Instituto de Ingeniería y Tecnología
Diseño Mecatronico

Sistema de Estacionamiento en Paralelo para Plaforma movil tipo Ackermann

Luis Armando Enciso Nava	215766
Luis Eli Soto Arredondo	201018
Francisco Javier Hernandez Gutierrez	201215
Angel Reynaldo Garcia Saucedo	201957
Jorge Ivan Ruiz Espinoza	201621

21 de mayo de 2025

Índice

Índice de figuras

1. Introducción

La adquisición de datos en un entorno dinámico es un proceso fundamental que permite la recopilación de información mediante diversos métodos y tecnologías, con el objetivo de analizar y comprender el funcionamiento del sistema en estudio. En la industria y la investigación, esta práctica se ha convertido en un pilar esencial, especialmente en el desarrollo de plataformas móviles capaces de ejecutar maniobras.
Este proyecto propone la implementación de un sistema de estacionamiento en paralelo para una plataforma con configuración Ackermann. Para ello, se utilizarán dispositivos ESP32, los cuales serán programados para que sean capaces tanto del control de movimientos de la plataforma móvil como de la recopilación y procesamiento de datos esenciales. A través de la programación, el sistema realizará las mediciones necesarias para ejecutar la maniobra de estacionamiento . Además, los datos obtenidos serán transmitidos a otros dispositivos, permitiendo así la simulación digital del movimiento y su implementación en el modelo físico.

1.1. Objetivo

Implementar un sistema de estacionamiento en paralelo para una plataforma móvil tipo Ackermann.

1.2. Objetivos Particulares

- Equipar una plataforma móvil tipo Ackermann con un sistema con la propiedad de realizar maniobras de estacionamiento en paralelo.
- Crear una representación tridimensional de la plataforma de manera sincronizada con su ejecución física.
- Implementar un sistema de comunicación entre el microcontrolador, la simulación 3D y el robot móvil físico.

2. Desarrollo

2.1. Estacionamiento en Paralelo

Es una maniobra en la que se busca estacionar la plataforma movil ocupando el espacio entre dos objetos los cuales ya están previamente estacionados. Se define estacionamiento paralelo porque, una vez estacionado el automóvil luego de la maniobra, este queda en paralelo al borde de la calzada.

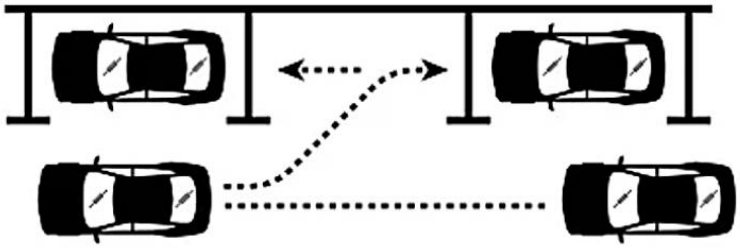


Figura 1: estacionamiento en paralelo

2.2. Plataforma de desarrollo robótico en Linux

Para esta parte se planea usar ROS2 que proviene de las siglas en inglés de Robot Operating System el cual es un conjunto de bibliotecas y herramientas de software que sirven para construir aplicaciones de robotica en linux. Desde controladores hasta algoritmos de vanguardia y con potentes herramientas para desarrollo. ROS2 tiene todo lo que se podría necesitar a la hora de realizar proyectos de robótica ademas es muy accesible ya que es de codigo abierto.

2.3. Lista de Materiales

- Chasis con dirección tipo Ackerman
- Microcontrolador ESP-32
- Sensor láser "time of flight"VL053X
- Motores con Encoder de dos canales
- Unidad de medición inercial MPU
- Batería 12v 3600 mah
- Servomotor MG996R
- Buzzer

2.3.1. Chasis con dirección tipo Ackerman

El chasis está equipado con un par de motores de reducción de CC de alto rendimiento, neumáticos de goma, servo digital MG-996R. El chasis adopta una estructura de aleación de aluminio totalmente metálica con una superficie anodizada. El chasis adopta una estructura de dirección Ackerman.
Ambos motores cuentan con un encoder rotativo de dos canales, el cual necesita una alimentación de 3.3v para su correcto funcionamiento. Dichos motores funcionan con un voltaje de 12v.

Figura 2: Plataforma movil

2.3.2. Microcontrolador ESP-32

El ESP32 es un microcontrolador creado por Espressif Systems. Este dispositivo es el sucesor del ESP8266. Tiene WiFi y Bluetooth y funciona con bajo consumo. Esto lo hace perfecto para proyectos que usan baterías. Además, es más rápido que un Arduino. Tiene pines sensibles al tacto y un sensor de efecto Hall.

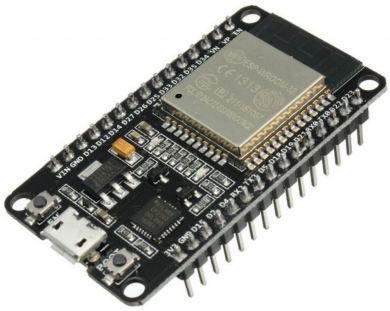


Figura 3: ESP32

2.3.3. Sensor láser "time of fly"VL053X

Sensor laser, tambien llamado "time of fly.[es] un sensor muy preciso el cual proyecta un laser para medir la distancia hacia un objeto

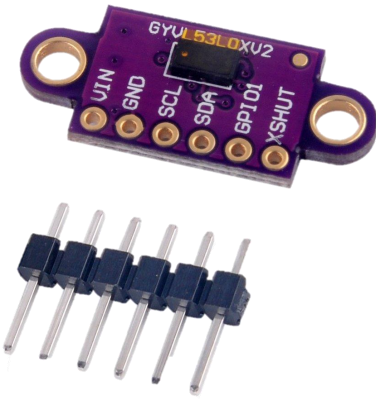


Figura 4: Sensor láser

2.3.4. Motorreductor 12V con encoder de dos canales

Motorreductor de 12 voltios, el cual tiene acoplado un encoder de dos canales de efecto hall de 350 rpm

Figura 5: Motoreductor de 12V

2.3.5. MPU-6050

El MPU-6050 es una unidad de medición inercial (IMU) de seis grados de libertad (6DOF) fabricado por Invensense, que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes.

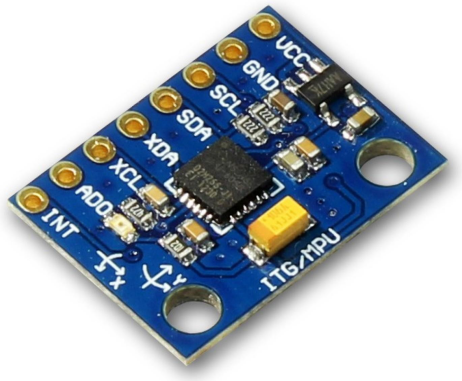


Figura 6: Sensor de acelerómetro y giroscopio MPU-6050

Acelerómetro

- Mide la aceleración lineal en tres ejes (x, y, z).
- Detecta cambios en la velocidad del objeto.
- Proporciona datos sobre la orientación del objeto en el espacio.

Giroscopio

- Mide la rotación angular en tres ejes (x, y, z).
- Detecta cambios en la orientación del objeto.
- Proporciona datos sobre la velocidad angular del objeto.

