UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

Developing a Semiautonomous Terrestrial Vehicle

Gabriel Alejandro Gómez Guerra

Ingeniería Electrónica y Automatización

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero Electrónico

Quito, 21 de diciembre de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Developing a Semiautonomous Terrestrial Vehicle

Gabriel Alejandro Gómez Guerra

Nombre del profesor, Título académico **René Játiva E., Ph.D.**

Quito, 21 de diciembre de 2023

© DERECHOS DE AUTOR

Nombres y apellidos: Gabriel Alejandro Gómez Guerra

Código: 00332459

Cédula de identidad: 1721837084

Lugar y fecha: Quito, 21 de diciembre de 2023

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

El desarrollo de vehículos autónomos ha ido tomando importancia en la industria tecnológica debido a las potenciales ventajas que presentan, como la reducción de accidentes y la reducción de tráfico. Actualmente, el estado del arte muestra un desarrollo heterogéneo, reportándose pequeñas flotas de vehículos entregando paquetes en áreas suburbanas en vías de muy bajo tráfico y servicios semiautónomos que conmutan el control entre el usuario y su vehículo según el nivel de complejidad del escenario de tráfico, permitiendo al usuario tomar el control en cualquier momento y particularmente en condiciones de emergencia. En este trabajo se desarrolla un vehículo semiautónomo, al cual se le proporciona una ruta a partir de un sistema de coordenadas referencial. Se diseña un controlador basado en el modelo dinámico de Ackerman, el cual permite el seguimiento de la ruta programada por parte del vehículo. Además, se presenta el diseño de dos controladores PID para permitir que la velocidad lineal del vehículo y el ángulo de la dirección sean capaces de seguir la referencia generada por el controlador principal. Se utilizan dos filtros de Kalman unidimensionales para reducir la varianza de medición de los sensores de velocidad y de posición angular de la dirección. Se utiliza un BMW i8 Spyder a escala 4:1 como el robot móvil sobre el cual se diseña el control. Asimismo, se dispone de un módulo de expansión CAN, el cual le permite al procesador principal comunicarse con otros computadores para obtener información de sus alrededores tales como por ejemplo un sistema de procesamiento de imágenes en tiempo real. Finalmente se realizan pruebas de la operación del controlador en seguimiento de rutas de referencia y de la capacidad del sistema para recibir y atender mensajes por medio del protocolo CAN.

Palabras clave: Vehículo autónomo, Modelo dinámico de Ackerman, Filtro de Kalman de una dimensión, Protocolo de comunicaciones CAN, Seguimiento de trayectoria, Controlador PID, Controlador basado en álgebra lineal.

ABSTRACT

The development of autonomous vehicles has been taking more relevance because of the potential advantages they offer, as the reduction in traffic accidents and a better and smother mobility. Nowadays, the state of the art shows an heterogeneous development, reporting small fleets delivering packages on low traffic suburban areas and semiautonomous services that are able to commute control between the driver and the vehicle according to the traffic scenario complexity, allowing the driver to take control anytime, especially in emergency scenarios. In this work, a semiautonomous vehicle is developed to which a predesigned trajectory is uploaded in reference to a coordinate system. A controller is designed based in the Ackerman Dynamic model and two PID controllers are implemented to allow the linear velocity and the steering-angle to follow the reference generated by the main controller. Two one-dimensional Kalman filters are used to reduce the measure variance of the linear velocity and the steering angle of the vehicle. The prototype is a 4:1 scaled BMW i8 Spyder model. Additionally, a CAN expansion hat is installed to the main processor to allow it to communicate with a secondary processor to obtain additional information about the surroundings like for example a real time image processing unit. Finally, the controller operation using a reference trajectory and the ability to receive and react of the system to a CAN protocol message is tested.

Key words: Autonomous vehicle, Ackerman Dynamic model, One-dimensional Kalman filter, CAN communication protocol, Trajectory tracking, PID controller, Linear algebra based controller.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Un vehículo autónomo se considera un vehículo que puede percibir su entorno y tomar decisiones de acuerdo con los datos obtenidos. Esta máquina es capaz de operar sin la intervención de un usuario. El vehículo puede realizar las mismas tareas que un vehículo normal y puede ingresar a los mismos lugares que un vehículo tradicional [1].

La Sociedad de Ingenieros Automotrices (SAE) define seis niveles de automatización comenzando desde el nivel cero donde el operador humano realiza todo el control, hasta el nivel 5 en donde el vehículo ya no tendría pedales ni volante para el control externo, por lo que sería completamente autónomo [2]. Actualmente, la tecnología se encuentra entre el nivel dos y tres, el paso más drástico, en donde se salta de un manejo manual por un operario a un sistema en el cual se puede entregar el control a la computadora del vehículo, pero en el cual, el operario debe estar alerta a todo momento, para recuperar el control en el caso de situaciones de emergencia que los sistemas a bordo del vehículo no puedan manejar [3].

Este trabajo consiste en el desarrollo de un vehículo semiautónomo, el cual sea capaz de seguir una ruta de referencia programada previamente. El vehículo se puede controlar manualmente por medio de un control remoto. En este trabajo se sube el vehículo a un nivel de autonomía superior, en el cual este puede seguir una trayectoria programada sin la intervención directa de un usuario. Asimismo, se instala un módulo de comunicación CAN, el cual puede recibir información de una computadora adicional, por medio de la cual el vehículo sea capaz de tomar decisiones en dependencia de su entorno.

El dispositivo desarrollado es un vehículo tipo Ackerman, es decir, tiene el tren de potencia en el eje trasero y el tren de dirección en el delantero. Este tipo de vehículo es el que describe la dinámica de los autos tradicionales [4]. La columna de dirección del vehículo permite el

cambio de ángulo de las llantas delanteras, lo que permite el movimiento lateral del vehículo. Por otra parte, cabe destacar, que este vehículo no puede girar sin la velocidad lineal del tren de potencia del vehículo, por lo que la velocidad angular depende directamente de la velocidad lineal [4]–[10].

El controlador implementado en el vehículo se diseña por medio del modelo dinámico del vehículo basado en álgebra lineal. Este controlador utiliza la orientación del vehículo y la posición en un sistema de coordenadas para el cálculo de la velocidad lineal y el ángulo de giro de las ruedas delanteras [5]. Por otra parte, se introduce un controlador tipo PID, el cual calcula el ancho de pulso del PWM que se alimenta al motor del tren de potencia, el cual asegura que la velocidad del vehículo sea la misma que se calcula por medio del controlador [7]. Finalmente, se diseña un controlador tipo PID que permita el seguimiento del ángulo de giro calculado por el actuador del robot móvil [4], [5], [7]–[9].

Los sensores que el equipo utiliza tienen un rango de incertidumbre en la medida, por lo cual se utiliza un filtro de Kalman. Este filtro consiste en un algoritmo que utiliza medidas observadas durante un tiempo para estimar el estado de un sistema, a pesar de que las medidas sean imprecisas [11], [12]. El filtro consiste en predecir el siguiente estado del sistema a partir de un modelo dinámico, comparar el valor medido con el valor predicho y actualizar el estado del sistema en función del valor medido, la varianza de la medición y el valor predicho [11]–[13]. En este trabajo, se utiliza el filtro de Kalman con el sensor de velocidad y el sensor de posición angular de la columna de dirección. Se utiliza una versión simplificada del filtro en la que se asume que la dinámica del sistema es constante dentro del tiempo de muestreo en el que se implementa el filtro (0.1 segundos), y por tanto que el valor del estado anterior se mantiene en el próximo estado. Por otra parte, el filtro de Kalman reduce la

varianza del valor predicho por medio de las iteraciones [11]–[13]. Como el sistema en el que se implementa no se mantiene constante, sino que sufre cambios bruscos, cuando el error entre el valor medido y el valor anterior es mayor a un umbral determinado, se reestablece el valor de la varianza predicha con la varianza de la medición, con el fin de que el valor filtrado sea más cercano al valor medido que al valor filtrado en el estado anterior [12].

En este trabajo se instala un módulo de comunicación CAN al procesador principal, con el fin que este se comunique con una computadora adicional que potencialmente permita el procesamiento de imágenes en tiempo real, con el fin de detectar obstáculos y facilitar el seguimiento de una ruta exterior.

El protocolo Controller Area Network (CAN) es un estándar de comunicación serial diseñado para la industria automotriz. Este protocolo permite que varios microcontroladores puedan compartir información entre sí, por medio de un bus de datos común bidireccional [14]–[16]. Este protocolo de comunicación es de tipo transmisión, es decir, la información es transmitida por un nodo y todo el resto de los nodos la recibe. Sin embargo, solamente por el contenido del mensaje se discrimina el nodo que debe reaccionar en el sistema [14].

A continuación, se presenta la descripción del sistema físico utilizado, la descripción del sistema controlado en lazo cerrado, se introduce el modelo dinámico de Ackerman, se presenta el diseño del controlador basado en álgebra lineal, además, se describe el método de identificación de sistemas en lazo abierto. Se introduce el filtro de Kalman de una dimensión y el protocolo de comunicación CAN. También, se sintonizan los controladores PID de velocidad y de posición angular de la columna de dirección del vehículo. Finalmente se realizan pruebas de seguimiento de trayectoria y la emulación de recepción de mensajes por medio del protocolo CAN.

Objetivo general

Desarrollo de un sistema de control que permita el manejo autónomo de un vehículo terrestre.

Objetivos específicos

- Desarrollo de un sistema de control basado en álgebra lineal que permita dar seguimiento a una ruta de referencia previamente suministrada con un tiempo de muestreo fijo.
- Minimizar el error de posición del vehículo real con respecto a la ruta programada.
- Implementar el filtrado de Kalman asumiendo que la dinámica del sistema es constante entre dos estados separados por el tiempo de muestreo T0 de 0.1 segundos.
- Mitigar la incertidumbre de las medidas tomadas por medio del sensor de velocidad lineal y el sensor de posición angular de la columna de dirección.
- Utilizar los sensores de proximidad en cada tiempo de muestreo para detectar objetos que se encuentren obstaculizando el paso del vehículo.
- Detener el vehículo a una cierta distancia de un obstáculo detectado.
- Instalación de un módulo CAN para el Raspberry Pi 4b.
- Emulación de recepción de señales por medio del protocolo CAN utilizando un segundo Raspberry Pi 4b que se comunique con el procesador principal.
- Decodificación del mensaje recibido por medio del protocolo CAN para conseguir que el vehículo responda a una potencial interrupción del sistema de reconocimiento del entorno y modifique pertinentemente la ruta de navegación.

DESCRIPCIÓN DEL SISTEMA FÍSICO

En la siguiente sección se describirá el sistema utilizado durante este trabajo, lo cual incluye el vehículo utilizado, los sensores que tiene la planta, el sistema de alimentación de los sensores y los actuadores, y la conexión con el procesador principal.

Descripción del vehículo

El vehículo utilizado para el desarrollo de este proyecto es un modelo a escala 4:1 de un BMW i8 Spyder. El vehículo es un juguete que se movía por medio de un conductor que se encargaba de acelerar por medio de un pedal y de dirigir la orientación del vehículo por medio de un volante. El vehículo fue modificado para que se conduzca por medio de control remoto. El procesador utilizado es un Raspberry Pi 4B, el cual recibe las direcciones del control remoto y manda las señales correspondientes a los actuadores del vehículo. En la Figura 1 se muestra el vehículo en cuestión.

Figura 1. Modelo a escala 4:1 BWM i8 Spyder.

El vehículo cuenta con un motor DC en la rueda trasera derecha, el cual es el encargado del movimiento lineal del vehículo. Asimismo, el vehículo tiene un motor DC en el eje del volante, el cual permite controlar el ángulo de giro de las ruedas delanteras. Ambos motores son alimentados por una batería de 12 V.

La distancia entre ejes del vehículo es de 70 cm y la distancia lateral entre ruedas es de 55 cm. El ángulo máximo de giro de las ruedas es de 30° hacia cada lado. Finalmente, el diámetro de las llantas es de 27 cm. Las dimensiones del vehículo se muestran en la Figura 2.

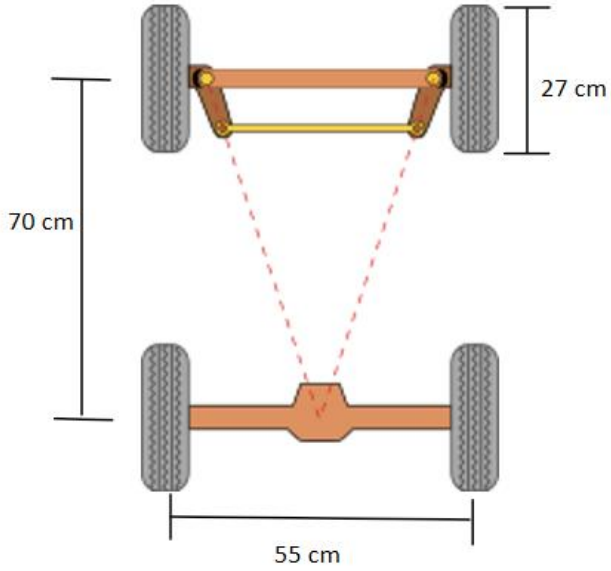


Figura 2. Dimensiones del vehículo.

Sensores

El vehículo cuenta con varios sensores que permiten la obtención de datos de posición y velocidad. Asimismo, el dispositivo cuenta con sensores de distancia ultrasónicos que

permiten obtener información de los alrededores, como la posición de obstáculos. La descripción de los sensores utilizados se encuentra en el Anexo A.

Esquema de alimentación de sensores y actuadores

Se utiliza una batería de 12 V y de 7000 mAh para alimentar los sensores descritos en la subsección anterior, además del módulo CAN que se instala en el vehículo para que sea capaz de comunicarse con un procesador adicional, y una batería de las mismas condiciones para alimentar los dos motores DC que se utilizan para el movimiento del vehículo. Se utiliza el driver Monster Moto Shield para el manejo de los motores, ya que estos requieren de una corriente alta de funcionamiento y este puede soportar hasta 14 A de corriente nominal y 30 A de corriente pico [17]. El Raspberry Pi 4B se alimenta solamente de 5 V, por lo que se utiliza un reductor de voltaje para disminuir de 12 a 5 V la salida de la batería.

El Raspberry Pi 4B alimenta a los sensores y el módulo CAN con un voltaje de 5 V y para el encoder, el potenciómetro y el módulo MCP3008 se utiliza un reductor de voltaje a 3.3V para asegurar que la salida de estos sea de máximo 3.3 V o los pines del GPIO del Raspberry se pueden averiar.

El diagrama de conexión simplificado de alimentación de los dispositivos se muestra en la Figura 3.

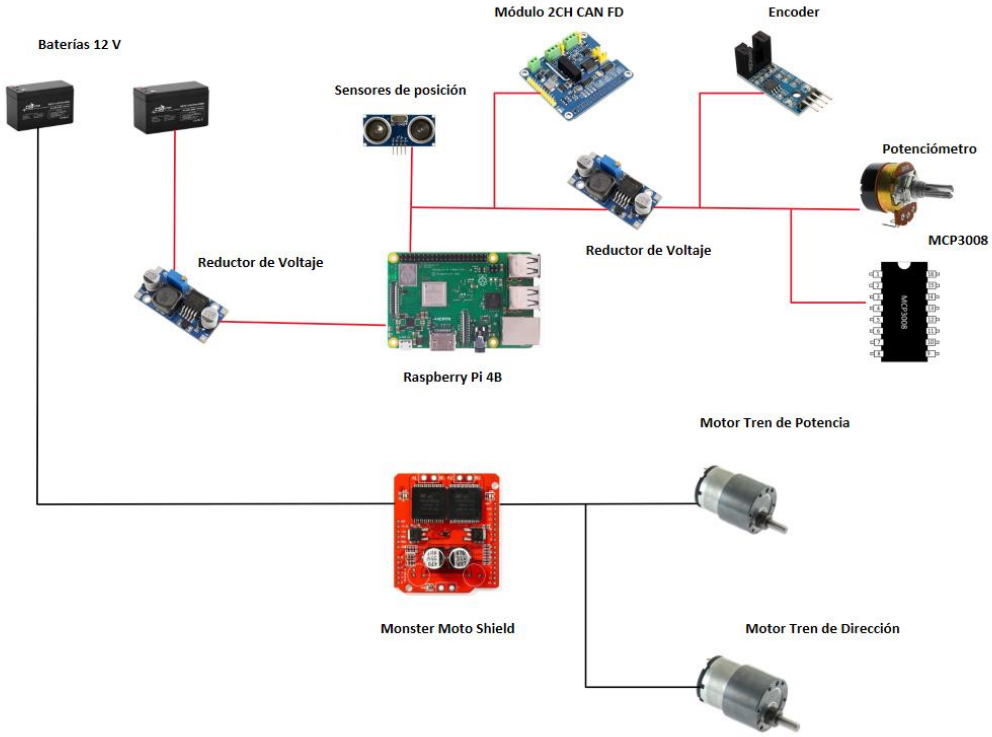


Figura 3. Diagrama de alimentación simplificado de los componentes del vehículo.

Descripción de la conexión de pines GPIO del Raspberry Pi 4B

El Raspberry Pi 4B tiene un bus de conexión GPIO de 40 pines, los cuales se utilizan para manejar las señales eléctricas de salida y de entrada para el control de los motores y de los sensores instalados en el robot móvil.

El diagrama de los pines GPIO del Raspberry, junto con la tabla de conexiones entre las entradas y salidas de los dispositivos instalados en el vehículo se detallan en el Anexo B. Además, se diseña una placa PCB que sirve para la conexión entre los dispositivos descritos anteriormente y los pines de control GPIO del Raspberry. El diseño de la placa se detalla también en el Anexo B.

DESCRIPCIÓN DEL SISTEMA CONTROLADO EN LAZO CERRADO

El sistema del vehículo autónomo consiste en los sensores descritos anteriormente, los actuadores, que son los motores DC de velocidad lineal y de posición angular de la columna de dirección, y de los controladores que se describirán en las próximas secciones.

Para el control de seguimiento de ruta se diseña un controlador basado en álgebra lineal, el que permite calcular la velocidad y la posición angular que debe tener la columna de dirección en dependencia de una trayectoria de referencia programada previamente.

Con el fin de traducir los valores calculados en el controlador de álgebra lineal al sistema real, se diseña un controlador PID para la velocidad lineal y un controlador PID para la posición angular de la columna de posición. Ambos controladores utilizan las salidas del controlador de álgebra lineal como referencia.

Por otra parte, las entradas del controlador de álgebra lineal son las medidas de velocidad y de posición angular dadas por los sensores instalados en el vehículo. Sin embargo, la medida en bruto de los sensores tiene una varianza considerable, lo que hace que la acción de control se vuelva muy agresiva al grado que pueda dañar a los actuadores. Para reducir la varianza de la medida tomada por los sensores, se instala un filtro de Kalman para cada medida, el cual considera que la dinámica del sistema se mantiene constante entre estados separados por un tiempo de muestreo.

El diagrama de bloques del sistema controlado en lazo cerrado se muestra en la Figura 4.

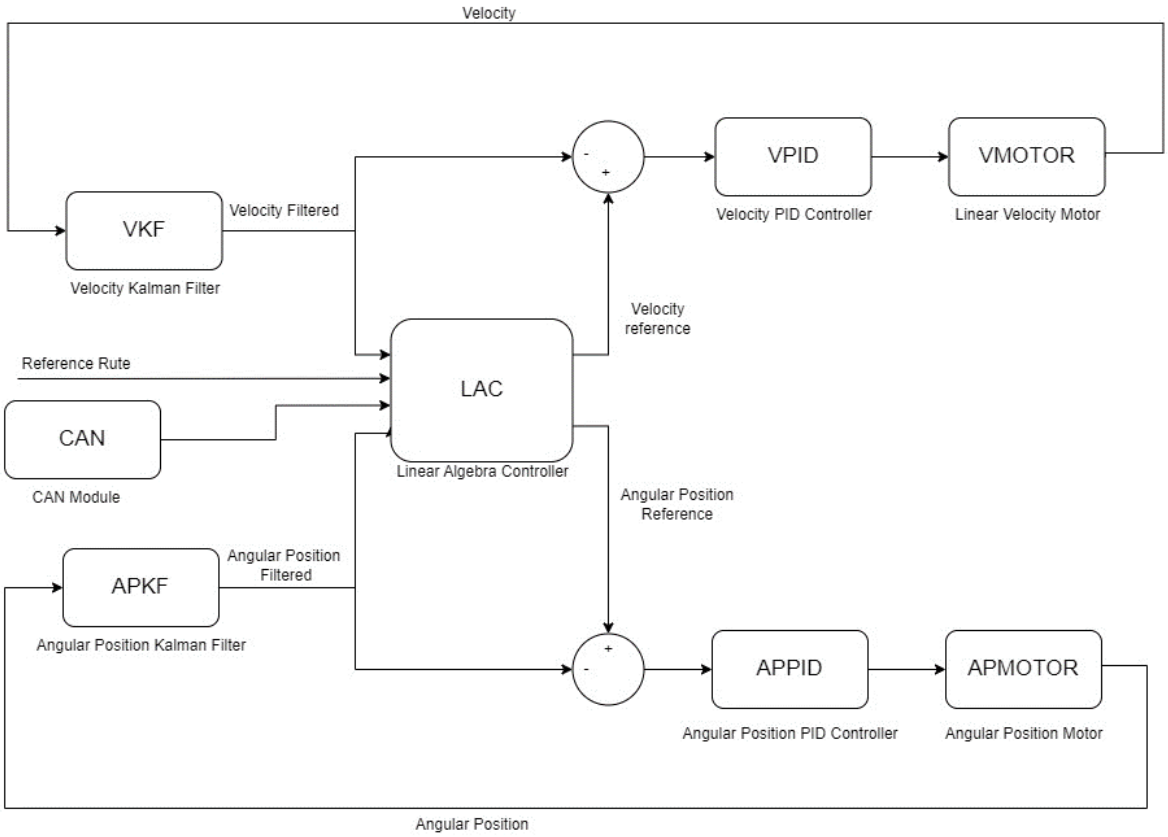


Figura 4. Diagrama de bloques del sistema controlado en lazo cerrado

MODELO DINÁMICO DE ACKERMAN

El modelo dinámico de Ackerman describe el comportamiento de un vehículo del tipo automotriz con cuatro ruedas, en el cual la velocidad lineal es dada por un impulso en el eje trasero del vehículo y el movimiento lateral se produce por una columna de dirección que permite que las ruedas delanteras giren a un cierto ángulo [7]. El modelo del vehículo se representa en la Figura 5.

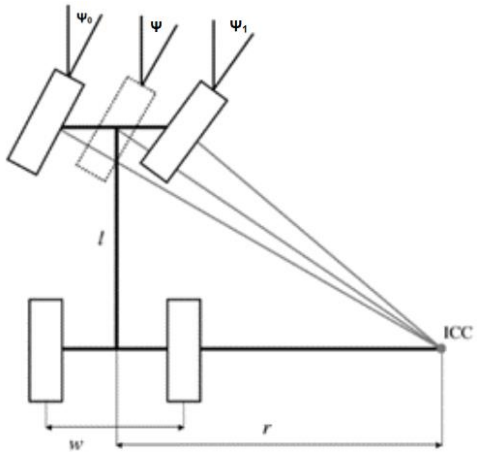


Figura 5. Representación del Modelo Dinámico de Ackerman [18].

En el modelo es importante tomar en cuenta las dimensiones geométricas del vehículo a modelar, ya que el giro del vehículo y la velocidad angular dependen de la velocidad lineal del vehículo, del ángulo de giro de la rueda y de la distancia entre el eje delantero y el eje trasero [5], [7], [9], [18].

Para simplificar aún más el modelo, se puede considerar el eje delantero como una sola rueda en la mitad del eje, cuyo ángulo determina la velocidad angular del vehículo [18]. El modelo simplificado se muestra en la Figura 6.

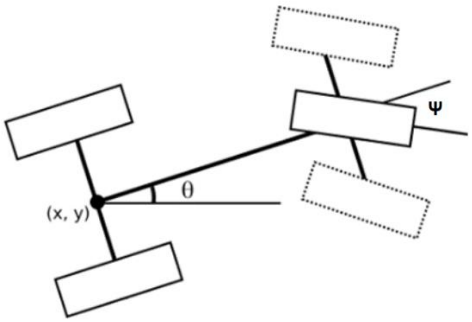


Figura 6. Modelo simplificado de Ackerman.

En donde la posición se mide en coordenadas x y y, que se ubican en el centro del eje de potencia del vehículo, así como con la inclinación theta, que representa el ángulo al que el carro se va a dirigir en el siguiente movimiento. Por otra parte, el ángulo $\psi(t)$ determina el ángulo de giro de la rueda delantera [4], [7], [9], [18].

Las ecuaciones del modelo dinámico de Ackerman para el vehículo son las siguientes [5].

$$\dot{x}(t) = v(t) \cdot \cos\left(\theta(t)\right)$$
$$\dot{y}(t) = v(t) \cdot \sin\big(\theta(t)\big)$$
$$\dot{\theta}(t) = \frac{v(t)}{L} \cdot \tan\left(\psi(t)\right)$$

Donde $\dot{x}(t)$ es la velocidad en dirección x, $\dot{y}(t)$ es la velocidad en dirección y, $\theta(t)$ es la orientación del vehículo, $v(t)$ es la velocidad lineal del vehículo y $\psi(t)$ es el ángulo de giro de las llantas delanteras.

