UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA INDUSTRIAL EN PROCESOS DE AUTOMATIZACIÓN

Tema:

MONITORIZACIÓN DE PARÁMETROS ELÉCTRICOS DE UN MANIPULADOR MÓVIL

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero Industrial en Procesos de Automatización.

ÁREA: Industrial y Manufactura

LÍNEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de Control

AUTOR: Montenegro Garófalo Frank Janio

TUTOR: Dr. Marcelo Vladimir García Sánchez, Mg

Ambato - Ecuador

agosto – 2021

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación sobre el tema: MONITORIZACIÓN DE PARÁMETROS ELÉCTRICOS DE UN MANIPULADOR MÓVIL, desarrollado bajo la modalidad de Proyecto de Investigación por el señor Frank Janio Montenegro Garófalo, estudiante de la Carrera de Ingeniería Industrial en Procesos de Automatización, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, agosto 2021.

Firmado electrónicamente por:
MARCELO VLADIMIR GARCIA SANCHEZ

Dr. Marcelo Vladimir García Sánchez, Mg.

TUTOR

AUTORÍA

El presente proyecto de investigación titulado: MONITORIZACIÓN DE PARÁMETROS ELÉCTRICOS DE UN MANIPULADOR MÓVIL, es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, agosto 2021.

Frank Janio Montenegro Garófalo

C.C. 150095580-0

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Frank Janio Montenegro Garófalo , estudiante de la Carrera de Ingeniería Industrial en Procesos de Automatización, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad de Proyecto de Investigación, titulado MONITORIZACIÓN DE PARÁMETROS ELÉCTRICOS DE UN MANIPULADOR MÓVIL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Ambato, agosto 2021.

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Pilar Urrutia, Mg.
PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
SANTIAGO MAURICIO ALTAMIRANO MELENDEZ

Ing. Santiago Altamirano, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
FRANKLIN WILFRIDO SALAZAR LOGRONO

Ing. Franklin Salazar, Mg.
PROFESOR CALIFICADOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la Institución.

Ambato, agosto 2021.

Frank Janio Montenegro Garófalo

C.C. 150095580-0

AUTOR

DEDICATORIA

Con gran admiración y respeto a mis padres
Carlos Montenegro y Jacqueline Garófalo,
gracias por su apoyo incondicional en
cada momento de este proceso,
quienes me brindaron el consejo
necesario para poder continuar
mi camino con esfuerzo
y dedicación.

A mis hermanos por saber comprender
que toda meta trazada lleva su
tiempo y apoyarme a culminar,
siempre brindándome sus
palabras de aliento.

Frank Janio Montenegro Garófalo

AGRADECIMIENTO

A DIOS por darme el entendimiento y conocimiento necesario en el trascurso de este proceso de Titulación, por darme la fuerza necesaria e iluminarme por el camino para encontrar las adecuadas soluciones a cada problema presentado.

A mis Padres por el apoyo en todo sentido, por estar siempre dispuestos a estrecharme su mano sin excusas y en todo momento.

A mis hermanos por estar presentes todos estos años, siendo fieles testigos del esfuerzo y constancia que he dedicado.

A Marcelo García, mi tutor de Tesis, por ser una gran persona, un excelente docente y fomentar en mi la investigación.

Frank Janio Montenegro Garófalo

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

Actualmente, se han implementado sistemas robóticos para la automatización en los procesos de fabricación, creando así mejora en la calidad de los productos, no obstante, acorde evolucionan estas tecnologías ha sido necesario implementar sistemas de monitorización en tiempo real para cada dispositivo. De forma que, conociendo el valor de los parámetros eléctricos del robot se previenen fallos del controlador, sobrecargas del sistema, daño de motores, de las cajas reductoras y de la estructura mecánica del dispositivo en general. Además, hoy en día para el desarrollo de la robótica, se considera como principal complejidad la implementación de algoritmos que definan el comportamiento del robot, a tal punto, que construir un robot desde cero, ya no es considerado como aporte significativo a la investigación, de ahí la importancia de la utilización de un Sistema Operativo para robots.

Es así que la presente investigación se desarrolla con ROS (Sistema Operativo Robótico), usando un microcontrolador de bajo costo Raspberry Pi, donde se monitoriza la corriente eléctrica de las articulaciones del KUKA youBot, siendo este un manipulador móvil desarrollado para la investigación, con interfaces abiertas y que permiten el acceso al sistema en casi todos sus niveles de control de hardware. Para la comunicación se emplea el protocolo MQTT y para la interfaz gráfica se usa un Portal Web.

Finalmente, se define un plan de pruebas con diferentes pesos acoplados al manipulador, donde los resultados mostraron un 97% de eficacia del Sistema, garantizando el envío de datos en tiempo real y sin pérdida de los mismos.

Palabras Clave: Robótica, manipuladores móviles, ROS, IoT, monitorización, parámetros eléctricos.

ABSTRACT

Currently, robotic systems have been implemented for automation in manufacturing processes, like creating improvement in the quality of products, however, according to the evolution of these technologies, it has been necessary to implement real-time monitoring systems for each robotic device. That way knowing the value of the robot's electrical parameters, it prevents driver failures, system overloads, engine damage, reducing boxes, and the mechanical structure of the device in general. Also, today it is considered as the main complexity in robotics for its development, the implementation of algorithms that define the behaviour of the robot, therefore building a robot from scratch is no longer considered a significant contribution to research, wherefore the importance of using a Robot Operating System.

So, this research is developed with ROS (Robotic Operating System), using a low-cost Raspberry Pi microcontroller, where the electrical current of the KUKA youBot joints is monitored, this being a mobile manipulator developed for research, which has open-source interfaces and that allow access to the system in almost all its levels of hardware control. Communication is developed under the MQTT protocol and for the graphical monitoring interface, Web Portal is used.

Finally, a test plan is defined with different weights coupled to the manipulator, where the effectiveness of the System is 97%, guaranteeing the sending of data in real-time and without data loss.

Keywords: Robotics, mobile manipulators, ROS, IoT, monitoring, electrical parameters.

INTRODUCCIÓN

En la industria manufacturera, gran parte del trabajo manual ha sido, y sigue siendo, reemplazado por robots industriales debido a su repetitividad, precisión, velocidad y eficiencia. A medida que la tecnología ha evolucionado, la industria se ha vuelto más automatizada para proporcionar líneas de producción eficientes, adaptables y en continua evolución [1].

La fabricación de automóviles fue la primera industria en beneficiarse del uso de robots incorporándolos en plantas de fabricación. Normalmente, estos robots realizaron tareas repetitivas de recoger y colocar de manera exitosa, al contar con la naturaleza repetitiva de sus movimientos, mismos que facilitaron y agilizaron las operaciones, que caso contrario de ser llevadas a cabo manualmente se habría tenido una gran cantidad de problemas y los procesos serían poco sostenibles [2].

Los robots se consideran máquinas sofisticadas pero su tarea suele ser tan simple, como la de realizar una secuencia mínima de movimientos, a su vez estos forman parte de una planta altamente automatizada, y deben funcionar de manera confiable para mantener la disponibilidad de esa planta. En la industria se puede tener más de 200 robots en una sola planta. Los robots realizan muchas tareas, como manipulación de materiales, soldadura, entre otras y el funcionamiento de la fábrica depende de ellos [3].

Aunque se diseñan planes de contingencia en líneas de producción automatizadas, se producen averías y el tiempo de inactividad es muy caro, por lo que se han estudiado tanto métodos de mantenimiento destinados a prevenir fallas, así como destacar la importancia de incluir técnicas basadas en condiciones de monitoreo del rendimiento del dispositivo robótico en tiempo real [4].

Actualmente, la robótica en la industria se ha ido incorporando en mayor cantidad, y de entre los dispositivos más usados se presentan los robots manipuladores, mismos que deben realizar tareas de alta precisión, tener capacidad de interacción con el medio, contar con una eficiente comunicación entre ello,s al tener gran cantidad de procesos ejecutándose al mismo tiempo, para todo esto es necesario conocer de entre muchas de las variables existentes el valor sus parámetros eléctricos, ya que en base a esto y a

lo especificado en los datos técnicos del robot se genera el correcto funcionamiento del mismo, una buena manera de obtener estos valores en tiempo real es implementando sistemas de monitorización [5].

Para la monitorización de datos, actualmente los portales web han tomado gran relevancia en la industria, ya que estos pueden transformar los controladores de cualquier sistema en puertas de enlace de IoT y conectarlos con la nube, sin complicadas herramientas de ingeniería ni configuraciones de protocolo se puede comenzar a configurar directamente en línea [6]. MQTT es una herramienta que se utiliza en una amplia variedad de industrias al ser un protocolo de mensajería estándar para IoT, este ha sido diseñado como transporte de mensajería de publicación/suscripción sumamente liviana lo cual es ideal para conectar dispositivos remotos con un ancho de banda de red mínimo [7].

Además, hoy se considera como un eje principal la utilización de SO (Sistemas Operativos) para la programación y manipulación de robots, el SO debe incorporar cierta abstracción de hardware, además de varias de las tareas básicas que faciliten el desarrollo en la plataforma robótica. ROS es un sistema operativo robótico, el cual sirve para el desarrollo de software dirigido a estas aplicaciones [8]. Este Sistema ofrece servicios de control de dispositivos de bajo nivel, transmisión de mensajes en medio de procesos, abstracción de hardware, incorporación de funcionalidades que comúnmente son utilizadas y la administración de datos en paquetes [9]. ROS es una potente herramienta ya que integra la simulación, visualización y gestión de los procesos en una misma distribución del software.

Para dar solución a lo descrito, la presente investigación desarrolla un Sistema de monitorización en tiempo real del valor de la corriente eléctrica de las juntas de un robot manipulador, se emplea el uso de un portal en la nube para la visualización de datos bajo el protocolo MQTT, con ROS se crea la comunicación entre el robot y un mini controlador de bajo costo. El eje principal del proyecto es mantener una comunicación estable y en tiempo real sin pérdida de datos, brindando al usuario la facilidad de monitorizar las variables desde cualquier lugar que él desee.

CAPITULO I.- MARCO TEÓRICO

1.1 Tema de Investigación

"MONITORIZACIÓN DE PARÁMETROS ELÉCTRICOS DE UN MANIPULADOR MÓVIL"

1.2 Antecedentes Investigativos

1.2.1 Contextualización del problema

El incremento y la demanda del uso de sistemas robóticos que año tras año se han ido introduciendo en la industria, específicamente los robots manipuladores que actualmente se consideran como un elemento más en el proceso productivo de una fábrica donde se requiere movimientos repetitivos, fuerzas elevadas y orientar piezas siguiendo trayectorias preestablecidas en la ejecución de tareas variadas, hace que cada vez sea más indispensable tener un control preciso de estos robots para su correcto desempeño.

Dentro de una fábrica es importante conocer el valor de los parámetros mecánicos, electrónicos y eléctricos de cada máquina-herramienta que interviene en el proceso, estos parámetros son de utilidad para sacar el máximo potencial de cada dispositivo robótico que forma parte del proceso con el objetivo de minimizar los riesgos que conlleva el ocasionarse fallos, tanto al mismo instrumento que ejecuta la actividad o al proceso en sí, de tal modo que no solo se debe conocer los parámetros eléctricos de un manipulador móvil, sino también es necesario llevar una lectura en tiempo real que permita controlar y monitorizar las variables eléctricas de estos dispositivos.

Desconocer estas variables en un manipulador móvil conlleva a no aprovechar varias de las ventajas que implica el sí monitorizarlas, de entre ellas tenemos; reducir la duración de los movimientos improductivos, conocer el valor de torque que ejerce cada articulación, evitar fallos al sistema mecánico o al controlador del robot, entre otras. Todo esto se podría evitar si conocemos el valor de los parámetros eléctricos en tiempo real, siendo de entre estos parámetros uno de los más importantes el valor de la corriente eléctrica que fluye por cada articulación del manipulador.

1.2.2 Estudio del arte

A partir de la inclusión de sistemas automatizados en la industria se reduce el trabajo humano y surge la necesidad de monitorizar y controlar estos sistemas, tanto en los procesos de fabricación como al hacer uso de dispositivos robóticos en las diferentes tareas ejecutadas, así en la actualidad se ha creado el requisito de implementar estos sistemas de monitorización, de los cuales hoy por hoy existen varios métodos para este fin, el objetivo principal de estos métodos es proporcionar sistemas confiables y eficientes; por esta razón, los sistemas convencionales son sustituidos por sistemas modernos e inteligentes [5].

Muchos estudios que investigan los sistemas de control modernos generalmente se centran en monitorizar y controlar los parámetros del motor eléctrico, ya que en un proceso de fabricación, se pueden usar múltiples motores eléctricos al mismo tiempo, un ejemplo de estos sistemas es la aplicación de manipuladores móviles y brazos robóticos industriales, donde monitorizar cada motor utilizado del robot es muy importante para sacar el máximo beneficio del mismo y evitar fallos del proceso en general [10].

A continuación, se detallan investigaciones que demuestran la importancia de conocer el valor de los parámetros eléctricos de un robot, aquí se describen trabajos en referencia a manipuladores móviles que prueban los beneficios de aprovechar la lectura de estas variables. Es así como demuestra la investigación según Dong-Eon et al. en [11], donde se propone un control óptimo anti deslizamiento debido a los factores externos del entorno por donde circula el robot, se utiliza un método de detección de corriente de los motores de un manipulador móvil, que consiste en un controlador PID Fuzzy, el sistema usa un sensor de corriente que emite datos necesarios para el control antideslizante, mismo que trabaja en conjunto con el PI del sistema de control.

Así también la investigación de Zhenwei et al. como describe en [12], para la monitorización de parámetros de una articulación robótica de un manipulador móvil, donde para identificar estos parámetros se basa en información multisensorial, tales como el sensor de par articulado y sensores de corriente del motor para así lograr un modelado de par y fricción constantes.

Otro documento es el realizado por Abele et al. en [13], donde esta publicación presenta un concepto para medir el comportamiento de vibración de los robots industriales utilizando sensores de bajo costo, éste fue desarrollado como parte de un sistema para monitorizar el comportamiento dinámico de los robots industriales en todo su espacio de trabajo, aquí se usa un sistema de monitorización de corriente en las juntas del robot, además se registra las frecuencias y amplitudes de excitación con suficiente precisión, para así poder sacar conclusiones sobre el comportamiento de vibración del robot industrial.

En estos trabajos se ha aprovechado la lectura de corriente de los motores a favor de su objetivo deseado, en los mismos se han implementado sensores e integrando una comunicación en particular para la lectura de esta variable eléctrica en cada manipulador móvil descrito. Sim embargo, uno de los problemas que rigen al desarrollo de aplicaciones para robots es la complejidad de crear su arquitectura desde cero y para cada sistema en particular, es así como surgen los Sistemas Operativos para robots, de entre los más relevantes que hoy existen es ROS [14].

ROS es un Sistema que puede integrarse a una gran cantidad de manipuladores móviles existentes, donde esta compatibilidad presenta una gran ventaja en relación a las investigaciones descritas previamente, al ser ROS de código abierto, en constante mantenimiento y con soporte de una gran comunidad que actualmente va en constante crecimiento, ya que cada vez se van integrando más dispositivos a este Sistema Operativo Robótico y denominado por muchos como el Android de la robótica [9].

En base a lo descrito se realiza el presente proyecto, donde se propone leer los valores de corriente eléctrica de cada articulación del brazo robótico del KUKA youBot, esto es factible, gracias al acceso que se tiene a este manipulador, al residir en las instalaciones de la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato. Para el control se usa la tarjeta Raspberry Pi, además, se utiliza el sistema ROS para establecer el modelo de comunicación y para la lectura de los valores emitidos por los sensores, donde finalmente los datos son enviados a la nube para su respectiva monitorización.

1.2.3 Fundamentación teórica

KUKA youBot

El KUKA youBot es un manipulador móvil omnidireccional que se ha desarrollado y diseñado para que pueda servir como plataforma de referencia para la industria, la investigación y la educación al mismo tiempo. La manipulación móvil es la perfecta integración y sincronización de mano con la movilidad al ser una tecnología clave para la robótica de servicios profesionales, así como para futuros escenarios de fabricación flexible y cognitiva.

La plataforma móvil omnidireccional KUKA youBot consta del chasis, de cuatro ruedas mecanum, motores, alimentación y una placa de PC donde los usuarios pueden ejecutar varios programas en este o controlarlo desde una computadora de manera remota, además la plataforma viene con un dispositivo Live-USB con Linux-Ubuntu preinstalado y con controladores para el hardware [15].

El brazo de KUKA youBot consta de cinco juntas rotativas y una pinza de dos dedos como efector final. Alcanza una altura de 655 mm, tiene un peso de 6,3 kg y está diseñado para transportar una carga útil de hasta 0,5 kg de peso. Las articulaciones rotativas del brazo giran alrededor de dos ejes diferentes. El máximo de la velocidad de rotación de una articulación es de 90 grados/s y el efector final se puede abrir 11,5 mm por dedo.

Figura N° 1: Manipulador KUKA youBot.

El robot KUKA youBot usa el SO Ubuntu, cuenta con una API (Interfaz de Programación de Aplicaciones) desarrollada en C++, con esta API el desarrollador puede acceder y controlar el hardware del manipulador KUKA youBot. Este software es de código abierto y puede ser visto como un controlador de alto nivel del robot, consta de en un conjunto de subsistemas funcionales y desacoplados, donde el brazo del manipulador se representa como una cadena cinemática 5 grados de libertad y su plataforma móvil omnidireccional es tratada como un conjunto de juntas giratorias [16].

Además, la API utiliza como clases principales: La clase YouBotManipulator que representa el brazo como una serie de articulaciones y una pinza, la clase YouBotBase que representa la plataforma omnidireccional y la clase YouBotJoint que representa una articulación ya sea del brazo o de la plataforma omnidireccional. En ROS el acceso a las propiedades del robot está debidamente organizado de manera jerárquica para cada sistema o subsistema, al contar con un conjunto de clases de configuración, las cuales permiten obtener o establecer configuraciones para esa entidad en particular [17].

Modelo cinemático del KUKA youBot

El Modelo cinemático de la *plataforma omnidireccional* cuenta con una plataforma omnidireccional de cuatro ruedas (Mecanum), estas permiten el desplazamiento libre del sistema en el plano cartesiano, no obstante, se debe considerar en este tipo de ruedas la disposición de los rodillos para obtener un modelo cinemático adecuado. A continuación, se detallan las medidas de la plataforma en milímetros [16].

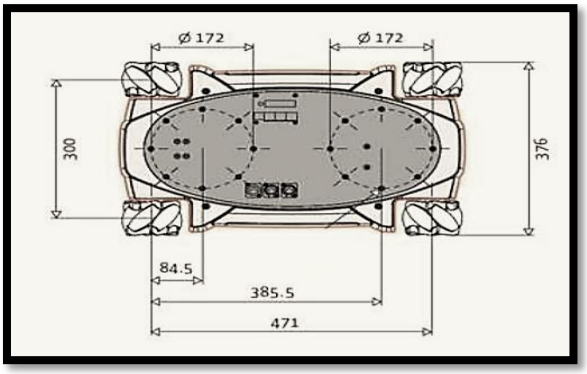


Figura N° 2: Medidas de la plataforma

La configuración cinemática de un robot móvil omnidireccional se expresa por el siguiente conjunto de ecuaciones:

$$\dot{x} = v_l = \frac{1}{4}(d_1 + d_2 + d_3 + d_4)$$

$$\dot{y} = v_t = \frac{1}{4}(-d_1 + d_2 + d_3 - d_4)\tan(\alpha_b)$$

$$\dot{\theta_b} = v_a = \omega = \frac{1}{4}(d_1 + d_2 + d_3 + d_4)\beta$$

En el cual $\dot{x}$ y $\dot{y}$ son la velocidad en los ejes X y Y , o a su vez las velocidades longitudinal (v_l) y transversal (v_t), $\dot{\theta_b}$ es la velocidad angular (v_a) de la plataforma móvil. la velocidad lineal de cada rueda está dada por d_i. Para los parámetros α_b y β se los obtiene experimentalmente ya que estos dependen del ángulo al que están ubicados los rodillos en de las ruedas Mecanum [16].

El *brazo manipulador* del Kuka youBot consta de 5 grados de libertad rotacionales, el efector final está compuesto por una pinza de dos elementos, mismos que permiten la manipulación y sujeción de pequeños objetos. Mediante el algoritmo de Denavit-Hartenberg se obtiene el modelo cinemático del robot manipulador, los marcos de referencia se aprecian en la figura N° 3.

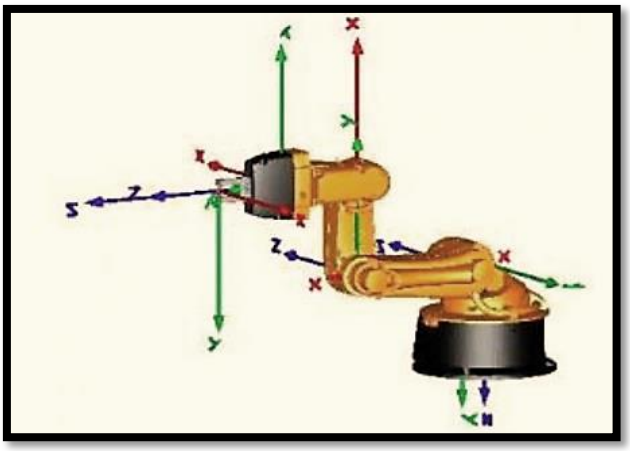


Figura N° 3: Posicionamiento de marcos de referencia
Fuente: Mirelez-Delgado et al., 2015

En la Figura N° 4 se observa las dimensiones longitudinales y rotacionales del brazo robótico y los Parámetros de Denavit-Hartenberg como lo muestra Tabla 1 mientras que las distancias y ángulos entre los marcos de referencia para el brazo manipulador se encuentra en la Tabla 2.

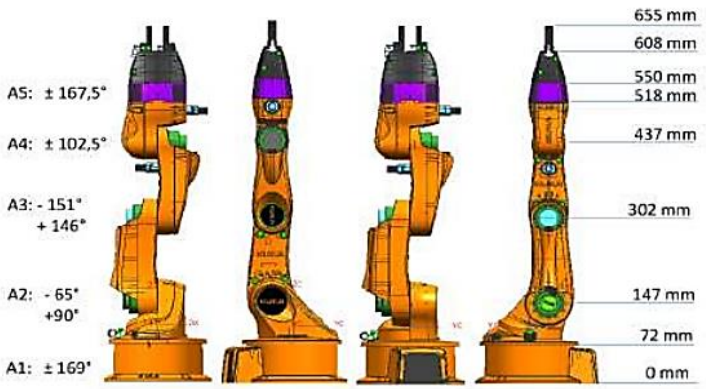


Figura N° 4: Descripción del manipulador del Kuka youBot.
Fuente: Mirelez-Delgado et al., 2015

Se presenta la siguiente tabla donde se detalla los parámetros de Denavit-Hartenberg de cada escabón del manipulador.

Tabla 1: Parámetros de Denavit-Hartenberg para el manipulador

Eslabón	θ	d	a	α
1	q1	0.147	0.0330	$\frac{\pi}{2}$
2	q2	0	0.1550	0
3	q3	0	0.1350	0
4	$q4 + \frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
5	q5	0.2175	0	0

Fuente: Mirelez-Delgado et al., 2015

Se detalla a continuación la cadena cinemática de cada articulación en relación a los marcos de referencia detallados con anterioridad.

Tabla 2: Cadena cinemática para el manipulador del youBot

	Marco anterior	Traslación (cm)			Rotación (grados)		
		x	y	z	x	y	Z
Articulación 1	Base	2.4	0	11.5	180	0	0
Articulación 2	Articulación 1	3.3	0	0	90	0	-90
Articulación 3	Articulación 2	15.5	0	0	0	0	-90
Articulación 4	Articulación 3	0	13.5	0	0	0	0
Articulación 5	Articulación 4	0	11.36	0	-90	0	0
Pinza	Articulación 5	0	0	5.716	90	0	180

Fuente: Mirelez-Delgado et al., 2015

Para obtener las matrices de transformación homogéneas se toman los parámetros de las Tablas 1 y 2 de acuerdo a la siguiente ecuación:

$$\left(\mathbf{A}_i^{i-1}\right) = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & -S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_n^0 = \mathbf{A}_1^0 \dots \mathbf{A}_n^{n-1}$$

S y C denotan las funciones seno y coseno respectivamente, para la matriz de transformación total, se obtiene de multiplicar de manera sucesiva a cada una de las matrices homogéneas.

ROS

Es un software que actúa como puente entre un sistema operativo o base de datos y una variedad de aplicaciones, esencialmente en una red, ésta orientado a la construcción para el control de robots. Se instala sobre un GNU/Linux y más sistemas Operativos compatibles como Ubuntu, Windows, Raspbian, etc. Donde ROS nos permite incorporar cualquier tipo de dispositivos a una red modular y distribuida que en conjunto tiene el control del sistema.

La relevancia de ROS radica en que cada vez más investigadores, empresas y amantes de la robótica, usan software para programar sus robots, ROS busca crear una comunidad que comparta al mundo sus conocimientos adquiridos [14].