Funcionalidades del MPU6050

- Detección de movimiento y orientación.
- Estabilización de imágenes y videos.
- Control de drones y robots.
- Seguimiento de gestos y movimientos en dispositivos móviles.
- Monitoreo de actividad física en dispositivos *wearables*.
- Navegación en sistemas de posicionamiento global (GPS).

2.3.6. Bateria de 12V 3600 mah

Bateria de 12v de 3600 mah, ideal para alimentar nuestro circuito.

Figura 7: Bateria de 12v

2.3.7. Servomotor MG996R

Servo motor que será ocupado para la dirección de nuestro motor, dicho servomotor soporta hasta 10 kg y funciona con 5vdc.

Figura 8: Servmotor

Algoritmos de Control y configuración

Implementaremos los algoritmos de control para los motores y el servomotor, usando Bluetooth para enviar comandos desde una aplicación de Android. Calcular RPM, velocidad angular y velocidad lineal a partir de las lecturas de los encoders.

Despues definiremos las configuraciones geométricas del sistema de dirección Ackermann y los ángulos de las ruedas, para garantizar un movimiento preciso y menor desgaste en las ruedas.

Teorema de $\tan^{-1}(2)$

La función $\tan^{-1}(2)$ o $\tan^{-1}(y, x)$, también conocida como la función arcotangente de dos parámetros, es una función matemática que calcula el ángulo θ entre el eje positivo de las x y el punto (x, y) en el plano cartesiano. Este ángulo θ se expresa en radianes y está limitado a un rango de $-\pi$ a π.

$\tan^{-1}(y, x)$ se utiliza para convertir coordenadas cartesianas (x, y) en coordenadas polares (r, θ), donde:

- r = distancia desde el origen hasta el punto (x, y)
- θ = es el angulo entre el x positivo y la linea que conecta con el origen en el punto (x, y).

Esta función toma en cuenta el signo de ambos parámetros para determinar el cuadrante correcto del angulo.

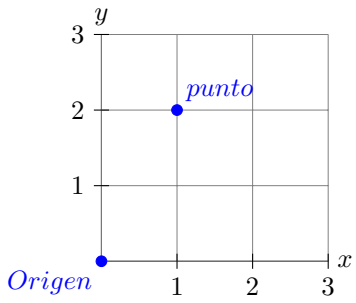


Figura 9: Plano cartesiano en el primer cuadrante de $\tan^{-1}$

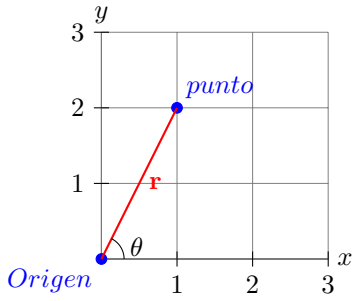


Figura 10: Plano cartesiano en el primer cuadrante de $\tan^{-1}(y, x)$

Las figuras muestran un plano cartesiano donde se representa un punto en el primer cuadrante y la línea que une el origen con dicho punto. Usamos la función atan2(y,x) para calcular el ángulo de esta línea respecto al eje x positivo, ya que atan2 considera correctamente el signo de ambos parámetros, permitiendo identificar el cuadrante exacto. Esto es fundamental en la dirección Ackermann de un coche, porque necesitamos conocer con precisión la orientación del vehículo para controlar adecuadamente el giro de las ruedas y su trayectoria.

Donde podemos aplicar este teorema:

- **Gráficos por computadora:** Para rotar objetos en un plano.
- **Robótica:** Para determinar la orientación de un robot en un espacio bidimensional.
- **Navegación:** Para calcular la dirección entre dos puntos en un mapa

2.4. Dirección Ackerman

La ecuación para calcular el ángulo de dirección Ackerman en un sistema de dirección de una plataforma móvil programable se basa en la geometría de la configuración de las ruedas. La idea principal detrás del diseño de Ackermann es que las ruedas delanteras estén alineadas de manera que todos los radios de giro de las ruedas se crucen en el centro de rotación del vehículo, lo que minimiza el deslizamiento y el desgaste de las ruedas.

Este diseño se basa en la geometría de la configuración de las ruedas, donde los radios de giro de todas las ruedas se cruzan en un punto común, optimizando así la maniobrabilidad del vehículo. La fórmula básica para el ángulo de dirección Ackerman para las ruedas delanteras utilizando el teorema de $\tan^{-1}(y, x)$ para obtener el ángulo de giro más exacto se muestra en la ecuación 1:

$$\theta_i = \tan^{-1}\left(\frac{L}{R_i}\right) \tag{1}$$

donde:

- θ_i es el ángulo de dirección de la rueda delantera medido en grados
- i (donde i puede ser 1 o 2 para las ruedas del lado izquierdo o derecho, respectivamente)
- L es la distancia entre el eje de las ruedas delanteras (ancho del tren delantero) medida en metros(m)
- R_i es el radio de giro de la rueda i medida en metros(m)

Para un diseño de dirección Ackerman ideal, las ruedas delanteras deben estar orientadas de manera que todos los radios de giro de las ruedas se crucen en el mismo punto en el suelo. Esto asegura que las ruedas interiores y exteriores del vehículo giren con los ángulos adecuados para el radio de giro deseado.

Para calcular la velocidad lineal y angular de un vehículo que utiliza una dirección Ackerman, primero debemos entender cómo las ruedas afectan el movimiento del vehículo. Aquí está el proceso:

La velocidad lineal (v) del vehículo se calcula como en la ecuación 2:

$$v = r \cdot \omega \tag{2}$$

donde:

- r es el radio de giro del vehículo
- ω es la velocidad angular de rotación del vehículo alrededor del centro de giro

La velocidad angular (ω) del vehículo se relaciona con el radio de giro (r) y la velocidad lineal (v) en la ecuación 3:

$$\omega = \frac{v}{r} \tag{3}$$

Cálculo del Radio de Giro

Para calcular el radio de giro usando $\tan^{-1}(y, x)$, necesitamos expresar la relación en términos de coordenadas cartesianas (x, y). Podemos considerar:

- x como la diferencia en la posición horizontal
- y como la diferencia en la posición vertical

La ecuación para el radio de giro, llamada Ackermann, se representa en la ecuación 4:

$$\text{Ackermann} = \tan^{-1}\left(\frac{d_1}{\frac{d_1}{\tan(\beta)} - d_2}\right) \tag{4}$$

donde:

- d_1 es la distancia entre los ejes delantero y trasero del vehículo
- d_2 es la distancia entre las ruedas del mismo eje
- β es el ángulo de giro interior de la rueda

Esta ecuación asegura que las ruedas del vehículo giren alrededor de un punto común, lo que mejora la estabilidad y reduce el desgaste de los neumáticos.

Ángulo de giro de rueda interior

$$\beta = \tan^{-1}\left(\frac{L}{r - \frac{W}{2}}\right) \tag{5}$$

donde:

- L es la distancia entre ejes
- r es el radio de giro
- W es la distancia entre las ruedas del mismo eje (vía)

Ángulo de giro de rueda exterior

$$\alpha = \tan^{-1}\left(\frac{L}{r + \frac{W}{2}}\right) \tag{6}$$

donde:

- L es la distancia entre ejes (batalla)
- r es el radio de giro
- W es la distancia entre las ruedas del mismo eje (vía)

Porcentaje de Ackermann
El porcentaje de Ackermann se puede calcular como en la ecuación 7:

$$\text{Porcentaje} = 100 \cdot \left(\frac{\alpha}{\text{Ackermann}}\right) \tag{7}$$

Un 100 % de Ackermann significa que las prolongaciones de las bielas se cortan en el centro del eje trasero. En caso de que el porcentaje sea superior a 100, éstas se cortarán por delante del eje trasero; y detrás si es inferior.