DISEÑO DEL CONTROLADOR BASADO EN ÁLGEBRA LINEAL

Para el controlador de seguimiento de ruta del vehículo, se toma como entrada los datos del encoder de dirección y los datos del sensor hall que proporciona la velocidad lineal del vehículo en un determinado momento. Para el controlador se utiliza un tiempo de muestreo constante de 0.1 s.

A partir de los datos de velocidad y designando el punto de partida del vehículo en la coordenada (0,0) con un ángulo de orientación 0 con respecto del eje x, se calcula la posición discretizada del vehículo por medio de las siguientes ecuaciones.

$$x_n = x_{n-1} + T_0 \cdot V_{n-1} \cdot \cos(\theta_{n-1})$$

$$y_n = y_{n-1} + T_0 \cdot V_{n-1} \cdot \sin(\theta_{n-1})$$

$$\theta_n = \theta_{n-1} + \frac{T_0 \cdot V_{n-1}}{L} \tan(\psi_{n-1})$$

Por medio de las ecuaciones anteriores, se determina la posición actual del vehículo.

Por otra parte, para diseñar el controlador, se debe discretizar las ecuaciones del modelo dinámico que describe el sistema, las cuales quedan como lo siguiente [5].

$$x_{n+1} = x_n + T_0 \cdot V_n \cdot \cos(\theta_n)$$

$$y_{n+1} = y_n + T_0 \cdot V_n \cdot \sin(\theta_n)$$

$$\theta_{n+1} = \theta_n + \frac{T_0 \cdot V_n}{L} \tan(\psi_n)$$

A continuación, se determina que las variables de control que se necesita calcular en cada instante de muestreo es la velocidad lineal y el ángulo de giro de las ruedas, los cuales serán los que se configuran en los actuadores del vehículo.

$$V_n \cdot \cos(\theta_n) = \frac{x_{n+1} - x_n}{T_0}$$

$$V_n \cdot \sin(\theta_n) = \frac{y_{n+1} - y_n}{T_0}$$

$$\tan(\psi_n) = \frac{\theta_{n+1} - \theta_n}{\frac{V_n}{L} \cdot T_0}$$

Se escriben las ecuaciones en forma matricial.

$$\begin{bmatrix} \cos(\theta_n) & 0 \\ \sin(\theta_n) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V_n \\ \tan(\psi_n) \end{bmatrix} = \frac{1}{T_0} \cdot \begin{bmatrix} x_{n+1} - x_n \\ y_{n+1} - y_n \\ \theta_{n+1} - \theta_n \\ \frac{V_n}{L} \end{bmatrix}$$

A partir de esta ecuación se determina que lo siguiente tiene que cumplirse, con el fin de que el sistema de ecuaciones tenga solución.

$$\begin{bmatrix} \cos(\theta_n) \\ \sin(\theta_n) \end{bmatrix} \cdot V_n = \frac{1}{T_0} \cdot \begin{bmatrix} x_{n+1} - x_n \\ y_{n+1} - y_n \end{bmatrix}$$

De esta condición, se concluye que la única manera que se el sistema tenga solución es que ambos vectores sean paralelos entre sí, por lo cual se cumple la condición de paralelismo.

$$\frac{\sin(\theta_n)}{\cos(\theta_n)} = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}$$

Ya que esta condición se debe cumplir, se designa una variable adicional que permita que se cumpla esta condición. Se designa la orientación como una variable sacrificable que no sigue la referencia, con el fin de que las coordenadas en x y en y puedan seguir la referencia.

$$\tan(\theta_{ez_n}) = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}$$

Con el fin de que las coordenadas horizontales y verticales tiendan a la referencia, se calcula el siguiente punto de coordenadas de la siguiente manera.

$$x_{n+1} = x_{ref_{n+1}} - k_x \cdot (x_{ref} - x_n)$$

$$y_{n+1} = y_{ref_{n+1}} - k_y \cdot (y_{ref} - y_n)$$

$$\theta_{n+1} = \theta_{ez_{n+1}} - k_\theta \cdot (\theta_{ez_n} - \theta_n)$$

Donde k_y, k_x y k_θ son constantes entre 0 y 1 que determinan que tan agresiva es la acción de control. Entre más cercano a uno, más suave la acción de control.

De esta manera, se determina las siguientes relaciones.

$$\Delta x = x_{ref_{n+1}} - k_x \cdot (x_{ref} - x_n) - x_n$$

$$\Delta y = y_{ref_{n+1}} - k_y \cdot (y_{ref} - y_n) - y_n$$

$$\Delta \theta = \theta_{ez_{n+1}} - k_\theta \cdot (\theta_{ez_n} - \theta_n) - \theta_n$$

Finalmente, se resuelve el sistema de ecuaciones original utilizando la siguiente relación de álgebra lineal $A^T \cdot A \cdot x = A^T \cdot b$. De esta manera, el sistema de ecuaciones original queda como lo siguiente.

$$\begin{bmatrix} \cos(\theta_{ez_n}) & \sin(\theta_{ez_n}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_{ez_n}) & 0 \\ \sin(\theta_{ez_n}) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V_n \\ \tan(\psi_n) \end{bmatrix} = \frac{1}{T_0} \cdot \begin{bmatrix} \cos(\theta_{ez_n}) & \sin(\theta_{ez_n}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \\ \frac{V_n}{L} \end{bmatrix}$$

$$\begin{bmatrix} V_n \\ \tan(\psi_n) \end{bmatrix} = \frac{1}{T_0} \cdot \begin{bmatrix} \Delta x \cdot \cos\left(\theta_{ez_n}\right) + \Delta y \cdot \sin\left(\theta_{ez_n}\right) \\ \Delta\theta \\ \frac{V_n}{L} \end{bmatrix}$$

De esta manera, se calculan las acciones de control por medio de las siguientes ecuaciones.

$$V_n = \frac{1}{T_0} \cdot \left(\Delta x \cdot \cos\left(\theta_{ez_n}\right) + \Delta y \cdot \sin\left(\theta_{ez_n}\right)\right)$$

$$\psi_n = \arctan\left(\frac{\Delta\theta}{\frac{V_n}{L} \cdot T_0}\right)$$

MÉTODO DE IDENTIFICACIÓN DE SISTEMAS EN LAZO ABIERTO

Para el desarrollo del controlador de posición angular de la columna de dirección y el controlador de velocidad que se requiere implementar para poder asegurar que el vehículo siga la trayectoria programada con la velocidad y el ángulo de giro calculados por el controlador descrito en la sección anterior, se requiere identificar el sistema que incluyen los motores y las cajas reductoras en dependencia de la señal de control de velocidad de los motores. Debido a que no se conoce la dinámica precisa de ambos sistemas, se requiere utilizar un método de identificación de procesos en lazo abierto. Se utiliza el método de dos puntos para obtener un sistema de primer orden más retardo [19], [20].

El método utilizado para la identificación del sistema de velocidad lineal y de posición angular de la columna de dirección es un método de dos puntos desarrollado por Alfaro, el cual devuelve como modelo dinámico una función de transferencia de un sistema de primer orden más retardo [19].

La identificación del proceso se basa en la curva de reacción de este, por lo cual se realiza un cambio a la entrada del sistema y se mide la dinámica del cambio de la salida del sistema.

En el Anexo C se describe el proceso de identificación utilizado para los sistemas de velocidad lineal y de posición angular de la columna de dirección del vehículo.

EL FILTRO DE KALMAN

El filtro de Kalman es un algoritmo que se utiliza para estimar el estado futuro de un sistema en dependencia de las medidas tomadas anteriormente. En este caso, se utiliza un filtro de Kalman de una dimensión para reducir la incertidumbre de la medición de velocidad y de posición angular de la columna de dirección [12]. El filtro de Kalman de una dimensión consiste en 5 pasos, inicializar el filtro, tomar la medida actual, predecir el estado siguiente, comparar la predicción con la medida actual y calcular la estimación del estado siguiente a partir de la comparación realizada anteriormente [11]–[13].

En el filtro de Kalman de una dimensión, se tiene la variable medida y la variable estimada. Estas dos variables tienen el mismo valor esperado. Para el desarrollo de las ecuaciones del filtro se denomina la variable estimada como x y la variable medida como y.

$$x \sim N(\mu, \sigma_x)$$

$$y \sim N(\mu, \sigma_y)$$

$$E\{x\} = E\{y\} = \mu$$

La ecuación para calcular la estimación del estado es la siguiente.

$$x_n = x_{n-1} + (1 - g_n) \cdot (y_n - x_{n-1})$$

En donde g es la ganancia del filtro. El filtro pretende minimizar la varianza de la variable estimada, por lo que la ganancia tiene que ser la siguiente [12].

$$g_n = \frac{\sigma_{yn}^2}{\sigma_{xn-1}^2 + \sigma_{yn}^2}$$

Por medio de la ganancia del filtro y la ecuación de estimación del estado siguiente, se calcula la estimación del sistema y la varianza de la estimación por medio de las siguientes ecuaciones [12], [13].

$$x_n = \frac{\sigma_{y,n}^2 \cdot x_{n-1} + \sigma_{x,n-1}^2 \cdot y_n}{\sigma_{x,n-1}^2 + \sigma_{y,n}^2}$$

$$\sigma_{x,n}^2 = \left(\frac{1}{\sigma_{x,n-1}^2} + \frac{1}{\sigma_{y,n}^2} \right)^{-1}$$

El valor filtrado de salida depende de la ganancia del filtro. Si se tiene una ganancia alta, el valor filtrado tiende al valor medido, mientras que, con una ganancia baja, el valor filtrado se acerca más al valor predicho.

En este caso, como el tiempo de muestreo es muy pequeño, se asume que el valor predicho es el mismo que el del estado anterior, debido a que se asume que la dinámica del sistema es constante entre dos tiempos de muestreo [12].

$$x_{n+1} = x_n$$

La derivación de la ecuación de actualización de estado y el comportamiento del filtro en dependencia a la ganancia se muestran en el Anexo D.

Implementación del filtro de Kalman para la medición de velocidad lineal

Para implementar el filtro de Kalman se debe inicializar el filtro, es decir, asignar un valor inicial de la estimación del valor de la velocidad y de la varianza de la velocidad medida. Para obtener la varianza de la velocidad medida, se realiza una prueba de medida de velocidad a una entrada escalón y se mide la varianza de los valores obtenidos. De esta manera, se obtiene que la varianza de la velocidad medida es la siguiente.

$$\sigma_v^2 = 1500$$

Se inicializa el filtro asumiendo que la varianza de la estimación es grande, ya que se inicia con una velocidad 0, cuando el carro está parado.

$$\sigma_{v,est,1}^2 = 1500$$

$$v_{est,1} = 0 \left[\frac{cm}{s}\right]$$

Del mismo modo, cuando existe un cambio en la velocidad del vehículo, la ganancia del filtro tiene que ser mayor, ya que la estimación debe ser más cercana al valor medido que al valor estimado en el periodo anterior. Para aumentar la ganancia, si el error entre la velocidad medida y la velocidad estimada en el periodo anterior es mayor a un valor umbral, se restablece la varianza de la estimación a la varianza de la velocidad medida. La siguiente figura muestra el comportamiento del filtro de Kalman aplicado en la medición de velocidad.

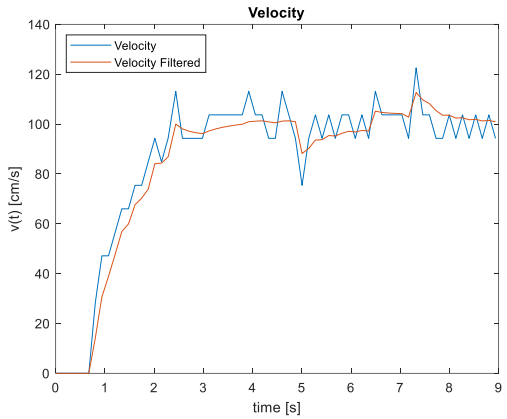


Figura 7. Comportamiento del filtro de Kalman para la medición de velocidad.

Por medio de la implementación del filtro se reduce la varianza de la medida en estado estable de 60 $\left[\frac{cm^2}{s^2}\right]$ a 30 $\left[\frac{cm^2}{s^2}\right]$. Esto demuestra que el filtro implementado en la medida de velocidad reduce la varianza de la medida en la mitad de la varianza de los valores medidos en el sensor.

Implementación del filtro de Kalman para la posición angular de la columna de dirección

Del mismo modo, se mide la varianza de la medida del valor que lee el Raspberry a partir del voltaje del potenciómetro por medio de una entrada escalón, con el fin de obtener la varianza que inicializa el filtro.

$$\sigma_a^2 = 0.1$$

Se asume que el vehículo inicia con las llantas rectas, por lo que el valor inicial de la estimación del potenciómetro es de 0.5, y se asume que la varianza de estimación es igual a la de medida inicialmente.

$$\sigma_{a,est,1}^2 = 0.1$$

$$a_{est,1} = 0.5$$

Análogamente al filtro de velocidad, cuando existe un cambio en el ángulo de dirección, la ganancia del filtro debe ser grande, por lo que se reestablece el valor de la varianza de la estimación al valor de la medida. De esta manera, el comportamiento del filtro aplicado en la medición del ángulo se muestra en la siguiente figura.

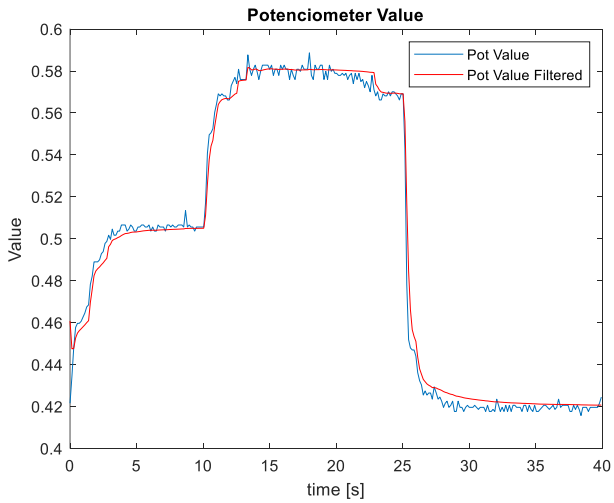


Figura 8. Comportamiento del filtro de Kalman para la medición del ángulo de giro.

Del mismo modo, se mide la varianza de la medida del valor que devuelve el potenciómetro en estado estable. La varianza del valor sin el filtro es de $4.38 \cdot 10^{-6}$, mientras que el valor de la varianza medida con el filtro implementado es de $2.23 \cdot 10^{-6}$, lo cual comprueba que el filtro reduce la varianza de la medida aproximadamente en la mitad.

PROTOCOLO DE COMUNICACIÓN CONTROLLER AREA NETWORK (CAN)

El protocolo CAN (Controller Area Network) es un estándar de comunicación serial, diseñado principalmente para la industria automotriz. Este protocolo permite a microcontroladores y dispositivos compartir información entre sí, por medio de un bus de datos común bidireccional, lo que permite la reducción de conexiones cableadas, lo que ha permitido que los arneses en los vehículos sean más reducidos [14]. El bus CAN es de tipo de transmisión, lo cual implica que todos los nodos conectados al bus reciben el mensaje. No hay posibilidad de que un mensaje se mande a solamente un nodo, sin embargo, el protocolo es diseñado para que cada nodo solamente reaccione al mensaje que le corresponde [15]. Este protocolo se puede descomponer en capas, como la capa de aplicación, de objeto, de filtración de mensaje de estatus y de transferencia [14].

Los mensajes en el protocolo CAN consisten típicamente de 10 bytes o de 94 bits en el protocolo extendido. Esta información se estructura en un marco determinado. Existen 4 tipos de marcos: el marco de datos, el marco remoto, el marco de error y el marco de sobrecarga, siendo el marco de datos el más utilizado. Los mensajes transmitidos no tienen dirección específica, ya que todos los nodos reciben la transmisión, sin embargo, el contenido de los mensajes determina el nodo de interés [14], [15].

El marco de datos es el encargado de enviar información relevante a los nodos conectados al bus. El marco remoto es el encargado de solicitar información. El marco de error es un mensaje que viola las reglas de transmisión del protocolo, el cual se transmite solamente cuando un nodo detecta un error, lo que hace que todos los nodos detecten este error y automáticamente se vuelva a enviar el mensaje original. Finalmente, el marco de sobrecarga

es similar al marco de error, pero este solicita un retraso en la transmisión de los datos dentro del bus [16].

El protocolo CAN fue desarrollado para tener una alta inmunidad para interferencia eléctrica y para tener la capacidad de detectar errores internos y repararlos, por lo que el protocolo se ha expandido de la industria automotriz hacia distintas industrias como la automatización industrial, la industria médica y la manufactura de productos [14].

El protocolo es basado en el modelo estándar de interconexión de sistemas abiertos (OSI). Este estándar se lo conoce también como el modelo de 7 capas, ya que se compone de elementos independientes que describe los requisitos de comunicación a distintos niveles de abstracción [14], [16].

El formato del marco estándar del protocolo CAN, así como las capas de abstracción de sistemas OSI y en los que el protocolo trabaja se describen en el Anexo E, así como el módulo que se utiliza en el vehículo compatible con el Raspberry Pi 4B y sus pines de conexión.

SINTONIZACIÓN DEL CONTROLADOR DE VELOCIDAD

Para la sintonización del controlador PID de velocidad se debe identificar el sistema primeramente por medio del método descrito anteriormente. El proceso de identificación del sistema se muestra en el Anexo F. La función de transferencia del sistema de velocidad es el siguiente.

$$G_v(s) = \frac{3.5 \cdot e^{-0.36 \cdot s}}{2 \cdot s + 1}$$

. El controlador tiene la siguiente forma [20].

$$PID = K_c \cdot \left(e + \frac{1}{T_i} \int e \cdot dt + T_d \cdot \frac{d}{dt} e \right)$$

Donde e es el error entre la referencia y el valor medido, Kc es la ganancia del controlador, T_i es el término integral del controlador y T_d es el término derivativo.

Para calcular los parámetros del controlador PID del sistema de velocidad lineal, se utiliza el método de Dahlin, el cual produce una respuesta suave y sin sobre pico. Las ecuaciones de Dahlin para la sintonización del controlador son las siguientes [20].

$$K_c = \frac{1}{2 \cdot K} \cdot \frac{\tau}{t_0}$$

$$T_i = \tau$$

$$T_d = \frac{t_0}{2}$$

De esta manera los parámetros del controlador PID son los siguientes.

$$K_c = 0.79$$

$$T_i = 2$$

$$T_d = 0.18$$

Implementación del controlador en el sistema real

Se implementa el controlador en el sistema real y se observa el comportamiento de la velocidad lineal en el robot móvil. La Figura 9 muestra el comportamiento de la velocidad lineal por medio de un cambio de referencia de 0 a 100 cm/s.

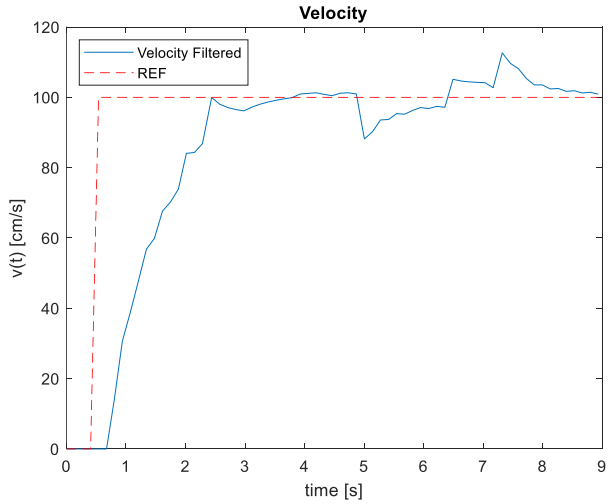


Figura 9. Velocidad Lineal Controlada en el Sistema Real.

Se observa que el controlador permite que la velocidad lineal siga la trayectoria. Sin embargo, se tiene una incertidumbre en la medición que el filtro de Kalman no alcanza a eliminar, lo que el controlador considera como una perturbación. Se observa en la Figura 9 que el controlador fuerza a los picos que se generan en la medida a seguir la referencia, los cuales son considerados por el controlador como perturbaciones, lo que demuestra que el controlador PID diseñado es capaz de eliminar perturbaciones.

SINTONIZACIÓN DEL CONTROLADOR DE POSICIÓN ANGULAR DE LA COLUMNA DE DIRECCIÓN

Asimismo, se debe identificar el sistema de posición angular de la columna de dirección del vehículo, lo cual se detalla en el Anexo G. La función de transferencia de la posición angular de la columna de dirección es la siguiente.

$$G_a(s) = \frac{5.93 \cdot e^{-0.17 \cdot s}}{s \cdot (0.09 \cdot s + 1)}$$

Del mismo modo, se utiliza un controlador PID para la posición angular de la columna de dirección. Sin embargo, como el sistema de posición es un sistema de segundo orden, se calculan los parámetros del controlador por medio de las siguientes ecuaciones [20].

$$K_c = \frac{2 \cdot \tau_c + \tau + t_0}{K \cdot (\tau_c + t_0)^2}$$

$$T_i = 2 \cdot \tau_c + \tau + t_0$$

$$T_d = \frac{(2 \cdot \tau_c + t_0) \cdot \tau}{2 \cdot \tau_c + \tau + t_0}$$

Donde τ_c es un parámetro de sintonización que se escoge de la siguiente manera [20].

$$\frac{\tau_c}{t_0} > 0.8 \; y \; \tau_c > 0.1 \cdot \tau$$

De esta manera los parámetros del controlador PID son los siguientes.

$$K_c = 0.5$$

$$T_i = 1.25$$

$$T_d = 0.08$$

Implementación del controlador en el sistema real

Del mismo modo, se implementa el controlador en el sistema real. Sin embargo, la dinámica del sistema cambia a medida que el error tiende a cero, ya que el motor de la dirección no alcanza a girar las ruedas con un valor de PWM muy bajo, por lo que se incrementa el valor de la ganancia a medida que el error entre la referencia y el valor medido disminuye. Los valores que se utilizan son determinados por medio prueba y error. Del mismo modo, se presenta la respuesta del controlador a un cambio de referencia.

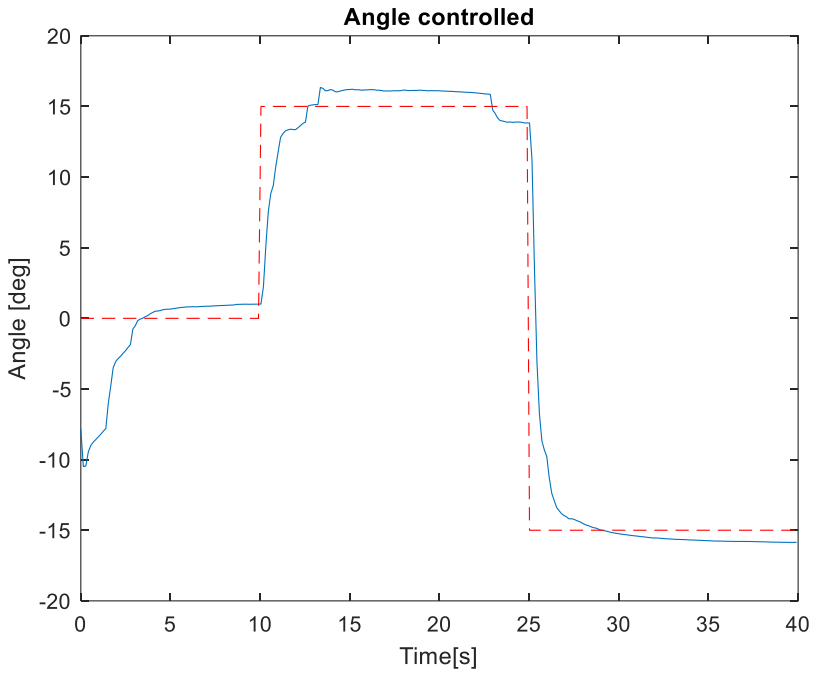


Figura 10. Posición Angular Controlada en el Sistema Real.

Se observa que la posición angular en el sistema real puede seguir de manera adecuada la referencia, pero en estado estacionario tiene un error de aproximadamente 1 grado, esto se debe a que el sistema no avanza a girar las ruedas con un PWM muy bajo debido al peso de las ruedas. Sin embargo, el error es mínimo, por lo que se considera aceptable.

IMPLEMENTACIÓN DEL CONTROLADOR DE ÁLGEBRA LINEAL

Se implementa el controlador de seguimiento de trayectoria basado en álgebra lineal en el vehículo. Para esto, se diseña una ruta circular en la cual debe circular a una velocidad de 60 cm/s con un radio de 2 m. Este controlador se encuentra acoplado con los controladores PID descritos en las secciones anteriores, por lo que el sistema controlado es el que se muestra en la Figura 4.

Se realiza una prueba de seguimiento de trayectoria en la ruta diseñada y programada previamente en el procesador del vehículo. Los resultados de esta prueba se reportan a continuación.