En ROS se denomina un nodo maestro, el cual permite a los demás nodos existentes establecer una comunicación, esta puede ser por mensajes o también llamados tópicos y la comunicación por servicio y cliente, donde la comunicación por servicio es uno a uno y la comunicación por Topics es uno a varios [18]. Es el nodo maestro ROSCORE que ejecuta ROS el cual permite que el mecanismo de comunicación entre nodos funcione, este organiza la información, da turnos a cada nodo. Cuando un nodo se añade prácticamente este nodo advertirá al nodo master de su presencia dando a conocer sus mensajes de entrada y de salida [14].

En los sistemas ROS existen una cantidad de nodos donde estos por sí solos no suelen ser muy útiles, hasta el momento que se comunican entre sí, pueden intercambiar información y datos, la forma más común de hacerlo es a través de temas, el cuál es un nombre para una secuencia de mensajes con un tipo definido [18].

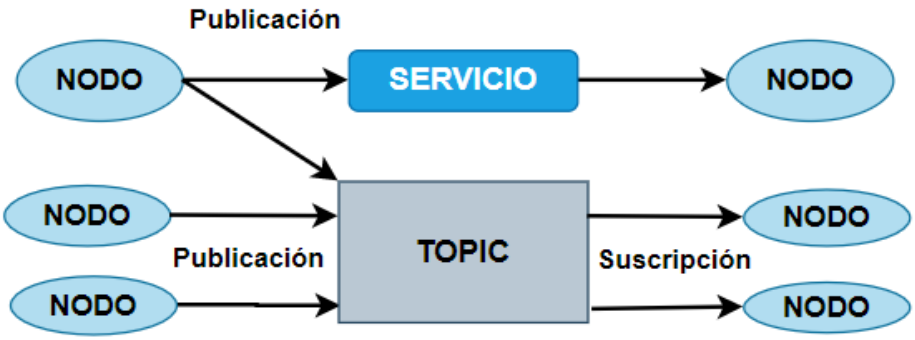


Figura N° 5: Medios para la comunicación de nodos en ROS.

Una de las principales ventajas de usar ROS es que todo se configura para las conexiones necesarias cuando los nodos que se anuncian o se suscriben a temas sean manejados, creando un mecanismo de comunicación autónomo y así no tenga que preocuparse de esto. En ROS, todos los mensajes sobre el mismo tema deben ser del mismo tipo de datos. Aunque ROS no lo aplica, los nombres de los temas a menudo describen los mensajes que se envían a través de ellos [18].

La comunicación en ROS puede darse por **Servicios,** donde se envían y reciben los mensajes a través del medio de comunicación, es cuando solo reciben la información los 2 extremos de la comunicación. La comunicación **Publicación/suscriptor** se presenta de uno a varios, donde el publicador habla y los suscriptores escuchan, se basa en tópicos o cadenas de textos que son publicadas, este tópico estará asociado a un publicador o a un suscriptor de tal forma que pueda enviar o recibir información respectivamente.

Publicación/suscriptor es una de las formas más comunes de intercambiar datos en un sistema distribuido. Se debe considerar que antes de que los nodos comiencen a transmitir datos sobre temas, estos primero se deben anunciar para el Sistema, tanto el nombre del tema como los tipos de mensajes que se enviarán, de este modo ya pueden comenzar a enviar o publicar, así los nodos que desean recibir mensajes sobre un tema pueden suscribirse a ese tema haciendo una solicitud a ROSCORE o nodo maestro, ya una vez suscrito, todos los mensajes en los temas se entregan al nodo que realizó la solicitud con anterioridad [14].

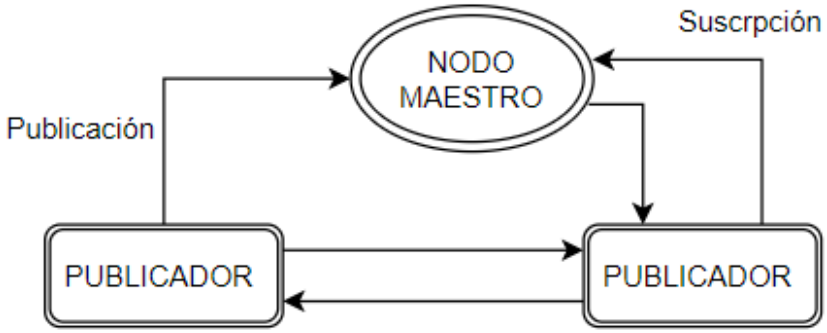


Figura N° 6: Arquitectura ROS, publicación/suscriptor

Los tipos de mensajes en ROS es la información que viaja al establecerse la comunicación entre los nodos y estos pueden ser de varios tipos ya sea de cadena de caracteres, números, imágenes, audios etc. ROS en sí tiene ya definidos todos estos tipos de mensajes los cuales son muy utilizables, y de uso común para cualquier tipo de proyecto a desarrollarse, estos usan datos primitivos como enteros, flotantes, booleanos, arrays, estructuras entre otras.

El Sistema de Archivos de ROS está divido y ordenado en varias carpetas y archivos, donde la carpeta Manifest contiene información sobre los paquetes, como licencia, dependencias y configuraciones del compilador. Los Meta Paquetes es la unión de varios paquetes. La información que se envía se denomina Mensajes. Los Servicios establecen como se estructura la petición y respuesta en ROS. Los algoritmos que describen el funcionamiento del proceso está alojado en el archivo de Códigos.

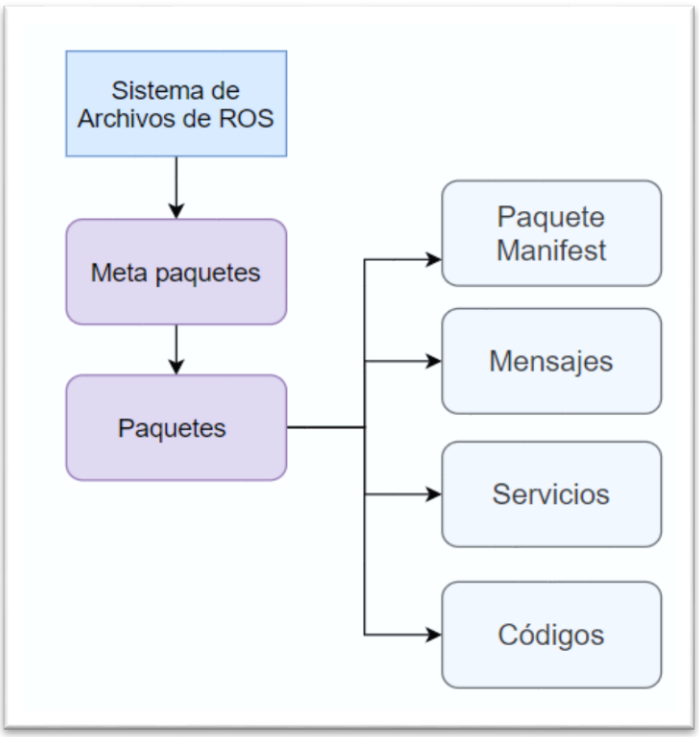


Figura N° 7: Sistema de archivos en ROS.

Los Paquetes poseen la estructura básica y necesaria para crear programas en ROS, cada paquete contiene las siguientes carpetas:

- **src. -** ficheros fuente (java, C++, Python, etc.).
- **lib. -** librerías generadas al correr la compilación con rosmake.
- **config. -** parámetros para la configuración del paquete.
- **data. -** información de usuario necesaria.
- **msg. -** definición de los mensajes usados.
- **srv. -** servicios usados por el paquete.
- **launch. -** scripts que van a permitir lanzar el paquete y todos sus nodos al ingresar una sola instrucción.

Catkin Workspace

Un espacio de trabajo catkin es una carpeta donde se modifica, se crean e instalan los paquetes denominados catkin. Un espacio de trabajo del Workspace puede contener hasta cuatro espacios diferentes, cada uno de los cuales cumple una función diferente en el proceso de desarrollo del software, estos trabajan en conjunto y deben establecerse de manera correcta para que el sistema tenga fluidez y una correcta compilación del programa antes de ejecutarse. Se detalla en la siguiente figura la estructura general de las carpetas o archivos que implica el utilizar el Workspace.

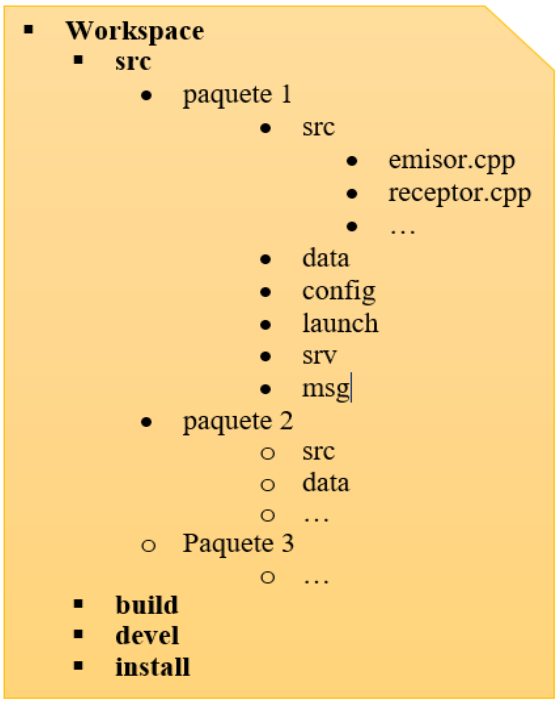


Figura N° 8: Estructura de carpetas Catkin-workspace

- **Source Space (src). -** el espacio que contiene el código fuente de los paquetes catkin. Aquí es donde se puede extraer, pegar o clonar el código fuente de los paquetes que se desea compilar. Este espacio no debe modificarse al configurar, construir o instalar.

- **Build Space (build). -** es el espacio de construcción donde se invoca CMake para construir los paquetes catkin en el espacio fuente. Aquí CMake y catkin mantienen su información de caché y otros archivos intermedios. El espacio de compilación no tiene que estar contenido dentro del espacio de trabajo ni debe estar fuera del espacio de origen.

- **Development Space (devel). -** el espacio de desarrollo donde se colocan los objetivos construidos antes de ser instalados. La forma en que se organizan los objetivos en el espacio de desarrollo es la misma de su diseño cuando se instalan. Esto proporciona un entorno útil de prueba y desarrollo que no requiere invocar el paso de instalación.

- **Install Space (install). -** una vez que se construyen los objetivos, se pueden instalar en el espacio de instalación invocando al objetivo de instalación, generalmente con make install. El espacio de instalación no tiene que estar contenido dentro del espacio de trabajo. Dado que el espacio de instalación lo establece CMAKE_INSTALL_PREFIX [19].

Ubuntu

Es un software libre que brinda la libertad de usarlo a favor en cada necesidad presentada, esta libertad tiene enormes beneficios ya que al compartir la experiencia se crean nuevos conocimientos que impulsa al desarrollo de manera colectiva para mejorar continuamente. Por otro lado, tiene acceso a software esencial para una variedad de aplicaciones, donde el código abierto hace que la comunidad sea libre de ejecutar sus programa para cualquier propósito [20]. Para el manipulador móvil KUKA youBot se tiene el Live-USB que es un dispositivo de arranque el cual contiene una distribución de Ubuntu, controladores básicos y el software para el control y ejecución del robot [17].

Simuladores de entorno virtual para robots

Los simuladores actualmente son una herramienta esencial al momento de llevar a cabo una investigación previa al desarrollo en el entorno real, su diseño permite poner a prueba rápidamente varios algoritmos de programación e integrarlos con el entorno virtual, diseñar un robot, ejecutar pruebas, así como implementar y entrenar sistemas de inteligencia artificial en escenarios realistas propios del simulador. La simulación en general se refiere a la práctica de desarrollo y programación de objetos y de modo que juntos son capaces de emular tareas específicas, ideas o un proceso de propuesta en el mundo real. Es importante seleccionar un simulador acorde al fin del proyecto de investigación desarrollado para no tener inconvenientes con el sistema físico [21].

GAZEBO es un simulador capaz de llevar a cabo interacciones entre robots en interiores y entornos al aire libre, generando una respuesta realista de los sensores que incluye por defecto la simulación, varios de estos son utilizados con frecuencia al incorporar inteligencia artificial a un robot. Gazebo está diseñado para reproducir la dinámica de los entornos que un robot puede encontrar en tareas comunes propias de su naturaleza. Este funciona con dos procesos, cliente y servidor, el cual puede simular desde una máquina remota. Además, es completamente de código abierto y de libre acceso, contiene una amplia base de contribuyentes que apoyan esta herramienta. Además, los motores de física predeterminados integrados con Gazebo incluyen ODE, Bullet, Simbody y DART [22].

Gazebo con ROS ha incorporado y desarrollado varias herramientas para su integración, donde el simulador incluso puede agregar algunas restricciones físicas al entorno, por lo que cuando se ejecuta la simulación y el robot real, el resultado es prácticamente el mismo, de tal modo que, Gazebo se ha convertido en una de las principales herramientas utilizadas en la comunidad ROS.

V-REP se introdujo como un versátil y escalable marco de simulación, al ofrecer una multitud de lenguajes de programación diferentes, mismos que permiten incrustar controladores y funcionalidad en modelos de simulación, esto facilita la tarea a los desarrolladores y reduce la complejidad de implementación para los usuarios. Ahora se ha convertido en un sistema robusto y ampliamente utilizado como simulador de robots, está disponible tanto académicamente como en el campo industrial. V-REP es de código cerrado con una licencia educativa. Además, V-REP tiene diferentes opciones para motores de física, incluida Bullet Physics, ODE, Newton y Vortex Dynamics [23].

WEBOTS es un simulador de código abierto que proporciona un entorno de desarrollo completo para el modelado, la programación y simulación de robots [24]. Gracias a su interfaz amigable y fácil de usar, puede añadir o eliminar objetos o robots y evaluar su posible beneficio del escenario de simulación, que requiere una pequeña cantidad de tiempo para el desarrollo. Además, incluye un compilador, que hace posible probar y validar algoritmos de control que involucren datos complejos procesando rápidamente. Como Gazebo y V-REP, viene con el motor físico ODE integrado [25].

Mini ordenadores de control de Placa única

El mini ordenador ODROID-C4 de placa reducida en relación a sus antecesores es más eficiente desde el punto de vista energético y de rendimiento, como el primer ordenador ARM de 64 bits del mundo a un precio muy asequible. El ODROID-C4 presenta una CPU Amlogic S905X3 que es un clúster Cortex-A55 de cuatro núcleos con una GPU Mali-G31 de nueva generación. Los núcleos A55 funcionan a 2.0Ghz sin estrangulamiento térmico utilizando un disipador de calor de serie, dando lugar a un ordenador recudido y muy silencioso.

Posee LAN de 10/100/1000 Gigabit Ethernet con chipset Realtek RTL8211F, No tiene Wi-Fi incluido, 4 USB de 3.0, 1 USB de 2.0 OTG para usarlo como dispositivo externo, Ubuntu 20.04, CoreELEC, Android 9 (64bit) como sistemas operativos, GPIO 40 pines I/O hasta 25 GPIO, 6 PWM, 2 ADC (12-bit, 1.8V max), 2 I2C, 1 SPI, 1 UART, y salidas de alimentación a 5V, 3.3V, 1.8V, GND. Su temperatura máxima de funcionamiento alcanza los 75°C [26].

Figura N° 9: Placa ODROID C4

BANANA PI es una computadora de placa única de código abierto. Puede ejecutar Android 4.2, Ubuntu, Debian, Fedora y muchas otras distribuciones de Linux. Utiliza una CPU Allwinner A64 de 1,2 GHz de cuatro núcleos, tiene un encabezado GPIO de 40 pines que coincide con el del Modelo B + Raspberry Pi, GPU Mali-400 MP2, 2 GB de RAM y 8 GB de eMMC [27].

Posee Ethernet de 10/100/1000 Mbps, Dongle USB WIFI opcional, un receptor de infrarrojos, dos USB de 2.0, un USB 2.0 OTG todo directo desde el chip, Admite pantalla HD multicanal con HDMI 1.4 (Tipo A - completo), una interfaz de pantalla LVDS.

Se puede ejecutar fácilmente para videojuegos, admite salida de video de alta definición 1080P, el GPIO es compatible con Raspberry Pi y puede ejecutar imágenes raspbian. El rango de temperatura de funcionamiento de Banana Pi va desde un mínimo de -15 °C hasta los 75 °C.

Figura N° 10: Placa Banana Pi

LATTEPANDA es un modelo básico que tiene tres puertos USB, WiFi y Bluetooth 4.0 integrados,un coprocesador compatible con Arduino que le permite ser utilizado como una placa Arduino para controlar la electrónica conectada a sus 20 pines GPIO. En otros lugares, hay soporte para HDMI y Fast Ethernet, y una ranura microSD que admite hasta 32 GB de almacenamiento. Está pre-instalado con una edición completa de Windows 10.

Los informes sobre el uso indican que LattePanda tiende a funcionar bastante caliente, por lo que deberá investigar una solución de enfriamiento para evitar el estrangulamiento de la CPU [28]. La temperatura de funcionamiento de este dispositivo va desde lo 5°C hasta los 50 °C.

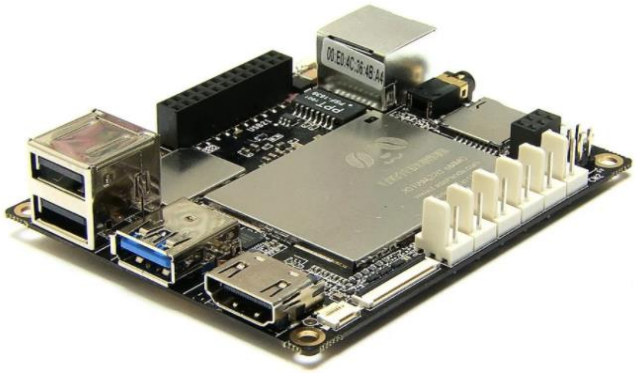


Figura N° 11: Placa LattlePanda

Raspberry PI 3 es básicamente una computadora del tamaño de una tarjeta de crédito. Tiene un ARM/Broadcom SoC (System on Chip) con GPU, RAM y opciones de conectividad inalámbrica y por cable. El micro-ordenador mantiene 1 GB de memoria RAM, pero utiliza una revisión del procesador al BCM2837 ARMv8 de 4 núcleos, funciona a 1.2 Ghz. Esta versión es la primera en utilizar instrucciones de 64 bits.

Además, la mayor novedad de esta versión es que finalmente incluyó Wi-Fi y Bluetooth 4.1 de base, sin necesidad de adaptadores, ampliando los horizontes de su usabilidad mucho más allá. Las especificaciones técnicas muestran que el límite de funcionamiento es de 85 °C de temperatura, Es compatible cons Sistemas Operativos como Raspbian, Linux, Pidora, Windows 10, Ubuntu y RISC OS. [29].

Figura N° 12: Placa Raspberry Pi 3 – Modelo B – ARMv8 – 1G RAM

Node-RED

Es un software que funciona mediante flujos con enfoque a Internet de las Cosas (IoT), nos permite definir de manera gráfica los nodos de servicios a través de varios protocolos estandarizados, esta es una herramienta ligera que se presenta de manera visual, misma que está programada en NodeJS, esta puede correr en dispositivos de bajo procesamiento. Node-RED es una herramienta de código abierto establecido por IBM y permite interactuar e interconectarse con varios conceptos de IoT [30].

Node-RED minimiza la conceptualización sobre eventos de la vida real, al agregar nodos que integran a toda clase de sistemas de comunicación de mensajería, incluyendo los más utilizados en la industria, además de permitir el desarrollo de nuevos nodos al ser de código abierto, de los protocolos funcionales y probados se presenta para IoT el protocolo MQTT entre otros.

Cada nodo procede con los datos dados acorde la forma en que el desarrollador lo ha diseñado, luego esos datos pasan al siguiente nodo en la misma serie. Además, Node-RED proporciona una interfaz de usuario visual, misma que es fácil de entender, modificar y depurar. Una aplicación Node-RED existente se puede compartir de manera sencilla con otros desarrolladores tan solo exportando archivos JSON [30].

Protocolo MQTT

Es un protocolo que permite el transporte de datos de tipo publicador/subscriptor, está creado para tener una gran facilidad de implantación en los entornos que se usa, como por ejemplo en la comunicación máquina-máquina (M2M) e Internet de las cosas, al poseer gran ancho de banda, característica que es de suma importancia en la red.

Este protocolo resalta en el modo de transferir sus paquetes a través de cable al igual que con HTTP, así mismo tiene mucha facilidad de implementarse en el caso de ser cliente. Su facilidad de uso es clave por lo tanto se ha convertido en un modo de comunicación perfecto para dispositivos con recursos limitados en la actualidad. Aunque MQTT comenzó como un protocolo utilizado para comunicarse con sistemas de control de supervisión y adquisición de datos SCADA en la industria del petróleo y el gas, se ha vuelto popular en el campo de los dispositivos inteligentes y hoy es el protocolo de código abierto líder para conectar Internet de las cosas (IoT) e IoT industrial (IIoT) [7].

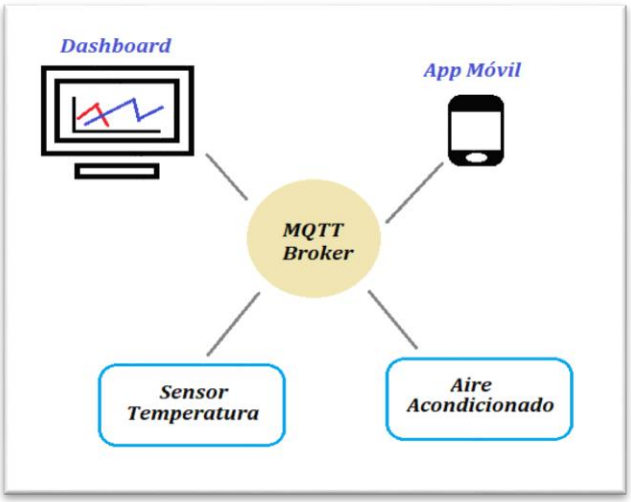


Figura N° 13: Funcionamiento entre MQTT y dispositivos.

La **Calidad de Servicio** en la entrega de los mensajes en MQTT está dada por niveles, de los cuales se presenta el nivel 0 (At most once*)*, mismo que garantiza una entrega como máximo una vez, en vista que el destinatario no reconoce la recepción del mensaje recibido y a su vez el remitente no almacena ni retransmite el mensaje, lo que puede ocasionar pérdidas o duplicación de mensajes.

En el nivel 1 (At least once), se garantiza que el mensaje será enviado y entregado al menos una vez, para este nivel el remitente debe almacenar el mensaje hasta que llegue el mensaje de confirmación correspondiente. En este nivel es muy probable que se presenten duplicación de los mensajes.

Finalmente, se presenta el nivel 2 (Exactly once), donde este nivel brinda del servicio más alto, garantizando que cada mensaje sea recibido una sola vez por el destinatario, es el nivel más seguro, pero también el más lento debido al hacer uso de por lo menos dos flujos de solicitud/respuesta entre el destinatario y el remitente. En caso de que el mensaje no llegue el remitente es responsable de retransmitir el mensaje luego de cierta cantidad de tiempo [31].

Para los **Niveles de Seguridad** en MQTT se tiene el Nivel de red para crear una conexión fiable, es necesario utilizar una red físicamente segura o VPN para la comunicación entre clientes y agentes, esto puede funcionar con aplicaciones de gateway donde este se conecta a la VPN y a los dispositivos.

Luego se presenta el Nivel de transporte, mismo que se usa para la encriptación del transporte de datos cuando sea necesario confidencialidad, aquí los datos no pueden ser leídos en el proceso de transmisión al ser un método seguro y que proporciona una autentificación de certificado de cliente para validar la identidad de ambos lados, de este modo se crea una comunicación muy sólida y con alta seguridad para la transmisión de datos.

Por último, está el Nivel de aplicación, donde se proporciona un identificador del cliente por medio de credenciales, tanto el nombre de usuario y contraseña, estas son propiedades dadas por el propio protocolo. En el nivel de aplicación también es posible usar encriptación de la carga útil para de esta manera asegurar la información transmitida sin ser necesario una encriptación de transporte [32].

Broker Mosquitto

Mosquitto es uno de los brokers más populares y estables, y proporciona una implementación de servidor ligera para el Protocolo MQTT-SN. Ligero significa que solo se incluyen las funciones necesarias, mismas que se codifican de la forma más eficaz posible [13]. Mosquitto es un agente de mensajería instantánea, es liviano y puede ser usado en cualquier dispositivo inteligente. Este proporciona un método sencillo para la interacción en una red de topología tipo estrella empleando un modelo de publicación / suscripción.

Mosquitto también puede traducir y transferir mensajes fácilmente lo que lo hace ideal para manejar información para IoT, al actuar como puerta de enlace cuando los dispositivos se comunican con protocolos. Además, Mosquitto se puede instalar fácilmente en Raspberry Pi [33].

La arquitectura publicación/suscripción de MQTT está basada en eventos, donde cada mensaje se envía a los diferentes receptores los cuales se hayan suscrito a la publicación en particular, el Broker Mosquitto es el encargado de distribuir estos mensajes. El tema al que se suscriben los receptores para recibir los mensajes se denomina Topic. Cuando un cliente que se suscribe a un topic, es decir, a quién va dirigido el mensaje, esto se lo puede comparar al asunto de un correo electrónico o email, donde el mensaje no tiene el destinatario en concreto, ya que puede haber uno, varios o a su vez ninguno.