2.5. Modelo matemático de estacionamiento en paralelo.

Para la plataforma movil tenemos una limitación mecánica por lo cual el angulo menor que se puede utilizar en el modelo es de 38 grados en la llanta interior de acuerdo al giro: esta limitación de angulo se obtiene con un servomotor el cual para obtener esos 38 grados se define por una escritura pwm en el servo con un angulo igual a 15 grados, que en traducción haciendo referencia a que los 90 grados son los 0 grados entonces el servomotor gira 75 grados hacia la derecha de acuerdo a su construcción teniendo en cuenta que ahora los 90 grados es el origen del giro.

Dado a que el carro puede alcanzar un ackermann al 100 % entonces las formulas del modelo se mantienen por lo cual:

$$\cot(\delta_L) - \cot(\delta_R) = \frac{TW}{WB} \tag{8}$$

$$\delta_{\text{Ack}} = \frac{\delta_p}{\gamma} \tag{9}$$

$$\delta_L = \tan^{-1}\left(\frac{WB \cdot \tan(\delta_{\text{Ack}})}{WB + 0{,}5 \cdot TW \cdot \tan(\delta_{\text{Ack}})}\right) \tag{10}$$

$$\delta_R = \tan^{-1}\left(\frac{WB \cdot \tan(\delta_{\text{Ack}})}{WB - 0{,}5 \cdot TW \cdot \tan(\delta_{\text{Ack}})}\right) \tag{11}$$

Por lo tanto una función que dependa de δ_{Ack} tal que nos proporcione los valores de los ángulos de las llantas seria:

$$(\delta_L, \delta_R) = f(\delta_{\text{Ack}}) \tag{12}$$

$$f(\delta_{\text{Ack}}) = \tan^{-1}\left(\frac{WB \cdot \tan(\delta_{\text{Ack}})}{WB \pm 0{,}5 \cdot TW \cdot \tan(\delta_{\text{Ack}})}\right) \tag{13}$$

En la cual la función radica en el mas menos para entregar los valores correspondientes para cada llanta, y sin importar el giro la evaluación en adición el angulo siempre sera correspondiente a la llanta del lado izquierdo y la evaluación en sustracción siempre sera correspondiente al lado derecho.

En donde:

- δ_L Angulo de la llanta Izquierda.
- δ_R Angulo de la llanta Derecha.
- TW Ancho de vía.
- WB Distancia entre ejes.
- δ_{Ack} Angulo Ackermann.
- δ_p Angulo del piñón.
- γ Relación de dirección.

Teniendo lo anterior en cuenta procedemos a sacar R_c la cual nos describirá el radio con el que debemos trabajar, por lo tanto:
La relación γ esta definida por la relación del angulo de giro del servomotor y el angulo de giro de la rueda central, al no tener rueda central, podemos optar por usar la función inversa, la cual parte de:

$$f(\delta_{\text{Ack}}) = \tan^{-1}\left(\frac{WB \cdot \tan(\delta_{\text{Ack}})}{WB \pm 0{,}5 \cdot TW \cdot \tan(\delta_{\text{Ack}})}\right) \tag{14}$$

$$(\delta_L, \delta_R) = \tan^{-1}\left(\frac{WB \cdot \tan(\delta_{\text{Ack}})}{WB \pm 0{,}5 \cdot TW \cdot \tan(\delta_{\text{Ack}})}\right) \tag{15}$$

Una vez teniendo la ecuación igualada a los valores que necesitamos encontrar comenzamos con el despeje:

$$\tan(\delta_L, \delta_R) = \frac{WB \cdot \tan(\delta_{\text{Ack}})}{WB \pm 0{,}5 \cdot TW \cdot \tan(\delta_{\text{Ack}})} \tag{16}$$

$$\tan(\delta_L, \delta_R)\,(WB \pm 0{,}5 \cdot TW \cdot \tan(\delta_{\text{Ack}})) = WB \cdot \tan(\delta_{\text{Ack}}) \tag{17}$$

$$\frac{WB \cdot \tan(\delta_L, \delta_R) \pm 0{,}5 \cdot TW \cdot \tan(\delta_{\text{Ack}}) \cdot \tan(\delta_L, \delta_R)}{WB \cdot \tan(\delta_{\text{Ack}})} = 1 \tag{18}$$

$$\frac{WB \cdot \tan(\delta_L, \delta_R)}{WB \cdot \tan(\delta_{\text{Ack}})} \pm \frac{0{,}5 \cdot TW \cdot \tan(\delta_L, \delta_R)}{WB} = 1 \tag{19}$$

$$\frac{\tan(\delta_L, \delta_R)}{\tan(\delta_{\text{Ack}})} = 1 \mp \frac{0{,}5 \cdot TW \cdot \tan(\delta_L, \delta_R)}{WB} \tag{20}$$

$$\frac{1}{\tan(\delta_{\text{Ack}})} = \frac{1}{\tan(\delta_L, \delta_R)} \pm \frac{0{,}5 \cdot TW}{WB} \tag{21}$$

Hacemos una sustitución trigonométrica equivalente a $1/tan()$ lo que nos dejaría como:

$$\cot(\delta_{\text{Ack}}) = \cot(\delta_L, \delta_R) \pm \frac{0{,}5 \cdot TW}{WB} \tag{22}$$

$$\delta_{\text{Ack}} = \cot^{-1}\left(\cot(\delta_L, \delta_R) \pm \frac{0{,}5 \cdot TW}{WB}\right) \tag{23}$$

$$\delta_{\text{Ack}} = f^{-1}(\delta_L, \delta_R) \tag{24}$$

$$f^{-1}(\delta_L, \delta_R) = \cot^{-1}\left(\cot(\delta_L, \delta_R) \pm \frac{0{,}5 \cdot TW}{WB}\right) \tag{25}$$

En la ecuación 25 sustituimos el valor de $\frac{TW}{WB}$ por la ecuación igualitaria (8) dejándonos así:

$$f^{-1}(\delta_L, \delta_R) = \cot^{-1}\left(\cot(\delta_L, \delta_R) \pm 0{,}5 \cdot (\cot(\delta_L) - \cot(\delta_R))\right) \tag{26}$$

$$f^{-1}(\delta_L, \delta_R) = \cot^{-1}\left(\cot(\delta_L, \delta_R) \pm 0{,}5 \cdot \cot(\delta_L) - 0{,}5 \cdot \cot(\delta_R)\right) \tag{27}$$

$$f^{-1}(\delta_L) = \cot^{-1}\left(-0{,}5 \cdot \cot(\delta_L) + 0{,}5 \cdot \cot(\delta_R)\right) \tag{28}$$

$$f^{-1}(\delta_R) = \cot^{-1}\left(0{,}5 \cdot \cot(\delta_L) + 0{,}5 \cdot \cot(\delta_R)\right) \tag{29}$$