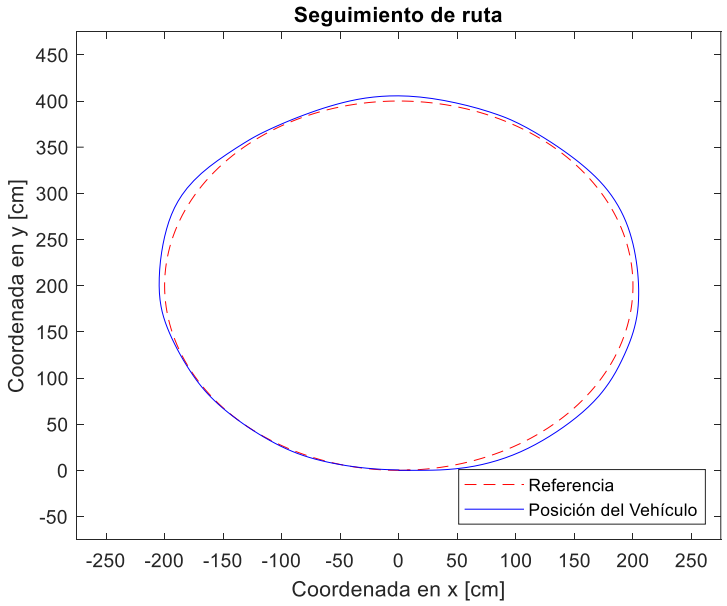


Figura 11. Resultados de la prueba de seguimiento de ruta.

Se observa que el vehículo puede seguir de manera adecuada la ruta programada previamente, utilizando los tres controladores diseñados, por lo que se concluye que estos controladores funcionan de manera adecuada. Sin embargo, se observa que existe un error

entre la ruta programada y el camino del vehículo. El error cuadrático medio entre las coordenadas en x de la ruta de referencia y la ruta que realiza el vehículo es de 103.46 $[cm^2]$. Mientras que el error cuadrático medio de las coordenadas en y es de 93.64 $[cm^2]$.

Asimismo, se reporta el ángulo de orientación del vehículo al realizar la ruta programada.

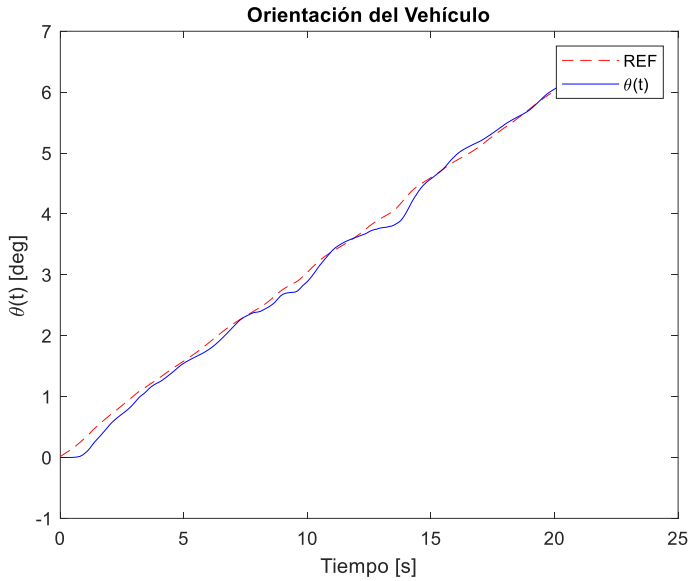


Figura 12. Ángulo de orientación del vehículo durante la prueba de seguimiento de trayectoria.

Se observa que el vehículo sigue la referencia de orientación calculada por medio del controlador de álgebra lineal. Sin embargo, existe un error entre la orientación real del vehículo y la orientación de referencia. El error cuadrático medio del valor real y de la referencia es de 0.012 [deg^2]. Este es un valor bastante cercano a cero, por lo que se concluye que el error entre la orientación real y la orientación de referencia es insignificante.

Por otra parte, se muestra en la siguiente figura el comportamiento de la posición angular de la columna de dirección del vehículo durante la trayectoria.

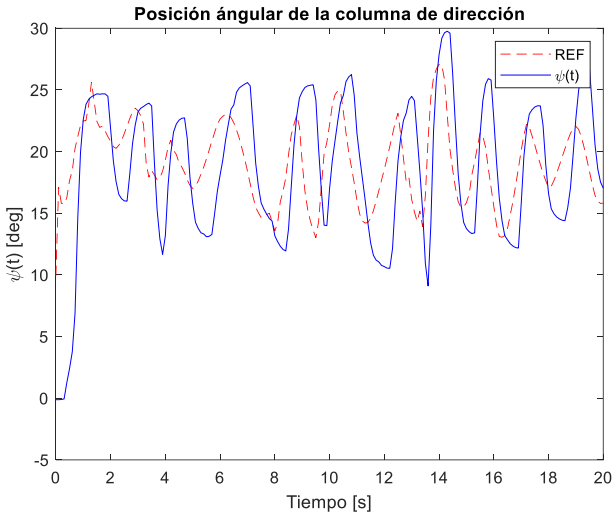


Figura 13. Posición angular de la columna de dirección durante la prueba de seguimiento de trayectoria.

Se observa que la posición angular de la columna de dirección es capaz de seguir la referencia generada por el controlador de álgebra lineal. Sin embargo, se observa un retraso entre el valor medido de la posición y la referencia, lo que causa que el vehículo tenga un error en el seguimiento de la ruta y lo que se observa en la Figura 12, que la orientación del vehículo oscila alrededor de la referencia en lugar de estabilizarse. El error cuadrático medio de la posición angular es de 366.47 [deg^2]. Este es un valor grande para el error, lo cual indica que el controlador para la posición angular de la dirección del vehículo no es preciso, por lo cual se concluye que este debe ser mejor sintonizado o cambiar de tipo de controlador para permitir que el seguimiento de ruta sea más preciso.

Finalmente, se reporta el comportamiento de la velocidad lineal del vehículo.

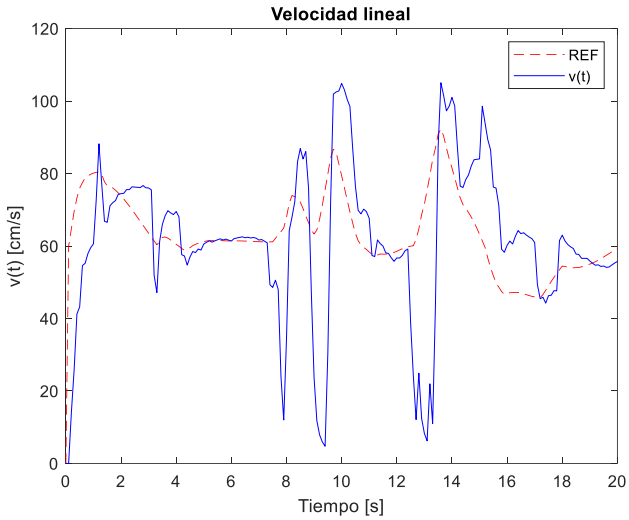


Figura 14. Velocidad lineal del vehículo durante la prueba de seguimiento de trayectoria.

Se observa que el vehículo se mueve con una velocidad de alrededor 60 cm/s. Asimismo, se observa que el controlador PID de velocidad acoplado al sistema es capaz de seguir la referencia calculada por el controlador de seguimiento de trayectoria. La variación que se observa en la velocidad real del vehículo se debe a la interacción del filtro de Kalman y de la velocidad medida en el sistema. Asimismo, cuando el PWM calculado por el controlador PID de velocidad disminuye, es más difícil mover el vehículo, por lo que la velocidad disminuye y se generan los picos hacia abajo, lo cual compensa en controlador de manera brusca, lo que genera los picos por encima de la referencia. Por otra parte, el vehículo se mueve ligeramente por encima de la velocidad límite baja de 50 cm/s, lo que hace que cuando la referencia disminuya se vuelva más difícil mover el vehículo. Debido a esto se aumenta el error entre la velocidad medida y la velocidad de referencia del vehículo. El error cuadrático medio entre la velocidad medida y la velocidad de referencia es de 421.27 $\left[\frac{cm^2}{s^2}\right]$. Si se aumenta la velocidad de seguimiento de trayectoria del vehículo este error disminuye, sin embargo, en la realidad la ruta que realiza el vehículo es distinta debido a la inercia del mismo que causa que el radio del círculo aumente a una mayor velocidad.

EMULACIÓN DE RECEPCIÓN DE SEÑALES POR MEDIO DEL PROTOCOLO CAN

Se realiza una prueba de funcionamiento del prototipo utilizando un segundo Raspberry Pi 4B que se conecta al canal 0 del módulo de expansión CAN. El segundo Raspberry es el encargado de enviar un mensaje a través del protocolo CAN del canal 0 al canal 1. El procesador principal recibe el mensaje, detiene el camino del vehículo, detiene el seguimiento de la ruta del vehículo, endereza la columna de dirección, hace retroceder el vehículo por un tiempo de 1.5 segundos y cambia de ruta de referencia. Cuando se recibe el mensaje, se reinicia el controlador de álgebra lineal, por lo que el procesador actúa como que iniciara una nueva ruta a partir de la posición en la que se encuentra después de retroceder. Cabe destacar que el controlador de seguimiento de ruta no es capaz de hacer que el vehículo vaya en reversa, por lo que el momento que el vehículo retrocede, se comprueba que el mensaje ha sido recibido y que el programa del vehículo entra a una interrupción en el controlador principal.

Para esta prueba, se realiza una ruta en línea recta a una velocidad de 60 cm/s como ruta principal. La segunda ruta es una circunferencia de 3 m de diámetro, con una velocidad lineal del vehículo de 60 cm/s. Las rutas programadas en este experimento se muestran en el Anexo H.

En la siguiente figura se muestra la trayectoria que el vehículo sigue y la referencia que el controlador observa para el seguimiento de ruta.

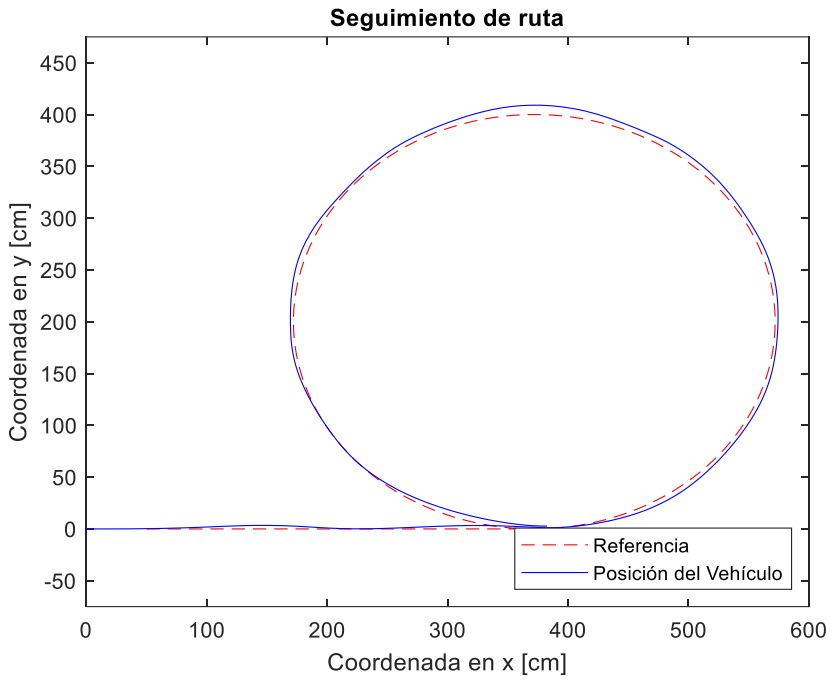


Figura 15. Seguimiento de trayectoria durante la prueba de emulación de recepción de mensaje por CAN.

Se observa que el vehículo empieza en la ruta en línea recta. Cuando se recibe el mensaje enviado por el segundo Raspberry, el vehículo cambia de ruta a una circunferencia de radio 2 m. Se observa que el vehículo es capaz de seguir las dos rutas programadas, y que este cambia de ruta en la recepción del mensaje. Del mismo modo, existe un error entre la ruta que realiza el vehículo con la referencia. El error cuadrático medio entre las coordenadas x referencia y de la ruta realizada por el vehículo es de 368.48 $[cm^2]$. Por otro lado, el error cuadrático medio de las coordenadas en y es de 57.79 $[cm^2]$. El error en x es mayor, ya que la ruta de referencia comienza 50 cm por delante del vehículo, lo cual aumenta el error en x hasta que el vehículo llegue a la referencia.

Asimismo, se reporta la orientación del vehículo durante esta prueba.

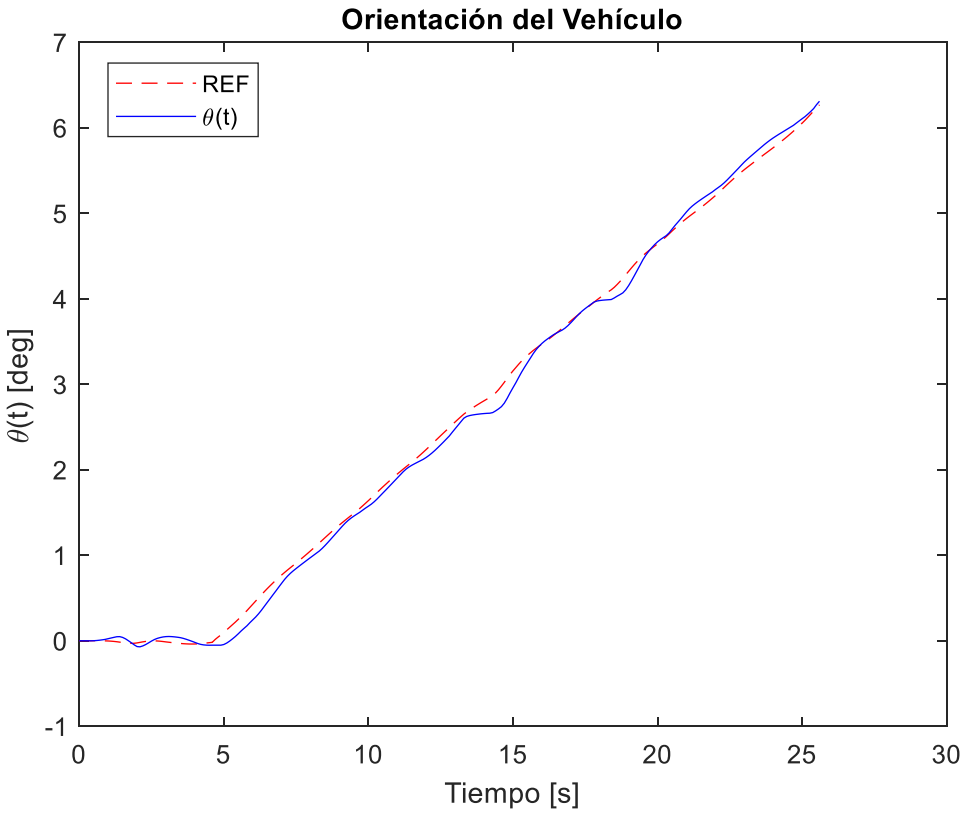


Figura 16. Orientación del vehículo durante la prueba de emulación de recepción del mensaje por CAN.

Del mismo modo, se observa que el vehículo tiene un cambio de orientación en el momento en el que este recibe el mensaje del segundo Raspberry. Se observa que el vehículo oscila alrededor de la referencia, lo cual se debe a la oscilación en la medida del ángulo de dirección que se observa en la siguiente figura. El error entre la referencia y la orientación real del vehículo es de 0.0085 [deg^2], lo que indica que el error en la orientación del vehículo es insignificante.

Asimismo, se reporta la posición angular de la columna de dirección durante el experimento.

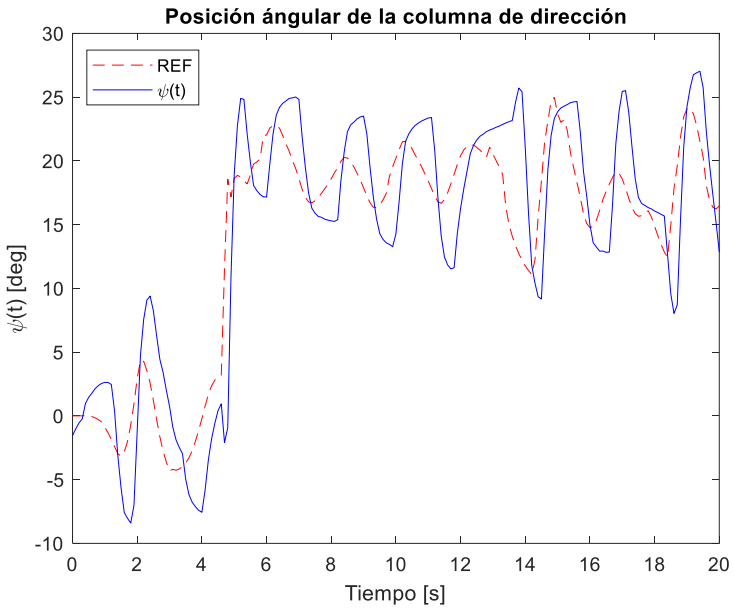


Figura 17. Posición angular de la columna de dirección durante la prueba de emulación de recepción del mensaje por CAN.

Del mismo modo, se observa que existe un retraso en la posición angular de la columna de dirección con respecto a la referencia generada con el controlador de álgebra lineal. Además, se observa que la posición angular de la dirección medida sobrepasa la referencia en los cambios, lo que genera error entre la medida y la referencia. Asimismo, se observa el cambio en la posición angular en el momento en el que el procesador principal recibe el mensaje. El error cuadrático medio entre la posición angular medida y la referencia es de 308.73 [deg^2], lo cual comprueba que el controlador de posición angular de dirección debe ser sintonizado de mejor manera para permitir que el vehículo siga la ruta de referencia de manera más precisa.

Finalmente, se reporta el comportamiento de la velocidad lineal del vehículo durante el experimento.

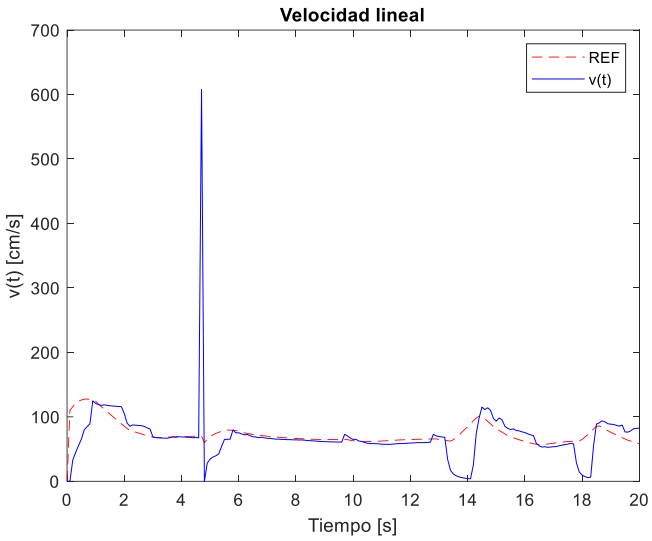


Figura 18. Velocidad lineal durante la prueba de emulación de recepción del mensaje por CAN.

La velocidad lineal del vehículo es capaz de seguir la referencia generada en el controlador de álgebra lineal. Se observa un pico de velocidad en la Figura 18, el cual representa el momento en el que el vehículo vuelve a arrancar en la segunda ruta, ya que el motor debe dar más potencia para iniciar el movimiento. El error cuadrático medio de la velocidad d medida y la velocidad de referencia es de 1650.9 $\left[\frac{cm^2}{s^2}\right]$. El error tan grande es debido al pico que se genera en el momento que el vehículo vuelve a arrancar después de recibir el mensaje.

LIMITACIONES DEL SISTEMA

El sistema cuenta con limitaciones físicas y mecánicas que no le permiten seguir rutas que requieran que el sistema actúe más allá de sus limitaciones. El prototipo es un vehículo grande que es pesado. Debido a esto, la ruta se debe diseñar con una velocidad de avance mayor a 50 cm/s, de lo contrario la señal de PWM para hacer que el motor gire más lento no podrá mover el vehículo.

Asimismo, ya que el vehículo tiene una masa grande, a medida que aumenta la velocidad la inercia del vehículo incrementa, lo que causa que en las curvas este se desvíe, ya que el radio de curvatura incrementa en la realidad. Debido a esto, el vehículo no alcanza a completar la ruta de referencia circular en la realidad, pero el controlador si ve que el vehículo la completa. Solamente en un caso alcanza a dar una vuelta completa, este caso es con un radio de 2 m y con una velocidad de 50 cm/s.

Por otra parte, el sistema de la columna de dirección tiene topes mecánicos que no le permiten girar a las ruedas delanteras en un ángulo superior a 30 grados a cada lado. Si se intenta girar en un ángulo mayor a lo que los topes mecánicos permiten, los engranes de a caja de reducción del sistema de dirección se desgastan y se pueden llegar a romper, por lo que se programa en el controlador un limitante de giro que, si se llega a los 30 grados y se quiere seguir girando, se apague el motor de dirección.

Asimismo, el controlador de álgebra lineal no puede reaccionar a discontinuidades bruscas en la orientación del vehículo, ya que esto hace que el vehículo se separe de la ruta y este se pierda. Debido a esto, en la ruta circular, el vehículo solamente es capaz de dar una vuelta completa, si se requiere dar dos vueltas, al momento de llegar a 360 grados en la orientación del vehículo, el controlador hace un salto a 0 grados de orientación que no le permite al

vehículo dar una segunda vuelta, ya que el controlador entiende este salto como dar una vuelta completa en la dirección opuesta a la dirección, la cual está limitada mecánicamente y esto hace que el vehículo se salga de la ruta establecida. El método más simple para resolver este problema es introducir condiciones en el programa para evitar el salto, pero esto implica que las condiciones dentro del programa deben aumentar indefinidamente en dependencia de cada ruta programada, lo cual vuelve al programa ineficiente y lento. Otra solución posible es introducir estar restricciones en el modelo matemático del vehículo dentro del controlador. Finalmente, otra solución posible propuesta es linealizar las ecuaciones trigonométricas del controlador de álgebra lineal para evitar que se produzcan discontinuidades en el cálculo del ángulo de dirección.

CONCLUSIONES

En este trabajo se pretende diseñar un vehículo terrestre semiautónomo que sea capaz de seguir una ruta preestablecida con una intervención mínima de un usuario. El vehículo utilizado para el prototipo es un modelo a escala 4:1 de un BMW i8 Spyder. El vehículo tiene un largo entre ejes de 70 cm y un ancho entre llantas de 55 cm.

El vehículo cuenta con sensores de distancia ultrasónicos para detección de obstáculos, un sensor de velocidad lineal y un sensor de posición angular de la columna de dirección. Además, el vehículo cuenta con un motor DC para sistema de tracción y un motor DC para el sistema de dirección como actuadores. Finalmente, se instala un módulo de expansión para el protocolo de comunicación CAN que le permite al procesador principal, un Raspberry Pi 4B, comunicarse con otro procesador, que le permite obtener información adicional a los sensores instalados en el sistema principal.

Para el seguimiento de trayectoria se diseña un controlador basado en álgebra lineal que utiliza la posición actual del vehículo, calculada a través de las medidas de velocidad lineal y del ángulo de la dirección, para calcular la velocidad y el ángulo de la dirección del siguiente tiempo de muestreo en base a la ruta de referencia programada.

Por otra parte, para traducir el valor calculado por el controlador principal, se diseña dos controladores PID que calculan la señal PWM que se alimenta a los actuadores para alcanzar la velocidad y el ángulo de dirección calculados en el controlador principal.

Los tres controladores diseñados trabajan conjuntamente y hacen que el vehículo siga la ruta programada. Sin embargo, se observa que existe un error en la reproducción de la ruta, lo que se debe a los cambios dinámicos del sistema de potencia y el sistema de dirección a medida

que el PWM de control de los motores cambia y el peso del prototipo. Asimismo, el controlador de posición angular de la dirección es capaz de seguir la referencia calculada por medio del controlador de seguimiento de trayectoria, pero existe un retardo entre la medición y la referencia, lo cual es debido a la dinámica del sistema y las limitaciones mecánicas de la columna de dirección. El controlador PID diseñado es capaz de seguir la referencia de posición angular, sin embargo, el error entre el valor medido y el valor de referencia no es cero, debido a los cambios de dirección que se generan durante la trayectoria, por lo cual se debe realizar una sintonización más fina del controlador o cambiar a un controlador que sea capaz de seguir estos cambios en la referencia con un tiempo de asentamiento menor al propuesto. Asimismo, cabe recalcar que los engranes de la caja de reducción de velocidad en el sistema de dirección son de plástico, por lo que cambios demasiado bruscos pueden llegar a desgastarlos y forzarlos a un estado de fallo mecánico.

Por otra parte, se observan picos en las gráficas de velocidad lineal, los cuales son causados por un error en la medición con el sensor de velocidad el cual consiste en una rueda ranurada. El error se causa debido a que en el tiempo de muestreo se cuentan distintos pases por las ranuras, lo que hace que el error de medición sea bastante grande y el filtro de Kalman no lo pueda eliminar. Esto también es causa del error en el seguimiento de la trayectoria programada, ya que la velocidad influencia directamente en el controlador principal. El error cuadrático medio en el seguimiento de trayectoria en el estado actual del vehículo es de aproximadamente 100 $[cm^2]$ en ambas coordenadas.

Finalmente, se realizan pruebas con la emulación de recepción de mensajes por medio del protocolo CAN con el módulo de expansión instalado en el controlador principal y un

segundo Raspberry. En el momento de recepción del mensaje se detiene el seguimiento de la ruta, el vehículo retrocede una distancia y cambia a una segunda ruta programada.

