UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y COMUNICACIONES

Tema:

SISTEMA DE CONTROL PARA LA TELEOPERACIÓN DEL ROBOT KUKA YOUBOT MEDIANTE COMUNICACIONES INALÁMBRICAS Y EL USO DE TARJETAS EMBEBIDAS

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Electrónica y Comunicaciones

ÁREA: Física y Electrónica
LÍNEA DE INVESTIGACIÓN: Sistemas Electrónicos
AUTOR: Alexis Geovanny Peñaloza Leguisamo
TUTOR: Ing. Franklin Salazar Logroño Mg.

Ambato - Ecuador
septiembre – 2021

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: SISTEMA DE CONTROL PARA LA TELEOPERACIÓN DEL ROBOT KUKA YOUBOT MEDIANTE COMUNICACIONES INALÁMBRICAS Y EL USO DE TARJETAS EMBEBIDAS, desarrollado bajo la modalidad Proyecto de Investigación por el señor Alexis Geovanny Peñaloza Leguisamo, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, 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, septiembre 2021.

Firmado electrónicamente por:
FRANKLIN WILFRIDO SALAZAR LOGRONO

Ing. Franklin Salazar Logroño Mg.

TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: SISTEMA DE CONTROL PARA LA TELEOPERACIÓN DEL ROBOT KUKA YOUBOT MEDIANTE COMUNICACIONES INALÁMBRICAS Y EL USO DE TARJETAS EMBEBIDAS 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, septiembre 2021.

Alexis Geovanny Peñaloza Leguisamo

C.C. 1500750201

AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Alexis Geovanny Peñaloza Leguisamo, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE CONTROL PARA LA TELEOPERACIÓN DEL ROBOT KUKA YOUBOT MEDIANTE COMUNICACIONES INALÁMBRICAS Y EL USO DE TARJETAS EMBEBIDAS, 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, septiembre 2021.

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Pilar Urrutia, Mg.
PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
ANA PAMELA CASTRO

Ing. Pamela Castro Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
CARLOS DIEGO GORDON GALLEGOS

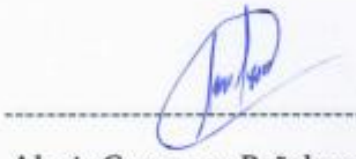
PhD. Carlos Gordón Gallegos
PROFESOR CALIFICADOR

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, septiembre 2021.



Alexis Geovanny Peñaloza Leguisamo

C.C. 1500750201

AUTOR

DEDICATORIA

A mis Padres, debido que gracias a todas sus enseñanzas y apoyo incondicional lograron construir un espíritu motivado para avanzar cada día un poco y lograr cumplir mis metas planteadas. Gracias por el gran ejemplo a seguir que son. LOS AMO.

Alexis Geovanny Peñaloza Leguisamo

AGRADECIMIENTO

Agradezco a mi esposa Jessenia Garrido por su amor, cariño y apoyo incondicional para poder seguir adelante dándome ánimos ayudándome a superar todos mis tropiezos a lo largo de la carrera.

.

Alexis Geovanny Peñaloza Leguisamo

ÍNDICE GENERAL

Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

Debido al alto costo que representa comprar accesorios para el control de maquinaria dentro de la industria es indispensable el conocimiento necesario para poder resolver este tipo de inconvenientes por el bienestar de cualquier empresa. El presente proyecto de investigación tiene como objetivo la creación de un sistema de control para la teleoperación del robot Kuka Youbot con el uso de hardware y software libre debido a que las aplicaciones actuales con el robot en su mayoría son con secuencias previamente programadas, limitando así un uso continuo e independiente en tiempo real si es requerido, dentro del proyecto se explica el diseño de un circuito electrónico creado en EasyEDA con la utilización del microcontrolador ESP32-WROOM-32, el cual es un módulo dirigido a una amplia variedad de aplicaciones debido a su comunicación inalámbrica Wi-Fi y Bluetooth, además de doble núcleo de CPU los cuales pueden ser controlados individualmente. Aprovechando que este microcontrolador se puede programar con una variedad de lenguajes se usó para la creación de una comunicación inalámbrica con el Kuka Youbot usando el protocolo TCP y el empleo de sockets en cada extremo. Para poder recibir la información del control remoto se creó un nodo capaz de recibir datos por medio del protocolo TCP/IP y publicar continuamente mensajes al nodo del robot con el tema respectivo según sean las necesidades del operador. El sistema fue creado y puesto a prueba con un robot Kuka Youbot real y otro dentro del simulador Gazebo con un sistema operativo Ubuntu 16.04 y ROS Kinetic.

Este sistema para controlar el Kuka Youbot beneficia de forma directa al área de investigación de la Universidad Técnica de Ambato, así como a sus estudiantes y docentes por la entrega de un circuito PCB con licencia abierta a modificaciones según aparezcan nuevas necesidades, además hace un gran aporte al entendimiento de la robótica.

Palabras claves: Kuka, Youbot, WiFi, ESP32, socket, python, ROS, control inalámbrico

ABSTRACT

Due to the high cost of buying accessories for the control of machinery in the industry, the necessary knowledge is essential to be able to solve this type of problems for the wellness of any company. The main objective of this research project is the creation of a control system for the teleoperation of the Kuka Youbot robot with the use of free hardware and software because the current applications with the robot are mostly with previously programmed sequences, limiting a continuous and independent use in real time if required, inside the project the design of an electronic circuit is explained which was created in EasyEDA with the use of the ESP32WROOM32 microcontroller, which is a module focused on a wide variety of applications with its Wi-Fi and Bluetooth wireless communication, taking advantage of the fact that this microcontroller can be programmed with a variety of languages, it was used to create a wireless communication with the Kuka Youbot using the TCP protocol and the use of sockets at each end. To receive the information from the remote control, a node was created with the ability to receive data through the TCP / IP protocol and continuously publish messages to the robot node with the respective topic according to the needs of the operator. The system was created and tested with a real Kuka Youbot robot and another inside the Gazebo simulator with an Ubuntu 16.04 operating system and ROS Kinetic.

This system to control the Kuka Youbot directly benefits the research area of the Technical University of Ambato, as well as its students and teachers by delivering a PCB circuit with a license open to modifications, it also makes a great contribution to the understanding of the robotics.

Keywords: Kuka, Youbot, Sockets, Wireless control, WiFi, Python, ROS, ESP32

CAPÍTULO I

MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA DE CONTROL PARA LA TELEOPERACIÓN DEL ROBOT KUKA YOUBOT MEDIANTE COMUNICACIONES INALÁMBRICAS Y EL USO DE TARJETAS EMBEBIDAS

1.2 Antecedentes investigativos

La automatización de procesos va creciendo con rapidez para dar solución al mejoramiento y optimización de la producción, aumentando la eficiencia sin interrupciones por errores humanos. La instalación de sensores o actuadores inalámbricos dentro de una industria puede reducir costos significativos, la tecnología inalámbrica presenta superioridad a la tecnología cableada al momento de requerir cambio de posicionamiento de los equipos o la necesidad de pruebas como dispositivos portátiles. En Ecuador y alrededor del mundo se han realizado múltiples trabajos de investigación relacionados con la robótica de forma general y proyectos específicos para robots Kuka Youbot, los mismos se detallan a continuación.

En Ecuador el 2018, la Universidad Técnica de Ambato se presenta la investigación "Autonomous Robot KUKA YouBot Navigation Based on Path Planning and Traffic Signals Recognition" de Henry Lema, Carlos Gordón, Cristian Peñaherrera, Patricio Encalada y Diego León. Entran en el desarrollo y planificación de rutas para un robot omnidireccional Kuka Youbot bajo el reconocimiento de posturas humanas mediante la integración de los ambientes de trabajo ROS, Matlab y OpenCV, teniendo como resultado una detección perfecta con un tiempo aproximado a 0.2 segundos.

[1]

En Colombia el 2016, LA Universidad Pedagógica y Tecnológica presentó el estudio "Análisis de elementos en zona local y remota para la teleoperación del brazo robótico AL5A" por María Pinto y Jhon Montañez. Un proyecto en el cual implementan un modelamiento sistemático integrado con herramientas computacionales para comparar el desempeño de tres tipos de dispositivos maestros para el control de un brazo robótico

AL5A; joystick de video juegos, un teléfono móvil y la interfaz haptica Novint Falcon. Teniendo con un mejor nivel de inmersiñon a la interfaz Novint Falcon, por el efecto de un movimiento tridimensional desde la estación de teleoperación.

[2]

En Bolivia el 2014, la Universidad Pontificia Bolivariana presenta el trabajo de grado "Implementación de una Aplicación Móvil de Teleoperación bajo la Plataforma Android para Controlar un Robot Industrial" de José R. Martínez, Juan C. Yepes, Juan J. Yepes. Mediante la comunicación TCP y la creación de una aplicación para Android en java el proyecto explica el control maestro esclavo del robot Kuka KR6 por medio de un servidor TCP para mandar comandos a la controladora KR C2 y tener el control del robot con un smartphone conectado a wifi, el microcontrolador y el controlador Kuka KR-C2dando cuentan con una comunicación con retardo en la transferencia de datos. [3]

En Ecuador el 2019, La universidad Técnica de Ambato presentó la investigación "Diseño de sistemas de control industrial de robots basados en industria 4.0" teniendo como autores a Juan Camilo Escobar Naranjo y Dr.Mg. Marcelo Vladimir García Sánchez. Se creó el control del brazo robótico Kuka Youbot desde un ordenador enviando datos mediante wifi a un controlador ejecutándose sobre tarjeta Raspberry Pi, el mismo que reenvía de mensaje desde el terminal de comandos ROS, mediante la construcción de plugins que es el encargado de la comunicación entre el robot y la terminal de comandos. [4]

En España el 2012, en la Universidad Politécnica de Valencia presentó la investigación "Programación de un robot industrial Scara mediante un dispositivo móvil" teniendo como autores a Ángel Fernández Cañada Vilata y Martín Mellado Arteche. Desarrollaron una comunicación mediante bluetooth de un smartphone para controlar el Robot Scara de Kuka el mismo que se comunica a un PC y este reenvía las órdenes a la controladora propia del robot Scara para el control de posición y velocidad de movimiento, el sistema es ineficiente debido a que tiene una respuesta lenta en comparación a cuando el robot es manejado por un humano directamente.

[5]

En vista de todo lo analizado previamente se creó un sistema de control para la teleoperación del robot Kuka Youbot con ayuda del microcontrolador ESP-WROOM-32 el cual es un módulo con comunicación inalámbrica wifi y bluetooth facilitando la comunicación inalámbrica mediante el protocolo TCP y la creación de socket en los extremos de la comunicación, este MCU cuenta con software y hardware libre de bajo costo para realizar diferentes tipos de aplicaciones y asegurar con el perfecto funcionamiento del Kuka Youbot.

1.3 Contextualización del problema

A nivel mundial, el progreso tecnológico de las últimas décadas ha crecido de forma acelerada, llegando del uso de grandes computadores de escritorio hasta la era digital, se han convertido en productos muy pequeños con características del procesamiento de datos en el campo de la robótica, la automatización busca optimizar y aumentar la eficiencia en el proceso de los negocios. [6]

La combinación de la informática, electrónica y mecánica nos llevan a los sistemas autónomos inteligentes de múltiples diciplinas para realizar tareas sencillas como limpiar suelos, transportar materiales y servicios hospitalarios incluso a actividades de alto riesgo, como manipular químicos peligroso o exploración del espacio estelar. La gran parte de empresas dedicadas a estas actividades cuentan con maquinaría de código cerrado lo cual impide a los investigadores utilizar todas las características del equipo adquirido. [7]

Para construir un robot es necesario dominar los campos de la electrónica, mecánica, eléctrica, comunicaciones y programación para construir estructuras, configurar animaciones, establecer interfaces de comunicación y elaborar un software para ejecutar rutinas que permitan mantener un enlace entre hombre y máquina. El campo de la educación es semejante a la industrial, debido a los altos costos de adquisición en robots para la investigación y desarrollo de la automatización, debido a que el costo de requisición de los robots utilizados en la automatización es alto, esto se debe al software y hardware controlado y diseñado por el fabricante, impulsando hacer programas de bajo rendimiento, obstaculizando la programación directa de rutinas en el controlador del propietario [8] [9].

En el 2019 se produjo el incendio de la catedral de Notre Dame en el cual se pudo apreciar al robot COLOSSUS, un robot bombero equipado con un cañón de agua y manipulado a control remoto para proteger la vida de los bomberos evitando el contacto cercano a las llamas. En la actualidad la gran parte de los robots industriales cuentan con una manipulación mediante una previa codificación de rutinas en el controlador dificultando así la manipulación en tiempo real en caso de necesitarlo, debido que para poder controlar los diferentes autómatas de forma inalámbrica en tiempo real se necesita adquirir equipos directamente a la marca registrada del mismo, aumentado drásticamente el costo para la empresa o institución que adquirió el robot.

Actualmente en LATAM (América Latina) inicio a usar con mayor reiteración sistemas robóticos con la finalidad de mejorar la industria, adquiriendo exactitud y velocidad en la producción para lo cual es necesario tener personal capacitado antes de la compra de los equipos para enfrentar cualquier daño, sabiendo solucionar de una manera eficaz que represente un bajo costo para la empresa.

En la Universidad Técnica de Ambato se cuenta por el momento con tres equipos robot Kuka Youbot para la desarrollo e investigación de la automatización. La gran parte de las aplicaciones realizadas con Kuka Youbot provienen de investigaciones con previa programación en el ordenador del robot, donde se debe cargar instrucciones para la planificación de rutas desde la API de Youbot, con el envío de datos al nodo correspondiente según el tema con el cual opere la plataforma y al brazo Kuka, limitando un control preciso y rápido adaptándose a las necesidades del operador en caso de requerir una manipulación continua a distancia, esto limita al programador en crear aplicaciones con integración de nuevas tecnologías de comunicación, forzando a crear programas con bajo rendimiento con tiempos de respuesta lentos por las diferentes interfaces de comunicación debido que se requiere la utilización de software y hardware controlado y diseñado por el fabricante. [8] [9] Para mejorar la manipulación del robot Kuka Youbot se ha puesto en marca esta investigación la cual tiene como finalidad de controlar al robot a través de la navegación controlada por comunicaciones inalámbricas desarrollando un sistema de control inalámbrico para manipular del robot Kuka Youbot beneficiando directamente a los docentes, investigadores y estudiantes de la Universidad Técnica de Ambato, proporcionando un diseño PCB de licencia abierta que puede ser modificado los códigos programable de

operación, originando una sociedad capaz de apoyar el incremento el conocimiento de la robótica.

El proyecto de investigación realizado utiliza software y hardware libre para controlar el robot Kuka Youbot, la comunicación es mediante un sistema inalámbrico wifi. Además, se adquirió los elementos electrónicos con facilidad en el mercado ecuatoriano por medio del comercio local y bajo importaciones como persona natural por medio de correos nacionales.

1.4 Fundamentación teórica

1.4.1 Historia de la robótica moderna

La ingeniería comenzó a desarrollarse a mediados del siglo XX en diferentes ramas como son: electrónica, mecánica, telecomunicaciones e informática, contribuyendo en la creación de robots. La empresa Devilviss en 1938 construyó el primer manipulador para pintura en aerosol, siendo un brazo articulado el cual fue uno de los primeros robots integrados a la industria debido a su utilidad dentro de la producción en cadena. En 1950 el término "robótica" fue utilizado por primera vez por Asimov en su libro "Yo Robot". [10]

Las investigaciones algorítmicas de los años 40 dadas por Huffman fueron de gran alcance debido que se pudo ir paralelamente con la inteligencia artificial para la robótica permitiendo el desarrollo de sistemas secuenciales. Los automatismos se formalizaron gracias al álgebra de Boole. Uno de los primeros sensores fue patentado en 1956 por Gc Devol, el cual por medio de magnetos para accionar un dispositivo mecánico registraba señales eléctricas en un controlador. [10]

Un aporte muy grande a favor de la robótica es considerado las palabras de R. Brooks y AM Flynn en 1989 la cuales cambiaron la filosofía de la construcción de robots, manifestando que deben ser en grandes cantidades, de dimensiones reducidas y económicos. [10]

1.4.2 Robótica

En la ciencia la rama que estudia el diseño y construcción de máquinas capaces de realizar tareas llevadas a cabo por humanos es llamada Robótica, estas máquinas deben ser capaces de completar dichas tareas de forma eficiente sin necesidad de estar

presente una persona para controlarla. [11]

1.4.3 Robótica de desarrollo

La robótica de desarrollo es una parte de la robótica dedicada al desarrollo de sistemas de control general, por medio de un proceso de organización autónoma, teniendo como reacción un robot capaz de continuar desarrollando aplicaciones complejas como: cognitivas, capacidades perceptuales y comportamentales. Siendo un área de investigación encargada de reunir la robótica situada, neurociencia y psicología del desarrollo. [12]

1.4.4 Robot

Un robot es una máquina que por medio de programación se puede controlar el posicionamiento y seguir trayectorias para llevar al cabo diferentes tareas, para lo cual debe contar con una unidad de memoria, actuadores y en algunos casos con sensores con la capacidad de captar el entorno para realizar tareas repetitivas con la capacidad de variar la secuencia cíclica sin afectar su estructura física. [11]

1.4.5 Robot Kuka Youbot

El robot Kuka Youbot siendo un manipulador móvil fue creado principalmente para el desarrollo de la educación e investigación por lo cual es considerado un robot de escritorio. Kuka Youbot se considera un robot muy significativo dentro de la investigación y desarrollo robótico debido haber abierto un nuevo camino con la transferencia de conocimientos tecnológicos en la comunidad de la robótica. [13]

Figura 1: Kuka Youbot *[13]*

1.4.6 Estructura Física del Kuka Youbot

El robot Kuka Youbot fue diseñado y desarrollado con la intención de simular un entorno con un robot industrial para poder entrenar al futuro personal encargado de controlar grandes empresas automatizadas. Cuenta con una configuración básica, la cual consiste en un brazo robótico con 5 ejes y una pinza con 2 dedos al final de la extremidad, este brazo puede ser transportado gracias a una plataforma móvil omnidireccional con la capacidad de acercarse hacia objetos con gran alcance como se muestra en la Figura 1. [13]

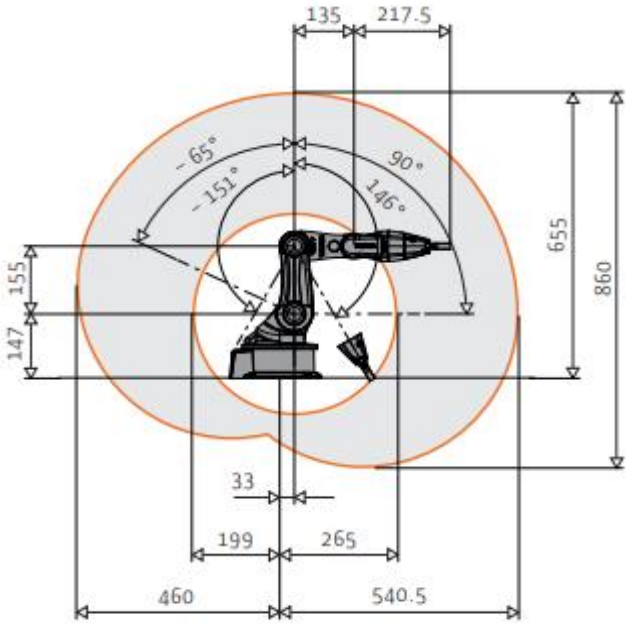


Figura 2: Movilidad del Brazo Kuka Youbot *[13]*

El brazo cuenta con un amplio rango de giro de 338° como se puede apreciar en la Figura 2. La estructura abierta del brazo robótico permite observar y aprender sobre el funcionamiento y el mecanismo interno del robot con el fin de ayudar a los estudiantes e investigadores poder avanzar en sus estudios. [13]

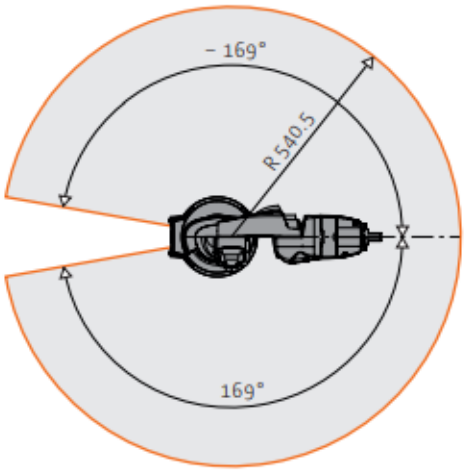


Figura 3: Giro del brazo del Kuka Youbot *[13]*

El robot cuenta con una plataforma móvil omnidireccional encargada de movilizar el brazo con la finalidad de ampliar su rango de operabilidad, dicha plataforma cuenta con llantas Mecanum especiales para un desplazamiento lateral sin necesidad de girar el plano frontal de la plataforma como se muestra en las Figuras 3 y 7. [13]

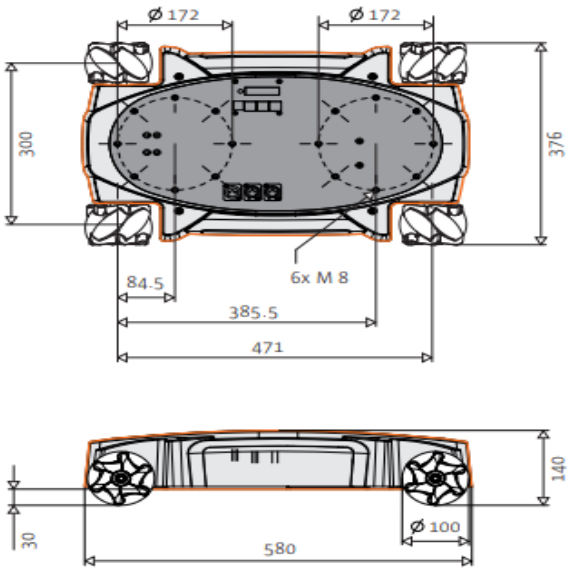


Figura 4: Plataforma Móvil del Kuka Youbot *[13]*

1.4.7 Motores del Kuka Youbot

El robot Kuka Youbot es un pequeño robot móvil de escritorio desarrollado con una plataforma de código abierto para la investigación científica, el cual cuenta con un sistema de motores brushless de MAXON MOTOR, estos motores facilitan su

operabilidad debido al reducido espacio en el interior del brazo y el chasis del robot, dichos componentes están ensamblados directamente en las articulaciones del brazo robótico. MAXON MOTOR y KUKA trabajaron conjuntamente para el desarrollo de un motor especial, particularmente ligero, preciso y de uso robusto. El brazo robótico trabaja con un total de cinco motores brushless planos de marca maxon, es motor sin escobillas "EC45 flat, EC32 flat" en combinación con reductores especiales y encoders. La plataforma trabaja con cuatro motores brushless planos sin escobillas "EC45", estos motores suministran una potencia entre 15 y 50 W, manteniendo un mínimo peso de 46 g y 110 g. [14]

Figura 5: Motores Maxon del robot Kuka Youbot *[14]*

1.4.8 Encoder del Kuka Youbot

Las articulaciones del brazo robótico cuentan con encoders que usan el mínimo espacio para permitir la medición de ángulos gracias a la interpolación, además tienen un gran número de pulsos. Maxon Encoder MR, tipo L. [14]