Por lo tanto:

$$f^{-1}(\delta_L) = -0{,}5 \cdot \delta_L + 0{,}5 \cdot \delta_R \tag{30}$$

$$f^{-1}(\delta_R) = 0{,}5 \cdot \delta_L + 0{,}5 \cdot \delta_R \tag{31}$$

La cual evaluamos con los valores de nuestro auto Ackermann corresponderían:

- $cot(\delta_L) = 26°$
- $cot(\delta_R) = 45°$

Por lo cual la evaluación correspondiente para cada una de ellas seria:

$$f^{-1}(\delta_L) = -0{,}5 \cdot \delta_L + 0{,}5 \cdot \delta_R \tag{32}$$

$$f^{-1}(\delta_R) = 0{,}5 \cdot \delta_L + 0{,}5 \cdot \delta_R \tag{33}$$

$$f^{-1}(\delta_L) = 22{,}5 - 13 \tag{34}$$

$$f^{-1}(\delta_R) = 22{,}5 + 13 \tag{35}$$

$$f^{-1}(\delta_L) = 9{,}5 \tag{36}$$

$$f^{-1}(\delta_R) = 35{,}5 \tag{37}$$

Como se puede ver al no ser una función lineal los el valor mas aproximado es de 35.5, por lo tanto procederemos a utilizar una función estable de un solo angulo la cual es:

$$f = \tan^{-1}\left(\frac{d_1}{\frac{d_1}{\tan(\beta)} - d_2}\right) \tag{38}$$

2.6. Nodos y Tópicos

Nodo	**Tipo**	**Descripción**
esp32_sensores	Micro-ROS	Adquiere datos de sensores
esp32_control	Micro-ROS	Controla motores y dirección
parking_controlador_node	ROS 2	Lógica de estacionamiento
gazebo_sim_node	Gazebo	Simulación
Rviz_node	RViz	Visualización de datos

Cuadro 1: Nodos principales del sistema

Nodo	**Tipo (Pub/Sub)**	**Tópicos**	**Tipo de Mensaje**
esp32_sensores_node	Publicador	/distancia/frontal(p)	sensor_msgs/Range
		/distancia/trasera(p)	sensor_msgs/Range
		/llantas/odometria(p)	nav_msgs/Odometry
		/imu/data_raw(p)	sensor_msgs/Imu
		/servo/angulo(p)	std_msgs/Float32
parking_controlador_node	Publicador/Suscriptor	/distancia/frontal(s)	sensor_msgs/Range
		/distancia/trasera(s)	sensor_msgs/Range
		/llantas/odometria(s)	nav_msgs/Odometry
		/imu/data_raw(s)	sensor_msgs/Imu
		/servo/data$_raw(s)$	std_msgs/Float32
		/velocidad/angular(p)	geometry_msgs/Twist
		/velocidad/lineal(p)	geometry_msgs/Twist
		/direccion/angulo(p)	std_msgs/Float32
		/control/velocidad(p)	geometry_msgs/Twist
esp32_control_node	Suscriptor	/velocidad/angular(s)	geometry_msgs/Twist
		/velocidad/lineal(s)	geometry_msgs/Twist
		/direccion/angulo(s)	std_msgs/Float32
gazebo_ros_node	Suscriptor	/direccion/angulo(s)	std_msgs/Float32
		/imu/data_raw(s)	sensor_msgs/Imu
		/control/velocidad(s)	geometry_msgs/Twist
		/simulacion/odometria(p)	nav_msgs/Odometry
rviz_node	Suscriptor	/simulacion/odometria(s)	nav_msgs/Odometry

Cuadro 2: Diseño de nodos y tópicos para el sistema mecatrónico

En la tabla mostrada anteriormente se pueden observarr los nodos que serán utilizados en nuestro proyecto, destacando cuáless son publicadores y cuáless sonn suscriptores. De igual manera en la tercera columna de izquierda a derecha se muestran los tópicos de cada uno donde podemos destacar un (p) para el cual es un tópico el cual es publicado y una (s) para el tópico al cual se suscribe dicho nodo, finalmente en la ultima columna de nuestra tabla podemos ver el tipo de mensaje para cada tópico.

2.7. Micro-ROS

Micro-ROS es una herramienta necesaria para la comunicación entre un microcontrolador, como ESP32, y ROS2. Con esto podemos implementar las herramientas como RViz y Gazebo que permiten la visualización y la obtención de datos obtenidos de sensores. Este ejercicio se realizó en Ros 2 Humble, si deseas replicarlo hay que cambiar el nombre de dependencias a tu ROS instalado.
Se usará para:

- Comunicación entre nodos, establecidos en el cuadro 2 de la sección 2.6, como publicadores y subscriptores.
- Representación gráfica del modelo físico al gemelo virtual, en los cuales se realizará el movimiento.

Para la instalación de Micro-ROS con FreeRTOS que es el agente que permite la comunicación entre el ESP32 y ROS2, con los cuales obtenemos la lectura de sensores y de actuadores. La comunicación entre el ESP32 y el agente se basa por comunicación serial o por Wifi.

2.8. Lógica de estacionamiento

En el diseño del modelo de estacionamiento automático optamos por aplicar una serie de pasos y de condiciones, unas de las condiciones es pensar en diversificar los puntos de control en una rutina, de este modo empezamos la configuración con los sensores; el sensor ultra sónico acoplado en el chasis del automóvil, este se centrara en medir la diferencia de la distancia, dicha diferencia debe estar siendo calculada constantemente, y por consecuencia publicar el/los datos correspondientes.

Después estableceremos una velocidad para el automóvil, a la cual se suscribirá el controlador de los motores con encoder, los cuales a partir de los cálculos de radio y revoluciones por minuto de la rueda se seguirá la velocidad establecida en el publicador.

Así mismo utilizaremos un tercer controlador el cual se encargara de a través de una diferencia en el tiempo y suscrito a la diferencia de distancia del primer controlador y a la velocidad del automóvil podrá hacer cálculos en paralelo los cuales estarán orientados exclusivamente a determinar si el espacio calculado es el necesario para que el automóvil efectúe el intento de estacionamiento en paralelo, si el espacio no es el necesario el automóvil seguirá avanzando, en cambio si cumple las condiciones de espacio el automóvil empezara la etapa de estacionamiento automático.

Para esta secuencia se utilizaran los cálculos previamente realizador los cuales describen una línea a seguir, gracias a estos cálculos se configuraran los parámetros correspondientes para realizar un correcto seguimiento del calculo de estacionamiento, los cuales activaran el servomotor (el cual controla la dirección del carro) dándole el ángulo de giro correspondiente, se recorre una primera distancia para entrar al cajón de estacionamiento de reversa, después girara el servomotor para realizar un ajuste en la posición del carro dejando al carro estacionado y en la posición inicial.

Observe que las distancias se medirán utilizando el segundo y tercer controlador previamente mencionados en la etapa de espacio necesario.