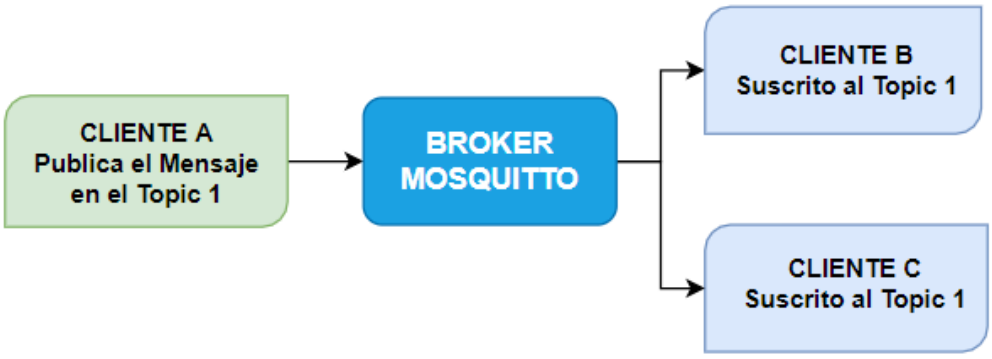


Figura N° 14: Arquitectura Publicación/Suscripción MQTT

Servicio en la nube Anyviz

Es un portal web fácil de usar con entorno de programación gráfica, se puede encontrar para el servidor web como bloque de programa directamente descargable para lógicas basadas en CODESYS, Windows CE, dispositivos Linux basados en ARM y adaptadores directamente a OPC UA y para el protocolo MQTT. La interfaz de usuario es un entorno de programación gráfica basado en HMTL5 basado en el principio de arrastrar y soltar. Por lo tanto, el servicio funciona en un navegador de Internet en un sitio web.

El entorno de programación gráfica se ha considerado muy minimalista y es muy fácil de usar. La variable deseada denominada etiqueta, se arrastra a la página y luego se configura cómo se desee representar el valor de la variable. Las opciones son directamente como texto, una tabla, una tendencia, un gráfico o alguna otra vista deseada. El entorno operativo gratuito está limitado a diez variables y el almacenamiento de datos está limitado a cierta cantidad de valores numéricos por hora.

Esto es comprensible, ya que el servicio es bueno, financiado y hay actividad. El mayor gasto del servicio Anyviz es también el almacenamiento de todos los datos del cliente, datos que pueden ser muy importante para los clientes y debe realizarse una copia de seguridad en varios servidores diferentes [34].

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un sistema de monitorización de corriente eléctrica de las articulaciones del robot KUKA youBot.

1.3.2 Objetivos Específicos

- Crear un ambiente virtual en Gazebo de la monitorización de corriente eléctrica de las articulaciones del robot KUKA youBot.
- Especificar los parámetros del KUKA youBot API en ROS para recibir los valores de corriente eléctrica de las articulaciones del manipulador.
- Implementar la monitorización de corriente eléctrica en el entorno real.

CAPÍTULO II.- METODOLOGÍA

2.1 Materiales

Ubuntu 16.04 LTS (Xenial Xerus)

Es un sistema operativo de código abierto para computadoras y mini controladores, forma parte de las distribuciones Linux, basa su arquitectura en Debian de tipo Intel, AMD y ARM, este Sistema permite la fácil integración por parte de desarrolladores. La imagen completa de Ubuntu Desktop es grande, pero contiene todo lo que necesita para convertir una Raspberry Pi en una PC principal.

Este Sistema Operativo ha sido elegido al ser la plataforma principal para ROS desde el principio, esa es la razón por la que cada versión de ROS es compatible con exactamente un Ubuntu LTS, misma que posee soporte a largo plazo. Esta versión cuenta además con la mayoría de paquetes ROS probados en robots reales y en simulación, esto facilita la compatibilidad de drivers y la estabilidad de la comunicación del sistema en general.

ROS Kinetic Kame

Es un entorno que facilita el desarrollo de aplicaciones en la robótica. Incluye herramientas y bibliotecas que proporcionan abstracción de hardware, visualizadores, dispositivos controladores, transmisión de mensajes, etc. Con una serie de comandos simples el usuario puede interactuar con el nodo maestro ROS donde puede publicar mensajes o llamar a servicios de manera manual.

ROS Kinetic Kame se seleccionó ya que está dirigido principalmente a la versión Ubuntu 16.04 (Xenial) como lo sugiere la empresa de programación Canonical y a su vez la codependencia de ambos crea la estabilidad del sistema a tiempo que agilizan el desarrollo de aplicaciones, todo esto de manera fácil al contener una gran variedad de codificación ya probada, así como una cantidad de documentación extensa que permite comprender el sistema desde la práctica. Otra de las grandes ventajas es la facilidad de migración a nuevas versiones en caso de ser requerido por el desarrollador, ya que con un mínimo de configuraciones esto se puede llevar a cabo.

Gazebo

Es un simulador que permite probar rápidamente algoritmos, diseñar robots, realizar pruebas de regresión y entrenar el sistema de Inteligencia Artificial utilizando escenarios realistas. Gazebo ofrece la capacidad de simular de forma precisa y eficiente poblaciones de robots en entornos complejos de interior y exterior

Se utilizará este software libre debido a su nulo gasto de inversión en comparación a los otros mencionados y la compatibilidad excelente que brinda con el sistema ROS y Ubuntu, además de tener todas sus características y funcionalidades centradas y dedicadas de manera concreta al sector de la robótica, permitiendo desde la sintonización de controladores hasta la detección de colisiones.

Es multiplataforma, pudiendo ser utilizado por otros investigadores que requieran usarlo en otros sistemas operativos. Dispone de un motor de físicas renovado con el que se puede simular incluso comportamientos mecánicos, en este aspecto, es sencillo encontrar simulaciones con drones, submarinos e incluso estrategias de despegue de aviones. Lo más relevante para el desarrollo del presente proyecto, es que en su biblioteca contiene al manipulador móvil KUKA youBot disponible.

KUKA youBot

Este manipulador móvil es compatible con ROS y se puede comunicar a través de Ethernet y EtherCAT, posee una PC mini ITX integrado. Cuenta con un brazo que posee cinco grados de libertad y una pinza al final que se mueve linealmente. En su base cuenta con cuatro ruedas Mecanum las que pertenecen a su plataforma omnidireccional, estas ruedas son modeladas de manera que usan fricción asimétrica.

Este robot manipulador ha sido seleccionado al contar con el acceso total al mismo. El manipulador KUKA youBot reside en las Instalaciones de la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato. El mismo fue diseñado para la investigación, de cierto modo tiene gran similitud a brazos robóticos utilizados en la industria, tanto en su tipo de movilidad, agarre de objetos, asi como también sus movimientos de traslación.

Comparativa de dispositivos de control

Para la elección del mini ordenador se toman en cuenta varios factores, como el rendimiento del dispositivo, su capacidad de procesamiento, memoria, compatibilidad con Sistemas Operativos, su temperatura de funcionamiento, accesibilidad y costos de adquisición.

Tabla 3: Tabla comparativa de microprocesadores

Nombre de Placa	Procesador	GPU	Memoria	Sistemas Operativos	Temperatura de funcionamiento	Valor
Odroid-C4	Amlogic ARM S905X3 (4x Cortex-53 @ up to 2GHz)	Mali-G31	2GB DDR3 RAM; optional 8GB eMMC	Ubuntu 20.04, CoreELEC, Android 9 (64bit)	Su temperatura máxima de funcionamiento alcanza los 75°C	$70
Banana Pi	Allwinner A64 (4x Cortex-A53 @ 1.2GHz)	Mali-400 MP2	2GB DDR3 RAM; 8GB to 64GB eMMC	Android 4.2, Ubuntu, Debian, Fedora	Mínimo de -15 °C hasta los 75 °C	$70
LattePanda	Intel Cherry Trail Z8350 Quad Core 1.8GHz	Intel HD Graphics @200-500 Mhz	2GB DDR3L	Pre-instalado con una edición completa de Windows 10.	Desde los 5°C hasta los 50 °C.	$140
Raspberry Pi 3 Model B	Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz	Broadcom VideoCore IV	1GB	Raspbian, Linux, Pidora, Windows 10, Ubuntu,RISC OS	Su temperatura máxima de funcionamiento alcanza los 85°C	$60

La Raspberry Pi 3 Model B se eligió ya que presenta varias ventajas en cuanto a su fácil accesibilidad, bajo precio en relación a las demás placas, puede trabajar con varios Sistemas Operativos estables, soporte en varios campos y áreas de la tecnología, cuenta con una gran cantidad de repositorios compatibles con hardware existente, además de poseer las características suficientes para poder trabajar como dispositivo de control en el presente proyecto.

Protocolo MQTT

MQTT es un protocolo muy práctico que hoy en día se está implementado en varias aplicaciones IoT y M2M, donde en este campo, la aplicación se desarrolla para el monitoreo, control y para el envío de datos entre servidores, sensores y actuadores.

Se eligió este protocolo en vista de los beneficios que incluye, como presentar una huella de código liviana donde los dispositivos solo necesitan unas pocas líneas de código para comenzar a funcionar con el protocolo. Los paquetes de datos son minimizados lo que significa eficiencia desde el punto de vista energético, esto es beneficioso si un dispositivo funciona con batería o tiene poca potencia de CPU.

La velocidad es óptima y funciona en tiempo real sin retrasos, presenta facilidad de implementación al contar con bibliotecas en lenguajes de programación comúnmente usados. Actualmente, se ha ido incorporando en la Industria para el desarrollo de sistemas de control y registro de datos, así mismo es empleado en el monitoreo de dispositivos que requieran ser leídos en tiempo real.

Node-RED

Es una herramienta Open Source basada en Node.js misma que facilita la integración de hardware con software de manera sencilla y rápida por medio de una programación visual basada en nodos. El sistema puede recibir datos de los sensores y controlar los actuadores por medio de un servidor, donde para el presente proyecto se utiliza el protocolo MQTT para la comunicación.

Se eligió esta herramienta porque la aplicación a desarrollar necesita trabajar con fluidez, tanto con el robot como su interacción de manera ágil y ligera con el portal web, se emplea Node-RED ya que procesa los datos del sensor de manera gráfica y con tiempos de transmisión de datos muy elevados, esto quiere decir que es de fácil acoplamiento para sistemas que requieran lectura de datos en tiempo real. Además, esta puede incluir la interacción de datos provenientes de varios sistemas, que usen diferentes tipos de protocolos e integrarlos fácilmente en la red de nodos, así se puede crear un sistema de control de datos con una variedad de dispositivos que usen diferentes sistemas de comunicación.

Broker Mosquitto

Mosquitto es un servidor MQTT, el cual permite el envío y recepción de mensajes en este protocolo. Gracias a su ligereza permite fácilmente emplearlo en gran número de ambientes, inclusive si éstos fueran de pocos recursos.

Se acopla de manera sencilla a varios Sistemas Operativos y portales web actualmente utilizados en tecnologías IoT. Se seleccionó este bróker debido a su sencilla instalación en SO, a su fácil uso, configuración y al tener gran cantidad de documentación sobre su funcionamiento, además de ser un servidor de mensajes de código abierto.

Portal web

Se eligió AnyViz al poseer una de las formas más fáciles de monitorear, operar y analizar los controles de máquinas y plantas de forma remota, gracias a la nube desde cualquier ubicación que el usuario desee, el portal puede transformar sus controladores existentes en puertas de enlace de IoT y conectarse con su planta a AnyViz Cloud, todo esto sin complicadas herramientas de ingeniería ni configuraciones de protocolo.

Caso de estudio

En el presente proyecto se pretende dar lectura de la corriente eléctrica proveniente de las articulaciones del brazo robótico de un manipulador móvil, al ser estos datos de suma importancia para sacar el máximo potencial del dispositivo, además con estos valores y en base a los datos técnicos del robot se puede reducir la duración de los movimientos improductivos, realizar un análisis de eficiencia de consumo de energía eléctrica, conocer el valor de torque que ejerce cada articulación, evitar fallos al sistema mecánico o al controlador del robot y de entre muchos otros aspectos relevantes para la prevención, el control y monitoreo del dispositivo robótico.

Para controlar el hardware del robot, se desea emplear un mini controlador de bajo costo que cumpla con los requisitos y características necesarias para el correcto funcionamiento del sistema, se considera la utilización de un Framework dedicado al software robótico para la integración del robot con el dispositivo de control.

A fin de comprender el uso del Framework mencionado, se contempla como primer paso la utilización de un entorno virtual que integre la simulación con el sistema de control, así de esta manera se logre evitar fallos y no tener inconvenientes al momento de trasladar la arquitectura del sistema hacia el robot físico.

Con la implementación de un protocolo ligero y que no demande consumo excesivo de recursos, se desea crear una comunicación estable que traslade los datos desde el robot hasta un portal web, donde los datos obtenidos se visualicen de manera amigable para el usuario. A continuación, se presenta la arquitectura general del sistema.

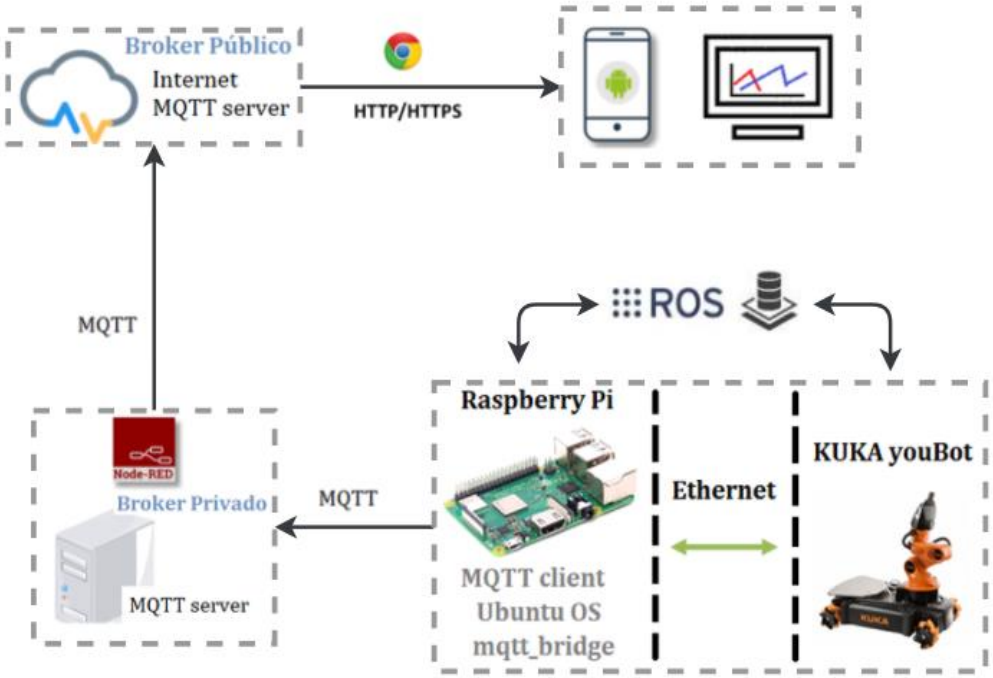


Figura N° 15: Arquitectura del sistema

Fuente: Elaboración propia del Investigador

2.2 Métodos

2.2.1 Modalidad de la investigación

Investigación Bibliográfica – documental

Debido a la fundamentación de conceptos que sirven como sustento para estudios y trabajos similares en referencia al tema basado en libros, artículos, manuales, congresos académicos, publicaciones, revistas y páginas web para ampliar y fortalecer conocimientos como la programación, modelos de comunicación y codificación que puedan emplearse en el desarrollo de la investigación.

También, es útil la documentación de información recolectada sobre los resultados alcanzados a fin de servir como soporte técnico y contribución científica a trabajos que puedan realizarse posteriormente.

Investigación de campo

Ya que se debe acudir a trabajar directamente con el robot una vez terminado y probado en la simulación, esta se realizará en su totalidad dentro de las instalaciones de la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato.

Investigación Experimental

En vista que se llevó a cabo las pruebas, el control y la experimentación del Sistema hasta obtener el óptimo funcionamiento del mismo, se lo realizó en los laboratorios de la Facultad implementando el software de control instalado en la tarjeta para la manipulación del robot.

2.2.2 Recolección de la información

Para la recolección de información se llevará a cabo la recopilación de datos que proporcionen información técnica científica al proyecto, así como trabajos de investigación, libros, tesis, congresos, revistas y temas en relación. Se clasificarán estos datos de acuerdo a la importancia de los mismos desechando los no relevantes, para la interpretación de estos datos se desarrollarán gráficos, tablas estadísticas que permitan analizar los resultados obtenidos acorde a los objetivos planteados.

2.2.3 Procesamiento y análisis de datos

Software Wireshark. – permite la captura de paquetes y mostrar en tiempo real todos los mensajes transmitidos y recibidos en la red al cual la PC está conectada. Posee opciones de filtrado para organizar la información obtenida, al poseer este software la capacidad de ver el tráfico total que pasa por la red.

Indicador de eficacia. – es el grado en que se logran los resultados esperados, es decir la comparación entre lo alcanzado y lo esperado (RA/RE). Los niveles superiores de eficacia corresponden a porcentajes de ejecución muy elevados, donde una calificación alta es cada vez más difícil de obtener.

T-Student. – esta prueba se empleará para comprobar la hipótesis H_0 de la variable dependiente de la hipótesis general. Es una prueba estadística que se fundamenta en dos principios, la distribución de normalidad y la independencia de muestras. Se basa en una prueba muestral para 30 datos. T-student es de común uso, pero una poderosa aplicación, la cual permite comparar dos variables, aun cuando una de ellas no cumpla el criterio de normalidad.

2.2.4 Hipótesis

Hipótesis de Investigación

Con la utilización del protocolo MQTT se pueden leer en tiempo real los datos de corriente eléctrica de las articulaciones del robot KUKA youBot.

Identificación de las variables

Variable independiente. – adquisición de datos de corriente eléctrica de las articulaciones del robot KUKA youBot.

Variable dependiente. – lectura de datos en tiempo real con el uso del protocolo MQTT.

2.2.5 Desarrollo del proyecto

Para la elaboración del proyecto se desarrollarán las siguientes actividades: una minuciosa investigación acerca de programación en C++. Conocer el funcionamiento general de ROS así como sus mecanismos de comunicación. Establecer la integración de ROS con Gazebo para la posterior Simulación del robot. Entender el uso del dispositivo de control con Linux-Ubuntu. Crear los modelos de comunicación para el entorno real. Desarrollar la interfaz de usuario con herramientas IoT. Realizar pruebas y resultados en el entorno real, para finalmente desarrollar la elaboración del Informe Final.

CAPÍTULO III.- RESULTADOS Y DISCUSIÓN

3.1 Análisis y discusión de los resultados

3.1.1 Desarrollo de la propuesta

Entorno virtual para la Simulación

Para el entorno de simulación se inicia creando la integración entre ROS y Gazebo, donde se proporciona al sistema de las interfaces necesarias para la simulación del robot usando el conjunto de paquetes de *gazebo_ros_pkgs*. Luego, se configura el modelo y el entorno de simulación, aquí deben ser incluidos parámetros de sombras, suelo, gravedad, iluminación y ambiente en general, para posteriormente cargar el modelo URDF del robot KUKA youBot desde el lanzador con el que se ejecuta el simulador. Para la comunicación se usa el modo publicación/suscripción por topics. Se desarrolla finalmente el complemento o plugin mediante el cual se genera la manipulación y la lectura de datos de la simulación.

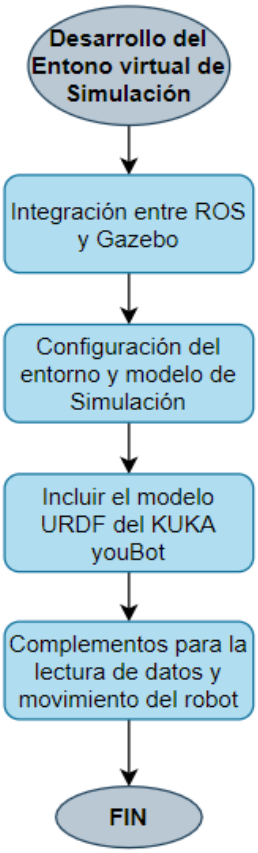


Figura N° 16: Creación del entorno virtual

Fuente: Elaboración propia del Investigador

Con el uso de las herramientas ROS, se establece el espacio de trabajo catkin denominado “workspace_ws”, donde se puede modificar o crear los paquetes catkin. Dentro de este espacio de trabajo se crea la carpeta principal **src**, aquí se desarrolla la programación en lenguaje C++. Como los nodos se distribuyen en paquetes, ahora se crean estos paquetes para el desarrollo de los nodos requeridos por el Sistema. A continuación, se presenta de manera gráfica la estructura general de carpetas.

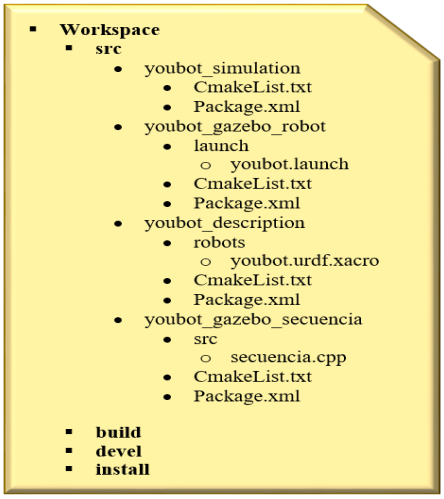


Figura N° 17: Workspace de la Simulación

El paquete Youbot simulation, se define como un meta paquete el cual incluye todas las librerías de Gazebo y Ros necesarias para la correcta compilación y ejecución del programa. Youbot gazebo robot, contiene al lanzador de paquetes mismo que incorpora todos los archivos requeridos para correr simultáneamente, las dependencias y nodos necesarios para la ejecución del sistema**.** Se incluye aquí al complemento el cual publica valores de velocidad, posición y esfuerzo de cada articulación, estos datos son anunciados al tema */joint_states*. La herramienta ROS Rqt Graph permite detallar un diagrama en tiempo real de la comunicación e interacción del sistema, se presenta a continuación de manera gráfica el flujo de datos, la elipse representa un nodo y un rectángulo a un tema.

Figura N° 18: Flujo de datos publicados desde la Simulación

La información física de los componentes del robot, así como los atributos de masa, de inercia, los tipos de articulaciones, colores y más características están situados en el paquete Youbot gazebo description. Finalmente, en el archivo Youbot gazebo secuencia, se aloja el paquete el cual crea al nodo */youbot_gazebo_secuencia* que publica sobre el topic */arm_1/arm_controller/command* generando la secuencia de movimientos del manipulador simulado en Gazebo. Se representa con ROS Rqt Graph el siguiente diagrama el flujo de datos.

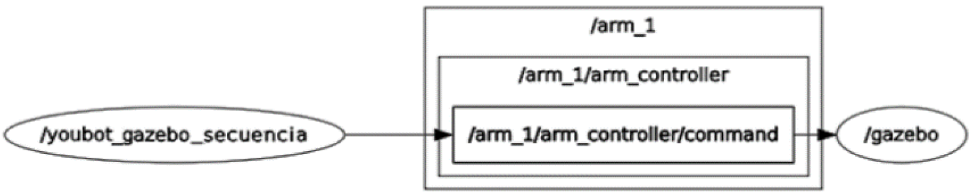


Figura N° 19: Flujo de datos para el control de movimientos sobre Gazebo

Para dar movimiento a la simulación se usa Gazebo plugin, este es una biblioteca desarrollada en C++ que Gazebo carga en tiempo de ejecución, el mismo brinda a los modelos URDF una mayor funcionalidad y crear una comunicación de la simulación con mensajes de ROS, la función de este complemento es controlar el movimiento de cada articulación. Para la secuencia del robot se toma en cuenta los valores rotacionales del manipulador, con esto se evita tener problemas y sobre esfuerzo de los motores al trasladar estos movimientos al entorno real. En base a la tabla de los límites rotacionales presentados en el Anexo N° 1, se experimenta por individual en cada articulación hasta lograr la secuencia deseada, en la siguiente figura se muestra captura de la simulación.

