



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**Control de navegación de un robot móvil
asistente enfermero en hospitales covid 19**

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Alan García Martínez

DIRECTOR DE TESIS

Dr. Víctor Manuel Lomas Barrié



Ciudad Universitaria, Cd. Mx., 2024

Jurado asignado:

Presidente.	Dr. Juan Mario Peña Cabrera
Vocal.	Dr. Víctor Manuel Lómas Barrié
Secretario.	Ing. Ricardo Federico Villareal Martínez
Primer Suplente.	Dr. Pablo Roberto Pérez Alcazar
Segundo Suplente.	Dr. Mauricio Enrique Reyes Castillo

Sitio en donde se desarrolló el tema:

Instituto de Investigaciones de Matemáticas Aplicadas y en Sistemas.

Laboratorio de electrónica y automatización para la industria 4.0.

Ciudad Universitaria.

Universidad Nacional Autónoma de México.

Dr. Víctor Manuel Lomas Barrié

Asesor de Tesis

Agradecimientos

La presente tesis fue posible de realizar gracias a las personas que han contribuido en mi proceso de formación académica, brindando su apoyo y esfuerzo, motivándome para cumplir mi meta de convertirme en ingeniero electrico y electrónico. Por este motivo deseo brindar un sincero agradecimiento con total sinceridad y cariño a mi familia; a mi madre Blanca Estela, quien siempre me ha escuchado y atendido en mis necesidades básicas desde mi nacimiento, me logro motivar todos estos años a no rendirme y seguir adelante a pesar de las dudas y oportunidades de declinar y tomar otras decisiones que habrían pospuesto mi deseo de poder cumplir mi meta de obtener mi título de profesionista. Un agradecimiento a mi padre Arnulfo, quien con su apoyo y constante esfuerzo hizo posible que lograra enfocarme en la realización de esta tesis, dandome la oportunidad de poder estudiar sin preocupación de carencias económicas y siempre brindando su esfuerzo para poder cumplir con los materiales y gastos económicos que estuvieron presentes, sacrificando el sudor de su trabajo para que no desistiera por preocupaciones económicas. Un agradecimiento a mis hermanos Etzael y Edith quienes con su ejemplo me motivaron a poder concluir mis estudios y poder seguir desarrollando mi formación personal y me brindaron un ejemplo a continuar a pesar de los constantes desvelos y sacrificios que son necesarios para poder lograr cumplir los objetivos propuestos.

Un agradecimiento especial a mi actual prometida y futura esposa Erika Palacios Reyes, quien es la persona más especial para mí, ya que ha depositado su confianza en mí, quien más me ha apoyado emocionalmente para tomar la decisión de concluir esta tesis, a pesar de las propias dudas y problemas que se han llegado a presentar en los últimos años, donde en ocasiones me sentia incapaz de cumplir y completar este proyecto de tesis, ella siempre ha estado conmigo en los peores y mejores momentos, en donde sentía que nunca podría lograr cumplir mis metas propuestas, ella logró darme la principal motivación para cada día despertar temprano y buscar seguir adelante para completar mis metas personales, en las que se encuentra completar la presente tesis y poder obtener mi cédula profesional para seguir creciendo juntos y cumplir no solo mis metas personales, si no también cumplir nuestras metas futuras.

Finalmente y no menos importante, doy un importante agradecimiento a la UNAM por darme la oportunidad de desarrollarme académicamente, así como a la Facultad de Ingeniería por la formación académica de poco más de 5 años para lograr convertirme en un ingeniero electrico electrónico al servicio de México, así como al Instituto de Investigaciones de Matemáticas Aplicadas y en Sistemas donde tuve la oportunidad de realizar mi servicio social y tesis desarrollándome en el ámbito de la robótica, el cuál se ha convertido en mi principal campo de interés.

*Alan García Martínez
Álvaro Obregón Ciudad de México, 2023.*

*Cada libro, cada tomo que ves, tiene alma.
El alma de quien lo escribió,
y el alma de quienes lo leyeron y vivieron y soñaron con él*

-La sombra del viento. Carlos Ruiz Zafón.

*Se trabaja con imaginación, intuición y una verdad aparente;
cuando esto se consigue, entonces se logra la historia que uno quiere dar a conocer.
Creo que eso es, en principio, la base de todo cuento, de toda historia que se quiere contar.*

- El desafío de la creación. Juan Rulfo.

*No importa lo que pases, no pierdas el rumbo, porque no hay
nada más genial que ver tu realidad transformada en algo que
refleje a la perfección lo que en realidad eres.*

-Jen Sincero.

Índice general

Índice general	VII
Índice de figuras	VIII
Índice de tablas	XIII
1 Introducción	7
1.1 Planteamiento del problema	7
1.2 Importancia de la robótica para hospitales	8
2 MARCO TEÓRICO	9
2.1 Consideraciones para un robot omnidireccional de servicio	9
2.2 Sistema motriz de un robot móvil Omnidireccional	20
2.3 Sistemas de control de robots móviles	52
2.4 Robot Operating System	58
3 METODOLOGÍA	70
3.1 Modulación por ancho de pulso para Raspberry PI Pico.	71
3.2 Análisis dinámico de los motores.	73
3.2.1 Modelado del sistema.	75
3.3 Diseño de control del robot móvil omnidireccional.	79
3.4 Tren motriz del robot móvil	83

3.4.1	Fuente de alimentación.	83
3.4.2	Protección de la electrónica de potencia	84
3.4.3	Convertidor de corriente directa.	84
3.4.4	Configuración de modulación por ancho de pulso para los controladores de motores de corriente directa.	85
3.5	Sensores de distancia y retroalimentación	89
3.6	Sensor de corriente y nivel de batería	90
3.7	Diagrama de Flujo para el algoritmo de navegación	91
3.8	Diagramas de conexión	96
3.8.1	Propuesta de diseño del circuito.	96
3.8.2	Propuesta de Diseño del circuito en PCB.	100
3.9	Configuración de ROS2, Rosbridge, micro-ROS y ROS Con Controller	101
3.9.1	Instalación de <i>ROS 2</i> Humble	101
3.9.2	Configuración del entorno de <i>ROS 2</i> Humble	102
3.9.3	Configuración del control desde navegador con <i>ROS Bridge</i>	103
3.9.4	<i>micro-ROS</i> para el dispositivo <i>Raspberry Pico</i>	106
3.9.5	ROS PID.	108
3.9.6	Configuración de tópicos.	110

4 RESULTADOS Y DISCUSIÓN 111

4.1	Resultado del planchado y serigrafía de la PCB	111
4.2	Resultado obtenido en la señal PWM y corriente del motor.	112
4.2.1	Resultado obtenido en la señal PWM.	115
4.3	Resultado de la lectura en terminal.	117
4.4	Resultado de diseños 3D para componentes y chasis.	120
4.4.1	Diseños propios.	120
4.4.2	Diseños de la comunidad de Thingiverse.	121

4.5	Resultados del robot.	125
5	CONCLUSIONES	140
6	Apéndice A. Planos realizados	144
6.1	Plano del Diseño esquemático de la conexión de los componentes.	144
6.2	Plano del Diseño esquemático general del robot.	145
6.3	Plano del Diseño de la PCB.	147
6.4	Planos del chasis.	151
7	Apéndice B. Lista de componentes	154
8	Apéndice C. Códigos realizados	155
8.1	Código utilizado para el Push Button.	155
8.2	Código utilizado para la calibración del Talon SR	157
8.3	Código utilizado para el Display.	158
8.4	Código utilizado para los sensores ultrasónicos.	161
8.5	Código utilizado para el RTC DS3231.	163
8.6	Código utilizado para el Driver BTS7960.	164
8.7	Código general con nodo en micro-ROS.	166
	Bibliografía	175

Índice de figuras

2.1	Vista de una sala de hospital en Ciudad de México.	10
2.2	Ejemplo de un pasillo de hospital.	12
2.3	Ejemplo de un pasillo de hospital con señalizaciones.	17
2.4	Ejemplo de un robot móvil omnidireccional con brazo robótico.	20
2.5	Rueda Suecano o rueda de Ilon.	22
2.6	Rueda de tipo <i>mecanum</i>	22
2.7	Direcciones para una plataforma compuesta de cuatro ruedas Mecanum. . . .	23
2.8	Ejemplo de un montacargas industrial con ruedas <i>mecanum</i>	24
2.9	Diseño para el prototipo del robot.	25
2.10	Primer versión del chasis y soporte del robot asistente.	26
2.11	Segunda versión del chasis y soporte del robot asistente.	27
2.12	Tipos de movimientos con una plataforma omnidireccional.	28
2.13	Propuesta de numeración de ruedas para definir la tabla de movimientos. . . .	30
2.14	Controlador de motores de corriente directa modelo Talon SR	31
2.15	Controlador de motores de corriente directa modelo BTS7960.	33
2.16	Motor de corriente directa AM800-001A.	36
2.17	Sensor ultrasonico modelo US100.	37
2.18	Sensor de corriente ACS712.	39
2.19	Reloj de tiempo real modelo DS3231.	41

2.20	Display LCD 16x2.	43
2.21	Módulo pcf8574 para pantalla LCD 16x2.	44
2.22	Voltímetro amperímetro digital.	45
2.23	Control modelo KY-023	46
2.24	Configuración de movimientos del control KY-023	47
2.25	Tira led RGB WS2812B.	48
2.26	Botón de paro de emergencia.	50
2.27	Raspberry Pi Pico.	53
2.28	Circuito Integrado RP2040.	53
2.29	Pinout del dispositivo Raspberry Pi Pico.	54
2.30	Raspberry Pi 4 Modelo B.	55
2.31	Características del dispositivo Raspberry PI 4 Model B.	56
2.32	Software Imager disponible para Windows y Linux	57
2.33	Sistemas operativos disponibles.	57
2.34	Sistemas operativos disponibles Linux.	57
2.35	Logo de la distribución de ROS2 HUMBLE	58
2.36	Interacción de sensores, actuadores y controladores de ROS2.	60
2.37	Interacción de elementos de <i>ROS 2</i>	61
2.38	Interacción de Nodos y Tópicos de <i>ROS</i>	62
2.39	Interacción de servicios de <i>ROS</i>	62
2.40	Interacción del nodo maestro en <i>ROS</i>	63
2.41	Interacción las acciones en <i>ROS</i>	64
2.42	Ejemplo de una interacción de nodos de un robot por medio de RVIZ	65
2.43	Ejemplo de la visualización de un modelo de un robot móvil en Gazebo	66
2.44	Arquitectura de comunicación de <i>Micro-ROS</i>	69
2.45	Comunicación entre cliente y agente.	69

3.1	Configuración de pines de la raspberry Pi Pico compatibles con canal de señal de modulación por ancho de pulso.	71
3.2	Señal de modulación por ancho de pulso en función del tiempo.	72
3.3	Esquema del circuito eléctrico del motor de corriente directa.	73
3.4	Caracterización espacial para el modelo del robot omnidireccinal.	75
3.5	Dirección vectorial de una rueda mecanum.	76
3.6	Diagrama de bloques del diseño de control de robot omnidireccional.	79
3.7	Diagrama de bloques del diseño de control de robot omnidireccional.	80
3.8	Entradas de referencia de corriente.	81
3.9	Gráfica de los errores: e_1 , e_2 , e_3 , e_4	81
3.10	Gráfica de la salida Deseada después de la cinemática dle robot omnidireccional con el controlador PID.	82
3.11	Batería seleccionada para la alimentación general del robot	83
3.12	Fusible de tipo americano para corrientes de 5 [A].	84
3.13	Convertidor de corriente directa 12 [V] - 5 [V] a 5[A].	84
3.14	Retroalimentación de los sensores de distancia "US-100".	89
3.15	Retroalimentación del sensor de corriente ASC712.	90
3.16	Diagrama de flujo de navegación del robot.	91
3.17	Diagrama de flujo de navegación del robot, segmento A.	91
3.18	Diagrama de flujo de navegación del robot, segmento B.	92
3.19	Diagrama de flujo de navegación del robot, segmento C.	92
3.20	Diagrama de flujo de navegación del robot, segmento D.	93
3.21	Diagrama de flujo de navegación del robot, segmento E.	94
3.22	Diagrama de flujo de navegación del robot, segmento F.	94
3.23	Diagrama de flujo de navegación del robot, segmento G.	94
3.24	Diagrama de flujo de navegación del robot, segmento H.	95
3.25	Diagrama de flujo de navegación del robot, segmento I.	95

3.26	Propuesta de los diagramas de conexión del circuito impreso o PCB.	96
3.27	Diagrama de conexión eléctrico.	98
3.28	Propuesta de los diagramas de conexión.	99
3.29	Propuesta del diseño de la PCB visto desde simulación 3D para la parte frontal.	100
3.30	Propuesta del diseño de la PCB visto desde simulación 3D para el reverso de las placas donde se encuentran la mayoría de pistas.	100
3.31	Ejecución desde la terminal para el Joystick.	103
3.32	Visualización desde el control desde navegador con nuestro navegador web. . .	104
3.33	Visualización de la terminal web al interactuar con el Joystick.	104
3.34	Vista de los tópicos en general con los que estamos interactuando.	105
3.35	Valor que esta recibiendo el tópico para una de las ruedas.	105
3.36	Conexión con el cliente de <i>micro-ros</i>	107
3.37	Gráfica de la entrada y salidas del controlador PID.	108
3.38	Gráfica de la entrada y salidas del controlador PID.	109
3.39	Esquema de configuración de ROS para los topicos para el controlador PID. .	110
4.1	Resultado de nuestra PCB de en nuestra cara frontal con los componentes soldados lista para realizar nuestra conexión con el robot.	111
4.2	Ciclo del trabajo cercano al 0%.	115
4.3	Ciclo del trabajo cercano al 25%.	115
4.4	Ciclo del trabajo cercano al 50%.	116
4.5	Ciclo del trabajo cercano al 75%.	116
4.6	Ciclo del trabajo cercano al 100%.	116
4.7	Lectura de las distancias de los 4 sensores de manera simultánea.	117
4.8	Lectura del mando KY-023 y selección de movimiento en alto.	117
4.9	Lectura del "mando KY-023" y selección de movimiento delantero.	118
4.10	Lectura e impresión del RTC DS3231.	118

4.11	Establecimiento de la conexión entre el agente y cliente de <i>micro-ROS</i>	119
4.12	Tópicos de conexión de <i>ROS</i>	119
4.13	Fallo de conexión entre el agente y cliente de <i>micro-ROS</i>	119
4.14	Diseño de soporte a chasis para los sensores de distancia "US-100" que utilizará el robot para lecturas de proximidad de distancia.	120
4.15	Nodos de conexión para los cables utilizados en las conexiones del robot omnidireccional.	121
4.16	Soporte para la camara Intel Realsense D455.	122
4.17	Versión preliminar de soporte para el sensor de corriente ACS712.	122
4.18	Diseño preliminar de una caja protectora para la pantalla LCD 16x2.	123
4.19	Visión preliminar de una caja protectora para un voltímetro amperímetro. . .	123
4.20	Diseño preliminar de mando para un único mando analógico con el componente "KY-023".	124
4.21	Sensor ultrasónico en la parte inferior del chasis del robot.	125
4.22	Batería de 12 [V], junto a los controladores de motores de corriente directa para pruebas preliminares.	125
4.23	Estado del robot en desorden de componentes y herramientas.	126
4.24	Parte trasera del chasis del robot.	127
4.25	Base para soportar la batería de 12 [V].	127
4.26	Parte frontal del robot con controladores de motores de corriente directa. . .	128
4.27	Base de la batería de ácido plomo de 12 [V].	128
4.28	Robot con cámar intel realsense D455 fijada al robot.	129
4.29	Configuración electronica sobre el chasis en su primer avance.	129
4.30	Segundo avance sobre la configuración electrónica sobre el chasis.	130
4.31	Tercer avance sobre la sección de la parte trasera de la configuración electrónica del chasis.	131
4.32	Batería colocada sobre su base para el chasis del robot.	132
4.33	Diseño de los soportes de los sensores para los sensores ultrasónicos "US-100".	132

4.34	Conexiones sobre la parte trasera del robot.	133
4.35	Botón de paro de emergencia del robot y pantalla táctil.	134
4.36	Pantalla LCD del tablero.	134
4.37	Vista lateral del robot en fase avanzada.	135
4.38	Vista de animación en la pantalla táctil del robot.	135
4.39	Propuesta de tablero de configuración electrónica final con prospectos de cambios del robot Ayudame 2.0.	136
4.40	Conexión electrónica en la propuesta final al desarrollo de esta tesis para tablero del robot asistente hospitalario "Ayudame 2.0".	137
4.41	Propuesta final al desarrollo de la tesis del robot omnidireccional asistente en entornos hospitalarios.. . . .	138
4.42	Vista frontal final del robot omnidireccional asistente de entornos hospitalarios.	139
6.1	Diseño esquemático de la conexión física de los componentes electrónicos. . . .	144
6.2	Diseño esquemático de la conexión física de los componentes electrónicos de manera general.	145
6.3	Tabla de simbología de la figura 8.2.	146
6.4	Diseño esquemático de la conexión física de los componentes electrónicos para la PCB.	147
6.5	Diseño impreso para la realización dela PCB sobre una tarjeta fenólica ajustada a la escala correcta para las pistas.	148
6.6	Diseño impreso para la realización dela PCB sobre una tarjeta fenólica ajustada a la escala correcta para la parte frontal de la PCB.	149
6.7	Diseño de los ejes del robot	150
6.8	Prototipo Versión 1 A del robot Ayudame 2.0. Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0.	151
6.9	Prototipo Versión 1 B del robot Ayudame 2.0. Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0	152
6.10	Prototipo Versión 1 C del robot Ayudame 2.0. Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0.	153

Índice de tablas

2.1	Caracterización de movimientos para 4 ruedas Mecanum.	30
2.2	Características del controlador de motores de corriente directa modelo Talon SR.	32
2.3	Especificaciones de pines del controlador de motores de corriente directa modelo BTS7960.	34
2.4	Especificaciones del controlador de motores de corriente directa modelo BTS7960.	34
2.5	Comparativa de las principales características entre cada controlador de motores de corriente directa.	35
2.6	Características del motor de corriente directa AM800-001A.	36
2.7	Funciones de los pines del sensor US-100.	38
2.8	Especificaciones del sensor de distancia US-100.	38
2.9	Funciones de los pines del sensor ACS712.	39
2.10	Especificaciones del sensor de corriente ACS712.	40
2.11	Funciones de los pines del reloj de tiempo real DS3231.	41
2.12	Especificaciones del reloj de tiempo real DS3231.	42
2.13	Funciones de los pines del módulo pcf8574 para la pantalla LCD 16x2.	44
2.14	Especificaciones modulo para la pantalla LCD 16x2.	44
2.15	Especificaciones modulo para la pantalla LCD 16x2.	45
2.16	Movimientos limitados para el control modelo KY-023.	46
2.17	Funciones de los pines del componente KY-023.	47

2.18	Funciones de los pines de la tira led WS2812B.	48
2.19	Especificaciones del módulo para la tira led RGB WS2812B.	49
2.20	Código de color y descripción de acción para la tira led RGB WS2812B.	49
2.21	Breve tabla de especificaciones para los calibres de cable seleccionados.	51
3.1	Caracterización de los controladores para motores de corriente directa para el robot sin movimiento.	85
3.2	Caracterización de los controladores para motores de corriente directa para un movimiento frontal.	85
3.3	Caracterización de los controladores para motores de corriente directa para un movimiento de reversa.	86
3.4	Caracterización de los controladores para motores de corriente directa para un movimiento lateral derecho.	86
3.5	Caracterización de los controladores para motores de corriente directa para un movimiento lateral izquierdo.	86
3.6	Caracterización de los controladores para motores de corriente directa para un movimiento Diagonal 1.	87
3.7	Caracterización de los controladores para motores de corriente directa para un movimiento Diagonal 2.	87
3.8	Caracterización de los controladores para motores de corriente directa para un movimiento Diagonal 3.	87
3.9	Caracterización de los controladores para motores de corriente directa para un movimiento Diagonal 4.	88
3.10	Caracterización de los controladores para motores de corriente directa para un movimiento de Giro en 360 grados $[H]$	88
3.11	Caracterización de los controladores para motores de corriente directa para un movimiento de Giro en 360 grados $[AH]$	88
3.12	Caracterización de Pines y asignaciones del microcontrolador RP2040.	97
4.1	Caracterización del Talon SR.	112
4.2	Caracterización del driver BTS7960.	112
4.3	Caracterización de los Motores 1 y 2.	112

4.4	Caracterización de los Motores 3 y 4.	113
4.5	Caracterización de los Motores a una salida de 6 [V].	113
4.6	Caracterización de los Motores a una salida de 0 [V].	113
4.7	Caracterización de los Motores a una salida de -6 [V].	114
5.1	Resumen de mejoras en el robot.	142

RESUMEN

El propósito de la presente tesis es plantear una solución a uno de los problemas sociales que derivaron como consecuencia del aislamiento y distanciamiento social que ocurrió durante la pandemia del COVID-19, la cual tuvo como principal acción preventiva establecer protocolos de seguridad, de manera más específica el conocido como distancias seguras entre los individuos, principalmente dirigidos a un sector de la población con mayor probabilidad de riesgo en salud debido a causas naturales como una determinada edad avanzada, entre diferentes padecimientos médicos. Como consecuencia de lo anterior, fueron cerrados establecimientos públicos y privados de mucha concurrencia, interrumpiendo las actividades cotidianas en escuelas, centros comerciales, restaurantes, entre otros lugares, evitando el aumento de contagios. No obstante, estas medidas de seguridad se implementaron también en hospitales y centros médicos, debido a que el personal médico y sanitario suele ser bastante propenso a contagiarse durante la interacción con los pacientes, se propone en esta tesis utilizar la robótica como herramienta principal para atender la problemática anteriormente mencionada, y de manera consecuente pueda ser de utilidad ante futuras epidemias y situaciones de peligro donde se requieran de una atención médica cercana entre médicos y pacientes, sin necesidad de aumentar el riesgo de contagio dentro de las instancias médicas, facilitando en los protocolos ya establecidos un medio para optimizar estos procesos, distribuyendo las tareas a un prototipo de robot móvil omnidireccional.

La motivación de esta propuesta es brindar una solución innovadora implementando un prototipo de robot móvil omnidireccional, que sea capaz de desempeñar labores específicas y de apoyo dentro de un entorno hospitalario, tomando como inspiración en el avance tecnológico constante de los robots, cada vez más presentes en múltiples actividades cotidianas, no pensando solo en robots que se encuentran dentro de diferentes industrias como pueden ser por ejemplo los diseñados para producción y automatización de fábricas o grandes industrias. De la misma manera, se tomaron como inspiración los nuevos y modernos prototipos de robots que asemejan el comportamiento y características físicas de un humano para la realización de nuestro propio robot. En esta visión se contempla no solo dar un paso en el aporte tecnológico patentando un diseño que no solo sea atractivo de ver, sino que también debe ser capaz de contribuir en la sociedad a mejorar la forma de recibir una atención médica más rápida y de calidad.

La robótica, actualmente se encuentra presente en el área de la salud y medicina, así como procesos de atención en pacientes, no es una novedad debido a que esta ya se encuentra presente desde hace algunos años dentro del apoyo en hospitales, particularmente con el uso de brazos robóticos para realizar cirugías de alto riesgo, donde se requiere de una mayor precisión en la manipulación de materiales quirúrgicos para realizar diversas operaciones en órganos del cuerpo humano, algunos ejemplos de estos modelos son el robot "Da Vinci" o "Zeus". No obstante aunque sea un aporte considerable en relación con la tecnología y la medicina, la principal desventaja de estos robots radica particularmente en que el costo para realizar estas operaciones es en ocasiones a precios poco accesibles para la población en general, además de poca probabilidad de acceso al público en general, por lo que tienen tiempos prolongados de espera para brindar servicio, sin considerar que el personal médico debe estar bien capacitado para operar estos dispositivos. Por este motivo, el propósito de nuestro robot móvil omnidireccional fue realizado considerado para ser de uso público, con un sistema de manejo sencillo para el usuario y pueda ser capaz de asistir en tareas donde se requiera realizar funciones como pueden ser entrega y transporte de diferentes insumos y suministros, de manera rápida y controlada, entre los que se pueden encontrar suministros médicos, muestras de laboratorio, ropa y materiales para desinfección, ahorrando tiempo y esfuerzo del personal para poder concentrarse en diferentes tareas específicas. Por otra parte, también puede desempeñar el papel de dar seguimiento y atención hacia los pacientes, recopilando y almacenando información para brindar una atención más eficiente.

El proceso de realización del robot se resume en un proceso de diferentes etapas, desde la planificación del diseño, la selección de componentes electrónicos y mecánicos, el proceso de ensamble del chasis principal, la programación y configuración electrónica. No habría sido posible, sin la colaboración con diferentes docentes y alumnos de tesis y servicio social, con principal apoyo del Instituto de Matemáticas Aplicadas y en Sistemas "IIMAS", el centro de Desarrollo de Ingeniería Industrial "CIDI", las cuales son instancias de la Universidad Nacional Autónoma de México "UNAM". En todo este tiempo de desarrollo se aportaron diferentes propuestas e ideas para contribuir en la participación en diferentes actividades enfocadas según fueran requeridas para realizar este robot. No obstante, un proyecto ambicioso requiere de un equipo de trabajo completo, ya que tomaría demasiada inversión en tiempo y capital para su realización, por este motivo y la limitación de tiempo de supervisión entre otros factores, por este motivo el principal punto de interés individual de esta tesis se concentra en la realización del diseño electrónico y comunicación con *ROS*, tomando consideraciones de seguridad eléctrica y el diseño de un código principal para realizar un sistema de control y navegación.

Se utilizaron múltiples herramientas disponibles para esta realización, por lo que aunque se realizaron algunas tareas de diseño e impresión 3D simples para lograr el funcionamiento electrónico, estas tareas no fueron el principal enfoque, por lo que algunas decisiones se tomaron en cuenta para el funcionamiento inmediato para realizar pruebas dentro del laboratorio.

El enfoque es hacer posible la capacidad de movimiento en nuestro robot, se tomaron en cuenta diferentes factores importantes presentes en la Norma Oficial Mexicana "NOM", así como el posible escenario de funcionamiento del robot en un entorno hospitalario ante situaciones posibles que puedan ser consideradas de riesgo en su operación, estableciendo un fundamento de carácter técnico y profesional, que en caso de ser implementado en un entorno de operación real, tomando en consideración no interferir o causar fallas en los protocolos ya establecidos en los entornos hospitalarios, como por ejemplo no interferir con señalamientos de seguridad, obstruya salidas de emergencia, colisione con objetos inmóviles, o si el terreno presenta obstáculos que entorpezcan el propósito original del robot, como podría ser un bote de basura, una silla, y de manera primordial si el entorno no es adecuado para el tipo de ruedas omnidireccionales. También se considera si el robot puede sufrir daños por el tipo de objetos del entorno como telas o basura tirada que se pueda atascar con las ruedas y dañar los motores, si un obstáculo difícil de detectar puede ocasionar colisiones o daños a las instalaciones como individuos presentes mientras el robot está en funcionamiento. Aun con estas consideraciones, lo cierto es que el número de casos no son totalmente predecibles debido a la incertidumbre del tipo de situación que puede causar una falla o mal funcionamiento del robot, y en un caso más grave ocasionar accidentes, por ejemplo durante el movimiento del robot este colisione con algún humano como puede ser un descuido durante la realización de alguna tarea en particular, así como en situaciones de emergencia de carácter natural en los que podría ocurrir desastres naturales como sismos, inundaciones o alguno otro pueda ser más perjudicial para el robot si se encuentra en estado de operación, por este motivo, para esta propuesta se tiene pensando que pueda emplearse este robot en hospitales certificados y regulados para normas oficiales para evitar contemplar otros inconvenientes que pueden ser corregidos al momento de implementar esta solución en un periodo de prueba, considerando un entorno de operación ideal.

Finalmente, el punto más importante en esta tesis es sentar una base de consulta que pueda ser referencia a futuro como consulta para continuar proyectos similares o buscar mejorar esta propuesta de solución, ya sea por el interés en el tema de la robótica, o realizar alguna patente que pueda contribuir un beneficio a la sociedad, superando los objetivos que se proponen para este proyecto, no en el sentido que pueda ser una solución a la problemática del aislamiento social debido al covid-19, también sea una solución a múltiples problemas que puedan presentarse a futuro, buscando que sea capaz de evolucionar para afrontar posibles nuevas epidemias que puedan suscitarse en un futuro, así como optimizar procesos de atención en pacientes, reducir riesgos de contagio en otras enfermedades que sean catalogadas de riesgo y principalmente sea de fácil acceso para poder operar de la manera en que se propone.

OBJETIVOS

Objetivo General

Se diseñara sistema de control para la navegación de un robot móvil sobre una plataforma omnidireccional que cumpla con funciones de asistente enfermero para hospitales para brindar un apoyo y soporte a personal médico y enfermero.

Objetivos específicos

Identificar las propiedades y características de un robot omnidireccional.

El robot asistente utilizará ruedas omnidireccionales de tipo *mecanum* para desplazarse de manera eficiente y precisa en entornos específicos de hospitales con el fin de optimizar la navegación y facilitar las tareas relacionadas con la asistencia médica.

Implementar un tren motriz para navegación del robot.

Se propondrá un prototipo de modelo de tren motriz con una propuesta de diseño de circuito electrónico, considerando aspectos como alimentación, protección contra picos de corriente y control del movimiento del robot móvil con relación a un algoritmo de navegación y un esquema de conexiones electrónicas con apoyo en diferentes sensores y actuadores.

Implementar un algoritmo para la navegación del robot móvil omnidireccional.

Se desarrollará un algoritmo que pueda controlar los movimientos y sistemas de navegación con basado en lecturas de sensores de nuestro robot omnidireccional, así como también diseñar e implementar un programa que permita gestionar de manera efectiva y precisa el tren motriz del robot, brindando control y maniobrabilidad.

Implementar un algoritmo de comunicación basado en ROS para el diseño del robot.

Se propondrá un sistema de comunicación con base en un entorno de *ROS* para establecer una comunicación entre tópicos y nodos para el funcionamiento del robot, utilizando las librerías que nos proporciona este entorno.

ANTECEDENTES

“La robótica se define como una ciencia que reúne diferentes campos tecnológicos, con el principal objetivo de diseñar máquinas robotizadas capaces de realizar diferentes tareas automatizadas en función de la capacidad de su software”.

-EDS Robotics.

“Un robot es una máquina capaz de extraer información de su ambiente y usa el conocimiento acerca de su mundo para moverse de manera segura para realizar un propósito específico”.

-1942, Isaac Asimov.

Introducción

Sección 1.1.

Planteamiento del problema

Los avances científicos del último siglo han hecho posible que la humanidad tenga una mayor esperanza de vida, gracias a la facilidad de conseguir productos de higiene personal, así como facilitar un mayor acceso a medicamentos y tratamientos para controlar enfermedades a un costo económico accesible, disminuyendo el riesgo en enfermedades que hasta hace algunas décadas llegaron a ser mortales para la población en general, debido en mayor medida a un mal cuidado de higiene personal o por el acceso restringido de medicamentos, como por ejemplo los antibióticos y antivirales. Si bien en la última década tener acceso a estos productos y servicios garantizaba una mayor calidad de vida en la sociedad mexicana, la epidemia del COVID-19 nos planteó una realidad que cambió la perspectiva sobre el manejo moderno y reacción para prevenir contagios ante la aparición de nuevas enfermedades. Esta epidemia logró en pocos meses tener un alto índice de contagios en pacientes, por lo que fue necesario declarar un estado de emergencia y tomar un aislamiento entre individuos que fue un punto de inflexión en distintas actividades sociales, laborales y educativas, entre muchas otras.

Si bien esta medida contuvo en gran medida los contagios en la población en general, fue necesario no detener actividades de carácter prioritario para la población en general, principalmente en sectores proveedores de alimentos, energía eléctrica y salud fueron primordiales para poder garantizar el funcionamiento de la sociedad aun cuando la mayoría de actividades fueron suspendidas durante este periodo. De manera particular, el área médica que se encontraba en labores fue principalmente expuesta al contacto directo con los pacientes del coronavirus, principalmente en los primeros picos de contagio con tratamiento prolongado en pacientes. Es ante esta situación que se propone una solución a este problema, utilizar un medio externo, que permita manejar de manera eficiente la atención médica con pacientes sin poner en riesgo la salud de médicos en casos innecesarios, donde actividades sencillas como el traslado de insumos y medicamentos puedan ser delegadas a un robot que sea capaz de cumplir tareas de manera ordenada.

Sección 1.2.

Importancia de la robótica para hospitales

Actualmente, la robótica se encuentra presente en la industria para desempeñar múltiples actividades específicas como la manufactura, soldadura, montaje de piezas mecánicas entre otras actividades de manera autónoma o semiautómata, según la función para la que se encuentra programada, son capaces de hacer estas tareas con una alta precisión, mejorando la eficiencia y productividad. Por este motivo, el campo de la robótica en la actualidad se encuentra más presente debido a la inversión en investigación y en desarrollo para emplearse no solo en función de uso industrial, sino también uso comercial y residencial. No obstante, las posibilidades que brinda son ilimitadas y se ajustan a las necesidades actuales en la sociedad. La propuesta de diseñar un robot onmidireccional asistente a hospitales, busca ofrecer una solución haciendo uso de nuevas herramientas y tecnologías que puedan ofrecer un servicio a los operadores médicos en atención a pacientes, especialmente a pacientes que requieren un aislamiento en su tratamiento, impulsado en gran medida debido a los efectos colaterales que tuvo la pandemia del covid-19, de manera que pueda ser utilizada en entornos hospitalarios para optimizar tareas, tales como transportar materiales e insumos dentro de diferentes zonas, niveles de charolas, vasos, platos, bandeja para medicinas, tener acceso a webcam, una interfaz de usuario accesible, y que sea capaz de realizar movimiento sin romper alguna normativa de seguridad o tener un problema en particular en su operación, de manera que sea capaz de ayudar a reducir el riesgo de infección al limitar la cantidad de interacciones entre el personal del hospital y los pacientes, mejorando la eficiencia y la calidad de la atención y liberando a los profesionales de la salud de tareas repetitivas y de bajo valor, con el propósito de ayudar en diferentes partes de la asistencia que se puede brindar para optimizar tareas de personal médico y de enfermería.

MARCO TEÓRICO

Sección 2.1.

Consideraciones para un robot omnidireccional de servicio

Es muy importante tener algunas consideraciones sobre los aspectos para el diseño de nuestro robot asistente a hospitales, principalmente teniendo en cuenta el entorno en el que nuestro robot estará operando para adaptar las condiciones de manejo en un entorno hospitalario para brindar un servicio seguro y eficaz, haciendo un breve análisis basado en lo que dicen las Normas Oficiales Mexicanas o por su abreviatura "NOM" sobre el tipo de especificaciones de entornos hospitalarios.

Las especificaciones recopiladas en los artículos propuestos por la Normativa Oficial Mexicana a tomar en cuenta se pueden listar de la siguiente manera.

- Especificaciones de sala.
- Dimensiones de pasillo.
- Tipo de suelo o superficie.
- Condiciones electromagnéticas.
- Normas de seguridad.
- Accesibilidad.
- Control de infecciones.

A continuación abordaremos brevemente estas consideraciones mencionadas en la lista anterior, resaltando los aspectos que consideramos importantes, debido a que cada norma oficial, mexicana o norma internacional es extensa, nos enfocaremos únicamente en los puntos que podrían considerarse para la implementación de nuestro robot.

Especificaciones de sala

Para este punto debemos tomar en cuenta el tamaño de la sala y las dimensiones del robot y el de las habitaciones que utilizarán, principalmente debido a que el robot debe poder moverse y maniobrar con facilidad sin causar molestias a pacientes y personal, por lo que las salas deben permitir el movimiento seguro con el mobiliario y otros equipos del hospital, principalmente tomando en cuenta como puede ser un entorno hospitalario en operación constante, tomando como ejemplo la figura 2.1, donde se observa un entorno de una sala de recepción de pacientes como ejemplo, donde habitualmente hay mobiliario como sillas, columnas y en operación habrá pacientes y personal en constante circulación.

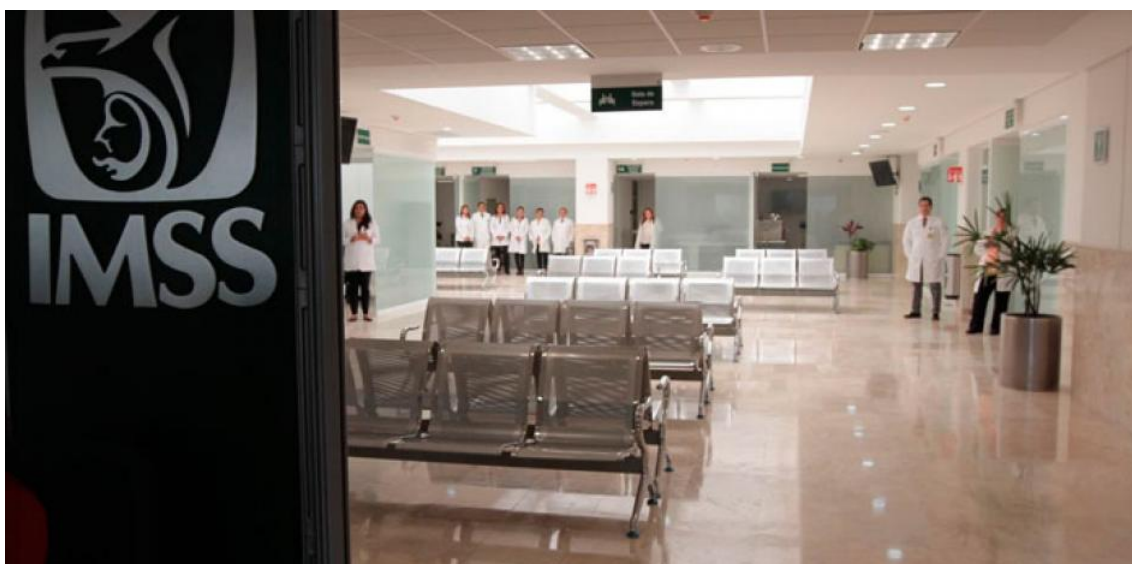


Figura 2.1: Vista de una sala de hospital en Ciudad de México.

Imagen obtenida en:

`"https://www.cronica.com.mx/notas-inaugura_el_imss_su_clinica_numero_48_en_la_ciudad_de_mexico-1018752-2017.html"`.

Con base en las consideraciones de la norma oficial mexicana NOM-016-SSA3-2012¹, que establece las características mínimas de infraestructura y equipamiento de hospitales y consultorios de atención médica especializada a tomar en cuenta para el servicio del robot omnidireccional toman en cuenta principalmente.

- Sección 6.6.1.1.1 "Las áreas, dimensiones y circulaciones deberán permitir el desarrollo de las funciones y actividades propias del personal de enfermería".

¹NOM-233-SSA1-2003, disponible en:

`"https://www.gob.mx/cms/uploads/attachment/file/512104/NOM-016-SSA3-2012.pdf"`

- Sección 6.6.4.6.1 "Los cubículos deberán tener las dimensiones suficientes para instalar y permitir la fácil movilidad de una cama-camilla y contar con su área tributaria correspondiente. Requieren instalaciones para suministrar oxígeno y realizar aspiración controlada, pudiendo ser esta última a través de instalaciones fijas o equipos portátiles".
- Sección 6.6.7.2 "Las dimensiones de las áreas de hospitalización, individual o colectiva, deben ser suficientes para la instalación del mobiliario y equipo apropiados para el tipo de pacientes a que esté dirigido el servicio, así como para permitir la movilidad y el desarrollo de las actividades del personal del área de la salud para la atención de los pacientes hospitalizados".
- Sección 6.6.9.1.2.12 "Las dimensiones de las áreas señaladas y sus respectivos accesos, pasillos y puertas, deben ser lo suficientemente amplios, para permitir la movilidad fácil y expedita de las camillas, equipos y del personal que interviene en la atención de los pacientes".
- Sección 7.10 "Cada consultorio deberá contar con una sala de espera, la cual podrá ser compartida por un conjunto de consultorios que se encuentren en una misma planta. Las dimensiones y el mobiliario de dicha sala, deberán ser proporcionales al número de consultorios que se disponga, preferentemente con un mínimo de 6 lugares de espera por consultorio".

Dimensiones de pasillo

Es indispensable tomar en cuenta las dimensiones de los pasillos y corredores en los que el robot se encontrará en servicio, ya que el robot debe estar diseñado para navegar en espacios estrechos y limitados de movimiento, de la misma manera los pasillos deben ser lo suficientemente anchos para permitir el movimiento seguro tanto del robot como del resto del personal. Podemos tomar de ejemplo la figura 2.2, donde se puede observar que hay señalizaciones, equipo de transporte de insumos, botes específicos de residuos, etc. Se debe tener el cuidado de no obstruir señalizaciones, que los sensores detecten obstáculos que puedan entorpecer la navegación del robot.



Figura 2.2: Ejemplo de un pasillo de hospital.

Imagen obtenida en:

"<https://www.pinterest.com.mx/pin/67e2b11bf2dfb3b8680e057b96a3f425/>".

Para la consideración de pasillos, tomaremos en cuenta principalmente la norma oficial mexicana, la NOM-233-SSA1-2003², donde resumiendo los principales puntos sobre la estructura se resalta lo siguiente.

- "Sección 6.1.3.1 Los cambios de nivel en piso se deben compensar con rampas ubicadas en esquinas y para distancias prolongadas se colocarán por lo menos cada 25.0 [m] y los peraltes máximos a una altura de 0.16 [m]."
- Sección 6.5.4 "Los peraltes deben ser verticales o con una inclinación al interior máxima de 0.025 [m]."

²NOM-233-SSA1-2003, disponible en:

"https://www.cndh.org.mx/sites/all/doc/Programas/Discapacidad/NOM_233_SSA1_2003.pdf"

- Sección 6.6.1 "Pasillos de comunicación al público: Ancho libre de 1.20 [m] como mínimo."
- Sección 6.6.4 "Tapajuntas en piso por cambio de acabado o junta constructiva: Diferencia máxima 0.013[m] altura."
- Sección 6.6.6 "Circulaciones internas en sanitarios, auditorios, comedores, regaderas y vestidores: 1.20 [m] libres."
- Sección 6.7.5 "Elevadores de servicio al público: Área interior libre 1.20 [m] * 1.50[m] como mínimo."
- Sección 6.7.6 "Ancho mínimo libre de puerta en elevadores 0.90 [m] como mínimo."
- Sección 6.14.3 "Mobiliario de atención al público: Espacio de 1.00[m] en sentido horizontal, altura mínima de 0.73[m] entre piso y parte inferior de cubierta y remetimiento inferior de 0.40[m]."
- Sección 6.15.1.4 "Para rampas y escaleras en el arranque y en la llegada deben de prolongarse 0.60 [m], con remate curvo y anclaje firme a piso o a muro que permita un apoyo seguro."
- Sección 6.15.3.5 "En circulaciones donde se utilice rejillas captadoras de agua con tapas de tipo perfil de ángulo o solera metálica: Separación máxima en uno de los lados, entre las piezas que la constituyen distancia mínima de 0.013[m]."
- Sección 6.15.3.7 "En salas de espera y auditorios se debe destinar un área cercana al acceso de 1.00[m] * 1.25[m] para personas con discapacidad en silla de ruedas y se indicará el área reservada."
- Sección 6.15.3.10 "En comedores se deben considerar mesas de altura libre de 0.76 [m] y asientos móviles."

Tipo de suelo o superficie

Para el modelo de nuestro robot omnidireccional con ruedas omnidireccionales de tipo *mecanum*, es necesario considerar el tipo de suelo o superficie de la instalación en la que deberá operar el robot, es de vital importancia que cuente con un suelo antideslizante y plano que evite la tracción o afecte el movimiento con múltiples vibraciones así para evitar deslizamientos accidentales o se vuelque, evitando causar posibles daños o afectaciones dentro de las instalaciones, en las figuras 2.1 y 2.2, se observa claramente que el tipo de suelo corresponde a uno liso, por esto mismo se tomara en cuenta con base a la norma oficial mexicana NOM-233-SSA1-2003³, establece algunas consideraciones que debemos tomar en cuenta, por ejemplo:

- Sección 6.1.2 "El acabado de pisos para el desplazamiento de personas con discapacidad, debe ser firme, uniforme y antiderrapante.",
- Sección 6.1.3.2 "En obras exteriores como plazas y banquetas considerar rampas para cambio de nivel en piso, con dimensiones mínimas de 1.00 [m] de ancho, pendiente no mayor de 8.0 % para un peralte de 0.16 [m] y de 6.0 % para desniveles mayores de dos p ralles o 0.32 [m], con acabado antiderrapante, de color contrastante que indique su presencia y señalización, conforme a lo señalado en el numeral 6.2 de esta Norma(NOM-233-SSA1-2003). "
- Sección 6.3.5 ".3.5 El piso de rampas debe ser firme, uniforme y antiderrapante, evitando acumulación de agua en descansos."

Condiciones electromagnéticas.

*El ambiente hospitalario es rico tanto en número como en diversidad de equipos electrónicos, que son susceptibles de recibir señales que posiblemente produzcan perturbaciones funcionales en ellos. El aumento de fuentes de campos electromagnéticos en tales ambientes, ha llevado a los investigadores alrededor del mundo a cuantificar el riesgo potencial de interferencia electromagnética en los equipos médicos. Para reducir los riesgos en estos equipos, es importante conocer el ambiente electromagnético que puede ser encontrado en los hospitales y clínicas.*⁴

³NOM-233-SSA1-2003, disponible en:

"https://www.cndh.org.mx/sites/all/doc/Programas/Discapacidad/NOM_233_SSA1_2003.pdf"

⁴Escobar, Johao .Silva Héctor Cadavid. Mapeo del campo magnético a frecuencia industrial en un ambiente hospitalario. Revista Tecnológica No.24. Julio 2010, disponible en:

"<https://www.redalyc.org/pdf/3442/344234318005.pdf>"

La norma internacional IEC 61000-4-8⁵, la cual establece la compatibilidad electromagnética a frecuencia industrial de 50/60 [Hz] de frecuencia en instalaciones eléctricas para la limitación de equipos electrónicos en entornos residenciales y comerciales, y principalmente industriales para evitar la interferencia de señales en equipos de electrónicos y de medición. Aunque la norma no proporciona límites específicos para frecuencias más altas, ya que constantemente varía en función de los equipos médicos manejados por cada país, establece pautas generales y principios para abordar perturbaciones magnéticas en un rango amplio de frecuencias. Para eso propone algunos métodos básicos para el análisis, destacando principalmente los siguientes métodos:

- *Método de proximidad: Consiste en la aplicación del campo magnético en los equipos electromédicos. El método consiste en pasar una bobina de inducción pequeña a lo largo de un equipo electromédico para descubrir áreas particularmente sensibles. IEC 61000-4-8⁶*
- *Método de la inmersión: Este método consiste en la aplicación del campo magnético en los equipos electromédicos, colocando los equipos en el centro de una bobina de inducción, realizar las mediciones de intensidad del campo magnético y valorar en el equipo la posible pérdida de funciones o la degradación de su actuación. IEC 61000-4-8⁷*

Estas consideraciones son para que las perturbaciones no afecten de manera considerable a los equipos médicos. En este caso, podría haber equipos electromédicos como pueden ser por ejemplo:

- Electrocauterio
- Bisturí armónico
- Colchón térmico
- Monitor de signos vitales
- Bomba de infusión
- Carro de anestesia
- Desfibrilador
- Carro rojo

⁵IEC 61000-4-8, disponible en:
”<https://webstore.iec.ch/publication/22272&preview>”

⁶IEC 61000-4-8

⁷IEC 61000-4-8

Los equipos anteriores están regulados por sus propias normas referidas en las siguientes normas internacionales enfocadas en los equipos electromédicos, principalmente las siguientes:

- IEC 60601-1-1: esta norma internacional principalmente regula las normativas fundamentales para la seguridad eléctrica de los equipos electromédicos.⁸
- IEC 60601-1-2: Esta norma internacional principalmente detalla los requisitos generales de los equipos electromédicos.⁹

Normas de seguridad

Existen normas de seguridad específicas que deben seguirse para operar en un entorno hospitalario para salvaguardar la salud y cuidado de los pacientes dentro de las instalaciones hospitalarias. Estas normas se enfocan principalmente en el cuidado de equipos médicos, el cuidado de las instalaciones, normas de higiene y protección civil. Por otra parte, la robótica al ser un campo aún en vías de desarrollo, no existen normas que regulen específicamente las dimensiones o capacidades que deber tener un robot en un entorno hospitalario, como por ejemplo algún diseño electrónico específico, por lo que mayormente debemos plantear con base en las normas ya existentes los cuidados al momento de desarrollar un robot para realizar diferentes labores, por lo que pueden plantearse algunas directrices sobre cómo debe funcionar el robot, cómo debe mantenerse y cómo debe limpiarse para evitar la propagación de infecciones y protocolos que se deben seguir para el cumplimiento de las normativas de higiene de los hospitales.

