Control de trayectoria para un vehículo autónomo a escala

Autor: Edward Soto, Director: Enrique Estupiñan

February 2, 2022

Abstract

Este documento muestra los resultados obtenidos del proyecto dirigido de énfasis en el área de robótica y automatización sobre el desarrollo de un Vehículo Autónomo a Escala (VAE), el cual se centra en el desarrollo de un controlador para el motor BLDC principal, un control de trayectorias y la integración de diversos sensores que le servirán al sistema de decisión el cual no está enmarcado en este proyecto.

Contenido

1 Introducción

El proyecto de investigación que se está llevando a cabo en el semillero de robótica de pregrado de ingeniería electrónica, consiste en la elaboración de un Vehículo de conducción Autónoma a Escala **VAE**, el cual requiere de la implementación del sistema de movimiento para que este pueda seguir una trayectoria previamente definida, de manera confiable, asegurando errores pequeños en el seguimiento y determinando con mediciones, el consumo de energía para realizar dicha trayectoria, con el

objetivo de que estas puedan ser usadas por el sistema de toma de decisiones. Para lograrlo también es necesario, además del Control de Movimiento **CM**, el Control de la Potencia entregada a los Motores **CPM**, con el objetivo de que la salida de CM tenga una influencia inalterada en la salida del motor, y CPM al ser realimentado afecte de forma poco notoria al primer lazo de control CM. Para este objetivo se debe tener en cuenta, que el motor seleccionado es de tipo BLDC, diferente en construcción a los motores DC que normalmente se trabajan ya que los BLDC no poseen escobillas, eliminando los problemas de rozamiento que disminuyen el rendimiento, no desprende tanto calor, no es tan ruidoso y no necesita una sustitución periódica, tiene una mayor relación velocidad-par motor que es fundamental para la aplicación que se quiere realizar, ya que es probable que el VAE llegue a tener un peso considerable. Lo anterior ayuda a pensar en el sensor que se usa para cerrar el lazo de CPM y el actuador que se utiliza para controlarlo, sin embargo, el motor encajado en el chasis tiene bastantes limitaciones por espacio y por diseño, que reducen las opciones de sensores (también se desea modificar el chasis poco), y ya que en este proyecto no se incluirá el diseño del hardware controlador, se usara el Módulo de Control de lazo Abierto MCA que se encarga de generar las señales necesarias a una determinada entrada analógica. El encoder de cuadratura es uno de los sensores más usados para lazos de control de velocidad angular, sin embargo, el motor no cuenta con un eje extendido que permita adicionar dicho sensor, por lo cual no será posible usar un sensor de este tipo. Adicionalmente las llantas están unidas a un mecanismo diferencial que además de permitir que una llanta gire a una velocidad diferente de la otra, esta no tiene espacio para agregar sensores de este tipo, por lo cual es descartado. La opción más viable teniendo en cuenta las posibilidades, es realizando mediciones de la corriente en las 3 fases del motor BLDC, como salida del motor e indicador del parámetro conocido como la fuerza contraelectromotriz, así con una búsqueda de cruce por cero, se puede determinar la posición del eje del motor y realizar la respectiva retroalimentación del lazo CPM. Así mismo, debido a la complejidad a la hora de medir la eficacia del seguimiento de trayectorias, esta medición se hace desde dentro del VAE con la información obtenida del sensor usado para retro-alimentar el lazo CM que consistirá en una IMU inercial, las cuales son sensores que basados en mediciones del campo magnético determinan la aceleración y ángulo de rotación dependiendo del número de ejes que estas especifiquen.

2 Objetivo Principal

Investigar acerca de plataformas para el desarrollo de tecnologías de conducción autónoma, así como implementar los algoritmos de control más esenciales como base para el objetivo de construir el primer vehículo de conducción autónoma a escala, producido en el semillero de robótica de la Escuela Colombiana de Ingeniería, enfocado a generar un mínimo producto viable con el objetivo de participar en la competencia como la Carolo Cup.

3 Objetivos Secundarios

- Conocer, aprender y seleccionar software o programas para simulación de vehículos autónomos
- Aprender y aplicar correctamente métodos para la identificación de motores BLDC y validar los modelos obtenidos.
- Investigar los métodos usados para el modelamiento de movimiento de vehículos no holonomicos e implementarlos en las simulaciones.
- Aprender a implementar leyes de control que puedan ser evaluadas con distintos criterios de acuerdo con el tipo de implementación a realizar.
- Reforzar la habilidad para modelar la interacción entre 2 sistemas e implementarlo.
- Implementar un sistema de control difuso y el respectivo hardware asociado a él, incluyendo sensores.
- Implementar sistemas controladores en dispositivos programables.

4 Metodología

4.1 Revisión Teórica

Para el desarrollo del proyecto se realiza una revisión teórica de la problemática de la generación de trayectorias, los métodos de control de motores BLDC, las técnicas de modelado del sistema móvil, las técnicas de identificación de motores BLDC y de igual manera se investiga acerca de las plataformas de simulación existentes y software potencial que se

usa para la simulación de sistemas de vehículos autónomos e identificaciones de motores BLDC.

4.2 Simulación

Una vez estudiados los métodos para la identificación de motores BLDC, se procede a realizarla sobre el motor que se posee en el laboratorio. En la plataforma de simulación, se incorpora el modelo resultante de la identificación de los motores BLDC y el modelo de movimiento del VAE que permite generar coordenadas matemáticas en un mapa que representa el entorno donde se ubica el VAE. El modelo de movimiento del VAE en función de la velocidad angular y lineal de entrada, necesita contar con el lazo cerrado de la potencia del motor CPM para que la dinámica no se vuelva muy compleja, se pueda reducir el error, no varíe por condiciones externas a las propias medidas físicas del VAE y también depende del punto de control deseado. Las simulaciones tienen unas entradas de referencia que se suponen vendrán del sistema de percepción de posición y de toma de decisiones que no se implementaran en este proyecto, las trayectorias usadas en este proyecto se generaran a partir de funciones matemáticas.

4.3 Implementación

La fase de implementación se divide en 2, según el lazo que se este implementando, ya sea CPM (control de potencia del motor) o CM (control de movimiento). Para la implementación se dispone de una Raspberry PI, que es un dispositivo programable que puede adaptarse al consumo de energía y medidas del vehículo. Así mismo no se integra todas las partes del proyecto, solo se trabaja las partes relevantes al control de motor BLDC y control de movimiento, incluyendo sensores de variables internas que sirven para evaluar el desempeño del controlador.

4.3.1 Control de motor CPM

Primero se implementa el hardware requerido para realizar la retroalimentación de la potencia del motor, una vez hecho esto, se procede a implementar el controlador seleccionado de la parte de simulación y se cierra el lazo, posterior a esto, se realizan las respectivas pruebas de funcionamiento para evaluar la eficacia del controlador.

4.3.2 Control de movimiento CM

Es necesario implementar sensores que permitan realizar una retro alimentación de la posición y la velocidad del vehículo, así como su orientación, sensores como IMUs y otros hacen parte de esta implementación. Una vez instalados estos sensores, se procede a realizar la implementación del control de trayectorias.

4.4 Fase de Pruebas

Una vez implementado los controladores CPM y CM, se evalúan en el VAE diferentes tipos de trayectorias, que se dividen en trayectorias simples y trayectorias complejas, en búsqueda de hallar una fiabilidad en el seguimiento de estas, así como de realizar ajustes en el controlador que de forma general mejoren esta funcionalidad. Estas mediciones, se evalúan con el sensor usado para la retroalimentación del lazo CM, suponiendo entonces condiciones donde hay mínimo deslizamiento y ninguna obstrucción en las llantas.