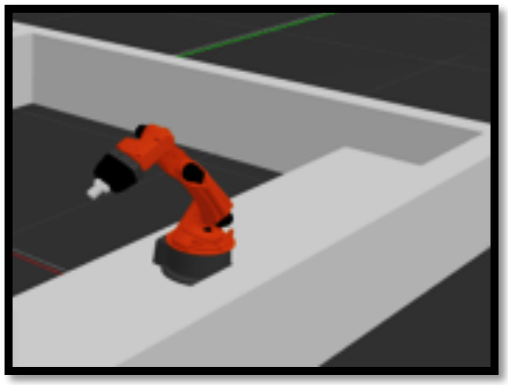


Figura N° 20: Simulación del manipulador KUKA youBot en Gazebo

Para la lectura de corriente eléctrica se parte del análisis de las constantes motoras del motor CC (maxon EC 45 flat) que usa el manipulador de la presente investigación. Donde el par de salida de un motor de CC es directamente proporcional a la corriente real del motor, y la velocidad angular del motor es directamente proporcional al EMF trasero que genera. Estas relaciones simples suelen estar dadas por las ecuaciones:

$$\mathbf{T = I \times K_T} \quad (1)$$

Dónde: T = par (Nm), I = corriente (A), K_T = constante de par (Nm/A)

$$\boldsymbol{\omega = \frac{V_E}{K_E}} \quad (2)$$

Dónde: ω = velocidad angular (rad/s), V_E = voltaje EMF trasero (V), K_E = constante de EMF inversa (Vs/rad)

La constante de par $\mathbf{K_T}$, se encuentra en la hoja de especificaciones del motor tomada del ANEXO N° 2. La pendiente de la curva de par-corriente del motor está determinada por la constante de par. De la ecuación (1) se despeja la corriente:

$$\mathbf{I = \frac{T}{K_T}} \quad (3)$$

De este modo se obtiene una lectura aproximada del valor de la corriente eléctrica de las articulaciones del manipulador de la Simulación. Se toman los valores de esfuerzo de cada articulación proveniente del topic */joint_states*, estos valores son enviados por MQTT a Node-RED donde se ejecuta la ecuación (3) para obtener el valor de la corriente eléctrica. ROS Rqt Graph detalla a continuación el flujo de datos general de la simulación.

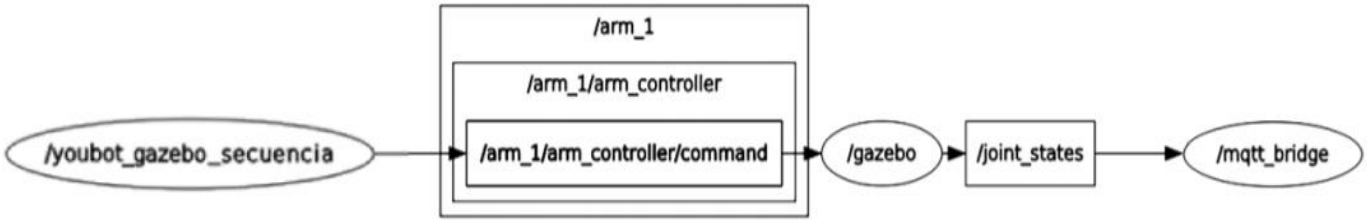


Figura N° 21: Flujo de datos general del Sistema – Simulación.

Entorno real

Etapa 1: Definición del Sistema. - Para una representación comprensible del sistema, se utilizan casos de uso, mismos que describen al conjunto de secuencias de las acciones del sistema, donde se muestra el comportamiento deseado del programa. Este ayuda a validar la arquitectura y a la verificación del sistema. Primero se representa el diagrama de caso de uso por parte del administrador para posterior detallar el caso de uso por parte del Usuario.

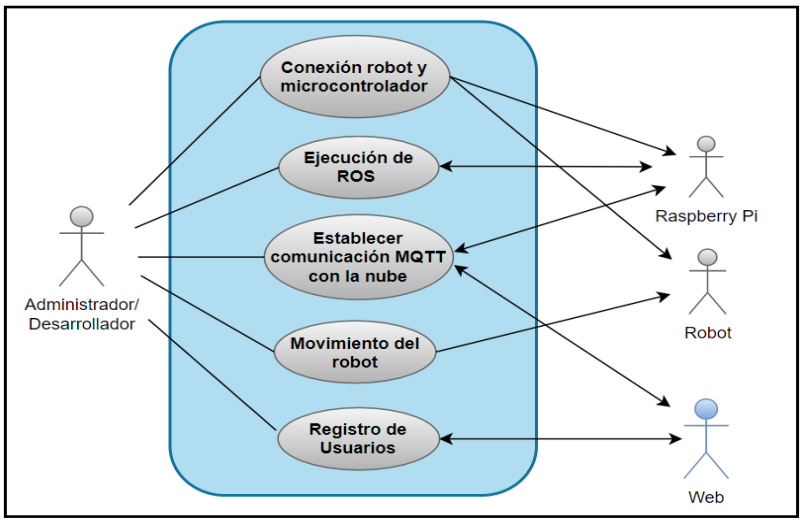


Figura N° 22: Diagrama de caso de uso, desarrollador.

A continuación, se detallan los casos de uso presentados para el desarrollador. Para la conexión del robot con el microcontrolador se conectan los periféricos necesarios para el uso del sistema operativo desde el dispositivo de control, se conecta el puerto ethercat pre configurado del robot, hacia el mini controlador, finalmente se encienden los motores.

Tabla 4: Caso de Uso – Conexión robot y microcontrolador

Caso de uso: Conexión robot y microcontrolador	
Actor	Desarrollador
Descripción	Caso de uso de conexión de equipos
Precondiciones	-
Flujo básico	1) Conectar periféricos necesarios a la Raspberry Pi. 2) Encender Raspberry Pi. 3) Conectar a la alimentación eléctrica el robot. 4) Conectar vía ethernet la Raspberry Pi y el robot. 5) Encender motores del robot.

Para que el sistema funcione y se cree la interacción entre el robot y el mini controlador es necesario ejecutar el ROS Master, mismo que valida la conexión del sistema, esto se lo lleva a cabo mediante la ejecución del nodo maestro desde la terminal de Ubuntu instalado en la Raspberry Pi, para una correcta ejecución se verifica la conexión vía ethernet y el encendido de motores para evitar fallos de conexión al momento de ejecutar el driver que nos permite la interacción del sistema con el robot.

Tabla 5: Caso de Uso – Ejecución de ROS

Caso de uso: Ejecución de ROS	
Actor	Desarrollador
Descripción	Caso de uso para ejecutar el ROS master
Precondiciones	Verificar estado activo del Sistema Operativo. Encender motores del robot.
Flujo básico	1) Ejecutar el youbot driver del robot. 2) Validar la conexión vía ethernet.

Para establecer la comunicación del dispositivo de control con el portal web, primero se debe ejecutar el nodo que permita la comunicación del sistema ROS por medio de MQTT al Boker MQTT privado, se debe correr y configurar en Node-RED las direcciones y puertos de los Brokers tanto público como privado para que la conexión sea exitosa. Se comprueba la conexión mediante el envío de un mensaje de tipo ROS por medio de MQTT hacia la nube, donde este debe actualizarse cada que el mensaje sea enviado sin ningún retardo ni inconveniente alguno.

Tabla 6: Caso de Uso – Establecer comunicación con la nube

Caso de uso: Establecer comunicación con la nube	
Actor	Desarrollador
Descripción	Caso de uso para comunicar el sistema con el portal web
Precondiciones	Correr el servidor Node-RED. Correr Broker MQTT privado.
Flujo básico	1) Establecer comunicación desde ROS a Broker MQTT privado. 2) Establecer conexión de Broker MQTT privado con Node RED. 3) Establecer conexión de Node-RED con Broker MQTT de público (nube).

Para acceder a la información proveniente de los sensores emitidos de las articulaciones del robot y dar lectura a sus datos, se verifica la ejecución del ROS master y que se haya establecido la comunicación del sistema con la nube. Se ingresa al Sistema ROS para listar los topics disponibles, aquí se debe identificar el nombre del topic desarrollado, mismo que contiene los datos provenientes de los sensores, donde para acceder a estos se ejecuta la suscripción respectiva del topic en mención.

Tabla 7: Caso de Uso – Lectura de datos de sensores

Caso de uso: Lectura de datos de sensores	
Actor	Desarrollador
Descripción	Caso de uso para leer el valor de corriente de las articulaciones del robot
Precondiciones	Ejecución de ROS master. Establecer conexión del sistema con la nube.
Flujo básico	1) Ingreso al sistema ROS. 2) Suscripción a topic publicador de los datos deseados.

Para la manipulación del robot se verifica la ejecución del sistema, la conexión adecuada del robot con el controlador, se define el topic que permite el ingreso de valores para dar movimiento a las juntas rotacionales del robot, se ingresa valores en particular para dar movimiento a cada articulación o se ejecuta directamente el programa que permita la publicación de la secuencia de movimientos deseados.

Tabla 8: Caso de Uso – Movimiento del robot

Caso de uso: Movimiento del robot	
Actor	Desarrollador
Descripción	Caso de uso para realizar la secuencia de movimientos del robot
Precondiciones	Ejecución de ROS master. Establecer conexión del sistema .
Flujo básico	1) Ingreso al sistema ROS. 2) Ejecución de la secuencia para el movimiento.

En esta sección se detallan las interacciones entre el usuario y el sistema. A continuación, se presenta el diagrama de caso de uso por parte del Usuario.

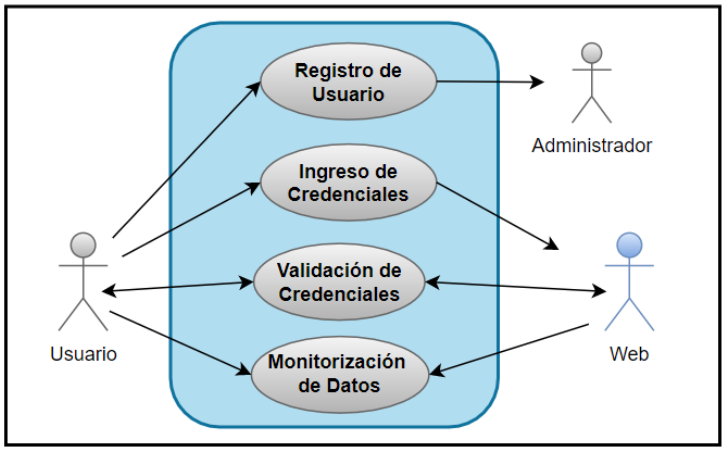


Figura N° 23: Diagrama de caso de uso, usuario.

Para poder acceder al portal web donde se alojan los datos a monitorizar, se debe como primer punto realizar un registro de usuario con el administrador, se dotan de las credenciales requeridas para la autenticación en el sistema.

Tabla 9: Caso de Uso – Registro de Usuario

Caso de uso: Registro de Usuario	
Actor	Usuario
Descripción	Caso de uso para registrar el usuario y contraseña
Precondiciones	-
Flujo básico	1) Dotar de una dirección mail y contraseña. 2) Validar la información con el administrador.

Para acceder a la página web se ubica la dirección del portal a utilizarse, se da clic en la pestaña Portal, para acceder a la Monitorización de datos.

Tabla 10: Caso de Uso – Ingreso de credenciales

Caso de uso: Ingreso de credenciales	
Actor	Usuario
Descripción	Caso de uso para ingresar al portal web AnyViz
Precondiciones	Registrar credenciales con el administrador
Flujo básico	1) Ingresar al portal 2) Ingresar el usuario y contraseña

Para la validación de credenciales se ingresa la contraseña y usuario, se da clic en la pestaña Login y se espera la confirmación para acceder al portal.

Tabla 11: Caso de Uso – Validación de credenciales

Caso de uso: Validación de credenciales	
Actor	Usuario
Descripción	Caso de uso para verificar credenciales de usuario
Precondiciones	Registrar credenciales con el administrador
Flujo básico	1) Dar clic en ingresar. 2) Esperar confirmación.

Ya ingresado en el portal, el usuario puede desplazarse en la página como el desee, llevar un registro y control de los datos presentados acorde las necesidades requeridas.

Tabla 12: Caso de uso – monitorización de datos

Caso de uso: Monitorización de datos	
Actor	Usuario
Descripción	Caso de uso para visualización de datos en tiempo real
Precondiciones	Ingreso al portal
Flujo básico	1) Navegar por la interfaz. 2) Registro y control de datos.

Etapa 2: Diseño del Sistema. – para la implementación del caso de estudio propuesto inicialmente, se desarrolló un sistema con ROS donde interaccionan tres nodos como se observa en la Figura N°24, esto permitirá controlar el brazo robótico del KUKA youBot por medio del API, misma que proporciona una interfaz bajo un framework de ROS. Con este framework puede mover la base y el brazo del youBot realizando un envío de mensajes ROS.

Para que los mensajes enviados llegue por medio del ROS master se ejecutan las acciones predefinidas a través de paquetes escritos en C ++, se describe a través de la API youBot el formato del mensaje a ser publicado y leído por el ROS master.

El controlador de brazo robótico se ejecuta con sus respectivas configuraciones, así como la ejecución del módulo de compensación gravitacional, con sus parámetros de calibración predefinidos.

Finalmente, con la transmisión de mensajes enviados por comando al formato URDF del robot se crea el acceso por medio de un tercer nodo al dispositivo de hardware a través del protocolo EtherCAT. A continuación, se representa la arquitectura del driver del manipulador, y cómo interactúan los nodos del controlador con el robot KUKA youBot, para de esta manera introducirnos al comportamiento general del Sistema.

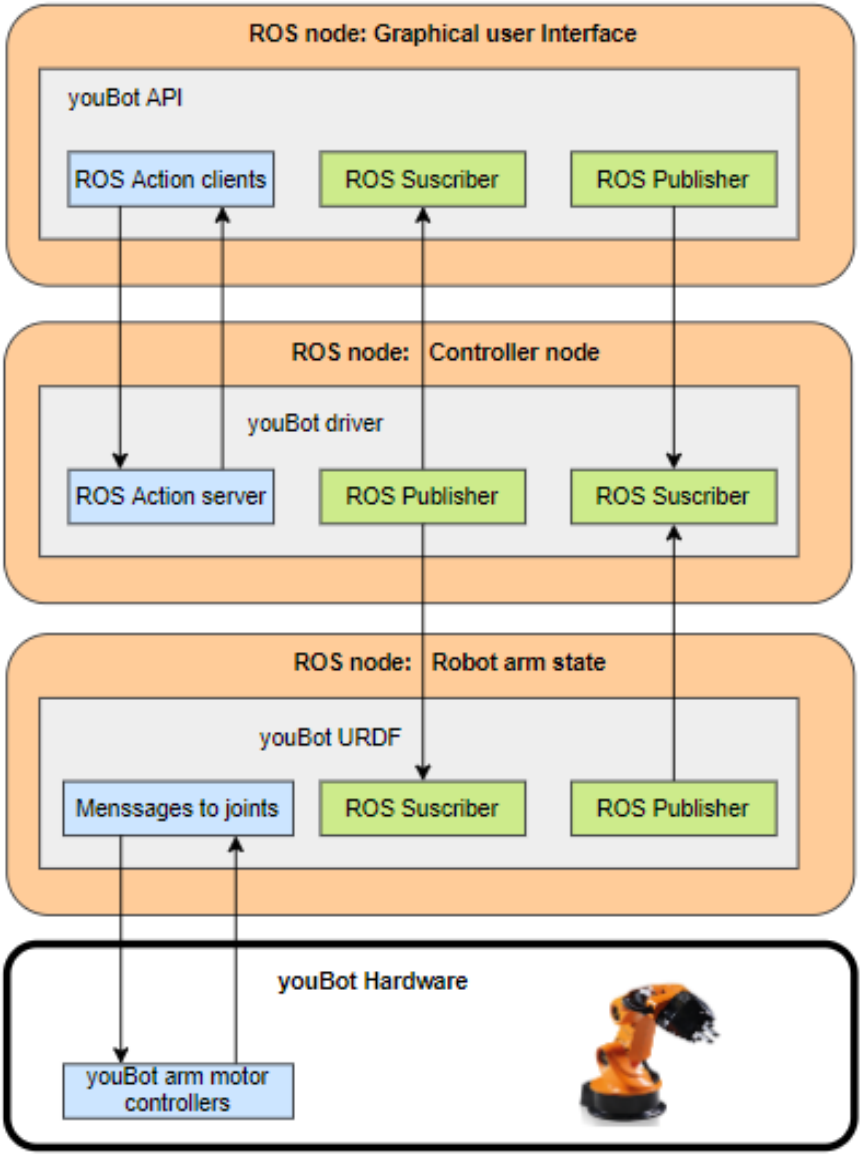


Figura N° 24: Arquitectura de comunicación youbot driver – manipulador

Para establecer la comunicación con el KUKA youBot, se representa a cada articulación del robot como una clase, es así como usando un diagrama de clases se describe la API del manipulador, aquí se muestran las clases con acceso a las juntas del brazo robótico.

Donde la clase *YouBotManipulator* tiene el acceso a una instancia o articulación en particular del brazo robótico a través de *YouBotJoint*, con *JointData* se pueden establecer y obtener los valores de setPoint o leer algunos datos de los sensores, en consecuencia, se establece la clase *JointSensedCurrent* para obtener los valores de corriente eléctrica provenientes de cada uno de los sensores que poseen las articulaciones. A continuación, se presenta el diagrama de clases utilizadas del API del robot.

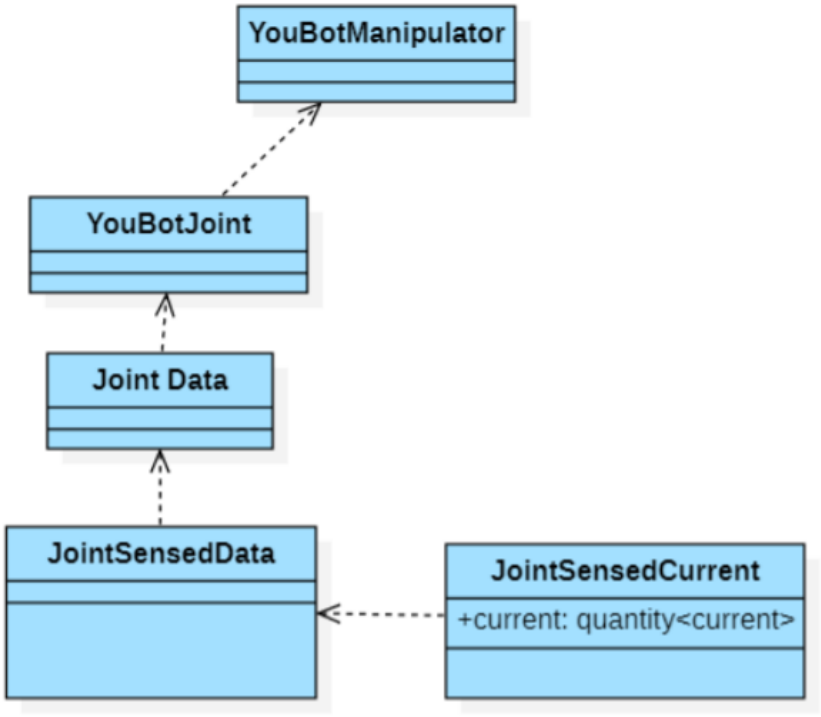


Figura N° 25: Diagrama de Clases

Para obtener los datos se fija un vector de cinco posiciones, donde cada posición se ocupará por cada una de las articulaciones del brazo robótico, correspondiente a los cinco grados de libertad establecido por la API de youBot. Los mensajes enviados desde el nodo del controlador al nodo del robot URDF son de tipo **sensor_msgs/JointState**, los cuales son usados para transmitir los valores en formato float64 dentro de las matrices creadas, para esto, la matriz irá incluyendo valores cada vez que se envíe un valor de tipo **current**, es así como de esta manera se accede a los sensores del motor de cada articulación para obtener el valor de corriente eléctrica en tiempo real y sin interferir con el envío de mensajes para el movimiento de brazo, al usar una comunicación asíncrona predefinida por el sistema ROS.

Para representar la comunicación, interacción y secuencia del sistema entre el dispositivo de control con la API o interfaz del robot, el controlador, hasta finalmente llegar a obtener los datos de los sensores del robot, se detalla a continuación un diagrama de secuencia del recorrido de datos.

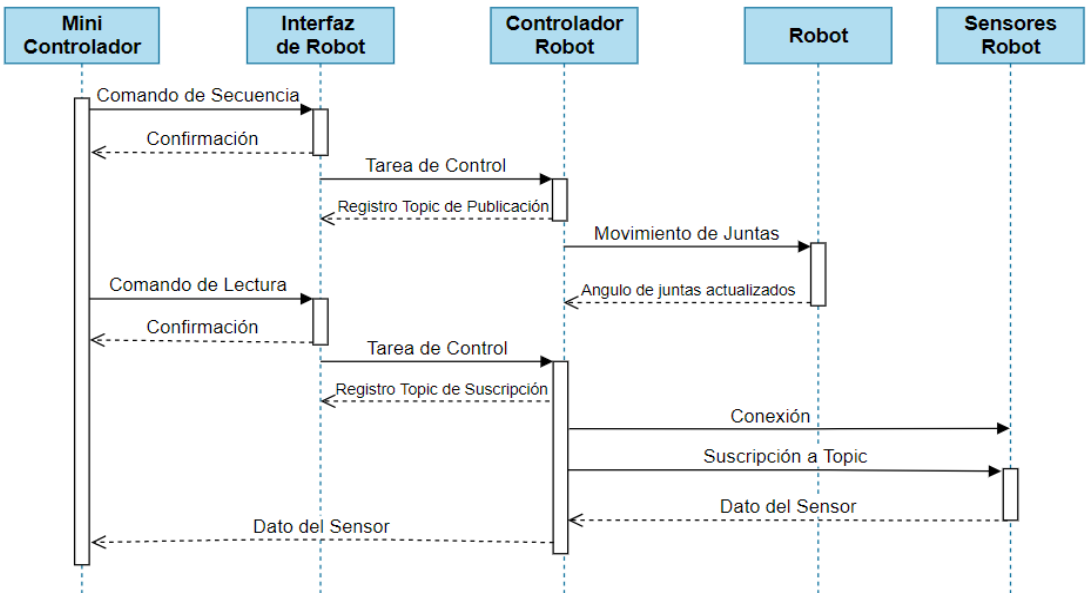


Figura N° 26: Diagrama de secuencia de flujo de datos

Se obtienen los datos del sensor mediante el paso de mensajes de ROS a MQTT con la ayuda del nodo */mqtt_brige* y se los envía a Node-RED para finalmente subirlos a la nube donde los datos serán representados gráficamente. Para dar control a las juntas giratorias del brazo robótico se usan mensajes de tipo brics actuator los cuales permiten controlar los motores a través del youBot API. Finalmente, la herramienta ROS Rqt Graph se ejecuta para detallar un diagrama en tiempo real de la comunicación e interacción entre nodos, topics y mensajes del sistema, se observa aquí el flujo de información dentro del sistema ROS desarrollado.

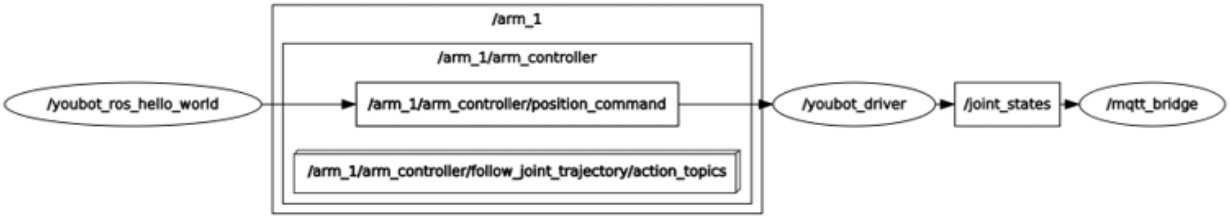


Figura N° 27: Flujo de datos general del Sistema – entorno real.

Para la secuencia del manipulador se desarrolló una tarea que se asemeja al movimiento de coger y poner una pieza desde diferentes lugares como se lo llevaría a cabo en una fábrica, donde el brazo del manipulador se acerca a la plataforma para simular el tomar un objeto, para luego girar hasta el extremo opuesto donde se simula que el objeto tomado se suelta en otra estación, se simula este agarre debido a que no se usó la apertura y cierre del gripper, en vista que la presente investigación da lectura solo a las articulaciones giratorias del robot, además se realizó de esta manera para facilitar el acople del contenedor al gripper donde se ubicaron los diferentes pesos para la secuencia de movimientos.

Se utilizaron varios pesos que van desde los 130, 230 y 330 gramos, para así dar lectura de los valores de corriente y verificar que, a mayor carga soportada por el robot, mayor es el consumo de energía por parte de las articulaciones, se llevaron a cabo varias pruebas con los diferentes pesos, tanto para la secuencia de movimientos como para diferentes posiciones de cada articulación en particular.