En este punto se considerará que el hospital debe cumplir con la Norma Oficial Mexicana NOM-026-STPS-2008¹⁰, que señala principalmente los colores y señales de seguridad e higiene dentro de las instalaciones hospitalarias, así como la identificación de riesgos por fluidos conducidos en tuberías, indicaciones y letreros de precaución e indicación de salidas de emergencia, como se observa en la figura 2.3, donde se observan algunas señalizaciones a una altura visible y una salida de emergencia sin obstrucción.

⁸IEC 61000-4-8, disponible en:

<https://www.bsigroup.com/globalassets/meddev/localfiles/en-gb/services/bsi-md-en-60601-leaflet-uk-en.pdf>

⁹IEC 61000-4-8

¹⁰NOM-026-STPS-2008, disponible en:

<https://www.stps.gob.mx/BP/secciones/dgsst/normatividad/normas/Nom-026.pdf>



Figura 2.3: Ejemplo de un pasillo de hospital con señalizaciones.

Imagen obtenida en:

"<https://www.pinterest.com.mx/pin/498421883735519602/>".

También se toma en consideración la Norma Oficial Mexicana NOM-004-SSA3-2012, del expediente clínico: Esta norma regula la correcta integración, organización y manejo de los expedientes clínicos en las instituciones de atención médica en México. Aunque no se enfoca exclusivamente en la seguridad física, garantiza que la documentación médica sea adecuada y segura, ya que en el caso del robot se debe llevar un registro para garantizar el correcto uso y manejo del robot en estado de operación, y finalmente considerando la Norma Oficial Mexicana NOM-026-SSA2-1998, Seguridad e higiene en los establecimientos de atención médica: Esta norma establece requisitos mínimos de seguridad e higiene en los establecimientos de atención médica, incluidos los hospitales. Cubre aspectos como la prevención de incendios, la evacuación de pacientes y el manejo de residuos médicos, nos ayudará a tener en cuenta la prevención del robot como un foco de infección para recomendaciones de cuidado en higiene.

Accesibilidad

El robot debe diseñarse de modo que sea fácilmente accesible para su mantenimiento y reparación, si no también accesible al personal médico y usuarios. Esto significa que los componentes deben ser fácilmente extraíbles y reemplazables en caso de que algún componente del sistema pueda fallar, también debe haber un fácil acceso a la batería del robot y a otros componentes críticos, profundizaremos sobre este punto más adelante al momento de hablar sobre los componentes seleccionados e implementados en la configuración electrónica del robot.

Por otra parte, el robot no debe estorbar con la accesibilidad de las instalaciones hospitalarias, es decir que no bloquee las señalizaciones de seguridad, los pasamanos o rampas de camino para pacientes y personal, no debe bloquear puntos de salidas de emergencia y finalmente que no colisione con mobiliario del hospital, nuevamente considerando las condiciones de suelo mencionadas en la norma oficial mexicana NOM-026-STPS-2008¹¹ mencionada en el punto anterior.

Control de infecciones

En un entorno hospitalario, el control de infecciones es de suma importancia, ya que existe la posibilidad de que el propio robot, si no es bien cuidado, sea una fuente de infección o enfermedades. El robot debe diseñarse para que sea fácil de limpiar y desinfectar, y debe estar fabricado con materiales resistentes a la proliferación bacteriana, misma mencionada en la Norma Oficial Mexicana NOM-026-STPS-2008 y la Norma Oficial Mexicana NOM-034-SSA3-2013, para la prevención y control de enfermedades en la atención primaria de salud: Aunque se centra en la prevención y control de enfermedades, esta norma tiene implicaciones para la seguridad en los hospitales al establecer directrices para la identificación y gestión de enfermedades infecciosas.

Esta consideración debe tomar en cuenta los materiales con los que se desarrolló este prototipo de robot, ya que la parte de la base del chasis consiste en un material de aleación de aluminio y acero inoxidable, mientras que la estructura física, consiste mayormente de material de impresión 3D, principalmente PETG y PLA, así como acrílico, estos son principalmente los materiales que constituyen el chasis del robot. La decisión de estos materiales fue por conveniencia y alcance para poder diseñar y proponer el modelo del robot, por decisión general del equipo Ayudame 2.0. Teniendo en cuenta lo anterior, algunas de las consideraciones que se deben tomar en cuenta pueden ser las siguientes:

¹¹NOM-026-STPS-2008, disponible en:
”<https://www.stps.gob.mx/BP/secciones/dgsst/normatividad/normas/Nom-026.pdf>”

- Superficie porosa.
- Desarrollo y crecimiento bacterianos
- Efectos de temperatura y humedad
- Crecimiento de agentes fungicos

La norma oficial mexicana NOM-249-SSA1-2010¹², establece algunos puntos que debemos tomar en consideración pensando en los puntos anteriores para evitar que nuestro robot pueda ser un foco de infección o cuidados que se deben tener:

- Sección 9.3, Los medicamentos e insumos se deben colocar sobre tarimas o anaqueles de tal manera que se facilite la limpieza, inspección y manejo.
- Sección 16.1, Las áreas utilizadas para la preparación y acondicionamiento deben estar separadas y comunicarse entre sí de acuerdo con un orden que corresponda a la secuencia de las operaciones y a los niveles de limpieza requeridos, de forma que se minimice el riesgo de confusión, se evite la contaminación y se disminuya el riesgo de omisión o ejecución errónea de cualquier fase del proceso.
- Sección 18.2 Todas las instalaciones, equipos, sistemas críticos y computarizados, que impacten en la calidad y control de las mezclas, deben estar calificados y los métodos de limpieza deben validarse, incluyendo limpieza y sanitización.
- Sección 18.9 Para la calificación de las instalaciones, equipos y servicios en uso debe existir evidencia que asegure el cumplimiento de los parámetros y límites de operación de las variables críticas. Adicionalmente, deben documentarse los procesos de calibración, limpieza, mantenimiento preventivo, de operación y de capacitación del personal.

¹²NOM-026-STPS-2008, disponible en:
["https://www.dof.gob.mx/normasOficiales/4327/salud/salud.htm"](https://www.dof.gob.mx/normasOficiales/4327/salud/salud.htm)

Sección 2.2.

Sistema motriz de un robot móvil Omnidireccional

El sistema de tren motriz de un robot móvil es un conjunto de elementos constituido principalmente de componentes mecánicos y electrónicos, que le permite tener la capacidad de realizar sus movimientos para desplazarse de un punto inicial a uno final. Es el sistema principal en el que se basa la estructura de un robot que le permite brindar la estabilidad y control necesarios para realizar los movimientos que le permitan cumplir sus tareas, siendo en el caso de un robot omnidireccional la cualidad de desplazarse sobre direcciones horizontales y verticales, girar en 360° sobre algún eje y maniobrar en direcciones diagonales para maniobrar mejor en entornos de tamaño reducido o que impliquen múltiples pasillos u obstáculos. Observemos, por ejemplo, el robot móvil omnidireccional con brazo robótico que se muestra en la figura 2.4.



Figura 2.4: Ejemplo de un robot móvil omnidireccional con brazo robótico.

Imagen obtenida en:
”<https://robotnik.eu>”.

El tren motriz de un robot móvil cuenta con un sistema de alimentación o energía eléctrica brindado por medio de una batería, que será necesaria para poder activar los motores eléctricos que le darán la potencia para poder ejecutar los movimientos, también energizará otros sensores y componentes electrónicos necesarios para que el robot pueda interactuar con su entorno y poder tomar decisiones o acciones establecidas previamente. Por ejemplo, un sensor de obstáculos o de detección de distancia puede influir con un actuador o sistema de frenado para reducir la velocidad en su totalidad en medio de una determinada tarea y poder trazar alguna ruta alterna o esperar un determinado tiempo en estado de reposo antes de continuar con su tarea.

Por lo general, los robots móviles utilizan ruedas convencionales y en ocasiones orugas o piernas. Sin embargo, el enfoque que se desarrollará a lo largo de esta tesis se basa en proponer nuestro robot con la capacidad de maniobrar en giros de 360° en sentido horario y antihorario, movimientos diagonales, horizontales y verticales para brindar una buena maniobrabilidad en un entorno hospitalario, por lo que utilizaremos ruedas omnidireccionales, que son capaces de brindarnos esta movilidad como se ha estado planteado con anterioridad.

El sistema motriz de un robot móvil es crucial para su capacidad de realizar movimientos continuos que podrán brindar la posibilidad de realizar tareas específicas dentro de un entorno hospitalario, enfocado el propósito de realizar determinadas tareas específicas, en el propósito de nuestro robot la capacidad de navegar en un entorno hospitalario, sin dejar de tomar en consideración las normativas ya existentes de seguridad y cuidados mencionados durante las NOM, para ayudar en tareas de un hospital.

Rueda omnidireccional de tipo *mecanum*

Existen dos tipos de ruedas omnidireccionales que proporcionan un movimiento omnidireccional, siendo la primera conocida como rueda sueca o rueda de *Ilon*, coloquialmente conocida solo como rueda omnidireccional, patentada por el ingeniero sueco *Bengt Ilon* en 1973, este prototipo se ejemplifica mejor en la figura 2.5.

*“La rueda sueca es aquella rueda que contiene en su periferia una corona de rodillos, colocados a cierto ángulo respecto a su eje de giro”.*¹³

¹³Adán Esteban Suárez Arriaga, Alina María Sánchez Balpueta, 2015, Plataforma móvil omnidireccional de cuatro llantas suecas (Mecanum) en configuración “AB”, [tesis de licenciatura, Universidad Nacional Autónoma De México], disponible en: http://www.ptolomeo.unam.mx:8080/jspui/bitstream/132.248.52.100/8392/1/Tesis_Perfect.pdf



Figura 2.5: Rueda Suecano o rueda de Ilon.

Imagen obtenida en:

"<https://www.piscinarobots.es/rueda-omnidireccional-de-152-mm-cojinete-de-laton-para-rodillos-14085>"

Por otra parte, la rueda omnidireccional de tipo *mecanum*, tiene la característica de tener rodillos alrededor de la rueda, cuya función es ser el soporte principal, mientras que el rodillo es un tambor montado en el tubo como se muestra en la figura 2.6, la cual a diferencia de la rueda sueca tiene la capacidad de desplazamiento lineal y angular a 45° grados.

La disposición de los rodillos en ángulos oblicuos permite que las ruedas *mecanum* generen fuerzas laterales cuando se mueven. Al variar las velocidades y las direcciones de rotación de las ruedas individuales, se pueden lograr diferentes tipos de movimiento, como desplazamiento lateral, giros en el lugar y desplazamientos diagonales.



Figura 2.6: Rueda de tipo *mecanum*.

Imagen obtenida en:

"<https://es.lamdageeks.com/mecanum-wheeled-robot-design-applications/>"

Esto permite que las ruedas se muevan en dos direcciones y se pueda mover conforme a un sistema holónomo, asegurando que puedan moverse instantáneamente en cualquier dirección.

Un sistema holónomo, tiene una restricción que se puede expresar como una función de las coordenadas generalizadas con respecto al tiempo, se dice entonces que es un sistema holónomo. Por tanto, si q_i , donde $i = 0, 1, 2, \dots, n$; son las coordenadas generalizadas del sistema, entonces la restricción geométrica se podrá escribir como una función de la forma:

$$f_i(q_0, \dots, q_n, t) = 0 \quad (2.1)$$

Entonces decimos que un sistema mecánico cuyas restricciones son todas holónomicas, se dice que es un sistema holónomo.¹⁴

Como se observa en la figura 2.7, al tener una dirección de giro en la rueda, por ejemplo, en el eje "Y", tendremos a la para una componente horizontal sobre un eje "X", y en consecuencia también tendremos una fuerza resultante representada con un vector en un ángulo a 45°.

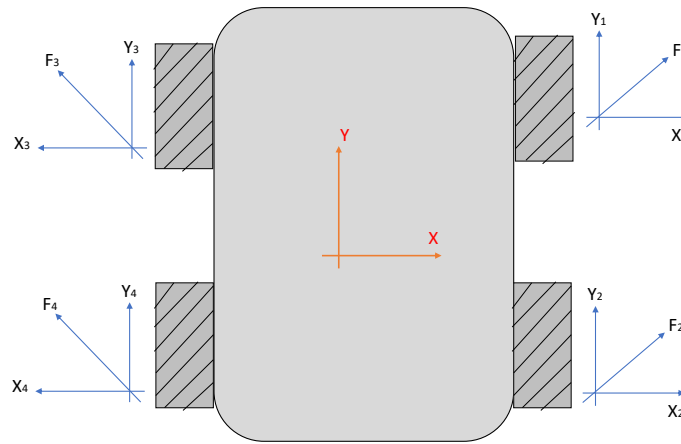


Figura 2.7: Direcciones para una plataforma compuesta de cuatro ruedas Mecanum.

Las ruedas *mecanum* se utilizan en una amplia gama de aplicaciones, especialmente en robótica móvil y vehículos autónomos. Permiten una mayor agilidad y maniobrabilidad, lo que facilita la navegación en espacios reducidos por medio de la realización de movimientos complejos, lo cual es conveniente al momento de implementar un robot dentro de un entorno hospitalario. Por otra parte, las ruedas *mecanum* también se utilizan en aplicaciones industriales, como transportadores y manipuladores robóticos, donde se requiere un alto grado de movilidad en entornos reducidos de espacio a modo de desplazarse de una manera rápida y efectiva, como por ejemplo aplicaciones en montacargas como se observa en la figura 2.8.

¹⁴Eduardo de la Rosa Bocanegra, 2009, Tesis para obtener el grado de doctor en ciencias e ingeniería, Universidad Nacional de Trujillo, Escuela de posgrado, pp.30. Disponible en "<https://dspace.unitru.edu.pe/server/api/core/bitstreams/c93dfdcdb-dab3-42b6-85d2-1460c52ecf79/content>"



Figura 2.8: Ejemplo de un montacargas industrial con ruedas *mecanum*.

Imagen obtenida en:

"<https://www.imssupply.com/blog/2018/04/12/Mecanum-Wheels-Means-Your-Forklift-Can-Moonwalk.html>"

En resumen, las ruedas omnidireccionales de tipo *mecanum* son un tipo especial de ruedas utilizadas en robótica y vehículos móviles para permitir movimientos multidireccionales. A diferencia de las ruedas convencionales, que solo pueden moverse en un sentido delantero y de reversa con la necesidad de maniobrar para realizar un giro, por esta cuestión las ruedas *mecanum* pueden moverse en todas las direcciones, incluyendo desplazamientos laterales y diagonales, debido a que se produce un desplazamiento lateral que al interactuar con las otras ruedas cancela o produce un vector de desplazamiento en la dirección que se desea, por lo que los hace altamente maniobrables y versátiles en entornos cerrados, por lo que la idea de basar el tren motriz con estas ruedas que nos permitirá desarrollar el tren motriz de nuestro robot para maniobrar en un entorno hospitalario.

Chasis del robot

El diseño del prototipo para el chasis fue considerado a las condiciones hospitalarias mencionadas en las normas oficiales mexicanas, el diseño del chasis a emplear, fue planeado y diseñado por el equipo de AYUDAME 2.0, en colaboración con el equipo asignado en el centro de Ingeniería y diseño industrial, a cargo del Dr. Mauricio Reyes Castillo. En las siguientes figuras se observa el desarrollo que se llevó a cabo para la realización del chasis que servirá de base para el robot. Posteriormente, en este diseño se basará la base y el desarrollo electrónico del robot. La figura 2.9, es una versión preliminar diseñada e impresa sobre el modelo que se planteó para el robot asistente a hospitales, considerando el tren motriz con ruedas omnidireccionales de tipo *mecanum*.



Figura 2.9: Diseño para el prototipo del robot.
Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0.

En la siguiente figura 2.10, se llevó a cabo el prototipo físico del robot asistente a hospitales. Este prototipo fue llevado a cabo de igual manera por el equipo de AYUDAME 2.0 compuesto por diferentes alumnos de servicio social.



Figura 2.10: Primer versión del chasis y soporte del robot asistente.
Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0.

En la figura 2.11 se observa el mismo prototipo para el robot de servicio en el que se implementará la parte electrónica para el diseño del tren motriz basado en las ruedas omni-direccionales de tipo "*mecanum*", que será el primer prototipo en el que se trabajara para nuestro robot asistente a hospitales, este ya fijado sobre el chasis del robot, con material de impresión 3D para el soporte de las secciones donde se encontrará la pantalla del robot, cajones para colocar los insumos como medicamentos o materiales.



Figura 2.11: Segunda versión del chasis y soporte del robot asistente.
Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0.

Todos los diseños correspondientes a esta planeación de este robot y proporcionados por el equipo AYUDAME 2.0, se encuentran en la sección del apéndice B, estos planos detallan más en profundidad las dimensiones que se tomaron para la implementación y ensamble del robot que fueron considerados en el proceso de desarrollo del robot.

Movimiento Omnidireccional de un robot móvil.

El movimiento omnidireccional en la aplicación de los robots móviles es cada vez más común debido a la gran versatilidad y ventajas que propone este sistema de movimientos para los robots móviles, que representa una mayor libertad de movimiento para desplazarse. Como se detalló anteriormente, el movimiento omnidireccional se debe al uso de 4 ruedas *mecanum* que pueden viajar en cualquier dirección desde cualquier ángulo sin girar de antemano, permitiendo desplazamientos horizontales, verticales, diagonales y girar sobre un eje de referencia. Como se muestra en la figura 2.12 tendremos 10 principales formas de movimiento.

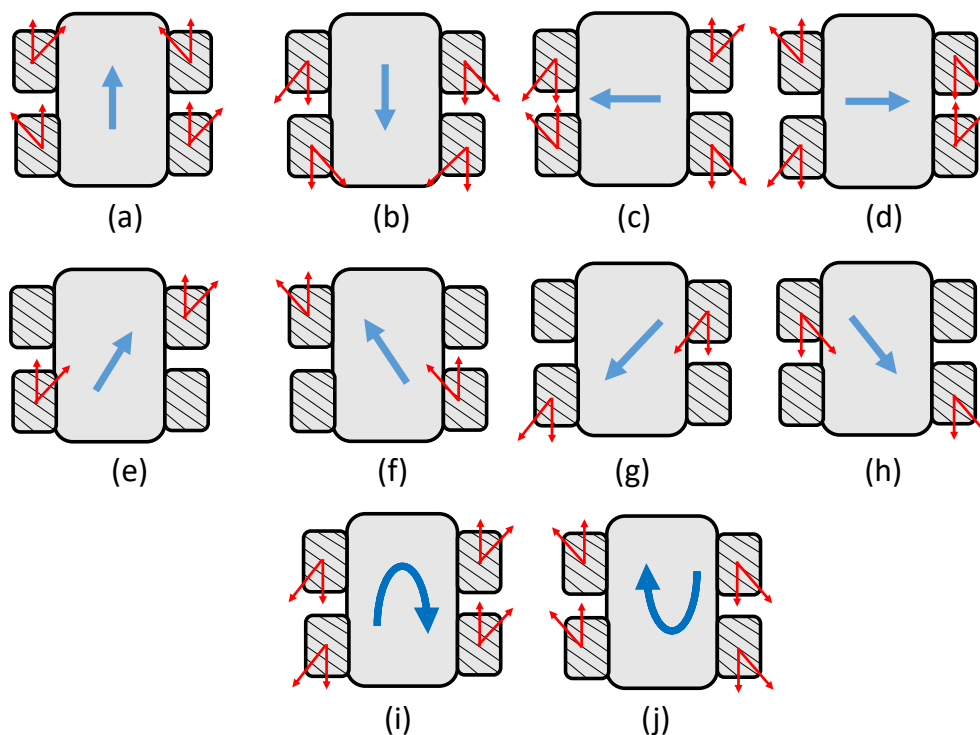


Figura 2.12: Tipos de movimientos con una plataforma omnidireccional.

De la figura 2.12(a). Se observa que la dirección de movimiento es hacia adelante debido a que todos los motores están girando hacia un mismo sentido de referencia en el plano, cancelando las componentes resultantes que harían que el movimiento sea horizontal, dejando un movimiento vertical delantero.

De la figura 2.12 (b). Observamos que, de igual manera, el movimiento resultante es vertical; sin embargo, si se cambia la dirección del giro de los motores, el movimiento resultante solo será en sentido contrario en el plano.

De la figura 2.12 (c) y (d). Se observa en ambos casos que tenemos un movimiento horizontal en direcciones que se pueden denominar como derecha e izquierda. Los movimientos horizontales tienen la característica de que los sentidos de giro están determinados porque las ruedas correspondientes al lado lateral en el que se encuentran deberán girar en sentidos contrarios al modo horario y antihorario deseados. Es decir que en este caso tendremos las ruedas del lado izquierdo girando de manera que ambas direcciones coincidan en un vector de movimiento resultante a la dirección izquierda y derecha respectivamente, cancelando las resultantes de las componentes verticales y empujando en la dirección resultante hacia el lado que se desea ir, en el caso del inciso (c) hacia la izquierda y (d) hacia la derecha respectivamente.

De la figura 2.12 (e), (f), (g), (h). Son movimientos con vectores resultantes en diagonal, resultado de tener funcionando únicamente dos ruedas omnidireccionales en sentido diagonal hacia una misma dirección, teniendo cuatro posibles casos que se muestran en los incisos, (e), (f), (g) y (h), donde para cada caso solo se deben tener ambas ruedas opuestas girando en un mismo sentido, mientras las otras ruedas restantes permanecerán sin movimiento, por lo que la componente resultante del ángulo de 45° permitirá el movimiento en sentido diagonal deseado.

De la figura 2.12 (i) y (j). Tenemos un movimiento de giro en 360° para la plataforma del robot omnidireccional, para lograr este sentido de giro ambas ruedas de cada lateral debe girar en el mismo sentido; sin embargo, cada lateral debe girar en sentido contrario el uno al otro, es decir, las dos ruedas laterales izquierdas deben girar en sentido contrario a las dos ruedas laterales derechas, esto dará como resultado un movimiento de giro en 360° sobre el eje del robot, en el caso del inciso (i) puede decirse que es en sentido acorde a las manecillas de reloj y en el inciso (j) contrario a las manecillas del reloj.

Resumiendo la información anterior, la caracterización de los movimientos anteriormente mencionados, tomaremos como referencia el movimiento y el sentido de giro que deben tener, como se muestra en la tabla 2.1, con la siguiente simbología abreviada para simplificar los sentidos de giro de las ruedas, tomando como referencia la figura 2.13, para evitar confundir los movimientos que se tomarán como referencia para la planificación del robot y se utilizara la siguiente abreviatura para simplificar la tabla 2.1.

- **SM:** Sin movimiento
- **SH:** Sentido horario o acorde a las manecillas de reloj
- **SAH:** Sentido antihorario o contrario a las manecillas de reloj

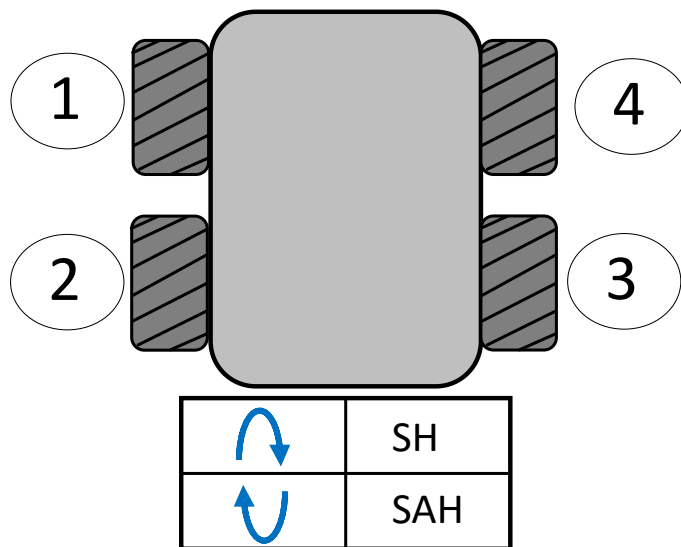


Figura 2.13: Propuesta de numeración de ruedas para definir la tabla de movimientos.

Movimiento	Motor 1	Motor 2	Motor 3	Motor 4
Alto	SM	SM	SM	SM
Adelante	SH	SH	SH	SH
Atras	SAH	SAH	SAH	SAH
Derecha	SH	SAH	SH	SAH
Izquierda	SAH	SH	SAH	SH
Diagonal 1	SH	SM	SH	SM
Diagonal 2	SAH	SM	SAH	SM
Diagonal 3	SM	SH	SM	SH
Diagonal 4	SM	SAH	SM	SAH
Giro de 360° en sentido anti horario	SH	SH	SAH	SAH
Giro de 360° en sentido horario	SAH	SAH	SH	SH

Tabla 2.1: Caracterización de movimientos para 4 ruedas Mecanum.

Controlador de motores de corriente directa Talon SR.

El driver controlador para motores de corriente directa modelo Talon SR, que se muestra en la siguiente figura 2.14, es un dispositivo que controla la dirección de giro y control de velocidad para motores de corriente directa, que serán de utilidad para configurar la dirección de giros mencionados en la tabla 2.1, mencionada anteriormente. Por otra parte también sirve ayuda a controlar las sobrecorrientes del motor.



Figura 2.14: Controlador de motores de corriente directa modelo Talon SR

Imagen obtenida de:

”http://files.andymark.com/Talon_User_Manual_1_3.pdf”

El funcionamiento de este dispositivo es básicamente controlar la corriente directa de un motor de corriente directa mediante una señal modulada por ancho de pulsos o conocida como PWM para determinar la velocidad y sentido de giro del motor.

Como se observa en la figura 2.14, el talon principalmente tiene como características una terminal de lado izquierdo con polaridad para alimentar el dispositivo con un rango de voltaje de 6 a 28 [V]. Tiene capacidad para soportar una corriente de hasta 36 [A], finalmente tiene una terminal de tres pines de los cuales tendremos una terminal con referencia a tierra y una para la señal PWM con la que controlaremos el dispositivo.

En el centro tenemos un disipador de temperatura al que se le puede adaptar un ventilador cuya principal adición es disipar altas temperaturas en el controlador. De lado derecho tenemos la terminal de salida donde se conecta al motor de corriente directa. Este controlador cuenta con led bicolor que ilumina en verde y rojo respectivamente para identificar el sentido de dirección de giro del motor y una intermitencia de color naranja que indica que no está actuando el motor o porque la señal de modulación enviada no está llegando adecuadamente.

Adicionalmente, este dispositivo puede ser calibrado siempre y cuando la calibración se encuentre un intervalo de 1-2 milisegundos, para la calibración se debe enviar una señal PWM desde el intervalo de lectura para establecer nuestro mínimo y máximo de nuestra señal, es decir en un ciclo de trabajo de 0% a 100 %.

Se puede observar mejor la información resumida de la hoja de especificaciones en la tabla 2.2.

Especificación	Minimo	Máximo
Voltaje de entrada de motor	6 [V]	28 [V]
Voltaje de entrada lógico	0.3 [V]	5.3 [V]
Corriente continua	1 [A]	60 [A]
Pico de corriente	1 [A]	100 [A]

Tabla 2.2: Características del controlador de motores de corriente directa modelo Talon SR.

Sin embargo, la principal desventaja de este controlador radica en que este modelo en particular fue discontinuado y retirado del mercado desde hace tiempo, por lo que actualmente no es posible contemplar con un remplazo ante posibles fallas técnicas en este controlador, por lo que será necesario contemplar una segunda opción de controlador para los motores en caso de que este modelo pueda fallar. Finalmente, las especificaciones técnicas están anexadas en el apéndice B de este documento.

Controlador de motores de corriente directa BTS7960.

El siguiente controlador de motor de corriente directa, que modelo BTS7960, es un controlador opcional que contemplamos debido a la falla del controlador modelo Talon SR y no poder encontrar algún remplazo debido a que fue retirado del mercado, por lo que encontrar un repuesto es complicado, por este motivo y en la búsqueda de un controlador de motor de corriente directa que pueda soportar los motores de nuestro tren motriz, encontramos un nuevo controlador a precio accesible y de fácil adquisición, este es el modelo que se muestra en la figura 2.15. Este controlador funciona de una manera diferente al controlador Talon

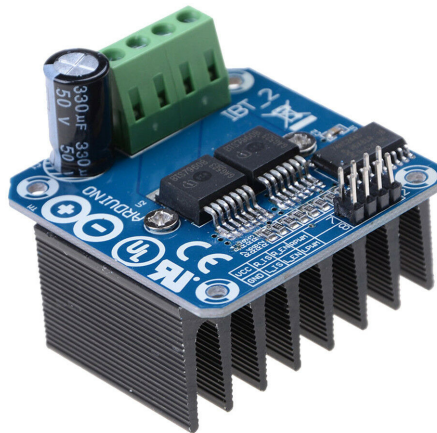


Figura 2.15: Controlador de motoresde corriente directa modelo BTS7960.

Imagen obtenida de:

”<https://udvabony.com/product/bts7960-motor-driver-43a/>

SR, como observamos previamente, el funcionamiento del controlador modelo Talon SR, se controlaba con un pin por medio del ciclo de trabajo la dirección de giro y velocidad, esto debido a que la principal diferencia se encuentra en como interactúa el ciclo de trabajo en ambos controladores y a los circuitos integrados propios de cada controlador, mientras que con el controlador de motores de corriente directa Talon SR la interacción de la señal PWM y el ciclo de trabajo controla la velocidad y sentido de giro en un intervalo limitado entre 1 a 2 [KHz], por el contrario, para este controlador el ciclo de trabajo únicamente controla la velocidad en un intervalo máximo de 25 [KHz], mientras que la dirección depende del canal que esté enviando la señal PWM, ya que ocupa dos pines para dirección en sentido horario y antihorario, contrario al controlador de motores Talon SR, donde solo se ocupa un de señal PWM para el control del motor, por lo que la desventaja radica principalmente en que soporta picos de corriente menores y que ocupa una mayor cantidad de pines, estos se encuentran detallados en la tabla 2.3.

PIN	Funcion	Descripcion
1	R PWM	Señal de activación PWM de dirección delantera
2	L PWM	Señal de activación PWM de dirección reversa
3	R EN	Activación y apagado de pin delantero de dirección delantera
4	L EN	Activación y apagado de pin delantero de dirección reversa
5	R IS	Accionamiento de alarma dirección delantera
6	L IS	Accionamiento de alarma dirección reversa
7	Vcc	Alimentación de microcontrolador
8	GND	Conección a tierra

Tabla 2.3: Especificaciones de pines del controlador de motores de corriente directa modelo BTS7960.

Por otro lado, como se menciona anteriormente, la corriente que soporta es menor a la del controlador de motores de corriente directa modelo Talon SR, sin embargo, soporta picos de hasta 43 [A], y voltajes de entrada desde 6 a 27 [V], por lo que su operación para el tren motriz propuesto resulta conveniente.

Otra diferencia notable es que este controlador, no debe ser calibrado previamente para definir su funcionamiento, debido a que, en función de su voltaje de alimentación y ciclo de trabajo, la salida del voltaje del motor será de manera proporcional en este controlador.

Resumiendo las especificaciones de este controlador de motores de corriente directa modelo BTS7960 mostrado en la tabla 2.4, las especificaciones se encuentran referenciadas en el apéndice B de este documento.

Especificación	Minimo	Máximo
Voltaje de entrada de motor	6 [V]	45[V]
Voltaje de entrada lógico	0.3 [V]	5.3 [V]
Corriente continua	1 [A]	43 [A]
Pico de corriente	1 [A]	60 [A]

Tabla 2.4: Especificaciones del controlador de motores de corriente directa modelo BTS7960.

Haciendo una breve comparativa, podemos hacer una tabla comparativa mostrada en la tabla 2.5, entre ambos controladores de motores de corriente directa a modo de observar las principales diferencias y características, no obstante descartando alguna posible mejor alternativa que pueda presentarse en algún futuro.

Cracterísticas	Controlador Talon SR	Controlador BTS7960
Pines	3	8
Pico de corriente máxima	100 [A]	60 [A]
Corriente continua (máx/min)	1-60 [A]	1-43 [A]
Voltaje de alimentación (máx/min)	6 - 28 [V]	6 - 45 [V]
Voltaje lógico(máx/min)	0.3 - 5 [V]	0.3 - 5.3[V]
Rango de periodo de opeeación	1 - 2[KHz]	0 - 25 [KHz]
Calibración necesaria	si	no
Comercialmente disonible	no	si

Tabla 2.5: Comparativa de las principales características entre cada controlador de motores de corriente directa.

Finalmente, de manera adicional, en el apéndice C, se encontrarán una sección de códigos de ejemplo en lenguaje de programación *C* para la calibración del controlador de motor de corriente directa modelo Talon SR y un breve ejemplo para el controlador de motor de corriente directa modelo BTS7960.

Motor DC modelo AM802-001A

El modelo de motor seleccionado para nuestro tren motriz, debe tener la potencia necesaria para el soporte del chasis de nuestro robot, por lo que este modelo de motor de corriente directa que se muestra en la figura 2.16, o motor de corriente directa modelo AM802-001A, fue seleccionado como modelo del motor para el diseño del robot.



Figura 2.16: Motor de corriente directa AM800-001A.

El motor de corriente directa modelo AM802-001A, es un motor con las siguientes características que se observan en la tabla 2.6.

Especificación	Valor nominal
Voltaje de alimentación continuo	12 [V]
Capacidad de corriente continua máxima	67.9 [A]
Corriente de arranque	2.3 [A]
Potencia máxima	377 [W]
Capacidad máxima de velocidad máxima	5310 [RPM]
Eficiencia máxima	63 [%]

Tabla 2.6: Características del motor de corriente directa AM800-001A.

Como se observa, este modelo particular de motor de corriente directa, tiene capacidad para mover el chasis y el robot en general, únicamente puede ser alimentado mediante dos terminales positiva y negativa, totalmente compatibles con nuestros controladores de corriente directa anteriormente mencionados, es decir, para los modelos BTS7960 y Talon SR, por lo que la tarea de controlar la velocidad y torque del motor como su dirección de sentido de giro facilitarían los movimientos mencionados en la tabla 2.1 para nuestro robot. Las especificaciones técnicas de este motor se encuentran en la sección del apéndice B.

Sensor Ultrasonico US-100

Un sensor de distancia es un dispositivo electrónico que permiten obtener el valor de la distancia mediante el uso de ondas ultrasónicas, es decir, hay una onda emitida desde un cabezal, la cual será reflejada al entrar en contacto con alguna superficie u objeto, tomando lectura de esta misma onda el otro cabezal. Para obtener la distancia se toma como referencia el tiempo de entre la emisión y la recepción de la onda, haciendo una conversión de distancia con base en la unidad de segundos que se especifique para cada modelo de sensor.

La distancia está determinada comúnmente por:

$$L = \frac{1}{2} * T * C \quad (2.2)$$

Donde T es el periodo del inverso del tiempo entre la emisión y la recepción del sensor y C es una constante correspondiente a la velocidad del sonido que es asociada a: $C = 343[m/s]$.

El sensor ultrasónico US-100 que se muestra en la figura 2.17, es un modelo particular de sensor ultrasónico diferente a otros modelos, tomando por ejemplo uno de los modelos más conocidos que es el HC-SR04, conocido principalmente por su facilidad de utilizar con arduino, en comparación con este modelo la característica principal es que ofrece mayor rango para lecturas y que cuenta físicamente con 5 pines correspondientes a tierra (estas se encuentra en dos pines), Vcc, Trigger PIN y Echo PIN, sus funciones se detallan en la tabla 2.7.



Figura 2.17: Sensor ultrasonico modelo US100.

PIN	Nombre	Descripcion
1	Vcc	Alimentación del sensor
2	Trigger	Envia una señal de onda
3	Echo	Recibe la señal del retorno de la onda
4	GND	Referencia a tierra
5	GND	Referencia a tierra

Tabla 2.7: Funciones de los pines del sensor US-100.

Sus especificaciones técnicas resumidas son las que se muestran en la tabla 2.8:

Especificación	Minimo	Máximo
Voltaje de alimentación	2.4 [V]	5.5 [V]
Voltaje lógico	2.4 [V]	5.5 [V]
Corriente continua	0 [A]	0.02 [A]
Detección de distancia	0.002 [m]	4.5 [m]

Tabla 2.8: Especificaciones del sensor de distancia US-100.

La hoja de especificaciones completa se encuentra en la sección del apéndice B de este documento. Adicionalmente, en el apéndice C se encuentra un ejemplo de un código en lenguaje de programación en C, para el dispositivo Raspberry Pico.

Sensor de corriente ASC712

Este módulo es un sensor de corriente multifuncional que cuenta con un rango de lectura de corriente directa y también corriente alterna según el propósito que se desea utilizar y la configuración que se le dé para el uso del modelo de este sensor, sin embargo, dependiendo del modelo de circuito integrado que tenga, debido a que existen algunos circuitos integrados con capacidad de 5, 20 y 30 amperes de rango de medición según se indique en su circuito integrado, donde la sensibilidad cambiara para cada caso. Sin embargo, el que nosotros utilizaremos es el de rango de 20 [A] que se muestra en la figura 2.18, la capacidad de alta lectura en todos los casos es gracias a un sensor electromagnético de tipo hall, que además de tener una implementación sencilla de lectura analógica contando únicamente con los pines de alimentación de poder, tierra y lectura analógica, así como dos terminales para hacer la conexión en serie de la carga de la cual se desea tomar la lectura de corriente, como se muestra en la tabla 2.9, resulta útil y sencillo para tomar las lecturas del proceso de descarga de la batería de nuestro robot y poder interpolar el consumo de potencia de nuestro robot y establecer tiempos de carga y descarga.

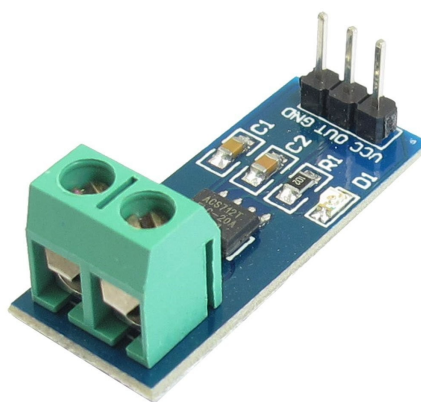


Figura 2.18: Sensor de corriente ACS712.

Imagen obtenida de:

"[https://articulo.mercadolibre.com.mx/](https://articulo.mercadolibre.com.mx/MLM-551036682-sensor-medidor-de-corriente-hall-ac712-5-aarduinopic-avr-JM)

MLM-551036682-sensor-medidor-de-corriente-hall-ac712-5-aarduinopic-avr-
JM"

PIN	Nombre	Descripcion
1	Vcc	Alimentación del sensor
1	Out	Lectura analógica del sensor
3	GND	Referencia a tierra
4	+	Referencia a entrada de la lectura de corriente
5	-	Referencia a salida de la lectura de corriente

Tabla 2.9: Funciones de los pines del sensor ACS712.

Las especificaciones técnicas resumidas de este componente se observan en la tabla 2.10. De manera adicional, en el apéndice C, se encuentra un código de programación en lenguaje C para ejemplificar el funcionamiento de este sensor, diseñado para el dispositivo Raspberry Pico.

Especificación	Mínimo	Máximo
Voltaje de alimentación	4.5 [V]	5.5 [V]
Corriente continua	0 [A]	0.13 [A]
Máxima corriente de lectura	0 [A]	20 [A]

Tabla 2.10: Especificaciones del sensor de corriente ACS712.

*”La salida de este dispositivo cuando la lectura de la corriente es cero. Para el voltaje de tensión unipolar nominalmente es la relación de $V_{cc}/2$, por lo que si la alimentación $V_{cc} = 5[V]$, entonces la salida continua es de 2.5 [V]”.*¹⁵

La lectura del sensor debe ser la relación directa dada por una ecuación de una sola variable, que sería la lectura de la corriente cuando el voltaje de salida es 2.5 [V]. Considerando que la lectura de la corriente es cero, por lo que expresando lo anterior en una ecuación sería entonces:

$$V[I] = m * I + 2.5 \quad (2.3)$$

Donde I es la lectura de corriente y m es una constante correspondiente a la sensibilidad del sensor dependiendo del rango de lectura, por lo que para el caso de nuestro circuito integrado correspondiente de 20 [A], entonces este valor es de $100 [\frac{mV}{A}]$. De la expresión anterior para obtener un valor de nuestra corriente despejando de la expresión anterior, entonces la nueva expresión de lectura de corriente sería entonces:

$$I = \frac{V - 2.5}{m} \quad (2.4)$$

Donde el voltaje de nuestro rango será la expresión del voltaje correspondiente por la variable V a nuestro valor de alimentación de la carga de nuestra batería que se desea medir.

¹⁵ ALLEGRO, microsystems, Fully Integrated, Hall-Effect-Based Linear Current Sensor IC ACS712. pp. 9. disponible en:
<https://docs.google.com/gview?embedded=true&url=www.allegromicro.com/-/media/files/datasheets/acs712-datasheet.pdf>

Módulo RTC DS3231

Este dispositivo es un reloj de tiempo real por su traducción del inglés *"real time clock"*, cuya principal función es guardar los registros de hora actual de manera continua con batería integrada para funcionar por prolongados tiempos de duración sin depender de la alimentación directa de una conexión de alimentación, debido a una batería integrada al módulo para que pueda guardar los registros de fecha y seguir tomando información del tiempo del mismo modo en que un reloj de mano funcionaria. Sin embargo, este modelo en particular, que es el que se muestra en la figura 2.19, es bastante compacto en comparación con otros modelos. Su funcionamiento es con ayuda del protocolo I2C. Este modelo en particular cuenta con 5 pines que se detallan en la tabla 2.11 para su funcionamiento.

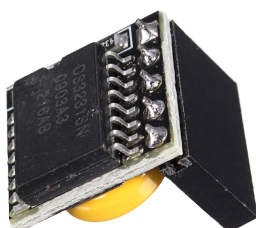


Figura 2.19: Reloj de tiempo real modelo DS3231.

Imagen obtenida de:

"[https:](https://alexnlld.com/product/ds3231-precision-rtc-memory-module-for-arduino/)

[//alexnlld.com/product/ds3231-precision-rtc-memory-module-for-arduino/](https://alexnlld.com/product/ds3231-precision-rtc-memory-module-for-arduino/)"

PIN	Nombre	Descripcion
1	Vcc	Alimentación del sensor
1	SDA	Pin de comunicación de datos
3	SCL	Pin de comunicación de reloj
4	SC	Propósito de programación de fabrica, no utilizable
5	GND	Referencia a tierra

Tabla 2.11: Funciones de los pines del reloj de tiempo real DS3231.

Las especificaciones técnicas resumidas se encuentran en la tabla 2.12 y la hoja de especificaciones completa se encuentra en la sección del apéndice B. Adicionalmente se encontrará un código de programación en lenguaje C, diseñado para el dispositivo Raspberry Pico, que incluye las características del uso del protocolo I2C del dispositivo. Cabe recordar que este protocolo involucra tener en consideración una resistencia de "*Pull Up*" para un funcionamiento adecuado.

Especificación	Mínimo	Máximo
Voltaje de alimentación(V_{CC})	2.3 [V]	5.5 [V]
Voltaje lógico de entrada SDA	$0.7 \cdot V_{CC}$ [V]	$V_{CC} + 0.3$ [V]
Voltaje lógico de entrada SCL	0.3 [V]	$0.3 \cdot V_{CC}$ [V]
Corriente continua	0 [A]	0.03 [A]

Tabla 2.12: Especificaciones del reloj de tiempo real DS3231.

Pantalla LCD 16X2

Una pantalla de tipo LCD, por sus abreviaturas del inglés "*Liquid Crystal Display*", o pantalla de tipo LCD 16x2 que se muestra en la figura, 2.20, es un monitor de cristal líquido que consta de 16 columnas y 2 filas de que facilita la representación de caracteres alfanuméricos. Cada carácter es de tamaño 5x8 píxeles, lo que significa que puede mostrar hasta 16 caracteres en una línea y hasta 2 líneas de texto, por lo que es útil al momento de mostrar información en mensajes que puede ser de utilidad. El display LCD 16x2 es ampliamente utilizado en proyectos electrónicos y sistemas embebidos por su tamaño compacto, relativa facilidad de uso con ayuda de un módulo y disponibilidad de bibliotecas y ejemplos que hacen que sea una opción popular en muchas aplicaciones que requieran entregar un mensaje a un usuario, como podría ser una lectura, un menú de decisiones, entre muchas otras aplicaciones.

Las características de esta pantalla es que cuenta con 16 Pines que funcionan para el manejo adecuado de esta pantalla, que por sí solo puede ser complicado su manejo si no se tienen conocimientos previos de su funcionamiento, sin embargo, este método convencional requiere de una gran cantidad de pines que podemos utilizar para otro propósito en el diseño del robot, por lo que buscando optimizar la cantidad de pines, es posible reducir el uso de estos 16 pines a 4 pines por medio de un módulo que utiliza el protocolo I2C, mediante un módulo expensor de pantalla LCD 16x2 modelo PCF8574 que se muestra en la figura 2.21, y las características de sus pines se muestran en la tabla 2.13. La necesidad de este módulo no es solo por simplificar el uso del display, sino de reducir la cantidad de pines que requieren para poder darle un uso más óptimo al robot al momento del diseño general de la electrónica.

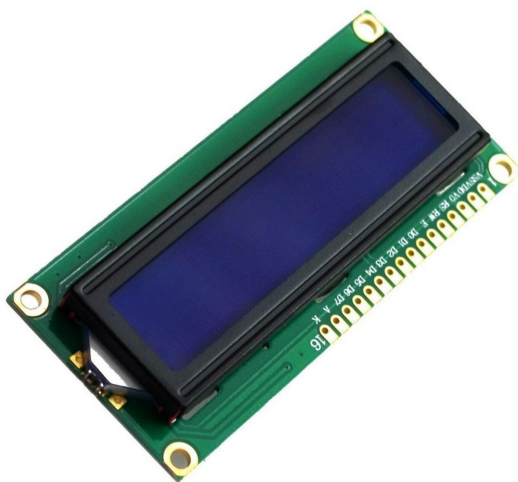


Figura 2.20: Display LCD 16x2.