5 Revisión Teórica

5.1 Investigación Plataformas de Simulación

La investigación de software para la simulación de vehículos autónomos, incluye frameworks para software que se usa comúnmente para la simulación de sistemas de control como lo es Matlab, así mismo la investigación busca la viabilidad de crear un software desde la base, en C++ para windows, java o python, sin embargo, al realizar esta investigación se encuentra con el software CARLA un software open-source que se estima por la investigación, es el mayor software de simulación libre que hay actualmente y se propone sea el eje central del proyecto cuando este avance paulatinamente.

5.1.1 Automated Driving Toolbox and Navigation Toolbox

Es un toolbox para Matlab que provee un entorno de simulación que permite observar el comportamiento en tiempo real de un vehículo con ciertas características.

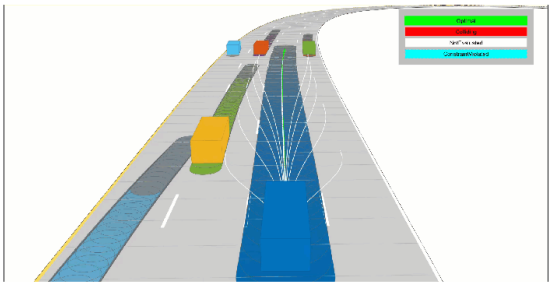

Figure 1: Imagen del simulador de trayectorias con obstáculos dinámicos en la vía. Tomada de: https://la.mathworks.com /products/navigation.html

Sin embargo, se entiende que si en este toolbox se desea incorporar una simulación del modelo de control de trayectorias, el cual es el que define que tan eficaz es el auto para realizar las trayectorias que este se propone (lineas blancas tenues que salen de la parte frontal del auto en la imagen anterior); Es probable que se tenga que editar parte de su código para introducir este trabajo que aumentaría la precisión de la simulación de este toolbox a una más realista. Documentación disponible en [4]

5.1.2 CARLA Simulator

En el trabajo de investigación acerca de plataformas diseñadas específicamente para esta aplicación se encuentran varias, que de una u otra forma son propietarias y necesitan de una licencia para ser usadas, sin embargo, en ninguna de las antes mencionadas se encuentran tantas referencias como con CARLA, que es una plataforma de código abierto (no hay cargo por su uso) y que posee una potencia de simulación asombrosa, esta plataforma está desarrollada en Python, pero se encuentran numerosas referencias que apuntan a que esta plataforma se quiere pasar a Unity, que es un motor de juegos en donde se espera que esta pueda incluir mejores características y un mayor rendimiento, sin embargo recientemente se conoce que los desarrolladores se orientaron a el motor de videojuego Unreal Engine propietario de Epic Games.

Figure 2: Imagen del simulador de trayectorias CARLA con obstáculos dinámicos en la vía. Tomada de: https://www.youtube.com /channel/UC1llP9ekCwt8nEJzMJBQekg

Sin embargo, este simulador denota el mismo problema del toolbox de Matlab puesto que para incluir la simulación de control de trayectorias es probable que se necesite editar el código del simulador. El software está disponible en [5]

5.1.3 Unity

Por último, Unity invita de forma general [6] a usar su motor de videojuegos para generar entornos que sirven para el entrenamiento de IAs de conducción autónoma, en donde se pueden integrar mejores algoritmos y análisis más personalizados.

Figure 3: Un juego de carreras hecho en unity con físicas simuladas Tomada de: https://www.youtube.com /watch?v=ehDRTdRGd1w&t=23832s

5.2 Investigación de métodos de control y trayectorias

La investigación se realiza con respecto a los métodos para controlar el seguimiento de trayectorias **CM** y controlar la velocidad del motor BLDC **CPM** abarca métodos de fácil implementación y sintonización para este tipo de aplicaciones, con el objetivo de verificar si los resultados que por lo general ofrecen se adaptan a la aplicación requerida.

5.2.1 Control de motor BLDC

Debido a la evidencia encontrada en [2], se entiende que un controlador lineal es suficiente para lograr un control aceptable para un motor DC sin escobillas **BLDC**, por lo cual el estudio se centra en un grupo muy especifico de controladores.

- Control P
- Control PD
- Control PI
- Control PID
- Control Atraso, adelanto y adelanto-atraso

El control de atraso, adelanto y adelanto-atraso, se descarta debido a la complejidad de la implementación, dejando los controladores restantes. Basado en la formación en control automático y contrastando con otras implementaciones [2], se puede decir que para este lazo de control **CPM** lo que más se necesita es que la respuesta sea rápida y precisa en estado estacionario, es decir, la respuesta en estado transitorio no es de tanto interés. Por lo cual, un controlador tipo PI es la primera opción a tener en cuenta, puesto que si la componente derivativa es muy alta la dinámica se puede ver seriamente retrasada, sin embargo, la opción de un controlador PID queda abierta, puesto que si el sobre pico en la respuesta transitoria es muy alta (el único escenario que interesa de la respuesta transitoria) no sera bueno para la vida útil del motor, como se puede ver en la siguiente figura.

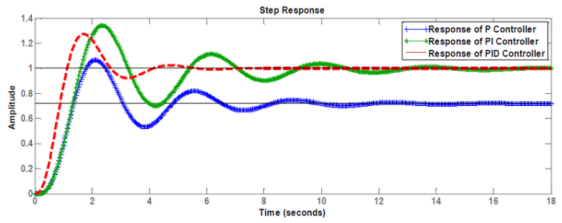

Figure 4: Relación de las constantes del controlador PID con su respuesta Tomado de: https://www.researchgate.net /figure /System-Response-of-P-PI-PID-Controller-tuned-with-Process-Reaction-Curve-Method _fig1_327816155

5.2.2 Control de trayectorias

Basando la investigación en [1] y con los conocimientos en la formación de control no lineal, se puede llegar a un análisis derivado del tipo de ecuaciones resultantes del modelamiento del movimiento del Vehículo Autónomo a Escala **VAE** como un Wheeled Mobile Robot **WMR** no holonomico sin deslizamiento. Estas ecuaciones se especifican en la sección 5.3.2 de este documento. como resultado de la investigación, concluye que existen 2 modelos que se aproximan de forma estrecha al movimiento del VAE, uno el cual está basado en WMR no holonomico descrito en coordenadas polares y otro que integra la interacción de dos del modelo descrito anteriormente en uno solo que es conocido como Car Like-WMR **CLWMR** [1]. El análisis de ambos modelos arroja que poseen dinámicas no lineales presentes en la naturaleza de sus ecuaciones, en los cuales un controlador difuso lineal no proporcionara los resultados deseados, como se evidencia en [3], en donde, la linealización al no hacerse alrededor del punto de operación este control de trayectorias presenta errores de distorsión y acumulación en el seguimiento para ángulos o trayectorias tan siquiera un poco complejas (ángulos de movimiento mayores a 15°), el problema de la acumulación del error se puede evidenciar en la siguiente figura, en donde la trayectoria del WMR no holonomico (en rojo) cada vez se va distanciando de la trayectoria deseada (en azul).

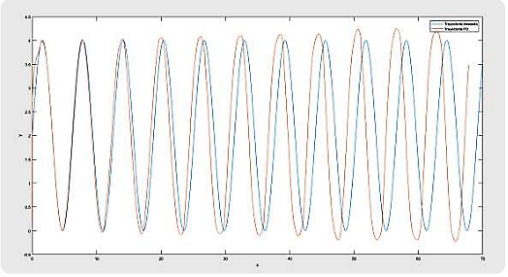

Figure 5: Trayectoria resultante de un controlador lineal (PD) en un robot no holonomico. Tomado de: [3]

En los resultados presentados en [3] se puede evidenciar que un controlador difuso no lineal, ofrece una mayor precisión como se muestra en la siguiente figura, y tomando como referencia la velocidad con la que crece el error. sin embargo, debido a la dificultad que muestra la revisión bibliográfica [9] la implementación de este controlador escapa al horizonte de este trabajo.