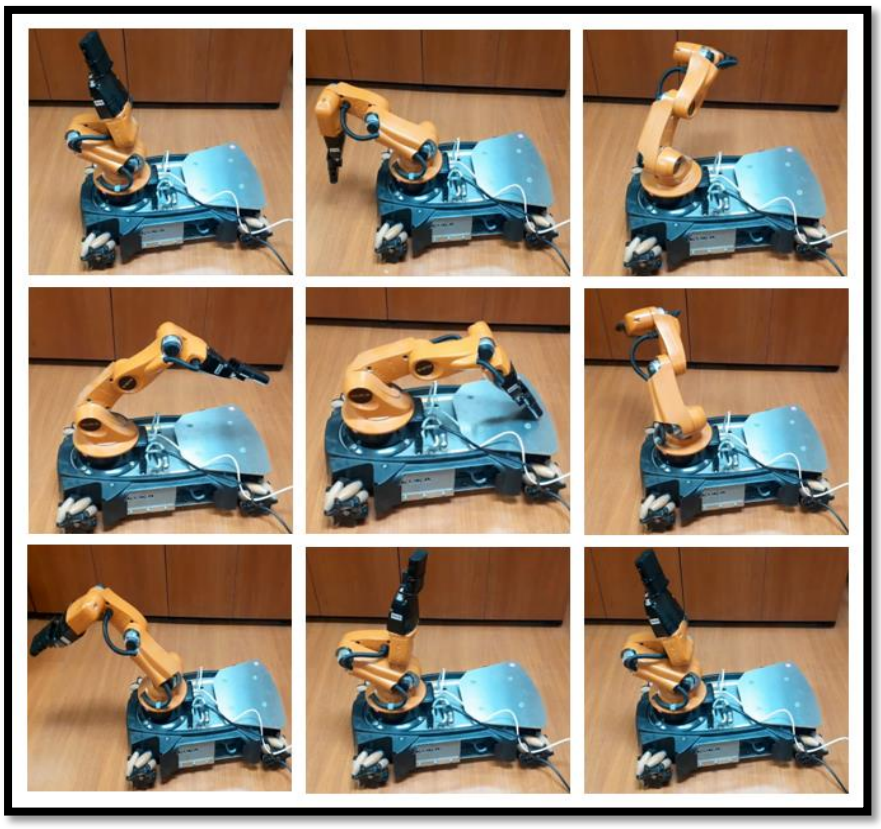


Figura N° 28: Secuencia con carga del KUKAyouBot

Etapa 3: Configuración del protocolo MQTT para la transmisión de datos. – la calidad de servicio utilizada en el presente proyecto es 0, de esta manera se garantiza la recepción del mensaje, y al mismo tiempo una entrega de datos en relación a las capacidades de la red. Para la seguridad se trabaja en el nivel de aplicación, se proporciona un identificador de cliente y credenciales de nombre de usuario y contraseña, con esto se evita el robo de información e ingreso de usuarios no deseados.

Del portal web **AnyViz** se obtienen varios datos pre configurados en la creación del proyecto MQTT en este portal, es así que para configurar al cliente MQTT en Node-RED es necesario tomar el nombre proyecto mismo que se asigna como nombre de usuario, al igual que el ID único del proyecto para autenticarlo y proceder al envío de datos, además se toma la contraseña respectiva creada por parte del administrador en el portal para su correcta autenticación.

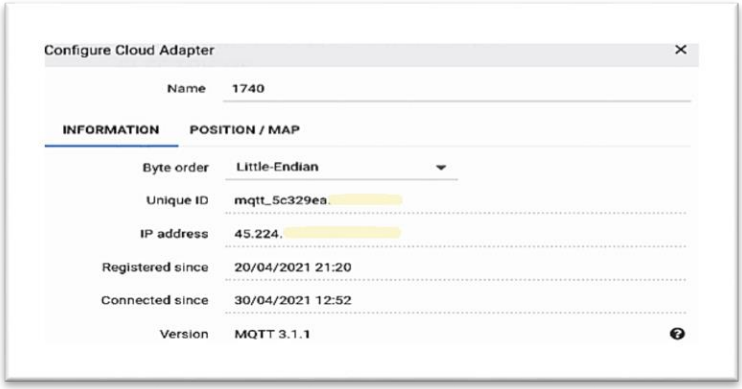


Figura N° 29: Datos de configuración requeridos - AnyViz

Para la configuración del bróker público se ingresa la dirección y puerto, el ID del proyecto, así como el nombre de usuario y contraseña previamente configuradas.

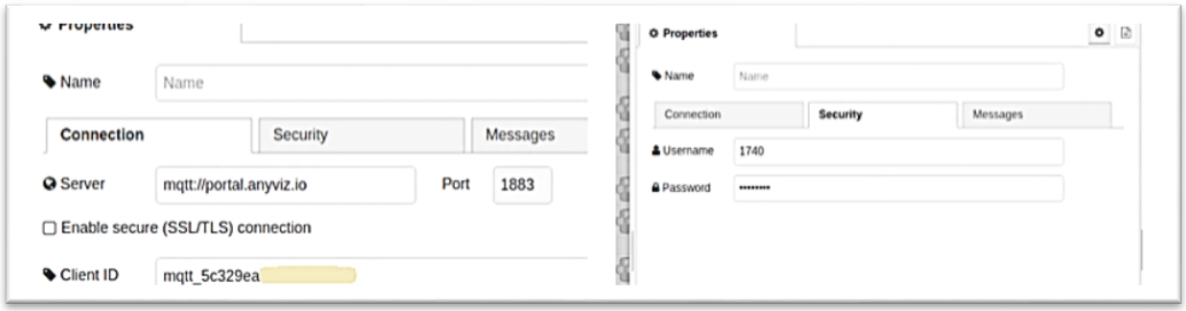


Figura N° 30: Configuración bróker público en node-red

Etapa 4: Obtención de datos en la nube. - tanto para la simulación como para el entorno real se usa el paquete Mqtt_bridge para enviar los datos desde ROS a Node-RED mediante el protocolo MQTT. En Node-RED se obtienen los valores del tema **currents** proveniente del topic de ROS denominado /joint_states. En la Simulación se toma el torque de las juntas y se realiza la función correspondiente para obtener el valor de corriente aproximado. Para el entorno real se accede directamente a los valores de corriente del topic en mención.

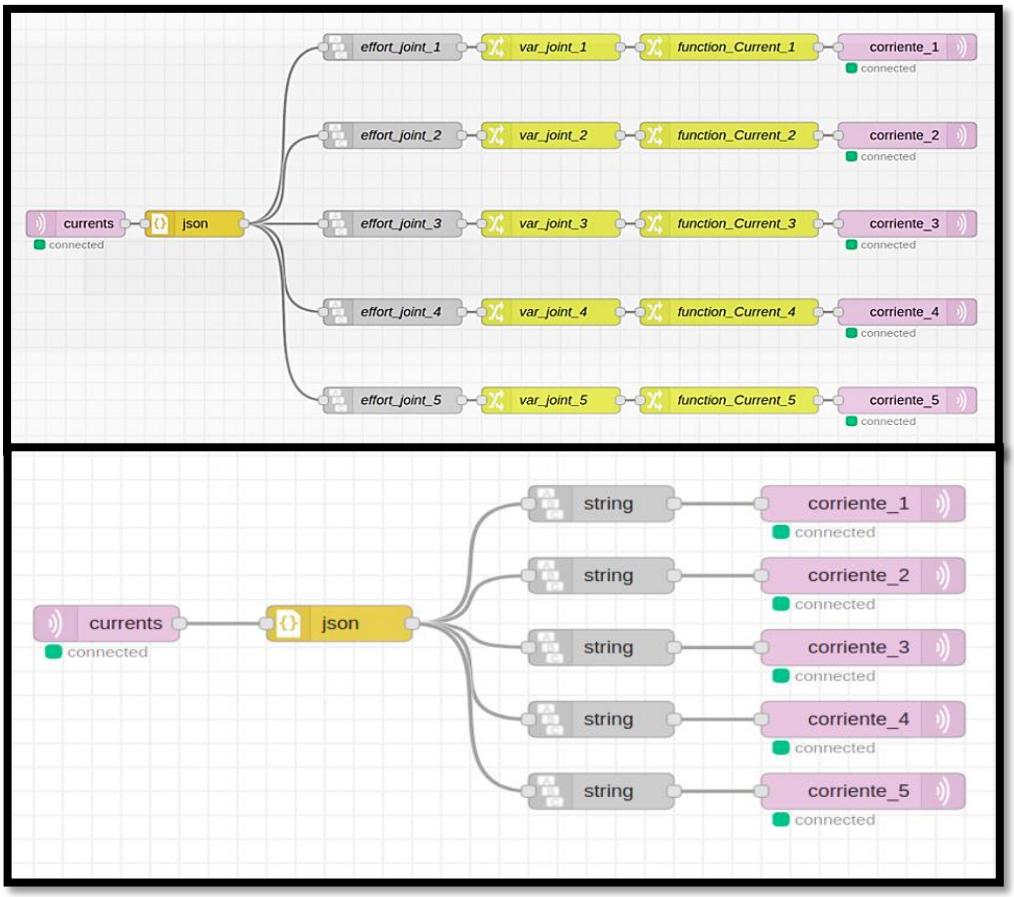


Figura N° 31: Red de nodos en Node-RED, Simulación – Entorno real

Etapa 5: Visualización de datos. - se muestra los valores de corriente eléctrica provenientes de cada una de las articulaciones del manipulador KUKAyouBot, se presenta la información de manera gráfica y en valores numéricos, se usa como unidad de medida eléctrica el Amperio. Al costado derecho superior se presenta una ilustración que detalla el número de juntas del manipulador mismo que coincide con el número que se detalla en las gráficas.

A continuación, se muestra los valores de la simulación y del entorno real. Estos datos son obtenidos al momento de ejecutar la secuencia de movimientos del manipulador.

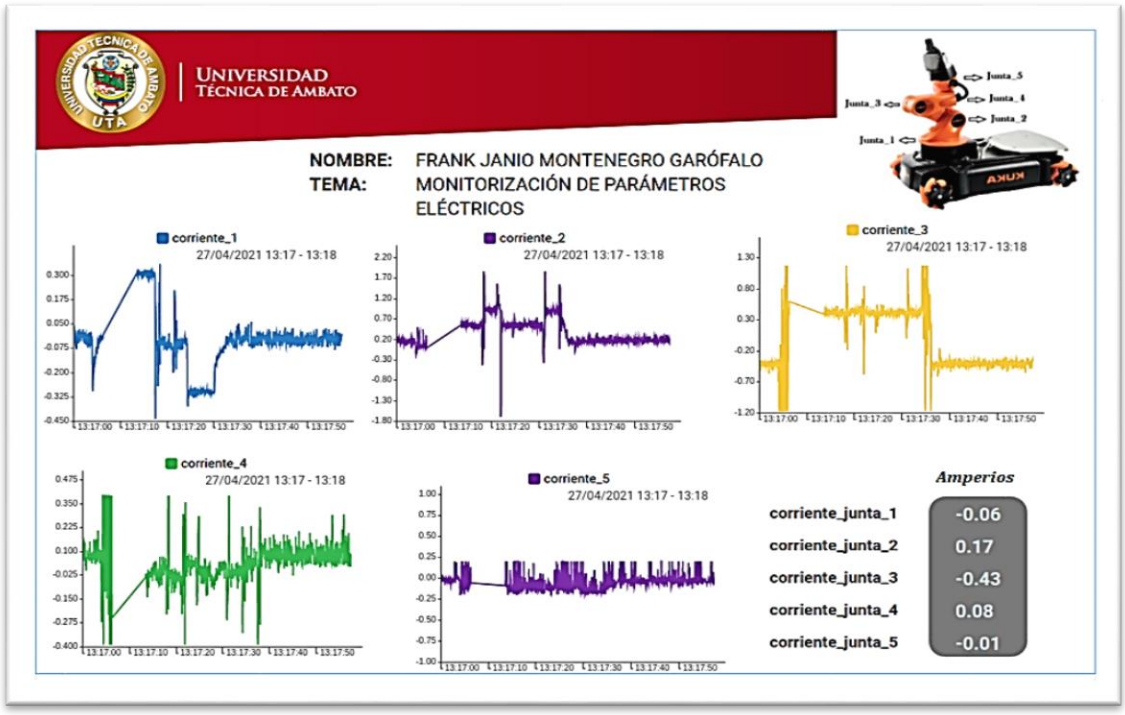


Figura N° 32: Monitorización de datos, Simulación

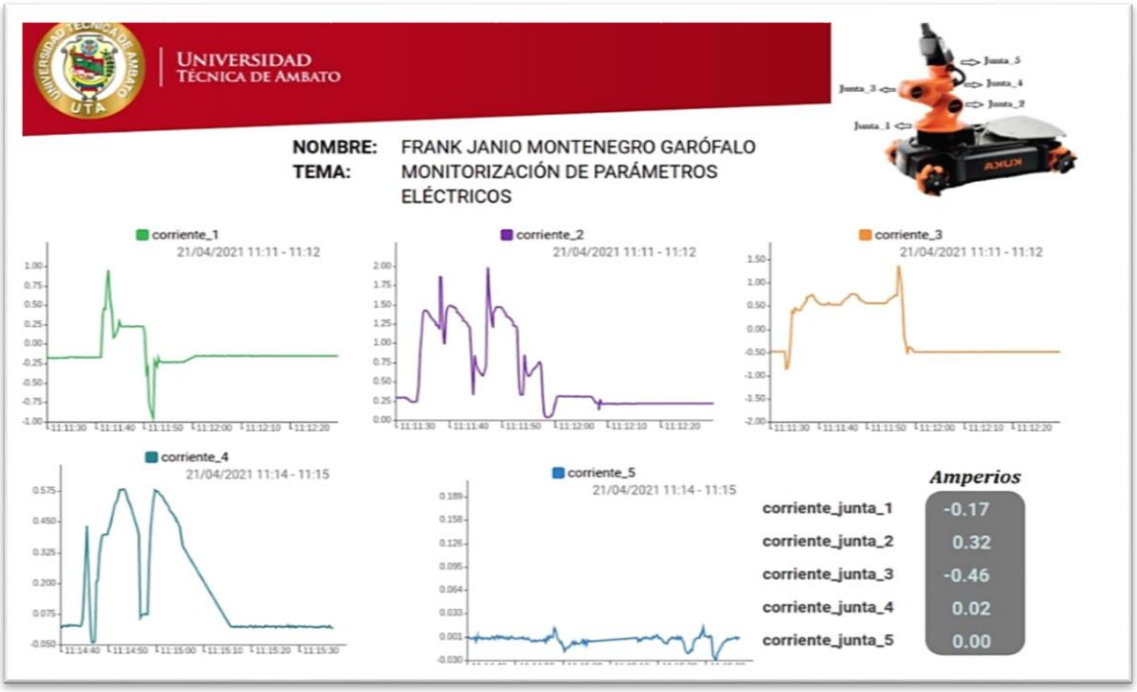


Figura N° 33: Monitorización de datos, entorno real

3.1.2 Análisis y discusión de los resultados

En el presente proyecto la simulación ha servido como una preparación para comprender el Sistema ROS, su funcionamiento y como realizar un uso adecuado del mismo. Introducirnos al comportamiento de los nodos, a la ejecución de los mismos, a publicar y suscribirse a topics y a entender su interacción entre ellos. Además, ha permitido relacionarse con el movimiento del manipulador, probar sus límites máximos rotacionales, de este modo se logra realizar el movimiento en el robot sin temor a causar daños a los motores. En las figuras 32 y 33 donde se realizó la lectura de datos con la secuencia en ejecución.

Análisis de tráfico de paquetes MQTT

Para el análisis de tráfico de red se utiliza el analizador de protocolos de código abierto WireShark, esta plataforma proporciona soporte para el tráfico de MQTT. WireShark se utilizó sobre el sistema operativo Ubuntu para la captura correspondiente del tráfico de datos y para su posterior análisis.

Se establece una secuencia de operación del Kuka Youbot para presentar la información sobre la interfaz del servidor web en AnyViz, durante la cual se realiza la captura de los paquetes de información publicados y subscritos por los clientes MQTT al bróker MQTT; esto se llevó a cabo durante un periodo superior a la duración de operación del manipulador, con el fin de identificar la constancia del tráfico en el intercambio de paquetes durante la frecuencia de operación en todo el periodo de captura.

En la Figura 34, se muestra de forma general el tráfico de red capturado, se utiliza las herramientas del software para generar el filtrado de protocolos para MQTT. En el cual, según el concepto del mismo define que su rendimiento es superior al de otros protocolos desde el punto de vista de volumen de tráfico; se observa un volumen que oscila entre 70 a 125 (paquetes/1segundo), donde su promedio se establece sobre los 100 (paquetes/1segundo).

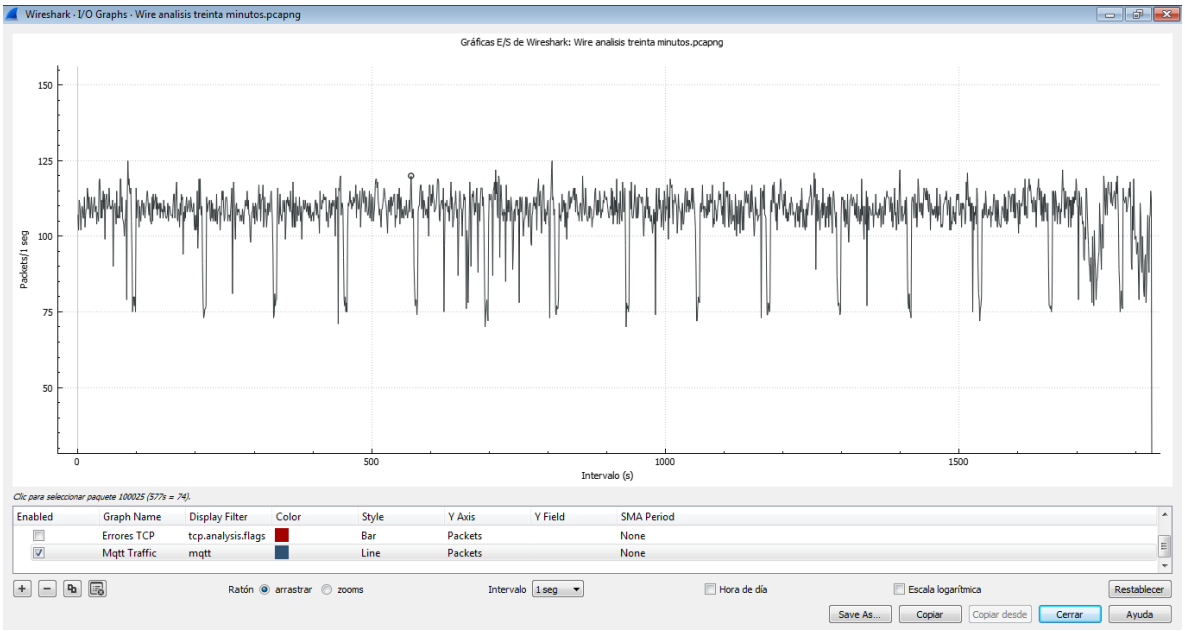


Figura N° 34: Gráfica de entrada y salida de paquetes MQTT.

Para una mejor apreciación del promedio general se tomó una sección de la captura del tráfico con mayores alteraciones y que cuente con el pico más alto y el pico más bajo, como se muestra en la figura N° 35, la sección va desde el intervalo del segundo 690 hasta el segundo 810, dando como promedio un tráfico de 107 (paquetes/1segundo).

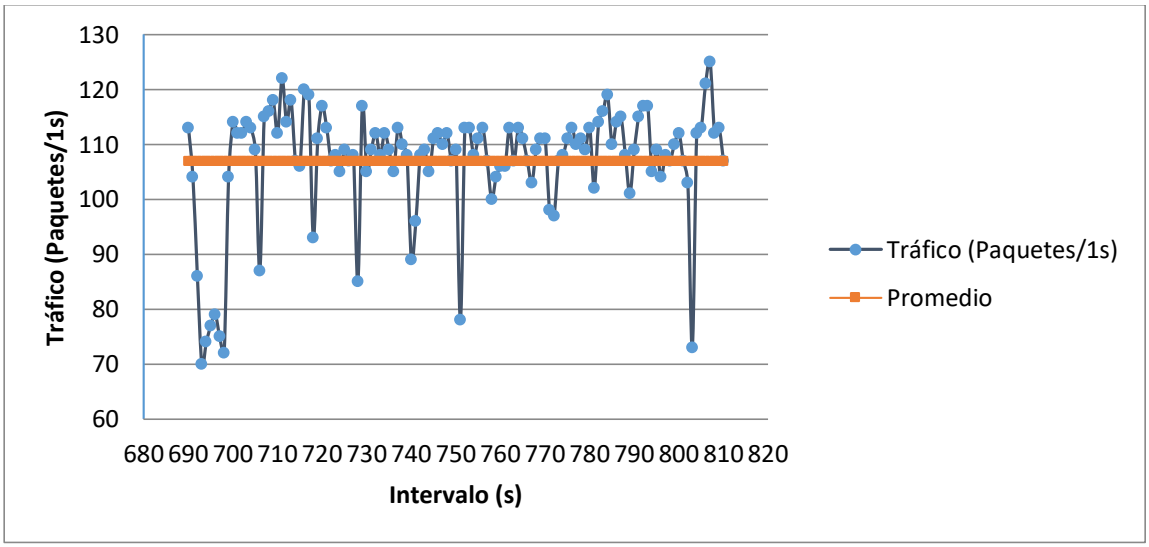


Figura N° 35: Promedió calculado durante un periodo de tráfico MQTT.

Análisis de tráfico de paquetes por cliente MQTT

Cliente publicador.

Es un dispositivo Raspberry en el nivel de planta, este cliente MQTT publica en el topic currents del broker MQTT el valor de la corriente de cada motor del KUKAyouBot. El tráfico de red de la comunicación entre este cliente MQTT y el bróker MQTT se puede observar en la Figura 36, su indicador es de color anaranjado, su tasa de transmisión es en tiempo real por lo que genera un tráfico mayor que oscila desde 53 hasta 98 (paquetes/1segundo).

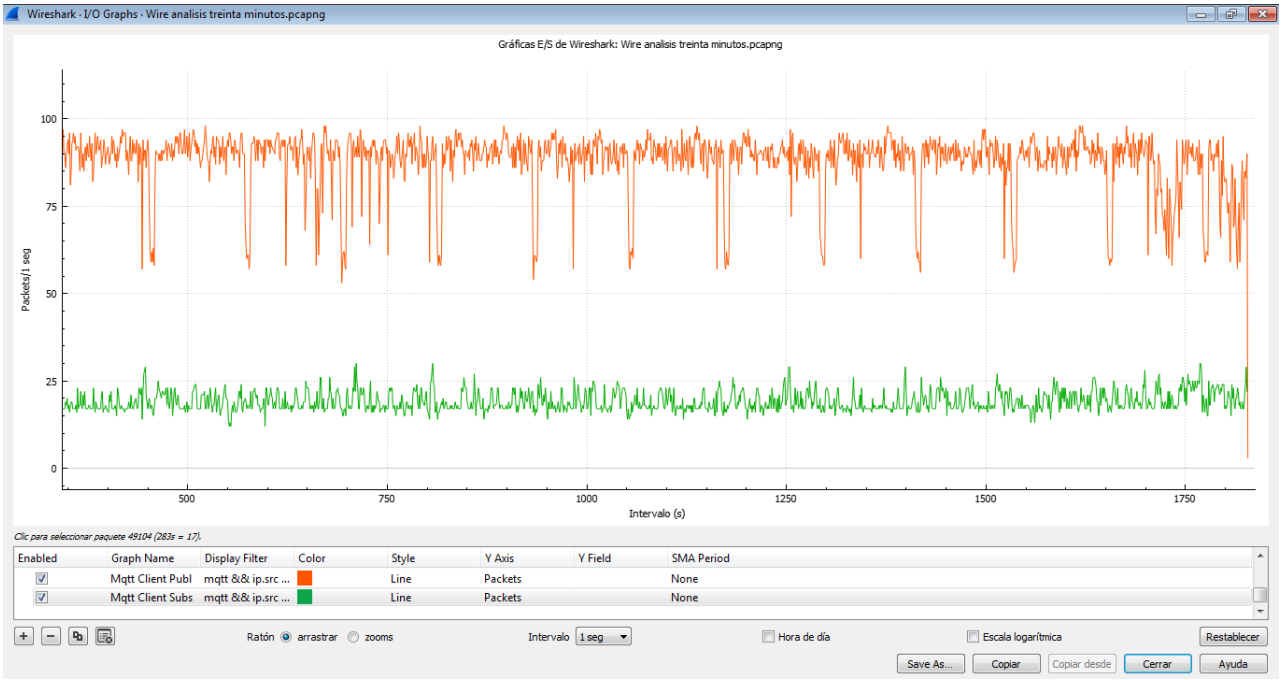


Figura N° 36: Gráfica de entrada y salida de paquetes de cada cliente MQTT.

Cliente subscriptor.

Es un dispositivo PC en el nivel de gestión, este cliente MQTT se subscribe al topic currents del bróker MQTT, de esta forma recoge el valor en los topics de la corriente de cada motor y lo presenta en una interfaz visual del servidor web de AnyViz (interfaz web). El tráfico de red de la comunicación entre este cliente MQTT y el bróker MQTT, también se presenta en la Figura 36, su indicador es de color verde, su tasa de refrezco depende de las características del indicador de la interfaz web por lo que genera un tráfico menor que oscila desde 12 hasta 33 (paquetes/1segundo).

Finalmente, como resultado de la medición del tráfico de red se pudo determinar la medición del tiempo de retardo de acceso entre la comunicación de paquetes MQTT, considerando el promedio general de 107 paquetes por 1segundos, entonces se tiene un retardo general en promedio de 0,0093s entre paquetes MQTT, lo que representa un retraso de acceso muy pequeño, consiguiendo de esta forma un acceso óptimo MQTT.

Tabla 13: Resumen de datos estadísticos de la captura del tráfico de red en WireShark.