Imagen obtenida de:

"https://articulo.mercadolibre.com.mx/MLM-602963638-display-lcd-16x2-1602a-para-arduino-pic-_JM"

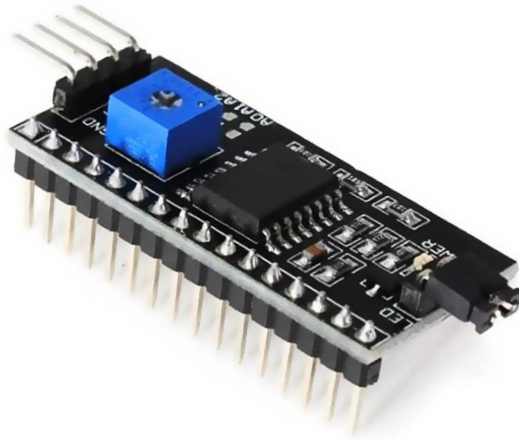


Figura 2.21: Módulo pcf8574 para pantalla LCD 16x2.

Imagen obtenida de:

"<https://www.tiendatec.es/maker-zone/modulos/1008-modulo-expansor-pcf8574-para-lcd1602-a-i2c-8472496015431.html>"

PIN	Nombre	Descripcion
1	Vcc	Alimentación del sensor
1	SDA	Pin de datos del sistema
3	SCL	Pin de reloj del sistema
4	GND	Referencia a tierra

Tabla 2.13: Funciones de los pines del módulo pcf8574 para la pantalla LCD 16x2.

Las especificaciones técnicas de este componente se muestran en la tabla 2.14.

Especificación	Minimo	Máximo
Voltaje de alimentación	2.5 [V]	6 [V]
Voltaje lógico	3.6 [V]	5.5 [V]
Corriente continua	0 [A]	0.03 [A]

Tabla 2.14: Especificaciones modulo para la pantalla LCD 16x2.

Voltímetro y amperímetro digital

Este dispositivo cuya imagen se observa en la figura ?? cumple la función de monitorear el voltaje y corriente de una carga en un rango de 0 a 100 [V] con una corriente máxima de lectura de 10 [A] y mostrar la lectura medida en un "Display de 7 segmentos".



Figura 2.22: Voltímetro amperímetro digital.

Imagen obtenida de:

"<https://leantec.es/tienda/voltmetro-amperimetro-100v-10a-digital-dc-rojo-azul-voltmeter-panel-sp/>"

Sus especificaciones técnicas resumidas se observan en la tabla 2.15. El rango de la resolución de lectura de voltaje es de 0.1 [V] y para la lectura de corriente es de 0.01 [A].

Especificación	Minimo	Máximo
Voltaje de alimentación	4.5 [V]	30 [V]
Rango de medición de voltaje	0 [V]	100 [V]
Rango de medición de corriente	0 [A]	10 [A]
Rango de funcionamiento de corriente continua	0.005 [A]	0.015 [A]

Tabla 2.15: Especificaciones modulo para la pantalla LCD 16x2.

Contro KY-023

Este componente funciona como un control simple para el tren motriz, si bien se plantea la idea de utilizar un control virtual accesible desde cualquier navegador web, este control físico fue propuesto como un método manual de y de fácil control para el robot en caso de requerir control inmediato en situaciones en las que no pueda conectarse directamente al control por medio desde navegador de internet. Este componente que se muestra en la figura 2.23 tiene la función de mover en determinadas direcciones el robot a las capacidades de este, es decir, no contempla en totalidad todos los movimientos pensados del robo debido a que su funcionamiento cuenta con dos ejes, conocidos como ejes "X" e "Y", que son controlados por dos potenciómetros de 10 $[K\Omega]$ y por un pulsador integrado.



Figura 2.23: Control modelo KY-023

Imagen obtenida de:

"<https://articulo.com.mx>"

La caracterización de movimientos disponibles se pueden mostrar en la tabla 2.16 con su respectivo movimiento según la configuración de este control, sin embargo, dependerá de la cantidad de bits de lectura analógicos, la configuración entre un rango de 0 y para nuestro caso con el dispositivo Raspberry Pi Pico, 4095 bits de lectura, por lo que estos serán el mínimo y máximo de la configuración, mientras que en un estado estático se encontrará en un valor de 2048.

Movimiento	Eje X	Eje Y
Alto	2047	2047
Delante	2047	4095
Reversa	2047	0
Derecha	4095	2047
Izquierda	0	2047
Diagonal 1	0	4095
Diagonal 2	4095	0
Diagonal 3	0	0
Diagonal 4	4095	4095

Tabla 2.16: Movimientos limitados para el control modelo KY-023.

En la figura 2.24 se puede observar de manera gráfica la configuración de los movimientos en función de los posibles ajustes del control KY-023, donde se observa la configuración de los movimientos adaptados al control KY-023. De igual forma, puede configurarse un giro de 360° si se presiona el pulsador integrando al control.

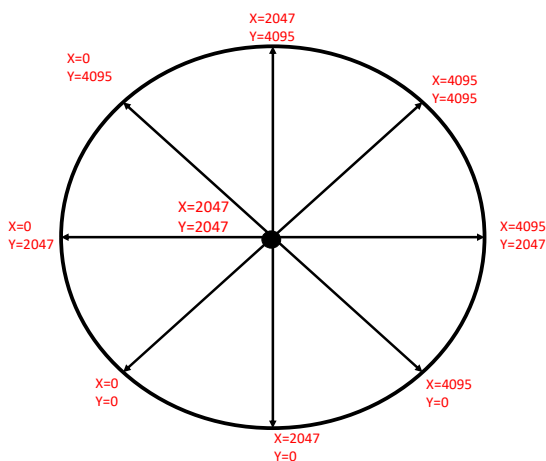


Figura 2.24: Configuración de movimientos del control KY-023

Este módulo cuenta con 5 pines que se detallan en la tabla 2.17. Las especificaciones completas de este componente se encuentran en la sección del apéndice B, adicionalmente en el apéndice C se incluye un código de programación en lenguaje C, para trabajar con el dispositivo Raspberry Pi Pico para ejemplificar.

PIN	Nombre	Descripcion
1	GND	Pin de referencia a tierra
1	Vcc	Pin de alimentación del componente
3	Eje Y	Pin de referencia del eje en Y
4	Eje X	Pin de referencia del eje en X
5	Botón pulsador	Es un botón integrado

Tabla 2.17: Funciones de los pines del componente KY-023.

Las especificaciones completas de este componente se encuentran en la sección del apéndice B, adicionalmente en el apéndice C se incluye un código de programación en lenguaje C, para trabajar con el dispositivo raspberry Pi Pico para ejemplificar.

Tira Led WS2812B

La inclusión de la tira de luces led es más bien un detalle más estético que un componente importante en el tren motriz del robot. Su propósito no es solo dar una impresión visual agradable a la vista del robot, ya que también tiene el propósito de dar indicaciones sobre el estado de acción del robot o un indicador de emergencia. Esta tira es similar a la que se muestra en la figura 2.25, y tiene la característica contraria a otras tiras leds donde se debía controlar el RGB individualmente por intensidad individual de vía para ajustar el color, aquí se controla por un único pin de transmisión de datos, mediante protocolo UART, por lo que sus pines especificados en la tabla 2.18, por lo que esta tira tiene la ventaja de controlar cualquier intensidad de secuencia o color por solo un pin, sin embargo, al se una tira de 150 leds de tipo RGB y SMD, el consumo de corriente ser el adecuado, ya que consumirá mínimo 2 [A].



Figura 2.25: Tira led RGB WS2812B.

Imagen obtenida de:

https://articulo.mercadolibre.com.uy/MLU-446712267-kit-rollo-rgb-tira-led-5-mts-ccontrol-remoto-efectos-220v-_JM

PIN	Nombre	Descripcion
1	Vcc	Pin de alimentación
1	Tx	Pin de transmisión de datos
3	GND	Pin de referencia a tierra

Tabla 2.18: Funciones de los pines de la tira led WS2812B.

La tabla de datos técnicos muestra las especificaciones técnicas resumidas de esta tira led, que se muestran en la tabla 2.19. Como se había mencionado, esta tira cambia la configuración de la iluminación de la tira led mediante un Pin de bus de datos para controlar las secuencias o color, por lo que puede ser útil para indicar mediante el color el estado del

robot en el que se encuentra, por ejemplo si se encuentra en movimiento o si se encuentra sin movimiento, así como indicar algún mensaje de alerta por poca carga en la batería o problemas con algún controlador.

La tabla 2.19 muestra sus especificaciones técnicas.

Especificación	Mínimo	Máximo
Voltaje de alimentación	2.5 [V]	6 [V]
Voltaje lógico	3.6 [V]	5.5 [V]
Corriente continua	0 [A]	0.03 [A]

Tabla 2.19: Especificaciones del módulo para la tira led RGB WS2812B.

La hoja de especificaciones completa se encuentra en la sección del apéndice B. Adicionalmente, en el apéndice C se encuentra un código de ejemplo de programación en lenguaje *C* para el dispositivo Raspberry Pi Pico, para el funcionamiento de esta tira led con una secuencia de ejemplo. Como se mencionó, se requiere de un pin con comunicación UART, por medio de un canal de transmisión de datos o *Tx*, debido a que la información transmitida para el control de secuencia y color se maneja por medio de la información de dirección de los registros.

La secuencia de indicación de estados se puede caracterizar por medio de la tabla 2.20.

Descripción	Color	Código de color RGB
En alto	Blanco	RGB(255, 255, 255)
En movimiento	Azul	RGB(0, 0, 255)
Batería Cargada	Verde	RGB(0, 255, 0)
Batería Baja	Rojo	RGB(255, 0, 0)
Objeto cerca del sensor	Naranja	RGB(255, 75, 10)
Inactividad	Púrpura	RGB(255, 0, 255)
Robot en servicio	Secuencia variable	RGB(n, n, n)

Tabla 2.20: Código de color y descripción de acción para la tira led RGB WS2812B.

Botón de paro de emergencia.

La selección de un botón de paro similar al de la figura 2.26, es como su nombre lo indica un botón de paro de emergencia para poder detener la alimentación del robot en caso de seguridad o una falla y detener completamente el movimiento del robot. Su tipo de acción es por enclavamiento, de contacto normalmente cerrado y tolerancia de 10 [A] de corriente.



Figura 2.26: Botón de paro de emergencia.

Imagen obtenida de:

`"https://articulo.mercadolibre.com.mx/`

`MLM-594492515-boton-paro-de-emergencia-con-enclavamiento-_JM"`

Selección de cables y terminales.

Selección de cables para la conexión a la batería y motores: Cable dúplex 14 AWG, bicolor, color rojo y negro para conexión en los motores.

Selección de cables para la conexión entre sensores y la pacb: Cable dúplex 22 AWG, bicolor, color rojo y negro para conexión con sensores y actuadores.

Calibre de Cable	Corriente soportada	Potencia máxima	Temperatura máxima
14	15 [A]	3600 [W]	60°C
22	5 [A]	1200 [W]	60 °C

Tabla 2.21: Breve tabla de especificaciones para los calibres de cable seleccionados.

Terminal de conexión: Terminal de latón de ojillo, con forro color amarillo, para conexión en nodos destinados a la conexión de la batería, motores y otros elementos.

El sistema de control de un robot móvil es aquel que le permite controlar y dirigir los movimientos, acciones y respuestas del robot con base en la lectura de sus sensores de proximidad y controlar la velocidad o dirección en función de lo que solicite el usuario.

Para nuestro sistema de control, utilizaremos como computadora principal el dispositivo *Raspberry Pi 4 Modelo B*, y como microprocesador auxiliar el modelo "RP2040" del dispositivo *Raspberry Pi Pico*, de manera adicional se plantea utilizar un dispositivo "Jetson nano", la cual no se abordó en profundidad en el planteamiento de la presente tesis, debido a que su función será el utilizar el algoritmo de navegación con "*Slam-ROS*" para el algoritmo de navegación autónoma, por lo que este enfoque se centró más en una navegación controlada por un control desde navegador y un control analógico con el control "ky-023"

El sistema de control de un robot móvil también puede incluir un conjunto de algoritmos de control y sistemas de retroalimentación que le permiten ajustar su movimiento y acciones en función de su entorno, por lo que se propondrá una base desde un diagrama de flujo para implementar en un código principal, dentro de algunas acciones donde por ejemplo, si el robot móvil detecta un obstáculo en su camino, el sistema de control puede ajustar su velocidad, dirección y trayectoria para evitar una colisión, o bien, detener cualquier movimiento que tenga indicado, así como si hay algún problema con la batería o con la comunicación con *ROS*.

Entonces el sistema de control de un robot móvil es una parte crítica del diseño del robot y es responsable de garantizar un movimiento preciso, seguro y eficiente del robot, ya que como observamos en algunas consideraciones de seguridad, deben tener en cuenta consideraciones para no obstruir con señalizaciones o salidas de emergencia.

Ahora a continuación detallaremos más a profundidad en los componentes electrónicos y dispositivos seleccionados que serán la base de control de nuestro robot móvil omnidireccional asistente en entornos hospitalarios, se detallará más en profundidad sobre lo que es *ROS*, por que utilizamos *ROS 2* y la comunicación con el microcontrolador "RP2040" por medio de *Micro-ROS*.

Raspberry PI Pico

El dispositivo Raspberry Pi Pico, que se muestra en la figura 2.27 es un SOC, por su abreviatura del inglés *"System on chip"*, o sistema en un chip, conformado principalmente por el microcontrolador RP2040 que se muestra en la figura 2.27, y la arquitectura interna de las características de este circuito integrado se muestra en la figura 2.28, permite el desarrollo de múltiples proyectos de electrónica analógica y digital, por ejemplo con los componentes mencionados anteriormente, desde la pantalla LCD, hasta los controladores de motores de corriente directa, el reloj de tiempo real, entre muchas más aplicaciones. Proporciona un entorno de trabajo accesible que brinda una facilidad de uso en lenguajes como *"Python"* por medio de una interfaz de usuario conocida por el entorno de *MicroPython* y en lenguaje de programación *C/C++*.

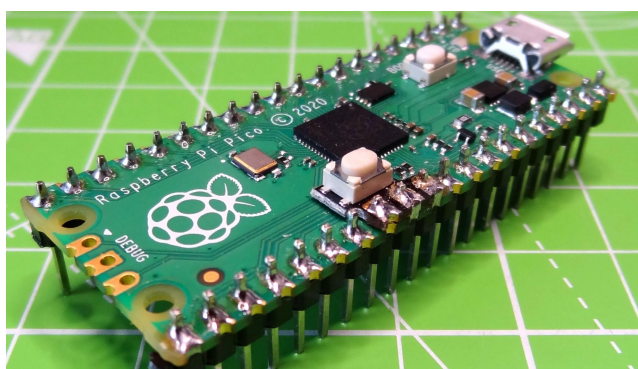


Figura 2.27: Raspberry Pi Pico.

Imagen obtenida de:

"<https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf>"

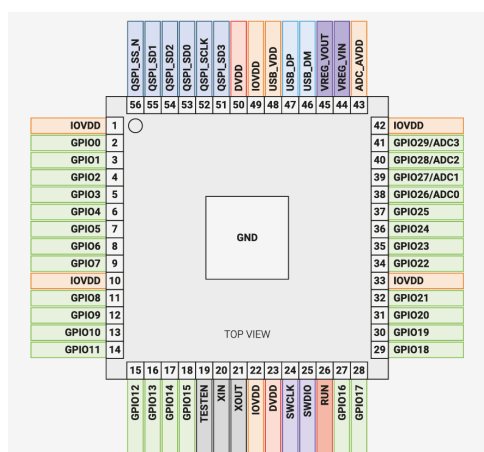


Figura 2.28: Circuito Integrado RP2040.

Imagen obtenida de:

"<https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>"

La configuración de los 43 pines disponibles del dispositivo Raspberry Pi Pico se observan en la figura 2.29.

Figura 2.29: Pinaut del dispositivo Raspberry Pi Pico.

"<https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf>"

Raspberry PI 4 MODEL B

El dispositivo *Raspberry Pi 4 MODEL B* que se muestra en la figura 2.30, es un *SOC*, o también conocido como un "dispositivo en un chip", que es capaz de manejar entornos de sistemas operativos mayormente basados en *Linux*, principalmente enfocado a su sistema operativo *Raspbian*. Desempeñando un entorno útil para realizar trabajo de una computadora de escritorio, pero en un dispositivo de bajo consumo de energía, en una ventaja adicional, no requiere un espacio amplio de trabajo, pues tiene un tamaño muy reducido. Por otra parte, también cuenta con pines para desarrollar múltiples aplicaciones de electrónica digital, similar al dispositivo *Raspberry Pi Pico*, sin embargo, tiene más restricciones al tener menos pines, también menos pines más restringidos si se desea trabajar con otros protocolos como UART, I2C, ISP, etc. Sin embargo, ofrece otras aplicaciones para trabajar con este dispositivo, ya que es ideal para desarrollar proyectos de robótica, o comunicación entre dispositivos.

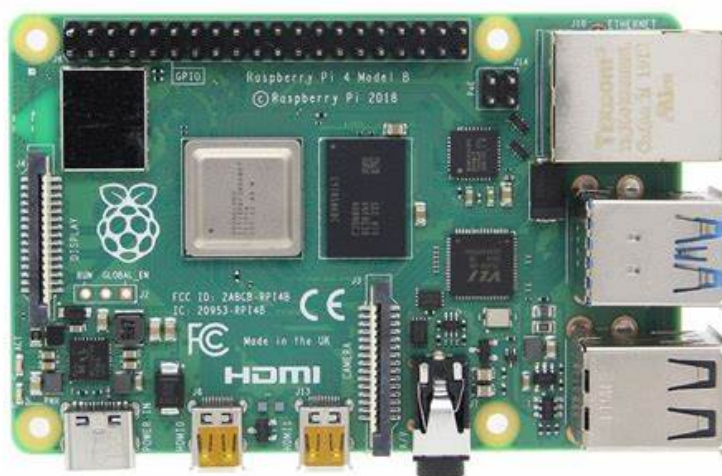


Figura 2.30: Raspberry Pi 4 Modelo B.

Imagen obtenida de:

"<https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf>"

Entre sus principales especificaciones técnicas tenemos:

CPU: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz

Memoria RAM: 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM (depende del modelo)

Storage: microSD card slot para cargar un SO y archivos

Conectividad: Gigabit Ethernet, dual-band 802.11ac wireless, Bluetooth 5.0, Alimentación por USB-C , dos puertos USB 3.0, dos puertos USB 2.0, dos puertos micro-HDMI, 3.5mm audio jack

Dimensiones: 88 x 58 x 19.5 mm, 46 g.

Su arquitectura física y guía de pines con sus pines reservados se muestran en la figura 2.31.

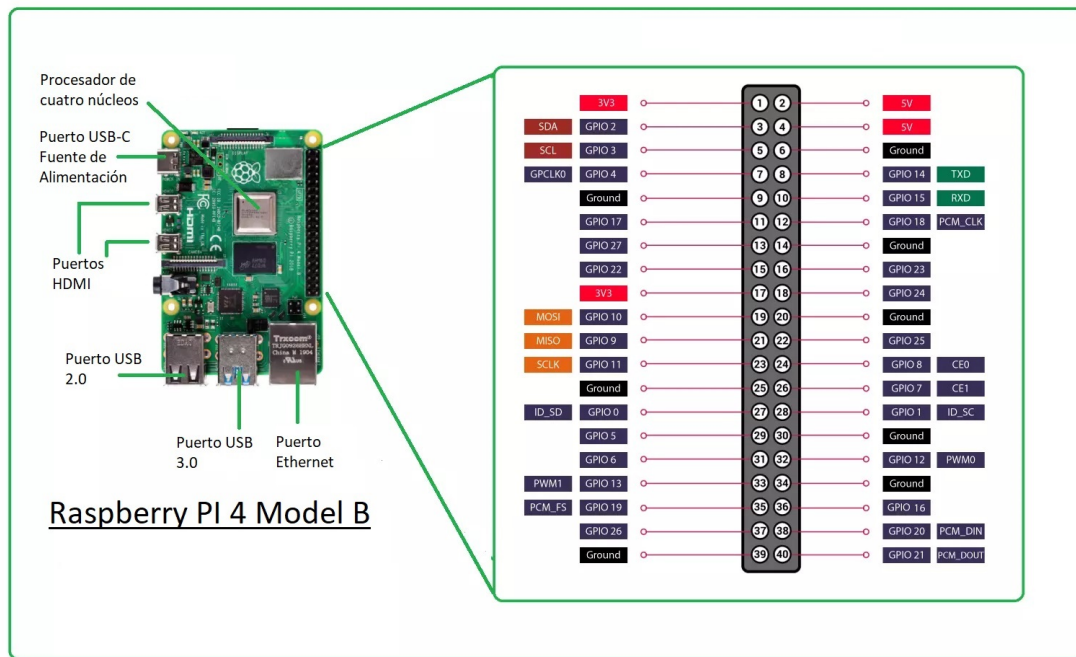


Figura 2.31: Características del dispositivo Raspberry PI 4 Model B.

Imagen disponible en:

”<https://theengineeringprojects.com>”

Por otra parte, *Raspberry UK*, que es la compañía que fabrica este dispositivo, proporciona un software llamado *Raspberry PI Imager*, con el cual se pueden cargar diferentes sistemas operativos compatibles en diferentes versiones, cada una recomendada para las capacidades de memoria RAM correspondientes a cada modelo, aunque también puede cargarse otras distribuciones de sistema operativo Linux entre las que se encuentran *Ubuntu*, *Arch* y *Debian*, e incluso versiones de solo terminales de comandos, incluida una versión de *Windows*. Sin embargo, no es la variedad de sistemas operativos que pueden cargarse a este dispositivo la razón por la cual se incluirá en nuestro proyecto del robot, es por las aplicaciones que se pueden incluir con protocolos de comunicación, como por ejemplo *MQTT*, *SSH*, *VNC*, entre algunos otros, es principalmente por su compatibilidad con *Robot Operating System*, también conocido como *ROS*.

Como se mencionó, el software de *Raspberry PI Imager*, que se puede observar en las figuras 2.32 proporciona un entorno para cargar un sistema operativo, como se observa en la figura 2.33 y figura 2.34 a una unidad de almacenamiento extratable de tipo *Micro-SD*, por lo que la cantidad de memoria puede variar dependiendo del tamaño del sistema operativo, es necesaria al menos una de 32 GB de almacenamiento para cargar el sistema operativo deseado. En algunos casos, donde se trabaje con una terminal de comandos sin interfaz de pantalla, será necesario configurar la conexión por SSH o VNC.



Figura 2.32: Software Imager disponible para Windows y Linux

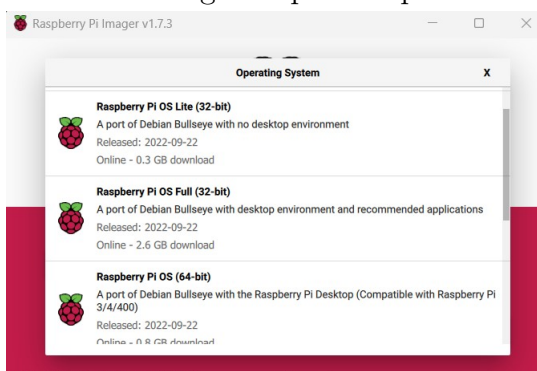


Figura 2.33: Sistemas operativos disponibles.

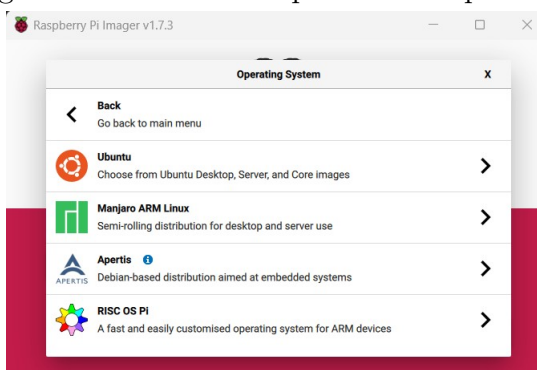


Figura 2.34: Sistemas operativos disponibles Linux.

Por sus siglas "*Robot Operating System*", o "*ROS*" en un entorno que se encarga de proporcionar una plataforma gratuita para el facilitar el desarrollo de aplicaciones en la robótica, dando a disposición múltiples librerías para facilitar la comunicación, el diseño, control y simulaciones de diferentes para realizar múltiples proyectos en la robótica, por lo que este entorno nos permitirá desarrollar la comunicación de nuestro robot.¹⁶

Por otra parte, *ROS* tiene compatibilidad con distintos sistemas operativos como *Mac OS X*, *Windows* y *Linux*, siendo este el más utilizado debido a que este entorno es de software libre. Otra característica de *ROS* es que cuenta con distintas distribuciones de diferentes librerías, facilitando a los usuarios el trabajar con una distribución diferente según las necesidades de cada usuario. Estas distribuciones se encuentran repartidas en lo que se podría considerar como dos principales ramas de *ROS*, siendo la primera conocida como *ROS 1*, la cual es la versión más antigua de *ROS*, teniendo un enfoque principalmente a simulaciones y aplicaciones de entorno de diseño, y las distribuciones que podemos encontrar aquí son por ejemplo *ROS Noetic*, *ROS Melodic*, las cuales son distribuciones que actualmente no reciben soporte por parte de la comunidad, por lo que ya no se esperan futuras actualizaciones, sin embargo, no implica que quedan obsoletas para trabajar. La otra rama es conocida como *ROS 2*, la cual es una versión más reciente de *ROS*, que contiene más librerías y contiene mayor contenido para que cualquier usuario pueda utilizar. Sus distribuciones, por mencionar, algunas son *ROS2 FOXY*, *ROS2 Humble*, *ROS 2 Galactic*. La distribución que se decidió utilizar para el desarrollo del robot es *ROS 2 Humble*, debido a que dentro de esta distribución se encuentra una librería conocida como *micro-ros*, que no se encuentra disponible en *ROS 1*, la cual facilita un entorno de comunicación con microprocesadores para el desarrollo electrónico de nuestro robot, y su logo se muestra en la figura. 2.35.



Figura 2.35: Logo de la distribución de ROS2 HUMBLE

Imagen obtenida de: Imagen disponible en:

"<https://docs.ros.org/en/humble/Installation.html>"

¹⁶"<https://wiki.ros.org/ROS/Introduction>"

Al decir que ROS proporciona diferentes librerías, nos referimos a que dentro del entorno de herramientas que proporciona al usuario y dependiendo de la distribución de *ROS* librerías en diferentes lenguajes de programación, siendo los más utilizados *C/C++*, *Python* y *Java* solo por mencionar algunos ejemplos, aunque existen más librerías para diferentes lenguajes de programación.

ROS se basa en un sistema de comunicación distribuido entre un sistema de nodos y tópicos que pueden comunicarse entre sí a través de mensajes a través de un nodo maestro. Estos mensajes pueden contener datos como comandos, datos de sensores, imágenes o alertas. Los nodos pueden publicar mensajes en ciertos temas y suscribirse a temas para recibir mensajes relevantes.

Sin embargo, no todo es sencillo al trabajar con el ambiente de *ROS*, debido a que una de las principales complicaciones que se pueden tener con este entorno se encuentra desde la instalación de cada distribución de *ROS*, siendo este problema común en usuarios nuevos debido a que cada distribución está enfocada para trabajar en un determinado sistema operativo, por lo que si no hay una compatibilidad adecuada de versiones no se podrá instalar una distribución deseada y se tendrá que optar por alguna alternativa de versión de sistema operativo. Por otra parte, la documentación que puede parecer confusa, ya que, al ser principalmente desarrollada por los usuarios, tienden a ser poco claras o específicas para una distribución de *ROS*. Y finalmente, el problema que se presenta a usuarios nuevos se encuentra en comprender el funcionamiento de *ROS* al momento de trabajar en aplicaciones específicas.

Las ventajas de ROS se pueden resumir en:

- Su uso en la industria, la investigación y entornos académicos ofrece una amplia gama de controladores, bibliotecas diseñadas para trabajar con sensores, actuadores y plantas de control.
- Al ser de código libre, permite un desarrollo y soporte en la aplicación a diferentes proyectos.
- Es multiplataforma y soportado en diferentes sistemas operativos y dispositivos compatibles.
- Al no requerir el pago de una licencia, sus aplicaciones son accesibles para todo tipo de aplicaciones en entornos académicos e industriales.

Por otra parte, las desventajas de *ROS* se pueden resumir en los siguientes puntos:

- Cada distribución de *ROS* es específica para trabajar en un determinado sistema operativo de *Linux*, *Windows* y *MAC*, por lo que ocasionalmente las distribuciones no suelen ser compatibles entre diferentes versiones, por ejemplo de *Ubuntu*, limitando a trabajar con cada distribución acorde a tu sistema operativo, por ejemplo, *ROS Humble* no es compatible con versiones anteriores a *Ubuntu 22.04*.
- Al ser de código abierto, la documentación proporcionada por la comunidad puede ser confusa para usuarios y puede ocasionar problemas al trabajar entre *ROS1* y *ROS2* debido principalmente a las diferentes distribuciones y enfoques al momento de trabajar con simulaciones y con hardware externo.
- Para usuarios nuevos que no tengan conocimientos básicos sobre el sistema operativo *Linux*, programación básica, puede ocasionar dificultades para entender el funcionamiento correcto de *ROS*, desde inicializar el nodo maestro, hasta la creación de una librería propia, hasta crear un entorno de trabajo en *ROS* debido a los múltiples pasos que se requieren para iniciar un nodo de comunicación.

Arquitectura de ROS.

Como se mencionó anteriormente en la breve introducción de *ROS*, el funcionamiento de *ROS2*, se basa en una arquitectura de comunicación distribuida entre un sistema de diferentes nodos y tópicos, donde los diferentes componentes de un sistema robótico se comunican entre sí con los siguientes elementos donde pueden interactuar diferentes controladores, sensores y actuadores por medio de mensajes, como se muestra en el diagrama de la figura 2.36.

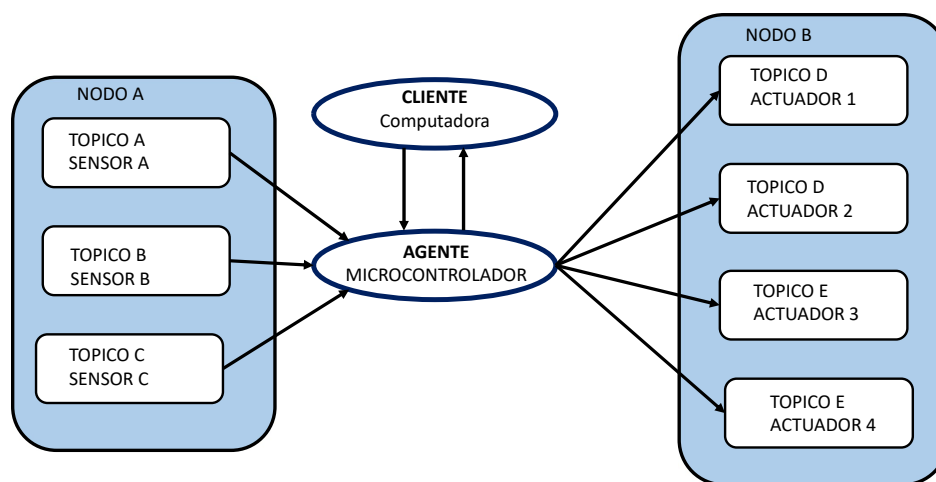


Figura 2.36: Intreracción de sensores, actuadores y controladores de ROS2.

La interacción de estos elementos constituyen el sistema de un robot con el que puede interactuar por medio de diferentes elementos, como se observa en la siguiente figura 2.37, donde se observa un ejemplo de como pueden interactuar por medio de *ROS2* diferentes elementos como pueden ser sensores, salidas de imagen en una pantalla u otros elementos como cámaras por un entorno de comunicación.

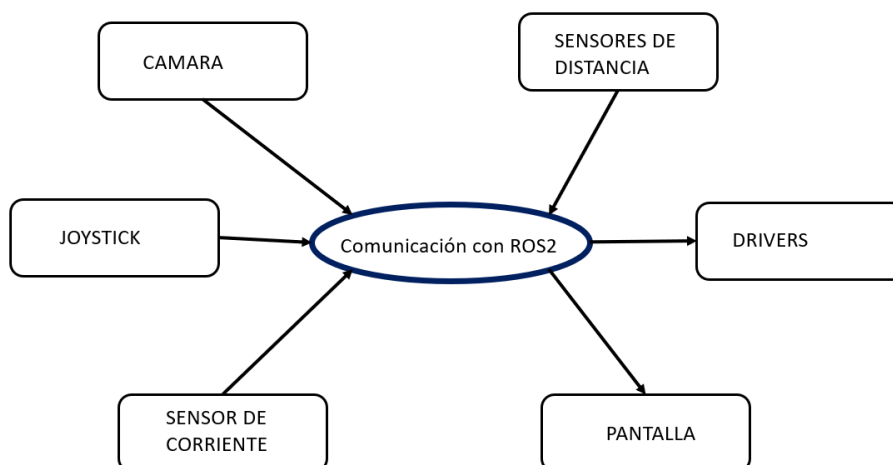


Figura 2.37: Intreracción de elementos de *ROS 2*.

Los conceptos básicos de ROS se explican a continuación.

- **Nodos:** Los nodos son los componentes básicos de *ROS2*. Cada nodo es una unidad de ejecución independiente que puede realizar tareas específicas, como el control de un sensor o un actuador, el procesamiento de datos o la planificación de movimientos. Los nodos se comunican entre sí a través de mensajes, servicios y acciones. Son archivos ejecutables de ROS, además de que existen múltiples paquetes predefinidos de ROS, así como la posibilidad de crear nuestros propios nodos.
- **Mensajes:** Los mensajes son estructuras de datos utilizadas para intercambiar información entre los nodos. *ROS2* proporciona un sistema de definición de mensajes que permite especificar la estructura y los tipos de datos de los mensajes utilizados en un sistema robótico. Los mensajes se envían y reciben a través de los llamados "tópicos", estos valores pueden representar las lecturas de algún sensor, o algún tipo de variable de nuestro interés. La interacción más básica de *ROS2* depende de los elementos anteriores, adicionando la interacción de un publicador y un subscriptor, como se observa en la figura 2.38, donde se puede ejemplificar de mejor manera como un nodo publicador que podría ser la información de un sensor envía un valor en forma de mensaje, por medio de un tópico y el nodo subscriptor interpreta ese valor del mensaje y tomar una decisión.

- Tópicos: Los tópicos son canales de comunicación unidireccionales a través de los cuales los nodos pueden enviar y recibir mensajes. Un nodo puede publicar mensajes en un tópico específico y otros nodos pueden suscribirse a ese tópico para recibir los mensajes. Esto permite una comunicación asíncrona y flexible entre los componentes del sistema, de igual manera, esta interacción se observa en la figura 2.38 .

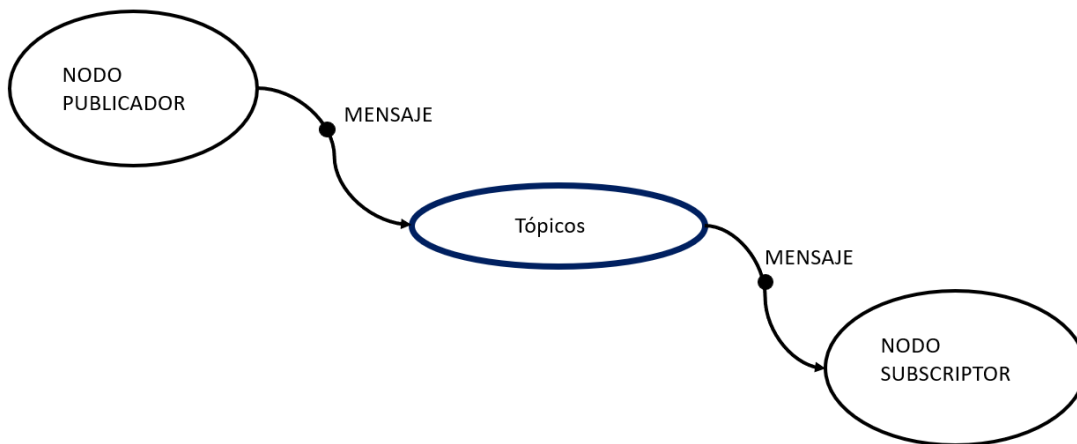


Figura 2.38: Interacción de Nodos y Tópicos de *ROS*.

- Servicios: Los servicios en ROS, no son más que una manera de comunicación del tipo "solicitud y respuesta". La estructura de datos de un servicio tiene dos campos, uno para la petición y uno para la respuesta. Un nodo puede ofrecer un servicio y otros nodos pueden enviar solicitudes a ese servicio para recibir una respuesta. Esta comunicación es síncrona y se basa en la invocación de un servicio específico. La interacción de un servicio se puede observar un ejemplo en la figura 2.39.

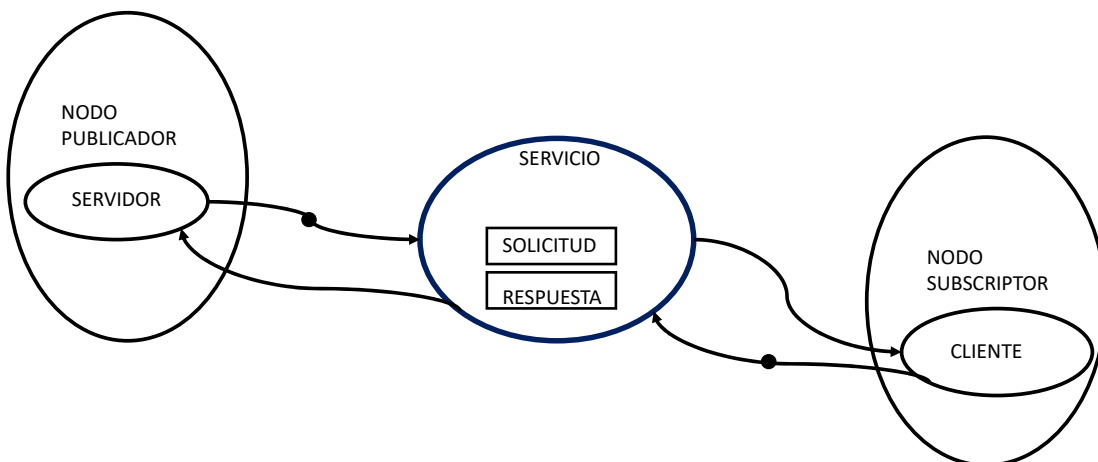


Figura 2.39: Interacción de servicios de *ROS*.

- Bags: Los Bags son ficheros de datos que salvan los mensajes que se publican durante una ejecución, estos resultan útiles para grabar información de una sesión de experimentos y crear un respaldo de datos que luego se pueden utilizar para aprendizaje automático o para estudio de un patrón de comportamientos, como puede ser reconocimiento de un entorno o aprendizaje de patrones.
- Nodo Maestro de ROS: El nodo maestro es el nodo central, es decir, se debe ejecutar este nodo maestro para el funcionamiento de *ROS*, simplemente con el comando: *"roscore"*. Se debe configurar el archivo *".bashrc"*, de cada ordenador a la red, para que este nodo principal se ejecute en un ordenador y otros ordenadores o dispositivos se conecte a él. Es importante mencionar que un sistema solo puede tener un nodo maestro a la vez, la interacción de este nodo maestro y tópicos se puede ejemplificar en la figura 2.40.

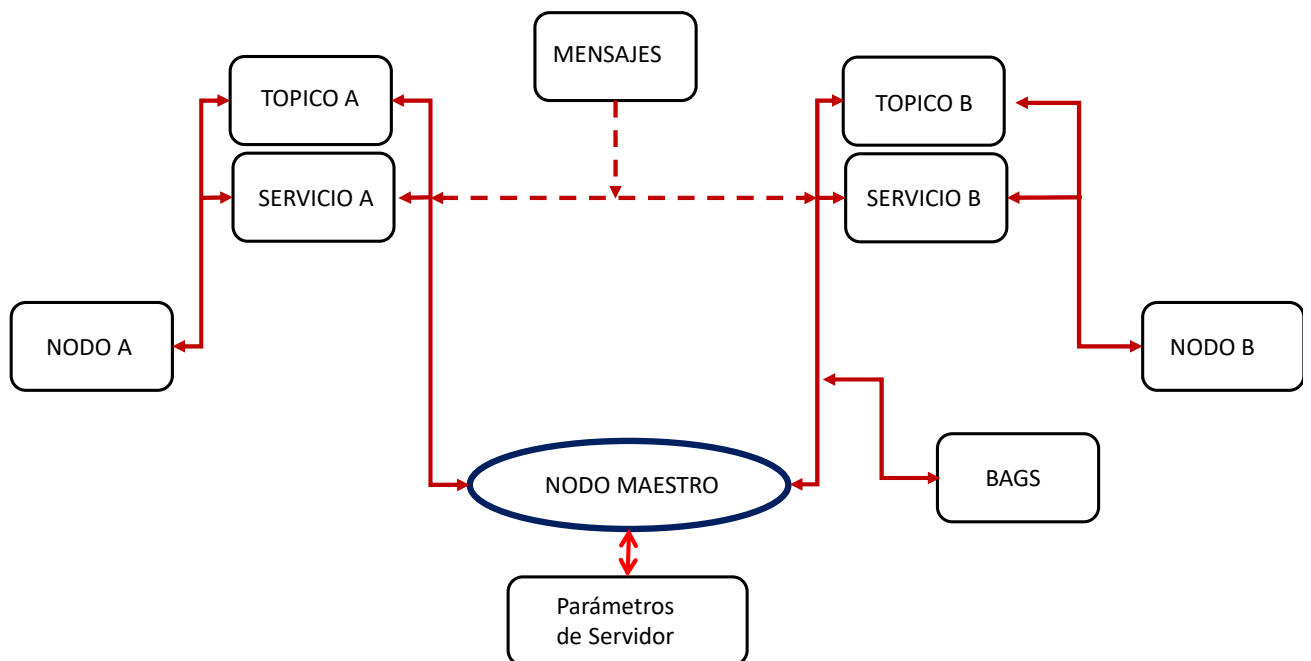


Figura 2.40: Interacción del nodo maestro en *ROS*

- Launch files: Los Launch files son ficheros de tipo *"XML"* que sirven para lanzar en orden una serie de nodos para la comunicación de nuestro robot. Se caracterizan por tener una extensión *".launch"*.

- **Acciones:** las acciones permiten la comunicación de tipo solicitud y respuesta extendida en el tiempo. A diferencia de los servicios, las acciones pueden tener múltiples respuestas o un flujo continuo de datos. Las acciones son útiles para tareas que requieren una interacción más compleja y un control fino, como la planificación de movimientos o la ejecución de tareas secuenciales, se puede observar un ejemplo de manera gráfica en la figura 2.41, entre la interacción de un cliente y un servidor, por medio de un tópico de acciones y solicitudes y respuestas de interacción y corrección mediante un tópico de retroalimentación.

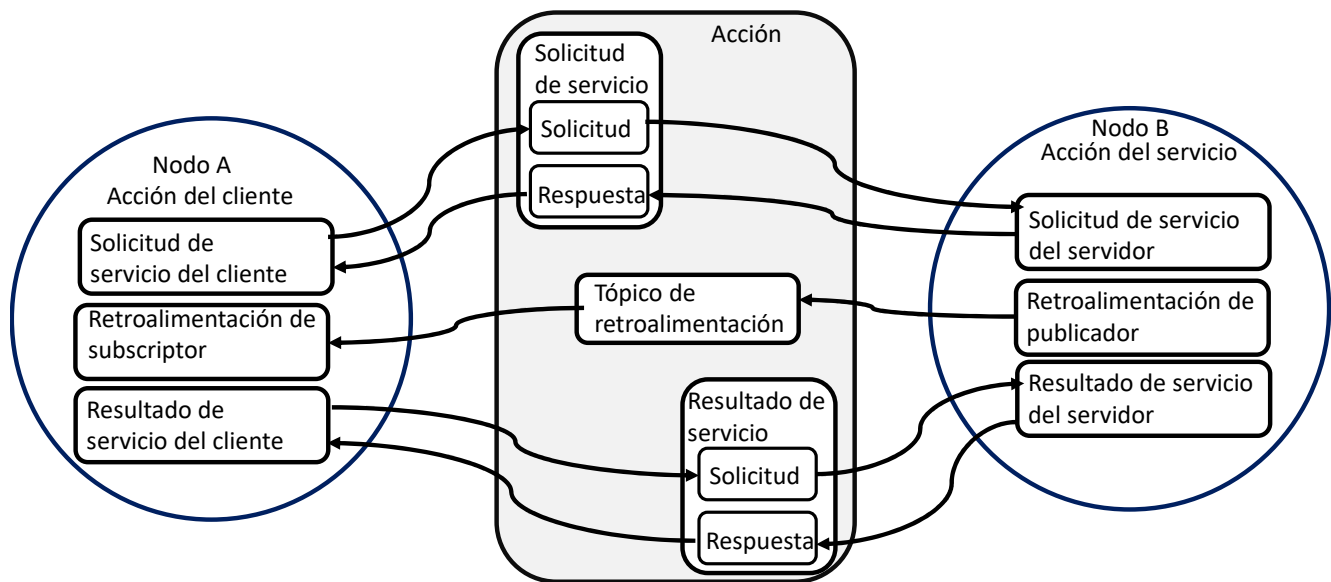


Figura 2.41: Interacción las acciones en *ROS*

Adicional a esto, *ROS* provee herramientas que resultan útiles, por ejemplo, para la simulación y visualización se disponen las herramientas:

- RVIZ: Es un entorno de visualización donde se puede observar la información de diferentes sensores para visualizar la información de sus respectivos nodos y tópicos en un entorno de visualización en 2D y 3D, así como modelar y simular entornos de robots para múltiples aplicaciones de *ROS* interactuando con la información de interés. Un ejemplo de esta visualización se puede observar en la figura 2.42.

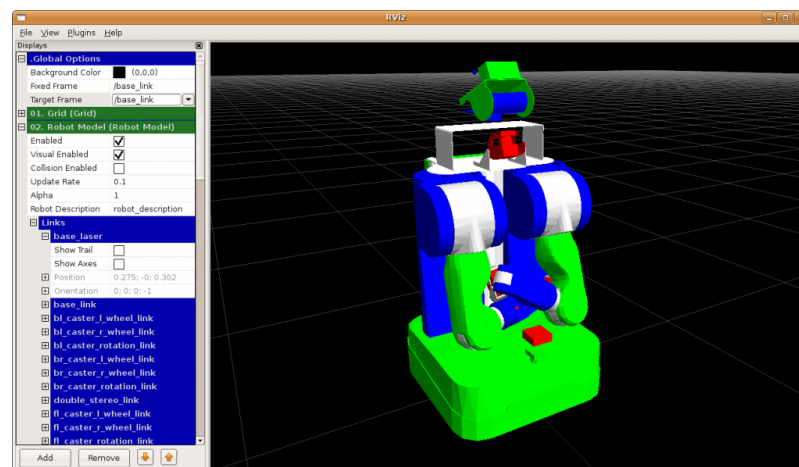


Figura 2.42: Ejemplo de una interacción de nodos de un robot por medio de RVIZ . Imagen disponible en:

”<https://automaticaddison.com/how-to-launch-rviz-and-rqt-in-ros/>”

- Gazebo: Es un entorno de simulación donde se pueden crear y diseñar prototipos, así mismo para realizar diversas pruebas de programación, un ejemplo de esta visualización se observa en la figura 2.43.

Micro-ROS

Micro-ROS es una herramienta propia integrada en la versión de *ROS 2*, siendo una extensión diseñada específicamente para trabajar con sistemas embebidos y microcontroladores, con el objetivo de permitir la integración de las herramientas de comunicación en nodos de *ROS* en robots y dispositivos con recursos limitados para no requerir un sistema de cómputo de mayor tamaño, costo y consumo energético en pequeños sistemas para lograr integrar una comunicación entre dos o más dispositivos por medio de un nodo de comunicación entre un agente y un cliente.