El prototipo tiene limitaciones físicas que se deben tomar en cuenta en el diseño de la ruta que debe seguir. El peso del prototipo hace que el motor de potencia no pueda mover el vehículo a una velocidad menor a 50 cm/s. Asimismo, el vehículo tiene una masa grande, lo cual hace que, a mayor velocidad, mayor inercia en las curvas. Esto genera que el radio de curvatura se incremente en las pruebas físicas, por lo que la ruta real del vehículo varía en cuanto a la ruta que el controlador realiza. Por otro lado, los topes mecánicos en el sistema de dirección no permiten un giro mayor a 30 grados a cada lado. Finalmente, el controlador de seguimiento de trayectoria no permite discontinuidades en la orientación del vehículo, ya que este se separa de la ruta programada.

OPORTUNIDADES DE MEJORA E INVESTIGACIÓN FUTURA

En este trabajo se desarrolla un vehículo semiautónomo capaz de reproducir una ruta de referencia programada previamente. Se utiliza un modelo simplificado de Ackerman para el controlador de algebra lineal de seguimiento de trayectoria. Sin embargo, el controlador no considera las limitaciones físicas del equipo para realizar los cálculos de la posición angular de la dirección y de la velocidad lineal del vehículo. En una futura investigación, se debe mejorar el diseño del controlador para que este tome en cuenta las limitaciones físicas del equipo y las características físicas, como el peso del vehículo, ya que esto permite eliminar los errores en el seguimiento de la ruta real, como la apertura del radio de giro que se genera por la inercia del vehículo. A una mayor masa del vehículo, se requiere una mayor fuerza centrípeta para mantenerlo en un movimiento circular. Asimismo, a una mayor aceleración angular del vehículo, el ángulo de giro de las llantas debe ser mayor para mantener el radio de curvatura de la trayectoria constante [21], [22].

Asimismo, para dar varias vueltas a una ruta circular, el controlador ve un salto del ángulo de orientación de 360° a 0°, lo cual el controlador interpreta como una vuelta completa a la posición angular de la columna de dirección hacia el lado opuesto, pero el tope mecánico de giro es de solamente 30°, lo cual causa que el vehículo se pierda de la ruta. Para corregir este error, se pueden introducir condiciones en el cálculo del ángulo de orientación, pero las condiciones incrementan dependiendo del caso. Se propone investigar la posibilidad de introducir restricciones en el modelo matemático del vehículo que permitan suprimir estas discontinuidades en el cálculo del ángulo de posición de la columna de dirección. Por otra parte, otra solución es la linealización de las ecuaciones trigonométricas del controlador de

álgebra lineal, de tal manera que esta discontinuidad no afecte el comportamiento del vehículo físico [23].

Por otra parte, el vehículo necesita que se diseñe por separado la ruta de referencia. Una oportunidad de mejora es que se active el modo manual del vehículo y este vaya tomando datos de la ruta realizada manualmente para poder replicarla posteriormente.

Finalmente, el módulo de expansión CAN se instaló con el fin de poder instalar un segundo procesador que recopile datos del entorno y le permita al vehículo tomar decisiones en referencia con la información recibida. En una investigación posterior se propone integrar el segundo procesador con un sistema de procesamiento de imágenes, que le permita al procesador principal recibir la información del entorno y reaccionar de manera apropiada.

REFERENCIAS BIBLIOGRÁFICAS

[1] "What is an Autonomous Car? – How Self-Driving Cars Work | Synopsys." Accessed: Oct. 16, 2023. [Online]. Available: https://www.synopsys.com/automotive/what-is-autonomous-car.html

[2] "SAE Levels of Driving Automation™ Refined for Clarity and International Audience." Accessed: Sep. 18, 2023. [Online]. Available: https://www.sae.org/blog/sae-j3016-update

[3] "The State of Level 3 Autonomous Driving in 2023 | AUTOCRYPT." Accessed: Sep. 18, 2023. [Online]. Available: https://autocrypt.io/the-state-of-level-3-autonomous-driving-in-2023/

[4] D. C. Conner, H. Kress-Gazit, H. Choset, A. A. Rizzi, and G. J. Pappas, "Valet parking without a valet," in *IEEE International Conference on Intelligent Robots and Systems*, 2007, pp. 572–577. doi: 10.1109/IROS.2007.4399374.

[5] F. A. Cheein and G. Scaglia, "Trajectory Tracking Controller Design for Unmanned Vehicles: A New Methodology," *J Field Robot*, vol. 31, no. 6, pp. 861–887, Nov. 2014, doi: 10.1002/rob.21492.

[6] L. Medina, G. Guerra, M. Herrera, L. Guevara, and O. Camacho, "Sliding Mode Control Approaches Applied to Trajectory Tracking for Non-holonomic Mobile Robots," *International Journal of Control, Automation, and Systems VV(X) (YYYY)*, pp. 1–17, doi: 10.1007/s12555-xxx-xxxx-x.

[7] A. Patnaik *et al.*, "Design and Implementation of Path Trackers for Ackermann Drive based Vehicles," Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.02978

[8] S. Yamakawa and K. Ebara, "Control of mobile robot by switching traveling direction and control gain," *ROBOMECH Journal*, vol. 4, no. 1, Dec. 2017, doi: 10.1186/s40648-017-0097-z.

[9] M. A. Sotelo, "Lateral control strategy for autonomous steering of Ackerman-like vehicles," *Rob Auton Syst*, vol. 45, no. 3–4, pp. 223–233, Dec. 2003, doi: 10.1016/j.robot.2003.09.002.

[10] M. R. Azizi, A. Rastegarpanah, and R. Stolkin, "Motion planning and control of an omnidirectional mobile robot in dynamic environments," *Robotics*, vol. 10, no. 1, Mar. 2021, doi: 10.3390/robotics10010048.

[11] E. Ulin-Avila and J. Ponce-Hernandez, "Kalman Filter Estimation and Its Implementation." [Online]. Available: www.intechopen.com

[12] A. Becker, *Kalman Filter From The Ground Up*, 2nd ed. 2023. Accessed: Nov. 13, 2023. [Online]. Available: https://www.kalmanfilter.net/kalman1d.html

[13] "Kalman Filters: A step by step implementation guide in python | by Garima Nishad | Analytics Vidhya | Medium." Accessed: Nov. 13, 2023. [Online]. Available: https://medium.com/analytics-vidhya/kalman-filters-a-step-by-step-implementation-guide-in-python-91e7e123b968

[14] S. Corrigan, "Introduction to the Controller Area Network (CAN) Application Report Introduction to the Controller Area Network (CAN)," 2002. [Online]. Available: www.ti.com

[15] "CAN Bus Protocol Tutorial | Kvaser." Accessed: Sep. 18, 2023. [Online]. Available: https://www.kvaser.com/can-protocol-tutorial/

[16] J. A. Cook and J. S. Freudenberg, “Controller Area Network (CAN) EECS 461, Fall 2008 *.”
[17] STMicroelectronics, “Automotive fully integrated H-bridge motor driver,” 2008. [Online]. Available: www.st.com
[18] “Ackerman Steering • Computer Science and Machine Learning.” Accessed: Oct. 16, 2023. [Online]. Available: https://www.xarg.org/book/kinematics/ackerman-steering/
[19] V. M. Alfaro Ruíz, “IDENTIFICACIÓN DE PROCESOS SOBREAMORTIGUADOS UTILIZANDO TÉCNICAS DE LAZO CERRADO,” *Revista Ingeniería*, vol. 11, no. 1–2, Jul. 2011, doi: 10.15517/ring.v11i1-2.604.
[20] O. Camacho, A. Rosales, and F. Rivas, *Control de Procesos*, 1st ed. Quito: Escuela Politécnica Nacional, 2020.
[21] “Curvature and acceleration.” Accessed: Dec. 21, 2023. [Online]. Available: https://web.ma.utexas.edu/users/m408m/Display13-4-3.shtml
[22] “6.6: Centripetal Force - Physics LibreTexts.” Accessed: Dec. 21, 2023. [Online]. Available: https://phys.libretexts.org/Bookshelves/University_Physics/Book:_University_Physics_(OpenStax)/Book:_University_Physics_I_-_Mechanics_Sound_Oscillations_and_Waves_(OpenStax)/06:_Applications_of_Newton’s_Laws/6.06:_Centripetal_Force
[23] “Lecture 24: Linearization.”
[24] E. Morgan, “HC-SR04 Datasheet,” 2014.
[25] Junye, “MOCH22A Datasheet.” 2006.
[26] “How to convert a DC motor to a servo motor with Arduino | Hobbyist.co.nz.” Accessed: Nov. 13, 2023. [Online]. Available: https://www.hobbyist.co.nz/?q=convert-dc-motor-to-servo-using-arduino
[27] Microchip Technology Inc., “MCP3004/3008.” 2008.
[28] “Raspberry Pi: Read Analog Inputs with Python (MCP3008) | Random Nerd Tutorials.” Accessed: Nov. 13, 2023. [Online]. Available: https://randomnerdtutorials.com/raspberry-pi-analog-inputs-python-mcp3008/
[29] “2-CH CAN FD HAT.” Accessed: Oct. 16, 2023. [Online]. Available: https://www.waveshare.com/wiki/2-CH_CAN_FD_HAT#Introduction
[30] E. and C. E. Military Institute of Science and Technology. Department of Electrical, Jāhāṅgīranagara Biśvabidyālaẏa. Institute of Life Sciences Information Technology, IEEE Communications Society. Bangladesh Chapter, Bangladesh Computer Society, Bangladesh Electronics Society, and Institute of Electrical and Electronics Engineers, “Design and Implementation of SPI Bus Protocol with Built-In-Self-Test Capability over FPGA”.
[31] P. Pallavi*, V. Priyanka, and Dr. Y. P. Sai, “Design and Verification of Serial Peripheral Interface (SPI) Protocol,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 6, pp. 793–796, Mar. 2020, doi: 10.35940/ijrte.F7356.038620.
[32] “Learn about the Serial Peripheral Interface (SPI) · VectorNav.” Accessed: Dec. 05, 2023. [Online]. Available: https://www.vectornav.com/resources/inertial-navigation-primer/hardware/synccomm

ANEXO A: DESCRIPCIÓN DE LOS SENSORES INSTALADOS EN EL VEHÍCULO

En esta sección se describirán los sensores que han sido instalados en el vehículo que permiten la obtención de información del entorno, así como la velocidad lineal del vehículo y la posición angular de la columna de dirección.

Sensores de distancia

El vehículo tiene 4 sensores de distancia ultrasónicos, los cuales se encuentran ubicados al frente, atrás y a los lados del vehículo. El modelo de los sensores es el HC-SR04, mostrados en la Figura 19.

Figura 19. Sensor Ultrasónico HC-SR04.

El voltaje de alimentación del sensor es de 5V y este tiene dos señales de control que permite la medición de la distancia. La señal TRIG es una señal de entrada que hace que el sensor emita un ultrasonido desde el emisor. Por otra parte, la señal ECHO es una señal de salida que permite medir la distancia a la que se encuentra un objeto por medio de la duración del pulso que emite esta señal. La duración del pulso es igual a la duración que se demora la señal desde que fue emitida hasta que es receptada en el receptor. Luego, utilizando la

velocidad del sonido se calcula la distancia a la que se encuentra el objeto en el que la señal rebotó. El sensor tiene un rango de medición de hasta 4 m. El sensor emite una señal de sonido, la cual se expande en forma de onda, sin embargo, el receptor mide la señal que rebota del obstáculo, por lo cual este debe encontrarse en línea directa del sensor y la superficie de reflexión debe ser perpendicular a la dirección de avance de la onda de sonido [24].

Medición de distancia

La distancia a la que se encuentra un obstáculo desde el vehículo se mide utilizando la velocidad del sonido. Se calcula al multiplicar el tiempo en el que se demora la señal desde que es emitida hasta que es recibida, multiplicada por la velocidad del sonido (34300 cm/s), dividido para dos, ya que la señal recorre dos veces la distancia.

$$s = \frac{t \cdot v_s}{2}$$

Donde s es la distancia a la que se encuentra el objeto, t es el tiempo en el que se demora la señal desde ser emitida hasta ser recibida y v_s es la velocidad del sonido.

Encoder

Asimismo, se instala un encoder en la llanta trasera del vehículo, con el fin de medir la velocidad lineal del vehículo. Se utiliza el encoder LM393, el cual debe ser alimentado con 3.3 V para que la señal de salida sea compatible con el nivel de entrada del Raspberry Pi 4B. El encoder utilizado es el que se muestra en la Figura 20.

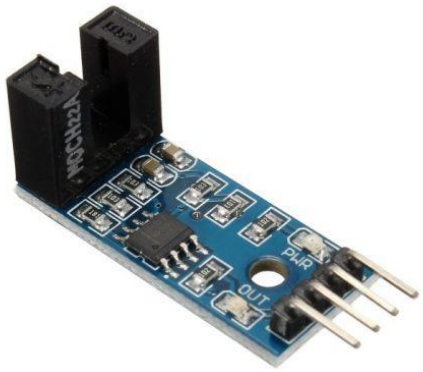


Figura 20. Encoder LM393.

Este dispositivo emite un rayo de luz que es receptado continuamente. Si este haz de luz es interrumpido, entonces el estado de la señal de salida cambia de alto a bajo. Este dispositivo tiene una salida analógica y una salida digital. Para el robot se utiliza la salida digital del encoder [25].

Para la medición del ángulo de dirección, se diseña una rueda ranurada con una resolución de 4°. Esta se acopla a la llanta trasera del vehículo, lo que le permite girar en conjunto con esta. La rueda ranurada tiene 45 ranuras, y se mide la transición de la señal de salida en ambas direcciones. El plano de la rueda ranurada que se diseña se muestra en la Figura 21, las medidas se encuentran en milímetros.

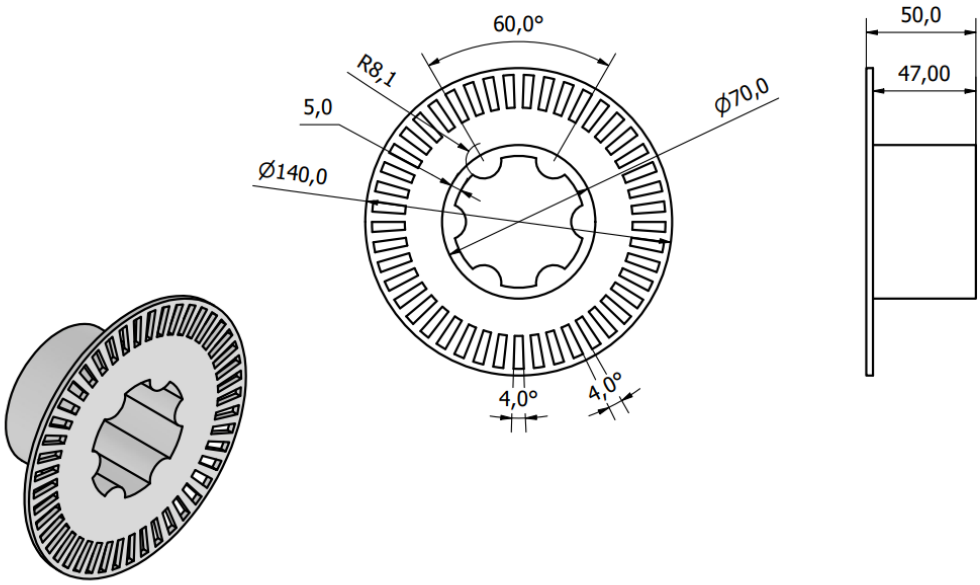


Figura 21. Rueda ranurada para la medición de velocidad.

Medición de velocidad lineal

La velocidad lineal del vehículo se mide en cada periodo de muestreo. Se cuenta en cada periodo el número de transiciones de la señal en ambos sentidos, se multiplica por pi y por el diámetro de la rueda que es de 27 cm y se divide para el tiempo de muestreo por el número de transiciones de la señal en una rotación, el cual es 90 con la rueda ranurada diseñada.

$$v = \frac{\pi \cdot n \cdot d_{wheel}}{90 \cdot T_0}$$

Donde v es la velocidad lineal del vehículo, n es el número de transiciones de la señal del sensor, d_{wheel} es el diámetro de la rueda y T_0 es el tiempo de muestreo.

Sensor de posición angular de la columna de dirección

Para medir la posición angular de la columna de dirección del vehículo se utiliza el mismo principio que utiliza un servomotor para medir la posición angular. Para esto, se utiliza el motor DC de dirección del vehículo con la caja reductora de velocidad acoplada con la columna de dirección del vehículo. Se mide la posición angular de la columna de dirección por medio de un potenciómetro y se controla la posición por medio del error entre la posición actual y la referencia [26]. En la Figura 22 se muestra el funcionamiento de este sistema.

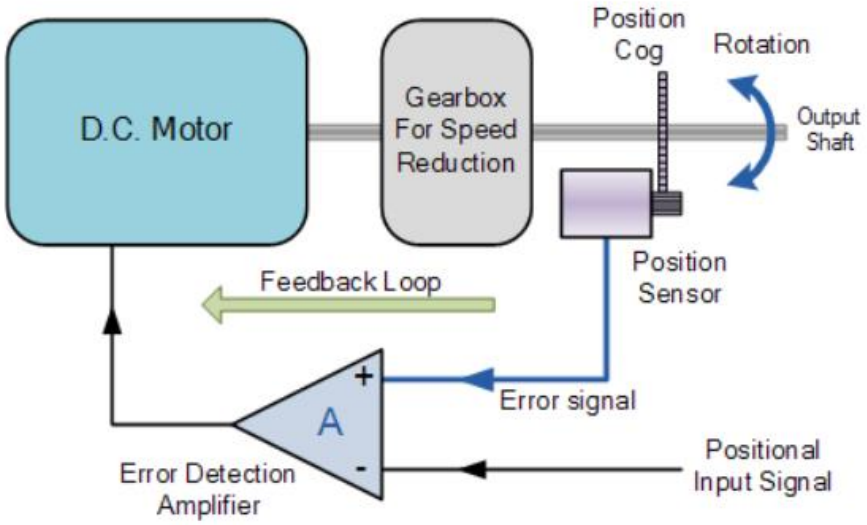


Figura 22. Diagrama de funcionamiento de un motor DC como un servomotor.

Potenciómetro

Se utiliza un potenciómetro con switch, debido a su mayor tamaño, para medir la posición angular de la columna de dirección, por medio del voltaje relativo medido en el potenciómetro con respecto al voltaje de alimentación. Se alimenta el potenciómetro con 3.3 V y se mide el voltaje de acuerdo con la posición del mando del potenciómetro. El potenciómetro utilizado se muestra en la Figura 23.

Figura 23. Potenciómetro para la medición de posición angular.

El diagrama de conexión del potenciómetro se muestra en la Figura 24.

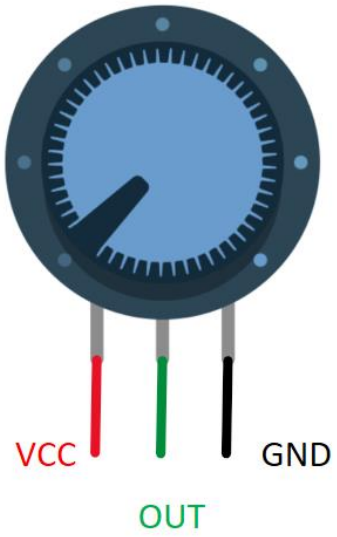


Figura 24. Diagrama de conexión del potenciómetro.

Convertidor análogo digital

La salida del potenciómetro es analógica. Sin embargo, el Raspberry Pi 4B solamente tiene entradas digitales. Debido a esto, se necesita utilizar un convertidor análogo digital para

transformar la señal de salida analógica del potenciómetro en una señal digital que sirva de entrada para el Raspberry PI. Se utiliza el módulo MCP3008, el cual es un convertidor análogo digital de 8 canales, que utiliza el protocolo de comunicación SPI del Raspberry PI para transmitir la señal digital al Raspberry, el cual se introduce en el Anexo I [27]. En la Figura 25 se muestran los pines del módulo MCP3008.

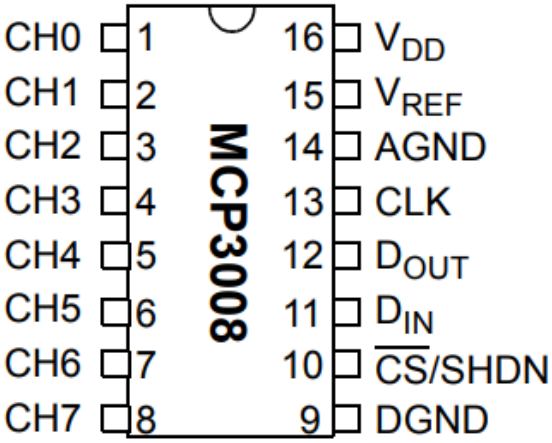


Figura 25. Pines del módulo MCP3008.

La salida del potenciómetro se conecta en el canal 0 del módulo. El pin 15 y 16 se conectan a 3.3V y los pines 14 y 9 se conectan a tierra. El pin 10 se conecta a la entrada serial SPI del Raspberry, en este caso el pin 26. Finalmente, los pines 11, 12 y 13 se conectan a los pines SPI MOSI, SPI MISO y SPI CLK del Raspberry, en este caso los pines 19, 21 y 23 respectivamente [27], [28]. En la Tabla 1 se muestra la conexión de los pines del módulo.

Tabla 1. Pines de conexión módulo MCP3008.

Pines	Conexión
1	Out Potenciómetro
9	GND
10	Pin 26 Raspberry

11	Pin 19 Raspberry
12	Pin 21 Raspberry
13	Pin 23 Raspberry
14	GND
15	VCC
16	VCC

El Raspberry Pi lee la salida digital del módulo MCP3008 como un valor entre 0 y 1 el cual es proporcional al voltaje emitido por el potenciómetro entre 0 y 3.3 V.

Placa PCB de Conexión del Módulo MCP3008

Se diseña una placa PCB para realizar la conexión del potenciómetro con el módulo MCP3008 y el Raspberry Pi. Para la placa se utiliza una bornera de 2 pines, un zócalo de 16 pines y espadines macho. El circuito diseñado en Proteus se muestra en la Figura 26.

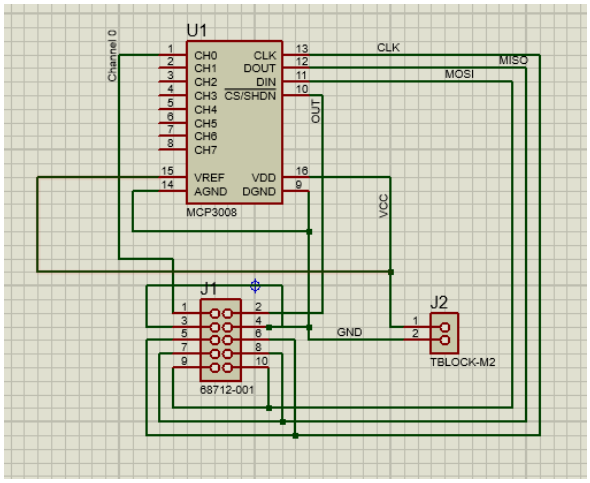


Figura 26. Circuito de conexión del módulo MCP3008.

Asimismo, en la Figura 27 se muestra el diseño de la placa PCB que se utiliza para la conexión entre el módulo y el Raspberry.

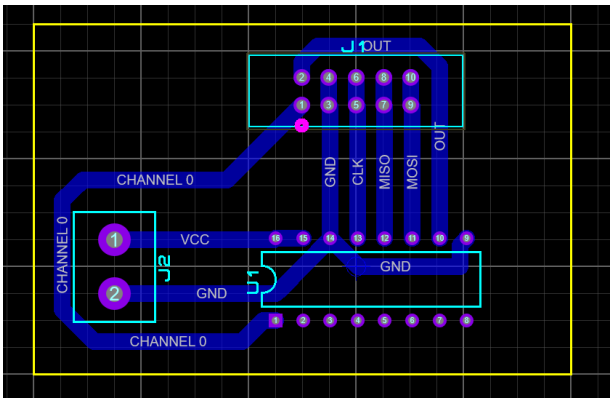


Figura 27. Diseño de la Placa PCB para la Conexión entre el Módulo MCP3008 y el Raspberry.

Cubierta del potenciómetro

Para ensamblar el potenciómetro en el vehículo, se diseña una cubierta impresa en 3D en forma de cubo que permita asegurar el potenciómetro al piso del vehículo por medio de amarras plásticas. El mando del potenciómetro sale de la cubierta para poder moverlo con la columna de dirección. El plano de la cubierta se muestra en la Figura 28.

