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: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CLOUD ROBOTICS PARA EL CONTROL CINEMÁTICO DE UNA CÉLULA ROBOTIZADA CONFORMADA POR DOS ROBOTS CRS A255

AUTORES: CORONEL QUINALUISA, JOEL ALEXANDER
PAREDES GONZÁLEZ, DIANA PAMELA

DIRECTOR: ING. TIPÁN CONDOLO, EDGAR FERNANDO, M.Sc.

SANGOLQUÍ
2019

i

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, " ***DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CLOUD ROBOTICS PARA EL CONTROL CINEMÁTICO DE UNA CÉLULA ROBOTIZADA CONFORMADA POR DOS ROBOTS CRS A255*** ", fue realizado por los señores ***CORONEL QUINALUISA, JOEL ALEXANDER***, y ***PAREDES GONZÁLEZ, DIANA PAMELA*** 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, 25 Noviembre del 2019

Ing. Edgar Fernando Tipán Condolo, M.Sc.

C.C. 1711391316

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Nosotros, ***CORONEL QUINALUISA, JOEL ALEXANDER, y PAREDES GONZÁLEZ, DIANA PAMELA***, declaramos que el contenido, ideas y criterios del trabajo de titulación: " ***DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CLOUD ROBOTICS PARA EL CONTROL CINEMÁTICO DE UNA CÉLULA ROBOTIZADA CONFORMADA POR DOS ROBOTS CRS A255***", es de nuestra 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, 25 Noviembre del 2019

Joel Alexander Coronel Quinaluisa
C.C. 1723838056

Diana Pamela Paredes González
C.C. 1723093124

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Nosotros, ***CORONEL QUINALUISA, JOEL ALEXANDER,*** *y* ***PAREDES GONZÁLEZ, DIANA PAMELA***, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación " *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CLOUD ROBOTICS PARA EL CONTROL CINEMÁTICO DE UNA CÉLULA ROBOTIZADA CONFORMADA POR DOS ROBOTS CRS A255*" en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 25 Noviembre del 2019

Joel Alexander Coronel Quinaluisa
C.C. 1723838056

Diana Pamela Paredes González
C.C. 1723093124

DEDICATORIA

La culminación de este proyecto se la dedicamos a nuestras familias quienes son las personas más importantes de nuestra vida, queremos agradecer a las familias Coronel y Paredes, ya que sin ellos no se hubiese podido culminar este Proyecto, ni cumplir varias de nuestras metas académicas.

Joel y Diana

AGRADECIMIENTO

Este trabajo agradecemos a cada una de las personas que estuvieron presentes de una manera u otra, en todo el transcurso del desarrollo de este proyecto con sus consejos, motivación y palabras de aliento, que nos motivaron a sobrellevar las diferentes adversidades que se presentaron en la elaboración del mismo.

Joel y Diana

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo es la creación de una celda robótica conformada por dos CRS A255, los mismos que cumplirán con la ejecución de su cinemática directa e inversa, al igual que permitirá la simulación de tres procesos industriales; el eje central de este proyecto es la operación remota del controlador ATxmega64D3 mediante una página web, la misma que será montada en un servidor web que en este caso es la tarjeta Raspberry Pi. Dicha página web contará con el ingreso de datos para la ejecución de la cinemática tanto inversa como directa, al igual que permitirá un registro de usuarios que posteriormente será almacenado en una base de datos; los gráficos históricos de la ejecución de la cinemática también podrán ser visualizados de forma dinámica, dichos gráficos pueden ser descargados en formatos xls, jpg, png, entre otros; en cuanto a los procesos industriales se podrán ejecutar de forma fácil, de igual forma existe el monitoreo de la celda de trabajo mediante una cámara. La página web fue desarrollada en el framework Django que usa el lenguaje de programación Python. Para este proyecto se utilizó Python 2.7 ya que permite la comunicación serial de la Raspberry Pi con el controlador ATxmega64D3. En cuanto al control que se desarrolló en este proyecto fue un control fuzzy el mismo que brindó un porcentaje de error de menos del 2% en la ejecución de la cinemática tanto inversa como directa al igual que en el desarrollo de los procesos.

PALABRAS CLAVE:

- **MANIPULADOR ROBÓTICO CRS A255**
- **DJANGO**
- **PYTHON**
- **ATXMEGA64D3**

ABSTRACT

The present work is the creation of a robotic cell made up of two CRS A255, which will comply with the execution of its direct and inverse kinematics, and the simulation of three industrial processes; the central axis of this project is the remote operation of the ATxmega64D3 controller through a web page, this page will be mounted on a web server, in this case a Raspberry Pi card was used. This web page will have the input of data for the execution of inverse and direct kinematics, as well as allowing a user registration that will later be stored in a database; the historical graphics of the execution of the kinematics can also be displayed dynamically, furthermore these graphics can be downloaded in xls, jpg, png formats, among others; as for the industrial processes, they can be executed easily, in the same way there is the monitoring of the work cell through of a camera. The website was developed in the Django framework that uses the Python programming language. Python 2.7 was used for this project because it allows serial communication of the Raspberry Pi with the ATxmega64D3 controller. As for the control that was developed in this project, it was a fuzzy control that provided an error rate of less than 2% in the execution of inverse and direct kinematics like in the development of the processes.

KEYWORDS:

- **CRS A255 ROBOTIC MANIPULATOR**
- **DJANGO**
- **PYTHON**
- **ATXMEGA64D3**

CAPÍTULO I

INTRODUCCIÓN

1.1. Antecedentes

Por el año de 1955, John McCarthy crea el término “Inteligencia artificial” incursionando con su teoría de tiempo compartido, similar a la nube que se conoce hoy en día; este concepto tiene como base el compartir recursos computacionales entre varios usuarios. En la década de los 80 salen al mercado los primeros computadores personales, el uso de los mismos se dispara en los siguientes años, hasta que en el año de 1996 George Favaloro y Sean O ‘Sullivan usan el término de “Cloud Computing” para poder relacionar los servicios que se pueden ofrecer a través de internet con el uso de computadores. (Cruz, 2015)

El modelo Cloud Computing se sustenta en tres pilares fundamentales como lo son:

- Software como un servicio (Software as a Service, SaaS): El proveedor brinda aplicaciones que son utilizadas por los consumidores finales.
- Infraestructura como un servicio (Infraestructure as a Service, IaaS): El proveedor brinda el hardware en el cual el usuario deberá colocar las aplicaciones que requiere para poder utilizarlo.
- Plataforma como un servicio (Plataform as a Service, PaaS): El proveedor brindará utilerías sobre las que se construyen aplicaciones.
- Estos pilares constituyen la arquitectura que todo Cloud debe poseer.

  (Silva, 2015) (Torres, 2016)

El concepto de Cloud va de la mano con la conexión de varios dispositivos al internet, por lo que se han presentado varios modelos que pueden ser implementados en las diversas empresas y organizaciones, estos modelos son:

- Nube Privada: El usuario gestiona sus servicios en la nube, manteniendo el control, se centralizan los recursos informáticos lo que brinda flexibilidad en la disponibilidad de los mismos.
- Nube Pública: El proveedor de servicios en la nube proporciona sus recursos de forma abierta a diversos usuarios; lo cual hace que en gestión de seguridad se tenga un control físico y lógico muy bajo.
- Nube Hibrida: Combina servicios que se ofrecen de forma pública y de forma privada. (Torres, 2016)

El uso de cada una de estas nubes variará de acuerdo con la aplicación en la que serán implementadas; ya que si se necesita mayor seguridad en el control de la información y un mayor volumen de almacenamiento de la misma es necesario que las entidades inviertan en contratar una nube privada con la finalidad de mantener a salvo su información ante posible intromisiones; pero si en la aplicación a implementar no es necesario guardar gran cantidad de datos y la seguridad no debe ser extrema es recomendable usar nubes públicas ya que trabajarán de la misma forma. El Cloud con los años se ha ido solidificando en el mercado; ya que se ha vuelto más rápido, eficaz y seguro es así que en el nuevo milenio debido a que varias empresas se ven en la necesidad de evaluar cómo usar el internet para brindar servicios; proporcionando de esta manera las condiciones perfectas para que la computación en la nube prospere, es así que nace lo que se conoce como "Cloud Robotics" que es un campo de la robótica que utiliza las tecnologías que le brinda la

computación en la nube para que los robots puedan poseer bastos recursos de computación, almacenamiento, comunicación y procesamiento. (Civera, 2015)

La industria 4.0 se la conoció por primera vez en Alemania, en la cual se pretende fomentar la eficiencia operativa y aumento de la producción industrial, fomentando la interconexión inteligente de la maquinaria y realizando estaciones totalmente productivas, es decir se estaría entrando a la nueva revolución industrial. (Semle, 2016)

Con la presencia de la nueva revolución industrial (Industria 4.0), en el mercado empresarial, las empresas se ven en la necesidad de trabajar con el Cloud, con la finalidad de obtener datos de manera más rápida y mejorar la productividad. Países como Alemania buscan mejorar la productividad de sus fábricas con la implementación de IoT y el desarrollo de computación en la nube, con el fin de desarrollar sistemas inteligentes que permitan interconectar estaciones de producción y llevar a la industria a un nivel más futurista. (Lee, 2014)

En la actualidad, no es común encontrar un artículo científico, tesis o paper sobre nuevas tecnologías o transformación digital que no incluya alguna mención sobre el IoT, ya sea ésta de forma directa o tangencial. Y es que es un hecho que la capacidad de poder conectar cualquier objeto y manejo de información es un concepto que despierta de forma inusual la imaginación.

Actualmente se han realizado diferentes trabajos de titulación, que utilizan IoT hacia dispositivos del hogar, educativos o aparatos dentro del ámbito industrial. Como base para el desarrollo de este proyecto, primero se elegirá la plataforma cloud, el protocolo de comunicación, el lenguaje de programación, el entorno de programación y finalmente el controlador óptimo y con prestaciones adecuadas al proyecto.

A continuación, se presentan trabajos realizados sobre el manipulador robótico CRS-A255, que ayudarán como base investigativa para el desarrollo del presente proyecto, en donde se pretende

mostrar un enfoque de la industria 4.0, para lo cual se detalla el tipo de controladores que se han ido implementado en estos manipuladores.

S. Minango presentó un artículo titulado “Control de Movimiento del Manipulador CRS-A255”, proponiendo que el control se realice mediante la implementación de algoritmos de visión artificial como la esqueletización de Zhang-Suen, permitiendo al manipulador imitar el movimiento de un brazo humano. La técnica de esqueletización permite reducir una imagen a un conjunto de líneas, y de esta forma no afectar al posicionamiento de las articulaciones y mantener una alta velocidad de procesamiento. (Zhang, 2014).

La implementación de un control por esqueletización brindó resultados favorables, puesto que se pudieron controlar cuatro de las cinco articulaciones del manipulador, teniendo una limitante en la velocidad de trabajo, ya que la comunicación entre el operador y el manipulador sufre un retardo al momento de realizar el movimiento (Minango, 2012).

M. García presentó una tesis “Diseño del sistema de control del brazo robótico CRS A255 utilizando la plataforma KINETIX de Allen Bradley”, proponiendo la sustitución del controlador C500 por la plataforma KINETIX de Allen Bradley, el mismo que permite que el robot se comunique mediante una red Ethernet/IP tipo anillo a otros manipuladores y así poder trabajar con el mismo, sin perder su funcionalidad (M.R, 2010).

D. Maldonado presentó un artículo titulado “Estudio, Diseño e Implementación de un Controlador (Hardware y Software) para Tres Grados de Libertad del Manipulador Robótico CRS-A255”, el cual basa el control en microcontroladores que funcionan como “Maestro-Esclavo “para poder obtener una vía de enlace entre el operador y el manipulador, esta red permite que las señales enviadas por la interfaz gráfica actúen como maestro hacia las señales de los motores que son los esclavos (Maldonado, 2013).

D. Casa presentó un artículo titulado “Desarrollo de Software para la Programación y Operación del Manipulador Robótico CRS A255” el cual mejora el controlador de Maestro-Esclavo y hace operables las otras dos articulaciones del manipulador con la implementación de una comunicación I2C, para mejorar la velocidad y respuesta de las cinco articulaciones que posee el robot y de esta forma poder trabajar en el desarrollo de trayectorias (Casa, 2014).

Wang S., Zhao H. y Hao X. realizaron un diseño de un robot de limpieza inteligente basado en IoT, las personas pueden operarlo de forma remota. El proceso básico de trabajo del robot es el siguiente. En primer lugar, el teléfono establece una red para comunicarse con el robot, de modo que el robot se agrega a la red interna. Después de agregar a la red, el teléfono se puede usar para controlar el robot mediante la aplicación. El operador usa la función de video en tiempo real de la aplicación para controlar que el robot se mueva en la casa; el ambiente interior se muestra en la pantalla del teléfono. El sistema de control de movimiento utiliza el último procesador Cortex-M4 como núcleo de control de hardware subyacente del robot. El sistema incluye sensores, comunicación inalámbrica, retroalimentación del motor y algoritmo PID. Además, posee un sensor de gas. El robot puede obedecer las teclas de control remoto por infrarrojos, por lo que el operador usa APP para ordenar al robot que emita la información de codificación infrarroja de aprendizaje, logrando la función de controlar los aparatos de la casa (Shuzhou Wang, 2015).

Li X., Zhang Y., Zhao H., Ding X. y Sun Z. diseñaron una videovigilancia en tiempo real y un robot móvil con tecnología IoT utilizando Raspberry Pi, como CPU integrada, combinada con el protocolo 802.11g y TCP/IP; empleo HTTP para realizar la transmisión de señal de distancia infinita y adoptó el esquema de codificación de video H264 para el monitoreo en tiempo real, para su transmisión de video utilizó RTP(Protocolo de transporte de tiempo real) / RTCP (Protocolo de control de RTP) y una arquitectura C/S (Cliente servidor) y transmisión B/S (bytes por segundo),

para el diseño de la base de datos para finalmente emplear el protocolo SSH (Protocolo de acceso remoto) garantizando el control remoto y la fiabilidad de seguridad del robot (Tucuru, 2012).

1.2. Justificación e importancia

Los controladores actuales presentes en los dos CRS A255 que se encuentran funcionales tanto en hardware como en software, están limitados para realizar una movilización de articulaciones independientes, sin un control entre sus motores mucho menos un control cinemático; además la comunicación entre el simulador y los robots es de forma alámbrica lo que no permite que el operario se encuentre fuera del espacio de trabajo de los manipuladores.

El presente proyecto pretende generar una celda robótica que permita trabajar a los dos manipuladores de forma conjunta realizando la simulación de distintos procesos industriales por medio de un control cinemático y una comunicación remota hacia una plataforma Cloud, a través de una página web, que proporcionarán la conexión a diferentes operarios, que pueden estar o no presentes dentro del espacio de trabajo de los robots CRS A255; se otorgará una monitorización de los mismos a través del uso de una base de datos y una cámara, las mismas que estarán integradas en la plataforma Cloud con la finalidad de identificar diferentes aspectos principales de la célula robotizada. Se tomará en cuenta las características actuales de los controladores, para mejorarlos o sustituirlos.

Al culminar este proyecto se dará a conocer nuevos aspectos desarrollados en la revolución industrial 4.0 o industria inteligente con el uso de una plataforma Cloud, la misma que posibilita tener datos en tiempo real, almacenamiento de datos, el uso de un control cinemático con una comunicación remota; contribuyendo así con la materia de robótica industrial dándole un enfoque y un sustento adicional a los conceptos y materiales utilizados actualmente.

1.3. Alcance del proyecto

El proyecto tiene la finalidad de desarrollar una mejora parcial o total a las prestaciones que brinda el controlador actual de cada uno de los robots CRS A255 para la realización de trayectorias y movimientos articulares con los manipuladores robóticos de cinco grados de libertad, los mismos que funcionarán en una celda robótica para simular procesos industriales.

La comunicación entre la celda robótica y los operarios será de forma remota, debido a que su servidor estará conectado a la nube de forma inalámbrica. La tarjeta de control se comunicará con el servidor mediante comunicación UART.

Uno de los aspectos más importantes es la inserción de multimedia en el sitio webs, lo cual permitirá la visualización de la celda robótica, con lo cual se tendrá una monitorización constante del operador hacia la celda robotizada.

El uso de base de datos dentro de la nube es para tener un censado completo de los diferentes actuadores y sensores incorporados en las articulaciones de los dos robots CRS A255. Posteriormente estos datos servirán para realizar gráficos históricos, los mismos que se mostrarán en la interfaz cliente web.

El proyecto incluirá la programación del controlador como se observa en la figura 1, al igual que el desarrollo de una interfaz cliente web que facilitará el control y visualización de las acciones desarrolladas por la celda robótica. Finalmente se desarrollarán pruebas de verificación de la comunicación al igual que el desempeño del controlador a través de una visualización de gráficas estadísticas dentro de la interfaz. A continuación, se presentan los diagramas esquemáticos del funcionamiento del sistema:

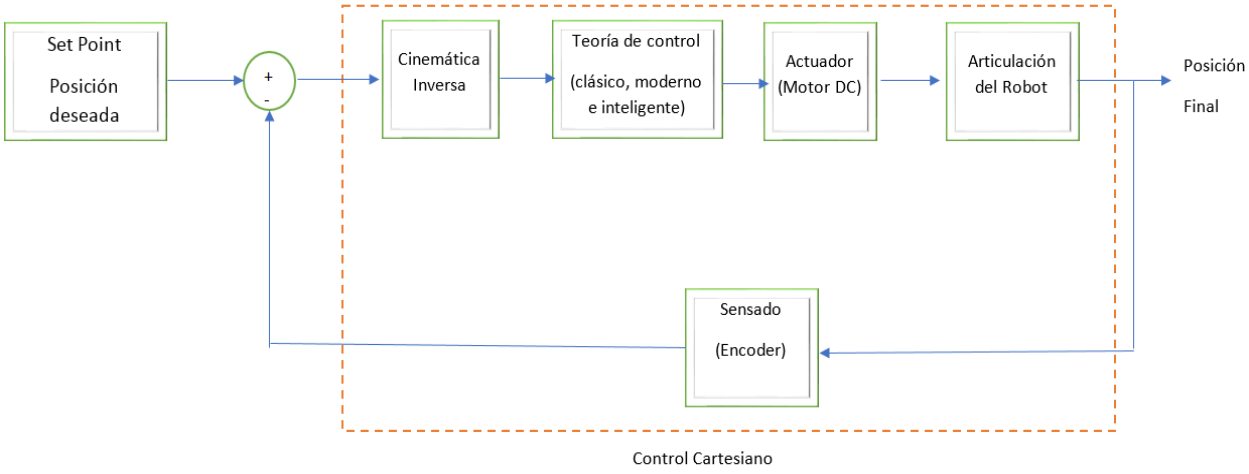

Figura 1. Diagrama de control cinemático de una articulación del manipulador robótico CRS A255

La figura 1 consiste en un control cartesiano de una articulación del manipulador robótico CRS A255, el cual consta de un set point que es la posición deseada que ingresa a un sumador, para luego ingresar a un bloque de cinemática inversa, después se establecerá un control mediante la aplicación de teoría de control, este control puede ser clásico moderno o inteligente; una vez establecido el control de la posición de la articulación del robot, se enviará una señal de activación al actuador que en este caso será un motor DC y a su vez a la articulación del robot; ya establecido el control del actuador del manipulador robótico, se censará mediante el uso de un encoder la posición del motor para la articulación, después del censado se realizará una retroalimentación. Este control será igual para el resto de articulaciones que conforman el manipulador robótico CRS A255

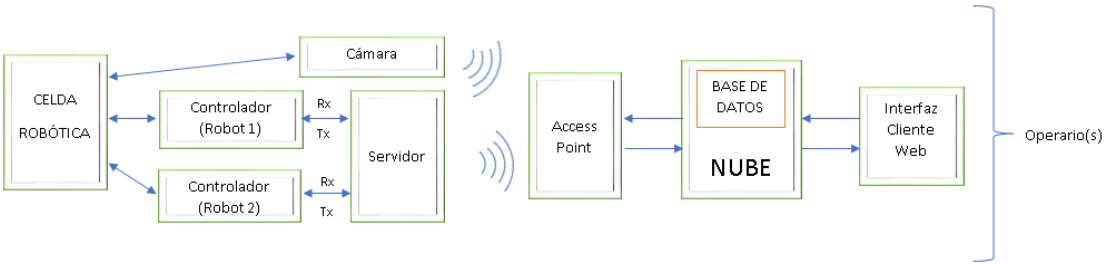

Figura 2. Conexión con plataforma Cloud

La figura 2 se tiene la celda robótica como se muestra en la figura 3; cada uno de los manipuladores robóticos CRS A255 tiene su controlador el mismo que se conectará con comunicación UART hacia el servidor, además de la presencia de una cámara dentro de la celda robótica. Posteriormente se conecta a la Nube en donde se tendrá una plataforma web, donde se pueda acceder y monitorizar la base de datos; teniendo como resultado una interfaz gráfica Cliente Web, a la que cualquier operario se podrá conectar remotamente. Tener en cuenta que la comunicación que se presenta es de tipo Full-Duplex.

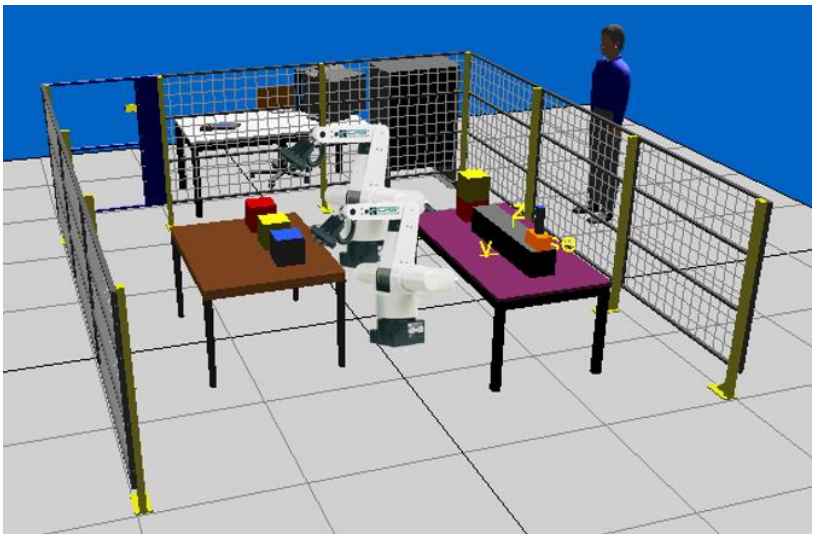

Figura 3. Diagrama Aplicativo de la celda robótica con dos CRS A255

En la figura 3 se puede observar que la célula robotizada consta de dos mesas; en la primera mesa de color café se encuentran tres objetos cuadrados de color rojo, azul y amarillo respectivamente; el primer manipulador robótico CRS A255 se encargará de tomar una caja seleccionada por el operario a través de la interfaz y la llevará hacia la mesa morada, a continuación el segundo robot CRS A255, sujetará otro objeto cuadrado seleccionado nuevamente por el operario a través de la interfaz y lo transportará hacia la mesa morada encima del objeto cuadrado previamente puesto por el otro manipulador robótico, por lo que una caja quedará encima de la otra, para que posteriormente el pistón de color negro despache los dos objetos cuadrados hacia una caja.

La interacción de estos tres esquemas expuestos en las figuras 1, 2 y 3 estarán comandados desde una interfaz de Cliente Web en donde se envían los diferentes movimientos, posiciones o procesos específicos hacia la celda robótica, se podrá observar gráficas estadísticas de los diferentes datos almacenados, permitiendo que el controlador envíe señales de accionamiento a los mecanismos conectados a las diferentes articulaciones de los manipuladores, realizando movimientos precisos para la diferentes tareas que el operario desee que en la celda robótica se lleve a cabo a través de un control cinemático en los robots.

El proyecto a realizar está dividido en seis etapas las cuales se detallan a continuación:

- En la primera etapa del proyecto, se elegirá el controlador adecuado que permita tanto un control rápido y preciso.
- La siguiente etapa del proyecto consiste en la elección y diseño del algoritmo de control (clásico, moderno, inteligente) que permita gobernar los movimientos de los diferentes eslabones de los manipuladores robóticos CRS A255.

- En la tercera etapa la elección de una plataforma Cloud que brinde todas las prestaciones necesarias para la realización del proyecto.
- La próxima etapa consiste en el desarrollo de una interfaz Cliente Web que ofrece la selección de procesos específicos, movimientos individuales de las articulaciones y la posición a diferentes puntos dentro del volumen de trabajo de la célula robótica a través de un control cinemático.
- La penúltima parte del proyecto se concentra en la realización de pruebas de funcionamiento, en la cual se busca medir la precisión y exactitud de respuesta del sistema, permitiendo analizar el desempeño del mismo a través de las gráficas estadísticas de los datos almacenados.
- Finalmente, se presentarán las conclusiones y recomendaciones obtenidas en el desarrollo del presente proyecto.

1.4. Objetivos

1.4.1. General.

Diseñar e implementar un sistema Cloud Robotics para el control cinemático de una célula robotizada conformada por dos robots CRS A255.

1.4.2. Específico.

- Implementar un controlador adecuado que permita realizar trayectorias y movimientos articulares a la celda robótica remotamente.
- Elaborar una interfaz gráfica Cliente Web mediante cuadros de opción, botones de control y plots para observar y manipular el comportamiento de la celda robótica.

- Determinar el control (clásico, moderno, inteligente) necesario, para gobernar los movimientos de los eslabones de los robots CRS A255 de acuerdo a parámetros que el usuario ingrese en la interfaz Cliente Web.
- Elegir una plataforma Cloud que permita el uso de un servidor de base de datos.
- Analizar las diferentes plataformas de Cloud públicas y privadas que se pueden usar.

CAPÍTULO II

ESPECIFICACIONES TÉCNICAS

2.1. Características mecánicas del manipulador robótico CRS A-255

´´El manipulador robótico consta de cinco articulaciones y una herramienta de trabajo conocida como gripper, el manipulador robótico trabaja en un espacio de trabajo de XYZ para desplazarse. En la figura 4 se tiene la distribución de las articulaciones´´. (Corporation, 2000)

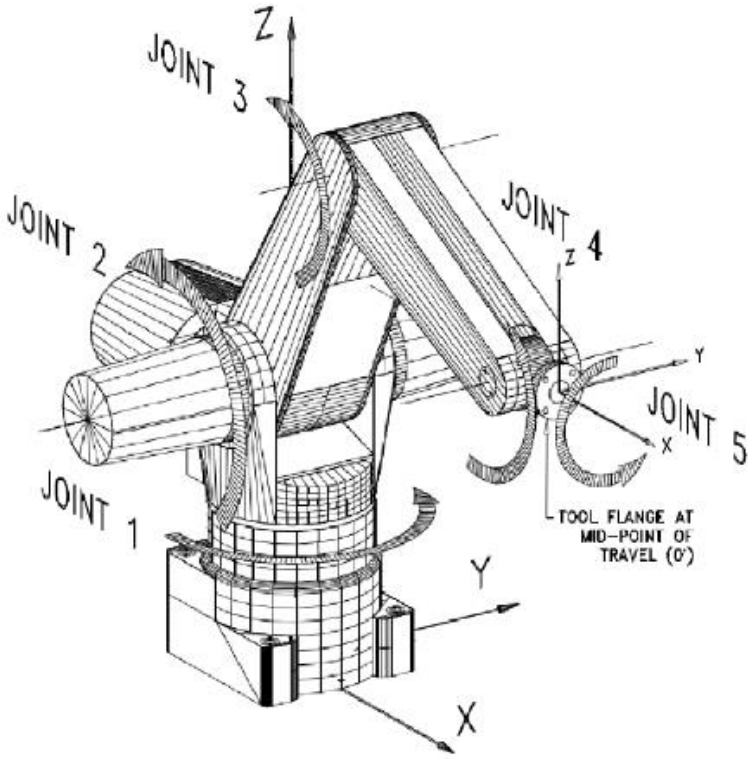

Figura 4. Distribución de articulaciones del manipulador robótico CRS-A255

Fuente: (Corporación, 2000)

Tabla 1

Articulaciones del manipulador robótico

Explicación de las articulaciones del manipulador robótico	
Articulación	**Significado**
Joint 1	Cintura
Joint 2	Hombro
Joint 3	Codo
Joint 4	Muñeca pitch
Joint 5	Muñeca roll

Fuente: (Corporación, 2000)

2.1.1. Rango de desplazamiento, dimensiones y peso.

El desplazamiento, las dimensiones y el peso del manipulador robótico se describirán en la siguiente tabla:

Tabla 2

Desplazamiento, Peso y Dimensiones CRS A255

Desplazamiento, Peso y Dimensiones CRS A255		
Peso	**lb.**	**kg**
Brazo robótico	37	17
Desplazamiento		
Articulación	**Eje**	**Rango de movimiento**
Cintura	1	$+175° \; a \; -175°$
Hombro	2	$+110° \; a \; 0°$
Codo	3	$0° \; a \; -125°$
Muñeca pitch	4	$+110° \; a \; -110°$
Muñeca roll	5	$+180° \; a \; -180°$
Dimensiones		
Sección	**Pulgadas**	**Cm**
Base de montaje superficie a hombre	10	25.4
Hombro a codo	10	25.4
Codo a muñeca roll	10	25.4
Muñeca a brida de la herramienta	2	5.08

Fuente: (Corporación, 2000)

2.1.2. Torque y Carga útil.

Otras características importantes para conocer la capacidad del manipulador robótico son su torque y carga útil, las mismas que se enuncian en la siguiente tabla:

Tabla 3
Torque y carga útil CRS A255

Torque y carga útil		
Articulación	**Eje**	**Torque /N-m**
Cintura	1	6.4
Hombro	2	6.4
Codo	3	6.4
Muñeca pitch	4	1.4
Muñeca roll	5	0.71
Carga útil		
Carga útil	**Velocidad**	**Masa**
Máxima	100%	2 Kg
nominal	80%	1Kg

Fuente: (Corporación, 2000)

2.2. Características eléctricas del manipulador robótico CRS A-255

El manipulador robótico CRS A255 en relación a la parte eléctrica está constituido por motores, frenos y encoders como se puede apreciar en la figura 5:

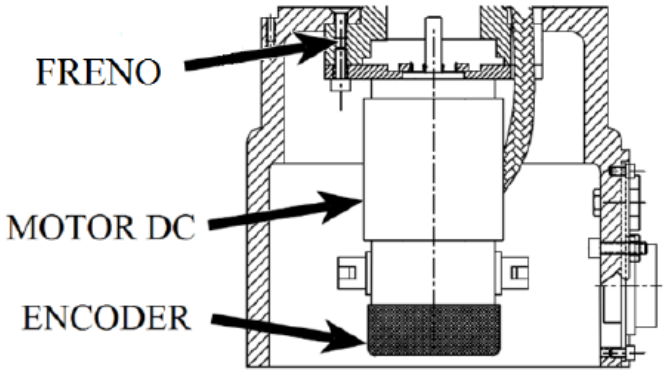

Figura 5. Distribución de freno, motor y encoder del CRS A255

Fuente: (Corporación, 2000)

Esta configuración eléctrica se encuentra presente solo en cuatro de las articulaciones del manipulador robótico, debido a que la cintura no presenta un freno, ya que el efecto de la gravedad no afecta a esta articulación como sucede en las otras cuatro. En la siguiente tabla se describe las características eléctricas de los motores de cada articulación:

Figura 6. Motor en las articulaciones del manipulador robótico

Tabla 4

Características eléctricas de los motores

Características eléctricas de los motores			
Articulación	**Tensión de trabajo**	**I máx. sin carga**	**I máx. con carga**
Cintura	12V - 36V	1.2 A	3.8 A
Hombro	12V - 36V	2.1 A	3.6 A
Codo	12V - 36V	1.3 A	3.6 A
Muñeca pitch	12V - 36V	0.7 A	3.4 A
Muñeca roll	12V - 36V	0.75 A	3.7 A
Especificaciones del motor			
Marca		CMC	
Código		CMC PM FIELD SERVO MOTOR	
Modelo		ME2110-057E	
Serie		1083778	

Fuente: (Corporación, 2000)

Como se observó en la figura 5, la parte eléctrica de los manipuladores robóticos está constituido por encoders y frenos los mismos que se describen a continuación:

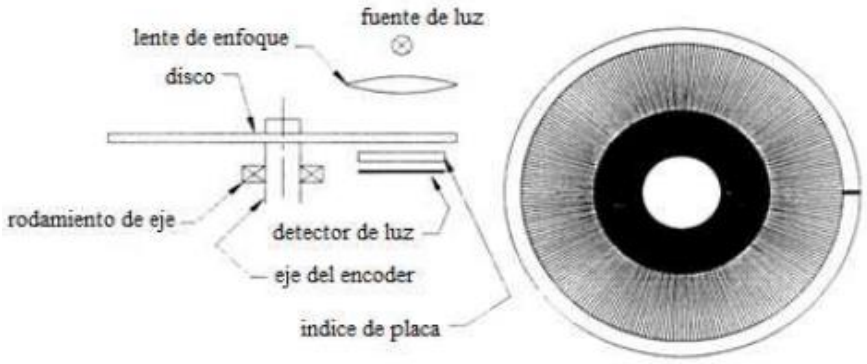

Figura 7. Esquema del encoder manipulador robótico CRS A255

Fuente: (García Saquicela, 2010)

- **Encoders:** los encoder que poseen los manipuladores robóticos son ópticos incrementales o decreméntales, y se encuentran posicionados al final de cada motor. Las características de estos encoder se describen en la siguiente tabla:

Tabla 5
Características de los encoders

Encoder CMS	
Especificación	**Característica**
Resolución	1000 pulsos/rev
Referencia	Marcador 1 pulso/rev
Salida	2 hilos
Resolución articulación	deg

Fuente: (García Saquicela, 2010)

- **Frenos:** los frenos que están instalados en los manipuladores robóticos son power off de disco a prueba de falla. Estos frenos funcionan con resortes de sujeción en un disco rotacional. Las características de los frenos se mencionan en la siguiente tabla:

Tabla 6
Características de los frenos

Frenos CMS	
Especificación	**Característica**
Voltaje	12 VDC
Tipo de solenoide	Magnético
Tipo de freno	Disco rotacional

Fuente: (García Saquicela, 2010)

CAPÍTULO III

ESTADO INICIAL

3.1. Hardware

El proyecto anterior realizó ciertos cambios a los manipuladores robóticos, los cuales se citarán en este capítulo con el fin de detallar en qué estado se encontraban los manipuladores y comprender de mejor manera los cambios que se efectuarán posteriormente para acoplar a las necesidades del proyecto que se desarrollará.

3.1.1. Activación de frenos independientes.

Se realizó un cambio al cableado interno de los manipuladores con relación a los frenos para hacer que trabajen de forma independiente haciendo posible que los motores se puedan activar con una señal PWM, para de esta manera poder efectuar el control ya que en la configuración original los frenos están unidos a la señal de referencia de la articulación a la que pertenecen. La conexión de los frenos quedo de esta forma:

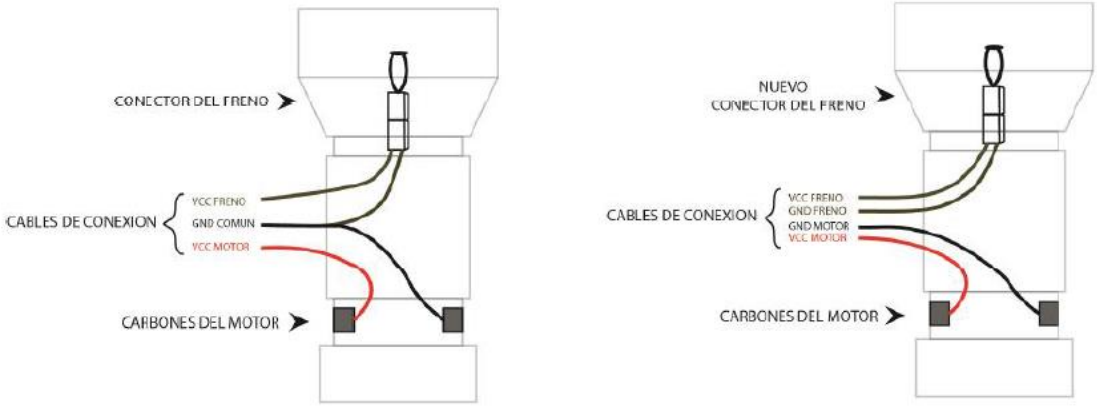

Figura 8. Conexión Original y actual de los frenos

Fuente: ***(Negrete, 2018)***

Como se evidencia en la figura 8 se hizo la separación de los frenos de cada manipulador robótico, haciéndolos independientes y controlables, se destaca también que la fuente de alimentación de los frenos es de 12V la cual debe de estar activa para que los frenos estén deshabilitados y permitan el funcionamiento del motor.

3.2. Encoders

3.2.1. Cambio de encoders

Uno de los manipuladores robóticos necesitó cambiar tres encoders: en la muñeca roll, en el codo y en el hombro, como no se encontró el encoder original del manipulador, se realizó una sustitución del mismo con características similares al original las cuales se cita a continuación:

Figura 9. Encoder Rotary instalado en las articulaciones del codo y hombro

Tabla 7

Características del encoder Rotary

Características del encoder codo/hombro	
Parámetro	**Característica**
PPR	600
Vin	3.3 VCC- 5VCC
Tamaño	38mm X 35.5mm
Velocidad mecánica máxima	5000R/min
Fases de salida	2
Tipo	LPD3806-600BM -05-24C

Figura 10. Encoder OMROM
Instalado en articulación
de la muñeca roll

Tabla 8
Características del encoder OMROM

Características del encoder muñeca roll	
Parámetro	**Característica**
PPR	1000
Vin	3.3 VCC- 5VCC
Tamaño	38mm X 35.5mm
Tipo de encoder	Incremental
Fases de salida	3
Tipo	E6B2-CWZ1X

Los encoder que se utilizaron como reemplazo necesitaron nuevos acoples para poder ser integrados a los motores, haciendo que de esta forma el manipulador robótico adquiera una nueva dimensión y se tenga que elaborar una nueva carcasa. La instalación de los encoders remplazados se pueden apreciar en la siguiente figura.

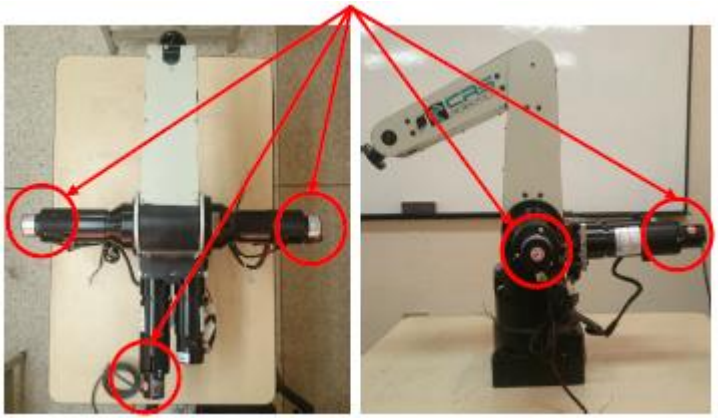

Figura 11. Montaje de encoders nuevos

Fuente: ***(Negrete, 2018)***

3.3. Potencia

3.3.1. Cambio de carcasa.

Como se mencionó en el apartado anterior el cambio de encoders generó nuevas dimensiones al manipulador por lo cual tuvieron que elaborar una nueva carcasa que recubra al manipulador como se observa en la figura 12.

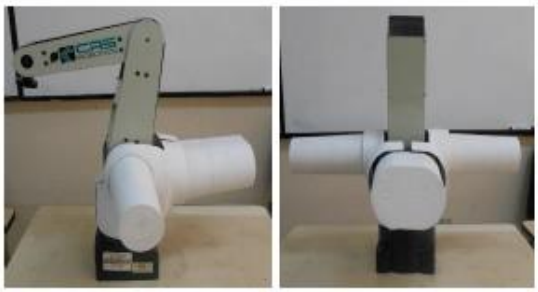

Figura 12. Manipulador robótico CRS A255 estado actual

3.3.2. Etapa de potencia.

Al ser los motores de las articulaciones de corriente continua es necesario la utilización de un puente H, en este proyecto utilizaron el módulo L298N, y como tensión de alimentación se tomó

12V para poder controlar a los motores. En la figura 13 se puede observar el diagrama de conexiones.

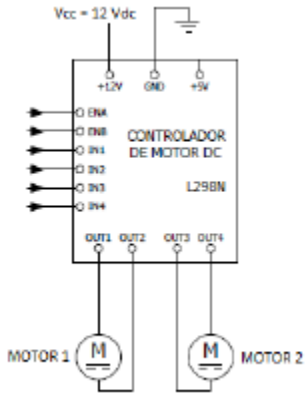

Figura 13. Conexión motores a puente H

Los frenos de cada articulación funcionan a 12VCC, por lo cual se elaboró un circuito de elevación de voltaje ya que la tarjeta de control que se implementó tiene sus pines de salida a una tensión de 3.3 V, es así que se implementó el siguiente circuito:

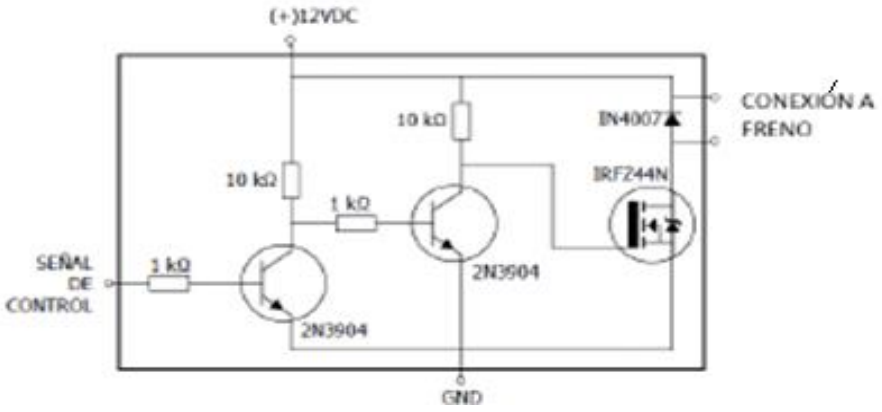

Figura 14. Estado de los frenos al momento de recibir el proyecto

Fuente***: (Negrete, 2018)***

3.3.3. Tarjeta de Control.

El microcontrolador con el que se trabajó en la ejecución de este proyecto fue la teensy 3.6 para cada manipulador utilizaron esta tarjeta por la facilidad de leer los encoders, ya que trabaja hasta con 10 interrupciones y cuenta con los pines PWM necesarios para desarrollar el control.

3.3.4. Implementación de circuitos.

Con los diseños efectuados tanto en la parte de potencia como de control se obtuvo la siguiente placa:

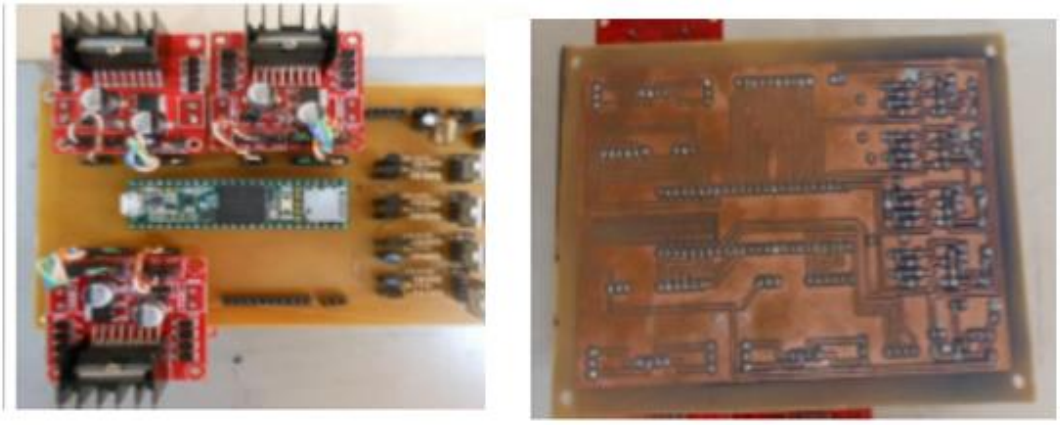

Figura 15. Placa de control y potencia

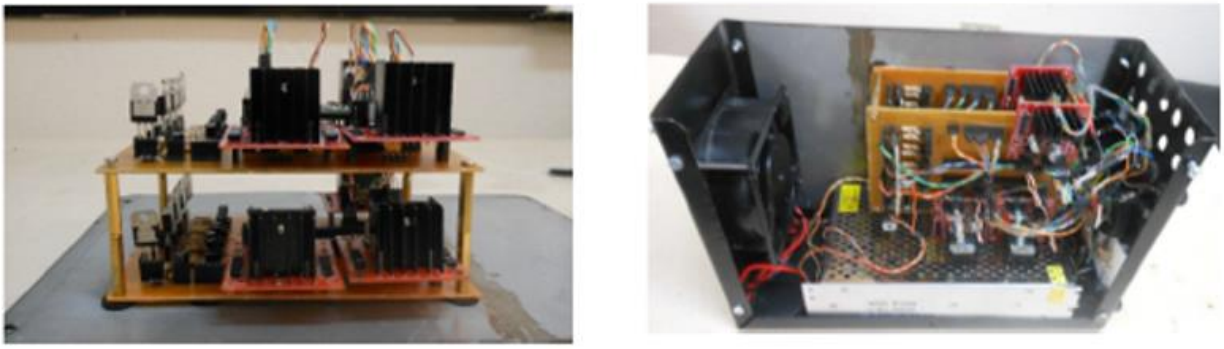

Figura 16. Circuitos de potencia y control dentro de la carcasa

3.3.5. Diseño de conectores.

La caja de control alberga la conexión de dos manipuladores robóticos, pero uno de ellos no dispone de conectores SPEC-MIL, es así que en el proyecto anterior se vieron en la necesidad de diseñar dichos conectores obteniendo el siguiente resultado.