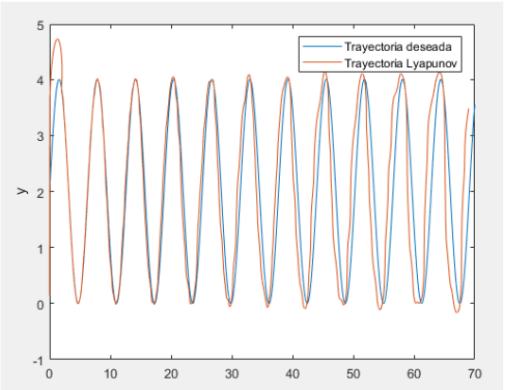

Figure 6: Resultados de trayectorias en un controlador no lineal (Lyapunov). Tomado de: [3]

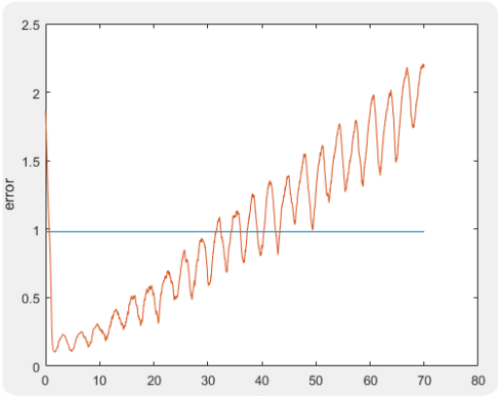

Figure 7: Error acumulado en el tiempo en un controlador lineal (PD). Tomado de: [3]

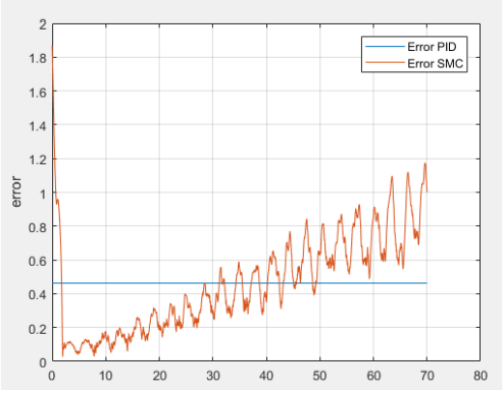

Figure 8: Error acumulado en el tiempo en un controlador no lineal (Lyapunov). Tomado de: [3]

Por lo cual, se halla que una de las mejores soluciones es implementar un controlador de Lyapunov en cualquiera de los 2 modelos de movimiento del VAE encontrados.

5.3 Investigación de la identificación del BLDC y modelamiento de movimiento

5.3.1 Identificación del motor BLDC

Para el sistema dinámico del motor BLDC se tiene en cuenta que es un actuador bastante complejo, por lo cual se descartan métodos de identificación que dan como resultado un sistema de primer orden. El método seleccionado es el método de identificación clásica: modelo de orden superior, el cual es capaz de dar como resultado un modelo de segundo orden o superior y no se espera que la respuesta sea subamortiguada. Dicho método expresa el modelo resultante como se muestra en la ecuación 1

$$G(s) = \frac{A}{(\tau s + 1)^2} e^{-Ls} \tag{1}$$

De la ecuación 1 y de la gráfica 9 se puede obtener las ecuaciones que relacionan los tiempos característicos del 28.4% y el 63.2% de la amplitud máxima de la respuesta al escalón con parámetros de la ecuación 1 como lo es L y τ. El cálculo aproximado de los parámetros característicos del sistema real se hace como indica el autor Smith quien aconseja calcular con a = 1.5 (Marlin, 2015, pags. 179-181), en las siguientes ecuaciones:

$$\tau = a * (t_{0.632} - t_{0.284}) \tag{2}$$

$$L = t_{0.632} - \tau \tag{3}$$

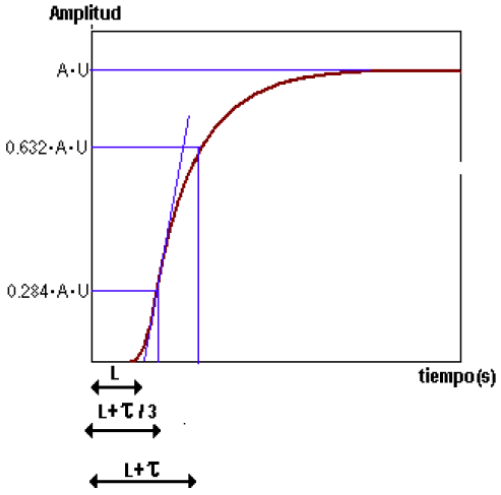

Figure 9: Ilustración con los puntos importantes de la identificación. Tomado de: Anexo J métodos de identificación

5.3.2 Modelamiento

La investigación del modelamiento de movimiento del VAE comienza con una indagación acerca de robots más simples y se encuentra acerca de los robots no holonomicos [3], los cuales tienen la característica de que su desplazamiento en el sistema de coordenadas está restringido, de tal manera que su movimiento es no omnidireccional, resulta que esta condición se extiende incluso a los robots tipo automóvil y a los automóviles. El modelamiento busca a través de expresiones físicas y matemáticas representar en términos de una entrada y una salida, el comportamiento de un sistema visto desde un punto particular de interés (punto de control), se diferencia de la identificación en cuanto no busca medir el comportamiento real de un sistema para intentar conocer como debería ser su expresión, si no que busca mediante consideraciones crear la expresión lo más real posible. Generalmente se usa cuando la identificación es muy compleja o no es necesaria por la simplicidad del sistema, en este caso medir la respuesta de movimiento que tendrá el VAE en el plano de desplazamiento es cuanto menos difícil y tedioso por lo cual se procede con un modelamiento.

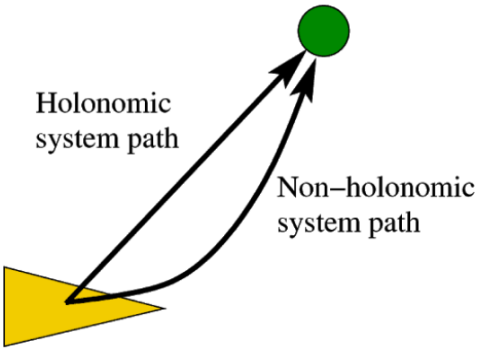

Figure 10: Definición gráfica de una trayectoria holonomica. Tomado de: [3]

Basándose en este hecho, en la investigación se encuentran 2 modelos que son lo suficientemente precisos, ambos conocidos popularmente en el modelado de este tipo de robots, encontrados en [1].

Modelo no holonomico diferencial. El primer modelo encontrado en realidad es una aproximación más precisa del modelo más básico que existe de los vehículos no holonomicos diferenciales, el cual posiciona el punto de control a una distancia A del punto donde se encuentra la mitad de la distancia D que es la distancia de separación entre los dos motores [1].

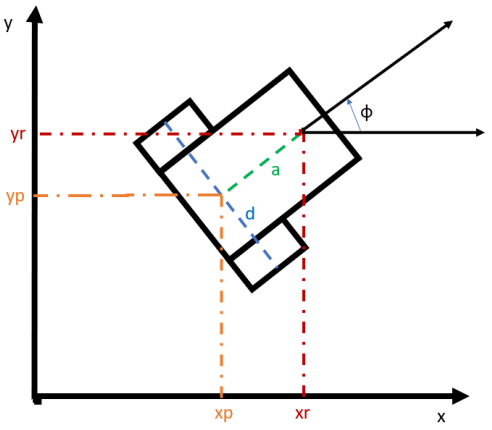

Figure 11: Modelo cinemático de un robot no holonomico diferencial que tiene su punto de control a una distancia A. Creación propia

Este modelo tiene como entrada la velocidad lineal y la velocidad angular del vehículo, U y W respectivamente; y tiene como salida las coordenadas rectangulares del punto de control xr, yr y ϕ. Finalmente las ecuaciones que definen este modelo son [1]:

$$\dot{x_r} = ucos\phi - awsen\phi \quad (4)$$

$$\dot{y_r} = usen\phi + awcos\phi \quad (5)$$

$$w = \dot{\phi} \quad (6)$$

Car-Like WMR (wheeled Mobile Robot) - Robot móvil con llantas tipo automóvil Este modelo es mucho más complejo que el anterior, puesto que está diseñado para tener en cuenta el movimiento de las llantas delanteras, lo que requiere que sean incorporados 2 modelos de robot diferencial en 1 y que su dinámica también sea modelada[1].