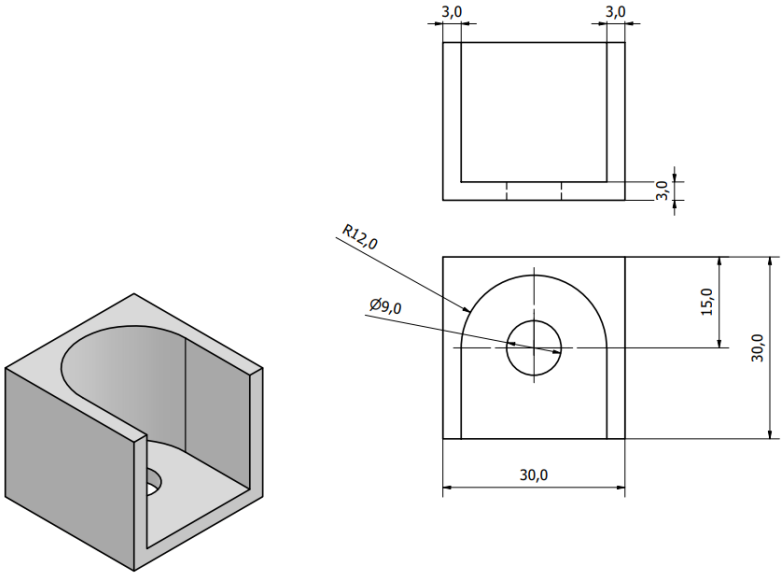


Figura 28. Cubierta del Potenciómetro.

Polea del potenciómetro

Para transmitir el movimiento de la columna de dirección al potenciómetro, se diseña un sistema de poleas para que el movimiento se transmita por medio de una banda. Cabe recalcar que para que la velocidad del movimiento sea la misma, la polea en la columna de dirección y la polea en el potenciómetro deben tener el mismo tamaño. En la Figura 29 se muestra el plano de la polea del potenciómetro. Esta tiene un agujero en la mitad que se acopla con el mando del potenciómetro que permite que los dos giren sin resbalar.

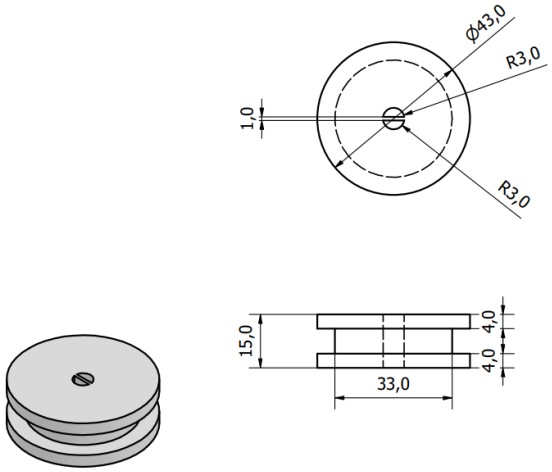


Figura 29. Polea del Potenciómetro.

Polea de la columna de dirección

Asimismo, se diseña la polea que se acopla a la columna de dirección para transmitir el movimiento rotacional al potenciómetro. Esta polea es del mismo tamaño que la polea del potenciómetro, para asegurar que el movimiento sea el mismo en la columna y el potenciómetro. Esta polea tiene un hueco transversal para instalar un prisionero en la columna de dirección, con el fin de que la polea no resbale durante el movimiento.

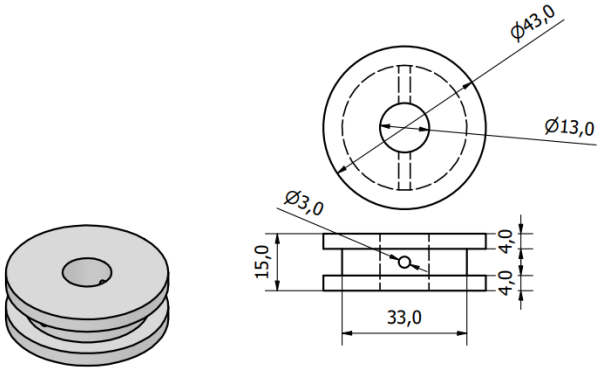


Figura 30. Polea de la Columna de Dirección.

Cálculo de la posición angular de la columna de dirección

El sistema de poleas y banda permite que el mando del potenciómetro gire en la misma cantidad que la columna de dirección del vehículo gira. Esto permite que el valor de la señal digital que recibe el Raspberry varie en dependencia del giro del potenciómetro.

La columna de dirección tiene un rango de -30° a 30°, lo que hace que el potenciómetro devuelva valores al Raspberry en un rango de 0.65 a 0.35. Ya que el potenciómetro tiene un comportamiento lineal, se calcula el ángulo de posición por medio de la siguiente ecuación lineal.

$$\psi = -200 \cdot pot + 100$$

Donde ψ es el ángulo de giro de la columna de dirección y pot es el valor que el Raspberry lee del voltaje que devuelve el potenciómetro.

ANEXO B: DIAGRAMA DE PINES GPIO DEL RASPBERRY PI 4B

En este anexo se detalla el diagrama de los pines GPIO del Raspberry Pi 4B, además de la conexión de las entradas y salidas de los dispositivos conectados a estos pines. Finalmente, se detalla el diseño de la placa PCB para la conexión entre el procesador y el hardware controlado por este.

En la Figura 31 se muestran los pines GPIO del Raspberry y su designación [28].

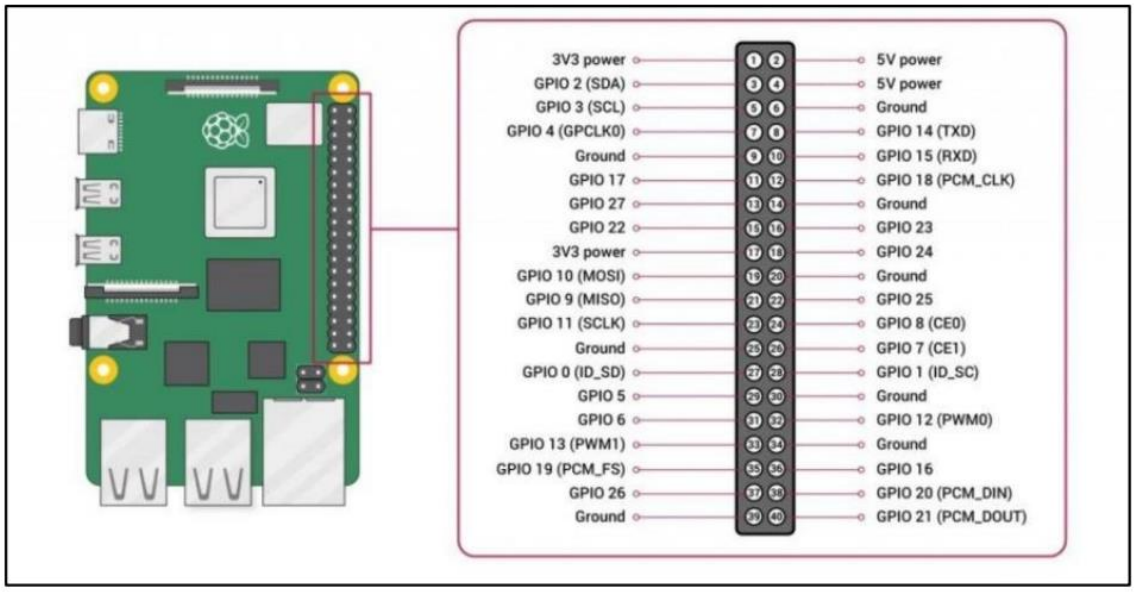


Figura 31. Diagrama de Pines del GPIO del Raspberry Pi 4B.

Los pines que se utilizan para el control de las señales del sistema se presentan en la Tabla 2.

Tabla 2. Pines GPIO.

Dispositivo	Señal	Pin
Sensores de Distancia	Trigger Frente	16
	Trigger Derecha	11
	Trigger Izquierda	13
	Trigger Atrás	15
	Echo Frente	18
	Echo Derecha	29
	Echo Izquierda	31
	Echo Atrás	22
Monster Moto Shield	INA1	36
	INA2	37
	INB1	27
	INB2	5
	PWM1	32
	PWM2	33
Encoder de velocidad	D	7
Luces	VCC	10
Buzzer	VCC	8
MCP3008	SPI MOSI_0	19
	SPI MISO_0	21
	SPI CLK_0	23
	SPI CE0_1	26
2-CH CAN FD HAT	SPI MOSI_1	38
	SPI MISO_1	35
	SPI CLK_1	40
	SPI CE1_0	12
	INT_1	28

Placa PCB para la conexión de los dispositivos con el Raspberry

Para simplificar la conexión entre los periféricos instalados en el vehículo con el Raspberry Pi, se diseña una placa PCB que permite la conexión en la placa de los periféricos y transmitir todo el bus de datos de 40 pines desde la placa hacia el Raspberry. Esto permite conectar y desconectar el Raspberry de forma sencilla sin necesidad de referirse al esquema de conexión de pines cada vez. Además, la placa permite que las conexiones sean más robustas, ya que se

utilizan cables planos y conectores IDC para la interfaz entre placas, sensores y el microcontrolador. En la Figura 32 se muestra el cable plano y el conector IDC que se utiliza.

Figura 32. Cable Plano y Conector IDC.

El circuito de la placa de conexión se diseña en Proteus. El circuito diseñado se muestra en la Figura 33.

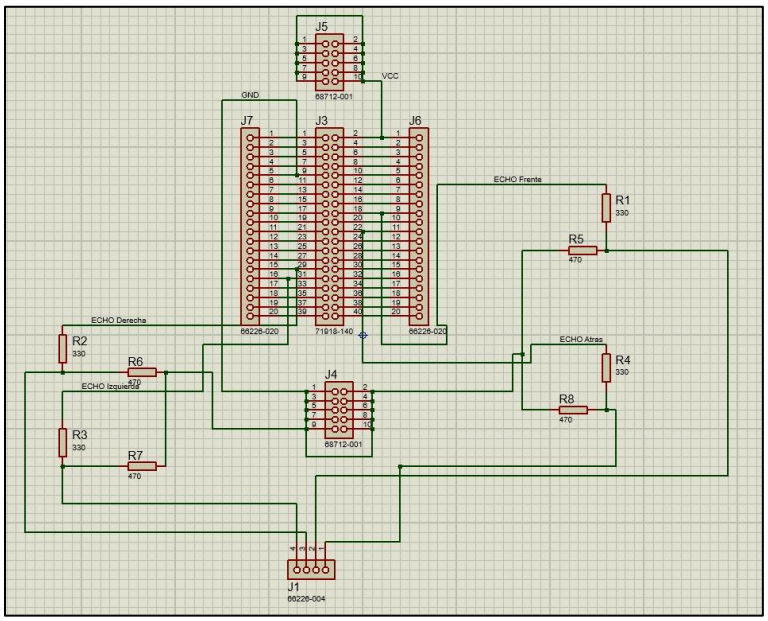


Figura 33. Circuito Diseñado para la Placa de Conexión de los Sensores con el Raspberry.

Asimismo, se presenta en la Figura 34 el diseño de la placa PCB que se utiliza en la conexión del vehículo.

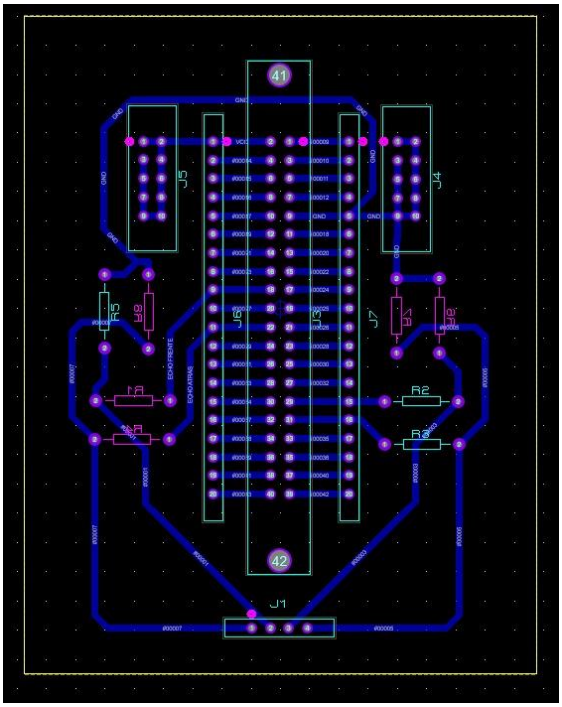


Figura 34. Diseño de la Placa PCB para la Conexión de los Sensores.

ANEXO C: MÉTODO DE DOS PUNTOS DE IDENTIFICACION DE SISTEMAS DINÁMICOS EN LAZO ABIERTO

La identificación de un proceso dinámico se basa en la curva de reacción del proceso con respecto a una entrada paso. Se utiliza un método de dos puntos para la identificación del proceso, la cual consiste en tomar medidas del tiempo en el que se demora la respuesta a llegar a dos puntos específicos en la curva de reacción y a partir de estos calcular los parámetros del sistema de primer orden más retardo [19]. La función de transferencia de un sistema de primer orden más retardo es el siguiente.

$$G(s) = \frac{K \cdot e^{-t_0 s}}{\tau \cdot s + 1}$$

Donde K es la ganancia del sistema se calcula como el cociente del cambio de la salida con respecto al cambio en la entrada y la constante de tiempo del sistema, τ, y el tiempo muerto, t_0, se calculan por medio de las siguientes ecuaciones [19].

$$K = \frac{\Delta y}{\Delta u}$$

$$\tau = a \cdot t_1 + b \cdot t_2$$

$$t_0 = c \cdot t_1 + d \cdot t_2$$

Donde $a, b, c\ y\ d$ son constantes que dependen del método utilizado para obtener la ecuación del sistema. En la Figura 35 se muestran los puntos de medida en la curva de reacción que se utilizan para determinar el modelo del sistema.

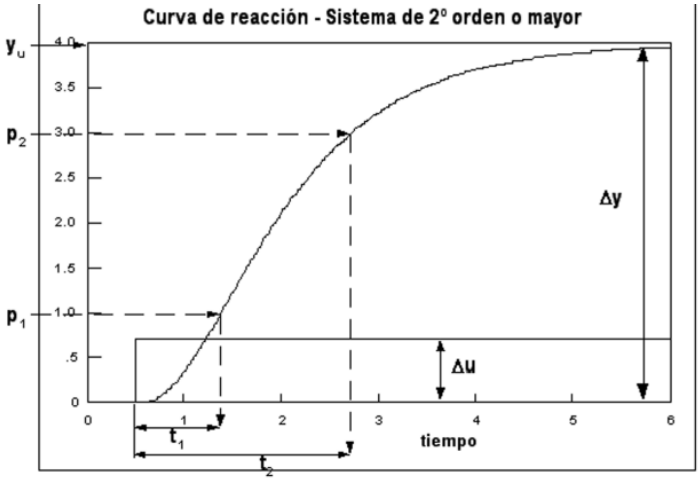


Figura 35. Curva de reacción para el método de dos puntos.

Método de Alfaro

El método de identificación de procesos en lazo abierto de Alfaro es un método de dos puntos que consiste en medir el tiempo que se demora en llegar la respuesta del sistema a un cambio escalón al valor del 25% de su valor en estado estable y al valor del 75% del valor en estado estable, los cuales representan el valor de t_1 y t_2 en las ecuaciones anteriores.

Para el método de Alfaro, se ha determinado que el valor de las constantes para determinar el modelo del sistema son las que se muestran en la Tabla 3 [19].

Tabla 3. Valor de las Constantes para el Método de Alfaro [19].

Constante	Valor
a	-0.91
b	0.91
c	1.262
d	-0.262

ANEXO D: EL FILTRO DE KALMAN UNIDIMENSIONAL

El filtro de Kalman de una dimensión sirve para reducir la varianza de la medida de una variable, por lo que este trabaja solamente con el valor de la variable medida y el valor estimado de la variable. Para el filtro, la variable medida tiene una distribución normal gaussiana, por lo que la variable estimada también tiene una distribución normal gaussiana y ambas variables tienen el mismo valor esperado. Para la derivación de la ecuación de actualización de estado se considera a x como la variable estimada y a y como la variable medida [11]–[13].

$$x \sim N(\mu, \sigma_x)$$
$$y \sim N(\mu, \sigma_y)$$
$$E\{x\} = E\{y\} = \mu$$

La ecuación de actualización de estado se calcula a partir de la ecuación de estimación del estado siguiente y la ganancia del filtro [12].

$$x_n = x_{n-1} + (1 - g_n) \cdot (y_n - x_{n-1})$$
$$g_n = \frac{\sigma_{yn}^2}{\sigma_{xn-1}^2 + \sigma_{yn}^2}$$

La ecuación de actualización de estado se deriva de la siguiente manera.

$$x_n = x_{n-1} + \left(1 - \frac{\sigma_{y,n}^2}{\sigma_{x,n-1}^2 + \sigma_{y,n}^2}\right) \cdot (y_n - x_{n-1})$$
$$x_n = x_{n-1} + \frac{\sigma_{x,n-1}^2}{\sigma_{x,n-1}^2 + \sigma_{y,n}^2} \cdot (y_n - x_{n-1})$$
$$x_n = \left(1 - \frac{\sigma_{x,n-1}^2}{\sigma_{x,n-1}^2 + \sigma_{y,n}^2}\right) \cdot x_{n-1} + \frac{\sigma_{x,n-1}^2}{\sigma_{x,n-1}^2 + \sigma_{y,n}^2} \cdot y_n$$

$$x_n = \frac{\sigma_{y,n}^2}{\sigma_{x,n-1}^2 + \sigma_{y,n}^2} \cdot x_{n-1} + \frac{\sigma_{x,n-1}^2}{\sigma_{x,n-1}^2 + \sigma_{y,n}^2} \cdot y_n$$

$$x_n = \frac{\sigma_{y,n}^2 \cdot x_{n-1} + \sigma_{x,n-1}^2 \cdot y_n}{\sigma_{x,n-1}^2 + \sigma_{y,n}^2}$$

Para una ganancia baja del filtro de Kalman el valor estimado se acerca mucho más al valor medido. La ganancia baja es cuando la varianza del valor estimado es grande y la de del valor medido es pequeña. La Figura 36 muestra el comportamiento del filtro de Kalman para una ganancia alta [12].

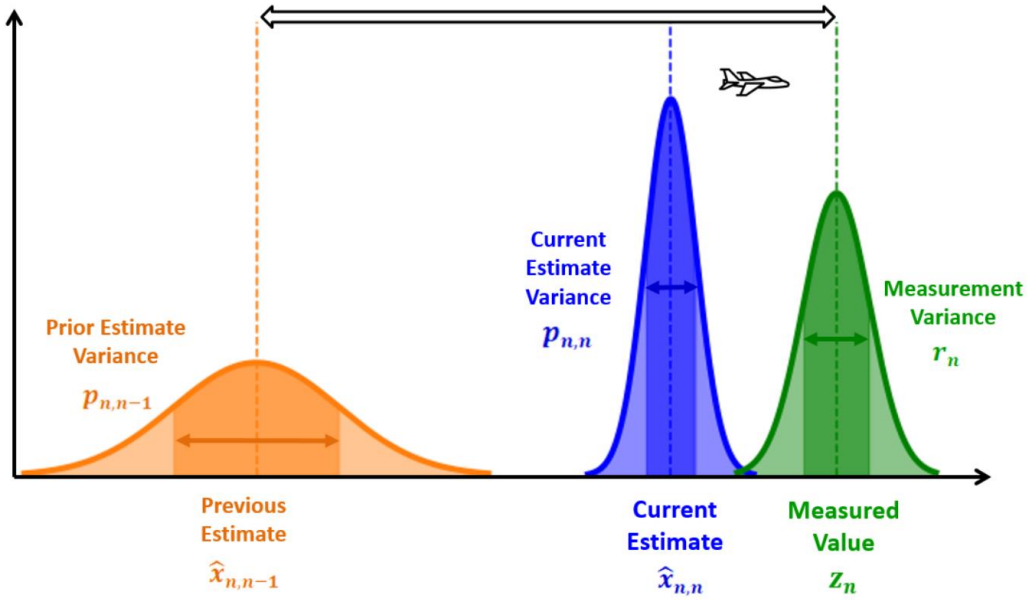


Figura 36. Comportamiento del filtro de Kalman para Ganancias Altas.

Por otra parte, el filtro de Kalman produce estimados más cercanos al estimado anterior si la ganancia es alta. Para esto, la varianza de la medida es alta y la varianza del estimado anterior es baja [12].

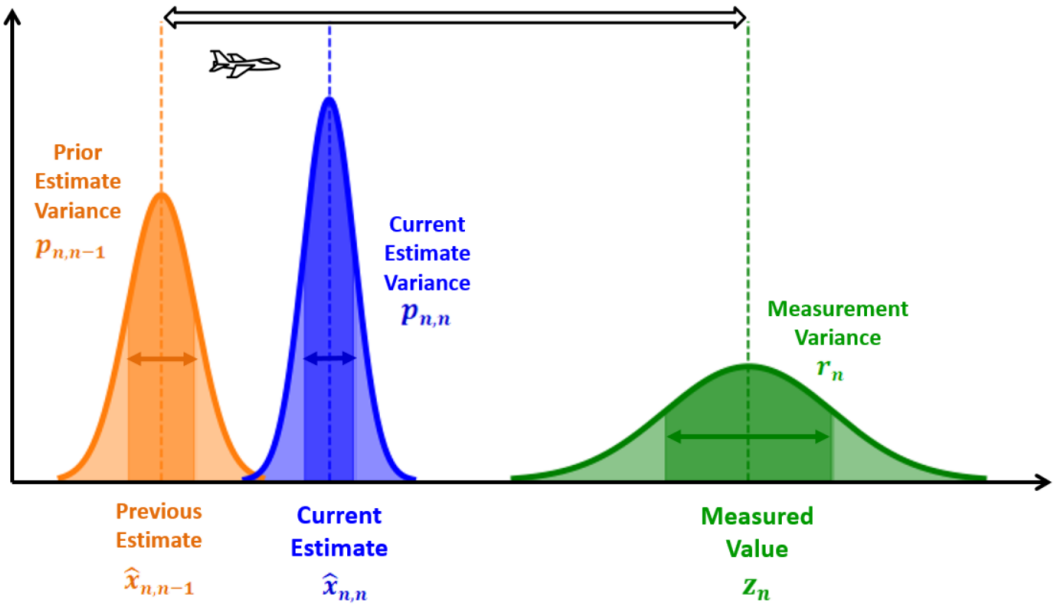


Figura 37. Comportamiento del Filtro de Kalman para Ganancias Bajas

ANEXO E: CARACTERÍSTICAS DEL PROTOCOLO DE COMUNICACIÓN CAN

El protocolo de comunicación CAN es un protocolo de tipo de transmisión de datos por medio de un bus común, lo que hace que todos los dispositivos conectados al bus de datos escuchen el mismo mensaje, pero dependiendo del arbitraje del mensaje y de su contenido, reacciona solamente el dispositivo relevante [14]–[16].

Para esto, los mensajes del protocolo tienen 4 tipos de marcos determinados: el marco de datos, el marco remoto, el marco de error y el marco de sobrecarga, siendo el marco de datos el más utilizado [14]–[16].

El marco de datos es el encargado de enviar información relevante a los nodos conectados al bus. El marco remoto es el encargado de solicitar información. El marco de error es un mensaje que viola las reglas de transmisión del protocolo, el cual se transmite solamente cuando un nodo detecta un error, lo que hace que todos los nodos detecten este error y automáticamente se vuelva a enviar el mensaje original. Finalmente, el marco de sobrecarga es similar al marco de error, pero este solicita un retraso en la transmisión de los datos dentro del bus [16].

El formato del marco estándar consiste en el inicio del marco de 1 bit; el campo de arbitraje del mensaje de 11 bits en el protocolo estándar y de 29 para el protocolo extendido, el cual se encarga de identificar el mensaje y determinar su importancia; el campo de control de 6 bits, el cual determina el largo del mensaje en el campo de datos y el tipo de marco transmitido; el campo de datos que es de 8 bytes para el protocolo estándar y de 64 bytes para el protocolo extendido, campo que contiene el dato transmitido; el campo CRC de 15 bits para el estándar y 17 o 21 bits para el extendido, este campo se encargade la detección y eliminación de errores por medio de un detector de redundancia cíclica; el campo de

reconocimiento de 2 bits, que se encarga de determinar si el mensaje enviado es válido; y, finalmente, el fin del marco de 7 bits, el cual marca el final de la transmisión [16].

En la Tabla 4 se muestra el formato del marco estándar del protocolo de comunicación CAN [16].

Tabla 4. Formato del mensaje CAN standard.

Field	Length (bits)	Description
Start of Frame (SOF)	1	Must be dominant
Identifier	11	Unique identifier indicates priority
Remote Transmission Request (RTR)	1	Dominant in data frames; recessive in remote frames
Reserved	2	Must be dominant
Data Length Code (DLC)	4	Number of data bytes (0–8)
Data Field	0–8 bytes	Length determined by DLC field
Cyclic Redundancy Check (CRC)	15	
CRC Delimiter	1	Must be recessive
Acknowledge (ACK)	1	Transmitter sends recessive; receiver asserts dominant
ACK Delimiter	1	Must be recessive
End of Frame (EOF)	7	Must be recessive

Por otra parte, el formato CAN extendido difiere el número de bits de cada parte del mensaje, como lo mencionado anteriormente. En la Tabla 5 se muestra el formato extendido de un mensaje en CAN [16].

Tabla 5. Formato extendido del mensaje en CAN.