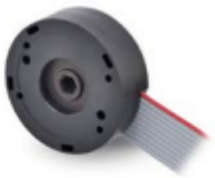


Figura 6: Maxon Encoder MR, tipo L *[14]*

1.4.9 Modelo cinemático del Kuka Youbot

El robot Kuka Youbot gracias a sus cuatro llantas tipo Mecanum la plataforma omnidireccional se puede un desplazamiento en el plano cartesiano dentro de los ejes "X" y "Y", las ruedas cuentan con rodillos colocados estratégicamente a 45º del eje principal para obtener un modelo cinemático confiable. Se puede cambiar la traslación transversal y rotación variando las velocidades lineales y sentido de giro de cada rueda de la plataforma. [15]

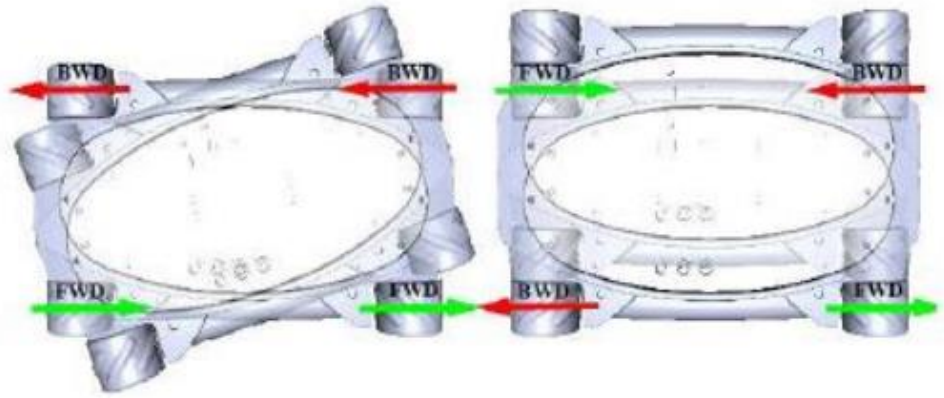


Figura 7: Movimiento con rueda tipo Mecanum *[15]*

1.4.10 Cadena cinemática para el manipulador del Youbot

Denavit-Hartenberg facilita parámetros sacados por un algoritmo que deben ser tomados en cuenta para el control de distancias y ángulos del brazo Youbot se pueden apreciar en la Tabla 1. [15]

Tabla 1: Parámetros dados por Denavit-Hartenberg *[15]*

	Marco anterior	Traslación (cm)			Rotación en 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	5.71	90	0	180

1.4.11 Sistema Operativo Robótico

Sistema Operativo Robótico es un entorno de trabajo también conocido como ROS con un set de herramientas capaz de proporcionar funciones de un variado grupo de computadores, utilizado para escribir software en diferentes robots. ROS tiene más de 2 mil paquetes, los cuales contienen drivers y funciones específicas capaces cada uno de proporcionar una funcionalidad especializada. [16]

Se puede decir que ROS es una colección de bibliotecas, herramientas y convenciones con el objetivo principal de simplificar tareas de crear un comportamiento robótico complejo y robusto en una amplia diferencia de robots, creado desde un principio para impulsar el aumento de software robótico colaborativo entre programadores de distintas ramas. [17]

Tema (topic)

Los temas dentro de ROS son canales de información encargados de comunicar los nodos entre sí, cada nodo puede emitir o suscribir información mediante los temas siendo estos nada más que el mensaje que se envía o recibe en distintos tipos de clases. [18]

Paquete (package)

Ros es un software que se encuentra organizado en paquetes, siendo un paquete capaz de contener un nodo, conjunto de datos, librería o cualquier cosa que pueda constituir un módulo, además los paquetes pueden organizarse en pilas. [18]

Nodo (node)

Un nodo es un proceso encargado de realizar algún tipo de computación en el sistema, siendo capaces de combinarse dentro de un grafo para compartir información entre ellos para crear ejecuciones complejas. Los nodos son capaces de controlar independientemente las diferentes partes de un robot de forma independiente. Controlando un nodo los motores del robot, otro los sensores y otro la construcción de mapas para su movilización. [18]

Pila (stack)

Conjunto de nodos capaces de proporcionar una funcionalidad específica, por ejemplo, una pila de navegación. [18]

Comandos de ROS

roscd: Comando utilizado para cambiar a un directorio de paquete o pila.

roscore: Utilizado para ejecutar todo lo necesario para dar soporte de ejecución al sistema completo de ROS. Es necesario su ejecución en todo momento para permitir que se comuniquen los nodos mediante un determinado puerto.

roscreate-pkg: Este comando debe ser ejecutado desde el espacio de trabajo válido para contener paquetes y es el encargado de crear e inicializar un paquete.

rosnode: Proporciona información sobre un nodo disponiendo de las siguientes opciones.

- rosnode info *nodo*: Muestra información general sobre un nodo específico.
- rosnode kill *nodo*: Mata al proceso.
- rosnode list: Muestra todos los nodos ejecutándose.
- rosnode machine *máquina*: Muestra los nodos ejecutándose en dicha máquina.
- rosnode ping *nodo*: Comprueba la conectividad con dicho nodo.

rosrun: Permite ejecutar aplicaciones de un paquete sin necesidad de cambiar a su directorio.

rostopic: Permite obtener información sobre un tema.

- rostopic bw: Identifica el ancho de banda consumido por un tema
- rostopic echo: Imprime datos del tema por la salida estándar
- rostopic find: Encuentra un tema
- rostopic info: Imprime información de un tema
- rostopic list: Imprime información de los temas activos
- rostopic pub: Publica datos a un tema activo
- rostopic type: Imprime el tipo de información de un tema

roswtf: Permite identificar errores por lo cual primero se debe ejecutar roscd y después roswtf. [19]

1.4.12 Simulador 3D Gazebo

Gazebo es un simulador de entornos 3D que facilita identificar el comportamiento de diferentes robots en un mundo virtual permitiendo diseñar robots de forma personalizada, crear mundos virtuales con herramientas CAD o importar modelos creados con anterioridad. Lo más importante del simulador es que puede ser sincronizado con ROS logrando que sus robots publiquen y subscriban información en nodos dentro del sistema operativo robótico. [20]

1.4.13 Circuitos Electrónicos

Los circuitos electrónicos están definidos como un conjunto de componentes electrónicos interconectados entre sí para cumplir una función en específico, comunicándose entre ellos mediante corrientes eléctricas que representan información necesaria para ser procesada y transmitirse hacia diferentes dispositivos. Los dispositivos electrónicos al recibir la información necesaria son encargados de convertir señales eléctricas procedente del medio ambiente como: luz, sonidos, cambios de temperatura, etc. [21]

1.4.14 Microcontrolador

Un microcontrolador siendo el mayor exponente del desarrollo de la electrónica digital debido que puede ser programado gracias a que cuentan con memoria y unidades de entrada y salida. En la actualidad los microprocesadores y microcontroladores son chips omnipresentes debido que los encontramos en todas partes para resolver tareas de diferente complejidad entre las comunicaciones, automatización y electrodomésticos. Los microcontroladores operan con millones de transistores en su interior encargados de crear circuitos complejos, también llamados unidad de procesamiento central o CPU debido que muchos pueden ser usados como el “cerebro” de un sistema computacional [22]

1.4.15 Tarjeta Embebidas

Las tarjetas embebidas son circuitos electrónicos que poseen como función controlar automáticamente los procesos de diferentes máquinas o actuadores teniendo en cuenta que contiene un microprocesador, existen 3 pasos fundamentales dentro de estos sistemas electrónicos para su funcionamiento, los cuales son: entrada de datos, procesamiento y salida de la información. [23]

ESP-32: El esp-32 fue creado por Espressif Systems, convirtiéndolo en un sistema de bajo consumo y bajo costo en un chip SoC (System On Chip), el cual cuenta con comunicación inalámbrica Wifi y Bluetooth, además de funcionar con un microprocesador Tensilica Xtensa LX6 de doble nucleo con 240 MHz como frecuencia de reloj permitiéndole poder operar multitareas con distintos periodos. Las características principales son:

- **SRAM:** 520 KB, para datos e instrucciones
- **ROM:** 448 KB, para arranque y funciones básicas
- **Bluetooth:** v4.2 BR/EDR/BLE
- **Wifi:** 802.11 b/g/n/e/i
- **Frecuencia de clock:** 600 DMIPS [24]

Raspberry Pi: En febrero del 2012 la Raspberry Pi fue creada por Raspberry Pi Foundation, esta placa electrónica en un inicio fue diseñada para fomentar y enseñar las ciencias básicas transformándose en un computador compacto de bajo costo con funcionamiento bajo el sistema operativo Linux que permite a personas de todas las edades explorar y aprender la computación, además de aprender lenguajes de programación como Scratch y Python.

Características principales:

- 5V/2.5A DC via micro usb
- HDMI full size
- Puerto MIPI CSI camera
- Bluetooth 4.2 BLE
- Puerto MIPI DSI display
- 40 pines GPIO header
- Wifi 2.4GHz-5GHz [25]

1.4.16 Teleoperación

El conjunto de tecnologías para el control, operación o gobierno a distancia de distintos dispositivos por una persona es considerado teleoperación. Por tanto, teleoperar es el acto que realiza una persona para controlar a distancia un dispositivo, mientras que un dispositivo teleoperado es teleoperado o controlado mediante un sistema de teleoperación. [26]

1.4.17 Telerobótica

Es la unión de tecnologías que comprenden el control, la monitorización y reprogramación a distancia de un robot por personas especializadas. Siendo así de la teleoperación de un robot, denominado telerobot o robot teleoperado. [26]

1.4.18 Tecnologías de comunicación inalámbrica

La transmisión de datos por medio del aire en dos o más dispositivos sin el uso de cables o cualquier tipo de soporte físicos se le conoce como comunicaciones inalámbricas, como no cuenta con un soporte físico la mayor desventaja que mantiene es su vulnerabilidad de seguridad debido a que cualquier persona se podría conectar desde cualquier lugar cercano, la distancia de operación varía según los distintos tipos de comunicaciones inalámbricas.

Las comunicaciones inalámbricas debido a que se basan en ondas de radio que permiten movilidad y flexibilidad a diferencia de las comunicaciones por cableado, a pesar de eso las comunicaciones inalámbricas tienen una velocidad de transmisión de 108 Mbps que es muy baja en comparación de las comunicaciones cableadas que llegan a 1 Gbps, una ventaja es que se pueden combinar los dos tipos de comunicaciones si es necesario debido a su compatibilidad para así poder alargar distancias o poder llegar a lugares donde la señal inalámbrica no puede llegar por obstrucción de obstáculos muy grandes. [27]

Wifi: La señal wifi tiene un alcance de hasta 250 metros en exteriores sin obstáculos, pertenece a tecnologías inalámbricas de corto alcance la cual en su mayoría de veces es utilizada en departamentos habitacionales, edificios con oficinas o centros educativos para comunicar dispositivos de forma inalámbrica, tecnología con un ancho de banda de entre 2.4 GHz y 5GHz basada en IEEE 802.11 de tecnología de

comunicaciones inalámbricas manteniendo 13 o 25 canales según su ancho de banda. [28]

Wimax: World Wide Interoperability for Microwave Access o Wimax conocido así por sus siglas, esta tecnología es un estándar tecnológico basado en IEEE 802.16 de comunicaciones inalámbricas de acceso de banda ancha aplicado para largas distancias con un alcance hasta 70km según los obstáculos presentes entre tx y rx, utiliza ondas de radio en las frecuencias superiores a 2.3GHz. [29]

Bluetooth: Bluetooth es un estándar tecnológico IEEE 802.15 diseñado para comunicaciones a corta distancia con un alcance máximo de 10m, a nivel mundial utiliza la frecuencia de 2.4GHz. Para comunicar dispositivos mediante Bluetooth se sigue un esquema maestro-esclavo el cual soporta hasta ocho dispositivos conectados simultáneamente en una configuración básica de comunicación. [30]

Tabla 2: Tabla comparativa entre Wimax, Wifi y Bluetooth

Especificación	Wimax	Wifi	Bluetooth
Estándar IEEE	802.16	802.11	802.15
Frecuencias de operación (GHz)	2.5/3.5/5.8	2.4/5	2.4
Canal de banda Ancha (MHz)	1.25 a 20	10 o 20 o 40	1
Velocidad de datos (bps)	124 M/1G	11-54M/1.2G	721K
Alcance (m)	40-70	100	10

1.4.19 Modelo TCP/IP

Desarrollado en los años 70 por DARPA (Defence Advanced Research Projects Agency) se convirtió en la base de internet sus siglas significan: TCP (protocolo de control de transmisión) e IP (protocolo de internet), es un conjunto de protocolos que permite el transporte seguro de información por las redes y se encuentra dividido en 4 capas que son: acceso a red, internet, transporte y aplicación. [31]

1.4.20 Protocolos de comunicación sobre TCP/IP

Los protocolos de comunicación son reglamentos o estándares formales que cada servidor y dispositivo debe tener para intercambiar información por medio de una red,

estas normas deben de usarse de forma obligatoria en todas las máquinas para que la comunicación no resulte caótica entre distintas marcas de dispositivos.

HTTP: Protocolo de Hiper-Texto es el protocolo de transferencia que permite el intercambio de información entre clientes web y servidores HTTP. Fue propuesto en 1999 por Tim Berners-Lee y es usado para los servicios web XML. [27]

HTTPS: Protocolo de Transferencia de Hiper-Texto que permite una conexión segura sin dejar interceptar la información entre el servidor y el cliente por personas no autorizadas convirtiéndola en una versión mejorada de HTTP. [32]

Socket TCP y UDP: Los sockets UDP utilizan el protocolo de datagramas de usuario, necesitando que le entreguen paquetes de datos que el usuario debe construir y los TCP utilizan el protocolo de control de la transmisión el cual admite bloques de datos cuyo tamaño puede ir desde 1 byte hasta muchos K bytes. UDP al perder información no hace nada en cambio TCP lo retransmitirá hasta que los datos sean entregados correctamente o se produzca un número máximo de retransmisiones. [33]

1.4.21 Socket

Socket es un método para la comunicación a través de la cual una aplicación puede enviar y recibir datos, de la misma manera que un identificador de archivo abierto permite que una aplicación lea y escriba datos en un almacenamiento estable. Un socket permite que una aplicación se conecte a la red y se comunique con otras aplicaciones que están conectadas a la misma red. [34]

1.5 OBJETIVOS

1.5.1 Objetivo general

Desarrollar un sistema de control para la teleoperación del robot Kuka Youbot mediante comunicaciones inalámbricas con el uso de tarjetas embebidas.

1.5.2 Objetivos específicos

- Estudiar el Estado del Arte del robot Kuka Youbot.
- Diseñar un modelo del sistema de teleoperación para asegurar el perfecto funcionamiento del robot Kuka Youbot en diferentes aplicaciones.
- Implementar el sistema de control con un controlador de hardware y software libre para la manipulación inalámbrica del robot Kuka Youbot.

CAPÍTULO II

METODOLOGÍA

2.1 Materiales

En la selección de materiales para este proyecto de investigación se tomó en cuenta el tipo de comunicación disponible de forma inalámbrica con el que cuenta el robot Kuka Youbot para poder elaborar el diseño del circuito PCB, la base bibliográfica se adquirió en libros, artículos científicos, documentos web, revistas técnicas y datasheet de fabricantes.

2.2 Métodos

2.2.1 Modalidad de la Investigación

En este trabajo de investigación se realizó con base en el concepto de investigación aplicada debido a que tiene como objetivo la generación de conocimientos que pueda ser aplicado directamente en temas sociales o al sector productivo, estos conocimientos adquiridos se usaron para dar solución a los generados por el robot Kuka Youbot, creando un sistema de control para la teleoperación del robot con la utilización de tecnología de hardware y software libre.

La investigación bibliográfica documentada se aplicó para obtener información con bases teóricas que facilite el diseño del módulo de control inalámbrico para el Kuka Youbot. Se tomó en cuenta el tipo de comunicación inalámbrica disponible para la búsqueda de información en libros, artículos científicos y proyectos desarrollados en otros países y en el Ecuador en los cuales realizaron estudios con: circuitos electrónicos, aplicaciones robóticas, módulos de control y sistemas micro controlados.

La investigación experimental se aplicó en el diseño del circuito para el controlador inalámbrico del robot, realizándose pruebas de funcionamiento y así verificar que el controlador cumpla con características adecuadas al momento de controlar el Kuka Youbot.

Además, la investigación de campo se realizó para determinar las características del prototipo y circuito a diseñarse. La recolección de información, adquisición de datos y validación de funcionamiento se utilizó de forma directa en el laboratorio de

Robótica de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

2.2.2 Recolección de la Información

El presente trabajo de investigación requirió la recopilación de información directamente de wiki.ros.org para un mayor entendimiento de bibliotecas y herramientas para la creación de aplicaciones robóticas, así mismo como en manuales de operación, libros, internet, revistas científicas, papers, trabajos de investigación y de la guía del docente tutor para el desarrollo del prototipo.

2.2.3 Procesamiento y Análisis de Datos

El procesamiento y análisis de datos se realizó mediante una clasificación de la documentación alcanzada, exponiendo los entornos a investigarse en el proyecto de una forma ordenada. Considerando los siguientes parámetros, se realizó una distinción crítica de los datos adquiridos a lo largo de la recolección de información.

- Obtener datos técnicos y específicos para determinar las características del sistema a diseñarse con un funcionamiento óptimo.
- Entender la información obtenida para alcanzar a plantear estrategias hacia la solución del problema que se puedan presentar a lo largo de la investigación.

2.2.4 Desarrollo del Proyecto

En el desarrollo e implementación del controlador electrónico de software libre para la manipulación del Kuka Youbot, fue necesario cumplir con la siguiente organización de actividades.

1. Análisis del funcionamiento del KUKA Youbot.
2. Identificación de los parámetros eléctricos y mecánicos que componen al KUKA Youbot.
3. Determinación de los componentes y etapas de control requeridas para la operación del KUKA Youbot.
4. Análisis de métodos y protocolos de comunicación para el intercambio de información entre el KUKA Youbot y un controlador.
5. Selección de la tecnología de comunicación y procesamiento de datos a

utilizarse en el módulo controlador.

6. Diseño de un circuito para el controlador del KUKA Youbot.
7. Diseño del sistema de alimentación eléctrica para el circuito controlador.
8. Programación del software de la tarjeta electrónica del módulo controlador del KUKA Youbot.
9. Elaboración de la tarjeta electrónica del módulo controlador.
10. Ensamblado y conexión de los componentes del módulo controlador en un contenedor de protección.
11. Evaluación y pruebas de funcionamiento del prototipo.
12. Corrección de errores.
13. Análisis de resultados.
14. Elaboración del informe final.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1 INTRODUCCIÓN

Dentro de la automatización de procesos la adquisición de software para manipular distintos equipos está sujeta a licencias, por lo cual el área de investigación en la comunidad universitaria es de suma importancia ya que permite el desarrollo de hardware y software libre en trabajo colaborativo. Este proyecto de investigación destaca con la manipulación inalámbrica del robot Kuka Youbot gracias a un controlador de software y hardware libre el cual usa comunicación WI-FI con ayuda de sockets para proporcionar un intercambio de flujo de datos y contar con un movimiento libre e independiente en ambientes de entorno incómodo, difícil acceso o peligroso para los humanos.

El sistema de control esta dado por la creación de una tarjeta electrónica con comunicación wifi, usando un microcontrolador esp32-wroom-32 el cual es un módulo genérico con capacidad de comunicación wifi y bluetooth, dando acceso al uso de sockets y tener una comunicación fluida entre la placa controladora y el robot, el controlador puede ser programado en distintos lenguajes de programación como Python, C++, ESP-IDF, entre otros. Los distintos botones y joysticks que se encuentran en el controlador se podrá modificar su acción dependiendo del usuario y el lenguaje en el que se programe, para lo cual solo se necesita revisar la Tabla 4 Para identificar los pines usados. La comunicación inalámbrica fue puesta a prueba con el simulador gazebo y luego utilizando elementos reales para hallar un óptimo funcionamiento del sistema. La comunicación WiFi entre el controlador y el Kuka funciona con el establecimiento de un enlace de comunicación bidireccional entre servidor-cliente con el protocolo TCP, creando un circuito virtual para el intercambio fiable de datos entre ambos extremos.

3.2 DESARROLLO DE LA PROPUESTA

3.2.1 Robot Kuka Youbot

El robot Kuka Youbot fue diseñado para la investigación y enseñanzas científicas como un manipulador móvil de interfaces abiertas permitiendo a los desarrolladores acceder a casi todos los niveles de control del hardware. Este robot debe ser utilizado solo en aplicaciones de laboratorio y no para aplicaciones industriales.

Es un robot conformado por una plataforma móvil con 4 ruedas Mecanum (son ruedas especiales compuesta de rodillos instalados a 45° Figura 8) accionadas por motores brushless de maxon, uno o dos brazos robóticos desmontables en el chasis, los cuales pueden operar de forma separada vía ethernet. Es considerado un robot de escritorio debido a su pequeño tamaño de 53 cm de longitud, 36 cm de anchura y 11 cm de altura (Tabla 3), las recomendaciones del manual nos dicen operarlo en un lugar amplio debido a su gran fuerza de movimiento lo que puede dañar objetos, herir personas o sufrir daños el robot de forma permanente.

Figura 8: Hardware Youbot
Fuente: Investigador (Photoshop)

La plataforma de Youbot cuenta con cuatro ruedas Mecanum con motores sin frenos mecánicos, el control de los motores es por medio de interfaz EtherCAT. La plataforma tiene la capacidad de moverse a cualquier dirección y posicionarse de forma precisa, cuenta con un eje oscilante para compensar las irregularidades del suelo.

Tabla 3: Especificaciones de la plataforma Youbot omni-direccional

Cinemática	4 ruedas Mecanum Kuka
Largo	580 mm
Ancho	380 mm
Altura	140 mm
Espacio	15 mm
Peso	20 kg
Carga útil	20 kg
Velocidad máxima	0.8 m/s
Comunicación	EtherCAT
Voltaje	24 VDC
Batería	Ácido de plomo 24V/5ª

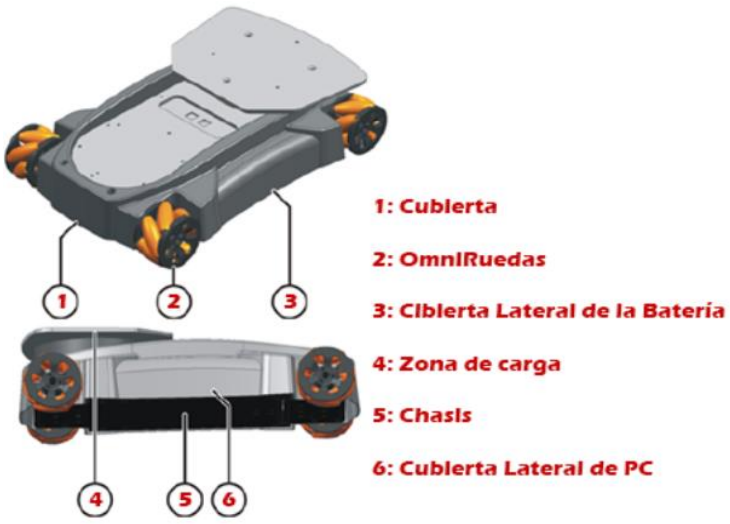


Figura 9: Partes de la Plataforma Youbot
Fuente: Investigador (Photoshop)

Al contar con ruedas Mecanum el robot Youbot se puede desplazar con facilidad solo con invertir el sentido de giro de distintas ruedas como se puede ver en la Figura 10.