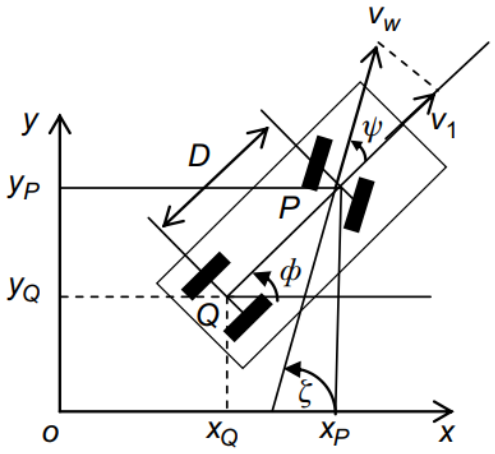

Figure 12: Modelo cinemático de un robot no holonomico diferencial tipo automóvil. Tomado de [1]

Como se menciono anteriormente este modelo está construido gracias a la unión de dos modelos, los cuales al eliminar la entrada U para poder tener las ecuaciones en una sola igualdad da como resultado lo siguiente:

$$-\dot{x_Q}sen\phi + \dot{y_Q}cos\phi = 0 \quad (7)$$

$$-\dot{x_p}sen(\phi + \psi) + \dot{y_p}cos(\phi + \psi) = 0 \quad (8)$$

Donde U es la velocidad lineal, ϕ es la velocidad angular y ψ es el ángulo de giro. Este modelo contempla la interacción de dos modelos que en un principio estaban separados, por lo cual, es necesario que se escoja un modelo como principal (punto de control/referencia) y el otro sea descrito en términos del primero. Es así como se considera el punto que está en la parte delantera del vehículo y entre las llantas como el punto de mayor importancia dentro de un vehículo tipo automóvil [1], y de igual forma el punto trasero puede expresarse a partir de este. Las ecuaciones que definen el modelo con este punto de control entonces son [1]:

$$\dot{x_Q} = v_1 cos\phi \quad (9)$$

$$\dot{y_Q} = v_1 sen\phi \quad (10)$$

$$\dot{\phi} = \frac{1}{D} v_w sen\psi \quad (11)$$

$$\dot{\phi} = \frac{1}{D} v_1 tg\psi \quad (12)$$

$$\dot{\psi} = v_2 \quad (13)$$

6 Implementación de sensores y actuador de motor

Para la implementación del actuador del motor BLDC se tiene en cuenta el controlador de lazo abierto que existe actualmente en el robot. Para la entrada de este dispositivo se tiene que entender que este espera una señal estándar de servo, la cual como se ve en la siguiente imagen, es una señal de onda cuadrada con un periodo de 20ms, en la cual el tiempo activo de la señal varia de entre 1ms y 2ms. Así mismo se debe mencionar que este controlador tiene una forma de calibración con respecto a la onda de entrada que recibe los valores mínimos, máximos y medios. Teniendo en cuenta que para el sistema de toma de decisiones (no incluido en este proyecto) se necesita tener disponibilidad de algoritmos de visión artificial y deep learning. Se selecciona como hardware de control una Raspberry Pi 3 (en realidad a medida que el proyecto avanza paulatinamente se encontra que es más recomendada la Raspberry Pi 4) y de esta manera se busca con dicho hardware generar esta señal. Se encuentran dos maneras de generarla: método polling y por interrupciones. El método de polling se puede describir como un método de consulta constante [7] en donde todo el tiempo se está preguntando si la ventana de tiempo para determinado estado de la señal ya se ha cumplido para en dado caso cambiar dicho estado. El método por interrupciones consiste en delegar el trabajo de contar tiempo a un periférico creado para este propósito conocido como timer, para que una vez cumplido el tiempo solicitado este interrumpa al sistema operativo para que este ejecute un trozo de código encargado de cambiar de estado la señal de salida. Es así que se crea el primer hilo/tarea, la cual consiste en una función que se ejecuta de manera repetitiva en un loop de forma ”paralela” con otras tareas, haciendo uso del threading propio de los sistemas

operativos. Esta tarea se encarga de activar una serie de interrupciones para generar la señal, usando métodos de una clase que se crea para controlar servos (código disponible en los anexos) con el objetivo de controlar el motor BLDC y el servo de dirección. Así mismo, queda abierta la implementación para el método polling, debido a que la implementación por interrupciones tiene la limitante de tener una resolución de tan solo 200us para el intervalo de la señal en alto de 1ms a 2ms, y si esto se desea mejorar es necesario intentar otra implementación que está considerada en la sección 6 de este documento delegando la generación de esta señal a un microcontrolador ARM adicional, sin embargo, como se menciona anteriormente en este proyecto se uso el método por interrupciones desde la Raspberry Pi 3.

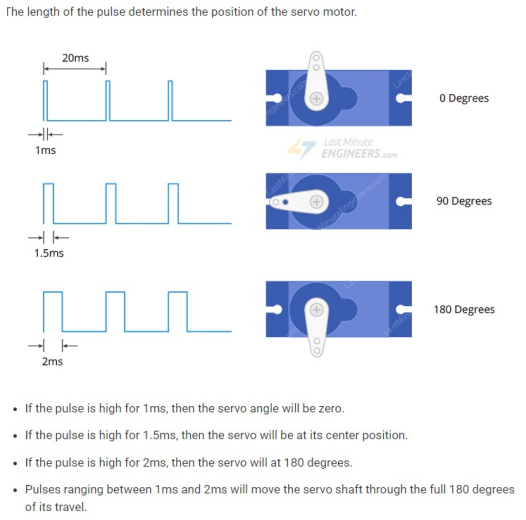

Figure 13: Señal de un servo estándar. Tomada de: https://lastminuteengineers.com /servo-motor-arduino-tutorial/

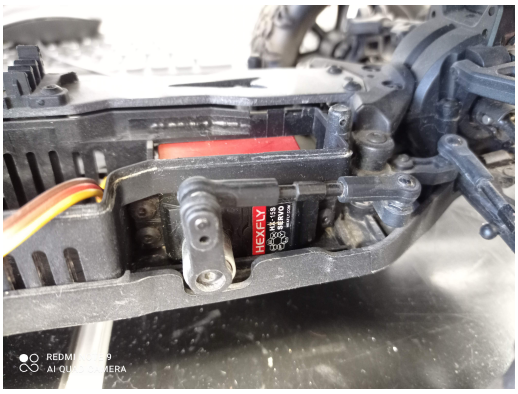

Figure 14: Servo encargado de la dirección. Creación propia

Los sensores para realizar la retroalimentación de velocidad del motor BLDC representan otro reto, debido a que para la implementación no es posible integrar sensores tipo encoder es necesario recurrir al uso de otro tipo de sensores, en este caso la opción más viable son sensores de corriente, que dan esta magnitud como respuesta a la posición del eje del motor.