2.9. Arquitectura de Funcionamiento

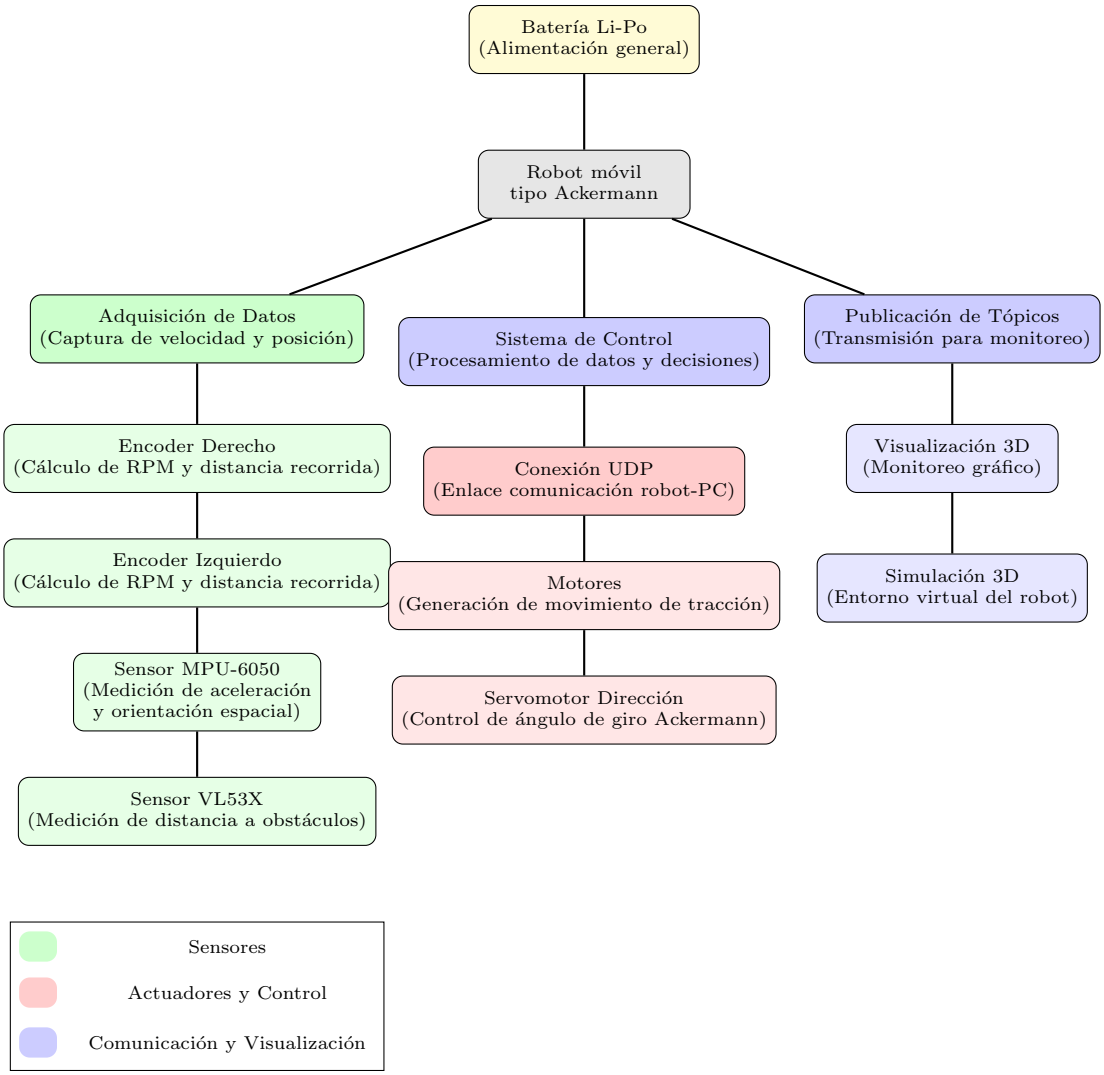


Figura 11: Diagrama del Sistema de Adquisición de datos y control del Robot Ackermann.

2.10. Ecuaciones y cálculos de velocidad desde pulsos de encoder:

Para el cálculo de las revoluciones, se investigó la hoja de datos de nuestros motores la cual nos arrojó que nuestros encoders de efecto Hall nos proporcionan once pulsos por revolución.
Teniendo en cuenta el valor de los pulsos por revolución de nuestros motores y teniendo en cuenta la reación reductora de ambos motorreductores de nuestros motores de 12v DC que es de 1:90; quiere decir que por cada 90 vueltas de nuestros motores, el eje central de la caja reductora dará una vuelta. Gracias los pulsos por revolución y la relación de la caja reductora, podemos establecer el valor de los pulsos por revolución del eje de la caja recutora:

$$90 \cdot 11 = 990 \tag{39}$$

- 90 = Relación reducora de la caja 1:90
- 11 = Pulsos por revolución
- 990 = Pulsos por revolución de la caja reductora

En la **ecuación (8)** se obtuvo el valor de los pulsos por revolución de un solo encoder, sin embargo, si medimos el cambio, el cual mide subidas y bajadas, este valor se duplicará, entonces obtenemos la **ecuación(9)**:

$$RPM = \frac{Pulsos \cdot 60}{990} \tag{40}$$

Donde:

- RPM = revoluciones por minuto
- 60 = 60 segundos que hay en un minuto
- 990 = Pulsos por revolución de la caja reductora

Una vez conocido el valor de las RPM gracias a la **ecuación (9)**, se puede medir la velocidad angular mediante la **ecuación (10)**:

$$\omega = \frac{2 \cdot \pi \cdot RPM}{60} \tag{41}$$

Donde:

- ω = Velocidad angular medida en radianes/segundo(rad/s).
- π = pi 3.1416
- 60 = segundos de un minuto
- RPM = Revoluciones por minuto

Finalmente, gracias a la **ecuación (10)** podemos deducir la ecuación para obtener la velocidad lineal de nuestro robot mediante la siguiente ecuación:

$$V = \omega \cdot r \tag{42}$$

Donde:

- V = velocidad lineal en centimetros/segundos(cm/s)
- ω = velocidad angular
- r = radio de la rueda de nuestros motores medida en centimetros

Consideraciones: Las ecuaciones (9 - 11) están dadas para el cálculo de 2 encoders, midiendo el cambio de la señal, sin embargo, habrá dos encoders por cada motor, por ende habrá 4 encoders, pero usarlos todos será opcional, sin embargo, en caso de usarlos, se puede realizar un promedio de ambas señales para obtener un valor general de pulsos y por ende, velocidad angular y velocidad lineal.

2.11. Modelo CAD

El modelo CAD (Diseño Asistido por Computadora) es una versión en 3D del sistema que estamos creando. Lo hicimos para tener una idea clara de cómo se vería el producto final. Para diseñarlo, usamos el programa SolidWorks y ahí armamos todas las piezas necesarias. Primero, medimos las partes físicas para poder dibujarlas en SolidWorks.
El modelo CAD incluye varios subensamblajes importantes, como la base, los actuadores, las placas electrónicas (PCBs) y los mecanismos de transmisión. Diseñamos cada parte pensando en las limitaciones físicas y lo que necesitaba el proyecto.
También usamos la página GrabCAD para descargar algunas piezas extras y así completar el diseño, lo que nos ayudó a ahorrar tiempo.
Además, usamos la página web GrabCAD para descargar algunas piezas adicionales y así completar el diseño final del ensamblaje, ahorrando tiempo y facilitando el proceso.
Las medidas para recrear las bases del robot móvil están en la sección de Anexos, en la **figura 30.** Además, en **la figura 28** de los Anexos, se muestra cómo se ven la dirección y los motores en el circuito real.