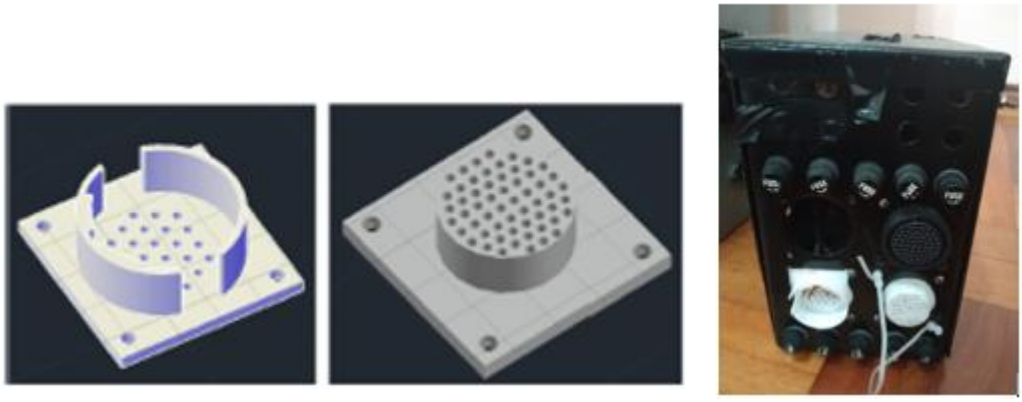

Figura 17. Diseño e implementación de conectores

3.4. Software

El sistema con el que trabajó este proyecto es una estructura ROS, la cual se ejecuta desde un PC en sistema operativo Ubuntu y un entorno de simulación para robot conocido como ViRed, el cual permite el monitoreo de los manipuladores robóticos desde el computador.

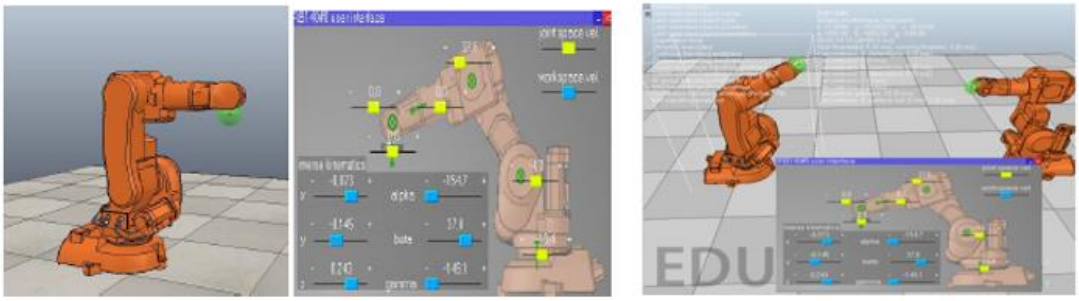

Figura 18. Interfaz de visualización del control de los manipuladores robóticos CRS A255 en ViRed

3.5. Problemas encontrados al momento de recibir los manipuladores robóticos

En esta sección se describirán problemas que se encontraron al momento de tomar el proyecto; los problemas abarcan problemas mecánicos y eléctricos haciendo que el alcance planteado desde un inicio fuera modificado a la realidad en la que se encontraban los manipuladores. Dichas consideraciones se describirán a continuación:

3.5.1. Encoders.

Cómo se describió en el ítem 3.2 un manipulador tubo cambios en sus encoders y por ende en sus acoplamientos, al momento de probar la funcionalidad de dicho manipulador se evidenció problemas de lectura de los encoders al momento de mover las articulaciones; en la revisión se descartó daños de los encoders pero si problemas en los acoplamientos debido a que los mismos son de plástico al ser impresos en 3D; debido a que es un manipulador industrial dichos acoplamientos deben ser diseñados con materiales más resistentes, ya que en la actualidad presentan un desgaste que imposibilita la lectura de los encoders; en este caso es recomendable hacer un estudio completo de que tipo y forma de acople permite una buena integración del motor con el encoder, siendo esta actividad fuera de los puntos previstos a desarrollarse dentro de este proyecto se descartó su arreglo; haciendo que se vea modificado el proceso industrial que se pensaba efectuar desde un inicio ya que solo se contará con un manipulador en óptimas condiciones aclarando que en lo que compete a la parte eléctrica y electrónica se sigue considerando para dos manipuladores.

3.5.2. Motores.

Los motores en ambos manipuladores se encuentran operativos; sin embargo, en el manipulador que no presentaba cambios se encontró que el motor de la muñeca pitch carecía de fuerza por lo

que limitaba el movimiento de la misma, al presentarse este inconveniente se realizó un cambio del motor de la muñeca pitch con el de la muñeca roll, dejando de esta forma al manipulador completamente operativo.

3.5.3. Software.

El software ViReD no fue contemplado en el diseño de este proyecto por eso se plantea un diseño nuevo para la monitorización de los manipuladores, el mismo que se describe en capítulos posteriores.

3.5.4. Controlador.

Cada manipulador trabajaba con una tarjeta controladora que era una Teensy 3.6 que cumplía con parte de los requerimientos para este proyecto, sin embargo, carecía de la cantidad necesaria de interrupciones para la lectura de los encoders, además de dificultar la comunicación serial con la Raspberry Pi debido a que la lectura solo se podía hacer carácter por carácter, por lo que se la descartó como controlador para el presente proyecto.

Si bien los problemas encontrados en los manipuladores limitaron de cierta forma el desarrollo del proyecto se dio alternativas y soluciones para cumplir con los objetivos planteados.

CAPÍTULO IV

REDISEÑO

4.1. Caja de control

El diseño de la caja de control para la celda robótica, está compuesta por los ítems que se tratarán a continuación:

4.1.1. Hardware.

Todos los elementos de control y de potencia se encuentran instaurados dentro de un armario metálico, que está recubierto por pintura electrostática, que actuará como aislante, dicho armario tiene los siguientes elementos:

- Alimentación
- Encendido general
- Encendido del manipulador robótico 1
- Encendido del manipulador robótico 2
- Paro de emergencia
- Conectores hacia los robots

En los siguientes apartados se dará una explicación detallada de cada elemento que conforma el armario de control de esta celda robótica.

Dentro de dicho armario se encuentra tanto la parte de potencia como de control, para lo cual se efectuó un esquemático de cómo se encuentran distribuidos los elementos y la alimentación de los mismos. (Anexo1).

4.1.2. Cableado.

Para facilidad del control en el cableado existente dentro del armario se estableció las siguientes directrices:

Tabla 9
Distribución de la alimentación de 24V

Fuente de 24 V		
Elemento	**Color**	**Código/Etiqueta**
VCC	Amarillo	+
GND	Azul	-

Tabla 10
Distribución de la alimentación de 12V

Fuente de 12 V		
Elemento	**Color**	**Código/ Etiqueta**
VCC	Amarillo	+.
GND	Rojo	-.

Tabla 11
Distribución de la alimentación de 5V – Raspberry Pi

Fuente de 5V Raspberry Pi		
Elemento	**Color**	**Código/Etiqueta**
VCC	Blanco	++
GND	Blanco	--

Tabla 12
Distribución de la alimentación de 5V – Tarjeta de frenos y encoders

Fuente de 5V Tarjetas para Frenos		
Elemento	**Color**	**Código/ Etiqueta**
VCC	Blanco	+/
GND	Blanco	-/

Tabla 13

Alimentación tarjetas de potencia Robot1

Fuente de 5V Tarjetas de potencia Robot 1		
Elemento	**Color**	**Código/ Etiqueta**
VCC	Blanco	A
GND	Blanco	B
Cables que ingresan a las tarjetas potencia Robot 1		
Elemento	**Color**	**Número de tarjeta**
VCC	Blanco	1
GND	Negro	
VCC	Azul	2
GND	Verde	
VCC	Morado	3
GND	Plomo	

CONTINÚA

Tabla 14

Alimentación tarjetas de potencia Robot2

Fuente de 5V Tarjetas de potencia Robot 2		
Elemento	**Color**	**Código/ Etiqueta**
VCC	Rojo	A
GND	Negro	B
Cables que ingresan a las tarjetas		
Elemento	**Color**	**Número de tarjeta**
VCC	Blanco	4
GND	Negro	
VCC	Azul	5
GND	Verde	
VCC	Morado	6
GND	Plomo	

Tabla 15

Distribución de pines de tarjetas de potencia hacia la tarjeta de control Robot1

Distribución de pines de tarjetas de potencia hacia la tarjeta de control Robot1			
Elemento	**Color**	**Código**	**Pin/ Tarjeta de control**
Tarjeta 1			
PW1	Tomate	E	P1
DIR1	Amarillo	F	P7
PW2	Café	C	P2
DIR2	Rojo	D	P8
Tarjeta2			
PW1	Café	I	P3
DIR1	Rojo	J	P9
PW2	Blanco	G	P4
DIR2	Negro	H	P10
Tarjeta3			
PW1	Verde	M	P5
DIR1	Azul	N	P11
PW2	Tomate	K	P6
DIR2	Amarillo	L	015

Tabla 16

Distribución de pines de tarjetas de potencia hacia la tarjeta de control Robot2

Distribución de pines de tarjetas de potencia hacia la tarjeta de control Robot2			
Elemento	**Color**	**Código/ Etiqueta**	**Pin/ Tarjeta de control**
Tarjeta 4			
PW1	Morado	Q	P1
DIR1	Plomo	R	P7
PW2	Verde	O	P2
DIR2	Azul	P	P8
Tarjeta 5			
PW1	Rojo	U	P3
DIR1	Café	V	P9
PW2	Amarillo	S	P4
DIR2	Tomate	T	P10
Tarjeta 6			
PW1	Azul	Y	P5
DIR1	Verde	Z	P11
PW2	Plomo	X	P6
DIR2	Morado	W	O15

CONTINÚA

Tabla 17

Distribución de pines de los motores con las tarjetas de potencia/ Robot1

Distribución de pines de los motores con las tarjetas de potencia/ Robot1			
Elemento	**Código**	**Color**	**Tarjeta**
Motor cintura (GND)	LM	Negro	Tarjeta 1
Motor cintura (VCC)	PQ	Blanco	
Motor hombro (GND)	NO	Amarillo	
Motor hombro (VCC)	VW	Tomate	
Motor codo (GND)	RS	Café	Tarjeta 2
Motor codo (VCC)	ZA	Plomo	
Motor muñeca pitch (GND)	TU	Rojo	
Motor muñeca pitch (VCC)	BC	Verde	
Motor muñeca roll (GND)	XY	Azul	Tarjeta 3
Motor muñeca roll (VCC)	DE	Blanco	
Motor gripper (GND)	GGD	Azul	
Motor gripper (VCC)	GGV	Amarillo	

Tabla 18

Distribución de pines de los motores con las tarjetas de potencia/ Robot2

Distribución de pines de los motores con las tarjetas de potencia/ Robot2			
Elemento	**Código**	**Color**	**Tarjeta**
Motor cintura (GND)	LM	Negro	Tarjeta 4
Motor cintura (VCC)	PQ	Blanco	
Motor hombro (GND)	NO	Amarillo	
Motor hombro (VCC)	VW	Tomate	
Motor codo (GND)	RS	Café	Tarjeta 5
Motor codo (VCC)	ZA	Plomo	

Motor muñeca pitch (GND)	TU	Rojo	
Motor muñeca pitch (VCC)	BC	Verde	
Motor muñeca roll (GND)	XY	Azul	
Motor muñeca roll (VCC)	DE	Blanco	Tarjeta 6
Motor gripper (GND)	GGD	Azul	
Motor gripper (VCC)	GGV	Amarillo	

Tabla 19

Distribución de los frenos de los motores y encoders del Robot1/Robot2

Distribución de los frenos de los motores y encoders del Robot1/Robot2/ Conector			
Número de pin	**Nombre**	**Código**	**Color**
1	Encoder canal A (cintura)	AA	Negro
2	Encoder canal B (cintura)	BB	Blanco
3	Encoder canal A (hombro)	CC	Plomo
4	Encoder canal B (hombro)	DD	Morado
7	Encoder canal Z (cintura)	EE	Azul
8	Freno GND (hombro)	FF	Café
11	Freno VCC (hombro)	GG	Rojo
17	VCC 3.3 (cintura)	HH	Tomate
18	Freno VCC (codo)	II	Amarillo
20	Encoder canal A (codo)	JJ	Verde
21	Encoder canal B (codo)	KK	Azul
22	Encoder canal A (muñeca pitch)	LL	Morado
23	Encoder canal B (muñeca pitch)	MM	Plomo
24	GND (cintura)	NN	Blanco
28	Freno GND (codo)	OO	Negro
31	Freno GND (muñeca pitch)	PP	Café
40	GND (codo/muñeca roll)	QQ	Rojo
43	Encoder canal A (muñeca roll)	RR	Tomate
44	Encoder canal B (muñeca roll)	SS	Amarillo
50	Freno GND (muñeca roll)	TT	Verde
51	VCC 3.3 (hombro/muñeca pitch)	UU	Morado
52	VCC 3.3 (codo/muñeca roll)	VV	Azul
55	GND (hombro/muñeca pitch)	WW	Plomo
53	Motor gripper VCC	NG	Amarillo
57	Motor gripper GND	MG	Azul

Tabla 20

Conexión de encoder a tarjeta de control/Robot1

Conexión de encoder a tarjeta de control/Robot1			
Nombre	**Código**	**Color**	**Pin/ Tarjeta de control**
Encoder canal A (cintura)	AA	Negro	O1
Encoder canal B (cintura)	BB	Blanco	O2
Encoder canal A (hombro)	CC	Plomo	O3
Encoder canal B (hombro)	DD	Morado	O4

Encoder canal A (codo)	JJ	Verde	O9
Encoder canal B (codo)	KK	Azul	O10
Encoder canal A (muñeca pitch)	LL	Morado	O17
Encoder canal B (muñeca pitch)	MM	Plomo	O18
Encoder canal A (muñeca roll)	LL	Morado	O13
Encoder canal B (muñeca roll)	MM	Plomo	O14

Tabla 21

Conexión de encoder a tarjeta de control/Robot2

Conexión de encoder a tarjeta de control/Robot2			
Nombre	**Código**	**Color**	**Pin/ Tarjeta de control**
Encoder canal A (cintura)	AA	Negro	O1
Encoder canal B (cintura)	BB	Blanco	O2
Encoder canal A (hombro)	CC	Plomo	O3
Encoder canal B (hombro)	DD	Morado	O4
Encoder canal A (codo)	JJ	Verde	O9
Encoder canal B (codo)	KK	Azul	O10
Encoder canal A (muñeca pitch)	LL	Morado	O17
Encoder canal B (muñeca pitch)	MM	Plomo	O18
Encoder canal A (muñeca roll)	LL	Morado	O13
Encoder canal B (muñeca roll)	MM	Plomo	O14

Tabla 22

Conexión de frenos a tarjeta de control/Robot1

Conexión de frenos a tarjeta de control/Robot1			
Nombre	**Código**	**Color**	**Pin/ Tarjeta de control**
Freno codo	FC1	Plomo	O6
Freno hombro	FH2	Morado	O5
Freno muñeca roll	FMR3	Azul	O7
Freno muñeca pitch	FMP4	Verde	O8

Tabla 23

Conexión de frenos a tarjeta de control/Robot2

Conexión de frenos a tarjeta de control/Robot2			
Nombre	**Código**	**Color**	**Pin/ Tarjeta de control**
Freno codo	FC5	Negro	O6
Freno hombro	FH6	Blanco	O5
Freno muñeca roll	FMR7	Tomate	O7
Freno muñeca pitch	FMP8	Amarillo	O8

Tabla 24

Conexión de frenos con pines de salida de tarjeta de frenos /Robot1/Robot2

Conexión de frenos con pines de salida de tarjeta de frenos /Robot1/Robot2			
Los frenos tienen una conexión de -12V			
Nombre	**Código/ Etiqueta**	**Color**	**Bornera de salida**
Freno GND (hombro)	FF	Café	P4
Freno VCC (hombro)	TF	Rojo	
Freno VCC (codo)	TF	Rojo	P1
Freno GND (codo)	OO	Negro	
Freno GND (muñeca pitch)	PP	Café	P6
Freno VCC (muñeca pitch)	TF	Rojo	
Freno GND (muñeca roll)	TT	Verde	P3
Freno VCC (muñeca roll)	TF	Rojo	

CONTINÚA

Tabla 25

Alimentación de 3.3V encoders

Alimentación de 3.3V encoders			
Nombre	**Código**	**Color**	**Bornera de salida**
VCC 3.3 (hombro/muñeca pitch)	UU	Morado	
VCC 3.3 (codo/muñeca roll)	V V	Azul	P8
GND (hombro/muñeca pitch)	QQ	Rojo	
GND (codo/muñeca roll)	WW	Plomo	

4.1.3. Etapa de potencia.

La mejora de la caja de control se hace con el rediseño de la parte de potencia que gobierna tanto a motores y frenos de los manipuladores, al igual que se hace un cambio del microcontrolador que albergará el control de estos manipuladores con el propósito de tener una mejor lectura de los encoders y por ende un mejor control; como también se puede efectuar la comunicación serial con el servidor que en este caso es una Raspberry Pi.

La parte de potencia consta de dos partes, la primera es referente a los motores que funcionarán a 24V, y la segunda es una tarjeta de control de los frenos, esto se puede apreciar de mejor manera en el esquemático de conexiones. (Anexo1).

La parte referente a la conexión de los motores de las articulaciones de los robots constará de una tarjeta de control de motor que es el Puente H Mosfet Irf3205 3-36v 10A, para robots como se explica a continuación:

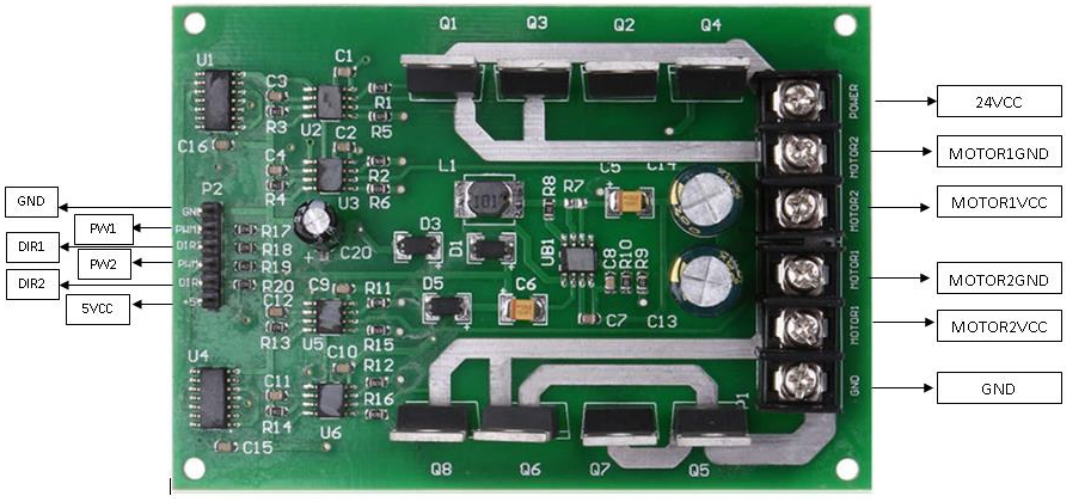

Figura 19. Esquemático Puente H de control de motores

"La tarjeta de Puente H para controlar motores se focaliza en mantener la eficiencia y potencia de los motores con los que trabaja, esto hace que la fuente de alimentación de la misma tenga una alta durabilidad" (Romero, 2017).

La principal razón por que se utilizó estas tarjetas en el rediseño de este proyecto se debió a la necesidad de mejorar la potencia con la que se trabajaba y obtener movimientos más certeros al momento de implementar el control cinemático; la elevación de potencia en los motores de cada articulación se realizó cambiando la fuente de alimentación a 24V. Debido a que estas tarjetas trabajan con un Mosfet IRF3205 de canal N de baja resistencia presenta pocas pérdidas de conmutación mejorando la utilización de energía y asegurando el control de los mismos.

El detalle de las ventajas que brinda esta tarjeta se describe en la siguiente tabla:

Tabla 26
Características Puente H control de motor

Características de Puente H control de motores	
Característica	**Especificación**
Voltaje nominal	3V-36V
Alimentación de control	5V
Corriente nominal	10A
Corriente máxima	30A

CONTINÚA

Número de motores que se conectan	2
Peso neto	53g
Tipo de control	PWM
Voltaje de control	0 – 5V

La parte referente a los frenos de los motores y alimentación de los encoders de las articulaciones de los robots se efectuó con el diseño de una placa de PCB como se visualiza en las siguientes figuras:

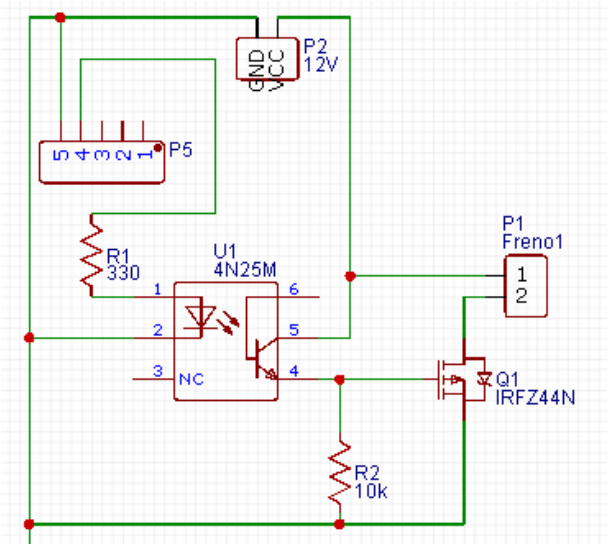

Figura 20. Circuito de potencia de un freno del motor

Como se puede observar en la figura 20 se tiene un circuito de control y potencia para la activación de los frenos de los motores de las articulaciones de los robots, el optoacoplador actúa como un interruptor y permite separar la parte de control de la de potencia, es así que la señal que recibe este circuito provendrá de la tarjeta de control formada por los microcontroladores Atxmega 64 Au3. La función de este elemento es recibir una activación (5V-control) y activar la tensión de salida que en este caso será de (12V-potencia).

El mosfet IRFZ44N se activará con voltaje, por datasheet se conoce que es necesario tener una tensión de 10VCC como mínimo en la entrada del gate, para la saturación del mosfet y posterior activación del freno, en la figura 21 el optoacoplador envía un voltaje de 12VCC, lo cual es suficiente para activación del mosfet.

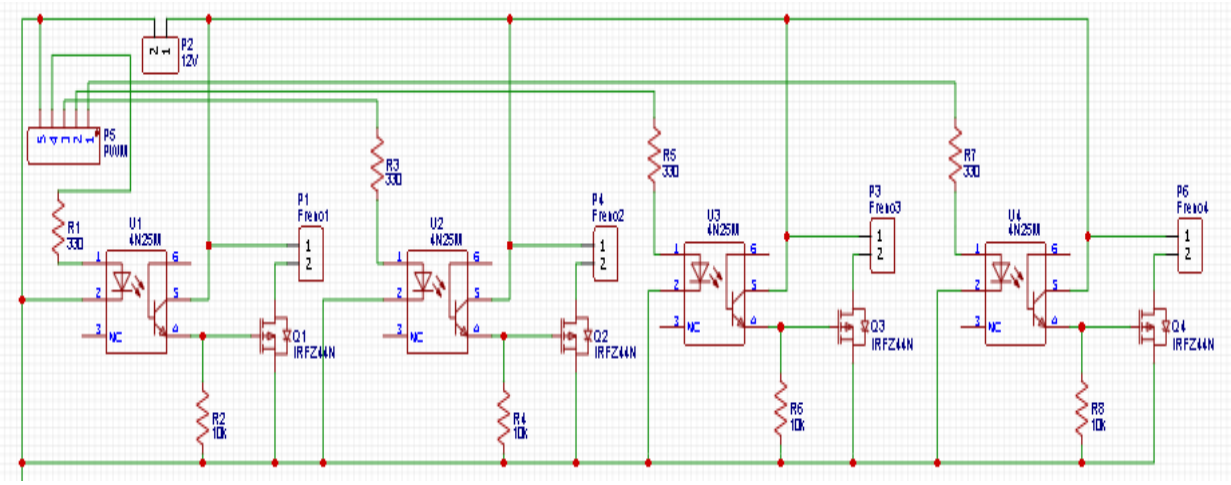

Figura 21. Esquemático de placa de frenos

Los encoders de cada robot funcionan a un voltaje de 3.3V; con la finalidad de evitar el uso de otra fuente de alimentación se utiliza un circuito simple con un LM317 para generar un voltaje de 3.3V a partir de una fuente de 5V.

"El LM317 es un regulador de voltaje que puede tener voltajes de entrada de entre 3 a 40 voltios y generar tensiones de 1,2 a 37 voltios" (Conde, 2016).

"Al momento de implementar un regulador de voltaje se recomienda que el voltaje de entrada tiene que estar entre 1,2 a 1,25 V por encima del voltaje de salida" (J.I.Huircan, 2012).

Debido a que el voltaje de entrada se convierte en un voltaje de referencial que tendrá ese valor. El voltaje de referencia se encuentra presente en los extremos de R1. La resistencia R2 es la que se va a calcular en función del voltaje salida y el valor de R1 que se escoja, en función de la siguiente formula:

$$R2 = \frac{R1}{Vref} * (Vout - Vref)$$

Para el cálculo de la resistencia se toma un Vref bajo que en este caso será de 1.25V, y una resistencia R1 de 330 ohmios la cual es comercial y fácil de encontrar en el mercado con estos datos se efectúa el siguiente calculo:

$$R2 = \frac{330}{1.25} * (3.3 - 1.25)$$

$$\boldsymbol{R2 = 545\Omega}$$

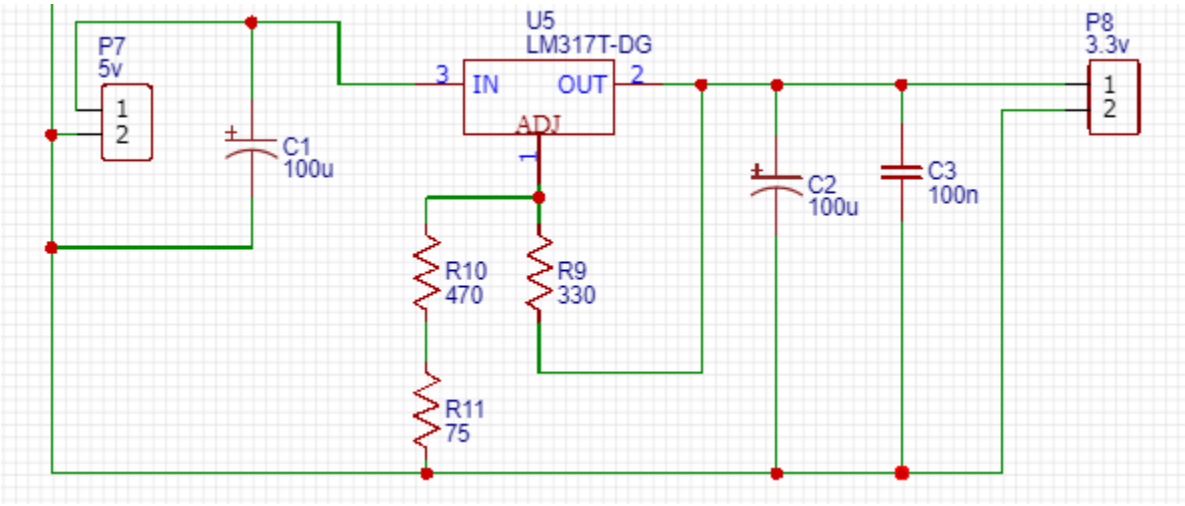

Figura 22. Circuito de alimentación de encoders

Con el diseño del circuito de potencia y control para frenos, al igual que el regulador de voltaje de 3.3V para alimentar los encoders se obtiene el siguiente PCB:

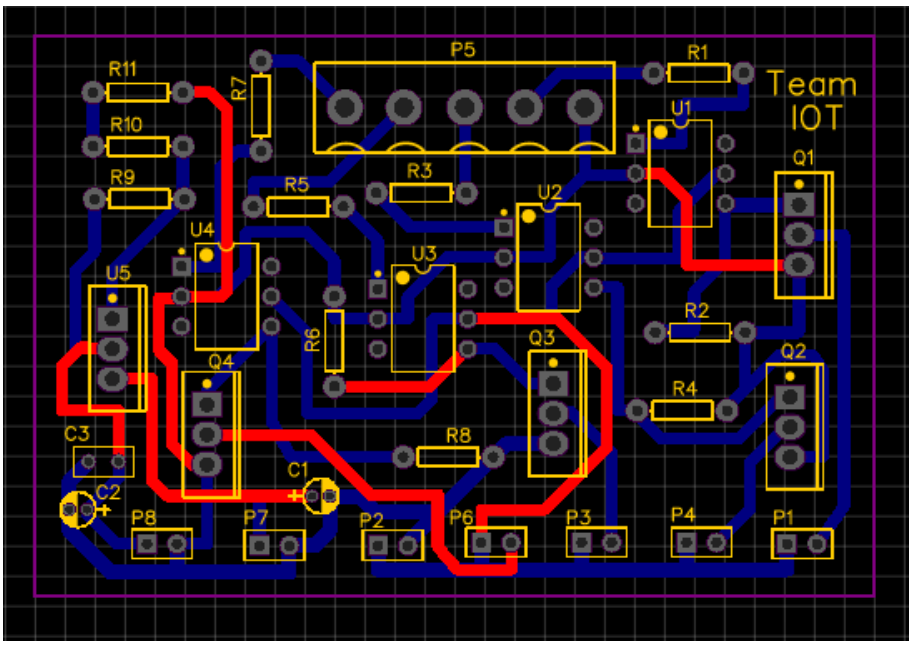

Figura 23. PCB de placa de frenos y alimentación de encoders

Figura 24. Placa de frenos armada en el tablero

4.1.4. Alimentación.

Si bien en el Capítulo 2 se detalla la corriente a la que funcionan los motores de las articulaciones y la que consumen los encoders es importante hacer un dimensionamiento de la corriente que se empleará en el sistema y en base a eso escoger el amperaje de las fuentes de alimentación. Lo inicial es aclarar que los motores de los manipuladores trabajarán con un voltaje de 24V y su movimiento se realizará articulación por articulación es así que el cálculo de corriente que necesitarán las fuentes se basará en un solo motor con carga tomando como base el motor de la cintura que es el que mayor carga tendrá.

Tabla 27

Consumo de corriente

Consumo de corriente del sistema		
Elemento del manipulador	**Tensión de trabajo**	**I máx. con carga**
Articulación Cintura	24V	2.5 A
Encoders totales (5)	5V	1A
Frenos totales (4)	12V	2A
Raspberry Pi	5V	2A
Controlador	5V	0.5 A
Tarjeta de encoders	5V	1A
Corriente total		**9A**

Con la finalidad de separar la parte de control de la de potencia y al no contar con una fuente variable se escogió que la caja de control tenga tres tipos de alimentación 24V, 12V y 5V, esto se debe a que las tarjetas de control funcionan a 5V, las tarjetas para los frenos a 12V, y la alimentación de las tarjetas de puente H para los motores se realizó con 24V. Para escoger el amperaje de las fuentes es necesario basarse en la tabla anterior que muestra la corriente máxima que tiene cada elemento, tomando en cuenta que para dicho cálculo se debe tomar el valor de la corriente máxima más un 20% del valor de la misma que deben tener como mínimo las fuentes a escoger; aclarando que en el caso de la tarjeta Raspberry Pi tiene su fuente independiente.

- Fuente de 24V

$$I = (Imax * 0.2) + Imax$$

$$I = (2.5 * 0.2) + 2.5$$

$$\boldsymbol{I = 3A}$$

- Fuente de 12V

$$I = (Imax * 0.2) + Imax$$

$$I = (2 * 0.2) + 2$$

$$\boldsymbol{I = 2.4A}$$

- Fuente de 5V

$$I = (Imax * 0.2) + Imax$$

$$I = (2.5 * 0.2) + 2.5$$

$$\boldsymbol{I = 3A}$$

Tomando en cuenta las corrientes mínimas que deben de poseer las fuentes se instalaron las siguientes fuentes en la caja de control:

Figura 25. Fuente de 24V

Tabla 28

Características de Fuente de 24V

Características de Fuente de 24V	
Característica	**Especificación**
Protección	Sobre voltaje de salida
	Variaciones de voltaje
	En cortocircuito
Entrada	100-240 VCA 50/60 Hz
Voltaje de salida	24 V
Amperaje	5A

Figura 26. Fuente de 12V

Tabla 29

Características de Fuente de 12V

Características de Fuente de 12V	
Característica	**Especificación**
Protección	Sobre voltaje de salida
	Variaciones de voltaje
	En cortocircuito
Entrada	100-240 VCA 50/60 Hz
Voltaje de salida	12 V
Amperaje	5A
Carcasa	Acero

Figura 27. Fuente PC de 5V

Tabla 30

Características de Fuente PC de 5V

Características de Fuente de 5V	
Característica	**Especificación**
Protección	Sobre voltaje de salida
	Variaciones de voltaje
	En cortocircuito
Entrada	115-230 VCA 50/60 Hz
Voltaje de salida	12 V/ 5V
Potencia	750W
Carcasa	Metal
Sistema de enfriamiento	1 ventilador

4.1.5. Sistema de enfriamiento.

Figura 28. Sistema de enfriamiento

La caja de control estará formada por cuatro ventiladores que funcionan a 12V, tres de ellos están colocados de forma que hacen circular el aire que se encuentre dentro de la caja de control, y el cuarto cumple la función de sacar ese aire hacia el exterior de la caja de control; este sistema de enfriamiento se instaló en la caja de control con la finalidad de evitar altas temperaturas en los componentes de control como de potencia por su continuo funcionamiento.

4.1.6. Botonera.

El diseño de la botonera para la caja de control contempla las normas expuestas por la guía GEDIS que menciona que los paneles de control deben de carecer de ambigüedad para su fácil manipulación y evitar errores en su ejecución. Tomando en cuenta que las funciones de dicho panel deben ser distribuidas de forma horizontal y vertical, al igual que las luces indicadoras deben ser colocadas sobre los selectores de mando en el caso de ejecución y funcionamiento; dichos selectores deben tener sus estados bien definidos sean estos de Paro/Marcha (para la ejecución) y ON/OFF (para el encendido); el panel de control siempre debe de contar con un Paro de emergencia en el caso de existir problemas en el funcionamiento del proceso que controla dicho armario de control. Cumpliendo con la normativa expuesta, la caja de control constará de una botonera para la fácil manipulación de los robots, es así que se tiene los siguientes elementos:

Figura 29. Botonera

Tabla 31
Botonera caja de control

Botonera	
Elemento	**Función**
ON-OFF	Encenderá o apagará toda la alimentación de la caja de control
Encendido Robot1	Habilita el funcionamiento del Robot1
Encendido Robot2	Habilita el funcionamiento del Robot2
Start	Luz indicadora del estado de todo el sistema
Robot1	Luz indicadora del robot1
Robot2	Luz indicadora del robot2
Paro de emergencia	Deshabilita el funcionamiento de los manipuladores Robóticos

4.1.7. Pines de conexión.

Los manipuladores robóticos cuentan con dos conectores SPEC-MIL, en la nueva caja de control que se elaboró. Se colocó los conectores de cada robot como se observa en la siguiente figura:

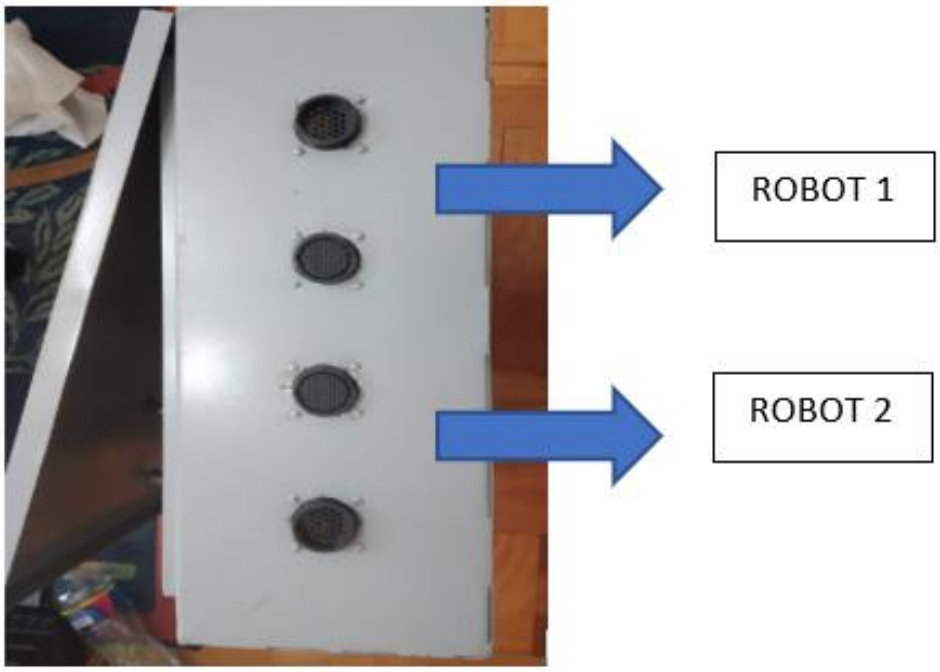

Figura 30. Distribución de conectores de los manipuladores robóticos

El significado de cada pin de conexión de los conectores se puede describir en las siguientes figuras:

Figura 31. Pines de conexión Robot CRS A255

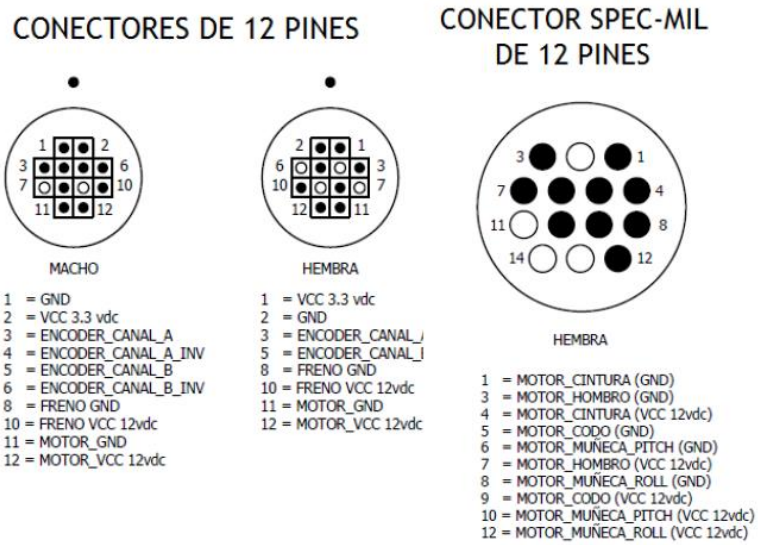

Figura 32. Pines de conexión Robot CRS A255 esquemático

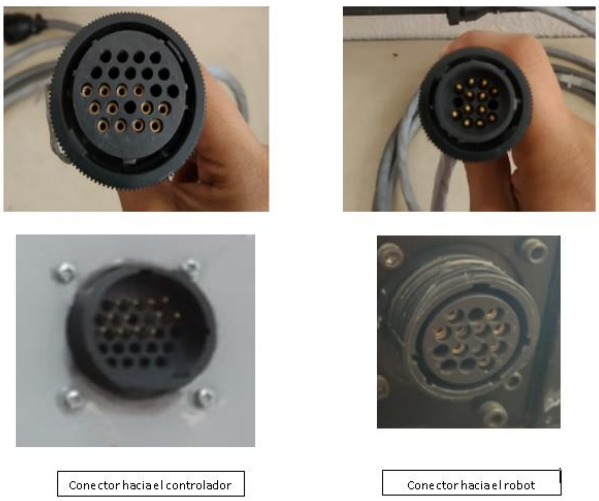

Figura 33. Pines de conexión Robot CRS A255 real

4.2. Servidor

Como servidor de este proyecto se contempló el uso de la tarjeta Raspberry Pi 3B, esta tarjeta tiene como función almacenar la plataforma cliente web, al igual que comunicarse con la tarjeta de control.

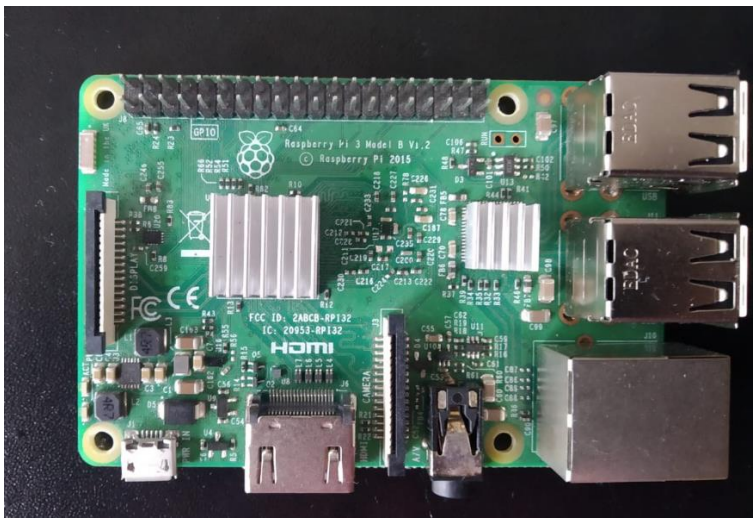

Figura 34. Raspberry Pi 3

Los orígenes de Raspberry Pi datan del año 2012 en Reino Unido en donde se lanzó el primer prototipo de este mini computador, donde nace con la finalidad de incentivar a niños a aprender de informática, es así donde Raspberry Pi empieza a realizar pequeños proyectos de electrónica bajo el concepto hazlo tú mismo.

Con el transcurso de los años Raspberry Pi ha sido utilizada en cuatro cosas principales:

- Media center: convertir una televisión en Smart.
- Videoconsola retro: para jugar grandes juegos retros como RetroPie.
- Ordenador: se puede instalar sistemas como Raspbian, Fedora o Ubuntu.
- Domótica: facilita hacer una casa un poco más inteligente.