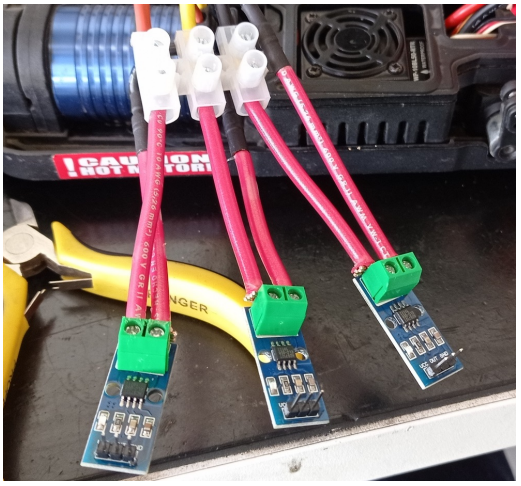

Figure 15: Implementación de sensores de corriente en las fases del motor BLDC. Creación propia

Para esta implementación se usan sensores tipo efecto hall ACS712 que permiten medir cada 5us la dinámica de corriente que estén teniendo las fases del motor BLDC. Sin embargo, la señal de salida de estos dispositivos necesita ser acondicionada para que pueda ser interpretada de forma rápida por la CPU central, por lo tanto es necesario realizar un acondicionamiento de señal con un circuito analógico que garantice un tiempo muy bajo de propagación. Se diseña la placa de acondicionamiento de señal y del circuito mencionado en la sección 6 y se realiza su fabricación posteriormente. Para fines de este proyecto dirigido, se realiza el circuito montado sobre una PCB de ensamble Universal que se acopla a la tarjeta de desarrollo Raspberry Pi 3.

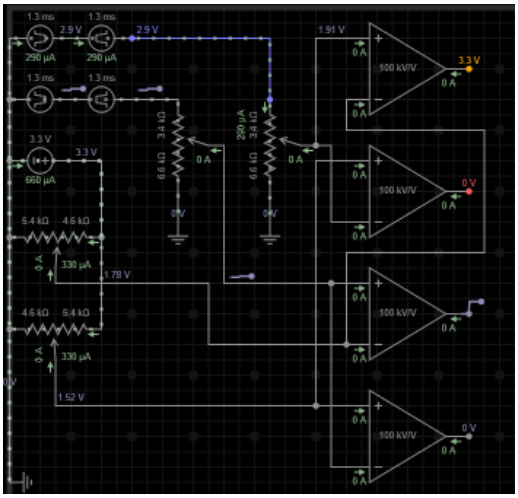

Figure 16: Esquemático de circuito acondicionador. Creación propia

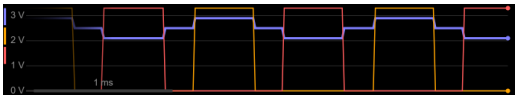

Figure 17: Señal de un sensor acondicionada. Creación propia

Durante la implementación, se encuentra un problema, puesto que la Raspberry Pi 3 B+ se usa con el sistema operativo de propósito general Raspberry pi OS de 64 bits basado en la distribución Debian de Linux [8] debido a su fácil instalación. El problema consiste en que al ser este sistema operativo de propósito general, el código de python que se ejecuta por encima del sistema operativo no puede garantizar tiempos exactos para la generación de señales ni para la lectura de muestras, lo cual presiona para que la implementación del actuador tenga que usar el periférico de PWM, descartando la opción de implementación por polling que antes se había considerado. Por otra parte, la idea general para poder controlar el motor BLDC es que los sensores de corriente permitan inferir la velocidad angular del motor, a través de un análisis de transformada rápida de Fourier **FFT** obteniendo la frecuencia de mayor energía de la señal de corriente. Debido a que las 3 fases del motor BLDC contienen la misma información, solo es necesaria una sola de ellas, por lo que en la implementación final 2 fases son retiradas y una sola 1 es dispuesta para la identificación y la retroalimentación en general.

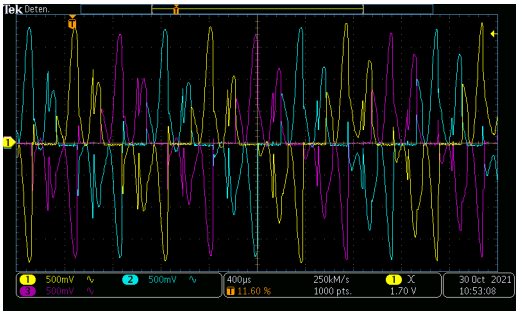

Figure 18: Señales de corriente de las 3 fases del motor. Creación propia

Figure 19: Señal de corriente de una fase y la salida del acondicionamiento de señal. Creación propia

El problema mencionado anteriormente, hace que la lectura sea errónea ya que el muestreo no tiene un periodo constante lo que genera que el delta de frecuencia no sea el mismo, lo cual es bastante problemático para hallar la frecuencia con mayor energía. Una solución es probar con un vector de tiempo variable, pero no es una solución aceptable puesto que el tiempo que le toma a la CPU hallar la frecuencia de mayor energía es demasiado, aproximadamente 1 segundo, por lo que esta solución es descartada. Esto conduce a la implementación de una solución distinta, que consiste en usar un microcontrolador adicional. El microcontrolador implementado es un STM32f103c8t6 de la compañía ST Microelectronics, que es un ARM Cortex M3 mono-núcleo en una tarjeta de desarrollo rápido conocida con el alias de ”Blackpill”.

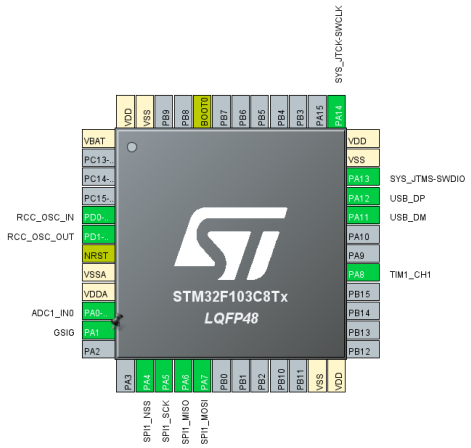

Figure 20: Microcontrolador ARM y su pinout. Creación propia

Este microcontrolador cuenta con un ADC tipo flash que puede leer hasta una frecuencia de 1MHz, que es más que suficiente para la frecuencia de muestreo que se selecciona con base en el análisis del FFT de la señal del sensor de corriente después del circuito de acondicionamiento, la frecuencia de muestreo seleccionada es de 2KHz puesto que más allá de 1KHz no hay información relevante para la aplicación.

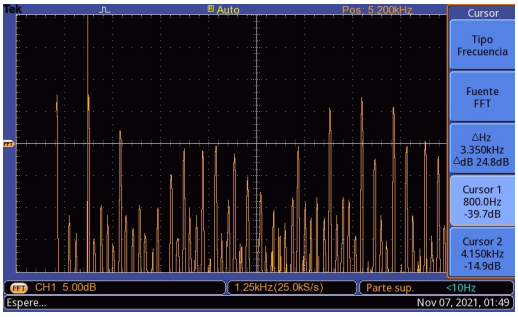

Figure 21: Análisis FFT al 80% de la señal. Creación propia

El código realizado en C para el Advanced Risk Machine **ARM** que integra un oscilador de cristal con el objetivo de generar mayor precisión, usa el timer 3 para generar un trigger out event sobre el ADC1 cada 500us quien realiza la lectura del canal 0 dispuesto en el pin A0, lectura que es trasladada a un bufer con un tamaño de 128 palabras de 16 bits (debido a que el ADC es de 12 bits) por el Direct Memory Access **DMA**, mientras que el SPI habilita una interrupción mediante Nested Vectored Interrupt Controller **NVIC** al núcleo ARM cuando reciba 2 bytes a manera de verificación de errores (si existe ruido en el canal) y de comando (que se está pidiendo al ARM realizar), este último se implementa con el objetivo de que más adelante la Raspberry se pueda apoyar aun más en el ARM (por ejemplo el actuador de los servos), en la interrupción el núcleo ARM verifica si la información recibida es valida y si el bufer está completo, de ser así, el núcleo ARM envía los bytes del bufer correspondiendo a una organización little endian (el byte menos significativo se envía primero) puesto que la conexión SPI es de 8 bits y los datos tienen un tamaño de 16 bits, así mismo, si se necesita que el tamaño de los datos no sean 128 si no aun más, el código es implementado para que de ser necesario pueda segmentar en paquetes de 512 bytes el bufer debido a que el SPI del ARM máximo puede almacenar esta cantidad de bytes. Por ultimo el núcleo ARM solamente transmite el bufer si el DMA ha terminado de llenarlo, de lo contrario el ARM no responde a la solicitud, además el tamaño del bufer debe estar dado por una potencia de 2.