Micro-ROS es compatible con diferentes arquitecturas de microcontroladores comerciales, lo que facilita la implementación de aplicaciones electrónicas en un amplio catálogo con documentación disponible para múltiples aplicaciones de robótica, aunque en algunos casos para algunos modelos de microprocesadores específicos se ve carente de documentación o con pocos ejemplos o directamente no es compatible y puede tener problemas al momento de buscar implementar el entorno de *micro-ROS* en un microprocesador que tenga características muy limitadas en propiedades de memoria, como por ejemplo el ATMEGA328P, que comercialmente corresponde al microprocesador de un dispositivo "*Arduino UNO*", al ser un microcontrolador de 8 bits, es decir que procesa 8 bits por ciclo de instrucción, será útil para aplicaciones sencillas, que no requieran de múltiple componente electrónicos, en este caso no podrá soportar la configuración necesaria para el entorno de *micro ROS* debido a la cantidad de memoria y el procesamiento que solicita para su funcionamiento, por lo que no resulta viable para nuestra propuesta de diseño de robot. Solo por mencionar algunos dispositivos compatibles tenemos la siguiente lista¹⁷.

- Renesas EK RA6M5 y e2studio
- Espressif ESP32
- Arduino Portenta H7
- Raspberry Pi Pico RP2040
- ROBOTIS OpenCR 1.0
- Teensy 3.2
- STM32L4 Discovery kit IoT
- Olimex LTD STM32-E407
- Arduino Due
- Arduino ZERO
- ST NUCLEO-F446ZE

¹⁷<https://micro.ros.org/docs/overview/hardware/>

Si bien *micro-ROS* no es una distribución de *ROS*, lo que hace esta extensión es que provee bibliotecas para diferentes dispositivos por medio de una conexión de comunicación serial USB, por medio del protocolo de comunicación *UART*, *Ethernet* y *Bluetooth* estableciendo las herramientas directas para la creación de nodos y tópicos en el dispositivo con la opción a modificar, crear y editar los nodos, tópicos y mensajes para la comunicación entre el agente y el cliente, por el cual se puede tener la comunicación e interactuar con diferentes componentes electrónicos conectados, como por ejemplo recibir la información de la lectura de algún sensor, y comunicarlo por medio de un mensaje entre diferentes dispositivos que tengan la comunicación para interactuar entre ellos.

Ahora recapitulando la decisión de elegir el dispositivo *Raspberry PI 4*, *Micro-ROS*, combinado con las características de la *Raspberry Pi Pico* y el procesador RP2040 que es microcontrolador de 16 bits, como microcontrolador, son compatibles con *micro-ROS*, al menos en el entorno de programación de *C/C++*, que resulta más ligero y optimizado que trabajar con el entorno de desarrollo en *Micro-Python*, principalmente porque este carga con un intérprete para poder facilitar el entorno de programación, el cual no es compatible con el entorno de *micro-ROS*, debido principalmente al espacio de memoria del microprocesador, debido a que este carga un archivo de extensión *.uf2* que funciona como un intérprete que facilita el entorno para cargar proyectos desde *Micro-Python*, haciendo que este entorno de trabajo sea ideal para aplicaciones de aprendizaje o aplicaciones que no requieran gran complejidad en su código de programación, sin embargo, este lenguaje ocupa un mayor espacio de memoria del microcontrolador y no es tan eficiente que hacerlo en un lenguaje de programación compilado, por lo que se tomó la decisión que para el desarrollo del robot se debe tener en cuenta que para trabajar con *ROS 2* y *Micro-ROS* se debe desarrollar nuestro código de programación en lenguaje de programación de *C/C++* con el dispositivo *Raspberry Pi Pico*, ya que al compilar nuestro código generamos el archivo con la extensión *.UF2*, que será más optimizada, compatible en memoria con los requisitos de *micro-ros* y podremos aprovechar una amplia cantidad de pines para colocar nuestros componentes electrónicos para la implementación de nuestro robot. La arquitectura de *Micro-ROS* se puede observar en la figura 2.44.

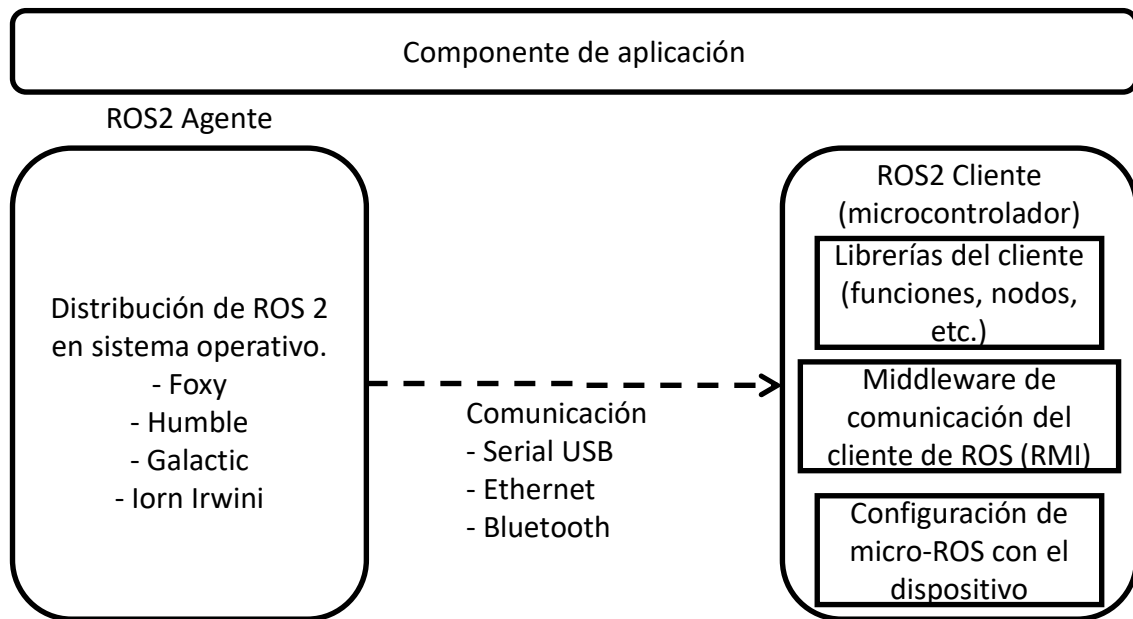


Figura 2.44: Arquitectura de comunicación de *Micro-ROS*

El funcionamiento de *Micro-ROS* está caracterizado principalmente de dos elementos principales, denominados como el cliente y agente.

El agente es la computadora que se encarga de los procesamiento principales y la comunicación con el nodo maestro, así como los otros nodos, así como encargarse de la interfaz de usuario del robot.

Por otra parte, el cliente se trata del microcontrolador, compatible con una amplia gama de microcontroladores por comunicación serial, el cual puede recopilar datos de sensores o enviar información a actuadores para el control de nuestro robot. Esto queda mejor ejemplificado en la figura 2.45, donde se observa mejor la interacción del agente y cliente y cómo funcionará para nuestro robot con los dispositivos seleccionados, es decir, el dispositivo *Raspberry PI 4* y la *Raspberry Pi Pico*.

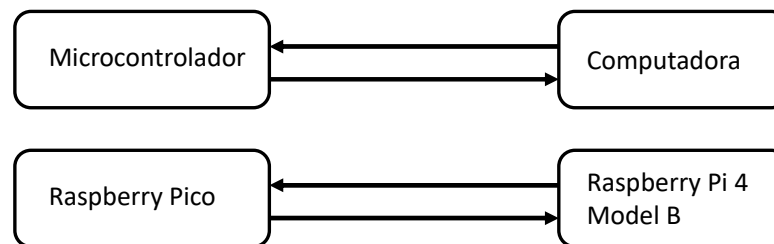


Figura 2.45: Comunicación entre cliente y agente.

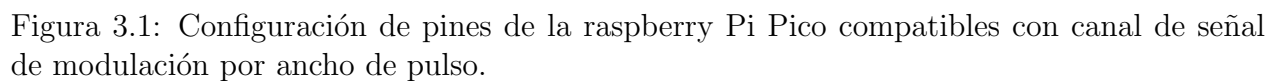
METODOLOGÍA

Para el desarrollo de estas tesis es necesario abordar diferentes temas y procedimientos que van a converger en un sistema para la propuesta general de nuestro robot móvil omnidireccional, debido a que son diferentes factores a tomar en cuenta con base en las normativas hospitalarias abordadas al principio del marco teórico.

Anteriormente, observamos la propuesta de diseño del chasis que será la base del robot omnidireccional asistente a hospitales ya tenía una estructura definida en función de cumplir en mayor medida con las normativas hospitalarias, adicionalmente consideramos un modelo de motores de corriente directa necesarios para el tren motriz de nuestro robot acoplados a ruedas omnidireccionales de tipo *mecanum* y se propusieron dos modelos diferentes de los controladores de motores de corriente directa que fueran tolerantes a la corriente necesaria, así como también se propusieron algunos sensores y componentes electrónicos para implementar en nuestro robot que tendrán una determinada función, en algunos casos incluyendo algunos programas con ejemplos de funcionamiento realizados que nos ayudarán a converger en el diagrama de flujo general para la comunicación por medio de *micro - ROS* que nos ayudarán en el diseño de nuestra propuesta del robot, por lo que para nuestra metodología nos centraremos en hacer una convergencia para la propuesta general del robot, desde el funcionamiento de una señal de modulación por ancho de pulso (PWM) para controlar los motores de corriente directa, el modelo matemático de nuestro robot omnidireccional, simulaciones de nuestro sistema de control, un diagrama de flujo de nuestro programa general, resolver el problema de alimentación de energía y las protecciones necesarias, así como el diseño electrónico en una placa de circuito impreso o abreviada como "PCB", y lograr establecer la comunicación en el entorno de *ROS* y abordar los resultados obtenidos, incluyendo algunos diseños realizados o disponibles en impresión 3D para nuestro robot y facilitar el diseño de nuestra propuesta.

Modulación por ancho de pulso para Raspberry PI Pico.

La configuración de los segmentos de configuración PWM distribuidos en los 30 pines, se observa en la figura 3.2, donde se observa que dentro de estos segmentos del 0 al 7 se repiten entre algunos pines digitales, por lo que la configuración que se desea dar dependerá mucho de la configuración realizada en el programa principal.



"<https://randomnerdtutorials.com/raspberry-pi-pico-pwm-micropython/>"

Como se observa en la figura 3.2, en esta distribución de pines se encuentran caracterizadas con las letras "A" y "B", donde la característica es que los pines "B" pueden ser configurados como una entrada para la medición de frecuencia y ciclo de trabajo; sin embargo, esta opción no la tomaremos en cuenta para el desarrollo de este proyecto. En el caso de este proyecto utilizaremos únicamente 6 salidas consideradas, correspondientes a nuestros controladores de motores de corriente directa, utilizando un rango de ciclo de trabajo de 0[%] a 100[%], como se observa en la figura ??.

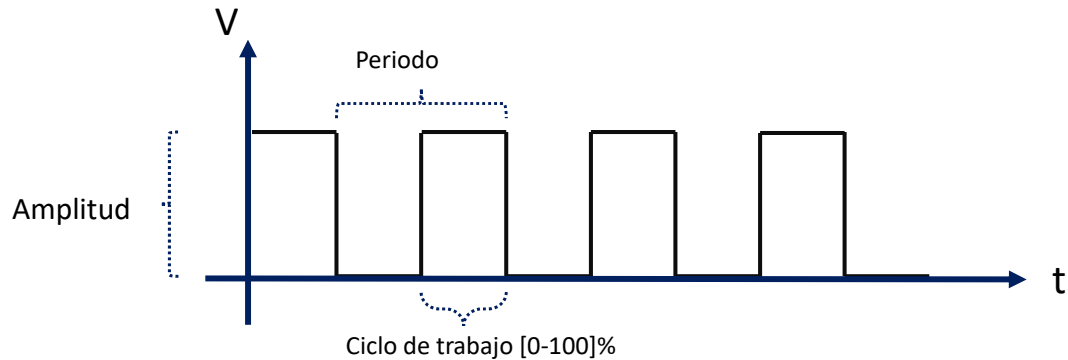


Figura 3.2: Señal de modulación por ancho de pulso en función del tiempo.

Este rango de ciclo de trabajo se encuentra entre 0 y 65535, correspondiente a sus correspondiente a 16 bits, y nos dará un voltaje de salida entre 0 [V] y 3.3 [V],. La librería para habilitar el ciclo del trabajo es:

```
1 #include "hardware/pwm.h"
```

La habilitación del canal PWM se realiza con los siguientes comandos:

```
1 gpio_set_function(ejemplo, GPIO_FUNC_PWM);
2 uint slice_num_A = pwm_gpio_to_slice_num(ejemplo);
3 pwm_set_wrap(slice_num_A, 65465);
4 pwm_set_chan_level(slice_num_A, PWM_CHAN_A, 7);
5 pwm_set_enabled(slice_num_A, true);
6 pwm_set_chan_level(slice_num_A, PWM_CHAN_A, 7);
```

La modificación del archivo Cmake.txt correspondiente para la habilitación del canal PWM es:

```
1 target_link_libraries(Ejemplo pico_stdlib hardware_adc hardware_pwm);
```

De manera adicional se incluye en el *apéndice C* un breve ejemplo para el uso de la configuración de los pines PWM, con un ejemplo sencillo de control de intensidad de un diodo led.

Sección 3.2.

Análisis dinámico de los motores.

A continuación se desarrollará de manera matemática la caracterización de los motores M1, M2, M3 y M4, correspondientes a cada uno de nuestros motores de corriente directa de nuestro robot. Ya que los motores son similares partiremos de un análisis individual.

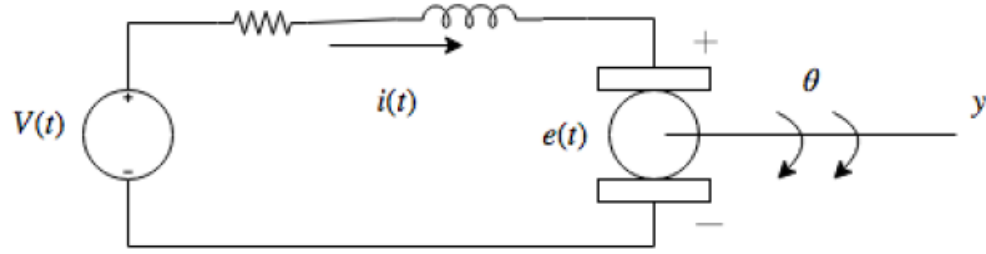


Figura 3.3: Esquema del circuito eléctrico del motor de corriente directa.

Realizando el análisis de malla del circuito anterior, entonces tenemos la siguiente expresión:

$$V(t) = R * i(t) + L \frac{di(t)}{dt} + e(t) \quad (3.1)$$

De la expresión anterior tenemos:

$V(t)$: voltaje de entrada en función del tiempo

R : Resistencia interna del motor

$i(t)$: Corriente en función del tiempo

L : valor de la inductancia

$e(t)$: Voltaje en función del tiempo

Sin embargo, para poder continuar con el modelado de esta ecuación diferencial, aplicaremos transformada de Laplace para reestructurar la expresión anterior en función de la frecuencia

$$V(S) = (R + LS)I(S) + E(S) \quad (3.2)$$

Ahora bien debemos considerar el torque generado en la rotación del motor de corriente directa, considerandolo a partir de la siguiente expresión:

$$\tau(S) = K_m I(S) \quad (3.3)$$

De la expresión anterior tenemos:

k_m Es la contante de la fem del motor.

$\tau(S)$: Es el torque del motor de corriente directa:

En consecuencia el torque del motor hace que se tenga una velocidad angular θ , descrito por la siguiente expresión:

$$\tau(S) = J\dot{\theta}(t) + f\theta(t) \quad (3.4)$$

De la expresión anterior tenemos:

J: Inercia

f: Fricción

θ : Velocidad angular [rad]

La expresión anterior reescrita aplicando la transformada de laplace tenemos la siguiente expresión:

$$\tau(S) = JS\theta(S) + f\theta(S) \quad (3.5)$$

Reescribiendo las ecuaciones 5.5 y 5.3 tenemos:

$$\tau(S) = K_m I(S) = JS\theta(S) + f\theta(S) \quad (3.6)$$

Despejando la expresión anterior en función de la velocidad angular tenemos:

$$\tau(S) = \frac{\tau(S)}{JS + f} = \frac{k_m I(S)}{JS + f} \quad (3.7)$$

Reescribiendo la ecuación 5.2 en función de I(S) tenemos:

$$I(S) = \frac{V(S) - e(S)}{R + LS} \quad (3.8)$$

Sustituyendo la expresión 5.8 en la 5.7 obtenemos:

$$\theta(S) = \frac{k_m \left(\frac{V(S) - e(S)}{R + LS} \right)}{JS + f} \quad (3.9)$$

El voltaje en el motor del torque se puede expresar como:

$$e(S) = k_m \theta(S) \quad (3.10)$$

Sustituyendo la expresión 5.11 en la expresión 5.9 obtenemos:

$$\theta(S) = \frac{\left(\frac{k_m V(S) - k_m^2 \theta(S)}{R + LS} \right)}{JS + f} \quad (3.11)$$

Simplificando la expresión obtenida entonces tenemos:

$$\theta(S) = \frac{k_m V(S) - k_m^2 \theta(S)}{(JS + f)(R + LS)} \quad (3.12)$$

Ahora es necesario buscar una expresión para $\theta(S)$, por lo que debemos despejar $k_m V(S)$, entonces:

$$k_m V(S) = \theta(S)(JS + f)(R + LS) + k_m^2 \theta(S) \quad (3.13)$$

Reescribiendo nuevamente $\theta(S)$:

$$\theta(S) = \frac{k_m V(S)}{(JS + f)(R + LS) + k_m^2} \quad (3.14)$$

De la expresión anterior podemos afirmar y establecer una función de transferencia definiendo a $V(S)$ como entrada y a $\theta(S)$ como la salida:

$$\frac{\theta(S)}{V(S)} = \frac{k_m}{(JS + f)(R + LS) + k_m^2} \left[\frac{\text{rad} * S}{v} \right] \quad (3.15)$$

Así entonces queda definida la función de transferencia de un motor de corriente directa, cabe mencionar que la rueda mecanum implica un valor de fricción mayor a las ruedas convencionales, por lo que al momento de definir los parámetros para el control dinámico del robot consideramos los 4 motores y el peso del robot.

3.2.1 Modelado del sistema.

El robot cuenta con cuatro motores de corriente directa, por lo que al describir un sistema dinámico debemos tomar en cuenta como está constituido internamente un motor de corriente directa, como se observa en la siguiente figura:

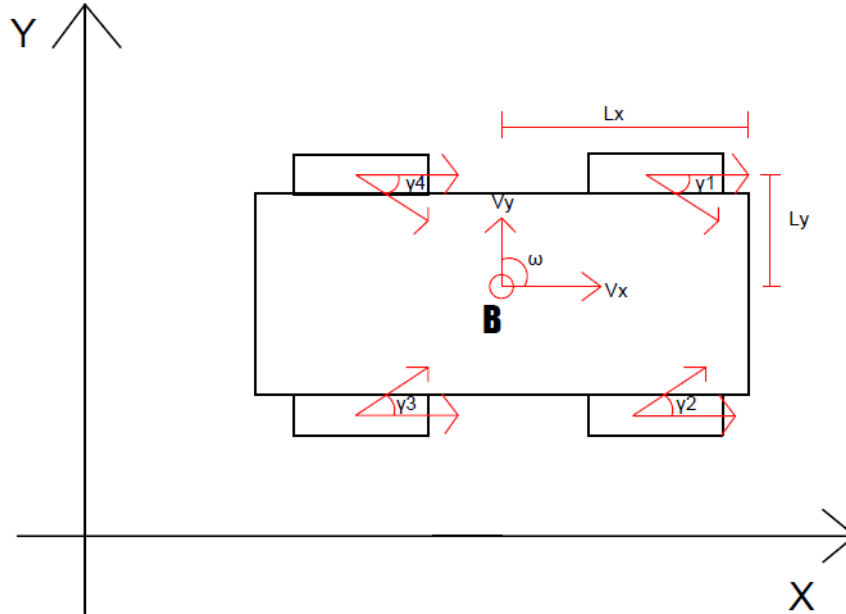


Figura 3.4: Caracterización espacial para el modelo del robot omnidireccional.

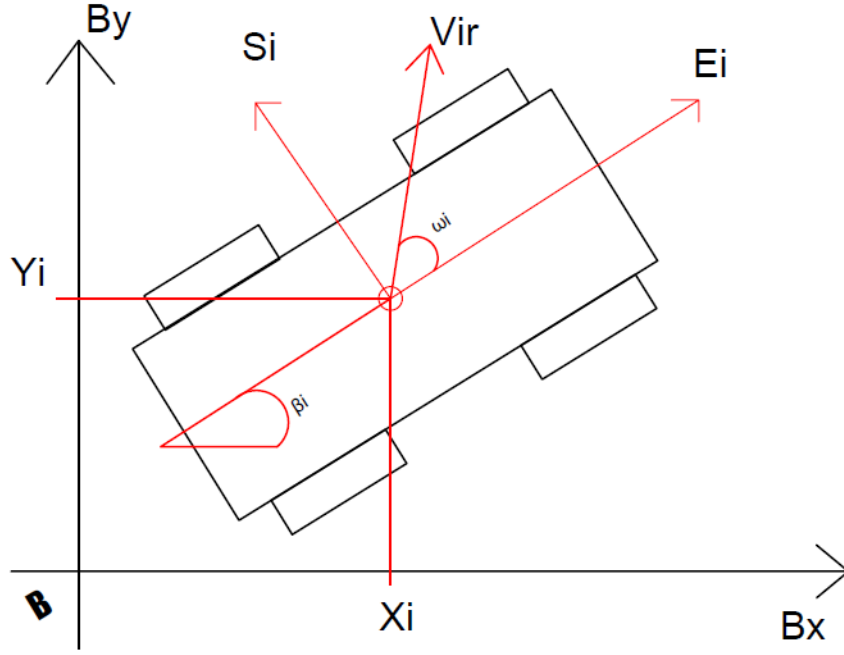


Figura 3.5: Dirección vectorial de una rueda mecanum.

Como se observa en la figura anterior, la relación dinámica de la rueda mecanum resulta en la siguiente expresión:

$$V_{ir} = \frac{r_r \omega_i}{\cos(\gamma_i)}; \quad (3.16)$$

Donde: $r_r > 0$ es el radio.

i : es el número de rueda

ω_i : es la velocidad angular de la rueda.

Entonces las expresiones respectivas en V_{Ei} y V_{Si} son:

$$V_{Ei} = r \omega_i + V_{ir} \cos(\gamma_i) \quad (3.17)$$

$$V_{Si} = V_{ir} \sin(\gamma_i) \quad (3.18)$$

Donde V_{Si} es la componente de V_{ir} . Podemos describir lo anterior en la siguiente matriz.

$$\begin{pmatrix} V_{Ei} \\ V_{Si} \end{pmatrix} = \begin{pmatrix} r & \cos(\gamma_i) \\ 0 & \sin(\gamma_i) \end{pmatrix} \begin{pmatrix} \omega_i \\ V_{ir} \end{pmatrix} = T_1 \gamma_i \begin{pmatrix} \omega_i \\ V_{ir} \end{pmatrix} \quad (3.19)$$

De la expresión anterior debemos resaltar que la expresión $T_1 \gamma_i$ es no singular siendo que el determinante de T_1 es diferente de cero.

Ahora las velocidades lineales de V_{Ei} y V_{Si} con respecto al punto B, se expresan de la siguiente manera:

$$\begin{pmatrix} V_{iXB} \\ V_{iYB} \end{pmatrix} = \begin{pmatrix} \cos(\beta_i) & -\sin(\beta_i) \\ \sin(\beta_i) & \cos(\beta_i) \end{pmatrix} \begin{pmatrix} V_{Ei} \\ V_{Si} \end{pmatrix} \quad (3.20)$$

Por ejemplo, si el valor de ángulo de β_i es cero para cada uno de los casos, entonces podemos afirmar que el movimiento es paralelo al eje x, considerando además que la alimentación eléctrica de cada uno de los motores para V_{Ei} , entonces podemos afirmar que la expresión resultante es:

$$\begin{pmatrix} V_iXB \\ V_iYB \end{pmatrix} = \begin{pmatrix} V_{Ei} \\ V_{Si} \end{pmatrix} \quad (3.21)$$

Considerando un movimiento lineal en el plano XY se obtienen las velocidades lineales de la rueda i

$$\begin{pmatrix} V_iXB \\ V_iYB \\ 0 \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ w \end{pmatrix} * \begin{pmatrix} x_i \\ y_i \\ 0 \end{pmatrix} \quad (3.22)$$

Obteniendo la siguiente expresión:

$$\begin{pmatrix} V_iXB \\ V_iYB \end{pmatrix} = \begin{pmatrix} 1 & 0 & -y_i \\ 0 & 1 & x_i \end{pmatrix} U \quad (3.23)$$

Donde $U = [v_x v_y \omega]^T$, siendo el control de entrada del vector del control de velocidad, y los puntos x_i, y_i son las posiciones de las coordenadas de puntos del centro de la rueda con respecto al punto B.

Tomando en cuenta las expresiones 5.23 y 5.21, podemos obtener la siguiente expresión:

$$\begin{pmatrix} \omega_i \\ V_{ir} \end{pmatrix} = \begin{pmatrix} r & \cos(\gamma_i) \\ 0 & \sin(\gamma_i) \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 & -y_i \\ 0 & 1 & x_i \end{pmatrix} U \quad (3.24)$$

La expresión anterior relaciona las velocidades del robot con las velocidades angulares de cada rueda.

Al mismo tiempo, estas velocidades angulares en ω están dadas por la expresión:

$$\omega_i = \frac{1}{r} \begin{pmatrix} -1 & \frac{-1}{\tan(\gamma_i)} & \frac{-1}{\tan(\gamma_i)} x_i - y_i \end{pmatrix} U \quad (3.25)$$

De la consideración anterior, donde el movimiento del robot es paralelo al eje X, entonces obtendremos las siguientes relaciones del ángulo gamma:

$$\gamma_1 = \gamma_3 = \frac{\pi}{4}$$

$$\gamma_2 = \gamma_4 = -\frac{\pi}{4}$$

$$x_1 = L_x, y_1 = L_y$$

$$x_2 = L_x, y_2 = -L_y$$

$$x_3 = -L_x, y_3 = -L_y$$

$$x_4 = -L_x, y_4 = L_y$$

Finalmente, la relación entre el vector de la entrada de velocidad del robot y la velocidad angular para cada rueda queda expresado en la siguiente expresión

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \frac{1}{r} \begin{pmatrix} 1 & -1 & -L_x - L_y \\ 1 & 1 & L_x + L_y \\ 1 & -1 & L_x + L_y \\ 1 & 1 & -L_x - L_y \end{pmatrix} U \quad (3.26)$$

Sustituyendo $U = (v_x \ 0 \ 0)^T$ en la ecuación 5.26 obtenemos los siguientes resultados:

$$r\omega_1 = v_x \quad r\omega_2 = v_x$$

$$r\omega_3 = v_x$$

$$r\omega_4 = v_x$$

y Sumando las expresiones anteriores obtenemos:

$$v_x = \frac{r(\omega_1 + \omega_2 + \omega_3 + \omega_4)}{4} \quad (3.27)$$

Siendo este el resultado de la velocidad en el eje x.

Por otra parte, sustituyendolo con el valor de U en el eje Y, siendo $U = (0 \ v_y \ 0)^T$ en la ecuación 5.26 obtenemos los siguientes resultados:

$$r\omega_1 = -v_y \quad r\omega_2 = v_y$$

$$r\omega_3 = -v_y$$

$$r\omega_4 = v_y$$

Y sumando estas componentes obtenemos entonces:

$$v_y = \frac{r(-\omega_1 + \omega_2 - \omega_3 + \omega_4)}{4} \quad (3.28)$$

Finalmente y de manera similar siendo U la velocidad angular y sustituyendo $U = (0 \ 0 \ \omega)^T$ en la ecuación 5.26 obtenemos los siguientes resultados:

$$\omega = \frac{r(-\omega_1 + \omega_2 - \omega_3 + \omega_4)}{4(L_x + L_y)} \quad (3.29)$$

Entonces, de las expresiones 5.27, 5.28 y 5.29, podemos obtener la relación de las velocidades angulares de las ruedas para el control de la velocidad de entrada del robot descrito por la expresión:

$$\begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix} = \frac{r}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ \frac{1}{L_x + L_y} & \frac{1}{L_x + L_y} & \frac{1}{L_x + L_y} & \frac{1}{L_x + L_y} \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} \quad (3.30)$$

De la expresión anterior finalmente se concluyen las siguientes relaciones:

La velocidad longitudinal es:

$$v_x(t) = (\omega_1 + \omega_2 + \omega_3 + \omega_4) * \frac{r}{4} \quad (3.31)$$

La velocidad transversal es:

$$v_y(t) = (-\omega_1 + \omega_2 + \omega_3 + \omega_4) * \frac{r}{4} \quad (3.32)$$

La velocidad angular es:

$$\omega(t) = (-\omega_1 + \omega_2 * \omega_3 + \omega_4) * \frac{r}{4(L_x + L_y)} \quad (3.33)$$

Finalmente, también el modelo cinemático del robot móvil con ruedas mecanum está dado por la expresión:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} U = R(\phi)U \quad (3.34)$$

Donde $R(\phi)$ es la matriz rotacional sobre el eje Z_G .

Sección 3.3.

Diseño de control del robot móvil omnidireccional.

El diagrama de bloques para el sistema de control en lazo cerrado partiendo de la ecuación de variables de estado.

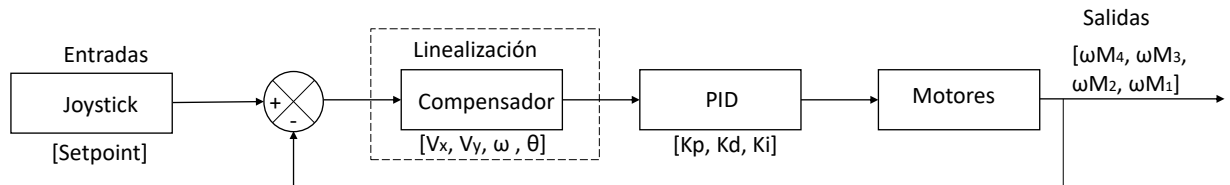


Figura 3.6: Diagrama de bloques del diseño de control de robot omnidireccional.

El diagrama anterior expresado en un sistema de simulación realizado en Matlab Simulink es el que se muestra en la siguiente figura.

De la primera sección de esta simulación tenemos se observa que debe tomar como referencia la velocidad que se tiene en los sentidos de los ejes en X, Y y omega, por lo que se tienen como referencia un valor de escalón para el instante $t = 0$, donde pasa a tener un estado de movimiento, como se observa en la siguiente figura.

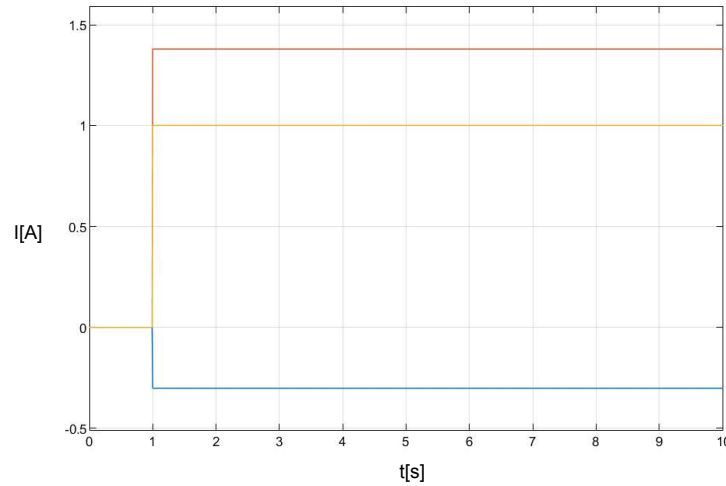


Figura 3.8: Entradas de referencia de corriente.

Antes de entrar en el proceso del PID debemos verificar las graficas arrojadas del error en los cuatro motores representados por las variables " $e[n]$ ", como se observa en la siguiente gráfica:

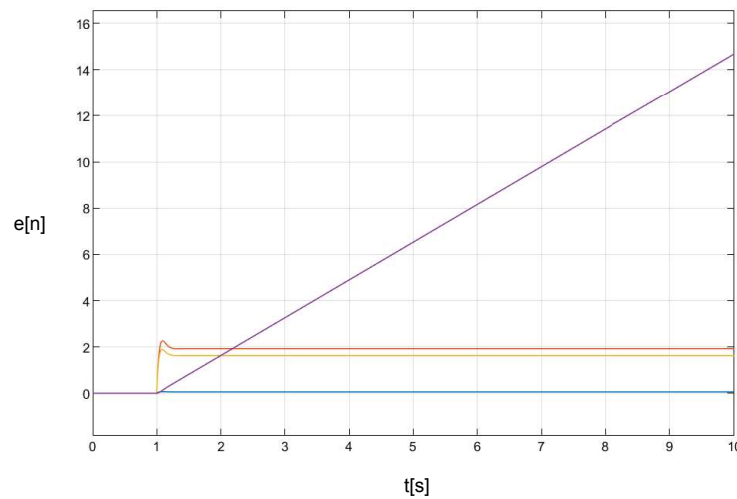


Figura 3.9: Gráfica de los errores: e_1 , e_2 , e_3 , e_4 .

La salida de los 4 motores de la velocidad después de pasar por la linealización de la matriz y el control PID después de pasar por la cinemática del robot en la gráfica resultante es:

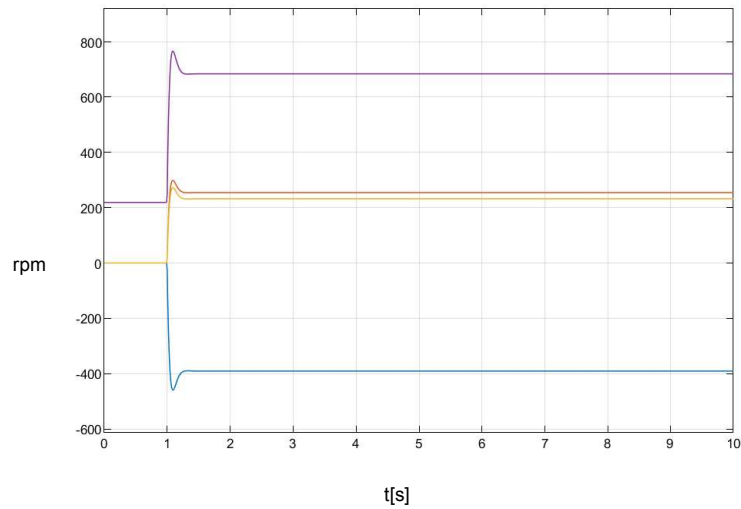


Figura 3.10: Gráfica de la salida Deseada después de la cinemática dle robot omnidireccional con el controlador PID.

3.4.1 Fuente de alimentación.

La alimentación seleccionada para el tren motriz del robot debe ser capaz de alimentar los 12 [V], que se observa en la figura 3.11, que será necesaria para los controladores del motor de corriente directa, al mismo tiempo alimentar dispositivo Raspberry Pi 4 model B, cuya alimentación es de 5 [V] a 1 [A] para que pueda funcionar y alimentar la raspberry Pi Pico 5 [V] a 1 [A], así como los sensores que funcionan también a un voltaje de 5 [V], que es el caso de los sensores de distancia o sensores ultrasónicos, así como el sensor de corriente y el display.

Por lo que fue seleccionada una batería de ácido plomo de 12 [v] a 24 [Ah] o amperes hora.



Figura 3.11: Batería seleccionada para la alimentación general del robot

Imagen obtenida en:

”[https:](https://www.steren.com.mx/bateria-sellada-de-acido-plomo-12-vcc-24-ah.html)

[//www.steren.com.mx/bateria-sellada-de-acido-plomo-12-vcc-24-ah.html](https://www.steren.com.mx/bateria-sellada-de-acido-plomo-12-vcc-24-ah.html)”.

3.4.2 Protección de la electrónica de potencia

Es importante mencionar que los motores pueden tener un riesgo de generar picos de corrientes o una sobrecorriente que pueda afectar la conexión de nuestro circuito.

Por este motivo fue necesario buscar una alternativa para proteger a los circuitos electrónicos, por lo que se plantea usar fusibles con soportes de picos de hasta 30 Amperes. El modelo seleccionado fueron fusibles de tipo americano cerámicos, que se observa en la figura 3.12.



Figura 3.12: Fusible de tipo americano para corrientes de 5 [A].

Imagen obtenida en:

["https://www.blueseas.com/products/5210/AGC_Fuse_-_5_Amp"](https://www.blueseas.com/products/5210/AGC_Fuse_-_5_Amp).

3.4.3 Convertidor de corriente directa.

Si bien la fuente de alimentación de nuestra batería es de 12 [V], necesaria para el funcionamiento de nuestros motores, no resulta conveniente para el resto de la electrónica implementada en nuestro robot, desde el dispositivo Raspberry Pi 4, Raspberry Pi Pico, y en general los sensores y componentes que funcionan a una alimentación de 5 [V] de voltaje de corriente directa, por lo que implementamos un modulo convertidor de voltaje o "*step down*" convertidor de 12[V] - 5[V] para la alimentación de estos dispositivos. En particular utilizamos un modelo que tiene capacidad de manejar corriente de hasta 5 [A], a 25 [W] de potencia, que se muestra en la figura 3.13.



Figura 3.13: Convertidor de corriente directa 12 [V] - 5 [V] a 5[A].

Imagen obtenida en:

["https://uelectronics.com/producto/convertidor-48v-a-12v-5a-60w-step-down/"](https://uelectronics.com/producto/convertidor-48v-a-12v-5a-60w-step-down/).

3.4.4 Configuración de modulación por ancho de pulso para los controladores de motores de corriente directa.

El dispositivo *Raspberry Pi Pico* tiene una salida de 3.3[V], por otra parte, considerando que tenemos dos controladores de motores de corriente directa de diferentes modelos, es decir, el modelo "Talon SR", y el modelo "BTS7960", donde este último utiliza dos pines de funcionamiento, por lo que en total tendremos 6 pines para los 4 motores. Las tablas incluirán las abreviaturas **sin movimiento**[SM] **sentido horario** [SH] y como **sentido anti horario** [SAH] para los movimientos, los cuales serán definidos por nuestra señal de modulación por ancho de pulso, tomando en cuenta la numeración de las ruedas establecida en la figura 2.13, donde las ruedas frontales son las correspondientes al motor 1 y 4 controladas por el controlador modelo "Talon SR", y las ruedas traseras son las controladas por el controlador "BTS7960". En el caso del Talon SR, un ciclo de trabajo aproximado al 50% implica que el controlador de motor "Talon SR" estará detenido, es decir, no dará un sentido ni velocidad de giro, por otro lado, implicará que para el controlador "BTS7960" dará en una velocidad del 50% en la dirección de giro correspondiente al pin de selección, sin embargo, un ciclo de trabajo cercano al 0% en el controlador "Talon SR" le dará una máxima velocidad de giro, sin embargo, en el controlador "BTS7960" implica que el motor estará detenido. Finalmente, un ciclo de trabajo cercano al 100% implicará que para el controlador "Talon SR" será una máxima velocidad de giro en el sentido contrario al 0%, y para el controlador "BTS7960" implicará la máxima velocidad de giro en dirección al pin correspondiente.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	32700	50	0	SM
2	15	0	0	0	SM
2	16	0	0	0	SM
3	17	0	0	0	SM
3	18	0	0	0	SM
4	19	32700	50	0	SM

Tabla 3.1: Caracterización de los controladores para motores de corriente directa para el robot sin movimiento.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	16700	25	6	SH
2	15	32700	50	6	SH
2	16	0	0	6	SH
3	17	32700	50	6	SH
3	18	0	0	6	SH
4	19	16700	0	6	SH

Tabla 3.2: Caracterización de los controladores para motores de corriente directa para un movimiento frontal.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	48700	25	6	SAH
2	15	0	0	6	SAH
2	16	32700	50	6	SAH
3	17	0	0	6	SAH
3	18	32700	50	6	SAH
4	19	48700	0	6	SAH

Tabla 3.3: Caracterización de los controladores para motores de corriente directa para un movimiento de reversa.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	16700	25	6	SH
2	15	0	0	6	SAH
2	16	32700	50	6	SAH
3	17	32700	50	6	SH
3	18	0	0	6	SH
4	19	48700	75	6	SAH

Tabla 3.4: Caracterización de los controladores para motores de corriente directa para un movimiento lateral derecho.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	48700	75	6	SAH
2	15	32700	50	6	SH
2	16	0	0	6	SH
3	17	0	0	6	SAH
3	18	32700	50	6	SAH
4	19	16700	25	6	SH

Tabla 3.5: Caracterización de los controladores para motores de corriente directa para un movimiento lateral izquierdo.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	16700	25	6	SH
2	15	0	0	0	SM
2	16	0	0	0	SM
3	17	32700	50	6	SH
3	18	0	0	6	SH
4	19	32700	50	0	SM

Tabla 3.6: Caracterización de los controladores para motores de corriente directa para un movimiento Diagonal 1.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	48700	75	6	SAH
2	15	0	0	0	SM
2	16	0	0	0	SM
3	17	0	0	6	SAH
3	18	32700	50	6	SAH
4	19	32700	50	0	SM

Tabla 3.7: Caracterización de los controladores para motores de corriente directa para un movimiento Diagonal 2.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	32700	50	0	SM
2	15	32700	50	6	SH
2	16	0	0	6	SH
3	17	0	0	0	SM
3	18	0	0	0	SM
4	19	18700	25	6	SH

Tabla 3.8: Caracterización de los controladores para motores de corriente directa para un movimiento Diagonal 3.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	32700	50	0	SM
2	15	0	0	6	SAH
2	16	32700	50	6	SAH
3	17	0	0	0	SM
3	18	0	0	0	SM
4	19	48700	75	6	SAH

Tabla 3.9: Caracterización de los controladores para motores de corriente directa para un movimiento Diagonal 4.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	19700	25	6	SH
2	15	32700	50	6	SH
2	16	0	0	6	SH
3	17	0	0	6	SAH
3	18	32700	50	6	SAH
4	19	48700	75	6	SAH

Tabla 3.10: Caracterización de los controladores para motores de corriente directa para un movimiento de Giro en 360 grados $[H]$.

Motor	GPIO	PWM	Ciclo de trabajo [%]	$ V_{Out} $	Sentido
1	14	48700	75	6	SAH
2	15	0	0	6	SAH
2	16	32700	50	6	SAH
3	17	32700	50	6	SH
3	18	0	0	6	SH
4	19	19700	25	6	SH

Tabla 3.11: Caracterización de los controladores para motores de corriente directa para un movimiento de Giro en 360 grados $[AH]$.

Sección 3.5.

Sensores de distancia y retroalimentación

La configuración de los sensores de distancia principalmente deben cumplir con la función de detección de próximos obstáculos y realizar una disminución de velocidad, es decir, si dentro del rango de distancia aceptable nuestra lectura en cada sensor indica una distancia promedio de un metro, entonces damos la indicación a nuestro robot a reducir su velocidad en su totalidad hasta que nuevamente una lectura indique una distancia segura mayor de un metro para que nuestro robot pueda recibir nuevamente los tópicos con la información para la modulación PWM para nuestros drivers controladores de motores y poder navegar nuevamente. Ejemplificando lo anterior, tomemos en cuenta el siguiente diagrama de flujo mostrado en la figura 3.14

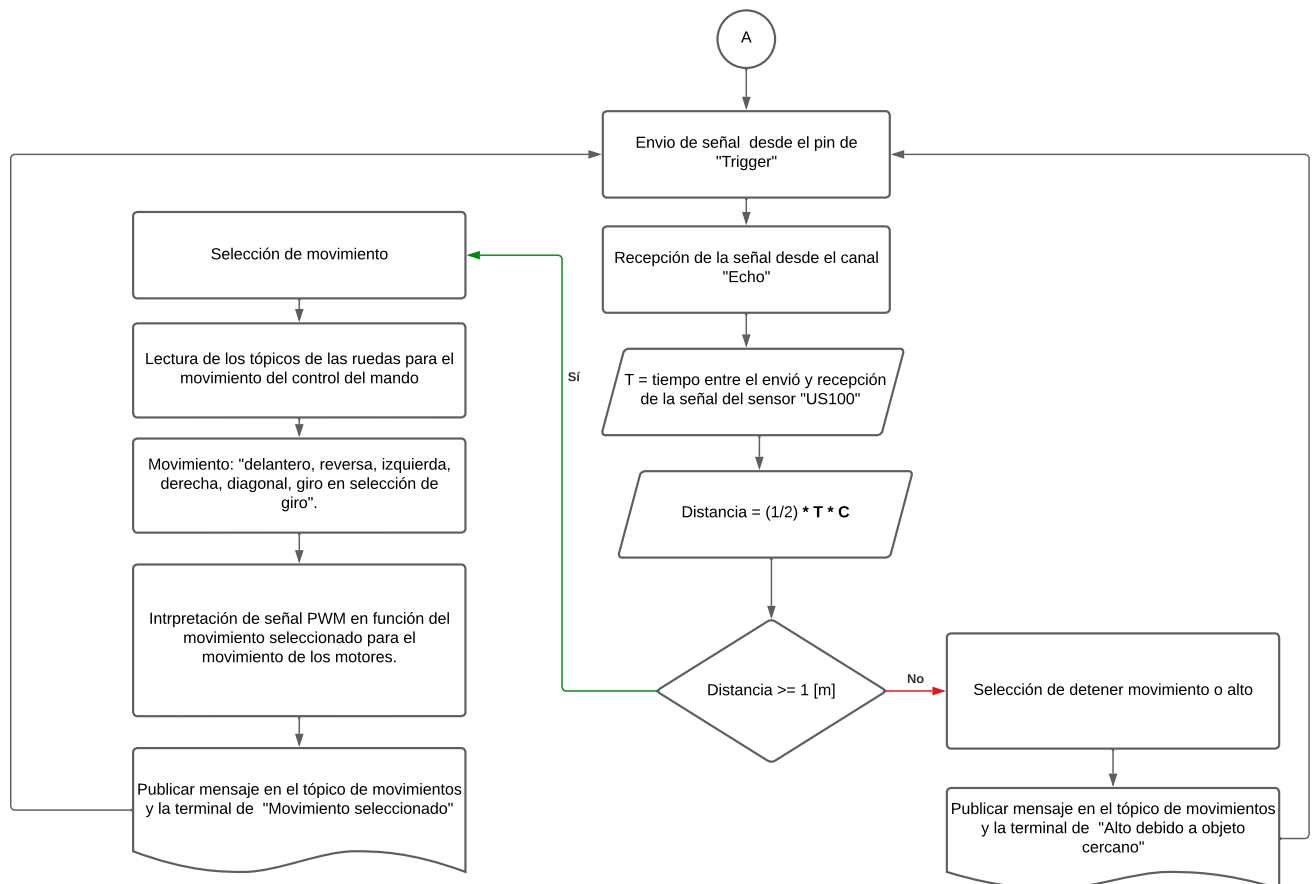


Figura 3.14: Retroalimentación de los sensores de distancia "US-100".