Field	Length (bits)	Description
Start of Frame (SOF)	1	Must be dominant
Identifier – Standard and Extended Formats	11	Unique identifier corresponds to Base ID in Extended Format
Identifier – Extended Format	29	Comprised of 11 bit Base ID and 18 bit Extended ID
Remote Transmission Request (RTR) – Standard and Extended Formats	1	Dominant in data frames; recessive in remote frames. In Standard Format, the 11 bit identifier is followed by the RTR bit.
Substitute Remote Request (SRR) – Extended Format	1	Must be recessive. SRR is transmitted in Extended Frames at the position of the RTR bit in Standard Frames. In arbitration between standard and extended frames, recessive SRR guarantees the standard message frame prevails.
IDE – Standard and Extended Frames	1	Must be recessive for Extended Format; dominant for Standard Format.
Reserved r0 – Standard Format	1	Must be dominant
Reserved r1, r0 – Extended Format	2	Must be recessive
Data Length Code (DLC)	4	Number of data bytes (0–8)
Data Field	0–8 bytes	Length determined by DLC field
Cyclic Redundancy Check (CRC)	15	
CRC Delimiter	1	Must be recessive
Acknowledge (ACK)	1	Transmitter sends recessive; receiver asserts dominant
ACK Delimiter	1	Must be recessive
End of Frame (EOF)	7	Must be recessive

El protocolo es basado en el modelo estándar de interconexión de sistemas abiertos (OSI). Este estándar se lo conoce también como el modelo de 7 capas, ya que se compone de elementos independientes que describe los requisitos de comunicación a distintos niveles de abstracción [14], [16]. Las 7 capas son las siguientes: la capa de aplicación, la cual describe como programas de aplicación se conectan a la red; la capa de presentación, la cual define la compresión de datos o la encriptación de datos; la capa de sesión, la cual se encarga de manejar y terminar las conexiones entre aplicaciones cooperativas; la capa de transporte, la cual transfiere datos entre usuarios y controla errores; la capa de red, la cual se encarga de enrutar la red; la capa de datos, la cual se encarga de sincronizar los datos transmitidos y controlar los errores generados; y la capa física, la cual define las especificaciones físicas de los dispositivos en la red como las especificaciones eléctricas de los dispositivos [16].

El protocolo de comunicación CAN se maneja en las últimas dos capas del estándar internacional, por lo que se maneja en la capa física y de datos, como lo observado en la Figura 38 [14].

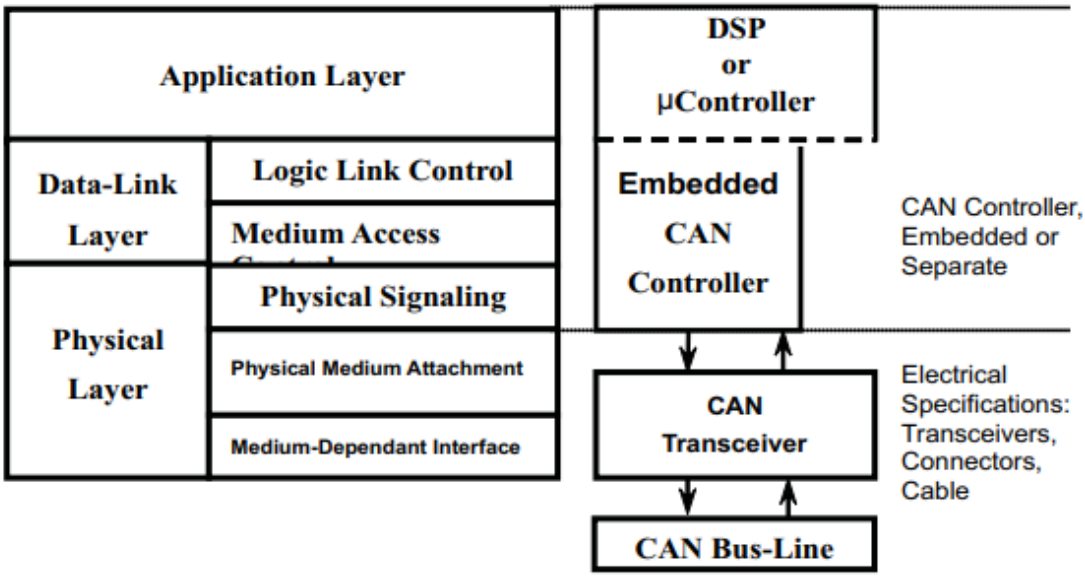


Figura 38. Arquitectura estándar del protocolo CAN.

Descripción del módulo CAN instalado en el Raspberry Pi 4B

El módulo de expansión CAN instalado en el vehículo es un módulo de dos canales aislados diseñado para el Raspberry Pi. Este módulo contiene varias protecciones contra sobre picos de corriente y voltaje, asimismo como una alta inmunidad a interferencias eléctricas. Los microcontroladores que utiliza el módulo de expansión es el MCP2515 y el SI65HVD230, los cuales son el controlador y el transmisor. El voltaje de operación de este módulo es de 5 V y se puede regular la salida a 3.3 o 5 V [29]. El módulo para instalar en el Raspberry se muestra en la Figura 39.

Figura 39. 2-CH CAN FD Module.

La interfaz de este módulo con el Raspberry es por medio del protocolo SPI, el cual se explica en el Anexo I. Este módulo tiene dos canales de comunicación aislados, los cuales se deben conectar al interfaz SPI 0 y SPI 1 del Raspberry Pi 4B [29]. El interfaz SPI 0 se utiliza en este caso para el conversor análogo digital MCP3008 descrito en el Anexo A, por lo que se utiliza el canal 1 de comunicación del módulo en el procesador principal. Para la emulación de la recepción de mensajes por medio del protocolo, se utiliza un segundo Raspberry Pi 4B conectado en el canal 0 del módulo. Los pines de conexión entre el módulo y los Raspberry se muestran en la Figura 40 [29].

PIN	Raspberry Pi (BCM2835)	Raspberry Pi (WPI)	Description
5V	5V	5V	5V Power input
GND	GND	GND	Ground
MISO_0	9 (MISO)	13 (MISO)	SPI_0 Data output
MOSI_0	10 (MOSI)	12(MOSI)	SPI_0 Data input
SCK_0	11 (SCK)	14 (SCK)	SPI_0 Clock input
CS_0	8(CE0)/7(CE1)	10(CE0)/11(CE1)	CAN_0 Chip select
INT_0	25/13	6/23	CAN_0 Interrupt Pin
MISO_1	19(MISO)/9	24(MISO)/13	SPI_1 Data output
MOSI_1	20(MOSI)/10	28(MOSI)/12	SPI_1 Data input
SCK_1	21(SCLK)/11	29(SCLK)/14	SPI_1 Clock input
CS_1	18(SPI1_CE0)/17/16/26	1/0/27/25	CAN_1 Chip select
INT_1	24/23/22/16	5/4/3/27	CAN_1 Interrupt Pin

Figura 40. Pines de conexión entre el módulo de expansión CAN y los Raspberry.

Para utilizar el módulo de expansión CAN con el Raspberry, se deben instalar las librerías correspondientes en el Raspberry, además de habilitar el protocolo de comunicación SPI de este. Asimismo, se debe configurar la tasa de transmisión de bits por medio de la comunicación serial. El proceso de instalación de librerías y de configuración de la tasa de datos de transmisión se detallan en el manual de usuario del vehículo que se adjunta como anexo a este documento.

ANEXO F: IDENTIFICACIÓN DEL SISTEMA DE VELOCIDAD LINEAL

Para realizar la identificación del sistema de velocidad lineal, se realizan varios cambios en el PWM del motor, que es la entrada del sistema. Se realizan cambios en el ancho del pulso del PWM desde 20% hasta 50% en pasos de 5%. Se mide la velocidad en cada uno de estos cambios, Las curvas obtenidas se muestran en la Figura 41.

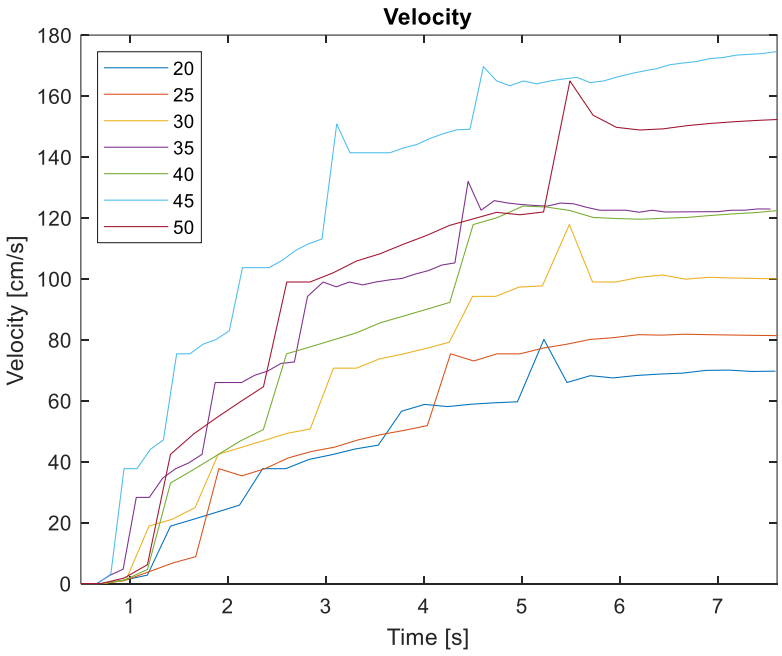


Figura 41. Curvas de reacción del sistema de velocidad lineal.

Se realiza el proceso de identificación del sistema, como se describe en la sección 5, en cada una de las curvas medidas. Los valores obtenidos se promedian entre sí y se obtiene un sistema global que se acopla de mejor manera a cada una de las curvas. El sistema identificado es el siguiente.

$$G_v(s) = \frac{3.5 \cdot e^{-0.36 \cdot s}}{2 \cdot s + 1}$$

Del mismo modo, se debe validar el modelo obtenido, al comparar la simulación del sistema a diferentes entradas con las curvas reales.

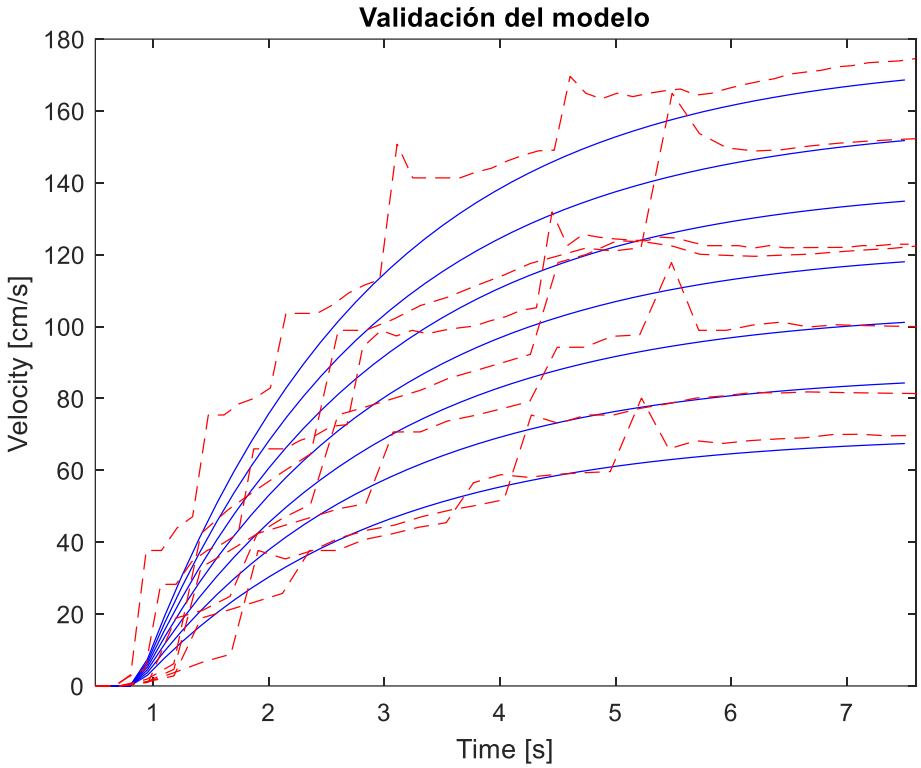


Figura 42. Validación del modelo del sistema de velocidad.

Se observa que el modelo obtenido se acopla de manera adecuada con las curvas medidas en el sistema real. Sobre todo, cuando el ancho de pulso del PWM es bajo. Sin embargo, se observa que el sistema sigue la dinámica correctamente para todos los cambios realizados con un pequeño error en el estado estable del sistema.

ANEXO G: IDENTIFICACIÓN DEL SISTEMA DE POSICIÓN ANGULAR DE LA COLUMNA DE DIRECCIÓN

Para la identificación del sistema de posición angular, se mide la velocidad en la que gira la columna de dirección con un cambio escalón con PWM con un ancho de pulso de 30% con un cambio hacia la derecha y un cambio a la izquierda. Las curvas de velocidad medidas se muestran en la Figura 43.

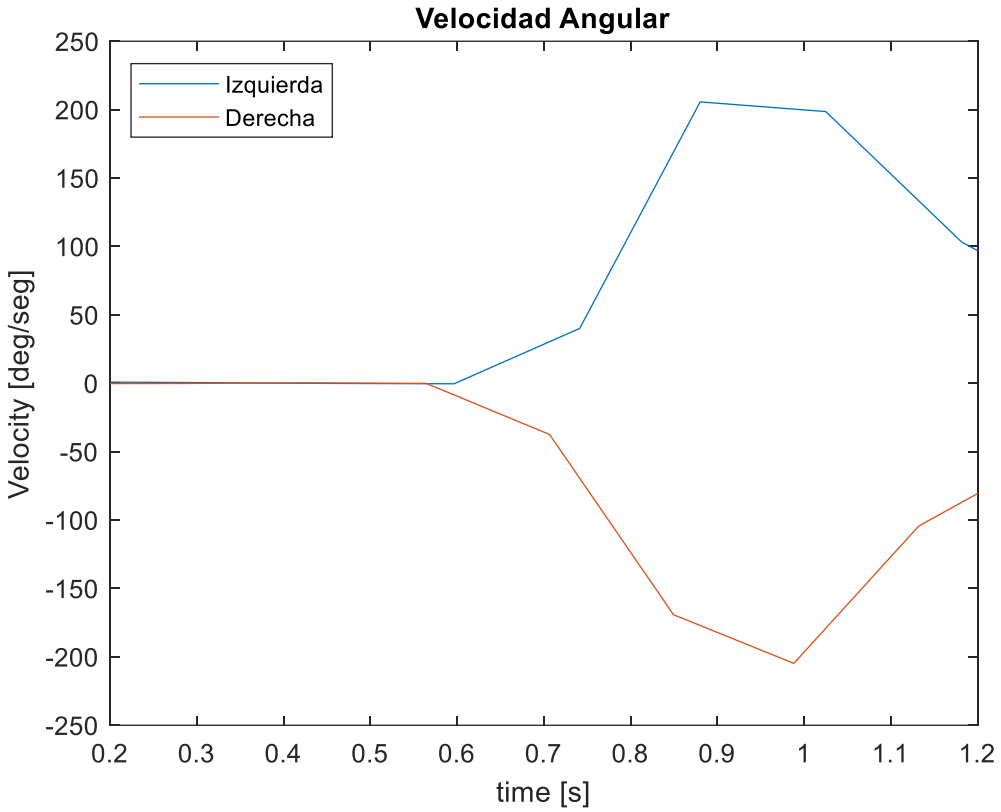


Figura 43. Velocidad Angular medida con un cambio de escalón del 30%

Se observa que la velocidad cuando gira a la derecha aumenta ligeramente más rápido que cuando gira a la izquierda. Sin embargo, se calcula el modelo de primer orden para ambas curvas y se utiliza el modelo promedio de las dos curvas para sintonizar el controlador. El modelo de velocidad angular del sistema es el siguiente.

$$G_{va}(s) = \frac{5.93 \cdot e^{-0.17 \cdot s}}{0.09 \cdot s + 1}$$

Finalmente, para determinar el sistema de posición angular a partir del sistema de velocidad angular, se agrega un integrador al sistema de velocidad. El sistema de posición angular de la columna de dirección es el siguiente.

$$G_{a}(s) = \frac{5.93 \cdot e^{-0.17 \cdot s}}{s \cdot (0.09 \cdot s + 1)}$$

Se valida el sistema obtenido con la medición de posición angular en el sistema real. La validación del sistema se muestra en la Figura 44.

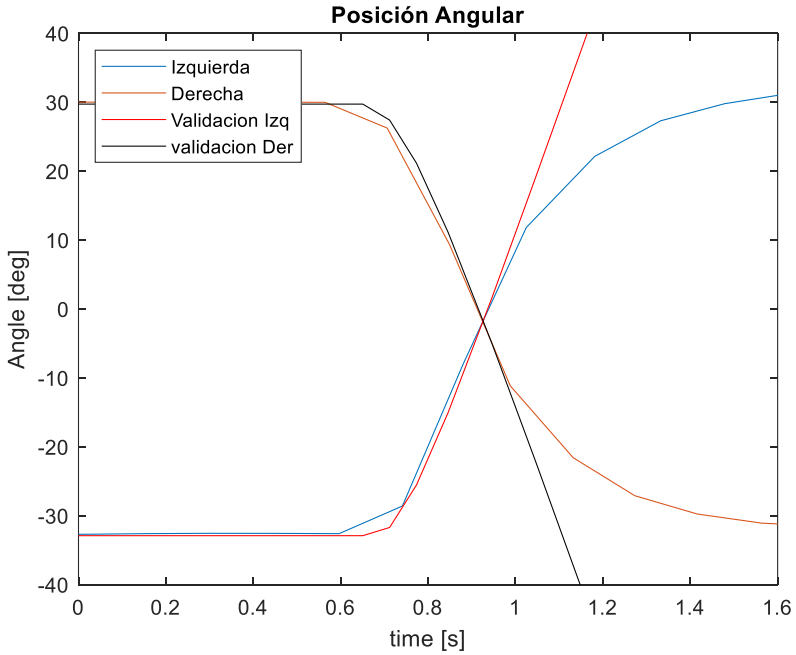


Figura 44. Validación del sistema de posición angular.

Se observa que el sistema modelado se acopla al cambio de posición angular de la columna de dirección. Debido a esto, se concluye que el modelo es preciso para la sintonización del controlador de posición angular.

ANEXO H: RUTAS DISEÑADAS PARA LA PRUEBA DE EMULACIÓN DE RECEPCIÓN DE MENSAJES POR CAN

Para la prueba de seguimiento de ruta con la emulación de la recepción de un mensaje por medio del protocolo CAN se diseñan dos rutas distintas que se cargan en el procesador principal del vehículo, el cual cambia de ruta el momento en el que el mensaje es recibido.

La primera ruta diseñada es simplemente una recta en dirección x positivo que avanza con una velocidad de 70 cm/s. La ruta diseñada es la que se muestra en la Figura 45.

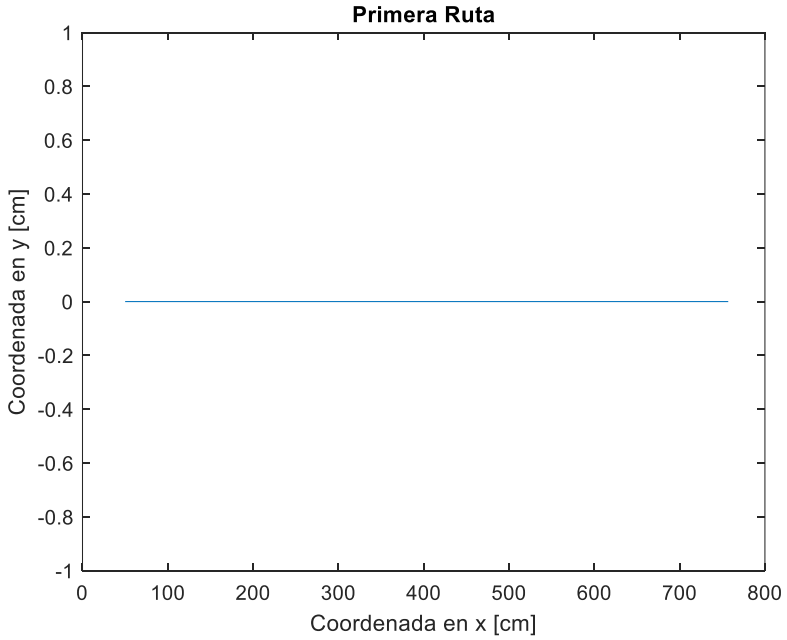


Figura 45. Primera ruta para la prueba con recepción de mensaje CAN.

Por otra parte, la segunda ruta diseñada es una circunferencia de 3 m de radio y con una velocidad lineal de 70 cm/s. La segunda ruta se muestra en la Figura 46.

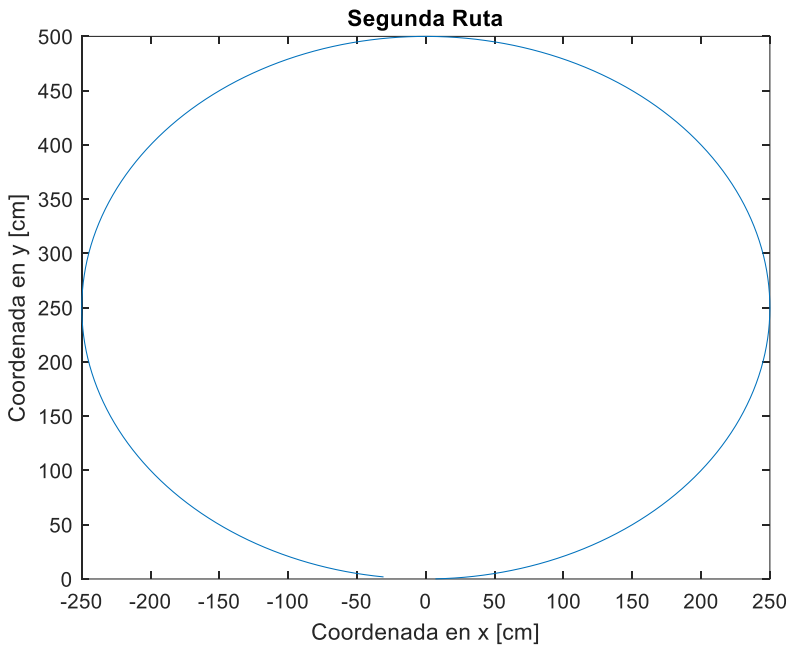


Figura 46. Segunda ruta para la prueba con recepción de mensaje CAN.

ANEXO I: PROTOCOLO DE COMUNICACIÓN SERIAL PERIPHERAL INTERFACE (SPI)

Serial Peripheral Interface (SPI) es un protocolo de comunicación serial inventado por Motorola. Este protocolo de comunicaciones se caracteriza por su rapidez de transmisión de datos y por su facilidad de instalación. Sin embargo, este protocolo se utiliza para comunicación entre dispositivos a una corta distancia, ya que la señal de reloj en cables largos puede distorsionarse por la caída de voltaje en la línea de transmisión [30]–[32].

El protocolo SPI utiliza un bus de datos full dúplex, lo que significa que el dispositivo conectado al bus de transmisión puede enviar y recibir datos al mismo tiempo. Asimismo, este protocolo es sincrónico, es decir, necesita una señal de reloj para sincronizar el muestreo de datos recibidos y enviados, lo cual permite una mayor velocidad de transmisión, ya que esta solo depende de la frecuencia del reloj, la cual es determinada por las características físicas del dispositivo conectado al bus. Los datos pueden ser muestreados en cada transición de reloj positiva o negativa [31].

Los dispositivos que se comunican por medio de este protocolo tienen una relación de maestro y esclavo. El maestro es el que determina la comunicación entre los dispositivos conectados al bus de transmisión y determina si el esclavo necesita dar una respuesta a la solicitud del maestro. EL dispositivo maestro genera la señal de reloj que se utiliza para el muestreo de los datos transmitidos. Lo que genera una ventaja en el tamaño de los datos enviados, ya que no se debe tener en cuenta los bits utilizados para la interrupción de inicio y final de la lectura de datos en el esclavo, además que no se debe configurar previamente la velocidad de transmisión entre los dispositivos conectados [30]–[32].

El protocolo de comunicación necesita un mínimo de 4 señales de datos para establecer la comunicación entre dispositivos. La primera es la señal de reloj que la genera el dispositivo maestro cuando se inicia la comunicación y este determina la velocidad de transmisión y muestreo de los datos durante la comunicación. La segunda señal es la salida de datos seriales del maestro al esclavo MOSI (Master Output Slave Input). Esta es la línea de transmisión de datos desde el dispositivo maestro hacia el dispositivo esclavo. La siguiente línea se encarga de la transmisión de datos desde el esclavo hacia el maestro. Esta línea se denomina por sus siglas en ingles MISO (Master Input Slave Output). Estas dos líneas de transmisión separadas es lo que le permite al protocolo tener la capacidad de recibir y enviar datos al mismo tiempo. Finalmente, la señal de selección de chip (CS) sirve para seleccionar el dispositivo con el que el maestro inicia la comunicación. Esta línea se mantiene en alto hasta que se inicia la comunicación, donde el maestro fuerza esta señal a bajo y permite que el esclavo se conecte al bus de comunicación. Esto permite eliminar dentro del mensaje la dirección del esclavo que debe establecer la comunicación con el maestro, lo cual permite una transmisión de datos aún más rápida [31].

El Raspberry Pi 4B cuenta con dos buses de transmisión de datos por medio de SPI. En este trabajo se utilizan los dos canales de comunicación en el procesador principal para el conversor análogo digital MCP3008 y para establecer la comunicación entre el módulo de expansión CAN y el procesador principal.