Modelo CAD del Sistema de Dirección Ackermann

Figura 12: Chasis

2.12. URDF

Unified Robot Description Format (URDF) que es un archivo XML que permite describir un robot. Para exportar a Gazebo, donde se llevará acabo el Gemelo Virtual, es necesario exportar en formato URDF del sistema realizado en SolidWorks.
Se debe identificar los "linksz "joints.º eslabones y articulaciones del modelo, para ello se debe de seleccionar "base link"que es lo que conforma todo lo del chasis de la plataforma móvil, incluyendo motores y el servomotor.
Una vez realizado el "base link"se deben de realizar la selección de "hijos"que estos son los que permitirán los movimientos continuos para el giro de las cuatro ruedas y dos movimientos de revolución el cual permitirán hacer el movimiento de Ackerman.
Para este trabajo se nombraron los links y joints de la siguiente manera:

Ruedas traseras – Movimiento continuo:

- Rueda-derecha-trasera-link
- Rueda-derecha-trasera-joint
- Rueda-izquierda-trasera-link
- Rueda-izquierda-trasera-joint

Ruedas delanteras – Movimiento de revolución:

- Giro-derecho-delantera-link
- Giro-derecho-delantera-joint
- Giro-izquierdo-delantera-link
- Giro-izquierdo-delantera-joint

Teniendo los movimientos de revolución, se deben aplicar como hijos las ruedas derecha e izquierda.

Ruedas delanteras – Movimiento continuo:

- Rueda-derecha-delantera-link
- Rueda-derecha-delantera-joint
- Rueda-izquierda-delantera-link
- Rueda-izquierda-delantera-joint
- Como paso fundamental se debe tener el centro de masa de la plataforma se encuentre ubicado en el centro del chasis, como se muestra a continuación.

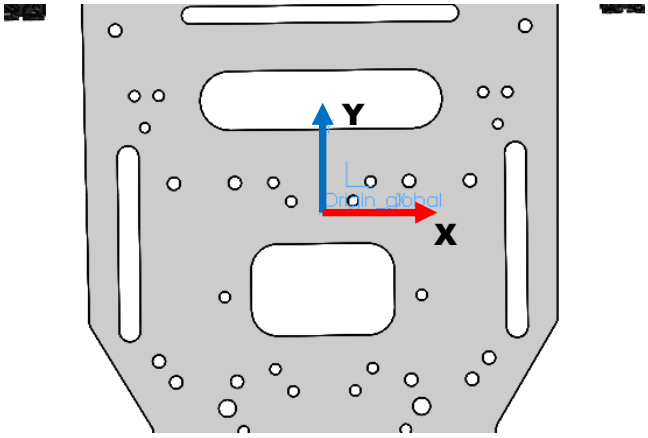


Figura 13: Centro de masa de Plataforma móvil

- Como segundo paso, es necesario definir un centro de cara en cada una de las 4 ruedas de la plataforma móvil, con el cual obtendremos el marco coordenado con respecto a la regla de la mano derecha. De esta manera se puede determinar el eje de rotación y su giro de cada rueda.

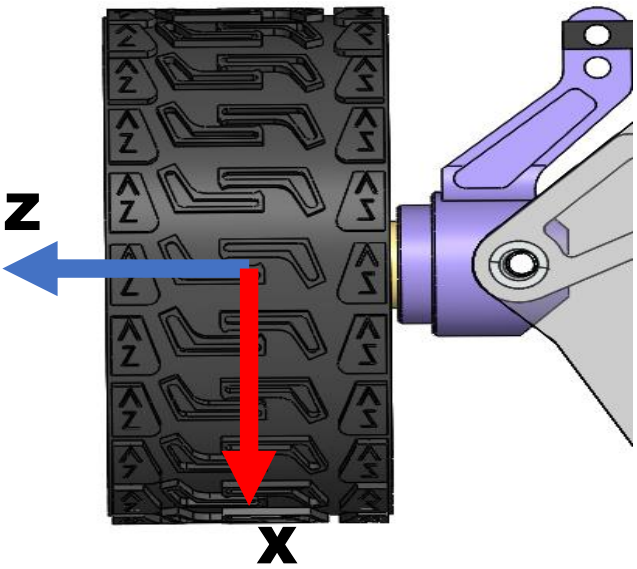
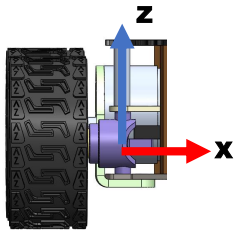
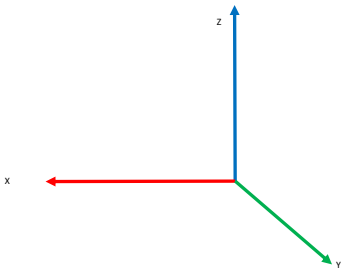


Figura 14: Centro de cara de la rueda.

- Tercer paso, para las llantas delanteras, además de contar con el marco coordenado de las ruedas, también se necesita ubicar un segundo marco coordenado en el brazo de dirección, el cual es el que permite el movimiento del giro.



(a) Mambos ejes, en la rueda y el en el brazo donde se realizará el movimiento de giro



(b) Marco coordenado para movimiento de Ackermann

Figura 15: Representación de marcos coordenados aplicados a las ruedas y brazo de giro del sistema móvil.

Una vez realizado estos pasos ya se podrá obtener el archivo que permitirá la visualización del urdf. Debemos tener en cuenta que hay que poner los marcos coordenados correctamente, poner los límites en los que se pueden rotar o girar las ruedas, todo esto para poder simular correctamente el modelo en un entorno de software, como Gazebo.

3. Anexos

La instalación de este agente se realizó siguiendo los pasos del repositorio oficial de ROS "primera aplicación con micro-Ros con FreeRTOS
Los pasos realizados en la terminal son los siguientes:
Activación de la instalación de ROS 2 en la terminal, este comando funge como señal para usar ROS 2.

```
source /opt/ros/$ROS_DISTRO/setup.bash
```

Creación de carpeta de trabajo, donde se realizará el trabajo con respecto a Micro-ros.

```
mkdir microros_ws
```

Este comando permite meternos a la carpeta creada anteriormente.

```
cd microros_ws
```

Permite la descarga de herramientas y librerías de Github, de la versión en la se esta trabajando, este caso la utilización de "Humble"

```
git clone -b $ROS_DISTRO https://github.com/micro-ROS/micro_ros_setup.git src/
    micro_ros_setup
```

Sudo apt update funge como la actualización de paquetes de Ros2, mientras que rosdep update funge para para ctualizar la herramienta rosdep, que es la que permite la instalación de dependencias de ROS.

```
sudo apt update && rosdep update
```

Este comando permite instalar todos los paquetes necesarios para que realice el correcto funcionamiento de librerías.

```
Rosdep install --from-paths src --ignore-src -y
```

Instalación de Pip, que permite gesionar y descargar librerías de Python para la utilización de Micro-ROS

```
sudo apt-get install python3-pip
```

Comando que permite la creación de todo lo realizado con anterioridad en el espacio de trabajo creado üros_ws"

```
colcon build
```

Para inicializar el espacio de trabajo creado de micro-ROS, junto con la implementación de paquetes de ROS2

```
source install/local_setup.bash
```

Ya una vez realizado los pasos anteriores, se realizará lo siguiente.
En este paso, se crea un espacio de trabajo para firmware, que es el que permitirá la comunicación entre ESP32 y ROS 2, el cual permite leer sensores o controle actuadores. Este paso es tardado, puesto que instala los paquetes necesarios para su funcionamiento.

```
ros2 run micro_ros_setup create_firmware_ws.sh freertos esp32
```