En este caso se lo utilizó como ordenador, ya que se instaló el sistema operativo Raspbian, en el cual está instalado el framework Django, descrito en capítulos posteriores, las características con las que se está trabajando en esta tarjeta se presentan en la siguiente tabla:

Tabla 32
Descripción de Raspberry pi 3B

Raspberry pi 3 modelo B	
Característica	**Descripción**
Procesador	Broadcom BCM2837, Cortex-A53 (ARMv8) 64-bit SoC
Frecuencia de reloj	1,2 GHz
GPU	VideoCore IV 400 MHz
Memoria	1GB LPDDR2 SDRAM
Conectividad inalámbrica	2.4GHz IEEE 802.11.b/g/n Bluetooth 4.1
Conectividad de red	Fast Ethernet 10/100 Gbps
Puertos	GPIO 40 pines HDMI 4 x USB 2.0 CSI (cámara Raspberry Pi) DSI (pantalla táctil) Toma auriculares / vídeo compuesto Micro SD Micro USB (alimentación)

4.3. Tarjeta de control

El control de los manipuladores robóticos se efectuará a través de una tarjeta diseñada con el microcontrolador ATXMEGA 64 D3, cada manipulador contará con su propia tarjeta y se comunicará con el servidor que es la Raspberry Pi mediante comunicación serial a través de Rx y Tx, como se observa en la siguiente figura:

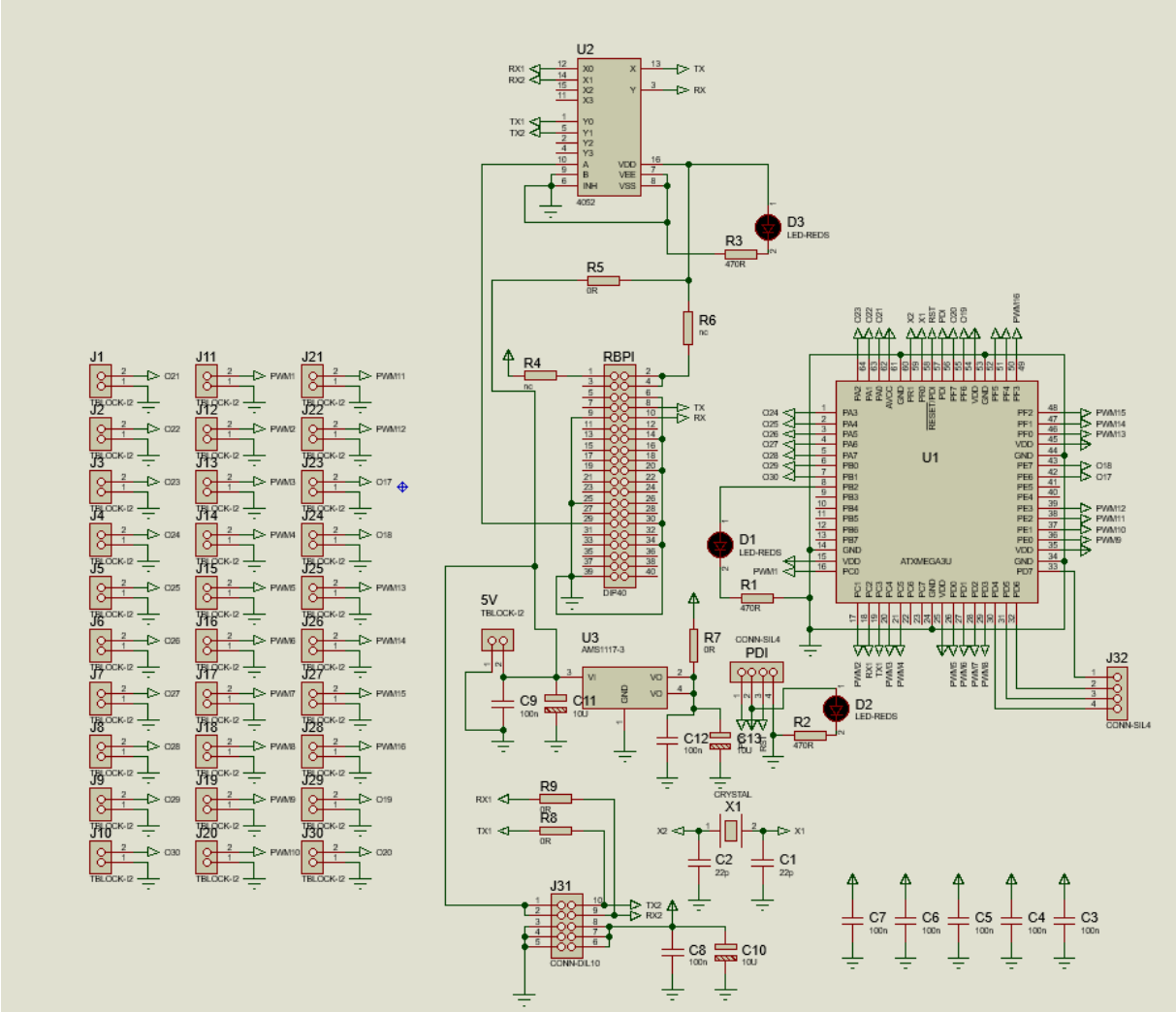

Figura 35. Diseño de tarjeta de control para los manipuladores robóticos

En la figura se observa que la tarjeta de control de cada manipulador consta de un microcontrolador ATXMEGA 64D3, un multiplexor CD74HC4052E y un regulador de voltaje

AMS1117, a continuación, se describirán cada uno de los componentes que conforman a esta tarjeta:

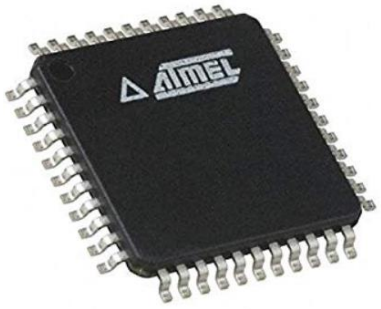

Figura 36. ATXMEGA 64 D3

El microcontrolador ATXMEGA 64D3 es un microcontrolador AVR de 8 bits cuyas características se enuncia en la siguiente tabla:

Tabla 33
Descripción de microcontrolador

ATXMEGA 64D3	
Característica	**Descripción**
Tipo de memoria de programa	Destello
Tamaño de la memoria del programa (KB)	64
Velocidad de CPU (MIPS / DMIPS)	32
Bytes SRAM (KB)	4
Datos EEPROM / HEF (bytes)	208
Periféricos de comunicación digital	3-UART, 5-SPI, 2-I2C
Capturar / Comparar / Periféricos PWM	18 captura de entrada, 18 CCP, 18PWM
Número de comparadores	2
Rango de temperatura (° C)	-40 a 85
Rango de voltaje de funcionamiento (V)	1.6 a 3.6
Pines totales	64
Baja potencia	Si

Una de las principales razones por las que se escogió este microcontrolador es porque maneja interrupciones multinivel, es decir que facilitará la lectura de los encoders de cada manipulador, y permitirá hacer un monitoreo simultaneo de cada articulación del robot.

Figura 37. Multiplexor CD74HC4052E

La utilidad de este multiplexor en la placa es actuar como un switch digital que permita elegir si se trabaja con la placa del robot 1 o la placa del robot 2, esto se debió a que la Raspberry Pi maneja una sola comunicación UART, es así que para solventar este problema se utiliza este multiplexor para seleccionar la placa que recibirá las ordenes desde el servidor. A continuación, se describen las características de este circuito integrado:

Tabla 34
Descripción de circuito integrado

CD74HC4052E	
Característica	**Descripción**
Número de canales	4
Configuración	2 x 4:1
Resistencia en modo encendido	130 Ohms
Voltaje de alimentación operativo:	2 V a 6 V
Temperatura de trabajo	- 55 C a +125 C
Ancho de banda:	185 Mhz
Tipo de producto:	Multiplexor Switch ICs
Máximo voltaje de alimentación doble:	+/- 5 V

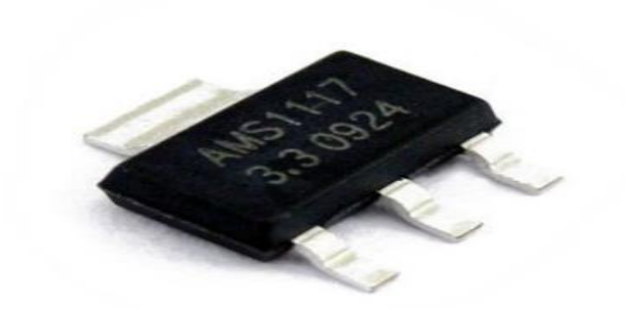

Figura 38. Circuito integrado de 3.3 V

El circuito integrado AMS1117 actúa como un regulador de voltaje a 3.3V, como se describió anteriormente el microcontrolador ATXMEGA 64D3, funciona a 3.3V, por lo cual es necesario realizar un regulador de tensión para alimentar el microcontrolador, sus características se describen a continuación:

Tabla 35
Descripción de circuito integrado

AMS1117	
Característica	**Descripción**
Voltaje de salida:	3.3V
Voltaje de entrada máximo	15 V
Corriente máxima	1A
Temperatura de operación	-20 a 120°C.
Dimensiones	36mm*17mm*14mm
Tipo de Integrado	Montaje superficial

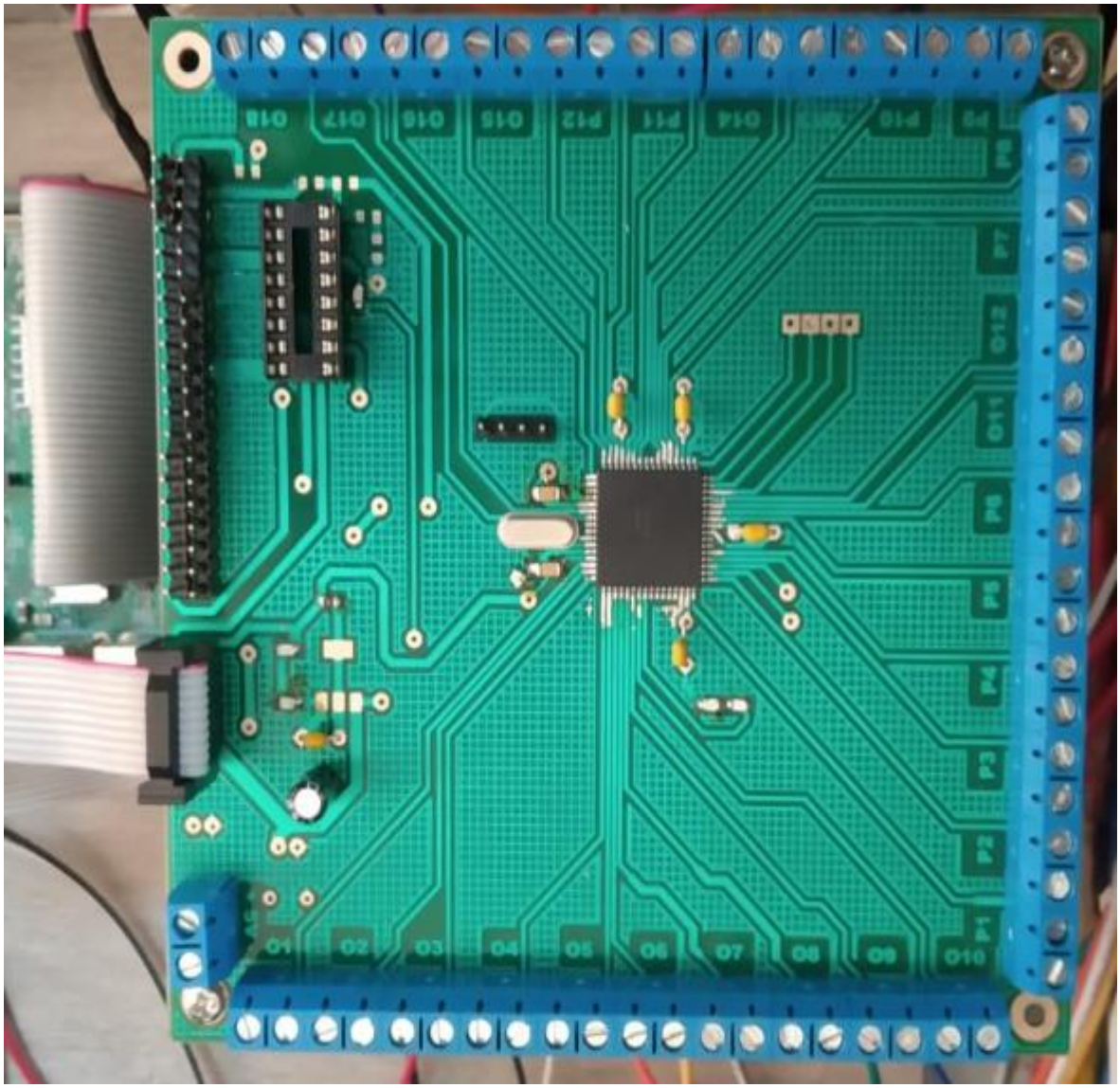

Figura 39. Tarjeta de control dentro del tablero

4.4. Acoplamiento de las señales provenientes del encoder

En el Capítulo 3 se describió los encoders con los que se trabaja en cada manipulador robótico, y el tipo de señal que entregan, siendo dicha señal una señal TTL es necesario que sea acondicionada para poder ser leída por el microcontrolador, que trabaja en un rango de 0 a 3.3V.

El acondicionamiento de dicha señal se efectuó con un comparador de nivel reduciendo de esta forma el ruido que puede provenir de la lectura de dichos encoders y obteniendo una señal más

pura con la que se pueda trabajar. Para realizar el comparador de nivel se utilizó el transistor 2N3904, esto debido a que si se utilizaba un amplificador operacional se necesita manejar fuentes simétricas tanto positivas y negativas para su funcionamiento generando posibles ruidos y distorsiones de la señal a acoplar y necesitando que la señal de los encoders sea lo más pura posible es preferible trabajar con transistores en este caso.

El circuito que se implementó se detalla en la siguiente figura:

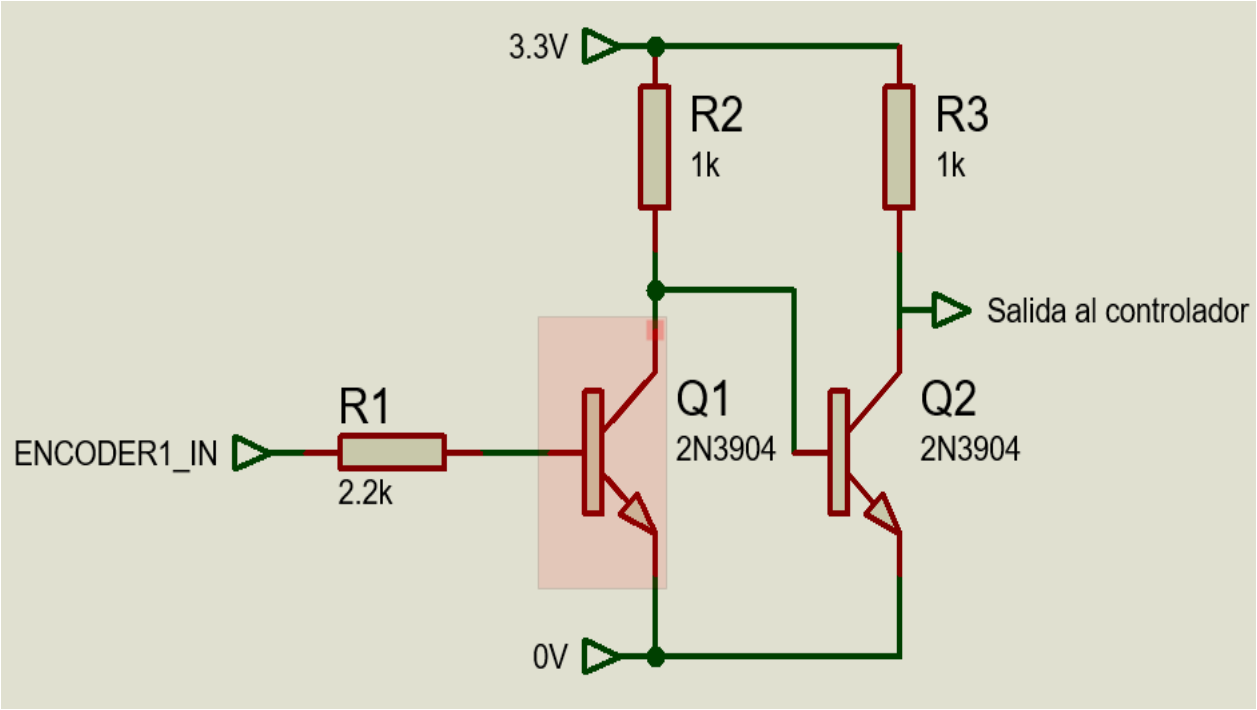

Figura 40. Circuito para acoplar la señal del encoder que va a ser ingresada al microcontrolador

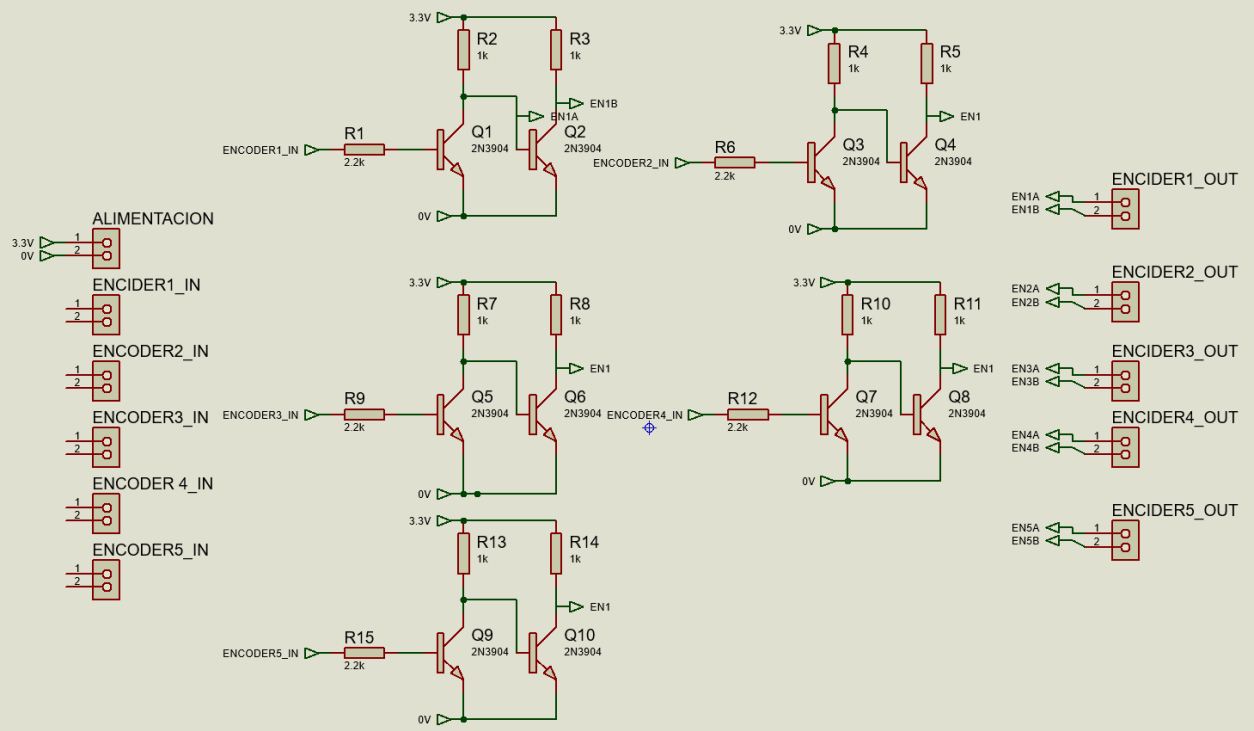

Figura 41. Diseño de placa para el acople de señal de los cinco encoders del manipulador

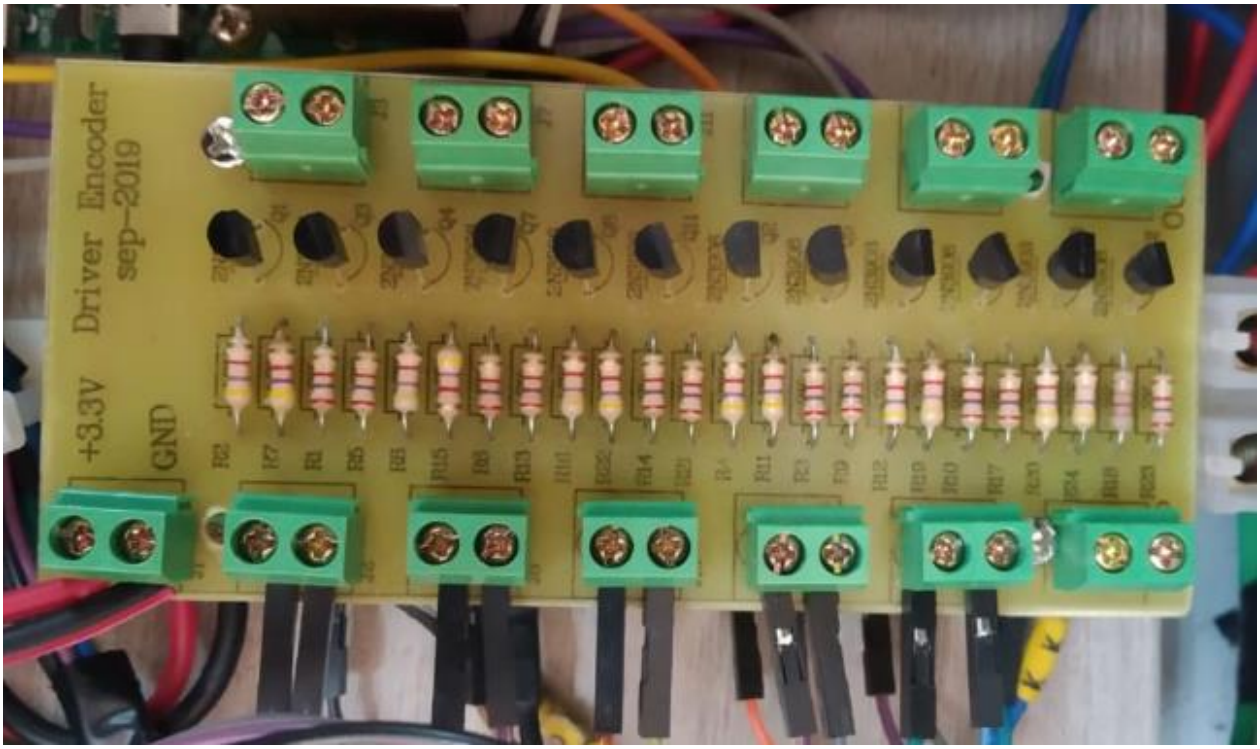

Figura 42. Driver de encoders dentro del tablero de control

4.5. Carcasa del Robot

En el proyecto se contempla modificaciones estructurales en un robot, por lo cual se realizó un rediseño de la carcasa que cubre las partes eléctricas y mecánicas del mismo.

El desarrollo de una nueva carcasa se efectuó por los cambios de dimensionamiento que tuvo el manipulador, siendo dichas piezas impresas en 3D, en el anterior proyecto se realizó la impresión y diseño de dichas piezas, pero no presentaban las características de dureza y resistencia requeridas para este manipulador, por lo que se deterioraron con el tiempo sufriendo rupturas y daños considerables como se observa en la siguiente figura:

Figura 43. Estado actual de la carcasa

En la mejora de la carcasa se utilizó la pieza que se mantenía en buen estado que corresponde a la que recubre la cola del robot donde se encuentran los motores de las muñecas.

Las piezas laterales del manipulador quedaron inservibles al sufrir rupturas, por lo que fue necesario volver a imprimirlas en base al diseño del anterior proyecto. Lo que se hizo con estas tres piezas antes de ser colocadas en el manipulador fue dotarlas de dureza y resistencia, por lo que se

las recubrió de una capa de masilla, dejándolas secar para posteriormente lijarlas y volver a masillarlas con la finalidad de que dichas piezas presente gran resistencia.

Figura 44. Masillado de las piezas a instalarse en el manipulador

Por último, se las pintó de color negro quedando de la siguiente manera:

Figura 45. Piezas masilladas y pintadas

La colocación de las piezas en el manipulador se realizó con pernos haciéndolas fijas al manipulador como se observa en la siguiente figura.

Figura 46. Manipulador robótico con nueva carcasa

CAPÍTULO V

COMUNICACIÓN CON´PLATAFORMA CLOUD

5.1. Django

"Django es un framework web diseñado para realizar aplicaciones de cualquier complejidad en unos tiempos muy razonables. Está escrito en Python y tiene una comunidad muy amplia, que está en continuo crecimiento" (Camino, 2018).

5.1.1. Introducción.

Se controlará una celda robótica de forma remota por lo cual es necesario el uso de una herramienta, a través de un entorno visual. Teniendo en cuenta que la herramienta escogida satisfaga las necesidades del proyecto y sea compatible con los recursos utilizados en el mismo.

Por lo que la herramienta a usar será un framework, dentro del entorno informático, "framework es un conjunto de componentes que te ayudan a desarrollar sitios web más fácil y rápidamente" (Zapke, 2013).

5.1.2. Selección de la plataforma Cloud.

La primera opción como plataforma Cloud, fue Node-RED, debido a que su programación es a base de nodos, similar a la programación en LabVIEW por lo que es una programación sencilla, de igual manera es de código abierto por lo que se puede descargar diferentes nodos gratuitamente; su enfoque principal es el IoT, usando diferentes protocolos como REST, MQTT, AMQP entre otros, lamentablemente tiene una desventaja la cual es que no permite el desarrollo libre de un sitio web, ya que posee plantillas preestablecidas para botones, gráficas, sliders, entre otros; además no había la posibilidad de introducir diferentes códigos escritos en Python, dentro del entorno de Node-RED.

La segunda opción fue buscar un framework que utilice el lenguaje de programación PHP, debido a que dicho lenguaje de programación permite una conexión entre una página web y la interacción entre la Raspberry Pi a través de sus puertos GPIO. Existen algunos framework que manejan este lenguaje como Laravel, Symfony, Aura, FuelPHP, entre otros. Lamentablemente no se podía programar dentro de PHP la comunicación serial entre la Raspberry Pi y el microcontrolador ATxmega64D3.

Finalmente, se consideró factible utilizar el framework web Django, debido a que presenta características útiles para el desarrollo del proyecto.

Antes de explicar sus diferentes particularidades es necesario conocer la historia del framework Django, para poner en contexto la razón por la cual Django fue creado y la manera en que trabaja.

Django nació naturalmente de aplicaciones de la vida real escritas por un equipo de desarrolladores Web en Lawrence, Kansas. Nació en el otoño boreal de 2003, cuando los programadores Web del diario Lawrence Journal-World, Adrian Holovaty y Simon Willison, comenzaron a usar Python para crear sus aplicaciones (uniwebsidad, 2006).

En el verano boreal de 2005, luego de haber desarrollado este framework hasta el punto en que estaba haciendo funcionar la mayoría de los sitios World Online, el equipo de World Online, que ahora incluía a Jacob Kaplan-Moss, decidió liberar el framework como software de código abierto. Lo liberaron en julio de 2005 y lo llamaron Django, por el guitarrista de jazz Django Reinhardt (uniwebsidad, 2006).

Se observa que Django es un framework que fue muy utilizado en el ámbito de sitios web de noticias los cuales necesitaban un manejo controlado de grandes bases de datos; sin embargo, esto no quiere decir que Django no sea una herramienta efectiva para la creación de cualquier sitio web.

Ya conociendo la historia de Django se puede comprender de mejor manera ciertas características peculiares del framework. Entre sus particularidades se observa que Django es un framework que está escrito en Python, por lo que puede heredar todas las funcionalidades que nos ofrece dicho lenguaje, y como ya se dijo anteriormente es de código abierto.

Es un framework de alto nivel y muy rápido por lo que se puede realizar un diseño de una página web compleja, en un tiempo razonable.

Tiene una comunidad muy amplia que está en continuo crecimiento, en donde puedes encontrar ayuda, módulos o paquetes los cuales son herramientas que te brinda Django para realizar actividades que se encuentran comúnmente en una página web, como, por ejemplo: registro, inicio de sesión, etc.

Django es un framework muy seguro y estable, además la consulta a la base de datos de Django se encuentra en una API muy ordenada y de fácil acceso a cualquier tabla de datos.

Por otro lado, se tiene que Django es un framework muy escalable y versátil, por lo que no es necesario comenzar desde cero, si ya se tiene alguna página web y se desea modificarla para hacerla más compleja.

5.1.3. Instalación en el entorno Raspbian Stretch.

Es necesario antes de instalar cualquier programa tener el sistema operativo Raspbian Stretch para lo cual, es necesario descargar la imagen .iso del sistema operativo desde la página oficial de la fundación Raspberry.

Ya descargada la imagen .iso, se almacena el sistema operativo en una memoria SD de 16GB clase 10 o superior. Por lo que es necesario en primer lugar, formatear la memoria SD, a través del

programa SDFormatter V4.0, posteriormente se monta la imagen .iso a la memoria SD con el uso del programa Win32 Disk Imager.

Lo primero que se debe realizar en cualquier instalación dentro del sistema Raspbian, es ejecutar las siguientes líneas de comando en el terminal.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

"El primer comando permite la actualización de la lista de paquetes disponibles y sus versiones, pero no instala ningún paquete, y el segundo comando instala los paquetes actualizados del primer comando" (Carazo, 2013).

El sistema operativo Raspbian Stretch viene por defecto instalado algunos programas como son Python 2.7.9, el cual se utiliza para instalar Django 1.11 u otra versión inferior, y es con la que se trabaja en el desarrollo de este proyecto.

Como se necesita una versión específica de Django es necesario instalarlo a través del sistema de gestión de paquetes pip, para lo cual es necesario instalar pip.

```
$ sudo apt-get install python-pip
```

Ya instalado pip se procede a instalar Django 1.11 con la siguiente línea de comando en el terminal.

```
$ sudo pip install django==1.11
```

Para verificar que versión de Django se instaló, se ejecuta el siguiente comando.

```
$ django-admin --version
```

5.1.4. Funcionamiento.

Para el uso de Django es necesario la instalación de un entorno virtual en Raspbian, para lo cual se ejecuta la siguiente línea de comandos en el terminal.

```
$ sudo pip install virtualenv
```

Para verificar la versión del entorno virtual que se instaló, se ejecuta el siguiente comando.

```
$ virtualenv --version
```

Se procede a crear la carpeta donde se almacenará el proyecto en Django y el activador del entorno virtual.

```
$ mkdir Proyectos-Django
```

A continuación, se crea el proyecto para lo cual se debe ingresa por el terminal a la carpeta anteriormente creada.

```
$ cd Proyectos-Django
$ django-admin startproject Diana4 .
```

La parte que dice “Diana4” es el nombre del proyecto el cual puede ser sustituido por cualquier otro, pero no se debe de olvidar colocar el punto que se encuentra luego del nombre del proyecto.

A continuación, se debe crear el entorno virtual, el cual se debe activar para poder ejecutar el proyecto de Django.

```
$ cd Proyectos-Django

$ virtualenv activate
```

La palabra “activate” es el nombre del entorno virtual a crear, el cual puede ser sustituido por cualquier otro nombre. Ya creado el entorno virtual se ingresa a la carpeta activate y se coloca el siguiente comando, para activar el entorno virtual.

```
$ source bin/activate
```

Esto permitirá activar el entorno virtual “activate”.

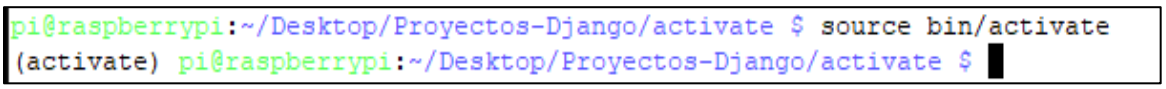

Figura 47. Entorno virtual “activate” activado

Desde este punto ya se puede realizar el diseño de la página web lo cual está detallado en el Capítulo 6.

Ya finalizado el diseño de la página web se procede a correr el servidor para lo cual se debe tener activado el entorno virtual, e ingresar a la carpeta del proyecto para ejecutar el siguiente comando.

```
$ python manage.py runserver 0.0.0.0:8000
```

Si todo se hizo correctamente se mostrará el siguiente mensaje.

```
(activate) pi@raspberrypi:~/Desktop/Proyectos-Django/Diana4 $ python manage.py runserver 0.0.0.0:8000
Performing system checks...

System check identified no issues (0 silenced).
August 07, 2019 - 20:57:55
Django version 1.11, using settings 'Diana4.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

Figura 48. Mensaje satisfactorio de ejecución del servidor Django

5.1.5. PostgreSQL.

Dentro de Django se puede escoger con que base de datos se va a trabajar entre las más usadas tenemos PostgreSQL, MySQL, Microsoft SQL Server, Oracle, etc.

Por lo cual se decide usar PostgreSQL, debido a que es muy popular, existe gran información de cómo usarla dentro del framework Django, es una base de datos escalable, es gratuito, libre, es fácil de administrar, además una característica primordial para este proyecto es que está diseñada para soportar grandes volúmenes de datos.

Para instalar PostgreSQL en Raspbian Stretch es necesario ingresar la siguiente línea de comando en el terminal.

```
$ sudo apt install postgresql libpq-dev postgresql-client postgresql-client-common
```

A pesar de que su manejo se puede realizar desde el terminal es de mayor comodidad tener una herramienta visual para poder administrarla, para lo cual se utiliza pgAdmin3, en donde se puede hacer búsquedas SQL y desarrollar bases de datos.

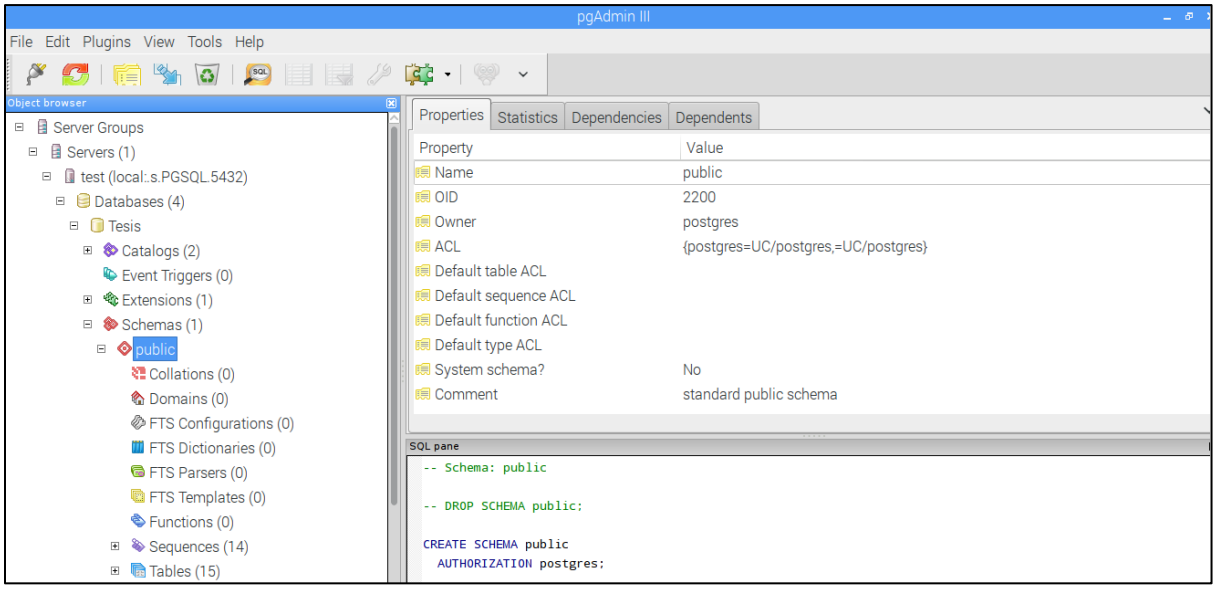

Figura 49. Entorno pgAdmin3

Para instalar pgAdmin3 en Raspbian Stretch es necesario ingresar la siguiente línea de comando en el terminal.

```
$ sudo apt install pgadmin3
```

5.2. Ngrok

"Es una herramienta que permite acceder al servidor local a cualquier persona en internet con la que se comparta una url generada dinámicamente" (NeverCracker, 2017).

5.2.1. Introducción.

Ngrok permite realizar lo mencionado anteriormente sin hacer ninguna modificación en el router o los Firewalls de la computadora.

Existen otras alternativas que se puede usar como es Tunnel o Serveo.net, pero tienen algunas desventajas. Tunnel tiene una gran desventaja la cual es la inseguridad en la creación del túnel, en

cambio Serveo.net era muy seguro e incluso se puede escoger la url generada; sin embargo, tiene la desventaja de no mantener la conexión en periodos largos de tiempo, por lo que se escogió al final Ngrok.

5.2.2. Interacción con el framework Django.

Teniendo en cuenta que Django lleva por defecto el levantamiento de un servidor web local; es decir, que la página o páginas web que se realicen en Django solo se las podrá visualizar en una red Local, por lo que era necesario encontrar una herramienta que permita crear un túnel entre el servidor local y el internet, dicha herramienta es Ngrok.

5.2.3. Instalación.

Para la instalación de Ngrok sé ingresa el siguiente comando.

```
$ sudo wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-arm.zip
```

El cual descargará en el sistema Raspbian el programa Ngrok. Lo único que hace falta es descomprimir el archivo para que empiece su instalación.

```
$ sudo unzip ngrok-stable-linux-arm.zip
```

5.2.4. Funcionamiento.

Es necesario correr el servidor de Django anteriormente para ello se debe seguir todos los pasos mencionados en el ítem 5.1.4. Ya concluido esos pasos se ingresa a una nueva ventana de terminal y se coloca el siguiente comando.

```
$ ./ngrok http 8000
```

El número 8000 es el puerto que se está utilizando para correr el servidor de Django se puede usar uno diferente, pero si se lo modifica igual se lo debe modificar al momento de correr el servidor de Django. Si todo se lo hizo correctamente se mostrará el siguiente mensaje.

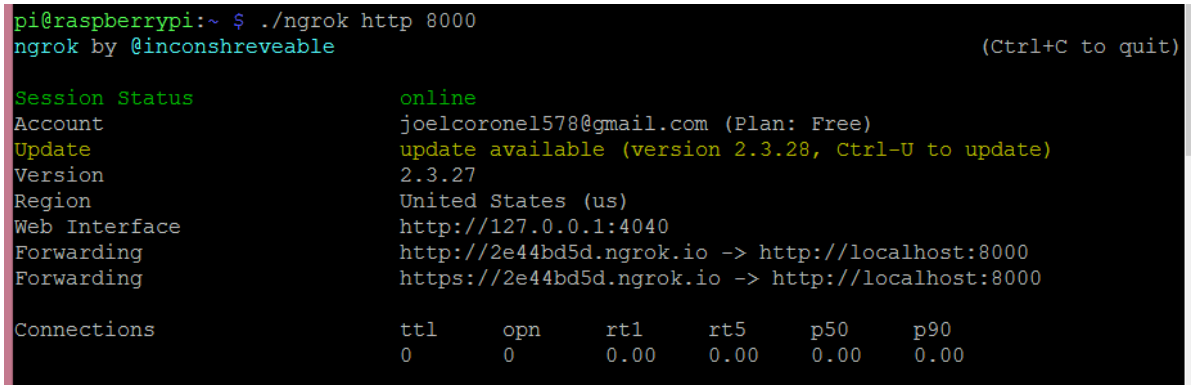

Figura 50. Programa Ngrok en funcionamiento

Una de las desventajas de Ngrok free es que arroja una URL dinámica la cual antes de ingresar a la página web creada por Django, es necesario modificar el archivo settings.py ingresando la URL que impone Ngrok, en este caso sería “2e44bd5d.ngrok.io”

Figura 51. Archivo setting.py, línea 28

CAPÍTULO VI

DISEÑO DE LA INTERFAZ CLIENTE WEB

6.1. Introducción

Al momento de crear un nuevo proyecto en Django se crea un archivo llamado manage.py, el cual sirve como ayuda de la administración del sitio, además se crea una carpeta con el nombre del proyecto, dentro de la carpeta mencionada se originan cuatro archivos con los siguientes nombres: __init__.py, settings.py, urls.py y wsgi.py.

De estos cuatro archivos solo dos archivos se utilizarán, el primero será el archivo settings.py en donde se encuentra la configuración del sitio web, el segundo archivo será el urls.py en donde se coloca las URL utilizadas es decir las diferentes rutas del sitio web. A pesar de no utilizar los otros dos archivos es importante no borrarlos, sino el sitio web no funcionará correctamente.

6.2. Configuraciones principales

Se abre el archivo settings.py, y lo primero que se modificará es el código de lenguaje, el cual tiene dos partes, la primera es el código del idioma y la segunda el código del país, además se cambia la zona horaria, para conocer el código de la zona horaria es necesario ingresar a la siguiente página *https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.*

```
LANGUAGE_CODE = 'es-mx'

TIME_ZONE = 'UTC'
```

Figura 52. Configuración del lenguaje y zona horaria de la página web

Debido a que la página web utilizará archivos estáticos como imágenes, archivos de tipo javascript y css; es necesario permitir su uso, para lo cual se debe generar una ruta escribiendo las siguientes líneas de código al final del archivo settings.py.

Con lo cual se dice que se posee una carpeta con el nombre static, y dentro de ella estarán todos los archivos estáticos.

```
STATIC_URL = '/static/'
STATICFILES_DIRS = (os.path.join(BASE_DIR, 'static'),)
```

Figura 53. Acceso para el uso de archivos estáticos

Por defecto viene configurada la base de datos sqlite3, por lo que hay que modificarla para que funcione con PostgreSQL.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'Tesis',
        'USER': 'pi',
        'PASSWORD': 'raspberry',
        'HOST': 'localhost',
        'PORT': 5432,
    }
}
```

Figura 54. Configuración para el uso de PostgreSQL

6.3. Archivos estáticos

Como ya se mencionó en el ítem 6.2, la página web tendrá archivos estáticos los cuales permiten el arreglo visual del sitio web. Para lo cual se creó una carpeta llamada static dentro de la misma se creó diferentes subcarpetas con los siguientes nombres: images, js, plugins, styles. En la primera existen todo tipo de imágenes png, jpg y svg las cuales han sido utilizadas dentro del sitio web. En la segunda carpeta se halla todos los archivos de javascript. En la tercera carpeta se encuentra

archivos tanto de javascript como css. Finalmente, en la última carpeta se visualiza archivos de tipo css.

Para el uso de archivos estáticos en cualquier página HTML es necesario colocar las siguientes líneas de código al principio del archivo *{% load staticfiles %}*.

Un ejemplo de permiso para el uso de un archivo css dentro de HTML es el siguiente.

```
<link rel="stylesheet" type="text/css" href="{% static 'styles/responsive.css' %}">
```

Figura 55. Ejemplo de permiso de un archivo css en HTML

Viendo la línea de código anterior se observa que lo único diferente en una programación de HTML normal, es el atributo href, el cual permite colocar la ruta del archivo, por lo tanto siempre que se usa Django, al comienzo del atributo href, se tiene que colocar *"{% static '*, en donde static es el nombre de la carpeta en donde están todos los archivos estáticos, luego se colocará la ruta del archivo dentro de comillas simples, por último se debe colocar *' %}"* para dejar por terminado el atributo href.

Por otro lado, si se desea dar permisos al uso de un archivo javascript dentro de HTML es muy similar a lo explicado anteriormente, lo único diferente es que se coloca en el atributo src, y no en el atributo href, todas las modificaciones de la ruta del archivo. Para un mejor entendimiento véase la siguiente figura.

```
<script src="{% static 'js/jquery-3.2.1.min.js' %}"></script>
```

Figura 56. Ejemplo de permiso de un archivo javascript en HTML

Por último, para dar permisos al uso de una imagen dentro de HTML se lo puede colocar tanto en el atributo src como en el atributo style, todas las modificaciones de la ruta del archivo, el cual se explicó con los archivos css. Para un mejor entendimiento véase los siguientes ejemplos de uso de imágenes en HTML.

```
<img src="{% static 'images/logo.png' %}" alt="">
```

Figura 57. Ejemplo de permiso de una imagen png en HTML con el atributo src

```
<div class="search_background" style="background-image:url({% static 'images/
    search_background.jpg' %})"></div>
```

Figura 58. Ejemplo de permiso de una imagen jpg en HTML con el atributo style

6.4. Templates

Los templates son los archivos que contienen el código de las páginas de su web site. Se trata, entre otras cosas, de determinar cómo las informaciones van a aparecer en la pantalla, incluso la posición de los placeholders los cuales son elementos predefinidos que deben configurarse para que puedan ser visualizados por los usuarios de su web site (Aleixo, 2019).

Para el proyecto se creó una carpeta con el nombre “templates” dentro de ella se creó cinco diferentes subcarpetas las cuales son: base, cámara, cinemática, gráficas y procesos.