Figura 10: Movimientos de la Plataforma Youbot según la rotación de sus ruedas
Fuente: Investigador (Photoshop)

Tabla 4: Especificaciones del brazo Youbot de Kuka *[35]*

Cinemática	Cadena serial con 5 ejes rotativos
Altura	655mm
área de trabajo	0.513 m^3
Peso	6.3 Kg
carga útil	0.5 Kg
Estructura	Yeso de magnesio
Comunicación	EtherCAT
Voltaje	24 VDC
Potencia	80W

El brazo robótico tiene un sistema cinemático de 5 ejes con servomotores los cuales no cuentan con frenos mecánicos, en la Tabla 4 se pueden apreciar las características principales. La base de este cuenta con una entrada de 24 VDC y de comunicación de datos RJ45. El brazo puede ser utilizado de forma independiente sin la necesidad de la plataforma y cuenta con las siguientes secciones como se puede ver en la Figura 11, además podemos observar en la Tabla 5 el alcance del brazo Youbot.

Figura 11: Partes del Brazo Youbot
Fuente: Investigador (Photoshop)

Tabla 5: Especificaciones de los motores de las articulaciones Youbot *[35]*

Datos del eje	Rango	Velocidad
Eje 1	+/- 169º	90 º/s
Eje 2	+ 90º / -65º	90 º/s
Eje 3	+146º / -151º	90 º/s
Eje 4	+/- 102.5º	90 º/s
Eje 5	+/- 167.5º	90 º/s

La pinza del Youbot tiene un motor en cada uno de sus dedos que pueden ser movidos de forma independiente con una apertura máxima de 10mm hacia cada lado, teniendo la capacidad de sujetar pequeños cubos o barras de 20mm (Figura 12) y cuenta con su

propia función antibloqueo para evitar soltar la pieza agarrada en un fallo energético.

Figura 12: Grippers creados por aquipo RoboCup *[36]*

Tabla 6: Datos del actuador *[37]*

Datos del actuador	Articulación 1	Articulación 2	Articulación 3	Articulación 4	Articulación 5	Ruedas
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
Torque nominal (Nm)	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 ($Kg*$	13.5	13.5	13.5	9.25	3.5	13.5

mm^2)						
Velocidad nominal (RPM)	5250	5250	5250	2850	2800	5250
Peso (gr)	110	110	110	75	46	110
CAJA DE CAMBIOS						
Relación de reducción	156	156	100	71	71	26
Momento de inercia ($Kg * mm^2$)	0.409	0.409	0.071	0.07	0.07	0.14
Peso (gr)	-	-	-	-	-	224
CODIFICADOR						
Revoluciones	4000	4000	4000	4000	4000	4000

Los dedos por defecto que llegan al comprar el robot son cortos y rígidos por lo cual no puede sujetar objetos redondos con facilidad, para poder manipular este tipo de objetos existen a la venta unos grippers adaptativos triple de FESTO llamados FinGrippers vistos en la Figura 13, gracias al aquipo RoboCup de la Universidad de Ciencias Aplicadas de Bonn-Rhine-Sieg que lograron adaptar al sistema original Youbot de dos dedos con un mecanismo paralelo sin la necesidad de algún software de control o placa electrónica nueva que puede ser comprado o impreso en 3D como se puede visualizar en la Figura 12. La fricción de los ejes dependerá de la precisión de la impresora 3D utilizada dando como resultado que si los orificios son muy pequeños la fricción sube a tal punto que los motores de la pinza no podrán mover la pinza Kuka Youbot. [38]

Tabla 7: Especificaciones de la Pinza

Cinemática	Desmontable con 3 dedos
Distancia de cada dedo	20mm
Distancia de apertura	50mm

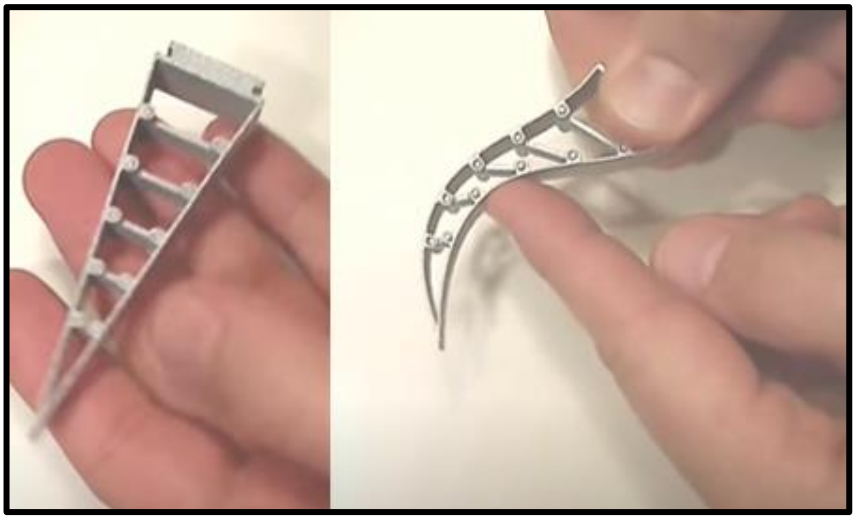


Figura 13: Dedos festo *[39]*

3.2.2 Sensores Disponibles de Kuka para Youbot

En la tienda Youbot-store se ofertan 3 sensores, entre ellos 2 tipos de cámaras y un láser telémetro.

ASUS Xtrion PRO-LIVE es una cámara 3D que infiere distancias a partir de un patrón capturado utilizando un proyector de infrarrojos y una cámara de infrarrojos, consta de una cámara web a color con un rango de detección de 0.8m – 3.5m. [40]

Microsoft LifeCam es una cámara web USB, la cual contiene controladores Linux UVC por lo que no necesita ningún tipo de instalación más que conectar la cámara al conector USB del robot. [41]

URG-04LX-UG01 es un telémetro láser con una distancia de escaneo máxima de 5.6m, cuenta con un ángulo de escaneo de 240º adquiriendo escaneos con una frecuencia de hasta 10Hz. [42]

3.2.3 Comunicación Inalámbrica del Robot Kuka Youbot

El robot cuenta con una placa Mini-ITX y un procesador Intel Atom Dual Core D510 (Tabla 8), por lo que al cargar el sistema operativo Ubuntu junto con ROS se convierte en un computador que solo necesita la conexión de una pantalla, teclado y mouse para usarse como cualquier otro computador, gracias a esto se puede instalar los drivers necesarios para obtener conexión WiFi mediante de un adaptador wifi USB

3.2.4 Versiones de ROS

En la Universidad de Stanford gracias a los estudiantes de doctorado Eric Berger y Keenan Wyrobek en el 2007 apareció SWITCHYARD hoy conocido como ROS, para lo cual las versiones conocidas son: Foxy Fitzroy, Melodic Morenia, Lunar Loggerhead, Kinetic Kame, Jade Turtle, Indigo Igloo, Hydro Medusa, Groovy Galapagos, Fuerte Turtle, Electric Emys, Diamondback, C Turtle, Box Turtle. Teniendo soporte técnico solo en una versión antigua (Indigo) y 4 versiones nuevas (Foxy, Melodic, Lunar, Kinetic). ROS desde sus inicios fue pensado para trabajar bajo código abierto para facilitar el uso de herramientas y bibliotecas elegidas por los usuarios y así poder cambiar los desarrollos de software robóticos con facilidad. [43]

3.2.5 Comparación entre distintas versiones de ROS mayormente usadas

Para poder comparar las versiones con mayor uso de ROS se debe tener en cuenta la compatibilidad que soporta con los distintos sistemas operativos disponibles, requerimientos de funcionamiento mínimo y programas necesarios para su funcionamiento o simulación para no tener limitaciones en la configuración del sistema a diseñarse.

Tabla 8: Diferencias entre versiones de ROS *[44]*

Versión de Ros	Requerimientos mínimos	Soporte recomendado	Año de lanzamiento	Requisitos Exactos
Noetic	C++14 Python 3.8 Lisp SBCL	Ubuntu Focal Fossa (20.04) Debian Buster	2020	Boost 1.71! CMake3.16.3! Gazebo 11.x*

	1.4.16	Fedora 32		Ignition Citadel Ogre 1.9! OpenCV 4.2! PCL 1.10! PyQt 5.14.1! Qt5 5.12.5!
Melodic	C++14 Python 2.7 Lisp SBCL 1.3.14	Ubuntu Artful (17.10) Ubuntu Bionic (18.04) Debian Stretch Fedora 28	2018	Boost 1.62 CMake3.9.1 Gazebo 9.0.0* Ogre 1.9 OpenCV 3.2* PCL 1.8.1 PyQt 5.7
Kinetic	C++11 Python 2.7 Lisp SBCL 1.2.4 CMake 3.0.2 Boost 1.55	Ubuntu Wily (15.10) Ubuntu Xenial (16.04) Debian Jessie Fedora 23 Fedora 24	2016	Ogre3D 1.9.x Gazebo 7 PCL 1.7.x OpenCV 3.x Qt 5.3.x PyQt5
Hydro	C++03 Boost 1.48 Lisp SBCL 1.0.x Python 2.7 CMake 2.8.3	Ubuntu Precise (12.04 LTS) Ubuntu Quantal (12.10) Ubuntu Raring (13.04)	2013	

3.2.6 Versión de ROS utilizadas

Para el robot real Kuka Youbot ha estado disponibles solo 3 versiones de ROS (Versión 0.3.0: Ubuntu 10.04 con ROS Electric, Versión 0.4.0: Ubuntu 12.04 con ROS Fuerte, Versión 1.0.1: Ubuntu 12.04 con ROS Hydro), lo cual al comprar el robot por

defecto desde febrero del 2015 llega con la versión 1.0.1 Ubuntu 12.04 con ROS Hydro, se adquiere un dispositivo USB Live que viene con Ubuntu y ROS listos para su instalación en casi cualquier computador, este dispositivo de instalación ya cuenta con todos los controladores y aplicaciones para el control del Youbot. [45]

A medida que se fue recopilando información en foros de ROS y en base de pruebas y errores en la instalación de diferentes versiones de Ubuntu y ROS se encontró una correcta instalación y ejecución de la aplicación ejemplo youbot_ros_hello_world sin errores con Ubuntu Xenial 16.04 y ROS Kinetic.

3.2.7 Lenguajes de programación para comunicarse con ROS

Python, JavaScript y C++ son los principales lenguajes de programación dentro de ROS por lo cual existen guías de estilo de codificación en wiki.ros.org para poder exponer pautas necesarias para obtener mejores prácticas y poder modificar paquetes creados por otros programadores siguiendo las convenciones de estilo en el paquete a editar. Esta guía está direccionada para todo el código ROS, básico y avanzado.

3.2.8 Diferencias entre lenguajes de programación usados con ROS

Debido que C++ y Python mantienen más información para publicar y suscribir a diferentes temas dentro de ROS son lenguajes de programación expuestos en la siguiente Tabla 9 comparativa siendo los mismos que pueden ser usados para programar nodos de comunicación con ROS, por lo cual se expondrá las características con mayor importancia para tener un mayor rendimiento y facilidades del programador.

Tabla 9: Benchmark entre Python y C++ *[46]*

Lenguaje de Programación	Tiempo del sistema en ejecutarse	Uso del CPU	Uso de RAM	Tiempo de programación empleado
Python	0.37s	99%	2101312kB	Ocupa menor tiempo de

				programación
C++	0.14s	99%	941808kB	Ocupa mayor tiempo de programación

3.2.9 Lenguaje de programación usado en el proyecto

Revisando las características de cada lenguaje de programación expuestos en la Tabla 9 se puede apreciar que C++ es superior a Python, esto también debido a que C++ es un lenguaje de programación antiguo en comparación con Python. La decisión de usar Python fue tomada por las facilidades presentadas al momento de programar y la disminución de líneas de código evitando definir variables y tipos de datos. Mientras se hacía pruebas de conectividad con el control y con el envío de datos al robot para su movimiento se facilitó el entendimiento de la funcionalidad de los temas y con eso poder concluir con un código estable y corto.

3.2.10 Simuladores para ROS

Gazebo mantiene paquetes con cinemática, dinámica, geometría y modelos visuales 3D de Kuka Youbot en formato URDF, además de herramientas para operar el robot y archivos de lanzamiento. [47]

V-REP proporciona interfaces para scripts lua integrados, nodos ROS y diferentes clientes API para un modelo Youbot de Kuka, además de interfaces de cinemática inversa y solucionadores de planificación de rutas. [48]

OpenRAVE cuenta con todo el DOF del robot Kuka Youbot, al integrar a Youbot con este simulador se puede usar IKFAST, estadísticas de enlaces, mapas de capacidades, varios planificadores y accesibilidad inversa. [49]

Webots contiene el DOF completo del Youbot de Kuka con un modelo preciso en la física del robot, con ruedas omnidireccionales extremadamente realistas, además cuenta con una facilidad de ampliación con los sensores Hokuyo o Kinect y la facilidad

de modificar el modelo de 2 brazos, 1 brazo o solo plataforma. [50]

3.2.11 Simulador usado para ROS

Gazebo es uno de los simuladores con mayor trabajo dentro de la robótica por ser calificado de porfa positiva por programadores experimentados, por lo cual además se puede obtener más información para su utilización. Se cuenta con muchos diseños funcionales del robot Kuka Youbot en internet para poder simular, además se tomó en cuenta la facilidad de instalación juntamente con ROS en un solo comando dentro del terminal de Ubuntu (sudo apt-get install ros-kinetic-desktop-full).

3.2.12 Requerimientos Técnicos del Sistema

Según las características propias del robot Kuka Youbot y el software disponible dentro del mismo se consideró necesario especificar los requerimientos técnicos para una comunicación segura con el robot que se detallan a continuación.

- Módulo de comunicación inalámbrica capaz de utilizar protocolos de comunicación dentro de la capa de red del modelo OSI con software libre y amplitud de programación en distintos lenguajes de programación.

- En base a revisar el ejemplo teleop_keyboard genérico para robots twist se considera necesario el uso de una palanca de mando o joystick para el control completo de la plataforma, aprovechando así al máximo el uso de las rudas Mecanum y su desplazamiento de la plataforma móvil.

- El brazo de Kuka Youbot se optó por el control según la planeación de trayectorias previamente programadas en posiciones específicas, por lo tanto, es necesario el uso de botones para activar el posicionamiento requerido.

- La apertura y cierre de la pinza tiene un control de dos dedos paralelos con un rango graduable por lo cual también es necesario el uso de un botón según las dimensiones del objeto que se desea agarrar.

3.2.13 Diseño General del Sistema

El controlador está diseñado para poder adaptarse a controlar el robot real o el simulado en gazebo mediante la capa de red del modelo OSI por el protocolo TCP y creación de sockets en cada extremo con configuración esclavo-maestro. El controlador mantiene configuración esclavo para poder conectar cualquier otro dispositivo con comunicación socket TCP y poder manipular el robot con cualquier aplicación Android de ser necesario. Figura 14

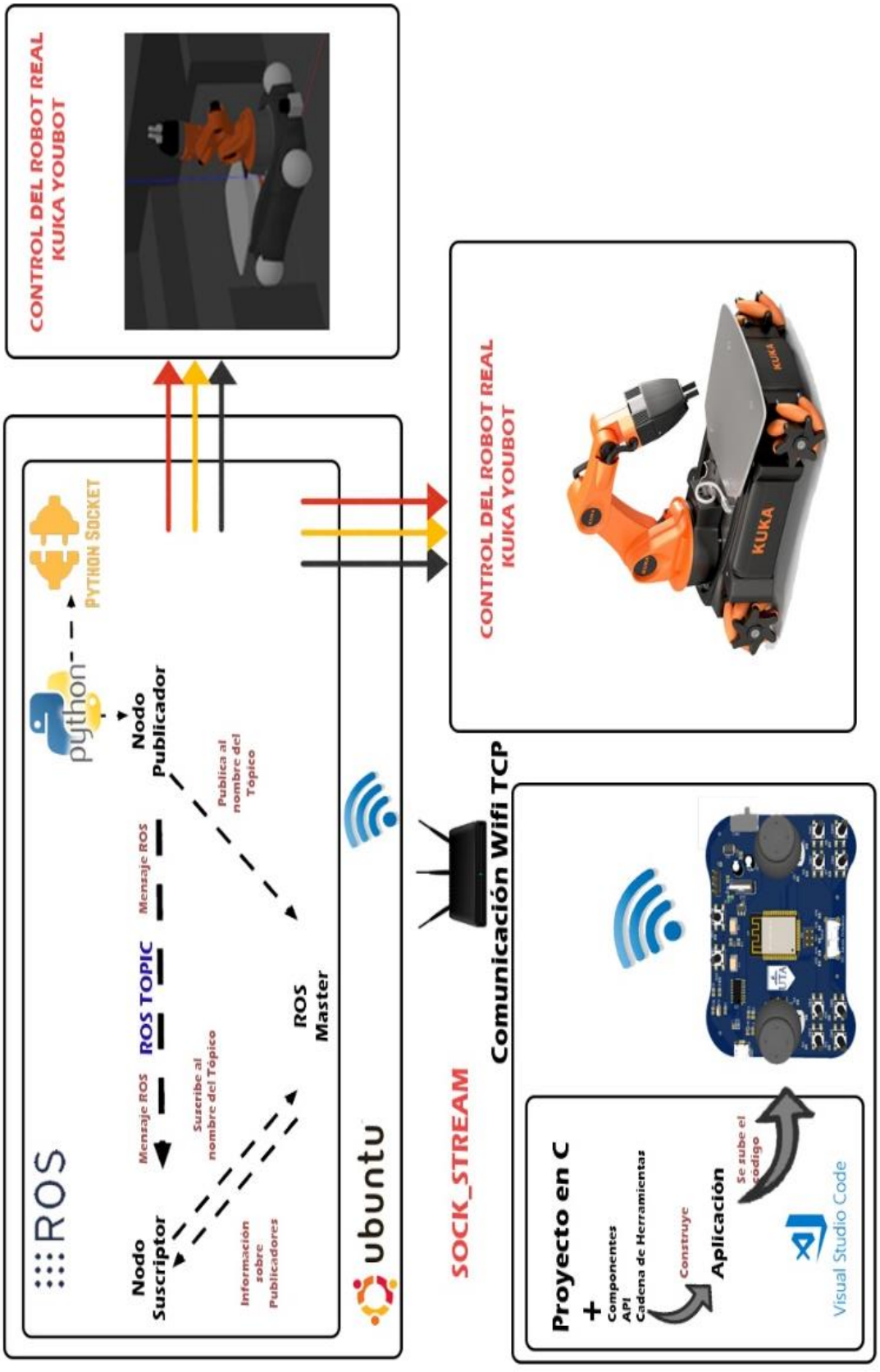


Figura 14: Diagrama del Sistema General
Fuente: Investigador (Photoshop)

3.2.14 Interfaz de Comunicación

La comunicación para controlar el robot fue diseñada con la creación de sockets mediante el protocolo TCP. La comunicación socket puede ser creada con TCP y UDP en donde el protocolo TCP se encarga de verificar una transmisión correcta de los datos entre transmisor y receptor, mientras que el protocolo UDP no lo verifica. Un socket debe ser creado con la especificación del puerto de enlace. La interfaz de comunicación tiene tres procedimientos principales, el del servidor y del cliente.

El primer procedimiento es del funcionamiento del servidor inician con el proceso "bind", encargado de asociar al servidor con una dirección IP y un puerto local. El segundo procedimiento dentro del servidor es "listen" encargado de administrar la cola de espera de los clientes pendientes, rechazando requerimientos en caso de llenarse la cola de espera. EL último procedimiento en el servidor es "accept", encargado de aceptar la conexión del cliente con un descriptor especial creado para el mismo para que el servidor pueda seguir aceptando clientes según su configuración.

El segundo procedimiento es del funcionamiento del cliente que cuenta solo con un proceso "connect", encargado de", encargado de establecer la conexión con el servidor debido que mantiene la dirección IP y puerto de enlace hacia el servidor el cual debe ser aceptado por "accept" anteriormente mencionado en los procesos del servidor.

El tercer y último procedimiento es del envío y recepción de datos que cumple con los procesos "send" y "sendto", encargadas del envío de datos con el control de número de bytes que retornan al ser enviados efectivamente y en donde se encuentra la dirección de memoria de los datos a enviar. Los procesos "recv" y "recvfrom" son encargados de recibir los datos enviados conteniendo la longitud en bytes y la dirección de memoria donde se depositan.

[51]

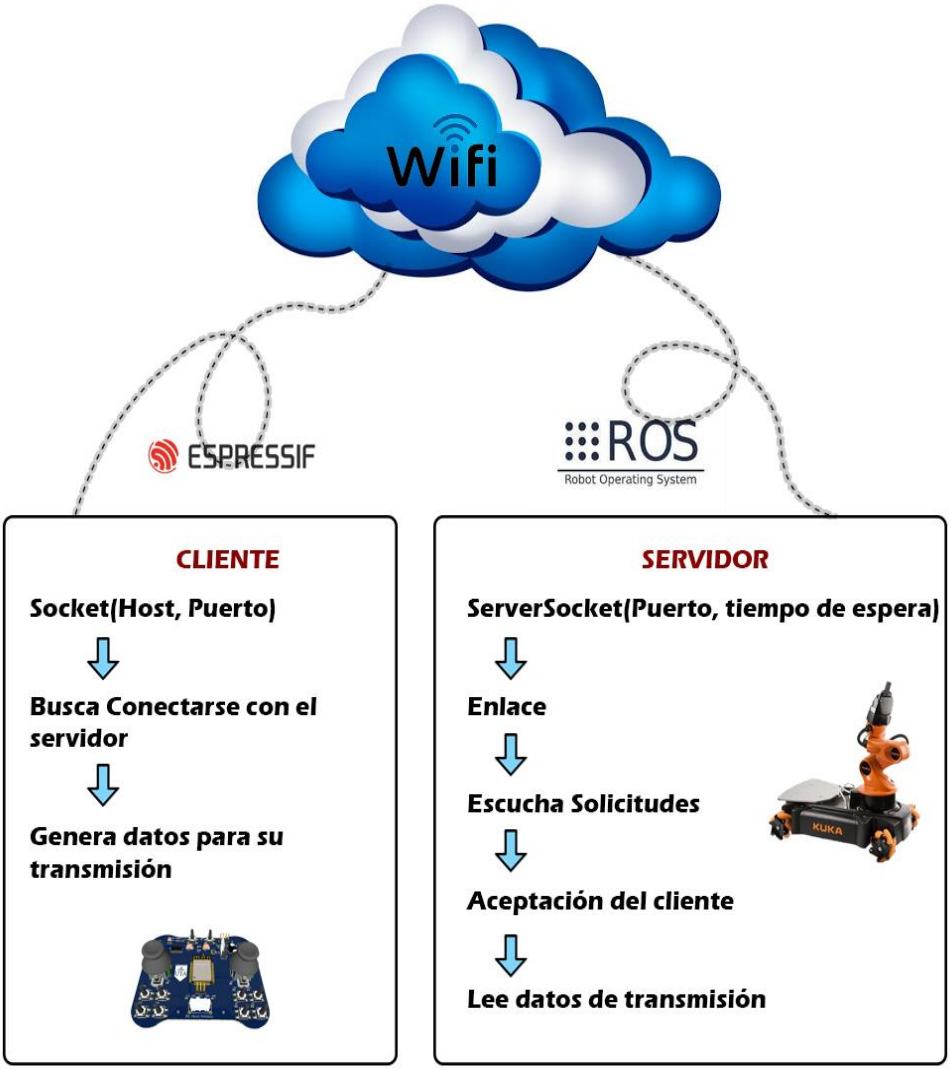


Figura 15: Interfaz de Comunicación
Fuente: Investigador (Photoshop)

3.2.15 Configuración del Hardware para el Youbot

Dentro de la plataforma Youbot viene integrado una batería y una placa Mini-ITX (Tabla 3) como computador, el cual es un PC industrial capaz de comunicarse en ciclos de 1ms a través de EtherCAT. El robot cuenta con conexión de dispositivos de entrada y salida para controlar la PC integrada directamente, para ello se necesita un teclado usb, un mouse usb y un monitor con cable VGA lo cual se conectan un lateral como se muestra en la Figura 16.