<table>
<tr><th colspan="8">Tráfico MQTT</th></tr>
<tr><td>Publicador</td><td colspan="7">Raspberry pi (mosquito-client-pub = “192.168.1.42”)</td></tr>
<tr><td>Subscriptor</td><td colspan="7">AnyViz (php-mqtt-client-sub = “52.174.100.142”)</td></tr>
<tr><td>Broker</td><td colspan="7">Pc Ubuntu (mosquito-broker = “192.168.1.47”)</td></tr>
<tr><td colspan="3">Escenario del entorno de trabajo</td><td colspan="5">Análisis WireShark</td></tr>
<tr><td rowspan="2">N° Items</td><td rowspan="2">N° Dispositivos</td><td rowspan="2">Tiempo medición (min)</td><td colspan="2">Paquetes capturados (publicador y broker)</td><td colspan="2">Paquetes capturados (broker y subscriptor)</td><td rowspan="2">Total Paquetes MQTT</td></tr>
<tr><td>Publicador a Broker</td><td>Broker a Publicador</td><td>Subscriptor a Broker</td><td>Broker a Subscriptor</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">3</td><td rowspan="2">30</td><td>161.218</td><td>79.907</td><td>36.129</td><td>38.680</td><td>315.934</td></tr>
<tr><td colspan="4">Paquetes Malformados</td><td>10.830</td></tr>
</table>

Los datos más significativos de la medición del tráfico de red se presentan en la Tabla 13, esta tabla contiene en resumen la estadística de conversación MQTT para el análisis de la tasa de entrega de las publicaciones entre cada extremo. Esta información nos permite determinar la eficacia del sistema mediante el análisis de entrega de paquetes de manera correcta en el procesamiento de datos, para esto se aplica la fórmula de indicador de eficacia presentada a continuación.

$$Eficacia = \frac{Paquetes\ entregados\ correctamente * 100}{Total\ de\ paquetes\ entregados} \quad (4)$$

Partiendo de la ecuación 4, se determina el porcentaje de eficacia del sistema desarrollado en el presente proyecto; en primer lugar, se identifican 305104 Paquetes Entregados Correctamente, 10830 Paquetes Entregados Malformados y un Total Paquetes Entregados de 315934, en la siguiente figura se representan estos valores porcentualmente.

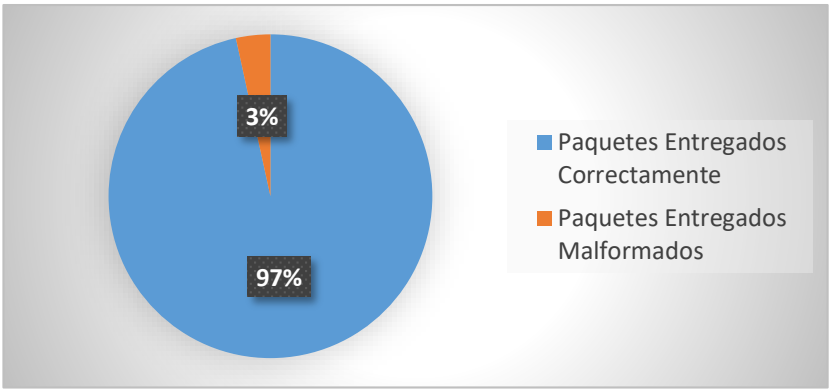


Figura N° 37: Grafica porcentual de paquetes entregados.

Se realiza una ponderación de cinco escalas para de esta forma determinar el porcentaje de eficacia del sistema obteniendo como resultado un 96,57%, mismo que según el equivalente de la Figura N° 38, el sistema es considerado como muy eficaz.

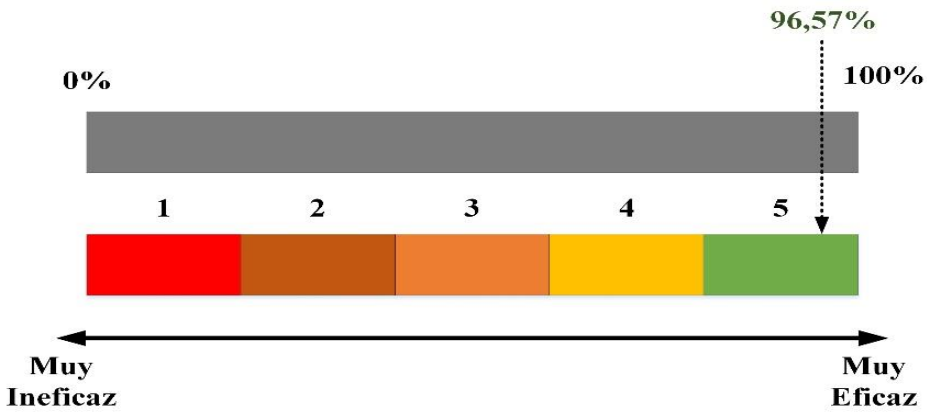


Figura N° 38: Gráfica del nivel de eficacia del sistema.

Ejecución con varios pesos

Esta se realizó con una variación de pesos de 100 gramos, iniciando en 130 gramos hasta los 330 gramos, en vista que el peso de carga máxima del manipulador es de 500 gramos como indica el Anexo N° 1, para de esta manera no forzar los motores, así se obtuvieron las gráficas de cada articulación con los 3 pesos. A continuación, se aprecia una de las gráficas obtenidas. Por ejemplo, para la articulación 2 con un peso de 330 gramos, el consumo de corriente eléctrica máxima en ciertos puntos al ejecutar la secuencia es de 2.68 A, sobrepasando el valor de corriente nominal de 2.32 A.

Figura N° 39: Articulación 2, secuencia con 130 gramos

Desde el Anexo N° 4 al 23 los valores obtenidos demuestran el aumento de consumo de corriente eléctrica de cada articulación acorde va incrementando el peso acoplado al brazo robótico KUKA youBot.

3.2 Verificación de hipótesis

Para la variable independiente se verifica la adquisición de datos de corriente eléctrica de las articulaciones del robot desde el Sistema ROS por medio del protocolo MQTT hasta llevarlos a la nube en la sección 3.1.2, con el análisis de Tráfico presentado con el software Wireshark, en la Figura Nº 34 se aprecia el trafico general obtenido de la comunicación desde el dispositivo de planta hasta la visualización de datos en el portal web, obteniendo una eficacia del 97%.

Para la variable dependiente, sobre identificar la velocidad con la que se transmiten los paquetes, de igual manera se utiliza el software Wireshark, ya que permite conocer la velocidad de trasmisión de paquetes del protocolo. Para la verificación de la hipótesis se toma una muestra de 30 datos que miden el tiempo de retardo entre paquetes transmitidos, estos son medidos en intervalos de segundo a segundo, se registra en un tiempo total de 30 segundos que va desde el intervalo del segundo 5 hasta el segundo 34, dado que en este periodo se ejecutó la secuencia del manipulador, ya que dicha secuencia tiene una duración total de 28 segundos, a continuación, se muestra la tabla de datos obtenidos.

Tabla 14: Retardos obtenidos por intervalos de tiempo

Fuente: Elaboración propia del Investigador

Intervalo de ejecución (s)	Retardo entre paquetes (ms)	Intervalo de ejecución (s)	Retardo entre paquetes (ms)
5	6,0241	**20**	5,8480
6	5,8480	**21**	58,8235
7	5,5866	**22**	5,6497
8	5,6818	**23**	5,3763
9	5,7803	**24**	5,6818
10	58,8235	**25**	5,8480
11	55,5556	**26**	5,6818
12	6,0241	**27**	5,7143
13	5,8140	**28**	55,5556
14	5,6818	**29**	5,7471
15	5,7471	**30**	5,7471
16	5,3476	**31**	5,7471
17	5,4348	**32**	6,1350
18	5,6818	**33**	5,7803
19	5,4645	**34**	5,7471

Se utiliza la prueba de T – student para la comprobación de la hipótesis, como primer paso se identifican la hipótesis nula y alternativa para la variable dependiente, siendo estas:

H_0: La velocidad de transmisión entre paquetes es suficiente para considerarse como un sistema en tiempo real.

H_1: La velocidad de transmisión entre paquetes no es suficiente para considerarse como un sistema en tiempo real.

Es necesario conocer que el retardo establecido para una transmisión en tiempo real se considera un máximo de 200 ms, de esta manera se asigna este valor como medida de referencia (**μ**). Se presenta a continuación las relaciones para la aceptación de las hipótesis planteadas. Para la hipótesis nula $H_0 <$ **μ** y para aceptar la hipótesis alternativa $H_1 >$ **μ,** se considera como nivel de significancia $\alpha = 0.05$, el número de datos de la muestra **n** =30, el grado de libertad calculado es 29 proveniente de (**n-1**) y una desviación estándar de la muestra ($\boldsymbol{\sigma}$) calculada de 317,026.

Al no conocer la desviación estándar de la población, tener un número de muestra igual a 30 y saber que los datos provienen de una distribución normal, en base a esto y como lo señala el método, la distribución adecuada a calcular es $\boldsymbol{t_c}$**,** dada por la siguiente fórmula.

$$t_c = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

$$t_c = \frac{12{,}58 - 200}{\frac{317{,}026}{\sqrt{30}}}$$

$$t_c = \frac{-187{,}41}{57{,}88} = -3{,}237$$

Para obtener el valor crítico se utiliza la tabla T – student, nos ubicamos en la columna 0.05 y localizando en las filas el número de grados de libertad calculado, se obtiene el valor $\boldsymbol{t}$ que corresponde a 1,6972.

Con el valor estadístico $\boldsymbol{t}$ y $\boldsymbol{t_c}$ calculado, se procede a la comprobación según la posición del dato en relación.

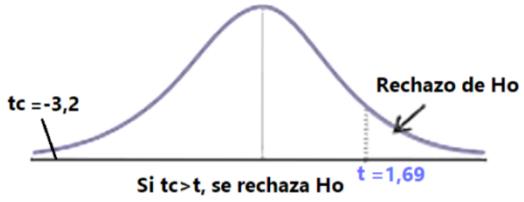


Figura N° 40: Localización de región crítica

Se procede a validar la hipótesis nula, al estar $\boldsymbol{t_c}$ en la zona de aceptación, de esta manera se tiene suficiente validación estadística para afirmar que "La velocidad de transmisión es suficiente para considerarse como un sistema en tiempo real".

Finalmente, se concluye en referencia a la hipótesis general de la sección 2.2.3, ya que se ha demostrado la veracidad de las variables identificadas, tanto para la variable independiente como dependiente, de esta manera se puede afirmar que:

"Con la utilización del protocolo MQTT se pueden leer en tiempo real los datos de corriente eléctrica de las articulaciones del robot KUKA youBot".

CAPITULO IV.- CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La Simulación en Gazebo permitió conocer el comportamiento del manipulador en cuanto a sus movimientos y límites rotacionales. Se realizaron pruebas controladas para reducir el riesgo de errores y comportamientos inesperados en el entorno real, se desarrolló la comunicación y manipulación mediante ROS, esto permitió comprender el funcionamiento del Sistema para luego trasladarlo al entorno real al poseer la misma arquitectura de funcionamiento. Para recibir datos similares a los del manipulador real, deben ser configurados en simulación tanto la velocidad como el esfuerzo de cada junta como lo describe la API del controlador para el KUKAyouBot físico, sin embargo, los datos resultantes fueron aproximados, pero simulares en los picos máximos de las gráficas generadas.

- Se establecieron los parámetros de comunicación desde ROS hacia el manipulador con el fin de poder controlarlo, para esto es necesario incluir todas las librerías, paquetes y drivers necesarios para el correcto funcionamiento del API del youbot. Además, se instalaron las dependencias necesarias directamente al Sistema operativo desde la terminal de Linux, con el fin de cumplir con todos los requisitos para la correcta compilación del código.

- Para la monitorización se usó el protocolo MQTT, este permitió la publicación de varios tópicos a través del broker Mosquitto, siendo un protocolo asíncrono donde cada sensor logra enviar los datos leídos a los topics de manera simultánea posibilitando un tiempo de respuesta rápido, apto para aplicaciones de tiempo real, resultando un sistema con una eficacia de 97%. Además, se realizó la Interfaz gráfica en el portal AnyViz, donde por medio de la herramienta Node-RED se estableció la conexión con el broker público MQTT. Se obtuvieron gráficas de la corriente con diferentes pesos, así como también las gráficas insertando ángulos de giro para cada articulación en particular, esto demostró que la corriente consumida es directamente proporcional a la carga que soporta el manipulador.

4.2 Recomendaciones

- Se debe considerar la Simulación como primer paso a realizar previo al uso del manipulador para establecer una comunicación estable y además comprender el funcionamiento de ROS en cuanto a su arquitectura, topics y nodos, así como generar su correcta ejecución, en vista que el modelo de comunicación entre ROS y la Simulación como con el manipulador real, se basan en el mismo Sistema. Se recomienda experimentar con el movimiento en la Simulación sin temor a equivocarse, para así tener seguridad al momento de llevar a cabo la ejecución en el brazo robótico real, evitando fallos y daños a los motores.

- El uso la imagen para Rapsberry Pi de la empresa Ubiquity Robotics es recomendable, debido a que encontrar las versiones exactas tanto de ROS como de los controladores del manipulador para la Raspberry Pi crean un tema complejo de definir, al tener librerías con poco mantenimiento y muchas de ellas ya obsoletas. Es así que trabajar con ROS Kinetic sobre Ubuntu 16.04 garantiza estabilidad del Sistema, fácil acoplamiento de nuevos nodos, la existencia de paquetes probados y funcionales y además evita que la comunicación con el robot se pierda.

- Se recomienda crear una comunicación estable para lo cual es necesario contar con un SO operativo y versión de ROS que trabajen en conjunto, así como también es fundamental que la conexión a Internet no se pierda, logrando de esta manera evitar la pérdida de datos. No es recomendable utilizar redes de telefonía móvil debido a que la capacidad de cobertura puede afectar en la QoS (Calidad de Servicio). Además, se recomienda utilizar plataformas que permitan escalabilidad, facilidad de uso y simple implementación de nuevos datos, esto debido a que todo proyecto debe tener predisposición a futuro, donde sea necesario incluir más sensores y lectura de los valores requeridos, asimismo poseer fácil interacción con los datos ya existentes. Se recomienda usar la seguridad necesaria para evitar el robo de información o alteración de los valores monitorizados.

C. MATERIALES DE REFERENCIA

Referencias Bibliográficas

[1] J. Engelberger, *Robotics in practice: management and applications of industrial robots*, 1980th ed. 2012.

[2] Paryanto, M. Brossog, M. Bornschlegl, and J. Franke, "Reducing the energy consumption of industrial robots in manufacturing systems," *Int. J. Adv. Manuf. Technol.*, vol. 78, no. 5–8, pp. 1315–1328, May 2015.

[3] D. Meike and L. Ribickis, "Energy efficient use of robotics in the automobile industry," in *2011 15th International Conference on Advanced Robotics (ICAR)*, 2011, pp. 507–511.

[4] L. D. Evjemo, T. Gjerstad, E. I. Grøtli, and G. Sziebig, "Trends in Smart Manufacturing: Role of Humans and Industrial Robots in Smart Factories," *Curr. Robot. Reports*, vol. 1, no. 2, pp. 35–41, Jun. 2020.

[5] A. G. Starr, I. J. Kennedy, and R. J. Wynne, "A System for On-Line Performance Monitoring of Industrial Robots," in *Proceedings of the Thirtieth International MATADOR Conference*, London: Macmillan Education UK, 1993, pp. 617–625.

[6] B. Sonkoly *et al.*, "Scalable edge cloud platforms for IoT services," *J. Netw. Comput. Appl.*, vol. 170, p. 102785, Nov. 2020.

[7] M. Kashyap, V. Sharma, and N. Gupta, "Taking MQTT and NodeMcu to IOT: Communication in Internet of Things," *Procedia Comput. Sci.*, vol. 132, pp. 1611–1618, 2018.

[8] M. Koseoglu, O. M. Celik, and O. Pektas, "Design of an autonomous mobile robot based on ROS," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, 2017, pp. 1–5.

[9] J. M. Aitken, S. M. Veres, and M. Judge, "Adaptation of System Configuration under the Robot Operating System," *IFAC Proc. Vol.*, vol. 47, no. 3, pp. 4484–4492, 2014.

[10] M. Isik and M. Haboglu, "Design and Implementation of Real Time Monitoring and Control System for Robot Arms Used in Industrial Applications," 2015.

[11] D.-E. Kim, H.-N. Yoon, K.-S. Kim, M. S. Sreejith, and J.-M. Lee, “Using current sensing method and fuzzy PID controller for slip phenomena estimation and compensation of mobile robot,” in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2017, pp. 397–401.

[12] Z. Huang, C. Zhang, C. Wang, G. Yang, and C.-Y. Chen, “Parameter identification of flexible robotic joint based on multi-sensor information,” in *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2017, pp. 376–380.

[13] E. Abele *et al.*, “Konzeption und Entwicklung eines Condition Monitoring Systems mit Low Cost Sensoren zur Überwachung von Roboterschwingungen,” *TUPrints*, 2016.

[14] “Documentation - ROS Wiki.” [Online]. Available: http://wiki.ros.org/. [Accessed: 04-May-2021].

[15] R. Bischoff, U. Huggenberger, and E. Prassler, “KUKA youBot - a mobile manipulator for research and education,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1–4.

[16] F. Mirelez-Delgado, A. Morales-Díaz, R. Ríos-Cabrera, and H. Pérez-Villeda, “Control Servovisual de un Kuka youBot para la manipulación y traslado de objetos,” *Congr. Nac. Control Automático, AMCA 2015*, no. 2012, pp. 239–244, 2015.

[17] Locomotec, “KUKA youBot User Manual,” pp. 1–43, 2011.

[18] P. González-Nalda, I. Calvo, I. Etxeberria-Agiriano, E. Zulueta, and J. Lopez-Guede, “Hacia un framework basado en ROS para la implementación de Sistemas Ciberfísicos,” 2015.

[19] “catkin/workspaces - ROS Wiki.” [Online]. Available: http://wiki.ros.org/catkin/workspaces. [Accessed: 22-May-2021].

[20] “Nuestra mision | Ubuntu.” [Online]. Available: https://ubuntu.com/community/mission. [Accessed: 17-Sep-2019].

[21] “Kiosko.” [Online]. Available: http://gazebosim.org/. [Accessed: 18-Sep-

2019].

[22] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154.

[23] E. Rohmer, S. P. N. Singh, and M. Freese, “V-REP: A versatile and scalable robot simulation framework,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326.

[24] O. Michel, “Cyberbotics Ltd. Webots ™: Professional Mobile Robot Simulation,” *Int. J. Adv. Robot. Syst.*, vol. 1, no. 1, p. 5, Mar. 2004.

[25] A. Ayala, F. Cruz, D. Campos, R. Rubio, B. Fernandes, and R. Dazeley, “A Comparison of Humanoid Robot Simulators: A Quantitative Approach,” in *2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2020, pp. 1–6.

[26] “Presentamos el nuevo ODROID-C4: Un Ordenador de Placa Reducida de Nueva Generación | ODROID Magazine.” [Online]. Available: https://magazine.odroid.com/es/article/introducing-the-new-odroid-c4-a-new-generation-single-board-computer/. [Accessed: 11-May-2021].

[27] “Banana Pi.” [Online]. Available: http://www.banana-pi.org/. [Accessed: 11-May-2021].

[28] “LattePanda – A Windows 10 Computer with integrated Arduino.” [Online]. Available: https://www.lattepanda.com/. [Accessed: 20-Sep-2019].

[29] E. Upton and G. Halfacree, *Raspberry Pi® User Guide*. 2016.

[30] “Node-RED.” [Online]. Available: https://nodered.org/. [Accessed: 11-May-2021].

[31] “Quality of Service 0,1 & 2 - MQTT Essentials: Part 6.” [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. [Accessed: 11-May-2021].

[32] “HiveMQ - Fundamentos de seguridad de MQTT.” [Online]. Available: https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals/.

[Accessed: 11-May-2021].

[33] “Mosquitto | The Eclipse Foundation.” [Online]. Available: http://www.eclipse.org/proposals/technology.mosquitto/. [Accessed: 11-May-2021].

[34] “AnyViz: la nube para su PLC.” [Online]. Available: https://www.anyviz.io/. [Accessed: 11-May-2021].

ANEXOS

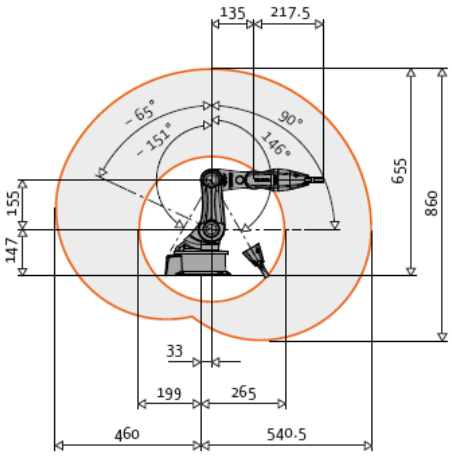
Anexo 1: Especificaciones técnicas del actuador

youBot Arm

Experience Manipulation

ENTER INTO THE AREA OF ROBOTIC MANIPULATORS. With the external, controllable 5-axis kinematics you can practically demonstrate and analyze the basics of robotics kinematics and dynamics. It is possible to perform basic grasping applications and, if extended with sensors, you can realize applications with grasp planning.

Datos del actuador	Junta 1	Junta 2	Junta 3	Articulación 4	Articulación 5	Rueda
Motor						
Voltaje nominal (V)	24	24	24	24	24	24
Corriente nominal (A)	2,32	2,32	2,32	1.07	0,49	2,32
Par nominal (mNm)	82,7	82,7	82,7	58,8		82,7
Inductancia terminal (mH)	0.573	0.573	0.573	2,24	7.73	0.573
Resistencia terminal (Ω)	0,978	0,978	0,978	4.48	13,7	0,978
Momento de inercia (kg * mm2)	13,5	13,5	13,5	9.25	3,5	13,5
Velocidad nominal (RPM)	5250	5250	5250	2850	2800	5250
Peso (gramos)	110	110	110	75	46	110
Caja de cambios						
Relación de reducción	156	156	100	71	71	26
Momento de inercia (kg * mm2)	0,409	0,409	0.071	0,07	0,07	0,14
Peso (gramos)						224
Codificador						
Recuentos por revolución	4000	4000	4000	4000	4000	4000



The arm on the Kuka youBot mobile robot

The technical specifications of the arm are as follows:

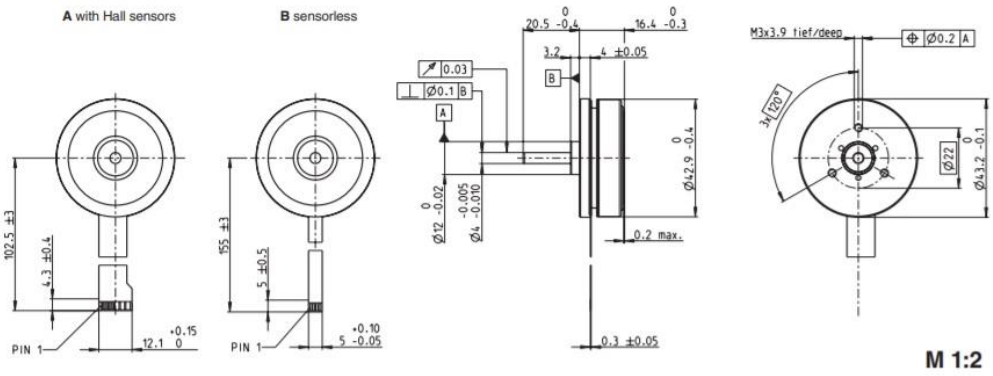
- Degrees of freedom: 5
- Height: 655 mm
- Working envelope: 0.513 m²
- Weight: 5.3 kg
- Payload: 0.5 kg
- Structure: magnesium casting
- Repeatability (position): 1 mm
- Communication: EtherCAT
- Voltage: 24V DC
- Drive power can be limited at 80W
- Axial velocity: 90 deg/s
- Encoder frequency: 1.000 Hz

Tabla de límites rotacionales

Número de Junta	Denominación	Límite Inferior	Límite Superior
1	arm_joint_1	0.0100692	5.84014
2	arm_joint_2	0.0100692	2.61799
3	arm_joint_3	-5.02655	-0.015708
4	arm_joint_4	0.0221239	3.4292
5	arm_joint_5	0.110619	5.64159

Anexo 2: Características de motores de las articulaciones del KUKAyouBot

EC 45 flat ∅42.9 mm, brushless, 30 Watt



Stock program
Standard program
Special program (on request)

		Article Numbers							
	A with Hall sensors	200142		339281		339282			
	B sensorless		200189		339283		339284		
Motor Data									
	Values at nominal voltage								
1	Nominal voltage	V	12	12	24	24	36	36	
2	No load speed	rpm	4380	4370	4380	4380	4760	4760	
3	No load current	mA	146	146	73	73	55.4	55.3	
4	Nominal speed	rpm	2940	2850	2940	2910	3290	3270	
5	Nominal torque (max. continuous torque)	mNm	55.5	53.2	55.3	54.7	66.6	66.1	
6	Nominal current (max. continuous current)	A	2.03	1.96	1.01	1	0.849	0.844	
7	Stall torque	mNm	241	206	239	230	337	330	
8	Starting current	A	10	8.58	4.97	4.77	5.38	5.22	
9	Max. efficiency	%	78	76	78	77	81	81	
	Characteristics								
10	Terminal resistance phase to phase	Ω	1.2	1.4	4.83	5.03	6.69	6.89	
11	Terminal inductance phase to phase	mH	0.56	0.56	2.24	2.24	4.29	4.29	
12	Torque constant	mNm/A	25.5	25.5	51	51	70.6	70.6	
13	Speed constant	rpm/V	374	374	187	187	135	135	
14	Speed/torque gradient	rpm/mNm	17.6	20.5	17.7	18.5	12.8	13.2	
15	Mechanical time constant	ms	17.1	19.9	17.2	17.9	12.4	12.8	
16	Rotor inertia	gcm²	92.5	92.5	92.5	92.5	92.5	92.5	

Specifications

	Thermal data	
17	Thermal resistance housing-ambient	5.7 K/W
18	Thermal resistance winding-housing	3.96 K/W
19	Thermal time constant winding	11.5 s
20	Thermal time constant motor	251 s
21	Ambient temperature	-40…+100°C
22	Max. permissible winding temperature	+125°C
	Mechanical data (preloaded ball bearings)	
23	Max. permissible speed	10000 rpm
24	Axial play at axial load < 5.0 N	0 mm
	> 5.0 N	typ. 0.14 mm
25	Radial play	preloaded
26	Max. axial load (dynamic)	4.8 N
27	Max. force for press fits (static)	53 N
	(static, shaft supported)	1000 N
28	Max. radial loading, 7.5 mm from flange	21 N
	Other specifications	
29	Number of pole pairs	8
30	Number of phases	3
31	Weight of motor	75 g

Values listed in the table are nominal.