ANEXO J: MANUAL DE INSTRUCCIONES VEHÍCULO SEMIAUTÓNOMO

Manual de Instrucciones
Vehículo Semiautónomo

Universidad San Francisco de Quito

Gabriel Gómez

Quito - Ecuador

PRECAUCIONES

1. Antes de encender el vehículo, comprobar las conexiones.

2. Verificar los alrededores antes de poner a funcionar el vehículo

3. Verificar que la carga de las baterías no disminuya de 10 V durante la operación, de lo contrario puede disminuir la vida útil de las mismas.

4. En la operación manual, no forzar la dirección a más de los topes mecánicos, de lo contrario se pueden romper los engranes de la caja de reducción.

5. No rodar las llantas traseras sin carga, los engranes del motor de potencia se pueden romper.

Contenidos

Lista de figuras

Lista de Componentes

Vehículo

Se utiliza un modelo a escala 4:1 de un BMW i8 Spyder.

Figura 1: Vehículo semiautónomo terrestre.

Motores

El vehículo cuenta con dos motores DC de 12 V que se utilizan para la potencia lineal del vehículo y en la columna de dirección.

Figura 2: Motores DC.

Cajas reductoras de velocidad

En los motores DC se acoplan dos cajas reductoras de velocidad. Estas contienen engranes de plástico de distintos tamaños que permiten la reducción de la velocidad de giro de los motores DC.

Figura 3: Caja reductora del motor de potencia.

Figura 4: Caja reductora del sistema de dirección.

Driver de Motores DC

Para controlar los motores el vehículo utiliza el driver Monster Moto Shield que es capaz de soportar una corriente nominal de 14 amperios y una corriente pico de 30 amperios [1].

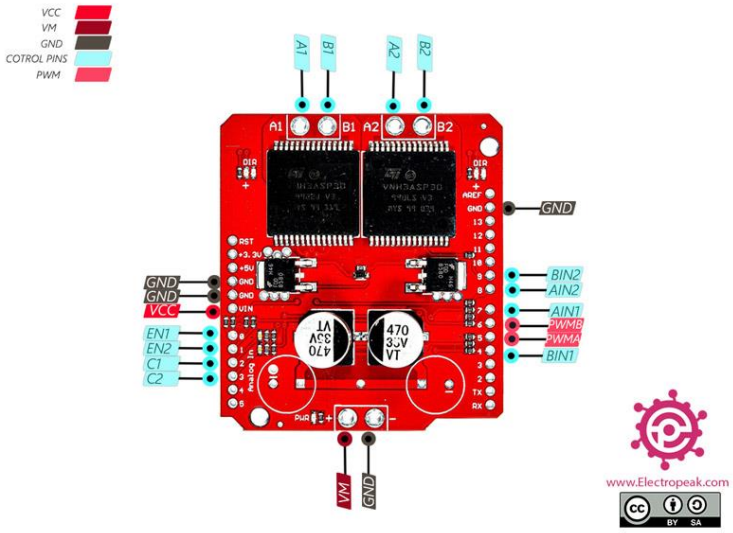


Figura 5: Driver Monster Moto Shield.

Raspberry Pi 4B

El controlador principal del vehículo es un Raspberry Pi 4B. Este se conecta a los sensores y al driver del motor, lo que permite el manejo del vehículo [2].

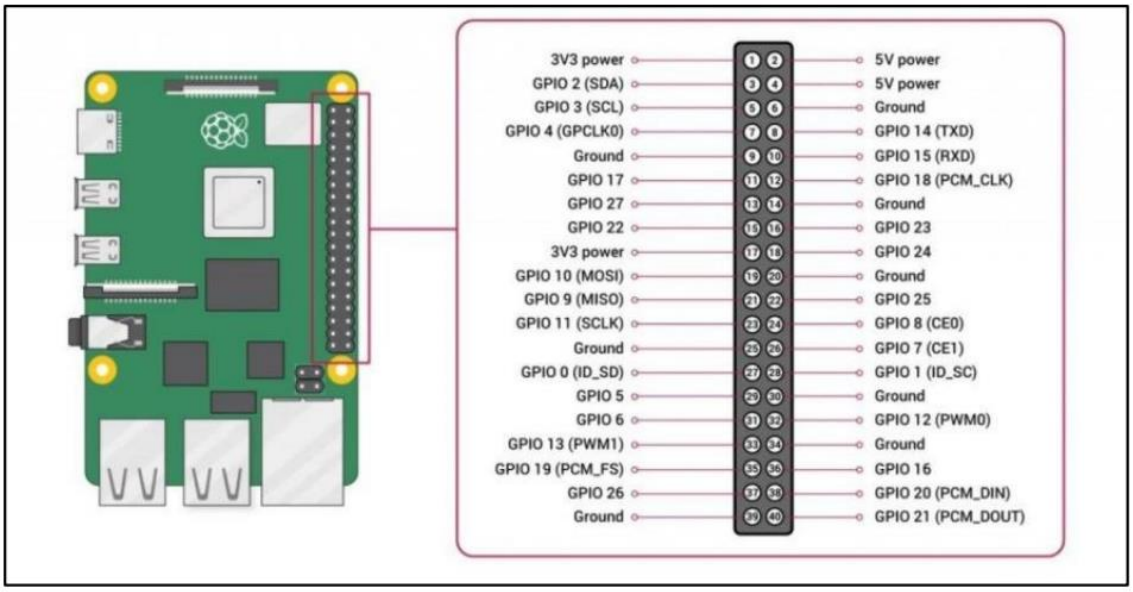


Figura 6: Raspberry Pi 4B.

Sensores de proximidad HC-SR04

El vehículo cuenta con 4 sensores HC-SR04. Uno en la parte delantera, uno en la parte trasera, uno a la izquierda y uno a la derecha [3].

Figura 7: Sensores HC-SR04

Encoder LM393

Para la medición de velocidad lineal, el vehículo tiene un Encoder LM393 instalado en la llanta trasera izquierda [4].

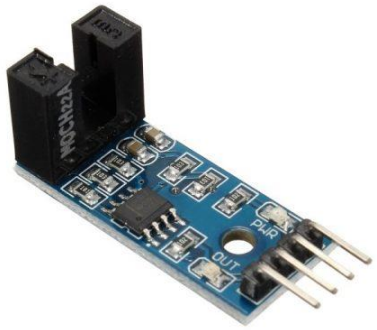


Figura 8: Encoder LM393.

Rueda dentada

Para el Encoder instalado, el vehículo cuenta con una rueda dentada con una resolución de 4 grados, instalada en la rueda trasera izquierda.

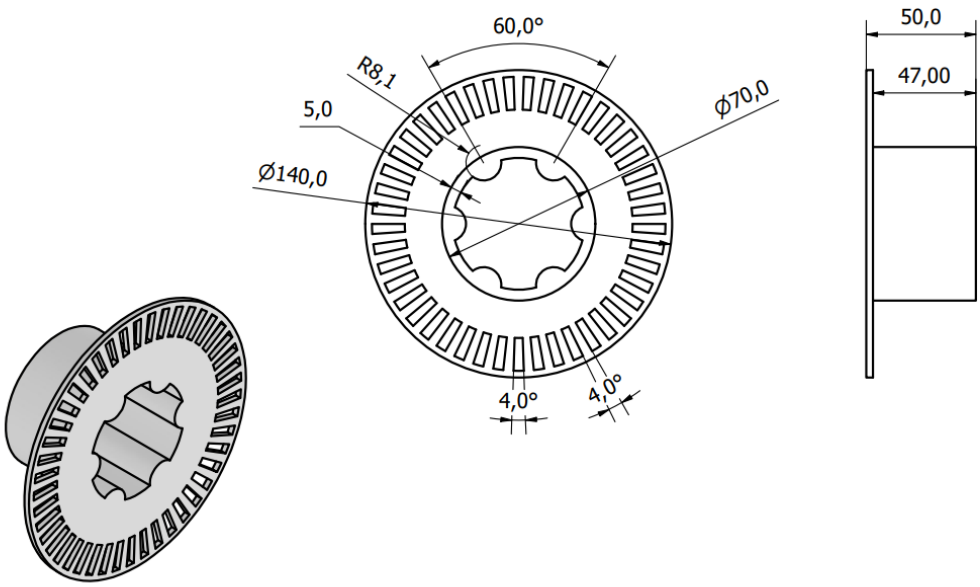


Figura 9: Rueda dentada para medición de velocidad.

Conversor Análogo Digital MCP3008

Para la medición del ángulo de posición de la columna de dirección se instala un sistema con un potenciómetro y de transmisión de movimiento con poleas. Para leer la señal del potenciómetro se utiliza un conversor análogo digital MCP3008 [5].

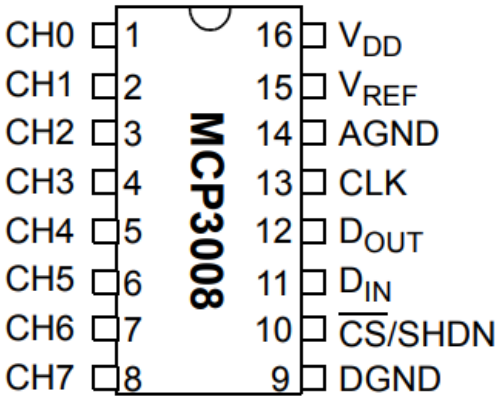


Figura 10: ADC MCP3008.

Potenciómetro

Se instala un potenciómetro a un lado de la columna de dirección para la medición de la posición angular de la dirección del vehículo.

Figura 11: Potenciómetro.

PCB para la conexión del ADC y del potenciómetro

El vehículo cuenta con una PCB para la conexión entre el potenciómetro y el ADC con el Raspberry.

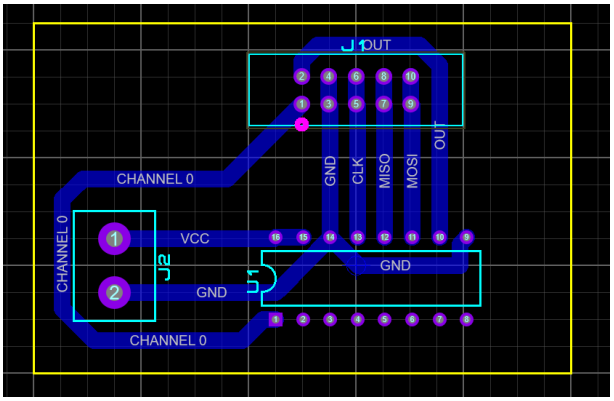


Figura 12: PCB para la conexión del ADC y el potenciómetro.

PCB para la conexión entre el Raspberry y los periféricos

El vehículo cuenta con una PCB para la conexión entre el controlador principal y los sensores instalados y el driver de los motores.

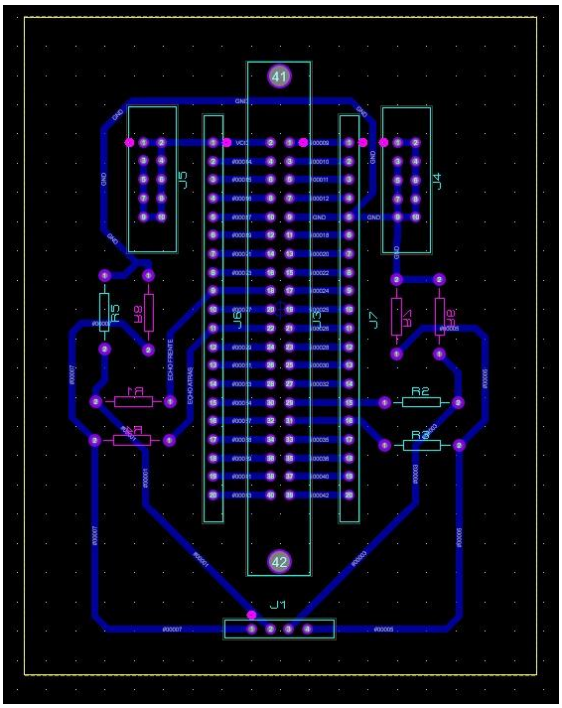


Figura 13: PCB para la conexión entre el Raspberry y los periféricos.

Placa de expansión CAN FD de dos canales

Para establecer el protocolo de comunicación CAN se instala una placa de expansión de dos canales CAN FD [6].

Figura 14: 2 CH CAN FD expansion Hat.

Baterías

El vehículo cuenta con tres baterías de 12 V y 7 Ah. Estas son baterías de plomo. Es importante no dejar que reduzca el voltaje de la batería por debajo de 10 V ya que esto puede reducir su vida útil.

Figura 15: Baterías 12 V 7Ah.

Control

El vehículo utiliza un control de play station 4 para su manejo.

Figura 16: Control de PS4.

Conexión de los pines del Raspberry Pi 4B

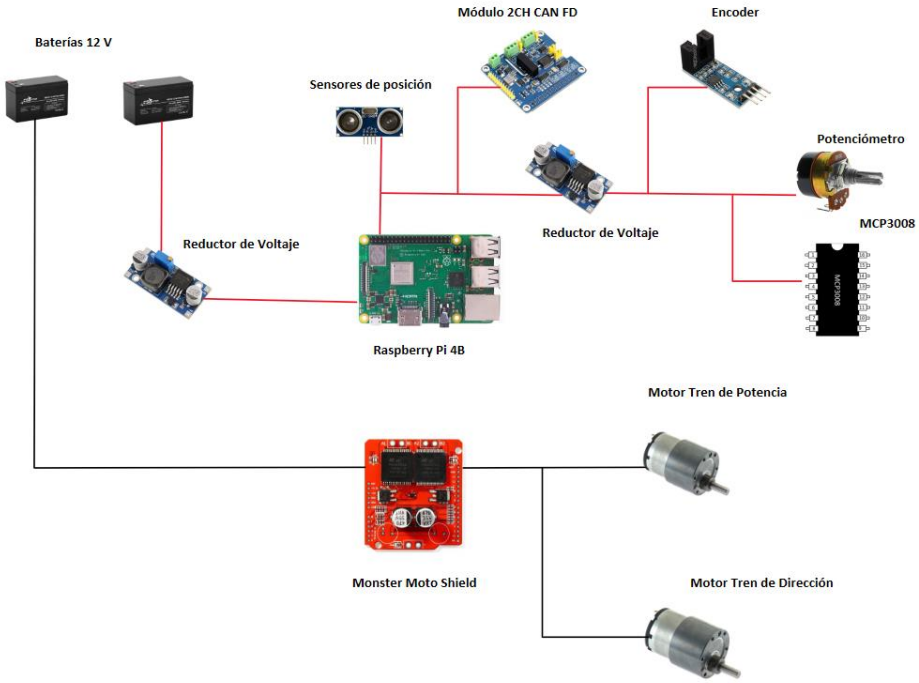
La siguiente tabla describe los pines de conexión del Raspberry Pi con las señales de control del vehículo.

Dispositivo	Señal	Pin
Sensores de Distancia	Trigger Frente	16
	Trigger Derecha	11
	Trigger Izquierda	13
	Trigger Atrás	15
	Echo Frente	18
	Echo Derecha	29
	Echo Izquierda	31
	Echo Atrás	22
Monster Moto Shield	INA1	36
	INA2	37
	INB1	27
	INB2	5
	PWM1	32
	PWM2	33
Encoder de velocidad	D	7
Luces	VCC	10
Buzzer	VCC	8
MCP3008	SPI MOSI_0	19
	SPI MISO_0	21
	SPI CLK_0	23
	SPI CE0_1	26

2-CH CAN FD HAT	SPI MOSI_1	38
	SPI MISO_1	35
	SPI CLK_1	40
	SPI CE1_0	12
	INT_1	28

Esquema de alimentación del vehículo

El siguiente diagrama muestra el esquema de alimentación de los dispositivos instalados en el vehículo.



Encendido del Vehículo

Para encender el vehículo seguir los siguientes pasos.

1. Conectar las dos baterías para el módulo central y para los motores.

Figura 17: Conexión de las baterías.

2. Verificar la carga de las baterías en los displays led.

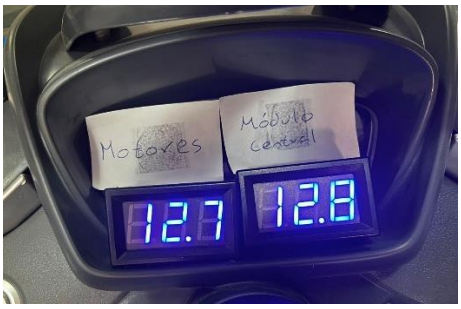


Figura 18: Displays de verificación de carga.

3. Verificar que el cable de alimentación del Raspberry se encuentre conectado.

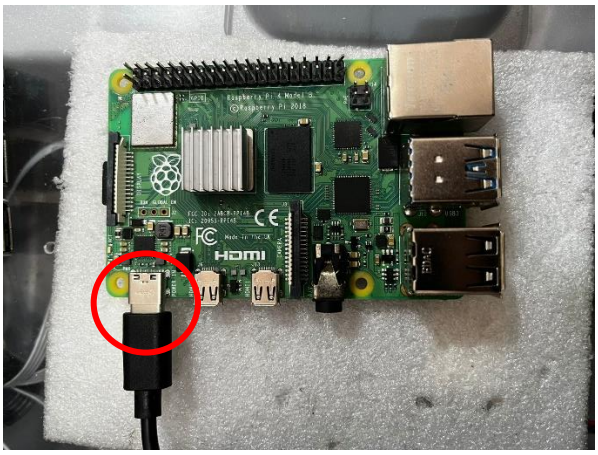


Figura 19: Conexión del cable de alimentación del Raspberry Pi 4B.

4. Verificar que el bus de datos del Raspberry esté conectado correctamente.

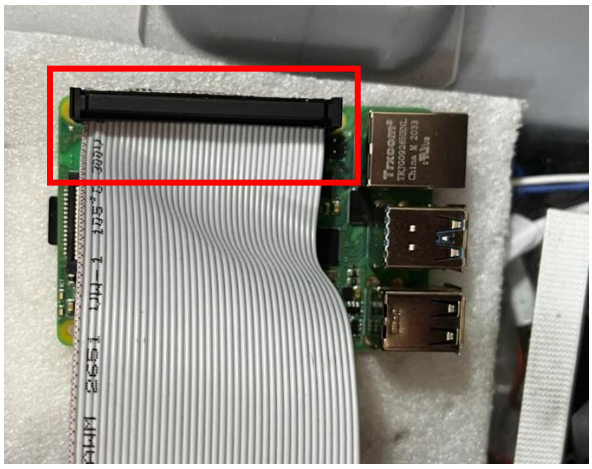


Figura 20: Conexión del bus de datos del Raspberry Pi 4B.

5. Pulsar los interruptores de encendido en el panel izquierdo del vehículo.

Figura 21: Interruptores de encendido.

Conectar el Raspberry a una red virtual

Se conecta el Raspberry a una red virtual para poder visualizar la interfaz del Raspberry en una computadora remota. Para vincular el procesador con una computadora se deben seguir los siguientes pasos.

1. Conectar el cable USB desde el Raspberry hacia la pantalla.
2. Conectar el cable micro HDMI a HDMI desde el puerto HDMI0 del Raspberry a la pantalla.

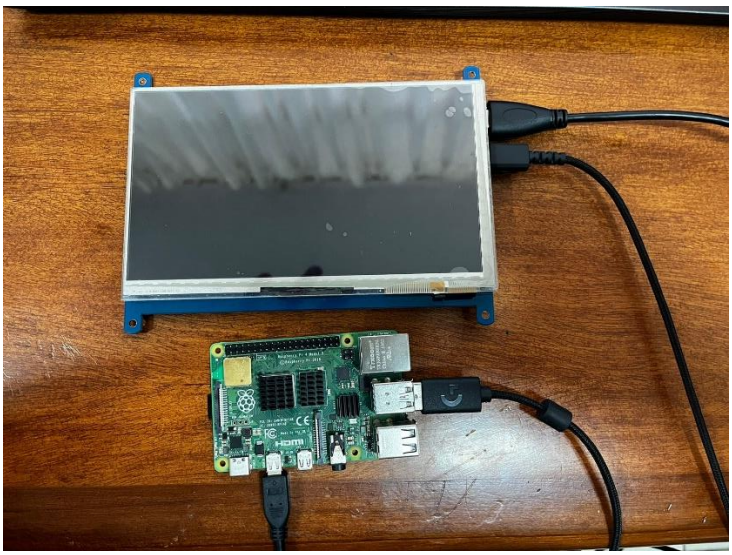


Figura 22: Conexión entre el Raspberry y la pantalla.

3. Conectar un teclado y un ratón a los puertos USB del Raspberry.
4. Encender el Raspberry.
5. Entrar al menú del Raspberry.
6. Entrar a “Preferences”.
7. Entrar a “Raspberry Pi Configuration”.
8. Entrar a “Interfaces”.

9. Verificar que las interfaces SSH y VNC se encuentren activas.

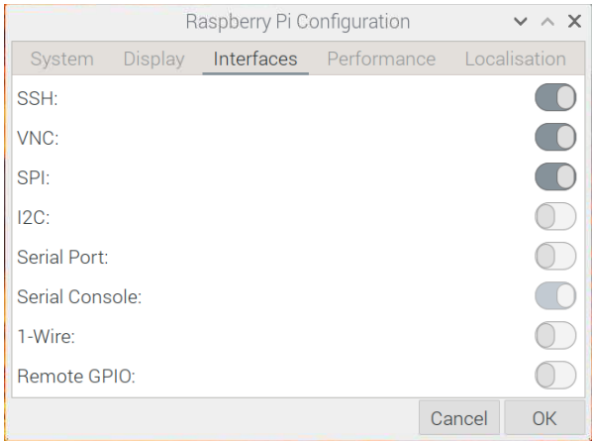


Figura 23: Interfaces del Raspberry.

10. Verificar que el Raspberry se encuentre conectado a una red Wi-Fi.

11. Entrar a una ventana de comandos.

12. Escribir el comando “ip a”.

13. Ubicar la dirección IP del Raspberry en la red conectada.

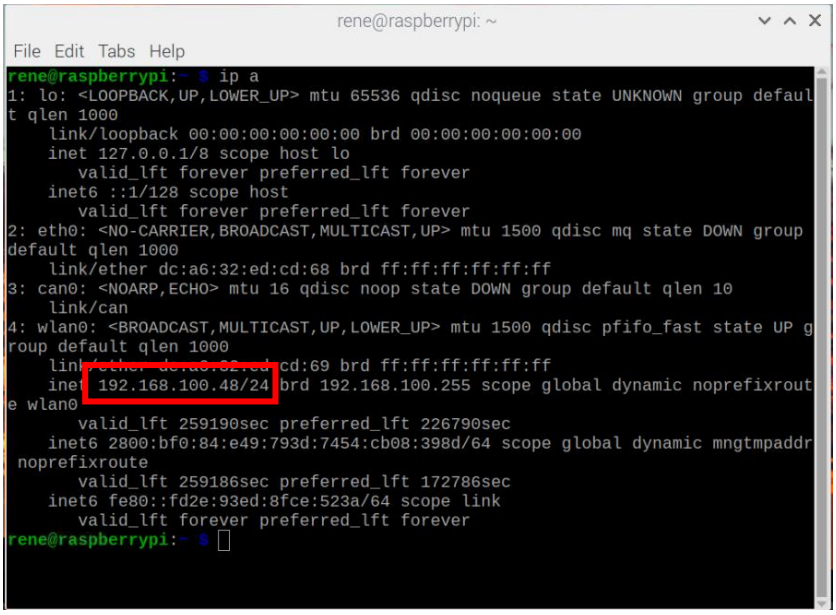


Figura 24: Dirección IP del Raspberry.

14. Verificar que la computadora remota se encuentre conectada a la misma red Wi-Fi que el Raspberry.
15. Iniciar en la computadora remota la aplicación “RealVNC Viewer”.
16. Ingresar la dirección IP del Raspberry en la aplicación “RealVNC Viewer”.
17. Conectar la computadora remota con el Raspberry por medio de la aplicación “RealVNC Viewer. El usuario del Raspberry principal es **rene** y la contraseña es **raspberry**. El usuario del Raspberry secundario es **rjativa** y la contraseña es **raspberry**.
18. Apagar el Raspberry.
19. Desconectar el teclado, el ratón y la pantalla del Raspberry.
20. Reiniciar el Raspberry.
21. Conectarse nuevamente por medio de la dirección IP y la aplicación “RealVNC Viewer” desde la computadora remota al Raspberry.

Transferencia de archivos entre la computadora remota y el Raspberry

Para transferir archivos entre la computadora remota y el Raspberry se necesita una aplicación en la computadora remota de cliente SSH, como la aplicación “Bitwise SSH Client”.

1. Iniciar la aplicación “Bitwise SSH Client”.
2. Ingresar la dirección IP, el usuario y la contraseña del Raspberry (usuario y contraseña detallados en la sección anterior).
3. Conectar el servidor SSH con el Raspberry.