Como se observa en la figura 3.14, la retroalimentación de los sensores decide si se continúa con el movimiento continuo o, por el contrario, hay una interrupción de algún obstáculo que indique una distancia menor a un metro determinará si el movimiento debe ser detenido en su totalidad.

Sección 3.6.

Sensor de corriente y nivel de batería

Por otra parte, la importancia de la alimentación y el nivel de batería determina por completo el funcionamiento de los motores, los sensores, el microcontrolador y la microcomputadora así como las pantallas o displays en el robot, por lo que la retroalimentación del nivel de batería debe estar definida por un rango de voltaje entre 5 [V] como mínimo y 14.5 [V] como máximo, debido a que nuestra batería de corriente directa será cargada y descargada constantemente, entonces podemos decir que en una escala de porcentaje el 0 porciento equivale a 4.9 [V] mientras que el 100 porciento equivale a 14.5 [V] o el rango de escala máximo de la batería seleccionada, entonces esta relación estará ligada a una interpolación determinada por la siguiente ecuación:

$$\text{PorcentajeBateria} = 4.9 * \frac{x}{100} * (14.5 - 4.9) \quad (3.35)$$

Donde la variable "V" es el nivel de lectura de nuestro sensor de corriente previamente ajustado para obtener el valor del voltaje desde nuestro canal de ADC, su diagrama de flujo se representa en la figura 3.15.

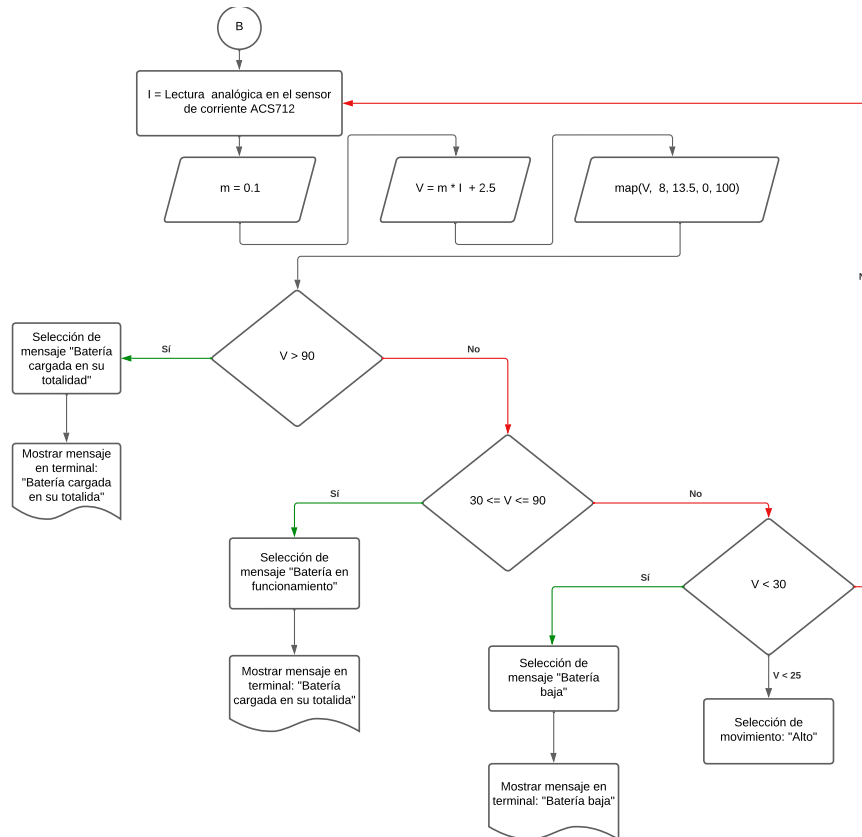


Figura 3.15: Retroalimentación del sensor de corriente ASC712.

Una vez caracterizados los movimientos posibles para nuestro robot móvil omnidireccional, y considerando la retroalimentación con los sensores y actuadores, el siguiente paso es pensar como es que nuestro robot debe interpretar los tópicos y mensajes de ROS para poder navegar de manera eficiente, por lo que tomando en consideración lo anterior el desarrollo del diagrama de flujo del robot es el que se muestra en la figura 3.16; Como se observa en

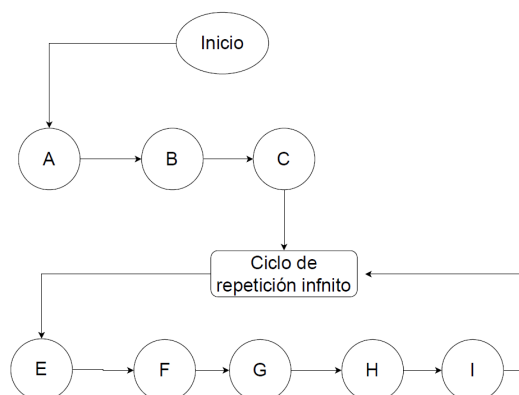


Figura 3.16: Diagrama de flujo de navegación del robot.

la figura 3.17, el diagrama del flujo quedo dividido en segmentos correspondientes a las letras "A,B,C,D,E,F,G,H,I", con el objetivo de simplificar el siagrama de flujo, a continuación detallaremos los procesos que se encuentran en cada segmento. En la figura 3.17, se observa la inicialización del programa cargando las librerías correspondientes y haciendo la declaración de variables globales, pines reservados y se crearan algunas funciones adicionales.

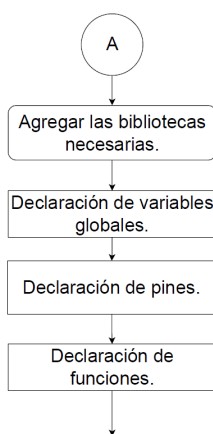


Figura 3.17: Diagrama de flujo de navegación del robot, segmento A.

En la figura 3.18, se observa la evaluación del agente de ROS para establecer comunicación con el cliente para la comunicación entre nodos y tópicos del robot, es decir que exista comunicación serial, en caso contrario se podrá asistir a tomar decisiones de movimiento desde el control "KY-024" para sus movimientos. En la figura 3.19, nos encontramos dentro

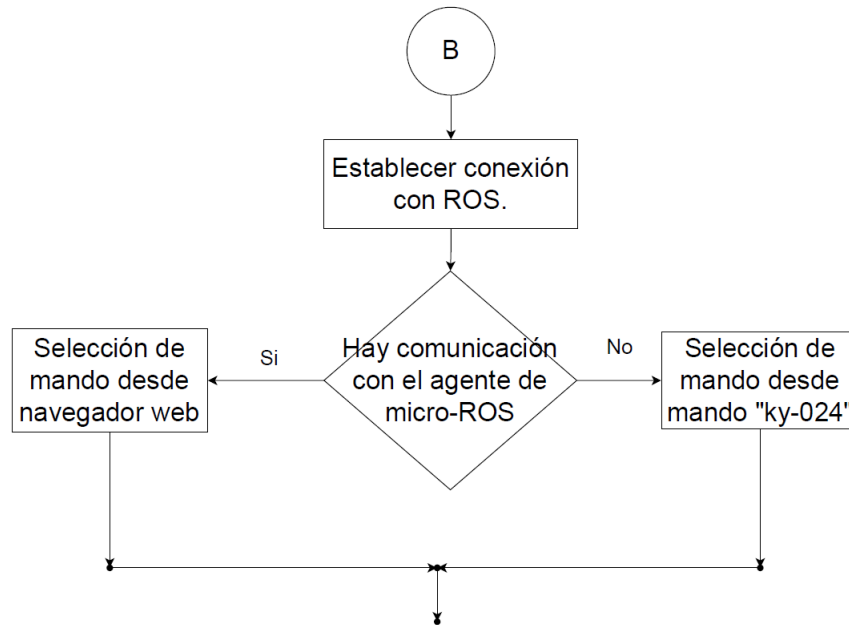


Figura 3.18: Diagrama de flujo de navegación del robot, segmento B.

de la función principal, donde antes de entrar al ciclo de repetición habilitaremos los pines de nuestro microcontrolador "RP2040", declararemos algunas variables locales dentro de la función.

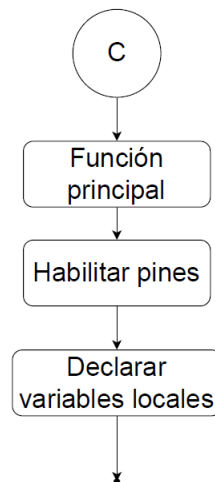


Figura 3.19: Diagrama de flujo de navegación del robot, segmento C.

En la figura 3.20, no se observa directamente en el la figura 3.16 del diagrama de flujo principal, sin embargó esta secuencia será importante, debido a que en esta sección se elejira en base a la selección del control por navegador desde la comunicación con ROS, o una comunicación desde el control "ky-024", para seleccionar los movimientos del robot en los sentidos configurados desde las tablas 5.1 a 5.11, con su correspondiente ciclo de trabajo correspondiente a su señal de modulación por ancho de pulso. y según el movimiento seleccionado para su navegación.

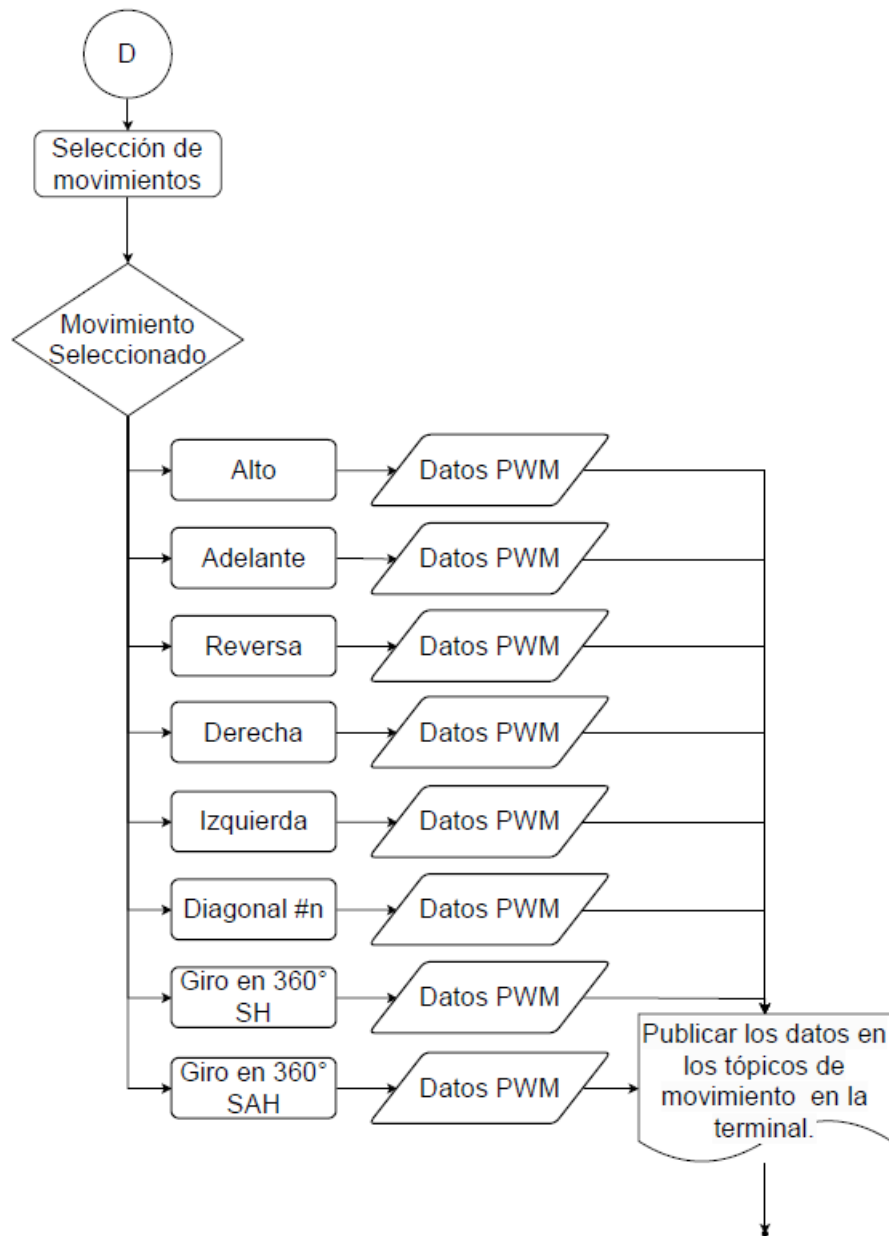


Figura 3.20: Diagrama de flujo de navegación del robot, segmento D.

En las figuras 3.21, 3.22, 3.21, son básicamente secuencias de lectura y mostrar mensaje de los componentes electronicos del reloj de tiempo real, "RTC", la pantalla LCD 16x2, y la secuencia de luces de la tira LED RGB correspondientemente.



Figura 3.21: Diagrama de flujo de navegación del robot, segmento E.

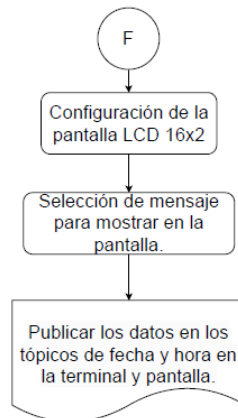


Figura 3.22: Diagrama de flujo de navegación del robot, segmento F.

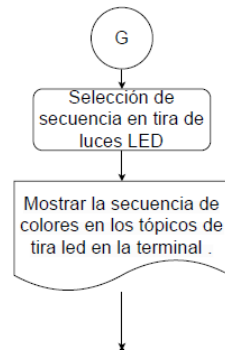


Figura 3.23: Diagrama de flujo de navegación del robot, segmento G.

Finalmente en las secuencias 3.24 y 3.25, que son similares a los diagramas ?? y 3.15, son básicamente la misma sentencia lógica pero adaptada a la compatibilidad de la propuesta del diagrama de flujo del robot de navegación de la figura 3.16.

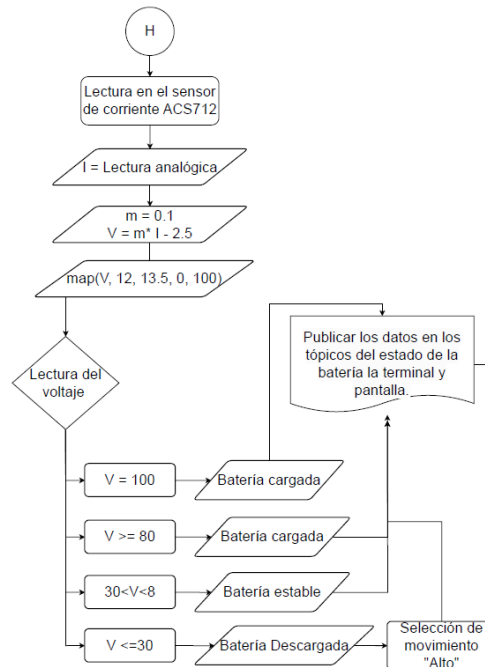


Figura 3.24: Diagrama de flujo de navegación del robot, segmento H.

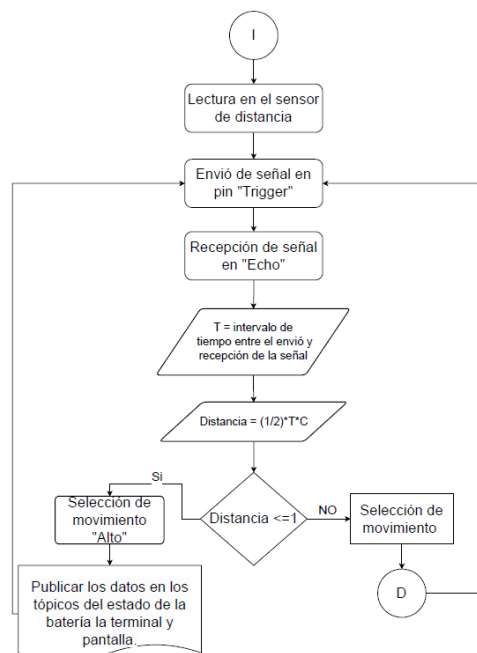
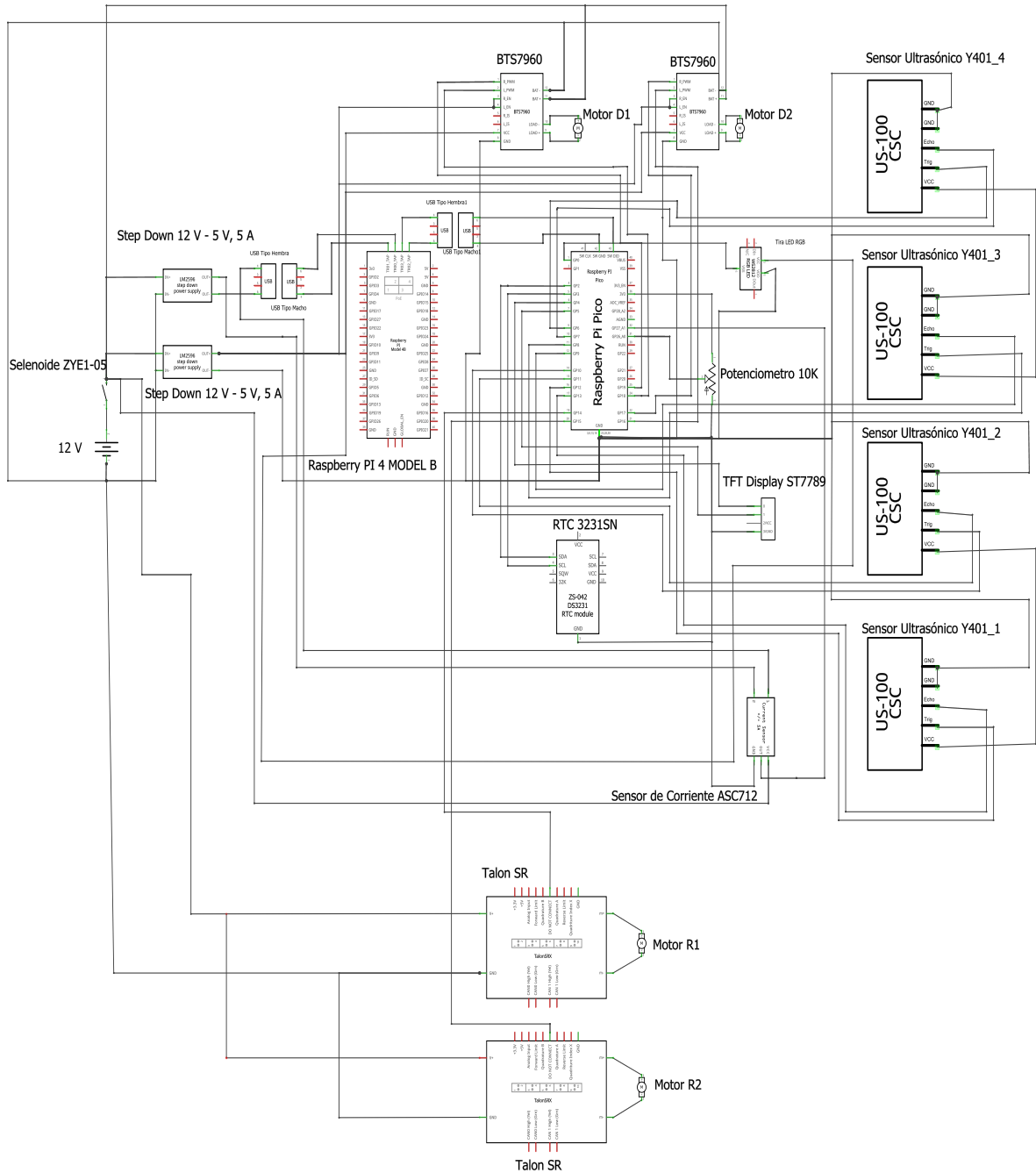


Figura 3.25: Diagrama de flujo de navegación del robot, segmento I.

3.8.1 Propuesta de diseño del circuito.

En la figura 3.26 se muestra la propuesta de los diagramas de conexión del circuito.



fritzing

Figura 3.26: Propuesta de los diagramas de conexión del circuito impreso o PCB.

En la tabla que se muestra a continuación se definen los GPIOs de la raspberry pi pico y la configuración de entrada salida, así como la función que tienen o si se dejan reservados esos pines.

PIN	GPIO	Configuración E/S	Configuración de GPIO	Función
1	0	Salida	UART:TX	Para la tira de leds
2	1	Sin asignar	Sin asignar	Para futura disponibilidad
4	2	Entrada	I2C:SDA	RTC DS3231:SDA
5	3	Salida	I2C:SCL	RTC DS3231:SCL
6	4	Salida	I2C:SDA	Display:SDA
7	5	Entrada	I2C:SCL	Display:SCL
9	6	Salida	Digital	Sensor Y4001:Trigger
10	7	Entrada	Digital	Sensor Y4001:Echo
11	8	Salida	Digital	Sensor Y4001:Trigger
12	9	Entrada	Digital	Sensor Y4001:Echo
14	10	Salida	Digital	Sensor Y4001:Trigger
15	11	Entrada	Digital	Sensor Y4001:Echo
16	12	Salida	Digital	Sensor Y4001:Trigger
17	13	Entrada	Digital	Sensor Y4001:Echo
19	14	Salida	PWM	Señal PWM
20	15	Salida	PWM	Señal PWM
21	16	Salida	PWM	Señal PWM
22	17	Salida	PWM	Señal PWM
24	18	Salida	PWM	Señal PWM
25	19	Salida	PWM	Señal PWM
26	20	Sin asignar	Sin asignar	Para futura disponibilidad
27	21	Sin asignar	Sin asignar	Para futura disponibilidad
29	22	Entrada	Push	Push Button de interrupción
31	26	Entrada	ADC	Lectura de sensor ASC712
32	27	Entrada	ADC	Lectura analógica
34	28	Entrada	ADC	Lectura analógica

Tabla 3.12: Caracterización de Pines y asignaciones del microcontrolador RP2040.

En la figura 3.27 se observa el esquema general de conexiones en el robot, tomando en cuenta la simbología y el esquema de conexiones desde la alimentación de la batería, los controladores de motores de corriente directa, los módulos reductores de voltaje a 5[V] que alimentarán la PCB y los sensores y otros componentes electrónicos. El diagrama completo se encuentra en el apéndice A de este documento.

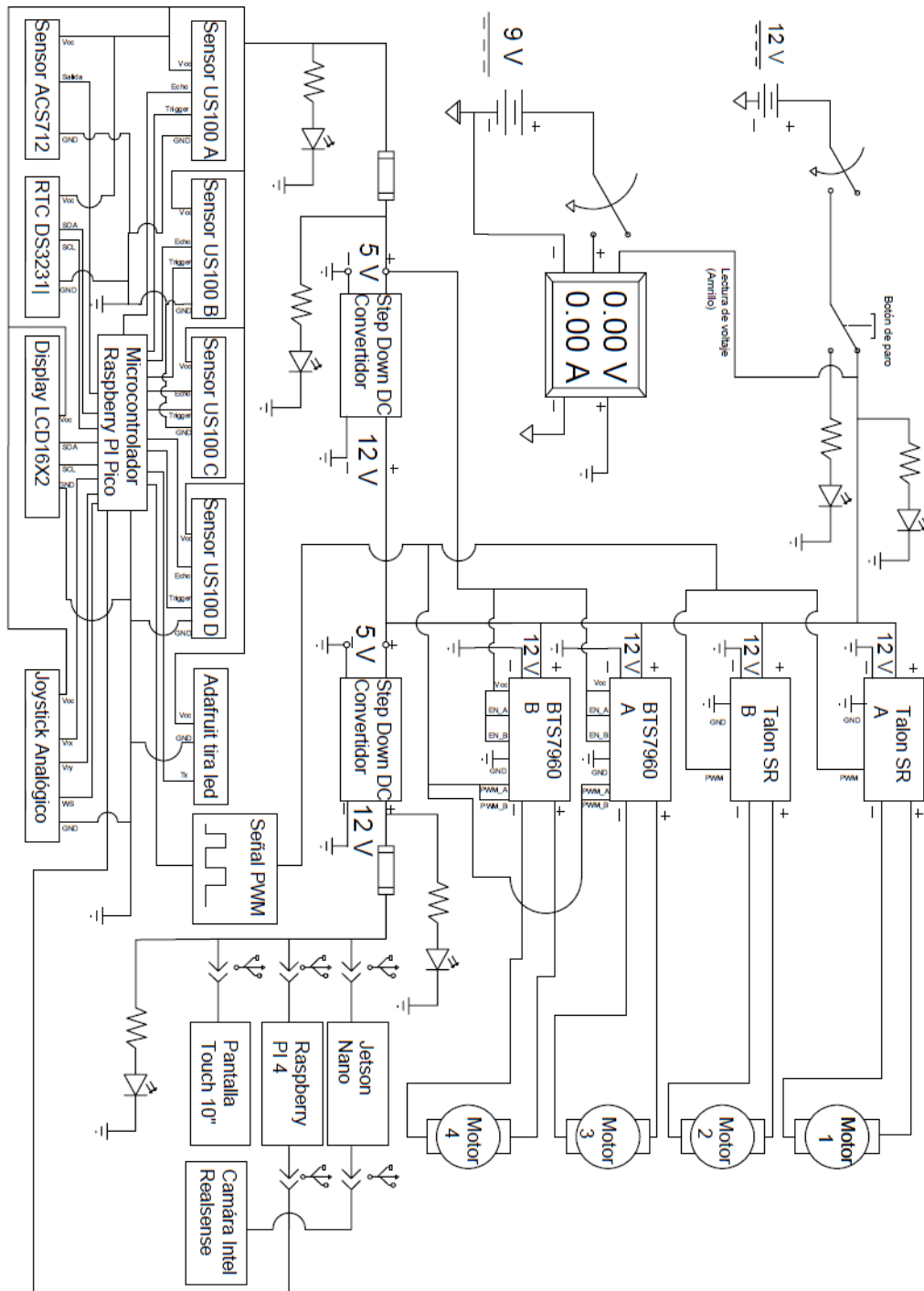


Figura 3.27: Diagrama de conexión eléctrico.

En la figura 3.28 se observa la simbología de algunas conexiones y referencias a componentes electrónicos de la figura 3.27, el esquema completo se encuentra en el apéndice A.

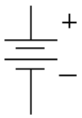
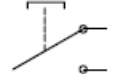
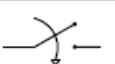
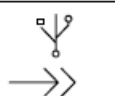
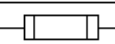
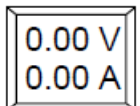
Tabla de Símbolos	
Símbolo	Descripción
	Fuente de alimentación con polarización de corriente directa
	Corriente directa
	Tierra de la batería de alimentación 12 V
	Tierra de componentes
	Tierra de batería de alimentación de 9 V
	Descripción
	Diodo led
	Botón de paro
	Interruptor
	Conector tipo USB
	Fusible de norma americana
	Voltímetro y Amperímetro Digital

Figura 3.28: Propuesta de los diagramas de conexión.

3.8.2 Propuesta de Diseño del circuito en PCB.

La propuesta del diseño fue realizada en el software de *kicad 7.0*. A continuación se muestra en la figura 3.29 la propuesta del diseño en visión 3D de la parte frontal que será implementada físicamente, mientras que el diseño de las pistas se muestra en la figura 3.30. Los planos se encuentran en el apéndice A. de este documento.

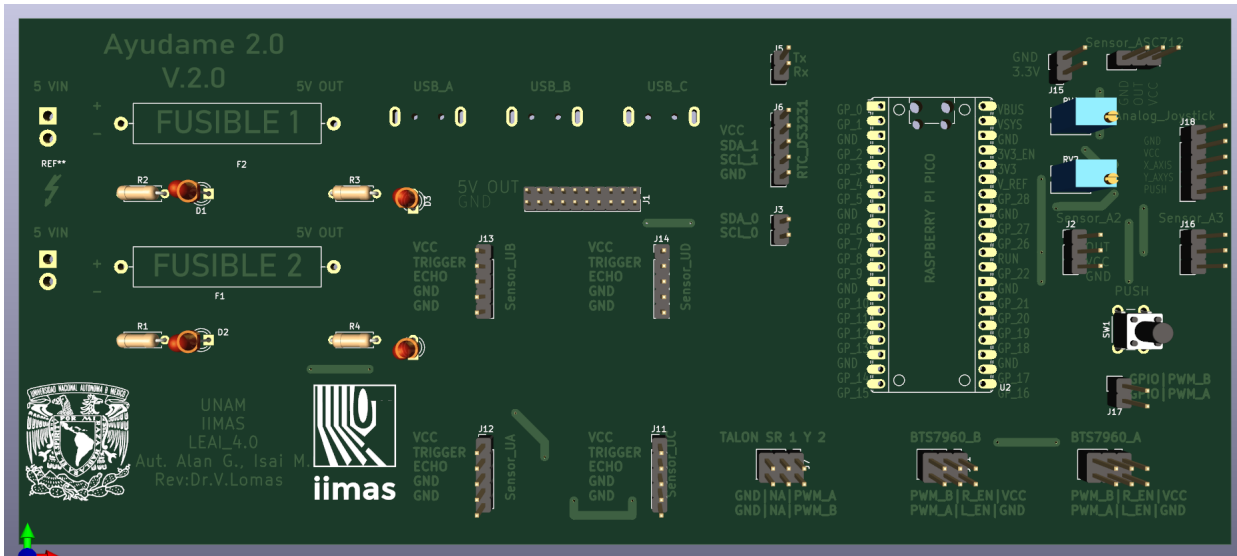


Figura 3.29: Propuesta del diseño de la PCB visto desde simulación 3D para la parte frontal.

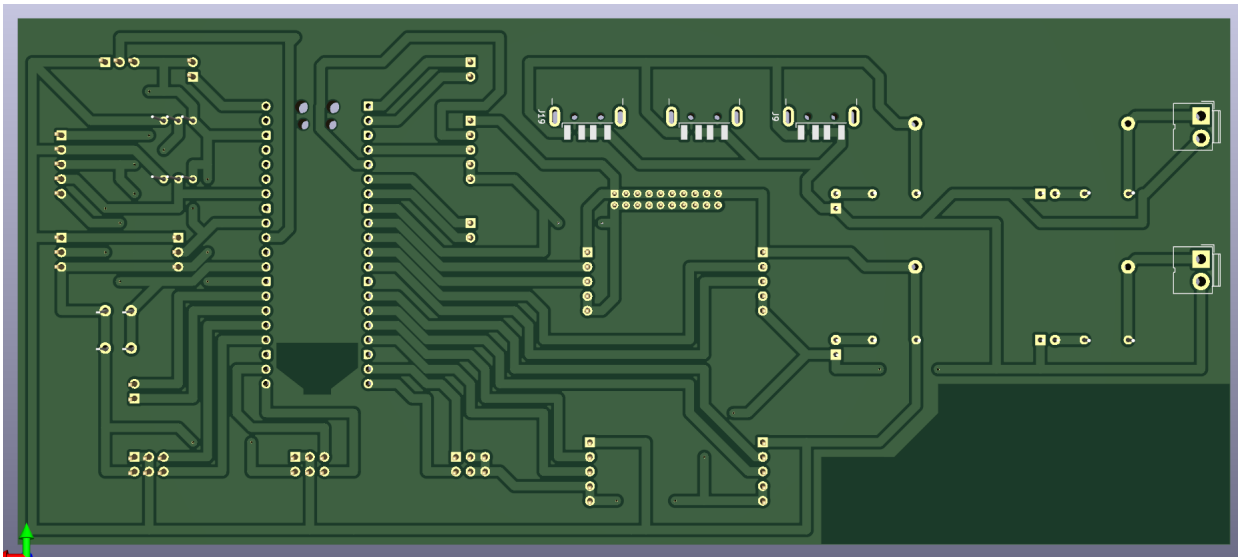


Figura 3.30: Propuesta del diseño de la PCB visto desde simulación 3D para el reverso de las placas donde se encuentran la mayoría de pistas.

3.9.1 Instalación de *ROS 2* Humble

Para lograr instalar ROS debemos seguir las siguientes instrucciones.

El primer paso es tener una configuración regional que admita UTF-8, entonces colocamos en la terminal los siguientes comandos.

```
1 locale # check for UTF-8
2 sudo apt update && sudo apt install locales
3 sudo locale-gen en_US en_US.UTF-8
4 sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
5 export LANG=en_US.UTF-8
6 locale # verify settings
```

Ahora se deberá agregar el repositorio apt de *ROS 2* a su sistema. Primero asegúrese de que el repositorio de Ubuntu Universe esté habilitado.

```
1 sudo apt install software-properties-common
2 sudo add-apt-repository universe
```

Agregamos la clave GPG de *ROS 2* con apt.

```
1 sudo apt update && sudo apt install curl
2 sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.
  key -o /usr/share/keyrings/ros-archive-keyring.gpg
```

Agregar el repositorio a su lista de fuentes.

```
1 echo "deb [arch=$(dpkg --print-architecture)signed-by=/usr/share/keyrings/
  ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/
  os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources
  .list.d/ros2.list > /dev/null
```

Actualizamos los cachés de su repositorio apt después de configurar los repositorios e instalar *ROS 2*.

```
1 sudo apt update
```

Los paquetes de *ROS 2* se basan en sistemas Ubuntu que se actualizan con frecuencia. Siempre se recomienda que se asegure de que su sistema esté actualizado antes de instalar nuevos paquetes.

```
1 sudo apt upgrade
2 sudo apt install ros-humble-desktop
3 sudo apt install ros-humble-ros-base
4 sudo apt install ros-dev-tools
```

Entonces ROS2 ha quedado instalado

3.9.2 Configuración del entorno de *ROS 2* Humble

Se debe ejecutar este comando en cada nuevo shell que abra para tener acceso a los comandos de *ROS 2*.

```
1 # Replace ".bash" with your shell if you're not using bash
2 # Possible values are: setup.bash, setup.sh, setup.zsh
3 source /opt/ros/humble/setup.bash
```

Se agrega el abastecimiento a su secuencia de comandos de inicio de shell.

Si no desea tener que buscar el archivo de instalación cada vez que abre un nuevo shell (omitiendo el paso 1), puede agregar el comando a la secuencia de comandos de inicio de su shell:

```
1 echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
```

La obtención de archivos de configuración de *ROS 2* establecerá varias variables de entorno necesarias para operar *ROS 2*. Si alguna vez tiene problemas para encontrar o usar sus paquetes de *ROS 2*, asegúrese de que su entorno esté configurado correctamente con el siguiente comando:

```
1 printenv | grep -i ROS
```

Se debe verificar que las variables como *ROSDISTRO* y *ROSVERSION* estén configuradas.

```
1 ROS_VERSION=2
2 ROS_PYTHON_VERSION=3
3 ROS_DISTRO=humble
```

Se debe consultar el artículo de ID de dominio para obtener detalles sobre los ID de dominio de ROS. Una vez que haya determinado un número entero único para su grupo de agentes *ROS 2*, puede configurar la variable de entorno con el siguiente comando:

```
1 export ROS_DOMAIN_ID=<your_domain_id>
```

Para mantener esta configuración entre sesiones de shell, puede agregar el comando a su secuencia de comandos de inicio de shell:

```
1 echo "export ROS_DOMAIN_ID=<your_domain_id>" >> ~/.bashrc
```

De forma predeterminada, la comunicación de ROS 2 no se limita a localhost. La variable de entorno *ROSLOCALHOSTONLY* le permite limitar la comunicación de *ROS 2* solo a localhost. Esto significa que su sistema ROS 2 y sus temas, servicios y acciones no serán visibles para otras computadoras en la red local. El uso de *ROSLOCALHOSTONLY* es útil en ciertos entornos, como las aulas, donde varios robots pueden publicar en el mismo tema y provocar comportamientos extraños. Puede establecer la variable de entorno con el siguiente comando:

```
1 export ROS_LOCALHOST_ONLY=1
```

Para mantener esta configuración entre sesiones de shell, puede agregar el comando a su secuencia de comandos de inicio de shell:

```
1 echo "export ROS_LOCALHOST_ONLY=1" >> ~/.bashrc
```

3.9.3 Configuración del control desde navegador con *ROS Bridge*

El planteamiento para utilizar el control desde navegador debe cumplir con la posibilidad de poder utilizarlo con html al abrirlo desde un enlace web accesible para todo el público, por lo que ROS2 bridge nos facilita esta opción.

La instalación de *ROS 2* bridge es bastante sencilla, ya que debemos ejecutar los siguientes comandos.

```
1 sudo apt install ros-humble-rosbridge-server
2 git clone https://github.com/LEAI-IIMAS/AYUDAME.git
3 source /opt/ros/humble/setup.bash
4 sudo apt install firefox
```

Ahora que hemos instalado *ROS 2* bridge, para ejecutar el control desde navegador debemos ejecutar el siguiente comando:

```
1 ros2 launch rosbridge_server rosbridge_websocket_launch.xml
```

Ahora, en una nueva terminal abriremos el navegador web para interactuar con el control desde navegador colocando nuestra dirección de instalación de *ROS 2* bridge:

```
1 cd AYUDAME-main/Software/ros/virtual-joystick-cuatro-ruedas
2 firefox index.html
```

Con el comando "F12" desde el navegador web podremos observar los valores que retornaran para el ciclo de trabajo que posteriormente movera los drivers de los motores para la caracterización del movimiento, como se observa desde la terminal en la figura 3.31.

```
alan@alan-virtual-machine:~$ cd AYUDAME-main/Software/ros/virtual-joystick-cuatro-ruedas
alan@alan-virtual-machine:~/AYUDAME-main/Software/ros/virtual-joystick-cuatro-ruedas$ firefox index.html
update.go:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/share/gimp/2.0/help /usr/share/gimp/2.0/help none bind,ro 0 0): cannot open directory "/var/lib": pe
on denied
update.go:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/share/xubuntu-docs /usr/share/xubuntu-docs none bind,ro 0 0): cannot open directory "/var/lib": pe
denied
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
Gtk-Message: 14:06:35.06: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please try to not load it.
[GFX-1]: glxtest: VA-API test failed: failed to initialise VA-API connection.
ATTENTION: default value of option mesa_glxthread overridden by environment.
ALSA lib conf.c:4120:(snd_config_update_r) Cannot access file /usr/share/alsa/alsa.conf
ALSA lib seq.c:935:(snd_seq_open_noupdate) Unknown SEQ default
```

Figura 3.31: Ejecución desde la terminal para el Joystick.

A continuación se abrirá un navegador web con el que podremos interactuar directamente, como se observa en la figura 3.32.

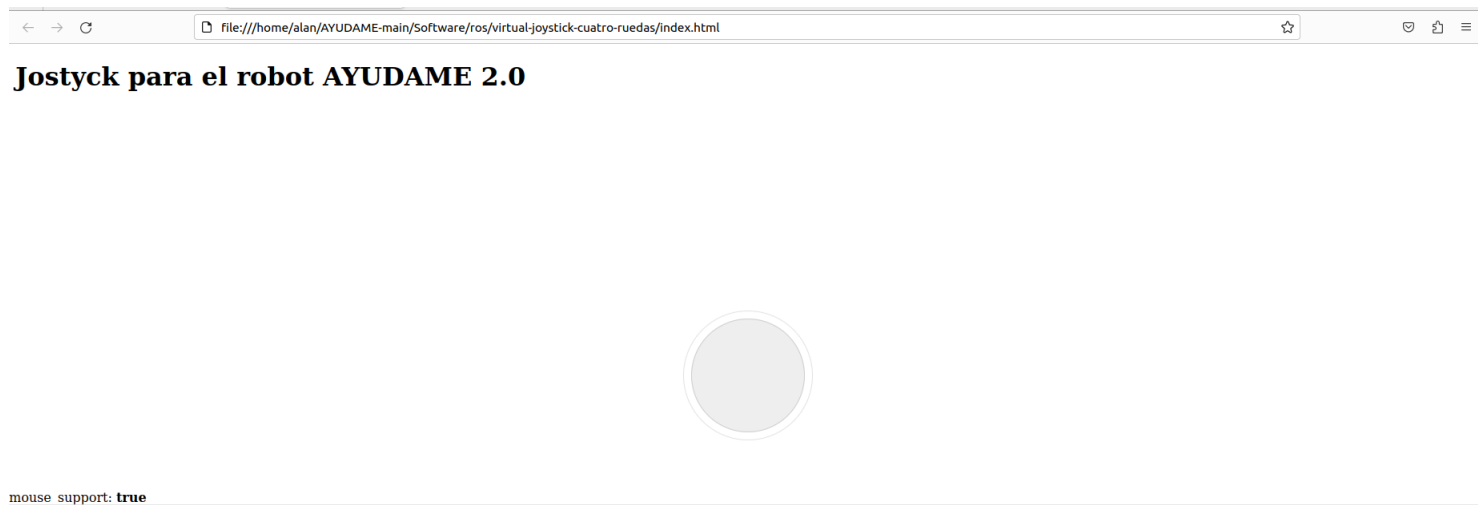


Figura 3.32: Visualización desde el control desde navegador con nuestro navegador web.

Al presionar F12 para observar la terminal, notaremos que al interactuar con el control desde navegador se enviará la respuesta correspondiente al necesario para el ciclo de trabajo, como se muestra a continuación en la figura 3.33.

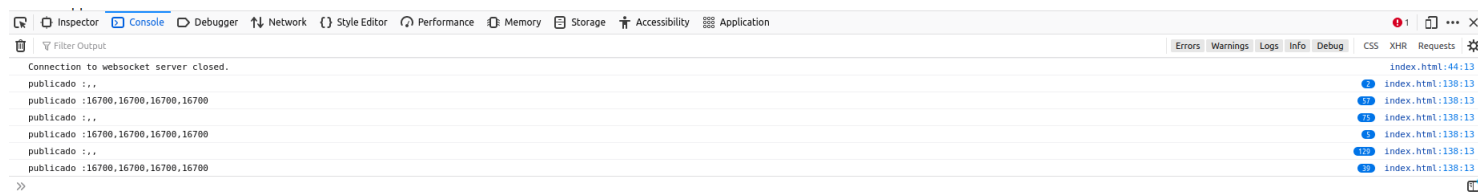


Figura 3.33: Visualización de la terminal web al interactuar con el Joystick.

Finalmente para observar los tópicos de *ROS 2 bridge*, en una nueva terminal introduciremos los comandos:

```
1 ros2 topic list
2 source /opt/ros/humble/setup.bash
3 ros2 topic echo /front_left_wheel
```


El resultado que veremos en la terminal serán los tópicos de las 4 ruedas que enviarán el valor acorde a los valores de nuestra señal PWM, que se observan en la figura 3.34.

```
alan@alan-virtual-machine:~/AYUDAME-main/Software/ros/virtual-joystick-cuatro-ruedas$ ros2 topic list
/back_left_wheel
/back_right_wheel
/client_count
/connected_clients
/front_left_wheel
/front_right_wheel
/parameter_events
/pico_publisher
```

Figura 3.34: Vista de los tópicos en general con los que estamos interactuando.

Al introducir el comando echo para uno de los tópicos de la rueda podremos observar el valor que esta enviando que coincide con el de la terminal del navegador web, que se observa en la figura 3.35.

```
data: 16700.0
---
data: 16700.0
---
data: 16700.0
---
data: 16700.0
---
data: 16700.0
---
data: 16700.0
---
data: 16700.0
---
data: 16700.0
---
```

Figura 3.35: Valor que esta recibiendo el tópicopara una de las ruedas.

3.9.4 *micro-ROS* para el dispositivo *Raspberry Pico*.

Como sabemos, *micro-ROS* es en pocas palabras la extensión de *ROS 2* enfocada a micro-controladores como el "RP2040". La configuración de *micro-ROS* para el dispositivo *raspberrypi pico*, no es compatible con el entorno de micropython, el cual es el "IDE" enfocado al procesador "RP2040", sin embargo, existe la opción de aplicar un entorno de *micro-ROS* en lenguaje de programación en C/C++, por lo que además de que es un lenguaje compilado ofrece una mayor optimización al momento de estructurar nuestro código con los sensores y drivers que estamos buscando, por lo que a continuación partiremos desde la instalación de *micro ros* para el entorno del dispositivo *raspberrypi pico* que se utilizó como base para formar los tópicos y servicios.

Con el siguiente comando se instalarán las dependencias necesarias:

```
1 sudo apt install build-essential cmake gcc-arm-none-eabi libnewlib-arm-  
   none-eabidxygen git python3
```

Ahora creamos un espacio de trabajo para poder guardar las fuentes necesarias para trabajar con el entorno de *micro-ros*, procediendo a crear una carpeta con el nombre de "*micro ros*" e instalando las dependencias necesarias con los siguientes comandos.

```
1 mkdir -p ~/micro_ros_ws/src  
2 cd ~/micro_ros_ws/src  
3 git clone --recurse-submodules https://github.com/raspberrypi/pico-sdk.git  
4 git clone https://github.com/micro-ROS/micro_ros_raspberrypi_pico_sdk.git
```

El primer repositorio tiene por nombre "Pi Pico SDK" proporcionado por la fundación "Raspberry". El segundo repositorio contiene un pre compilado de *micro-ROS* junto con un ejemplo.

Habilitamos el agente serial con el comando:

```
1 sudo snap set core experimental.hotplug=truett
```

Posteriormente reiniciamos el puerto usb con el comando:

```
1 sudo systemctl restart snapd
```

[language=bash] Verificamos que se muestre el serial port de snapd:pico
(Nota: Debe mostrarse la terminal el siguiente mensaje: name: serial-port)

```
1 summary: allows accessing a specific serial port  
2 plugs:  
3 micro-ros-agent  
4 slots:  
5 snapd:pico (allows accessing a specific serial port)
```

Conectamos el Plug al slot de comunicación serial con el comando:

```
1 snap connect micro-ros-agent:serial-port snapd:pico
```

Finalmente introducimos el comando para observar en la terminal el agente de micro ros en comunicación serial.

```
1 sudo micro-ros-agent serial --dev /dev/ttyACM0 baudrate=115200
```

Lo que veremos en la terminal despues de ingresar el comando anterior se observa en la figura 3.36.

```
alan@alan-virtual-machine:~$ sudo micro-ros-agent serial --dev /dev/ttyACM0 baudrate=115200
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
[1678391647.676568] info | TermiosAgentLinux.cpp | init | running...
[1678391647.677024] info | Root.cpp | set_verbose_level | logger setup
[1678391648.176598] info | Root.cpp | create_client | create
[1678391648.176689] info | SessionManager.hpp | establish_session | session established
[1678391648.245073] info | ProxyClient.cpp | create_participant | participant created
[1678391648.259482] info | ProxyClient.cpp | create_topic | topic created
[1678391648.273530] info | ProxyClient.cpp | create_publisher | publisher created
[1678391648.294370] info | ProxyClient.cpp | create_datawriter | datawriter created
█
```

Figura 3.36: Conexión con el cliente de *micro-ros*.

3.9.5 ROS PID.

El controlador de tipo Proporcional - Integrativo - Derivativo es un tipo de control o por sus siglas PID, se utiliza principalmente en sistemas de retroalimentación de lazo cerrado, en este caso lo utilizaremos para controlar la velocidad en la salida de los motores en función de la corrección de error en la velocidad para obtener nuestra velocidad en función del sistema de nuestro robot para poder lograr definir los movimientos y mejorar la salida de la velocidad deseada de nuestros motores para lograr los movimientos deseados.

La ecuación de este controlador está dada de la forma:

$$u(t) = K_c e(t) + \frac{K_c}{\tau_i} \int_0^t e(t) dt + K_c \tau_d \frac{de(t)}{dt} \quad (3.36)$$

La biblioteca que utilizaremos para ROS2 Controller PID, es necesario de igual manera ejecutar una serie de instrucciones.