Una vez ya descargado todos los paquetes, se procede a gestionar los archivos instalados.

```
cd microros_ws
/microros_ws/ ls
/microros_ws/ls/firmaware
/microros_ws/ls/firmaware/ ls
/microros_ws/ls/firmaware/ cd freertos_apps/
/microros_ws/ls/firmaware/freertos_apps/ ls
/microros_ws/ls/firmaware/freertos_apps/ cd apps/
```

Estando en esta sección deberá aparecer aparecer las aplicaciones para la realización en este caso guiados de la pagina oficial de ROS, el ejemplo "Ping pong".

```
/microros_ws/ls/firmaware/freertos_apps/apps/ ls
```

Comando a realizar para configurar la aplicación por via puerto serial y lograr la comunicación entre ESP32 Y Micro ROS.

```
ros2 run micro_ros_setup configure_firmware.sh ping_pong --transport serial
```

En este paso construimos el firmware para usar el protocolo "ping pong", enviando y recibiendo información.

```
ros2 run micro_ros_setup build_firmware.sh
```

Paso de flash, en este paso tenemos que tener conectado el ESP32 a la computadora por puerto USB

```
ros2 run micro_ros_setup flash_firmware.sh
```

Descargamos todos los paquetes del agente de Micro-ROS para la comunicación con ROS 2. Estos paquetes son bastantes, así que puede variar el tiempo en su descarga.

```
ros2 run micro_ros_setup create_agent_ws.sh
```

Se construyen los paquetes descargados del agente.

```
ros2 run micro_ros_setup build_agent.sh
```

Activación de paquetes descargados.

```
source install/local_setup.bash
```

Antes de corroborar el funcionamiento de este firmware, se debe de poner el siguiente comando, para saber con exactitud la dirección en el que esta conectado el ESP32.

```
/dev/serial/by-id/*
```

En este caso a nosotros nos arrojo la siguiente ubicación en donde esta conectado el ESP32.

```
/dev/serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-
    port0
```

Ya obtenido la dirección de conexión del ESP32, se debe poner el siguiente comando.

```
ros2 run micro_ros_agent micro_ros_agent serial --dev
```

Juntamos la dirección obtenida con el comando anterior, en este paso debemos de presionar el botón BOOT del ESP32 para acompletar la conexión. Realizado este comando debe de haber conexión establecida.

```
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/serial/by-id/usb-
    Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0
```

Para confirmar la comunicación se debe de abrir otra terminal y correr el siguiente comando.

```
ros2 topic list
```

Una vez puesto ese comando, deben aparecer los siguientes Tópicos y eso es señal de una comunicación entre ESP32 y Micro-Ros.

```
/microRos/ping
/microRos/pong
```

3.1. Codigo de Microros:

```
#include <micro_ros_arduino.h>
#include <rcl/rcl.h>
#include <rclc/rclc.h>
#include <rclc/executor.h>
#include <geometry_msgs/msg/twist.h>
#include <std_msgs/msg/float32.h>
#include <sensor_msgs/msg/imu.h>
#include <ESP32Servo.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

#define SERVO_PIN 13
#define TRIG_PIN 12
#define ECHO_PIN 14
#define ENCODER_PIN 15

#define ENA 32
#define ENB 33
#define IN1 2
#define IN2 4
#define IN3 16
#define IN4 17

const int VELOCIDAD_LENTA = 200;
const int PULSOS_POR_VUELTA = 1346;
const float DISTANCIA_CM_POR_PULSO = 21.99 / PULSOS_POR_VUELTA;

const int PULSOS_DIAGONAL = 400;
const int PULSOS_RECTO = 350;
const int PULSOS_AJUSTE = 400;
const int PULSOS_AVANCE = 50;

volatile long contador_pulsos = 0;
Servo direccion;
Adafruit_MPU6050 mpu;
bool maniobra_realizada = false;

rcl_publisher_t twist_publisher;
rcl_publisher_t distance_publisher;
rcl_publisher_t imu_publisher;
geometry_msgs__msg__Twist twist_msg;
std_msgs__msg__Float32 distance_msg;
sensor_msgs__msg__Imu imu_msg;
rclc_executor_t executor;
rclc_support_t support;
rcl_allocator_t allocator;
rcl_node_t node;

void IRAM_ATTR contarPulsos() {
  contador_pulsos++;
}

void setupMicroROS() {
  set_microros_transports();
  allocator = rcl_get_default_allocator();
  rclc_support_init(&support, 0, NULL, &allocator);
  rclc_node_init_default(&node, "esp32_robot", "", &support);
  rclc_publisher_init_default(&twist_publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(
    geometry_msgs, msg, Twist), "model/robot/cmd_vel");
  rclc_publisher_init_default(&distance_publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(
    std_msgs, msg, Float32), "distancia");
  rclc_publisher_init_default(&imu_publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(
    sensor_msgs, msg, Imu), "imu/data");
  rclc_executor_init(&executor, &support.context, 1, &allocator);
}

void setupMPU6050() {
  mpu.begin();
```

```
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
}

void publishIMUData() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  int64_t time_now = rmw_uros_epoch_millis();
  imu_msg.header.stamp.sec = time_now / 1000;
  imu_msg.header.stamp.nanosec = (time_now % 1000) * 1000000;
  imu_msg.linear_acceleration.x = a.acceleration.x;
  imu_msg.linear_acceleration.y = a.acceleration.y;
  imu_msg.linear_acceleration.z = a.acceleration.z;
  imu_msg.angular_velocity.x = g.gyro.x * (M_PI / 180.0);
  imu_msg.angular_velocity.y = g.gyro.y * (M_PI / 180.0);
  imu_msg.angular_velocity.z = g.gyro.z * (M_PI / 180.0);
  rcl_publish(&imu_publisher, &imu_msg, NULL);
}

void publishDistance(float distancia) {
  distance_msg.data = distancia;
  rcl_publish(&distance_publisher, &distance_msg, NULL);
}

void publicarVelocidades(float lineal, float angular) {
  twist_msg.linear.x = lineal;
  twist_msg.angular.z = angular;
  rcl_publish(&twist_publisher, &twist_msg, NULL);
}

void moverAdelanteIndefinido() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, VELOCIDAD_LENTA);
  analogWrite(ENB, VELOCIDAD_LENTA);
  publicarVelocidades(0.10, 0.0);
}

void moverAdelanteTiempo(int ms) {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, VELOCIDAD_LENTA);
  analogWrite(ENB, VELOCIDAD_LENTA);
  publicarVelocidades(0.10, 0.0);
  delay(ms);
  detenerMotores();
}

void moverAtras(long pulsos_objetivo) {
  contador_pulsos = 0;
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, VELOCIDAD_LENTA);
  analogWrite(ENB, VELOCIDAD_LENTA);
  publicarVelocidades(-0.10, 0.0);
  while (contador_pulsos < pulsos_objetivo) {
    publishIMUData();
    delay(10);
  }
  detenerMotores();
}

void detenerMotores() {
  digitalWrite(IN1, LOW);
```

```
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  publicarVelocidades(0.0, 0.0);
}

float leerDistancia() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duracion = pulseIn(ECHO_PIN, HIGH, 20000);
  float distancia = duracion * 0.034 / 2.0;
  publishDistance(distancia);
  return distancia;
}

void fase1RetrocederDiagonal() {
  direccion.write(140);
  publicarVelocidades(-0.10, 0.5);
  moverAtras(PULSOS_DIAGONAL);
}

void fase2RetrocederRecto() {
  direccion.write(90);
  publicarVelocidades(-0.10, 0.0);
  moverAtras(PULSOS_RECTO);
}

void fase3RetrocederAjustando() {
  direccion.write(40);
  publicarVelocidades(-0.10, -0.5);
  moverAtras(PULSOS_AJUSTE);
}

void fase4AvanzarCentrar() {
  direccion.write(90);
  publicarVelocidades(0.10, 0.0);
  moverAdelanteTiempo(1000);
}

void ejecutarSecuenciaEstacionamiento() {
  fase1RetrocederDiagonal();
  fase2RetrocederRecto();
  fase3RetrocederAjustando();
  fase4AvanzarCentrar();
}

void setup() {
  Serial.begin(115200);
  delay(2000);
  direccion.attach(SERVO_PIN);
  direccion.write(90);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(ENCODER_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENCODER_PIN), contarPulsos, FALLING);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  Wire.begin();
  setupMPU6050();
  setupMicroROS();
  detenerMotores();
}
```

```
209
210 void loop() {
211   static unsigned long last_imu_publish = 0;
212   if (millis() - last_imu_publish > 100) {
213     publishIMUData();
214     last_imu_publish = millis();
215   }
216   if (!maniobra_realizada) {
217     float distancia = leerDistancia();
218     if (distancia > 0 && distancia < 25) {
219       moverAdelanteTiempo(1700);
220       detenerMotores();
221       ejecutarSecuenciaEstacionamiento();
222       maniobra_realizada = true;
223     } else {
224       moverAdelanteIndefinido();
225     }
226   } else {
227     detenerMotores();
228   }
229   rclc_executor_spin_some(&executor, RCL_MS_TO_NS(100));
230 }
```