Para permitir el uso de la carpeta templates es necesario realizar una modificación en el archivo settings.py el cual se encuentra dentro de la carpeta Diana4. Lo que se debe modificar es la dirección de los templates.

```
'DIRS': [],
```

Figura 59. Dirección de templates por defecto

Al crear un proyecto nuevo, el archivo settings.py no tiene ninguna dirección de templates por defecto, por lo cual se debe escribir dentro de los corchetes la dirección de los archivos HTML.

```
'DIRS': [os.path.join(BASE_DIR, 'templates')],
```

Figura 60. Dirección de la carpeta templates

En la figura anterior se observa la dirección de los templates, siendo ‘templates’, el nombre de la carpeta contenedora de los archivos HTML.

6.4.1. Subcarpeta base.

En la cual se encuentran los archivos HTML de registro, login, y la página principal más conocida como índex.

6.4.1.1. Archivo index.html.

En informática es común llamarle a la página inicial o principal como índex, por ende, el motivo del nombre de esta página. Para entender mejor lo que tiene la página principal se dividirá en bloques.

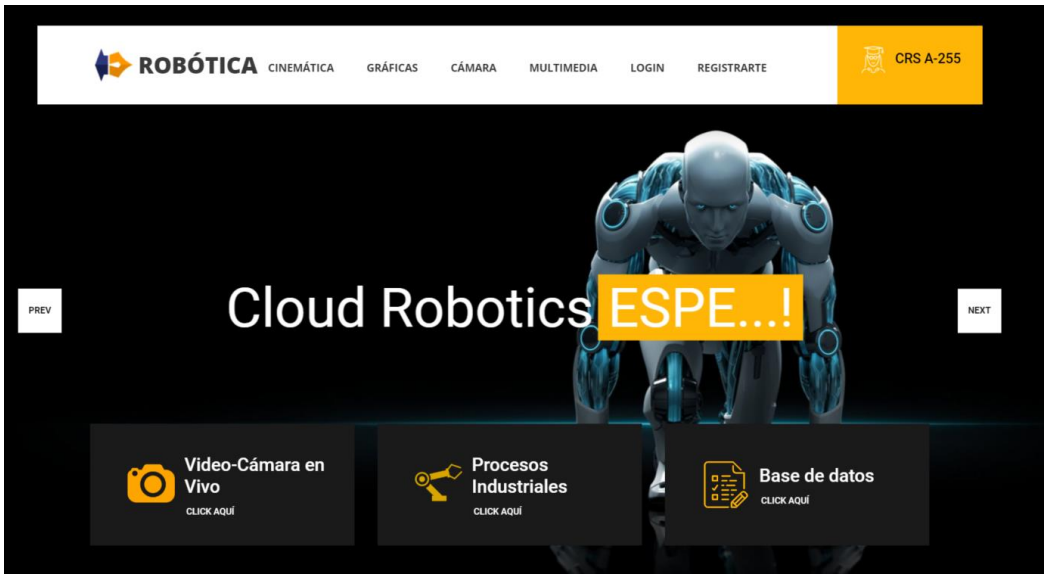

Figura 61. Primer bloque de la página principal index.html

En la figura anterior se observa un encabezado con diferentes alternativas a escoger, sin embargo, si se escoge las primeras cuatro alternativas contando de izquierda a derecha te llevarán

a una página web la cual indicará que primero se debe iniciar sesión; a pesar de ello a continuación se describe a donde te redirige cada una de las opciones del encabezado si se estuviera logueado, tener en cuenta que no se está logueado al sitio web debido a que en la parte derecha del encabezado se muestra el nombre de los robots y no un nombre de usuario.

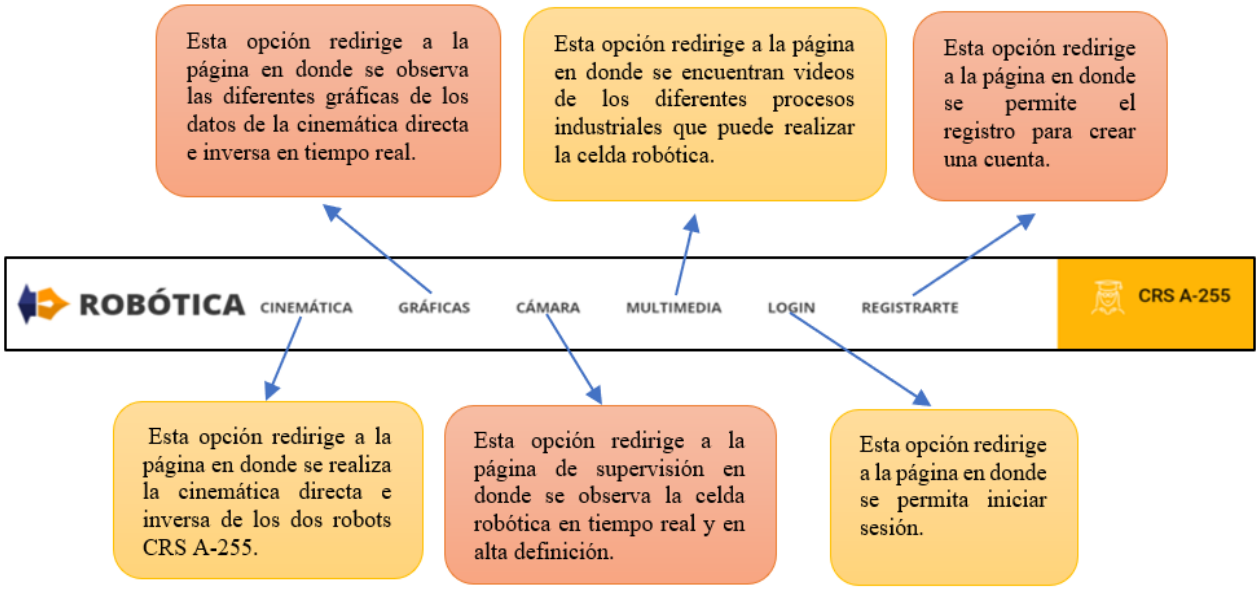

Figura 62. Esquema de navegación del encabezado

En la parte inferior del primer bloque de la página principal index.html se muestra tres opciones a escoger, a continuación, se detalla a donde se redirige cada una de las opciones, sin embargo, recuerda que debes estar logueado a la página web, para poder acceder a sus funcionalidades.

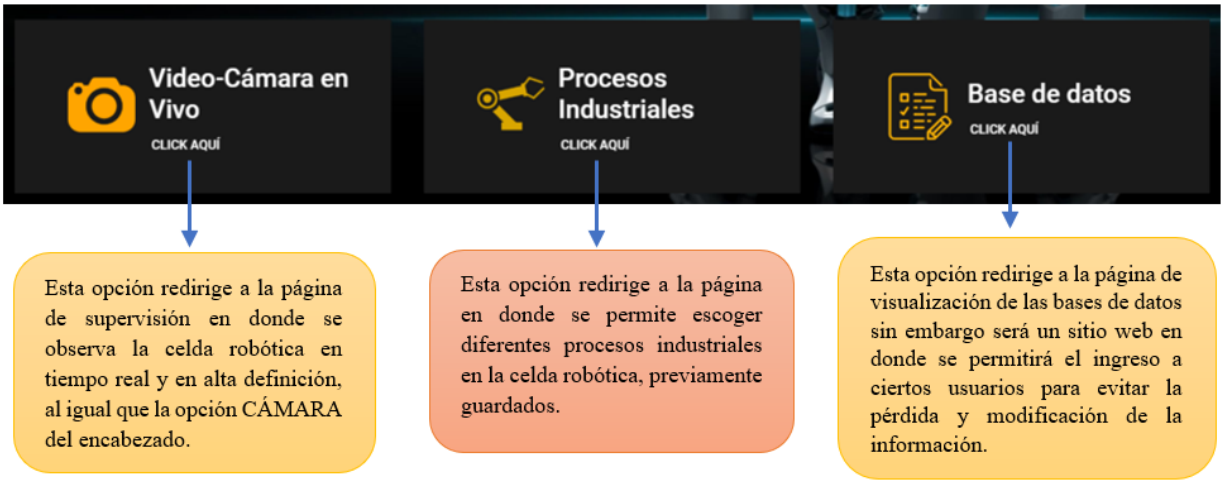

Figura 63. Esquema de navegación de las opciones inferiores el archivo index.html

Figura 64. Segundo bloque de la página principal index.html

En este bloque se observa que existe un botón llamado REGÍSTRATE AHORA el cual te redirige a la página web de registro la misma que está detallada en el ítem 6.4.1.3.

Además, a lado derecho se encuentra un formulario de inicio de sesión el mismo que puede ser llenado si ya se han registrado al sitio web anteriormente, dicho formulario está detallado en el ítem 6.4.1.4.

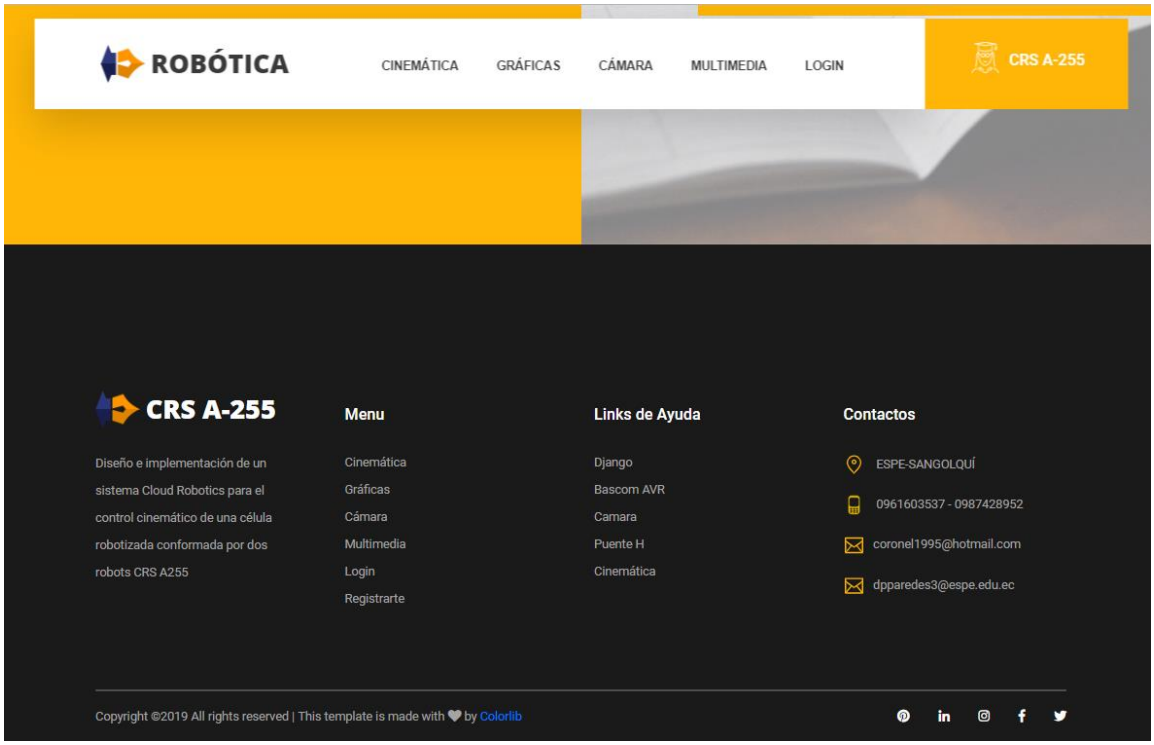

Figura 65. Tercer bloque de la página principal index.html

Se observa el pie de página, en donde se muestra el nombre del proyecto, un menú con las mismas opciones del encabezado, algunos links claves que fueron de ayuda para el desarrollo del proyecto, y finalmente el contacto de los creadores de la tesis.

6.4.1.2. Archivo index-login.html.

La página es muy similar con la anterior teniendo solo dos grandes diferencias, la primera es que se muestra el nombre del usuario logueado en la esquina superior derecha de la página, y la otra diferencia, es que permite el ingreso a todas las funcionalidades del sitio web, las mismas que fueron mencionadas en el ítem anterior.

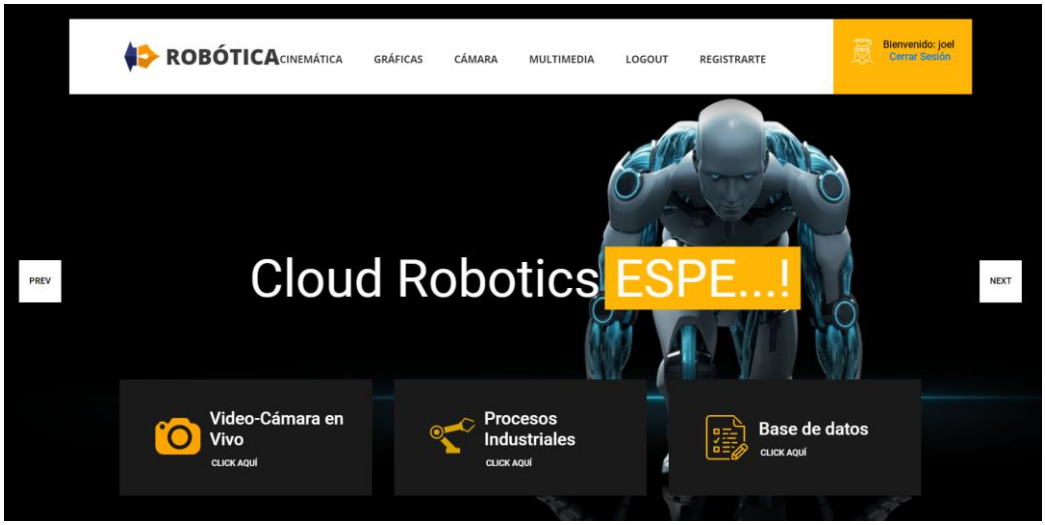

Figura 66. Página principal logueada index-login.html

6.4.1.3. Archivo registrarte.html.

Es el primer template que se debe crear, debido a que es donde el usuario va a registrarse para tener una cuenta y así poder acceder a todas las funcionalidades de la página web, para registrarse la persona deberá llenar los siguientes casilleros en blanco, con sus datos personales.

Figura 67. Formulario de registro

Para realizar el formulario de registro se escribió las siguientes líneas de código.

```
                        <h1 class="search_title"><font size=7><p style="color:rgb(242, 186, 13);">Registrate Ahora
                        </font></p></h1>
                        {% block content %}
                        <form method="post" autocomplete="off">
                        {% csrf_token %}
                        <input class="input_field search_form_name" type="text" name="first_name" placeholder="
                        Nombre" required="required" data-error="Usuario is required.">
                        <input class="input_field search_form_name" type="text" name="last_name" placeholder="
                        Apellido" required="required" data-error="Usuario is required.">
                        <input class="input_field search_form_name" type="text" name="username" placeholder="
                        Usuario" required="required" data-error="Usuario is required.">
                        <input class="input_field search_form_degree" type="email" name="email"  placeholder="
                        Correo Electrónico" required="required" data-error="Correo_Electrónico is required.">
                        <h7 class="search_content text-center"> <p style="color:black;">Marca el siguiente
                        casillero, si no eres un ROBOT </p></h7>
                        <input type="checkbox" name="is_staff"  value="TRUE" class="inline checkbox" id="checkbox1"
                        >
                        <input class="input_field search_form_degree" type="password" name="password1" placeholder=
                        "Contraseña"required="required" data-error="Contraseña is required.">
                        <input class="input_field search_form_degree" type="password" name="password2" placeholder=
                        "Confirmar Contraseña"required="required" data-error="Contraseña is required.">
                        <button class="search_submit_button trans_200" id="search_submit_button" type="submit"
                        value="Submit">Registrarme</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</form>
{% endblock %}
```

Figura 68. Formulario de registro

En el código anterior, existen algunos comandos o códigos exclusivos para Django como, por ejemplo, la línea *{% block content %}* la cual permite iniciar un bloque de contenido, por lo que se coloca al principio de todo el contenido del formulario registro; ya al final de todo el código se debe cerrar el bloque del contenido para lo cual se coloca la línea de código *{% endblock %}*.

Otro comando especial que usa Django es *{ csrf_token %}* el cual se debe colocar antes de usar un form.

Todos los registros realizados por los usuarios se guardarán en una base de datos de PostgreSQL.

6.4.1.4. Archivo login.html.

Es el que permite la creación de la página, en donde el operario tendrá que iniciar sesión colocando su usuario y contraseña. Antes de iniciar sesión el usuario tuvo que haberse creado una cuenta como se explicó en el ítem anterior.

Para crear la parte del login dentro de la página web se utilizó el siguiente código.

```
                <!-- LOGIN -->

            <div class="search_section d-flex flex-column align-items-center justify-content-center">
                <div class="search_background" style="background-image:url({% static 'images/course_1.jpg'%
                    })"></div>
                <div class="search_content text-center">
                        <h1 class="search_title"><font size=7><p style="color:rgb(242, 186, 13);">Iniciar
                        Sesión</font></p></h1>

                        {% block content %}
                        <form method="post">
                            {% csrf_token %}
                            <input name="username" class="input_field search_form_name" type="text"
                            placeholder="Usuario" required="required" data-error="Usuario is required.">
                            <input name="password" class="input_field search_form_degree" type="password"
                            autocomplete="off" placeholder="Contraseña" required="required" data-error="
                            Contraseña is required.">
                            <button id="search_submit_button" type="submit" class="search_submit_button
                            trans_200" value="Ingresar">Login</button>
                            </form>
                </div>
            </div>
        </div>
    </div>
</div>

                    </div>
                </div>


{% endblock %}
    <!-- Ultimo -->
```

Figura 69. Código para creación del login

Si se observa el código detenidamente se puede ver que tiene los mismos comandos especiales de Django, que se utilizaron en la creación del formulario del registro.

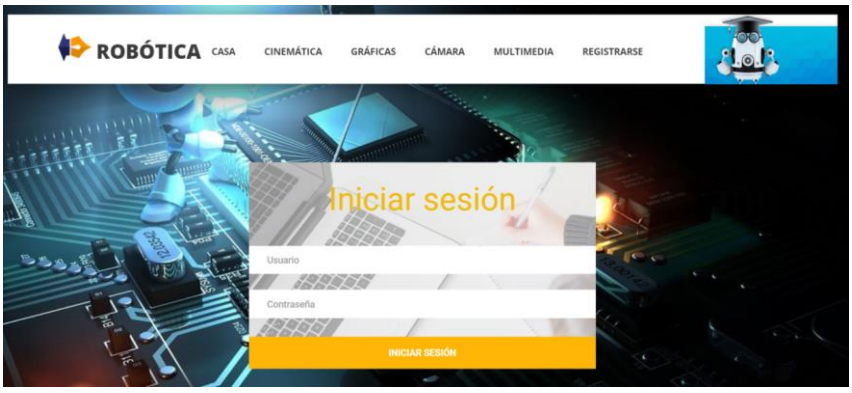

Figura 70. Página web login.html

6.4.1.5. Archivo First-login.html.

Al momento de acceder a cualquiera de las funcionalidades de la página, sin iniciar sesión anteriormente con alguna cuenta, se direccionará hacia una página web que te indicará que no puedes acceder a las funcionalidades del sitio web sin antes haberte logueado.

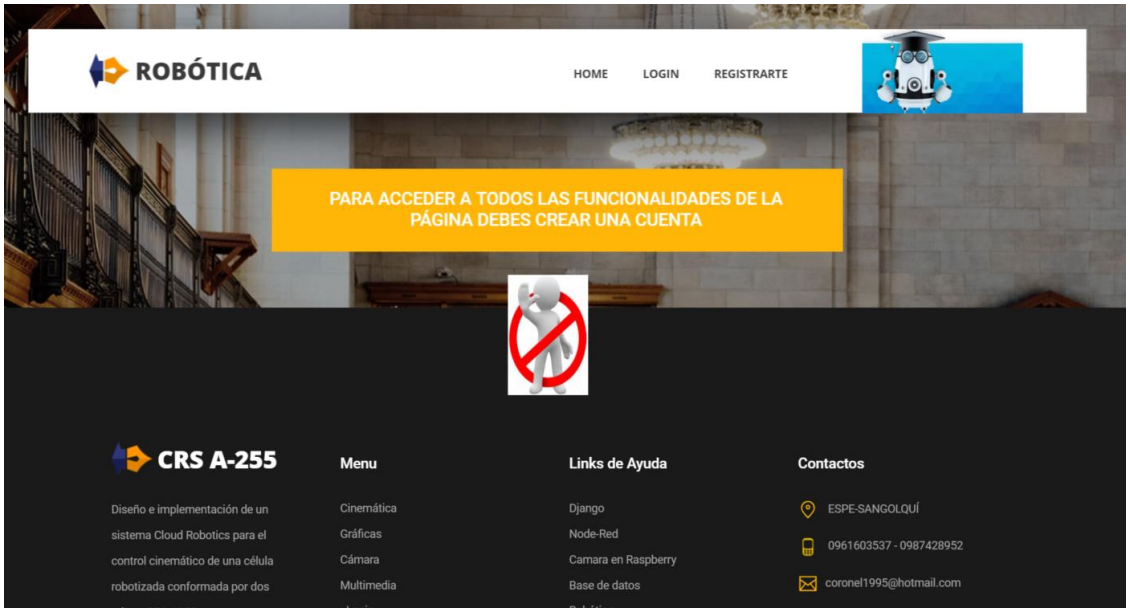

Figura 71. Página web First-login.html

6.4.2. Subcarpeta cinemática.

En la cual se encuentran los archivos HTML de la cinemática directa e inversa tanto individual como grupal, es decir para un solo robot o para ambos robots CRS A-255.

6.4.2.1. Archivo cinematica.html.

Permite la creación de la página web en donde el usuario ya logueado anteriormente, puede ejecutar la cinemática directa o inversa en uno o en ambos robots, de igual forma se puede llevar tanto al robot uno como al robot dos a la posición home para tener una perspectiva más clara, a continuación, se muestra la página html.

Figura 72. Posición Home de los robots uno y dos

Figura 73. Formulario para la cinemática directa individual

Figura 74. Formulario para la cinemática directa conjunta

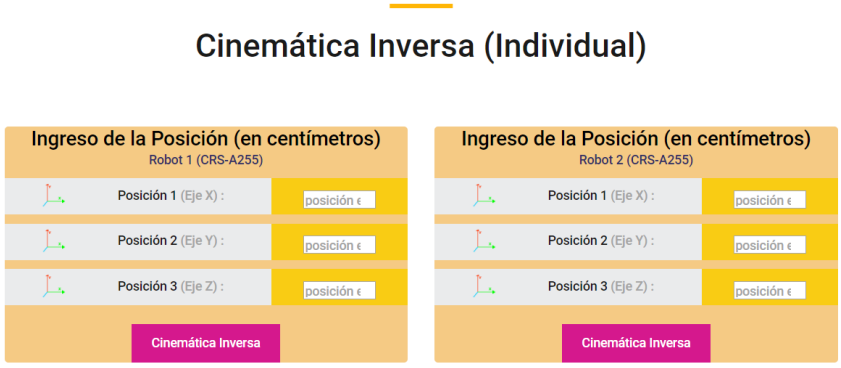

Figura 75. Formulario para la cinemática inversa individual

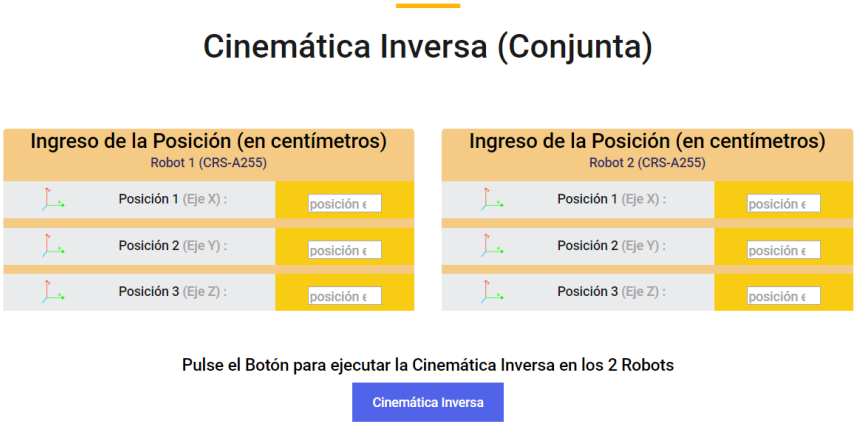

Figura 76. Formulario para la cinemática inversa conjunta

Para el diseño de cada formulario, se utilizó un form, dentro del cual se colocaron algunos inputs, cada input debe tener su nombre para identificarlos, debido a que el usuario ingresará información en cada input, la misma que debe ser enviada hacia otro archivo en donde se ejecutará las diferentes operaciones aritméticas, dicho archivo se encuentra en la app de cinemática, véase el ítem 6.5.1.3.

```
<form method="post" action='{% url "inversa1" %}'>
{% csrf_token %}
<!-- Caja respuesta 1 -->
<div class="price_box d-flex flex-row align-items-center">
    <div class="course_author_image">
        <img src="{% static 'images/eje.gif' %}" width="35" height="35"alt="https://unsplash.com/@mehdizadeh">
    </div>
    <div class="course_author_name">
        Posición 1 <span>(Eje X) : </span></div>
    <div class="course_price d-flex flex-column align-items-center justify-content-center"><span>
        <input name="posicion1" type="number" style="width:80px;height:20px" placeholder="posición en X" required="required" data-error=
        "Valor is required.">
    </span></div>
</div>

<!-- Caja respuesta 2 -->
<div class="price_box d-flex flex-row align-items-center">
    <div class="course_author_image">
        <img src="{% static 'images/eje.gif' %}" width="35" height="35"alt="https://unsplash.com/@mehdizadeh">
    </div>
    <div class="course_author_name">
        Posición 2 <span>(Eje Y) : </span></div>
    <div class="course_price d-flex flex-column align-items-center justify-content-center"><span>
        <input name="posicion2" type="number" style="width:80px;height:20px" placeholder="posición en Y" required="required" data-error=
        "Valor is required.">
    </span></div>
</div>

<!-- Caja respuesta 3 -->
<div class="price_box d-flex flex-row align-items-center">
    <div class="course_author_image">
        <img src="{% static 'images/eje.gif' %}" width="35" height="35"alt="https://unsplash.com/@mehdizadeh">
    </div>
    <div class="course_author_name">
        Posición 3 <span>(Eje Z) : </span></div>
    <div class="course_price d-flex flex-column align-items-center justify-content-center"><span>
        <input name="posicion3" type="number" style="width:80px;height:20px" placeholder="posición en Z" required="required" data-error=
        "Valor is required.">
    </span></div>
</div>

<!-- Boton Cinematica Inversa Robot1 -->
<br>
<div class="newsletter_form d-flex flex-md-row flex-column flex-xs-column align-items-center justify-content-center">
<button id="newsletter_submit" type="submit" class="newsletter_submit3_btn trans_300" value="Submit">Cinemática Inversa</button>
</div>
</form>
```

Figura 77. Código para la creación del formulario de la cinemática inversa para el Diana4

Para crear los otros formularios hay que seguir la misma sintaxis que se observa en la figura anterior, solo es necesario tener en cuenta el nombre de cada input, ya que este debe ser único para cada uno de los inputs de cada formulario.

6.4.2.2. Archivo cinematica_respuesta.html.

Permite la creación de la página web en donde el usuario puede observar la respuesta de la cinemática directa individual o conjunta, y el formulario o formularios que haya llenado. El cálculo de la cinemática directa e inversa se puede observar en los ítems 8.3.1 y 8.3.2, y la implementación de las mismas está descrito en el ítem 6.5.1.3.

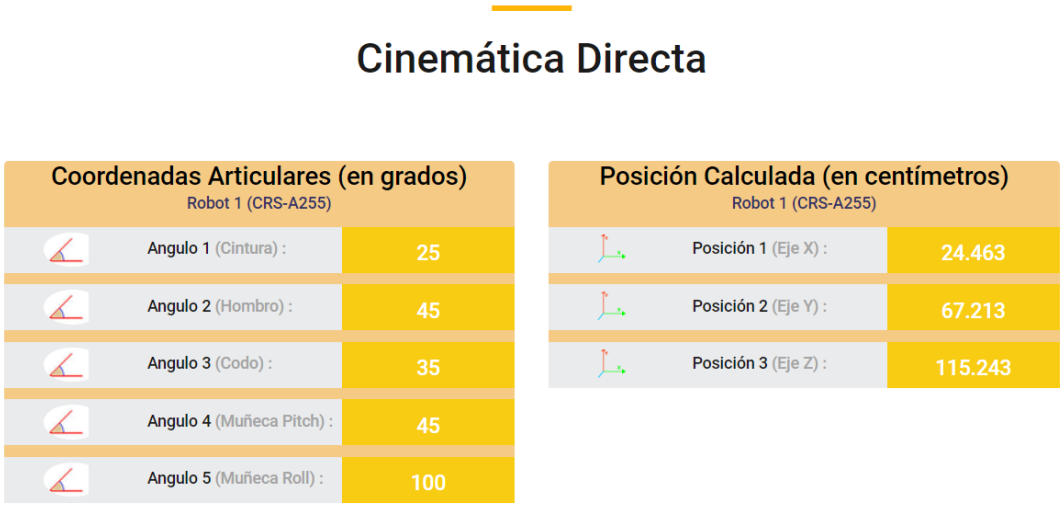

Figura 78. Resultado de la cinemática directa del Robot 1

Si se hubiese realizado la cinemática directa conjunta se tendría un formulario igual que el anterior para el primer robot y adicional los ángulos ingresados para el segundo robot y su respectiva posición calculada, como se observa en la siguiente figura.

Figura 79. Resultado de la cinemática directa del Robot 2

```
<!-- Popular Course Item -->
<div class="col-lg-5 course_box">
    <div class="card">
        <div class="card-body text-center">
            <div class="card-title"><h2 style="color:black;"> Posición Calculada (en centímetros)</h2></div>
            <div class="card-text">Robot 1 (CRS-A255)</div>
        </div>


        <!-- Caja respuesta 1 -->
        <div class="price_box d-flex flex-row align-items-center">
            <div class="course_author_image">
                <img src="{% static 'images/eje.gif' %}" width="35" height="35"alt="https://unsplash.com/@mehdizadeh">
            </div>
            <div class="course_author_name">
                Posición 1 <span>(Eje X) : </span></div>
            <div class="course_price d-flex flex-column align-items-center justify-content-center"><span>
                {{robot1.X0}}
                {{robots.X00}}
            </span></div>
        </div>

        <!-- Caja respuesta 2 -->
        <div class="price_box d-flex flex-row align-items-center">
            <div class="course_author_image">
                <img src="{% static 'images/eje.gif' %}" width="35" height="35"alt="https://unsplash.com/@mehdizadeh">
            </div>
            <div class="course_author_name">
                Posición 2 <span>(Eje Y) : </span></div>
            <div class="course_price d-flex flex-column align-items-center justify-content-center"><span>
                {{robot1.Y0}}
                {{robots.Y00}}
            </span></div>
        </div>

        <!-- Caja respuesta 3 -->
        <div class="price_box d-flex flex-row align-items-center">
            <div class="course_author_image">
                <img src="{% static 'images/eje.gif' %}" width="35" height="35"alt="https://unsplash.com/@mehdizadeh">
            </div>
            <div class="course_author_name">
                Posición 3 <span>(Eje Z) : </span></div>
            <div class="course_price d-flex flex-column align-items-center justify-content-center"><span>
                {{robot1.Z0}}
                {{robots.Z00}}
            </span></div>
        </div>


    </div>
</div>
```

Figura 80. Líneas de código para la visualización de la posición calculada del robot 1

En la anterior figura, se observa líneas de código utilizadas solo dentro del framework Django, las cuales son las siguientes: *{{robot1.X0}}*, *{{robots.X00}}*, *{{robot1.Y0}}*, *{{robots.Y00}}*, *{{robot1.Z0}}* y *{{robots.Z00}}*. Todas estas líneas de código mencionadas permiten obtener algún dato de cualquier tipo desde la app de cinemática.

Para muestra de todos los resultados de la cinemática directa individual o conjunta se utiliza el mismo sistema.

Para entender mejor hay que recordar que en el ítem 6.4.2.1. se menciona que se va a obtener datos ingresados por el usuario a través de inputs y enviárselos hacia la app de cinemática, dentro de esta app se ejecuta en un archivo de Python la cinemática directa y a través de etiquetas conocidas dentro de Django como contenido se envía la información, por lo cual estas etiquetas en la anterior figura son X0, X00, Y0, Y00, Z0 y Z00 en cambio la palabra que se encuentra a la izquierda de estas etiquetas, sirve para identificar de que vista viene cada uno de los datos, en el apartado de apps se explica de manera más detallada el archivo views donde se muestran todas las vistas.

A continuación, se muestra la manera de cómo se recibe los datos, se trabaja con ellos y finalmente se muestra los resultados.

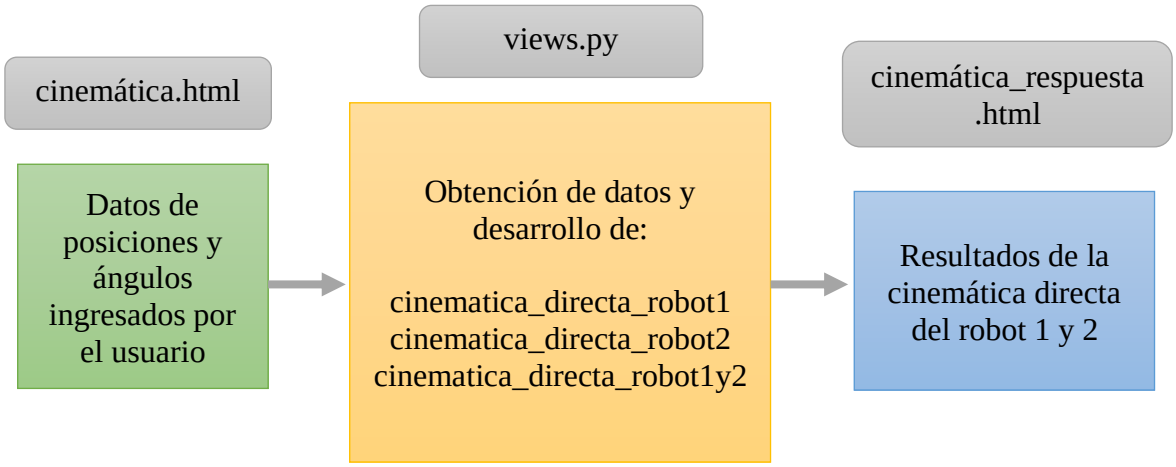

Figura 81. Flujo de datos para la obtención de la cinemática directa

6.4.2.3. Archivo cinematica_inversa_respuesta.html.

Permite la creación de la página web en donde el usuario puede observar la respuesta de la cinemática inversa individual o conjunta, y el formulario o formularios que haya llenado.

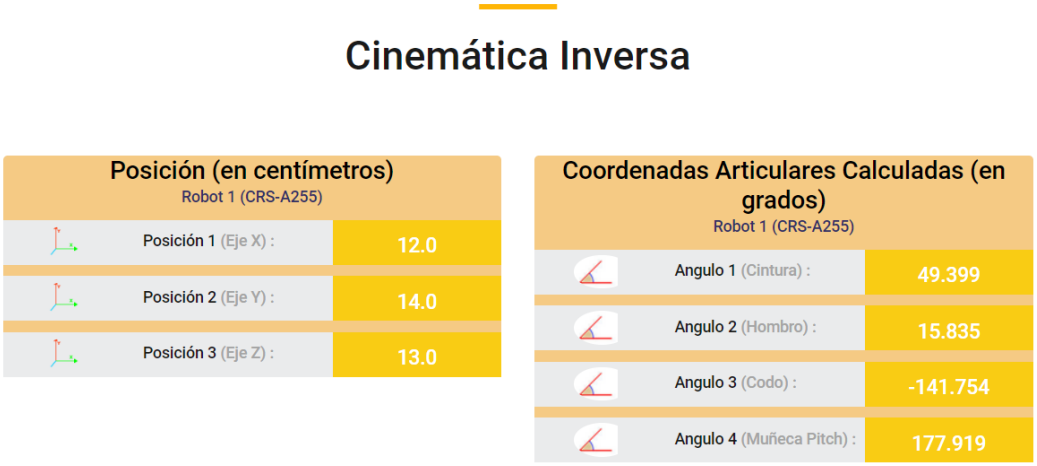

Figura 82. Resultado de la cinemática inversa del Robot 1

Si se hubiese realizado la cinemática inversa conjunta se tendría un formulario igual que el anterior para el primer robot y adicional las posiciones ingresadas para el segundo robot y sus respectivos ángulos calculados de cada articulación, como se observa en la siguiente figura.

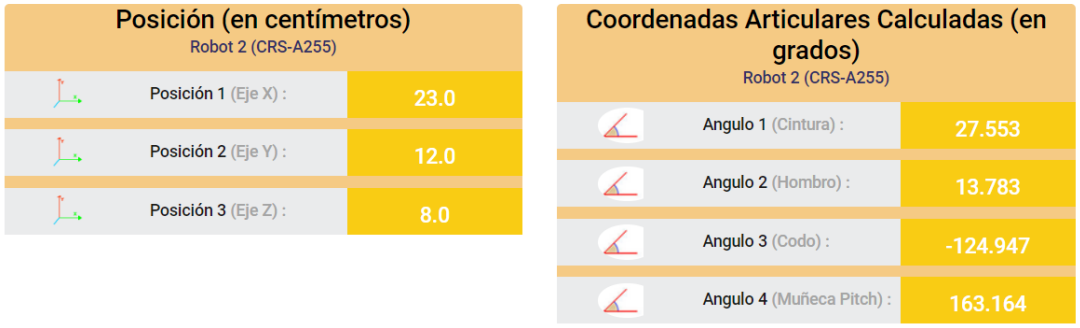

Figura 83. Resultado de la cinemática inversa del Robot 2

La creación de los formularios de respuesta para la cinemática inversa es de la misma forma que los formularios de respuesta de la cinemática directa, los cuales están explicados en el ítem anterior, su diferencia es el flujo de sus datos, para un mejor entendimiento observar la siguiente figura.

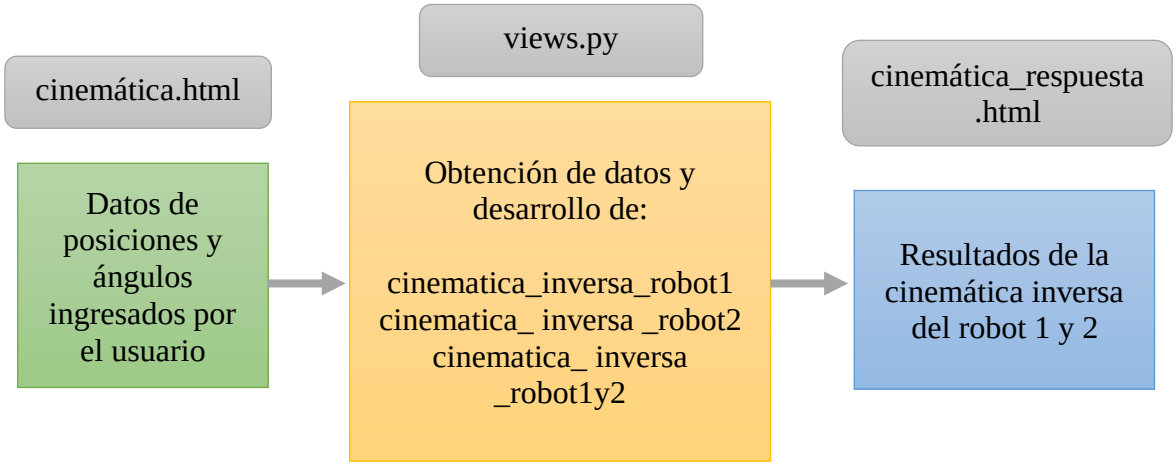

Figura 84. Flujo de datos para la obtención de la cinemática inversa

6.4.3. Subcarpeta gráficas.

En la cual se encuentran los archivos HTML de las gráficas estadísticas de la cinemática inversa y directa de los robots que conforman la celda robótica.

6.4.3.1. Archivo grafica_directa_robot1.html.

Permite la creación de la página web en donde el usuario ya logueado anteriormente, puede observar las gráficas de la cinemática directa del robot uno; tanto las tres gráficas de respuesta que son: coordenada en X, coordenada en Y, coordenada en Z; y también las cinco gráficas de entrada que son: ángulo de cintura, ángulo de hombro, ángulo de codo, ángulo muñeca pitch y ángulo muñeca roll. Para tener una perspectiva más clara, a continuación, se muestra las ocho gráficas mencionadas anteriormente.

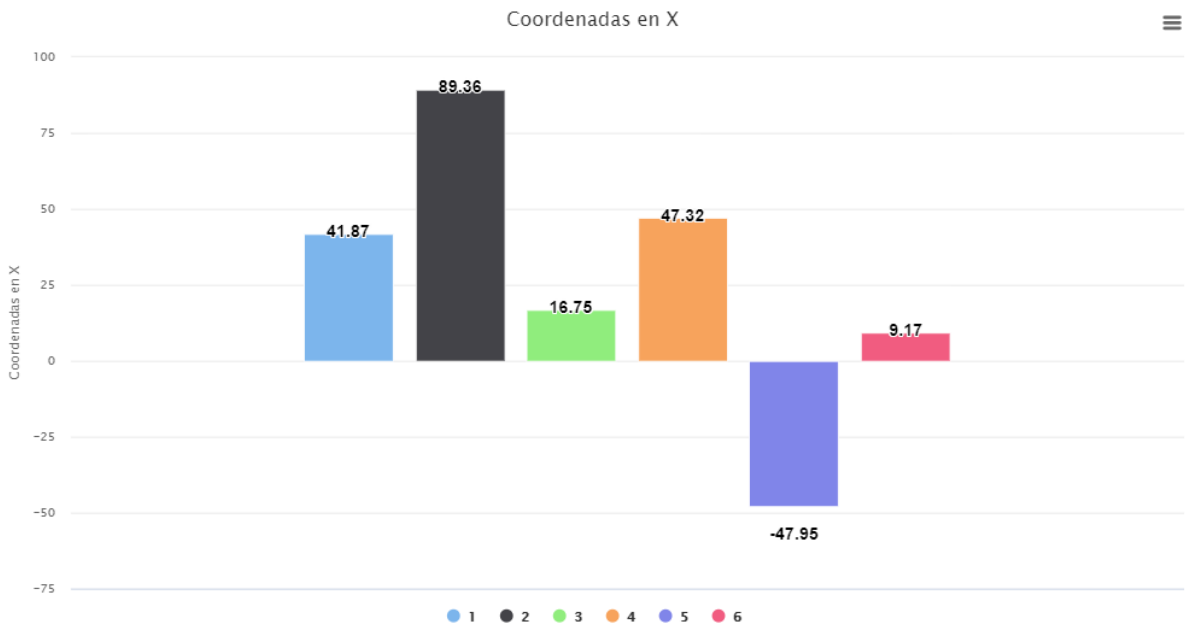

Figura 85. Coordenadas en X de la cinemática directa del robot uno

En la figura anterior se observa seis datos diferentes representados en columnas verticales, dichos datos corresponden a la respuesta en la coordenada en X al momento de realizar la cinemática directa del robot uno, por lo que se puede entender que se ha realizado la ejecución de la cinemática directa en el primer robot seis veces, se visualiza que en la primera ejecución el resultado de la coordenada en X es 41.87 y en la última ejecución el resultado en la coordenada en X es 9.17, de igual forma cada una de las gráficas cuenta con la opción de descargar tanto la imagen como sus datos en diferentes formatos como: JPEG, PNG, PDF, CSV, XLSX.

En las siguientes dos figuras se puede visualizar la respuesta de las coordenadas en Y y Z; de igual forma tienen seis datos cada una, por lo que se menciona anteriormente que la cinemática directa en el primer robot se ejecutó seis veces.