Procedemos a realizar la instalación para el controlador con los siguientes comandos.

```
1 sudo apt-get install ros-humble-pid
2 cd catkin_ws/src
3 git clone https://bitbucket.org/AndyZe/pid.git
4 cd ..
5 catkin_make
```

Si quedo instalado correctamente el complemento, podemos ejecutar un ejemplo de un servomotor con el siguiente comando:

```
1 roslaunch pid servo_sim.launch
```

A continuación se abre una ventana emergente que muestra las entradas y salidas del controlador PID.

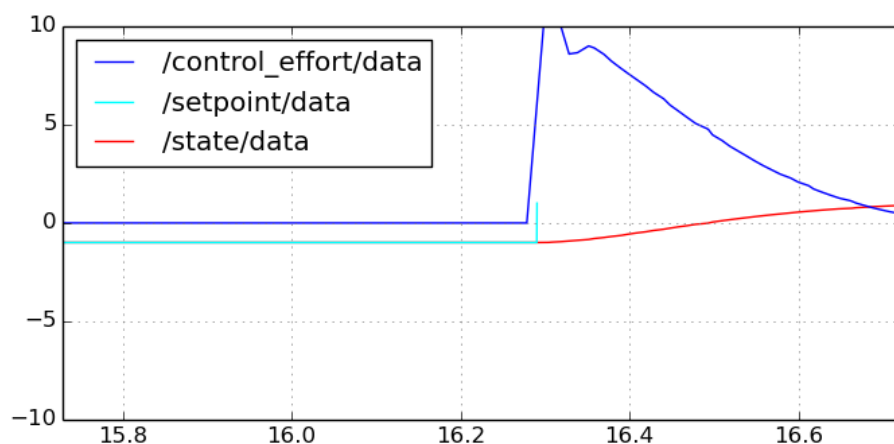


Figura 3.37: Gráfica de la entrada y salidas del controlador PID.

Los tópicos resultantes se muestran en la siguiente figura.

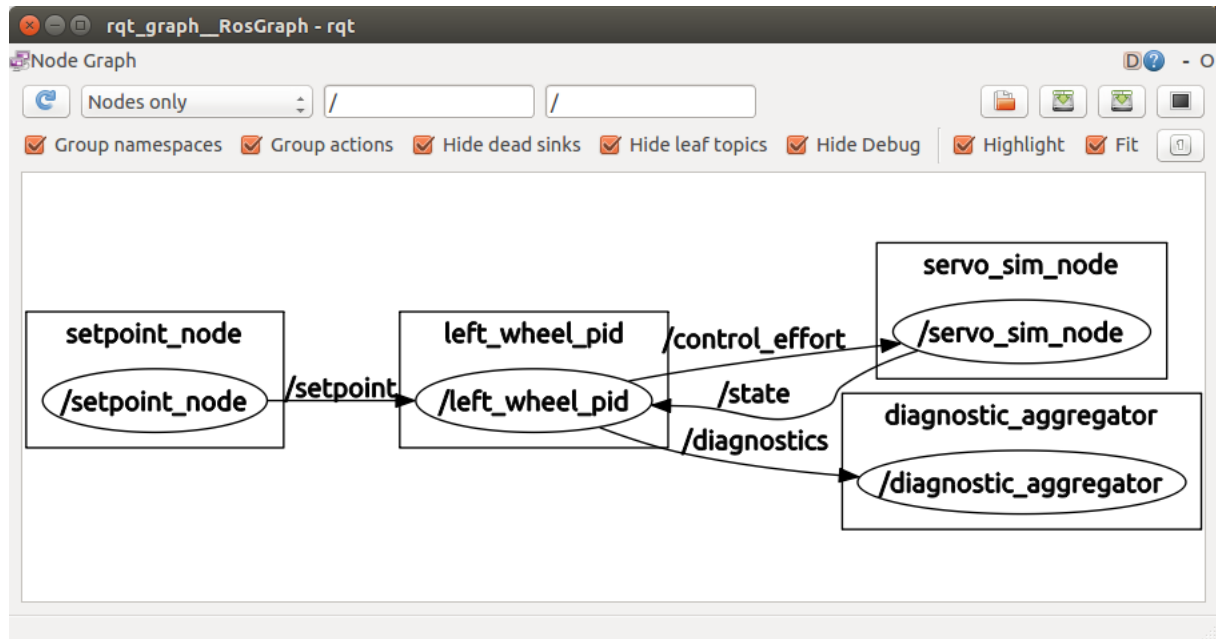


Figura 3.38: Gráfica de la entrada y salidas del controlador PID.

El *setpoint_node* publica su punto de ajuste variable en el tiempo en el controlador PID que se ejecuta en el nodo *left_wheel_pid*, que aplica correcciones a través del tema *control_effort* al *servo_sim_node*. El *servo_sim_node* publica el valor actual de la posición servo simulada en el tema *state*, al que se suscribe el controlador PID y en el que basa su *control_effort*.

3.9.6 Configuración de tópicos.

Una vez que hemos configurado nuestros entornos de ROS2 para trabajar con la raspberry pi pico ahora es necesario definir los tópicos y servicios que utilizaremos para nuestro robot, por lo que serán ejemplificados en el siguiente esquema.

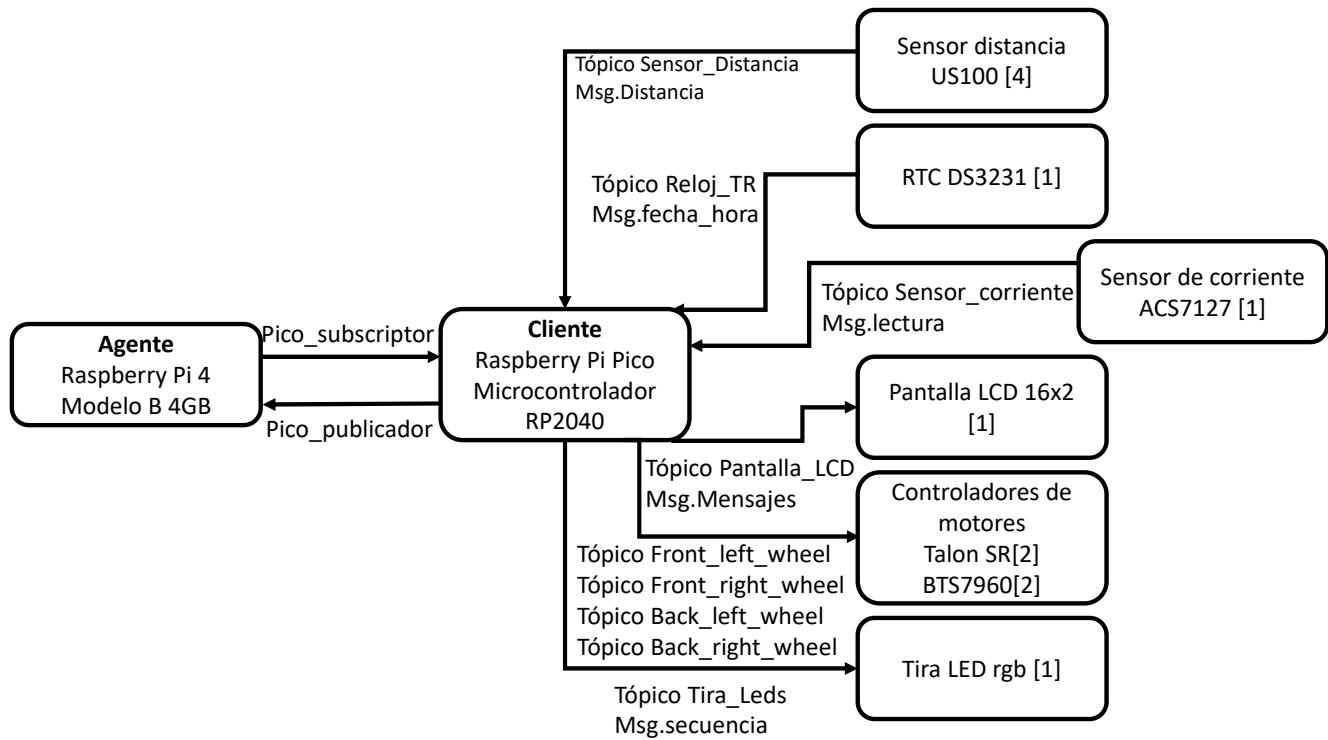


Figura 3.39: Esquema de configuración de ROS para los topicos para el controlador PID.

RESULTADOS Y DISCUSIÓN

Sección 4.1.

Resultado del planchado y serigrafía de la PCB

El proceso de impresión, diagramas que se encuentran en el apéndice A de este documento, el proceso de planchado y serigrafía de la PCB así como la perforación y la soldadura de la placa se obtuvo el siguiente resultado en la figura 4.1, donde comparada con la figura 3.29, el resultado de este diseño logra ser similar en la implementación al diseño propuesto en la simulación, por lo que es un resultado satisfactorio.

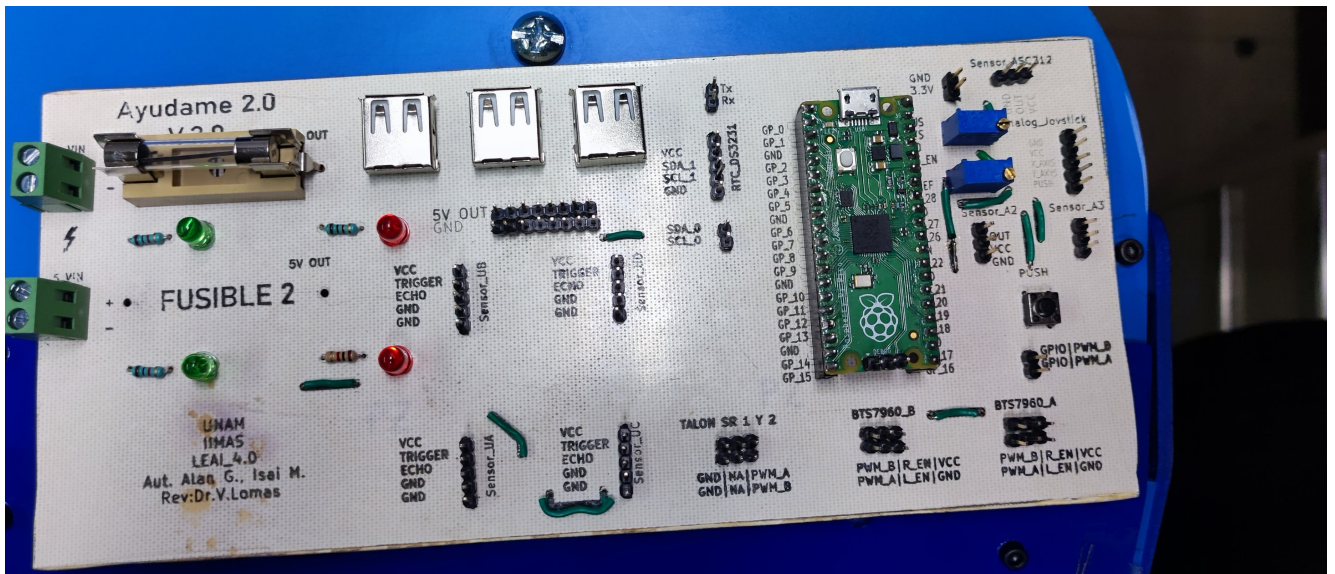


Figura 4.1: Resultado de nuestra PCB de en nuestra cara frontal con los componentes soldados lista para realizar nuestra coexión con el robot.

Sección 4.2.

Resultado obtenido en la señal PWM y corriente del motor.

Los resultados del driver de motor Talon SR se observan en la siguiente tabla.

RAW (BITS)	PWM Raspberry Pico	Ciclo de trabajo [%]	Voltaje de salida [V]
16	255	0	12
1024	16700	25	6
2 2048	32700	50	0
3060	49000	75	-6
4096	64400	100	-12

Tabla 4.1: Caracterización del Talon SR.

Los resultados del driver BTS7960 se observan en la siguiente tabla.

RAW (BITS)	PWM Raspberry Pico	Ciclo de trabajo [%]	Voltaje de salida [V]
16	255	0	0
1024	16700	25	3
2048	32700	50	6
3060	49000	75	9
4096	64400	100	12

Tabla 4.2: Caracterización del driver BTS7960.

Los valores obtenidos para cada uno de los motores se mostraran en las siguientes tablas. Los valores caracterizados para obtener un voltaje igual en la salida con el controlador Talon SR es para:

Motor 1 y Motor 2 para driver Talon SR:

PWM	Ciclo de Trabajo [%]	Voltaje [V]	Corriente [A]
255	0	12	4.2
16700	25	6	2.1
32700	50	0	0
49000	75	-6	2.1
64400	100	-12	4.2

Tabla 4.3: Caracterización de los Motores 1 y 2.

La caracterización cambia debido al funcionamiento del driver BTS7960 para los motores 3 y 4.

PWM	Ciclo de Trabajo [%]	Voltaje [V]	Corriente [A]
255	0	0	0
16700	25	3	1.1
32700	50	6	2.1
49000	75	9	3.1
64400	100	12	4.2

Tabla 4.4: Caracterización de los Motores 3 y 4.

La caracterización de los movimientos a un voltaje de salida definida se pueden expresar en la siguiente tabla.

MOTOR	Voltaje [V]	PWM	Ciclo de Trabajo [%]	PIN
1	6	16700	25	14
2	6	16700	25	15
3	6	32700	50	16,17
4	6	32700	50	18,19

Tabla 4.5: Caracterización de los Motores a una salida de 6 [V].

La caracterización de los movimientos a un voltaje de salida definida se pueden expresar en la siguiente tabla.

MOTOR	Voltaje [V]	PWM	Ciclo de Trabajo [%]	PIN
1	0	32700	50	14
2	0	32700	50	15
3	0	255	0	16,17
4	0	255	0	18,19

Tabla 4.6: Caracterización de los Motores a una salida de 0 [V].

La caracterización de los movimientos a un voltaje de salida definida se pueden expresar en la siguiente tabla.

MOTOR	Voltaje [V]	PWM	Ciclo de Trabajo [%]	PIN
1	-6	49000	50	14
2	-6	49000	50	15
3	-6	32700	0	16,17
4	-6	32700	0	18,19

Tabla 4.7: Caracterización de los Motores a una salida de -6 [V].

4.2.1 Resultado obtenido en la señal PWM.

En las siguientes figuras, se obtienen referencias a las señales del ciclo de trabajo obtenidas en la lectura de un osciloscopio, cada imagen correspondiente a un ciclo de trabajo que necesitaremos en los controladores de motores que se especificaron en las tablas 6.1 a 6.7. En este caso la figura 4.2, corresponde a un ciclo de trabajo cercano al 0%, sin llegar a ser exactamente es ciclo de trabajo, la figura 4.3, corresponde a un ciclo de trabajo cercano a 25 % respectivamente, la figura 4.4 corresponde a un ciclo de trabajo cercano a 25 % respectivamente, la figura 4.5 corresponde a un ciclo de trabajo cercano a 75 % respectivamente, y finalmente la figura 4.6 corresponde a un ciclo de trabajo cercano a 100 % respectivamente, sin llegar a ser exactamente este ciclo de trabajo.

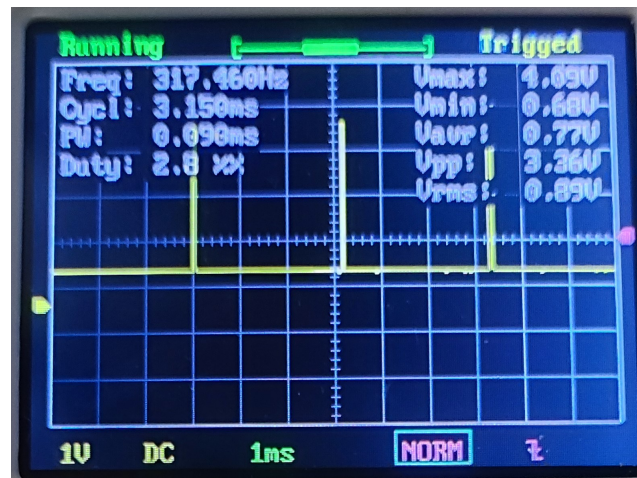


Figura 4.2: Ciclo del trabajo cercano al 0%.

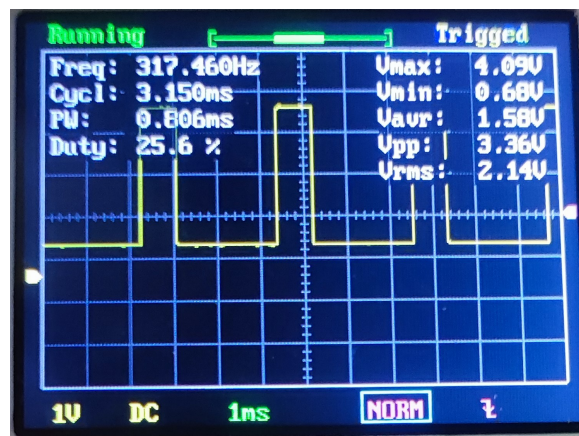


Figura 4.3: Ciclo del trabajo cercano al 25%.



Figura 4.4: Ciclo del trabajo cercano al 50%.

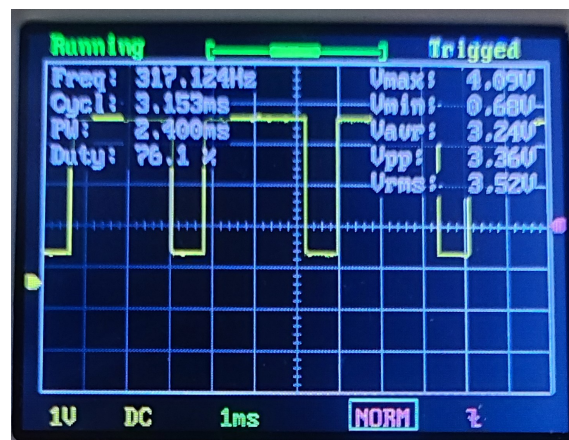


Figura 4.5: Ciclo del trabajo cercano al 75%.

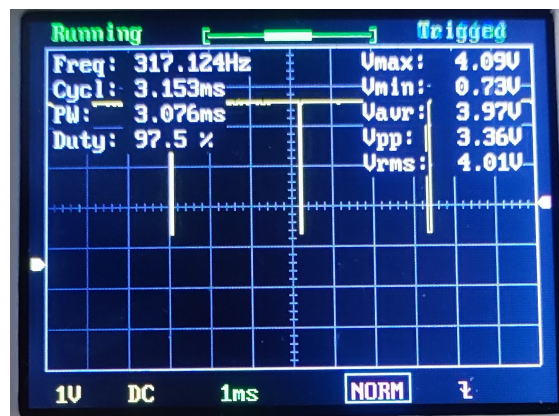


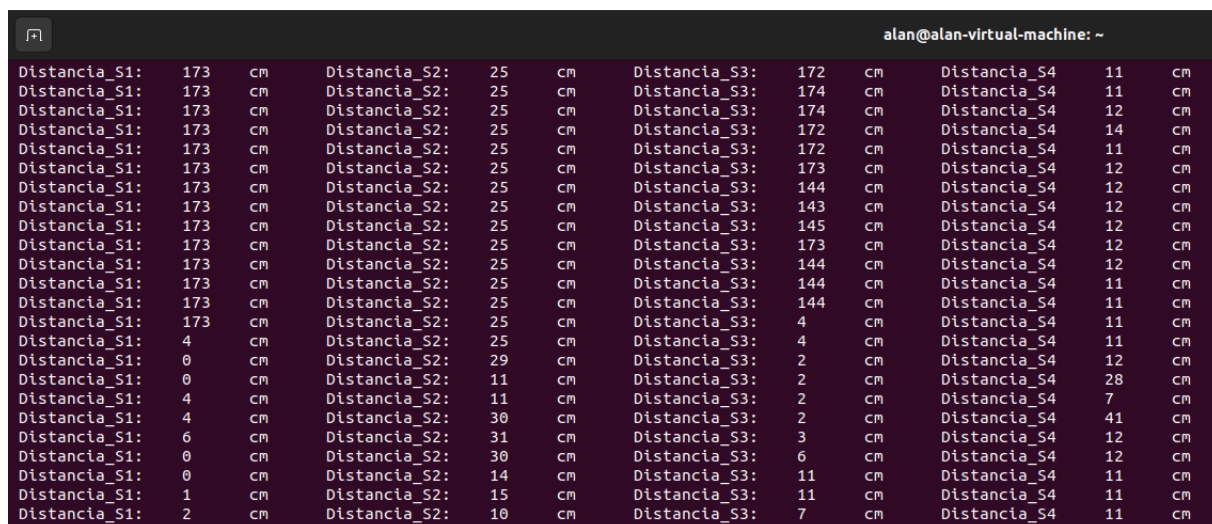
Figura 4.6: Ciclo del trabajo cercano al 100%.

Sección 4.3.

Resultado de la lectura en terminal.

A continuación se presentarán los resultados obtenidos en las lecturas de los sensores, desde la terminal de comandos, así como otros resultados que se observan.

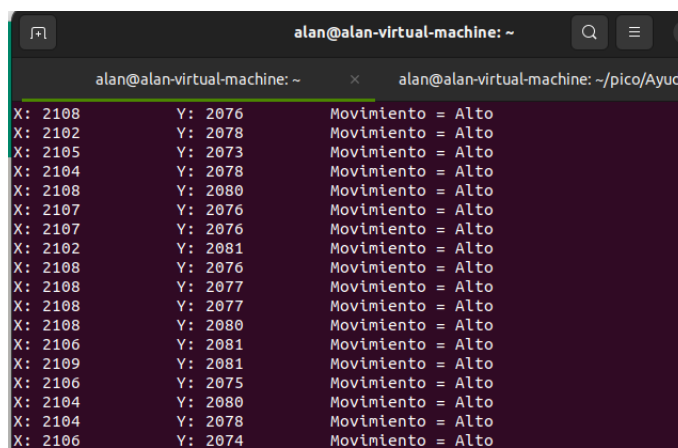
En la figura 4.7 se observan las lecturas de los sensores ultrasónicos tomadas desde el robot.



Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	172	cm	Distancia_S4:	11	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	174	cm	Distancia_S4:	11	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	174	cm	Distancia_S4:	12	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	172	cm	Distancia_S4:	14	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	172	cm	Distancia_S4:	11	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	173	cm	Distancia_S4:	12	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	144	cm	Distancia_S4:	12	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	143	cm	Distancia_S4:	12	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	145	cm	Distancia_S4:	12	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	173	cm	Distancia_S4:	12	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	144	cm	Distancia_S4:	12	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	144	cm	Distancia_S4:	11	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	144	cm	Distancia_S4:	11	cm
Distancia_S1:	173	cm	Distancia_S2:	25	cm	Distancia_S3:	4	cm	Distancia_S4:	11	cm
Distancia_S1:	4	cm	Distancia_S2:	25	cm	Distancia_S3:	4	cm	Distancia_S4:	11	cm
Distancia_S1:	0	cm	Distancia_S2:	29	cm	Distancia_S3:	2	cm	Distancia_S4:	12	cm
Distancia_S1:	0	cm	Distancia_S2:	11	cm	Distancia_S3:	2	cm	Distancia_S4:	28	cm
Distancia_S1:	4	cm	Distancia_S2:	11	cm	Distancia_S3:	2	cm	Distancia_S4:	7	cm
Distancia_S1:	4	cm	Distancia_S2:	30	cm	Distancia_S3:	2	cm	Distancia_S4:	41	cm
Distancia_S1:	6	cm	Distancia_S2:	31	cm	Distancia_S3:	3	cm	Distancia_S4:	12	cm
Distancia_S1:	0	cm	Distancia_S2:	30	cm	Distancia_S3:	6	cm	Distancia_S4:	12	cm
Distancia_S1:	0	cm	Distancia_S2:	14	cm	Distancia_S3:	11	cm	Distancia_S4:	11	cm
Distancia_S1:	1	cm	Distancia_S2:	15	cm	Distancia_S3:	11	cm	Distancia_S4:	11	cm
Distancia_S1:	2	cm	Distancia_S2:	10	cm	Distancia_S3:	7	cm	Distancia_S4:	11	cm

Figura 4.7: Lectura de las distancias de los 4 sensores de manera simultánea.

En la figura 4.8 se observan las lecturas analógicas del mando ky-023, y una selección de su movimiento, en este caso de alto.



X: 2108	Y: 2076	Movimiento = Alto
X: 2102	Y: 2078	Movimiento = Alto
X: 2105	Y: 2073	Movimiento = Alto
X: 2104	Y: 2078	Movimiento = Alto
X: 2108	Y: 2080	Movimiento = Alto
X: 2107	Y: 2076	Movimiento = Alto
X: 2107	Y: 2076	Movimiento = Alto
X: 2102	Y: 2081	Movimiento = Alto
X: 2108	Y: 2076	Movimiento = Alto
X: 2108	Y: 2077	Movimiento = Alto
X: 2108	Y: 2077	Movimiento = Alto
X: 2108	Y: 2080	Movimiento = Alto
X: 2106	Y: 2081	Movimiento = Alto
X: 2109	Y: 2081	Movimiento = Alto
X: 2106	Y: 2075	Movimiento = Alto
X: 2104	Y: 2080	Movimiento = Alto
X: 2104	Y: 2078	Movimiento = Alto
X: 2106	Y: 2074	Movimiento = Alto

Figura 4.8: Lectura del mando KY-023 y selección de movimiento en alto.

En la figura 4.9 se observan las lecturas analógicas del mando ky-023, y una selección de su movimiento, en este caso, en la selección de un movimiento hacia adelante.

```
alan@alan-virtual-machine: ~
X: 2805 Y: 4095 Movimiento = Adelante
X: 2805 Y: 4095 Movimiento = Adelante
X: 2803 Y: 4095 Movimiento = Adelante
X: 2803 Y: 4095 Movimiento = Adelante
X: 2801 Y: 4095 Movimiento = Adelante
X: 2808 Y: 4095 Movimiento = Adelante
X: 2802 Y: 4095 Movimiento = Adelante
X: 2802 Y: 4095 Movimiento = Adelante
X: 2805 Y: 4095 Movimiento = Adelante
X: 2802 Y: 4095 Movimiento = Adelante
X: 2795 Y: 4095 Movimiento = Adelante
X: 2798 Y: 4095 Movimiento = Adelante
X: 2800 Y: 4095 Movimiento = Adelante
X: 2804 Y: 4095 Movimiento = Adelante
X: 2803 Y: 4095 Movimiento = Adelante
X: 2802 Y: 4095 Movimiento = Adelante
X: 2805 Y: 4095 Movimiento = Adelante
X: 2798 Y: 4095 Movimiento = Adelante
```

Figura 4.9: Lectura del "mando KY-023" y selección de movimiento delantero.

En la figura 4.10 se observa el resultado de la terminal en las lectura del "mando ky-023", con adición a la impresión de fecha y hora del dispositivo "RTC DS3231".

```
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 42 cm Distancia_S2: 16 cm Distancia_S3: 23 cm Distancia_S4: 48 cm || Fecha: 2023/10/09 || Horario: 13:42:32 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 42 cm Distancia_S2: 16 cm Distancia_S3: 23 cm Distancia_S4: 49 cm || Fecha: 2023/10/09 || Horario: 13:42:32 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 42 cm Distancia_S2: 16 cm Distancia_S3: 23 cm Distancia_S4: 75 cm || Fecha: 2023/10/09 || Horario: 13:42:33 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 42 cm Distancia_S2: 16 cm Distancia_S3: 25 cm Distancia_S4: 145 cm || Fecha: 2023/10/09 || Horario: 13:42:33 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 43 cm Distancia_S2: 16 cm Distancia_S3: 21 cm Distancia_S4: 76 cm || Fecha: 2023/10/09 || Horario: 13:42:33 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 43 cm Distancia_S2: 16 cm Distancia_S3: 28 cm Distancia_S4: 75 cm || Fecha: 2023/10/09 || Horario: 13:42:34 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 42 cm Distancia_S2: 16 cm Distancia_S3: 17 cm Distancia_S4: 145 cm || Fecha: 2023/10/09 || Horario: 13:42:34 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 43 cm Distancia_S2: 16 cm Distancia_S3: 17 cm Distancia_S4: 75 cm || Fecha: 2023/10/09 || Horario: 13:42:34 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
Distancia_S1: 42 cm Distancia_S2: 16 cm Distancia_S3: 17 cm Distancia_S4: 74 cm || Fecha: 2023/10/09 || Horario: 13:42:35 || Dia: LUN ||
Hello RGB and other colors!
X: 4095 Y: 4095 Movimiento = ALTO
```

Figura 4.10: Lectura e impreseión del RTC DS3231.

En la figura 4.11 se observa que se establece la conexión entre el agente, es decir, el dispositivo *Raspberry Pi 4*, con el cliente o dispositivo *Raspberry Pi Pico*.

```

alan@alan-virtual-machine: ~
alan@alan-virtual-machine: ~
alan@alan-virtual-machine: ~
alan@alan-virtual-machine: ~/AY...
alan@alan-virtual-machine: ~/AY...
alan@alan-virtual-machine: ~/AY...

alan@alan-virtual-machine:~$ sudo snap set core experimental.hotplug=true
[sudo] password for alan:
alan@alan-virtual-machine:~$ sudo systemctl restart snapd
alan@alan-virtual-machine:~$ snap interface serial-port
name:    serial-port
summary: allows accessing a specific serial port
plugs:
  - micro-ros-agent
slots:
  - snapd:pico
alan@alan-virtual-machine:~$ snap connect micro-ros-agent:serial-port snapd:pico
alan@alan-virtual-machine:~$ sudo micro-ros-agent serial --dev /dev/ttyACM0 baudrate=115200
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
[1678391647.676568] info | TermiosAgentLinux.cpp | init | running... | fd: 3
[1678391647.677024] info | Root.cpp | set_verbose_level | logger setup | verbose_level: 4
[1678391648.176598] info | Root.cpp | create_client | create | client_key: 0x456F4541, session_id: 0x81
[1678391648.176689] info | SessionManager.hpp | establish_session | session established | client_key: 0x456F4541, address: 0
[1678391648.245073] info | ProxyClient.cpp | create_participant | participant created | client_key: 0x456F4541, participant_id: 0x000(1)
[1678391648.259482] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x456F4541, topic_id: 0x000(2), participant_id: 0x000(1)
[1678391648.273530] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x456F4541, publisher_id: 0x000(3), participant_id: 0x000(1)
[1678391648.294370] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x456F4541, datawriter_id: 0x000(5), publisher_id: 0x000(3)

```

Figura 4.11: Establecimiento de la conexión entre el agente y cliente de *micro-ROS*.

En la figura 4.12 se observan los tópicos de ROS que se pueden acceder desde la terminal con el comando " *ros2 topic echo*".

```

alan@alan-virtual-machine:~/AYUDAME-main$ ros2 topic echo
/back_left_wheel
/back_right_wheel
/client_count
/connected_clients
/front_left_wheel
/front_right_wheel
/parameter_events
/pico_publisher

```

Figura 4.12: Tópicos de conexión de *ROS*.

En la figura 4.13 se observa el mensaje de nula conexión entre el agente y cliente de *micro-ROS*, esto debido principalmente en que en algunas ocasiones se deberá reiniciar la conexión entre el cliente y el agente, debido principalmente a interrupciones que puede ocurrir al desconectar algún dispositivo.

```

alan@alan-virtual-machine:~$ snap connect micro-ros-agent:serial-port snapd:pico
alan@alan-virtual-machine:~$ sudo micro-ros-agent serial --dev /dev/ttyACM0 baudrate=115200
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
[1678511090.336360] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511091.339512] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511092.348378] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511093.352469] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511094.357060] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511095.359631] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511096.364939] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511097.376320] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511098.384960] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511099.388011] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...
[1678511100.389873] info | TermiosAgentLinux.cpp | init | Serial port not found. | device: /dev/ttyACM0, error 2, waiting for connection...

```

Figura 4.13: Fallo de conexión entre el agente y cliente de *micro-ROS*.

Sección 4.4.

Resultado de diseños 3D para componentes y chasis.

En esta breve sección recapitularemos sobre los diseños que se fueron utilizando sobre la marcha para ayudar al diseño de nuestro robot. Algunos diseños propios y otros tomados por parte de una comunidad de libre acceso para diferentes usuarios según las necesidades requeridas, en nuestro caso fueron tomados para facilitar el ordenamiento de los componentes electrónicos en el chasis del robot que se mostraran en el resultado del robot.

4.4.1 Diseños propios.

En la figura 4.14 se observa el diseño de la base que sostendrá a los sensores ultrasónicos para que se encuentren fijados al robot, en esta primera propuesta solo buscamos un resultado que nos permitiera tomar algunas lecturas con los sensores, por lo que el propósito no es un resultado para la presentación del robot, si no más bien uno que nos permita facilitar las lecturas de los sensores de distancia o sensores ultrasónicos "US-100".

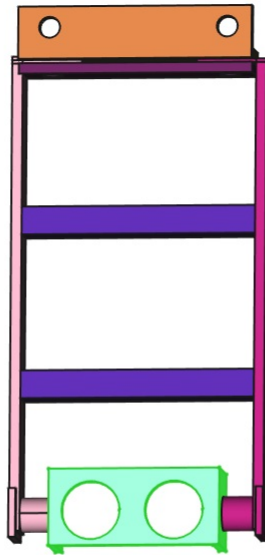


Figura 4.14: Diseño de soporte a chasis para los sensores de distancia "US-100" que utilizará el robot para lecturas de proximidad de distancia.

El diseño que se muestra en la figura 4.15, cumple la función similar al de una canaleta, la cual para pasar el cable y poder fijar los nodos de conexión entre diferentes cables, principalmente de la alimentación de la batería y los controladores para motores de corriente directa, así como de las zapatas de tipo ojillo para hacer la conexión de los nodos en general del sistema de batería, motores y alimentación de la PCB realizada, con el propósito de tener un orden para fijar y retirar estos nodos en caso de requerir cambios posteriores.

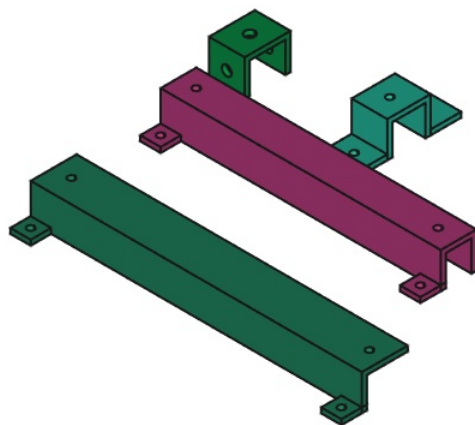


Figura 4.15: Nodos de conexión para los cables utilizados en las conexiones del robot omnidireccional.

4.4.2 Diseños de la comunidad de Thingiverse.

El objetivo de implementar diseños realizados por la comunidad de thingiverse es básicamente en reducir el tiempo de diseño para diferentes proyectos, por lo que en este sitio web y comunidad ya se encuentran realizados múltiples diseños realizados por una gran comunidad de pasión por el diseño 3D que brindan de manera gratuita sus diseños para algunos componentes electrónicos que resultan convenientes de utilizar, y si bien tienen un propósito diferentes se pueden utilizar de una manera útil en la implementación de nuestro robot omnidireccional, sin fines de lucro y dando crédito a los contribuidores.

El primer diseño que se muestra en la figura 4.16, es básicamente un soporte de la cámara "intel realsense D455", que será la encargada de procesar la navegación autónoma con la ayuda de "Slam - ROS", encargada por el equipo de "Ayudame Navegación".



Figura 4.16: Soporte para la camara Intel Realsense D455.
Este diseño fue realizado por el usuario Bradley BeMent y se encuentra disponible en el siguiente URL: "<https://www.thingiverse.com/vatbrain/designs>".

El siguiente diseño que se observa en la figura 4.17 corresponde a un sujetador para nuestro sensor de corriente "ACS712" que facilitará fijarla al chasis del robot, y tener una conexión para la lectura y los pines que irán conectados a la PCB diseñada para sus lecturás tomadas.

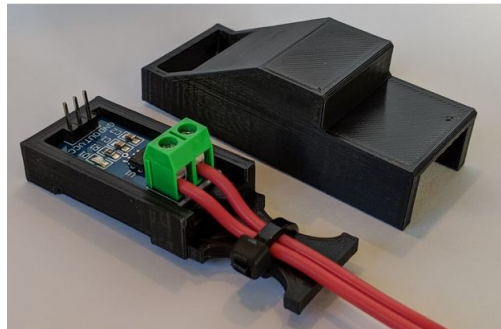


Figura 4.17: Versión preliminar de soporte para el sensor de corriente ACS712.
Este diseño realizado por el usuario "Rego Maclean" y se encuentra disponible en el siguiente URL: "<https://www.thingiverse.com/humble0/designs>".

El siguiente diseño mostrado en la figura 4.18 corresponde a una caja para l pantalla LCD16x2, encargado de ser una caja protectora y como un soporte que será ajustable a nuestro robot sobre un acrílico en la parte trasera del chasis del robot.

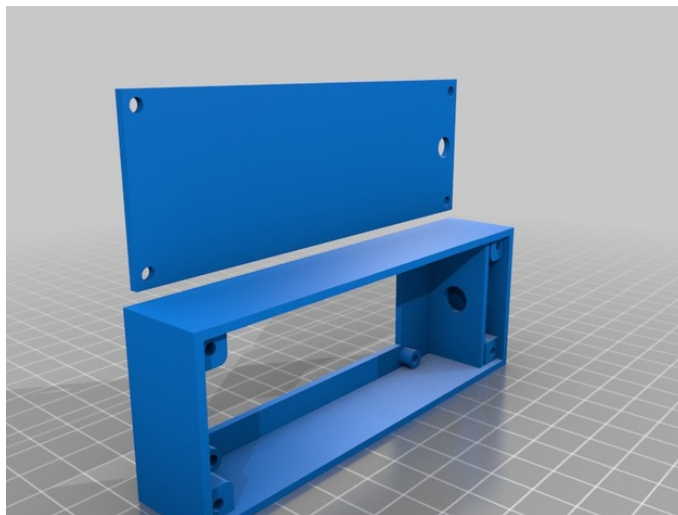


Figura 4.18: Diseño preliminar de una caja protectora para la pantalla LCD 16x2. Este diseño fue realizado por el usuario "Martin Pihrt", disponible en la siguiente URL: "<https://www.thingiverse.com/pihrt/designs>".

El siguiente diseño mostrado en la figura 4.19 corresponde a un voltímetro y amperímetro que será utilizado principalmente para tomar lectura de la corriente y voltaje del robot y también en el nivel de la carga de la batería en el sistema. Este diseño facilitará el fijar este componente en el chasis de nuestro robot.



Figura 4.19: Visión preliminar de una caja protectora para un voltímetro amperímetro. Este diseño fue realizado por el usuario Marco EmmeTi, diseño disponible en la URL: "<https://www.thingiverse.com/marcomt/designs>".

El siguiente diseño mostrado en la figura 4.20 corresponde a la base para el mando "KY-023", que será implementado como propuesta de control manual de nuestro robot omnidireccional, de manera adicional incluye opciones que se pueden realizar con interruptores para otros usos.



Figura 4.20: Diseño preliminar de mando para un único mando analógico con el componente "KY-023".

Este diseño fue realizado por el usuario "Clo V15", y esta disponible en la siguiente URL:
"<https://www.thingiverse.com/cl0/designs>".

Sección 4.5.

Resultados del robot.

En esta sección veremos el proceso y avance que se fue logrando con el paso del desarrollo del robot omnidireccional "Ayudame 2.0", donde las figuras del desarrollo del robot en la implementación electrónica del chasis, como se acomodó el tablero electrónico y los diseños realizados por medio de impresión 3D, que se fue desarrollando a lo largo del proyecto en las diferentes etapas y con los principales cambios realizados. En la siguiente figura se observa el sensor ultrasónico como componente utilizado en el robot.

En la figura 4.21, se observa el sensor de distancia "US-100", en un fondo el chasis del robot.

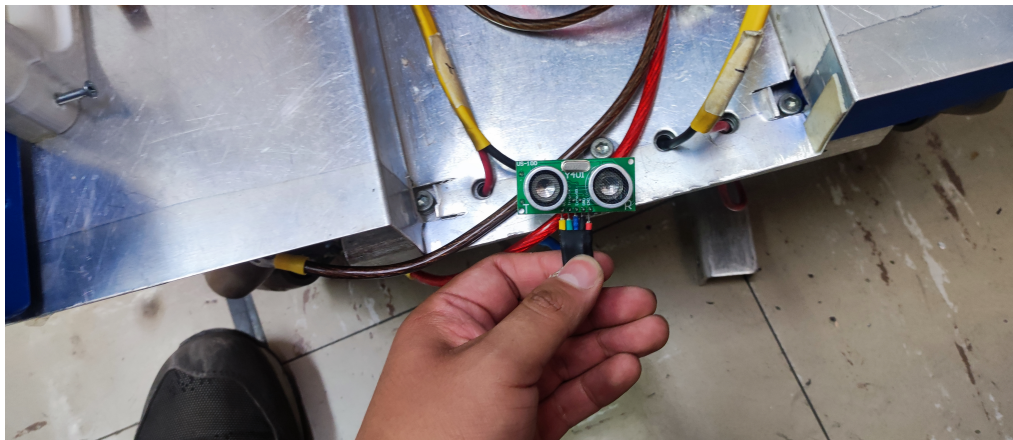


Figura 4.21: Sensor ultrasónico en la parte inferior del chasis del robot.

En la figura 4.22, se observa una versión preliminar sobre cómo sería el espacio del chasis, junto a una batería de 12 [V] de corriente directa, donde se realizaron primeras pruebas con los controladores de motores modelo Talon SR.

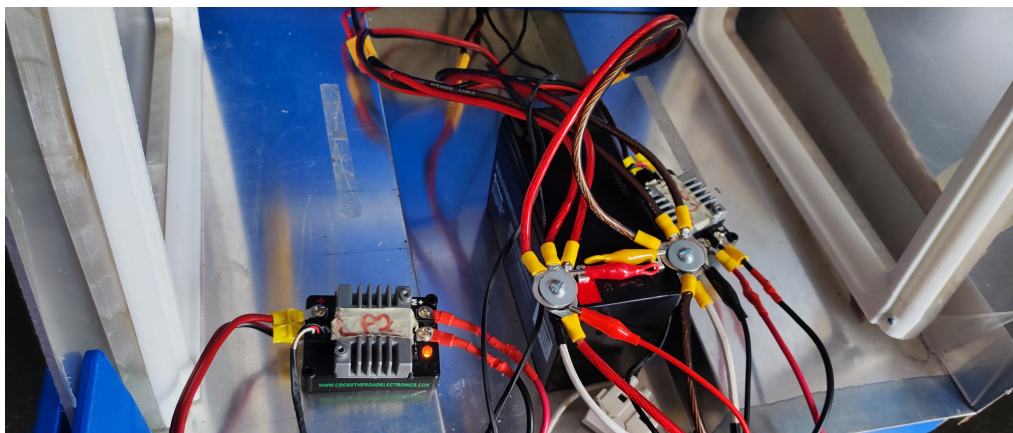


Figura 4.22: Batería de 12 [V], junto a los controladores de motores de corriente directa para pruebas preliminares.

En la siguiente figura 4.23 se observa un estado del robot en desorden de diferentes componentes y herramientas, más, sin embargo, se consideran los prospectos dimensiones y consideraciones de los cables que estarán destinados al uso de los sensores, el tipo de calibre que habíamos considerado anteriormente, ya que fue un constante reto considerar y proponer como ordenar los componentes electrónicos considerando posibles cambios a futuro.



Figura 4.23: Estado del robot en desorden de componentes y herramientas.

En la figura 4.24 se observa la preparación para la planeación de la colocación de la PCB o el circuito impreso que será el principal centro de conexión para los sensores y los controladores de motores de corriente directa, colocados sobre la parte trasera del robot, ya que se considerara esta sección para colocar la principal electrónica del robot debido a la accesibilidad y futuros cambios entre usuarios.



Figura 4.24: Parte trasera del chasis del robot.

En la figura 4.25 se observa que se ha colocado un soporte encargado de sostener la batería de ácido plomo de 12 [V].

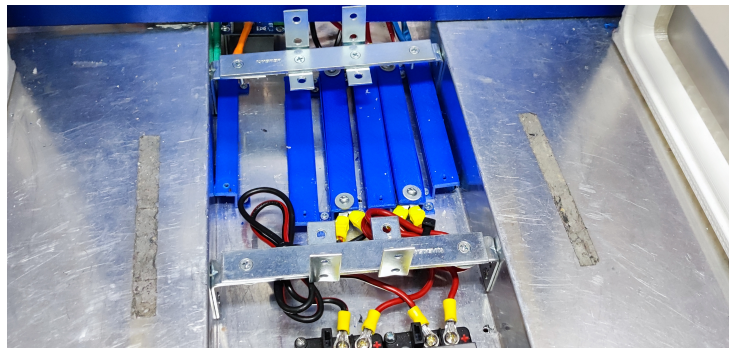


Figura 4.25: Base para soportar la batería de 12 [V].

En la figura 4.26 se observa la parte frontal del chasis del robot, en una sección que se observa más despejada. Aquí se tomó la decisión de que serían colocados los controladores de los motores de corriente directa del robot, principalmente a la accesibilidad para poder sustituir o realizar algún cambio posterior.

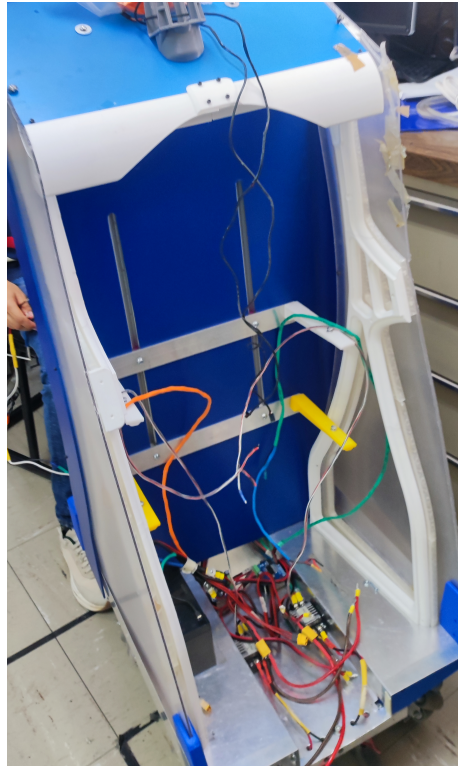


Figura 4.26: Parte frontal del robot con controladores de motores de corriente directa.

En la figura 4.27 se observa el diseño de la base que se colocó sobre el chasis para colocar la batería en el centro del robot, que será la principal fuente de alimentación de nuestro robot que le dará energía a los componentes electrónicos y los motores de nuestro robot omnidireccional.



Figura 4.27: Base de la batería de ácido plomo de 12 [V].

En la figura 4.28 se observa la cámara de Intel Realsense lidar D455 fijada sobre una altura considerable del robot, que será principalmente utilizada para el algoritmo de navegación con *SLAM-ROS*, encargada principalmente por el equipo de "Ayudame 2.0 Navegación"



Figura 4.28: Robot con cámara intel realsense D455 fijada al robot.

En la figura 4.29 se observa que sobre la consideración de colocar un improvisado tablero de la electrónica del robot resulta conveniente colocarla en la parte trasera del robot omnidireccional por términos de espacio y accesibilidad para cambios posteriores, como se había planteado ya anteriormente, siendo colocada sobre una superficie de acrílico, este sería nuestro primer punto de partida del avance en esta sección.



Figura 4.29: Configuración electrónica sobre el chasis en su primer avance.

En la figura 4.30 se observa que sobre esta misma sección fuimos progresando la configuración electrónica, ya que resultaba factible continuar en esta sección, considerando en esta ocasión una mejor distribución entre los cables y componentes electrónicos del robot, así como un interruptor general.