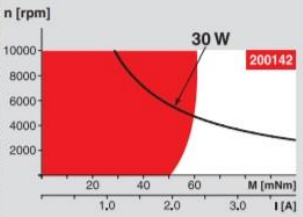
Connection	with Hall sensors	sensorless
Pin 1	V_{Hall} 4.5…18 VDC	Motor winding 1
Pin 2	Hall sensor 3*	Motor winding 2
Pin 3	Hall sensor 1*	Motor winding 3
Pin 4	Hall sensor 2*	⅄ neutral point
Pin 5	GND	
Pin 6	Motor winding 3	
Pin 7	Motor winding 2	
Pin 8	Motor winding 1	

*Internal pull-up (7…13 kΩ) on pin 1
Wiring diagram for Hall sensors see p. 29

Adapter	Article number	Article number
see p. 321	220300	220310
Connector	**Article number**	**Article number**
Tyco	1-84953-1	84953-4
Molex	52207-1185	52207-0485
Molex	52089-1119	52089-0419

Pin for design with Hall sensors:
FPC, 11-pol, Pitch 1.0 mm, top contact style

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 16 - 21

Planetary Gearhead
∅42 mm
3 - 15 Nm
Page 243

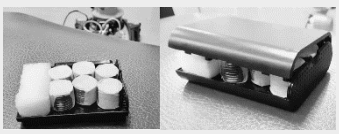
Spur Gearhead
∅45 mm
0.5 - 2.0 Nm
Page 244

Recommended Electronics:

ESCON 50/5	Page 292
DECS 50/5	297
DEC Module 24/2	298
DEC 24/3	298
DEC Module 50/5	299
EPOS2 24/2, Module 36/2	312
EPOS2 24/5	313
EPOS2 P 24/5	316
EPOS3 70/10 EtherCAT	319
Notes	**20**

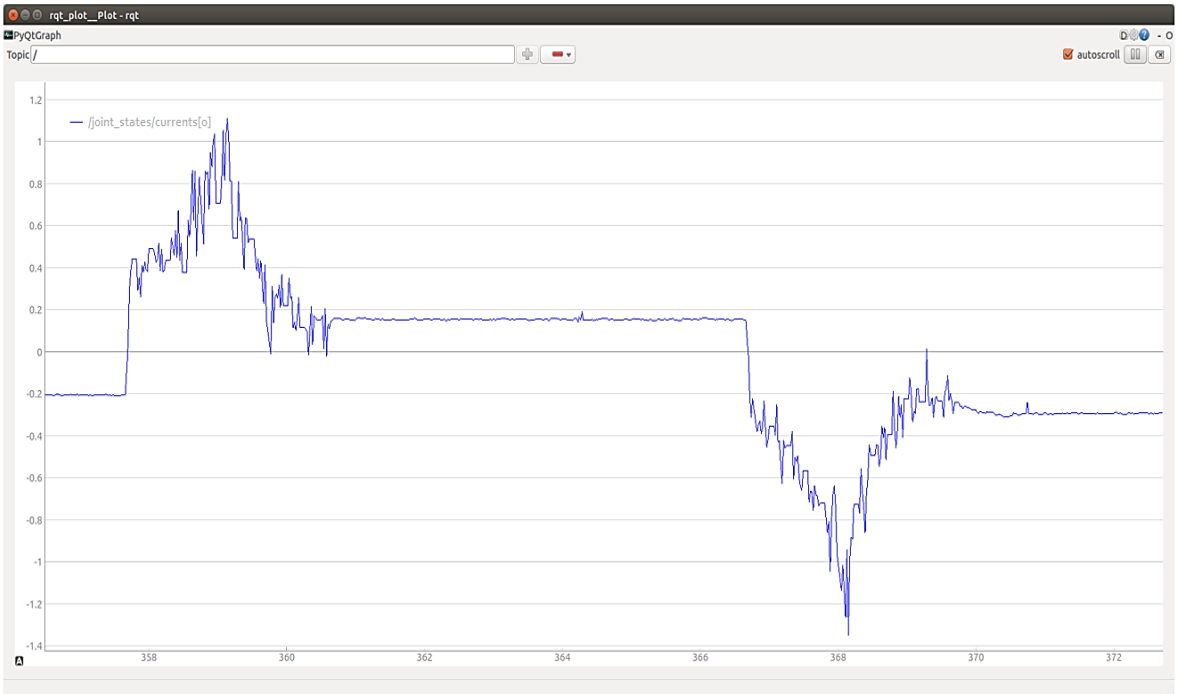
Option
With Cable and Connector
(Motor length +1.3 mm,
Ambient temperature -20…+100°C)

Anexo 3: Pesos establecidos para ejecución de la secuencia de movimientos

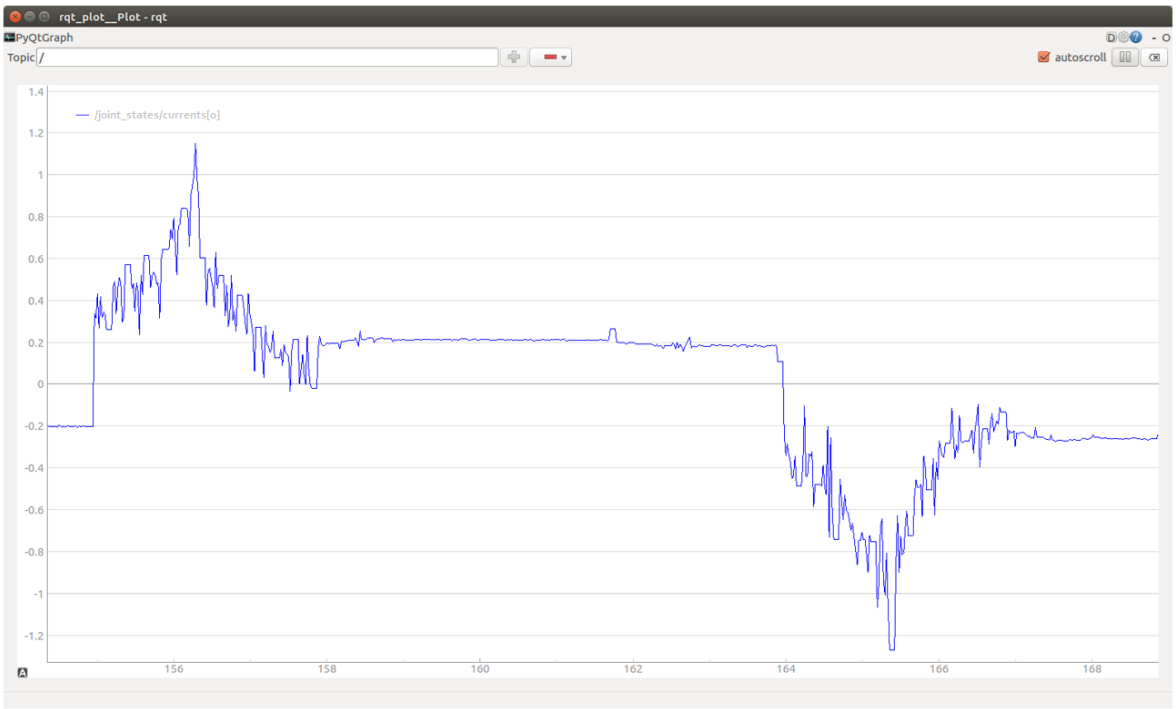
Descripción	Peso	Foto
Contenedor de pesos	30 gramos	
1° peso	100 gramos	
2° peso	200 gramos	
3° peso	300 gramos	
Ilustración	KUKAyouBot con 330 gramos 	

Anexo 4: Articulación 1, secuencia sin peso

Anexo 5: Articulación 1, secuencia con 130 gramos

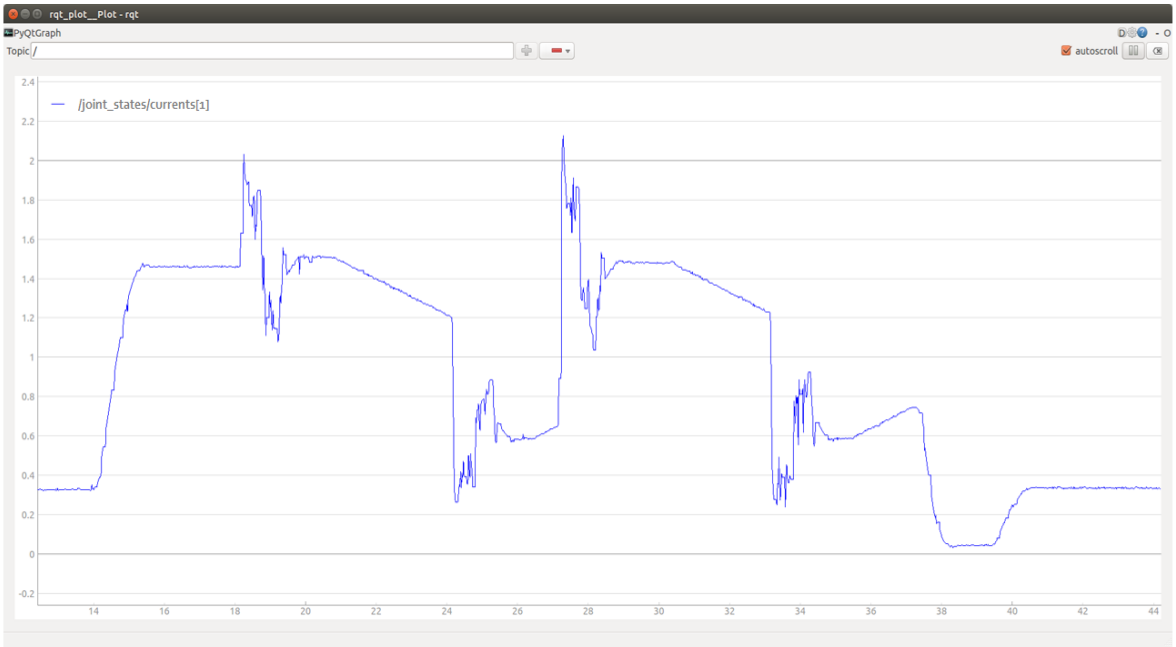


Anexo 6: Articulación 1, secuencia con 230 gramos



Anexo 7: Articulación 1, secuencia con 330 gramos

Anexo 8: Articulación 2, secuencia sin peso

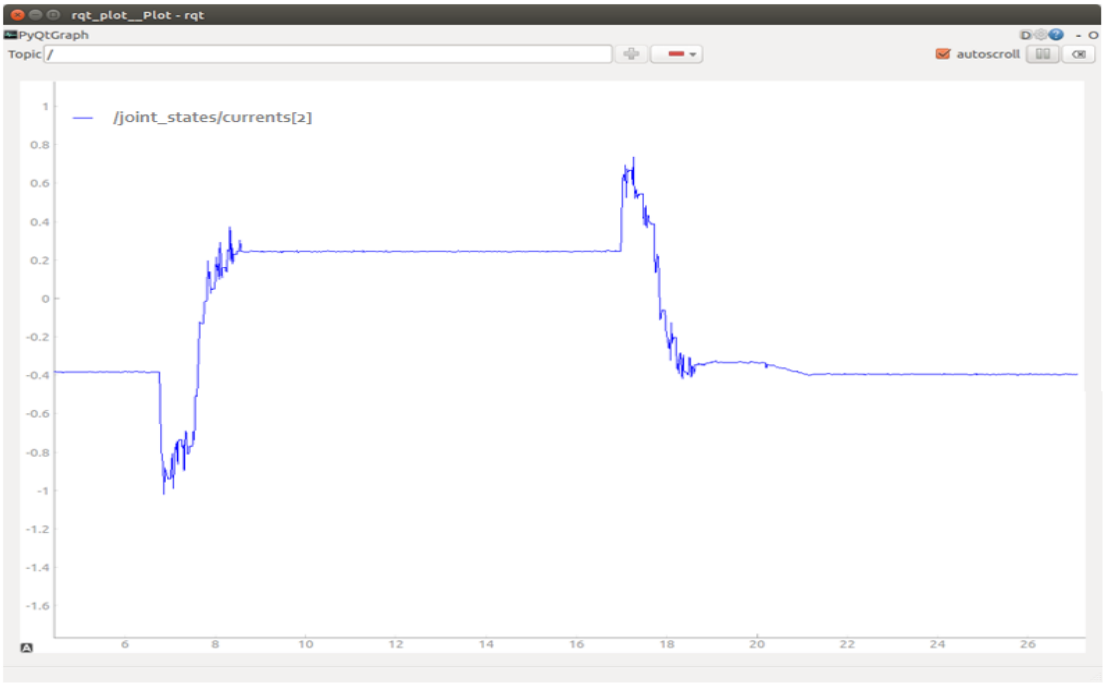


Anexo 9: Articulación 2, secuencia con 130 gramos

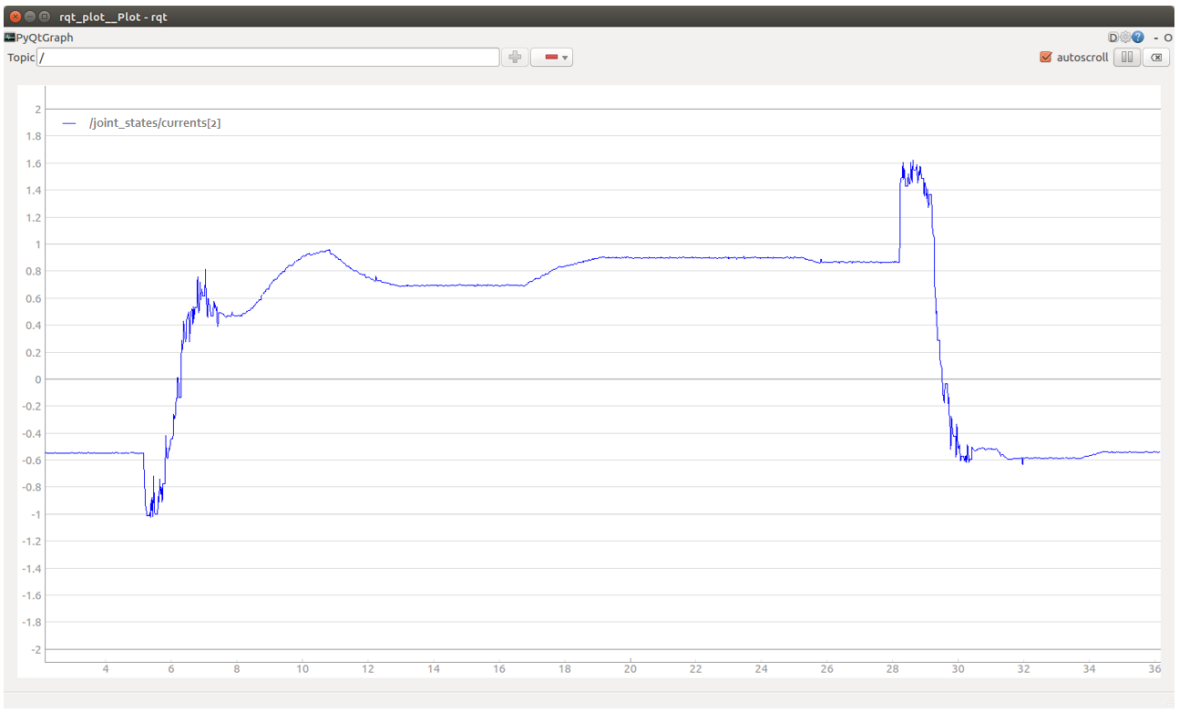
Anexo 10: Articulación 2, secuencia con 230 gramos

Anexo 11: Articulación 2, secuencia con 330 gramos

Anexo 12: Articulación 3, secuencia sin peso

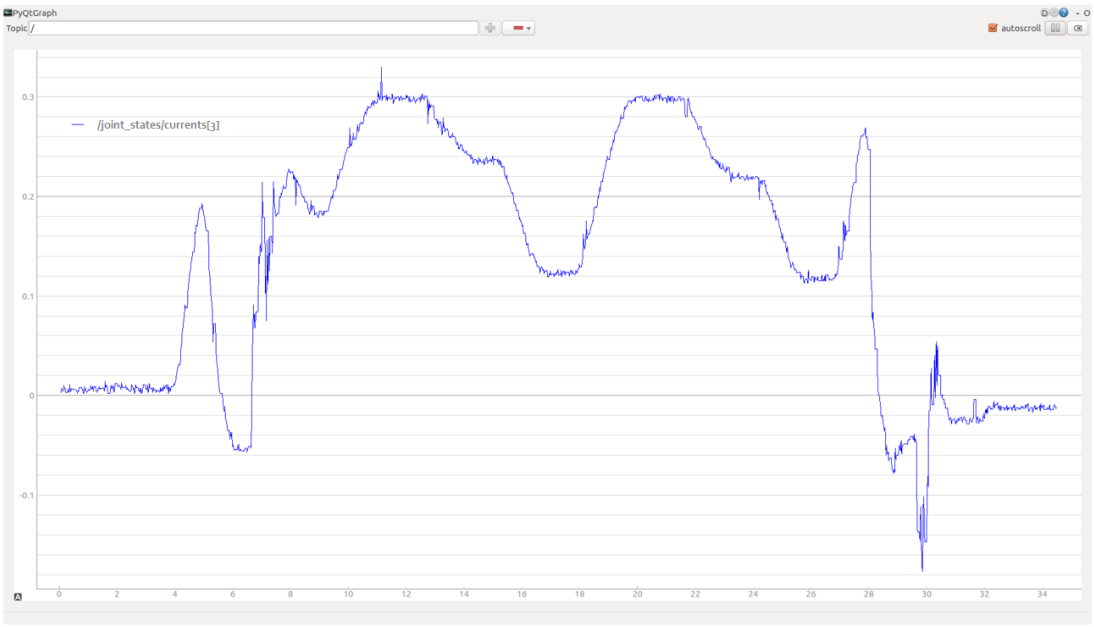


Anexo 13: Articulación 3, secuencia con 230 gramos

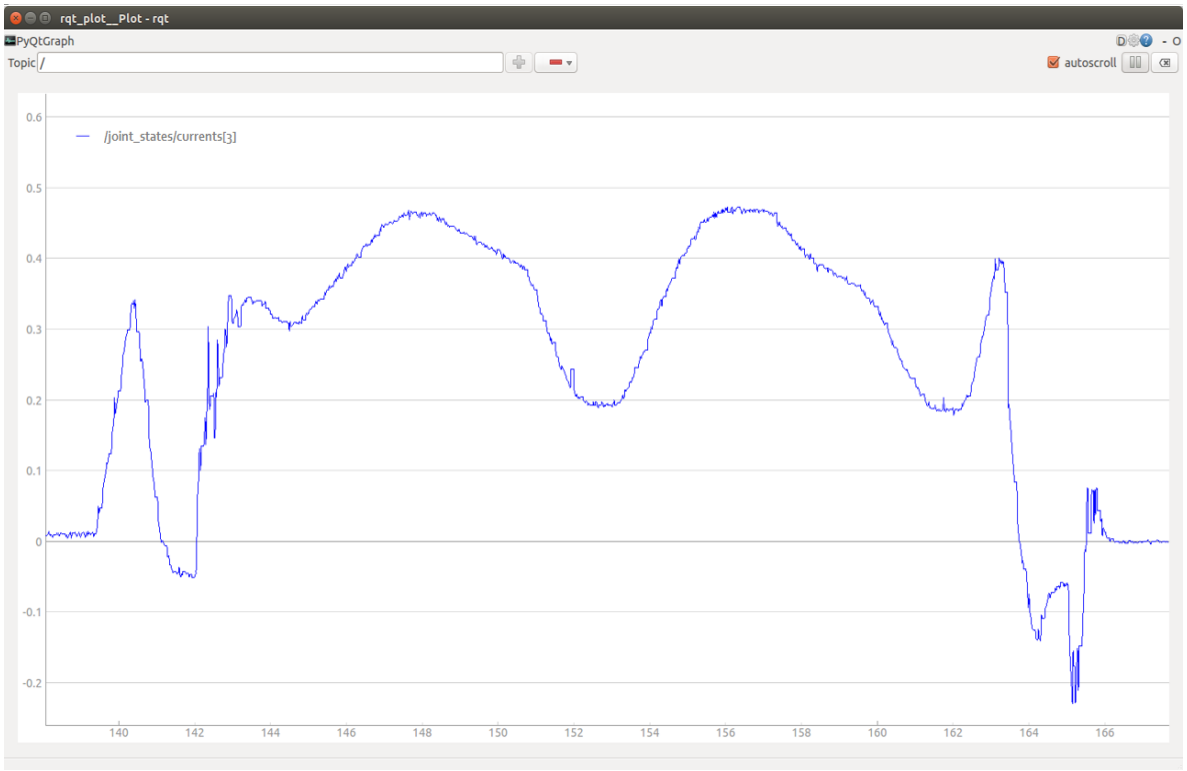


Anexo 14: Articulación 3, secuencia con 330 gramos

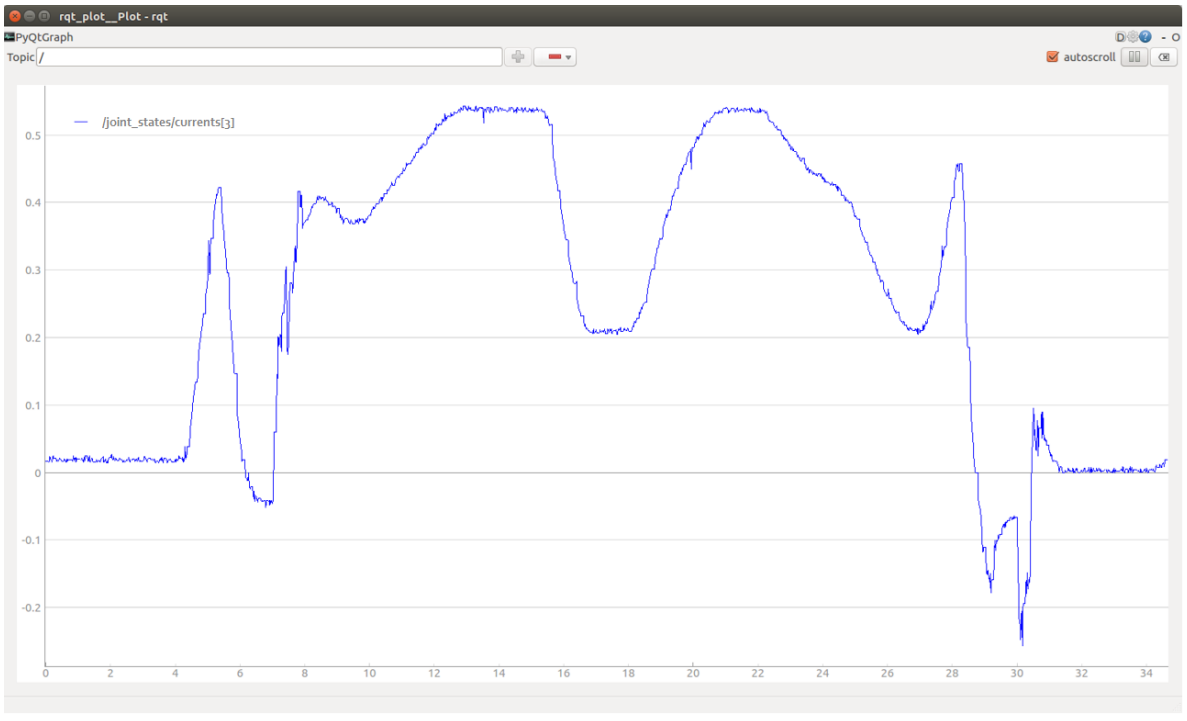
Anexo 15: Articulación 4, secuencia sin peso