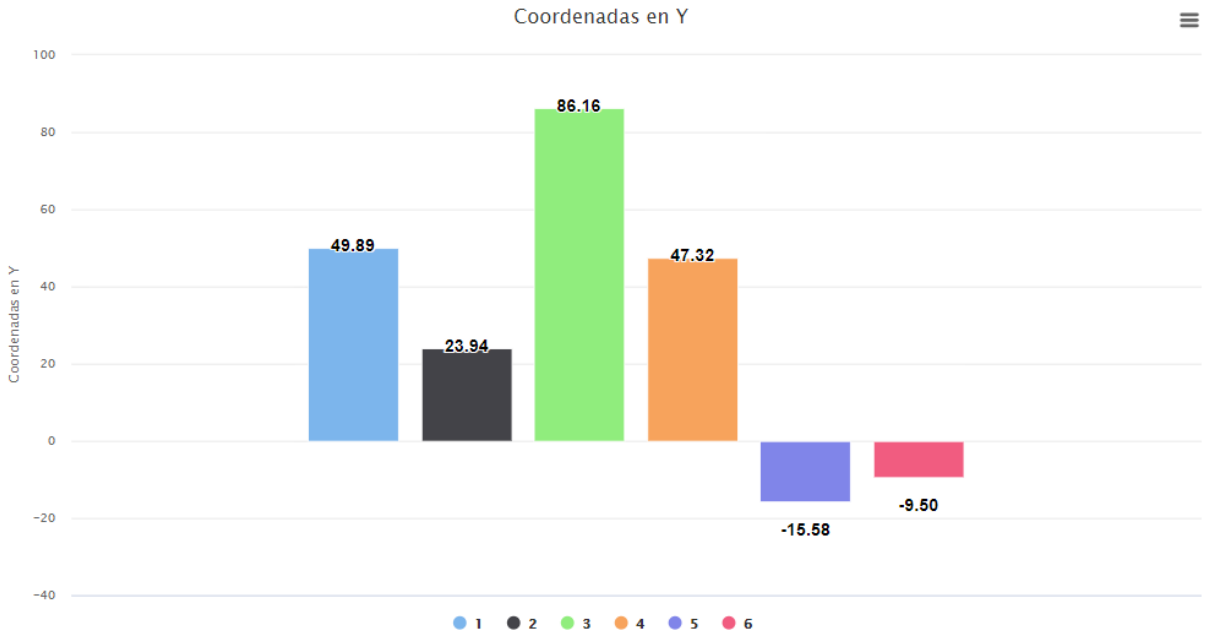

Figura 86. Coordenadas en Y de la cinemática directa del robot uno

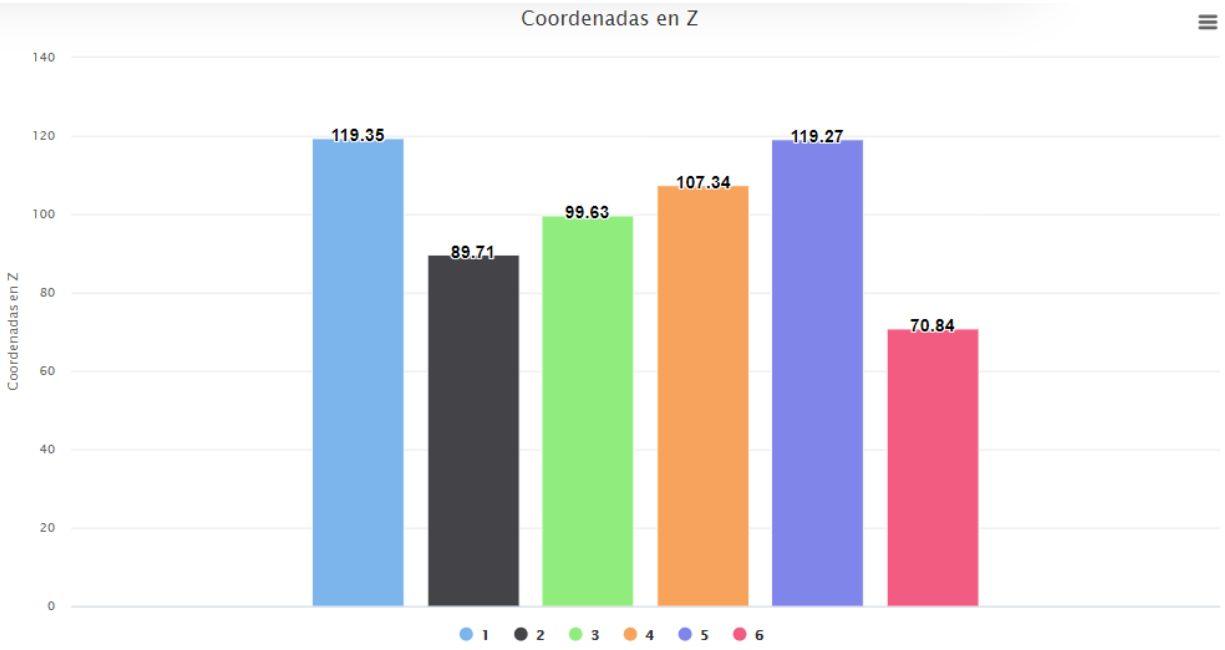

Figura 87. Coordenadas en Z de la cinemática directa del robot uno

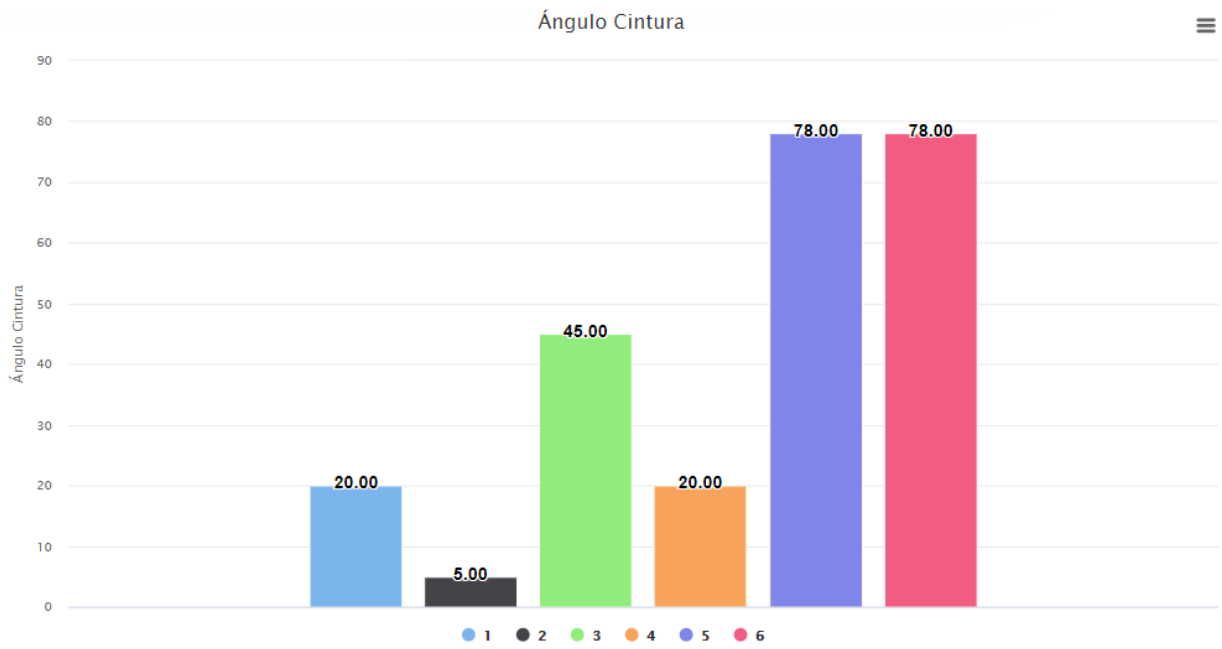

Figura 88. Ángulo cintura de la cinemática directa del robot uno

En la figura anterior se observa seis datos diferentes representados en columnas verticales, dichos datos corresponden al valor del ángulo de la cintura del primer robot, estos datos son ingresados por el usuario, dependiendo el valor en grados que deseen que el ángulo de la cintura adopte. Se visualiza que en la primera ejecución el usuario ingresó el valor de 20 grados para el ángulo de la cintura y en la última ejecución el usuario ingresó el valor de 78 grados.

En las siguientes cuatro figuras se puede visualizar el ingreso de los valores de los ángulos del hombro, codo, muñeca pitch y muñeca roll del primer robot respectivamente; de igual forma tienen seis datos cada una de las gráficas.

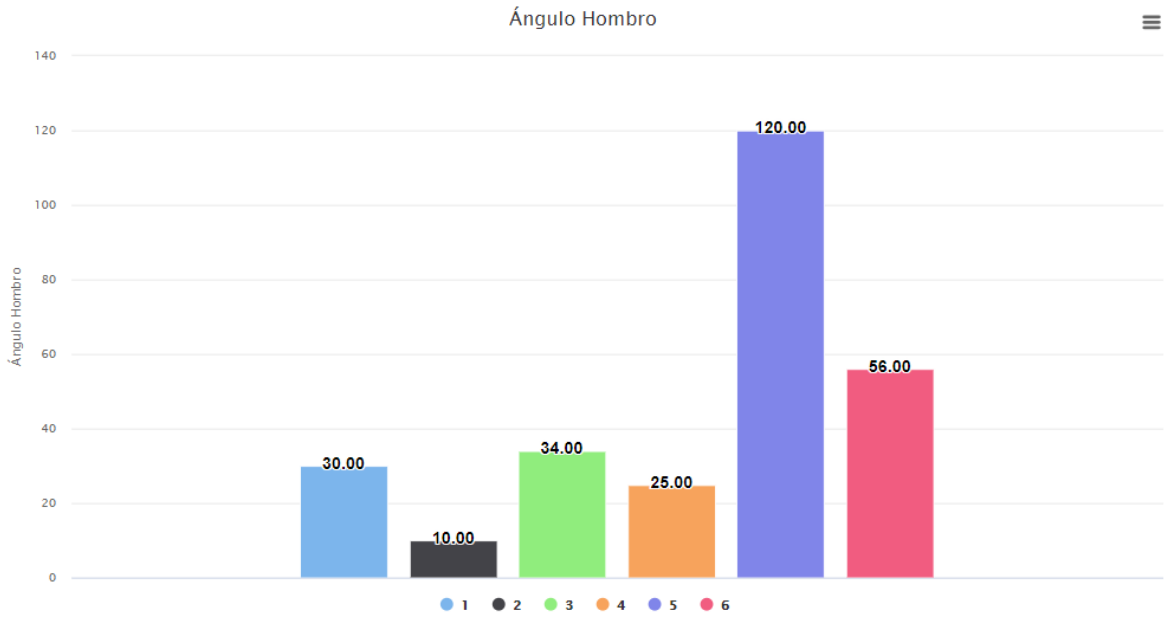

Figura 89. Ángulo hombro de la cinemática directa del robot uno

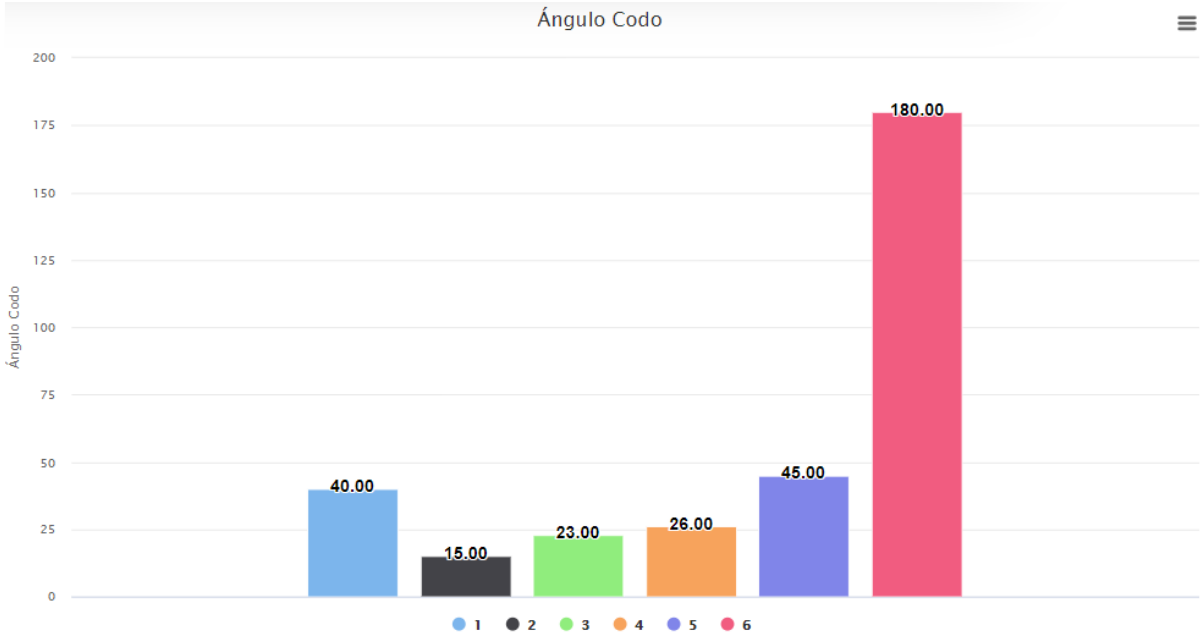

Figura 90. Ángulo codo de la cinemática directa del robot uno

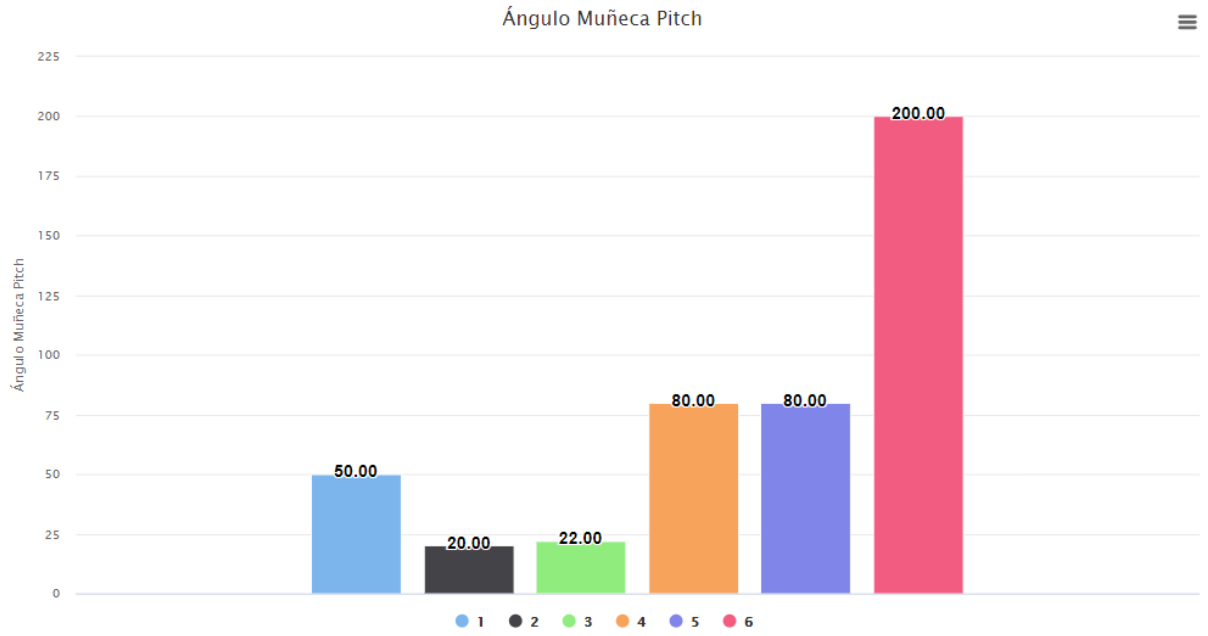

Figura 91. Ángulo muñeca pitch de la cinemática directa del robot uno

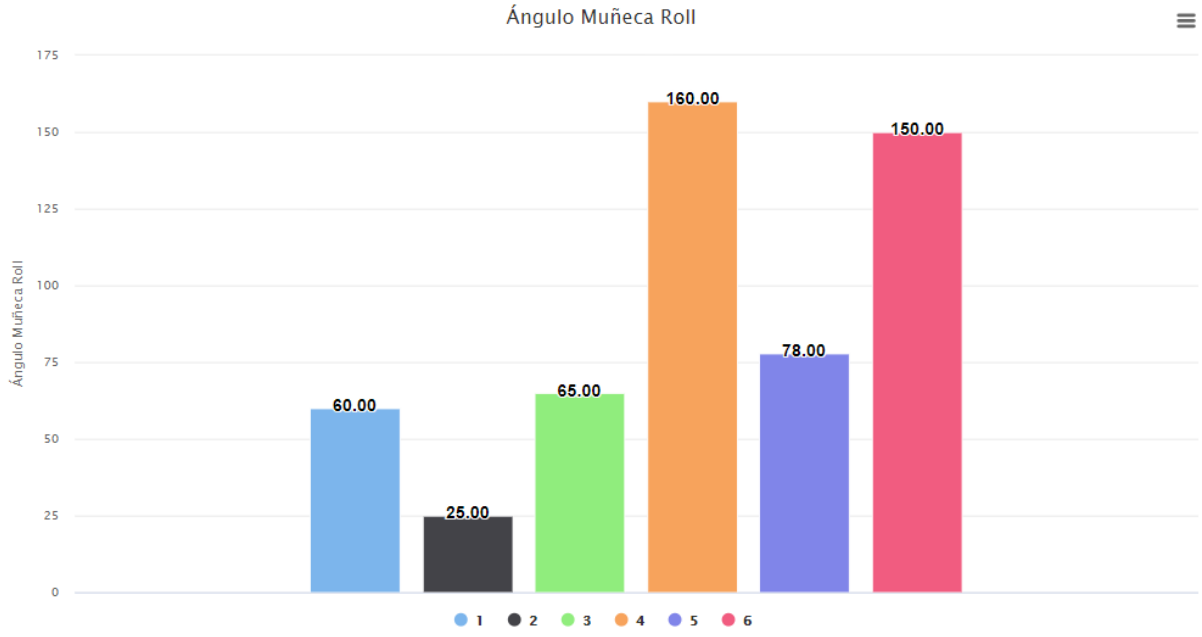

Figura 92. Ángulo muñeca roll de la cinemática directa del robot uno

Para la creación de las diferentes gráficas estadísticas, se utilizó la librería HighCharts, la misma que está escrita en Javascript, por lo cual lo primero que se debe hacer, es dar permisos al uso de los archivos Javascript dentro de HTML, como se mencionó en el ítem 6.3.

```
<script src="https://code.highcharts.com/highcharts.js"></script>
<script src="https://code.highcharts.com/modules/series-label.js"></script>
<script src="https://code.highcharts.com/modules/exporting.js"></script>
<script src="https://code.highcharts.com/modules/export-data.js"></script>
```

Figura 93. Permiso de los diferentes archivos de HighCharts

Se observa que la ruta de los diferentes archivos de HighCharts, deben ser llamados desde el sitio oficial de HighCharts a través de un link.

Como se trata de la cinemática directa, se tiene tres gráficas de entrada y cinco gráficas de salida. Para colocar el tamaño de las mismas se realiza con las siguientes líneas de programación.

```
<div id="salida1" style="width:100%; height:600px;"></div>
<div id="salida2" style="width:100%; height:600px;"></div>
<div id="salida3" style="width:100%; height:600px;"></div>
```

Figura 94. Tamaño de las gráficas de salida de la cinemática directa del robot uno

```
<div id="entrada1" style="width:100%; height:600px;"></div>
<div id="entrada2" style="width:100%; height:600px;"></div>
<div id="entrada3" style="width:100%; height:600px;"></div>
<div id="entrada4" style="width:100%; height:600px;"></div>
<div id="entrada5" style="width:100%; height:600px;"></div>
```

Figura 95. Tamaño de las gráficas de entrada de la cinemática directa del robot uno

```
<script>
Highcharts.chart('entrada1', {
    chart: {
        type: 'column'
    },
    title: {
        text: 'Ángulo Cintura'
    },
    xAxis: {
        categories: ['']
    },
    yAxis: {
        title: {
            text: 'Ángulo Cintura'
        }
    },

    plotOptions: {
        line: {
            dataLabels: {
                enabled: true
            },
            enableMouseTracking: false
        }
    },
    series: [
            {% for directa1 in robot1_directa %}
    {
        name: '{{directa1.id}}',
        data: [{{directa1.Ang_Cintura_R1}}],

        dataLabels: {
        enabled: true,
        rotation: 0,
        color: 'black',
        align: 'center',
        format: '{point.y:.2f}', // one decimal
        y: 10, // 10 pixels down from the top
        style: {
        fontSize: '16px',
        fontFamily: 'Arial, sans-serif'
        }
    }
    },
    {% endfor %}
    ]
});
</script>
```

Figura 96. Programación de la entrada 1 de la cinemática directa del robot uno

Para programar las gráficas en primer lugar se debe colocar el tipo de gráfica que se desee, en este caso es de tipo columna; después de proceder a escribir el título del gráfico en este caso Ángulo Cintura; a continuación, se coloca tanto el titulo para el eje “x” y eje “y” del gráfico, en este caso solo colocamos título en el eje “y”, el cual es Ángulo Cintura, después se escoge algunas opciones del gráfico en donde se coloca que exista líneas horizontales en la gráfica, y no haya seguimiento de mouse en la gráfica, por último, la parte de series en donde se escribe *{% for directa1 in robot1_directa %}* en donde directa1 es el nombre del objeto que obtendrá los datos, de la tabla de datos llamada Directa_Robot1, en cambio robot1_directa es donde esta guardado el objeto directa1. Además, la línea de código, *name: '{{directa1.id}}',* indica que del objeto directa1 va a escoger la columna llamada id. En cambio, para los datos se los llama con la siguiente línea de código, *data: [{{directa1.Ang_Cintura_R1}}],* en donde se llama del objeto directa1 la columna Ang_Cintura_R1, en donde se encuentran todos los datos de ángulo de cintura del robot uno. Finalmente se escribe diferentes opciones del gráfico de la columna como es el color, la alineación del texto de arriba de la columna, el formato, el estilo de letra, etc. No hay que olvidar que todo for se debe finalizar, por lo que se escribe casi al final la siguiente línea de código *{% endfor %}.*

Para las otras entradas y salidas se realiza de la misma forma lo único diferente será al momento de llamar los datos se llamará a una distinta columna de la base de datos Directa_Robot1.

6.4.3.2. Archivo grafica_directa_robot2.html.

Permite la creación de la página web en donde el usuario ya logueado anteriormente, puede observar las gráficas de la cinemática directa del robot dos; tanto las tres gráficas de respuesta que son: coordenada en X, coordenada en Y, coordenada en Z; y también las cinco gráficas de entrada que son: ángulo de cintura, ángulo de hombro, ángulo de codo, ángulo muñeca pitch y ángulo

muñeca roll. Las gráficas son de la misma forma que se muestran en el ítem anterior y su creación sigue los mismos patrones, lo único diferente será que el llamado a la base de datos se redirigirá a una distinta tabla de datos que contenga la información de la cinemática directa del robot dos.

6.4.3.3. Archivo grafica_inversa_robot1.html.

Permite la creación de la página web en donde el usuario ya logueado anteriormente, puede observar las gráficas de la cinemática inversa del robot uno; tanto las cinco gráficas de respuesta que son: ángulo de cintura, ángulo de hombro, ángulo de codo, ángulo muñeca pitch y ángulo muñeca roll; y también las tres gráficas de entrada que son: coordenada en X, coordenada en Y, coordenada en Z. Las gráficas son de la misma forma que se muestran en el ítem 6.4.3.1. y su creación sigue los mismos patrones, lo único diferente será que el llamado a la base de datos se redirigirá a una distinta tabla de datos que contenga la información de la cinemática inversa del robot uno.

6.4.3.4. Archivo grafica_inversa_robot2.html.

Permite la creación de la página web en donde el usuario ya logueado anteriormente, puede observar las gráficas de la cinemática inversa del robot dos; tanto las cinco gráficas de respuesta que son: ángulo de cintura, ángulo de hombro, ángulo de codo, ángulo muñeca pitch y ángulo muñeca roll; y también las tres gráficas de entrada que son: coordenada en X, coordenada en Y, coordenada en Z. Las gráficas son de la misma forma que se muestran en el ítem 6.4.3.1. y su creación sigue los mismos patrones, lo único diferente será que el llamado a la base de datos se redirigirá a una distinta tabla de datos que contenga la información de la cinemática inversa del robot dos.

6.4.4. Subcarpeta cámara.

En la cual se encuentra el archivo HTML de supervisión de la cámara.

6.4.4.1. Archivo camara1.html.

Permite la creación de la página web en donde el usuario puede monitorear a la celda robótica.

Figura 97. Página web cámara1.html

Lo más importante dentro del código para la creación del sitio web de supervisión, es el uso de un iframe.

Un iframe es un documento de una web que se inserta en otra página web. Se trata de un elemento que puede albergar cualquier tipo de contenido en su interior y que ayuda, entre otras

cosas, a ampliar el mensaje que se ofrece con fuentes externas o con material complementario que pueda resultar de interés para el usuario que entra (NeoAttack, 2018).

```
<div class="embed-container">
<center><iframe src="https://isa.ezvizlife.com/camera/cameraAction!goCameraInfo.action?cameraId=cd8b085ca26e4679824d9530676928c1&infoType=real" width="1000"
 height="700" allowfullscreen frameborder="0">
</iframe></center>
</div>
```

Figura 98. Uso del iframe en la página web camara1.html

El link colocado en el iframe, se trata de la página oficial de EZVIZ la cual es la marca de la cámara que se utiliza para la supervisión de la celda robótica, en el siguiente capítulo se explica a detalle el sitio web ezviz life y cómo usarlo para la monitorización de los robots CRS A255.

6.4.5. Subcarpeta procesos.

En la cual se encuentra los archivos HTML de los procesos industriales y de multimedia.

6.4.5.1. Archivo procesos.html.

Al principio del proyecto se planteó la realización de un proceso industrial utilizando dos robot CRS A-255, moviendo unas cajas de una mesa a otra para que finalmente un pistón las despache, lamentablemente se presentaron inconvenientes mecánicos (acoples de los encoders hacia los motores) en uno de los dos brazos robóticos de igual forma no hubo la disponibilidad del uso de una segunda mesa dentro de las instalaciones de la Universidad por lo que se optó por realizar tres diferentes procesos en los cuales la ejecución de todo el proceso se la realiza con un solo brazo robótico.

Dentro de este archivo se puede seleccionar tres diferentes procesos, el primero es el empaquetado de piezas de Jenga, el segundo es la simulación del transporte de neumáticos y el último proceso es la colocación de huevos en la cubeta. Para ejecutar cualquiera de los tres procesos

existe un botón debajo de cada una de las imágenes representativas de cada proceso como se observa en la siguiente figura.

Figura 99. Entorno Web de los procesos industriales

La programación del archivo procesos.html es muy similar, a la programación que se realizó en al archivo cinematica.html, el cual se explica en el ítem 6.4.2.1.

En la siguiente figura se puede observar la programación para el primer proceso industrial.

```
<!-- Proceso Uno JENGA-->
<div class="col-lg-5 course_box">
    <div class="card">
        <div class="card-body text-center">
            <div class="card-title"><h2 style="color:#67356E;">Transporte de piezas de JENGA</h2></div>
            <img src="{% static 'images/jenga.png' %}" width="350" height="250">
        </div>

        <form method="post" action='{% url "proceso1" %}'>
        {% csrf_token %}

        <!-- Boton Proceso 1 -->
        <br>
        <div class="newsletter_form d-flex flex-md-row flex-column flex-xs-column align-items-center
        justify-content-center">
        <button id="newsletter_submit" type="submit" class="newsletter_submit1_btn trans_300" value="submit">
        Ejecutar Proceso</button>
        </div>
        </form>
    </div>
</div>
```

Figura 100. Programación HTML para el primer proceso industrial (Transportes de piezas de JENGA)

Se tendrá tres formularios debido a que son tres procesos industriales. Para el diseño de cada formulario, se utilizó un form, dentro del mismo hay que colocar la línea de código *{% csrf_token %}* para no tener inconvenientes al momento de ejecutar el archivo, de igual manera se coloca un elemento button el cual permite la creación de un botón, dicho botón tiene la finalidad de redirigir internamente hacia una función dentro del archivo views.py de la app de procesos, en donde se encuentran las líneas de programación que permiten que el robot realice dicho proceso.

Para la creación de los otros formularios se sigue la misma sintaxis que se observa en la figura anterior, solo es necesario tener en cuenta a que función del archivo views.py de la app de procesos debe redirigirse.

6.4.5.2. Archivo multimedia.html.

El archivo multimedia.html permite la visualización de los tres diferentes procesos explicados en el ítem anterior, a través de videos pregrabados de cada uno de los mismos. El sitio web se visualiza de la siguiente manera.

Figura 101. Visualización de los tres procesos industriales en la página web

Para la programación del archivo multimedia.html se utiliza el elemento video el cual permite incrustar un video dentro del documento HTML.

```
<!-- VIDEO 1 JENGA-->
<div class="col-lg-5 course_box">
    <div class="card">
        <div class="card-body text-center">
            <div class="card-title"><h2 style="color:#67356E;">Transporte de piezas de JENGA</h2></div>
            <video src="{% static 'videos/proceso1.mp4' %}" type="video/mp4" width=350  height=250 controls poster="{%
            static 'images/jenga.png' %}">
            Lo sentimos. Este vídeo no puede ser reproducido en tu navegador.<br>
            </video>
        </div>
    </div>
</div>
```

Figura 102. Programación HTML para la visualización del primer proceso (Transportes de piezas de JENGA)

Las mismas líneas de programación se las realiza para los otros dos procesos con la única diferencia que cambia el título del video, y la ruta de alojamiento del video.

6.5. Instalación de apps

Para tener un orden adecuado en el diseño de nuestra página web, Django permite la creación de apps, las cuales son módulos con funcionalidades dentro del proyecto. En palabras más sencillas sería la distribución que se quiere dar a la página web.

En el proyecto se ha creado una carpeta llamada “apps” dentro de la cual se ha generado las diferentes aplicaciones o apps. Para crear una aplicación en Django se ingresa el siguiente comando en el terminal.

```
$ python manage.py startapp cinematica
```

El comando anterior debe ser ejecutado dentro de la carpeta “apps”; la palabra cinemática dentro del comando, es el nombre de la aplicación a crear.

Al momento de crear una aplicación, se genera una carpeta con el nombre de la aplicación, dentro de ella se encuentran seis archivos: __init__.py, admin.py, apps.py, models.py, tests.py, views.py. de los cuales solo se utiliza models.py, admin.py y views.py.

Dentro del archivo models.py se coloca las clases, en otras palabras, las tablas de base de datos que se vayan a usar, en el archivo models.py se coloca las bases de datos que se desee visualizar en el entorno de administración de Django. Finalmente, en el archivo views.py es donde se coloca todas las vistas, es decir todo lo que se desea mostrar dependiendo la url a la que se redirija.

Para el proyecto se ha generado cuatro diferentes aplicaciones las cuales son: cinemática, usuario, cámara, y procesos, las mismas que deben ser definidas en el archivo settings.py, lo cual permitirá la comunicación con todos los archivos que conforma el framework del proyecto.

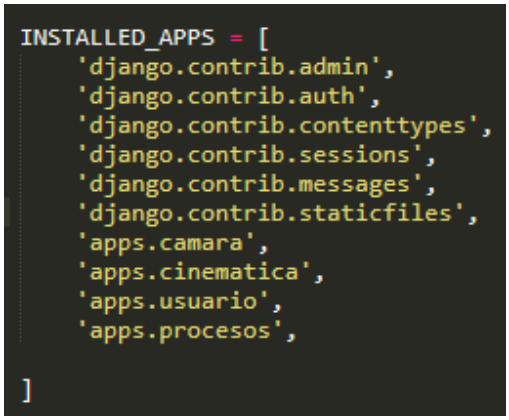

Figura 103. Definición de las apps instaladas

En la figura anterior se observa que en las últimas líneas de código se define las apps creadas, para el Proyecto.

6.5.1. Aplicación cinemática.

Dentro de esta aplicación estará todo lo relacionado a la cinemática directa e inversa y el control cinemático de los dos robots CRS-A255.

6.5.1.1. Archivo models.py.

Dentro de este archivo se creará cuatro diferentes clases llamadas: Directa_Robot1, Directa_Robot2, Inversa_Robot1, Inversa_Robot2; en donde se guardarán las coordenadas y ángulos de los diferentes robots al momento de realizar su cinemática inversa o directa individual, o grupal.

```
class Directa_Robot1(models.Model):
    Cinematica_R1 = models.CharField(max_length=30)
    Ang_Cintura_R1 = models.CharField(max_length=30)
    Ang_Codo_R1 = models.CharField(max_length=30)
    Ang_Hombro_R1= models.CharField(max_length=30)
    Ang_Muneca_Pitch_R1 = models.CharField(max_length=30)
    Ang_Muneca_Roll_R1 = models.CharField(max_length=30)
    Coord_X_R1 = models.CharField(max_length=30)
    Coord_Y_R1 = models.CharField(max_length=30)
    Coord_Z_R1 = models.CharField(max_length=30)
```

Figura 104. Clase Directa_Robot1 de la app cinemática

```
class Directa_Robot2(models.Model):
    Cinematica_R2 = models.CharField(max_length=30)
    Ang_Cintura_R2 = models.CharField(max_length=30)
    Ang_Codo_R2 = models.CharField(max_length=30)
    Ang_Hombro_R2 = models.CharField(max_length=30)
    Ang_Muneca_Pitch_R2 = models.CharField(max_length=30)
    Ang_Muneca_Roll_R2 = models.CharField(max_length=30)
    Coord_X_R2 = models.CharField(max_length=30)
    Coord_Y_R2 = models.CharField(max_length=30)
    Coord_Z_R2 = models.CharField(max_length=30)
```

Figura 105. Clase Directa_Robot2 de la app cinemática

```
class Inversa_Robot1(models.Model):
    Cinematica_IR1 = models.CharField(max_length=30)
    Coord_X_IR1 = models.CharField(max_length=30)
    Coord_Y_IR1 = models.CharField(max_length=30)
    Coord_Z_IR1 = models.CharField(max_length=30)
    Ang_Cintura_IR1 = models.CharField(max_length=30)
    Ang_Codo_IR1 = models.CharField(max_length=30)
    Ang_Hombro_IR1 = models.CharField(max_length=30)
    Ang_Muneca_Pitch_IR1 = models.CharField(max_length=30)
    Ang_Muneca_Roll_IR1 = models.CharField(max_length=30)
```

Figura 106. Clase Inversa_Robot1 de la app cinemática

```
class Inversa_Robot2(models.Model):
    Cinematica_IR2 = models.CharField(max_length=30)
    Coord_X_IR2 = models.CharField(max_length=30)
    Coord_Y_IR2 = models.CharField(max_length=30)
    Coord_Z_IR2 = models.CharField(max_length=30)
    Ang_Cintura_IR2 = models.CharField(max_length=30)
    Ang_Codo_IR2 = models.CharField(max_length=30)
    Ang_Hombro_IR2 = models.CharField(max_length=30)
    Ang_Muneca_Pitch_IR2 = models.CharField(max_length=30)
    Ang_Muneca_Roll_IR2 = models.CharField(max_length=30)
```

Figura 107. Clase Inversa_Robot2 de la app cinemática

Todas las clases creadas no son más que las tablas que se guardarán en la base de datos PostgreSQL, las cuales permitirán desarrollar las diferentes gráficas estadísticas.

6.5.1.2. Archivo admin.py.

En este archivo se registra los modelos, es decir las clases que se mencionaron en el anterior ítem, se realiza dicho registro para poder observar estas tablas de datos desde el sitio oficial de administración de base de datos de Django el cual está vinculado hacia la página principal del sitio web.

```
from django.contrib import admin

# Register your models here.
from .models import Directa_Robot1
from .models import Directa_Robot2
from .models import Inversa_Robot1
from .models import Inversa_Robot2


admin.site.register(Directa_Robot1)
admin.site.register(Directa_Robot2)
admin.site.register(Inversa_Robot1)
admin.site.register(Inversa_Robot2)
```

Figura 108. Archivo admin.py de la app de cinemática

Se puede observar que lo primero que toca hacer es importar el modelo desde el archivo models.py y posteriormente registrar cada uno de los modelos importados.

6.5.1.3. Archivo views.py.

Este archivo es de suma importancia debido a que es donde se realiza la definición de todas las funciones de cada vista de la app de cinemática.

```
from django.shortcuts import render
from django.db.models import Count, Q
from .models import Directa_Robot1
from .models import Directa_Robot2
from .models import Inversa_Robot1
from .models import Inversa_Robot2
import math
import numpy as np
import RPi.GPIO as GPIO
import serial
import time
```

Figura 109. Archivo views.py de la app

de cinemática (importación de librerías)

En la anterior figura, la línea uno y dos son importaciones propias del archivo views.py que vienen por defecto, en cambio la línea tres, cuatro, cinco y seis son las mismas que se usa en el archivo admin.py, las cuales permiten importar los modelos desde el archivo models.py; finalmente en las últimas líneas se importan las librerías propias de Python que se van a utilizar. Por defecto

no vienen instaladas la librería numpy, RPi.GPIO, y serial, por lo cual para poder hacer uso de las mismas es necesario instalarlas dentro de la carpeta contenedora del proyecto, a través del terminal, usando el administrador de paquetes pip.

```
$ pip install RPi.GPIO
$ pip install pyserial
```

Para la instalación de la librería numpy se le realiza de una forma diferente debido a que de la anterior manera, aparece un error en la mitad de la instalación, por lo cual su instalación debe ser como se muestra a continuación:

```
$ pip download numpy==1.13.0
$ unzip numpy-1.13.0.zip
$ cd numpy -1.13.0
$ pip install .
```

Luego de importar e instalar todas las librerías necesarias, se dispone a escribir todas las funciones que definen a una vista.

```
def cinematica(request):
    return render(request, "cinematica/cinematica.html")
```

Figura 110. Archivo views.py de la app de cinemática

(definición de la función cinemática)

Una función cuenta con las siguientes partes:

- Se escribe primero la palabra def lo cual indica que se está definiendo una función.
- Segundo se escribe el nombre de la función, en la figura anterior se puede observar el nombre de las funciones en color verde.
- Una función puede o no recibir algún parámetro, para lo cual se abre paréntesis y dentro de él se coloca el parámetro que recibirá, en este caso el parámetro que recibe es request.
- Finalmente, dentro de la función se coloca a donde retornará para lo cual se escribe la ubicación del archivo HTML a retornar.

Dentro de las funciones se puede escribir algún programa que puede ejecutarse al momento de redirigirse a esa función, se puede redirigirse a una función a través de un botón que se muestre en el sitio web. Los programas que están escritos dentro de algunas funciones de la app de cinemática son:

- Posición Home del primer robot.
- Posición Home del segundo robot.
- Cinemática directa del primer robot.
- Cinemática directa del segundo robot.
- Cinemática directa de ambos robots.
- Cinemática inversa del primer robot.
- Cinemática inversa del segundo robot.
- Cinemática inversa de ambos robots.

```python
def home_robot1(request):

    gpio.setmode(gpio.BOARD)
    gpio.setup(29, gpio.OUT)
    gpio.output(29, False)

    puerto = serial.Serial('/dev/ttyS0', baudrate=9600, timeout=1)

    qq1 = 0
    qq2 = 90
    qq21 = 0
    qq3 = -90
    qq31 = 0
    qq4 = 0
    qq5 = 0

    pulso11=round((qq1*(-35550))/180)
    pulso22=round((qq21*(-16920))/90)
    pulso33=round((qq31*(-24695))/(-125))
    pulso44=round((qq4*(-3980))/(-90))
    pulso55=round((qq5*2005)/90)

    pulso1=str(pulso11)
    pulso2=str(pulso22)
    pulso3=str(pulso33)
    pulso4=str(pulso44)
    pulso5=str(pulso55)

    pulsso1='<P1'+ pulso1 + '\r'
    pulsso2='<P2'+ pulso2 + '\r'
    pulsso3='<P3'+ pulso3 + '\r'
    pulsso4='<P4'+ pulso4 + '\r'
    pulsso5='<P5'+ pulso5 + '\r'
    pulsso6='<P8' + '\r'
    puerto.write(pulsso1)
    time.sleep(1)
    puerto.write(pulsso2)
    time.sleep(1)
    puerto.write(pulsso3)
    time.sleep(1)
    puerto.write(pulsso4)
    time.sleep(1)
    puerto.write(pulsso5)
    time.sleep(1)
    puerto.write(pulsso6)

    return render(request, "cinematica/cinematica.html")
```

Figura 111. Archivo views.py de la app de cinemática

(función para posición home del robot 1)

En la figura anterior se muestra el programa para que el robot uno se coloque en la posición HOME, en primer lugar se coloca el pin GPIO número 29 de la Raspberry Pi como salida y se lo pone en cero lógico, debido a que dicho pin es el que hace la función de switch entre las dos tarjetas

que tienen el microcontrolador ATxmega64D3; a continuación se define el puerto serial de la Raspberry Pi; después se coloca a las variables de cada uno de los ángulos del brazo robótico en cero debido a que en la posición HOME, los encoders marcan cero pulsos, por lo que dichos valores son transformados a String para posteriormente ser enviados vía serial hacia la tarjeta controladora del robot uno. La misma programación se realiza para la posición HOME del robot dos, con la única diferencia que el pin GPIO número 29 se lo coloca en uno lógico.

```
def cinematica_directa_robot1(request):

    if request.method == "POST":
        qqq1 = request.POST['angulo1']
        qqq2 = request.POST['angulo2']
        qqq3 = request.POST['angulo3']
        qqq5 = request.POST['angulo4']
        qqq4 = request.POST['angulo5']

    gpio.setmode(gpio.BOARD)
    gpio.setup(29, gpio.OUT)
    gpio.output(29, False)

    puerto = serial.Serial('/dev/ttyS0', baudrate=9600, timeout=1)

    qq1 = float(qqq1)
    qq2 = float(qqq2)
    qq21 = float(qqq2)-90
    qq3 = float(qqq3)
    qq31 = float(qqq3)+90
    qq4 = float(qqq4)
    qq5 = float(qqq5)

    ang3=qq21+qq31
    ang5=ang3+qq5
    ang4=-(ang5/2)

    pulso11=round((qq1*(-35550))/180)
    pulso22=round((qq21*(-16920))/90)
    pulso33=round((ang3*(-24695))/(-125))
    pulso44=round((ang4*(-3980))/(-90))
    pulso55=round((ang5*2005)/90)

    pulso1=str(pulso11)
    pulso2=str(pulso22)
    pulso3=str(pulso33)
    pulso4=str(pulso44)
    pulso5=str(pulso55)

    pulsso1='<P1'+ pulso1 + '\r'
    pulsso2='<P2'+ pulso2 + '\r'
    pulsso3='<P3'+ pulso3 + '\r'
    pulsso5='<P4'+ pulso4 + '\r'
    pulsso4='<P5'+ pulso5 + '\r'
```

CONTINÚA

```
pulsso1='<P1'+ pulso1 + '\r'
pulsso2='<P2'+ pulso2 + '\r'
pulsso3='<P3'+ pulso3 + '\r'
pulsso5='<P4'+ pulso4 + '\r'
pulsso4='<P5'+ pulso5 + '\r'

puerto.write(pulsso1)
time.sleep(1)
puerto.write(pulsso2)
time.sleep(1)
puerto.write(pulsso3)
time.sleep(1)
puerto.write(pulsso4)
time.sleep(1)
puerto.write(pulsso5)


q1 = math.radians(qq1)
q2 = math.radians(qq2)
q3 = math.radians(qq3)
q5 = math.radians(qq4)
q4 = math.radians(qq5)

L0=10.5
L1=16
a2=25
a3=26.5
a4=5.5
#Para q1
Cq1=math.cos(q1)
Sq1=math.sin(q1)
#Para q2
Cq2=math.cos(q2)
Sq2=math.sin(q2)
#Para q3
Cq3=math.cos(q3)
Sq3=math.sin(q3)
#Para q4
Cq4=math.cos(q4)
Sq4=math.sin(q4)
#Para q5
Cq5=math.cos(q5)
Sq5=math.sin(q5)

A1= np.array([[Cq1, 0, Sq1, 0],[Sq1, 0, -Cq1, 0],[0, 1, 0, L0+L1],[0, 0, 0, 1]])
A2= np.array([[Cq2, -Sq2, 0, a2*Cq2],[Sq2, Cq2, 0, a2*Sq2],[0, 0, 1, 0],[0, 0, 0, 1]])
A3= np.array([[Cq3, -Sq3, 0, a3*Cq3],[Sq3, Cq3, 0, a3*Sq3],[0, 0, 1, 0],[0, 0, 0, 1]])
A4= np.array([[Cq4, -Sq4, 0, a4*Cq4],[Sq4, Cq4, 0, a4*Sq4],[0, 0, 1, 0],[0, 0, 0, 1]])
A5= np.array([[Cq5, 0, Sq5, 0],[Sq5, 0, -Cq5, 0],[0, 1, 0, 0],[0, 0, 0, 1]])

T3=np.dot(A1,A2)
T2=np.dot(T3,A3)
T1=np.dot(T2,A4)
T=np.dot(T1,A5)

X0=round(T[0,3], 3)
Y0=round(T[1,3], 3)
Z0=round(T[2,3], 3)

ejeX0 = float(X0)
ejeY0 = float(Y0)
ejeZ0 = float(Z0)
```

CONTINÚA

```
cinematica1 = 'directa'

print('Registro CD-ROBOT1')

movimiento = Directa_Robot1(Cinematica_R1= cinematica1, Ang_Cintura_R1=qqq1,Ang_Hombro_R1=qqq2,Ang_Codo_R1=qqq3,
                            Ang_Muneca_Pitch_R1=qqq5,Ang_Muneca_Roll_R1=qqq4,Coord_X_R1=ejeX0,Coord_Y_R1=ejeY0,
                            Coord_Z_R1=ejeZ0,)
movimiento.save()

content ={
    'angulo1' : qqq1,
    'angulo2' : qqq2,
    'angulo3' : qqq3,
    'angulo4' : qqq4,
    'angulo5' : qqq5,
    'X0' : X0,
    'Y0' : Y0,
    'Z0' : Z0,
}

return render(request, "cinematica/cinematica_respuesta.html", {'robot1': content})
```

Figura 112. Archivo views.py de la app de cinemática

(función para la cinemática directa del robot 1)

En la figura anterior se muestra el programa para la cinemática directa del primer robot, se observa que se obtiene los ángulos de las articulaciones del robot ingresadas por el usuario, después se coloca el pin GPIO número 29 de la Raspberry Pi como salida y se lo pone en cero lógico, debido a que dicho pin es el que hace la función de switch entre las dos tarjetas que tienen el microcontrolador ATxmega64D3; a continuación se define el puerto serial de la Raspberry Pi; después se transforma en flotantes los ángulos ingresados por el usuario, luego se realiza una regla de tres para cada uno de los ángulos de las articulaciones del robot, debido a que se necesita transformar los ángulos que se encuentran en grados a número de pulsos, dichos número de pulsos son transformados a String para posteriormente ser enviados vía serial hacia la tarjeta controladora del robot uno.