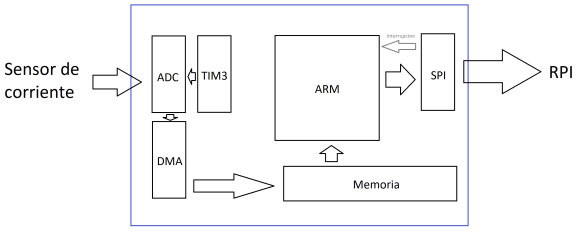

Figure 22: Diagrama de funcionamiento ARM. Creación propia

Esta implementación soluciona el problema satisfactoriamente puesto que la Raspberry solo debe leer cada cierto tiempo (dado por la cantidad de datos a almacenar en el bufer) el bufer que contiene el ARM y puede proseguir mediante el algoritmo de FFT para hallar la frecuencia con la mayor energía. Como se muestra en la siguiente gráfica los resultados del análisis en la frecuencia son bastante limpios mostrando de forma clara la frecuencia fundamental (de mayor energía).

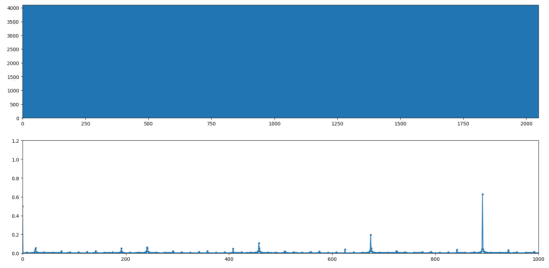

Figure 23: Resultado de implementación de lectura con ARM. Creación propia

7 Implementación de sensores de movimiento

Para esta implementación se tiene en cuenta el hecho de que el VAE no puede comunicarse con dispositivos externos, por lo cual, se plantea un dispositivo con un marco de referencia interno, descartando así el uso de un GPS debido a su complejidad y porque no necesariamente puede ofrecer la exactitud requerida, una unidad de movimiento inercial de 6 ejes es el dispositivo escogido para implementar.

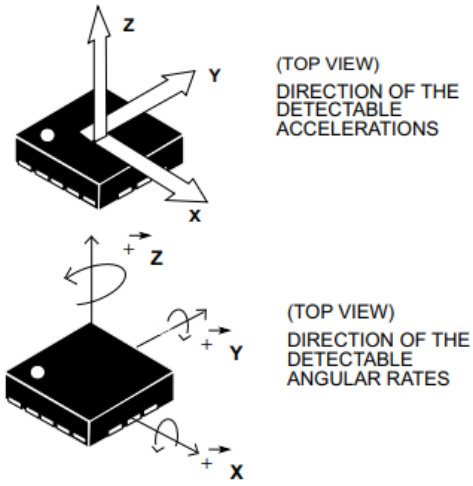

Figure 24: Unidad de movimiento inercial LSM6DS33 escogida y sus 6 ejes. Tomado de: https://www.st.com /en/mems-and-sensors/lsm6ds3tr-c.html

Por conveniencia este dispositivo se incluye dentro de la PCB que se construye para acondicionar los sensores de corriente, de tal manera que esté adherida al VAE de forma sólida. En el trabajo realizado sobre este sensor al igual que sobre el actuador, se incluye la elaboración de una clase en python la cual sirve para a través de diferentes métodos controlar y acceder a la información obtenida por la IMU.

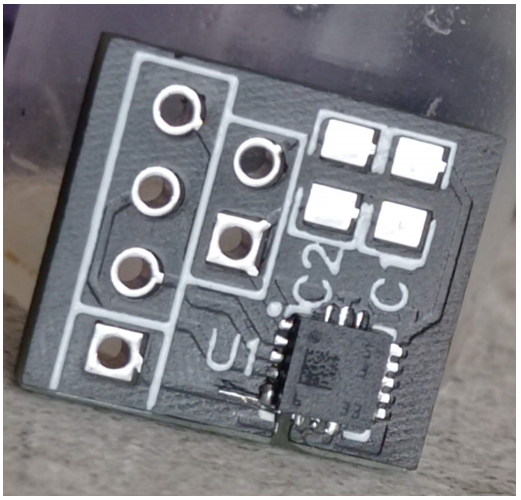

Figure 25: LSM6DS33 en placa de prototipado rápido. Creación propia

8 Simulación

8.1 Identificación de motor BLDC

Para realizar la identificación del motor BLDC se procede a usar la implementación del actuador tipo servo con PWM y el sensor de corriente con el ARM, lo que se hace es establecer un valor de entrada que es de -1000 hasta 1000, siendo 0 el punto de no acción sobre el motor, 1000 el punto de máxima marcha y -1000 dependerá de la configuración establecida en el controlador de lazo abierto que puede ser: freno máximo o punto de reversa máximo, usando threading se elaboraron 2 hilos, uno que está encargado de realizar todo el procesamiento de la lectura de los datos y otro encargado de incrementar la señal de salida en 100, esto con el objetivo de realizar la identificación sobre un escalón que este previamente sobre otro como es recomendado por la literatura, la señal obtenida se guarda en un archivo .CSV que es analizado en el software Matlab. La señal obtenida es la siguiente:

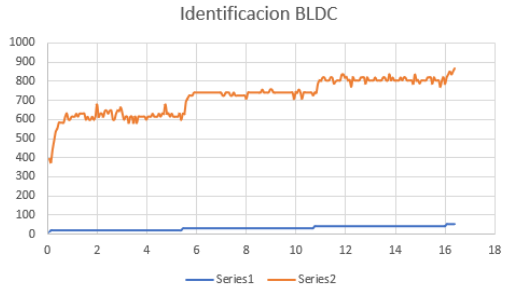

Figure 26: Señal usada para la identificación. Creación propia

Esta señal es tratada, escogiendo la respuesta dinámica al segundo escalón, retirándole el DC e importándola al software Matlab, en donde usando la aplicación integrada de identificación de sistemas, se identifica el sistema, con el siguiente resultado:

$$\frac{Y(z)}{X(z)} = \frac{0.4346}{1 - 1.224z^{-1} + 0.3542z^{-2}} \tag{14}$$

El porcentaje de precisión de la estimación es tan solo del 54.68% lo cual indica que el modelo puede apartarse de comportamientos dinámicos de la vida real, sin embargo, es un buen punto de partida y es algo con lo que se puede trabajar en la simulación, este resultado puede deberse a la insuficiente precisión y el ruido que posee la señal.