Figura 4.30: Segundo avance sobre la configuración electrónica sobre el chasis.

En la figura 4.31 se observa un mayor avance sobre esta sección, ya planificando como serían colocados algunos componentes electrónicos y el uso posterior que tendrán, así como las conexiones de los cables, controladores de motores de corriente directa y cables de alimentación.

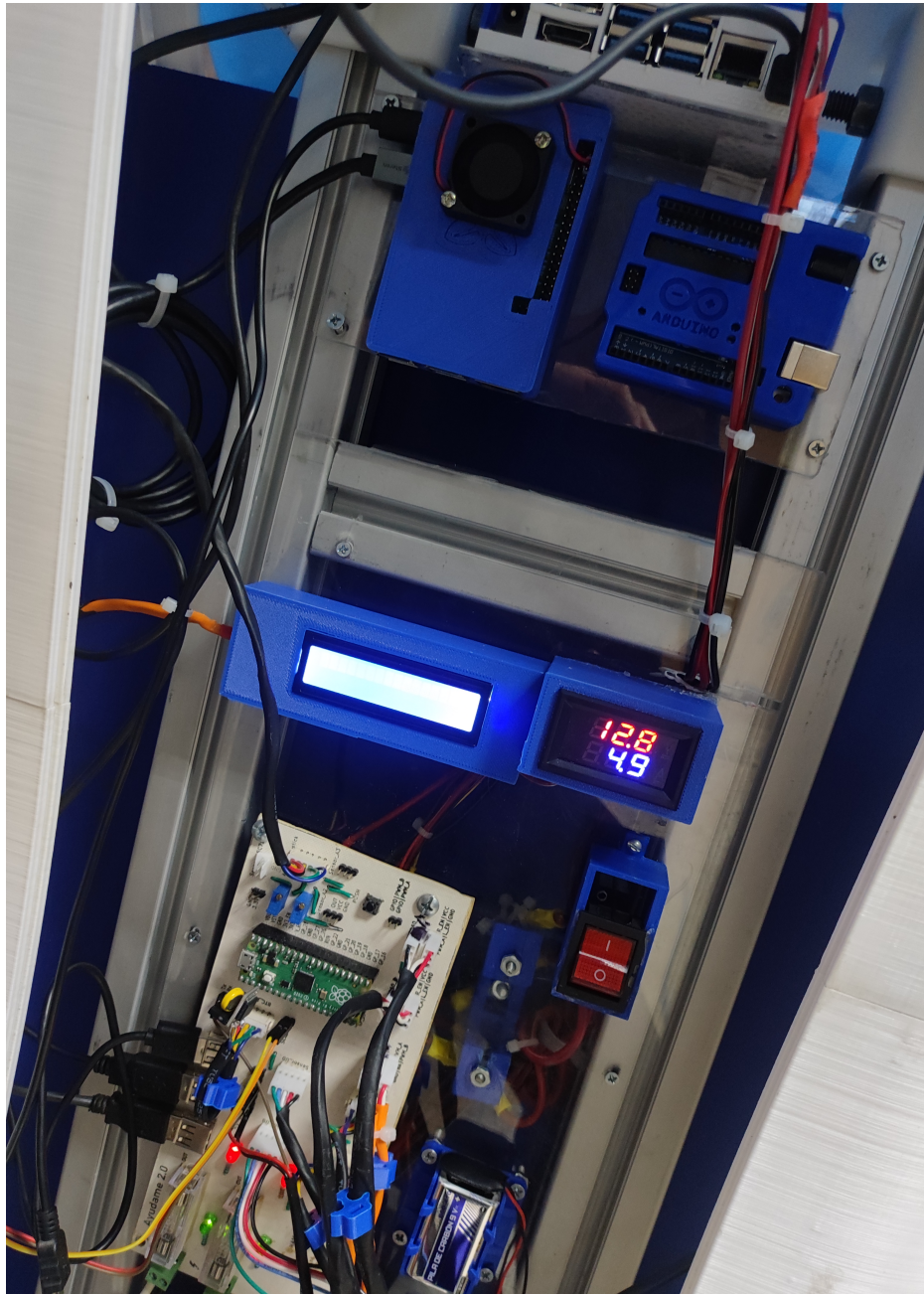


Figura 4.31: Tercer avance sobre la sección de la parte trasera de la configuración electrónica del chasis.

En la figura 4.32 se observa la batería colocada sobre la base previamente realizada para que se colocara sobre este punto.



Figura 4.32: Batería colocada sobre su base para el chasis del robot.

En la figura 4.33 se observa el diseño de la base para los sensores ultrasónicos "US-100" que serán colocados sobre el chasis del robot para las lecturas de distancia de proximidad del robot.

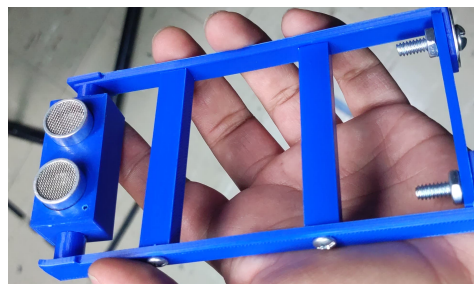


Figura 4.33: Diseño de los soportes de los sensores para los sensores ultrasónicos "US-100".

En la figura 4.34 se observa la parte trasera del robot con la configuración electrónica fijada sobre un acrílico en la parte del chasis del robot considerando también la conexión y componentes del robot.



Figura 4.34: Conexiones sobre la parte trasera del robot.

En la figura 4.35 se observa la configuración de la pantalla táctil del dispositivo *raspberrypi 4*, y detrás se observa el botón de paro de emergencia.



Figura 4.35: Botón de paro de emergencia del robot y pantalla táctil.

En la figura 4.36 se observa la pantalla LCD 16x2 y el voltímetro amperímetro digital con una lectura de voltaje en la batería y corriente de la carga hasta antes de colocar completamente los dispositivos electrónicos.

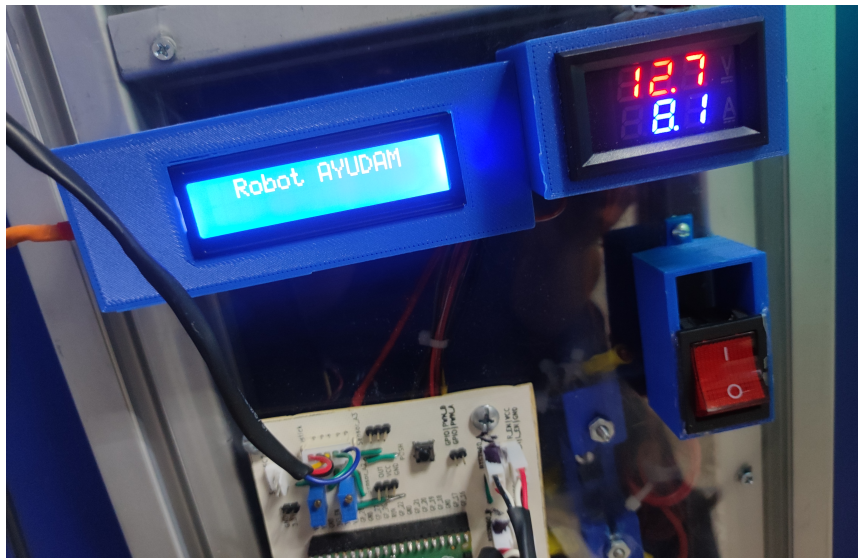


Figura 4.36: Pantalla LCD del tablero.

En la figura 4.37 se observa la una de las fases finales de la propuesta del diseño del robot desde una vista lateral del robot omnidireccional en un estado más avanzado que al comienzo de la implementación electrónica del tren motriz, donde ya se realizaron algunas pruebas de rendimiento y sincronización en los motores funcionando, aun sin tener movimiento.

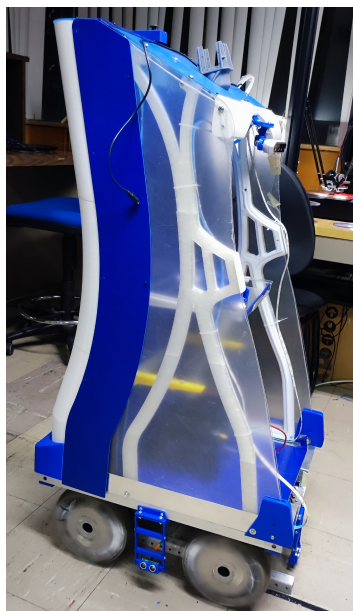


Figura 4.37: Vista lateral del robot en fase avanzada.

En la figura 4.38 se observa el robot con la pantalla táctil colocada, con una de las propuestas de animación al robot encargadas por el equipo "Ayudame 2.0 Animaciones", con el objetivo de tener la apariencia de un asistente.

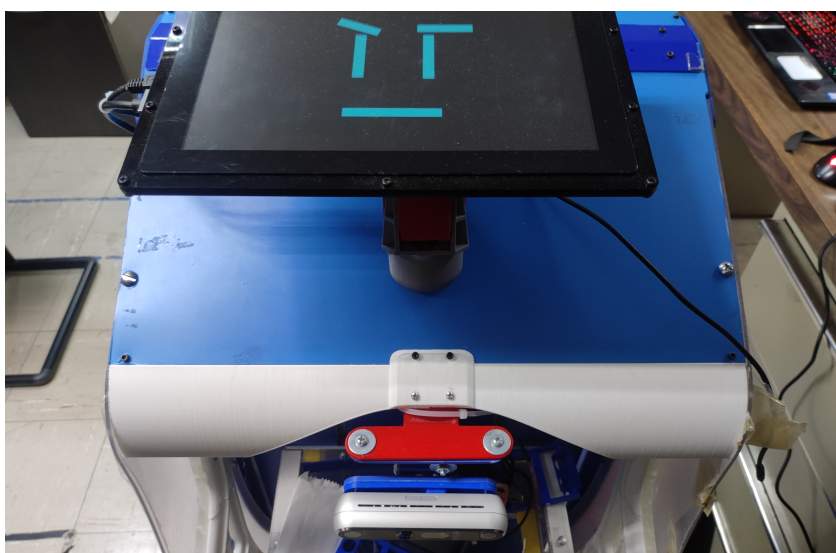


Figura 4.38: Vista de animación en la pantalla táctil del robot.

En la figura 4.39 se observa el resultado de la fijación de los componentes electrónicos especificados, con todas las consideraciones y componentes electrónicos situados, aun con accesibilidad a mejoras y cambios futuros.



Figura 4.39: Propuesta de tablero de configuración electrónica final con prospectos de cambios del robot Ayudame 2.0.

En la figura 4.40 se observa otra perspectiva de la propuesta en el tablero electrónico para la configuración electrónica, sujeta a cambios ante el claro problema entre el orden de exceso de cable de los componentes electrónicos.

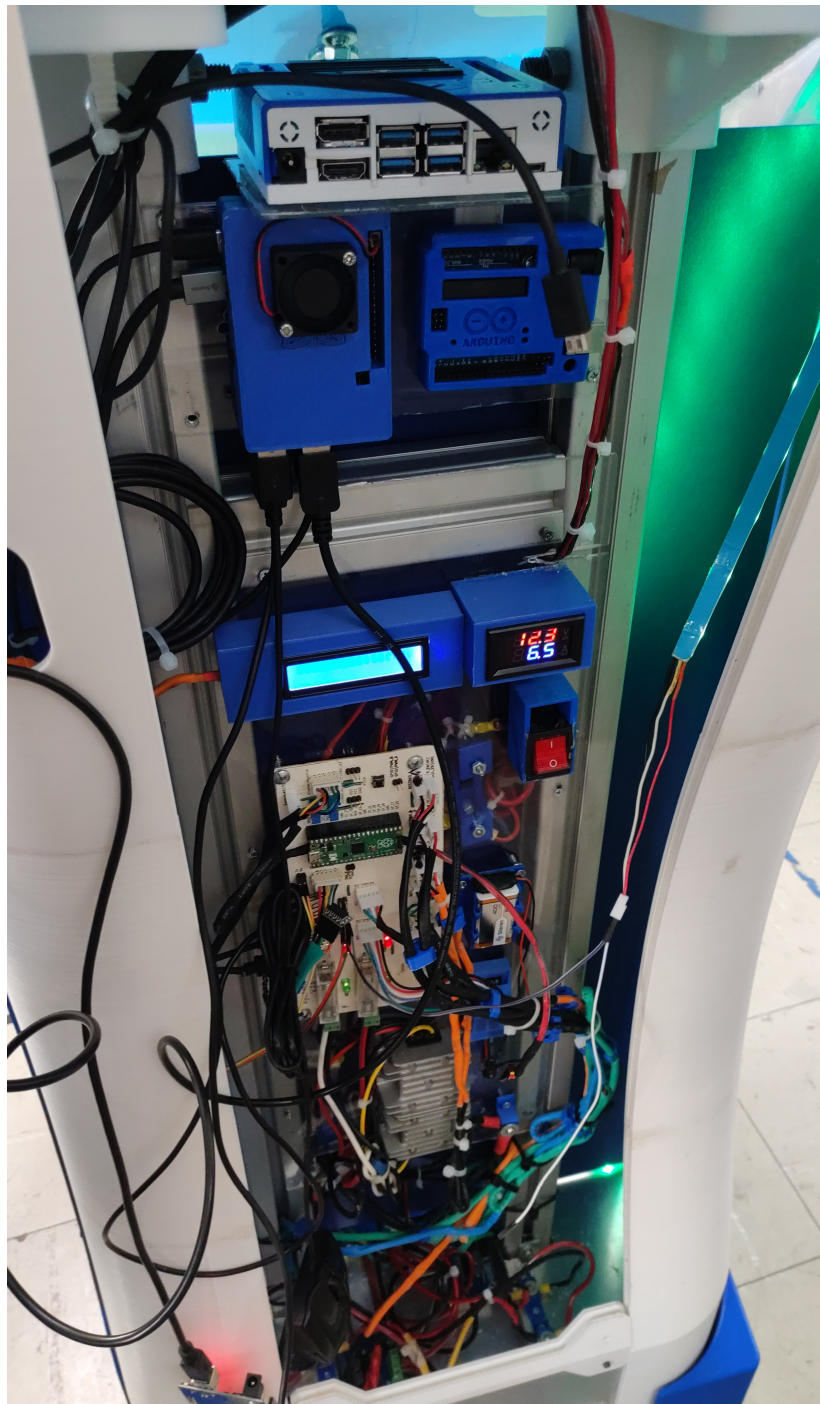


Figura 4.40: Conexión electrónica en la propuesta final al desarrollo de esta tesis para tablero del robot asistente hospitalario "Ayudame 2.0".

En la figura 4.41 se observa una vista general de la propuesta del robot de servicio en entornos hospitalarios, con un mayor orden y los componentes electrónicos fijados que se utilizaron para las pruebas de navegación en conjunto con el equipo de "Ayudame 2.0 Navegación", así como se observa un mayor orden y presentación del robot.

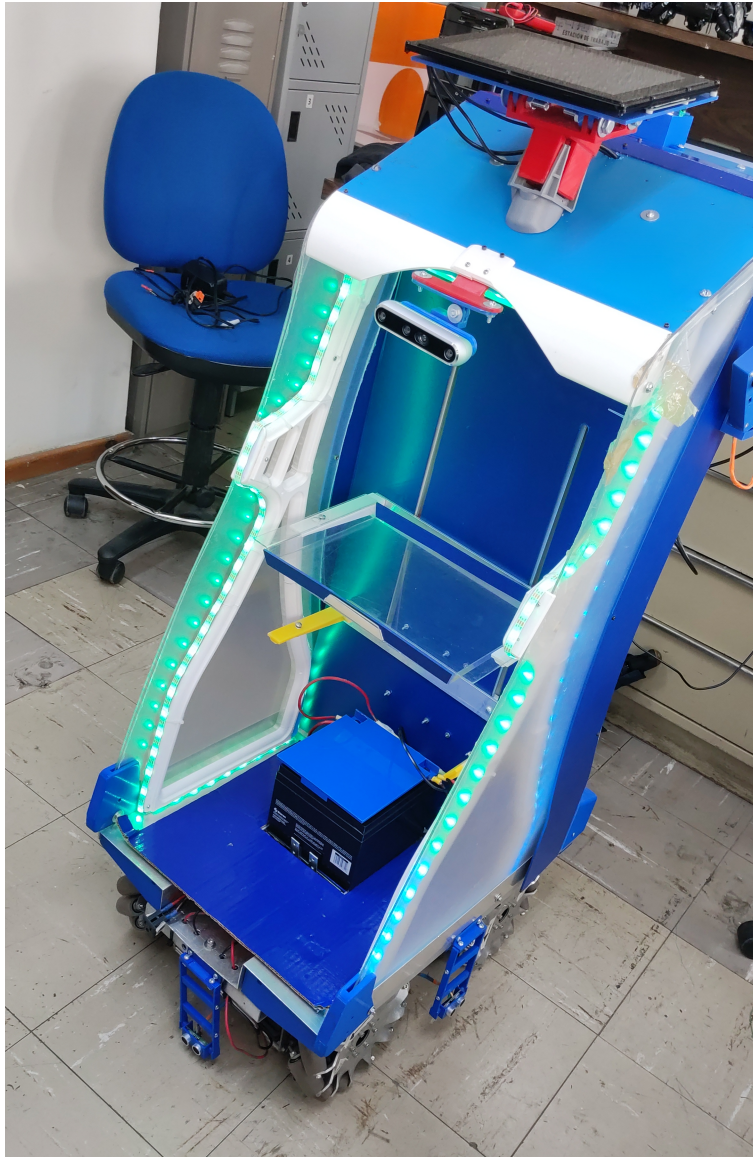


Figura 4.41: Propuesta final al desarrollo de la tesis del robot omnidireccional asistente en entornos hospitalarios..

En la figura 4.42 se observa otra perspectiva del robot asistente hospitalario con en una de las fases de propuesta finales sobre el diseño del tren motriz con el chasis y los componentes electrónicos ordenados y al resultado que se logró en el desarrollo de esta tesis.



Figura 4.42: Vista frontal final del robot omnidireccional asistente de entornos hospitalarios.

CONCLUSIONES

Se logró implementar un tren motriz para nuestro robot omnidireccional, tomando en consideración el trabajo realizado por el equipo "Ayudame 2.0" de manera general, donde logramos adaptar el trabajo en conjunto de diseño, que realizó el chasis para adaptar la configuración electrónica con los componentes propuestos y comentados durante todo el proceso de esta tesis. En algunos momentos fue necesario adaptar algunos cambios inmediatos de diseño y propuestas para realizar algunas pruebas en momentos específicos, siempre escuchando diferentes propuestas de todos los miembros que estuvieron implicados en el proyecto de manera general. También tomamos en consideración las normativas oficiales mexicanas vigentes para que nuestro robot cumpla con los lineamientos de seguridad que son considerados dentro de las instalaciones hospitalarias para un funcionamiento adecuado sin la necesidad de incumplir o provocar situaciones de riesgo, evitando que pueda obstruir señalizaciones y salidas de emergencia, así como obstruir pasillos.

No obstante, cabe recalcar que debido a que la llegada de la robótica móvil o de asistencia en hospitales aún va en crecimiento, no hay una normativa o consideraciones a seguir al pie de la letra o ya establecidas dentro de un manual o guía, por lo que muchas de las consideraciones tomadas en cuenta fueron propuestas al momento de adaptar el robot para solucionar algunos de los retos y consideraciones que tomamos en cuenta durante el desarrollo de estas tesis, por lo que gran parte de los procedimientos y trabajos realizados para la adaptación de este robot quedan abiertas a futuros cambios y mejoras.

Por ejemplo, la parte donde se colocó la electrónica principal, puede ser adaptada a un tablero con mayor protección contra las vibraciones de las ruedas, también consideración de seguridad ante posibles robos, debido principalmente a que los componentes tienen valor económico, por ejemplo el dispositivo Jetson Nano y la cámara de Intel RealSense Lidar tienen un valor de costo considerable, así como la batería de ácido plomo. En el estado actual del robot, los componentes anteriormente mencionados pueden ser fácilmente desmontables, por lo que es necesaria una mayor seguridad en estos componentes electrónicos.

Otra de las observaciones tomadas en cuenta al momento de realizar las pruebas de navegación del robot, estaba en el ruido producido por las ruedas omnidireccionales, las cuales no pasaban desapercibidas, y no consideramos adecuadas para encontrarse en un entorno hospitalario, por lo que hay posibles cambios para este tren motriz que pueden ser considerados en un futuro.

Uno de los principales retos de este proyecto fue la coordinación entre los diferentes equipos involucrados, navegación, diseño y control, ya que se presentaron en las propuestas de desarrollo para las diferentes versiones de *ROS*, por ejemplo, diferencias entre el uso de *ROS 2* y *ROS 1*, entre distintos puntos de vista que se ofrecían en cada versión y en ocasiones la coordinación entre pruebas requería de la disponibilidad de los involucrados en el proyecto, ocasionando una dificultad para realizar las diferentes pruebas requeridas o el cumplimiento en tiempo y formas para debatir los avances y siguientes pasos a tomar en cuenta.

Algunas propuestas diseños, como por ejemplo el diseño de la PCB y los materiales conseguidos en cortos tiempos para no postergar las entregas o por recorte de presupuesto, optamos en diversas ocasiones por contribuir propiamente con nuestro presupuesto o esperar en tiempo de espera larga para minimizar el tiempo y recursos del robot, los cuales pueden ser implementados en materiales de mejor calidad o dedicar más tiempo en el diseño de las decisiones del chasis o cambió en componentes electrónicos, adicionalmente, muchos de los diseños realizados en impresión 3D, son libres para el uso de los usuarios, sin embargo, algunos de los diseños propios solo fueron diseñados para ser funcionales y cumplir con las lecturas en las pruebas realizadas, pueden ser mejorados para ser no solo en apariencia mejor, sino también en funcionalidad, como por ejemplo los modelos utilizados para fijar los sensores de distancia, o el modelo para el mando "ky-023" y la cámara de "intel realsense".

La información sobre las principales consideraciones que se tienen a propuestas a futuro se puede resumir en la siguiente tabla 5.1, tomando en consideración que también se pueda acceder a un mayor tiempo de inversión al proyecto y presupuesto que pueda facilitar el desarrollo del robot.

Propuesta	Descripción
Cambió de material en el chasis del robot de impresión 3D	Implementar un chasis a futuro que sea más eficiente en la limpieza y menos reactivo con el entorno, siendo menos vulnerable a retener patógenos contagiosos y sea de fácil limpieza, por ejemplo, acero inoxidable
Diseño Electrónico	La primera versión de la PCB contemplaba de manera poco intuitiva las configuraciones de las conexiones con los componentes colocados, por lo que en primera instancia al momento de realizar la primera revisión de la PCB no había una revisión en cuanto a calidad y dimensión de las revisiones de la configuración electrónica, por lo que consideraría un cambio en la configuración del tamaño de pistas, dimensiones de la placa, referencia de tierra sobre la placa de cobre entre búsquedas de reducción de ruido eléctrico de la configuración electrónica.
Consideración de componentes electrónicos de mejor calidad	Consideración de adquirir mejores conectores para la configuración electrónica, como conectores JST de los sensores ultrasónicos, o cambiar la configuración de zapatas por conexiones de borneras.
Diseño de un mejor tablero electrónico	Diseñar una propuesta de mayor seguridad para la protección de los componentes electrónicos y para organizar mejor la configuración de conexión en los cables electrónicos.
Mejora en optimización del algoritmo de control	Buscar mejoras a solución de problemas continuos en el algoritmo de navegación y comunicación con los sensores y actuadores implementados para la mejora en tiempos de respuesta del robot.

Tabla 5.1: Resumen de mejoras en el robot.

En retrospectiva, la realización de esta tesis propone una forma de optimización y asistencia a personal médico dentro de los hospitales en la realización de diferentes tareas de asistencia en transporte de insumos, carga de material médico bajo ciertas normas de higiene y control de velocidad, no obstante, las aplicaciones que pueden implementarse en un futuro pueden ir más allá de estas tareas, y cumplir otras formas de optimización, por ejemplo en la asistencia de toma de signos vitales, estatura, temperatura, así como enviar un previo registro y enviar las lecturas tomadas a una base de datos con la información de los pacientes para facilitar la atención médica. Adicionalmente, también se puede dar mayor aprovechamiento a la pantalla para además de la realización de gestos poder tener una interfaz que ayude al personal médico a llevar un mejor control sobre los pacientes identificando las camas, habitaciones y pacientes que estén llevando a cabo para ofrecer asistencia en el cálculo de suministro de ciertos medicamentos específicos, las dosis y horarios asignados por medio de una interfaz sencilla y accesible.

No obstante, durante el desarrollo en general de este proyecto, se planteó la idea de hacer pruebas en un entorno hospitalario real, por lo que al investigar las posibilidades de llevar esta tarea a cabo en instalaciones hospitalarias, es necesario contar con algún responsable legal y una planificación de múltiples eventos para que se pueda llevar a cabo esta tarea, por lo que no es imposible implementar este proyecto en un entorno real, sin embargo, se requiere un mayor avance y planificación por parte de los responsables del desarrollo en el proyecto, por lo que puede tener un mayor alcance para seguir mejorando en las condiciones del desarrollo de este proyecto, así como la opinión de las personas que se encargan de laborar en este sector, desde el personal médico, hasta responsables de la seguridad dentro de las instalaciones hospitalarias correspondientes, independientemente de ser instituciones públicas o privadas, y que estas cuenten con las aprobaciones de las normativas mexicanas oficiales.

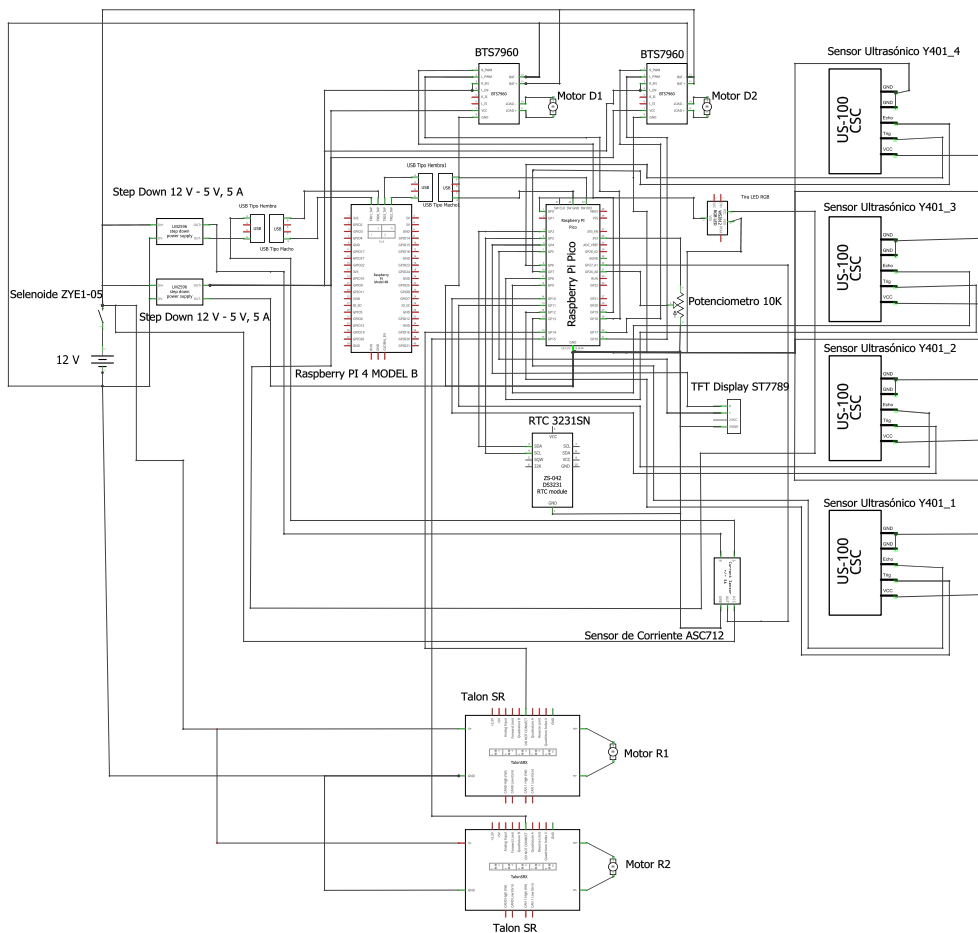
Personalmente, considero que la realización de esta tesis es un pequeño paso en un gran aporte para futuras aplicaciones de la robótica, principalmente debido al continuo avance de la tecnología y nuevas patentes, será en un futuro más fácil optimizar procesos y para afrontar futuros problemas sanitarios y de salud, tomando en cuenta lo visto durante la pandemia mundial del covid-19, donde la interacción del personal médico con los pacientes los colocaba en una mayor exposición en riesgos de contagio y propagación, por lo que en un futuro será más fácil buscar soluciones más próximas y salvaguardar el cuidado entre personal hospitalario y pacientes, sentando una base de referencia que pueda ser tomada como referencia para realizar prototipos similares que puedan funcionar en beneficio de la sociedad.

Finalmente, debo agradecer al Instituto de Matemáticas Aplicadas y en Sistemas, y a la Universidad Nacional Autónoma de México por darme la oportunidad de realizar mis prácticas profesionales en el desarrollo de este proyecto en conjunto con mi asesor el Dr. Victor Manuel Lomas Barrié, participando en el desarrollo del tren motriz de este proyecto que posteriormente permitió el desarrollo de la presente tesis, compartiendo ideas y propuestas que pueden ser llevadas a cabo en aplicaciones reales de ingeniería no solo por el deseo de una retribución monetaria o de fama, sino principalmente por lo el deseo de seguir contribuyendo a los avances tecnológicos, y fomentando el desarrollo tecnológico para que nuevas generaciones tomen mayor participación y se continúe con el desarrollo no solo en este proyecto, ya que el Instituto de matemáticas aplicadas y en sistemas puede fomentar el crecimiento profesional de académicos y alumnos que buscan impulsar múltiples proyectos como el proyecto "Ayudame 2.0", el cual con mayor participación, organización y presupuesto puede llegar a pasar de un prototipo de prueba a un robot de aplicación en un entorno hospitalario real para descubrir y proponer las normativas de un robot de servicio hospitalario, mejorando de manera constante y cumplir el objetivo de ser un beneficio a la sociedad cumpliendo los objetivos propuestos al principio de esta tesis.

Apéndice A. Planos realizados

Sección 6.1.

Plano del Diseño esquemático de la conexión de los componentes.



fritzing

Figura 6.1: Diseño esquemático de la conexión física de los componentes electrónicos.

Plano del Diseño esquemático general del robot.



Tabla de Símbolos	
Símbolo	Descripción
	Fuente de alimentación con polarización de corriente directa
	Corriente directa
	Tierra de la batería de alimentación 12 V
	Tierra de componentes
	Tierra de batería de alimentación de 9 V
	Descripción
	Diodo led
	Botón de paro
	Interruptor
	Conector tipo USB
	Fusible de norma americana
	Voltímetro y Amperímetro Digital

Figura 6.3: Tabla de simbología de la figura 8.2.

Sección 6.3.

Plano del Diseño de la PCB.

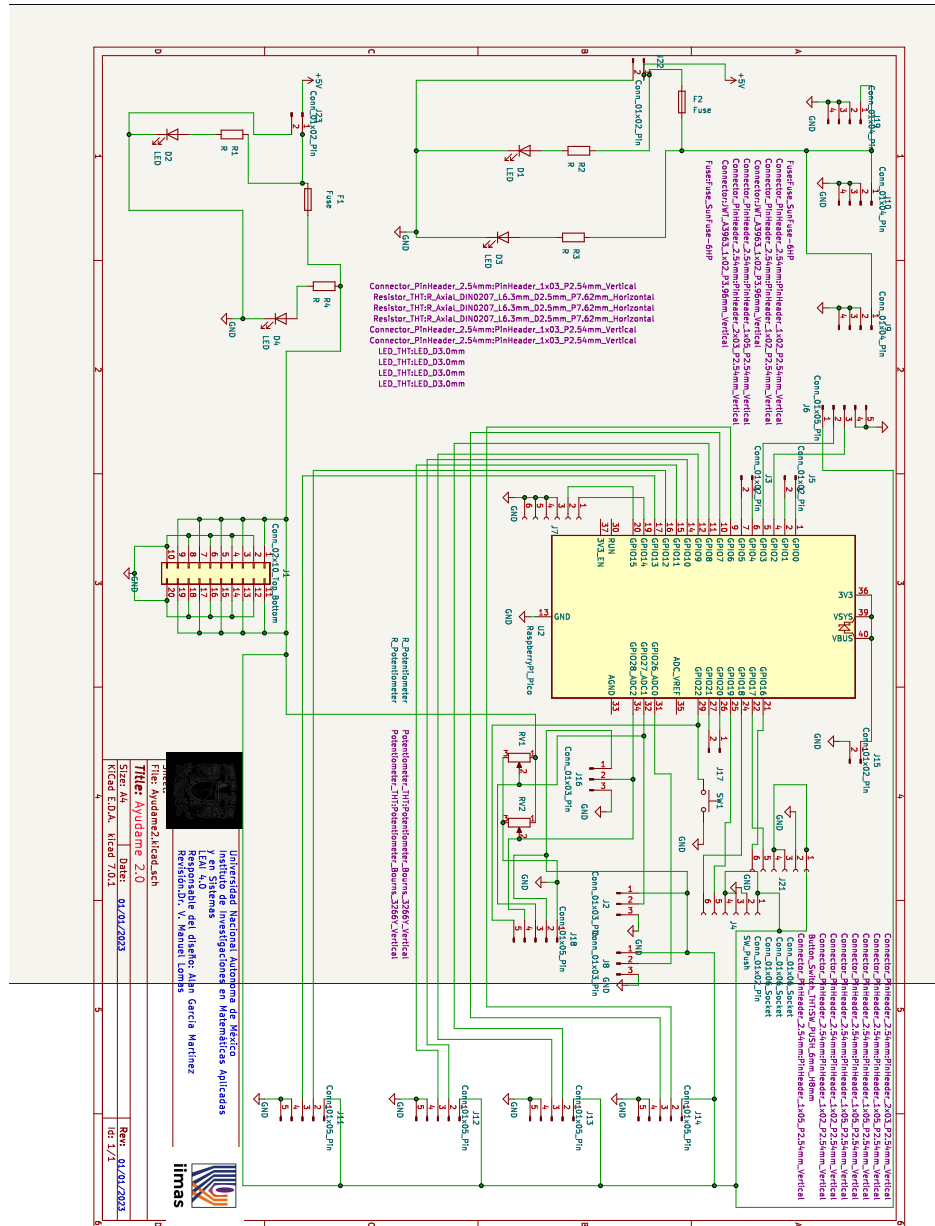


Figura 6.4: Diseño esquemático de la conexión física de los componentes electrónicos para la PCB.

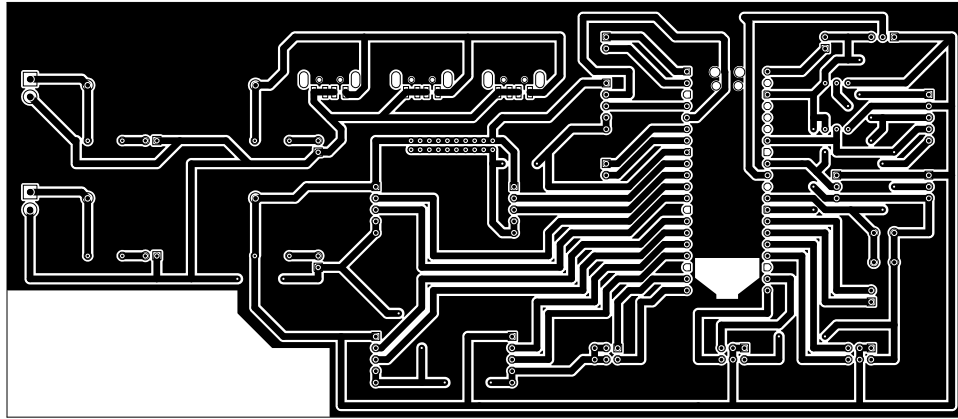


Figura 6.5: Diseño impreso para la realización dela PCB sobre una tarjeta fenólica ajustada a la escala correcta para las pistas.

6.3 Ejes del Robot.

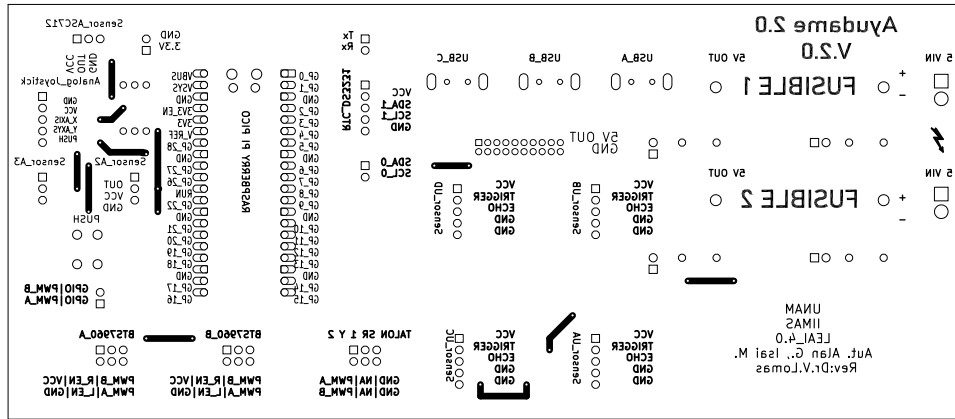


Figura 6.6: Diseño impreso para la realización de la PCB sobre una tarjeta fenólica ajustada a la escala correcta para la parte frontal de la PCB.

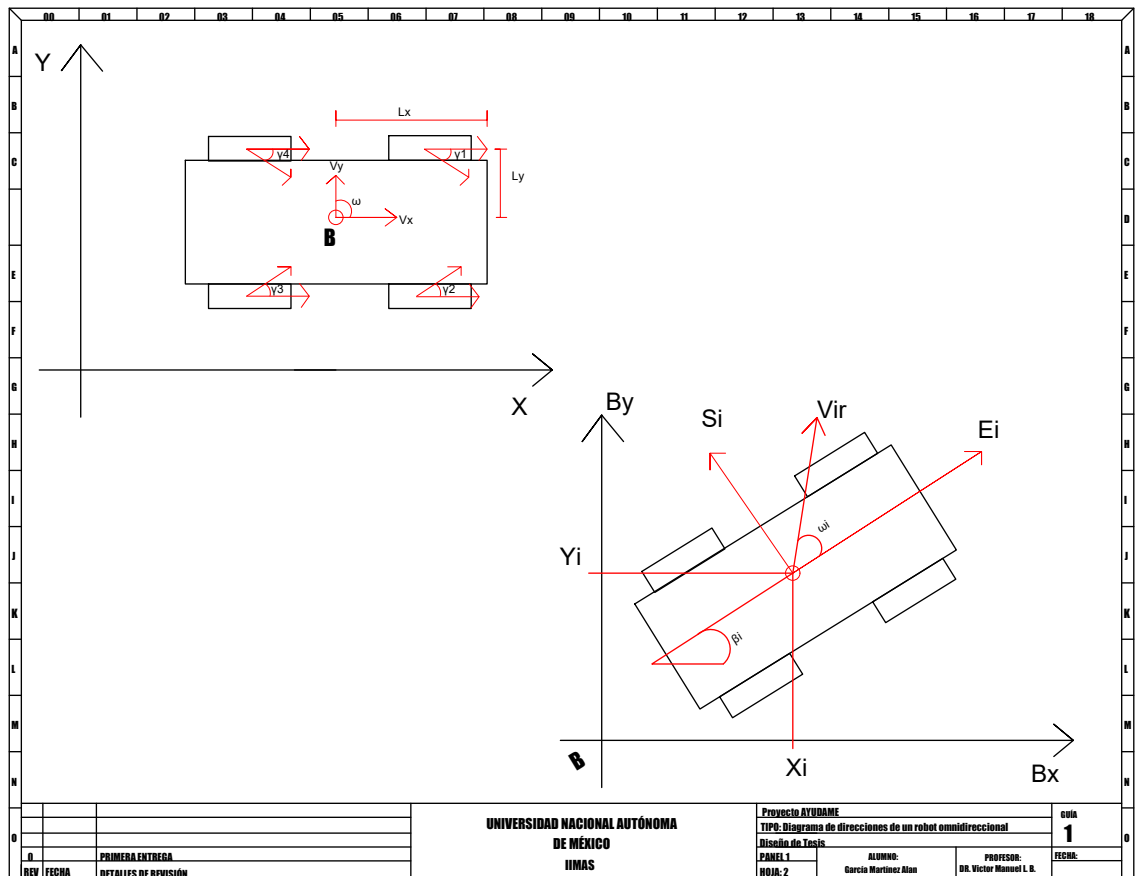


Figura 6.7: Diseño de los ejes del robot

Sección 6.4.

Planos del chasis.

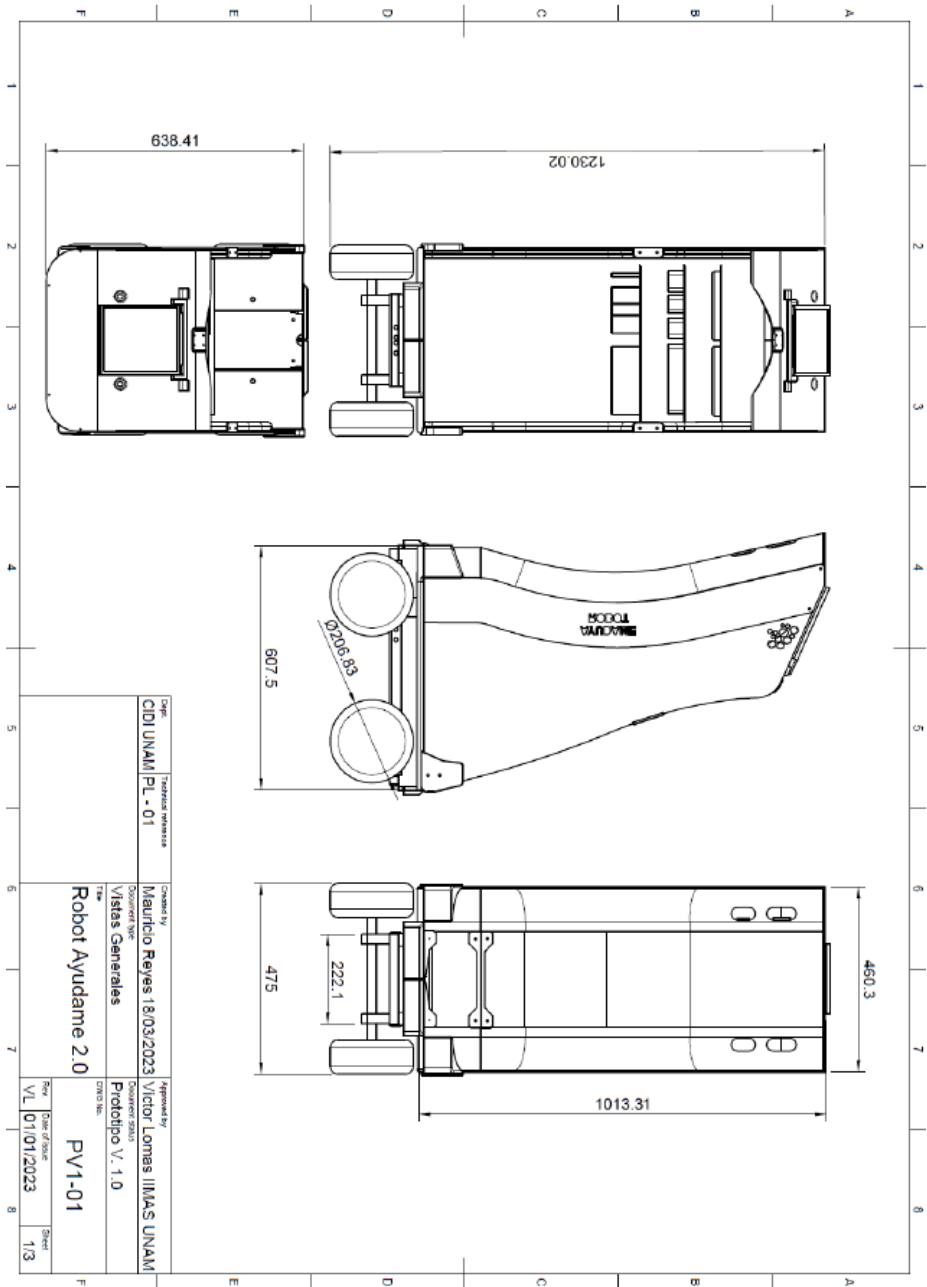


Figura 6.8: Prototipo Versión 1 A del robot Ayudame 2.0. Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0.

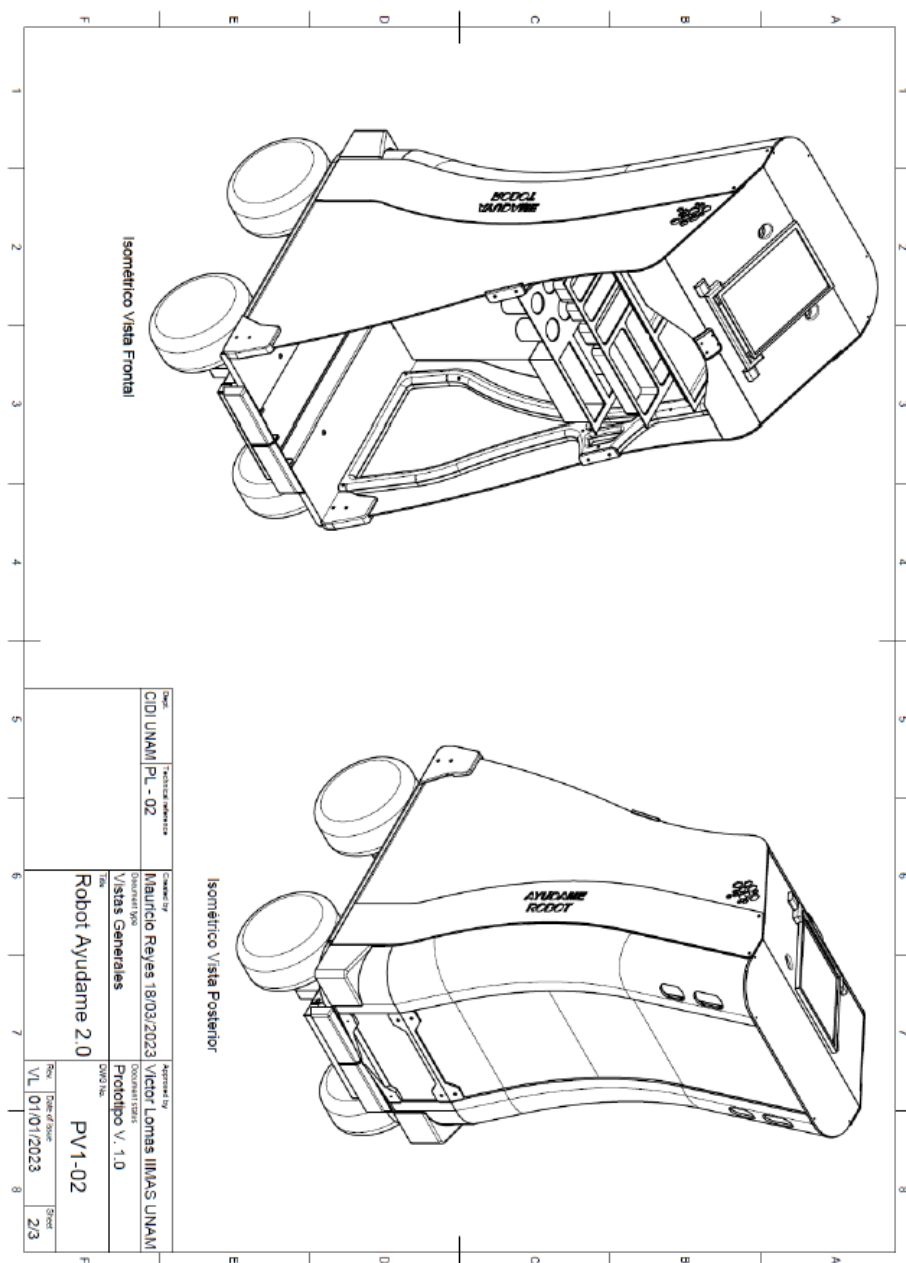


Figura 6.9: Prototipo Versión 1 B del robot Ayudame 2.0. Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0 .