Figura 16: Dispositivos E/S Youbot
Fuente: Investigador

Tabla 10: Características placa Mini-ITX

Procesador	Intel Atom Dual Core D510 (cache de 1M, 2x1.66GHz)
Memoria	DDR2 de 2 GB de un solo canal a 667 MHz
Gráficos	Gen3.5 + GFX Core, frecuencia de reloj renderizada de 400MHz hasta 224 MB de memoria compartida
Disco Duro	SSD de 32 GB

Para proceder con el funcionamiento el brazo robótico Youbot debe estar en su posición inicial como se muestra en la Figura 17 con eso se evita que mueva o empuje accidentalmente las articulaciones contra sus topes mecánicos. El procedimiento de inicio se debe realizar con el robot apagado.

Figura 17: Posición inicial del brazo Youbot
Fuente: Investigador

Cada articulación del brazo Youbot viene equipado con codificadores de posición relativa para medir los ángulos de las articulaciones desplazando angularmente con respecto a una posición de referencia, también llamada posición de descanso para indicar que al inicio de la programación se debe comenzar desde este punto. Como recomendación dentro del manual de usuario se debe mover las articulaciones con lentitud al acercarse a estos límites (puede verse en las Figuras 2 y 18) y así evitar daños.

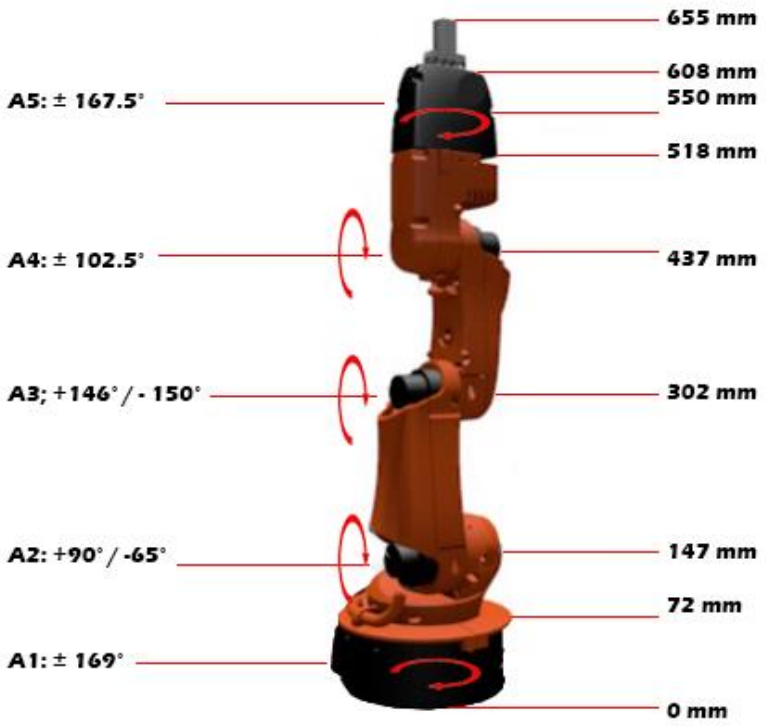


Figura 18: Estructura cinemática del brazo.
Fuente: Investigador (Photoshop)

El encendido del robot se inicia con los dispositivos de E/S necesarios previamente conectados y con el brazo Youbot en su posición inicial, es necesario presionar por unos segundos el botón ON/OFF para encender la alimentación, presionar por una segunda vez para iniciar el PC interno y una tercera vez para encender los motores de la plataforma Youbot, el brazo Youbot tiene su propio botón ON/OFF. En caso de querer apagar el robot se procede a presionar los botones de la misma manera para

encender, pero en el orden contrario sin ser necesario apagar desde el monitor.

Figura 19: Encendido del Kuka Youbot
Fuente: Investigador

3.2.16 Configuración del Software para Youbot

El robot Kuka Youbot viene con un USB cargado con el instalador de Ubuntu y los drivers necesarios para el control del robot. Puede ser instalado en un PC aparte para usar el brazo Youbot por separado o en la PC que viene internamente en la plataforma, debe ser conectado el USB en la parte lateral Figura 16. Se debe priorizar la inicialización de dispositivo de arranque en ““Hard Disk Drives” ingresando a BIOS.

Para iniciar a usar el Youbot se debe instalar la librería Youbot API con los comandos “sudo apt-get install ros-hydro-youbot-driver” y “sudo ldconfig /opt/ros/hydro/lib”. Se debe configurar el archivo:

youbot-ethercat.cfg: Cambiar el nombre de la unidad ethernet (EthernetDevice = eth1) en la dirección /opt/ros/hydro/share/youbot_driver/config/

Verificar las configuraciones de los siguientes archivos.

youbot-base.cfg: Está en la dirección /opt/ros/hydro/share/youbot_driver/config/

```
#in youbot-manipulator.cfg
[JointTopology]
ManipulatorJoint1 = 5
ManipulatorJoint2 = 6
ManipulatorJoint3 = 7
ManipulatorJoint4 = 8
ManipulatorJoint5 = 9
```

youbot-manipulator.cfg: En la dirección /opt/ros/hydro/share/youbot_driver/config/

```
#in youbot-manipulator.cfg
[JointTopology]
ManipulatorJoint1 = 5
ManipulatorJoint2 = 6
ManipulatorJoint3 = 7
ManipulatorJoint4 = 8
ManipulatorJoint5 = 9
```

La instalación de Youbot applications se hace ingresando por comandos a catkin_ws/src y digitando el comando "git clone https://github.com/wnowak/youbot_applications", se sale del directorio con "cd .." y finalmente "catkin_make".

La instalación de ROS se realiza solo con el ingreso de 3 comandos: "sudo apt-get install ros-hydro-youbot-driver-ros-interface", "sudo setcap cap_net_raw+ep /opt/ros/hydro/lib/youbot_driver_ros_interface/youbot_driver_ros_interface" y "sudo ldconfig /opt/ros/hydro/lib". Una vez culminada la instalación se puede empezar a utilizar el robot según las necesidades requeridas.

3.2.17 Nodo publicador en el robot Youbot

El desarrollo de un script en Python con las especificaciones requeridas para un control adecuado del robot al momento de manipularlo de forma inalámbrica con cualquier tipo de dispositivo con comunicación wifi y el uso de socket TCP se logró con la integración de la librería socket y con el uso de los temas tipo geometry_msgs/Twist, brics_actuator/JointPositions y trajectory_msgs/JointTrajectory para el control de la plataforma de ambos robots usados, el brazo del robot real y el brazo del robot simulado respectivamente como se puede ver en la Figura 20.

En la Figura 20 se puede apreciar los parámetros necesarios enviar según el tipo de tema para el control de del robot Kuka Youbot, en la sección 1 se aprecia el tipo de tema para mover la plataforma con el envío de los valores x,y,z dentro de un vector "linear", la sección 2 corresponde al movimiento del brazo y pinza del robot real que deben ser enviados todos los parámetros necesarios en un arreglo de vectores dentro en un vector positions, finalmente la sección 3 corresponde para el control del movimiento del brazo y pinza del robot simulado en gazebo donde se debe enviar el

arreglo de valores dentro de vectores y finalmente unir todos los vectores en uno llamado points.

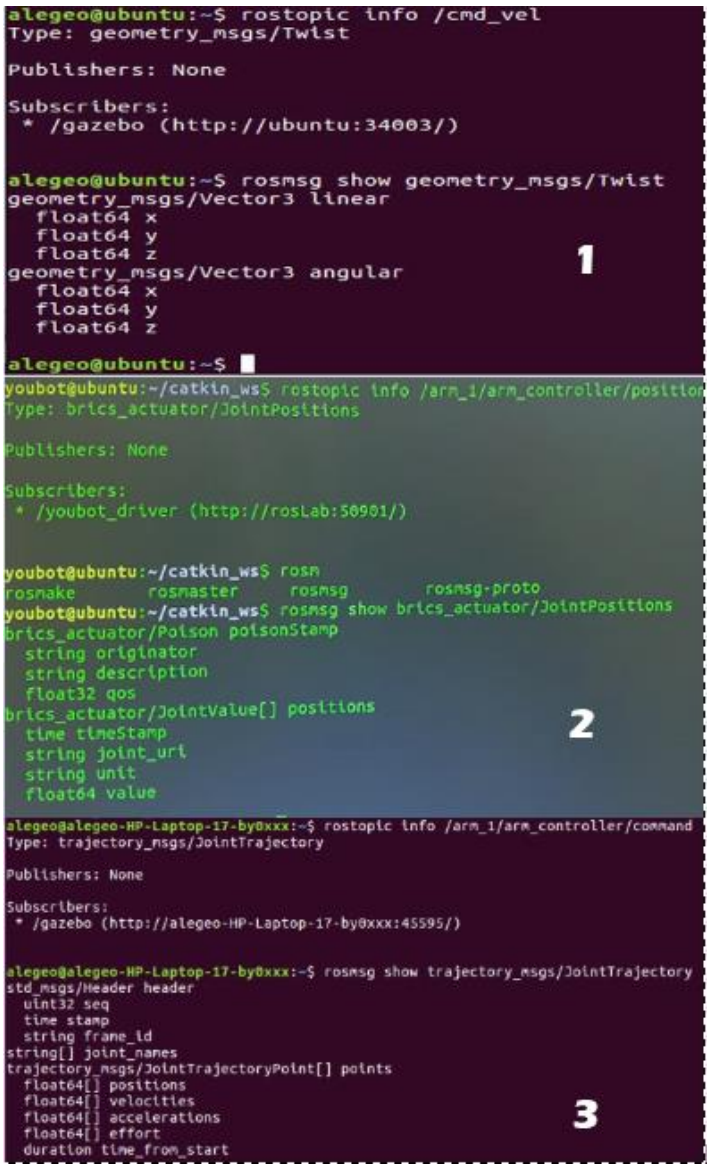


Figura 20: Tipo de temas usados en el robot real y simulado
Fuente: Investigador

3.2.18 Configuración del Software para Youbot dentro de Gazebo

Para el uso de Gazebo se usó Ubuntu 16.04 el cual es compatible con ROS Kinetic por lo cual se siguió las instrucciones para plataformas Ubuntu dentro de http://wiki.ros.org/kinetic/Installation/Ubuntu. Antes de iniciar con la instalación se debe configurar los repositorios de Ubuntu para permitir "multiverso" dentro de Software&Updates. Luego de debe configurar el computador para aceptar de

packages.ros.org los softwares necesarios, mediante el siguiente comando.

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

Para estar seguros de no obtener paquetes corruptos durante la instalación de debe configurar las llaves. Con estas llaves nos aseguramos de que la descarga es directamente desde packages.ros.org.

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Si se desea se puede agregar la sesión bash para cada vez que se lanza un nuevo Shell, así se agregan las variables de entorno ROS automáticamente.

```
echo "fuente /opt/ros/kinetic/setup.bash" >> ~ / .bashrc
fuente ~ / .bashrc
```

Se debe actualizar el índice de paquetes Debian y luego la instalación de ROS

```
sudo apt-get update
sudo apt-get install ros-kinetic-desktop-full
```

Se procede a instalar las dependencias para la construcción de paquetes

```
sudo apt install python-rosdep python-rosinstall python-rosinstall-
generator python-wstool build-essential
```

Finalmente, se inicializa rosdep para que nos permita instalar fácilmente dependencias del sistema.

```
sudo rosdep init
rosdep update
```

Con las configuraciones anteriores tenemos lista nuestra computadora para poder simular con ROS y Gazebo. Debido a que es un entorno simulado debemos descargar los paquetes necesarios para poder simular al robot Kuka Youbot. Para la simulación se clonó los paquetes directamente del repositorio de mas-group en github. Se escogió estos paquetes debido a que fueron los que no presentaron ningún tipo de error al ser ejecutado el nodo.

```
git clone https://github.com/mas-group/youbot_simulation.git
```

```
git clone https://github.com/mas-group/youbot_description.git
```

3.2.19 Software programado del Kuka Youbot en Python

El software programado del Youbot fue diseñado en el lenguaje de programación Python para mover al robot de forma inalámbrica por lo cual inicia llamando los contenedores ROS para los controladores Kuka Youbot necesarios para el control y los módulos necesarios para la comunicación inalámbrica mediante TCP socket.

```
#!/usr/bin/env python
import roslib
import rospy
from geometry_msgs.msg import Twist
import sys, signal
import socket
```

Se procede a determinar el host y el puerto con los que se realiza la comunicación, se crea un nuevo socket con valores por default, luego se establece la comunicación con el método bind que recibe una tupla con dos valores (host y puerto), por último establecemos la cantidad de peticiones que puede manejar en cola el socket con listen().

```
HOST = '192.168.43.85'
PORT = 33657
s = socket.socket()
s.bind((HOST, PORT))
s.listen(1)
```

Se crea variables tipo diccionario para poder acceder a los elementos necesarios mediante una clave para poder mover al robot, la clave son las letras enviadas desde el cliente. Los datos extraídos posteriormente se usan en el método correspondiente para realizar el movimiento requerido.

```
moveBindings = {
#                       x,y,relacion tetha
                    'i':(1,0,0),        # adelante
                    'o':(1,0,-1),       # adelante + rotación derecha
                    'j':(0,1,0),        # izquierda
                    'l':(0,-1,0),       # derecha
                    'u':(1,0,1),        # adelante + rotación derecha
```

```
                ',':(-1,0,0),       # atrás
                '.':(-1,0,-1),      # atrás + rotación derecha
                'm':(-1,0,1),       # atrás + rotación izquierda
                'v':(0,0,1),        # gira a la izquierda en su lugar
                'b':(0,0,-1),       # gira a la derecha en su lugar
            }
speedBindings={
                'q':(1.1,1.1),      # aumentar las velocidades
máximas en un 10%
                'z':(.9,.9),    # disminuir las velocidades máximas
en un 10%
                'w':(1.1,1),    # aumentar solo la velocidad lineal
en un 10%
                'x':(.9,1),     # disminuir solo la velocidad lineal
en un 10%
                'e':(1,1.1),    # aumentar solo la velocidad angular
en un 10%
                'c':(1,.9),     # disminuir solo la velocidad angular
en un 10%
            }

moveArm =   {
                'r':(5.80,1.05,-3.44,1.73,2.95),      # movimiento
específico
                't':(0.11,0.11,-0.11,0.11,0.11),    # movimiento a
home
            }
moveGripper =   {
                'f':(0.011),  # abre la pinza
                'g':(0),        # cierra la pinza
             }
```

Damos valores iniciales de velocidad y se crea un método para luego llamar en caso de querer aumentar la misma, se recomienda no variar la velocidad mucho ya que el robot cuenta con fuerza en sus movimientos y a la velocidad que se le da por defecto se mueve con la suficiente rapidez.

```
speed = 0.1
turn = 0.1
def vels(speed,turn):
        return "currently:\tspeed %s\tturn %s " % (speed,turn)
```

Por seguridad del robot se protege la programación con if_name_==”__main__” para evitar que partes del código se ejecuten al importarse módulos, los valores x,y,th inicializan en cero para que el robot no se mueva al ejecutar por primera vez el script, estos valores son cambiados luego al recibir información de cliente en la variable tipo

diccionario de Python moveBindings.

```
if __name__=="__main__":
        settings = termios.tcgetattr(sys.stdin)
        pub = rospy.Publisher('cmd_vel', Twist)
        rospy.init_node('teleop_twist_keyboard')
        x = 0
        y = 0
        th = 0
        status = 0
```

Se inicia dos ciclos infinitos, el primero se encarga de identificar si existe peticiones del cliente y cumplirlas mientras que el segundo acepta las peticiones que el cliente envíe, la información enviada se guarda en la variable content.

```
try:
        while(1):
                client, addr = s.accept()
                while True:
                        content = client.recv(1024)
                        if len(content)==0:
                                break
                        else:
                                key = content
                client.close()
```

Al llegar información desde el cliente se identifica donde se encuentra el movimiento deseado con el comando if en cascada y se asigna los valores respectivos a cada variable, caso contrario se regresa a valores iniciales para evitar fallos.

```
if key in moveBindings.keys():
        x = moveBindings[key][0]
        y = moveBindings[key][1]
        th = moveBindings[key][2]
elif key in speedBindings.keys():
        speed = speed * speedBindings[key][0]
        turn = turn * speedBindings[key][1]
        if (status == 14):
        status = (status + 1) % 15
elif key in moveArm.keys():
        join0 = moveArm[key][0]
        join1 = moveArm[key][1]
        join2 = moveArm[key][2]
        join3 = moveArm[key][3]
        join4 = moveArm[key][4]
elif key in moveGripper.keys():
```

```
        gripp = moveGripper[key][0]
else:
        x = 0
        y = 0
        th = 0
        join0 = 0.11
        join0 = 0.11
        join0 = -0.11
        join0 = 0.11
        join0 = 0.11
        gripp = 0
        if (key == '\x03'):
                break
```

3.2.20 Simuladores de circuitos electrónicos para diseño de placas PCB

Entre los programas más usados para el diseño de circuitos electrónicos y placas PCB podemos encontrar a Eagle, Proteus y EasyEDA. Estos softwares de diseño PCB son usados de forma profesional para generar archivos gerber con la información necesaria para crear los circuitos impresos con máquinas CNC especializadas para obtener placas PCB con exactitud milimétrica y acabados profesionales.

3.2.21 Diferencias entre simuladores de circuitos electrónicos

Para empezar con el diseño de la placa PCB de control se tomó en cuenta las principales características de los softwares de diseño presentadas en la siguiente Tabla 11 comparativa.

Tabla 11: Diferencias entre softwares de diseño electrónico

Características	Proteus	Eagle	EasyEDA
Diseño Online	No	No	si
Variedad de los elementos básicos	Si	Si	si
Capacidad de diseño electrónico con alta variedad de elementos	Si, se debe crear o descargar paquetes o	Si, se debe crear o descargar paquetes o	Si, se puede crear pero al ser una plataforma

electrónicos del mercado	librerías para ciertos elementos como placas, módulos o sensores	librerías para ciertos elementos como placas, módulos o sensores	online no es necesario buscar para descargar elementos diseñados por otros usuarios
Vínculo directo para trabajar con elementos disponibles en el mercado y finalmente comprarlos	No	No	si
Vinculo directo para enviar a fabricar la placa PCB diseñada	no	No	si
Simulación de los diseños electrónicos	Si	Si	Si

3.2.22 Simulador utilizado para la creación del circuito electrónico

El diseño del controlador fue creado en la plataforma online EasyEDA debido a las ventajas presentadas en la Tabla 11, principalmente por la interfaz amigable, la facilidad de trabajar directamente con elementos disponibles en el mercado y la capacidad de poder enviar a fabricar placas profesionales a un precio accesible.

Figura 21: Diseño 3D en EasyEDA y la placa real soldado todos los elementos
Fuente: Investigador (EasyEDA)

Esta plataforma online tiene la capacidad para montar un diseño 3D de la placa diseñada como se puede ver en la Figura 21. Existe una variedad amplia de modelos 3D que se pueden vincular con los elementos electrónicos utilizados para tener una perspectiva del resultado final al crear la placa PCB.

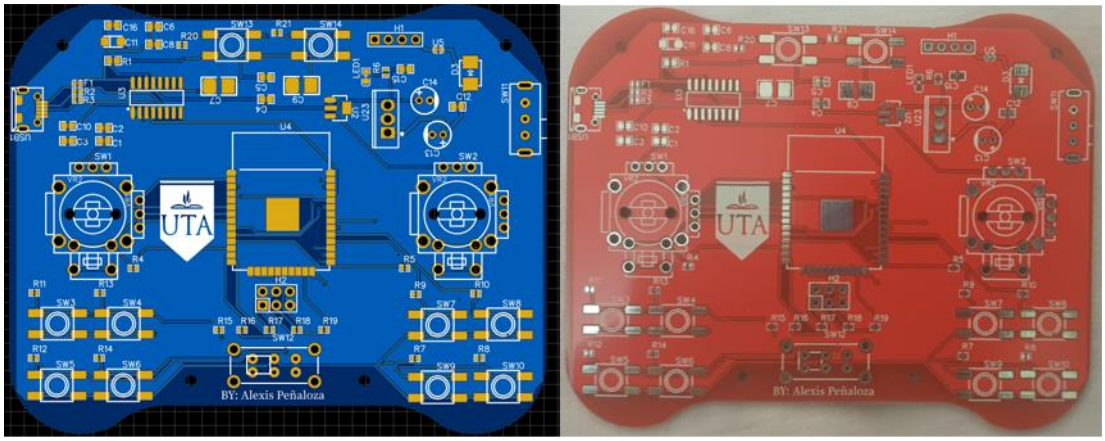


Figura 22: Modelado 2D en EasyEDA y la placa real sin soldar los elementos
Fuente: Investigador (EasyEDA)

3.2.23 Diagrama de pistas de la placa PCB

Las pistas fueron dimensionadas teniendo en cuenta el espesor con el que la fábrica JLCPCB proporciona para poder hacer el pedido de placas PCB. Esta fábrica da la opción de escoger entre 1 $\frac{oz}{ft^2}$ y 2 $\frac{oz}{ft^2}$ para el espesor entre las capas superior e inferior, teniendo en cuenta que la corriente máxima entre los pines del microcontrolador y dispositivos de entrada es de 80mA, la temperatura máxima con la cual se va a operar el controlador es a temperatura ambiente y solo se cuenta con dos capas en el diseño

se usó las siguientes fórmulas para determinar un ancho óptimo de las pistas.

$$Ancho = \frac{Área}{1.378\ Grosor} \tag{1}$$

$$Área = \left[\frac{I}{k_1 \Delta T^{k_2}}\right]^{\frac{1}{k_3}} \tag{2}$$

En donde:

I: Corriente máxima
$Área$: área de la pista entre el espesor y ancho de la pista
ΔT: diferencia de temperatura
$Grosor$: espesor de la pista
$Ancho$: ancho de la pista
$k_1,\ k_2, k_3$: constantes dependientes del lugar de la pista (interna o externa)

Debido que solo existe dos capas en el diseño (superior e inferior) los valores de las constantes son los siguientes: k_1=0.0647, k_2 =0.4281, k_3 =0.6732. La temperatura máxima de operabilidad del microcontrolador según su datasheet es 85ºC y la corriente máxima en los pines de entrada y salida es de 80mA.

$$Área = \left[\frac{80x10^{-3}}{0.0647(85-25)^{0.4281}}\right]^{\frac{1}{0.6732}}$$

$$Área = \left[\frac{80x10^{-3}}{0.0647(60)^{0.4281}}\right]^{1.485442662}$$

$$Área = [0.0028\ ]^{\frac{1}{0.6732}}$$

$$Área = 0.1014327471$$

$$Ancho = \frac{0.17699}{1.378 * 1}$$

$$Ancho = 0.073\ th = 0.002mm$$

Dado que para las pistas en las entradas y salidas del microcontrolador se necesita un grosor mínimo se decidió crear con un grosor estándar de 0.26mm y así en un futuro poder construir la placa de forma artesanal.