8.2 Modelo de movimiento del VAE

Para el modelo de movimiento se usa una función que permite dibujar y animar el movimiento del vehículo, teniendo en cuenta la posición central del VAE. El modelo que se usa es aquel que es más apropiado para la implementación de una estrategia de control por función candidata de Lyapunov [1]. Este modelo tiene como referencia el presentado en la sección **5.3.2 Modelo no holonomico diferencial**, sin embargo el punto de control no está a una distancia a y los estados están dados en coordenadas polares, que son los siguientes [1]:

$$\dot{l} = -v\cos\zeta$$

$$\dot{\zeta} = -\omega + (v/l)\sin\zeta$$

$$\dot{\psi} = (v/l)\sin\zeta$$

Figure 27: Variables de estado del modelo. Tomado de: [1]

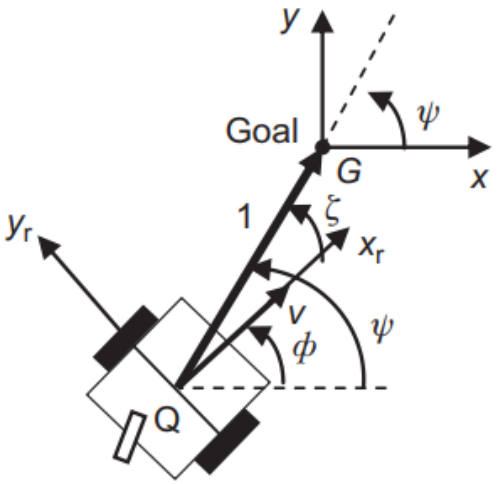

Figure 28: Representación gráfica del modelo usado. Tomado de: [1]

8.3 Simulación de la identificación y el modelo de movimiento del VAE

Para simular la interacción del sistema identificado (BLDC) y el sistema modelado (movimiento del VAE) se tiene en cuenta la particularidad de la implementación de este vehículo, en el cual, la dirección de movimiento solamente está dada por el servo de dirección y la velocidad de movimiento solamente está dado por el motor BLDC, por lo cual es la velocidad lineal la que se envía al sistema identificado del motor BLDC, puesto que el servo de dirección tiene su controlador ya implementado y se asume que funciona idealmente. Para integrar los dos modelos se simula la identificación como un sistema en espacio de estados. La interacción

de los dos sistemas sin el controlador del motor BLDC muestra que el sistema total tiende al descontrol como se muestra en la siguiente figura (trayectoria del robot errática en azul):

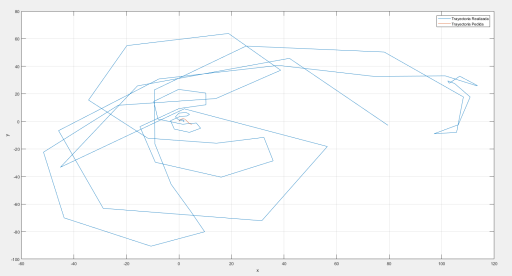

Figure 29: Simulación de los dos sistemas interactuando sin el control de motor BLDC. Creación propia

Sin embargo, para poder hacer un análisis mejor sustentado es apropiado ver los errores de mayor interés, la longitud l y el ángulo de dirección *zeta*.

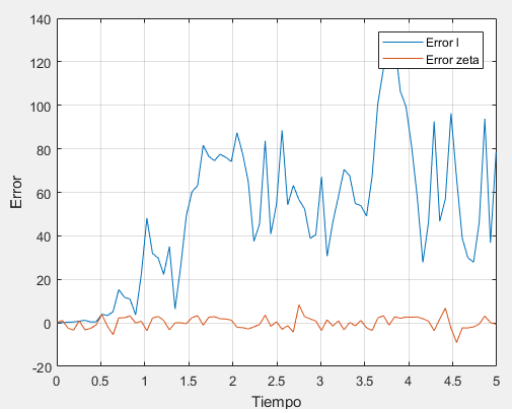

Figure 30: Evolución de los errores. Creación propia

Se puede observar que el error en *zeta* converge alrededor de cero, lo que indica que el modelo de movimiento funciona de forma deseada, mientras que l diverge, esto muestra que el controlador faltante afecta solamente a l en donde el motor BLDC es el único involucrado, esto puede ayudar a entender que los sistemas están interactuando de la forma esperada.

8.4 Seleccionar estrategias de control

8.4.1 Controlador de motor BLDC

Una vez identificada la planta, se hace uso de la aplicación de tuneo de controladores PID que posee Matlab para hallar el controlador más óptimo en cuanto a seguimiento de referencia se refiere, puesto que para una aplicación de este estilo un controlador lineal debería ser suficiente y estos son excepcionalmente los más sencillos de implementar.

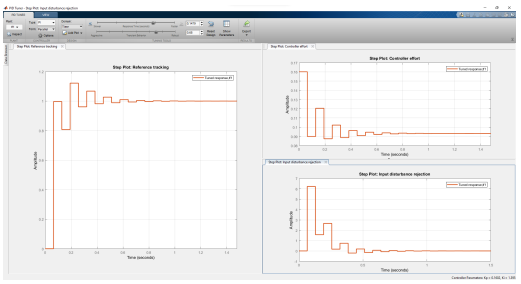

Figure 31: Uso de aplicación integrada de tuneo de PID. Creación propia

La herramienta de tuneo muestra una respuesta incontrolable para cualquier valor de Kd excepto para 0, por lo cual el controlador a escoger es un controlador tipo PI, con constantes Kp = 1.28 y Ki = 3.89 con un periodo característico de 0.064 segundos, que corresponde a 128 muestras de 2KHz.

8.4.2 Controlador de trayectorias

Para el controlador de trayectorias, puesto que el modelo claramente muestra no linealidades se escoge un controlador no lineal, en especifico control de trayectorias por función candidata de Lyapunov. Los cálculos que llevan a encontrar la velocidad lineal y la velocidad angular se encuentran en: [1]*pag. 176.*
Se obtiene el siguiente resultado:

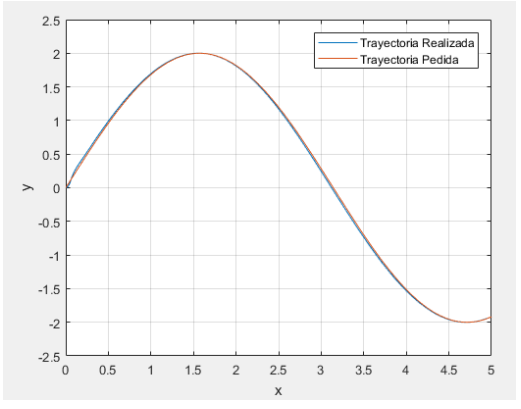

Figure 32: Respuesta del control de Lyapunov. Creación propia

$$v = K_1(\cos\zeta)l, \quad K_1 > 0$$

$$\omega = K_2\zeta + K_1(\cos\zeta)(\sin\zeta)(\zeta + q_2\psi)/\zeta,\ K_2 > 0$$

Figure 33: Leyes de control[1]

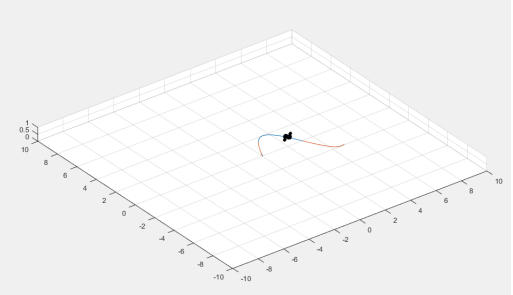

Figure 35: Resultado de trayectoria. Creación propia

8.5 Simulación del lazo de control completo

Simplemente se procede a implementar estas leyes de control e ingresar la velocidad lineal como referencia al lazo de control del motor, que para fines prácticos se hace de la misma forma que en un principio, es decir, se multiplica la identificación por el controlador PI se cierra el lazo y a esa función de transferencia resultante se le halla el espacio de estados para poderse integrar en el sistema completo (incluyendo el control de Lyapunov), dando como resultado el seguimiento de la trayectoria por parte del VAE, en la siguiente imagen se puede observar el resultado:

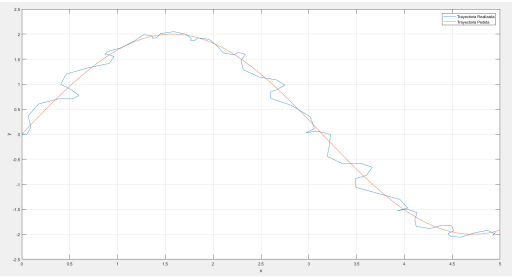

Figure 34: Resultado final del control de trayectorias. Creación propia

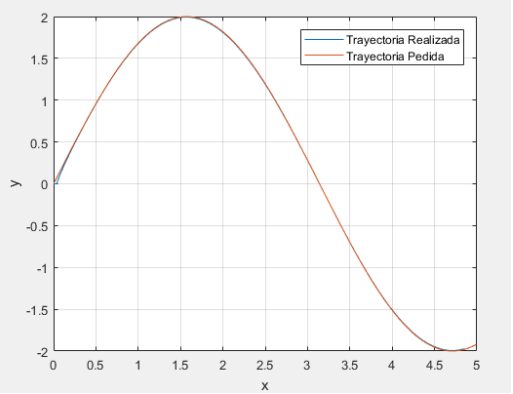

Figure 36: Trayectoria solicitada vs trayectoria realizada. Creación propia

8.6 Evaluar controlador simulado

Se puede observar que al agregar el controlador del motor BLDC a la simulación del sistema de trayectorias esta mejora, el sistema completo en si ya no tiende al descontrol pero tampoco sigue la trayectoria bien, ajustando un poco las constantes de la ley de control de Lyapunov se llega a una respuesta aceptable con los resultados que se muestran en las siguientes figuras:

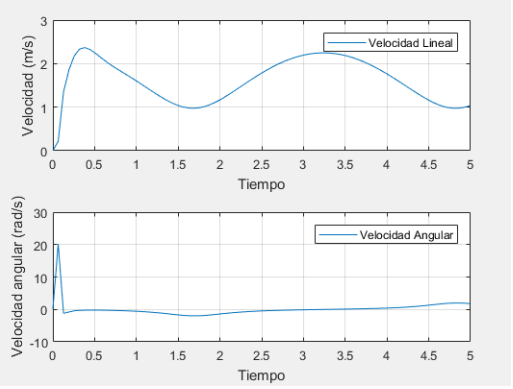

Figure 37: Señales de control. Creación propia

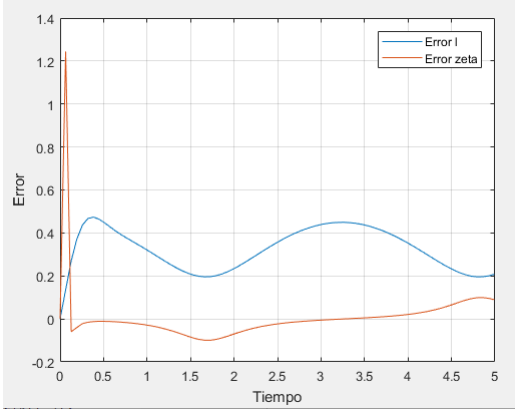

Figure 38: Errores. Creación propia

Figure 39: Salida del motor BLDC siguiendo una referencia de 600. Creación propia

9 Implementación

9.1 Implementación de controlador de velocidad

Para la implementación del control de velocidad se crea un programa en Python a partir de la identificación que continúa el proceso para que una vez obtenida la frecuencia de mayor energía del FFT este entre al controlador, el cual calcula la variación de la señal de control (du), puesto que este controlador posee un término integral el cual de acuerdo a un algoritmo incremental no se desea calcular ya que puede resultar en un uso de memoria muy alto y mayor tiempo de uso de la CPU. Finalmente, la implementación con las constantes Kp y Ki calculadas en la simulación no da los resultados esperados debido al problema de los sistemas operativos de propósito general y su precisión, por lo cual se debe buscar el termino Ki (puesto que este depende del periodo característico que no es constante) un poco más bajo de lo que resulta en la simulación.

9.2 Fase de pruebas de motor

En las primeras observaciones se puede evidenciar que el controlador tiene problemas con el ruido de la señal y con la constante integral, por lo tanto se decide disminuir la constante integral, adicionalmente pareciera que todo intento por limpiar la señal resulta en descontrol del sistema. Los resultados obtenidos del sistema de control siguiendo una referencia de frecuencia de rotación angular de 600Hz, se muestran en la siguiente figura:

Esta prueba se realiza sobre la plataforma armada en vacío en donde las llantas no realizan ningún esfuerzo. Así mismo, el control de trayectorias no se implementa debido a la complejidad de la elaboración del sensor de posición (su programación y uso) puesto que este aporta una componente importante de ruido.

10 Trabajo Futuro

El trabajo restante para la continuación del proyecto, considera la evaluación del controlador CPM con el robot en funcionamiento y armado, revisando la eficacia del seguimiento de la referencia ante el ruido y el movimiento propio del robot.
También considera la implementación del control de trayectorias y evaluar el seguimiento de las trayectorias generadas revisando que se puedan realizar trayectorias cada vez mas complejas (ángulos mas cerrados y con mayor dificultad de giros); el cual tendrá un área de trabajo importante en la implementación del sensor de posición y su retroalimentación. También incluirá la implementación del sistema de percepción y de toma de decisiones que generara la entrada al sistema de movimiento. El cual tendrá un trabajo importante en la visión y reconocimiento del entorno.

11 Conclusiones

- Las unidades de medida inercial IMUs requieren de un extenso acondicionamiento para que puedan ser usadas de forma efectiva en aplicaciones móviles, en especial donde se requiere conocer de forma precisa la ubicación en el espacio.

- Los controladores basados en funciones candidatas de Lyapunov pueden tener una matemática y análisis extensos pero son bastantes simples de sintonizar.
- Los sistemas operativos de propósito general no son aptos para afrontar muchos de los desafíos que presentan los sistemas de control, es más recomendado que estos realicen análisis más complejos por encima de las aplicaciones de control y dejar estas a sistemas operativos de tiempo real.
- La transformada rápida de Fourier es una herramienta poderosa para el análisis de señales, puesto que incluso en sistemas ruidosos esta herramienta se desempeña mucho mejor que los análisis por flancos.
- La comunicación SPI es la forma más rápida y fácil de implementar una comunicación entre procesadores y microprocesadores, y es muy útil para solucionar problemas como el mencionado en el trabajo, de los sistemas operativos y su precisión o en general relacionados con capacidad de procesamiento o arquitectura. Sin embargo, se prevé que en el futuro se adopte una comunicación CAN la cual tenga un mejor manejo de tiempos y manejo de errores en la comunicación.
- El controlador difuso no se implementó por que la revisión bibliográfica ilustró las dificultades del controlador difuso lineal.
- La señal usada para la identificación del motor BLDC usando sensores de corriente debe tener prestaciones de ruido bajo puesto que este puede causar una precisión de identificación bajo (por debajo del 70%), para este fin se comprobó que usando un mayor numero de muestras mejora la respuesta de la transformada rápida de Fourier.
- La selección del software de simulación se debe hacer teniendo en cuenta la arquitectura de software del proyecto puesto que algunos obligan a que solamente se use un tipo muy especifico de arquitectura: Machine Learning, Deep Learning, análisis de imágenes u otros. Y la salida no necesariamente está expresada como una trayectoria matemática.
- El modelamiento de sistemas no holonomicos conducen siempre a un sistema de ecuaciones no lineales que hace su estudio mas elaborado y extenso, al cambiar el tipo de vehículo a uno omnidireccional (de forma que las llantas permitan movimientos perpendiculares a la orientación del vehículo) resulta en un modelo que se puede solucionar con las herramientas de álgebra lineal convencionales y permitiría implementar un controlador lineal.

References

[1] S.G. Tzafestas, *Introduction to Mobile Robot Control*, Primera Edición, 2014, Atenas, Grecia.

[2] C.F. Rengifo, N. Castro y D.A. Bravo, Publicado en: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-74882017000100041, 27/10/2021

[3] E.D. Soto, *Diseño y simulación de controladores no lineales para robots no holonomicos*, 2020, Bogotá, Colombia

[4] Documentación en: https://la.mathworks.com/products/automated-driving.html

[5] Documentación en: https://carla.org/

[6] *Tomado de: https://unity.com/solutions/automotive-transportation/autonomous-vehicle-training*

[7] Publicado en: https://es.wikipedia.org/wiki/Polling

[8] Publicado en: https://www.raspberrypi.com/software/operating-systems/

[9] Publicado en: https://automaticaddison.com/extended-kalman-filter-ekf-with-python-code-example/