3.2. Comandos para la inicializacion en ros2:

3.3. Iniciar microros

```
1     ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/serial/by-id/usb-
      Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0
```

3.4. Iniciar gazebo

```
1     ros2 launch mobile_robot gazebo_model.launch.py
```

3.5. Vizualizar topicos de microros

```
1     ros2 topic list
```

3.6. Vizualizar sensores y actuadores

```
1     ros2 topic lecho \
```

3.7. Planos plataforma movil

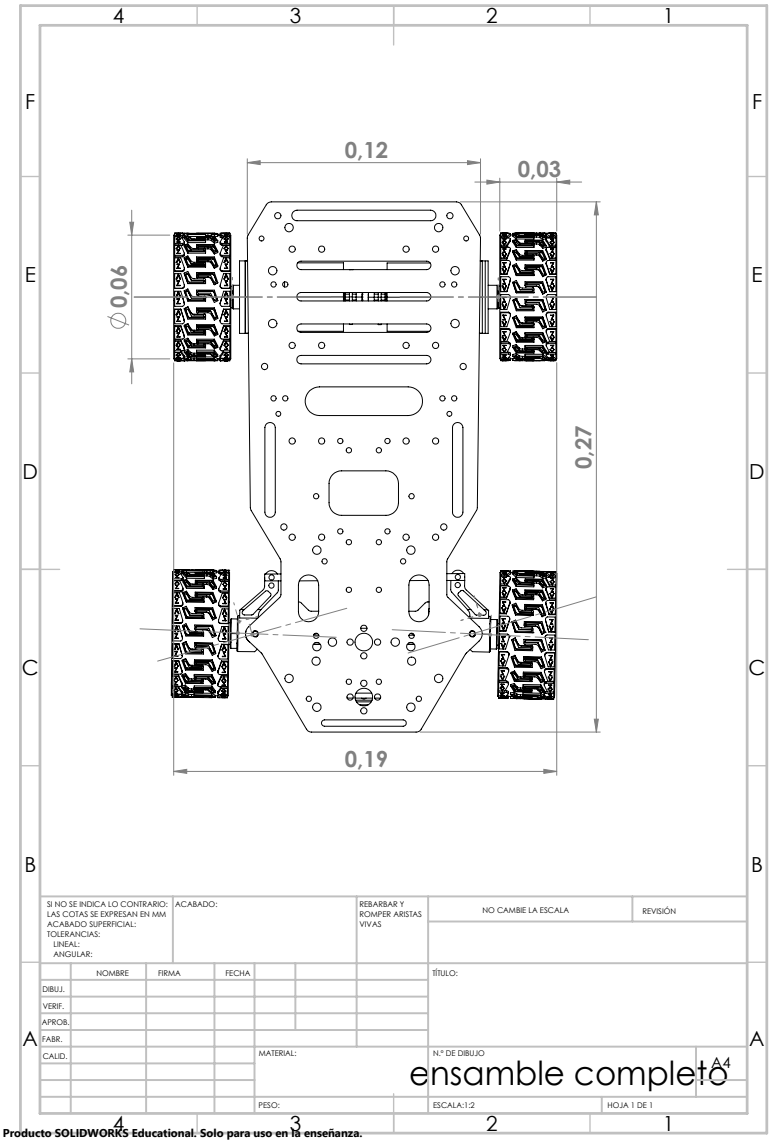


Figura 16: Planos de plataforma movil tipo ackermann

3.8. Renders

(a) Render de un servo motor MG996R hecho en solidworks

(b) Render del motor de 12v de 350 rpm hecho en solidworks

Referencias

[1] Ingenieria Y Mecanica Automotriz, "Qué es y como funciona el principio de Ackerman?," *Ingenieria Y Mecanica Automotriz*, Oct. 2019. [Online]. Available: https://www.ingenieriaymecanicaautomotriz.com/que-es-y-como-funciona-el-principio-de-ackerman/. [Accessed: Oct. 2, 2024].

[2] E. Roch Moraguez, "Todo sobre ESP32: Guía y Aplicaciones Prácticas," *LovTechnology*, Jun. 2024. [Online]. Available: https://lovtechnology.com/todo-sobre-esp32-guia-y-aplicaciones-practicas/. [Accessed: Oct. 2, 2024].

[3] J. L. R., "Baterías de Litio — Qué son, funcionamiento, partes y usos," *247Tecno*, Oct. 2019. [Online]. Available: https://247tecno.com/baterias-de-litio/. [Accessed: Oct. 2, 2024].

[4] Ingeniería Mecafenix, "Qué es el buzzer y cómo funciona (zumbador)," *Mecafenix*, Dec. 2023. [Online]. Available: https://www.ingmecafenix.com/electronica/componentes/el-buzzer/#:~:text=Un%20zumbador%20o%20mejor%20conocido%20como%20buzzer%20%28en,una%20bater%C3%ADa%20o%20cualquier%20fuente%20de%20corriente%20directa. [Accessed: Oct. 2, 2024].

[5] L. Llamas, "Determinar la orientación con Arduino y el IMU MPU-6050," *Luis Llamas*, Sep. 2016. [Online]. Available: https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/. [Accessed: Oct. 2, 2024].

[6] Open Robotics, "ROS (Robot Operating System)," *ROS.org*, 2007. [Online]. Available: https://www.ros.org. [Accessed: Apr. 4, 2025].