Para las pistas en el área de potencia se utiliza un máximo de 3ª por lo cual el grosor es el siguiente:

$$Área = \left[\frac{3}{0.0647(85-25)^{0.4281}}\right]^{\frac{1}{0.6732}}$$

$$Área = 22.09588$$

$$Ancho = \frac{22.09588}{1.378 * 1}$$

$$Ancho = 16.034\ th = 0.4072mm$$

Con estos datos se determinó aumentar un 35% el grosor de la pista para evitar la separación del cobre en la placa por altas temperaturas como medida preventiva. El ancho final de las pistas en el área de potencia fue de 0.55mm.

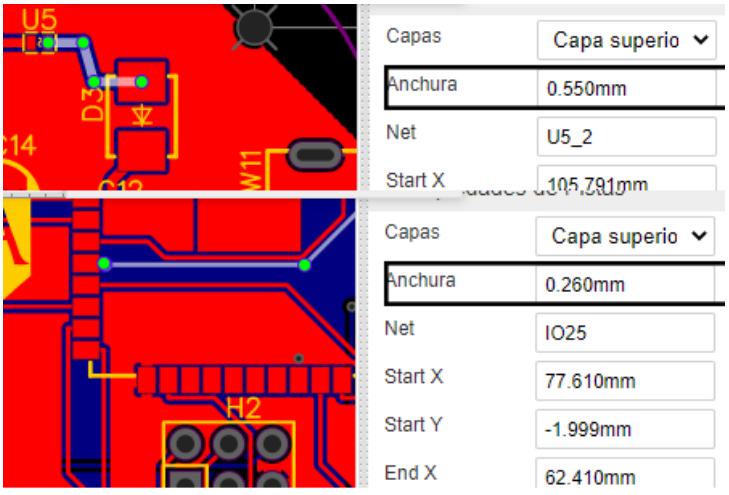


Figura 23: Grosor de las pistas utilizadas en la placa
Fuente: Investigador (EasyEDA)

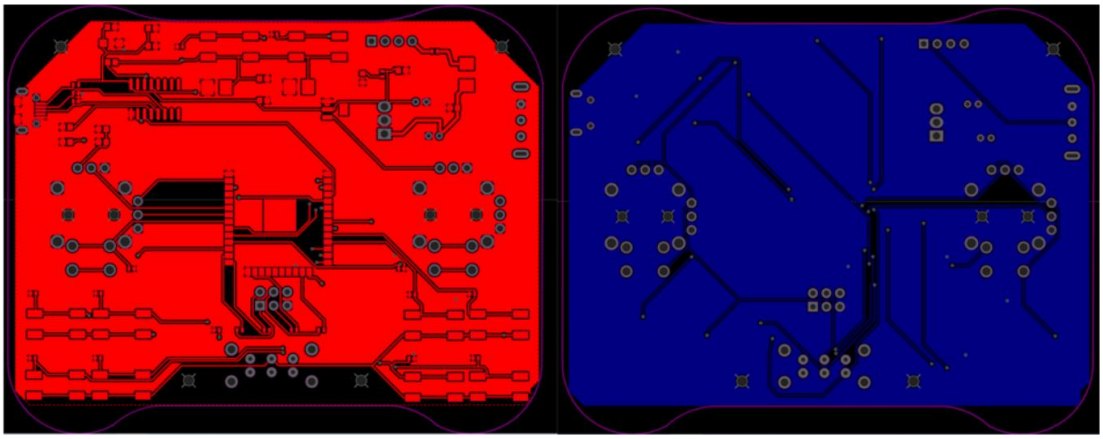


Figura 24: Diagramas de pistas superior e inferior
Fuente: Investigador (EasyEDA)

3.2.24 Microcontroladores con comunicación wifi

Espressif es una empresa multinacional de semiconductores encargada del desarrollo de soluciones AIoT de bajo consumo con conectividad Bluetooth y Wi-Fi, la empresa se dedica a la creación de chips, módulos y placas de desarrollo ESP8266 y ESP32. [52]

ESP8266 y ESP32 son los microprocesadores con mayor uso para la comunicación inalámbrica con aplicaciones muy exigentes como la transmisión de música, codificación y decodificación de voz. Se puede apreciar las diferencias entre los dos microprocesadores en la Tabla 12.

Tabla 12: Diferencias entre microcontroladores [53] [54]

Características	ESP32	ESP8266	ATmega2560	Teensy 3.2
Alimentación	3-3.6V	2.5-3.6V	5V	3.6-6V
Número de Bits	32	32	-	32
SRAM	520 KB	50 KB	8KB	64KB
ROM	448 KB	-	4KB	256KB

Procesador	Tensilica LX6	Tensilica L106	-	RM Cortex-M4
Número de núcleos	Dual Core	Single Core	-	-
Temperatura que soporta	-40ºC~+85°C	-40ºC~+125°C	-	-
Corriente de operabilidad	80mA		50mA	80mA
Pines GPIO utilizables	32	16	54	34
ADC	12	1	16	21
DAC	2	No	14	12
WiFi	Si		No	No
Bluetooth	Si	No	No	No

3.2.25 Microcontrolador utilizado en la placa de control

Después de investigar entre los microcontroladores más usados y entre ellos los que cuentan con comunicación WiFi expuesto en la Tabla 12, para la facilidad de no usar módulos extra en la conectividad inalámbrica y debido que se requiere el uso de 2 joysticks, los mismo necesitan 2 entras análogas cada uno se decidió usar el microcontrolador ESP32 ya que cuenta con 12 entradas ADC y conectividad directa mediante WiFi y Bluetooth a diferencia del ESP8266 que solo tiene una entrada análoga.

3.2.26 Placa Electrónica de Control Wifi

La placa electrónica de control wifi es la encargada de administrar el envío de

información hacia el robot Kuka Youbot con la ayuda de componentes análogos y digitales. La placa cuenta con un microcontrolador para identificar la información enviada por los componentes y poder reenviarla de forma inalámbrica al robot.

Esquema del Núcleo de Control y Comunicación ICSP y USB

El núcleo de control está formado por un microcontrolador ESP32-WROOM con dos núcleos de CPU que se pueden controlar individualmente, además cuenta con comunicación inalámbrica de WIFI-BT-BLE. La codificación puede ser ingresada al MCU de dos formas: La primera es conectando un programador serie ICSP (In Chip Serial Programmer) directamente a los pines MTMS, MTDI, MTCK y MTDO que son los IO14, IO12, IO13 y IO15 respectivamente. La segunda forma es con ayuda del integrado CH340C que es un chip de conversión bus USB a interfaz serie con la conexión de los pines TXD0 y RXD0, como se visualiza en la Figura 25.

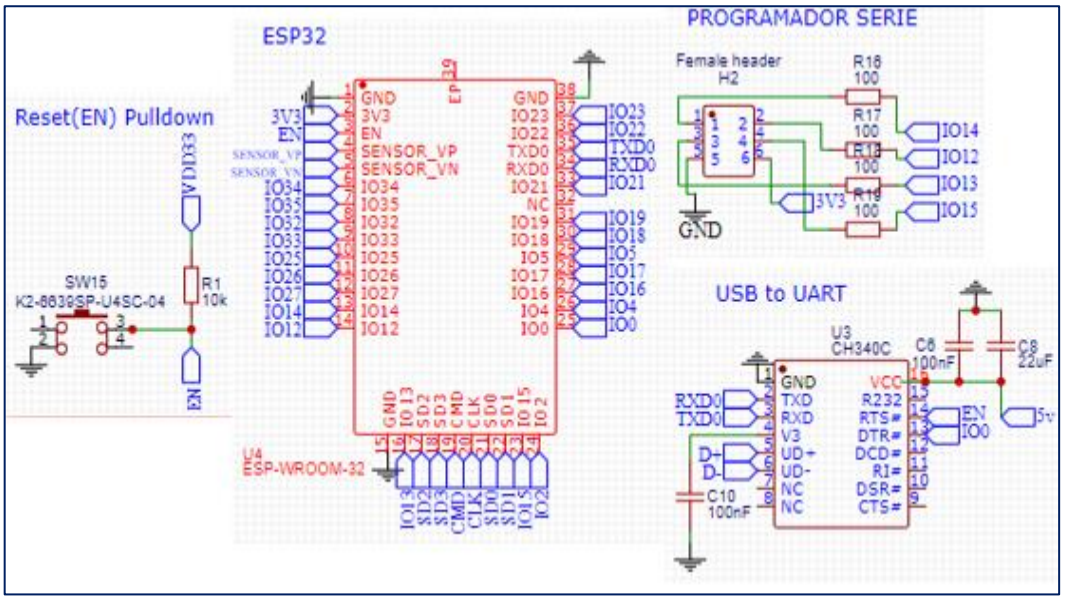


Figura 25: Esquema de las conexiones del núcleo de control
Fuente: Investigador (EasyEDA)

Varistores de protección

Los varistores de protección son usados comúnmente para proteger los circuitos contra variaciones de voltaje, bloqueando el paso de corriente al activarse y así poder proteger componentes sensibles.

La entrada USB del circuito cuenta con 2 varistores Panasonic EZJZ0V500AA para proteger la entrada de información de variaciones de tensión conectados a Tx y Rx, además de un pequeño fisible 0402L050SLKR de 6V/1A para poder alimentar todo el circuito con una fuente USB sin riesgos a dañar la placa como se muestra en la Figura 26.

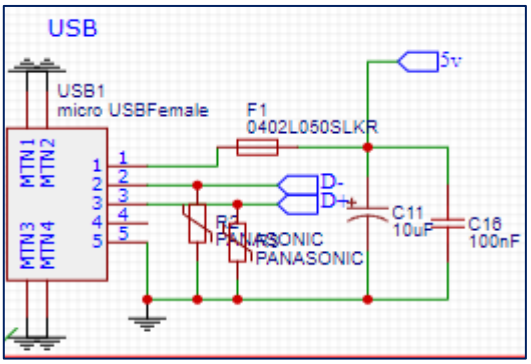


Figura 26: Esquema de entrada USB
Fuente: Investigador (EasyEDA)

Reguladores de voltaje disponibles

Los reguladores de voltaje son los encargados de recibir un voltaje y reducirlo al valor necesario según su diseño, estos dispositivos cuentan con 3 pines: GND, voltaje de entrada y voltaje de salida. Entre los reguladores de voltaje disponibles en el mercado se encuentran los de 3.3V, 5V, 9V y 12V, teniendo dicho voltaje en su pin de salida

Reguladores de voltaje utilizados

El esquemático diseñado para el controlador tiene un voltaje de funcionamiento de 3.3V debido al voltaje de operabilidad y voltaje de entrada en los pines del microcontrolador ESP32 utilizado. Para poder alimentar el sistema con un una batería de hasta 35V se necesitó el uso de dos reguladores conectados en cascada y así evitar que el regulador final de 3.3V sufra sobrecalentamiento, Holtek Semicon HT7333 es un regulador tipo smd para regular 3.3V y L7805CV-DG es un regulador de 5V capaz de soportar hasta 35V, en este caso el regulador principal se usó con un encapsulado TO0220 para evitar sobrecalentamiento y poder adicionar un disipador de calor con mayor facilidad en su estructura.

Esquema del Circuito de Potencia

El circuito de potencia para alimentar el circuito completo está diseñado para soportar

35V/1A ya que cuenta con un regulador de voltaje L7805CV-DG y como medidas de protección en la entrada del regulador cuenta con un fusible 0432003.KRHF capaz de soportar 3A, seguido de un diodo rectificador VS-10BQ030-M3/5BT para evitar daños al circuito en caso de conexión inversa en la polaridad de la batería como se puede visualizar en la Figura 27.

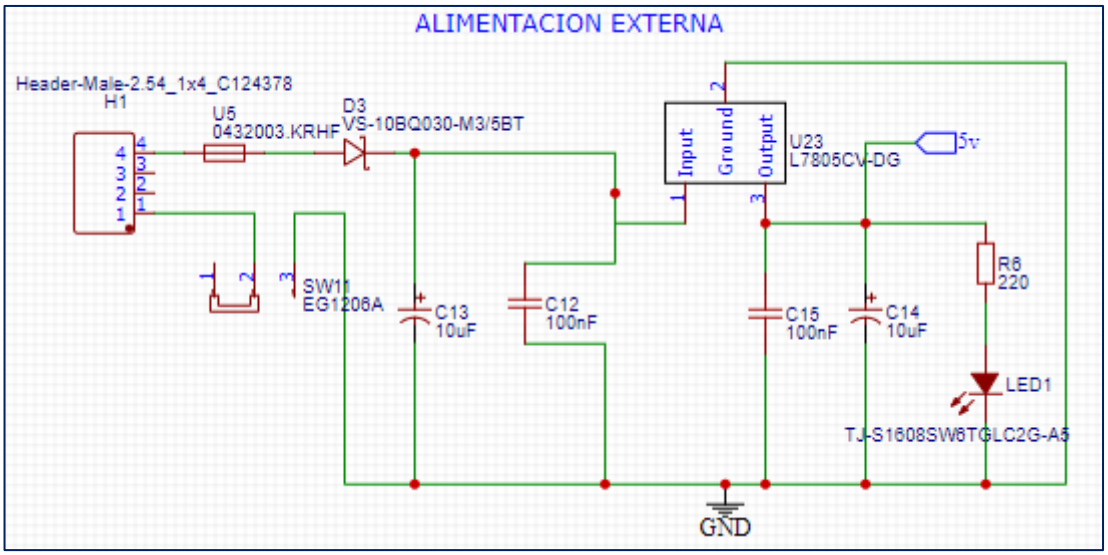


Figura 27: Esquema de Alimentación Externa
Fuente: Investigador (EasyEDA)

La alimentación del MCU debe ser de 3.3V, además los pines de entrada y salida del ESP32-WROOM soportan 3.3V por lo cual se implementó un regulador de voltaje Holtek Semicon HT7333 (Figura 28). El acondicionamiento de energía se diseñó para no dañar el ESP32-WROOM con sobretensiones debido a que tiene como un máximo de voltaje de 3.6V, lo cual es un rango muy estrecho entre el voltaje de operabilidad y el máximo soportable.

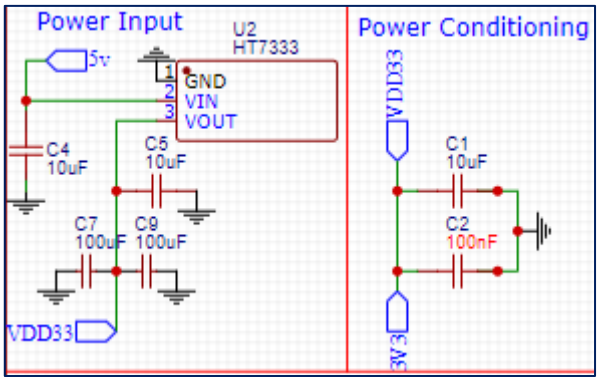


Figura 28: Esquema de 3.3V
Fuente: Investigador (EasyEDA)

Esquema de Entradas Digitales y Análogas

Los dispositivos de entrada análogos y digitales están alimentados directamente a la fuente de 3.3V explicado en el esquema anteriormente mencionado, las 4 entradas análogas están conectadas a los pines ADC disponibles mientras que todos los pines utilizables fueron conectados con pulsadores y un switch (Figura 29) para poder aprovechar al máximo el control de los periféricos del dispositivo a controlar.

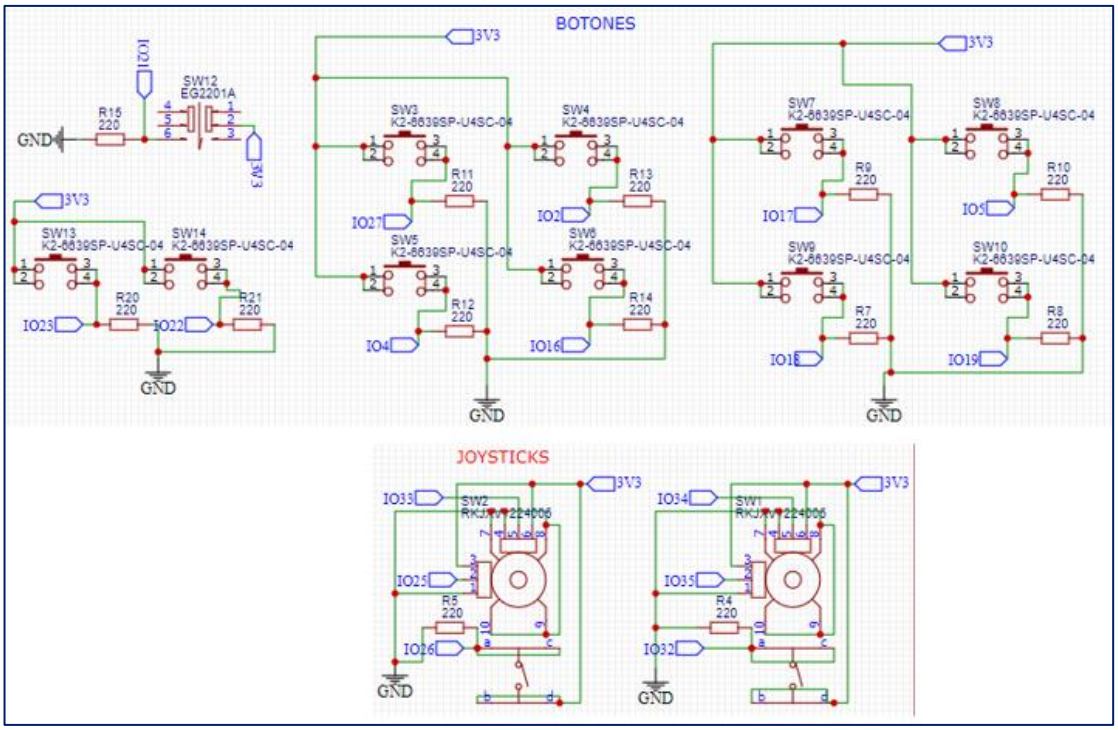


Figura 29: Esquema de entradas Digitales y Análogas
Fuente: Investigador (EasyEDA)

En la Figura 30 se muestra el diagrama de bloques de la placa electrónica diseñada, el cual cuenta con todas las funciones que realiza el sistema para su correcto funcionamiento, entre los cuales encontramos los reguladores de voltaje conectados en cascada con un acondicionamiento final de 3.3V, las entradas análogas y digitales se pueden accionar gracias a este acondicionamiento debido que los pines del microcontrolador se queman en caso de recibir un voltaje mayor a 3.6V, el bloque central del microcontrolador es el encargado de recibir los datos enviados por el operador al momento de activar los pulsadores y joysticks, luego procesa la información y envía al robot los datos para ejecutar las ordenes mediante WiFi

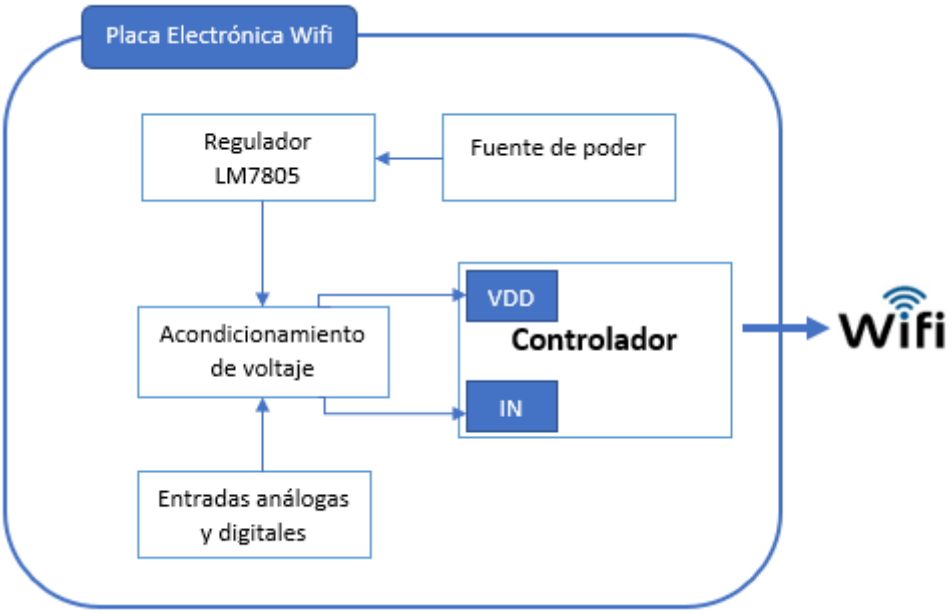


Figura 30: Diagrama de Bloques del Módulo Wifi
Fuente: Investigador

En la Tabla 13 se muestra los pines a los cuales se encuentran conectados los botones y joysticks para poder modificar el código fuente.

Tabla 13: Pines usados del ESP-32

Localización	Nombre ESP-32	Ping	Función
Pulsador superior-izquierda	IO23	37	GPIO23, VSPID, HS1_STROBE
Pulsador superior-	IO22	36	GPIO22, VSPIWP,

derecha			U0RTS, EMAC_TXD1
Pulsador superior-izquierda (conjunto de botones izquierda)	IO27	12	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
Pulsador superior-derecha (conjunto de botones izquierda)	IO2	24	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
Pulsador inferior-izquierda (conjunto de botones izquierda)	IO4	26	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
Pulsador inferior - derecha (conjunto de botones izquierda)	IO16	27	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
Pulsador superior-izquierda (conjunto de botones derecha)	IO17	28	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
Pulsador superior-derecha (conjunto de botones derecha)	IO5	29	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
Pulsador inferior-izquierda (conjunto de botones derecha)	IO18	30	GPIO18, VSPICLK, HS1_DATA7
Pulsador inferior - derecha (conjunto de botones derecha)	IO19	31	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
Interruptor central-	IO21	33	GPIO21, VSPIHD, EMAC_TX_EN

inferior			
Joystick derecho-arriba/abajo	IO35	7	GPIO35, ADC1_CH7, RTC_GPIO5
Joystick derecho-izquierda/derecha	IO33	9	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
Joystick derecho pulsador	IO32	8	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
Joystick izquierdo - arriba/abajo	IO25	10	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
Joystick izquierdo-izquierda/derecha	IO34	6	GPIO34, ADC1_CH6, RTC_GPIO4
Joystick izquierdo pulsador	IO26	11	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1

3.2.27 Código de Control

Para este sistema se ha creado 3 códigos de programación debido que dentro de la simulación de Gazebo el controlador del brazo de Kuka Youbot opera con un tema distinto al robot real y la publicación que se debe realizar para poder mover el brazo son distintas.

Código del Kuka Youbot Real

El robot Kuka Youbot real cuenta con un sistema operativo Ubuntu 12.04 junto con ROS Hydro, se puede controlar con la creación de un script dentro del ROS Package

designado a usarse, el script creado fue programado en el lenguaje de programación Python usando la librería Twist para el movimiento de la plataforma móvil de forma libre en sus partes lineales y angulares, para el movimiento del brazo se usó la librería JointPositions para posicionar el brazo en una dirección requerida con anterioridad. Además de la librería socket para la comunicación inalámbrica.

Código del Kuka Youbot en Gazebo

El robot Kuka Youbot simulado en Gazebo cuenta con un sistema operativo Ubuntu 16.04 junto con ROS Kinetic, se puede controlar con la creación de un script dentro del ROS Package designado a usarse, el script creado fue programado en el lenguaje de programación Python usando la librería Twist para el movimiento de la plataforma móvil de forma libre en sus partes lineales y angulares, para el movimiento del brazo se usó la librería JointTrajectory para posicionar el brazo en una dirección requerida con anterioridad. Además de la librería socket para la comunicación inalámbrica.