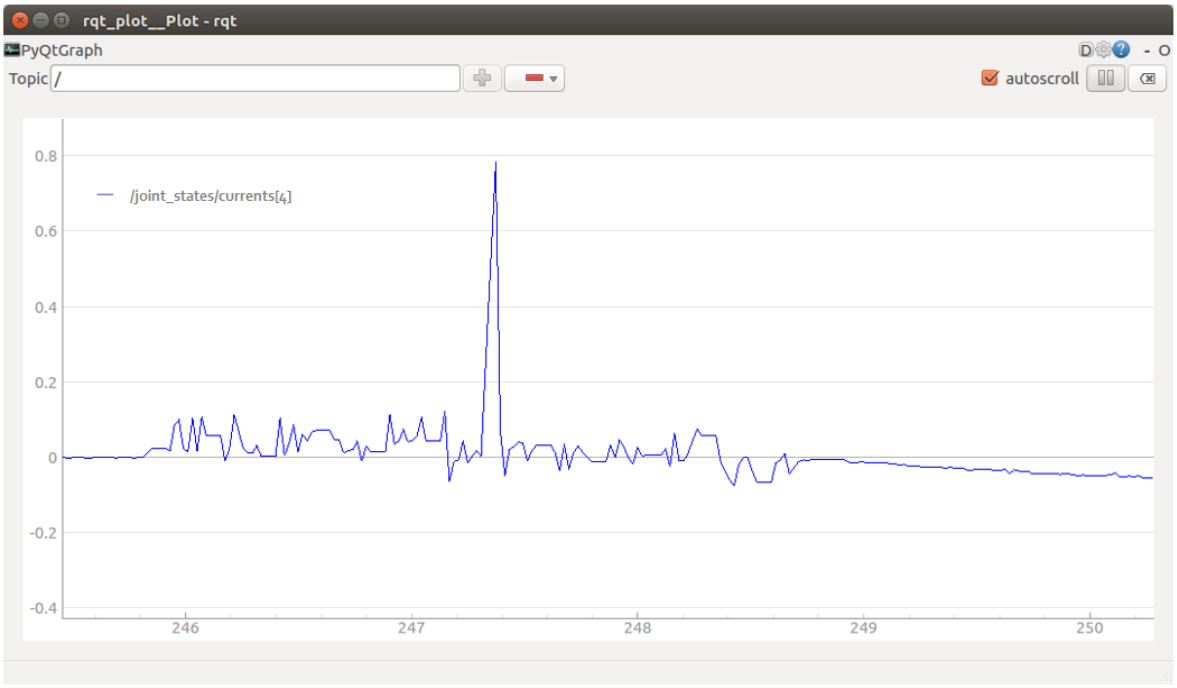
Anexo 16: Articulación 4, secuencia con 230 gramos



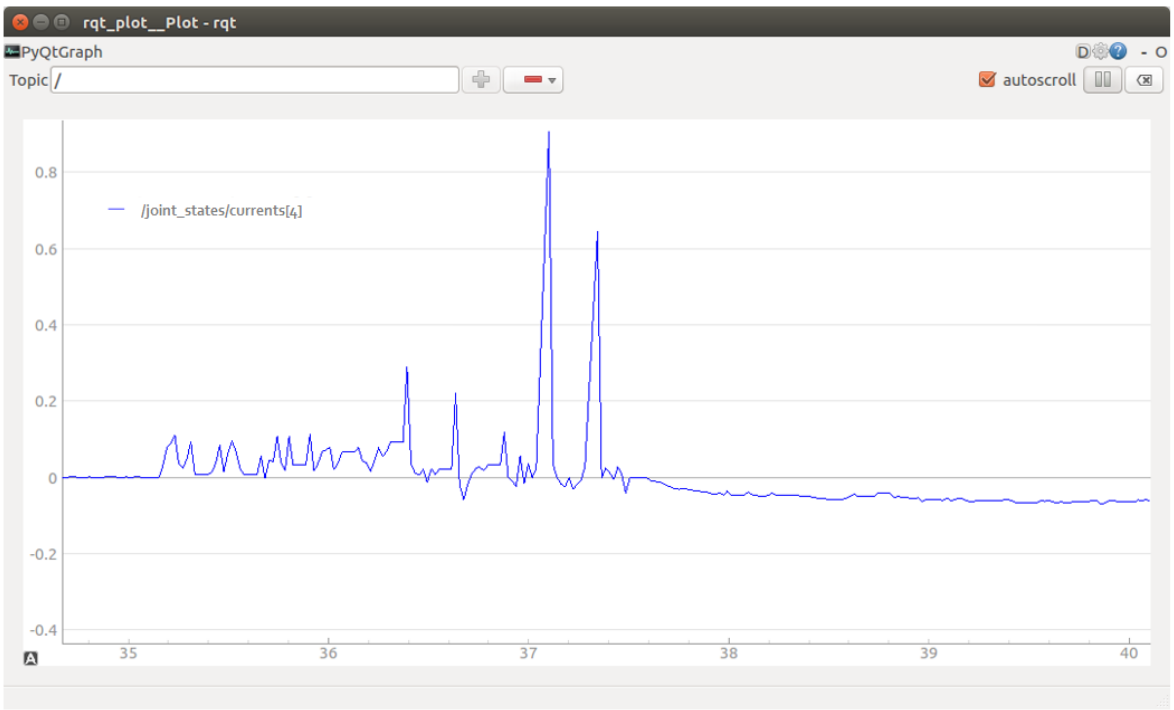
Anexo 17: Articulación 4, secuencia con 330 gramos



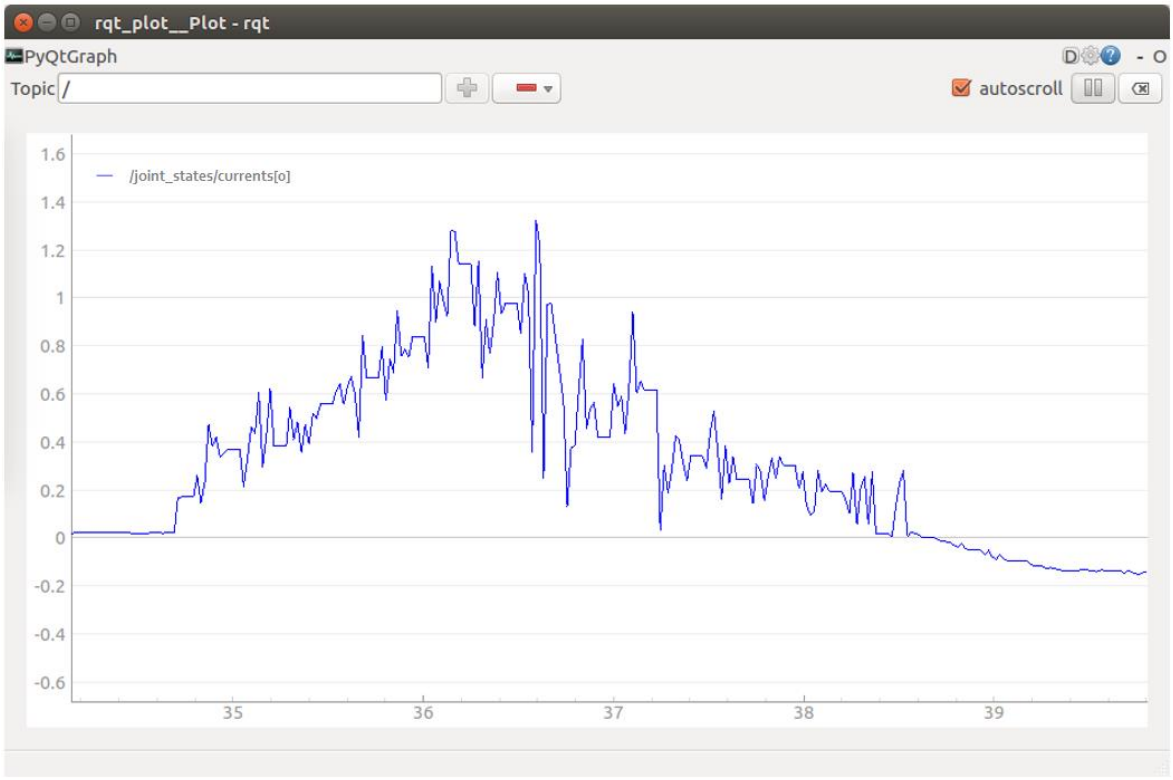
Anexo 18: Articulación 5, de 0 a 3 radianes



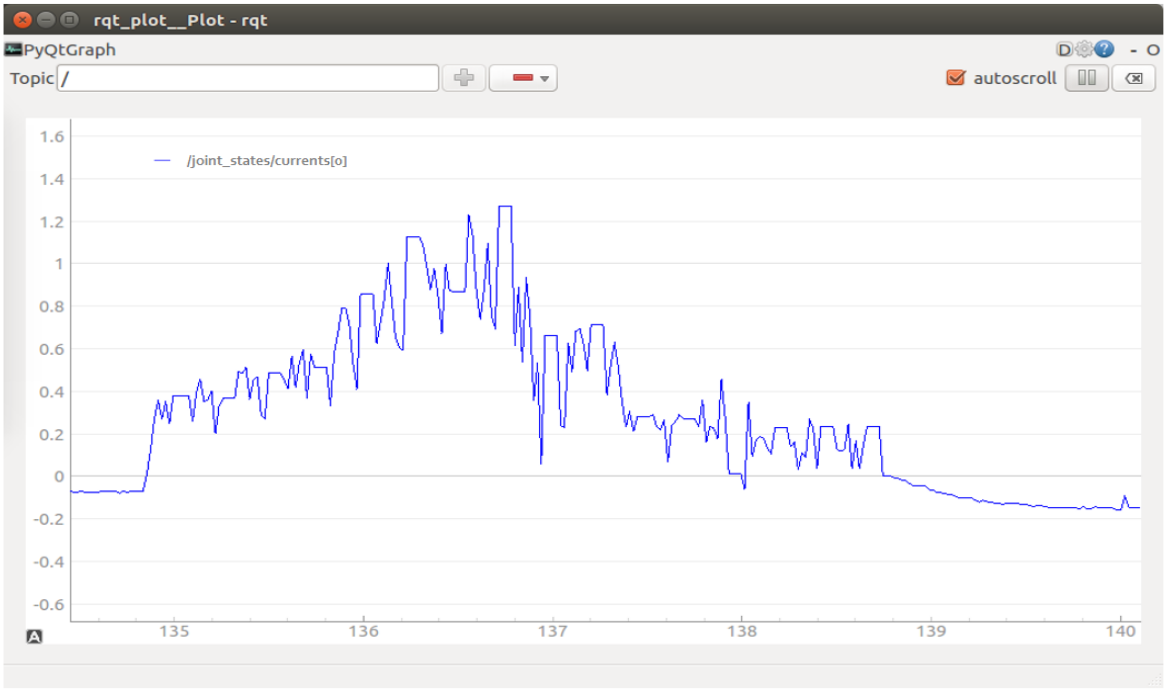
Anexo 19: Articulación 5, de 0 a 3 radianes con 330 gramos



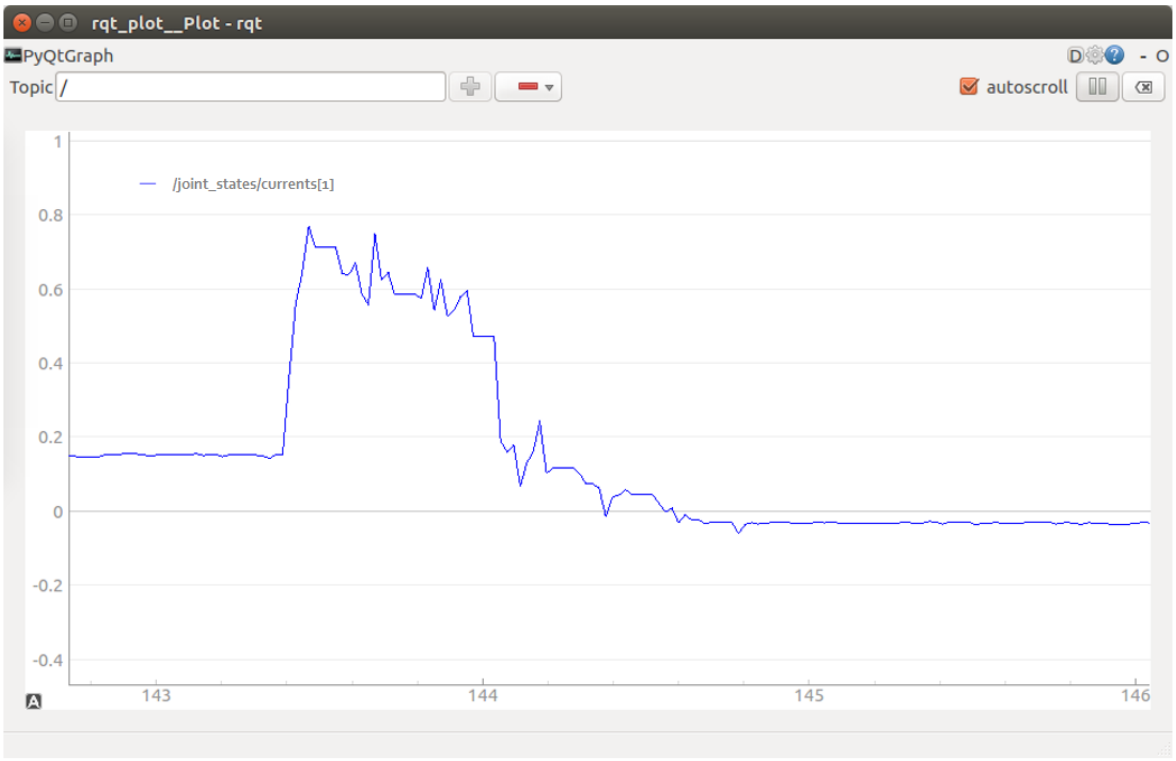
Anexo 20: Articulación 2, de 0 a 0.5 radianes sin peso



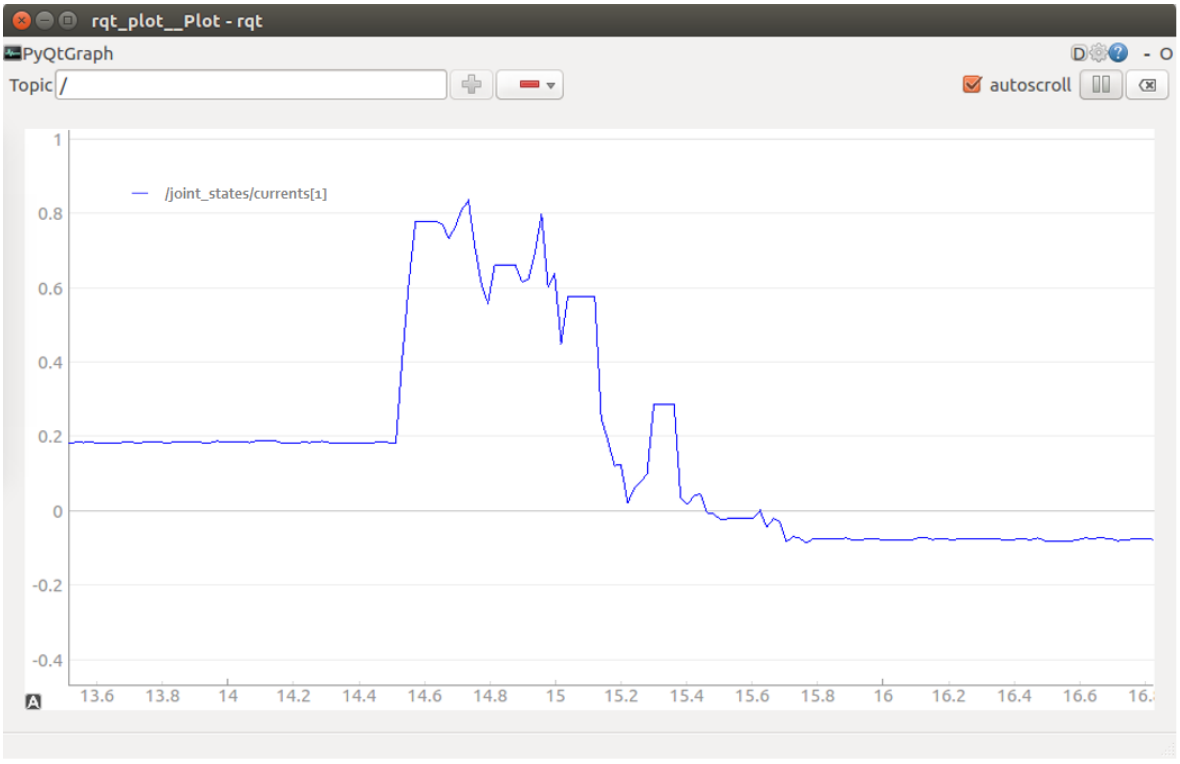
Anexo 21: Articulación 2, de 0 a 0.5 con 330 gramos



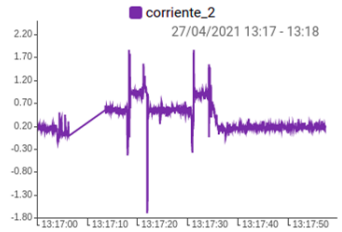
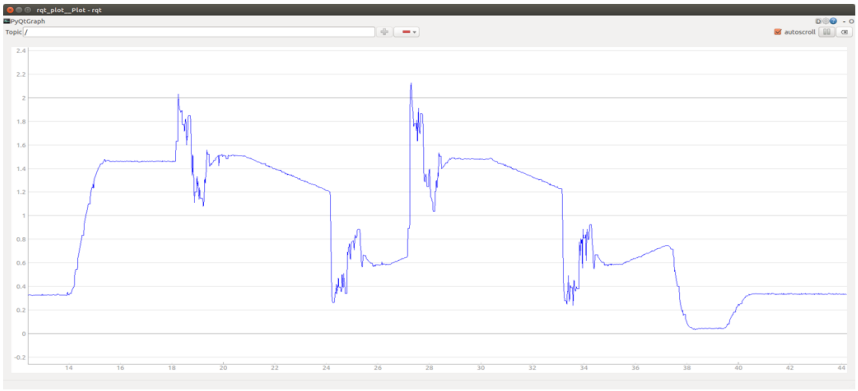
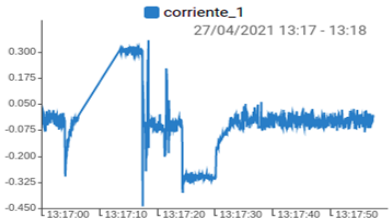
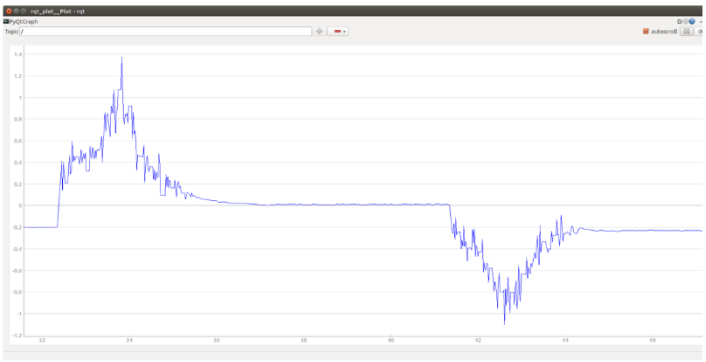
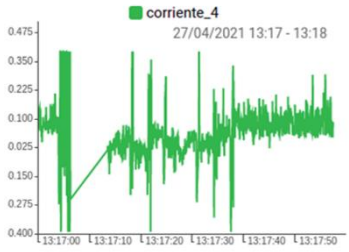
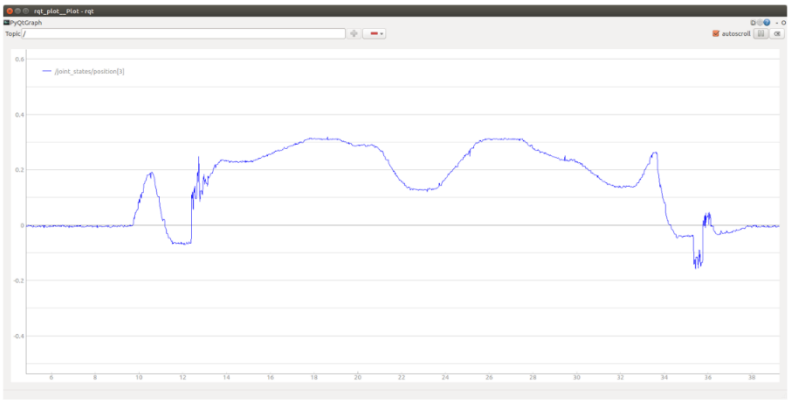
Anexo 22: Articulación 2, de 0 a 0.5 radianes sin peso



Anexo 23: Articulación 2, de 0 a 0.5 radianes con 330 gramos

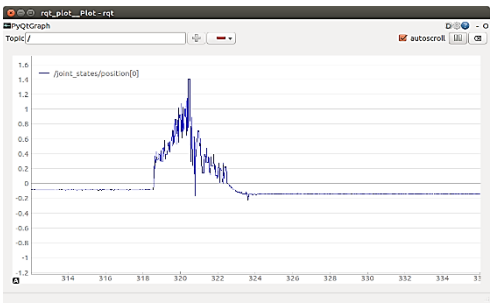
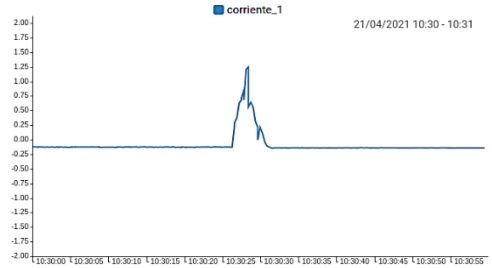
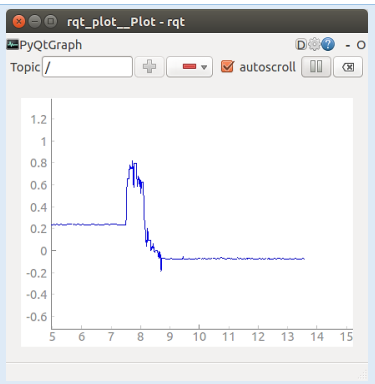
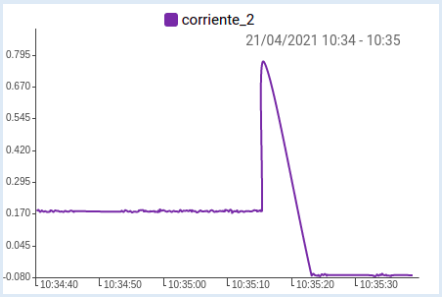
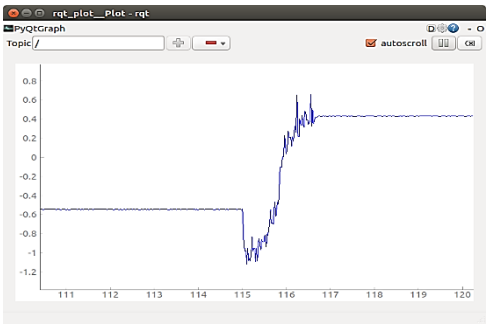
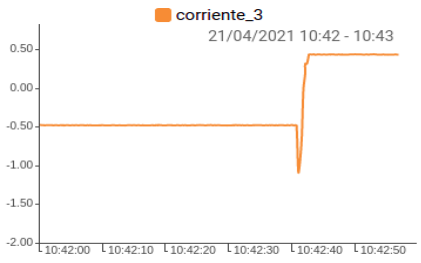


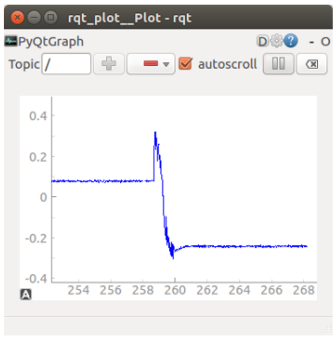
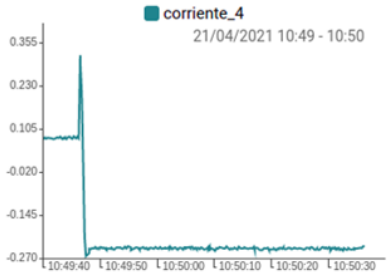
Anexo 24: Comparación entre simulación y entorno real

<table>
<tr><td colspan="2">Secuencia para la articulación 2</td></tr>
<tr><td>Simulación</td><td>Rqt_plot</td></tr>
<tr><td> 
</td><td>  </td></tr>
<tr><td colspan="2">Secuencia para la articulación 2</td></tr>
<tr><td>Simulación</td><td>Rqt_plot</td></tr>
<tr><td> 
</td><td>  </td></tr>
<tr><td colspan="2">Secuencia paara la articulación 4</td></tr>
<tr><td>Simulación</td><td>Rqt_plot</td></tr>
<tr><td> 
</td><td>  </td></tr>
</table>

Anexo 25: Análisis de datos con herramienta Rqt_plot e Interfaz Gráfica.

Tabla 15: Comparación de herramienta Rqt_plot con Interfaz gráfica

Articulación 1, de 0 a 5 radianes con 330 gramos	
Rqt_plot	Interfaz AnyViz
	
Articulación 2, de 0 a 0.5 radianes con 330 gramos	
Rqt_plot	Interfaz AnyViz
	
Articulación 3, de 0 a -1.5 radianes con 330 gramos	
Rqt_plot	Interfaz AnyViz
	

Articulación 4, de 0 a 1.2 radianes con 330 gramos	
Rqt_plot	Interfaz AnyViz
	
Articulación 5, de 0 a 3 radianes con 330 gramos	
Rqt_plot	Interfaz AnyViz