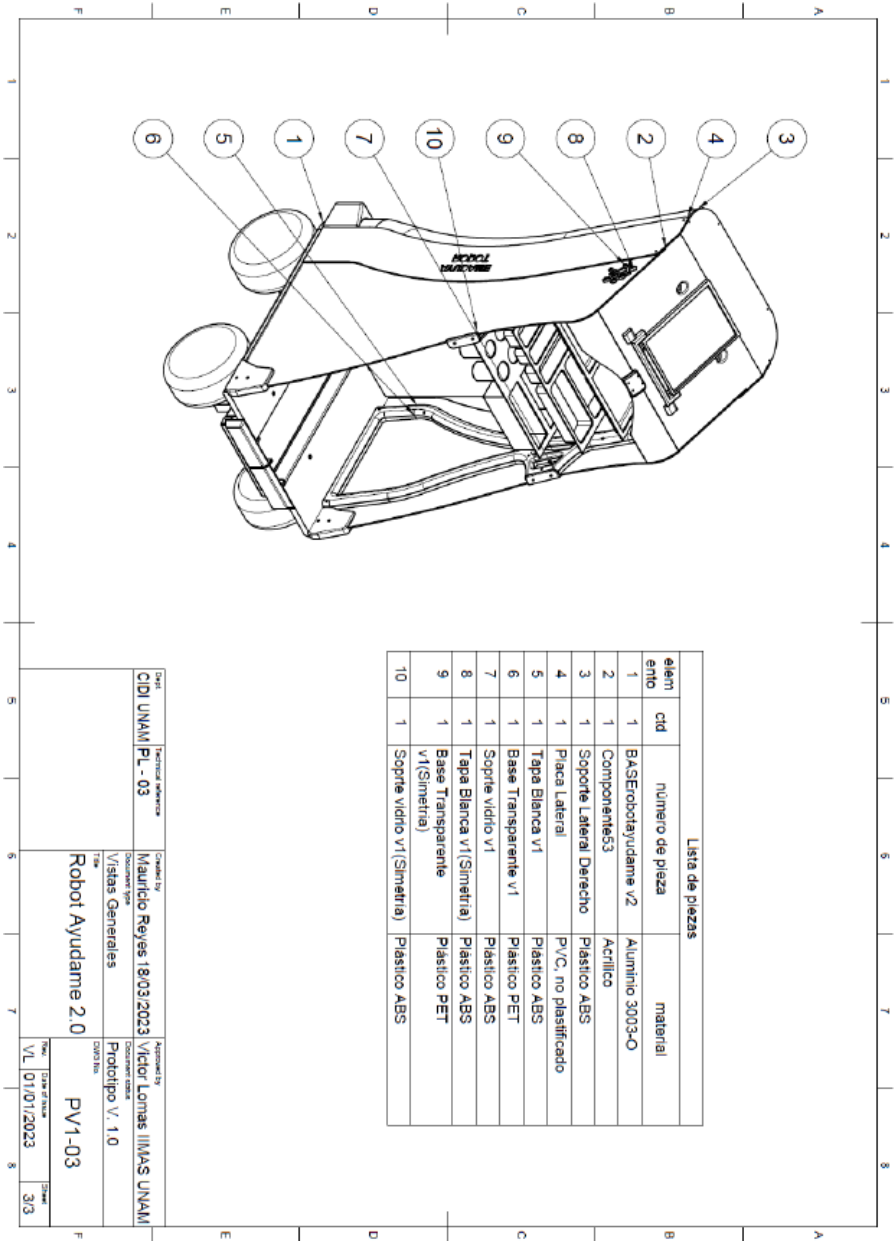


Figura 6.10: Prototipo Versión 1 C del robot Ayudame 2.0. Imagen tomada desde el documento de Memoria técnica de AYUDAME 2.0.

Disponible en: "<https://www.alldatasheet.com/datasheet-pdf/pdf/1179113/WORLDSEMI/WS2812B.html>"

Apéndice C. Códigos realizados

Todos los códigos mostrados se pueden descargar desde el siguiente repositorio de github.
”https://github.com/AlanGrcM/Ayudame_Raspberry_Pico_Microcontroller_C.git”

Sección 8.1.

Código utilizado para el Push Button.

```
1 #include <stdio.h>
2 #include "pico/stdlib.h"
3 #include "hardware/gpio.h"
4 #include "hardware/pwm.h"
5 #define LED_PIN 14
6 #define BUTTON_PIN 12
7 int main()
8 {
9     stdio_init_all();
10    gpio_init(BUTTON_PIN);
11    gpio_set_dir(BUTTON_PIN, GPIO_IN);
12    gpio_pull_up(BUTTON_PIN);
13    gpio_set_function(LED_PIN, GPIO_FUNC_PWM);
14    uint slice_num = pwm_gpio_to_slice_num(LED_PIN);
15    pwm_set_wrap(slice_num, 255);
16    pwm_set_chan_level(slice_num, PWM_CHAN_A, 0);
17    pwm_set_enabled(slice_num, true);
18    int pwm_value = 0;
19    while(1)
20    {
21        while(!gpio_get(BUTTON_PIN)){
22            pwm_value = 0;
23            printf("PWM: %d \n", pwm_value);
24            pwm_set_chan_level(slice_num, PWM_CHAN_A, pwm_value);
25            sleep_ms(500);
26            pwm_value = 64;
27            printf("PWM: %d \n", pwm_value);
28            pwm_set_chan_level(slice_num, PWM_CHAN_A, pwm_value);
29            sleep_ms(500);
30            pwm_value = 127;
31            printf("PWM: %d \n", pwm_value);
```

```

32  pwm_set_chan_level(slice_num, PWM_CHAN_A, pwm_value);
33  sleep_ms(500);
34  pwm_value = 191;
35  printf("PWM: %d \n", pwm_value);
36  pwm_set_chan_level(slice_num, PWM_CHAN_A, pwm_value);
37  sleep_ms(500);
38  pwm_value = 255;
39  printf("PWM: %d \n", pwm_value);
40  pwm_set_chan_level(slice_num, PWM_CHAN_A, pwm_value);
41  sleep_ms(500);
42  }
43  if(gpio_get(BUTTON_PIN)){
44  printf("Button pressed\n");
45  sleep_ms(1000);
46  }
47  }
48  }

```

Sección 8.2.

Código utilizado para la calibración del Talon SR

```

1 #include <stdio.h>
2 #include "pico/stdlib.h"
3 #include "hardware/gpio.h"
4 #include "hardware/adc.h"
5 #include "hardware/pwm.h"
6 //Definimos los nombres de los GPIO que vamos a usar
7 #define POT_PIN 26
8 #define LED_PIN 14
9 #define CAL_PIN 15
10 long map(long x, long in_min, long in_max, long out_min, long out_max)
11 {
12     return (x - in_min)*(out_max-out_min) / (in_max-in_min) + out_min;
13 }
14 int main(){
15     int frecuenciaDeseada = 300;
16     float Divider = (125000000/(4095*frecuenciaDeseada))/16; //= 3.3
17     long ciclos = 65535;
18     stdio_init_all();
19     adc_init();
20     adc_gpio_init(POT_PIN);
21     adc_select_input(0);
22     gpio_set_function(LED_PIN, GPIO_FUNC_PWM);
23     gpio_set_function(CAL_PIN, GPIO_FUNC_PWM);
24     uint slice_num = pwm_gpio_to_slice_num(LED_PIN);
25     pwm_set_clkdiv(slice_num, Divider);
26     pwm_set_wrap(slice_num, ciclos);
27     pwm_set_chan_level(slice_num, PWM_CHAN_A, 0);
28     pwm_set_chan_level(slice_num, PWM_CHAN_B, 0);
29     pwm_set_enabled(slice_num, true);
30
31     while (1)
32     {
33         uint16_t result = adc_read();
34         long pwm_value = map(result, 0, 4095, 0, ciclos);
35         float voltaje = pwm_value*3.3/65535;
36         printf("Raw: %d \t PWM: %d \t Voltaje PWM: %.1f \n", result,
pwm_value, voltaje);
37
38         pwm_set_chan_level(slice_num, PWM_CHAN_A, pwm_value);
39         pwm_set_chan_level(slice_num, PWM_CHAN_B, pwm_value);
40         sleep_ms(100);
41     }
42 }

```

Sección 8.3.

Código utilizado para el Display.

```
1 #include <stdio.h>
2 #include <string.h>
3 #include "pico/stdlib.h"
4 #include "hardware/i2c.h"
5 #include "pico/binary_info.h"
6 const int LCD_CLEARDISPLAY = 0x01;
7 const int LCD_RETURNHOME = 0x02;
8 const int LCD_ENTRYMODESET = 0x04;
9 const int LCD_DISPLAYCONTROL = 0x08;
10 const int LCD_CURSORSHIFT = 0x10;
11 const int LCD_FUNCTIONSET = 0x20;
12 const int LCD_SETCGRAMADDR = 0x40;
13 const int LCD_SETDDRAMADDR = 0x80;
14 const int LCD_ENTRYSHIFTINCREMENT = 0x01;
15 const int LCD_ENTRYLEFT = 0x02;
16 const int LCD_BLINKON = 0x01;
17 const int LCD_CURSORON = 0x02;
18 const int LCD_DISPLAYON = 0x04;
19 const int LCD_MOVERIGHT = 0x04;
20 const int LCD_DISPLAYMOVE = 0x08;
21 const int LCD_5x10DOTS = 0x04;
22 const int LCD_2LINE = 0x08;
23 const int LCD_8BITMODE = 0x10;
24 const int LCD_BACKLIGHT = 0x08;
25 const int LCD_ENABLE_BIT = 0x04;
26 static int addr = 0x27;
27 #define LCD_CHARACTER 1
28 #define LCD_COMMAND 0
29 #define MAX_LINES 2
30 #define MAX_CHARS 16
31 void i2c_write_byte(uint8_t val) {
32 #ifdef i2c_default
33     i2c_write_blocking(i2c_default, addr, &val, 1, false);
34 #endif
35 }
36 void lcd_toggle_enable(uint8_t val) {
37 #define DELAY_US 600
38     sleep_us(DELAY_US);
39     i2c_write_byte(val | LCD_ENABLE_BIT);
40     sleep_us(DELAY_US);
41     i2c_write_byte(val & ~LCD_ENABLE_BIT);
42     sleep_us(DELAY_US);
43 }
44 void lcd_send_byte(uint8_t val, int mode) {
45     uint8_t high = mode | (val & 0xF0) | LCD_BACKLIGHT;
46     uint8_t low = mode | ((val << 4) & 0xF0) | LCD_BACKLIGHT;
47     i2c_write_byte(high);
48     lcd_toggle_enable(high);
49     i2c_write_byte(low);
50     lcd_toggle_enable(low);
51 }
```

8.3 Ejes del Robot.

```
52 void lcd_clear(void) {
53     lcd_send_byte(LCD_CLEARDISPLAY, LCD_COMMAND);
54 }
55 void lcd_set_cursor(int line, int position) {
56     int val = (line == 0) ? 0x80 + position : 0xC0 + position;
57     lcd_send_byte(val, LCD_COMMAND);
58 }
59 static void inline lcd_char(char val) {
60     lcd_send_byte(val, LCD_CHARACTER);
61 }
62 void lcd_string(const char *s) {
63     while (*s) {
64         lcd_char(*s++);
65     }
66 }
67 void lcd_init() {
68     lcd_send_byte(0x03, LCD_COMMAND);
69     lcd_send_byte(0x03, LCD_COMMAND);
70     lcd_send_byte(0x03, LCD_COMMAND);
71     lcd_send_byte(0x02, LCD_COMMAND);
72     lcd_send_byte(LCD_ENTRYMODESET | LCD_ENTRYLEFT, LCD_COMMAND);
73     lcd_send_byte(LCD_FUNCTIONSET | LCD_2LINE, LCD_COMMAND);
74     lcd_send_byte(LCD_DISPLAYCONTROL | LCD_DISPLAYON, LCD_COMMAND);
75     lcd_clear();
76 }
77 int main() {
78     #if !defined(i2c_default) || !defined(PICO_DEFAULT_I2C_SDA_PIN) || !
defined(PICO_DEFAULT_I2C_SCL_PIN)
79         #warning i2c/lcd_1602_i2c example requires a board with I2C pins
80     #else
81         i2c_init(i2c_default, 100 * 1000);
82         gpio_set_function(PICO_DEFAULT_I2C_SDA_PIN, GPIO_FUNC_I2C);
83         gpio_set_function(PICO_DEFAULT_I2C_SCL_PIN, GPIO_FUNC_I2C);
84         gpio_pull_up(PICO_DEFAULT_I2C_SDA_PIN);
85         gpio_pull_up(PICO_DEFAULT_I2C_SCL_PIN);
86         bi_decl(bi_2pins_with_func(PICO_DEFAULT_I2C_SDA_PIN,
PICO_DEFAULT_I2C_SCL_PIN, GPIO_FUNC_I2C));
87         lcd_init();
88         static char *message[] =
89             {
90                 "Ejemplo LCD16X02", "Raspberry Pi Pico",
91             };
92         while (1) {
93             for (int m = 0; m < sizeof(message) / sizeof(message[0]); m +=
MAX_LINES) {
94                 for (int line = 0; line < MAX_LINES; line++) {
95                     lcd_set_cursor(line, (MAX_CHARS / 2) - strlen(message[m +
line]) / 2);
96                     lcd_string(message[m + line]);
97                 }
98                 sleep_ms(2000);
99                 lcd_clear();
100             }
101         }
102         return 0;
103     #endif
}
```


Sección 8.4.

Código utilizado para los sensores ultrasónicos.

```
1
2 #include <stdio.h>
3 #include "pico/stdlib.h"
4 #include "hardware/gpio.h"
5 uint trigPin_a = 0;
6 uint echoPin_a = 1;
7 uint trigPin_b = 2;
8 uint echoPin_b = 3;
9 uint trigPin_c = 4;
10 uint echoPin_c = 5;
11 uint trigPin_d = 6;
12 uint echoPin_d = 7;
13 int timeout = 26100;
14 void setupUltrasonicPins(uint trigPin, uint echoPin)
15 {
16     gpio_init(trigPin_a);
17     gpio_init(echoPin_a);
18     gpio_set_dir(trigPin_a, GPIO_OUT);
19     gpio_set_dir(echoPin_a, GPIO_IN);
20     gpio_init(trigPin_b);
21     gpio_init(echoPin_b);
22     gpio_set_dir(trigPin_b, GPIO_OUT);
23     gpio_set_dir(echoPin_b, GPIO_IN);
24     gpio_init(trigPin_c);
25     gpio_init(echoPin_c);
26     gpio_set_dir(trigPin_c, GPIO_OUT);
27     gpio_set_dir(echoPin_c, GPIO_IN);
28     gpio_init(trigPin_d);
29     gpio_init(echoPin_d);
30     gpio_set_dir(trigPin_d, GPIO_OUT);
31     gpio_set_dir(echoPin_d, GPIO_IN);
32 }
33
34 uint64_t getPulse(uint trigPin, uint echoPin)
35 {
36     gpio_put(trigPin, 1);
37     sleep_us(10);
38     gpio_put(trigPin, 0);
39     uint width = 0;
40     while(gpio_get(echoPin) == 0) tight_loop_contents();
41     while(gpio_get(echoPin) == 1)
42     {
43         width++;
44         sleep_us(2);
45         if(width > timeout) return 0;
46     }
47     return width;
48 }
49 int getCm(uint trigPin, uint echoPin)
50 {
51     uint64_t pulseLength = getPulse(trigPin, echoPin);
```

```

52     return pulseLength / 29 / 2;
53 }
54 int main()
55 {
56     stdio_init_all();
57     setupUltrasonicPins(trigPin_a, echoPin_a);
58     setupUltrasonicPins(trigPin_b, echoPin_b);
59     setupUltrasonicPins(trigPin_c, echoPin_c);
60     setupUltrasonicPins(trigPin_d, echoPin_d);
61     //gpio_pull_down(3);
62     while(true)
63     {
64         printf("\n Distancia_S1: \t  %d \t cm \t Distancia_S2: \t  %d \t cm \t
        Distancia_S3: \t  %d \t cm \t Distancia_S4 \t  %d \t cm \t ", getCm(
        trigPin_a, echoPin_a), getCm(trigPin_b, echoPin_b), getCm(trigPin_c,
        echoPin_c), getCm(trigPin_d, echoPin_d));
65         sleep_ms(200);
66     }
67 }

```

Sección 8.5.

Código utilizado para el RTC DS3231.

```
1 #include "ds3231.h"
2 char buf[]={0x00,0x50,0x59,0x17,0x04,0x05,0x03,0x21};
3 char *week={"SUN","Mon","Tues","Wed","Thur","Fri","Sat"};
4 int ds3231SetTime()
5 {
6     char buf[2];
7
8     buf[0]=0x00;
9     buf[1]=0x50;
10    i2c_write_blocking(I2C_PORT,addr,buf,2,false);
11
12    buf[0]=0x01;
13    buf[1]=0x59;
14    i2c_write_blocking(I2C_PORT,addr,buf,2,false);
15
16    buf[0]=0x02;
17    buf[1]=0x23;
18    i2c_write_blocking(I2C_PORT,addr,buf,2,false);
19
20    buf[0]=0x03;
21    buf[1]=0x04;
22    i2c_write_blocking(I2C_PORT,addr,buf,2,false);
23
24    buf[0]=0x04;
25    buf[1]=0x05;
26    i2c_write_blocking(I2C_PORT,addr,buf,2,false);
27
28    buf[0]=0x05;
29    buf[1]=0x03;
30    i2c_write_blocking(I2C_PORT,addr,buf,2,false);
31
32    buf[0]=0x06;
33    buf[1]=0x21;
34    i2c_write_blocking(I2C_PORT,addr,buf,2,false);
35    return 0;
36 }
37 void ds3231ReadTime()
38 {
39     uint8_t val = 0x00;
40     i2c_write_blocking(I2C_PORT,addr,&val,1,true);
41     i2c_read_blocking(I2C_PORT,addr,buf,7,false);
42 }
```

Sección 8.6.

Código utilizado para el Driver BTS7960.

```
1 #include <stdio.h>
2 #include "pico/stdlib.h"
3 #include "hardware/gpio.h"
4 #include "hardware/adc.h"
5 #include "hardware/pwm.h"
6 #define POT_PIN 26
7 #define L_PWM 14
8 #define R_PWM 15
9 long map(long x, long in_min, long in_max, long out_min, long out_max)
10 {
11     return (x - in_min)*(out_max-out_min) / (in_max-in_min) + out_min;
12 }
13 int main(){
14     int frecuenciaDeseada = 300;
15     float Divider = (125000000/(4095*frecuenciaDeseada))/16; //= 3.3
16     long ciclos = 65535;
17     stdio_init_all();
18     adc_init();
19     adc_gpio_init(POT_PIN);
20     adc_select_input(0);
21     gpio_set_function(L_PWM, GPIO_FUNC_PWM);
22     gpio_set_function(R_PWM, GPIO_FUNC_PWM);
23     uint slice_num_A = pwm_gpio_to_slice_num(L_PWM);
24     uint slice_num_B = pwm_gpio_to_slice_num(R_PWM);
25     pwm_set_clkdiv(slice_num_A, Divider);
26     pwm_set_clkdiv(slice_num_B, Divider);
27     pwm_set_wrap(slice_num_A, ciclos);
28     pwm_set_wrap(slice_num_B, ciclos);
29     pwm_set_chan_level(slice_num_A, PWM_CHAN_A, 0);
30     pwm_set_chan_level(slice_num_A, PWM_CHAN_B, 0);
31     pwm_set_chan_level(slice_num_B, PWM_CHAN_A, 0);
32     pwm_set_chan_level(slice_num_B, PWM_CHAN_B, 0);
33     pwm_set_enabled(slice_num_A, true);
34     pwm_set_enabled(slice_num_B, true);
35     while (1)
36     {
37         uint16_t result = adc_read();
38         long pwm_value = map(result, 0, 4095, 0, ciclos);
39         float voltaje = pwm_value*3.3/65535;
40         if(voltaje == 1.5 ){
41             pwm_set_chan_level(slice_num_A, PWM_CHAN_A, 256);
42             pwm_set_chan_level(slice_num_B, PWM_CHAN_B, 256);
43             printf("Alto \t Raw: %d \t PWM: %d \t Voltaje PWM: %.1f \n", result,
44                 pwm_value, voltaje);
45         }
46         if(voltaje <= 1.4){
47             pwm_set_chan_level(slice_num_A, PWM_CHAN_A, pwm_value);
48             pwm_set_chan_level(slice_num_B, PWM_CHAN_B, 256);
49             printf("Izquierda \t Raw: %d \t PWM: %d \t Voltaje PWM: %.1f \n",
50                 result, pwm_value, voltaje);
51         }
52     }
```

8.6 Ejes del Robot.

```
50  if(voltaje >= 1.6){
51      pwm_set_chan_level(slice_num_A, PWM_CHAN_A, 256);
52      pwm_set_chan_level(slice_num_B, PWM_CHAN_B, pwm_value);
53      printf("Derecha \t Raw: %d \t PWM: %d \t Voltaje PWM: %.1f \n",
54      result, pwm_value, voltaje);
55      }
56      sleep_ms(100);
57  }
```

```
1 #include <stdio.h>
2 #include <string.h>
3 #include "pico/stdlib.h"
4 #include "hardware/gpio.h"
5 #include "hardware/adc.h"
6 #include "hardware/pwm.h"
7 #include "hardware/i2c.h"
8 #include "pico/binary_info.h"
9 #include "ds3231.h"
10 #include <rcl/rcl.h>
11 #include <rcl/error_handling.h>
12 #include <rcl/rcl.h>
13 #include <rcl/executor.h>
14 #include <std_msgs/msg/int32.h>
15 #include <rmw_microros/rmw_microros.h>
16 #define LED_PIN 0
17 // #define I2C_PORT i2c1
18 // #define I2C_SCL 2
19 // #define I2C_SDA 3
20 // Definidos dentro de "ds3231.h"
21 #define PIN_SDA 4
22 #define PIN_SCL 5
23 uint trigPin_a = 6;
24 uint echoPin_a = 7;
25 uint trigPin_b = 8;
26 uint echoPin_b = 9;
27 uint trigPin_c = 10;
28 uint echoPin_c = 11;
29 uint trigPin_d = 12;
30 uint echoPin_d = 13;
31 #define TALON_1 14
32 #define TALON_2 15
33 #define BTS_1A 16
34 #define BTS_1B 17
35 #define BTS_2A 18
36 #define BTS_2B 19
37 #define ACS_OUT 26
38 #define JX_PIN 27
39 #define JY_PIN 28
40 #define BUTTON_PIN 22
41 const uint LED_PIN = 25;
42 #define LED_LENGTH 147
43 int timeout = 26100;
44
45 long map(long x, long in_min, long in_max, long out_min, long out_max)
46 {
47     return (x-in_min)*(out_max - out_min) / (in_max - in_min) + out_min;
48 }
49 void setupUltrasonicPins(uint trigPin, uint echoPin)
50 {
51     gpio_init(trigPin_a);
```

8.7 Ejes del Robot.

```
52  gpio_init(echoPin_a);
53  gpio_set_dir(trigPin_a, GPIO_OUT);
54  gpio_set_dir(echoPin_a, GPIO_IN);
55  gpio_init(trigPin_b);
56  gpio_init(echoPin_b);
57  gpio_set_dir(trigPin_b, GPIO_OUT);
58  gpio_set_dir(echoPin_b, GPIO_IN);
59  gpio_init(trigPin_c);
60  gpio_init(echoPin_c);
61  gpio_set_dir(trigPin_c, GPIO_OUT);
62  gpio_set_dir(echoPin_c, GPIO_IN);
63  gpio_init(trigPin_d);
64  gpio_init(echoPin_d);
65  gpio_set_dir(trigPin_d, GPIO_OUT);
66  gpio_set_dir(echoPin_d, GPIO_IN);
67  }
68  uint64_t getPulse(uint trigPin, uint echoPin)
69  {
70      gpio_put(trigPin, 1);
71      sleep_us(10);
72      gpio_put(trigPin, 0);
73      uint width = 0;
74      while(gpio_get(echoPin) == 0) tight_loop_contents();
75      while(gpio_get(echoPin) == 1)
76      {
77          width++;
78          sleep_us(2);
79          if(width > timeout) return 0;
80      }
81      return width;
82  }
83  int getCm(uint trigPin, uint echoPin)
84  {
85      uint64_t pulseLength = getPulse(trigPin, echoPin);
86      return pulseLength / 29 / 2;
87  }
88  int getInch (uint trigPin, uint echoPin)
89  {
90      uint64_t pulseLength = getPulse(trigPin, echoPin);
91      return (long)pulseLength /74.f /2.f;
92  }
93
94  const int LCD_CLEARDISPLAY = 0x01;
95  const int LCD_RETURNHOME = 0x02;
96  const int LCD_ENTRYMODESET = 0x04;
97  const int LCD_DISPLAYCONTROL = 0x08;
98  const int LCD_CURSORSHIFT = 0x10;
99  const int LCD_FUNCTIONSET = 0x20;
100 const int LCD_SETCGRAMADDR = 0x40;
101 const int LCD_SETDDRAMADDR = 0x80;
102 const int LCD_ENTRYSHIFTINCREMENT = 0x01;
103 const int LCD_ENTRYLEFT = 0x02;
104 const int LCD_BLINKON = 0x01;
105 const int LCD_CURSORON = 0x02;
106 const int LCD_DISPLAYON = 0x04;
107 const int LCD_MOVERIGHT = 0x04;
```

```

108 const int LCD_DISPLAYMOVE = 0x08;
109 const int LCD_5x10DOTS = 0x04;
110 const int LCD_2LINE = 0x08;
111 const int LCD_8BITMODE = 0x10;
112 const int LCD_BACKLIGHT = 0x08;
113 const int LCD_ENABLE_BIT = 0x04;
114
115 static int addr_2 = 0x27;
116
117 #define LCD_CHARACTER 1
118 #define LCD_COMMAND 0
119
120 #define MAX_LINES 2
121 #define MAX_CHARS 16
122
123
124 void i2c_write_byte(uint8_t val) {
125 #ifdef i2c_default
126     i2c_write_blocking(i2c0, addr_2, &val, 1, false);
127 #endif
128 }
129
130 void lcd_toggle_enable(uint8_t val) {
131
132
133 #define DELAY_US 600
134     sleep_us(DELAY_US);
135     i2c_write_byte(val | LCD_ENABLE_BIT);
136     sleep_us(DELAY_US);
137     i2c_write_byte(val & ~LCD_ENABLE_BIT);
138     sleep_us(DELAY_US);
139 }
140
141
142 void lcd_send_byte(uint8_t val, int mode) {
143     uint8_t high = mode | (val & 0xF0) | LCD_BACKLIGHT;
144     uint8_t low = mode | ((val << 4) & 0xF0) | LCD_BACKLIGHT;
145
146     i2c_write_byte(high);
147     lcd_toggle_enable(high);
148     i2c_write_byte(low);
149     lcd_toggle_enable(low);
150 }
151
152 void lcd_clear(void) {
153     lcd_send_byte(LCD_CLEARDISPLAY, LCD_COMMAND);
154 }
155
156
157 void lcd_set_cursor(int line, int position) {
158     int val = (line == 0) ? 0x80 + position : 0xC0 + position;
159     lcd_send_byte(val, LCD_COMMAND);
160 }
161
162 static void inline lcd_char(char val) {
163     lcd_send_byte(val, LCD_CHARACTER);

```


8.7 Ejes del Robot.

```
164 }
165
166 void lcd_string(const char *s) {
167     while (*s) {
168         lcd_char(*s++);
169     }
170 }
171
172 void lcd_init() {
173     lcd_send_byte(0x03, LCD_COMMAND);
174     lcd_send_byte(0x03, LCD_COMMAND);
175     lcd_send_byte(0x03, LCD_COMMAND);
176     lcd_send_byte(0x02, LCD_COMMAND);
177
178     lcd_send_byte(LCD_ENTRYMODESET | LCD_ENTRYLEFT, LCD_COMMAND);
179     lcd_send_byte(LCD_FUNCTIONSET | LCD_2LINE, LCD_COMMAND);
180     lcd_send_byte(LCD_DISPLAYCONTROL | LCD_DISPLAYON, LCD_COMMAND);
181     lcd_clear();
182 }
183
184 rcl_publisher_t publisher;
185 std_msgs__msg__Int32 msg;
186
187 void timer_callback(rcl_timer_t *timer, int64_t last_call_time)
188 {
189     rcl_ret_t ret = rcl_publish(&publisher, &msg, NULL);
190     msg.data==getCm(trigPin, echoPin);
191 }
192
193 extern char buf[];
194 extern char *week[];
195
196
197 int main()
198 {
199     #if !defined(i2c_default) || !defined(PICO_DEFAULT_I2C_SDA_PIN) || !
200         defined(PICO_DEFAULT_I2C_SCL_PIN)
201     #warning i2c/lcd_1602_i2c example requires a board with I2C pins
202     #else
203     int i = 0;
204     char *movimiento;
205     long T1;
206     long T2;
207     long BTS1A;
208     long BTS1B;
209     long BTS2A;
210     long BTS2B;
211     rmw_uros_set_custom_transport(
212     true,
213     NULL,
214     pico_serial_transport_open,
215     pico_serial_transport_close,
216     pico_serial_transport_write,
217     pico_serial_transport_read
218     );
219     stdio_init_all();
```

```

219
220     adc_init();
221
222     adc_gpio_init(JX_PIN);
223     adc_gpio_init(JY_PIN);
224
225     gpio_init(BUTTON_PIN);
226     gpio_set_dir(BUTTON_PIN, GPIO_IN);
227     gpio_pull_up(BUTTON_PIN);
228
229     gpio_set_function(TALON_1, GPIO_FUNC_PWM);
230     gpio_set_function(TALON_2, GPIO_FUNC_PWM);
231     gpio_set_function(BTS_1A, GPIO_FUNC_PWM);
232     gpio_set_function(BTS_1B, GPIO_FUNC_PWM);
233     gpio_set_function(BTS_2A, GPIO_FUNC_PWM);
234     gpio_set_function(BTS_2B, GPIO_FUNC_PWM);
235
236     uint slice_num_A = pwm_gpio_to_slice_num(TALON_1);
237     pwm_set_clkdiv(slice_num_A, 6.3); //6.3
238     pwm_set_wrap(slice_num_A, 65465);
239     pwm_set_chan_level(slice_num_A, PWM_CHAN_A, 7);
240     pwm_set_enabled(slice_num_A, true);
241
242     uint slice_num_B = pwm_gpio_to_slice_num(TALON_2);
243     pwm_set_clkdiv(slice_num_B, 6.3); //6.3
244     pwm_set_wrap(slice_num_B, 65465);
245     pwm_set_chan_level(slice_num_B, PWM_CHAN_B, 7);
246     pwm_set_enabled(slice_num_B, true);
247
248     uint slice_num_C = pwm_gpio_to_slice_num(BTS_1A);
249     pwm_set_wrap(slice_num_C, 65465);
250     pwm_set_chan_level(slice_num_C, PWM_CHAN_A, 0);
251     pwm_set_enabled(slice_num_C, true);
252
253     uint slice_num_D = pwm_gpio_to_slice_num(BTS_1B);
254     pwm_set_wrap(slice_num_D, 65465);
255     pwm_set_chan_level(slice_num_D, PWM_CHAN_B, 0);
256     pwm_set_enabled(slice_num_D, true);
257
258     uint slice_num_E = pwm_gpio_to_slice_num(BTS_2A);
259     pwm_set_wrap(slice_num_E, 65465);
260     pwm_set_chan_level(slice_num_E, PWM_CHAN_A, 1);
261     pwm_set_enabled(slice_num_E, true);
262
263     uint slice_num_F = pwm_gpio_to_slice_num(BTS_2B);
264     pwm_set_wrap(slice_num_F, 65465);
265     pwm_set_chan_level(slice_num_F, PWM_CHAN_B, 1);
266     pwm_set_enabled(slice_num_F, true);
267
268     i2c_init(i2c1, 100*1000);
269     gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
270     gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
271     gpio_pull_up(I2C_SDA);
272     gpio_pull_up(I2C_SCL);
273
274     printf("DS3231 Test Program ...\n\n");

```

```

275
276 ds3231SetTime();
277
278
279 i2c_init(i2c_default, 100 * 1000);
280 gpio_set_function(PICO_DEFAULT_I2C_SDA_PIN, GPIO_FUNC_I2C);
281 gpio_set_function(PICO_DEFAULT_I2C_SCL_PIN, GPIO_FUNC_I2C);
282 gpio_pull_up(PICO_DEFAULT_I2C_SDA_PIN);
283 gpio_pull_up(PICO_DEFAULT_I2C_SCL_PIN);
284
285 bi_decl(bi_2pins_with_func(PICO_DEFAULT_I2C_SDA_PIN,
286 PICO_DEFAULT_I2C_SCL_PIN, GPIO_FUNC_I2C));
286 lcd_init();
287 static char *message[] =
288 {
289     "Robot AYUDAME", "2.0",
290 };
291
292 setupUltrasonicPins(trigPin_a, echoPin_a);
293 setupUltrasonicPins(trigPin_b, echoPin_b);
294 setupUltrasonicPins(trigPin_c, echoPin_c);
295 setupUltrasonicPins(trigPin_d, echoPin_d);
296 T1 = 32700;
297 T2 = 32700;
298 BTS1A = 0;
299 BTS1B = 0;
300 BTS2A = 0;
301 BTS2B = 0;
302 gpio_init(LED_PIN);
303 gpio_set_dir(LED_PIN, GPIO_OUT);
304
305 rcl_timer_t timer;
306 rcl_node_t node;
307 rcl_allocator_t allocator;
308 rcl_support_t support;
309 rcl_executor_t executor;
310 allocator = rcl_get_default_allocator();
311
312 const int timeout_ms = 1000;
313 const uint8_t attempts = 120;
314
315 rcl_ret_t ret = rmw_uros_ping_agent(timeout_ms, attempts);
316
317 if (ret != RCL_RET_OK)
318 {
319     return ret;
320 }
321
322 rcl_support_init(&support, 0, NULL, &allocator);
323
324 rcl_node_init_default(&node, "pico_node", "", &support);
325 rcl_publisher_init_default(
326     &publisher,
327     &node,
328     ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
329     "pico_publisher");

```

```

330
331     rclc_timer_init_default(
332         &timer,
333         &support,
334         RCL_MS_TO_NS(1000),
335         timer_callback);
336
337     rclc_executor_init(&executor, &support.context, 1, &allocator);
338     rclc_executor_add_timer(&executor, &timer);
339
340     gpio_put(LED_PIN, 1);
341
342     msg.data = 0;
343
344     while (1) {
345         rclc_executor_spin_some(&executor, RCL_MS_TO_NS(100));
346         ds3231ReadTime();
347         buf[0] = buf[0]&0x7F;
348         buf[1] = buf[1]&0x7F;
349         buf[2] = buf[2]&0x3F;
350         buf[3] = buf[3]&0x07;
351         buf[4] = buf[4]&0x3F;
352         buf[5] = buf[5]&0x1F;
353
354         printf(" || Fecha: 20%02x/%02x/%02x ||", buf[6], buf[5], buf[4]);
355
356         printf(" || Horario: %02x:%02x:%02x || ", buf[2], buf[1], buf[0]);
357
358         printf(" || Dia: %s || \n ", week[(unsigned char)buf[3]-1]);
359
360         if(buf[6]==0x00 && buf[5]==0x01 && buf[4]==0x01)
361         {
362             ds3231SetTime();
363         }
364         adc_select_input(1);
365         uint adc_x_raw = adc_read();
366         adc_select_input(2);
367         uint adc_y_raw = adc_read();
368
369         if(adc_y_raw == 4095 && adc_x_raw == 4095){
370             T1 = 32700;
371             T2 = 32700;
372             BTS1A = 0;
373             BTS1B = 0;
374             BTS2A = 0;
375             BTS2B = 0;
376             movimiento = "ALTO";
377         }
378         else if(adc_y_raw == 4095 && adc_x_raw <= 4000){
379             T1 = 40600;
380             T2 = 40600;
381             BTS1A=0;
382             BTS1B=20000;
383             BTS2A=20000;
384             BTS2B=0;
385             movimiento = "Adelante";

```

8.7 Ejes del Robot.

```
386     }
387     else if (adc_y_raw <= 4000 && adc_x_raw == 4095){
388         T1 = 17700;
389         T2 = 17700;
390         BTS1A = 20000;
391         BTS1B = 0;
392         BTS2A = 0;
393         BTS2B = 20000;
394         movimiento = "Atras";
395     }
396
397     else{
398         printf("X: %d \t Y: %d \t Static %s \n", adc_y_raw, adc_x_raw);
399         sleep_ms(200);
400     }
401     while (!gpio_get(BUTTON_PIN)){
402         T1 = 21700;
403         T2 = 39200;
404         BTS1A = 0;
405         BTS1B = 18000;
406         BTS2A = 0;
407         BTS2B = 18000;
408         movimiento = "Giro en 360 grados";
409         pwm_set_chan_level(slice_num_A, PWM_CHAN_A, T1);
410         pwm_set_chan_level(slice_num_B, PWM_CHAN_B, T2);
411         pwm_set_chan_level(slice_num_C, PWM_CHAN_A, BTS1A);
412         pwm_set_chan_level(slice_num_D, PWM_CHAN_B, BTS1B);
413         pwm_set_chan_level(slice_num_E, PWM_CHAN_A, BTS2A);
414         pwm_set_chan_level(slice_num_F, PWM_CHAN_B, BTS2B);
415         printf("X: %d \t Y: %d \t Movimiento = %s \n", adc_y_raw,
adc_x_raw, movimiento);
416         sleep_ms(50);
417     }
418     pwm_set_chan_level(slice_num_A, PWM_CHAN_A, T1);
419     pwm_set_chan_level(slice_num_B, PWM_CHAN_B, T2);
420     pwm_set_chan_level(slice_num_C, PWM_CHAN_A, BTS1A);
421     pwm_set_chan_level(slice_num_D, PWM_CHAN_B, BTS1B);
422     pwm_set_chan_level(slice_num_E, PWM_CHAN_A, BTS2A);
423     pwm_set_chan_level(slice_num_F, PWM_CHAN_B, BTS2B);
424     printf("X: %d \t Y: %d \t Movimiento = %s \n", adc_y_raw,
adc_x_raw, movimiento);
425
426     printf("\n Distancia_S1: \t %d \t cm \t Distancia_S2: \t %d \t cm \t
Distancia_S3: \t %d \t cm \t Distancia_S4 \t %d \t cm \t ", getCm(
trigPin_a, echoPin_a), getCm(trigPin_b, echoPin_b), getCm(trigPin_c,
echoPin_c), getCm(trigPin_d, echoPin_d));
427
428     for (int m = 0; m < sizeof(message) / sizeof(message[0]); m += MAX_LINES
) {
429         for (int line = 0; line < MAX_LINES; line++) {
430             lcd_set_cursor(line, (MAX_CHARS / 2) - strlen(message[m +
line]) / 2);
431             lcd_string(message[m + line]);
432         }
433     }
434 }
```

```
435     sleep_ms(100);  
436 }  
437     return 0;  
438 #endif  
439 }
```

Bibliografía

- [1] WHO, “WHO Coronavirus Disease (COVID-19) Dashboard”. [En línea]. Disponible en: <https://covid19.who.int/>.
- [2] “Covid-19 México, Datos Abiertos Dirección General de Epidemiología”. [En línea]. Disponible en: <https://datos.covid-19.conacyt.mx/>
- [3] M. Igoe y V. Chadwick, “After the pandemic : How will COVID-19 transform global health and development ?”, Devex, núm. April, pp. 1–9, 2020.
- [4] V. M. Lomas-Barrié, M. Peña-Cabrera, y T. Alcantara-Concepcion, “AYUDAME 1.0”, 2020. [En línea]. Disponible en: http://www.leai4.iimas.unam.mx/?page_id=101. [Consultado : 07 – jul – 2020].
- [5] H. Durrant-Whyte y T. Bailey, “Simultaneous localization and mapping: Part I”, IEEE Robot. Autom. Mag., vol. 13, núm. 2, pp. 99–108, 2006.
- [6] Bailey y H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): Part II”, IEEE Robot. Autom. Mag., vol. 13, núm. 3, pp. 108–117, 2006.
- [7] S. Gatesichapakorn, J. Takamatsu, y M. Ruchanurucks, “ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera”, 2019 1st Int. Symp. Instrumentation, Control. Artif. Intell. Robot. ICA-SYMP 2019, pp. 151–154, 2019.
- [8] R. Liu, J. Shen, C. Chen, y J. Yang, “SLAM for Robotic Navigation by Fusing RGB-D and Inertial Data in Recurrent and Convolutional Neural Networks”, 2019 IEEE 5th Int. Conf. Mechatronics Syst. Robot. ICMSR 2019, pp. 1–6, 2019.
- [9] P. Henry, M. Krainin, E. Herbst, X. Ren, y D. Fox, “RGB-D mapping: Using depth cameras forense 3D modeling of indoor environments”, en Springer Tracts in Advanced Robotics, 2014, vol. 79, pp. 477–491.
- [10] F. Endres, J. Hess, J. Sturm, D. Cremers, y W. Burgard, “3-D Mapping with an RGB-D camera”, en IEEE Transactions on Robotics, 2014, vol. 30, núm. 1, pp. 177–187.
- [11] M. Labbé y F. Michaud, “Online global loop closure detection for large-scale multi-session graph-based SLAM”, en IEEE International Conference on Intelligent Robots and Systems, 2014, pp. 2661–2666.

- [12] X. Liu, B. Guo, y C. Meng, "A method of simultaneous location and mapping based on RGB-D cameras", en 2016 14th International Conference on Control, Automation, Robotics and Vision, ICARCV 2016, 2017.
- [13] V. Lomas-Barrie, M. Peña-Cabrera, y J. Durán-Ortega, "Determining humanoid soccer player position based on Goal detection", en Proceedings of the 2015 International Conference on Artificial Intelligence, ICAI 2015 - WORLDCOMP 2015, 2015, pp. 61–65.
- [14] P.-C. Mario, L.-J. Ismael, R.-C. Reyes, y C.-C. Jorge, "Machine vision approach for robotic assembly", *Assem. Autom.*, vol. 25, núm. 3, pp. 204–216, 2005.
- [15] IFR Press Release. Industrial Robots: Robot Investment Reaches Record 16.5 billion USD. Shanghai, Frankfurt, Sep 18, 2019 [consultado 9 Mar 2019] Disponible en: <https://ifr.org/ifr-press-releases/news/robot-investment-reaches-record-16.5-billion-usd>
- [16] Posibilidades de la robótica y perspectivas de la robótica en la medicina <https://www.revistadyna.com/busqueda/posibilidades-y-perspectivas-de-robotica-en-medicina>
- [17] Ramos, A. C. (2001). Estado del arte en cirugía robótica. *Revista Mexicana de Cirugía Endoscópica*, 2(2), 109–112.
- [18] Cornejo Aguilar, J. A., Cornejo, J., Vargas, M., Sebastian, R. (2019). La revolución de la cirugía robótica en latino américa y la futura implementación en el sistema de salud del Perú. *Revista de la Facultad de Medicina Humana*, 19(1), 16. [10] R.H. Taylor A Perspective on Medical Robotics *Proceedings of the IEEE*.
- [19] O. Diegel, A. Badve, G. Bright, J. Potgieter, and S. Tlale, Improved Mecanum Wheel Design for Omni-directional Robots, no. November, 2002.
- [20] I. Doroftei, V. Grosu, and V. Spinu, Omnidirectional Mobile Robot - Design and Implementation, *Bioinspiration and Robotics Walking and Climbing Robots*, no. September. I-Tech, 2007
- [21] Dickerson, S.L. & Lapin, B.D. (1991). Control of an omni directional robotic vehicle with Mecanum wheels. In: *National Telesystems Conference Proceedings*, pp.323-328, March 26-27, 1991, Atlanta, USA.
- [22] Everett, H.R. (1995). *Sensors for Mobile Robots: Theory and Application*. A K Peters, Ltd, MA, USA
- [23] M. West and H. Asada, "Design and control of ball wheel omnidirectional vehicles," *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, 1995, pp. 1931-1938 vol.2, doi: 10.1109/ROBOT.1995.525547.
- [24] Dickerson SL, Lapin DB. Control of Omni-Directional Robotic Vehicle with Mecanum Wheels. *IEEE Proceedings of National Tele-systems Conference* 1991;323-8.
- [25] Tlale NS. On Distributed Mechatronic Controller for Omni Direction Autonomous Guided Vehicles. *Industrial Robot: An International Journal* 2006.

- [26] Shimada A, Yajima S, Viboonchaicheep P, Samura K. Mecanum-wheel vehicle systems based on position corrective control. 32nd Annual Conference of IEEE Industrial Electronics Society 2005.
- [27] Reyes, Mauricio & Murillo, Andrés & Mendoza, Carlos. (2022). Consideraciones en el diseño de robots para la atención médica en el mundo post COVID-19. Cultura Científica y Tecnológica
- [28] López R., Leonardo J., Franco m., Zulay E., Pateti M., Antonio S.,”METODOLOGÍA DE IMPLEMENTACIÓN DE UN CONTROLADOR PID DIFUSO EN UN FPGA”, Universidad Nacional Experimental Politécnica “Antonio José de Sucre”, Julio 2006.
- [29] Jhonny A. Valencia V., Alejandro Montoya O., Luis Hernando Ríos, “Modelo Cinemático de un Robot Móvil tipo Diferencial y de Navegación a partir de la Estimación Odométrica”, Universidad Tecnológica de Pereira, Mayo de 2009.
- [30] J. Bacik, F. Durovsky, M. Biroš, K. Kyslan, D. Perdukova y S. Padmanaban, “Pathfinder–Development of Automated Guided Vehicle for Hospital Logistics”, IEEE Access, vol. 5, pp. 26892-26900, 2017, doi: 10.1109/ACCESS.2017.2767899.
- [31] T. Bányai, I. Maral, B. Illés, Á. Bányai y P. Tamás, “Optimization of Operation Strategy for Collection Systems of Biohazard Wastes in Hospitals Based on Autonomous Robots: A Heuristic Approach”, AJRCoS, pp. 33-43, jun. 2020, doi: 10.9734/ajrcos/2020/v5i430142.
- [32] Le-Jie Zhang, Zeng-Guang Hou, Min TAN, “Kalman Filter and Vision Localization Based Potential Field Method for Autonomous Mobile Robots”, the Chinese of Sciences, July 2005.
- [33] María del Rocío Ortiz Albarrán, “Implementación de Modelos Ocultos de Markov en la navegación de un robot móvil autónomo”, Universidad Nacional Autónoma de México, 2011.