Código del Controlador

La codificación del controlador utiliza entradas digitales y análogas para enviar sus datos mediante el protocolo TCP con ayuda de sockets en los extremos de la comunicación por lo cual se creó en el entorno de programación ESP-IDE, la cual es un software desarrollado por Espressif siendo el marco de desarrollo oficial para los microcontroladores de series ESP32 y así aprovechar al máximo al MCU.

3.3 Presupuesto

El presupuesto para realizar este proyecto fue proporcionado directamente por parte del investigador y la logística escogida para exportar los materiales tipo smd fue hecha mediante triangulación en España con un courrier más económico que con envío directo, debido a la crisis de salud presentada a nivel mundial inhabilitando la llegada del correo nacional.

ID	DESCRIPCIÓN	UNIDAD	CANTIDAD	PRECIO UNITARIO	TOTAL
1	Capacitor cerámico de 10uF SMD	c/u	3	$0,10	$0,30
2	Capacitor cerámico de 100nF SMD	c/u	6	$0,02	$0,12

3	Capacitor cerámico de 22uF SMD	c/u	1	$0,30	$0,30
4	Capacitor cerámico de 100uF SMD	c/u	2	$0,35	$0,70
5	Capacitor dieléctrico 10uF SMD	c/u	1	$0,70	$0,70
6	Capacitor dieléctrico de 10uF	c/u	2	$0,22	$0,44
7	Diodo Rectificador VS-10BQ030-M3/5BT SMD	c/u	1	$0,20	$0,20
8	Fusible 0402L050SLKR SMD	c/u	1	$0,18	$0,18
9	Conector Macho-2.54_1x4	c/u	1	$0,10	$0,10
10	Conector Hembra-2,54_2x3	c/u	1	$0,10	$0,10
11	Diodo Led emisor de luz verde SMD	c/u	1	$0,05	$0,05
12	Resistencia 10kΩ ohm SMD	c/u	1	$0,08	$0,08
13	Varistor PANASONIC SMD	c/u	2	$0,50	$1,00
14	Resistencia de 220Ω ohm SMD	c/u	14	$0,20	$2,80
15	Resistencia de 100Ω ohm SMD	c/u	4	$0,20	$0,80
16	Joystick RKJXV1224005	c/u	2	$2,80	$5,60
17	Botón Pulsador SMD	c/u	11	$0,11	$1,21
18	Interruptor de energía EG1206A	c/u	1	$0,64	$0,64
19	Interruptor EG2201A	c/u	1	$0,77	$0,77
20	Regulador de voltaje 3,3V HT7333 SMD	c/u	1	$0,38	$0,38
21	Integrado CH340C SMD	c/u	1	$0,77	$0,77
22	ESP-WROOM-32 SMD	c/u	1	$7,31	$7,31
23	Fusible 3A 0432003.KRHF SMD	c/u	1	$0,51	$0,51
24	Regulador de voltaje 5V L7805CV-DG	c/u	1	$0,25	$0,25
25	Conector micro USB-Hembra SMD	c/u	1	$0,27	$0,27

26	PCB circuito impreso	c/u	1	$5,00	$5,00
27	Batería Lipo Turnigy 7,4V / 300mAh	c/u	1	$11,99	$11,99
28	Corte laser de la carcasa	Hora	2	$8,00	$16,00
				SUBTOTAL	$58,57
				DISEÑO DEL PROTOTIPO	$100,00
				GASTOS DE ENVIO DE MATERIALES	$55,00
				TOTAL	$213,57

3.4 ANÁLISIS Y DISCUSIÓN DE RESULTADOS

3.4.1 Pruebas de envío de datos al robot

Para verificar que los datos lleguen correctamente se implementó un script programado en Python con socket TCP en modo servidor y así conectar el control en modo cliente, los datos que ingresan al servidor son de tipo chart para aprovechar el uso de condicionales dentro de la programación del nodo publicador. En la Figura 31 se puede apreciar los datos recibidos desde el control que fueron impresos en consola para facilitar al control de errores.

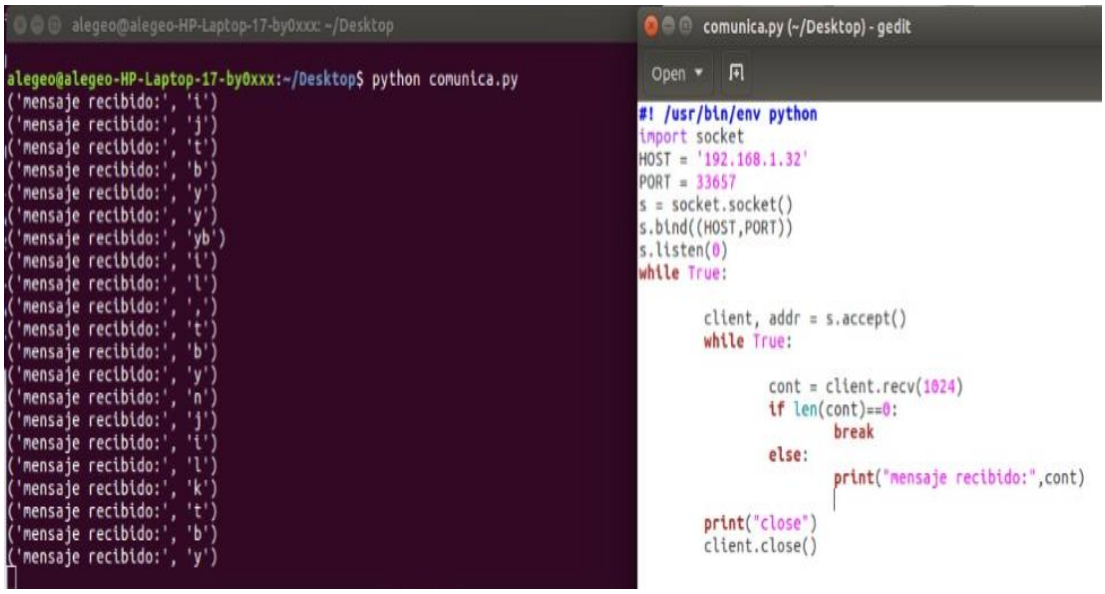


Figura 31: Pruebas de comunicación entre el control y Ubuntu
Fuente: Investigador

3.4.2 Prueba de ejecución de datos recibidos para el movimiento de la plataforma Youbot simulada en Gazebo

La prueba de movimiento de la plataforma se realizó imprimiendo los valores de “x” y “y” después de ejecutar el script anterior mostrado en la Figura 32, este script se añadió al ejemplo teleop_twist_keyboard.py y cambiando la elección de valores mediante los datos recibidos en vez de la lectura del teclado.

La plataforma se puede modificar el modo de desplazamiento con el cambio de valores en los ejes x,y,z con valores combinados 1/-1 y así lograr movimientos diagonales, laterales y rotación en el origen.

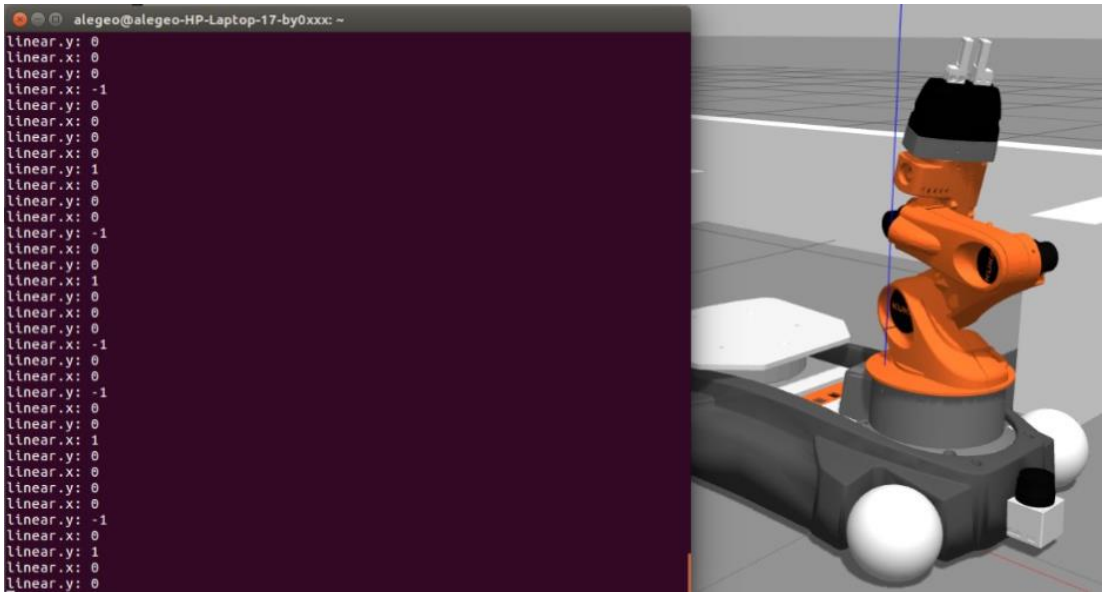


Figura 32: Pruebas de movimiento de la plataforma Youbot en Gazebo
Fuente: Investigador

3.4.3 Prueba de ejecución de datos recibidos para el movimiento de brazo y pinza Youbot simulados en Gazebo

Las pruebas necesarias se realizaron con el envío de datos encapsulados en un vector desde el nodo publicador hacia el nodo en gazebo mediante los temas necesarios para el brazo y la pinza, además de un nodo suscriptor para visualizar los datos recibidos correctamente y los movimientos ejecutados por el robot dentro del simulador.

Las primeras pruebas para identificar una correcta codificación de envió de datos por el tema que utiliza el brazo Youbot se realizó con el control mediante el teclado del computador, siendo semejante al ejemplo youbot_keyboard_teleop.py con la plataforma. Una vez controlado el robot por completo por el teclado se procedió a cambiar la recepción de datos mediante teclado por comunicación socket como se puede ver en la Figura 33.

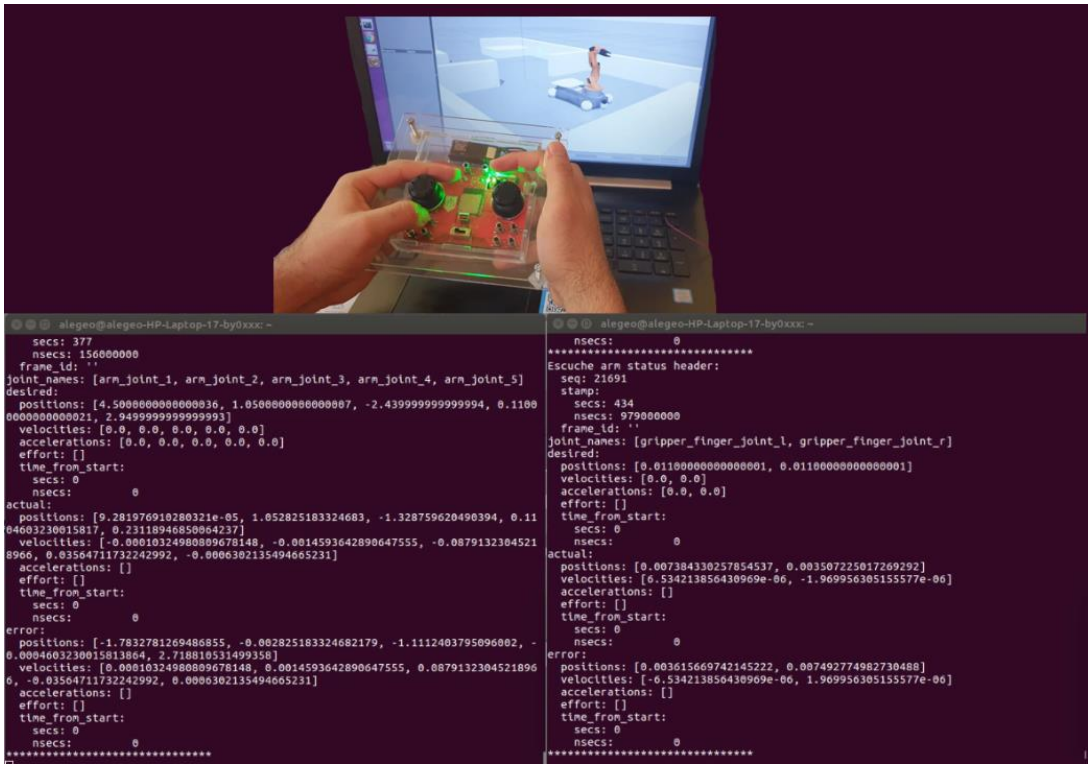


Figura 33: Prueba de movimiento del brazo y pinza del robot simulado
Fuente: Investigador

3.4.4 Script creado para identificar los datos recibidos

Se codificó un archivo Python para identificar los valores actuales del brazo y pinza. La configuración del archivo consiste en suscribirse a los temas necesarios, en este caso los del brazo y pinza terminados en “state”. Se imprimen los valores obtenidos y finalmente una fila de asteriscos para separar los valores nuevos.

```python
#!/usr/bin/env python
from std_msgs.msg import String
from control_msgs.msg import JointTrajectoryControllerState
from trajectory_msgs.msg import JointTrajectory
import roslib
import rospy
import trajectory_msgs.msg as tm

def callback_Arm(data):
        print("Escuche arm status "+str(data)+"\n******************************")

def listener():
        rospy.init_node('listener',anonymous=True)
        rospy.Subscriber("/arm_1/arm_controller/state", JointTrajectoryControllerState, callback_Arm)
        #rospy.Subscriber("/arm_1/gripper_controller/state", JointTrajectoryControllerState, callback_Arm)
        rospy.spin()

if __name__ == '__main__':
        listener()
```

Figura 34: Script suscriptor
Fuente: Investigador

3.4.5 Prueba de ejecución de datos recibidos para el movimiento de la plataforma Youbot del robot real

Las pruebas de ejecución de los datos recibidos se realizaron de manera directa con la visualización del movimiento del robot ya que en la simulación se logró corregir las fallas de comunicación y en comparación con el robot simulado se utiliza el mismo tipo de tema para el envío de datos desde el nodo publicador.

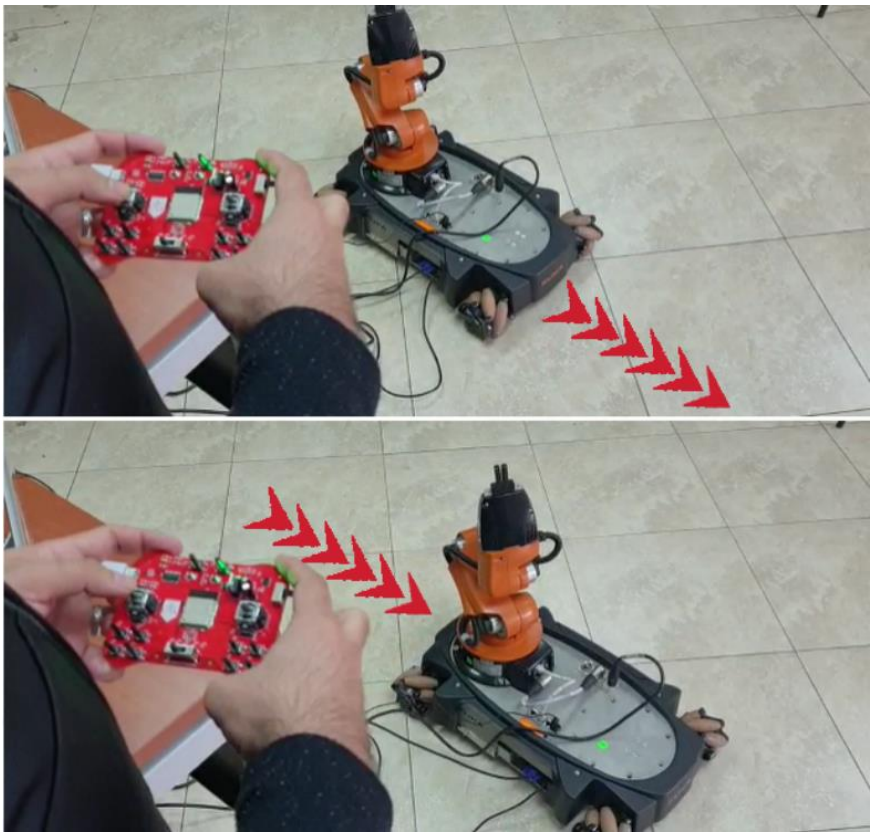


Figura 35: Pruebas de movimiento de la plataforma Youbot
Fuente: Investigador

3.4.6 Prueba de ejecución de datos recibidos para el movimiento del brazo y pinza Youbot del robot real

Para iniciar con las pruebas de control con el brazo y pinza Youbot se procedió a identificar el tipo de tema y después a identificar el tipo de mensaje ROS a enviarse y así poder hacer el arreglo de vectores dentro del vector final "positions" que nos pide el tema.

Para poder identificar los valores necesarios y tipo de mensaje a enviarse se debe seguir los siguientes pasos:

1. rostopic list - > nos muestra todos los temas activos
2. rostopic info TEMA - > imprime la información sobre el tema consultado
3. rosmsg show TIPO DE TEMA - > muestra información sobre el tipo de mensaje ros

El tema con el que opera el robot real se puede observar en la Figura 36. Con el tema necesario se procedió a configurar el arreglo de vectores para enviar el mensaje ROS y modificar el ejemplo teleop_twist_keyboard.py para poder tener movimiento del brazo y pinza con ayuda del teclado, finalmente se ajustó la comunicación WiFi al igual que con el robot simulado.

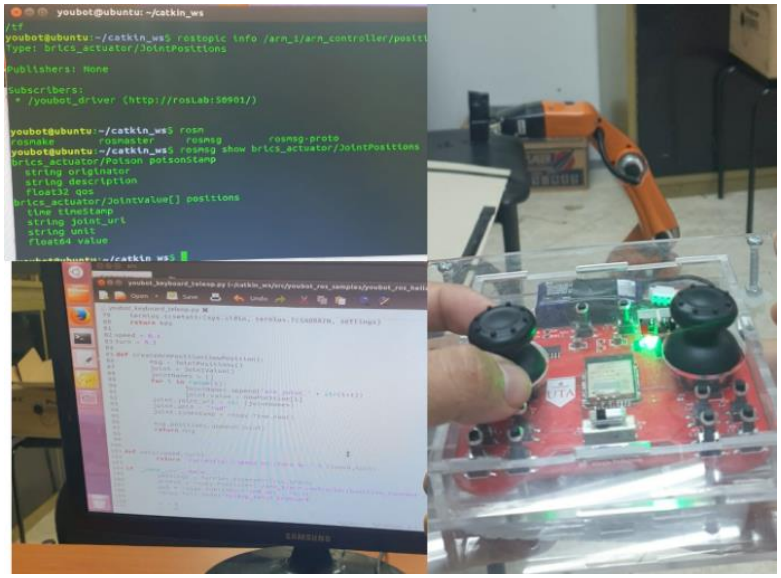


Figura 36: Prueba de movimiento del brazo y pinza del robot Youbot
Fuente: Investigador

El proyecto concluido alcanzó a cumplir los objetivos planteados con la fabricación del control remoto inalámbrico elaborado con un microprocesador ESP32 para la manipulación libre del robot Kuka Youbot real y simulado en gazebo, este proyecto cuenta con una amplia gama de aplicaciones futuras ya que el nodo receptor en ROS puede ser programado para ejecutar acciones o trayectos completos al accionar los periféricos del controlador, además el controlador al ser una placa con periféricos de entrada se pueden cambiar sensores según sean las necesidades futuras. Este dispositivo tiene la facilidad para operar con alimentación mediante USB o baterías de hasta 35V.

3.4.7 Trayectorias predeterminadas controladas a ejecutar mediante WiFi

Se pre-programó una ruta completa a ejecutarse mediante la activación de un botón en el controlador, gracias a la programación hecha mediante métodos para el control por separado de la pinza, brazo y plataforma, siguiendo la misma configuración para el control independiente mediante el controlador explicado anteriormente se activó aun secuencia para repetirse cada ves que se aplaste un botón en específico del controlador wifi.

```
if traje= 1:
        x = 1
        y = 0
        th=0
        twist.linear.x = x*speed
        twist.linear.y = y*speed
        twist.angular.z = th*turn
        x = 0
        y = 1
        twist.linear.x = x*speed
        twist.linear.y = y*speed
        twist.angular.z = th*turn
        x = 1
        y = 0
        twist.linear.x = x*speed
        twist.linear.y = y*speed
        twist.angular.z = th*turn
        grip = 0.011
        msg = createGripperPositionCommand(grip)
        jointvalues = [4.5, 1.05, -2.44, 0.11, 2.95]
        msgd = createArmPositionCommand(jointvalues)
        grip = 0.0
        msg = createGripperPositionCommand(grip)
        jointvalues = [0.11, 0.11, -0.11, 0.11, 0.11]
        x = -1
        y = 0
        th=0
        twist.linear.x = x*speed
        twist.linear.y = y*speed
        twist.angular.z = th*turn
        x = 0
        y = -1
        th=0
        twist.linear.x = x*speed
        twist.linear.y = y*speed
        twist.angular.z = th*turn
        x = -1
        y = 0
        th=0
        twist.linear.x = x*speed
        twist.linear.y = y*speed
        twist.angular.z = th*turn
```

Figura 37: Trayectoria pre-programada del robot Kuka Youbot
Fuente: Investigador

3.4.8 Trayectoria predeterminada controlada a ejecutar mediante teclado del computador

Con la ayuda de “len” se puede identificar el número de elementos que contiene un objeto por lo cual se implementó una selección con el condicional “if” para controlar la entrada de datos a todas las opciones pre-configuradas mediante el controlador WiFi y el teclado del computador como se puede visualizar en la Figura 38. Gracias a esta selección de puede tener un control total del robot para su funcionamiento independiente o con la activación de las secuencias pre-configuradas.

```
client, addr = s.accept()
while True:
    cont = client.recv(1024)
    if len(cont)==0:
        key = getKey()
    else:
        key = cont
```

Figura 38: Selección de ordenes mediante teclado y wifi
Fuente: Investigador

3.4.9 Trayectoria automática evitando colisionar con objetos

Con el sensor laser en el robot simulado y la librería LaserScan se implementó la configuración automática para detectar objetos y poder esquivarlos y así evitar choques, el código es capaz de ayudar al robot a salir del un laberinto como se puede apreciar en la Figura 39. Estos escenarios fueron creados para poner a prueba el código diseñado y poder garantizar que responde ante cualquier tipo de escenario