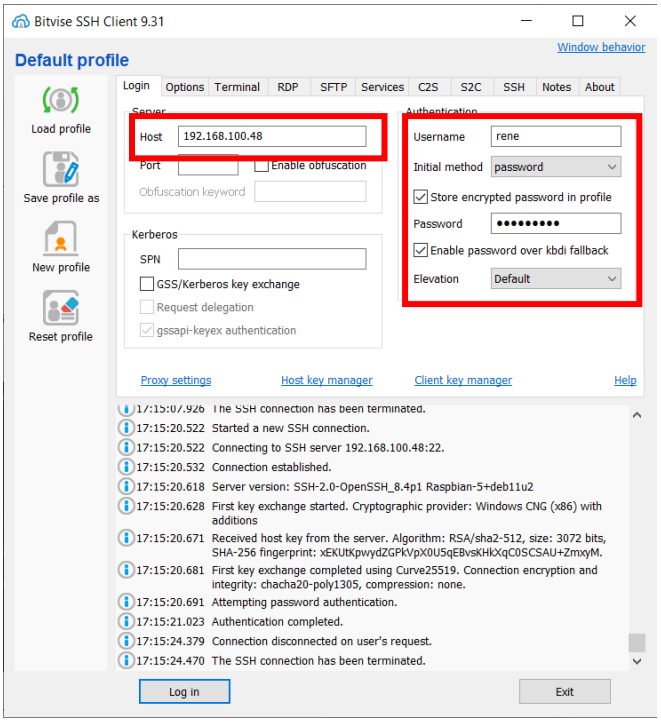


Figura 25: Conexión entre cliente SSH y el Raspberry.

4. Una vez conectado el servidor SSH, abrir una ventana SFTP.

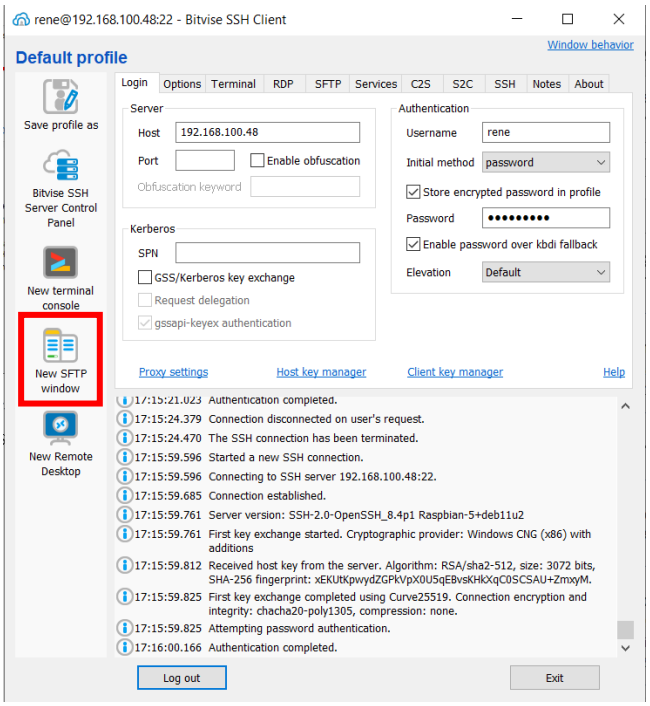


Figura 26: Ventana SFTP.

5. En la ventana SFTP ubicar la carpeta de donde salen los archivos y a donde se van a cargar tanto en el Raspberry como en la computadora remota.

6. Seleccionar los archivos a traspasar.

7. Seleccionar el botón de cargar si es el traspaso desde la computadora remota hacia el Raspberry o descargar si es desde el Raspberry hacia la computadora.

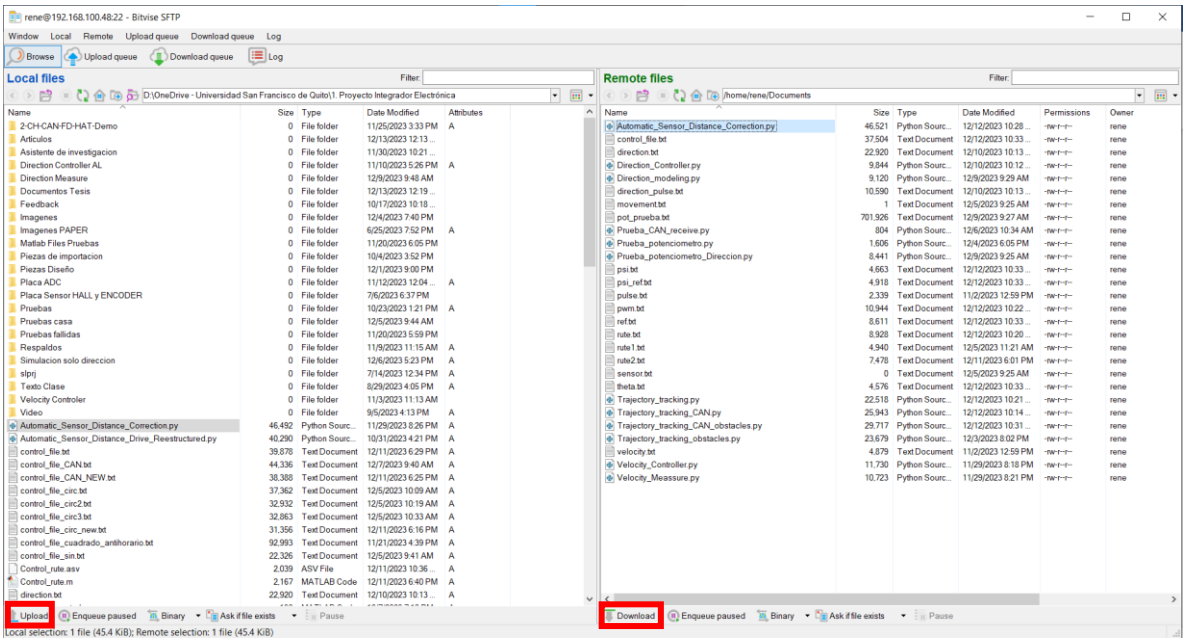


Figura 27: Traspaso de archivos entre la computadora remota y el Raspberry.

Conexión del mando de PS4 al Raspberry

El control de PS4 se conecta al Raspberry por medio de Bluetooth. Si el mando ya se encuentra emparejado, solamente aplastar el botón PS del mando y esperar a que la luz se ponga azul. Para emparejar el mando por primera vez seguir los siguientes pasos.

1. Activar el Bluetooth en el Raspberry.
2. Mantener aplastados los botones PS y Share en el mando hasta que la luz empiece a titilar.

Figura 28: Emparejamiento Bluetooth del mando con el Raspberry.

3. Entrar al menú de Bluetooth y esperar a que el Raspberry detecte el control.
4. Seleccionar el control y esperar a que la luz en el control se vuelva azul.

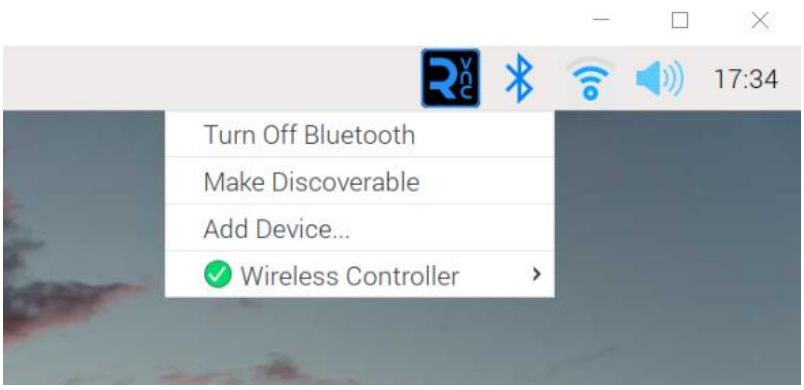


Figura 29: Menú de dispositivos conectados por Bluetooth.

Manejo manual del vehículo

Para el manejo manual del vehículo se utiliza el programa “Automatic_Sensor_Distance_Correction.py”.

1. Encender el vehículo.
2. Conectar el Raspberry a la computadora remota.
3. Entrar a “Documentos” en el Raspberry.
4. Abrir el programa “Automatic_Sensor_Distance_Correction.py”.
5. Conectar el control de PS4 al Raspberry.
6. Correr el programa.
7. Aplastar el botón triángulo.

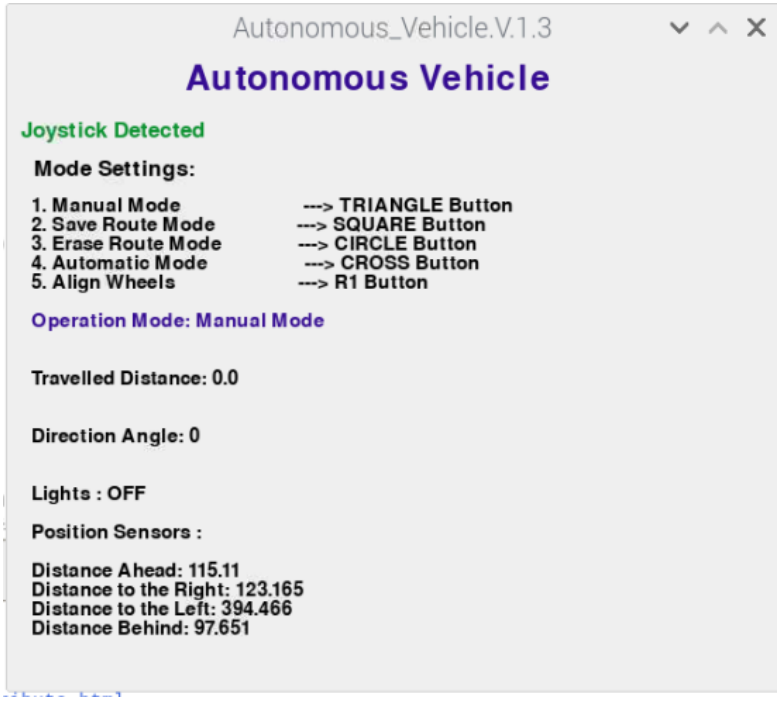


Figura 30: Interfaz del programa.

8. Los controles son los siguientes:

Figura 31: Controles modo manual.

NOTA: El vehículo no se mueve hacia adelante o atrás si los sensores de proximidad miden una distancia menor a 30 cm.

Seguimiento de trayectoria

El vehículo tiene programas instalados que le permite seguir una trayectoria predefinida en un sistema de coordenadas cartesianas en centímetros. Los programas que se pueden utilizar son “Trajectory_tracking.py”, “Trajectory_tracking_obstacles.py”, “Trajectory_tracking_CAN.py” y “Trajectory_tracking_CAN_obstacles.py”. Estos programas permiten seguir la trayectoria, seguir la trayectoria y parar el vehículo si se detecta un obstáculo por medio de los sensores de proximidad, seguir una primera trayectoria; si recibe un mensaje por medio del protocolo CAN; dar retro por un periodo de 1 segundo y cambiar a una segunda ruta, y seguir la trayectoria y parar el vehículo si se detecta un obstáculo por medio de los sensores de proximidad, seguir una primera trayectoria; si recibe un mensaje por medio del protocolo CAN; dar retro por un periodo de 1 segundo y cambiar a una segunda ruta y detectar obstáculos por medio de los sensores de proximidad y detener el vehículo si se detecta un obstáculo respectivamente.

Para el seguimiento de trayectoria se necesita diseñar una trayectoria.

Diseño de la trayectoria

1. Abrir en Matlab el programa gentray.m.
2. Descomentar las líneas del programa para la ruta deseada.
3. Ejecutar el programa.
4. Verificar en las figuras si la ruta diseñada es la deseada.
5. En el command window de Matlab, ejecutar las siguientes líneas para escribir la ruta diseñada en un archivo de texto.

```
Command Window
>> TABLA = table(xref',yref');
>> writetable(TABLA,'rute.txt');
```

Figura 32: Comandos para escribir la ruta en un archivo de texto.

6. Abrir el archivo de texto generado en el bloc de notas.

```
rute - Notepad
File  Edit  Format  View  Help
Var1,Var2
50,0
1.22464679914735e-14,0
5.99910004049915,0.0899932502024914
11.9928012958889,0.359892012959162
17.9757098396022,0.809453397601145
23.9424414577839,1.43827282922675
29.8876264947198,2.24578441279155
35.8059146851648,3.23126144237571
41.6919799692199,4.39381705517035
47.5405252854269,5.73240502959409
53.3462873377662,7.24582072682189
```

Figura 33: Archivo de ruta generado.

7. Reemplazar la separación de los datos de coma por un espacio, que los datos queden como en la figura. (Sugerencia: Utilizar la función de reemplazar todos)

```
rute - Notepad
File  Edit  Format  View  Help
50 0
1.22464679914735e-14 0
5.99910004049915 0.0899932502024914
11.9928012958889 0.359892012959162
17.9757098396022 0.809453397601145
23.9424414577839 1.43827282922675
29.8876264947198 2.24578441279155
35.8059146851648 3.23126144237571
41.6919799692199 4.39381705517035
47.5405252854269 5.73240502959409
```

Figura 34: Archivo de ruta editado.

8. Subir el archivo de ruta generado a Documentos del Raspberry.

Ejecutar el programa de seguimiento de ruta

Una vez subida la ruta diseñada al Raspberry, se puede ejecutar el programa seleccionado en el Raspberry para el seguimiento de ruta.

1. Seleccionar el programa que se quiere ejecutar.
2. Conectar el mando de PS4 al Raspberry.
3. Ejecutar el programa.
4. Los controles son los siguientes.

Figura 35: Controles para el seguimiento de trayectoria.

Descarga de los archivos de control de ejecución

Una vez ejecutado uno de los programas de seguimiento de trayectoria, se descargan los archivos de control de la ejecución del programa del Raspberry a la computadora remota. Los archivos para descargar se muestran en la siguiente figura.

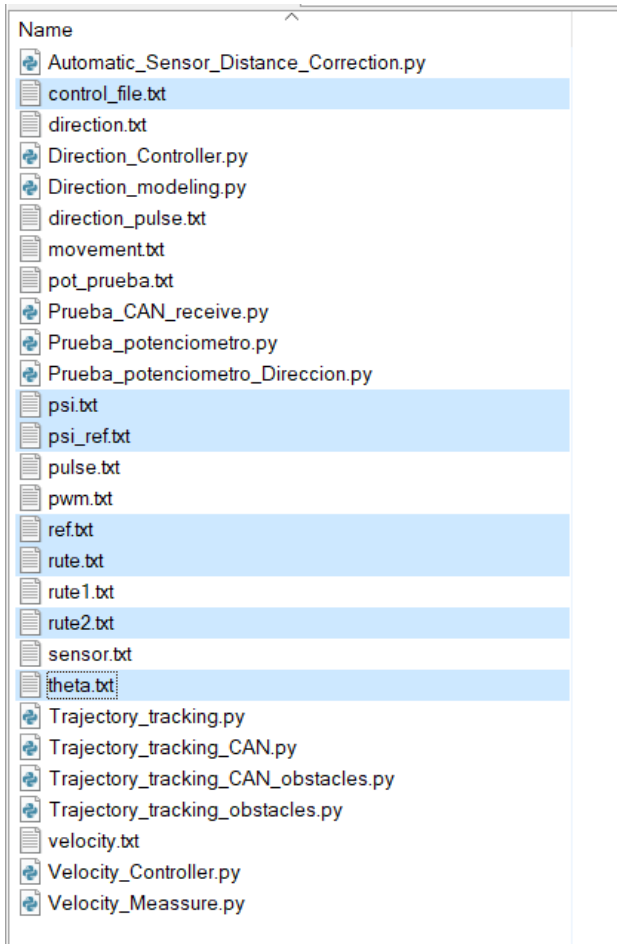


Figura 36: Archivos de control.

Una vez descargados los archivos a la computadora, se abre el programa “Control_rute.m” en Matlab y se ejecuta. Se observan en las figuras generadas si el vehículo ha seguido de manera exitosa la ruta programada.

Configuración del módulo CAN

Para el seguimiento de ruta uno de los programas, se configura el canal de comunicación CAN dentro del mismo programa. Sin embargo, para configurar el canal de comunicación por fuera del programa se siguen los siguientes pasos [6].

1. Abrir una ventana de comandos.
2. Ejecutar las siguientes líneas.

- sudo ip link set can0 up type can bitrate 1000000 dbitrate 8000000 restart-ms 1000 berr-reporting on fd on
- sudo ifconfig can0 txqueuelen 65536

3. Por medio de las líneas anteriores se configura la tasa de transmisión de bits por medio del canal.
4. Una vez que se deja de utilizar el canal, se debe desactivar la conexión, para lo cual se ejecuta la siguiente línea.

- sudo ifconfig can0 down

IMPORTANTE: Cuando se ejecuta el programa con recepción de mensajes CAN se debe finalizar el programa por medio del mando de PS4 con el botón Options. Si no se realiza este paso, el canal de comunicación queda abierto y puede ocasionar problemas en la recepción de siguientes mensajes.

Calibración de la dirección

Para calibrar la dirección el vehículo tiene instalado un programa llamado “Prueba_potenciometro_Direccion.py”. El proceso para calibrar la dirección es el siguiente.

1. Abrir el programa “Automatic_Sensor_Distance_Correction.py”.
2. Conectar el mando de PS4.
3. Ejecutar el programa.
4. Poner el modo manual.
5. Girar las llantas a un extremo.
6. Cerrar el programa.
7. Abrir el programa “Prueba_potenciometro_Direccion.py”.
8. Ejecutar el programa.
9. Si las llantas están giradas a la derecha, calibrar el valor de “Direction Angle Potenciometer Value” en 0.65, y si las llantas se encuentran giradas a la izquierda, calibrar el valor a 0.35.

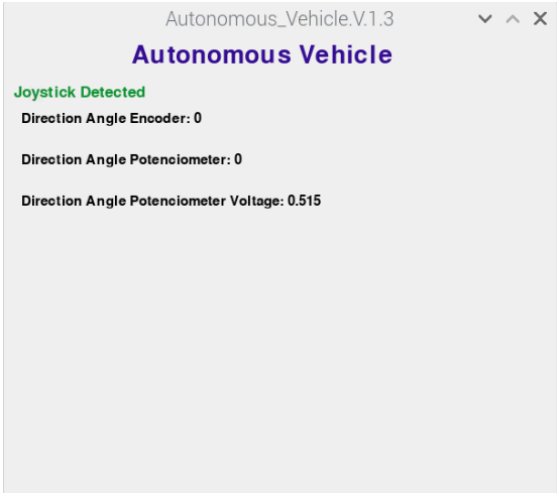


Figura 37: Interfaz de calibración de dirección.

Actualizar la versión de Python del Raspberry

Los programas de seguimiento de trayectoria y del manejo manual del vehículo utilizan una estructura de “match-case”, la cual convierte a la ejecución del programa en una especie de máquina de estado en lugar de una ejecución por polling. Esta estructura se encuentra disponible a partir de python1.3.10. Sin embargo, el Raspberry viene por defecto con la versión python1.3.9.

El proceso para actualizar la versión de Python se muestra a continuación [7].

1. Abrir una ventana de comandos.
2. Descargar la versión de Python escogida por medio de la siguiente línea, reemplazando los números por la versión de Python que se quiere descargar.

- wget https://www.python.org/ftp/python/3.10.2/Python-3.10.2.tgz

3. Extraer los archivos descargados.

- tar -zxvf Python-3.10.2.tgz

4. Entrar a la carpeta de instalación de Python.

- cd Python-3.10.2

5. Configurar la instalación.

- ./configure --enable-optimizations

6. Instalar las dependencias faltantes.

- sudo apt update
- sudo apt install -y build-essential tk-dev libncurses5-dev libncursesw5-dev libreadline6-dev libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-dev libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev libffi-dev

- sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev wget

7. Compilar la versión de Python.

- sudo make altinstall

Crear un entorno virtual

Es recomendable crear un entorno virtual para ejecutar la nueva versión de Python instalada, ya que esto permite que las versiones anteriores se mantengan instaladas, las cuales sirven para la ejecución del sistema operativo del Raspberry.

Para crear y utilizar un entorno virtual se deben seguir los siguientes pasos [8].

1. En la ventana de comandos, ejecutar la siguiente línea para crear el entorno virtual. En este caso, Autocar es el entorno virtual utilizado.

- python -m venv Autocar

2. Activar el entorno virtual.

- source Autocar/bin/activate

Definir la nueva versión de Python como defecto en el entorno

Con el entorno activado, ejecutar las siguientes líneas [7].

- cd /usr/bin
- sudo rm python3
- sudo ln -s /usr/local/bin/python3.10 python3

Finalmente, verificar la versión de Python asociada al entorno virtual.

- python3 -V

Instalación de librerías

En la nueva versión de Python se deben instalar las librerías faltantes. Es importante verificar que la librería que se requiera instalar sea compatible con la versión de Python instalada. Verificar que las librerías se instalen en el entorno virtual.

IMPORTANTE: No utilizar comandos que comiencen con sudo en el entorno virtual, ya que el entorno virtual no reconoce estos comandos.

Definir el uso de la nueva versión de Python en el compilador Thonny.

1. Abrir el programa Thonny.
2. Entrar a “Tools”.
3. Entrar a “Options”.
4. Entrar a “Interpreter”.
5. Seleccionar el ejecutable entrando a la carpeta donde se instaló el entorno virtual y seleccionar el intérprete de la nueva versión de Python.

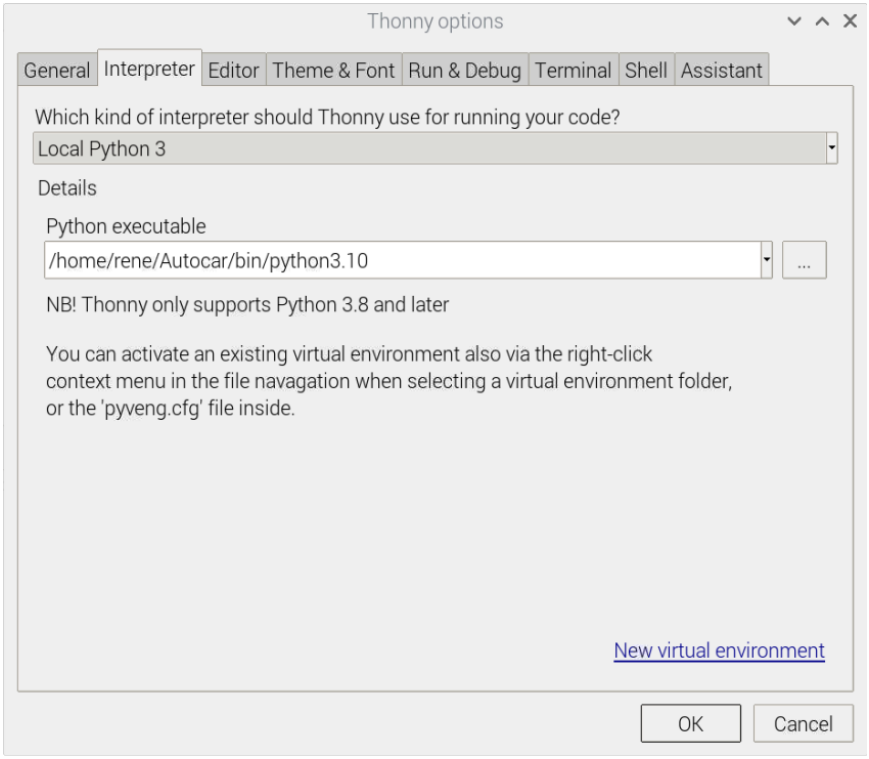


Figura 38: Selección del intérprete para el compilador.

Uso del segundo Raspberry para enviar el mensaje por CAN.

Se utiliza un segundo Raspberry para enviar un mensaje con el protocolo CAN y emular la recepción de este en el Raspberry principal. Para esto se deben seguir los siguientes pasos.

1. Conectar el Raspberry a la fuente de poder adicional instalada en el vehículo.

Figura 39: Conexión del segundo Raspberry.

2. Conectar el segundo Raspberry con el canal 0 del módulo de expansión CAN.

Figura 40: Conexión del Raspberry con el módulo de expansión CAN.

3. Conectar una tercera batería a los puertos adicionales en el vehículo.
4. Conectar el Raspberry con la computadora remota utilizando el método descrito anteriormente.

5. Configurar el canal de comunicación CAN por medio de la ventana de comandos como lo descrito en secciones anteriores.
6. Abrir el programa “Prueba_CAN_send.py”
7. Cada que se ejecuta el programa se envía el mensaje determinado en el programa.

Referencias

[1] STMicroelectronics, “Automotive fully integrated H-bridge motor driver,” 2008. [Online]. Available: www.st.com

[2] “Raspberry Pi: Read Analog Inputs with Python (MCP3008) | Random Nerd Tutorials.” Accessed: Nov. 13, 2023. [Online]. Available: https://randomnerdtutorials.com/raspberry-pi-analog-inputs-python-mcp3008/

[3] E. Morgan, “HC-SR04 Datasheet,” 2014.

[4] Junye, “MOCH22A Datasheet.” 2006.

[5] Microchip Technology Inc., “MCP3004/3008.” 2008.

[6] “2-CH CAN FD HAT.” Accessed: Oct. 16, 2023. [Online]. Available: https://www.waveshare.com/wiki/2-CH_CAN_FD_HAT#Introduction

[7] “Install latest Python version on Raspberry Pi.” Accessed: Dec. 13, 2023. [Online]. Available: https://allurcode.com/install-latest-version-of-python-on-raspberry-pi/

[8] “12. Virtual Environments and Packages — Python 3.12.1 documentation.” Accessed: Dec. 13, 2023. [Online]. Available: https://docs.python.org/3/tutorial/venv.html