglés), con la cual se puede realizar el control de cada articulación de forma paralela y no secuencialmente como se lo realiza actualmente.

- Se puede mejorar el controlador difuso implementado, tomando en cuenta la dinámica del robot, y así lograr una mayor precisión en sus movimientos.
- Se recomienda dotar de soportes de ajuste para los conectores SPEC-MIL diseñados, o en su defecto comprar los originales, para no tener problemas en la conexión entre el circuito de control y potencia con el manipulador robótico CRS A255.
- Se recomienda hacer uso de encoders del mismo tipo para todas las articulaciones del manipulador robótico CRS A255, y así no tener lecturas erróneas el momento de entrar en funcionamiento.
- Realizar el análisis, mantenimiento preventivo y correctivo de la parte mecánica de los manipuladores robóticos CRS A255, para estar seguros de que sus elementos (cajas de reducción, cadenas, engranes, piñones, soportes, junturas), se encuentran en un buen estado.

REFERENCIAS BIBLIOGRÁFICAS

Barrios Gutierrez, E. F., Bernal Castillo, F. D., & Tejada Ome, C. A. (2011). *Diseño De Sistema Electrónico Para Manejo De Señales Digitales Para Control De Articulaciones De Robot Movemaster Rv M1 De Mitsubishi*. Bogotá, Colombia.

Bauzano, E., Fernandez/Iribar, A., López, M., Renteria, A., & Muñoz, V. (04 de Septiembre de 2015). *Integración De Dispositivos En Un Robot Quirúrgico*. Comité Español de Automática de la IFAC (CEA-IFAC), 815-822.

Calvo, C. M. (Enero de 2006). *Estudio de factibilidad de la repotenciacion de una planta de generacion electrica de turbinas de combustion*. Caracas, Venezuela.

Castañeda, C. C. (2015). *ROS-Gazebo. una Valiosa Herr amienta de Vanguardia para el Desarrollo de la Robótica*. Tunja, Colombia.

Chimarro Amaguaña, J., & Enríquez Herrera, A. (2015). *Diseño, Construcción E Implementación De Un Robot Esférico De 4 Grados De Libertad Para Manipulación De Objetos Utilizando La Plataforma Robotic Operating System (Ros)*. Sangolquí, Ecuador.

Coque, D., & Casa, D. (Marzo del 2014). *Desarrollo de software para la programacion y operacion del manipulador robotico CRSA255*. Sangolquí, Ecuador.

Corporation, C. R. (2000). *A255 Robot Arm User Guide.* Recuperado el 08 de Junio del 2017, de https://www.eecs.yorku.ca/course_archive/2011-2/W/4421/doc/a255/A255UserGuide

Davis, K., & Bankieris, D. (8 de agosto de 2016). *ROS Hexápodo*. Houston, Estados Unidos. Recuperado el 24 de Febrero del 2018, de https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160011282.pdf

Escudero, J. L. (2016). *Análisis de un brazo robotico con Gazebo y ROS para tareas de inspeccion remota en el CERN*. Madrid, España.

Ferguson, M. (2015). *ROS*. Recuperado el 19 de Agosto de 2018, de http://www.ros.org/

Garcia, A. (2016). *Diseño, Construcción Y Control De Un Robot Manipulador De 3 Grados De Libertad De Bajo Coste Para El Desarrollo De Un Manipulador Móvil*. Valencia, España.

Garzón Jaramillo, P., & Obando Maldonado, C. (2016). *Desarrollo De Un Sistema De Ubicación En La Plataforma Robótica Para Exteriores Teleoperada Del Laboratorio De Manufactura*. Sangolquí, Ecuador.

Gomez Tejada, V. (2015). *Operación de remachado mediante robot manipulador industrial basado en ROS.* Sevilla, España.

González Díaz, A. F., & Jiménez Reyes, D. A. (2014). *Diseño E Implementación De Tarjetas De Potencia Para El Control De Movimiento Del Robot Movemaster Rv-M1 De Mitsubishi*. Bogotá, Colombia.

Haegele, M. (Octubre de 2016). *Executive Summary World Robotics 2016 Service Robots*. Publication of World Robotics 2016 – Service Robots. Frankfurt , Alemania.

Herrera, W. (2011). *Diseño del control para el posicionamiento de un servomecanismo cartesiano de tres ejes.* Nueva Guatemala de la Asuncion, Guatemala.

Kouro samir, M. R. (2010). *Control mediante logica difusa*. Técnicas modernas de automática,1-7.Valparaiso, Chile.

Maldonado Daniel, S. A. (2013). *Estudio diseño e implementación de un controlador (hardware y software) para tres grados de libertad del manipulador robótico CRS-A255.* Sangolquí, Ecuador.

Mejía Silva, M., & Núñez Arroba, J. (2016). *Diseño Y Construcción De Un Robot Social Para Interacción Hombre Máquina*. Sangolquí, Ecuador.

Ochoa, M., Aguiar, G., & Erazo, A. (2016). *RHINO–an autonomous interactive surveillance robot for the needed ones: design and study case*. MATEC Web Conferences 56, ICCAE 2016. Recuperado el 16 de Noviembre del 2018, de https://doi.org/10.1051/matecconf/ 20165605607003

Rapado, J. (2016). *Diseño e implementación de una interfaz gráfica de usuario para mapeado de entornos y navegación en ROS*. Valencia, España.

Rivera, P. A. (Julio de 2015). *Diseño E Implementación De Técnicas De Control Para El Controlador CAD Del Manipulador Robotico Crs A255.1*. Sangolquí, Ecuador.

Sandoval Daniela, T. A. (2016). *Sistema replicador de movimiento articular de estremidad superior derecha en brazo robótico industrial por estudio electromiográfico y uso de kinect.* Sangolquí, Ecuador.

Stoffregen Paul, C. R. (2015). *PJRC Electronic Projects Componet Available Worldwide*. Recuperado el 19 de Junio del 2018, de https://www.pjrc.com

Tumbaco Mendoza, D., & Quimbita Zapata, W. (2014). *Diseño Y Construcción De Un Prototipo De Robot Delta Con Implementación De Un Cortador Láser Cnc Utilizando La Plataforma Robotic Operating System (Ros) Para La Elaboración De Artículos Publicitarios*. Latacunga, Ecuador.