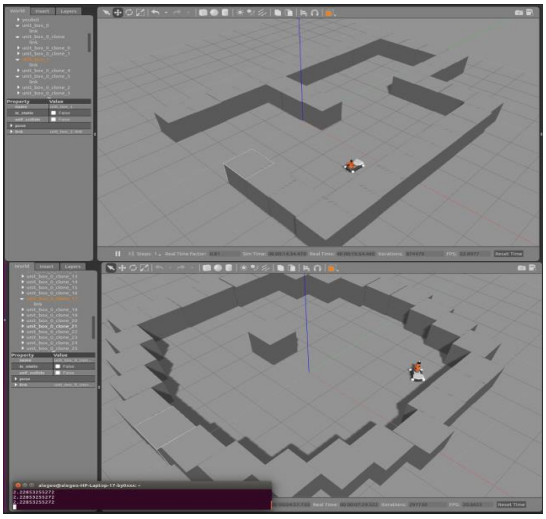
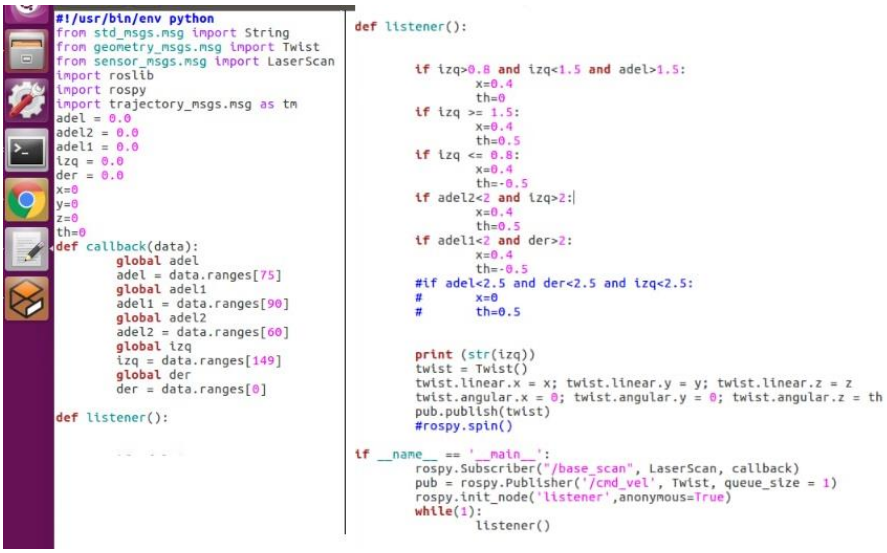


Figura 39: Kuka Youbot evitando obstáculos
Fuente: Investigador

En la Figura 40 podemos apreciar el código con el cual se produjo las pruebas de funcionamiento, teniendo un éxito del 100% sin presentar ni un solo choque con el código finalizado, el código se suscribe al tema /base_scan para identificar los valores necesarios, con “data.ranges” localizamos a la distancia y orientación que se encuentran los objetos, además el script fue diseñado para que el robot se mantenga siempre cerca a una pared en la parte izquierda y así poder salir con facilidad de laberintos. Este código es basado en la ley de la mano derecha para laberintos, la cual menciona que si te encuentras atrapado solo debes caminar hacia adelante con la mano sin separar de la pared, teniendo así siempre una pared al lado derecho y recorriendo toda la superficie hasta encontrar la salida.



```
#!/usr/bin/env python
from std_msgs.msg import String
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan
import roslib
import rospy
import trajectory_msgs.msg as tm
adel = 0.0
adel2 = 0.0
adel1 = 0.0
izq = 0.0
der = 0.0
x=0
y=0
z=0
th=0
def callback(data):
        global adel
        adel = data.ranges[75]
        global adel1
        adel1 = data.ranges[90]
        global adel2
        adel2 = data.ranges[60]
        global izq
        izq = data.ranges[149]
        global der
        der = data.ranges[0]

def listener():
```

```
def listener():

        if izq>0.8 and izq<1.5 and adel>1.5:
                x=0.4
                th=0
        if izq >= 1.5:
                x=0.4
                th=0.5
        if izq <= 0.8:
                x=0.4
                th=-0.5
        if adel2<2 and izq>2:
                x=0.4
                th=0.5
        if adel1<2 and der>2:
                x=0.4
                th=-0.5
        #if adel<2.5 and der<2.5 and izq<2.5:
        #       x=0
        #       th=0.5


        print (str(izq))
        twist = Twist()
        twist.linear.x = x; twist.linear.y = y; twist.linear.z = z
        twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = th
        pub.publish(twist)
        #rospy.spin()

if __name__ == '__main__':
        rospy.Subscriber("/base_scan", LaserScan, callback)
        pub = rospy.Publisher('/cmd_vel', Twist, queue_size = 1)
        rospy.init_node('listener',anonymous=True)
        while(1):
                listener()
```

Figura 40: Código evita obstáculos
Fuente: Investigador

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- El robot Kuka Youbot es considerado un robot industrial de escritorio debido que es utilizado para la investigación y estudio de la automatización logrando a simular un entorno industrial, este robot llega a ser un entrenador para el futuro personal especializado y encargado de dirigir grandes empresas automatizadas. La cinemática de Youbot mantiene dos partes: La primera es el modelo de la plataforma que contiene ruedas Mecanum lo cual es una de las mejores tecnologías en llantas por su posibilidad de reparaciones en los rodillos sin desmontar la rueda por completo, además que gracias a este sistema el robot puede tener un movimiento casi ilimitado ahorrando espacio en giros y dedicando mayor área a la producción. La segunda es el modelo del manipulador de 5 ejes de amplio movimiento y una pinza de dos dedos, alcanzando un amplio giro de 338º y una carga útil de 0.5 Kg.

- La configuración de comunicación inalámbrica mediante sockets TCP demostró tener una conectividad rápida y sencilla para su implementación.

- La creación de la placa PCB con el microcontrolador ESP32-WROOM-32 facilitó las pruebas de funcionamiento debido que su codificación puede ser hecha en múltiples lenguajes de programación como: microPython, C, C++ y ESP-IDF.

- El controlador fue puesto a prueba con un robot simulado en gazebo en el cual fue corregidos todos los errores encontrados a lo largo de la investigación, una vez el sistema funcionó correctamente con el simulador se probó con el robot real y se logró concretar la funcionalidad correcta con ambos robots sin necesidad de hacer cambios en la comunicación. Este sistema creado cuenta con multifuncionalidad debido que solo es necesario cambiar la IP del servidor a conectarse para poder controlar ambos robots.

4.2 Recomendaciones

Si se quiere recrear este proyecto de investigación se debe tomar en cuenta las siguientes recomendaciones:

- Para iniciar con ROS es necesario revisar los temas de ayuda en http://wiki.ros.org/ debido que nos ayudan desde lo más básico, además cuenta con un foro de ayuda mutua entre programadores. Es importante familiarizarse con los términos utilizados para poder continuar con la codificación y poder resolver los problemas presentados.
- Para colocar los dispositivos de entrada a los pines GPIO del microcontrolador se debe tener en cuenta que existen pines que se desactivan al activarse la comunicación WiFi por lo cual es fundamental revisar la hoja de datos del microcontrolador.
- Trabajar con el entorno de simulación debido que es muy similar con real y así no se tendrá daños en el robot por fallos en códigos. Se debe tener en cuenta que el brazo Youbot simulado y el real funcionan con distinto tema de mensaje de ROS. Para identificar el orden de vectores en el mensaje a enviar con el tipo de tema de ROS podemos verificar con "rosmsg show *tema*".
- Para tener una conectividad directa con wifi de la máquina Ubuntu es recomendable instalar la versión a utilizar juntamente con Windows, sin la necesidad de máquinas virtuales y así evitar problemas con los puertos de la máquina anfitrión.
- A diferencia del robot simulado en el robot real se debe especificar las configuraciones de ROS_MASTER_URI, ROS_HOSTNAME Y ROS_IP antes de ejecutar cualquier nodo en los terminales, debido que así le decimos al sistema en donde puede ubicar al nodo maestro.
- Siguiendo los pasos de instalación y configuración de ROS y Gazebo en este documento se puede obtener la facilidad de practicas con estudiantes utilizando el robot simulado dentro de una máquina virtual.
- Se debería ampliar el estudio dentro de la movilidad completamente autónoma del robot siguiendo los primeros pasos explicados en este documento para

obtener como resultado final la localización de objetos deseados y movilización de un lugar a otro.

- El prototipo diseñado en este proyecto de investigación debe ser usado de forma educativa para el beneficio de los estudiantes de la Universidad Técnica de Ambato.

C. MATERIAL DE REFERENCIA

Bibliografía

[1] C. Gordón y P. Encalada, «Robot autónomo KUKA YouBot Navegación basada en la planificación de rutas y reconocimiento de señales de tráfico,» de *Avances en Sistemas Inteligentes y Computación*, 2019.

[2] J. A. Montañez Barrera y M. L. Pinto Salamanca, «researchgate,» 15 11 2016. [En línea]. Available: https://www.researchgate.net/publication/314167056_Analisis_de_elementos_en_zona_local_y_remota_para_la_teleoperacion_del_brazo_robotico_AL5AAnalysis_of_elements_in_local_area_and_remote_for_teleoperation_of_AL5A_robotic_arm. [Último acceso: 03 11 2020].

[3] J. C. Yepes, J. J. Yepes y J. R. Martínez Torres, «researchgate,» 2014. [En línea]. Available: https://www.researchgate.net/profile/Juan_Yepes3/publication/310300374_Implementacion_de_una_Aplicacion_Movil_de_Teleoperacion_bajo_la_Plataforma_Android_para_Controlar_un_Robot_Industrial/links/582b1b9708ae004f74af9b99/Implementacion-de-una-Aplicacion-Mo. [Último acceso: 28 09 2020].

[4] J. C. Escobar Naranjo y M. V. García Sánchez, «uta.edu.ec,» 2019. [En línea]. Available: https://repositorio.uta.edu.ec/bitstream/123456789/29952/1/Tesis_1601id.pdf. [Último acceso: 03 11 2020].

[5] Á. F. Cañada Vilata y M. Mellado Arteche, «riunet.upv,» 02 09 2012. [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/17152/Memoria.pdf?sequence=1&isAllowed=y. [Último acceso: 28 09 2020].

[6] S. D. FRANCISCO SILVA, F. Silva y S. Deltiode, «Automatización Robótica de Proceso,» Deloitte, Febrero 2017. [En línea]. Available: https://www2.deloitte.com/content/dam/Deloitte/ec/Documents/deloitte-analytics/Estudios/Automatizacion_Rob%C3%B3tica_Procesos.pdf.

[7] C. Gallegos y E. S. Barahona, «Navegación Autónoma Basada en Maniobras Bajo Estimación de Posturas Humanas para un Robot Omnidireccional Kuka Youbot", Universidad Técnica de Ambato,» Universidad Tecnica de Ambato, Ingenieria en Electronica y Comunicaciones, Ambato, 2019.

[8] S. M. GONZALES, «A study on the usefulness of educationalrobotics from educators' perpective,» *Revista de Pedagogía*, vol. 32, nº 90, 2011.

[9] S. N. NAVARRO, «Smart Robots y Otras Máquinas Inteligentes en Nuestra Vida Cotidiana,» Revista CESCO de Derecho de Consumo, No 20, 2016. [En línea]. Available: vailable: http://www.revista.uclm.es/index.php/cesco.

[10] M. Sánchez, J. Salvador Bayarri, F. Millán Rodriguez, P. Jiménez Schlegl y F. Sánchez Martín, 2007. [En línea]. Available: http://scielo.isciii.es/pdf/aue/v31n3/v31n3a02.pdf. [Último acceso: 08 02 2021].

[11] itnuevolaredo, «itnuevolaredo,» 2005. [En línea]. Available: http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Inteligencia%20Artificial/Apuntes/tareas_alumnos/Robotica/Robotica%20_2005-Verano_.pdf.

[12] A. y. U. H. Kovács, *What They Are and What They Learn*, Página personal del autor, 2016.

[13] Locomotec, «generationrobots,» 06 12 2012. [En línea]. Available:

https://www.generationrobots.com/img/Kuka-YouBot-Technical-Specs.pdf. [Último acceso: 03 10 2020].

[14] maxongroup, «maxongroup,» 2013. [En línea]. Available: https://www.maxongroup.es/medias/sys_master/8815739797534.pdf?attachment=true. [Último acceso: 04 10 2020].

[15] G. M. Mafla Medina, «dspace.espoch.edu,» marzo 2019. [En línea]. Available: http://dspace.espoch.edu.ec/bitstream/123456789/10286/1/20T01182.pdf. [Último acceso: 13 10 2020].

[16] E. S. Barahona Guamani y C. Gordón Gallegos, 08 2019. [En línea]. Available: https://repositorio.uta.edu.ec/jspui/bitstream/123456789/30119/1/Tesis_t1642ec.pdf. [Último acceso: 03 11 2020].

[17] Open Robotics, «ROS,» [En línea]. Available: https://www.ros.org/about-ros/. [Último acceso: 21 6 2021].

[18] Moodle UA Cloud, «moodle2018-19.ua,» 2019. [En línea]. Available: https://moodle2018-19.ua.es/moodle/mod/book/view.php?id=8465&chapterid=186. [Último acceso: 22 06 2021].

[19] Moodle UA Cloud, «moodle2018-19.ua,» 2019. [En línea]. Available: https://moodle2018-19.ua.es/moodle/mod/book/view.php?id=8465&chapterid=190. [Último acceso: 22 06 2021].

[20] Moodle UA Cloud, «moodle2018-19.ua,» [En línea]. Available: https://moodle2018-19.ua.es/moodle/mod/book/view.php?id=8465&chapterid=181. [Último acceso: 22 06 2021].

[21] xunta, «edu.xunta,» 20 01 2020. [En línea]. Available: https://www.edu.xunta.gal/centros/iespedrofloriani/aulavirtual2/pluginfile.php/3245/mod_resource/content/0/Electronica/Electr17-02-13-2/circuitos_electrnicos.html.

[22] D. Benchimol, «books.google,» 2011. [En línea]. Available: https://books.google.com.ec/books?id=nNUeRa78FBUC&printsec=frontcover&dq=microcontrolador&hl=es&sa=X&ved=2ahUKEwjAmdrzlr3vAhWEtVkKHeWVALcQ6AEwAHoECAAQAg#v=onepage&q&f=true. [Último acceso: 19 03 2021].

[23] H y S. Altamirano, *Sistema de Control de Acceso por Reconocimiento de Iris para el Ingreso dePersonal a la Empresa Electroservicios Querubín de la Ciudad de Puyo*, Ambato: Universidad Técnica de Ambato, 2018.

[24] A. R. Bruno Saravia, «microelectronicash,» 2019. [En línea]. Available: https://www.microelectronicash.com/downloads/ESP32_MANUAL.pdf. [Último acceso: 19 03 2021].

[25] raspberrypi, «raspberrypi,» [En línea]. Available: https://raspberrypi.cl/que-es-raspberry/. [Último acceso: 19 03 2021].

[26] E. Nuño Ortega y L. Basañez Villaluenga, «upcommons.upc.edu,» 04 2004. [En línea]. Available: https://upcommons.upc.edu/bitstream/handle/2117/570/IOC-DT-P-2004-05.pdf?sequence=1&isAllowed=y. [Último acceso: 03 11 2020].

[27] J. Paredes, «sites.google,» 2017. [En línea]. Available: https://sites.google.com/site/capllevantcomunicaciones/home/comunicaciones-inalambricas. [Último acceso: 03 10 2020].

[28] W. Zhang y L. L. Xiaoguang Hu, «sci-hub,» 2009. [En línea]. Available: https://sci-hub.tw/10.1109/ICIEA.2009.5138237. [Último acceso: 19 03 2021].

[29] J. Benavente Carmona, «openaccess.uoc.edu,» [En línea]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/43113/6/jbenaventecTFC0615memoria.pdf. [Último acceso: 10 10 2020].

[30] J. L. De La Cruz Fernández, M. Torres Roldán, F. J. Bellido Outeiriño y J. A. Gistas

Peyrona, «tecnicaindustrial,» [En línea]. Available: http://www.tecnicaindustrial.es/tiadmin/numeros/15/06/a06.pdf. [Último acceso: 10 10 2020].

[31] D. Benchimol, Microcontroladores, 1a ed., Buenos Aires: FOX Andina en coedición con DALAGA S.A., 2011, p. 320.

[32] Ryte Wiki, «ryte,» 2020. [En línea]. Available: https://es.ryte.com/wiki/HTTPS. [Último acceso: 19 03 2021].

[33] V. González Ruiz, «hpca,» 10 04 2008. [En línea]. Available: http://www.hpca.ual.es/~vruiz/docencia/imagen_y_sonido/practicas/html/texputse57.html. [Último acceso: 19 03 2021].

[34] K. Calvert y M. Donahoo, TCP/IP Sockets in Java, vol. 2, D. Penrose, Ed., New York: Elsevier Inc., 2008.

[35] youbot store, «youbot-store,» [En línea]. Available: http://www.youbot-store.com/developers/hardware-specification. [Último acceso: 23 06 2021].

[36] Hochschule Bonn-Rhein-Sieg, «youtube,» 08 06 2020. [En línea]. Available: https://www.youtube.com/watch?v=lZaZ-f1kFl8. [Último acceso: 23 06 2021].

[37] youbot store, «youbot-store,» 12 02 2015. [En línea]. Available: http://www.youbot-store.com/wiki/index.php/YouBot_Detailed_Specifications. [Último acceso: 24 06 2021].

[38] M. Füller, F. Hegger y E. Solda, «youbot-store,» [En línea]. Available: http://www.youbot-store.com/developers/soft-two-finger-gripper-75. [Último acceso: 23 06 2021].

[39] M. Ahlers, «youtube,» 29 08 2011. [En línea]. Available: https://www.youtube.com/watch?v=90cXfaFM4O8. [Último acceso: 23 06 2021].

[40] youbot store, «youbot-store,» 2014. [En línea]. Available: http://www.youbot-store.com/developers/asus-xtion-pro-live-driver. [Último acceso: 23 06 2021].

[41] youbot store, «youbot-store,» 2014. [En línea]. Available: http://www.youbot-store.com/developers/microsoft-lifecam-driver. [Último acceso: 23 06 2021].

[42] youbot store, «youbot-store,» 2014. [En línea]. Available: http://www.youbot-store.com/developers/hokuyo-laser-range-finder-driver. [Último acceso: 23 06 2021].

[43] Creative Commons Atribución Compartir Igual 3.0, 25 06 2021. [En línea]. Available: https://es.wikipedia.org/wiki/Robot_Operating_System. [Último acceso: 25 06 2021].

[44] T. Foote y K. Conley, «ros,» 21 09 2010. [En línea]. Available: https://www.ros.org/reps/rep-0003.html. [Último acceso: 25 06 2021].

[45] youbot store, 06 07 2016. [En línea]. Available: http://www.youbot-store.com/wiki/index.php/USB_Live-Stick. [Último acceso: 28 06 2021].

[46] O. Campos, «genbeta.,» 11 07 2011. [En línea]. Available: https://www.genbeta.com/desarrollo/benchmark-entre-c-c-c-python-y-go. [Último acceso: 28 06 2021].

[47] youbot store, «youbot store,» 01 06 2015. [En línea]. Available: http://www.youbot-store.com/wiki/index.php/Gazebo_simulation. [Último acceso: 28 06 2021].

[48] youbot store, «youbot store,» 2014. [En línea]. Available: http://www.youbot-store.com/v-rep. [Último acceso: 28 06 2021].

[49] youbot store, «youbot store,» 2014. [En línea]. Available: http://www.youbot-store.com/openrave. [Último acceso: 28 06 2021].

[50] youbot store, «youbot store,» 2014. [En línea]. Available: http://www.youbot-store.com/webots. [Último acceso: 28 06 2021].

[51] A. J. González, «profesores.elo.utfsm,» [En línea]. Available:

http://profesores.elo.utfsm.cl/~agv/elo309/lectures/interfazSocket.pdf. [Último acceso: 23 06 2021].

[52] espressif, «espressif,» 2021. [En línea]. Available: https://www.espressif.com/en/company/about-espressif. [Último acceso: 30 06 2021].

[53] espressif, «espressif,» 2021. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. [Último acceso: 30 06 2021].

[54] espressif, «espressif,» 2020. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. [Último acceso: 30 06 2021].

ANEXOS

ANEXO A

Código del controlador

```
#include <WiFi.h>
#include "config.h"  // Sustituir con datos de vuestra red
#include "ESP32_Utils.hpp"

const int joystick1 = 35;  // joystick izquierdo arriba y abajo
const int joystick2 = 34;  // joystick izquierdo lados
const int joystick3 = 25;  // joystick derecho arriba y abajo
const int joystick4 = 33;  // joystick derecho lados
const int button2 = 4;     // boton izquierdo-abajo     lado derecho
const int button8 = 23;     // boton centro izquierdo
const int button9 = 22;     // boton centro derecho
int jo1 = 0;        // variable lee valor potenciometro joystick1
int j2 = 0;        // variable lee valor potenciometro joystick2
int j3 = 0;        // variable lee valor potenciometro joystick3
int j4 = 0;        // variable lee valor potenciometro joystick4
int b2 = 0;         // variable lee valor boton izquierdo-abajo
int b8 = 0;         // variable lee valor boton centro izquierdo
int b9 = 0;         // variable lee valor boton centro derecho
char msg[25];
long count=0;
int status = WL_IDLE_STATUS;
WiFiClient client;

void setup() {
  Serial.begin(115200);
  Serial.println("inicio");
  ConnectWiFi_AP();
  Serial.println("fin");
}

void loop() {
  Serial.println("inicio2");
  if(client.connect(ip, 33657)){
     Serial.println("Conección éxitosa");
     jo1 = analogRead(joystick1);
     j2  = analogRead(joystick2);
     jo1 = analogRead(joystick1);
     j2  = analogRead(joystick2);
     char valor = 'k';

     if(j2<1400){
          //Serial.println("DERECHA");
```

```
      valor='l';
     }if(j2>2210){
          //Serial.println("IZQUIERDA");
      valor='j';
     }if(jo1>2400){
          //Serial.println("ARRIBA");
      valor='i';
     }if(jo1<1600){
          //Serial.println("ABAJO");
      valor=',';
     }if((jo1>=1605 && jo1<=1945)&&(j2>=1485 && j2<=1925)){
      valor='k';
     }
     char op;
     String str = String(valor);
     str.toCharArray(msg,25);
     if(op!=valor){
      op=valor;
      client.print(msg);
     }
     if ((v == HIGH) && (old_val == LOW)){
    state=1-state;
    delay(10);
    }
    old_val = v; // valor del antiguo estado
    if (state==1&&valor=='k'){
     valor='t'; // enciende el LED
    }
    if(state==0&&valor=='k'){
     valor='b'; // apagar el LED
    }if(digitalRead(button8)==HIGH){
     valor='y';
    }if(digitalRead(button9)==HIGH){
     valor='n';
    }
     client.stop();
    }
}
```

ESP32_Utils.hpp

```
void ConnectWiFi_STA(bool useStaticIP = false)
{
  Serial.println("");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
```

```
    delay(100);
    Serial.print('.');
  }

  Serial.println("");
  Serial.print("Iniciado STA:\t");
  Serial.println(ssid);
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());
}

void ConnectWiFi_AP(bool useStaticIP = false)
{
  Serial.println("");
  WiFi.mode(WIFI_AP);
  while(!WiFi.softAP(ssid, password))
  {
    Serial.print(".");
    delay(100);
  }

  Serial.println("");
  Serial.print("Iniciado AP:\t");
  Serial.println(ssid);
  Serial.print("IP address:\t");
  Serial.println(WiFi.softAPIP());
}
```

config.h

```
const char* ssid     = "alexis1";
const char* password = "12345678";
const char* hostname = "ESP32_1";

IPAddress ip(192,168,43,64);
```