Los ángulos que fueron transformados en flotante son utilizados para poder aplicar el algoritmo de Denavit-Hartenberg, a través del método matricial, el mismo que se describe en el ítem 8.3.1, finalmente se obtiene las coordenadas X, Y, Z y tanto los ángulos como las posiciones se guardan dentro de tabla de datos Directa_Robot1 y se muestran en el sitio web para que el usuario observe el resultado de la cinemática directa.

El mismo programa es para el segundo robot, con la única diferencia que se encuentra dentro de una función diferente llamada cinematica_directa__robot2, los POST que se coloca al inicio cambian debido a que se refiere a los inputs del segundo robot, ya que cada input tiene su propio nombre como se mencionó en el ítem 6.4.2.1. además, el pin GPIO 29 se colocará en uno lógico para que escoja la segunda tarjeta controladora.

```
def cinematica_directa_robot2(request):

    if request.method == "POST":
        qqq6 = request.POST['angulo6']
        qqq7 = request.POST['angulo7']
        qqq8 = request.POST['angulo8']
        qqq9 = request.POST['angulo9']
        qqq10 = request.POST['angulo10']
```

Figura 113. Archivo views.py de la app de cinemática

(POST de la función para la cinemática directa del robot 2)

Para la cinemática directa conjunta de los dos robots se une los dos programas tanto del robot uno como del robot dos y se lo coloca en una sola función llamada cinematica_directa_robot1y2.

```
def cinematica_directa_robot1y2(request):

    if request.method == "POST":
        qqq1 = request.POST['angulo1']
        qqq2 = request.POST['angulo2']
        qqq3 = request.POST['angulo3']
        qqq4 = request.POST['angulo4']
        qqq5 = request.POST['angulo5']
        qqq6 = request.POST['angulo6']
        qqq7 = request.POST['angulo7']
        qqq8 = request.POST['angulo8']
        qqq9 = request.POST['angulo9']
        qqq10 = request.POST['angulo10']
```

Figura 114. Archivo views.py de la app de cinemática

(POST de la función para la cinemática directa del robot 1 y 2)

Para el programa de la cinemática inversa tanto individual como conjunta se tiene como post las posiciones o coordenadas cartesianas y se obtiene como resultado los ángulos de rotación de los elementos del brazo robot; el desarrollo de la cinemática inversa de ambos robots, se la realizó a través del método geométrico, el mismo que se describe en el ítem 8.3.2. Al igual que en la cinemática directa, tanto los ángulos como las posiciones se guardan dentro de una tabla de datos y se muestran en el sitio web para que el usuario observe el resultado de la cinemática inversa. Al momento de realizar el método geométrico se tomará dichos ángulos de las articulaciones del robot se las transformará a número de pulsos, y finalmente serán enviadas hacia la tarjeta controladora vía serial.

```
def cinematica_inversa_robot1(request):

    if request.method == "POST":
        xx0 = request.POST['posicion1']
        yy0 = request.POST['posicion2']
        zz0 = request.POST['posicion3']

    gpio.setmode(gpio.BOARD)
    gpio.setup(29, gpio.OUT)
    gpio.output(29, False)

    puerto = serial.Serial('/dev/ttyS0', baudrate=9600, timeout=1)

    x0 = float(xx0)
    y0 = float(yy0)
    z0 = float(zz0)

    q11= math.atan2(y0,x0)
    q1=round(math.degrees(q11),3)

    modulo=math.sqrt((x0**2)+(y0**2))

    qq=0
    d1=26.5
    l2=25
    l3=26.5
    l4=5.5

    q= math.radians(qq)

    lx=l4*(math.cos(q))
    lz=l4*(math.sin(q))

    x1=modulo-lx
    z1=z0-lz-d1
    h=math.sqrt((x1**2)+(z1**2))

    a11=math.atan2(z1,x1)
    a1=math.degrees(a11)

    a22=math.acos(((l2**2)-(l3**2)+(h**2))/(2*h*l2))
    a2=math.degrees(a22)
    q2=round(a1+a2,3)
```

CONTINÚA

```
a33=math.acos(((l2**2)+(l3**2)-(h**2))/(2*l2*l3))
a3=math.degrees(a33)

q3=round(a3-180,3)

q4=round(qq-q2-q3,3)

angulo1 = float(q1)
angulo2 = float(q2)-90
angulo22 = float(q2)
angulo3 = float(q3)+90
angulo33 = float(q3)
angulo4 = float(q4)
angulo5 = 0

ang3=angulo2+angulo3
ang5=ang3+angulo4
ang4=-(ang5/2)

pulso11=round((angulo1*(-35550))/180)
pulso22=round((angulo2*(-16920))/90)
pulso33=round((ang3*(-24695))/(-125))
pulso55=round((ang5*(-3980))/(-90))
pulso44=round((ang4*2005)/90)

pulso1=str(pulso11)
pulso2=str(pulso22)
pulso3=str(pulso33)
pulso4=str(pulso44)
pulso5=str(pulso55)

pulsso1='<P1'+ pulso1 + '\r'
pulsso2='<P2'+ pulso2 + '\r'
pulsso3='<P3'+ pulso3 + '\r'
pulsso4='<P4'+ pulso4 + '\r'
pulsso5='<P5'+ pulso5 + '\r'

puerto.write(pulsso1)
time.sleep(1)
puerto.write(pulsso2)
time.sleep(1)
puerto.write(pulsso3)
time.sleep(1)
puerto.write(pulsso4)
time.sleep(1)
puerto.write(pulsso5)

cinematica2 = 'inversa'

print('Registro CI-ROBOT1')
movimiento = Inversa_Robot1(Cinematica_IR1= cinematica2, Coord_X_IR1=xx0,Coord_Y_IR1=yy0,Coord_Z_IR1=zz0,
                            Ang_Cintura_IR1=angulo1,Ang_Hombro_IR1=angulo22,Ang_Codo_IR1=angulo33,
                            Ang_Muneca_Pitch_IR1=angulo4,Ang_Muneca_Roll_IR1=0)
movimiento.save()

content ={
    'posicion1' : x0,
    'posicion2' : y0,
    'posicion3' : z0,
    'angulo1' : q1,
    'angulo2' : q2,
    'angulo3' : q3,
    'angulo4' : q4,
}

return render(request, "cinematica/cinematica_inversa_respuesta.html", {'robot1I': content})
```

Figura 115. Archivo views.py de la app de cinemática

(Función para la cinemática inversa del robot 1)

```
def cinematica_inversa_robot2(request):

    if request.method == "POST":
        xx0 = request.POST['posicion4']
        yy0 = request.POST['posicion5']
        zz0 = request.POST['posicion6']
```

Figura 116. Archivo views.py de la app de cinemática

(POST de la función para la cinemática inversa del robot 2)

```
def cinematica_inversa_robot1y2(request):

    if request.method == "POST":
        XX0 = request.POST['posicion1']
        YY0 = request.POST['posicion2']
        ZZ0 = request.POST['posicion3']
        xx0 = request.POST['posicion4']
        yy0 = request.POST['posicion5']
        zz0 = request.POST['posicion6']
```

Figura 117. Archivo views.py de la app de cinemática

(POST de la función para la cinemática inversa del robot 1 y 2)

```
# GRAFICAS ESTADISTICAS

def graficas_estadisticas1(request):
    directa1 = Directa_Robot1.objects.values()#obtenemos nuestros datos
    return render(request,'graficas/grafica_directa_robot1.html',{'robot1_directa':directa1})

def graficas_estadisticas2(request):
    directa2 = Directa_Robot2.objects.values()#obtenemos nuestros datos
    return render(request,'graficas/grafica_directa_robot2.html',{'robot2_directa':directa2})

def graficas_estadisticas3(request):
    inversa1 = Inversa_Robot1.objects.values()#obtenemos nuestros datos
    return render(request,'graficas/grafica_inversa_robot1.html',{'robot1_inversa':inversa1})

def graficas_estadisticas4(request):
    inversa2 = Inversa_Robot2.objects.values()#obtenemos nuestros datos
    return render(request,'graficas/grafica_inversa_robot2.html',{'robot2_inversa':inversa2})
```

Figura 118. Archivo views.py de la app de cinemática (Función para las gráficas estadísticas)

Dichas funciones que se observa en la figura anterior sirven para la redirección hacia las páginas de las gráficas estadísticas mencionadas en el ítem 6.4.3. Se observa que en cada una de las funciones existe un objeto, para el caso de la primera función el objeto es directa1 el cual permite

obtener los datos desde la base de datos PostgreSQL, para que a través del método for se obtengan sus valores en la página HTML como se menciona en el ítem 6.4.3.1.

6.5.2. Aplicación usuario.

Dentro de esta aplicación estará todo lo relacionado al login, registro y página principal o índex del sitio web.

6.5.2.1. Archivo views.py.

Este archivo es de suma importancia debido a que es donde se realiza la definición de todas las funciones de cada vista de la app de usuario y además de las diferentes clases propias de Django.

```python
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from django.views.generic import CreateView
from django.core.urlresolvers import reverse_lazy
from django.shortcuts import render, redirect
from .forms import RegistroForm
from django.http import HttpResponse
#from django.contrib.auth.forms import UserCreationForm
#from django.contrib import messages

# Vistas de la mayoria de HTML para la pagina

def index(request):
    return render(request, 'base/index.html')

def index_logueado(request):
    return render(request, 'base/index-login.html')

def first_login(request):
    return render(request, 'base/First-login.html')

def index_usuario(request):
    return render(request, 'base/webcam.html')

class RegistroUsuario(CreateView):
    model = User
    template_name = 'base/registrarte.html'
    form_class = RegistroForm
    success_url = reverse_lazy('login')
```

Figura 119. Archivo views.py de la app de usuario

En la figura anterior, la primera línea de código viene por defecto en el archivo, las otras importaciones a excepción de HttpResponse, se las realiza para poder crear la clase del registro de usuarios, en cambio HttpResponse se lo utiliza para crear objetos o instancias de tipo http.

Luego de importar todas las librerías se comienza con la definición de las funciones tanto de la página principal del sitio web como de login del usuario, la creación de estas funciones es de la misma forma que se explicó en el ítem 6.5.1.3.

Finalmente se crea una clase con el nombre RegistroUsuario, dicha clase tiene como parámetro CreateView, el cual crea una vista del registro. Dentro de la clase se colocó un modelo User, después se escribe el template en donde se encuentra el registro de usuario escrito en código HTML, a continuación, se coloca un formulario de registro que se encuentra dentro del archivo forms.py el cual se explica en el siguiente ítem, ya para finalizar se escribe a donde debe redirigirse el sitio web, si su registro es satisfactorio.

6.5.2.2. Archivo forms.py.

Por defecto en una app no viene incluido el archivo forms.py por lo cual es necesario crearlo, el mismo servirá para escribir el formulario del registro de usuario, en el cual se puede agregar cosas adicionales al formulario que Django provee por defecto.

```
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class RegistroForm(UserCreationForm):

    class Meta:
        model = User
        fields = [
                'username',
                'first_name',
                'last_name',
                'email',
                'is_staff',
            ]
        labels = {
                'username': 'Nombre de Usuario',
                'first_name': 'Nombre',
                'last_name': 'Apellido',
                'email': 'Correo',
                'is_staff': 'Permisos',
        }
        unique_together = ('email', 'username')
```

Figura 120. Archivo forms.py de la app de usuario

En la figura anterior se puede observar que se importa un UserCreationForm el cual permite crear un formulario de usuario, después se observa que se crea una clase llamada RegistroForm la misma que es llamada desde la clase RegistroUsuario descrita en el ítem anterior, esta clase RegistroForm tiene como parámetro el formulario que se importó, dentro de la clase se definen los campos del formulario conocidos como fields y las etiquetas o labels. El nombre de los fields son los mismos que se deben colocar en los name de los inputs del archivo registrarte.html.

Todos los datos ingresados en el input del formulario registro son guardados en una tabla de datos que viene por defecto con Django lo único necesario es migrar las tablas de datos a la base de datos PostgreSQL antes de utilizarla para lo cual es necesario escribir las siguientes líneas código dentro de la carpeta del proyecto desde el terminal.

```
$ python manage.py makemigrations

$ python manage.py migrate
```

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
joel	coronel1995@hotmail.com			✓
joelin	coronel123@hotmail.com	joeli	coronel	✓
jou	jou123@hotmail.com			✓

3 users

Figura. 121 Tabla de datos de la base de registro de usuarios desde el administrador de Django

6.5.2.3. Archivo urls.py.

Por defecto en una app no viene incluido el archivo urls.py por lo cual es necesario crearlo, el mismo servirá para escribir la url del formulario de registro.

```
from django.conf.urls import url , include
from .views import RegistroUsuario

urlpatterns = [

    url(r'^nuevo/', RegistroUsuario.as_view(), name='registrar'),
    #path('', views.index, name='inicio'),
]
```

Figura 122. Archivo urls.py de la app de usuario

En la figura anterior se puede observar que se importan las librerías url, include y además la función RegistroUsuario de la view de la app usuario. Finalmente se escribe la url de la función RegistroUsuario, la cual se la denota como registrar.

6.5.3. Aplicación cámara

Dentro de esta aplicación estará todo lo relacionado a la supervisión de la celda robótica.

6.5.3.1. Archivo views.py

Este archivo es de suma importancia debido a que es donde se realiza la definición de todas las funciones de cada vista de la app de cámara.

```
from django.shortcuts import render

# Create your views here.
def camara1(request):
    return render(request, 'cámara/camara1.html')
```

Figura 123. Archivo views.py de la app de cámara

En la figura anterior se observa que solo se tiene una definición de función llamada cámara1, debido a que toda la supervisión de la celda robótica, se la realiza en una sola página web.

6.5.4. Aplicación procesos

Dentro de esta aplicación estará todo lo relacionado a los procesos industriales de los dos robots CRS-A255.

6.5.4.1. Archivo views.py

Este archivo es de suma importancia debido a que es donde se realiza la definición de todas las funciones de cada vista de la app de procesos y además los diferentes programas para la ejecución de cada uno de los procesos.

```
from __future__ import unicode_literals

from django.shortcuts import render
import math
import numpy as np
import RPi.GPIO as gpio
import serial
import time


def ProcesosIndustriales(request):
    return render(request, "procesos/procesos.html",{})

def Multimedia(request):
    return render(request, "procesos/multimedia.html",{})
```

Figura 124. Archivo views.py de la app de procesos, primera parte

En la figura anterior, la primera y segunda línea de código vienen por defecto en el archivo, las otras importaciones son las mismas librerías propias de Python que se importaron en el archivo views.py de la app de cinemática, en donde se explica cómo deben ser instaladas a través del terminal.

Luego de importar todas las librerías se comienza con la definición de las funciones tanto de la página de procesos industriales como la de multimedia; la creación de estas funciones es de la misma forma que se explicó en el ítem 6.5.1.3.

```
def proceso_uno(request):

    gpio.setmode(gpio.BOARD)
    gpio.setup(29, gpio.OUT)
    gpio.output(29, False)

    puerto = serial.Serial('/dev/ttyS0', baudrate=9600, timeout=1)


    q1 = str(-33254)
    q2 = str(0)
    q3 = str(0)
    q4 = str(0)
    q5 = str(0)

    pulso0 = '<P6\r'
    pulso1 = '<P1' + q1 + '\r'
    pulso2 = '<P2' + q2 + '\r'
    pulso3 = '<P3' + q3 + '\r'
    pulso4 = '<P4' + q4 + '\r'
    pulso5 = '<P5' + q5 + '\r'

    pulso0 = str(pulso0)
    pulso1 = str(pulso1)
    pulso2 = str(pulso2)
    pulso3 = str(pulso3)
    pulso4 = str(pulso4)
    pulso5 = str(pulso5)

    puerto.write(pulso0)
    time.sleep(1)
    puerto.write(pulso1)
    time.sleep(1)
    puerto.write(pulso2)
    time.sleep(1)
    puerto.write(pulso3)
    time.sleep(1)
    puerto.write(pulso4)
    time.sleep(1)
    puerto.write(pulso5)
    time.sleep(1)
```

Figura 125. Archivo views.py de la app de procesos,

segunda parte (primera posición de la función proceso uno)

En la figura anterior se muestra el programa para la ejecución del proceso uno, en principio se coloca el pin GPIO número 29 de la Raspberry Pi como salida y se lo pone en cero lógico; a continuación, se define el puerto serial de la Raspberry Pi; posteriormente se escribe una variable para cada una de las articulaciones, asignando el valor correspondiente a la primera posición del proceso uno, a dichas variables se adiciona una etiqueta, para que la tarjeta controladora admita ese valor enviado por el puerto serial.

Para las siguientes posiciones que forman los diferentes procesos, siguen el mismo procedimiento de programación explicado en la figura anterior.

6.6. URL

En el proyecto se puede crear tantas URL como se crea necesario, las cuales permiten la redirección o navegación entre una pestaña a otra. En Django es necesario primero definir cada una de las vistas a través de funciones, esto se lo realiza en los archivos views.py los cuales se encuentran dentro de cada app creada por el diseñador de la página web, para entender mejor este tema de las vistas véase el ítem 6.5.

Luego de crear correctamente las vistas de cada URL que se necesite, lo siguiente es dirigirse al archivo urls.py, este archivo se encuentra en la carpeta Diana4.

```
from django.conf.urls import url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/' admin.site.urls),
]
```

Figura 126. Archivo urls.py sin modificaciones

En la figura anterior se observa las líneas de código que vienen por defecto en el archivo urls.py. En las primeras dos líneas de código se está importando recursos, el primero es url desde conf.urls, el cual permite el escrito de las URL dentro de la función urlpatterns = [ ]. Dentro de los corchetes de la función mencionada anteriormente se escribe todas las diferentes URL del sitio web.

En la segunda línea de código se importa admin el cual es importado desde contrib, el mismo que permite el ingreso al sitio web de administración de base de datos de Django.

```
from django.conf.urls import url , include
from django.contrib import admin
from apps.camara import views as camara
from apps.usuario import views
from apps.cinematica import views as cinematica
from apps.procesos import views as procesos

from django.contrib.auth import views as auth_views

urlpatterns = [
    url(r'^robotica/admin', include(admin.site.urls)),
    url(r'^camara/', camara.camara1, name='camara'),
    url(r'^first_login/', views.first_login, name='first_login'),
    url(r'^robotica/', views.index_logueado, name='inicio-logueado'),
    url(r'^registro/', include('apps.usuario.urls'), name='registro'),
    url(r'^login/',auth_views.LoginView.as_view(template_name='base/login.html'), name='login'),
    url(r'^inicio/',auth_views.LoginView.as_view(template_name='base/index.html'), name='inicio'),
    url(r'^logout/',auth_views.LogoutView.as_view(template_name='base/login.html'), name='logout'),

    #url(r'^cinematica1/', cinematica.cinematica1, name='cinematica1'),
    url(r'^cinematica/', cinematica.cinematica, name='cinematica'),
    url(r'^posicion_home1/', cinematica.home_robot1, name='home1'),
    url(r'^posicion_home2/', cinematica.home_robot2, name='home2'),
    url(r'^cinematica_directa1/', cinematica.cinematica_directa_robot1, name='directa1'),
    url(r'^cinematica_directa2/', cinematica.cinematica_directa_robot2, name='directa2'),
    url(r'^cinematica_directa/', cinematica.cinematica_directa_robot1y2, name='directa1y2'),

    url(r'^cinematica_inversa1/', cinematica.cinematica_inversa_robot1, name='inversa1'),
    url(r'^cinematica_inversa2/', cinematica.cinematica_inversa_robot2, name='inversa2'),
    url(r'^cinematica_inversa/', cinematica.cinematica_inversa_robot1y2, name='inversa1y2'),

    url(r'^graficas_directa_robot1/', cinematica.graficas_estadisticas1, name='estadisticas1'),
    url(r'^graficas_directa_robot2/', cinematica.graficas_estadisticas2, name='estadisticas2'),
    url(r'^graficas_inversa_robot1/', cinematica.graficas_estadisticas3, name='estadisticas3'),
    url(r'^graficas_inversa_robot2/', cinematica.graficas_estadisticas4, name='estadisticas4'),

    url(r'^Proceso_Tres/', procesos.proceso_tres, name='proceso3'),
    url(r'^Proceso_Dos/', procesos.proceso_dos, name='proceso2'),
    url(r'^Proceso_Uno/', procesos.proceso_uno, name='proceso1'),
    url(r'^Procesos-Industriales/', procesos.ProcesosIndustriales, name='procesos'),
    url(r'^Multimedia/', procesos.Multimedia, name='multimedia'),
    url(r'^Ajuste2/', procesos.freno_hombro, name='hombro'),
    url(r'^Ajuste3/', procesos.freno_codo, name='codo'),
    url(r'^Ajuste4/', procesos.freno_muneca_roll, name='roll'),
    url(r'^Ajuste5/', procesos.freno_muneca_pitch, name='pitch'),
    url(r'^Ajustes_Apagados/', procesos.apagar_ajustes, name='apagar'),
    url(r'^Ajustes/', procesos.Ajustes, name='ajustes'),
]
```

Figura 127. Archivo urls.py modificado

En el archivo urls.py modificado, se importan las views de las diferentes aplicaciones creadas como se observa en la líneas de código tres, cuatro, cinco y seis de la figura anterior, además se importa desde contrib.auth las views y para llamarlas se coloca el nombre de auth_views, este nombre puede modificarse dependiendo el gusto del desarrollador.

Dentro de la función urlpatterns se coloca las nuevas URL, para colocar cada URL, se debe seguir la siguiente sintaxis:

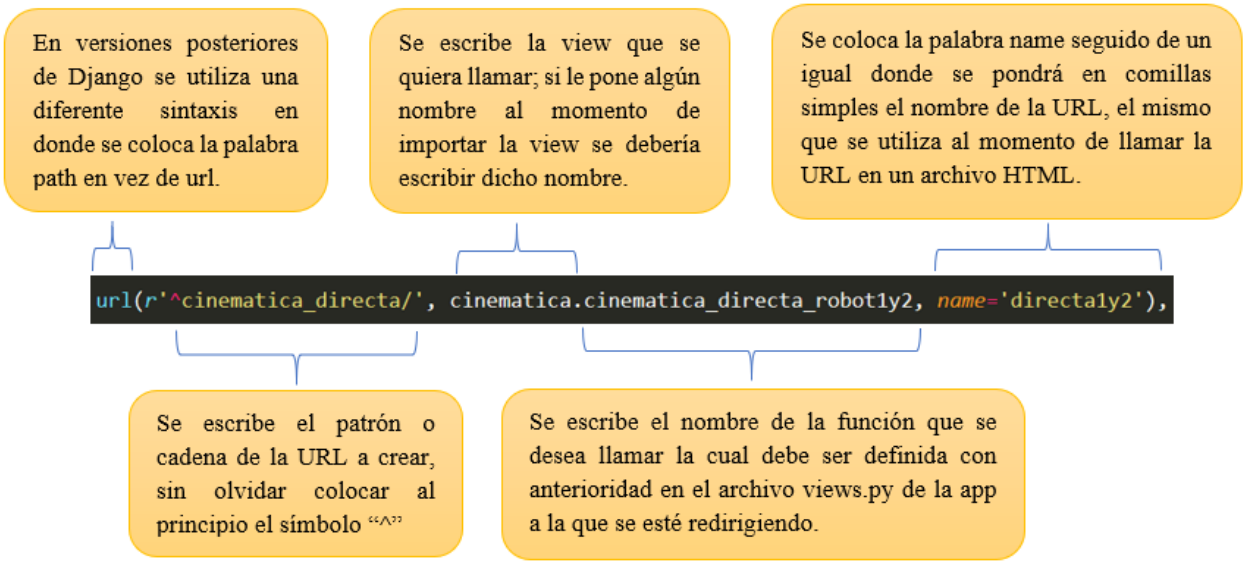

Figura 128. Sintaxis para para escritura de una URL dentro del framework Django

CAPÍTULO VII

SUPERVISIÓN REMOTA

7.1. Videocámara

Para tener un monitoreo constante de la celda robótica, fue necesario el uso de una videocámara, la cual tenga una grabación de video constante sin retrasos, con una imagen nítida en la cual se pueda observar todos los objetos de la celda y por último la posibilidad de ingresar en ella de manera remota.

7.1.1. Selección.

Se probó algunas opciones entre las cuales fueron la Arducam, una cámara USB y la webcam de la laptop. La desventaja de las dos primeras era que su video no era constante sino existía retardos los cuales distorsionaban la imagen, por ese motivo se eliminó esas dos posibilidades, en cambio la webcam de la laptop no tenía este problema sin embargo su visualización era de manera local y no permitía una visualización remota.

Otras posibilidades como son: la PI camera, las cámaras IP Wifi de marca China; las cuales no se probó por la falta del equipo, pero se descartaron, ya que la primera tenía el mismo problema de la Arducam es decir su video no era constante, y la segunda el problema fue que solo permitía el ingreso de manera remota a través de una aplicación en el celular o en la computadora, esto se pudo constatar a través de foros y videos tutoriales.

Es por todo lo mencionado anteriormente que se decidió el uso de la cámara EZVIZ HD Indoor WiFi modelo CS-CV206, la cual tiene muchas características interesantes.

Figura 129. Cámara

Ezviz HD Indoor WiFi

modelo CS-CV206

Como su nombre mismo lo dice tiene una alta resolución de 1280x720p, su conexión al router es inalámbrica, aun cuando necesita conexión alámbrica hacia el toma corriente, tiene un slot en donde se puede colocar una MicroSD para guardar fotos y videos, sin embargo existe otra manera de guardar las fotos y videos en la computadora que se esté visualizando la grabación en tiempo real de la cámara, tiene doble audio esto quiere decir que se escuchará todo lo que suceda en la celda robótica y además se puede enviar cualquier mensaje hacia la celda desde el lugar en donde se realice la supervisión, su ingreso hacia la cámara es a través de una página web o por una aplicación en el celular o en la computadora, además posee visión nocturna por lo que la supervisión hacia la celda robótica será 24/7.

Es importante mencionar que la marca EZVIZ pertenece a Hikvision por lo que una cámara de marca Hikvision podría igual ser útil para el propósito del proyecto, otra alternativa sería una cámara Motorola de la línea MBP, pero la desventaja es que su precio es más elevado que las dos anteriores y no se encuentra muy fácilmente en el mercado local.

7.1.2. Configuración.

Primero se conecta el cable de alimentación tanto al toma corriente como a la cámara, se sabe que está lista al momento que el led indicador de la cámara se torne de un azul permanente.

Para el manejo remoto desde el celular es necesario desde la App Store descargar la app llamada EZVIZ, ya descargada la app se ingresa y se crea una cuenta de usuario, si no se posee alguna, ya creada la cuenta se ingresa y se agrega una cámara a través del símbolo "+", posteriormente se escanea el código QR el cual viene en el manual, en la cámara o en la caja del equipo, finalmente nos pedirá el código de verificación el cual se encuentra en la parte inferior de la cámara.

En cambio, para la supervisión de la cámara desde la web es necesario ingresar a la siguiente URL https://ius.ezvizlife.com donde se mostrará la siguiente página web.

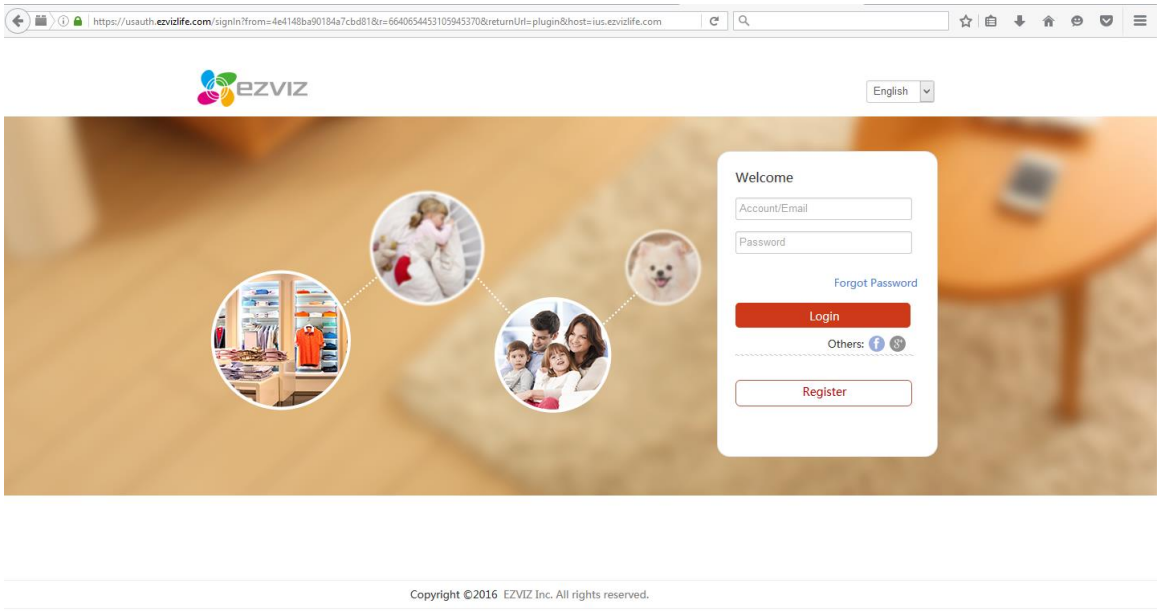

Figura 130. Sitio oficial de EZVIZ

La página web usa algunos plugins que son bloqueados por algunos navegadores web como son Google Chrome, Opera, Brave e incluso Mozilla Firefox en sus últimas versiones por lo que es

necesario bajar la versión de Mozilla Firefox a las versiones menores de 50.0 así no se tendrá ningún inconveniente al momento del ingreso hacia la plataforma.

Si ya se tiene una cuenta la cual puede ser la misma que se creó en la aplicación EZVIZ, solo hace falta loguearse, posteriormente se mostrará la siguiente página web.

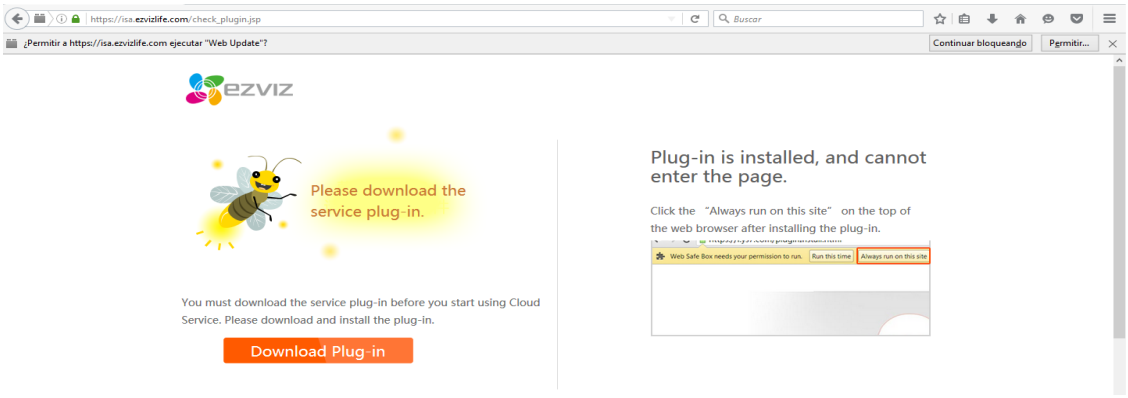

Figura 131. Plugin necesario para el correcto funcionamiento de la página de EZVIZ

Se indica que es necesario bajar un plugin para el funcionamiento de la página solo es cuestión de descargarlo y ejecutar el .exe, esto se realizará solo una vez ya que el plugin quedará instalado en la computadora, luego solo será cuestión dar clic en permitir y recordar.

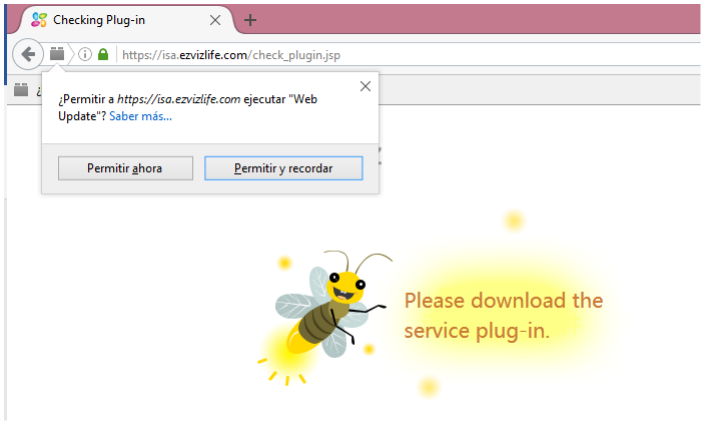

Figura 132. Aprobar el uso del plugin para el funcionamiento de la página EZVIZ

Finalmente, solo es necesario agregar la cámara como se hizo en la aplicación EZVIZ y se observará como se muestra en la siguiente imagen.

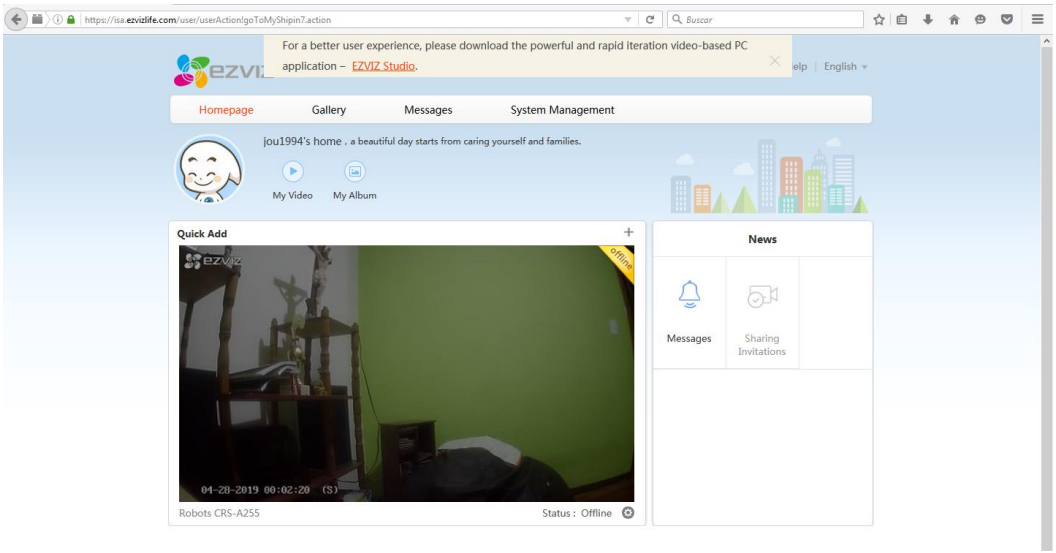

Figura 133. Visualización de la imagen de la cámara en la página web EZVIZ

CAPÍTULO VIII

DISEÑO DEL CONTROLADOR

8.1. Descripción de las diferentes teorías de control

Las teorías de control van desde la clásica hasta la digital, en donde el diseño de todos los controladores cumplen con la finalidad de mejorar las características de respuestas de los sistemas y mejorar el funcionamiento de los mismos. En el desarrollo de procesos automáticos e industriales se toma en cuenta la realimentación en lazo cerrado con el fin de comparar la salida con la entrada de referencia y utilizar esta diferencia como unidad de control (Dorf, 2005).

Para el diseño de cualquier tipo de controlador industrial es necesario considerar su robustez y su capacidad de mitigar varias de las perturbaciones que puede presentar el sistema al momento de entrar en funcionamiento, sin olvidar que a nivel industrial se tiene como fuente de energía la electricidad o algún fluido presurizado como lo es el aceite o el aire, a continuación, se describirán algunos de los controladores que son implementados a nivel industrial:

8.1.1. Control de dos posiciones o de encendido y apagado.

En un sistema on/off el actuador presentará solo dos posiciones fijas, que en la mayoría de las ocasiones será encendido o apagado, lo cual lo hace simple y barato al momento de ser implementado, este controlador poseerá una brecha diferencial en donde conmutará para cambiar de posición es común que los controladores de dos posiciones sean dispositivos eléctricos, pero en neumática los controladores de ganancias muy altas también tienen una función de dos posiciones.

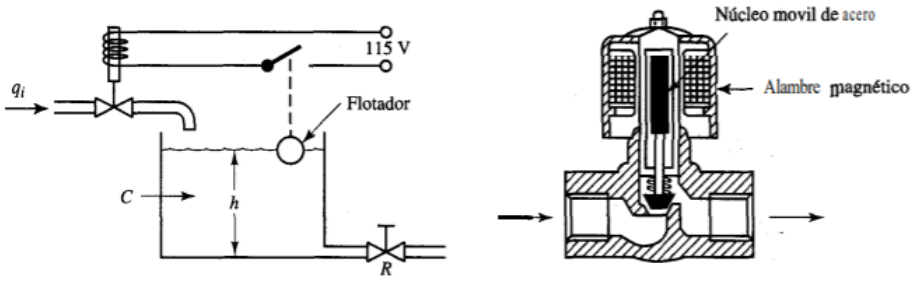

Figura 134. Ejemplos de control on/off sistema del control del nivel del líquido, válvula electromagnética

Fuente*: **(Ogata, 2000)***

8.1.2. Control proporcional e integral.

El control proporcional e integral está definido por la siguiente ecuación:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt$$

En donde se tendrá que K_p es la ganancia proporcional que actúa al instante en el sistema, mientras que la acción integral T_i actúa en un intervalo de tiempo que regula la velocidad de la acción de control, ambos parámetros son ajustables a las necesidades del sistema.

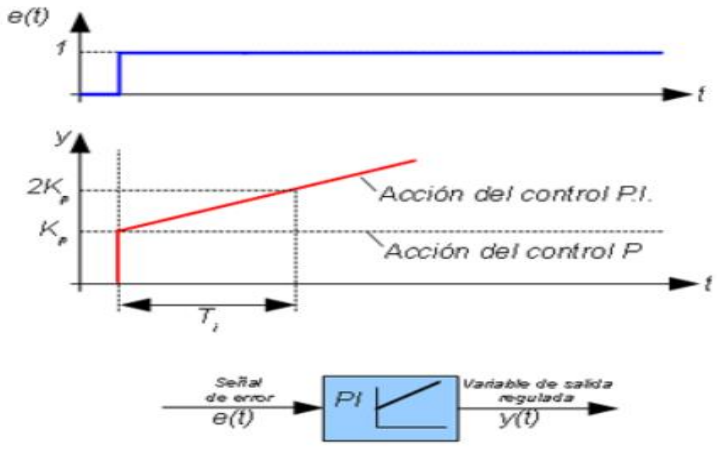

Figura 135. Control de acción proporcional e integral

Fuente: ***(Ortiz, 2015)***

En la figura 135 se puede notar que un controlador PI detecta de forma instantánea el error y después de un cierto tiempo la acción integral será la encargada de atenuar el error existente entre la señal de entrada con respecto a la salida.

8.1.3. Control proporcional integral y derivativo.

El controlador proporcional integral y derivativo viene dado por la siguiente ecuación:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt}$$

El controlador PID es el controlador más utilizado a nivel industrial esto se debe a que este controlador agrupa tres acciones básicas en el control que son la acción proporcional (K_p) encargada del error actual, la acción integral (T_i) la cual depende de las acciones pasadas y la acción derivativa (T_d) predice futuras perturbaciones o errores, por lo cual hace de este controlador el más robusto al momento de recibir la señal de realimentación. El uso de un controlador PID se puede presentar en procesos que involucren motores, por lo cual es muy utilizado en casi toda la maquinaria industrial.

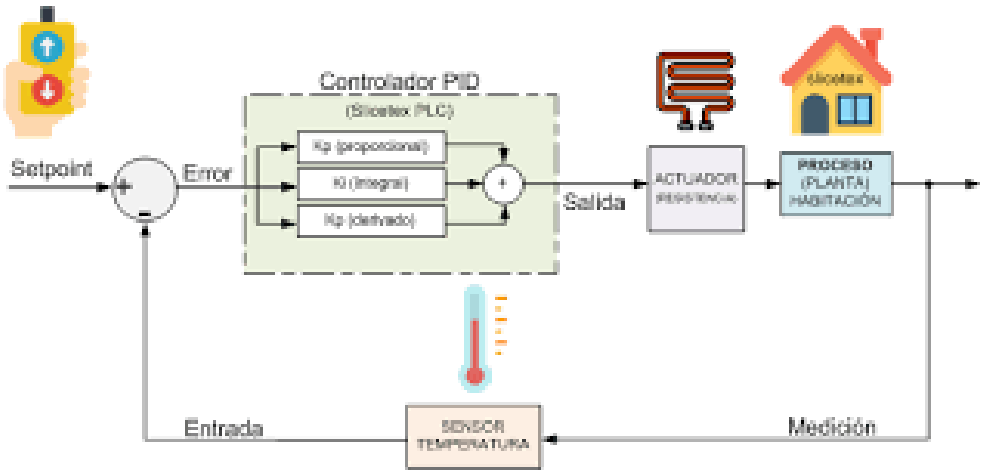

Figura 136. Control PID a nivel industrial

Fuente: ***(Ogata, 2000)***

8.1.4. Control fuzzy.

El control fuzzy está asociado a lo lógica de conjuntos difusos la misma que trata las ambigüedades que se pueden presentar en el medio, es decir los elementos que pueden existir o no en un determinado grupo; con esto se puede decir que la lógica difusa intenta modelar estas ambigüedades. Al momento de colocar lógica difusa en un control industrial se intenta emular conocimientos o experiencia de un operario dentro del cerebro de una máquina es decir que lo que se intenta es que una maquina pueda emular el pensamiento humano.

Al tener controles con lógica difusa en el cual se emula el pensamiento humano para ser utilizado en el cerebro de una máquina se tiene más ventajas prácticas con respecto a los controladores tradicionales, ya que el modelamiento matemático del proceso no es necesario y se puede desarrollar un controlador práctico sin complicaciones matemáticas que mejoren el rendimiento de los procesos desempeñados por dicha máquina.

El control fuzzy a pesar de no trabajar con un modelamiento matemático como tal sino con la lógica de un ser humano al momento de procesar las variables del controlador se hace de forma numérica, por lo que es necesario una etapa de fusificación que dota a los datos de entrada un grado de importancia, es decir transforma los valores numéricos de la entrada a expresiones que dependerán de la función de pertenencia. El controlador difuso traza sus reglas en que SI (IF) sucede una acción ENTONCES (THEN) se tendrá una consecuencia, siendo estas reglas lo más claras y entendibles para el cerebro de la máquina. El defusificador se encargará de transformar las expresiones a variables numéricas; esto se puede visualizar en la siguiente figura:

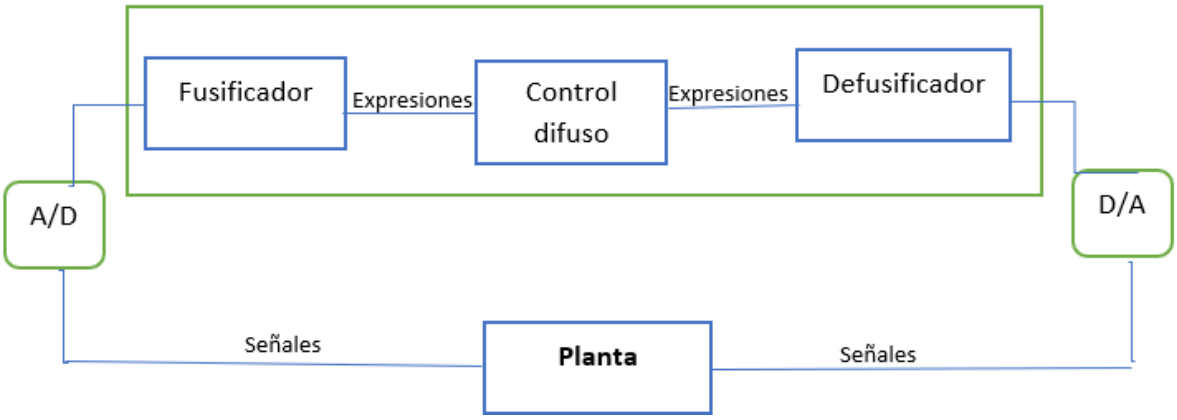

Figura 137. Componentes de un controlador difuso

8.2. Selección de la teoría de control

Como se puede apreciar en el apartado 8.1 se desglosa los principales controladores utilizados a nivel industrial; en el presente trabajo se utiliza manipuladores robóticos que son empleados a nivel industrial y de aprendizaje, se ve la factibilidad de emplear un controlador fuzzy debido a que este controlador permite tomar decisiones de forma más concisa al momento de su ejecución de procesos y evita la implementación de un modelo dinámico del sistema mejorando así el rendimiento de la celda de trabajo que se propone en el Capítulo 9.

Para obtener un control del manipulador robótico, lo que se realiza es un control de cada una de las articulaciones de dicho manipulador, por lo cual se controla cada motor de las mismas. A continuación, se describe el diagrama de bloques que debe de presentar el controlador difuso a ser implementado en cada uno de los manipuladores robóticos CRS A255.

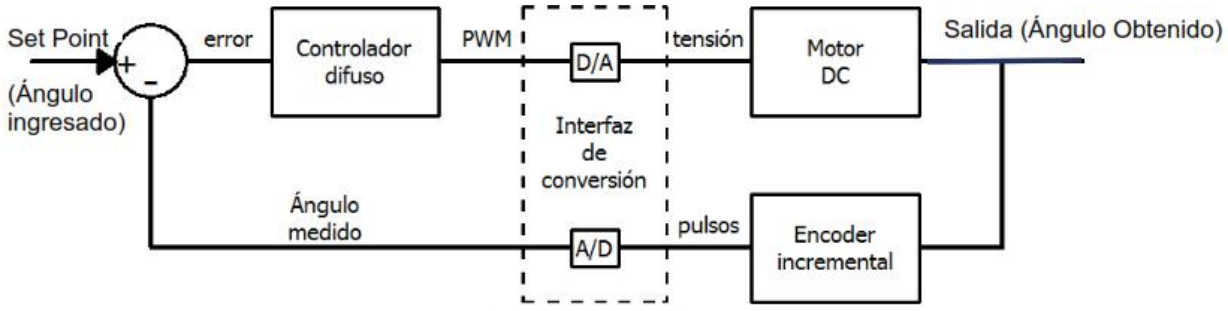

Figura 138. Diagrama de bloques de control fuzzy para manipuladores robóticos CRS A255

En la figura 138 se detalla el control que se desea efectuar en cada uno de los manipuladores, siendo la entrada de dicho controlador la diferencia entre el ángulo deseado del ángulo medido; al utilizar como sensor un encoder incremental se trabaja con pulsos es decir que todos los ángulos son transformados a pulsos definiendo las variables lingüísticas en la siguiente tabla:

Tabla 36
Entrada de la variable lingüística que ingresa al controlador

Variable lingüística que ingresa al controlador	
Variable lingüística Nomenclatura	**Descripción**
PNG	Pulso negativo grande
PNB	Pulso negativo bajo
PC	Pulso cero
PPP	Pulso positivo pequeño
PPG	Pulso positivo grande

Los valores de las variables lingüísticas de entrada estarán entre -1000 a 1000 debido a la resolución que tiene el encoder de cada uno de los motores como se observa en la siguiente figura:

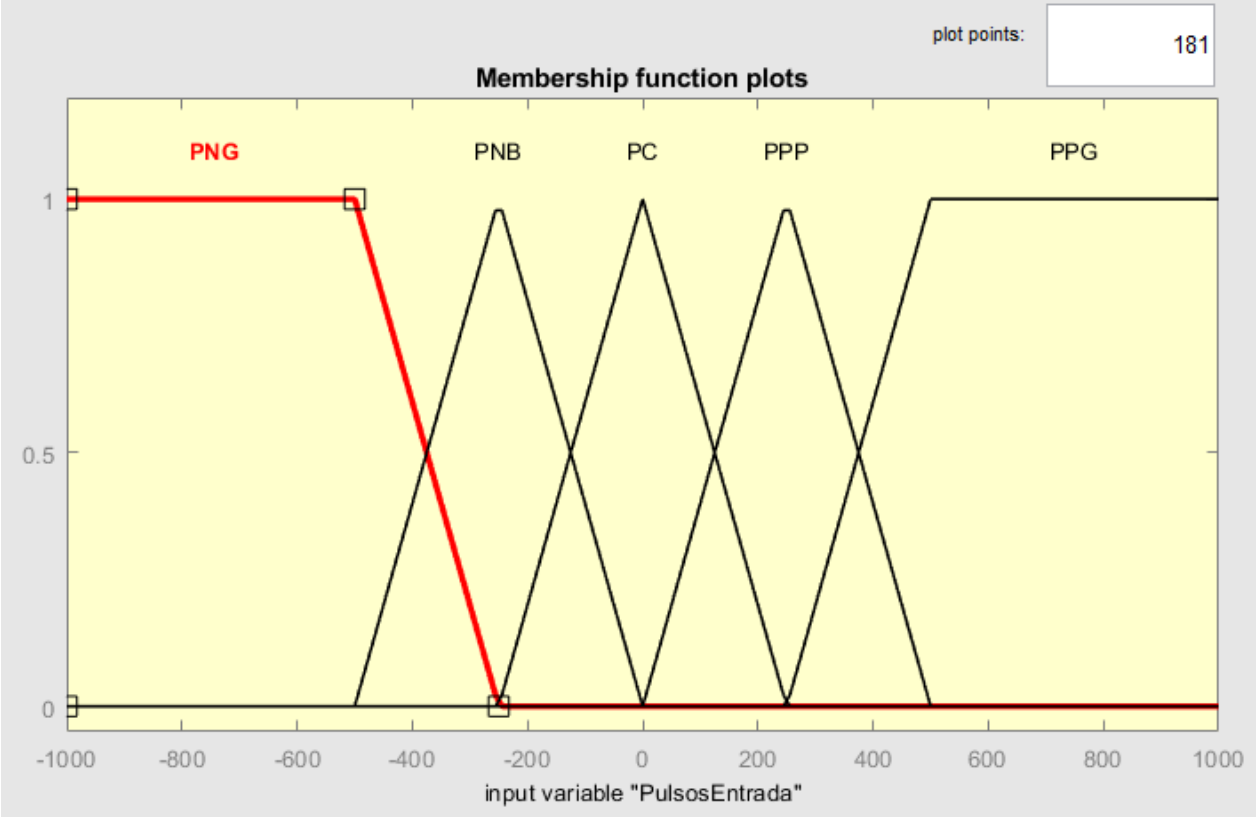

Figura 139. Funciones de pertenencia para la entrada del controlador

- PNG= *trapmf* (PulsoEntrada; -1000,-1000,-500,-250)
- PNB= *trinf* (PulsoEntrada; -500,-250,0)
- PC= *trinf* (PulsoEntrada; -250,0,250)
- PPP= *trinf* (PulsoEntrada; 0,250,500)
- PPG= *trapmf* (PulsoEntrada; 250,500,1000,1000)

La salida del controlador será en voltaje, como se mencionó en el capítulo 4 el voltaje de alimentación de los motores será de 24 voltios; al ser un control de posición se tendrá que trabajar con el sentido de giro del motor, por lo que en el diseño del controlador se toma el rango de -24 a 24 como se detalla en las funciones de relación y en sus variables lingüísticas citadas a continuación:

Tabla 37

Salida de la variable lingüística del controlador

Variable lingüística de salida del controlador	
Variable lingüística Nomenclatura	**Descripción**
VNG	Voltaje negativo grande
VNB	Voltaje negativo bajo
VC	Voltaje cero
VPP	Voltaje positivo pequeño
VPG	Voltaje positivo grande

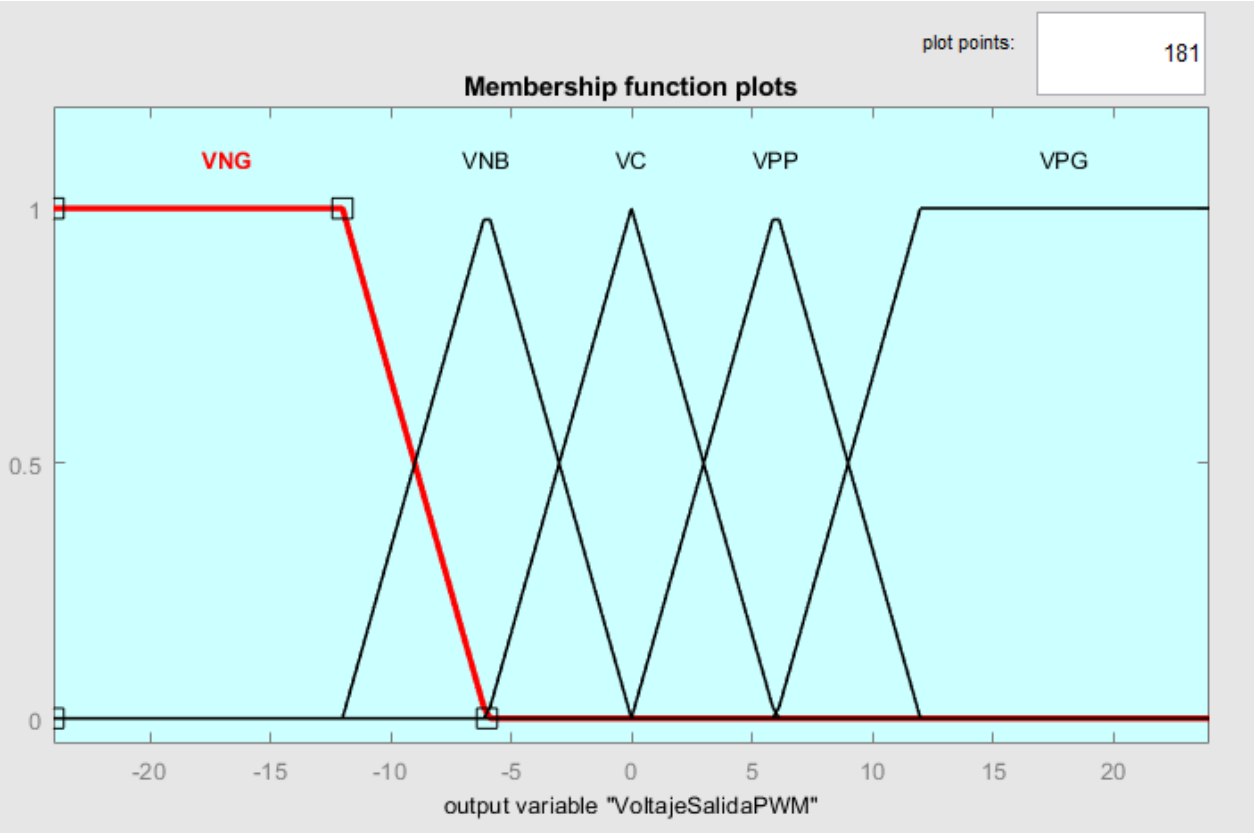

Figura 140. Funciones de pertenencia para la entrada del controlador

- VNG= *trapmf* (PulsoEntrada; -24,-24,-12,-6)
- VNB= *trinf* (PulsoEntrada; -12,-6,0)
- VC= *trinf* (PulsoEntrada; -6,0,6)
- VPP= *trinf* (PulsoEntrada; 0,6,12)

- VPG= *trapmf* (PulsoEntrada; 6,12,24,24)

Ya cuando se tienen los términos lingüísticos tanto de entrada como de salida del controlador se establecen las reglas del controlador detallándose en la siguiente tabla:

Tabla 38
Relaciones de pertenencia del controlador

Reglas del controlador	
SI	**ENTONCES**
PNG	VNG
PNB	VNB
PC	VC
PPP	VPP
PPG	VPG

Con estos datos se puede trabajar con la herramienta Toolbox de Matlab y obtener el controlador difuso, a continuación, se muestra la gráfica del controlador:

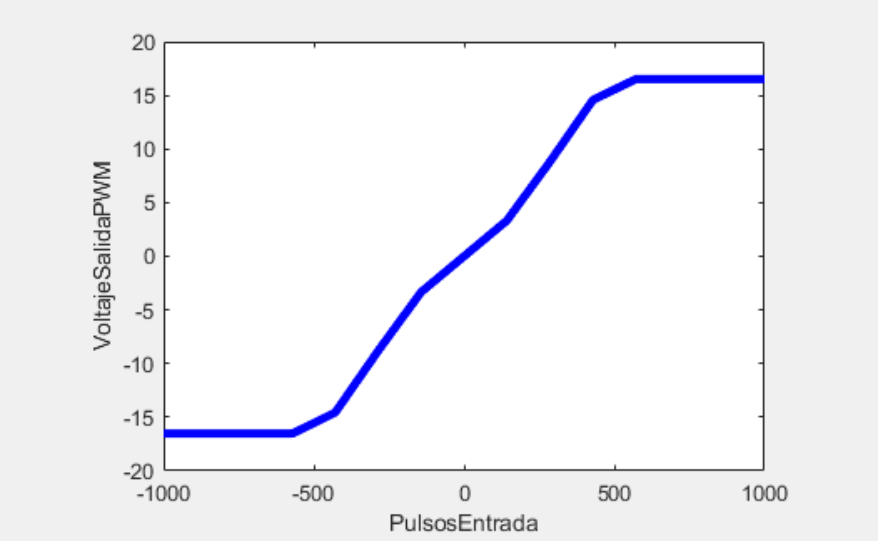

Figura 141. Curva del controlador a implementar

En la figura se ejemplifica los valores que obtendría la salida del controlador dependiendo de la entrada del mismo (pulsos).

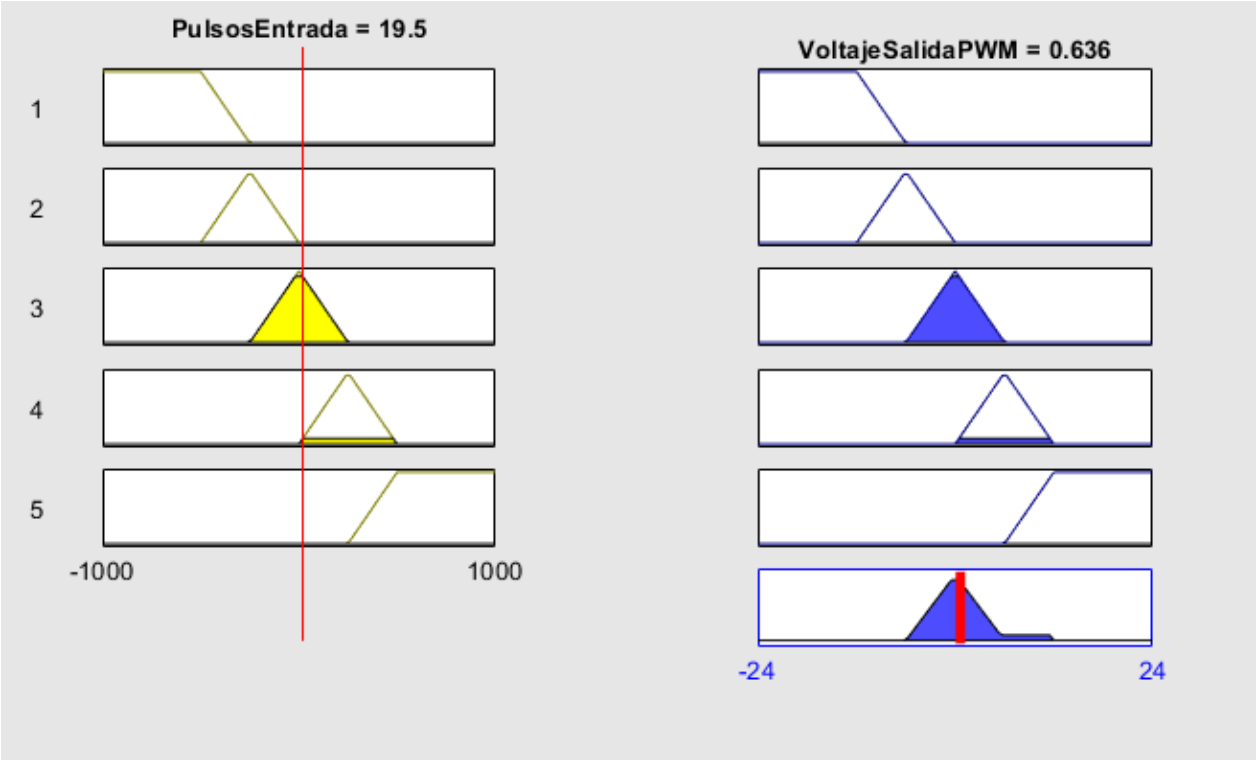

Figura 142. Reglas del controlador

Como se observa en la figura se presenta las reglas del controlador y la desfusificacion del mismo, como se observa las 5 reglas de entrada tienen una respuesta en la salida, es decir se tendrá el valor de tensión y el sentido al que se deberá mover el motor para cumplir los pulsos que se ingresó.

8.3. Descripción de la cinemática

La cinemática de un manipulador robótico estudia el movimiento del mismo con respecto a un sistema de referencia. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares. (Barrientos, Peñín, Balaguer, & Aracil, 1997).

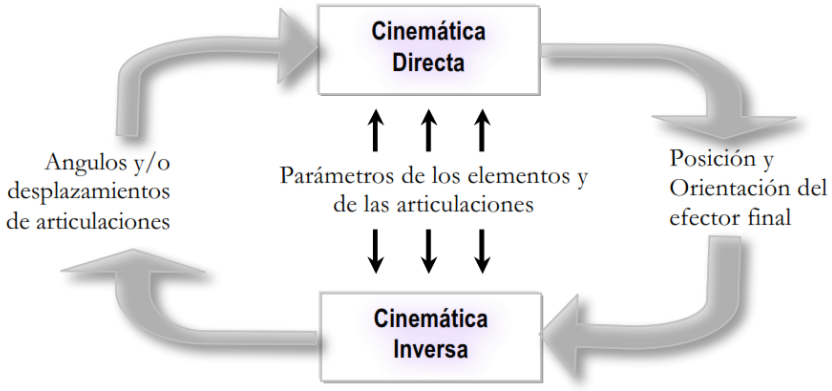

Figura 143. El problema cinemático

Fuente: (Jaramillo, 2005)

8.3.1. Cinemática directa.

"Determina la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas de referencia, conocidos los ángulos de las articulaciones y los parámetros geométricos de los elementos del robot" (Hernández, 2013).

Para el análisis de la cinemática directa en el manipulador robótico de configuración angular CRS A-255, de cinco grados de libertad, se utiliza el algoritmo de Denavit-Hartenberg.

El algoritmo de Denavit-Hartenberg es un método matricial que permite establecer de manera sistemática un sistema de coordenadas, para cada eslabón del brazo robot. Así mediante transformaciones de rotación y traslación se pueden conocer las coordenadas cartesianas (X_0, Y_0, Z_0) de la herramienta, referido a un sistema de coordenadas fijo con base en los ángulos de todos los elementos del brazo robot. (Medina, Villafuerte, & Mejía, 2016).

Por lo tanto, para el brazo robótico CRS A-255 de cinco grados de libertad presentado en este proyecto, la cinemática directa se reduce a encontrar la matriz de transformación homogénea ${}^0T_5 = {}^0A_1, {}^1A_2, {}^2A_3, {}^3A_4, {}^4A_5$

En la siguiente figura se muestra representación gráfica de la vista superior del brazo robótico, en la que se indica que no existen distancias verticales entre eslabones, de igual manera están señaladas las distancias de brazo, antebrazo y muñeca, y con un círculo amarillo, la posición espacial del TCP (Tool Center Point), él cual es el extremo final del robot, en donde se conecta con la herramienta.

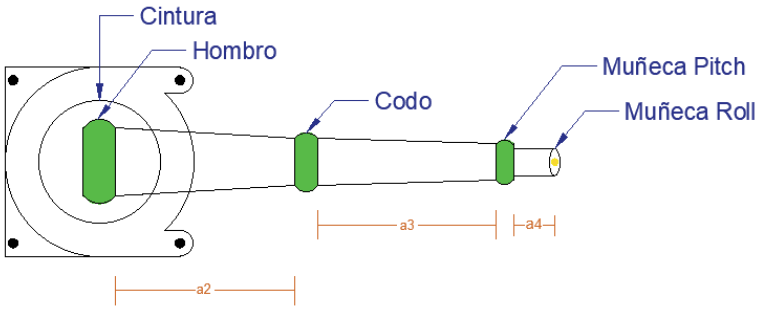

Figura 144. Representación gráfica de la vista superior del brazo robótico CRS A-255

En la siguiente figura se muestra una representación gráfica de la vista lateral del brazo robótico, donde se indican las diferentes dimensiones de los eslabones, y los diferentes ángulos de cada articulación, los cuales son necesarios para el análisis cinemático directo.

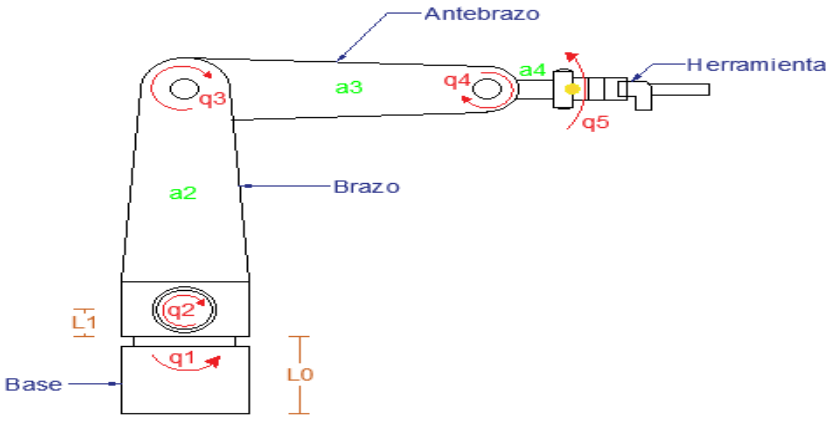

Figura 145. Representación gráfica de la vista lateral del brazo robótico CRS A-255

Con las dos figuras anteriores se puede realizar un gráfico simbólico del brazo robótico de cinco grados de libertad. Las articulaciones son representadas con cilindros, en cambio los eslabones son representados únicamente con una línea que conecta a las articulaciones del robot manipulador, de igual manera están señaladas las distancias entre articulaciones e incluso está incluido los sistemas de coordenadas seleccionados, en cada una de las articulaciones.

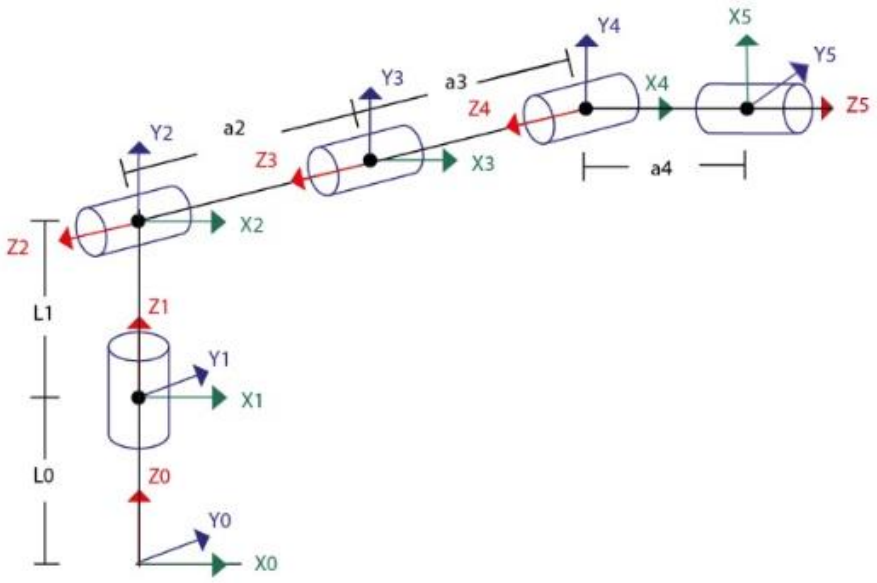

Figura 146. Representación simbólica del robot CRS A-255

La representación simbólica del robot CRS A-255, permite facilitar el análisis de los movimientos del robot y agilizar el proceso de obtención de los parámetros de Denavit-Hartenberg. Al aplicar el algoritmo de Denavit-Hartenberg se llenan los parámetros que se presentan en la siguiente tabla, los cuales sirven para obtener las diferentes matrices de transformación homogénea.

Tabla 39
Parámetros Denavit-Hartenberg del manipulador propuesto

Articulaciones	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{a_i}$	$\boldsymbol{\alpha_i}$
1	q_1	$L_0 + L_1$	0	90
2	q_2	0	a_2	0
3	q_3	0	a_3	0
4	q_4	0	a_4	0
5	q_5	0	a_5	90

Matriz de transformación homogénea general

$$^{i-1}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ Sq_1 & C\alpha_i C\theta_i & S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de transformación de la cintura

$$^{0}A_1 = \begin{bmatrix} Cq_1 & 0 & Sq_1 & 0 \\ Sq_1 & 0 & -Cq_1 & 0 \\ 0 & 1 & 0 & L0 + L1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de transformación del hombro

$$^{1}A_2 = \begin{bmatrix} Cq_2 & -Sq_2 & 0 & a2 * Cq_2 \\ Sq_2 & Cq_2 & 0 & a2 * Sq_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de transformación del codo

$$^{2}A_3 = \begin{bmatrix} Cq_3 & -Sq_3 & 0 & a3 * Cq_3 \\ Sq_3 & Cq_3 & 0 & a3 * Sq_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de transformación de la muñeca pitch

$$^{3}A_4 = \begin{bmatrix} Cq_4 & -Sq_4 & 0 & a4 * Cq_4 \\ Sq_4 & Cq_4 & 0 & a4 * Sq_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de transformación de la muñeca roll

$$^{4}A_5 = \begin{bmatrix} Cq_5 & 0 & Sq_5 & 0 \\ Sq_5 & 0 & -Cq_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Al realizar la multiplicación de las matrices y simplificando términos se obtiene la matriz de transformación homogénea ${}^0T_5$, la misma que será una matriz de cuatro filas y cuatro columnas.

$$ {}^0T_5 = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{23} & a_{33} & a_{34} \\ a_{41} & a_{24} & a_{43} & a_{44} \end{bmatrix} $$

En dicha matriz, los elementos a_{14}, a_{24}, a_{34} describen las coordenadas cartesianas ${}^0X_5$, ${}^0Y_5$, ${}^0Z_5$ de la localización espacial del extremo final del brazo robótico es decir la traslación, en cambio los elementos $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33}$ representan la matriz de rotación del robot.

8.3.2. Cinemática inversa.

Determina la configuración que debe adoptar el robot es decir hallar los valores de las coordenadas articulares del robot, para una posición y orientación del extremo del robot conocidas (Hernández, 2013).

A diferencia del problema cinemático directo, en este caso no existe una manera sistemática de obtener una solución, siendo el procedimiento de obtención fuertemente dependiente de la configuración del robot. Por este motivo, para obtener las ecuaciones que permitan calcular los ángulos requeridos de cada una de las articulaciones del brazo robótico CRS A-255, se utiliza el método geométrico.

En la siguiente figura se puede observar una vista 3D del robot angular, en donde se aprecia que tanto X_0 como Y_0 forman un triángulo rectángulo, en donde su hipotenusa, se la denomina "Módulo"; la cual no es más que la distancia horizontal desde el origen de coordenadas hasta el extremo final del brazo robótico. Se aprecia de igual manera uno de los ángulos internos del triángulo es q_1, que es el ángulo rotacional de la cintura.

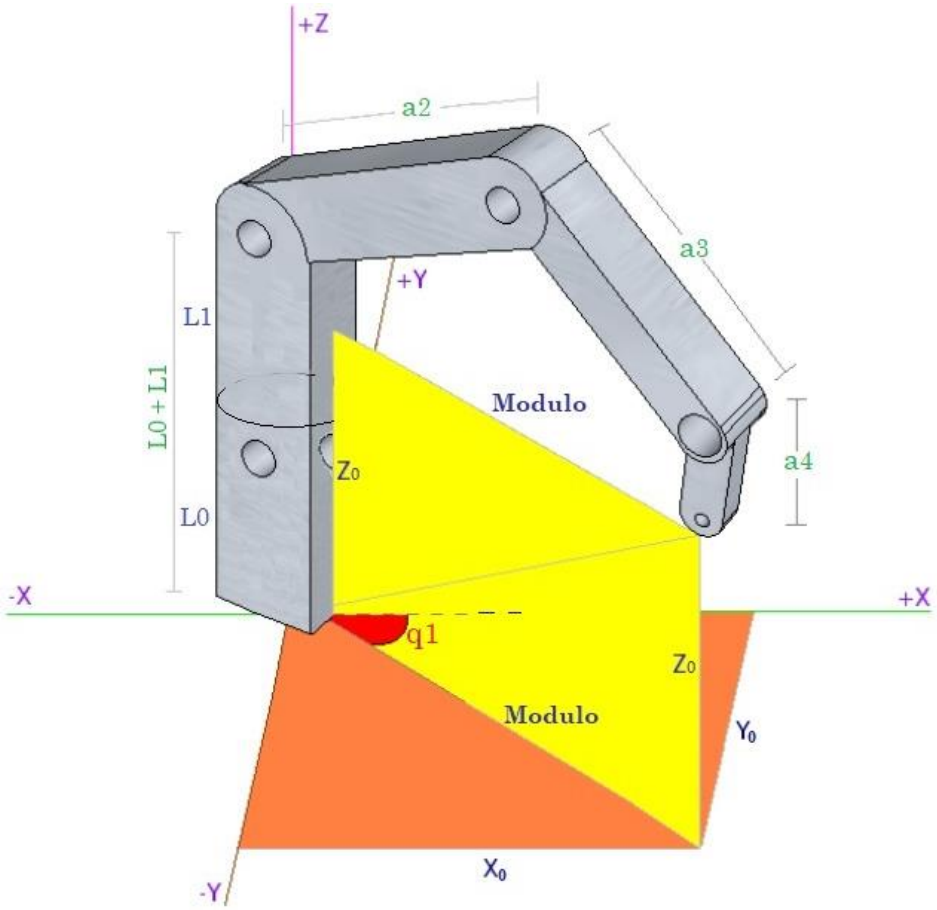

Figura 147. Vista 3D del brazo robótico CRS A-255

El ángulo de la cintura q_1, se observa que es un ángulo interno de un triángulo rectángulo como se aprecia en la siguiente figura.

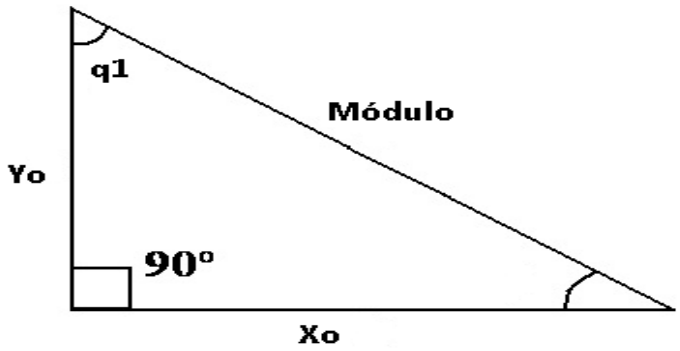

Figura. 148 Triángulo rectángulo formado

por las coordenadas iniciales de la base del robot

Por lo cual el ángulo q_1 se calcula de la siguiente manera:

$$\boldsymbol{q_1 = \tan^{-1}\left(\frac{Y_0}{X_0}\right)}$$

Se puede deducir:

$$Modulo = \sqrt{X_0^2 + Y_0^2}$$

En la siguiente figura se muestra otra vista lateral del brazo robótico, donde se indican otros parámetros necesarios que permitirán establecer las ecuaciones para el cálculo de los ángulos de las articulaciones faltantes.

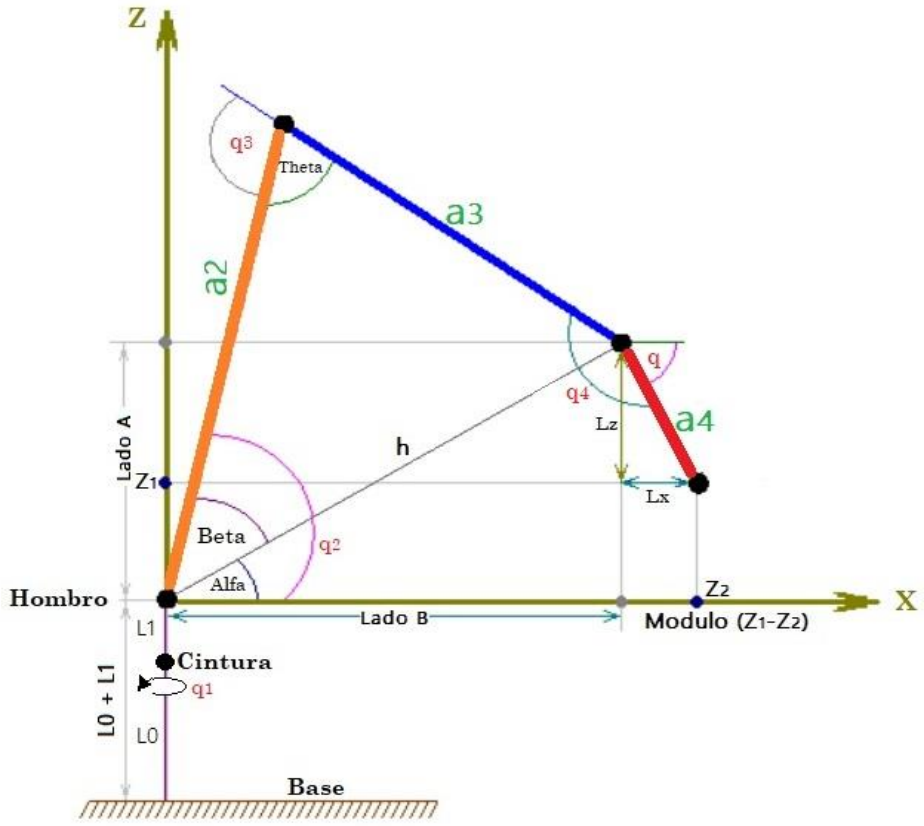

Figura 149. Representación gráfica de la vista lateral del brazo robótico con parámetros para cinemática inversa

En la anterior figura se puede observar un ángulo q el cual es conocido como ángulo de la herramienta; dicho ángulo permanece constante y su valor debe ser escogido por el usuario. Continuando con la resolución de la cinemática inversa se plantean las siguientes ecuaciones, las cuales son deducidas de la anterior figura.

El valor de L_x es:

$$L_x = a_4 \cdot \cos(q)$$

El valor de L_z es:

$$L_z = a_4 \cdot \sin(q)$$

$Lado\_B$ se calcula como:

$$Lado\_B = Modulo - L_x$$

El valor de $Lado\_A$ se obtiene a partir de:

$$Lado\_A = Z_0 - L_z - (L_0 + L_1)$$

h se calcula como:

$$h = \sqrt{Lado\_A^2 + Lado\_B^2}$$

El valor de $Alfa$ es:

$$Alfa = \tan^{-1}\left(\frac{Lado\_A}{Lado\_B}\right)$$

El valor de $Beta$ se obtiene a través de ley de cosenos:

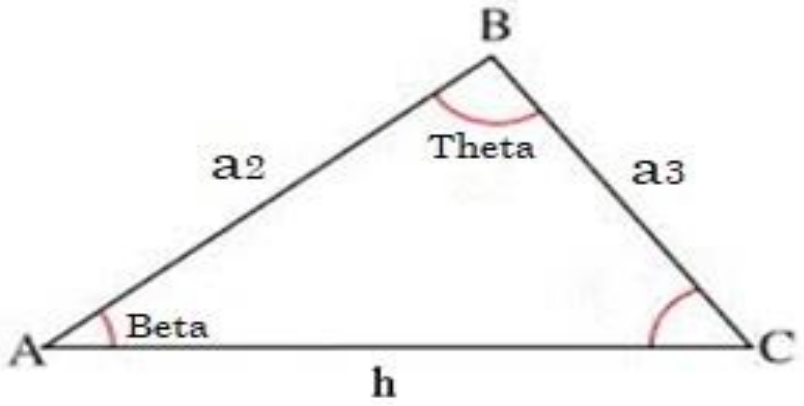

Figura 150. Triángulo formado por el brazo, antebrazo e hipotenusa del brazo robótico

$$Beta = \cos^{-1}\left(\frac{a_2^2 - a_3^2 + h^2}{2 \cdot a_2 \cdot h}\right)$$

Así, el ángulo de rotación q_2 del hombro del robot Angular es:

$$\boldsymbol{q_2 = Alfa + Beta}$$

El valor de $Theta$ también se obtiene por ley de cosenos del mismo triángulo anterior:

$$Theta = \cos^{-1}\left(\frac{a_2^2 + a_3^2 - h^2}{2 \cdot a_2 \cdot a_3}\right)$$

Entonces, el ángulo de rotación q_3 del codo del robot Angular es:

$$\boldsymbol{q_3 = Theta - 180°}$$

Por último, el ángulo de rotación de la muñeca pitch q_4 es:

$$\boldsymbol{q_4 = q - q_2 - q_3}$$

El robot manipulador CRS A255 cuenta con dependencias de eslabones y movimientos en articulaciones; en total tiene tres dependencias las cuales son:

- El eslabón del antebrazo depende del brazo.
- El eslabón que une la muñeca pitch con la muñeca roll depende del antebrazo.

- El movimiento de la muñeca roll se ve influenciado por el movimiento de la muñeca roll

Para que dichas dependencias no afecten a la ejecución correcta de la cinemática directa e inversa del brazo robótico, es necesario realizar una compensación en los ángulos del codo, la muñeca pitch y la muñeca roll, las cuales se muestran a continuación:

$$ang3 = q2 + q3$$

$$ang5 = ang3 + q5$$

$$ang4 = -\left(\frac{ang5}{2}\right)$$

8.4. Explicación del código fuente del controlador

El programa del microcontrolador ATxmega64D3 fue escrito en un compilador básico llamado BASCOM-AVR 2.0.7.8, el cual es un compilador original de Windows para la familia AVR. El lenguaje de programación que utiliza BASCOM AVR es BASIC, el cual es un lenguaje estructurado con etiquetas.

A continuación, se describe el código escrito en BASCOM-AVR 2.0.7.8 para el microcontrolador -ATxmega64D3.

```
$regfile = "xm64D3def.dat"          'Tipo de microcontrolador a usar (ATxmega64D3)
$crystal = 32000000                 'Cristal a usar (32MHz)
$hwstack = 230                      'Tamaño de la pila de hardware
$swstack = 180                      'Tamaño de la pila de software
$framesize = 180                    'Tamaño del marco o del frame se lo utiliza para Print, LCD, input, FP y Format
```

Figura 151. Configuraciones iniciales para el microcontrolador ATxmega64D3

```
Config Osc = Disabled , Extosc = Enabled , Pllosc = Enabled , Range = 12mhz_16mhz ,
             Startup = Xtal_256clk , Pllsource = Extclock , Pllmul = 2

Config Sysclock = Pll , Prescalea = 1 , Prescalebc = 1_1     'EXTERNAL 16MHz
Do
Loop Until OSC_STATUS.4 = 1                                  'Check if RC2MRDY is ready

Config Priority = Static , Vector = Application , Lo = Enabled , Hi = Enabled

Config Com1 = 9600 , Mode = Asynchroneous , Parity = None , Stopbits = 1 , Databits = 8
Open "COM1:" For Binary As #1
```

Figura 152. Configuraciones específicas para el microcontrolador ATxmega64D3

La primera configuración específica, es la configuración del oscilador, básicamente se deshabilita el oscilador interno de 2Mhz, se habilita el oscilador externo, de igual manera se habilita la pila del oscilador a través de una señal del oscilador externo, dicha pila permite mantener un correcta fase y reloj en la comunicación, después se coloca el rango del oscilador externo el cual será de 12 a 16MHz, se específica un tipo de cristal de 0.4-16 MHz para 256 de CLK (señal de reloj).

La segunda configuración específica, es la del oscilador usado para la generación del sistema de reloj, se coloca un Sysclock igual al Pll es decir igual a la pila del reloj del oscilador externo, además se señala un prescaler de uno, que es la división del reloj, se observa que para los tres prescaler A, B y C se coloca uno.

A continuación, se coloca un Do Loop, donde se indica, que el estado del oscilador debe ser estable para comenzar a ejecutar las siguientes líneas de programación.

De igual manera se configura las interrupciones, colocando la palabra static, las interrupciones se comportan como un chip Xmega, se escoge application lo cual permite que los vectores de interrupción se coloquen en la dirección 0, finalmente se habilita las interrupciones bajas y de alta prioridad debido a que las medias ya están habilitadas por defecto.

En la penúltima línea de código se configura la comunicación UART, en donde se coloca que se va a usar el Com1 uno a 9600 baudios, con una operación asíncrona, con ninguna paridad, con un bit de paro y con ocho bits de datos. Ya en la última línea de código se abre el COM1.

```
On Porta_int0 Encoder1 : Enable Porta_int0 , Lo
On Porta_int1 Encoder2 : Enable Porta_int1 , Lo
On Portb_int0 Encoder3 : Enable Portb_int0 , Lo
On Porte_int0 Encoder4 : Enable Porte_int0 , Lo
On Portf_int0 Encoder5 : Enable Portf_int0 , Lo

Config PORTA.0 = Input : Config Xpin = PORTA.0 , Sense = Rising
Config PORTA.2 = Input : Config Xpin = PORTA.2 , Sense = Rising
Config PORTB.0 = Input : Config Xpin = PORTB.0 , Sense = Rising
Config PORTE.6 = Input : Config Xpin = PORTE.6 , Sense = Rising
Config PORTF.6 = Input : Config Xpin = PORTF.6 , Sense = Rising

Config Xpin = PORTA.1
Config Xpin = PORTA.3
Config Xpin = PORTB.1
Config Xpin = PORTE.7
Config Xpin = PORTF.7

PORTA_INT0MASK = &B0000_0001
PORTA_INT1MASK = &B0000_0100
PORTB_INT0MASK = &B0000_0001
PORTE_INT0MASK = &B0100_0000
PORTF_INT0MASK = &B0100_0000
```

Figura 153. Puertos del microcontrolador ATxmega64D3 para interrupciones de los encoders

En la anterior figura se observa todas las configuraciones necesarias para las interrupciones de los encoders, debido a que cada robot tiene 5 encoders y cada uno de ellos tiene dos hilos A y B, se necesita en total configurar 10 puertos del microcontrolador, hay que tener en cuenta que cada puerto del microcontrolador ATxmega64D3 tiene dos interrupciones por lo que se ha usado la interrupciones cero y uno del puerto A, la interrupción cero del puerto B, E y F. No hay que olvidarse de establecer las máscaras de interrupciones como se observa en las últimas líneas de código, dichas máscaras de interrupciones son los pines que serán activados para generar las interrupciones.

```
Config Serialin = Buffered , Size = 20
'Enable Urxc , Hi
Enable Usartc0_rxc , Hi
```

Figura 154. Configuraciones de comunicación del microcontrolador ATxmega64D3

Se configura el serial0 es decir el que se encuentra en el puerto C, y se coloca el tamaño del buffer de entrada a la SRAM a 20, el máximo tamaño que se puede colocar es de 255. En la última línea de código se habilita en alto el Urxc, la cual es una interrupción de recepción.

```
Config PORTA.4 = Output    'Freno1
Config PORTA.5 = Output    'Freno2
Config PORTA.6 = Output    'Freno3
Config PORTA.7 = Output    'Freno4
Freno1 Alias PORTA.4
Freno2 Alias PORTA.5
Freno3 Alias PORTA.6
Freno4 Alias PORTA.7

'--------------------Inicializacion de variables
Freno1 = 0
Freno2 = 0
Freno3 = 0
Freno4 = 0
'-----------------------------------------------

Config PORTE.0 = Output    'Direccion1
Config PORTE.1 = Output    'Direccion2
Config PORTE.2 = Output    'Direccion3
Config PORTE.3 = Output    'Direccion4
Config PORTF.0 = Output    'Direccion5
Config PORTF.1 = Output    'Direccion6

Direccion1 Alias PORTE.0
Direccion2 Alias PORTE.1
Direccion3 Alias PORTE.2
Direccion4 Alias PORTE.3
Direccion5 Alias PORTF.0
Direccion6 Alias PORTF.1
```

Figura 155. Configuraciones de puertos para frenos y dirección de los diferentes motores

Se configura cuatro puertos para los frenos y se los coloca como puertos de salida; son solo cuatro debido a que la cintura no posee freno, a todos estos puertos se los inicializa en cero lógico para que ningún freno del robot se desactive antes de iniciar el movimiento de sus articulaciones. De igual manera se configura seis puertos para dirección debido a que son cinco articulaciones y la sexta dirección corresponde al motor del gripper o herramienta.

```
Dim Enc1 As Long
Dim Enc2 As Long
Dim Enc3 As Long
Dim Enc4 As Long
Dim Enc5 As Long

Dim Enc1a As Long
Dim Enc2a As Long
Dim Enc3a As Long
Dim Enc4a As Long
Dim Enc5a As Long

Dim Enc1str As String * 30
Dim Enc2str As String * 30
Dim Enc3str As String * 30
Dim Enc4str As String * 30
Dim Enc5str As String * 30

Dim Dato As String * 30
Dim Dato1 As String * 30

Dim pulso1 As Long
Dim pulsoActual1 As Long
Dim pulsoAn1 As Long
Dim pulsoDe1 As Long
Dim cintura As Integer

Dim pulso2 As Long
Dim pulsoActual2 As Long
Dim pulsoAn2 As Long
Dim pulsoDe2 As Long
Dim hombro As Integer

Dim pulso3 As Long
Dim pulsoActual3 As Long
Dim pulsoAn3 As Long
Dim pulsoDe3 As Long
Dim codo As Integer

Dim pulso4 As Long
Dim pulsoActual4 As Long
Dim pulsoAn4 As Long
Dim pulsoDe4 As Long
Dim mu_roll As Integer

Dim pulso5 As Long
Dim pulsoActual5 As Long
Dim pulsoAn5 As Long
Dim pulsoDe5 As Long
Dim mu_pitch As Integer

Const c1 = 3
```

Figura 156. Configuración de variables

Al principio se configura las variables que se utilizará en la lectura de los encoders, después se configura la variable Dato y Dato1 en donde se guardará toda la información enviada desde la Raspberry Pi hacia el microcontrolador a través de la comunicación UART, las siguientes variables a configurar son las que se utilizará al momento de mover cada una de las articulaciones del robot

(cintura, hombro, codo, muñeca roll y muñeca pitch), por último se tiene una constante c1 la cual es el rango de fallo entre el pulso deseado, y el pulso obtenido.

```
'-------------Inicializacion de variables
pulsoAn1 = 0
pulsoDe1 = 0
pulsoActual1 = 0
cintura = 0

pulsoAn2 = 0
pulsoDe2 = 0
pulsoActual2 = 0
hombro = 0

pulsoAn3 = 0
pulsoDe3 = 0
pulsoActual3 = 0
codo = 0

pulsoAn4 = 0
pulsoDe4 = 0
pulsoActual4= 0
mu_roll = 0

pulsoAn5 = 0
pulsoDe5 = 0
pulsoActual5= 0
mu_pitch = 0
```

Figura 157. Inicialización de variables

Se inicializan las variables que se van a utilizar para el movimiento de las diferentes articulaciones del robot en cero.

```
Config PORTC.0 = Output    'PWM1
Config PORTC.1 = Output    'PWM2
Config PORTC.4 = Output    'PWM3
Config PORTC.5 = Output    'PWM4
Config PORTD.0 = Output    'PWM5
Config PORTD.1 = Output    'PWM6

Config Tcc0 = Pwm_topbot , Prescale = 8 , Comparea = Enabled , Compareb = Enabled , Resolution = 8
Config Tcc1 = Pwm_topbot , Prescale = 8 , Comparea = Enabled , Compareb = Enabled , Resolution = 8
Config Tcd0 = Pwm_topbot , Prescale = 8 , Comparea = Enabled , Compareb = Enabled , Comparec = Enabled
              , Compared = Enabled , Resolution = 8

'Tcc0_per = &H7FFF
TCC0_PER = &HFF
'Tcc0_cca = 10000
'Tcc0_ccb = 10000

TCC1_PER = &HFF
'Tcc1_cca = 10000
'Tcc1_ccb = 10000

TCD0_PER = &HFF
'Tcd0_cca = 10000
'Tcd0_ccb = 10000
'Tcd0_ccc = 10000
'Tcd0_ccd = 10000
```

Figura 158. Configuración de salidas PWM

Se configura seis puertos como salida tipo PWM para los seis motores que posee el brazo robótico incluido el gripper o herramienta, Tcc0 corresponde tanto al puerto C0 y C1, Tcc1 en cambio corresponde al puerto C4 y C5, por último, Tcd0 corresponde al puerto D0 y D1. Todas las salidas PWM están configuradas en el rango de 0 a 255.

```
Enable Interrupts

'LECTURA PULSOS
Do
   S = Inkey(#1)
   If S = "<" Then
      Input Dato Noecho
      Dato1 = Mid(dato , 1 , 2)
      If Dato1 = "P1" Then
         Dato1 = Mid(dato , 3 , 6)
         pulso1 = Val(dato1)
         pulsoAn1 = pulso1-c1
         pulsoDe1 = pulso1+c1
         cintura = 1
      End If

      If Dato1 = "P2" Then
         Dato1 = Mid(dato , 3 , 6)
         pulso2 = Val(dato1)
         pulsoAn2 = pulso2-c1
         pulsoDe2 = pulso2+c1
         hombro = 1
      End If
```

Figura 159. Lectura de pulsos desde la Raspberry Pi hacia el microcontrolador

Se habilita las interrupciones y se abre el bucle Do Loop en donde se ingresa todo el código del programa. Se puede observar que se crea una sentencia If en donde se indica, que cuando el COM1 es decir S reciba un < la variable Dato guardará la información que se esté enviando en ese momento, de la información enviada se coge los valores desde la posición uno hasta la dos; después si dicha información que se escogió corresponde a P1, P2, P3, P4, P5, P6, P7 o P8 se ingresará en las diferentes sentencias If de cada una de las articulaciones dentro de ellas se activa una variable la cual indica el inicio del movimiento de dicha articulación.

```
'Encoder Cintura
  If Enc1 < 0 Then
    Enc1a = 0 - Enc1
    Enc1str = Str(enc1a)
    Enc1str = Format(enc1str , "000000")
    Enc1str = "-" + Enc1str
    pulsoActual1 = Val(Enc1str)
  Else
    Enc1str = Str(enc1)
    Enc1str = Format(enc1str , "000000")
    Enc1str = "+" + Enc1str
    pulsoActual1 = Val(Enc1str)
  End If
```

Figura 160. Lectura del encoder ubicado en la cintura del brazo robótico

La variable Enc1 incrementa o decrementa dependiendo la señal cuadrada del pin A del encoder la cual se está leyendo constantemente en la interrupción Encoder1, por ende, la sentencia If lee dicho dato y lo único que hace es colocar un signo negativo en la lectura del encoder si el dato es menor que cero, caso contrario le coloca un signo positivo. Este procedimiento se realiza para los cuatro encoders restantes.

```
'Cintura
  If cintura = 1 Then
  If pulsoActual1 <=pulsoDe1 AND pulsoActual1 >=pulsoAn1 Then
    TCC0_CCA = 0
    Direccion1 = 1
    cintura = 0
  Elseif pulsoActual1 < pulsoDe1 Then
    Direccion1 = 0
    TCC0_CCA = 100
  Else
    Direccion1 = 1
    TCC0_CCA = 100
  End If

  Else
    TCC0_CCA = 0
    Direccion1 = 1
    pulsoDe1 = 0
    pulsoAn1 = 0
  End if
```

Figura 161. Movimiento de la cintura del brazo robótico

Se observa una sentencia If, la cual se ejecuta al momento de que la variable cintura se encuentre en uno, luego la siguiente sentencia if compara los pulsos actuales del encoder de la cintura con los pulsos deseados, si dichos pulsos están iguales o con un pequeño rango de diferencia, los pines de

control tanto de dirección uno, y PWM uno se mantienen en cero, por otro lado si los pulsos actuales son menores que los pulsos deseados el pin de dirección uno se mantiene en cero y el pin de PWM uno en 100, caso contrario si los pulsos actuales serían mayor a los pulsos deseados el pin de dirección uno se mantiene en uno y el pin de PWM uno en 100. Este procedimiento se realiza para las cuatro articulaciones restantes.

```
Loop

End

Encoder1:
  If PINA.1 = 0 Then Incr Enc1
  If PINA.1 = 1 Then Decr Enc1
Return
```

Figura 162. Interrupción Encoder1 para la articulación cintura del brazo robot

En la anterior figura se observa el final del bucle Do Loop, después se finaliza el programa con la palabra End, ya finalizado el programa, se colocan las diferentes interrupciones, en este caso se observa la interrupción llamada Encoder1, la cual lee el pin A1 en donde está conectado el hilo A del encoder de la cintura, dicha interrupción es atendida cada flanco ascendente de la señal cuadrática del hilo B del enconder de la cintura. Aumentando o disminuyendo la variable Enc1 dependiendo si el pin A1 detecta un cero o un uno lógico.

CAPÍTULO IX

DISEÑO Y CONSTRUCCIÓN DE LA CELDA DE TRABAJO

9.1. Cambios en la mesa de trabajo

En el Capítulo 1 se establece ya una celda de trabajo con la que se va a contar en el desarrollo del proyecto, pero debido a falta de espacio físico y la entrega de una mesa de trabajo amplia en la cual se puedan instalar los dos manipuladores, se tuvo que rediseñar la celda robótica que ocuparían los manipuladores al igual que el tipo de procesos que se efectuarían por el daño mecánico de un manipulador el mismo que se describe en el ítem 3.5. Sin embargo, es importante mencionar que a pesar del daño mecánico del manipulador fue considerado su espacio físico como sus conexiones eléctricas dentro de esta nueva mesa de trabajo, la misma que se describirá en los siguientes apartados.

9.2. Mesa de trabajo

La mesa de trabajo estará conformada por dos robots CRS A-255 colocados uno frente al otro, estos robots se encontrarán en su posición home con la finalidad de cumplir la cinemática que se expuso en el capítulo anterior y efectuar la simulación de procesos industriales. En la parte inferior estará ubicada la caja de control que alimenta y controla a cada uno de los manipuladores, la mesa de trabajo estará conformada de un pistón neumático con su respectiva electroválvula para que pueda sacar las piezas como se observa en la siguiente figura.

Figura 163. Mesa de trabajo de los CRS A255

9.2.1. Piezas de trabajo.

Las piezas de trabajo que moverán los manipuladores robóticos para cumplir los procesos serán tres: unos de forma rectangular de un ancho de dos centímetros, huevos ovalados y círculos de dos centímetros de ancho para poder ser sostenidos por las pinzas que tienen cada manipulador y ser trasladadas para ser expulsadas mediante el pistón.

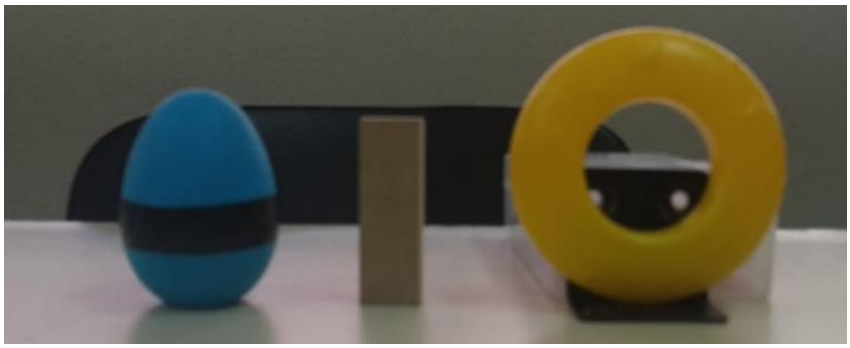

Figura 164. Piezas de trabajo

9.2.2. Robots.

En el Capítulo 3 se detalla el estado de los manipuladores robóticos a utilizar en la celda de trabajo y su funcionamiento, como se puede observar en la figura 162 estos manipuladores están distribuidos de forma simétrica en la mesa de trabajo para poder cumplir con los procesos que se les impone, para poder asegurar el funcionamiento de los mismos y garantizar su adecuado movimiento se los fija mediante pernos a la mesa de trabajo con la finalidad de brindar estabilidad y evitar errores en la ejecución de procesos.

9.2.3. Elementos neumáticos.

Para expulsar las piezas de la mesa de trabajo se utilizó un pistón neumático de doble efecto, el mismo que presentó algunos desperfectos debido a que era de segunda mano por lo que fue necesario un mantenimiento en el cambio de los o-rings, los cuales evitan las fugas de aire del pistón.

Figura 165. Pistón de doble efecto YONQ de 1MPa de presión

Para la activación del pistón se utilizó una electroválvula 5/2 con retorno a resorte de 24VDC.

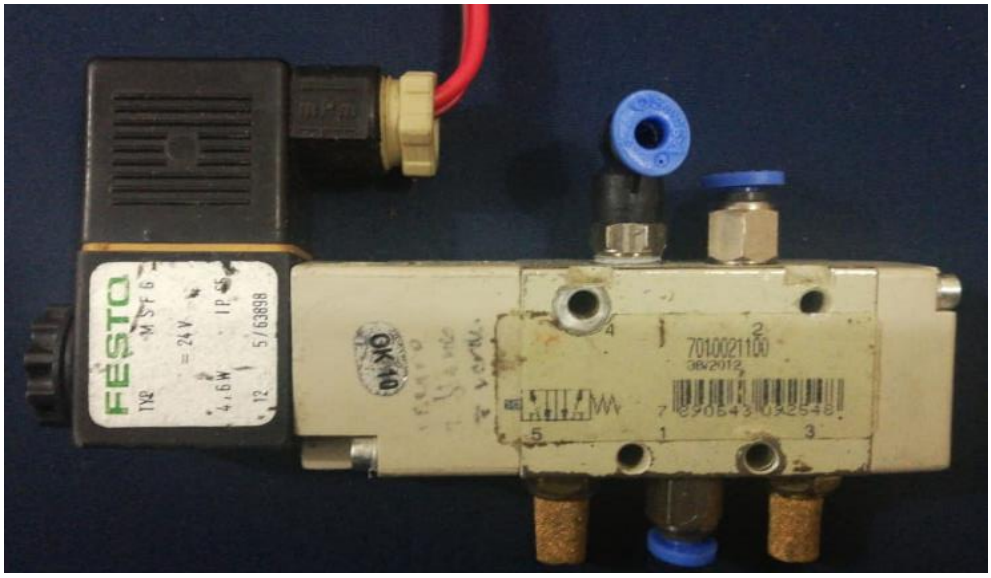

Figura 166. Electroválvula de 5/2 con bobina de marca FESTO

Para entender el funcionamiento del pistón en la mesa de trabajo se realizó un diagrama de bloques, un neumático y un eléctrico que se observa en las siguientes figuras:

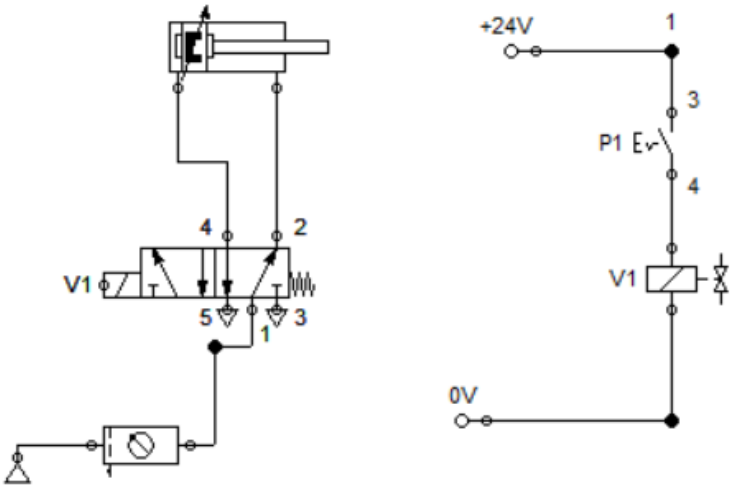

Figura 167. Diagrama de control del pistón instalado en la mesa de trabajo

Como se observa en la figura 167 el pistón de doble efecto está conectado a una electroválvula 5/2, esta electroválvula recibe aire desde una unidad de mantenimiento, la cual está conectada a un compresor. La electroválvula se alimenta con 24 voltios DC, que al momento de recibir el pulso de activación cerrará el circuito y activará el movimiento del pistón como se observa continuación:

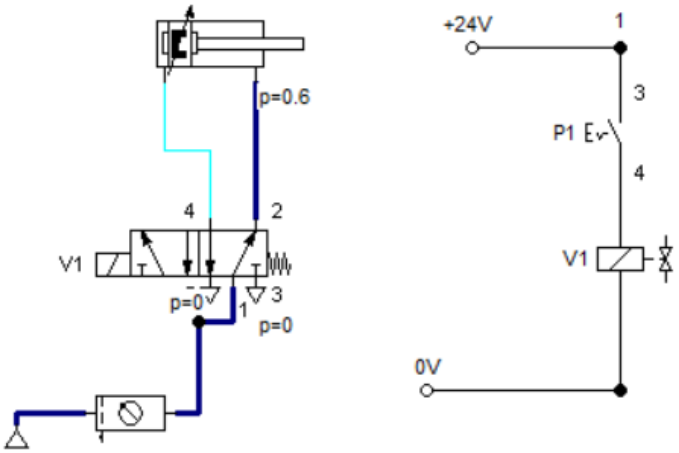

Figura 168. Circuito energizado antes de presionar el pulsador

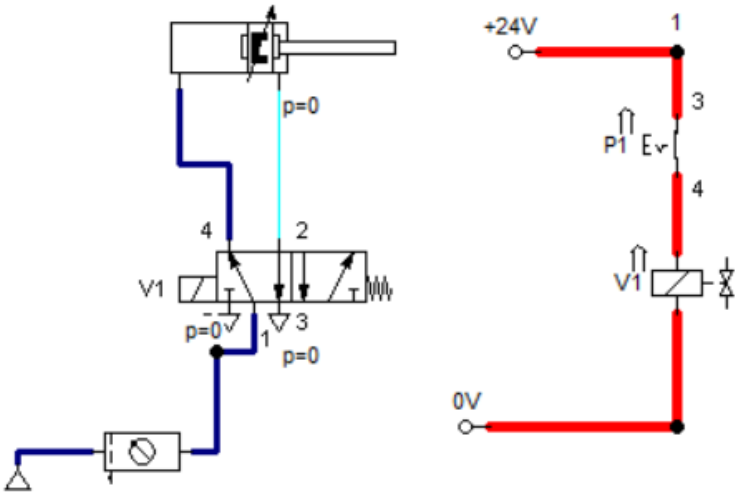

Figura 169. Circuito energizado al momento de recibir el pulso

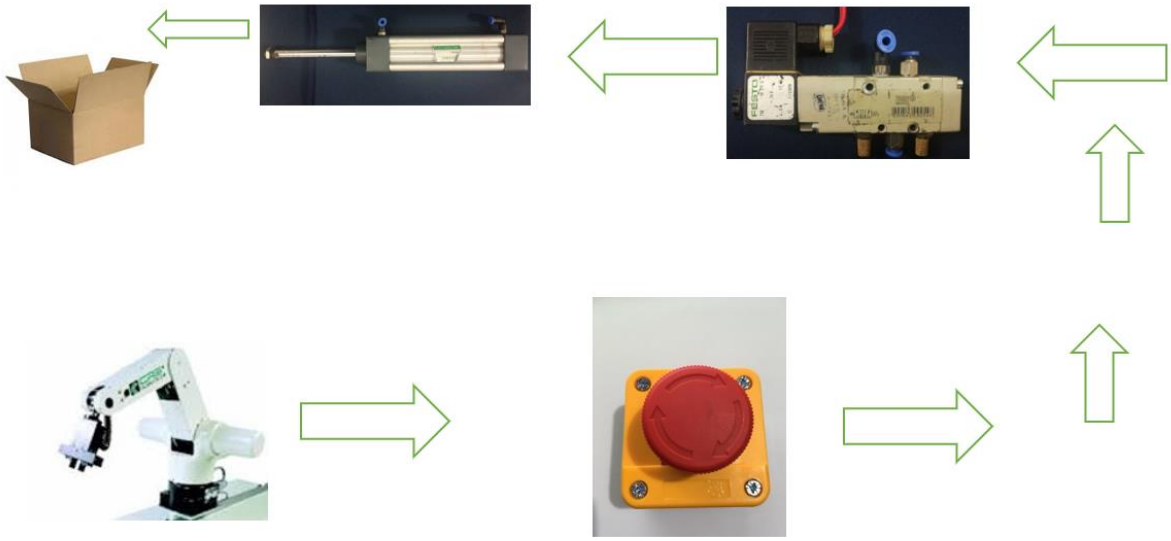

Figura 170. Activación del pistón mediante el robot

En la figura se tiene el control del pistón el mismo que se activará cuando el robot presione el botón, esta señal será transmitida para activar la electroválvula y esta a su vez el pistón para poder empujar la caja que contenga las piezas.

9.2.4. Caja de control.

En el Capítulo 4 se detalla cada uno de los componentes de la caja de control, en la figura se puede ver su colocación dentro de la mesa de trabajo y como alimenta a cada uno de los manipuladores, esta caja cumple con protección y ventilación necesaria dentro de un nivel industrial.

Figura 171. Caja de control en la mesa de trabajo

CAPÍTULO X

PRUEBAS Y RESULTADOS

10.1. Protocolos de comunicación

Para este proyecto se ha utilizado como servidor, la tarjeta Raspberry Pi 3 Model B, la cual se conecta de forma serial, por comunicación UART Full-Duplex hacia los microcontroladores ATxmega64D3 de cada uno de los dos brazos robóticos CRS A-255.

En la celda robotizada se encuentra una cámara IP WiFi, la cual usa el protocolo HTTP para la comunicación entre la página web y la videocámara.

Tanto la cámara como el servidor están conectados al internet de forma inalámbrica a través del protocolo 802.11b, a través de un Access Point, el cual está conectado, con cable ethernet hacia un punto de red.

10.2. Análisis de resultados

Los resultados de este trabajo fueron realizados tomando en cuenta dos aspectos, el desarrollo de la cinemática tanto directa como inversa y la ejecución de los procesos industriales, aclarando que estos resultados son referentes a un solo manipulador debido a que se presentaron daños mecánicos en el otro como se describió en el Capítulo 3; al igual que en el Capítulo 9 se justificó el motivo del cambio de la celada de trabajo tomando en cuentas estos aspectos se efectuará el análisis de resultados. Primero se dará a conocer los resultados obtenidos en la aplicación de la cinemática exponiendo dichos resultados en las siguientes tablas:

Tabla 40
Cinemática directa prueba 1

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
q1=0	X:32	Xexp:31	3.22
q2=90	Y:0	Yexp:0	0
q3=-90	Z:51.5	Zexp:49.5	4
q4=0			
q5=0			

Tabla 41
Cinemática directa prueba 2

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=45	X:25.35	Xexp:24	5.6
q2=80	Y:25.35	Yexp:24	5.6
q3=-70	Z:56.67	Zexp:56	1.19
q4=0			
q5=0			

Tabla 42
Cinemática directa prueba 3

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=-180	X:0	Xexp:0	0
q2=80	Y: -33.67	Yexp: -33	2.03
q3=-70	Z:59.93	Zexp:60	0.11
q4=40			
q5=0			

Tabla 43
Cinemática directa prueba 4

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=-100	X: -8.68	Xexp:8	1.85
q2=40	Y:49.25	Yexp: -50	1.5
q3=-50	Z:40.71	Zexp:40	1.77
q4=40			
q5=0			

Tabla 44
Cinemática directa prueba 5

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=-90	X:42.13	Xexp:42	0.3
q2=60	Y:0	Yexp:0	0
q3=-50	Z:56.96	Zexp:57.5	0.93
q4=40			
q5=50			

Tabla 45
Cinemática directa prueba 6

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=-120	X:18.65	Xexp:18	3.16
q2=70	Y: -32.30	Yexp: -32	0.93
q3=-55	Z:61.35	Zexp:61	-0.57
q4=40			
q5=50			

Tabla 46
Cinemática directa prueba 7

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=-90	X:20.84	Xexp:21	0.76
q2=70	Y:0	Yexp:0	0
q3=-10	Z:78.35	Zexp:80	2.06
q4=40			
q5=50			

Tabla 47
Cinemática directa prueba 8

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=-90	X:0	Xexp:0	0
q2=50	Y: -37.32	Yexp: -36	3.6
q3=-10	Z:68.10	Zexp:68	0.14
q4=40			
q5=40			

Tabla 48
Cinemática directa prueba 9

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=-100	X: -8.48	Xexp:8	0.6
q2=40	Y: -48	Yexp: -47	2.12
q3=-30	Z:51.38	Zexp:51	0.74
q4=40			
q5=60			

Tabla 49

Cinemática directa prueba 10

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
q1=-50	X:33.65	Xexp:33	1.96
q2=30	Y: -40.11	Yexp: -40	0.27
q3=-30	Z:42.53	Zexp:42	1.26
q4=40			
q5=40			

Para conocer el comportamiento de la ejecución de la cinemática directa del robot se realizaron 10 pruebas las cuales fueron expuestas en las tablas anteriores y con la finalidad de conocer el porcentaje de error en la ejecución de la misma se presenta la siguiente tabla:

Tabla 50

Resumen de los resultados en cinemática directa

Error relativo en cinemática directa			
Pruebas	**X**	**Y**	**Z**
Prueba 1	3.22	0	4
Prueba 2	5.6	5.6	1.19
Prueba 3	0	2.03	0.11
Prueba 4	1.85	1.5	1.77
Prueba 5	0.3	0	0.93
Prueba 6	3.16	0.93	0.57
Prueba 7	0.76	0	2.06
Prueba 8	0	3.6	0.14
Prueba 9	0.6	2.12	0.74
Prueba 10	1.96	0.27	1.26
Error relativo total %	**1.74**	**1.61**	**1.27**

Como se observa en la tabla resumen de resultados de cinemática directa los errores relativos en las coordenadas X, Y, Z son por debajo del 2% lo cual nos indica que el controlador del manipulador es eficaz. A continuación, se exponen las pruebas realizadas en la cinemática inversa:

Tabla 51
Cinemática inversa prueba 1

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=0	q1=90	q1ex=90	0
Y=25	q2=88.78	q2ex=88	0.88
Z=70	q3=-44.48	q3ex=-44	1.09
	q4=-44.29	q4ex=-43	3

Tabla 52
Cinemática inversa prueba 2

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=25	q1=-45	q1ex=-44	2.27
Y=-25	q2=78.63	q2ex=78	0.81
Z=60	q3=-58.80	q3ex=58	1.03
	q4=-19.83	q4ex=20	0.85

Tabla 53
Cinemática inversa prueba 3

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=25	q1=0	q1ex=0	0
Y=0	q2=88.78	q2ex=90	1.35
Z=70	q3=-44.48	q3ex=-45	1.15
	q4=-44.29	q4ex=-45	1.57

Tabla 54
Cinemática inversa prueba 4

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=40	q1=-34.99	q1ex=35	0
Y=-28	q2=46.43	q2ex=46	0.93
Z=40	q3=-56.45	q3ex=56	0.8
	q4=10.02	q4ex=11	3.8

Tabla 55
Cinemática inversa prueba 5

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=0	q1=-90	q1ex=90	0
Y=-40	q2=67	q2ex=65	3.07
Z=40	q3=-88.04	q3ex=-90	2.17
	q4=21.03	q4ex=22	2.11

Tabla 56
Cinemática inversa prueba 6

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=33	q1=-50.45	q1ex=50	0.9
Y=-40	q2=33.41	q2ex=33	1.24
Z=42	q3=-36.73	q3ex=36	2.02
	q4=0	q4ex=0	0

Tabla 57

Cinemática inversa prueba 7

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=0	q1=90	q1ex=-90	0
Y=-30	q2=94.50	q2ex=94	0.53
Z=50	q3=-97.57	q3ex=-97	0.51
	q4=3.078	q4ex=4	1.3

Tabla 58

Cinemática inversa prueba 8

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=0	q1=-90	q1ex=-90	0
Y=-30	q2=91.36	q2ex=90	1.51
Z=60	q3=-72.64	q3ex=-71	2.33
	q4=-18.72	q4ex=18	0.88

Tabla 59

Cinemática inversa prueba 9

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo %
X=32	q1=0	q1ex=0	0
Y=0	q2=90	q2ex=89	1.12
Z=51.5	q3=-90	q3ex=-89	1.12
	q4=0	q4ex=0	0

Tabla 60

Cinemática inversa prueba 10

Entradas	Respuesta Teórica	Respuesta Experimental	Error Relativo%
X=0	q1=-90	q1ex=-89	1.12
Y=-37	q2=66.16	q2ex=66	0
Z=65	q3=-30.02	q3ex=-29	1.13
	q4=-36.14	q4ex=-35	3.2

Para conocer el comportamiento de la ejecución de la cinemática inversa del robot se realizaron 10 pruebas las cuales fueron expuestas en las tablas anteriores y con la finalidad de conocer el porcentaje de error en la ejecución de la misma se presenta la siguiente tabla:

Tabla 61

Resumen de los resultados en cinemática inversa

Error relativo en cinemática inversa				
Pruebas	**q1**	**q2**	**q3**	**q4**
Prueba 1	0	0.88	1.09	3
Prueba 2	2.27	0.81	1.03	0.85
Prueba 3	0	1.35	1.15	1.57
Prueba 4	0	0.93	0.8	3.8
Prueba 5	0	3.07	2.17	2.11
Prueba 6	0.9	1.24	2.02	0
Prueba 7	0	0.53	0.51	1.3
Prueba 8	0	1.51	2.33	0.88
Prueba 9	0	1.12	1.12	0
Prueba 10	1.12	0	1.13	3.2
Error relativo total %	**0.33**	**1.14**	**1.34**	**1.67**

Como se observa en la tabla resumen de resultados de cinemática inversa los errores relativos en los ángulos q1, q2, q3, y q4 son por debajo del 2% lo cual nos indica que el controlador del manipulador es eficaz en la ejecución de la cinemática inversa.

En cuanto a los procesos industriales se efectuaron diez pruebas por cada uno de ellos, obteniendo como resultado una ejecución con fallas en al menos dos de las diez pruebas realizadas.

10.3. Trabajos futuros

Existen diferentes aspectos que pueden desarrollarse como posibles trabajos futuros y complementarse con el estudio de este proyecto; a continuación, se nombran algunos de ellos:

- La ejecución paralela de los dos robots CRS A255, a través del cambio de la tarjeta Raspberry Pi, por otra tarjeta que disponga más de un puerto UART, sin olvidar que debe tener la posibilidad de ser usada como servidor y además permitir el uso del lenguaje de programación Python.
- Cambio de los acoples mecánicos entre los motores DC y sus encoders, para que exista una lectura precisa y correcta de pulsos; debido a que en un proyecto anterior de tesis se llevó a cabo la sustitución de tres encoders (muñeca pitch, hombro, codo) y la construcción de sus acoples entre el motor y el encoder sustituido. Lamentablemente estos acoples fueron realizados de plástico a través de impresión 3D, por lo cual tienen un desgaste acelerado, afectando en la lectura de pulsos de dichos encoders.
- La implementación de un modelo dinámico, en cada uno de los robots CRS A-255 para un control más fino en los movimientos del manipulador robótico.
- Modificación del cableado del gripper o pinza, debido a que sus cables de conexión desde el brazo robótico hacia el tablero de control viajan por el mismo enmallado que los otros cables utilizados para la conexión de encoders, similar a un cable UTP, por esta razón existe un

campo magnético generado por el voltaje del gripper, el cual afecta las señales de lectura de los encoders.

CAPÍTULO XI

CONCLUSIONES Y RECOMENDACIONES

11.1. Conclusiones

- El diseño del sistema cloud robotics comprende tanto la parte de hardware como software, con lo que respecta a la parte de software se tiene la unión de varios softwares tales como Python, Basic, Java Script y CSS mediante el framework de Django y el entorno de programación Bascom AVR en cambio lo que respecta a la parte de hardware comprende la tarjeta Raspberry Pi que sirve como servidor y alojamiento de la página web, los puente h IRF3205 permiten el control de los motores a 24 voltios y la tarjeta con el microcontrolador ATxmega64D3 sirve para el control de todo el sistema.
- La ejecución de la cinemática directa e inversa como la realización de procesos industriales desde una página web remota, facilita el control de los mismos, brindando mayor seguridad a sus operarios y un mejor monitoreo de la celda robótica, logrando así la incorporación de nuevas tecnologías como el Internet of Things (IoT).
- Django es un framework muy preciso en el reporte de los errores dentro sus líneas de código, facilitando el trabajo del programador, al no tener que buscar el o los errores dentro del código de programación, sino sólo solucionarlos.
- La eficiencia del controlador fuzzy en conjunto con las buenas prestaciones de la tarjeta de control se pudo evidenciar en el análisis de resultados.
- Al momento de tener lectura de varios encoders es necesario usar un controlador capaz de tener un buen oscilador tanto interno como externo, además es necesario que el mismo tenga la posibilidad del uso de al menos cinco interrupciones, debido a que cada uno de los

brazos robóticos tienen cinco encoders, en los cuales su lectura debe ser permanente, para no perder ningún dato al momento de la lectura, continua de los encoders de cada articulación del brazo robótico.

- El uso del microcontrolador ATxmega64D3, fue una buena elección como tarjeta controladora para cada uno de los brazos robóticos, por sus diferentes prestaciones, como: el número de interrupciones por puerto, por su oscilador externo separado de los osciladores internos precisos con PLL y preescalador, por su arquitectura RISC, por su velocidad de procesamiento MIPS es decir un millón de instrucciones por segundo, por sus puertos de tipo PWM, por sus interfaces UART de 8/16 bits y por contar con 50 entradas y salidas de propósito general.
- El microcontrolador ATxmega64D3 facilitó la lectura continua de los cinco encoders que posee cada brazo robótico, por su cantidad de interrupciones y factibilidad de utilizar un oscilador externo y varios osciladores internos con su respectivo PLL y preescalador.
- La tarjeta Raspberry Pi limita el manejo simultaneo de los brazos robóticos ya que solo dispone de un puerto UART, es así que para poder trabajar con ambos manipuladores fue necesario el uso de un multiplexor CD74HC4052E que permite elegir el puerto UART de las tarjetas controladoras se hará la recepción y transmisión de datos.

11.2. Recomendaciones

- Los brazos robóticos usan encoders incrementales, lo cual es una ventaja y desventaja al mismo tiempo, debido a que por su precio son más económicos que los encoders absolutos, pero su gran desventaja es que antes de ser alimentados se debe posicionar al encoder en un

cero ya predeterminado, en caso de estos manipuladores robóticos, es necesario volver a la posición HOME (cero predeterminado para este proyecto) antes de apagar todo el sistema.

- Es obligatorio el uso de la versión 2.7 de Python o una menor a esta, debido a que, en las versiones posteriores, la librería serial no funciona correctamente, por ende, se debe de utilizar la versión Django 1.11 o una versión menor ya que dichas versiones admiten la versión 2.7 de Python.
- Al momento de usar librerías propias de Python dentro del framework Django se debe de instalar dentro del entorno virtual de Django y no solo en la Raspberry Pi.
- Se debe alimentar la parte de potencia antes de activar y alimentar la parte de control de las tarjetas de Puente H Robot Zumo, ya que esto hará que las mismas no se dañen perdiendo su potencia en la salida.
- Es necesario tener en cuenta que al momento de enviar un dato por la comunicación serial, el microcontrolador lea el dato no bit a bit sino todo el byte. Una forma de corroborar que el microcontrolador está leyendo un byte es con un programa de lectura y escritura entre la Raspberry Pi y el microcontrolador.
- Hay que tener en cuenta que si se necesita el movimiento paralelo de los dos robots es necesario el uso de otra tarjeta que no sea la Raspberry Pi debido a que esta tarjeta solo dispone de un puerto serial.
- Verificar las señales que entregan los diferentes encoders no solo con un multímetro sino con un osciloscopio para ver si dichas señales están dentro del rango, y así no tener falsas lecturas de pulsos al momento de los movimientos de las diferentes articulaciones del robot.
- Siempre es necesario visualizar las señales de los encoders, para visualizar si su rango de trabajo es el correcto, y así no tener falsas lecturas de pulsos.

- Es indispensable revisar el sitio oficial de AVR, al momento de usar un nuevo microcontrolador debido a que existen configuraciones específicas en cada microcontrolador de igual manera entre los microcontroladores Atmel y Atxmega hay algunas diferencias en las instrucciones de programación dentro del programa bascom.
- Se presentó problemas en la instalación de la cámara para supervisión de la celda robótica por políticas de seguridad en la Universidad; por lo que es necesario la creación de una red libre de restricciones, siempre y cuando se tenga el permiso y la asesoría del departamento de Tic’s.
- Para la visualización en tiempo real de la videocámara es necesario el uso del navegador Mozilla Firefox versión 47 o una inferior debido a que el sitio oficial de EZVIZ, no permite su correcto funcionamiento en versiones posteriores a dicho navegador y de igual forma sucede con los navegadores más usados como son: Opera, Google Chrome e Internet Explorer.

BIBLIOGRAFÍA

Aleixo, D. (25 de enero de 2019). *Tutoriales y soluciones*. obtenido de https://help.vtex.com/tutorial/que-son-placeholders?locale=es

Barrientos, A., Peñín, L., Balaguer, C., & Aracil, R. (1997). *Fundamentos de robótica* (primera ed.). Madrid, España: McFRAW-HILL.

Camino, P. (3 de Agosto de 2018). *Qué es Django y por qué usarlo*. obtenido de https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/

Carazo, J. (3 de mayo de 2013). *Diferencia entre apt-get update y apt-get upgrade*. obtenido de http://www.linuxhispano.net/2013/05/03/diferencia-entre-apt-get-update-y-apt-get-upgrade/

Casa, D. (2014). Desarrollo de software para la programación y operación del manipulador róbotico CRS A255. *Artículos Científicos- Carrera de Ingeniería en Electrónica, Automatización y Control*, 1-9.

Civera. (2015). Special issue on cloud robotics and automation. *EEE Transactions on automation science and engineering.*, 1-4.

Conde. (20 de Junio de 2016). *MicroControllerElectronics*. Obtenido de MicroControllers: https://microcontrollerelectronics.com/lm317-3-3v-source/

Corporation, C. R. (2000). *A255 Robot Arm 255.*

Cruz, F. (20 de Agosto de 2015). *Clinic cloud*. Obtenido de Clinic Cloud: https://clinic-cloud.com/blog/breve-historia-la-nube-del-cloud-computing/.

Dorf, R. (2005). *Sistemas de control moderno.* Madrid: Pearson education.

García Saquicela, M. (2010). *Diseño del sistema de control del brazo robótico CRS A255 utilizando la plataforma kinetix de Allen Bradley*. Sangolquí: ESPE.

Hernández, M. (22 de Febrero de 2013). *Cinemática del robot*. Obtenido de https://es.scribd.com/document/126812384/tema4

J.I.Huircan. (2012). *Reguladores de voltaje.* Temuco,Chile: Universidad de la Frontera.

Jaramillo, A. (20 de Junio de 2005). *Cinemática de manipuladores robóticos.* Obtenido de http://www.wag.caltech.edu/home/ajaramil/libro_robotica/cinematica.pdf

Lee, J. (2014). Recent advances and trends of Cyber-Physical systems and Big Data analytics in industrial informatics. *Center for Intelligent Maintenance Systems*, 1-6.

M.R, G. (2010). Diseño del sistema de control del brazo robòtico CRS-A255. *ESPE*.

Maldonado, D. (2013). Estudio, diseño e implementación de un controlador (Hardware y Software) para tres grados de libertad del manipulador robótico CRSA255. *Articulos Cientificos-Carrera de Ingenieria en Electrónica, Automatización y Control*, 1-6.

Medina, J., Villafuerte, R., & Mejía, E. (2016). Simulador 3D para brazo robot de 4 grados de libertad. *Iberoamericana para la Investigación y el Desarrollo Educativo*, 6-11.

Minango, S. (2012). Control de movimiento del manipulador CRS-A255. *IEEE*, 1-5.

Negrete, C. (2018). *Repotenciación del sistema de control y potencia de 2 brazos robóticos CRS A255.* Sangolqui: ESPE.

NeoAttack. (Enero de 2018). *Concepto de iframe*. obtenido de https://neoattack.com/neowiki/iframe/

NeverCracker. (30 de Septiembre de 2017). obtenido de https://nevercrackerblog.wordpress.com/2017/09/30/que-es-ngrok/

Ogata, K. (2000). *Ingeniería de control moderna.* Mexico: Pearson education .

Ortiz, J. (5 de Julio de 2015). *E-educativa*. obtenido de E-educativa: http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio/4750/4926/html/2_controlador_analgico_digital_hbrido_el_ordenador_como_elemento_de_control.html

Pamela, T. (25 de Julio de 2016). *Departamento de Ingenieria Industrial Universidad de la Concepcion*. obtenido de Universidad de la Concepcion: http://www2.udec.cl/~aldconcha/cyp/tipos.html

Romero, C. (18 de Agosto de 2017). *NewFrog*. obtenido de NewFrog: https://www.diyhomedeals.com/es/product/dual-motor-driver-module-board-h-bridge-dc-mosfet-irf3205-3-36v-10a-peak30a-121977

Semle, A. (2016). Protocolos IIOT para considerar. *AADECA*, 1-4.

Shuzhou Wang, H. Z. (2015). Design of an intelligent housekeeping robot based on IOT. *IEEE*, 1-4.

Silva, R. B. (2015). Metamodelo de aprendizaje estratégico (MAE): Arquitectura de la capa de infraestructura, solución basada en la Cloud Computing. *IEEE*, 1-14.

Torres, P. (Julio de 2016). *Departamento de Ingeniería Industrial Universidad de la Concepción*. (Mexico) recuperado el agosto de 2018, de http://www2.udec.cl/~aldconcha/cyp/tipos.html

Tucuru, C. (2012). Integrating robots into the Internet of Things. *IEEE*, 1-8.

uniwebsidad. (2006). *La historia de Django*. Obtenido de https://uniwebsidad.com/libros/django-1-0/capitulo-1/la-historia-de-django

Zapke, M. (2013). *Tu primer proyecto en Django*. obtenido de https://tutorial.djangogirls.org/es/django_start_project/

Zhang, C. T. (2014). A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 194.