ANEXO B

Código en el nodo del robot Kuka Youbot real

```
#!/usr/bin/env python
# @brief python_file

from __future__ import print_function

import roslib
import rospy
from numpy import inf, zeros
from geometry_msgs.msg import Twist
import trajectory_msgs.msg as tm
import sys, select, termios, tty, signal
import socket

HOST = '192.168.43.85'
PORT = 33657
s = socket.socket()
s.bind((HOST, PORT))
s.listen(0)


moveBindings = {
        'k':(0,0,0,0),
        'i':(1,0,0,0),
        'o':(1,0,0,-1),
        'j':(0,0,0,1),
        'l':(0,0,0,-1),
        'u':(1,0,0,1),
        ',':(-1,0,0,0),
        '.':(-1,0,0,1),
        'm':(-1,0,0,-1),
    }

moveArmBin = {
        't':(4.5, 1.05, -2.44, 0.11, 2.95),
        'b':(0.11, 0.11, -0.11, 0.11, 0.11),
    }
moveGrepperBin = {
        'y':(0.011),
        'n':(0),
    }

speedBindings={
        'q':(1.1,1.1),
        'z':(.9,.9),
```

```
        'w':(1.1,1),
        'x':(.9,1),
        'e':(1,1.1),
        'c':(1,.9),
    }

def getKey():

    tty.setraw(sys.stdin.fileno())
    select.select([sys.stdin], [], [], 0)
    key = sys.stdin.read(1)
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
    return key


def vels(speed,turn):

    return "currently:\tspeed %s\tturn %s " % (speed,turn)

def createArmPositionCommand(newPositions):
    msg = JointPositions()
    point = JointValue()
    jointNames = []
    joint.value = newPosition
    for i in range(5):
        jointNames.append("arm_joint_" + str(i+1))
    joint.joint_uri = jointNames()
    joint.unit = "rad"
    joint.timeStamp = rospy.time.now()
    msg.positions.append(joint)
    return msg

def createGripperPositionCommand(newPosition):

    msg = JointPositions()
    point = JointValue()
    jointNames = []
    joint.value = [newPosition, newPosition]
    joint.joint_uri = ["gripper_finger_joint_l", "gripper_finger_joint_r"]
    joint.unit = "rad"
    joint.timeStamp = rospy.time.now()
    msg.positions.append(joint)
    return msg


if __name__=="__main__":
    settings = termios.tcgetattr(sys.stdin)
    rospy.init_node('teleop_kuka')
```

```
    # Publisher for velocity command fed to Whiskeye
    pub = rospy.Publisher('/cmd_vel', Twist, queue_size = 1)
    armPublisher = rospy.Publisher("/arm_1/arm_controller/position_command", Joint
Positions, queue_size=1)
    gripperPublisher = rospy.Publisher("/arm_1/gripper_controller/position_command
", JointPositions, queue_size=1)
    speed = rospy.get_param("~speed", 0.5)
    turn = rospy.get_param("~turn", 1.0)

    # Position variables
    x = 0
    y = 0
    z = 0
    jointvalues = [0.11, 0.11, -0.11, 0.11, 0.11]
    grip = 0
    # orientation variables
    th = 0

    try:

        while(1):
            print("1")
            client, addr = s.accept()
            while True:
                print("2")
                cont = client.recv(1024)

        if len(cont)==0:
                    break
                else:
                    print(cont)
                    key = cont
                if key in moveBindings.keys():
                    x = moveBindings[key][0]
                    y = moveBindings[key][1]
                    z = moveBindings[key][2]
                    th = moveBindings[key][3]

                elif key in speedBindings.keys():
                    speed = speed * speedBindings[key][0]
                    turn = turn * speedBindings[key][1]
                elif key in moveArmBin.keys():
                    jointvalues = moveArmBin[key]
                elif key in moveGrepperBin.keys():
                    grip = moveGrepperBin[key]
                else:
                        # Reset parameters if arbitrary key is pressed
```

```
                x = 0
                y = 0
                z = 0
                th = 0
                jointvalues = [0.11, 0.11, -0.11, 0.11, 0.11]
                grip = 0
                if (key == '\x03'):  # CTRL-C pressed
                    break

            twist = Twist()
            twist.linear.x = x*speed
            twist.linear.y = y*speed
            twist.linear.z = 0
            twist.angular.x = 0
            twist.angular.y = 0
            twist.angular.z = th*turn
            msgd = createArmPositionCommand(jointvalues)
            msg = createGripperPositionCommand(grip)
                # Publish commands to the robot
            pub.publish(twist)
            armPublisher.publish(msgd)
            gripperPublisher.publish(msg)
        client.close()
except Exception as e:
    print(e)

finally:
    twist = Twist()
    msgd = createArmPositionCommand(jointvalues)
    twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
    twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
    jointvalues = [0.11, 0.11, -0.11, 0.11, 0.11]
    pub.publish(twist)
    armPublisher.publish(msgd)
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
```

ANEXO C

Código en el nodo del robot Kuka Youbot simulado en Gazebo

```
#!/usr/bin/env python
# @brief python_file

from __future__ import print_function

import roslib
# roslib.load_manifest('teleop')
import rospy
from numpy import inf, zeros
from geometry_msgs.msg import Twist
import trajectory_msgs.msg as tm
import sys, select, termios, tty, signal
import socket

HOST = '192.168.43.244'
PORT = 33657
s = socket.socket()
s.bind((HOST, PORT))
s.listen(0)

msg = """
Moving around:
   u    i    o
   j    k    l
   m    ,    .

t : up (arm)
b : down (arm)

CTRL-C to quit
"""

moveBindings = {
        'k':(0,0,0,0),
        'i':(1,0,0,0),
        'o':(1,0,0,-1),
        'j':(0,0,0,1),
        'l':(0,0,0,-1),
        'u':(1,0,0,1),
        ',':(-1,0,0,0),
        '.':(-1,0,0,1),
        'm':(-1,0,0,-1),
    }
```

```
moveArmBin = {
        't':(4.5, 1.05, -2.44, 0.11, 2.95),
        'b':(0.11, 0.11, -0.11, 0.11, 0.11),
    }
moveGrepperBin = {
        'y':(0.011),
        'n':(0),
    }

speedBindings={
        'q':(1.1,1.1),
        'z':(.9,.9),
        'w':(1.1,1),
        'x':(.9,1),
        'e':(1,1.1),
        'c':(1,.9),
    }

def getKey():

    tty.setraw(sys.stdin.fileno())
    select.select([sys.stdin], [], [], 0)
    key = sys.stdin.read(1)
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
    return key


def vels(speed,turn):

    return "currently:\tspeed %s\tturn %s " % (speed,turn)

def createArmPositionCommand(newPositions):
    msgd = tm.JointTrajectory()
    point = tm.JointTrajectoryPoint()
    point.positions = newPositions
    point.velocities = zeros(len(newPositions))
    point.accelerations = zeros(len(newPositions))
    point.time_from_start = rospy.Duration(0.5)
    msgd.points = [point]
    jointNames = []
    for i in range(5):
        jointNames.append("arm_joint_" + str(i+1))
    msgd.joint_names = jointNames
    msgd.header.frame_id = "arm_link_0"
    msgd.header.stamp = rospy.Time.now()
    return msgd

def createGripperPositionCommand(newPosition):
```

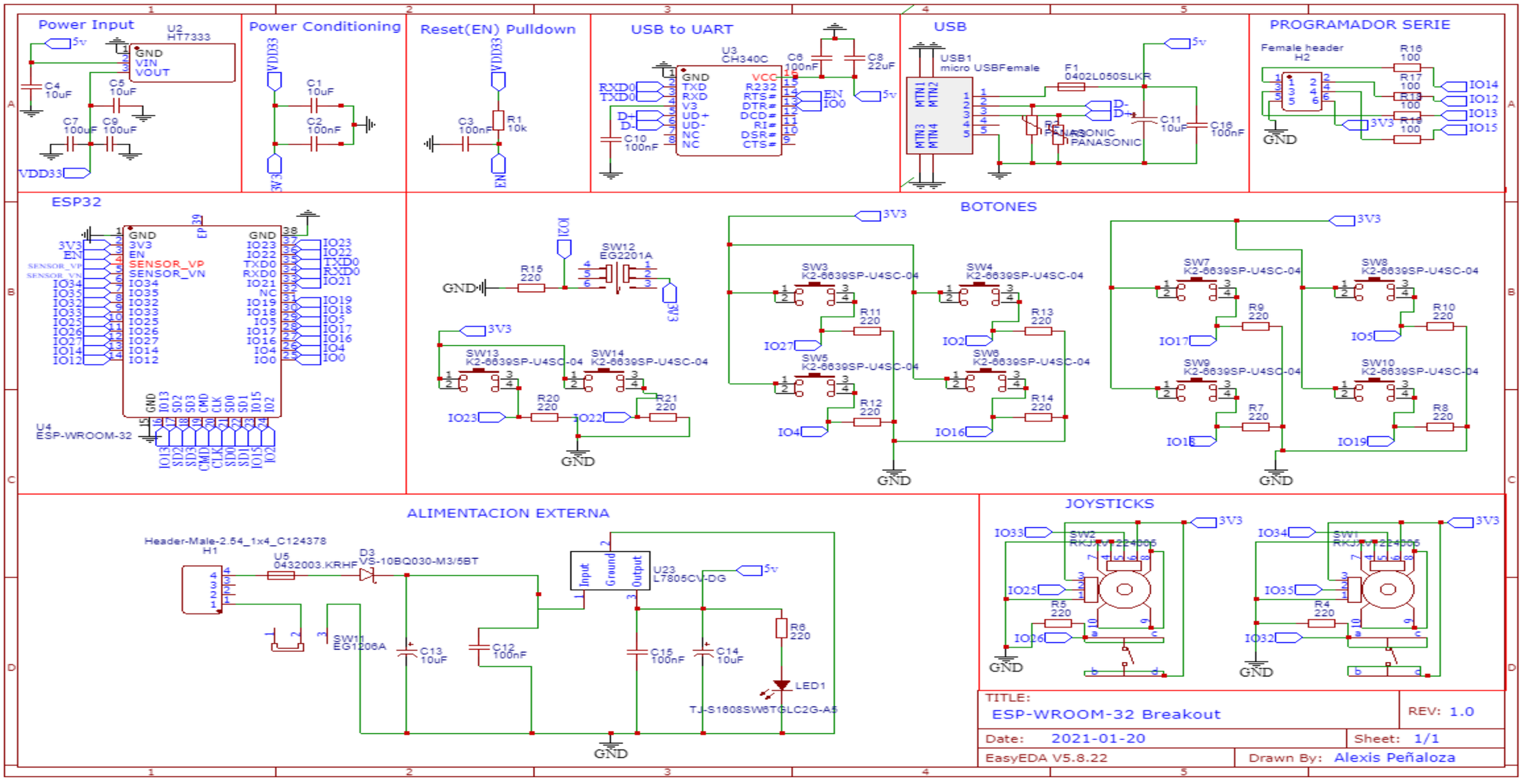
```
    msg = tm.JointTrajectory()
    point = tm.JointTrajectoryPoint()
    point.positions = [newPosition, newPosition]
    point.velocities = zeros(2)
    point.accelerations = zeros(2)
    point.time_from_start = rospy.Duration(0.5)
    msg.points = [point]
    jointNames = []
    msg.joint_names = ["gripper_finger_joint_l", "gripper_finger_joint_r"]
    msg.header.frame_id = "gripper_pointer_link"
    msg.header.stamp = rospy.Time.now()
    return msg


if __name__=="__main__":
    settings = termios.tcgetattr(sys.stdin)
    rospy.init_node('teleop_kuka')

    # Publisher for velocity command fed to Whiskeye
    pub = rospy.Publisher('/cmd_vel', Twist, queue_size = 1)
    armPublisher = rospy.Publisher("/arm_1/arm_controller/command", tm.JointTraje
ctory, queue_size=1)
    gripperPublisher = rospy.Publisher("/arm_1/gripper_controller/command", tm.Join
tTrajectory, queue_size=1)
    speed = rospy.get_param("~speed", 0.5)  # target linear velocity
    turn = rospy.get_param("~turn", 1.0)  # angle for turning

    # Position variables
    x = 0
    y = 0
    z = 0
    jointvalues = [0.11, 0.11, -0.11, 0.11, 0.11]
    grip = 0
    # orientation variables
    th = 0

    try:

        while(1):
            print("1")
            client, addr = s.accept()
            while True:
                print("2")
                cont = client.recv(1024)

        if len(cont)==0:
                    break
```

```
            else:
                print(cont)
                key = cont
            if key in moveBindings.keys():
                x = moveBindings[key][0]
                y = moveBindings[key][1]
                z = moveBindings[key][2]
                th = moveBindings[key][3]

            elif key in speedBindings.keys():
                speed = speed * speedBindings[key][0]
                turn = turn * speedBindings[key][1]
            elif key in moveArmBin.keys():
                jointvalues = moveArmBin[key]
            elif key in moveGrepperBin.keys():
                grip = moveGrepperBin[key]
            else:
                    # Reset parameters if arbitrary key is pressed
                x = 0
                y = 0
                z = 0
                th = 0
                jointvalues = [0.11, 0.11, -0.11, 0.11, 0.11]
                grip = 0
                if (key == '\x03'):  # CTRL-C pressed
                    break

            twist = Twist()
            twist.linear.x = x*speed
            twist.linear.y = y*speed
            twist.linear.z = 0
            twist.angular.x = 0
            twist.angular.y = 0
            twist.angular.z = th*turn
            msgd = createArmPositionCommand(jointvalues)
            msg = createGripperPositionCommand(grip)
                # Publish commands to the robot
            pub.publish(twist)
            armPublisher.publish(msgd)
            gripperPublisher.publish(msg)
        client.close()
except Exception as e:
    print(e)

finally:
    twist = Twist()
    msgd = createArmPositionCommand(jointvalues)
    twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
```

```
twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
jointvalues = [0.11, 0.11, -0.11, 0.11, 0.11]
pub.publish(twist)
armPublisher.publish(msgd)
termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
```

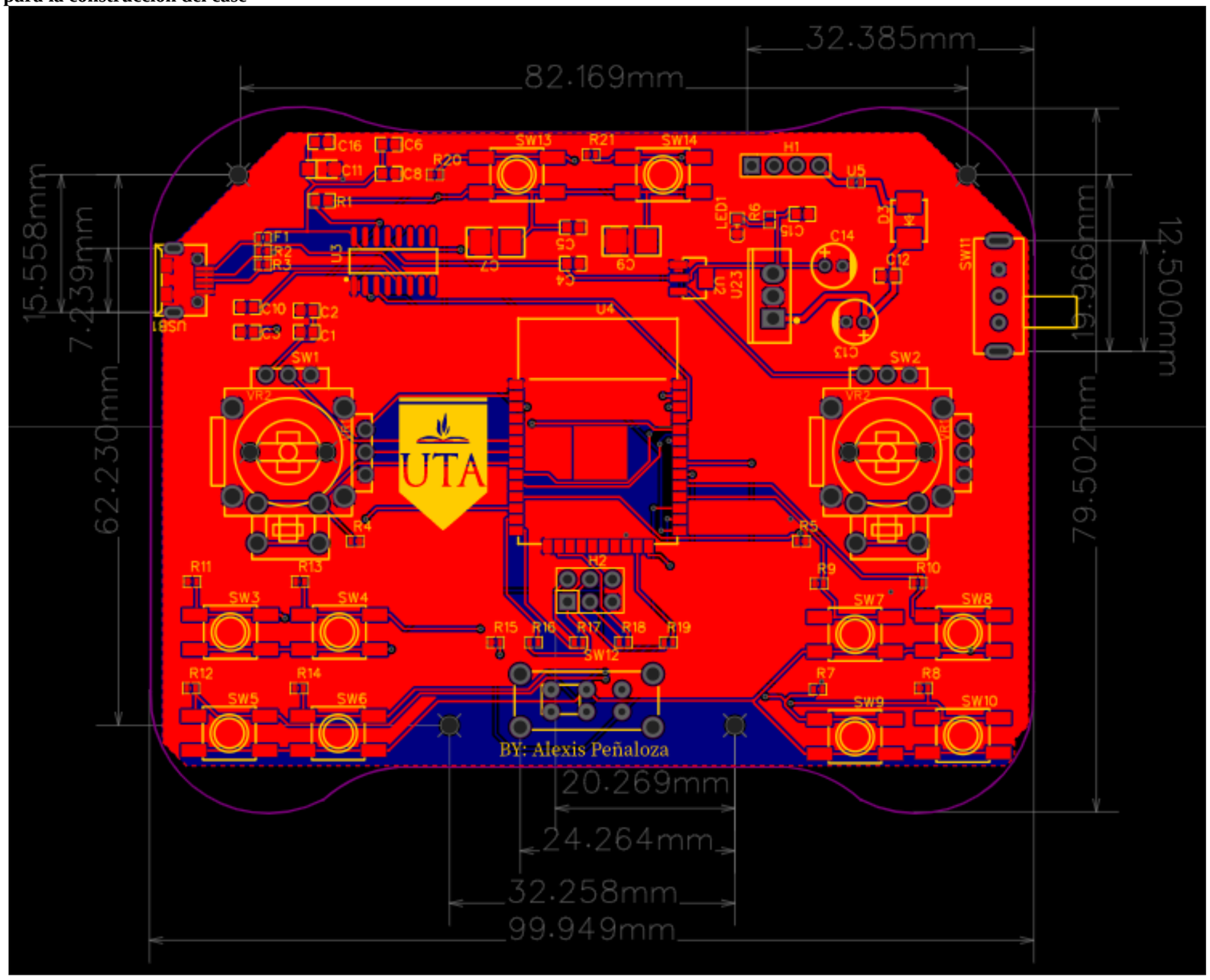
ANEXO D

Diagrama del circuito electrónico del controlador



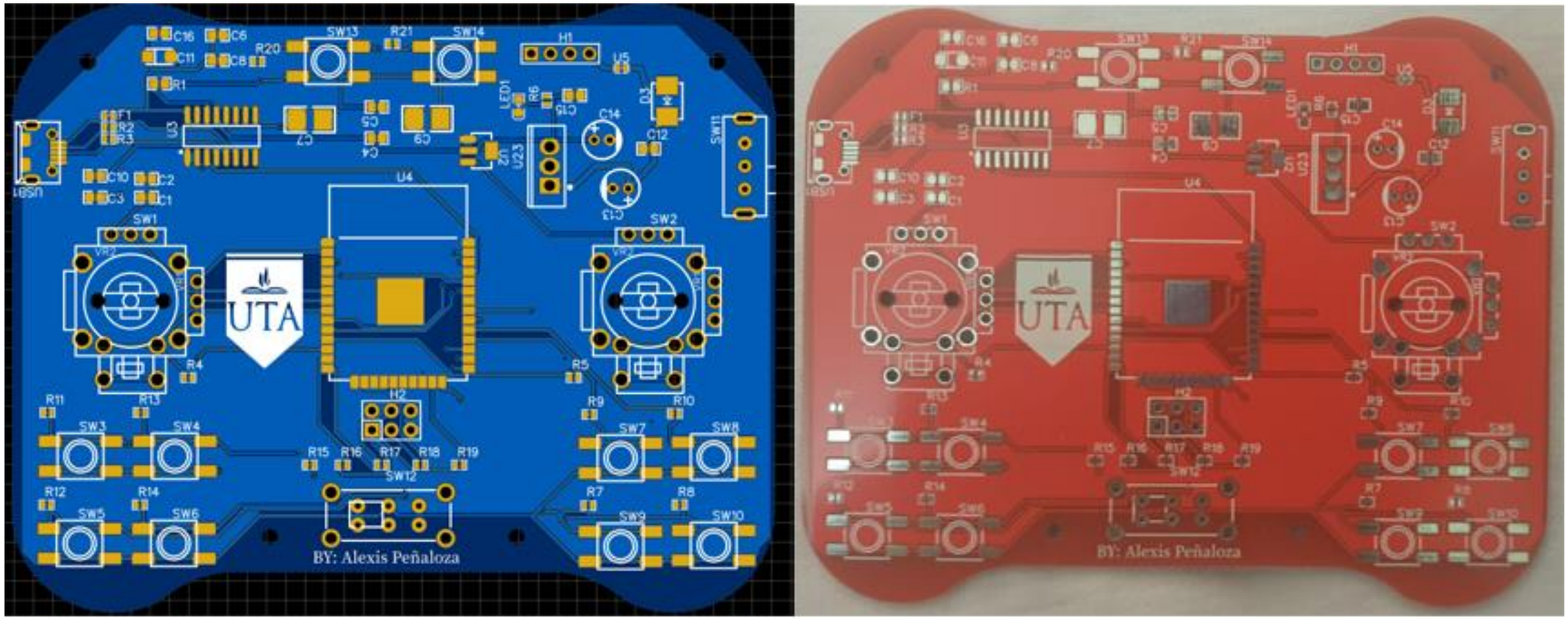
ANEXO E

Dimensiones de la placa PCB para la construcción del case



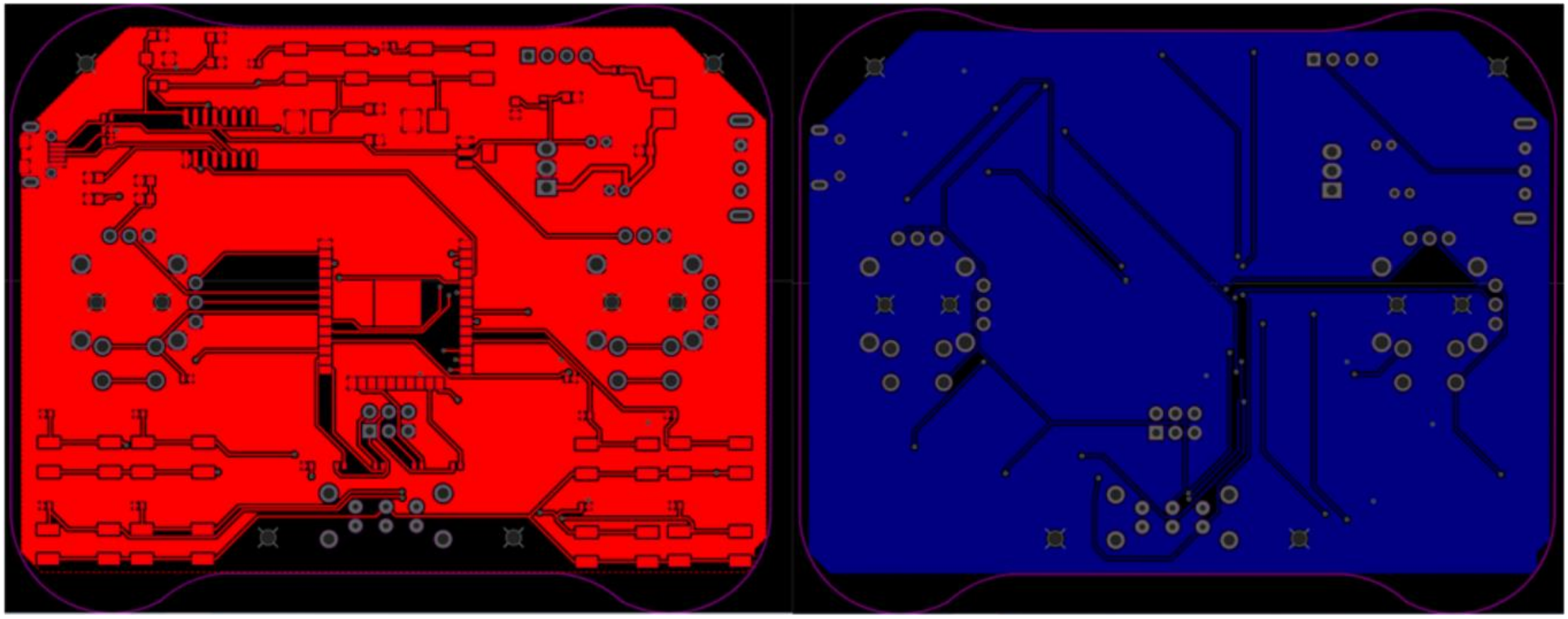
ANEXO F

Diseño 2D en EasyEDA en comparación con la placa completamente terminada



ANEXO G

Diseño del diagrama de pistas por capas



ANEXO H

Hoja de datos técnicos del microcontrolador ESP32
(la ficha técnica completa se puede encontrar en el link: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf)