Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Mecánica, Mecánica-Eléctrica y Mecatrónica

"CONTROL PD DIFUSO DE TRAYECTORIA PARA MODELO DE VEHÍCULO MEDIANTE LA DETECCIÓN DE LÍNEAS DE CARRIL UTILIZANDO LA TRANSFORMADA DE HOUGH EN MATLAB"

Tesis presentada por el Bachiller:

Mayhua Álvarez, André Mijaíl

Para optar el Título Profesional de
Ingeniero Mecatrónico

Asesor:

Dr. Quispe Ccachuco, Marcelo Jaime

Arequipa – Perú

2022

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA MECATRONICA

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 25 de Abril del 2022

Dictamen: 004850-C-EPIMMEM-2022

Visto el borrador del expediente 004850, presentado por:

2016200901 - MAYHUA ALVAREZ ANDRE MIJAIL

Titulado:

CONTROL PD DIFUSO DE TRAYECTORIA PARA MODELO DE VEHÍCULO MEDIANTE LA DETECCIÓN DE LÍNEAS DE CARRIL UTILIZANDO LA TRANSFORMADA DE HOUGH EN MATLAB

Nuestro dictamen es:

APROBADO

1530 - FERNANDEZ BARRIGA CAMILO GRIMALDO
DICTAMINADOR

1936 - MESTAS RAMOS SERGIO ORLANDO
DICTAMINADOR

2776 - COLLADO OPORTO CHRISTIAM GUILLERMO
DICTAMINADOR

DEDICATORIA

A mis padres, que siempre estuvieron apoyándome en todos mis proyectos y me dieron la oportunidad de poder estudiar esta carrera.

A mi hermana, por tenerme paciencia cuando estaba ocupado y no podía ayudarla.

A mi enamorada, que siempre estuvo conmigo en todos mis proyectos.

A mis amigos de 5to año, con los cuales pasé buenos momentos a pesar de la distancia y la cuarentena en clases.

AGRADECIMIENTOS

A mis padres, por darme la oportunidad de estudiar y haberme formado durante todo este tiempo.

A mis amigos, quienes me acompañaron en la aventura del aprendizaje y con quienes tengo muy buenos recuerdos.

A mis profesores, que me enseñaron no solo los conocimientos científicos, sino modos y formas de actuar y desempeñarme como profesional.

Al coro de la universidad, en donde pude terminar de formar mi personalidad, conocer nuevos amigos y desarrollar nuevas habilidades, fue mi verdadero soporte para liberar estrés, hacer cosas que jamás haría y conocer a mis 2 amores: La música y mi novia Karen.

RESUMEN

El propósito de este estudio es determinar el porcentaje de error que se obtiene de realizar el control PD difuso de trayectoria utilizando visión artificial en un modelo de vehículo. Así mismo se desea contribuir con una guía para motivar al estudio de la visión artificial en la escuela profesional de Ingeniería Mecatrónica de la Universidad Católica de Santa María. Para ello, se utilizó el método de "Transformada Hough", que en visión artificial permite identificar líneas. Se descubrió que se obtenía un mejor resultado de trabajar por separado los tramos "rectos" y "curvos" en la carretera. Así mismo se probaron diferentes valores para definir las mejores funciones de membresía para el control difuso. Finalmente, después de la adquisición del video (utilizando una aplicación para cámara de celular), procesamiento de las líneas de carril (utilizando transformaciones globales, geométricas y Transformada Hough para identificar las líneas de carril) y definición de los controladores difusos (determinados al experimentar empíricamente con el modelo de vehículo), se obtuvo un error máximo de 10% en tramos rectos y del 25% en tramos con curva. Es decir, considerando que el ancho de pista es 140 mm, el máximo descentramiento que tiene el vehículo en tramos rectos es de 14mm, y en tramos curvos de 35mm.

Palabras Clave: Visión artificial, Transformada Hough, Detección de carril, Control difuso

ABSTRACT

The purpose of this study is to determine the percentage of error that is obtained from performing the trajectory diffuse PD control using artificial vision over a vehicle model. Likewise, it is desired to contribute with a guide to motivate the study of artificial vision in the professional school of Mechatronics Engineering of the Catholic University of Santa María. To do this, it was derived from the "regression" method, which in artificial vision allows lines to be identified. It was found that a better result was obtained by working the “straight” and “curved” sections of the road separately. Likewise, different values were tested to define the best membership functions for fuzzy control. Finally, after video acquisition (using a cell phone camera app), lane line processing (using global, geometric, and linear and quadratic regression transformations to identify lane lines), and definition of fuzzy controllers (determined by experimenting empirically with the vehicle model), an error of 10% was obtained in straight sections and 25% in curved sections. That is, considering that the width of the track is 140 mm, the maximum offset that the vehicle has in straight sections is 14mm, and in curved sections 35mm.

Keywords: Machine Vision, , Lane Detection, Fuzzy Control

ÍNDICE

Pág.

ÍNDICE DE FIGURAS

Página

ÍNDICE DE TABLAS

Página

INTRODUCCIÓN

En este trabajo se desarrolló la aplicación de transformaciones geométricas y globales, junto con “Transformada Hough” para detectar líneas de carril de una autopista, con esta información se hizo un análisis de la geometría de estas líneas de carril y se determinó si el vehículo en cuestión se encontraba en el medio o no de la vía, para que, mediante controladores PD difusos se corrigiera su posición dentro del carril y se controlara el giro de las ruedas del auto, consiguiéndose una navegación autónoma.

Para el correcto desarrollo de este proyecto, se decidió dividir la tesis en cinco capítulos, los cuales con: “Marco metodológico”, “Marco teórico”, “Diseño conceptual”, “Diseño de detalle” y “Pruebas y resultados”; con el fin de dar una estructurar correctamente el trabajo y seguir una secuencia lógica de pasos que permitieron llegar a la culminación del proyecto.

Iniciando con el marco metodológico, se realizó la descripción de los problemas a solucionar con este proyecto, estos problemas son: Los accidentes de tránsito causados por el factor humano y la falta de desarrollo de trabajos relacionados a visión artificial en la escuela; después se escribió la justificación de este proyecto, como una manera viable a dar solución a los anteriores problemas, se establecieron las limitaciones que se tuvieron para la elaboración del proyecto y los objetivos que se deseaban alcanzar, finalmente se citaron los avances tecnológicos conseguidos hasta el momento, referidos al tema a desarrollar.

En un segundo capítulo se plasmaron todos los conceptos, fórmulas y detalles técnicos que se necesitó conocer para trabajar con visión artificial, se describieron además detalles técnicos que se necesitó conocer para poder trabajar adecuadamente con el proyecto, por lo cual en este capítulo se trataron conceptos sobre visión artificial, conceptos relacionados al procesamiento de imágenes conceptos sobre el color para tener las nociones básicas sobre las que trabajamos, así mismo hablar sobre visión artificial en la tecnología automotriz,

robótica móvil, que fueron base de nuestro trabajo y se terminará este capítulo hablando sobre los conceptos del método elegido para identificación de carriles: “Transformada Hough” y lógica difusa.

En el tercer capítulo del diseño conceptual se habló sobre la metodología de acción que se llevó a cabo para realizar el proyecto en base a los antecedentes leídos; en líneas generales, se realizó primero la adquisición de datos, utilizando una cámara de celular y una conexión inalámbrica, estos datos pasaron a través del procesamiento de imágenes y reconocimiento de líneas (Transformada Hough), los datos obtenidos de este procesamiento se utilizaron para determinar la posición del modelo de auto dentro del carril y los cambios en el ángulo de giro de las llantas que se necesitaba realizar, finalmente de la nueva posición que se generó en el modelo de vehículo se realimentó al controlador, para que siguiera corrigiendo el error de centro mediante el giro de las llantas y así se repitiera este proceso hasta conseguir que el modelo de vehículo se posicionara al medio del carril, además en este capítulo se especificaron los componentes y programas a utilizar.

En el cuarto capítulo se muestran y explican los programas realizados, también se muestran las adaptaciones que se realizó al vehículo utilizado para la comprobación del código realizado.

En el quinto capítulo se recogieron las experiencias vividas para poder obtener las metas planteadas, así como también se mostraron los resultados obtenidos.

Finalmente se mostraron las conclusiones y recomendaciones obtenidas de la experimentación; finalmente se anexaron procedimientos adicionales para la adaptación de algunos sistemas para el correcto funcionamiento de estos dentro del código elaborado.

Capítulo I

MARCO METODOLÓGICO

1. DESCRIPCIÓN DEL PROBLEMA

Con la tesis se buscó dar solución a 2 problemas en concreto: La falta de información respecto al porcentaje de error que se obtiene cuando se aplica un control PD difuso a una trayectoria utilizando visión artificial. Así mismo también se observó la falta de desarrollo de trabajos utilizando visión artificial en nuestra escuela, temas que serán desarrollados a continuación:

1.1 FALTA DE INFORMACIÓN RESPECTO AL PORCENTAJE DE ERROR QUE SE OBTIENE CUANDO SE APLICA UN CONTROL PD DIFUSO A UNA TRAYECTORIA UTILIZANDO VISIÓN ARTIFICIAL

Hernández Millán, G., Ríos Gonzales, L. H., & Bueno López, M. (2017) trabaja un método de control PID sintonizado con una red neuronal para el control de un robot móvil con sistema diferencial de movimiento, con este método simula y calcula el modelo cinemático del robot móvil a controlar. De esta forma consigue demostrar la eficiencia del control de posición para el seguimiento de tramos rectos y curvos.

Además, la interfaz que ellos desarrollaron permite hacer un ajuste en las ganancias de acuerdo a la necesidad. Sin embargo, a pesar de los avances mostrados, todo el sistema desarrollado se queda en solo una simulación, y no se presentan pruebas realizadas de un modelo de vehículo real.

Khan, M.A.-M.; et al. (2022) hizo un estudio aprendizaje profundo para identificar carriles, con el objetivo de conseguir un sistema ligero de aplicar, y en el que se utilicen redes neuronales convolucionales. De este modo, se consiguió la identificación de carriles en condiciones adversas, por ejemplo, con carreteras con líneas de carril borrosas, o en condiciones de lluvia, o cuando simplemente no se tienen líneas de carril. Sin embargo, las pruebas fueron elaboradas sobre videos pregrabados y, aunque los resultados son muy buenos, el método no brinda información respecto a la aplicación del método en algún modelo de vehículo real.

De manera similar Tosini & Leiva, (2018) desarrolla un método de detección de líneas de carril utilizando una variante de la transformada Hough, para carreteras que no tienen líneas de carril definidas. Es decir, esta investigación será utilizada para reconocer bordes de carretera y no líneas de carril y será aplicada en carreteras no pavimentadas. La variante de la transformada de Hough es llamada ARHT y es capaz de identifica líneas rectas y curvas dentro del carril. Finalmente se obtuvo una buena identificación de bordes de carretera. Al igual que los demás métodos, la aplicación se hizo sobre videos pregrabados.

Se puede observar entonces, que la mayoría de los trabajos de investigación desarrollan métodos y sistemas que no llegan a ser contrastados o probados en un entorno real. Por medio del metabuscador “Biznar” se observa en la Figura 1. que hay una creciente tendencia referida a proyectos referidos a “modelos de vehículo y detección de líneas”.

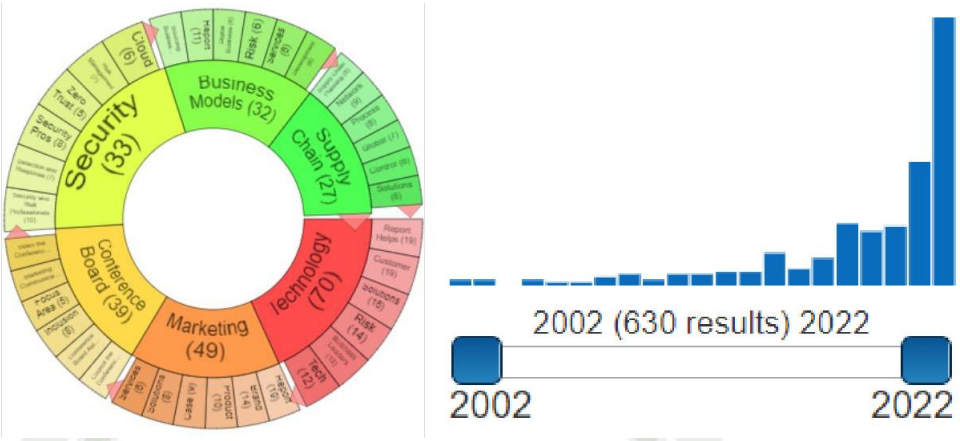


Figura.1 Biznar, situación de las investigaciones utilizando las palabras clave "modelo de vehículo" y "detección de líneas al 17/08/2022

Fuente: Elaboración propia

1.2 ANÁLISIS DE LA FALTA DE DESARROLLO DE TRABAJOS UTILIZANDO VISIÓN ARTIFICIAL EN NUESTRA ESCUELA

Al hacer una búsqueda en el repositorio de tesis de la Universidad Católica de Santa María con palabra clave "visión artificial" encontramos que desde el año 2013 hasta la actualidad solo se han presentado 6 trabajos que contienen esta palabra, como se observa en la Figura 1.2, además se observa que para el año 2019 hubo una creciente en la presentación de trabajos referidos a este tema.

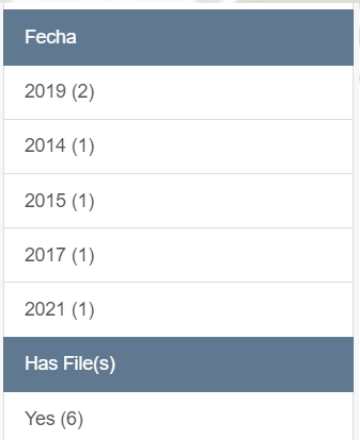


Figura I.2 Repositorio de tesis de la UCSM, número de trabajos con la palabra clave "visión artificial" al 23/08/2021

Fuente: Elaboración propia

Por otro lado, al hacer una búsqueda similar en un motor de búsqueda llamado "Biznar", encontramos que, en el mismo periodo, desde el 2013 al 2021 se han presentado 1044 investigaciones y se observa una fuerte tendencia de aumento en cuanto a investigaciones de este tipo, ya que para la fecha del 23/08/2021 hay 502 trabajos sobre este tema en lo que va del año del 2021, esto se observa en la Figura 1.3.

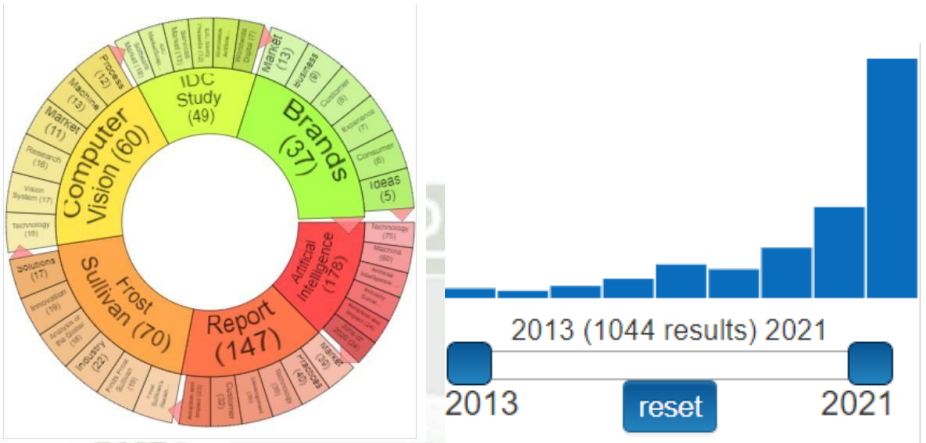


Figura I.3 Biznar, situación de las investigaciones en visión artificial al 23/08/2021

Fuente: Elaboración propia

De igual forma, se hizo una investigación sobre el tema de "visión artificial mantenimiento carril" que es el tema general sobre el que trata la tesis a desarrollar, para ver el estado de las investigaciones al respecto y se observa una fuerte tendencia de desarrollo de proyectos y trabajos referidos al mantenimiento de carril en los últimos 6 años, por lo que se concluye que el tema elegido es de interés científico actual y es necesario encontrar nuevos aportes que ayuden a profundizar o brindar una nueva perspectiva referida al mantenimiento de carril en vehículos, se observa en la Figura 1.4:

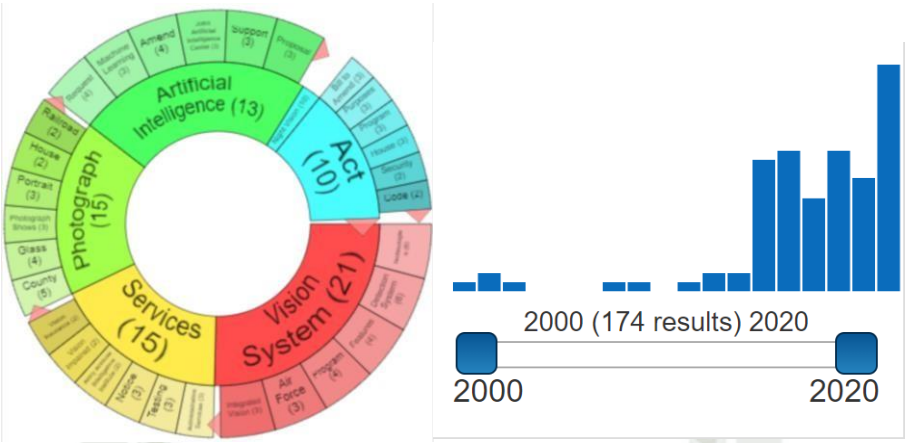


Figura I.4 Biznar, situación de las investigaciones en artificial visión mantenimiento de carril al 22/12/2020

Fuente: Elaboración propia

Teniendo en cuenta los datos anteriormente expuestos, caemos en cuenta que se necesita apoyar al desarrollo de la tecnología e investigaciones al respecto de visión artificial en nuestra escuela para estar a la par de los adelantos tecnológicos actuales.

3. JUSTIFICACIÓN

El proyecto pretende dar solución a los problemas planteados, a través del desarrollo e implementación en un modelo de vehículo con controladores PD difusos y elaboración de una guía académica para los estudiantes de la carrera de Ingeniería Mecatrónica, el sistema a elaborar deberá ser capaz de distinguir las líneas de carril y en base a ellas tomar decisiones de acuerdo con la curva que tenga la pista o si esta sigue de forma recta, de modo que pueda servir como asistente para la conducción

Se implementará este sistema utilizando el software Matlab como una manera de presentar un enfoque diferente utilizando un software que también podría ser utilizado en campo, ya que esta es una herramienta computacional potente, que posee varios toolbox para trabajar de manera más didáctica. En la actualidad existen investigaciones sobre diversas aplicaciones de modelos matemáticos y algoritmos para la detección de carril en autopista, utilizando Open CV y Python, pero no las hay utilizando el software Matlab, residiendo aquí

el aporte del proyecto en la educación, debido a que Matlab es utilizado ampliamente en la educación.

Por lo expuesto en la parte de descripción de problema, se puede observar que el factor humano tiene una importante influencia en el índice de accidentes a nivel Perú, por lo que la implementación de sistemas difusos de conducción autónoma o asistencia a la conducción ayudarían a disminuir este porcentaje de accidentes.

Finalmente, se ha considerado enfocar este estudio en modelos de auto, para poder comprobar de una manera más sencilla y a la vez real el funcionamiento del algoritmo, y adicionalmente como complemento para proyectos posteriores en mecatrónica.

4. LIMITACIONES

- Las pruebas se realizaron sobre modelos de auto y de carretera, es decir, no serán aplicados en prototipos, ya que se buscó con esta investigación la aplicación en proyectos de robot móviles para ingeniería mecatrónica. Así mismo se considerará una carretera plana, sin obstáculos, donde las líneas estén correctamente definidas y no existan otros elementos distractores
- La investigación se planteó para ser implementada en un modelo de vehículo Ackerman, controlado por Arduino y MATLAB, reconstruido de un auto a control remoto.
- Los resultados dependieron de la calidad de la imagen obtenida, por lo que influyó bastante el tipo de cámara utilizada; así mismo influyó la capacidad de procesamiento de la computadora que se utilizó, por lo que estos factores variaron ciertos resultados.
- Los costos de los componentes influyeron en la calidad de estos y en los resultados del trabajo, por lo que se moderó el precio-calidad para buscar buenos resultados.
- Adicionalmente otro factor limitante fue la falta de experiencia en el campo que se investigó, lo que conllevó a algunos retrasos de tareas o dificultades en algunos aspectos.

- Se debió tener en cuenta que este trabajo fue elaborado en tiempos de pandemia, por lo que se dificultó la obtención de algunos componentes retrasando plazos o generando que se tuvieran que adaptar algunas ideas originales
- Se deseó obtener los parámetros mínimos para considerar el control como correcto, es decir máximo sobre impulso menor al 20% del setpoint, error en estado estacionario cero y tiempo de establecimiento pequeño.
- La velocidad de desplazamiento del modelo de vehículo fue pequeña, lo que facilitó al automóvil tener una mejor respuesta, esta velocidad fue constante ya que solo se desea controlar el giro de las ruedas del modelo de vehículo.

5. OBJETIVOS

5.1. Objetivo General

Determinar el porcentaje de error al aplicar un control PD difuso de trayectoria para un modelo de vehículo mediante Transformada Hough para la detección de líneas de carril en MATLAB.

5.2 Objetivos Específicos

- Descubrir un método efectivo para la adquisición y procesamiento de datos de video
- Determinar el método más conveniente para identificación de líneas de carril
- Seleccionar las funciones de membresía correctas para el control PD difuso
- Elaborar una guía de estudio con la información recabada para la Escuela Profesional de Ingeniería Mecatrónica.

6. ANTECEDENTES

En diversos trabajos, tales como los desarrollados por Gómez Zurita, (2015); Ibarra Arenado et al., (2015); Tosini & Leiva, (2018); Ladero García, (2019); Khan, M.A.-M.; Haque, M.F.; et al (2022) y Correa Sandoval & Díaz Zapata, (2019) han trabajado con el algoritmo de Hough para poder desarrollar sus proyectos, teniendo como soporte el software Open CV y Python o C++ como lenguajes de programación, sin embargo, únicamente Torres Guaita, (2017) trabaja este mismo algoritmo de Hough en el software de Matlab; así mismo es importante resaltar que Matlab posee un toolbox especializado en navegación autónoma

de vehículos, mediante el cual se pueden realizar diversas aplicaciones como la que se busca en este proyecto, esto se evidencia en MathWorks, (2019).

Un enfoque diferente es el que emplean Khairdoost, N.; Beauchemin, S. & Bauer, M. (2021); Montoya Baidal & Pachacama Tamayo, (2016); Espada & Araya, (2015) y Mathalagu et al., (2020); quienes realizan una investigación bibliográfica más profunda a fin de poder realizar una comparación entre métodos.

Finalmente, Rodríguez Garavito, (2017) y Khairdoost, N.; Beauchemin, S. & Bauer, M. (2021). hacen la detección de carril utilizando redes neuronales, así que este será el orden en que se presentarán los avances de cada autor respecto a el problema de detección de líneas de carril.

Gómez Zurita, (2015) inicia delimitando su trabajo exclusivamente a autopistas rectas, ya que el método utilizado tendría variantes si se tratara de una autopista con curvas, lo que él busca desarrollar es un sistema robusto para detección de líneas bajo la condición mencionada anteriormente (pista recta), menciona diferentes problemas al momento de adquirir las imágenes de una carretera, tales como pueden ser la poca visibilidad de las mismas, debido a la falta de luz o poca conservación de las marcas, lo que las hace verse borrosas, también menciona que la presencia de objetos sobre la carretera o la presencia de otros autos, otras líneas dificultan la correcta detección de líneas de carril, por lo cual empieza seleccionando un algoritmo potente e selección de bordes, que es el algoritmo Hough en vez de otros como el Canny o Sobel; seguidamente incorporará a su programa otras herramientas matemáticas, tales como el Filtro de Kalman, Tracking de ancho de carretera y finalmente el Tracking del punto de fuga a fin de conseguir que los problemas mencionados anteriormente sean resueltos de la mejor forma posible.

El autor termina indicando que por lo general el detector en autopista, con los filtros de Kalman activos y manteniéndose en el carril tiene más de un 95% de acierto, que consiguió detectar las líneas de carril aun cuando estas eran continuas o discontinuas, como se ve en la imagen inferior y que consiguió implementar un sistema de cambio de carril para evitar confusiones del detector de líneas.

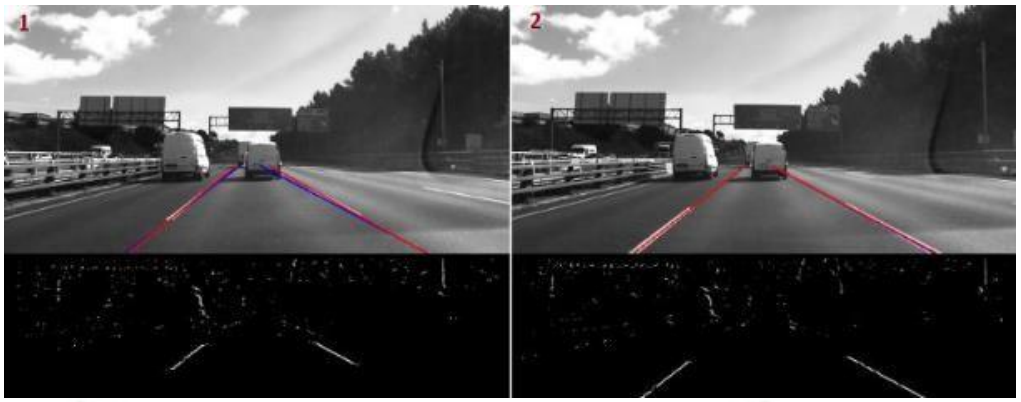


Figura I.5 Detección de líneas discontinuas de carril con el filtro Kalman

Fuente: Gómez Zurita, (2015)

Ibarra Arenado et al., (2015) hace un estudio profundo del método de puntos de fuga para detección de líneas de carril, siendo este otro método útil para conocer los límites de la carretera y completar las líneas de carril en caso presenten los problemas mencionados anteriormente; la ventaja de este artículo reside en que el método a utilizar sirve tanto para carreteras retas, como también para carreteras curvas, como se observa en la siguiente imagen:

Figura I.6 Puntos de fuga en tramo recto (izquierda) o en tramo curvo (derecha)

Fuente: Ibarra Arenado et al., (2015)

A pesar de que la idea parece sencilla a primera vista, es necesario realizar varios procesos para poder llegar a un resultado óptimo, por lo cual primero realiza un filtrado paso bajo a fin de reducirle el ruido a la imagen, seguidamente aplica el método de Canny para detectar los bordes de la imagen y aplica un algoritmo para poder detectar que bordes o líneas no pertenecen a la carretera, para la cual se condicionan ciertos valores angulares de inclinación de las líneas, y los bordes que no cumplen con esos valores angulares de orientación son descartados.

Seguidamente con los bordes que quedan se hace una división por zonas de la imagen, a fin de tener diferentes puntos de fuga debido a la distancia en que se encuentra la línea en profundidad en comparación del vehículo, se hace un "proceso iterativo de votación" para determinar el punto de fuga de la región, se tiene en cuenta para ello las longitudes de los segmentos que participan de dicha votación para asignarle un mayor o menor peso directamente proporcional a su longitud.

Tosini & Leiva, (2018) proponen una investigación para detección de bordes en caminos rurales no pavimentados, es decir que ya no se tienen las líneas típicas de una autopista como en los casos anteriores, sino que esta vez el método será aplicado en caminos de tierra, grava ripio o estabilizados, que no poseen delimitaciones laterales parejas o eje central. Para conseguirlo primero el autor realiza unos filtros de suavizado a fin de disminuir el ruido propio de este tipo de carreteras, por lo que utiliza el histograma de la imagen para mejorar algunos aspectos de contraste y posteriormente a la misma imagen la hace pasar por procesos de suavizado por media, para poder umbralizar la imagen y finamente detectar los bordes de la misma con el método de Roberts.

Posteriormente, para poder determinarlas líneas del camino se utiliza una variante del método de Hough denominada ARHT (Adaptative Random Hough Transform) a fin de incluir ciertos parámetros adicionales que ayuden en la detección de puntos, sobre todo en las partes de curva de la carretera, finalmente se encuentra el centro y radio más aproximados que contenga a la mayoría de los puntos en la imagen, detectando de esta forma el borde de la carretera. En la imagen inferior se observa la aplicación del algoritmo descrito anteriormente.

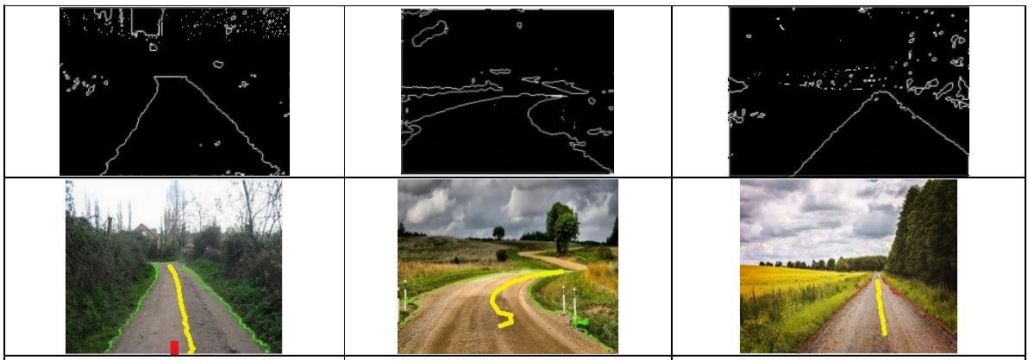


Figura I.7 Detección de bordes de carretera mediante método ARHT

Fuente: Tosini & Leiva, (2018)

Ladero García, (2019) desarrolla varios métodos a fin de conseguir la navegación autónoma de un prototipo pata un concurso “Seat Autonomous Driving Challenge”, para lo cual, desarrolla una estructura de control para el prototipo, esta estructura es la siguiente:

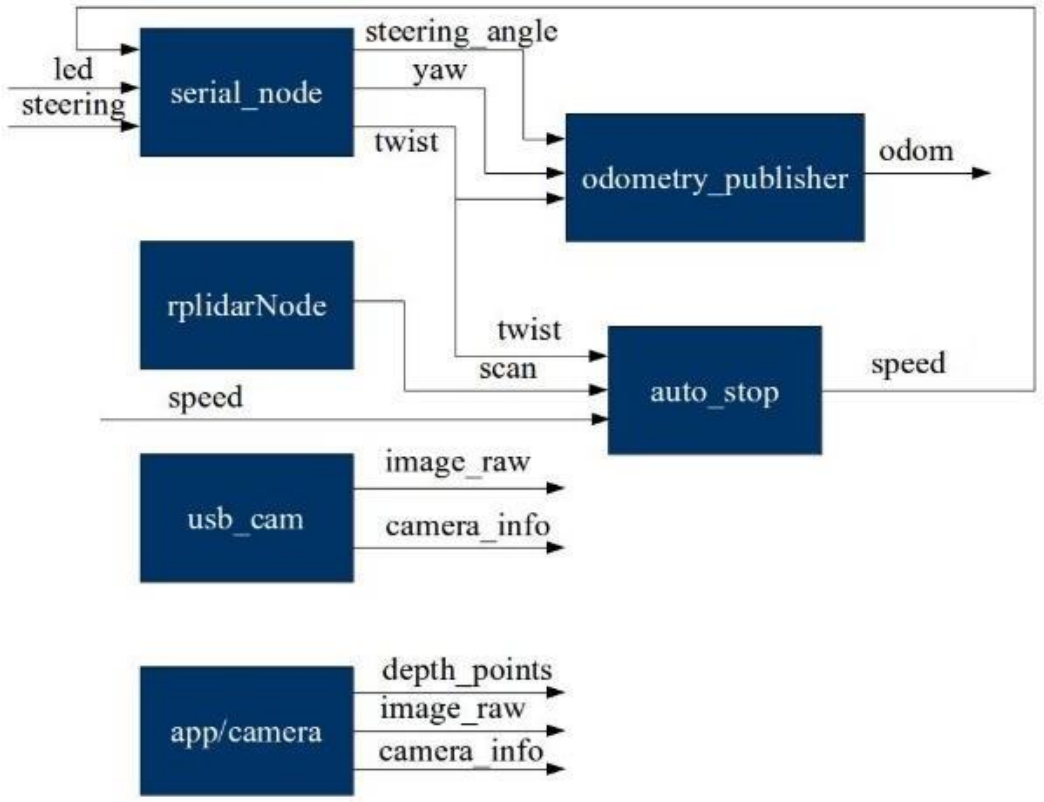


Figura I.8 ARHT Estructura de software para control del prototipo

Fuente: Ladero García, (2019)

Allí se observa el bloque Serial_Node, encargado de recibir la imagen de la pista, mediante una cámara y ayuda del Arduino, con lo cual obtendrán las propiedades del carril, y de otras señalizaciones como colores de semáforo y giros a la derecha, izquierda, etc. Se llevarán estas señales al bloque Odometry_publisher, que será el encargado de determinar la velocidad del vehículo de acuerdo con las condiciones recibidas anteriormente.

Los otros bloques se ocupan del frenado frente a diferentes objetos que pudieran atravesarse delante del auto o del procesamiento de imágenes en sí, pero para efectos del desarrollo del antecedente tomaremos únicamente la parte que compete a la detección de bordes de la carretera.

Al igual que en tesis anteriores, se utiliza el software Open CV para el procesado de la imagen y posterior detección de los objetos deseados, lo primero que realiza es transformar la imagen de color a blanco y negro, esto se hace con el fin de facilitar el proceso computacional derivado de trabajar con imágenes, a esta imagen se le realiza una corrección de distorsión que suele pasar desapercibida, pero que es importante para las futuras trasformaciones.

Después realiza una función de suavizado, probando con filtros de media y gaussiano a fin de eliminar ruidos y seguidamente binariza la imagen para posteriormente aplicar un método de detección de bordes a fin de poder trabajar con las líneas obtenidas para el control del vehículo.

Realiza un algoritmo matemático para determinar si el auto se encuentra a la derecha, al centro o a la izquierda del carril y en base a esto generar trayectorias, tanto si se encuentra en un carril recto o en curva, determinando el ángulo de la curva en el que se desplaza. Algunos de los resultados se muestran a continuación, donde la imagen de la izquierda es el modo en que lo ve la cámara, la imagen central el resultado del detector de borde y el de la derecha el resultado del algoritmo de Hough implementado.

Figura I.9 Comparación de los resultados obtenidos

Fuente: Ladero García, (2019)

Khan, M.A.-M.; Haque, M.F.; et al (2022); realizaron un sistema de Machine Learning aplicado a carreteras en condiciones no óptimas para el reconocimiento de líneas, de forma que el trabajo que obtuvieran fuera fácil de procesar, aplicable a la realidad y brindara a la

vez información confiable respecto a la identificación de carril. En el paper, los autores aseguran haber superado a modelos similares en cuanto a coeficiente de datos IoU y en el tamaño de los modelos. Así mismo la aplicación de este método fue realizada sobre videos pregrabados, en la figura 1.10 se observa que se realizó con éxito el reconocimiento de carril en condiciones como lluvia y noche, las cuales son un reto para la visión artificial, concluyéndose que el modelo desarrollado por los autores sería capaz de funcionar en la aplicación práctica.

Figura I.10 Identificación de carril en condiciones adversas

Fuente: Khan, M.A.-M.; Haque, M.F.; et al (2022)

Montoya Baidal & Pachacama Tamayo, (2016); desarrollan el diseño y la implementación de un prototipo de detección de carril, la particularidad en este trabajo es que, además de utilizar el Open CV utiliza el software QT Creator, encargado de crear interfaces gráficas para celular o Tablet, de modo que los resultados obtenidos se visualicen con facilidad en estas portátiles, además mediante el uso de una Raspeberry Pi 2B y una cámara PI CAM, se pretende armar un sistema completo para visualizar el carril detectado y crear una alarma en caso de que se esté saliendo del mismo, con la siguiente configuración:

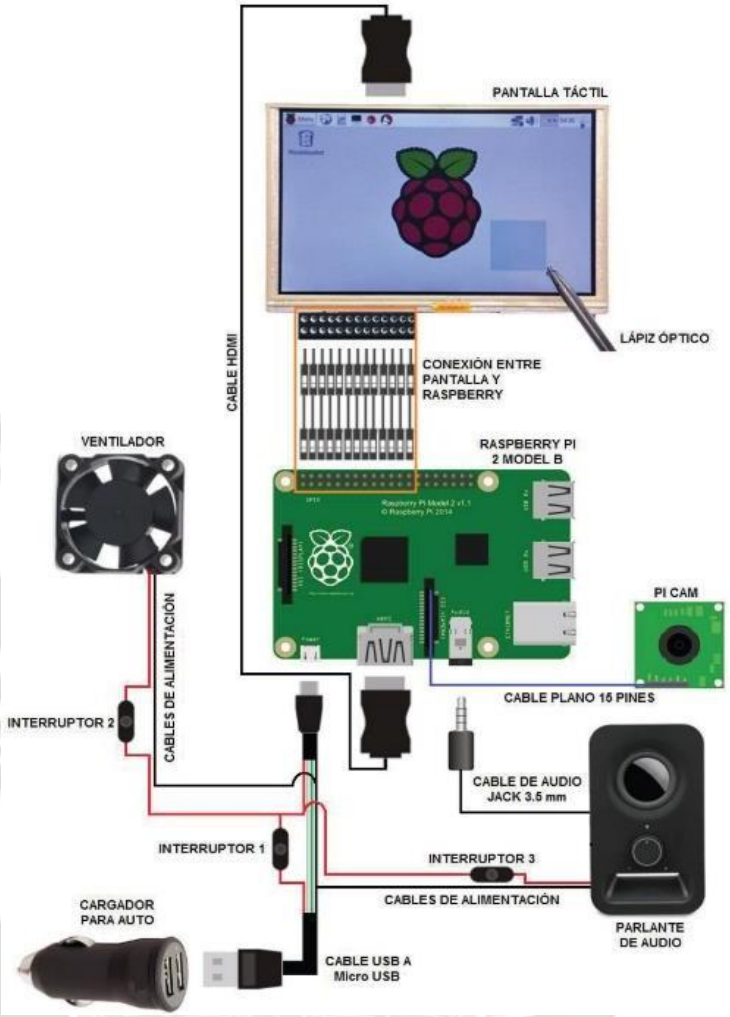


Figura I.11 Hardware para obtención y procesamiento de datos de la cámara

Fuente: Montoya Baidal & Pachacama Tamayo, (2016)

Era importante detallar el armado del hardware para tenerlo en cuenta en el momento de la implementación en el proyecto, de modo que se tuviera una referencia del modo correcto de utilizar estos componentes u otros similares en un futuro.

En cuanto a la obtención de líneas de carril se presentan 2 posibles métodos, el clásico visto en varias tesis o trabajos de investigación de determinación de bordes mediante el método de Canny y posterior detección de bordes de carril mediante la transformada de Hough, y se tiene un segundo método, que utiliza un Filtro adaptativo y la transformada Hough. Entonces se debe de obtener la imagen, llevarla a escala de grises y aplicar los filtros de eliminación de ruido que hemos visto, se umbraliza y se aplica la transformada de Hough, se observa que no se realiza el método de detección de bordes de Canny, por lo que se utiliza

un algoritmo denominado RANSAC para poder eliminar las líneas que no corresponden al carril. Se destaca que se hace una conversión interesante de pixeles a metros que no se había visto en ninguna otra tesis o publicación, determinando que cada pixel es aproximadamente 35,55 cm.

Correa Sandoval & Díaz Zapata, (2019); se desarrolla un sistema de percepción para un vehículo del tipo Ackermann, para el desarrollo del problema prueba con diversos métodos para la obtención de las líneas de carril, como el conocido método de Hough y técnicas de Machine Learning, utilizando 4 clasificadores que son el RBF, Lineal , Polinómico y Sigmoide, finalmente utiliza un método denominado Haar-Like, que considera regiones rectangulares adyacentes , en una ubicación específica de una ventana de búsqueda para filtrar la imagen y realzar las características del carril. Además, se trabaja con un área de interés (ROI) y con una transformación de perspectiva se genera una vista superior o Bird's Eye View (BEV), en ella se aplica el método de ventanas deslizantes para la detección del carril del vehículo. Finalmente, utiliza el filtro de Kalman para darle robustez al sistema frente a obstáculos, como se había visto en tesis anteriores.

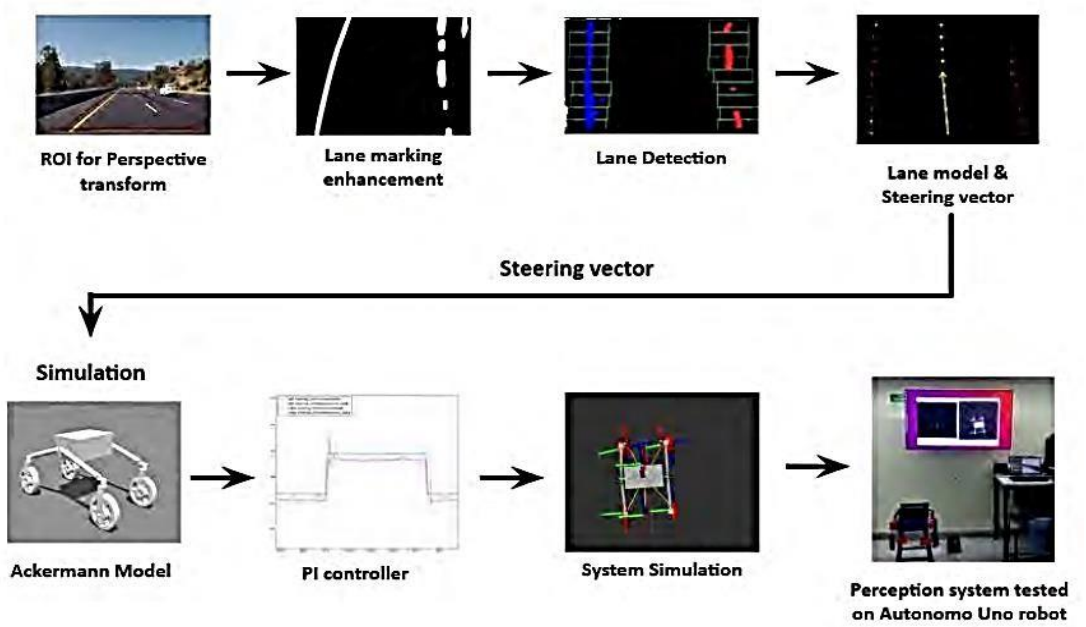


Figura I.12 Esquema de implementación del algoritmo

Fuente: Correa Sandoval & Díaz Zapata, (2019)

El esquema de trabajo básicamente es el que se muestra en la imagen superior, se observa que la primera parte corresponde al procedimiento detallado anteriormente, y la

segunda parte corresponde a un proceso constructivo para implementar el algoritmo anterior en un modelo de automóvil, y controlar su dirección. Como se observa en el gráfico anterior, seguidamente de todos los casos seguidos anteriormente se obtendrá un "vector de dirección", encargado de generar la trayectoria y un ángulo a seguir por el vehículo, ya que este vector se encuentra en el centro del carril detectado. Además, se realiza el control de la posición y la velocidad del vehículo mediante cinemática inversa para un robot móvil (modelo Ackerman) utilizando un controlador PI para pruebas de campo y en simulación.

Figura I.13 Vector de dirección obtenido de la detección de carril

Fuente: Correa Sandoval & Díaz Zapata, (2019)

Finalmente, se probó el modelo armado con el algoritmo propuesto y se evidenció que el sistema cumple con las funciones de detección y seguimiento del carril bajo diferentes condiciones. Además, el control de posición y velocidad también tiene resultados buenos, pero con presencia de ruido leve en la señal de control.

Barroso Rodríguez, (2016) indica un método para que un robot siga el camino que se ha calculado mediante los algoritmos de generación de trayectorias, en nuestro caso, para visión artificial. Para ello se utiliza el algoritmo de seguimiento de trayectoria, en este algoritmo le vamos introduciendo puntos de destino a medida que el robot los va alcanzando, estos van variando, de manera que los puntos objetivo van cambiando a lo largo del tiempo. Este tipo de algoritmo es de seguimiento puro y en el podemos elegir la distancia a la que queremos seguir la trayectoria.

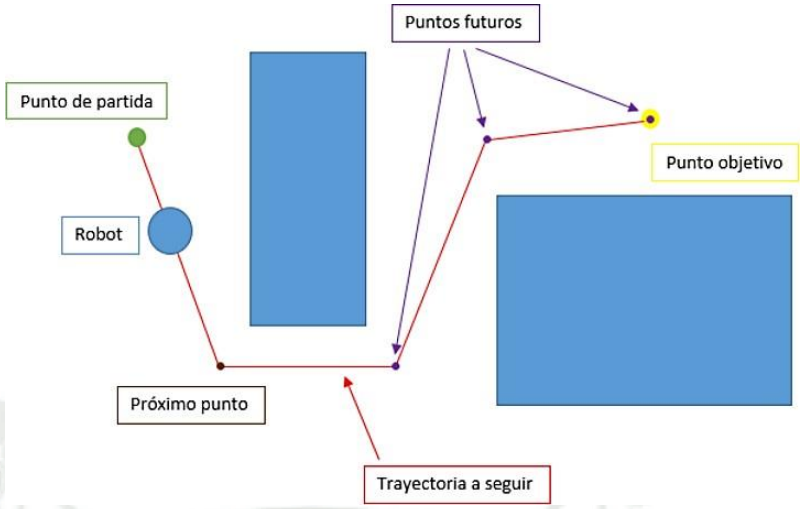


Figura I.14 Algoritmo de seguimiento de trayectoria
Fuente: Barroso Rodríguez, (2016)

En este tipo de algoritmos para obtener la velocidad a la que debe moverse nuestro robot es necesario usar una ley de control proporcional más integral (PI), mientras que para calcular la orientación solo es necesario una ley de control proporcional (P). La implementación de dicho algoritmo en Simulink sería la siguiente:

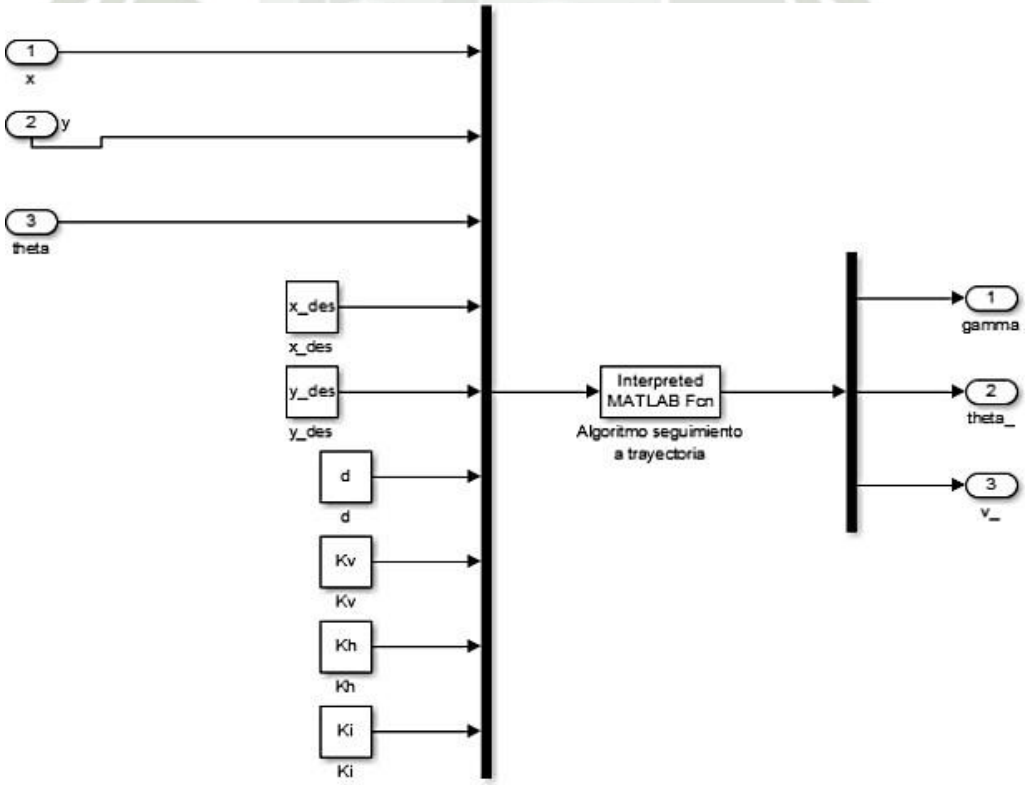


Figura I.15 Algoritmo de búsqueda implementado en Simulink
Fuente: Barroso Rodríguez, (2016)

Podemos ver como tenemos nueve valores de entrada, de ellos seis son constantes y tres son variables. Las cuatro constantes finales (d, Kv,Kh,Ki), variarán como hemos dicho antes según el robot y el algoritmo usado y serán introducidas externamente, mientras que las otras dos, también introducidas externamente, se corresponden con el punto que queremos alcanzar en cada instante y estas irán cambiando a medida que se vayan alcanzando los puntos de la trayectoria. Dicho algoritmo lo hemos logrado comparando el punto en el que nos encontramos, el cual es conocido ya que tenemos sus coordenadas x e y, con el punto objetivo de ese instante. En el momento que coincidan dichos valores se pasa al siguiente punto objetivo. También encontramos en la parte central una función de Matlab en la que se realizan las operaciones del algoritmo de seguimiento de trayectoria mostradas anteriormente, para así conseguir las variables de salida necesarias.

Capítulo II

MARCO TEÓRICO

1. VISIÓN ARTIFICIAL

La primera pregunta que surge al hablar sobre "visión artificial" es ¿Qué es la visión?, al respecto Justo de Frías, (2018) indica que "visión es recuperar de la información de los sentidos (vista) propiedades validas del mundo exterior". Y el primero continúa indicando que: "La visión artificial, o visión computacional, pretende replicar esta capacidad en los ordenadores, de forma que se puedan detectar diferentes objetos en el espacio y su posición a través de la imagen que se recibe, por ejemplo, con una cámara".

La visión artificial tuvo sus orígenes aproximadamente en los años 50, Gómez Hidalgo, R. (2019) indican que "La facilidad con la que los seres humanos podemos conocer todo lo que contiene nuestro entorno hizo creer que un ordenador podría interpretar las imágenes que recibe del exterior de una manera relativamente fácil", hubieron numerosas investigaciones al respecto, sin embargo, contrario a lo que se pesaba por aquel entonces, la visión artificial no fue tan sencilla de implementar como se esperaba, convirtiéndose en un reto para los programadores de aquel entonces, quienes, al no contar con las herramientas necesarias para el procesado de imágenes, se veían limitados con sus avances.

A pesar de todo, como refiere Gómez Hidalgo, R. (2019), entre los años e 1965 y 1970 se pudieron desarrollar algoritmos de gran relevancia computacional, muchos de los cuales se siguen aplicando a la actualidad, aunque para ese entonces estos algoritmos se veían aún limitados, nos referimos a los algoritmos desarrollados por los doctores Lawrence G. Roberts en 1965, William Wickam en 1967, Prewitt y Sobel en 1970.

Curiosamente se detienen las investigaciones hasta la década del 80, este periodo es denominado por (Ferro, 2018) como el "Invierno de la inteligencia Artificial", desde aquel punto a la actualidad ha habido varios avances, los cuales son listados en la Tabla II.1

Tabla II.1 A*plicaciones y problemas actuales relacionados con la visión computarizada*

	APLICACIONES	PROBLEMAS
1	Uso de sensores LiDAR y AMRs para conseguir vehículos autónomos. Se encuentra en esta investigación TESLA.	Automoción
2	Uso de cámaras para identificación de los productos que lleva el cliente, evitando pasar por un cajero. Se encuentran en esta investigación: Standard Cognition, Trigo Visión, Zippin, AiFi	Gran consumo
3	Recojo y reubicación de productos, empaquetado y paletizado de productos, control de calidad, montaje, cámaras inteligentes con sistemas de visión integrados	Industria
4	Reconocimiento e interpretación de objetos en la escena, Reconocimiento de eventos que podrían causar un potencial daño para trabajadores y parar procesos. Automatización de control de procesos 24 horas.	Seguridad
5	Detección de tumores y asistencia de robots médicos de alta precisión, utilizados para operaciones mínimamente invasivas	Medicina
6	Dotar al sistema de la capacidad de "aprender", y utilizar la información para diversos fines.	Deep Learning
7	Dotar al sistema de n comportamiento similar la "sistema visual humano", aplicando filtros a las imágenes para extraer características	Redes neuronales

Fuente: Elaboración propia

Se observan diversas aplicaciones para la visión artificial o "visión computarizada", según lo nombran los citados autores, es evidente que, la tecnología que se poseía hasta el momento iría adaptándose poco a poco a estas nuevas tecnologías, hasta que en la actualidad se pudiera, "realizar el reconocimiento de objetos mediante el análisis de sus características, utilizando procesos ligados a inteligencia artificial, aplicar este conocimiento a sistemas de

navegación de robots móviles, detección de tumores, análisis celular, reconocimiento de caracteres, identificación de objetivos militares, investigación criminal, etc.” (Fernández García, 2017), pudiéndose observar que, para este momento, la tecnología había dado un gran salto en cuanto a aplicaciones en diferentes campos de la tecnología.

En concordancia con el tema que se está estudiando, se tratará exclusivamente la aplicación de la visión artificial en la automoción y los robots móviles.

1.1 Visión artificial y procesamiento de imágenes

De acuerdo con Gonzáles Marcos, A., et al. (2006) “La visión artificial o comprensión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de ese mundo”. En donde las propiedades geométricas de un objeto comprenden su tamaño, forma y el lugar en que se ubica y las propiedades materiales son su color, textura, iluminación y composición. De la misma forma, es necesario que se pueda diferenciar el entorno del objeto en sí, además de poderse reconocer los cambios que sufre este entorno para evitar algunos errores de identificación.

Al igual que en el proceso de visión humana, los procesos básicos que se debe seguir para poder captar el mundo exterior son “captar la imagen” e “interpretar la imagen captada”. En la actualidad la primera fase de la visión ha sido solucionada, los “ojos” de un ordenador son actualmente las cámaras de video. Sin embargo, actualmente el mayor reto de la visión por computador es la de distinguir información relevante de la imagen captada del exterior, para que a partir de esa información se puedan realizar las acciones correspondientes a lo que se desea realizar. De acuerdo con Fernández García, N. (2017) algunas de las dificultades que presenta la visión artificial son: Ambigüedad en la definición, cambios de iluminación, cambios de escala, deformación, oclusión, movimiento, perdida de información, los cuales deben superar los algoritmos utilizados para la identificación de objetos.

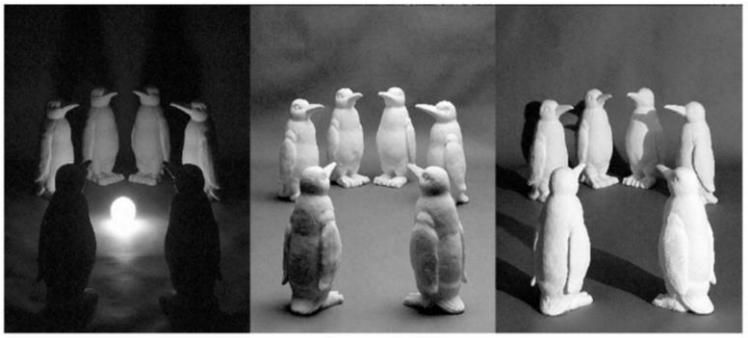


Figura II.1 Dificultad de la iluminación en la visión artificial
Fuente: Fernández García, N. (2017)

Figura II.2 Dificultad del movimiento en a visión artificial
Fuente: Fernández García, N. (2017)

Se colocaron ejemplos de algunas de las dificultades a las que se enfrenta la visión por computador, y que se presentaron en el desarrollo de esta tesis, lo cual será explicado en su debido momento en los capítulos posteriores.

A pesar de todos los avances alcanzados, aún sigue habiendo una enorme brecha entre el proceso de interpretación que hace un cerebro humano respecto a la interpretación que pudiera hacer un ordenador sobre la misma; por lo que, con el fin de facilitar el proceso de reconocimiento de objetos dentro de una imagen se procura colocar fondos contrastantes y planos que resalten el objeto de estudio, así como eliminar la mayor cantidad de ruido que se pueda encontrar en la imagen (ruido generado por sensores). El preprocesamiento de una imagen se lleva a cabo mediante algoritmos que calculan nuevas intensidades luminosas para cada uno de los pixeles de la imagen, sin embargo, esto puedesuponer una gran cantidad de cálculo, el cual se extiende cuanto mayor sea la resolución de

la imagen. En el caso de trabajarse con situaciones en tiempo real, es decir, que se tenga una respuesta instantánea a como se producen las imágenes, Gonzáles Marcos, A., et al. (2006) recomienda que en caso de trabajarse en tiempo real es necesario estudiar todos los métodos posibles y la forma de realizarlos con la menor carga computacional posible.

Gonzáles Marcos, A., et al. (2006) agrega que la visión computacional es una rama de la ciencia que está en fase de crecimiento. Al poseer una gran complejidad, se van solucionando poco a poco algunos problemas que ayudan a facilitar algunos otros procesos de reconocimiento de imágenes. Algunas líneas de estudio generadas comprenden visión en movimiento, visión tridimensional, segmentación de imágenes y agrupamiento de múltiples objetos diferentes en entornos no controlados, etc.

Por esto mismo, Fernández García, N. (2017) identifica 3 niveles de dificultad en aplicación de la visión artificial:

- Nivel bajo: Identificar propiedades directamente de los pixeles de la imagen, tales como gradiente, profundidad, textura, color, etc.
- Nivel intermedio: Obtener bordes, líneas, regiones, con el fin de segmentar la imagen, mediante la agrupación de elementos obtenidos en el nivel anterior
- Nivel alto: Interpretar los entes obtenidos en los niveles anteriores y utilizarlos con el fin de resolver algún problema.

Como se había mencionado anteriormente, existen algunas características propias de los sensores captadores de la imagen que modifican la calidad con que se capta la imagen, estas características son el muestreo y la cuantificación

A) EFECTO DEL MUESTREO

Está relacionado directamente a la resolución espacial de la imagen. En la imagen inferior se hace una comparación de una misma imagen captada con diferentes resoluciones y transformadas al mismo tamaño. Se puede observar que existe información perdida conforme el muestreo es más burdo, así como también se observa que aparece ruido en forma de patrones rectangulares conforme disminuye la resolución de la imagen.

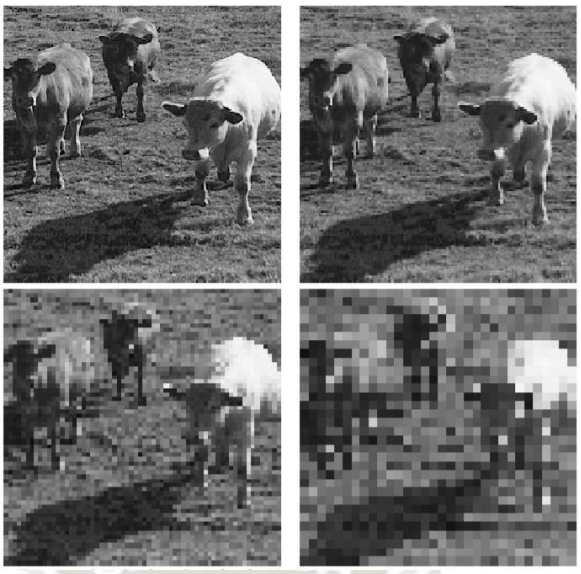


Figura II.3 Diferentes muestreos para la misma imagen
Fuente: Gonzáles Marcos, A., et al. (2006)

B) EFECTO DE LA CUANTIFICACIÓN

Hace referencia a la capacidad de asignar valores de medida para los niveles de intensidad de brillo en cada píxel, esto se encuentra relacionado a la cantidad de bits de información que dispone el procesador para poder almacenar estos valores lumínicos. Por lo general se utilizan 8 bits que equivalen a 256 niveles de gris identificables en una figura. EN la imagen inferior se muestra una comparación entre imágenes representadas con diferente número de bits

Figura II.4 Diferente cuantificación en la misma imagen
Fuente: Gonzáles Marcos, A., et al. (2006)

Un apunte interesante es anotado por Fernández García, N. (2017), quien hace énfasis en comparar la forma en que una persona ve una imagen digitalizada y el modo en como la interpretaría un ordenador, representada en la siguiente imagen

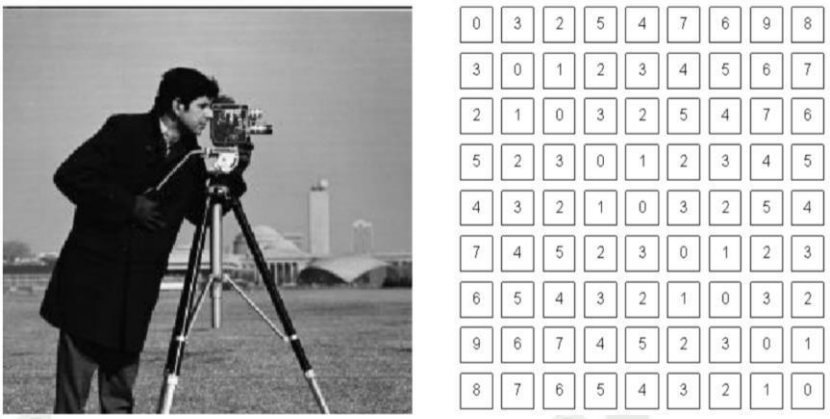


Figura II.5 Comparación de visión humana y visión del computador
Fuente: Fernández García, N. (2017)

De este modo es como se explican las dificultades mencionadas anteriormente, cuando la imagen presenta ruidos, deformaciones, movimiento, etc. que dificultan el "entendimiento" del ordenador de la imagen digitalizada.

No hay un criterio específico para determinar un número óptimo de bits para muestrear una imagen, sin embargo, existen valores umbrales a partir de los cuales no se aprecian diferencias significativas conforme se aumente el número de bits. Por lo que Gonzáles Marcos, A., et al. (2006) concluyen en utilizar la mayor resolución y calidad posible para realizar los trabajos de visión artificial.

1.2 Etapas en el proceso de visión artificial

Diversos autores Fernández García, N. (2017) y Gonzáles Marcos, A., et al. (2006) tienen ideas similares respecto a los procesos o etapas que se llevan a cabo para poder realizar una correcta visión por computador del mundo exterior, los cuales son:

- Digitalización: Es el proceso de capturar una imagen del mundo exterior y almacenarla dentro del ordenador, para que pueda ser manipulada.
- Preprocesamiento: Proceso de corrección de los ruidos y degradaciones de la imagen, se utilizan supresores de ruido o se modifica el contraste de esta.
- Segmentación: Proceso que consiste en descomponer e identificar los objetos que participan en la imagen.

- Representación y descripción: Proceso para determinar las propiedades de los objetos encontrados anteriormente, tamaño, color, texturas, etc.
- Reconocimiento e interpretación: Identificar cada objeto encontrado y darle un significado a lo que se está apreciando.

Como se observa en la figura inferior, cada uno de estos procesos está en constante relación con los datos que posea la computadora respecto al tema.

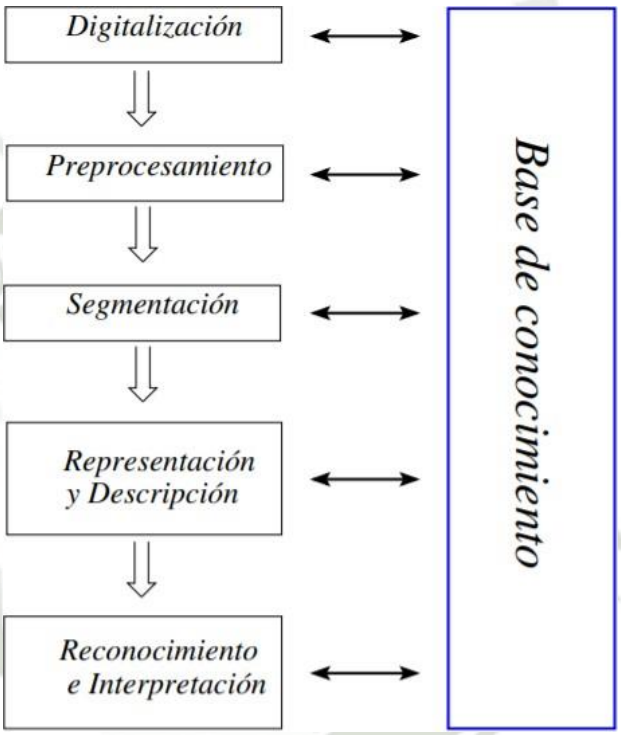


Figura II.6 Etapas en la visión artificial
Fuente: Fernández García, N. (2017)

1.3 Componentes en la visión artificial

En forma general, los componentes básicos para poder realizar trabajos de visión por computador son los siguientes

- Dispositivo de captura: Será el encargado de producir una señal eléctrica acorde al nivel de energía que capte del exterior. Este dispositivo tiene una sensibilidad específica a las bandas electromagnéticas que se desea medir.
- Conversor A/D: Es el encargado de transformar la señal analógica enviada del dispositivo de captura en una señal digital, que pueda ser utilizada por el computador.

- Memoria de video: Se encarga del almacenamiento de los datos obtenidos, para que puedan ser utilizados en el futuro.
- Procesador: Es el encargado de aplicar las transformaciones y algoritmos necesarios para poder trabajar con la imagen obtenida.

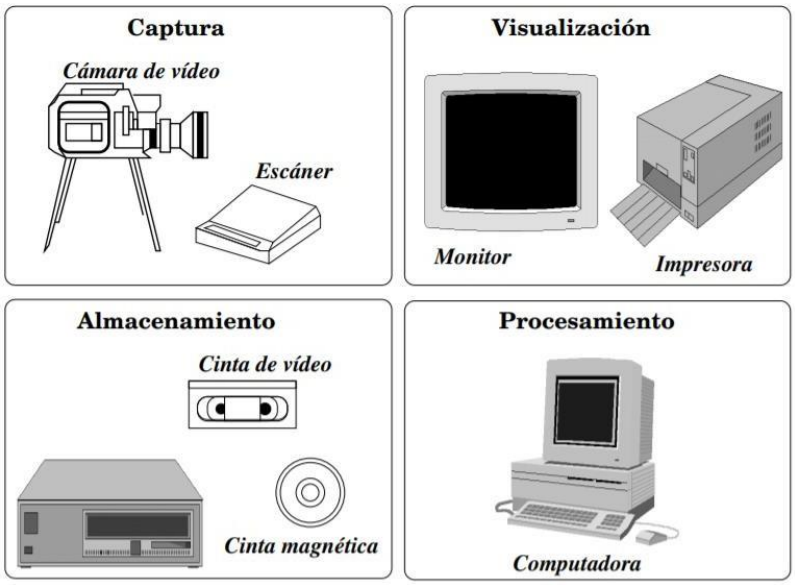


Figura II.7 Componentes d la visión artificial
Fuente: Fernández García, N. (2017)

1.4 Procesamiento de imágenes

En este apartado se tratarán con mayor profundidad las operaciones utilizadas en el proyecto, mientras que otras operaciones secundarias únicamente serán mencionadas

A) OPERACIONES DE PROCESAMEINTO GLOBAL

De acuerdo con García Mateos, G. (2017) las operaciones globales son aquellas en las que cada píxel es tratado de forma independiente, ya sea con una o con varias imágenes, de modo que las modificaciones realizadas afecten a toda la imagen en sí. Este tipo de operaciones se suelen utilizar para mejorar el histograma, reducir el ruido, componer imágenes, ajustar el color, etc., además que estas operaciones suelen combinarse con otros tipos de operaciones, las operaciones más comunes son:

– Aritméticas: sumar, restar, multiplicar, máximo, etc.
– Booleanas: and, or, not, etc.
– Transformaciones generales: Transformaciones de histograma, transformaciones de color, binarización, etc.

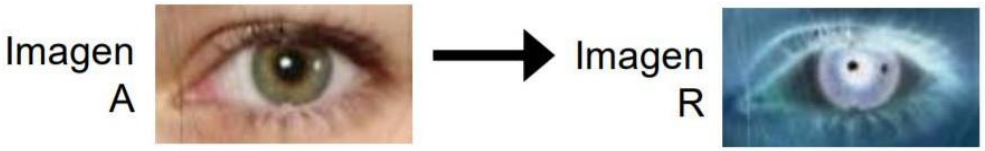


Figura II.8 Ejemplo de "inversión", operación global
Fuente: García Mateos, G. (2017).

a) Transformaciones de color

García Mateos, G. (2017) indica que toda imagen a color contiene 3 canales RGB, los cuales pueden ser modificados a la vez, determinando un cambio en la intensidad del color de la imagen, sin embargo, si se modifica cada canal por separado se tiene un cambio de color en la imagen.

Esto determina 2 posibilidades para realizar un cambio de color en una imagen:

- Aplicar transformaciones (suma, producto, ajuste de histograma, etc.), en distintos canales de la imagen para obtener el cambio de color
- Realizar transformaciones basadas en modelos de color. Cambiar el modelo de color (RGB, HSV, HLS, XYZ, YUV, etc.) y aplicar la función en ese modelo.

Figura II.9 Transformación de color en RGB
Fuente: García Mateos, G. (2017).

En nuestro caso se realizó una transformación de calor cambiando el modelo de color de RGB de la entrada a HSV, estos modelos son explicados más adelante.

b) Binarización

Gonzáles Marcos, A., et al. (2006) indica que la binarización consiste en convertir a cero ("0" que representa al color negro) los pixeles menores a un umbral determinado, y los pixeles restantes mayores o iguales al umbral convertirlos en uno ("1" que representa al color blanco).

Alegre, E., Pajares, G. & De la Escalera, A. (2016) cita a Ajenjo, 1993, quien indica que, "un umbral es un valor de intensidad a partir del cual un grupo determinado de píxeles serán considerados como pertenecientes a un subconjunto determinado y catalogados como blancos o negros"

En el ejemplo que se aprecia a continuación, observamos lo que sucede al variar el valor del umbral para realizar la binarización, por lo que se concluye que el éxito de este tipo de operaciones reside en determinar correctamente el valor de umbral para realizar las binarizaciones.

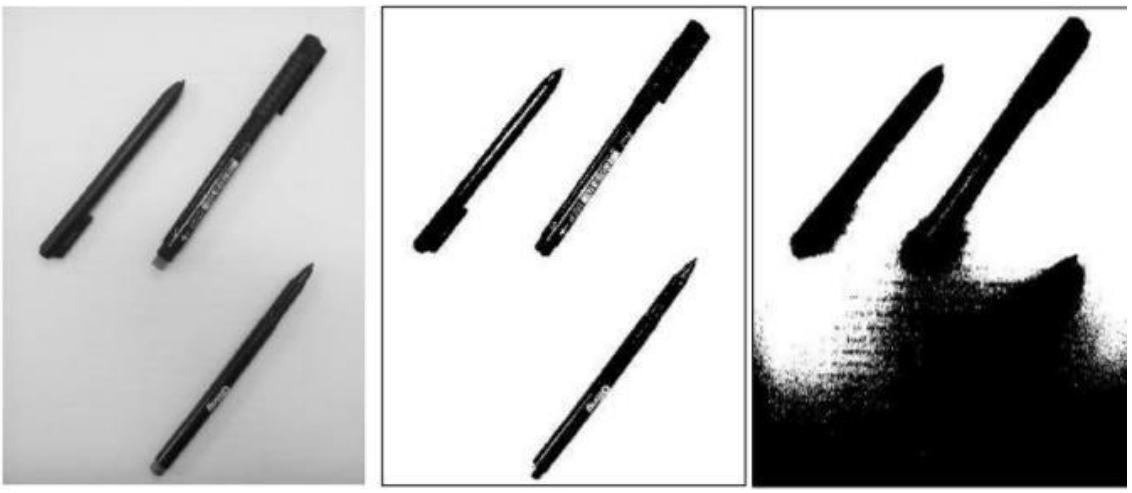


Figura II.10 Ejemplo de diferentes umbrales en binarizaciónFuente: Alegre, E., Pajares, G. & De la Escalera, A. (2016)

B) TRANSFORMACIONES GEOMÉTRICAS

Gonzáles Marcos, A., et al. (2006) indica que este tipo de algoritmos modifican las características geométricas de las imágenes, por lo que se utilizan en la reconstrucción de imágenes deformadas, para girarlas, realizarles ajustes, o darles alguna deformación intencional necesaria para su análisis.

Para conseguir estas modificaciones se debe redistribuir los píxeles de acuerdo con lo que se desee obtener Todos los algoritmos correspondientes a transformaciones geométricas se basan en realizar una nueva distribución de los pixeles según lo que se pretenda. Al calculo para conocer las nuevas posiciones a partir de las antiguas se llama interpolación.

García Mateos, G. (2017) termina indicando que las aplicaciones con transformaciones geométricas en entornos en tiempo real son muy reducidas, porque requieren bastante cálculo computacional. Termina indicando que el tamaño de la imagen de salida puede ser distinto del tamaño de la imagen de entrada, como se puede observar en la siguiente imagen

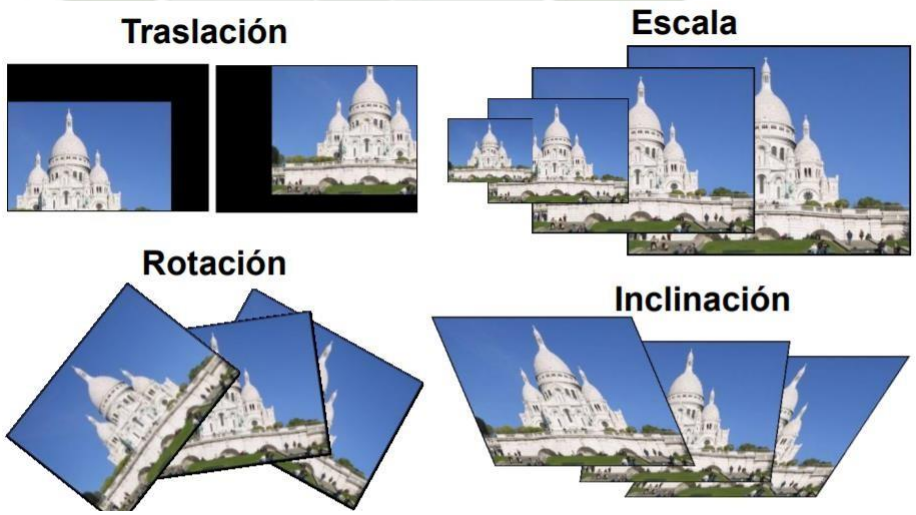


Figura II.11 Transformaciones geométricas varias
Fuente: García Mateos, G. (2017).

1.5 El color

Alegre, E., Pajares, G. & De la Escalera, A. (2016) indican que "la percepción visual del color se genera en nuestro cerebro a partir de las ondas electromagnéticas reflejadas por los objetos y captadas por los ojos". Sin embargo, para que un ordenador pueda describir los colores y descomponerlos en diferentes canales es necesario utilizar "espacios de color", los cuales son fórmulas matemáticas que realizan esa función. Así mismo Alegre, E., Pajares, G. & De la Escalera, A. (2016) indican que los espacios de color más utilizados son el RGB y el CMYK, donde el modelo RGB es utilizado por varios periféricos, tales como cámaras, escáneres, pantallas, etc.; además el modelo CMYK es utilizado en impresoras.

Nos centraremos en el estudio de los colores RGB y HSV que fueron utilizados en la elaboración del proyecto, sin embargo, mencionaremos de forma genérica a los otros modelos de color. En líneas generales, el espacio de color HSI es un sistema de coordenadas tridimensional (tono, saturación e intensidad), este sistema representa con un punto en el espacio a cada color. El espacio de color XYZ se utiliza para definir colores percibidos por el ojo humano; el espacio de color Lab identifica cada color de forma precisa.

Alegre, E., Pajares, G. & De la Escalera, A. (2016) brindan algunas definiciones básicas para comprender el color:

- Brillo: Indica si un área está más o menos iluminada.
- Tono: Indica si un área es similar al rojo, amarillo, azul o a una combinación de los anteriores.
- Colorido: Indica el grado de intensidad de los colores.
- Luminosidad: Es el brillo de una zona respecto a una zona blanca en la imagen.
- Croma: Es el colorido de un área respecto al brillo de un blanco de referencia.
- Saturación: es la relación entre el colorido y el brillo.

Así mismo los autores citan a Gevers y col. (2012) que indica que los parámetros relacionados con la visión humana son la luminancia, el tono y la saturación, parámetros que para ser transmitidos primero deben de ser transformarlos en parámetros eléctricos. El sistema más intuitivo comprende a las señales de luminancia cromática RGB, que es en la que se basan las cámaras de video para adquirir las imágenes

A) ESPACIOS DE COLOR

Alegre, E., Pajares, G. & De la Escalera, A. (2016) definen a los espacios de color como "métodos de representación que sirven para crear, visualizar o representar cualquier color". Indican así mismo que cualquier color puede ser representado mediante 3 cantidades o luminancias para definir cualitativa y cuantitativamente al color.

Como se había mencionado anteriormente, los periféricos como monitores, cámaras e incluso televisores trabajan con RGB, mientras que las impresoras trabajan con CMYK.

a) Espacio RGB

Fernández García, N. (2017) indica que este espacio es un sistema tridimensional de coordenadas cartesianas, donde cada color aparece debido a sus componentes espectrales primarias rojo, verde y azul, de donde deriva el nombre del espacio. Alegre, E., Pajares, G. & De la Escalera, A. (2016) recalcan que la naturaleza de los colores que puedan ser obtenidos es trivariante, ya que estos colores rojo, verde y azul son independientes entre sí, por lo que cualquier color "X" puede ser representado de la siguiente forma

$$X=R+G+B$$

Es de esta forma, que cualquier color queda definido como un punto dentro de un cubo de 3 dimensiones, pudiendo estar este punto en la superficie o interior del cubo, lo que se explica en la imagen que se ve a continuación:

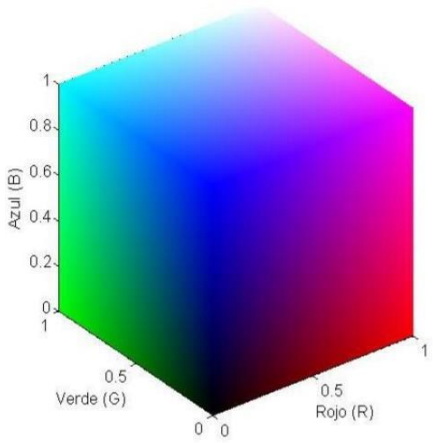


Figura II.12 Representación tridimensional del color en RGB
Fuente: Alegre, E., Pajares, G. & De la Escalera, A. (2016)

Para evitar problemas al momento de segmentar imágenes (que deberían de ser trabajadas en el espacio 3D del espacio de color) se puede normalizar el espacio de color RGB. Así mismo, Alegre, E., Pajares, G. & De la Escalera, A. (2016) realizan la descomposición de una imagen a color en sus 3 canales, rojo, verde y azul, que se observa en la imagen inferior, se observa que cada plano de color se representa en escala de grises.

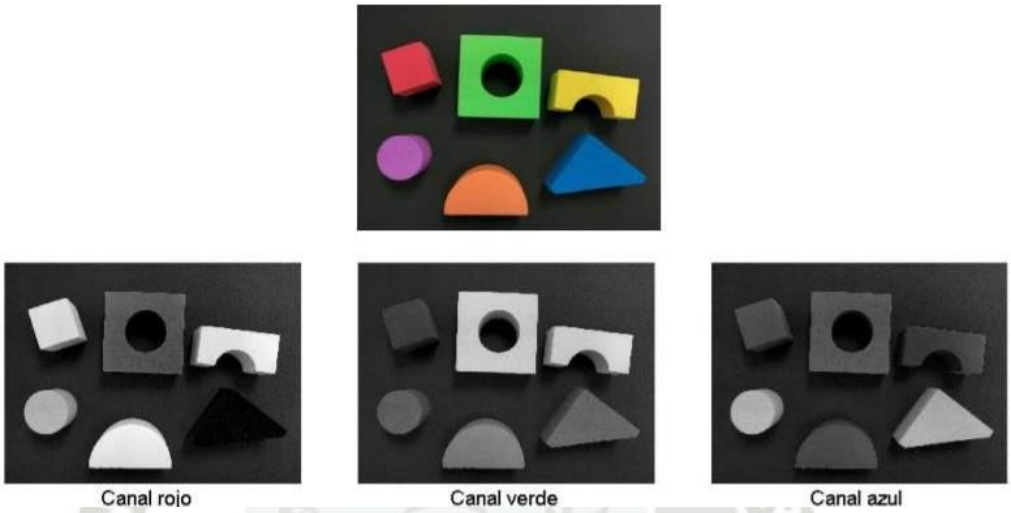


Figura II.13 Canales de una imagen en RGB
Fuente: Alegre, E., Pajares, G. & De la Escalera, A. (2016)

b) Espacio HSI-HSV

Los autores indican que este espacio emula la forma de percibir los colores que tenemos los seres humanos. El sistema determina al color por el tono o tinte (hue), saturación (saturation) y brillo (intensity), esto brinda ventajas al momento de segmentar imágenes, lo que se explicará con la siguiente comparación de imágenes:

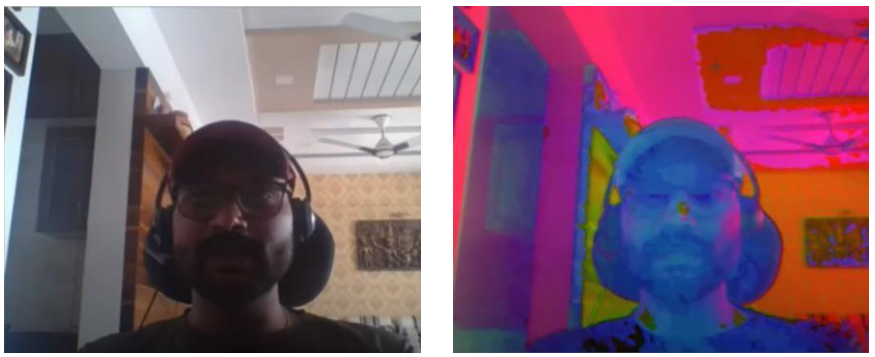


Figura II.14 Comparación entre RGB y HSV
Fuente: https://www.youtube.com/watch?v=uq7IIX4Ua3g

Debido a que el modelo HSI se centra en la detección del tono, se observa que los colores en la imagen en HSI se uniformizan a pesar de las sombras, o iluminaciones, ya que estos valores son captados por otros canales, lo que permite tener mejores resultados al trabajar con colores en este sistema, los autores citana Yoshida (2010) quien hace hincapié en estas ventajas del sistema HSI sobre elRGB.

Variaciones de este espacio de color son las siguientes:

- HSL (Hue, Saturation and Lightness)
- TSD (Hue, Saturation and Darkness)
- HSV (Hue, Saturation and Value)
- HVC (Hue, Value and Chroma)
- HCI (Hue, chroma and Intensity)

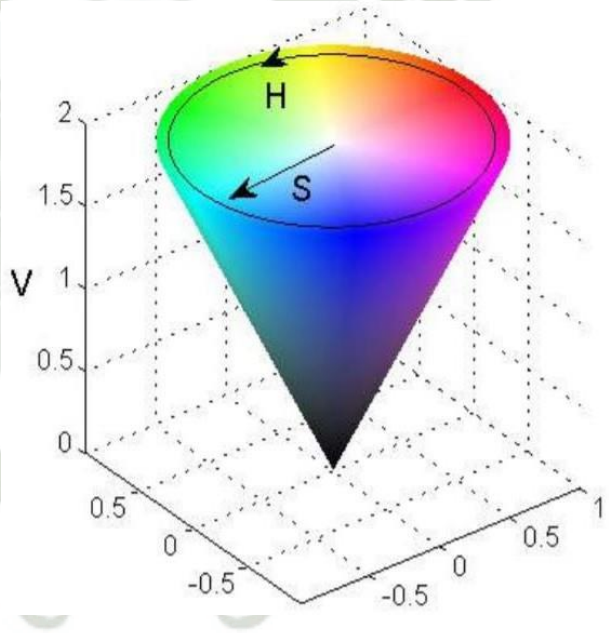


Figura II.15 Representación del color en HSV
Fuente: Alegre, E., Pajares, G. & De la Escalera, A. (2016)

En el caso del espacio HSV, se representan los colores a través de un cono invertido, donde pare seleccionar un color, primero se debe elegir un tono (H) de la región circular en valores angulares, luego se selecciona la saturación del color, es decir cuánto de “blanco” posee este color, en el eje horizontal del cono (S) y el valor o brillo del color en el eje vertical (V).

c) Espacio XYZ, Luv, Lab

La CIE "Commission Internationale de l'Eclairage" estableció un espacio de color "XYZ" que tenían el fin de corregir los errores que presentan los espacios de color anteriores (aparecen valores negativos para algunas longitudes de onda). De esta forma la CIE determina que la ordenada Y (luminosidad) será perpendicular a la "coloridad", determinada por el plano "XZ". Al trabajar con este espacio, se suele trabajar con valores normalizados entre 0 y 1.

Derivado de este espacio aparecen el espacio de color Luv y Lab, que representarán el tono o tinte, cromatismo e intensidad del color,

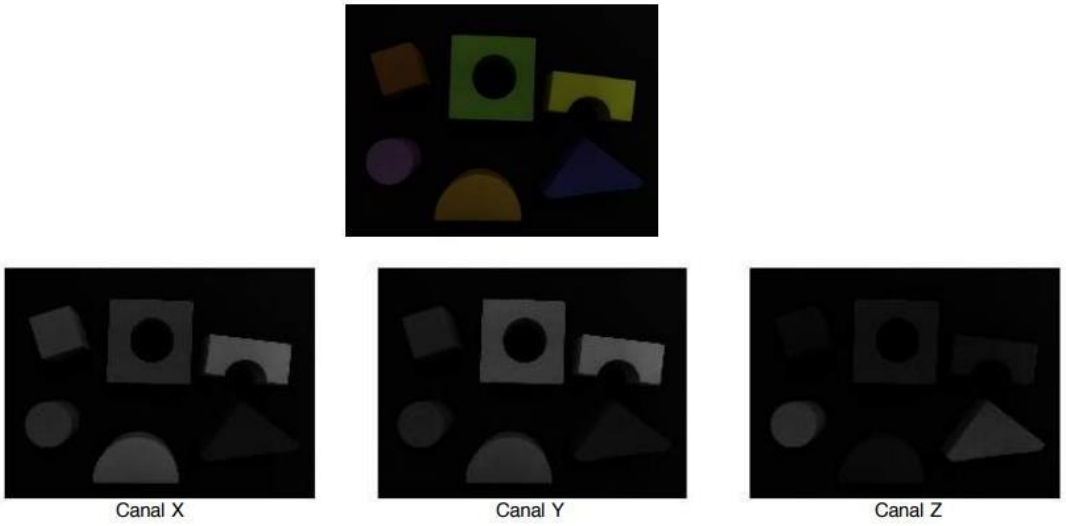


Figura II.16 Visualización de imagen en XYZ y descomposición en sus tres canales
Fuente: Alegre, E., Pajares, G. & De la Escalera, A. (2016)

d) Espacio YIQ, YUV, YCbCr, YCC

Se basan en valores de luminancia (luminosidad) y crominancia (color). Estos sistemas de color son los que utilizan algunas cámaras de color y las que también tienen las televisiones, YIQ para NTSC, YUV para PAL y YCbCr digital. En este espacio de color, la luminancia será calculada a partir de RGB con la siguiente ecuación:

$$Y = 0.30R + 0.59G + 0.11B$$

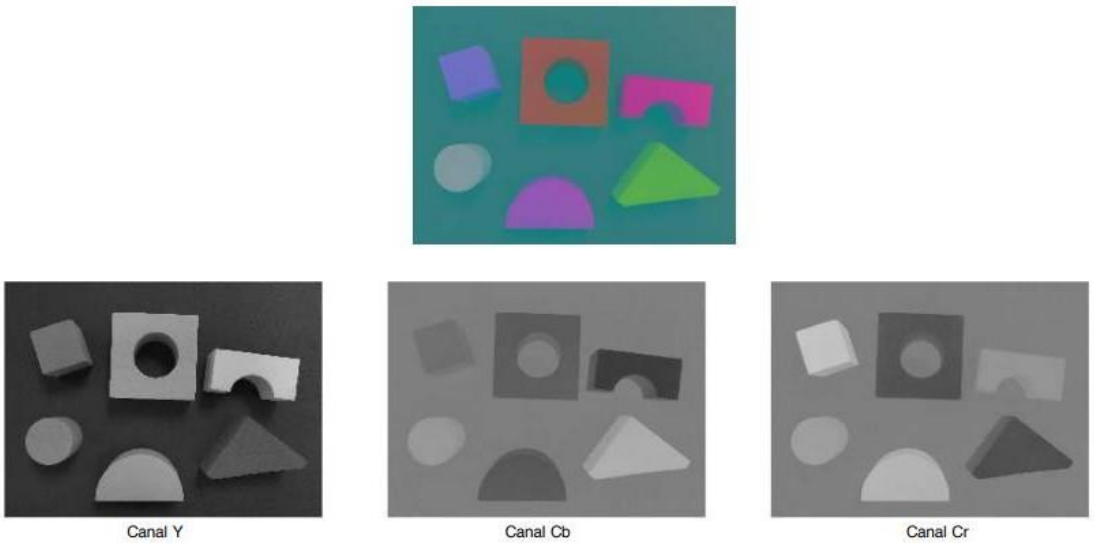


Figura II.17 Representación de imagen en YCbCr y descomposición en sus tres canales
Fuente: Alegre, E., Pajares, G. & De la Escalera, A. (2016)

1.6 Visión artificial en industria automotriz

Diversos autores, tales como Barba Guamán, (2015), Sánchez, (2015), Ponz Camps, (2016) y Barroso Rodríguez, (2016) coinciden en la gran aplicación de elementos inteligentes en los vehículos modernos, y dentro de estos elementos inteligentes, destaca la presencia de sensores y cámaras para identificar los elementos que están alrededor del vehículo, el objetivo común de la automoción actualmente es la "navegación autónoma", sin embargo, aún la tecnología se encuentra lejos de conseguir algo similar a esto.

El tema que más se relaciona con la detección de trayectorias es el de "navegación autónoma", por lo que se hablará al respecto. De acuerdo con lo expuesto por Ponz Camps, (2016), la tecnología de navegación autónoma debería antes resolver los siguientes parámetros:

- **Detección del entorno:** Se debe hacer un mapeo completo de los elementos que se encuentran alrededor nuestro, ya sean personas, señales de tráfico, semáforos, otros vehículos, actualmente este proceso lo realizan casi por completo los humanos, siendo algunas veces ayudados por diversas tecnologías de visión artificial u otras de sensores infrarrojos o ultrasonidos.

- **Ajuste de parámetros de acuerdo con las condiciones del entorno:** Este concepto viene ligado al anterior, ya que, de acuerdo al comportamiento de los elementos mapeados anteriormente, el sistema (vehículo), debe ser capaz de reaccionar de una manera óptima, ya sea aumentando o disminuyendo la velocidad y girando del volante esencialmente, para esto se deben de tener en cuenta los parámetros geométricos del vehículo.

- **Selección de la ruta a seguir**: Para generar la ruta deseada podemos indicar, desde el punto de partida y el punto de destino, un conjunto de puntos de paso. De esta forma se puede generar la ruta a partir de todo este conjunto de referencias, y, el proyecto propuesto consiste en poder capturar esta ruta, de acuerdo a imágenes referenciales y poder ordenar al vehículo a recorrerla.

Como se puede deducir, existen múltiples soluciones para cada uno de los ítems mencionados anteriormente, y adicionalmente se crean nuevos problemas internos que convierten a la navegación autónoma en un verdadero reto para la industria. Cabe mencionar que, este problema viene siendo tratado desde el 1990, cuando en el país de Holanda, se dio a conocer al mundo el primer proyecto de vehículo autónomo, este proyecto denominado "the PATH programa" planteaba poder conducir los coches en modo de un pelotón o convoy de vehículos.

Los sensores utilizados para este propósito fueron principalmente radares, utilizándose además sistemas de comunicación entre los diferentes vehículos y el control central, termina mencionando Ponz Camps, (2016)

Barba Guamán, (2015) menciona algunas de las actuales tecnologías utilizadas en automoción, las cuales son denominadas Sistemas Avanzados de Asistencia a la conducción (Advanced Driver Assistence Systems) (ADAS), dentro de las cuales encontramos a las siguientes:

- Sistemas de navegación a bordo. GPS
- Control de Crucero Adaptativo.
- Sistema anticolisión.
- Sistema de adaptación de velocidad inteligente.

- Sistemas de ayuda de asistencia en aparcamiento.
- Sistemas de comunicación Vehicular.
- Sistema de control de luces adaptativo.
- Sistema de visión nocturna.

En la imagen inferior se muestra la tecnología en visión artificial nocturna denominada BMW Night Vision, que es capaz de reconocer a los peatones y los animales de gran tamaño, y determinar su posición y proximidad respecto del coche

Figura II.18 BMW Night Vision

Fuente: https://www.abc.es/motor/reportajes/abci-vision-nocturna-puede-coche-jabali-200-metros-distancia-201808020310_noticia.html

2. ROBÓTICA MÓVIL

Se ha observado los avances que existen en el campo de la industria automotriz, sin embargo, esta tecnología sigue siendo aún costosa, y no del todo eficiente, por lo que existen debates sobre si "se puede uno fiar de la tecnología actual de los coches" o no, y peor aún, queda la duda de si "¿Será posible alcanzar la navegación autónoma?". En tanto, en el mundo de los robots móviles, nos encontramos con una gran variedad de tipos de robots móviles, como así lo mencionan Sosa Cervantes, (2016) y Llivisaca Aucapiña, C. (2018), quienes hacen una clasificación de los diferentes tipos de robots móviles de ruedas existentes, dentro de los que encontramos:

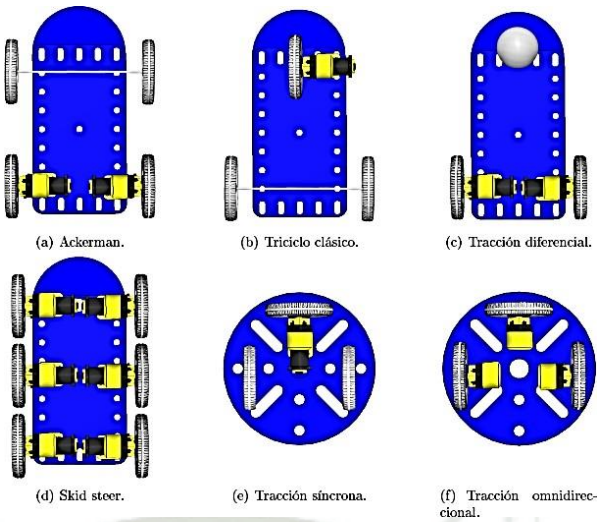


Figura II.19 Clasificación robots móviles con ruedas

Fuente: Técnicas de control automático para robots móviles de ruedas

Para nuestro estudio es conveniente analizar el modelo de coche "Ackerman", que será el utilizado para el proyecto, este robot móvil es el que utiliza cuatro ruedas convencionales, siendo muy semejante a un modelo real.

Llivisaca Aucapiña, C. (2018), explica este tipo de vehículos como sistemas basados en dos ruedas traseras tractoras que se montan de forma paralela en el chasis principal del vehículo, y las ruedas delanteras se encargan principalmente del direccionamiento, estas serán las encargadas de darle la dirección deseada al vehículo. En la imagen inferior, se observa un modelo de ejemplo de coche del tipo "Ackerman", implementado con diferentes elementos controladores.

Según este modelo, entonces se puede decir que la rueda delantera interior gira un ángulo ligeramente superior a la rueda exterior, de forma tal que los ejes de prolongación de las ruedas delanteras (directrices) y se cortan en el CIR o centro instantáneo de rotación, que se sitúa en un mismo punto que en el eje de prolongación del eje de las ruedas traseras (motrices). Esto elimina el deslizamiento que provoca los sobre virajes de la plataforma. El lugar de los puntos trazados sobre el suelo por los centros de los neumáticos, son circunferencias concéntricas con centro el eje de rotación CIR. Si no se tienen en cuenta las fuerzas centrífugas, los vectores de velocidad instantánea son tangentes a estas curvas. Por lo que las velocidades de movimiento del móvil deberán evitar que las ruedas no resbalen.

En los robots móviles con configuración Ackerman, se presentan dos ángulos de giro, uno en cada rueda. Esto genera mayores problemas a la hora de realizar el control, por lo que en muchas ocasiones lo que se hace es unificar los ángulos de direccionamiento en uno sólo, por lo que los radios de giro para los cuales el robot no muestra deslizamiento lateral son mayores que en otras configuraciones. En la imagen inferior se puede observar este efecto sobre el centro instantáneo de rotación (CIR).

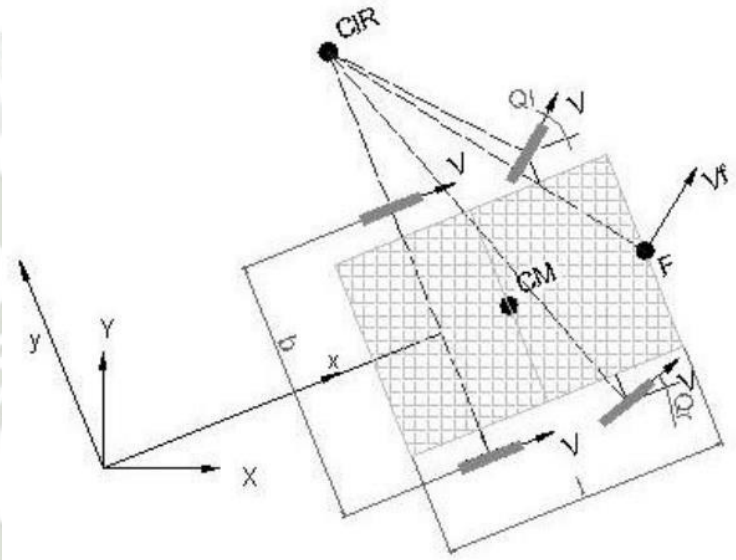


Figura II.20 Clasificación robots móviles con ruedas Sistema Ackermann

Fuente: Técnicas de control automático para robots móviles de ruedas

En la actualidad se producen varios tipos de "drones terrestres", como así lo evidencia Mori Virhuez, (2018) en su investigación, pudiendo mencionar a algunos de los drones terrestres tales como:

- Recon Scout XT, capaz de desplazarse en terrenos agrestes.
- Jumping Sumo: Drone de entretenimiento con funciones de exploración.
- Nerva LG: Posee una cámara capaz de brindar imágenes 360°.
- Summit – XL: Drone de investigación y vigilancia militar.

Este último robot móvil, es propiedad del Grupo de Innovación y Tecnología (GIT) de la Pontificia Universidad Católica del Perú, siendo utilizado en investigación y vigilancia, este robot móvil dispone de navegación autónoma ya que utiliza sensores de navegación, adicionalmente a esto puede ser tele operado, ya que cuenta con una cámara que transmite entiempo real.

Mori Virhuez, (2018) hizo una recopilación de características técnicas de robots móviles con ruedas, tales como son duración de la batería, peso, medidas, velocidades, etc., se adaptó de la tabla sólo la parte útil para el conocimiento del lector, respecto de robots móviles con ruedas, la tabla completa puede ser recuperadas de la fuente.

Estas características son:

Tabla II.2 *Características técnicas de robots móviles con ruedas*

Tipo de Drone		Duración	Peso	Medidas	Fuente de energía	Carga útil	Pendiente máxima	Velocidad	Rango de alcance
Terrestre	Jumping Sumo	20 min	0.21 kg	18.6 x 15.5 x 11.6 cm	Batería LiPo	No precisa	No precisa	8.5 km/h	0.05 km
	Recon Scout	60 min	0.64 kg	20.9 x 19.3 x 11.3 cm	Batería LiPo	No precisa	No precisa	1.7 km/h	0.09 km
	Nerva LG	120 min	5 kg	35 x 31 x 15 cm	Batería LiPo	No precisa	40°	13 km/h	1 km
	Summnit XL	300 min	45 kg	72.2 x 61.3 x 39.2 cm	LiFePo	20 kg	72°	10.8 km/h	No precisa

Fuente: Mori Virhuez, (2018)

3. DETECCIÓN DE LÍNEAS

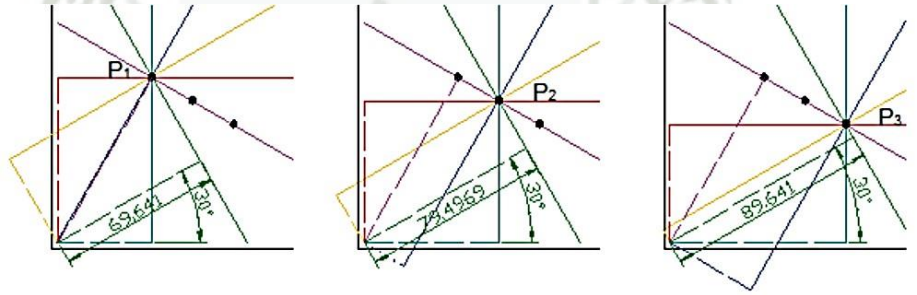
García, I.; Herrera, D. & Mina, Jorge (2017) Describe los métodos de detección de líneas que existen, utilizando únicamente algoritmos computacionales, de los cuales tenemos:

3.1 Transformada de Hough

De acuerdo con Montoya Baidal, D. F., & Pachacama Tamayo, A. R. (2016), utiliza la ecuación de recta en coordenadas polares, y ya no coordenadas cartesianas. Se considera que si tengo un punto (x0, y0), todas las posibles rectas que pasen por dicho punto se definen de la siguiente manera:

$$\rho_\theta = x_0 cos\theta + y_0 sin\theta$$

Esto significa que cualquier par de la forma (ρ_θ, θ) representa una recta que pasa por el punto requerido (x0, y0). Por ejemplo, si se toman puntos P1, P2 y P3, elegidos al azar, debe trazarse para cada punto una determinada cantidad de posibles rectas. Como se puede observar en la imagen inferior, en la Figura 2.19. Después se grafican los valores de recta de ρ y θ hallados, de esta forma se visualizarán curvas que representan cada posible combinación de ángulo y radio para rectas que contienen al punto. El gráfico obtenido se denomina "espacio de Hough"



P1 (x1, y1)	
θ	ρ
0	40
30	69,6
60	81,2
90	70
120	40.6
150	0,4

P2 (x2, y2)	
θ	ρ
0	57,1
30	79,5
60	80,5
90	60
120	23,4
150	-19,5

P3 (x3, y3)	
θ	ρ
0	74,6
30	89,6
60	80,6
90	50
120	6
150	-39,6

Figura II.21 Detección de líneas mediante la transformada de Hough
Fuente: Montoya Baidal & Pachacama Tamayo, (2016)

Para nuestro ejemplo observamos que las 3 curvas obtenidas presentan un valor en común (punto de intersección) que cumple con contener a los 3 puntos en una misma recta.

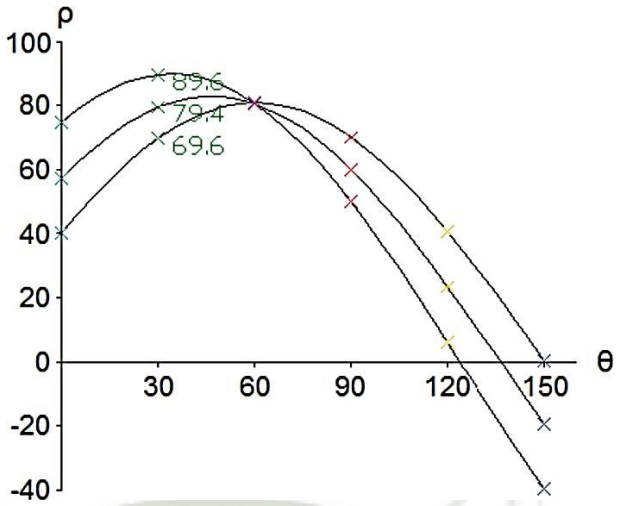


Figura II.22 Grafico del espacio de Hough
Fuente: Montoya Baidal & Pachacama Tamayo, (2016)

3.2 Regresión

Consiste en someter a la imagen a una serie de algoritmos que se encargarán de identificar líneas considerando una línea definida por la ecuación:

$$y = m * x + n$$

En base a la fórmula cartesiana de la recta y utilizando "mínimos cuadrados" para todos los puntos que se obtiene de la imagen, se encuentran los valores que definen la recta, los autores García, I.; Herrera, D. & Mina, Jorge (2017) citan varios trabajos donde se aplicó este método

3.3 Franjas horizontales

Se toman algunas Regiones de Interés (ROI) de la imagen, estas serán definidas al dividir la imagen en franjas horizontales y hallar centros de gravedad del conjunto de puntos que sean hallados en dichos ROI

3.4 Punto de fuga

En este método se proyectan las potenciales líneas y se encuentran puntos de fuga, que son cruces de líneas paralelas cuando tienden al infinito, de modo que aquellas líneas que presenten puntos de fuga son posibles candidatos

3.5 Filtrado

Cuando una imagen presenta líneas, se presentan determinados comportamientos después de filtrados de paso de banda o en el dominio de la frecuencia, de esta forma, García, I.; Herrera, D. & Mina, Jorge (2017) citan a Bossu et al. (2009) y Vioix et al. (2002), quienes aplicaron estos métodos en sus trabajos de información.

4. CONTROL DIFUSO Y CONTROL PID CONVENCIONAL

Según Guanrong, C. & Tat Pham, T. (2006) los controladores convencionales son ampliamente usado en las industrias, ya que estadísticamente el 90% de industrias aplica este tipo de controladores o algunas variantes del PID. Este tipo de controladores son simples, confiables y efectivos, siendo óptimos para sistemas lineales de bajo orden, en cuyos casos, debido a sus características son más fáciles de aplicar que los controladores difusos.

Sin embargo, para sistemas de un orden elevado, que no son lineales y que presenta delays en el tiempo, cuyos sistemas matemáticos son más inciertos o difíciles de calcular, o simplemente son imposibles de hallar a mano, el control difuso presenta un gran rendimiento, el único inconveniente en este tipo de sistemas es que su estructura es más complicada. Se vio, sin embargo, que a pesar de que la lógica detrás de los controladores convencionales y difusos es alfo diferente, en cuanto a su estructura es bastante similar, pudiéndose adaptar fácilmente un sistema difuso directamente desde uno convencional y viceversa. Es por esto que se hará un breve repaso del control PID y seguidamente de hablará del control difuso

4.1 Control PID

De acuerdo con Katsuhiko, Ogata (2010) los controladores PID permiten controlar un sistema en lazo cerrado para que alcance el “set point” o salida deseada. Este controlador PID está compuesto de tres métodos de control que proporcionan una acción Proporcional, Integral y Derivativa, de donde se deriva el nombre del controlador

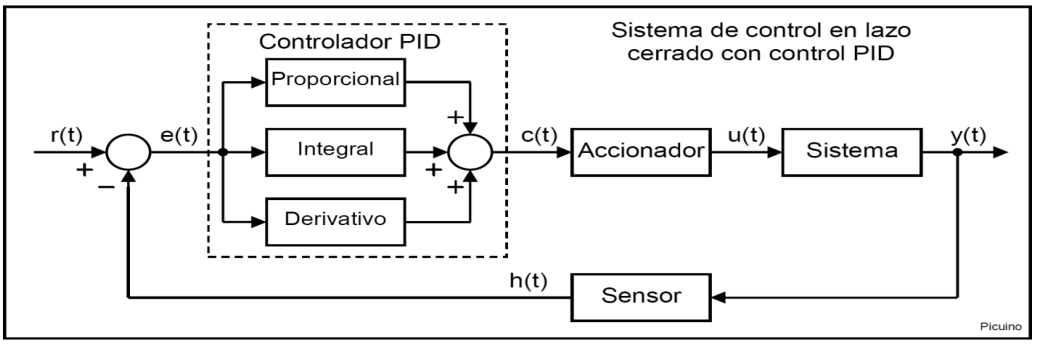


Figura II.23 Modelo de la estructura del controlador convencional PID
Fuente: https://www.picuino.com/es/control-pid.html

A) ACCIÓN DE CONTROL PROPORCIONAL

Esta acción de control es proporcional a la señal de error e(t), lo que significa que se multiplica la señal de error por una constante Kp, de modo que se pueda minimizar el error.

- Aumenta la velocidad de respuesta del sistema, se llega antes al set point deseado.
- Disminuye el error del sistema en estado estacionario, con Kp adecuados, aunque no siempre se consigue eliminar el error con la acción proporcional.
- Aumenta la inestabilidad del sistema, con valores muy altos de Kp.

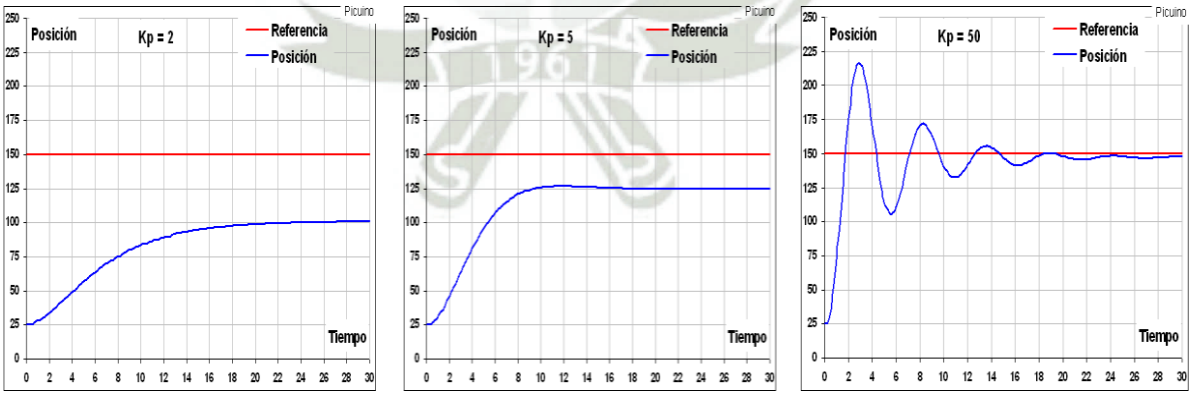


Figura II.24 Comportamiento de la planta con Controlador Proporcional
Fuente: https://www.picuino.com/es/control-pid.html

B) ACCIÓN DE CONTROL DERIVATIVA

Esta acción de control es proporcional a la derivada de la señal de error e(t), es decir se controlará en función a la diferencia de error que haya en el sistema.

- Aumenta la estabilidad del sistema controlado, sin embargo, con señales muy oscilantes o con frecuencia alta, podría tener más dificultades.
- Disminuye la velocidad del sistema, sin embargo, no es efecto muy grande.
- El error en estado estacionario permanecerá igual, por lo que tampoco es una acción definitiva.

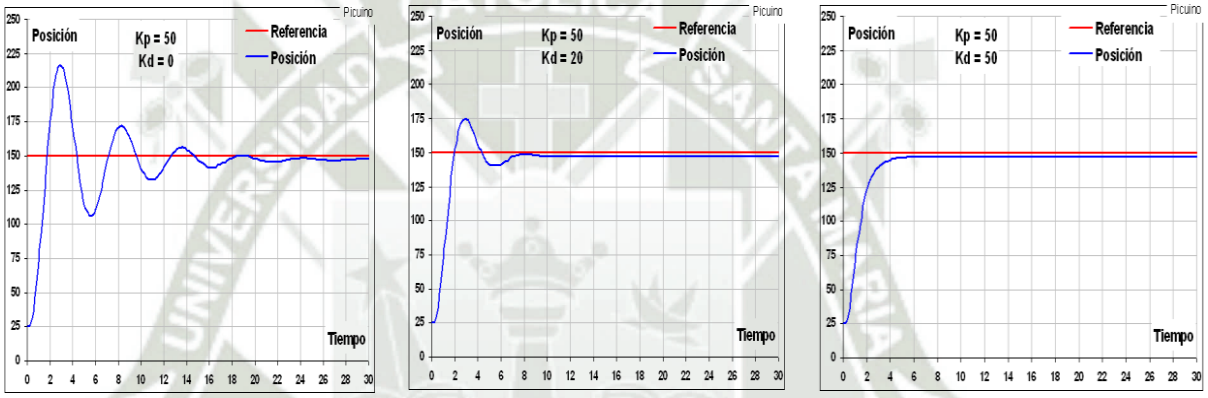


Figura II.25 Comportamiento de la planta con Controlador Derivativo
Fuente: https://www.picuino.com/es/control-pid.html

C) ACCIÓN DE CONTROL INTEGRAL

Esta acción calcula la integral de la señal de error e(t), es decir, trabaja con la suma o acumulación de errores, por lo que a medida que pasa el tiempo la acción integral es cada vez más grande. Mientras se va aumentando el valor Ki se tienen los siguientes efectos, que son visualizados en las gráficas inferiores:

- Disminuye el error del sistema en estado estacionario.
- Aumenta la inestabilidad del sistema, por lo mismo que depende de la suma de errores, creando una especie de inercia en el sistema.
- Aumenta la velocidad del sistema, aunque no es un efecto muy grande.

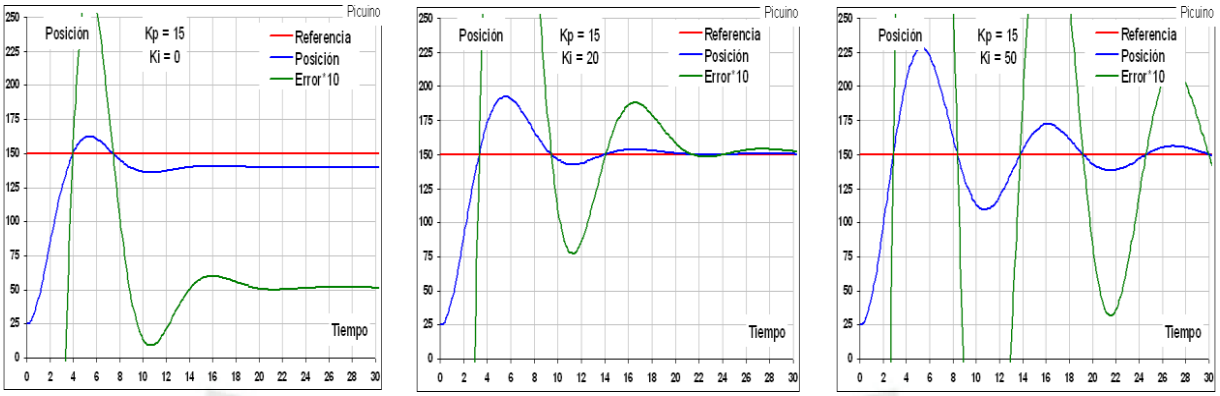


Figura II.26 Comportamiento de la planta con Controlado Integral
Fuente: https://www.picuino.com/es/control-pid.html

D) ECUACIÓN DEL CONTROLADOR

La ecuación del control PID es la siguiente:

$$c(t) = Kp \times e(t) + Ki \times \int e(t)dt + Kd \times \frac{\partial e(t)}{\partial t}$$

Para:

c(t) = señal de control

e(t) = señal de error

Kp, Ki, Kd = parámetros del controlador PID

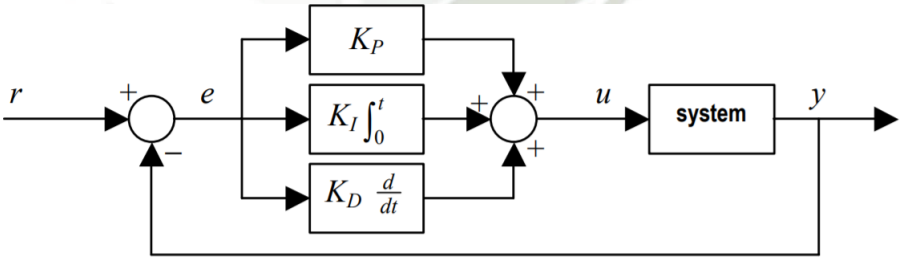


Figura II.27 Estructura del control convencional PID
Fuente: Guanrong, C. & Tat Pham, T. (2006)

Sin embargo, de acuerdo con Guanrong, C. & Tat Pham, T. (2006), los controladores PID sólo tienen una bonita estructura, pero no son prácticos en aplicaciones reales, por lo que los autores lo denominan “Controlador PID de libro de texto” y sugieren utilizar controladores PI+D, que es una variante que

utiliza los 3 controladores, pero que separa la componente Derivativa y la utiliza para analizar la salida del error luego de pasar por las componentes PI, es decir que analiza una señal más uniforme y suavizada. La estructura de este controlador se ve a continuación.

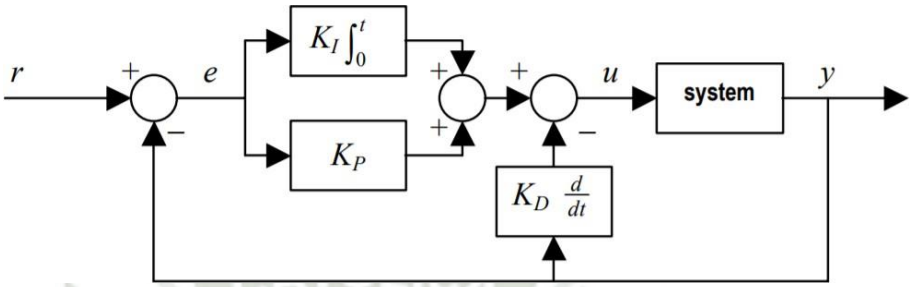


Figura II.28 Estructura del controlador PI+D convencional
Fuente: Guanrong, C. & Tat Pham, T. (2006)

4.2 Control difuso

En cuanto al control difuso Guanrong, C. & Tat Pham, T. (2006), indica que la estructura de este tipo de controlador es similar a la que sigue el PID convencional, por lo que estructura básica para este tipo de controladores sería la que se observa en la imagen inferior. Se reemplazan los controladores PID por sistema que consiste en un Fuzzyficador de la información que debe entrar en un controlador lógico-difuso (CLD), de modo que el controlador sea capaz de "entender" las señales de entrada y finalmente se cuenta con un Defuzzyficador que brinde señales de salida entendibles al sistema, además dentro del CLD, existirán "reglas difusas" que harán posible el control.

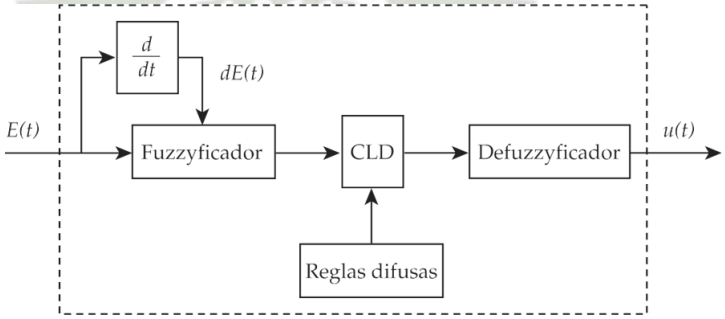


Figura II.29 Diagrama a bloques de un controlador difuso genérico.
Fuente: Diciembre Sanahuja, M. (2016)

El autor indica que los sistemas que trabajan con sistemas difusos se basan en el conocimiento de los expertos que diseñaron las reglas y parámetros adecuados para el correcto funcionamiento, desempeño y rendimiento del sistema.

Como se mencionó anteriormente, para el diseño del controlador difuso no es necesario conocer la fórmula matemática que define a la planta, que puede ser lineal o no lineal, puede ser de primer o segundo orden o ser de orden superior, el control difuso es capaz de trabajar sin conocer este aspecto del sistema.

La etapa de fuzzificación se encarga de transformar los valores de la señal de error en un subconjunto difuso normalizado. Existen diversas funciones de membresía, cada una de ellas con diferente rango de valores, por lo que, a la señal de error se le asignará una función de membresía debido a que el valor del error está comprendido dentro del rango de la función de membresía que le fue asignado. De esta manera se consigue hacer compatible la señal de error con la base de reglas del control difuso. Cada función de membresía debe ser cuidadosamente diseñado para poder satisfacer correctamente las necesidades del sistema.

La segunda etapa del controlador difuso es el motor de inferencia, que es el encargado de crear las acciones de control entre las entradas del controlador y las salidas que le corresponden a los mismos. Cada una de estas reglas sigue el modelo “Si-Entonces”

$$SI \quad entrada\ al\ controlador\ es\ E1 \quad Y \ldots Y$$
$$entrada\ al\ controlador\ es\ En$$
$$ENTONCES\ salida\ del\ controlador\ es\ U1$$

Esta definición de reglas dependerá de la experiencia del diseñador, de los conocimientos físicos que se tengan sobre la planta, y de las habilidades de análisis y diseño que se tenga.

La tercera etapa del control difuso es la defuzzificación se encarga de transformar las señales obtenidas en el área de inferencia y convertirlas en señales reconocibles por el sistema a controlar, en esencia, es el proceso contrario al proceso de fuzzificacion. Para poder realizar este proceso existen varias fórmulas efectivas disponibles.

Ya sea que en el fusificador se trabaje con la derivada de la señal de entrada, o incluso con la integral de esta, existirán ciertas arquitecturas para el CLD que se explicarán a continuación

A) ARQUITECTURA MAMDANI

De acuerdo con Diciembre Sanahuja, M. (2016), este método es el más utilizado. Fue propuesto en 1975 por Ebrahim Mamdani y consta de 4 pasos:

1. Fusificación de las variables de entrada.
2. Evaluación de las reglas.
3. Agregación de las salidas de las reglas.
4. Defusificación.

Este método utiliza reglas difusas del tipo "SI-ENTONCES" (IF-THEN), por lo que se trabaja con los valores de entrada, ya sea único o múltiples valores de entrada (unidos por conectores lógicos del tipo AND u OR) para poder asociarlos a un valor de salida establecido por la regla, de este modo, se pueden obtener varias reglas que hagan trabajar a los diferentes grupos de salida.

Este método trabaja con grupos difusos, de acuerdo con Ornelas Tellez, F. (2015) en el modelado de Mamdani, hay 2 métodos de inferencia más utilizados: Min-Max y Product-Max. La Figura 2.28 muestra una interpretación gráfica del método Min-Max de inferencia para 2 entradas y 2 reglas de inferencia.

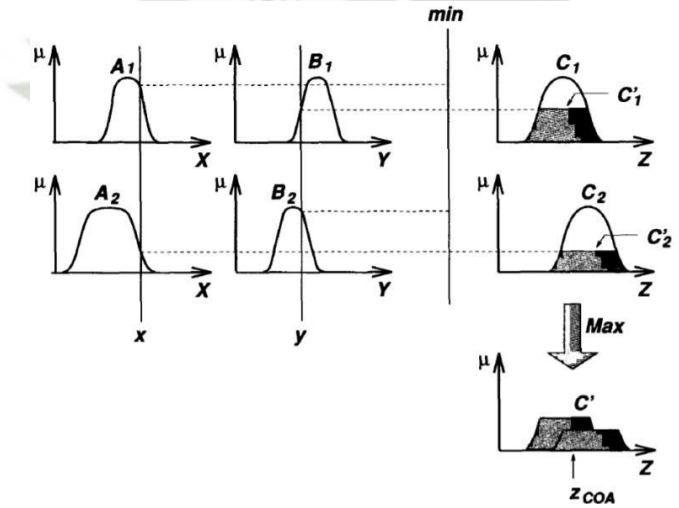


Figura II.30 Método de inferencia Min-Max
Fuente: Ornelas Tellez, F. (2015)

Así mismo se hace la demostración del comportamiento del método de inferencia Product-Max en la figura 2.29

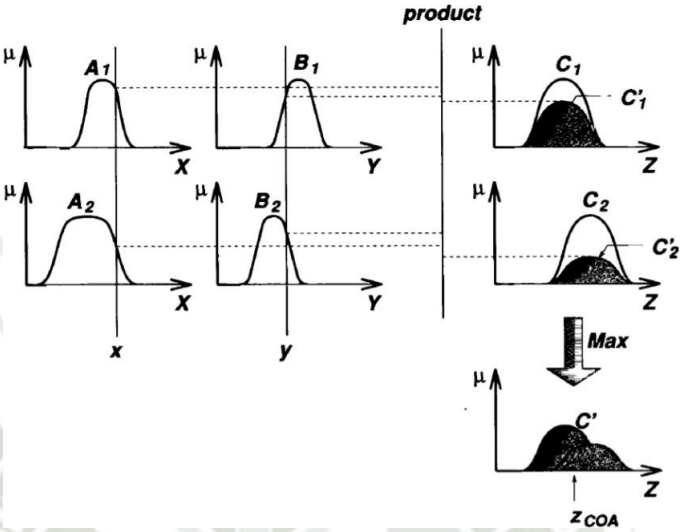


Figura II.31 Método de inferencia Product-Max
Fuente: Ornelas Tellez, F. (2015)

B) ARQUITECTURA TAKAGI-SUGENO-KANG (TSK)

La arquitectura TSK también tiene la forma de regla del tipo "si-entonces", pero relaciona los valores de entrada con los de salida utilizando funciones, es decir:

$$Regla\ i\text{: } si\ X(t)\ es\ A\ entonces\ Z\ es\ Y = ko + kiX(t)$$

donde A es un conjunto difuso cualquiera, de donde se tendrán los valores X(t), que son valores de entrada, con los cuales se podrán hallar los Z son los valores de salida o atributos controlables del problema a través de la función lineal Y.

Al momento de comparar las arquitecturas vistas anteriormente deducimos que la TSK es más precisa en cuanto a los valores de salida que se desean obtener porque disponemos de una función que es capaz de dar valores exactos, sin embargo, el método de Mamdani es más experimental y abierto a cambios

C) ARQUITECTURA TSUKAMOTO

Esta arquitectura tiene la particularidad de utilizar a la salida una función monótona, creciente o decreciente. De esta forma, la estructura de este modelo sería

$$Si\ Xi\ es\ Ai\ y\ \dots y\ Xn\ es\ An, entonces\ U\ es\ g(\alpha)$$

Esto implica que a la salida se tendrán valores certeros, el cual será calculado de forma similar que en un sistema TSK. Adicionalmente es interesante puntuar que el modelo Tsukamoto no sigue un proceso estricto de inferencia composicional difusa

Capítulo III

DISEÑO CONCEPTUAL

1. ARQUITECTURA DEL ROBOT

1.1 SOFTWARE

Para poder desarrollar este proyecto, se decidió seguir la siguiente ruta de acción, primero, para la adquisición de datos, se utilizó una cámara de celular y una conexión inalámbrica, estos datos pasaron a través del procesamiento de imágenes utilizando filtros geométricos y globales, para después pasar al reconocimiento de líneas (Transformada Hough), los datos obtenidos de este procesamiento se utilizaron para determinar la posición del modelo de auto dentro del carril y los cambios en el ángulo de giro de las llantas que se necesitó realizar, finalmente de la nueva posición que se generó en el modelo de vehículo se realimentó al controlador, para que siguiera corrigiendo el error de centro mediante el giro de las llantas. Así se repitió este proceso hasta conseguir que el modelo de vehículo se posicione al medio del carril, además en este capítulo se especificaron los componentes y programas a utilizar.

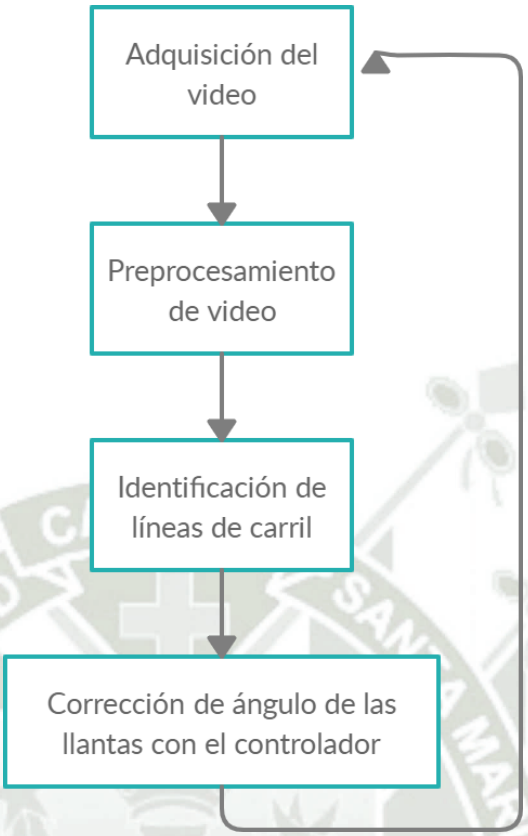


Figura III.1 *Grafico de secuencia de funcionamiento del programa*
Fuente: Elaboración propia

D) ADQUISICIÓN DEL VIDEO

En esta primera etapa se hizo la captación en video del ambiente que rodea al modelo de vehículo, a través del uso de la aplicación "DroidCam" es que se hizo la captación de la imagen y se la envió a la computadora en donde tendrá un preprocesamiento antes de realizar la identificación de líneas de carril, este procesamiento se encuentra detallado en el anexo A.3.

E) PREPROCESAMIENTO DE VIDEO

Para el preprocesamiento, los autores sugieren varios métodos para evitar introducir señales de error dentro del código de control, de modo que se tenga una mejor respuesta. AL respecto Correa Sandoval & Díaz Zapata, (2019) hicieron un esquema de procesamiento sencillo para detección de líneas. Para esta segunda etapa,

se tendrá en cuenta que al momento de realizarse las transformaciones globales y geométricas se recuperen datos relevantes para el código, de modo que se evite introducir muchos métodos de identificación de líneas.

De este modo los procesos a seguir son los que se especifican a continuación:

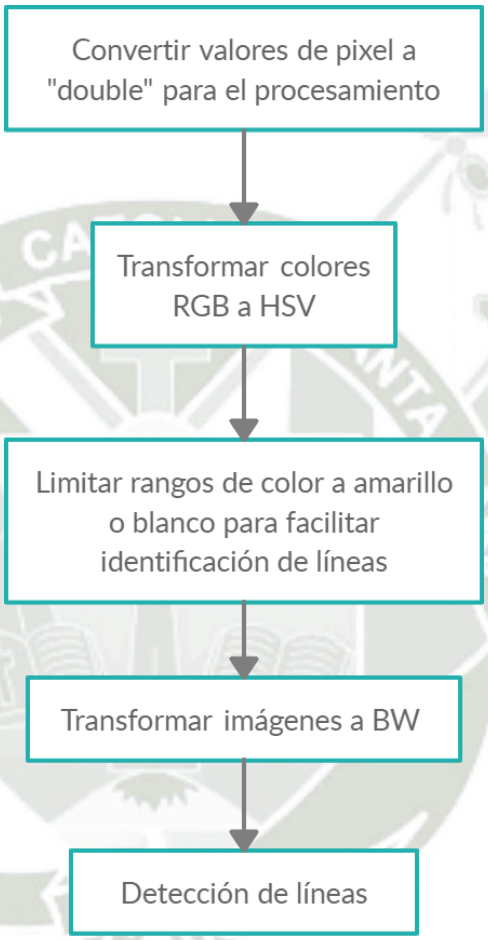


Figura III.2 *Grafico de secuencia de preprocesamiento de video*
Fuente: Elaboración propia

F) IDENTIFICACIÓN DE LÍNEAS DE CARRIL

En esta tercera etapa se llevó a cabo una de las partes más decisivas del código, ya que depende de la correcta identificación de las líneas de carril para que el controlador pueda realizar una correcta corrección del ángulo de giro de las llantas del modelo de vehículo, a la imagen en blanco y negro que llegó del paso anterior se le aplicó la Transformada Hough para poder determinar la presencia de líneas rectas o curvas en la carretera.

Adicionalmente se superpusieron las líneas halladas en este paso sobre las líneas del carril detectado, indicando que de esas líneas es que se realizará el procesamiento.

G) CORRECIÓN DEL ÁNGULO DE LAS LLANTAS CON EL CONTROLADOR

Del apartado anterior se obtuvieron los datos de los puntos que conforman las líneas de carril, es con estos datos se pudo saber si el automóvil se encontraba en el medio de estos valores (si el auto va por el medio del carril o se encuentra pegado a alguna línea de carril), y mediante los controladores difusos se ordenó el giro de las ruedas del modelo de vehículo a fin de corregir este error y se sitúe a la mitad del carril.

Se eligieron controladores difusos para tener flexibilidad en el control del vehículo y se decidió utilizar el modelo Mamdani PD, fue necesario experimentar luego de la elaboración del control para mejorar los resultados, ya que este tipo de controlador depende de la “experiencia del que diseñe el controlador”.

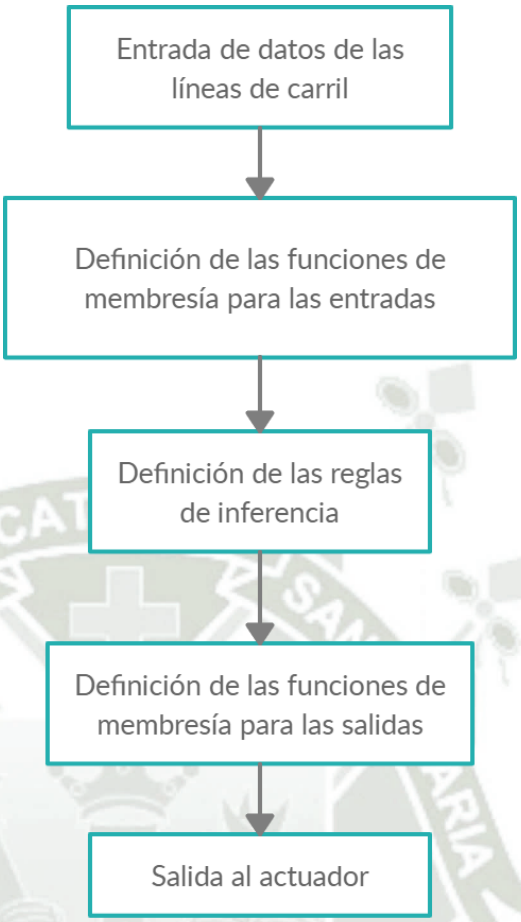


Figura III.3 *Grafico de secuencia de preprocesamiento de video*
Fuente: Elaboración propia

1.2 HARDWARE

También es importante definir los elementos mecánicos que serán gobernados por el código, ya que de acuerdo con sus características es que se definen los parámetros del código y controladores. El modelo de vehículo no será diseñado tal cual, sino que se utilizará un automóvil de juguete, en el cual se realizaran las modificaciones necesarias con el fin de adaptar los elementos que intervienen en el control del giro de las ruedas, se ha pensado en el siguiente esquema sobre el cual se puede trabajar.

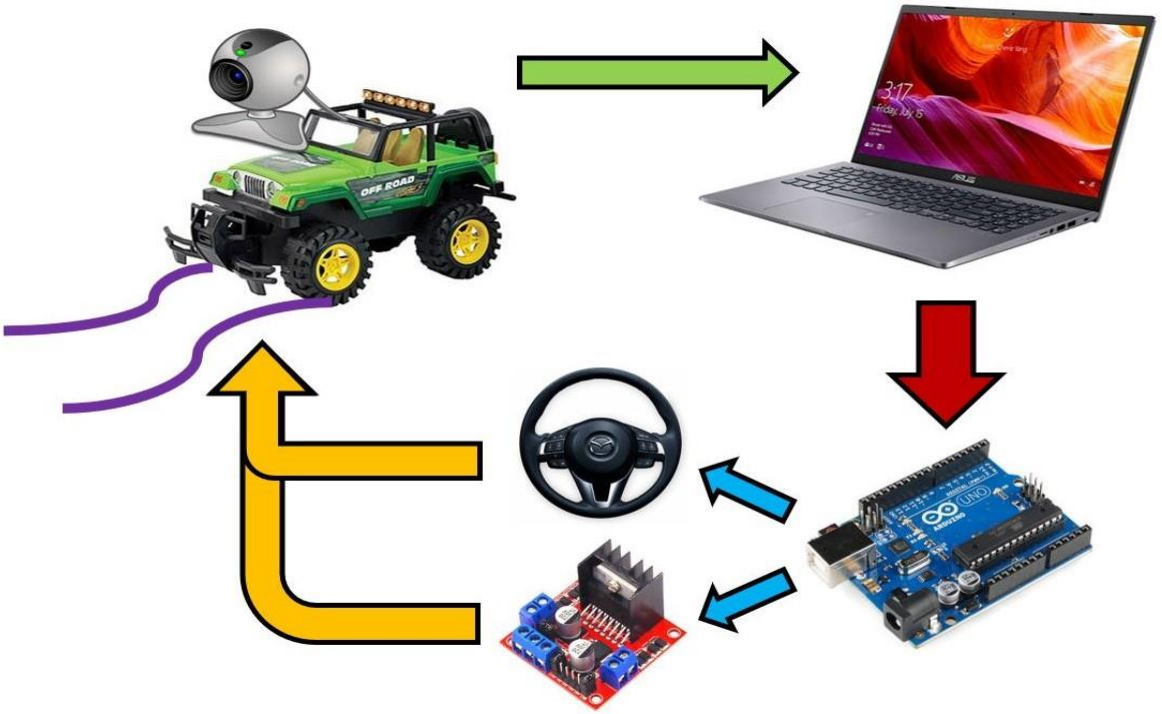


Figura III.4 *Grafico de secuencia de preprocesamiento de video*
Fuente: Elaboración propia

En la Figura 3.4 se observa el esquema de funcionamiento e interacción de los diferentes componentes del sistema entre sí, se observa que en el modelo de vehículo tiene integrado una cámara, que será la encargada de reconocer las líneas de carril y enviar esta información a la computadora, donde se realiza todo el procesamiento del video y se devuelve un resultado a través del Arduino, que será el encargado de controlar el ángulo de giro de timón, además también se encarga de enviar señal a un puente H que controla la velocidad del vehículo mientras se mueve, sin embargo esta velocidad será constante, por lo que la salida del Arduino al Puente H solo servirá para que este pueda brindarle al motor del vehículo una señal suficiente para tener una velocidad a la cual se pueda ejecutar el control de manera correcta. A continuación, se hace una descripción de los componentes a usar.

A) CÁMARA

En este proyecto se vio por conveniente utilizar una cámara de celular, debido a que las cámaras existentes necesitan una forma de adaptación especial para poder enviar los datos correctamente a la computadora, por lo que utilizar la cámara de celular era la opción más viable y práctica para la obtención del video del entorno.

Para poder adquirir la seña enviada por la cámara en la laptop y posteriormente poderla procesar, fue necesario utilizar la aplicación Droidcam, que generará una dirección IP para que las imágenes captadas por la cámara puedan ser visibles por la laptop, el proceso seguido a detalle se especifica en el anexo A.3.

B) COMPUTADOR

Será el encargado de realizar todo el procesamiento descrito en la parte de software, se utilizará un computador por su potencia de procesamiento.

C) ARDUINO

Se utilizará el Arduino UNO, que únicamente tendrá el rol de enviar las señales que envíe la computadora a los actuadores del modelo de vehículo, como se explicó anteriormente, se controlan el ángulo de giro de las llantas y la velocidad del vehículo, sin embargo, el único valor que variará será el valor del ángulo de las ruedas, ya que la velocidad del modelo de vehículo ser ajustará de modo que se tenga una velocidad que permita que el código funcione adecuadamente.

Capítulo IV

DISEÑO DE DETALLE

1. DISEÑO SOFTWARE

Diversos autores como Romero-Macas, H. (2020) y Andrade Fierro, L. & Chulca Simbaña, L. (2018) sugieren el uso de MATLAB como software para el procesamiento de las señales de visión computacional. Dentro del estudio del primer autor, se realizó un control PID sintonizado mediante redes neuronales, utilizando SIMULINK como soporte para dichas operaciones. El segundo autor estudió el control PID de un robot móvil de tres grados de libertad, utilizando SIMULINK para el procesamiento de los datos, terminando en un control eficiente y preciso. Se puede observar entonces que, dentro de MATLAB, la herramienta SIMULINK es de bastante ayuda.

Sin embargo, otros autores como Montoya Baidal & Pachacama Tamayo, (2016), Ladero García, (2019) o Tosini & Leiva, (2018) hacen uso de otra plataforma llamada Open CV y de placas de control tipo Raspe Berry.

Ante estos diferentes métodos de control, se eligió el método de control utilizado MATLAB y la herramienta SIMULINK para procesamiento de datos, debido a que se tiene

más información respecto a este programa, así mismo Aral Sarrafi (2021) propuso un modelo, utilizando SIMULINK, para la detección de líneas de carril, el cual puede ser de gran utilidad al momento de realizar la detección de líneas de carril. De la misma forma, y por facilidad en precio y adquisición de componentes, se eligió trabajar con la placa Arduino.

1.1 Interfaz Arduino y Matlab

En nuestro proyecto se utilizó una placa Arduino con el microprocesador AtMega328P, y para lograr controlar nuestro vehículo de acuerdo con la posición angular de las ruedas, se utilizó Matlab para el intercambio de datos y para realizar la observación y las pruebas de nuestro sistema definido.

Para ello se realizó primero la comprobación o instalación de algunos toolbox en el computador, los que fueron:

- MATLAB Support Package for Arduino Hardware.
- Simulink Support Package for Arduino Hardware.
- Simulink 3D Animation.

El proceso de instalación de estos toolbox se especificó en el anexo A.1.

1.2 Adquisición de video de una cámara externa

Al igual que el ítem anterior, este paso fue omitido, porque fue parte de la implementación física, se encuentra en los anexos (A.2).

1.3 Adquisición de video de una cámara externa en Matlab

Al igual que el ítem anterior, este paso fue omitido porque es un proceso que se repite en varias partes de la tesis, se encuentra en los anexos (A.3).

1.4 Descripción del Simulink

Se adaptó del Simulink desarrollado por Aral Sarrafi (2021) un sistema de control PD difuso de trayectoria para un modelo de vehículo mediante la detección de líneas de carril, quedando el siguiente sistema que será desarrollado a continuación:

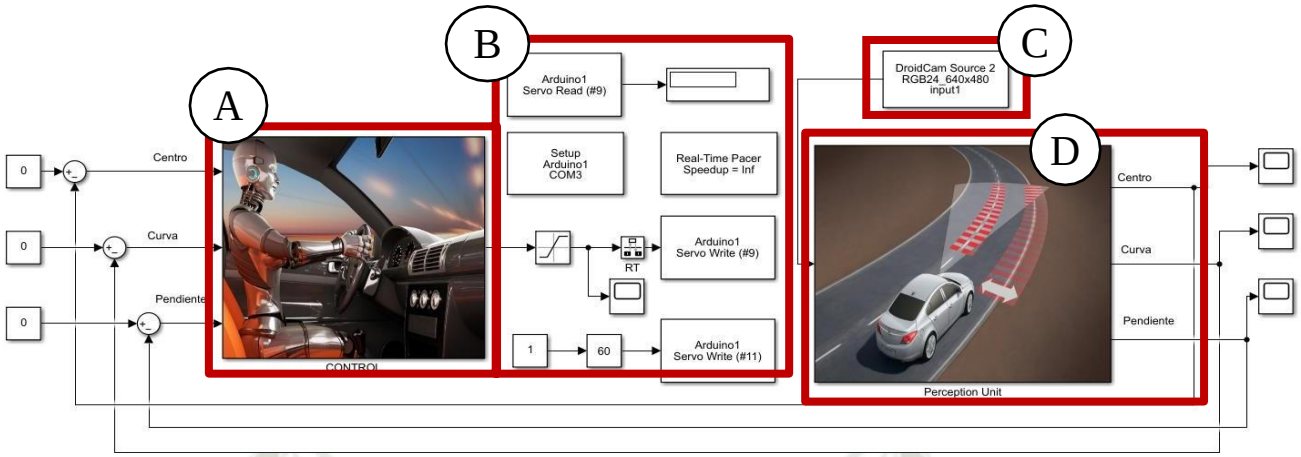


Figura IV.1 *Esquema de procesamiento de señal*
Fuente: Elaboración propia

Se observan las siguientes partes del sistema que serán descritas posteriormente:

A) Bloque de control.

B) Bloque de control del servomotor.

C) Bloque de visión.

D) Bloque de procesamiento.

A) BLOQUE DE CONTROL

Al existir varios factores que intervienen en el correcto control de trayectoria del modelo de vehículo, se decidió tomar 3 de los factores más importantes para trabajar con ellos, los que son llamados “centro”, “pendiente” y “curvatura”. Dentro de este subsistema, se puede apreciar la siguiente estructura de bloques.

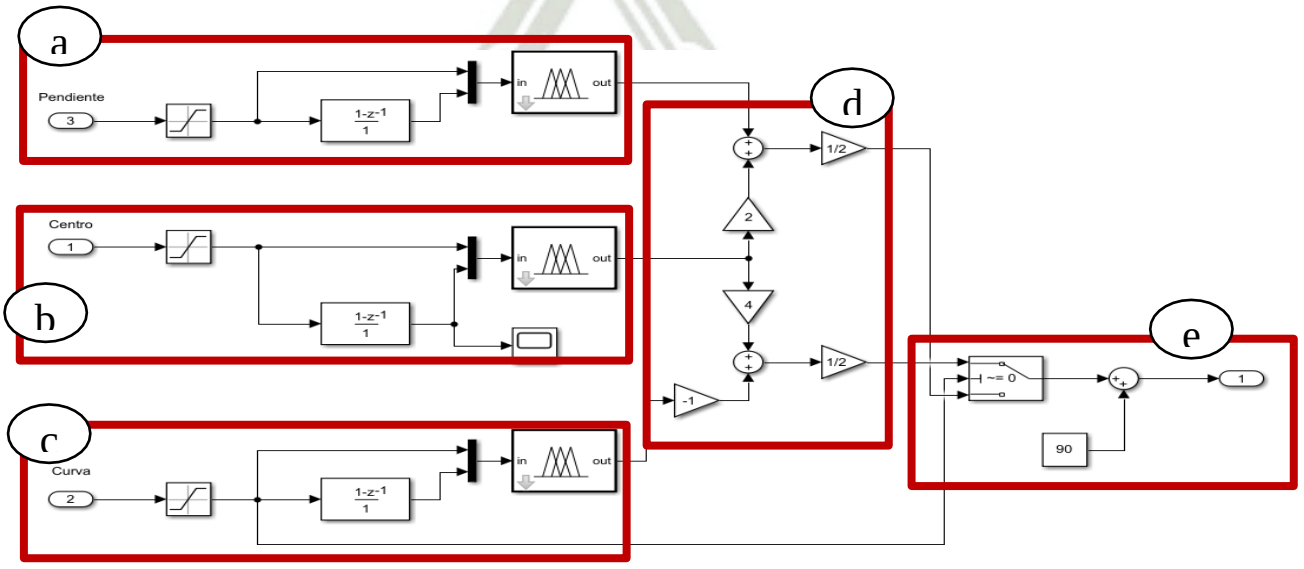


Figura IV.2 Bloque de control del sistema
Fuente: Elaboración propia

Donde las secciones que componen este subsistema son las siguientes:

a) Control PD difuso de Pendiente.

b) Control PD difuso de Centro.

c) Control PD difuso de Curva.

d) Adaptación de ángulo de salida.

e) Conmutador.

Antes de analizar cada parte del subsistema es necesario definir los parámetros con que trabajamos.

Centro fue la diferencia que existe entre la mitad de la imagen y la mitad entre las líneas de carril, esto indica que el modelo de vehículo se encuentra apegado a un lado u otro de la vía, es decir, indica si el auto está descentrado.

Pendiente fue un parámetro exclusivo cuando la vía es recta, este factor se encarga de indicar el nivel de paralelismo que tiene el vehículo respecto de las líneas de carril, ya que puede darse el caso en que el vehículo esté aparentemente centrado, pero no esté paralelo respecto a las líneas de carril, por lo que se deben de remediar estas situaciones para evitar mayores inconvenientes.

Curvatura fue un parámetro exclusivo cuando la vía tiene curvas, indicará cuando una curva es más abierta o más cerrada, por lo que dependiendo de estos valores se debieron de tomar diferentes decisiones con respecto al ángulo de giro de las llantas en el vehículo.

Estos 3 parámetros se complementan entre sí con diferentes pesos en importancia.

a) Control PD difuso de Pendiente

Entrada de Pendiente

Se vio por conveniente determinar un rango de [-0.5 0.5] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Medneg”, “Zero” y “Medpos”. Los parámetros y tipo de función se ven en la tabla 4.1.

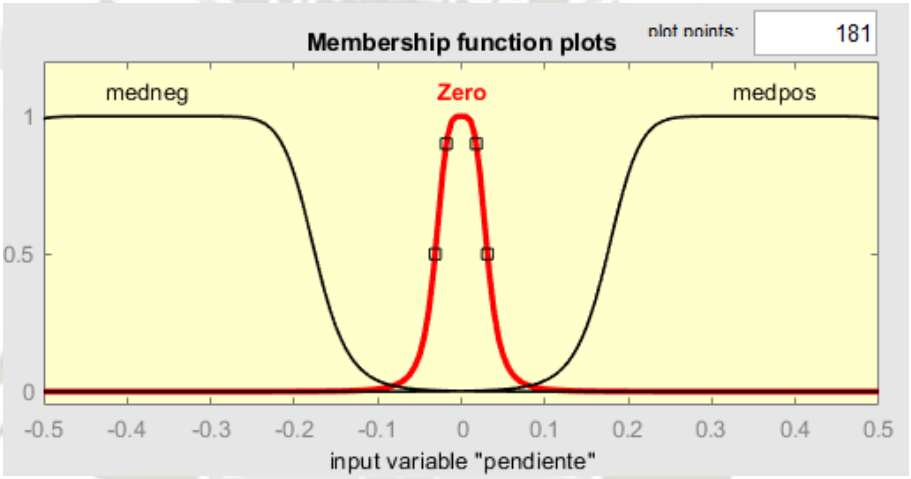


Figura IV.3 Función de membresía de la entrada "Pendiente"
Fuente: Elaboración propia

Tabla 4.1 Valores de las funciones de membresía de la entrada "Pendiente"
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
MEDNEG	gbellmf	[0.2 5 -0.375]
ZERO	gbellmf	[0.03125 2 0]
MEDPOS	gbellmf	[0.2 5 0.375]

Entrada deriv_pend

Se vio por conveniente determinar un rango de [-1.5 1.5] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Neg”, “Cero” y “Pos”.

Los parámetros y tipo de función se ven en la tabla 4.2.

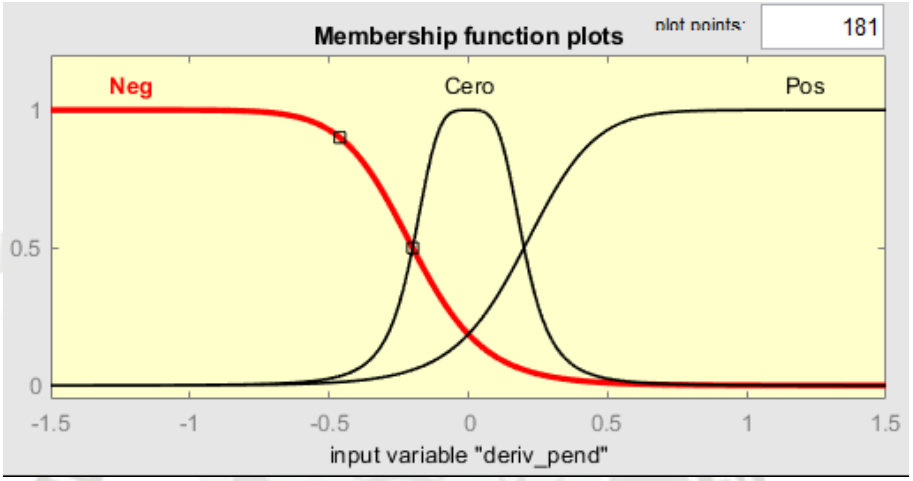


Figura IV.4 Función de membresía de la entrada "Derivada Pendiente"
Fuente: Elaboración propia

Tabla 4.2 Valores de las funciones de membresía de la entrada "Deriv_pend"
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
NEG	gbellmf	[1.8 7 -2]
CERO	gbellmf	[0.2 2 0]
POS	gbellmf	[1.8 7 2]

Salida

Se vio por conveniente determinar un rango de [-50 50] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Medizq”, “Queda” y “Medder”. Los parámetros y tipo de función se ven en la tabla 4.3.

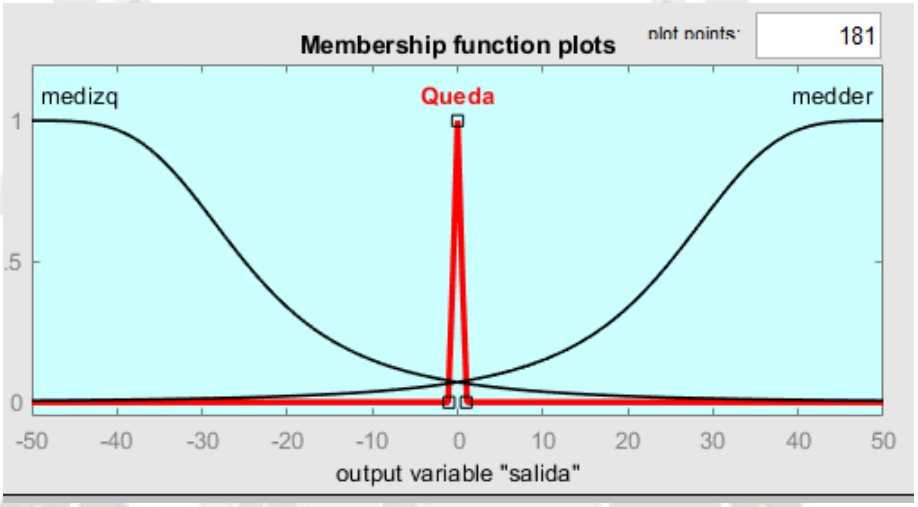


Figura IV.5 Función de membresía de la salida " Pendiente"
Fuente: Elaboración propia

Tabla 4.3 Valores de las funciones de membresía de la salida
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
MEDIZQ	gbellmf	[25 1.85 -50]
QUEDA	trimf	[-1 0 1]
MEDDER	gbellmf	[25 1.85 50]

Reglas de pertenencia

Fue de gran importancia determinar las reglas que determinarán la correlación entre las funciones de pertenencia de las entradas y de la salida, de modo que se tenga un buen funcionamiento del sistema a controlar, por esto se vio por conveniente establecer las reglas de la forma que se observa en la tabla 4.4.

Tabla 4.4 Reglas de pertenencia para las funciones de entrada y salida
Fuente: Elaboración propia

	NEG	**CERO**	**POS**
MEDNEG	Medizq	Queda	Queda
ZERO	Queda	Queda	Queda
MEDPOS	Queda	Queda	Medder

Con los valores de los parámetros de cada función de pertenencia de la entrada y salida, y viendo las reglas de pertenecía tenemos el siguiente gráfico que determina cada valor de salida de acuerdo con los parámetros de entrada que se tengan, en la Figura 4.6 que se ve a continuación:

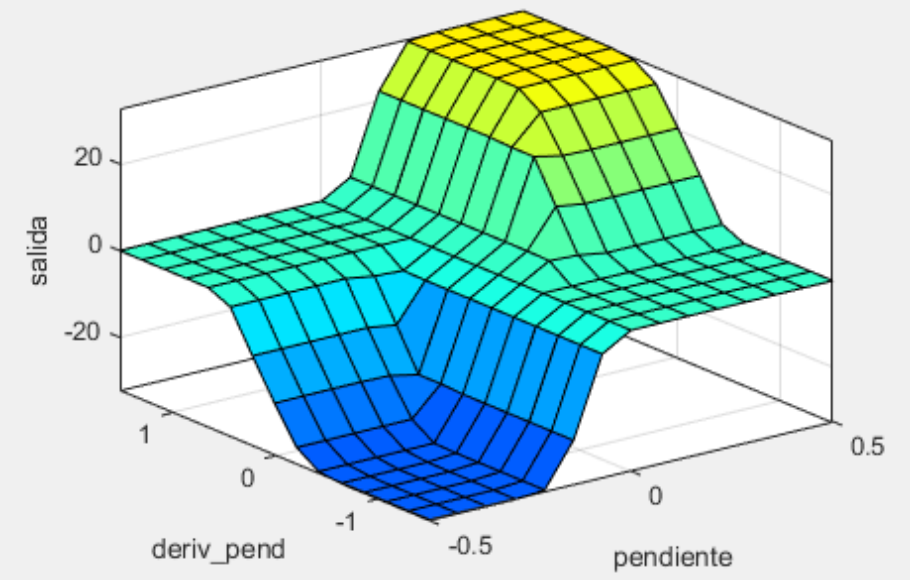


Figura IV.6 Superficie del comportamiento del controlador para la variable "Pendiente"
Fuente: Elaboración propia

Se buscó tener un control suave cuando el sistema alcanza el valor requerido, y tener un control más estricto cuando el sistema se aleja del valor requerido, esto explica que en el gráfico se formen algunas pendientes más empinadas en algunas zonas y haya otras más suaves en las zonas cerca a “0”.

b) Control PD difuso del Centro

Entrada de Centro

Se vio por conveniente determinar un rango de [-150 150] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Neg”, “Medneg”, “Zero”, “Medpos” y “Pos”.

Los parámetros y tipo de función se ven en la tabla 4.5.

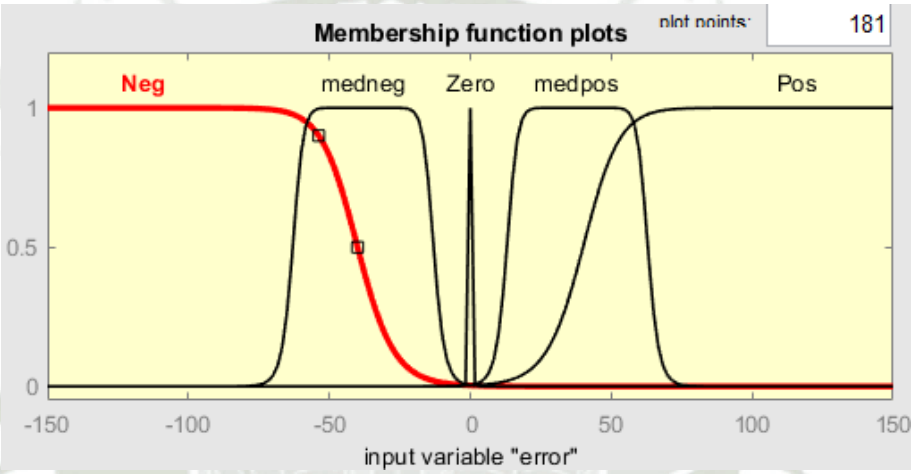


Figura IV.7 Función de membresía de la entrada "Centro"
Fuente: Elaboración propia

Tabla 4.5 Valores de las funciones de membresía de la entrada "Centro"
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
NEG	gbellmf	[195 15 -235]
MEDNEG	gbellmf	[25 6.5 -38]
ZERO	gbellmf	[1 5 0]
MEDPOS	gbellmf	[25 6.5 38]
POS	gbellmf	[195 15 235]

Entrada deriv_error

Se vio por conveniente determinar un rango de [-1.5 1.5] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Neg”, “Cero” y “Pos”.

Los parámetros y tipo de función se ven en la tabla 4.6.

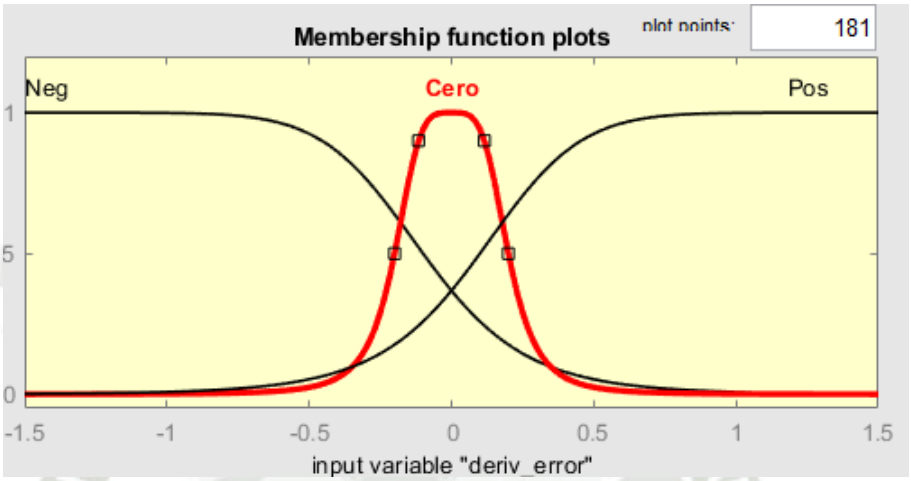


Figura IV.8 Función de membresía de la entrada "Derivada de centro"
Fuente: Elaboración propia

Tabla 4.6 Valores de las funciones de membresía de la entrada "Deriv_centro"
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
NEG	gbellmf	[1.8 5 -1.9]
CERO	gbellmf	[0.2 2 0]
POS	gbellmf	[1.8 5 1.9]

Salida

Se vio por conveniente determinar un rango de [-40 40] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Izquierda”, “Medizq”, “Queda”, “Medder” y “Derecha”.

Los parámetros y tipo de función se ven en la tabla 4.6.

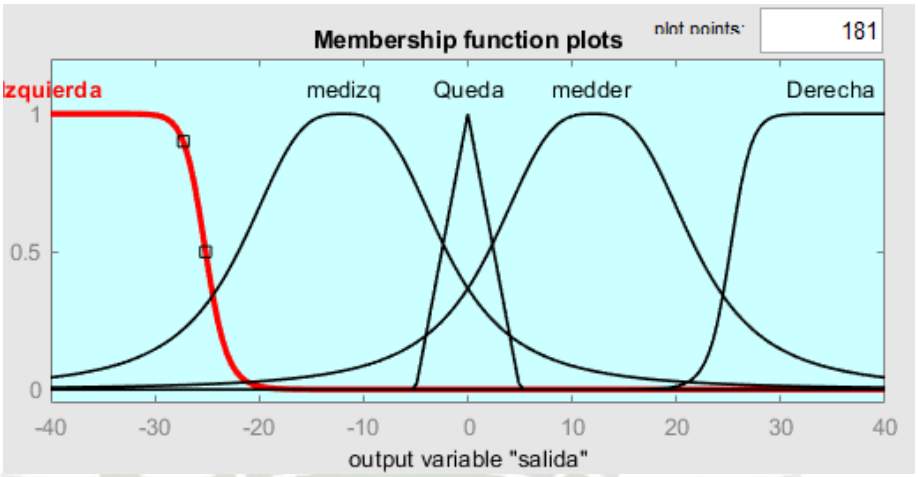


Figura IV.9 Función de membresía de la salida "Centro
Fuente: Elaboración propia

Tabla 4.7 Valores de las funciones de membresía de la salida "Centro"
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
IZQUIERDA	gbellmf	[20.57 10 -45.71]
MEDIZQ	gbellmf	[10 1.5 -12]
QUEDA	trimf	[-5 0 5]
MEDDER	gbellmf	[10 1.5 12]
DERECHA	gbellmf	[20.57 10 45.71]

Reglas de pertenencia

Fue de gran importancia determinar las reglas que determinarán la correlación entre las funciones de pertenencia de las entradas y de la salida, de modo que se tenga un buen funcionamiento del sistema a controlar, por esto se vio por conveniente establecer las reglas de la forma que se observa en la tabla 4.8.

Tabla 4.8 Valores de las funciones de membresía de la entrada "Derivada de pendiente"
Fuente: Elaboración propia

	NEG	**CERO**	**POS**
NEG	Izquierda	Izquierda	Medizq
MEDNEG	Medizq	Queda	Queda
CERO	Queda	Queda	Queda
MEDPOS	Queda	Queda	Medder
POS	Medder	Derecha	Derecha

Con los valores de los parámetros de cada función de pertenencia de la entrada y salida, y viendo las reglas de pertenecía tenemos el siguiente gráfico que determina cada valor de salida de acuerdo con los parámetros de entrada que se tengan, en la Figura 4.6 que se ve a continuación:

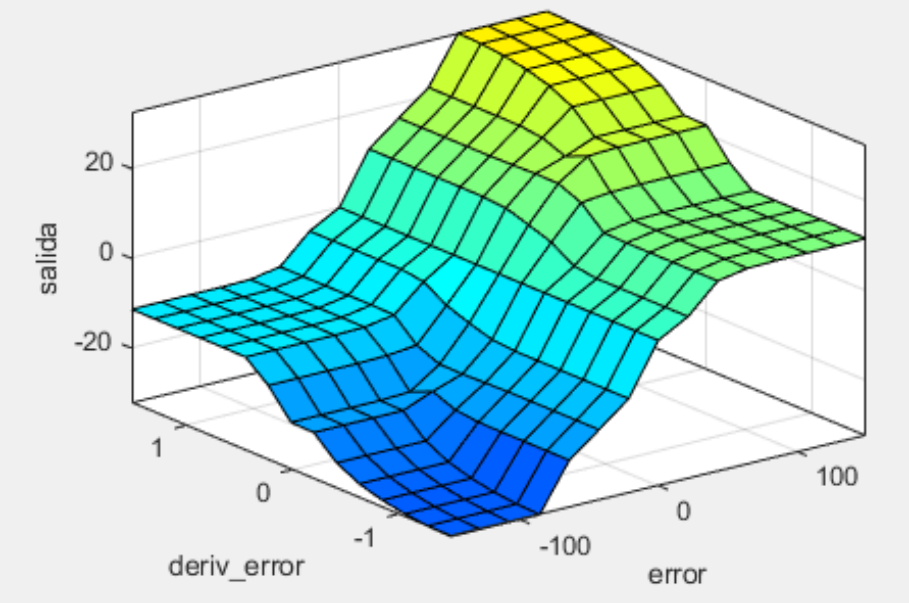


Figura IV.10 Superficie del comportamiento del controlador para la variable "Centro"
Fuente: Elaboración propia

Se buscó tener un control suave cuando el sistema alcanza el valor requerido, y tener un control más estricto cuando el sistema se aleja del valor requerido, esto explica que en el gráfico se formen algunas pendientes más empinadas en algunas zonas y haya otras más suaves en las zonas cerca a “0”.

c) Control PD difuso de la curva

Entrada de curva

Se vio por conveniente determinar un rango de [-0.0005 0.0005] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Neg”, “Medneg”, “Zero”, “Medpos” y “Pos”.

Los parámetros y tipo de función se ven en la tabla 4.9.

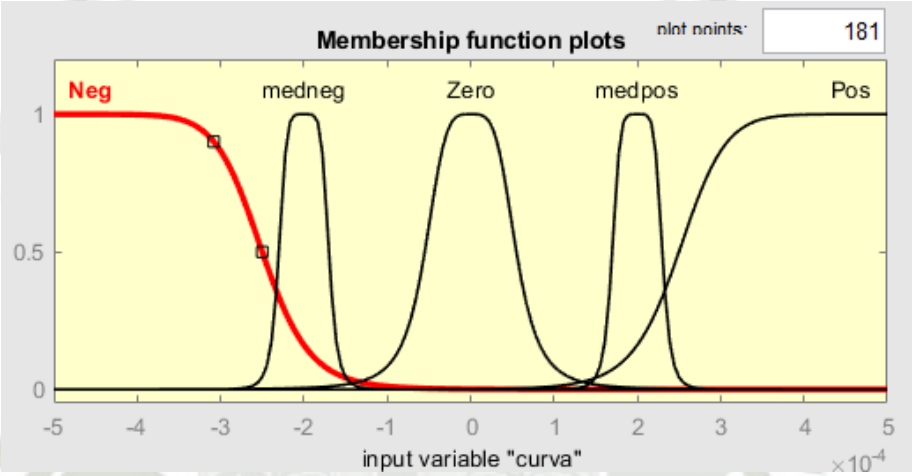


Figura IV.11 Función de membresía de la entrada "Curva"
Fuente: Elaboración propia

Tabla 4.9 Valores de las funciones de membresía de la entrada "Curva"
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
NEG	gbellmf	[0.0004 7 -0.00065]
MEDNEG	gbellmf	[3e-05 3 -0.0002]
ZERO	gbellmf	[5.497e-05 2 0]
MEDPOS	gbellmf	[3e-05 3 0.0002]
POS	gbellmf	[0.0004 7 0.00065]

Entrada deriv_curva

Se vio por conveniente determinar un rango de [-0.0001 0.0001] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Neg”, “Cero” y “Pos”.

Los parámetros y tipo de función se ven en la tabla 4.10.

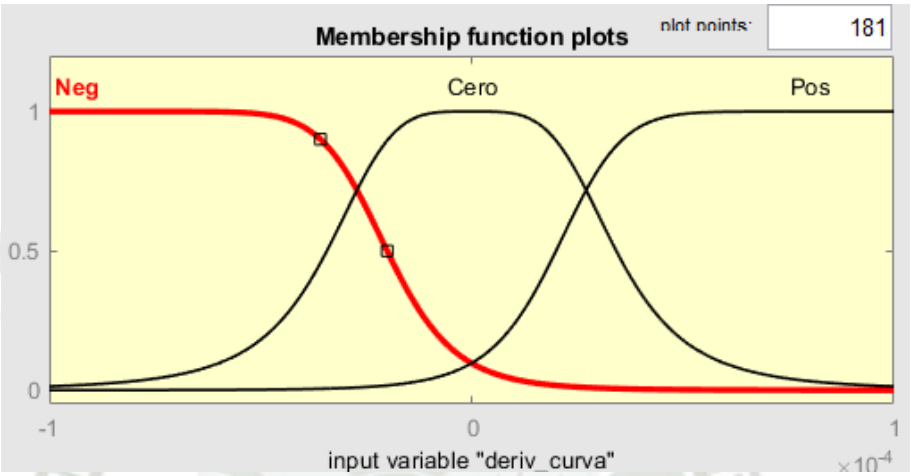


Figura IV.12 Función de membresía de la entrada "Derivada curva"
Fuente: Elaboración propia

Tabla 4.10 Valores de las funciones de membresía de la entrada "Derivada curva"
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
NEG	gbellmf	[8e-05 5 -0.0001]
CERO	gbellmf	[3.425e-05 2 0]
POS	gbellmf	[8e-05 5 0.0001]

Salida

Se vio por conveniente determinar un rango de [-80 80] entre el cual estarán determinadas las funciones de pertenencia, además de los datos propios de las mismas funciones de pertenencia denominadas “Izquierda”, “Medizq”, “Queda”, “Medder” y “Derecha”.

Los parámetros y tipo de función se ven en la tabla 4.11.

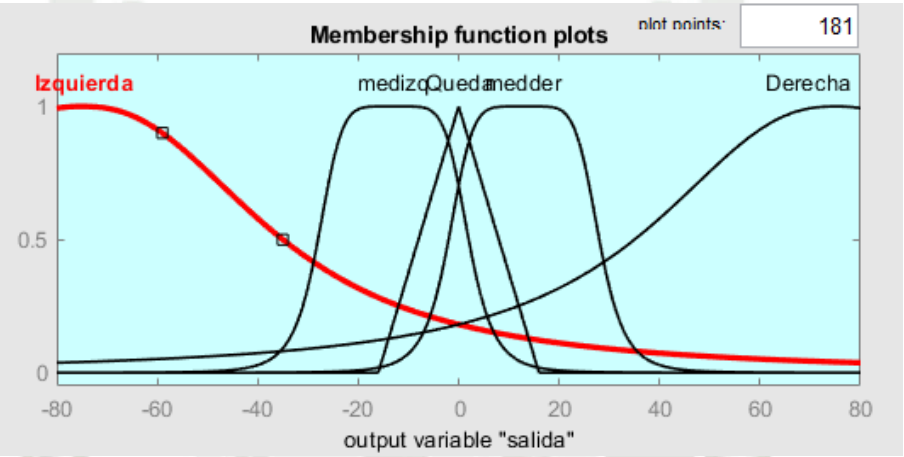


Figura IV.13 Función de membresía de la salida "Curva"
Fuente: Elaboración propia

Tabla 4.11 Valores de las funciones de membresía de la salida "Curva"
Fuente: Elaboración propia

NOMBRE	FUNCIÓN	PARÁMETROS
BAJAR	gbellmf	[65 10 -160]
MEDBAJS	gbellmf	[30 3 -72]
QUEDA	trimf	[-13 0 13]
MEDSUB	gbellmf	[30 3 72]
SUBIR	gbellmf	[65 10 160]

Reglas de pertenencia

Fue de gran importancia determinar las reglas que determinarán la correlación entre las funciones de pertenencia de las entradas y de la salida, de modo que se tenga un buen funcionamiento del sistema a controlar, por esto se vio por conveniente establecer las reglas de la forma que se observa en la tabla 4.12.

Tabla 4.12 Valores de las funciones de membresía de la entrada "Pendiente"
Fuente: Elaboración propia

	NEG	**CERO**	**POS**
NEG	Izquierda	Izquierda	Medizq
MEDNEG	Medizq	Queda	Queda
CERO	Queda	Queda	Queda
MEDPOS	Queda	Queda	Medder
POS	Medder	Derecha	Derecha

Con los valores de los parámetros de cada función de pertenencia de la entrada y salida, y viendo las reglas de pertenecía tenemos el siguiente gráfico que determina cada valor de salida de acuerdo con los parámetros de entrada que se tengan, en la Figura 4.6 que se ve a continuación:

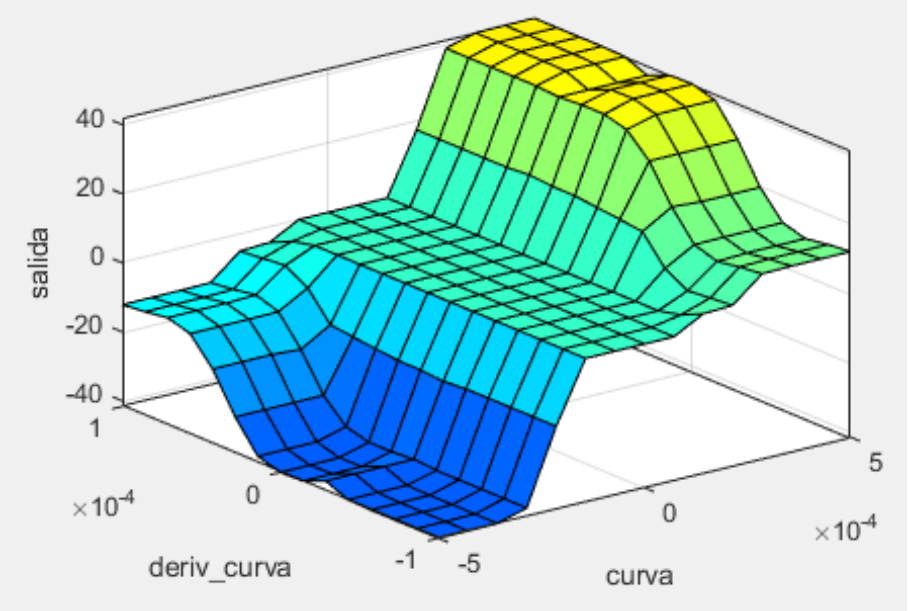


Figura IV.14 Superficie del comportamiento del controlador para la variable "Curva"
Fuente: Elaboración propia

Se buscó tener un control suave cuando el sistema alcanza el valor requerido, y tener un control más estricto cuando el sistema se aleja del valor requerido, esto explica que en el gráfico se formen algunas pendientes más empinadas en algunas zonas y haya otras más suaves en las zonas cerca a “0”.

d) Adaptación del ángulo de salida

Esta sección del subsistema se encargó de realizar un promedio de los ángulos enviados por los controladores, a pesar de que aparentemente los 2 controladores tienen los mismos pesos, las señales emitidas por el controlador del "Centro" tienen un rango menor de ángulo que las de los otros controladores, esto se pudo observar en la descripción de las salidas del controlador.

La decisión de darle internamente un mayor peso al controlador de "Centro" se tomó debido a que los controladores particulares "Pendiente" y "Curva" trabajan de forma drástica cada uno cuando ocurren pequeños cambios dentro de la vía, sin embargo, es necesario mantener el vehículo dentro de la vía, por lo que se le asigna un mayor peso a este valor. Así mismo se obtuvo mejores resultados asignándosele mayor peso a este controlador.

e) Conmutador

Se encargó de detectar si la señal de curva es cero o tienen un valor diferente en caso sea cero, significará que la función de regresión es lineal, por lo que deja pasar el ángulo calculado por los controladores de "Pendiente" y "Centro" sin embargo, si la señal que viene del controlador "Curva" es diferente de cero, significa que la regresión de la función es cuadrática, por lo que esta vez solo se tomará en cuenta el valor de los ángulos dado por los controladores "Curva" y "Centro".

B) BLOQUES DE CONTROL DEL SERVOMOTOR

Se colocan los siguientes bloques que ayudarán en la comunicación del Matlab y el Arduino, para tomar a la tarjeta de control como esclavo para controlar el servomotor de acuerdo con los datos obenidos del procesamiento.

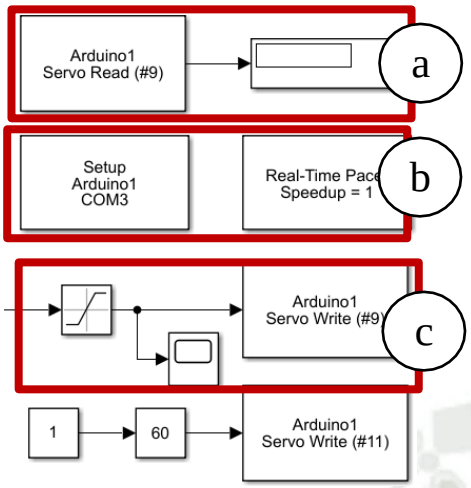


Figura IV.15 Bloque de control del servomotor
Fuente: Elaboración propia

a) Inicialización de los bloques

Fue necesario establecer estos bloques para que inicialicen los procesos en el Simulink, y a su vez determinen el tiempo de interacción entre el Simulink y el Arduino, el código de acondicionamiento del Arduino se encuentra en los anexos C.

b) Saturación y salida al Arduino

Del controlador ya sale el ángulo necesario para poder controlar de manera correcta el modelo de vehículo, en esta parte únicamente se limitó la salida de modo que no se excedan los límites físicos de movimiento que permite el Arduino y la juntura con las llantas, a pesar de todo, se ha acondicionado esta juntura para darle más libertad al movimiento, los límites de movimiento para el Arduino están entre 40° y 140°, finalmente esta señal debe de rastrearse con un retenedor de ceros (ZOH) para que no hallan problemas con los tiempos de muestreo entre los bloques de escritura en el Arduino y las señales que se recibe.

c) Visualización del ángulo del servo

Finalmente, en este bloque se visualizó el ángulo que debería estar girado el servomotor en el vehículo, este bloque es únicamente de control, no cumple ninguna función en específico, por lo que podría ser eliminado y no ocurrirían cambios significativos en el programa.

C) BLOQUE DE VISIÓN

En esta sección se trabajó junto con el toolbox “Image acquisition” de Matlab y una aplicación externa capaz de generar una dirección IP mediante la cual transmite la señal de video desde un celular para que pueda ser recibida por la computadora, se explica con mayor detalle en el Anexo A.2 (Adquisición de video desde una cámara).

En esta sección del Simulink, utilizamos el bloque “From video device” para poder enviar la señal de video del modelo de carretera al código de procesamiento y se pueda controlar el modelo de vehículo.

D) BLOQUE PROCESAMIENTO

Antes de realizar la identificación de líneas de carril se debe hacer un preprocesamiento de las imágenes, utilizando transformadas de globales y geométricas, así lo detallan diversos autores como: Correa Sandoval & Díaz Zapata, (2019); Romero-Macas, H. (2020) y Gonzáles Marcos, A., et al. (2006)

Después, de acuerdo con Gonzáles Marcos, A., et al. (2006) y García, I.; Herrera, D. & Mina, Jorge (2017), y según los métodos de línea que hemos estudiado, se consideró que el método de regresión era más efectivo que los otrs métodos. Nos basamos en esta decisión considerando que la Transformada de Hough de acuerdo con Tosini & Leiva, (2018) no era suficiente para el estudio de líneas curvas, por lo que aplicaba una variante del mismo, así mismo, el autor puntúa que a la salida del método se tenían datos de difícil procesamiento.

De la misma forma, Ibarra Arenado et al., (2015) experimenta con el método de punto de fuga, el cual necesita hacer varias iteraciones para poder resolverse sobre líneas rectas, representando una mayor carga computacional.

Finalmente, García, I.; Herrera, D. & Mina, Jorge (2017) se decantaron por el método de Transformada Hough, el cual fue brindaba valores fáciles de comprender y manejar. De la misma manera Aral Sarrafi (2021) estableció un esquema en Simulink para aplicaciones de mantenimiento de carril en simulaciones. El autor se valió del método de Transformada de Hough. Por lo que se utilizará dicho método como parte de la identificación de líneas de carril.

Aquí se definieron las operaciones necesarias para hacer el control de la dirección del vehículo, como se trabajó en simulación, es necesario determinar otras variables adicionales a solo la detección de las líneas, como también son la orientación y avance del vehículo, esto se explicará a continuación:

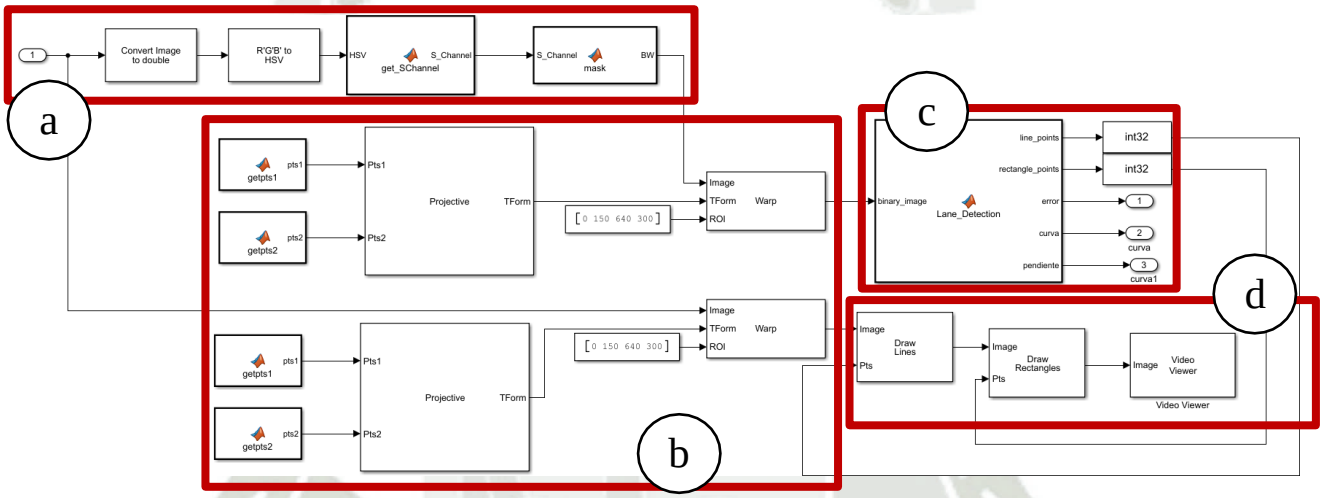


Figura IV.16 Bloque de procesamiento del sistema
Fuente: Elaboración propia

a) Transformaciones globales

En esta primera parte, se trabajaron transformaciones de color de la imagen, al trabajarse cada píxel independientemente se tratan de transformaciones globales.

Se inicia recibiendo la señal de video de la sección anterior, se la convirtió en una matriz con variables “double”, se convirtieron los colores de R’G’B’a HSV, se filtraron los colores de acuerdo con su “Valor” (Medida de blancoo negro), y posteriormente se le aplicó un filtro para binarizar la imagen con un umbral del 65%.

Sin embargo, para el correcto procesamiento de la imagen, se le deben de modificar parámetros geométricos.

Un ejemplo de las transformaciones que se realizan en esta sección es la siguiente:

Figura IV.17 Ejemplo del funcionamiento de las transformaciones globales del sistema
Fuente: Elaboración propia

b) Transformaciones geométricas

Se obtuvieron nuevas imágenes a partir de una imagen fuente, esta es la última modificación que se le hizo a la imagen antes de ser procesada, se consiguió una vista aérea o vista a vuelo de pájaro (o vista cenital, esta se explica en la parte b.2 de esta sección) a partir de una vista en picado, por lo que se elaboró el siguiente sistema:

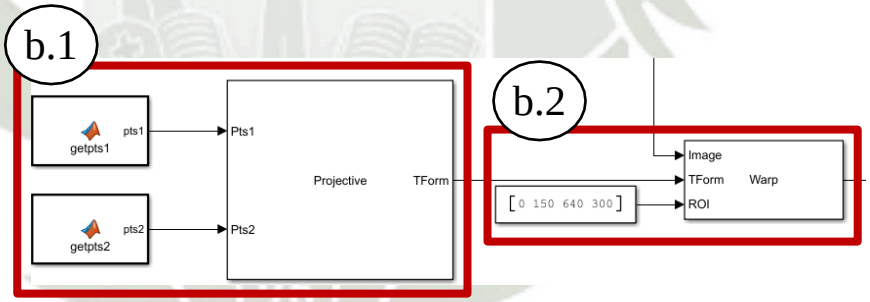


Figura IV.18 Bloque de las transformaciones geométricas
Fuente: Elaboración propia

Cabe aclarar que este sistema se encuentra duplicado, una de las transformaciones entrará al código de procesamiento y la otra transformación servirá como base sobre la cual se graficarán las salidas del procesamiento de la imagen, a continuación, se explican las partes de la sección de transformaciones geométricas.

b.1 Matriz de transformación

Esta sección se encarga de ingresar 2 matrices de coordenadas dentro del bloque “Projective”, de modo que como salida se obtuviera la matriz de transformación que permita pasar de la primera matriz de coordenadas a la segunda matriz de coordenadas.

Los valores que se usaron en las entradas de esta sección y sus representaciones son las siguientes:

```
pts1 = [220 120
        420 120
        0 460
        640 460];
```

```
pts2 = [170 0
        470 0
        170 480
        470 480];
```

Figura IV.19 Máscaras de las matrices de transformación
Fuente: Elaboración propia

Significa que la salida del bloque fue la matriz que transforme la primera figura en la segunda figura

b.2 Deformación: Vista a vuelo de pájaro

En esta sección se tienen 3 entradas, la primera es la señal de video, la cual será afectada de acuerdo con la matriz de transformación obtenida del bloque anterior y que es la segunda entrada, y finalmente una tercera entrada que son valores de recorte para la señal de video, ayuda a poder ser más preciso en la señal que se ingresa al código de procesamiento.

De acuerdo con la matriz de transformación definida anteriormente, tendríamos el siguiente ejemplo de lo que sucede en esta sección:

Figura IV.20 Ejemplo del funcionamiento del bloque de deformación
Fuente: Elaboración propia

Observamos que de la imagen de entrada se obtienen las 2 imágenes de la derecha, debido a que la imagen a blanco y negro entra al código de procesamiento y la imagen en color sirve de base para graficar las líneas de dirección.

La acción que se realiza de deformar la imagen de tal modo que parezca que las líneas de carril no tienen perspectiva, es decir, como si se las viera desde la parte superior, se denomina vista a vuelo de pájaro o vista aérea.

c) Procesamiento de la imagen

Aquí se definió la transformada de Hough para identificación de líneas y curvas, de modo que se puedan obtener las salidas necesarias, el código se encuentra explicado en el anexo B, sin embargo, un ejemplo gráfico de lo que sucede en esta sección se encuentra representado en la Figura 4.21.

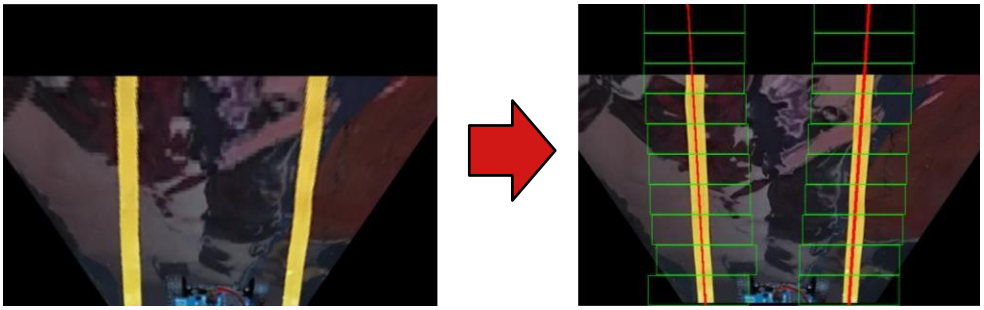


Figura IV.21 Ejemplo del bloque de procesamiento de imagen
Fuente: Elaboración propia

Se puede observar que las líneas de carril se encuentran demarcadas, así como también se encuentran enmarcadas con un marco color verde, estas gráficas corresponden a las salidas que arroja este código, específicamente las salidas "Line_points" y "Rectangle_points", las otras 3 salidas corresponden a datos necesarios para poder mantener el vehículo dentro de las líneas de carril, estas salidas serán estudiadas con detenimiento en la siguiente sección

d) Salidas

Finalmente, los datos de salida serán "Line_points", "Rectangle_points", "Centro", "Curva", "Pendiente".

Line_points en un vector fila que contiene las coordenadas de cada punto obtenido de la regresión lineal o cuadrática (si el valor del coeficiente de mayor grado cumple con las restricciones, la ecuación será cuadrática, de lo contrario será una ecuación lineal) que se ajusta a las líneas de carril tanto de la derecha, como al carril de la izquierda.

Rectangle_points contiene las coordenadas de cada vértice de los rectángulos que demarcan las líneas de carril tanto del lado derecho como del izquierdo.

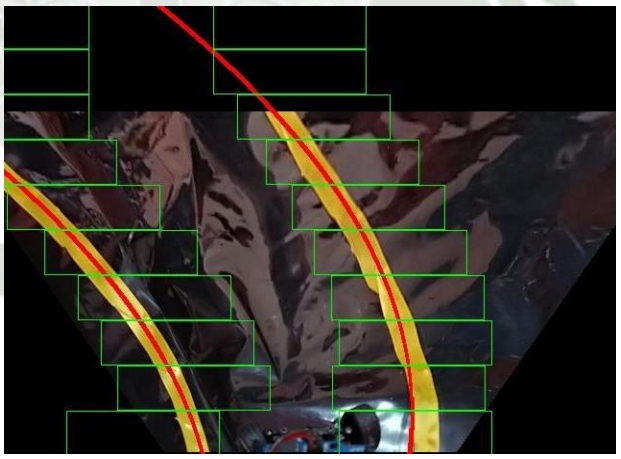


Figura IV.22 Representación de las salidas "Line_points" y "Rectangle_points"
Fuente: Elaboración propia

Centro determina el "grado de descentramiento" del centro vehículo con respecto a las líneas de carril. Como se ve en la figura inferior, el carro se encuentra apegado al carril izquierdo, por lo que está descentrado y envía una señal de error de centro alta, lo que provoca que el controlador también envíe una corrección angular grande.

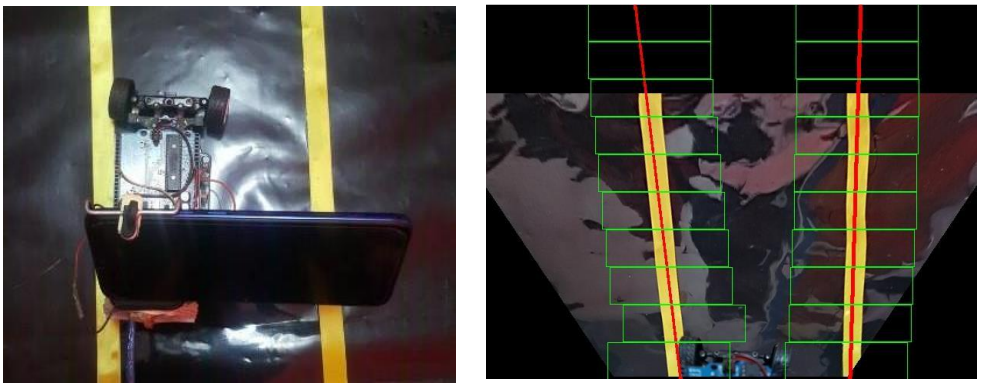


Figura IV.23 Representación de la salida "Centro"
Fuente: Elaboración propia

Curva determina el "grado de curvatura" cuando se determina que la función de regresión es cuadrática, este valor de curvatura es determinado por el valor del coeficiente de segundo grado, entonces este parámetro determinará que la curva es suave o más cerrada.

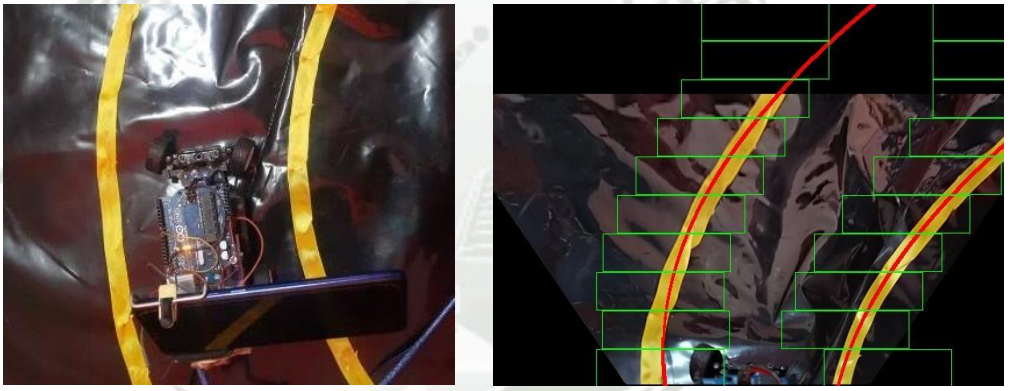


Figura IV.24 Representación de la salida "Curva"
Fuente: Elaboración propia

Pendiente determina el "grado de paralelismo" del vehículo con respecto a las líneas de carril, se diferencia del primer parámetro porque indica si el vehículo se encuentra cruzado respecto a las líneas de carril.

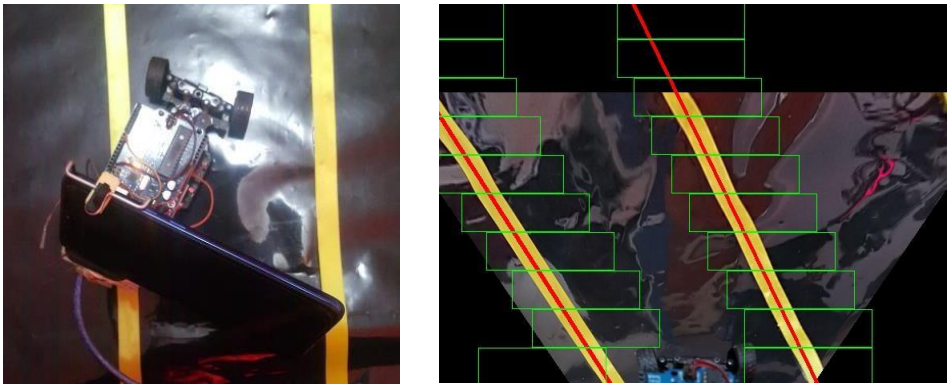


Figura IV.25 Representación de la salida "Pendiente"
Fuente: Elaboración propia

2. ESTRUCTURA MECÁNICA

2.1 Acondicionamiento del carro

Se trabajó sobre la base de un auto a control remoto, al cual se le realizaron modificaciones para poderle acondicionar las nuevas partes que necesita la estructura a fin de sostener el celular desde el cual se enviará la señal de video, así mismo se le cambió la placa de control por el Arduino y se le colocó un servomotor que ayude al movimiento de las llantas, quedando de la siguiente manera:

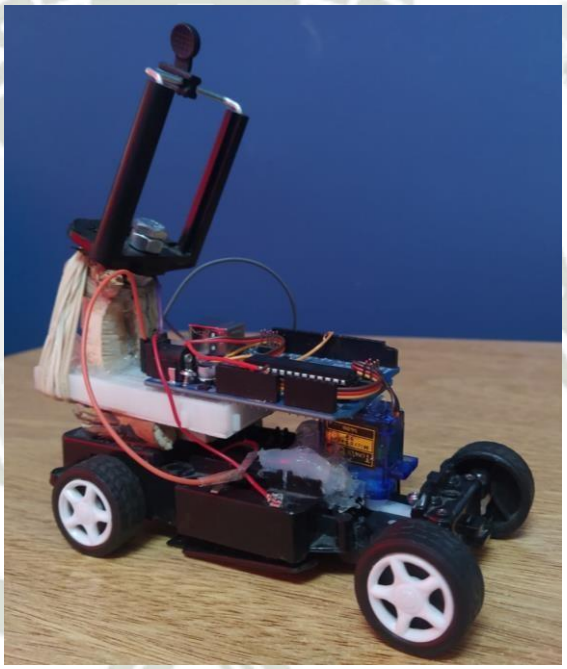


Figura IV.26 *Adaptación del primer modelo de vehículo*
Fuente: Elaboración propia

Debido a que no se necesitaban mayores datos físicos del modelo de vehículo para poder realizar el control PD difuso, para el acondicionamiento del vehículo no se realizaron ningún tipo de cálculos de esfuerzos sobre las piezas, únicamente se realizó una modificación en la juntura de las ruedas, para poderle dar mayor libertad al ángulo de giro de las ruedas del modelo de vehículo.

A continuación, se hará una descripción de las partes que constituyen la estructura mecánica.

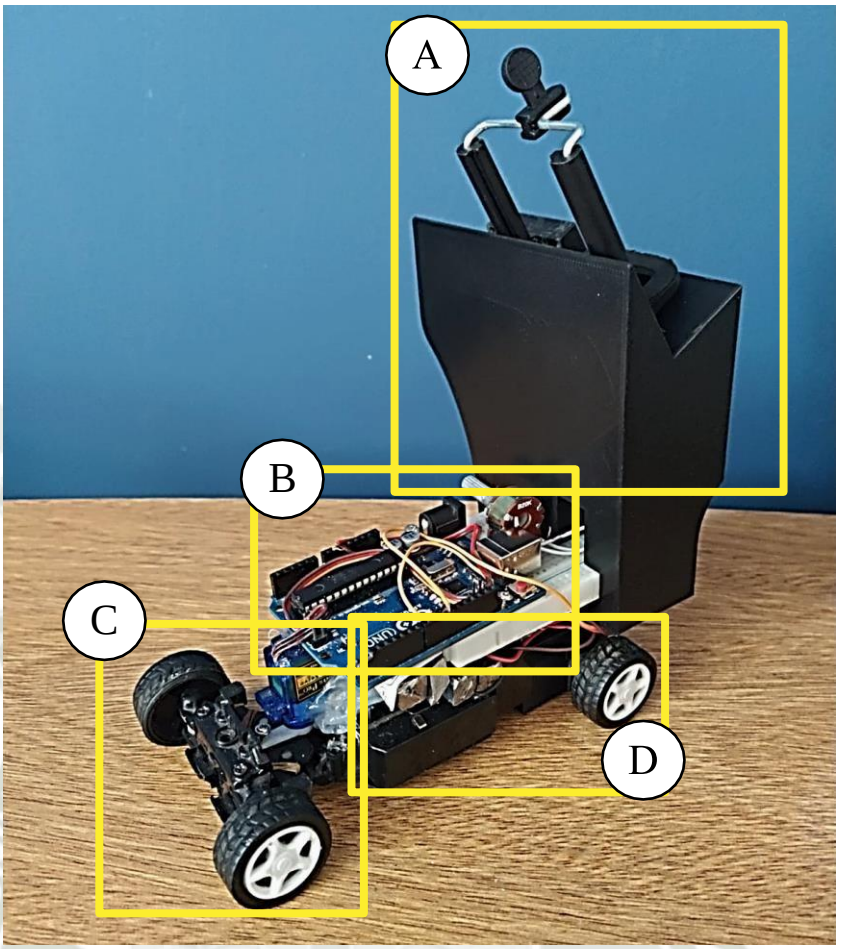


Figura IV.27 *Adaptación final del modelo de vehículo*
Fuente: Elaboración propia

A) SOPORTE DEL CELULAR

Consiste en una estructura fija que se encarga de servir de soporte para el celular, de modo que este permanezca fijo durante todo el recorrido que realice el vehículo, se compone de un sujetador de celulares unido mecánicamente por perno a una estructura echa en impresión 3D, acondicionada para soportar el peso del celular y la torción, este conjunto se encuentra unido mediante silicona sobre la estructura del vehículo en sí y sujeta también al protoboard.

Los planos de la impresión 3D se encuentran En la sección de Planos

B) ARDUINO Y PROTOBOARD

Serán la parte electrónica del modelo de vehículo, el Arduino está configurado como esclavo, para recibir las órdenes que envíe la computadora, y este a su vez transmitirlas al servomotor.

El protoboard cumplirá la función de servir como base para el Arduino, además proporciona puertos útiles para poder realizar conexiones con el motor. Este protoboard se encuentra sujetado también por la estructura que soporta al celular

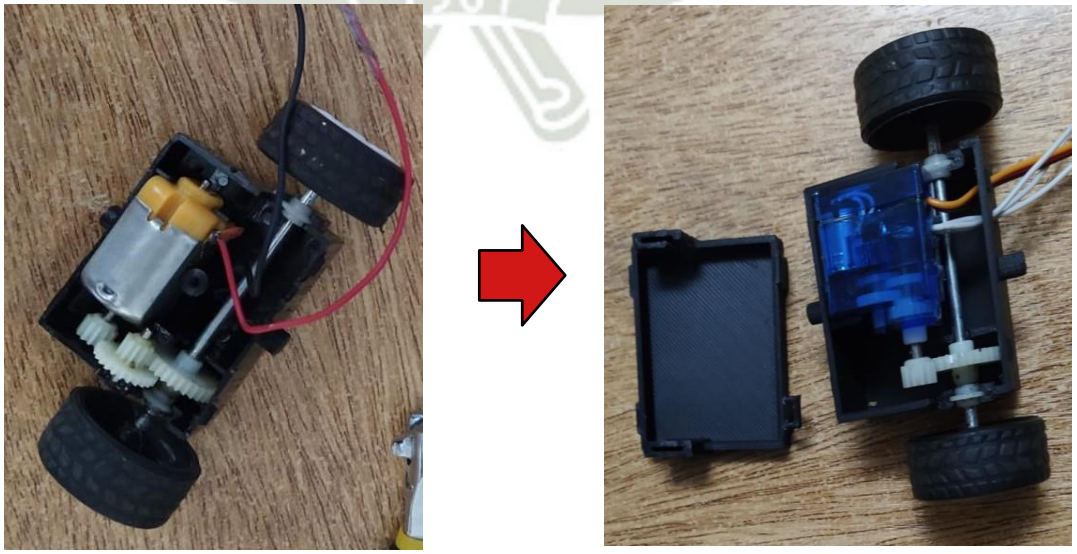
C) SERVOMOTOR Y SISTEMA DE DIRECCIÓN

Una vez la señal llega de la computadora al Arduino, este envía valores angulares al servomotor, que se encargará de mover la juntura de las ruedas, de modo que estas giren proporcionalmente y en el sentido contrario al que lo hace el servomotor. Como se mencionó anteriormente, se diseñó esta juntura para que se pudiera conseguir una mayor libertad de movimiento de las ruedas, el plano de este se encuentra también en la sección de Planos.

D) FUENTE DE PODER Y MOTOR

El auto a control remoto contaba con una porta pilas que formaba parte de la estructura del vehículo, el cual también es utilizado como principal fuente de poder para poder alimentar exclusivamente al motor (servomotor adaptado), ya que el Arduino es alimentadopor cable USB directamente desde el portátil, así mismo el motor que se utiliza es el mismoque viene con el auto a control remoto.

Como se mencionó anteriormente, el vehículo avanza gracias a que se cambió la estructura que soportaba al motor por una estructura diseñada e impresa en 3D y un servomotor adaptado para fungir de motor, de modo que se pudiera controlar la velocidad de giro del servomotor y por consiguiente la velocidad del vehículo



Capítulo V

PRUEBAS Y RESULTADOS

1. PRUEBAS

El procedimiento general para poder poner en marcha el modelo de vehículo fue el siguiente:

Primero se cargó un código IDE "ADIOES" que se encuentra en el Anexo C, que viene con el toolbox "Legacy MATLAB and Simulink support for Arduino" para definir al Arduino como esclavo y pueda leer los valores que envía el Matlab y a su vez enviarlos al servomotor, en la figura 5.1 se observa la compilación del programa en el Arduino, con esto el Arduino quedó listo para trabajar.

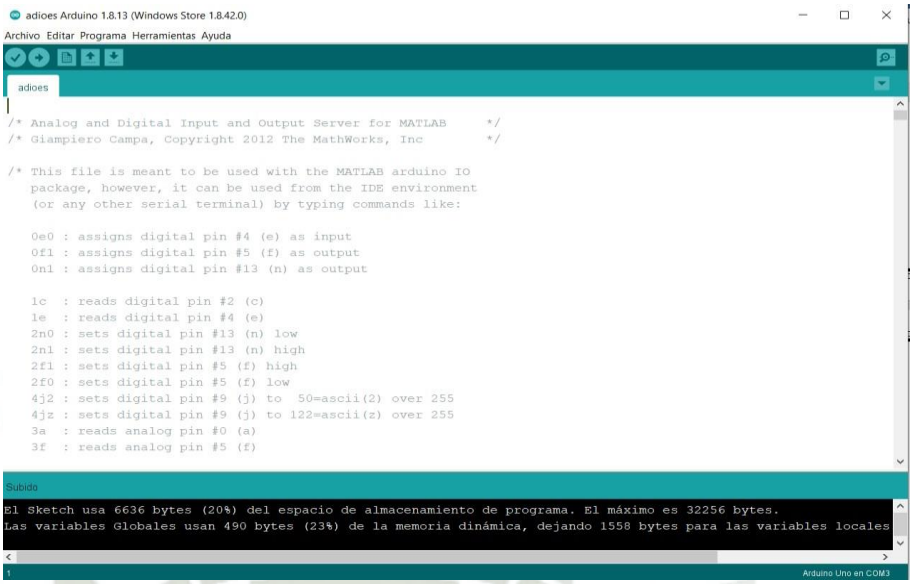


Figura V.1 IDE del código para el Arduino
Fuente: Elaboración propia

Seguidamente se digitó “fuzzy” en el command window de Matlab para abrir el toolbox de control difuso de Matlab, se configuró con los parámetros indicados anteriormente y se hizo clic en “Exportar variable al Workspace”. Se realizó esta acción 3 veces, debido a que utilizamos 3 controladores difusos, la ventana se vio de la siguiente manera.

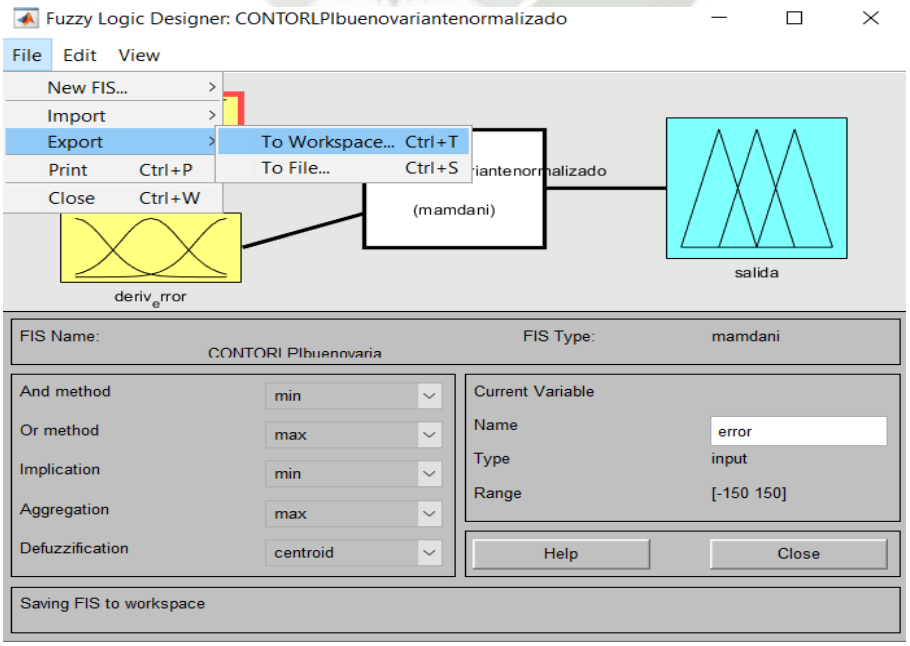


Figura V.2 Ventana FUZZY de Matlab
Fuente: Elaboración propia

Sabremos que el Simulink se encuentra listo para trabajar porque en la parte de control indica que se reconocieron los controladores como se muestra en la siguiente comparación

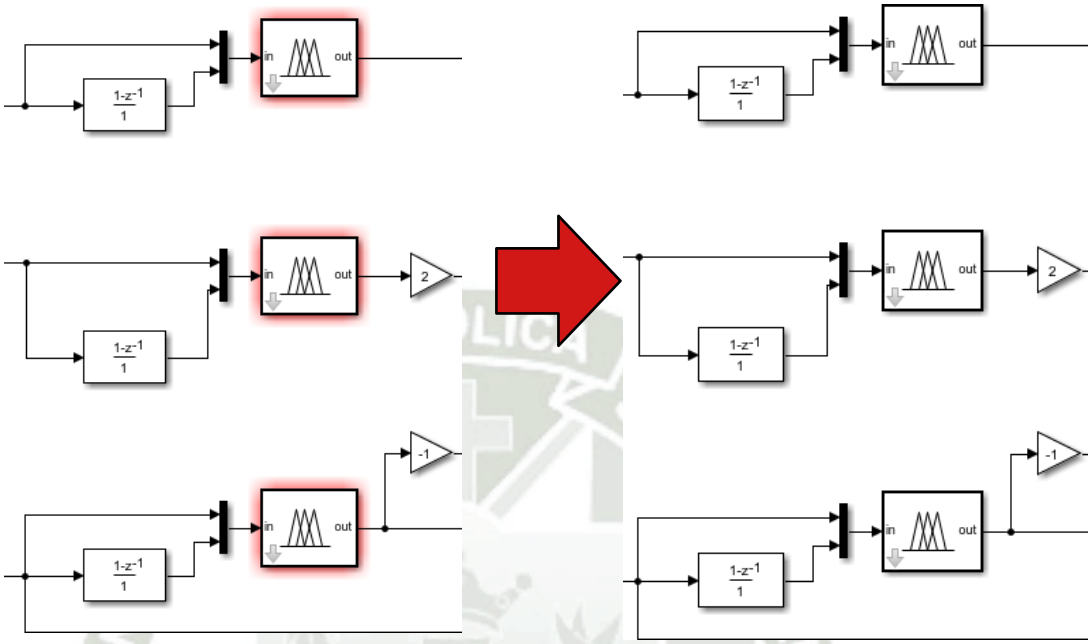


Figura V.3 Ejemplificación de parámetros FUZZY cargados
Fuente: Elaboración propia

Además, fue necesario enlazar la señal de video que lee el celular y que pueda leerlo el computador, esta parte ha sido explicada anteriormente, además se encuentra ampliamente explicada en el Anexo A.2 (Adquisición de video desde una cámara).

Todos estos preparativos fueron necesarios antes de poder ejecutar el Simulink, y deberán de realizarse cada vez que se desee correr el programa, a excepción del primer paso de subir el sketch "ADIOES" al Arduino, ya que esta tarjeta guarda el código incluso después de apagarse.

Las pruebas llevadas a cabo para poder ajustar los controladores difusos, cuyos valores finales fueron definidos en la sección "4.1.4 Descripción del Simulink", fueron realizadas de la siguiente manera:

Era necesario conocer que cada controlador trabajaba correctamente individualmente para luego ajustarlos y que trabajen bien en conjunto, así que se tomó la decisión de trabajar a la par con las salidas "Centro" y "Pendiente", para cuando la

regresión de las líneas de carril fuera lineal y trabajar con las salidas “Centro” y “Curva”, para cuando la regresión lineal de las líneas de carril fuera cuadrática.

Para el primer caso, cuando la regresión es lineal, se tomaron las lecturas de las señales “Centro” y “Pendiente”, como se había mencionado anteriormente, y a sus correspondientes ángulos arrojados por el controlador a través de sendos scopes, obteniéndose la siguiente imagen

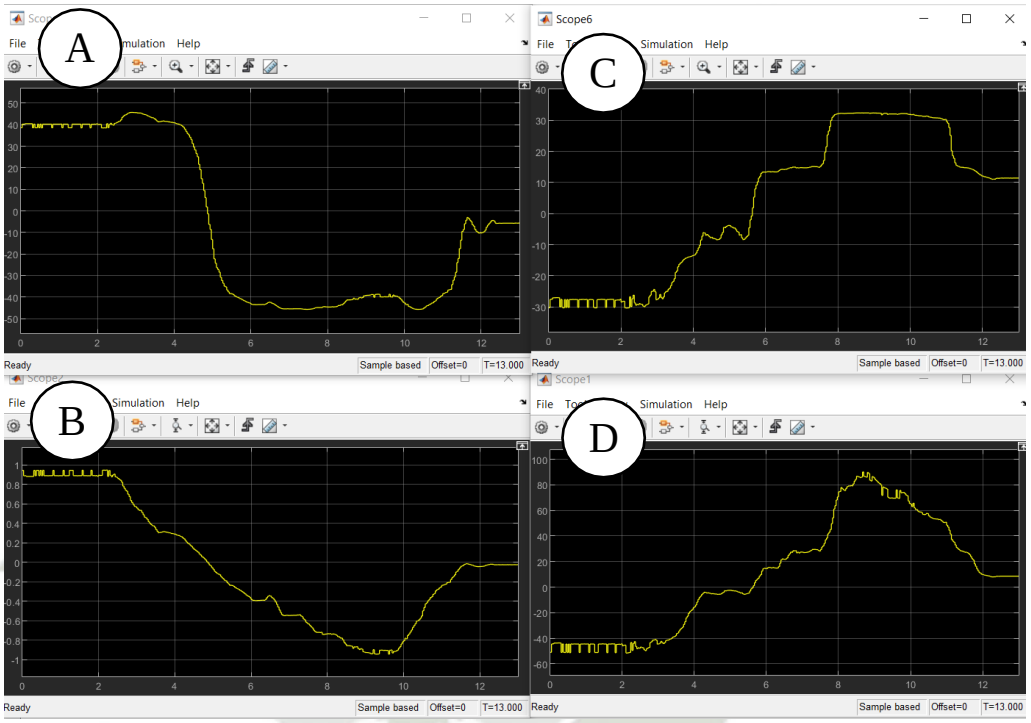


Figura V.4 Gráficos de valores "Pendiente" y "Centro"
Fuente: Elaboración propia

Donde A) es el ángulo que emite el controlador según la señal “pendiente”; B) es la señal “Pendiente”; C) es el ángulo que emite el controlador debido a la señal “Centro” y D) es la señal “Centro”.

Si analizamos la figura A y B detenidamente, notaremos que la señal “pendiente” oscila aproximadamente entre un rango de -1 a 1, lo que se ve reflejado en los parámetros de entrada para el control PD difuso de esta señal, además vemos que cuando la pendiente es positiva, la salida del ángulo es positiva también, esto se explica gráficamente de la siguiente manera:

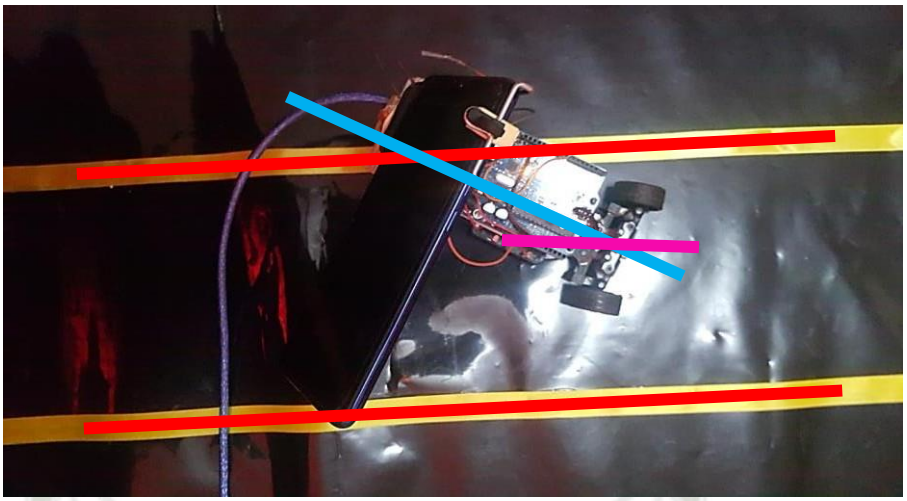


Figura V.5 Giro de las ruedas cuando el carro no está paralelo
Fuente: Elaboración propia

Como se pudo observar, cuando el modelo de vehículo se encuentra poco paralelo con relación a las líneas de carril, el ángulo de las llantas debe de sumarse al ángulo base de 90° para poder corregir de manera correcta la posición del vehículo.

Si se analiza las figuras C y D, notaremos que la señal de error tiene valores más grandes, llegando a valores de casi 100 de error, por lo que se toman estas referencias como entradas de los controladores correspondientes a esta señal. Así mismo se puede observar, que cuando el error es negativo la salida del ángulo es negativa, lo que quiere decir que restará el valor de la base de 90° para poder volver a centrar el vehículo, indicando que estos valores de control tienen una buena lógica.

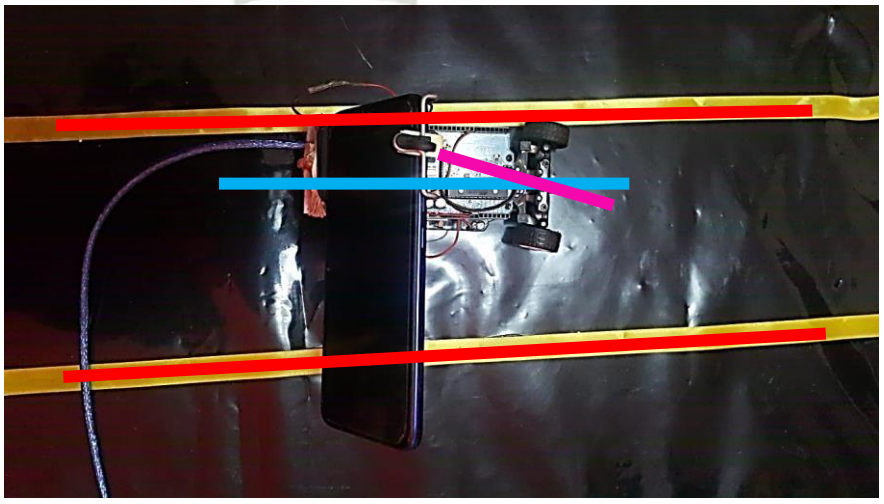


Figura V.6 Giro de las ruedas cuando el carro está descentrado
Fuente: Elaboración propia

Cabe resaltar que estos valores de señal son independientes entre sí, es decir, que no guardan ninguna correlación estricta entre ellas, por lo que ambas señales son necesarias para poder controlar el modelo de vehículo.

En el segundo caso, cuando la regresión es cuadrática, se toman las lecturas de las señales “Centro” y “curva”, y a sus correspondientes ángulos arrojados por el controlador, a través de sendos scopes, obteniéndose la siguiente imagen:

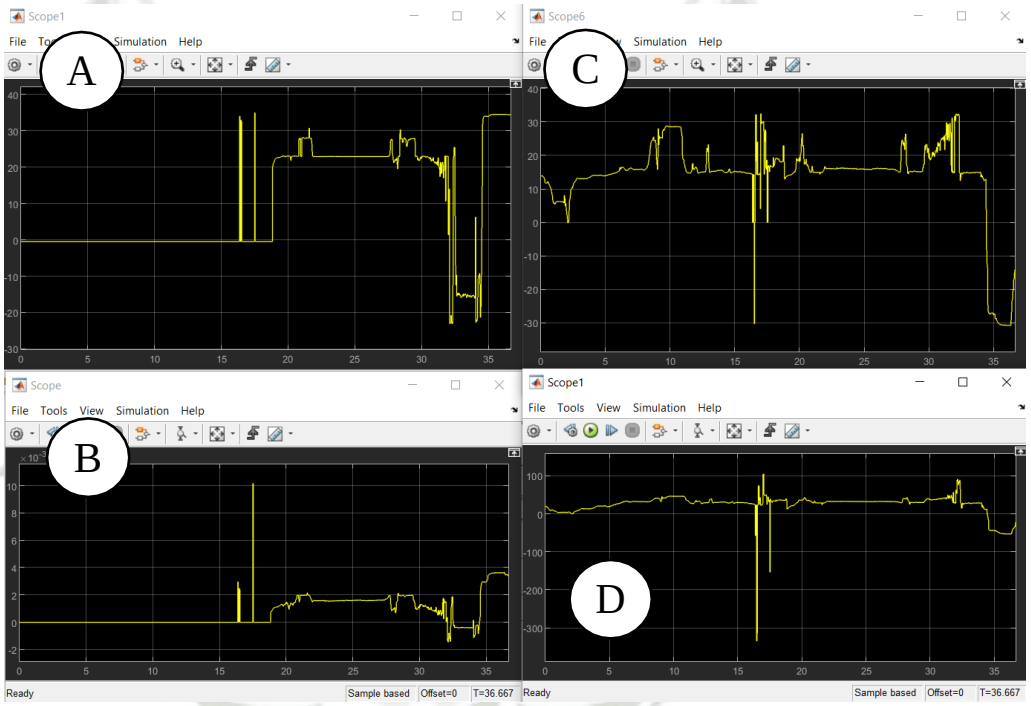


Figura V.7 Gráficos de valores "Curva" y "Centro"
Fuente: Elaboración propia

Donde A) es el ángulo que emite el controlador según la señal “curva”; B) es la señal “curva”; C) es el ángulo que emite el controlador debido a la señal “Centro” y D) es la señal “Centro”. El comportamiento de las señales C y D será el mismo que ya fue analizado anteriormente, por lo que solo nos queda estudiar el comportamiento de las señales A y B.

Se puede observar entonces, que la señal “curva” presenta un periodo en el que es cero, lo que indica que se encontraba el auto sobre un tramo recto, sin embargo, al llegar a cierto punto, se empiezan a reconocer curvas, por lo que la señal empieza a enviar valores diferentes de cero, así mismo el controlador empieza a arrojar valores angulares,

sin embargo, experimentalmente se observó que el ángulo de corrección angular no era el correcto, por lo que se decidió multiplicar por "-1" para poder invertir esta salida del controlador, de modo que pueda trabajar correctamente.

Se puede observar la corrección realizada por el controlador en el siguiente ejemplo

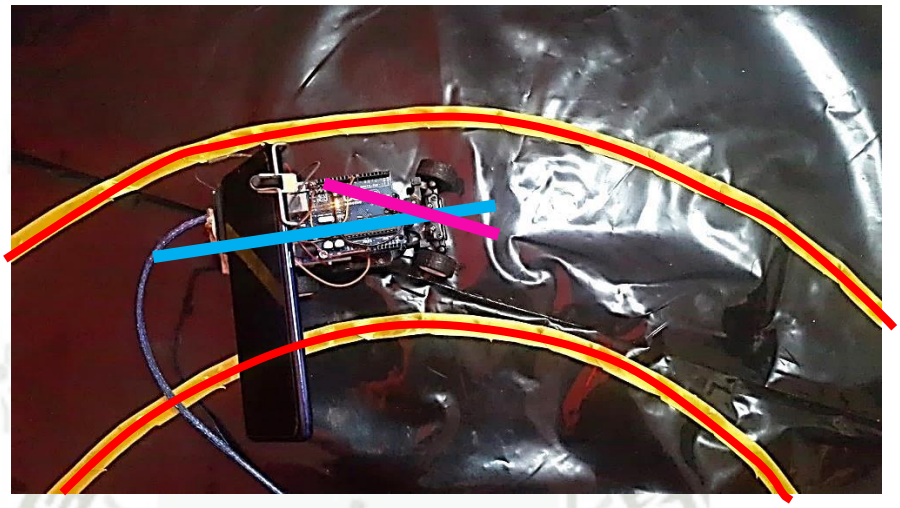


Figura V.8 Giro de las ruedas cuando el carro está en curva
Fuente: Elaboración propia

Siguiendo esta metodología se harán pruebas sucesivas hasta conseguir que el controlador logre el mejor desempeño para mantener el vehículo dentro y en el centro de las líneas de carril. Los resultados obtenidos son los siguientes.

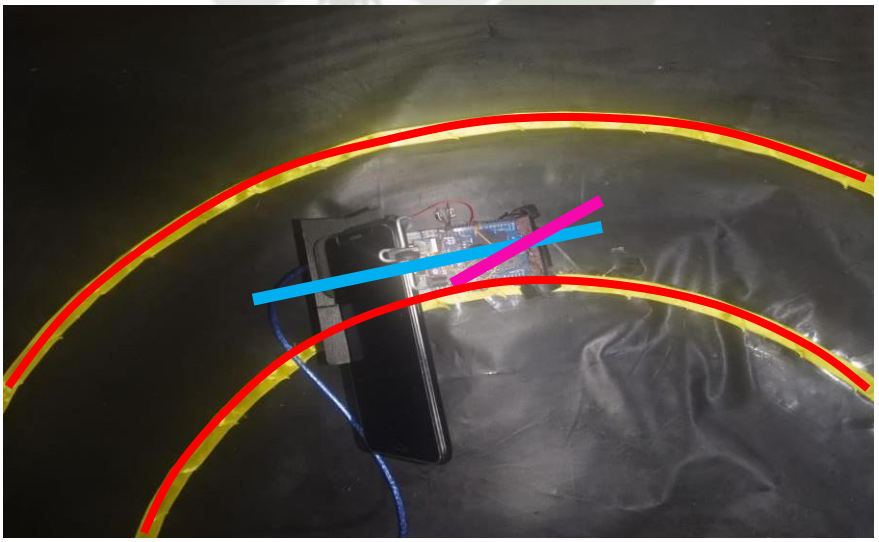
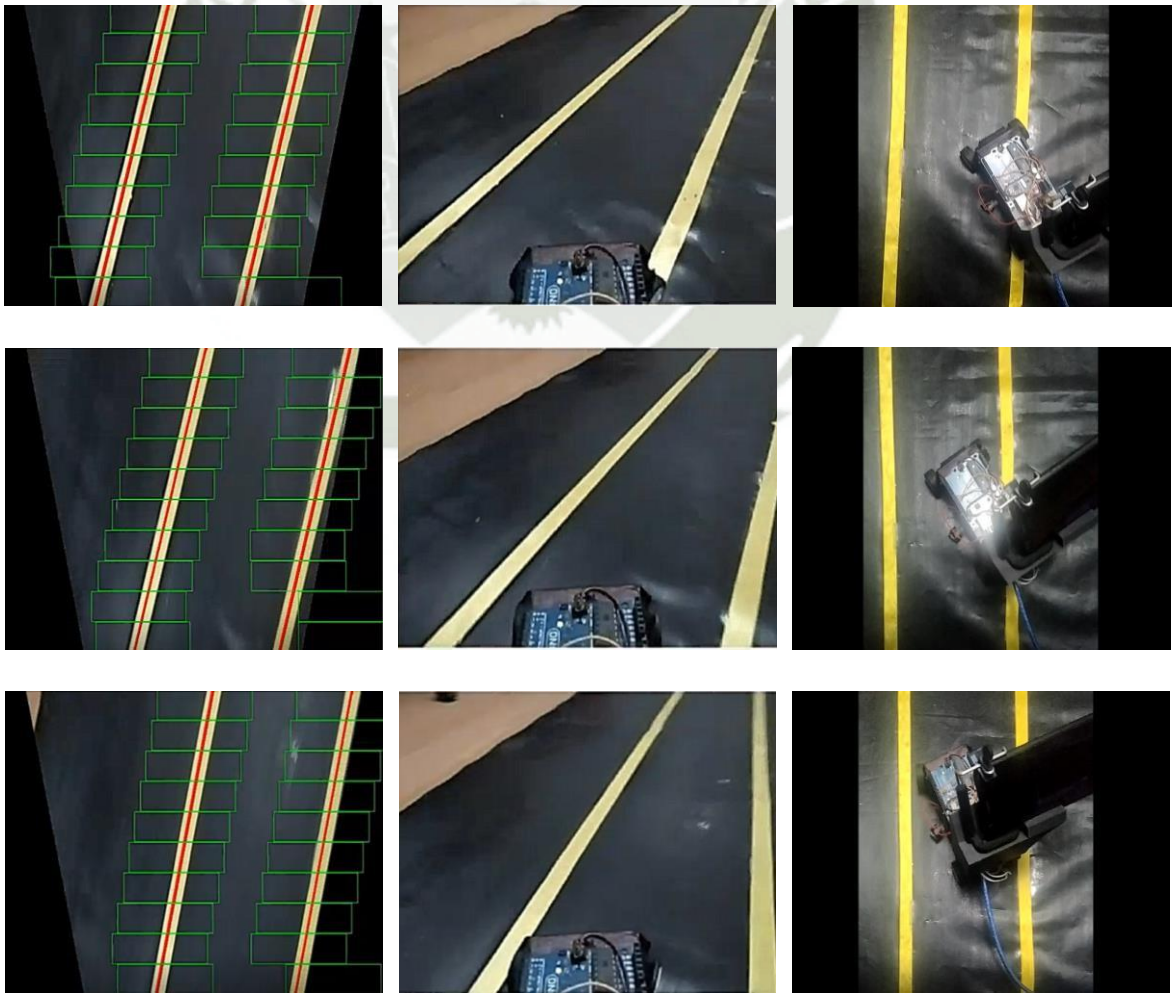


Figura V.9 Giro de las ruedas, cuando el carro está descentrado en curva
Fuente: Elaboración propia

2. RESULTADOS

Se debe tener en cuenta que nuestro “Set Point” sería mantener el vehículo centrado en medio de las líneas de carril, y el parámetro que se encarga de medir esto es el “Centro”, por lo que este error debería de ser constantemente cero o ser cercano al cero. Es necesario conocer que el máximo error que se puede tener de a curdo a las propiedades físicas del sistema es de “±200”.

Al poner a prueba el modelo de vehículo, fue necesario separar los casos en línea recta con perturbaciones y el vehículo funcionando en curva, ya que ambos casos se trabajan con controladores diferentes. Por lo tanto, se mostrarán los resultados de las mejoras generadas por los controladores que ayudan al controlador Centro, tanto en el caso de tramo recto, como en el tramo con curva. El primer caso es del control en línea recta con una perturbación inicial, la cronología de evolución de este proceso es el siguiente:



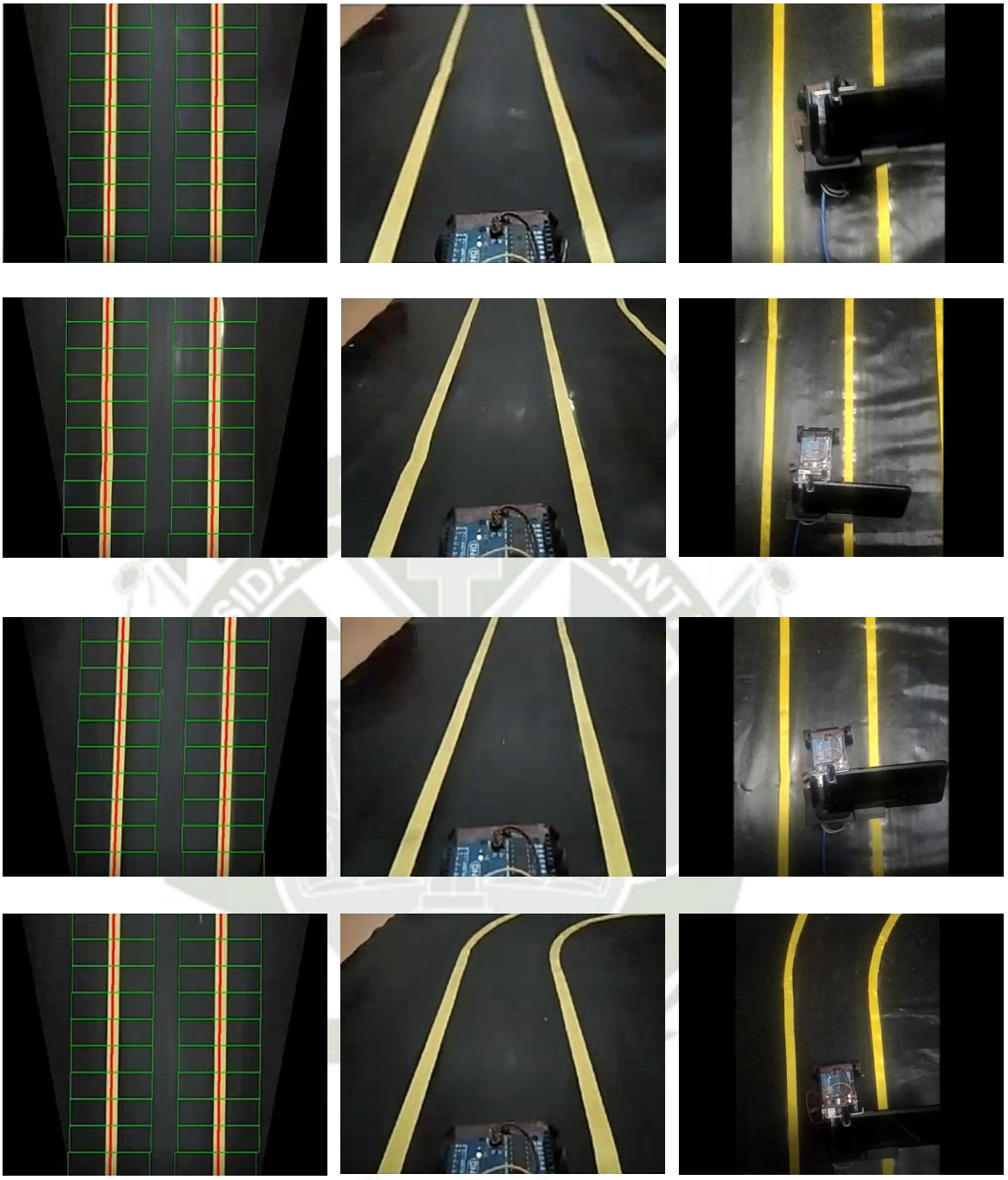


Figura V.10 Cronología del control del modelo de vehículo en línea recta con perturbación inicial
Fuente: Elaboración propia

El gráfico del error de centro obtenido por este mismo proceso es el siguiente:

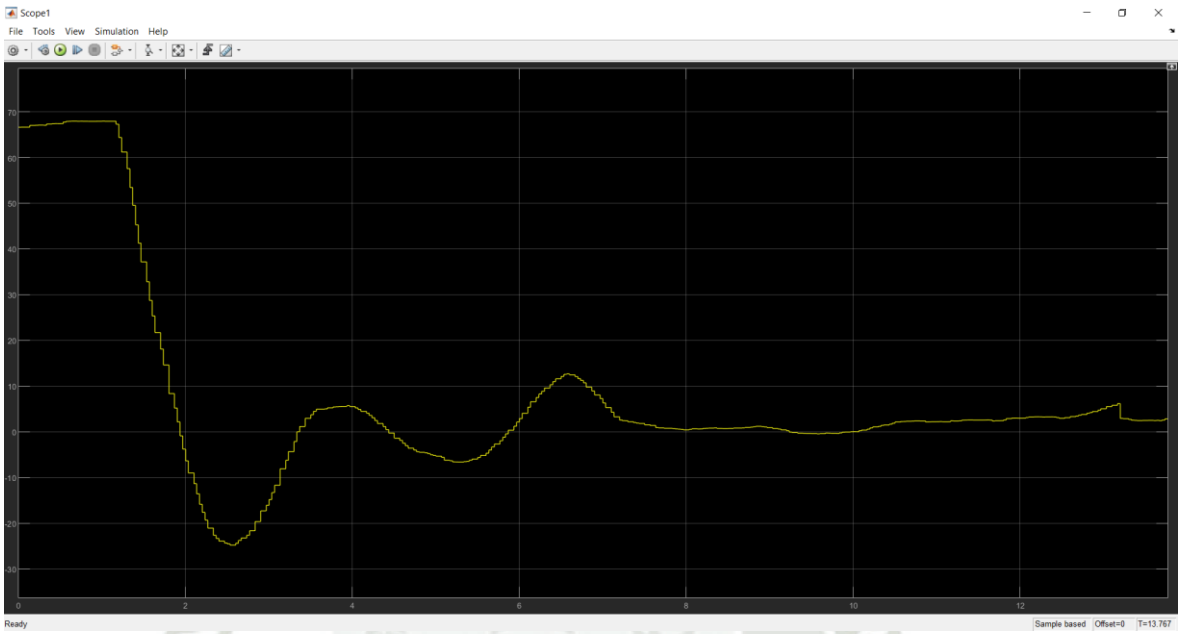


Figura V.11 Gráfico de la señal "Error de centro" en línea recta con perturbación inicial con 2 controladores
Fuente: Elaboración propia

Se pudo observar del gráfico que el sobre impulso al recuperar el centro nos da un error menor de 40, lo que implica un sobre impulso menor al 20%. Así mismo, se observa que el error oscila entre 20 y -20, haciendo que el error en estado estacionario es cero.

Como se mencionó al principio, se hará la comparación de este resultado, en el que trabajan juntos el controlador “Centro” y “Pendiente”, con el resultado del trabajo solo del controlador de “Centro”, el gráfico de los resultados de este controlador se observa a continuación:

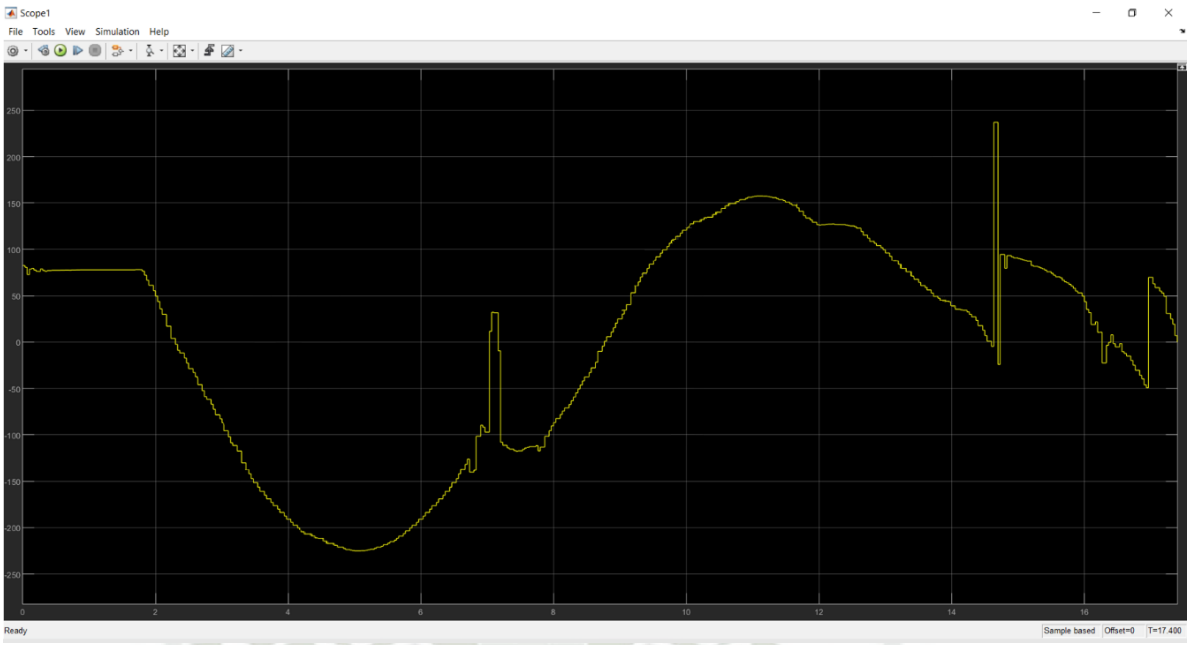


Figura V.12 Gráfico de la señal de "Error de centro" en línea recta con perturbación inicial, solo con controlador de "Centro"
Fuente: Elaboración propia

Se observa que el comportamiento del sistema es más errático con un solo controlador, y no se debe a que el controlador este mal diseñado, sino a que existen factores externos al sistema que provocan que las señales de centro sean muy erráticas cuando hay una perturbación muy grande, por lo que es necesario trabajar con señales que ayuden a la estabilización del modelo de vehículo que, en este caso, son las señales "Pendiente".

De la misma forma, se hizo la prueba con un filtro morfológico de cierre para observar si la señal mejoraba, obteniéndose la siguiente imagen:

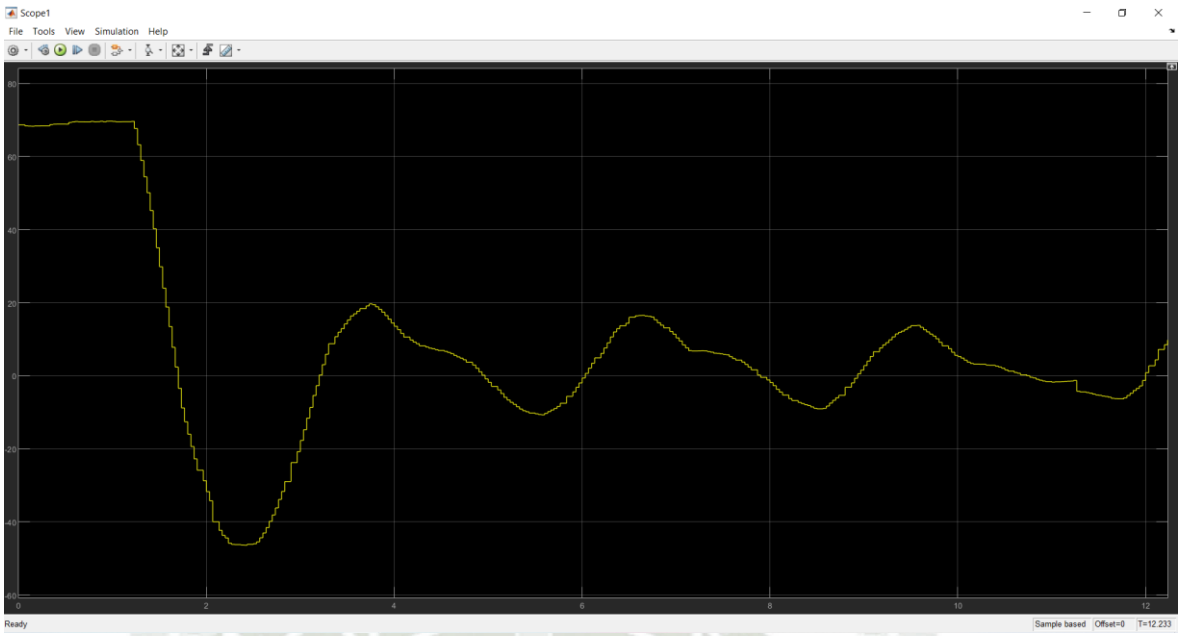
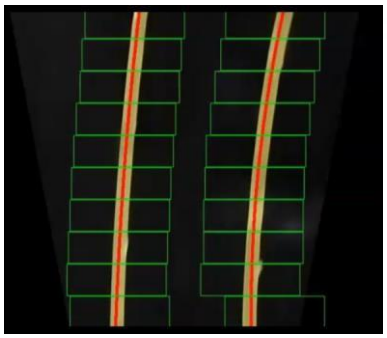
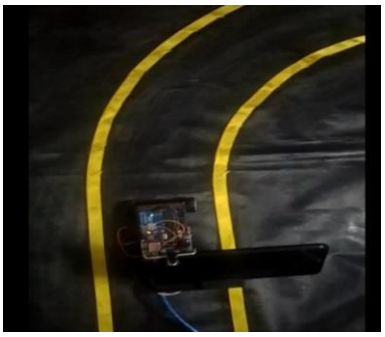


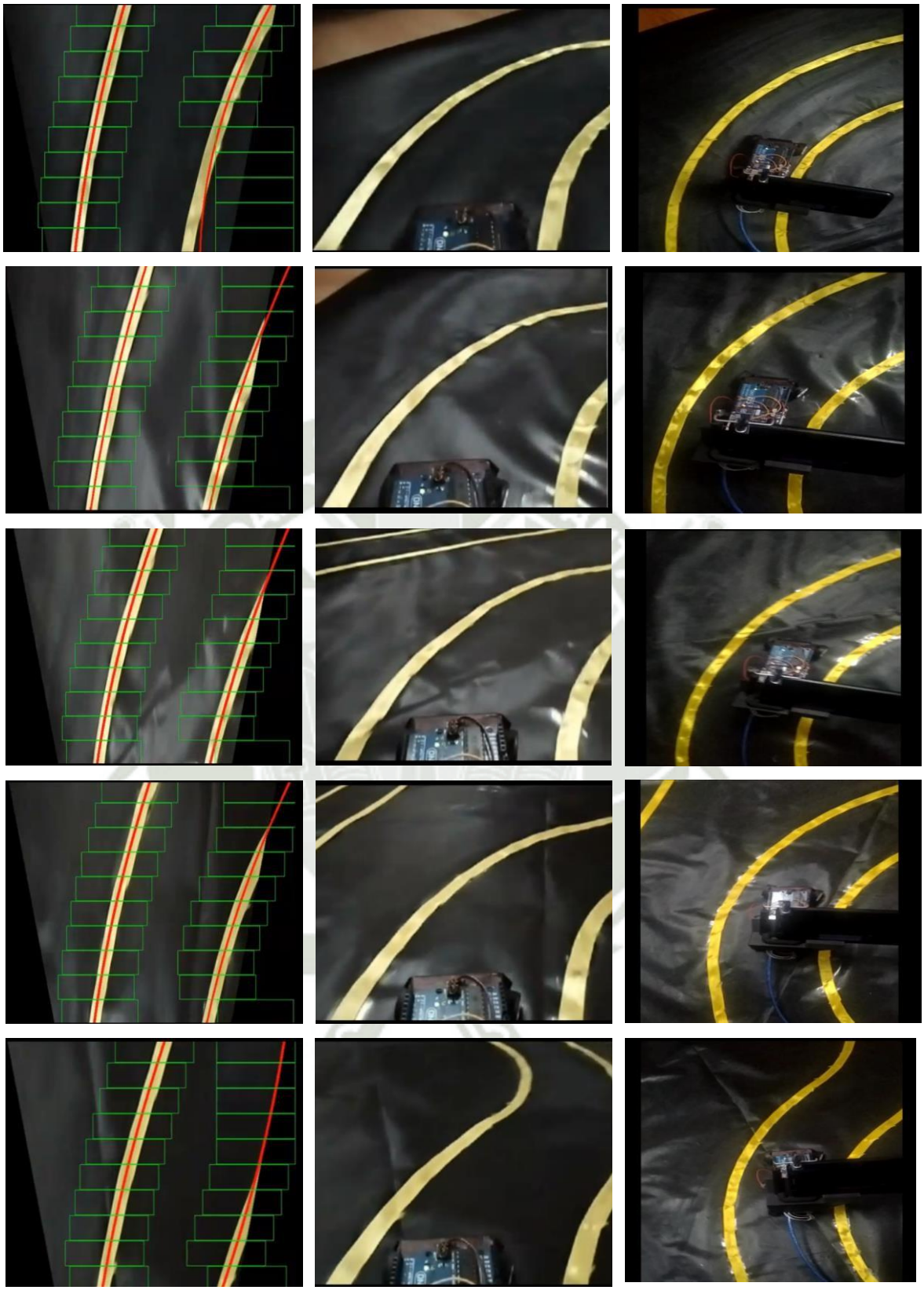
Figura V.13 Gráfico de la señal de "Error de centro" en línea recta con perturbación inicial, con transformaciones morfológicas
Fuente: Elaboración propia

Se observa, sin embargo, que contrario a lo que se pudiera pensar al aplicar el filtro morfológico de cierre, que ayuda a mejorar la nitidez de la imagen a procesar, que la señal de centro no mejora, sino que se hace un poco más inestable, esto es posiblemente debido a que se utilizan mayores recursos computacionales y el procesamiento se hace más lento.

Para el segundo caso, cuando la carretera posee curvas, se observará la cronología y el resultado de la señal de error de centro







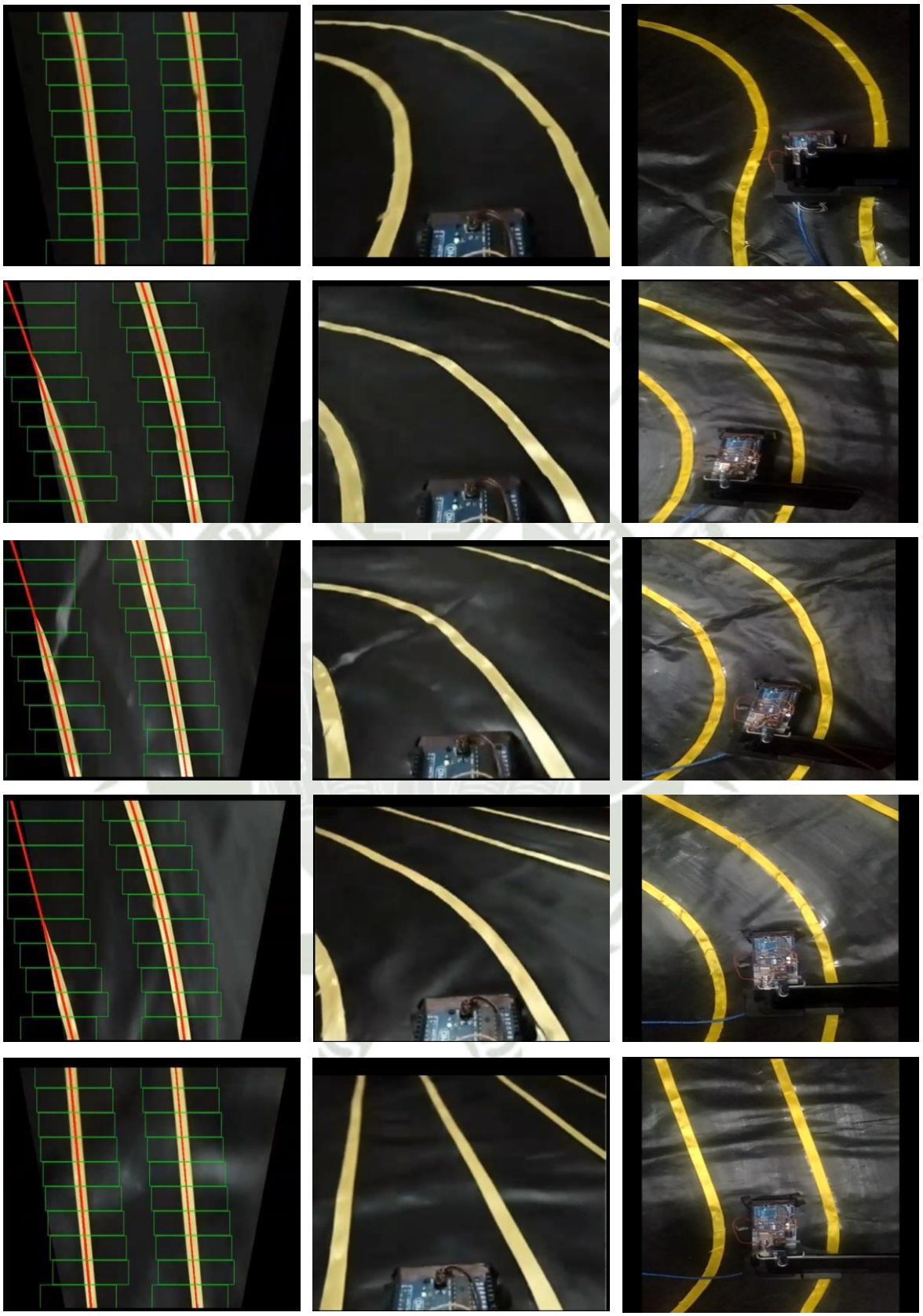


Figura V.14 Cronología del control del modelo de vehículo en tramo curvo
Fuente: Elaboración propia

A fin de poder hacer una comparación de las mejoras que introduce el controlador "Curva", se hicieron pruebas donde se puso a prueba el sistema en un tramo curvo, primero utilizando únicamente el controlador "Centro" y un segundo experimento utilizando los controladores "Curva" y "Centro", que se observan en los gráficos a continuación:

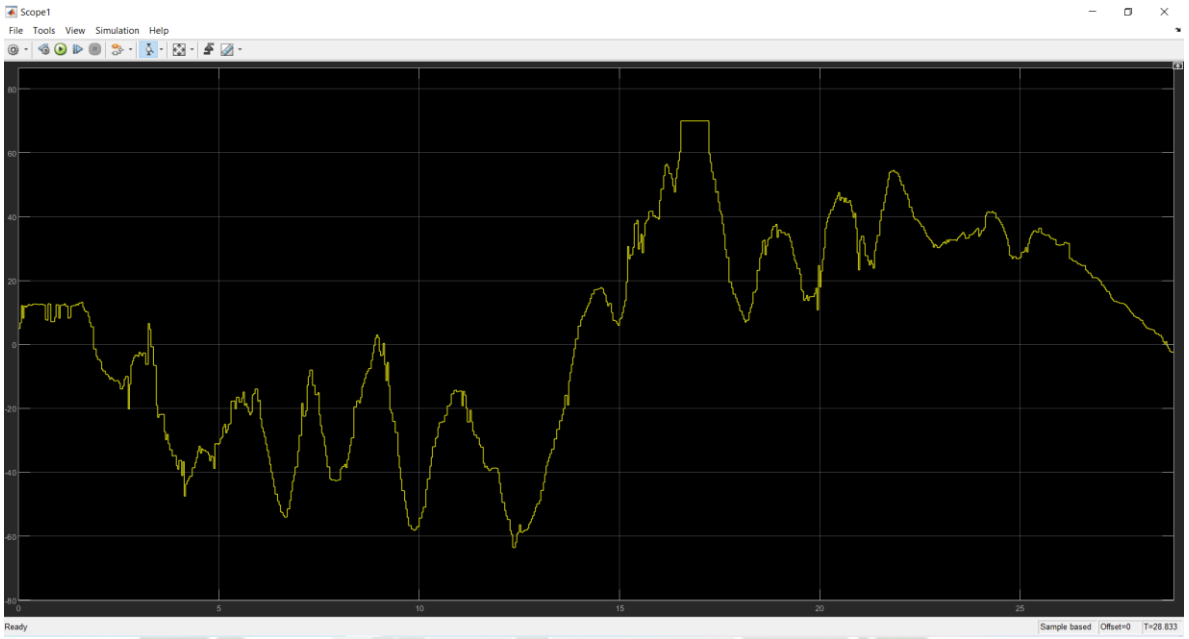


Figura V.15 Gráfico de la señal de error solo con controlador Centro
Fuente: Elaboración propia

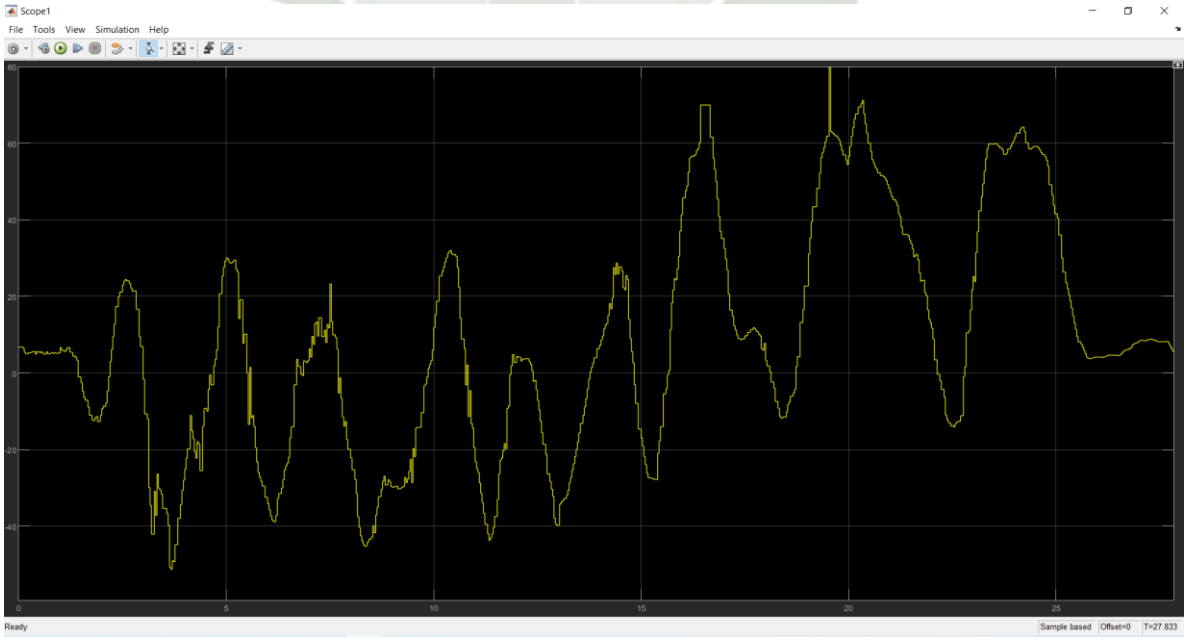


Figura V.16 Gráfico de la señal de erros con controladores Curva y Centro
Fuente: Elaboración propia

Se puede observar que cuando se utiliza solo el controlador “Centro”, la gráfica mantiene un error casi constante en 40, mientras que cuando se añade el controlador de curva el error oscila entre 0 y 60, sin embargo, tiene mayor tendencia a hacer que el error sea cero.

A continuación, se muestra la prueba completa del recorrido del vehículo en el circuito, que une las pruebas en línea recta y en tramos curvo con un poco más de recorrido, el cual será mostrado en el siguiente enlace:

https://www.youtube.com/watch?v=pQWY4sinoaM

A continuación, se mostrarán los datos obtenidos de hacer el recorrido entero por todo el circuito, se mostrarán los gráficos del “Error de centro”, de las señales “Curva”, “Pendiente” y “Ángulo” en las figuras que se muestran a continuación:

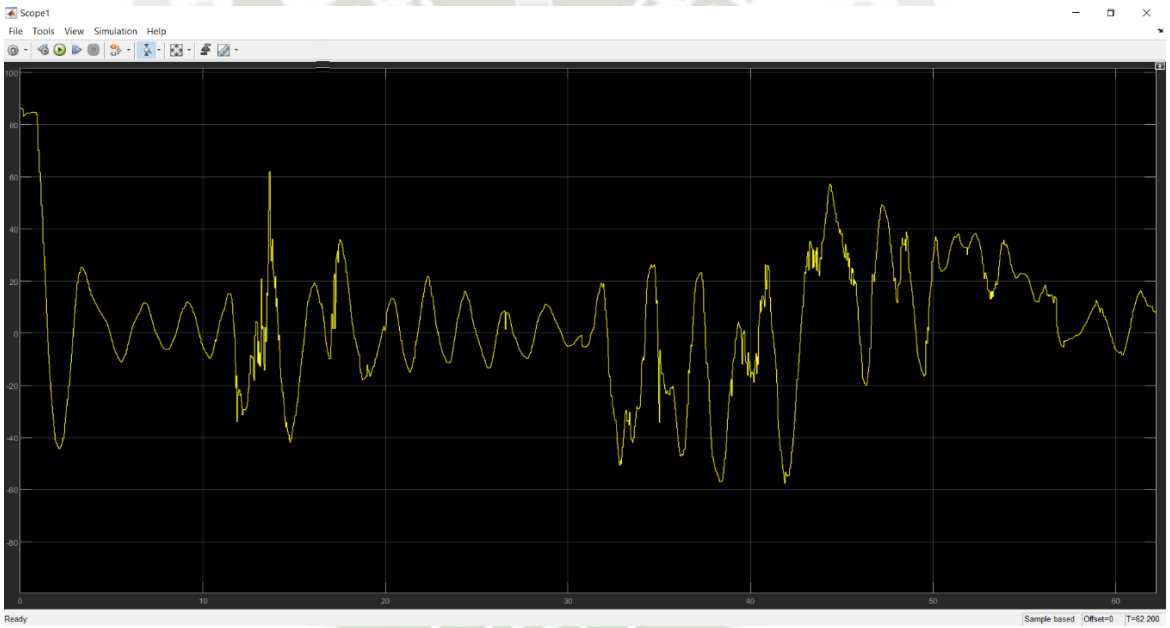


Figura V.17 Gráfica de la señal "Error de centro"
Fuente: Elaboración propia

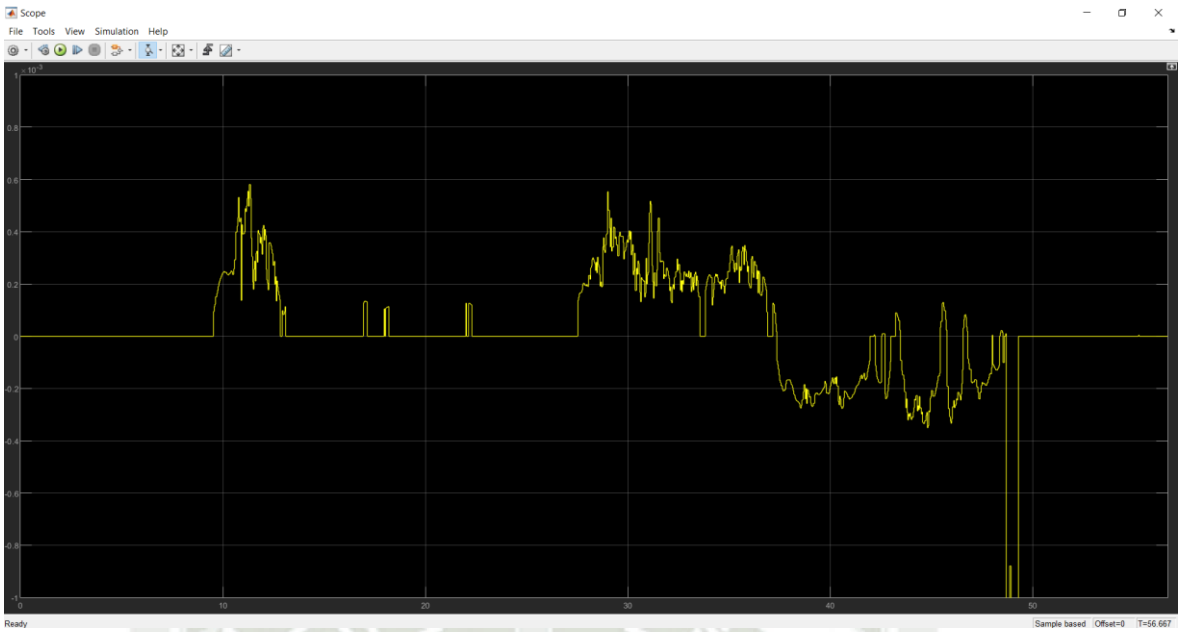


Figura V.18 Gráfica de la señal "Curva"
Fuente: Elaboración propia

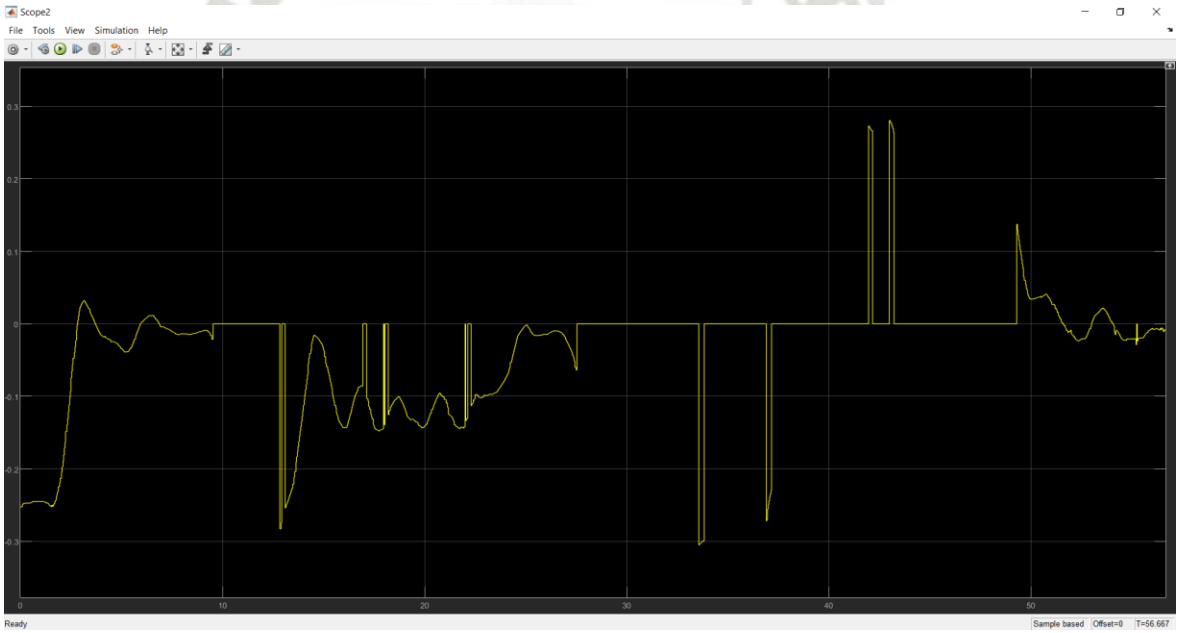


Figura V.19 Gráfica de la señal "Pendiente"
Fuente: Elaboración propia

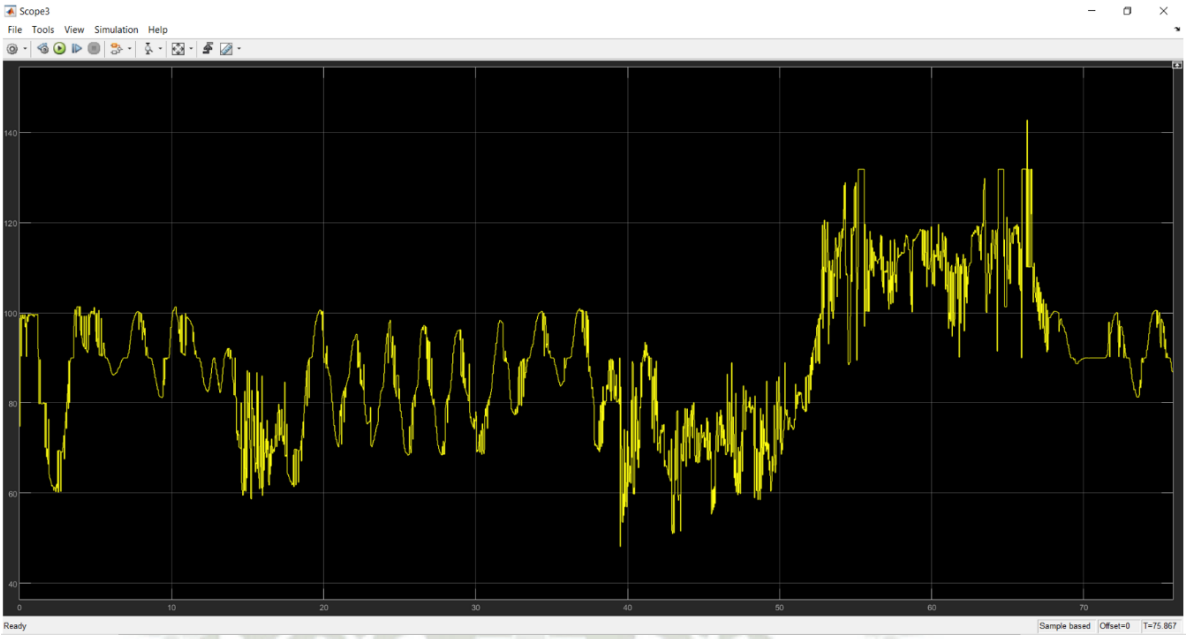


Figura V.20 Gráfica de los valores angulares enviados al servomotor
Fuente: Elaboración propia

De la figura 5.11 se puede observar que los valores de error en zonas de línea recta se mantienen inferiores a un valor de 20 de error de centro, que representa el 10% del máximo de error que se puede tener, además se observa que en la zona de curvas de la trayectoria el error es de aproximadamente 50, que equivale al 25% del máximo de error.

Para determinar qué zona es de curvas o tiene líneas rectas se utilizan las Figuras 5.12 y 5.13; debido a que ambas son excluyentes, es decir, mientras haya señal de “Pendiente” la señal de “Curva” será “0” y viceversa. Por lo tanto, se observa que hay zonas de curva en los periodos desde 9 a 13 aproximadamente y desde 27 a 48 de acuerdo con el tiempo de la gráfica, el resto de los periodos corresponden a zonas que contiene carriles en línea recta o que pueden ser tratados como línea recta, ya que su curvatura es muy pequeña.

Finalmente, de la Figura 5.14 observamos que se envían al servomotor varios valores angulares, sin embargo, en líneas generales, se tiene como eje central los 90°, es decir que este valor angular sería usado para ir únicamente de frente, y cuando existen valores por debajo de 90° es que se necesita ir hacia la derecha, ya sea que haya una curva o para buscar el centro del carril, de la misma forma, para valores mayores a 90° tendremos una dirección hacia la izquierda.

Se hará la correlación entre la gráfica de error de centro y la posición del carro en el trayecto, lo cual ayudará a una mayor comprensión del comportamiento del gráfico de error de centro, para esto se dividirá el trayecto en 7 zonas clave que brindarán una mejor visualización de los errores.

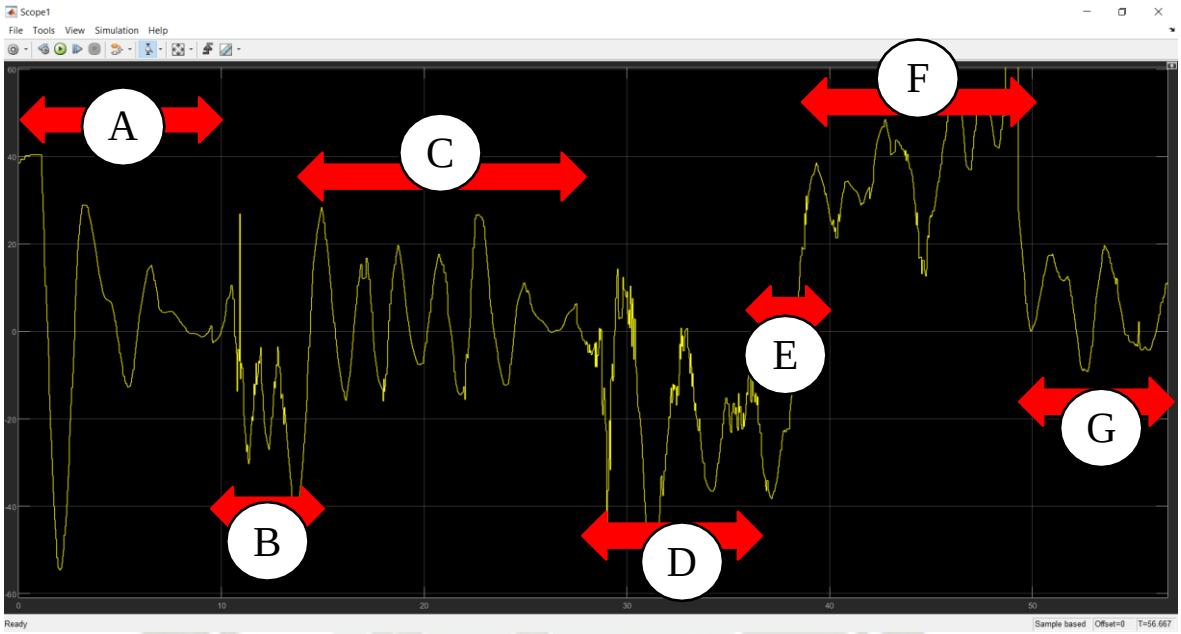


Figura V.21 Gráfica de errores de centro por secciones
Fuente: Elaboración propia

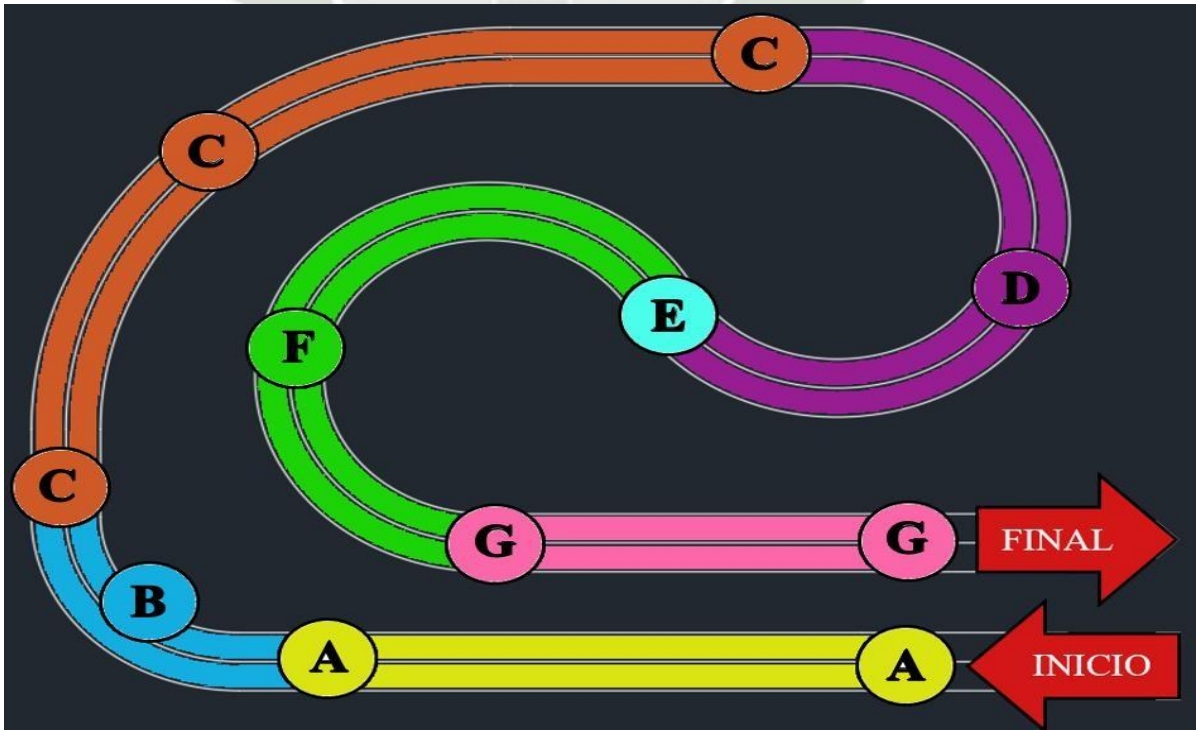


Figura V.22 Gráfico de las secciones de trayecto del gráfico de error de centro
Fuente: Elaboración propia

Como se puede ver las zonas A, C y G corresponden a zonas de línea recta, o que tiene un valor de curva pequeño y que puede ser considerado únicamente como línea recta; esto deja a las zonas B, D y F como zonas de curva. Así mismo la zona E es una zona de transición entre curvas.

En la zona A existe un error inicial alto, debido a que se coloca el vehículo fuera del carril para poder hacer una demostración de la efectividad del algoritmo, se observa que, posteriormente el vehículo no abandonará las líneas de carril.

La zona B es la primera curva a 90°, al igual que en las otras zonas tanto el control por "Error de centro" como el control por "Curva" actúan juntos para poder mantener el vehículo en el medio, ya que son complementarios.

La zona C es especial, porque presenta un primer tramo recto, luego un tramo con una curva no muy pronunciada y finalmente otro tramo recto, en este caso, el valor de curva es pequeño, por lo que el controlador trabaja perfectamente solo con las señales de "Error de centro" y "Pendiente".

La zona D y F son las curvas más pronunciadas del circuito, que describen la forma de casi todo un círculo, se trabajó de modo que el vehículo no tenga una reacción tan brusca al tomar la curva (lo que ocasionaría que el vehículo se descarrile) y a la vez tenga un giro suficiente para poder permanecer al medio del carril. Cabe aclarar que, a pesar de que en el gráfico de error de centro se observa que el error es de 40 y en ocasiones más alto, se debe considerar que la cámara acompaña el movimiento del vehículo, adicionalmente que en esta zona de curva se podría considerar que existen constantes perturbaciones en la imagen, ya que pequeños errores de giro dentro del carril crean grandes errores en la gráfica de error de centro; aun teniendo en cuenta todo esto, y luego de varias pruebas se llegó a la conclusión que el mejor resultado que se podría tener es el que se presenta aquí.

Una zona particularmente complicada es la zona E, ya que existe una transición o cambio entre curvas, lo que puede ocasionar algunos conflictos en la imagen que llega al controlador. Algunos de estos conflictos pueden ser:

- Pérdida de la visualización de alguna de las líneas de carril

- Contradicción entre las curvaturas, es decir que una línea de carril tendrá una curva positiva (que gira hacia la derecha) y otra línea de carril con una curva negativa (que gira a la izquierda).

Ambas situaciones fueron consideradas dentro del código del vehículo, de modo que cuando ocurran el vehículo no incurra en error.

Finalmente, la zona G representa el final del recorrido, es un tramo recto que no tiene gran complicación, salvo por el hecho de que se quiere evitar el “cruce de información”, es decir que se reconozcan las líneas que se encuentran frente a estas líneas.

CONCLUSIONES

En el trabajo se determinó que la mejor forma de medir el error del modelo de vehículo con el control PD difuso de trayectoria, era separar los tramos en “Rectos” y “Curvos”. De esta manera, se halló un error máximo de 10% en tramos rectos y del 25% en tramos con curva. Es decir, considerando que el ancho de pista es 140 mm, el máximo descentramiento que tiene el vehículo en tramos rectos es de 14mm, y en tramos curvos de 35mm.

Así mismo, se descubrió una forma efectiva de adquisición y procesamiento de datos de video, la cual fue utilizando el “Simulink” en Matlab, junto con una aplicación de video “DroidCam”, que permitía la adquisición de datos desde el celular al computador.

De la misma manera se estudió y determinó que la manera más conveniente de identificar líneas de carril es utilizando el método de Transformada Hough, que presenta un menor procesamiento y facilidad en la gestión de datos.

Además, se determinó que las funciones de membresía del tipo “triangular” y “campaña de Gauss” son las más adecuadas para el diseño del controlador, ya que permiten una mayor diferenciación de datos y separación de las respuestas a la salida del controlador, sin perder la suavidad a la salida del controlador.

Finalmente se realizó un manual de una aplicación virtual similar a la expuesta en esta tesis, la cual se deja a libre uso de quien que desee experimentar de primera mano una experiencia similar a la expuesta en esta tesis.

RECOMENDACIONES

Se recomienda tener en cuenta los niveles de luz a los cuales estará expuesto el proyecto que se quiera desarrollar, ya que, al momento de detectar las líneas de carril, si no se tienen condiciones óptimas puede incurrir en error.

Así mismo se recomienda realizar el proyecto en algún lugar completamente plano, evitar en lo posible irregularidades en el trayecto a seguir, porque algunos reflejos de superficies no lisas provocan errores.

En cuanto al montaje del vehículo, se recomienda ser muy riguroso para su confección, debido a que esta será la base sobre la que se desarrollará el proyecto, por lo que en general se recomienda tener bien seguro el lugar en el que se colocará la cámara, que sea entable y se encuentre bien centrada. De la misma forma hay que considerar que debe de haber suficiente peso en las llantas delanteras que giran para que el vehículo tome la dirección de esas llantas y no simplemente se vaya de frente. No se debe de vacilar al momento de hacer algún cambio en la estructura del vehículo.

Se recomienda ser muy crítico con los errores que se presenten mientras se experimenta con el vehículo, en esta experiencia se encontraron problemas como la variación constante de datos, lo que provoca errores pequeños que a la larga pueden ser peligrosos, así mismo se dan situaciones en las que existen contradicciones entre las líneas de curva o no se llega a observar alguna línea de carril, por lo que estas situaciones deben de ser consideradas en el código y seguir experimentando. Todos estos problemas hallados son propios de trabajar con sistemas reales, que es muy diferente de los sistemas teóricos.

En el caso del control difuso, tiene la ventaja y desventaja de ser flexible, por lo que muchas veces llega a corregir los errores, pero no lo realiza por completo, como se pudo observar en este proyecto, y al trabajar con funciones de membresía, es decir rangos de datos, muchas veces se hace complicado llegar a una mejor solución porque las posibilidades de combinación son infinitas, por lo que se recomienda ser constante al momento de hacer las pruebas de los sistemas desarrollados y saber reconocer el límite al que llega el sistema. Por lo mismo se recomienda hacerse bastante espacio de tiempo para poder hacer las pruebas necesarias, tomar apuntes de los progresos o mayores errores generados al hacer los cambios.

Esto conlleva a recomendar que, al momento de experimentar con el sistema difuso, se realice 1 solo cambio para ver cómo funciona, a que, si se hacen varios cambios a la vez, no

se tendrá certeza de cuál fue el factor que ayudó o empeoró el funcionamiento del sistema, cuando se vayan a realizar cambios grandes se recomienda generar copias de respaldo para poder saber los valores anteriores y no se pierda todo el avance si los cambios no son positivos.

Se recomienda tener en cuenta que el giro del servomotor puede ser muy pequeño, por lo que se recomienda tener una conexión lo más directa posible entre en servomotor y las llantas delanteras, todo esto ayudará a hacer que el sistema responda de la mejor forma posible.

Se recomienda tener un ordenador potente para la realización de este proyecto, debido a que como se utilizan diversos recursos computacionales, es posible que demore bastante en cargar los programas, sin embargo, se observa que es posible implementarlo en su totalidad y realizar las pruebas respectivas.

Se recomienda tener una velocidad de desplazamiento del vehículo baja, para que el sistema no se descontrole en el proceso en que se reconoce el error, se calcula la señal de corrección, se envía la señal al servomotor y este efectúa el cambio. En este aspecto un control más riguroso puede ayudar a mantener bajos los niveles de error, sin embargo, como se conoce por teoría, valores demasiado altos podrían causar un sistema inestable que aumente constantemente los errores.

REFERENCIAS

Andrade Fierro, L. & Chulca Simbaña, L. (2018). Desarrollo de un sistema de visión artificial para el reconocimiento, clasificación y maquinado de patrones con una tarjeta ARM. Universidad Politécnica Salesiana-Ecuador.http://dspace.ups.edu.ec/handle/123456789/15284

Alegre, E., Pajares, G. & De la Escalera, A. (2016). *Conceptos y métodos en visión por computador*. Grupo de visión del comité español de automática https://intranet.ceautomatica.es/sites/default/files/upload/8/files/ConceptosyMetodosenVxC.pdf

Aral Sarrafi (2021). Lane_Keeping_Assist_System (https://github.com/Aral-Sarrafi/Lane_Keeping_Assist_System), GitHub. Retrieved October 29, 2021.

Barba Guamán, L. R. (2015). *Utilización de Métodos de Visión Artificial Para PC Como Apoyo en la Automoción*.

Barroso Rodriguez, P. (2016). *Control y planificación de robots móviles*. Universidad de Sevilla.

Correa Sandoval, J. M., & Díaz Zapata, M. A. (2019). *Desarrollo de un Sistema de Percepción para Detección de Carril y Generación de Trayectorias para Vehículos Autónomos*. Universidad Autónoma de Occidente.

Diciembre Sanahuja, M. (2016). *Sistemas de Control con Lógica Difusa: Métodos de Mamdani y de Takagi-Sugeno-Kang (TSK).* Universidad Jaume I

Espada, V. N., & Araya, L. V. (2015). *Técnicas Computacionales Aplicadas a la Asistencia en la Conducción de Vehículos*. Universidad Nacional del Centro de la Provincia de Buenos Aires.

Fernández García, N. (2017). *Introducción a la visión artificial-Visión artificial avanzada*. Universidad de córdoba. Departamento de informática y análisis numérico http://www.uco.es/users/ma1fegan/2014-2015/vision/Temas/tema-1.pdf

Ferro, G. (2018). *Investigación en redes convolucionales y una aplicación a la detección de Peatones*. *July*, 19–21.

García, I.; Herrera, D. & Mina, Jorge (2017) *Detección automática de líneas de cultivo de papa utilizando imágenes digitales.* SATHIRI Vol. 12- N°2, pp 46-67 https://revistasdigitales.upec.edu.ec/index.php/sathiri/article/download/107/137/504

García Mateos, G. (2017). *Procesamiento de imágenes.* Universidad de Murcia. Departamento de informática y sistemas http://dis.um.es/profesores/ginesgm/pi.html

Gómez Hidalgo, R. (2019). Segmentación de imágenes RGB en alta resolución espacial. *Universidad Politécnica de Madrid*. https://oa.upm.es/55736/1/TFG_RUBEN_GOMEZ_HIDALGO.pdf

Gómez Zurita, J. L. (2015). *Detección de Líneas de Carril en Autopista*. Universidad Autónoma de Barcelona.

Gonzáles Marcos, A., et al. (2006). *Técnicas y algoritmos básicos de visión artificial*. Universidad de la Rioja. https://publicaciones.unirioja.es/catalogo/online/VisionArtificial.pdf

Guanrong, C. & Tat Pham, T. (2006) Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems. CRC Press LLC. https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/CHEN-PHAM-Introduction-to-Fuzzy-sets-Fuzzy-logic-and-Fuzzy-control-systems-Page-160.pdf

Hernández Millán, G., Ríos Gonzales, L. H., & Bueno López, M. (2017). Implementación de un controlador de posición y movimiento de un robot móvil diferencial. Revista Tecnura, 20(48), 123-136. doi: 10.14483/udistrital.jour.tecnura.2016.2.a09

Ibarra Arenado, M., Tjahjadi, T., Robla-Gómez, S., & Pérez-Oria, J. M. (2015). Localización del punto de Fuga para Sistema de Detección de Líneas de Carril. *XXXVIII Jornadas de Automática*, *67*, 983–990. https://doi.org/https://doi.org/10.17979/spudc.9788497497749.0983

Justo de Frías, C. (2018). *Visión artificial aplicada en la identificación de objetos y su parametrización geométrica*. https://e-archivo.uc3m.es/handle/10016/29643

Khairdoost, N.; Beauchemin, S. & Bauer, M. (2021). *Road Lane Detection and Classification in Urban and Suburban Areas Based on CNNs*. University of Western Ontario

Ladero García, R. (2019). *Detección De Señales y Líneas de Carril para Conducción Autónoma con Vehículo a Escala* [Escuela Técnica Superior de Ingenieros Industriales]. http://oa.upm.es/54102/1/TFG_RODRIGO_LADERO_GARCIA.pdf

Llivisaca Aucapiña, C. (2018). *Diseño y construcción del sistema de dirección de un vehículo de competencia Fórmula SAE eléctrico.* Universidad Politécnica Salesiana Ecuador. https://dspace.ups.edu.ec/handle/123456789/15520

Mathalagu, R., Bolimera, A., & Kalaichelvi, V. (2020). Lane Detection Technique Based on Perspective Transformation and Histogram Analysis for self-driving cars. *ELSEVIER*, 1–16. https://doi.org/https://doi.org/10.1016/j.compeleceng.2020.106653

MathWorks. (2019). *Sensor Fusion and Tracking for Autonomous Systems*.

Khan, M.A.-M.; Haque, M.F.; Hasan, K.R.; Alamjani, S.H.; Baz, M.; Masud, M.; Al-Nahid, A. (2022): *LLDNet: A Lightweight Lane Detection Approach for Autonomous Cars Using Deep Learning. Sensors 2022, 22, 5595*. https://doi.org/ 10.3390/s22155595

Montoya Baidal, D. F., & Pachacama Tamayo, A. R. (2016). Diseño e Implementación de unPrototipo de Detección de Carril Mediante Visión Artificial. In *Escuela Politécnica Nacional*. Escuela Politécnica Nacional.

Mori Virhuez, B. E. (2018). Diseño de un Vehículo Aéreo-Terrestre No Tripulado con Autonomía de Funcionamiento de Larga Duración Orientado a Operaciones de Búsqueda y Rescate. In *Pontificia Universidad Católica Del Perú*. Pontificia Universidad Católica Del Perú.

Ornelas Tellez, F. (2015) Control con lógica difusa. Universidad michoacana de San Nicolás de Hidalgo. Facultad de Ingeniería Eléctrica. http://dep.fie.umich.mx/~fornelas/data/uploads/pres_controldifuso_parte_2.pdf

Ponz Camps, F. de B. (2016). *Navegación automática en un vehículo con distribución Ackerman*.

Rodríguez Garavito, C. H. (2017). *Sistema Avanzado de Asistencia a la Conducción para Entornos Interurbanos*. Universidad Carlos III de Madrid.

Romero-Macas, H. (2020), *Algoritmo de reconocimiento de personas mediante procesamiento digital de imágenes usando MATLAB*, Polo del Conocimiento; Vol 5, No 8 (Año 2020). http://dx.doi.org/10.23857/pc.v5i8.1656

Sosa Cervantes, C. Y. (2016). *Técnicas de control automático para los robots móviles de ruedas*. INSTITUTO POLITÉCNICO NACIONAL.

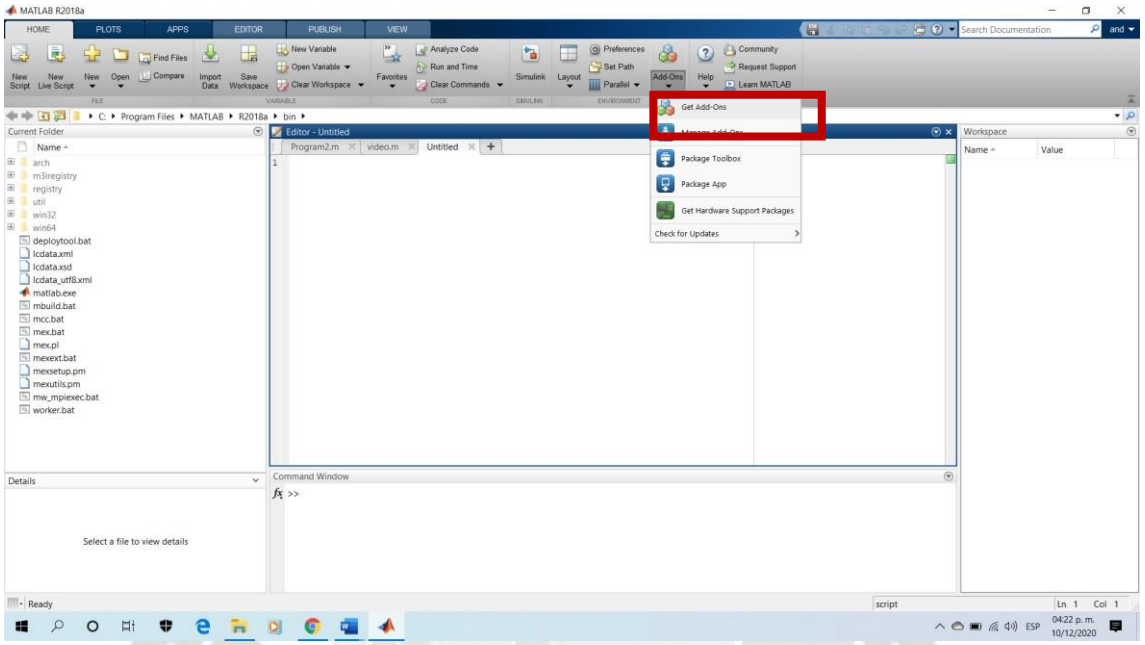
Torres Guaita, I. (2017). *Desarrollo de un Algoritmo de Visión Artificial para el Guiado Automático de un Robot Móvil*. Universidad Politécnica de Valencia.

Tosini, M., & Leiva, L. (2018). Detección de Carriles en Caminos Rurales no Pavimentados. *XXIV Congreso de Ciencias de La Computación*, 928–937.

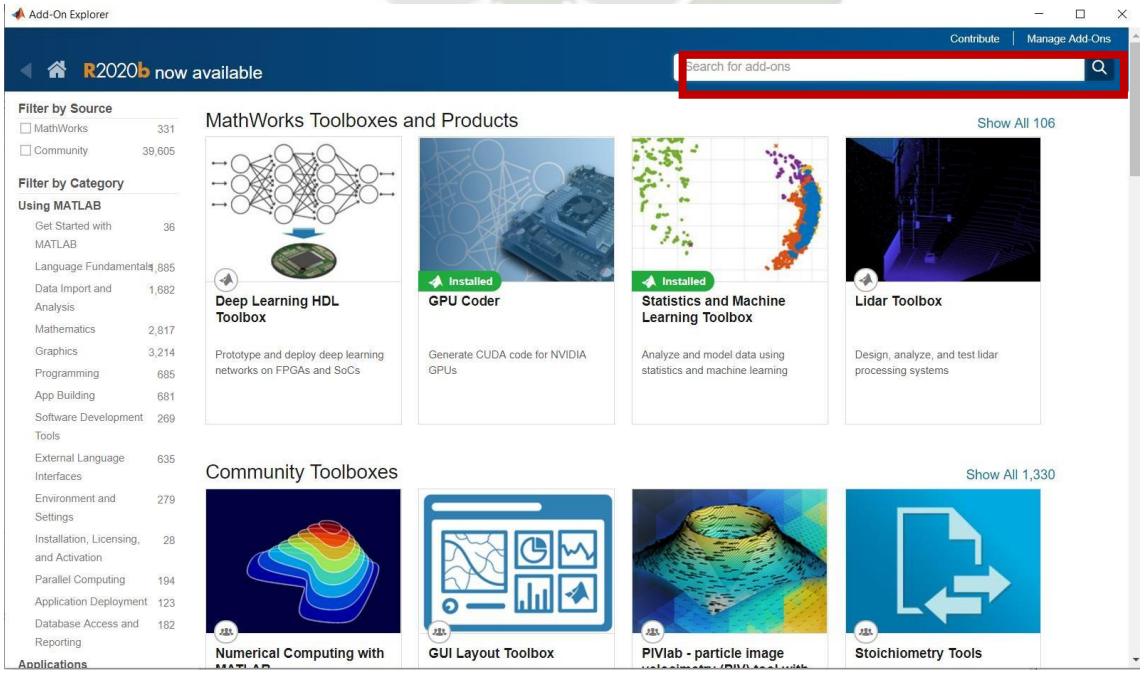
Anexo A:

A.1 PROCEDIMIENTO PARA INSTALACIÓN DE TOOLBOX EN MATLAB

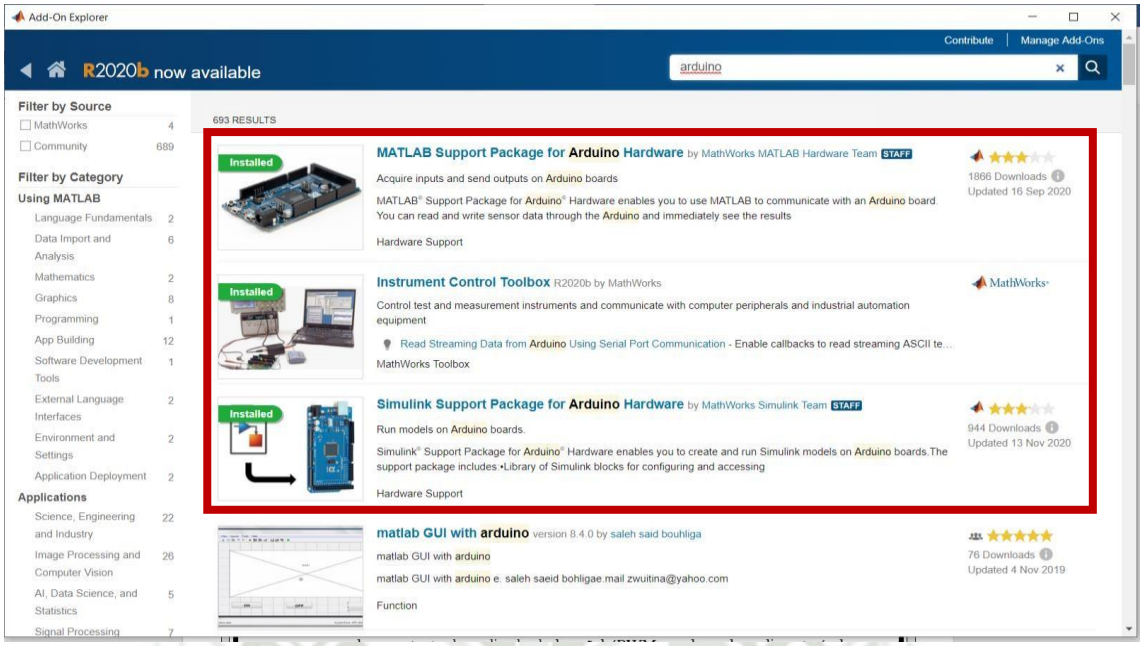
1. Se entra a la parte de ADD ONS



2. Usamos los paquetes descargados desde la plataforma de Matlab en internet

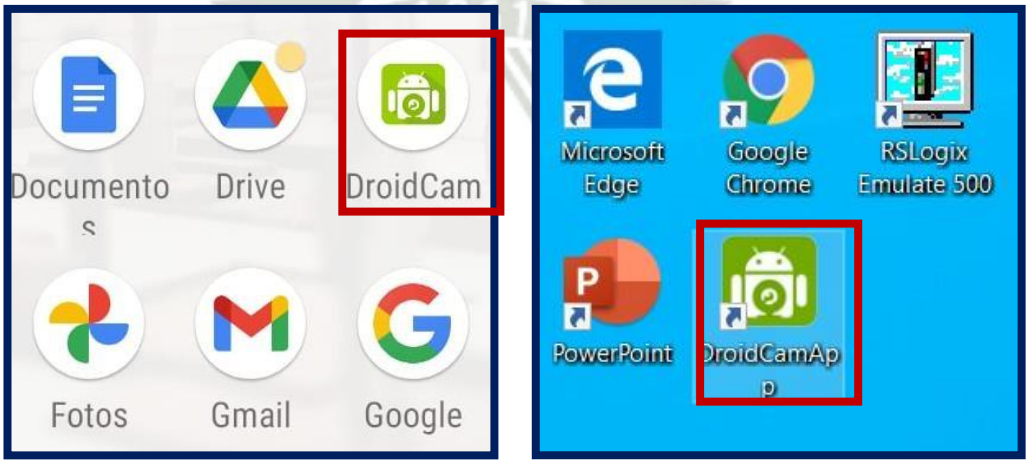


3. Se selecciona el paquete a querer instalar



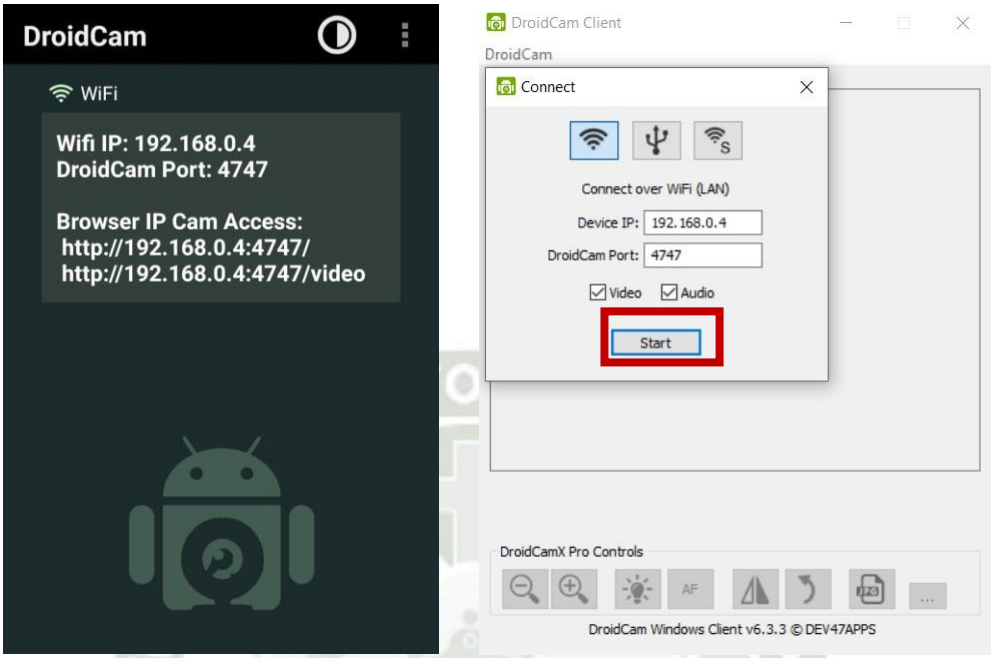
A.2. ADQUISICIÓN DE VIDEO DE UNA CÁMARA EXTERNA

Para adquirir video desde una cámara remota, se hace necesario utilizar algunas aplicaciones tanto en el celular como en la PC. En este caso se utiliza la aplicación "DroidCam", en las imágenes inferiores se observan las aplicaciones, en la izquierda en el celular y a la derecha en la PC.



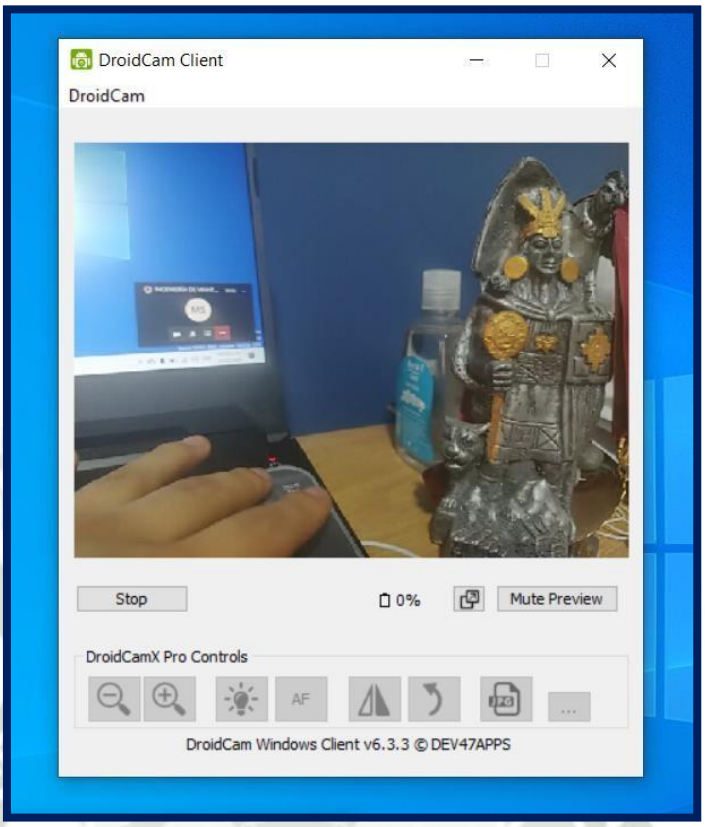
Una vez instaladas esas aplicaciones, se procede a abrir ambas aplicaciones teniendo que seguir los siguientes pasos para transmitir el video. Al abrir ambas aplicaciones se

observan las siguientes ventanas, a la izquierda la que se observa en el celular y a la derecha la que se observa en la PC



El celular se encarga de generar una dirección IP, por la cual enviará el video y en la computadora se observa que aparece un cuadro de diálogo en el cual se puede elegir una dirección IP y un puerto que están indicados en la aplicación del celular, al hacerse clic en "Start" se activará la cámara del celular y se empezará a enviar una señal al computador, en la imagen inferior se observa la imagen brindada por la cámara, y que será la misma que aparecerá en la computadora

Y la imagen recibida en el computador es la siguiente:



Se observa que las imágenes enviada y recibida son las mismas, sin embargo, la imagenes adquirida en una aplicación y no en un software adecuado para poder hacer el procesamiento del video, por lo que este es solo otro de los pasos necesarios para nuestro proyecto y el reto ahora consiste en poder llevar el video adquirido a la aplicación de Matlab

A.3. ADQUISICIÓN DE VIDEO DE UNA CÁMARA EXTERNA EN MATLAB

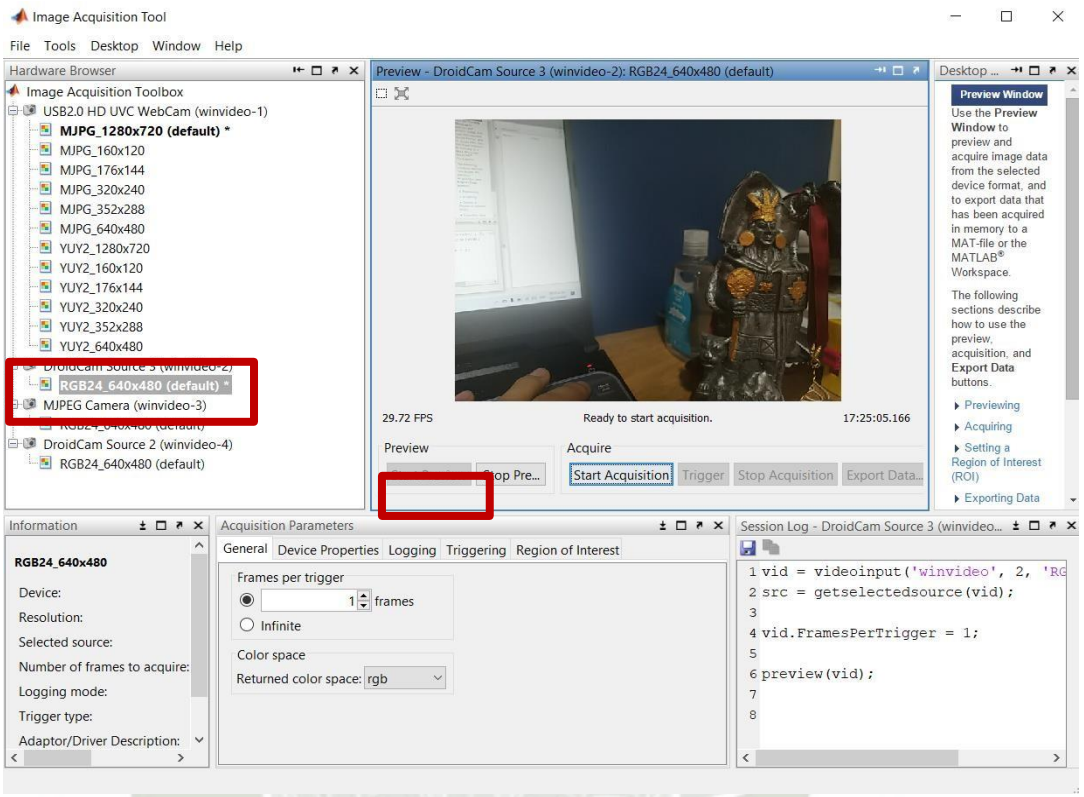
En esta parte utilizaremos una de las “Apps” de Matlab, para lo cual nos dirigimos a la sección de Apps en Matlab y seleccionamos “Image Acquisition”

Otra alternativa para adquirir a esta aplicación es escribiendo en el Command Window el comando: “imaqtool”.

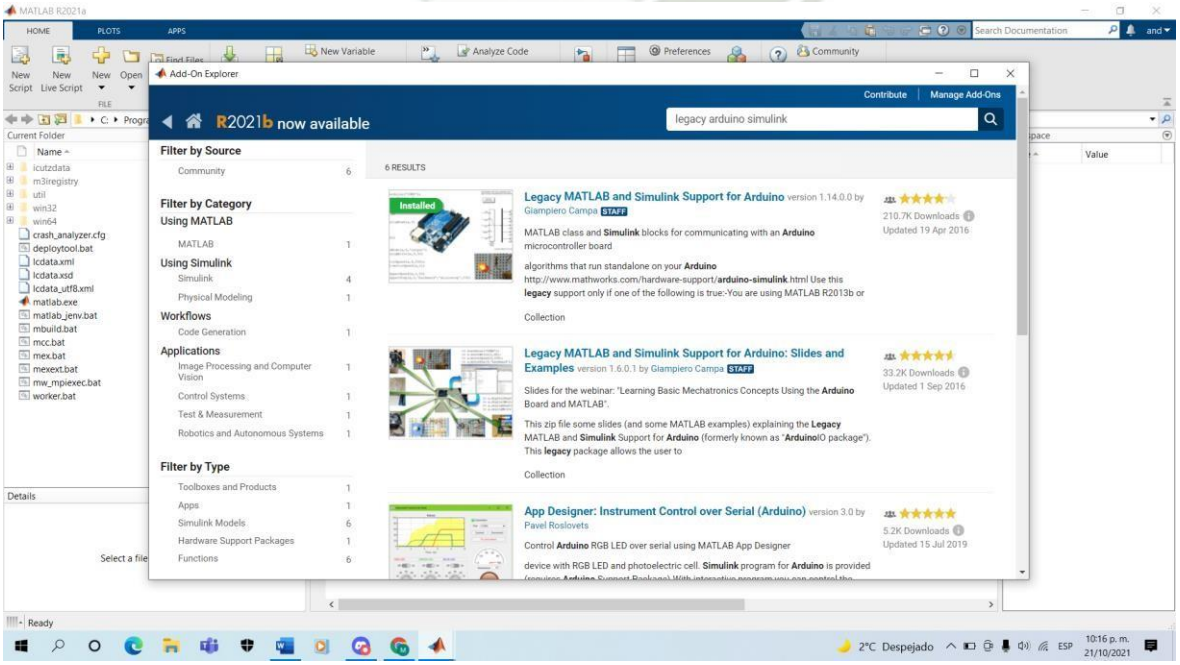
Nos aparecerá la siguiente ventana, con todas las entradas de cámara que existen en nuestra laptop:

- USB 2.0 HD UVC Webcam (winvideo) es la cámara incorporada en la laptop

- DroidCam Source 2 y 3: Son las cámaras adquiridas por la instalación de las apps DroidCam, por lo tanto, seleccionaremos esas opciones para poder adquirir el video



A.4. CONEXIÓN ARDUINO-SIMULINK



Legacy MATLAB and Simulink Support for Arduino

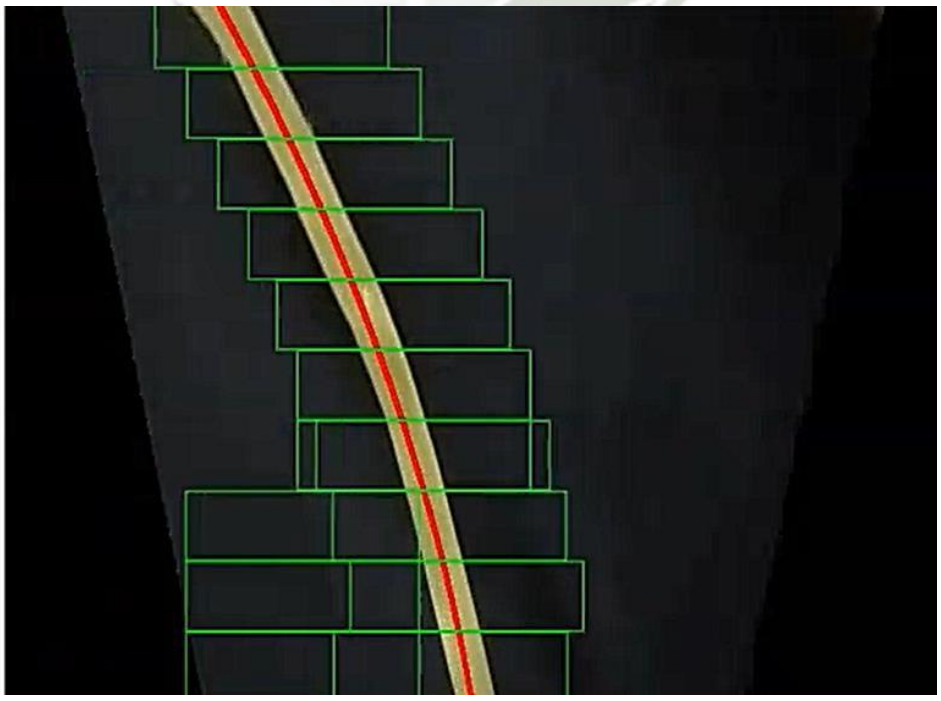
ANEXO B:

CÓDIGO PARA UTZAR AL ARDUINO

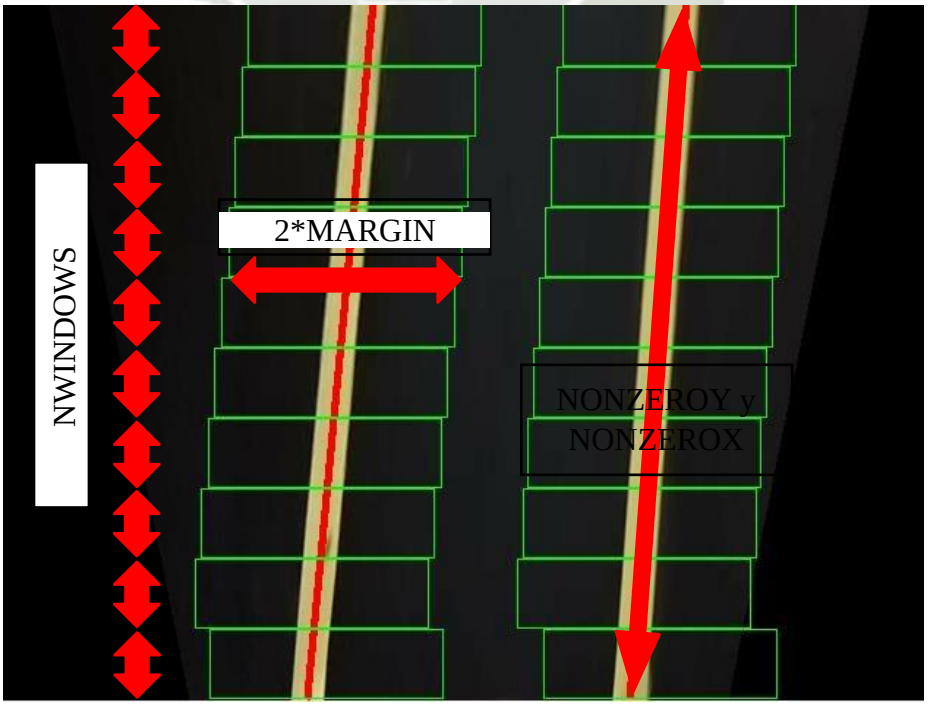
```
h = size(binary_image,1) %Halla la altura de la imagen
                          que ingresa
w = size(binary_image,2) %Halla el ancho de la imagen
                          que ingresa

mid_h = w/2 %Valor referencia para mediciones

base = sum (binary_image(mid_h:end,:)) %Halla la suma de
la matriz definida, desde el valor de referencia anterior
hasta el final, en todo el ancho de la imagen, los
resultados los coloca en un vector fila, se podría
representar de la siguiente manera
```


```
mid_w = w/2 %Valor referencia

[~, locs] = findpeaks(base) %Halla picos en el vector fila
hallada anteriormente, es decir, las zonas blancas
```

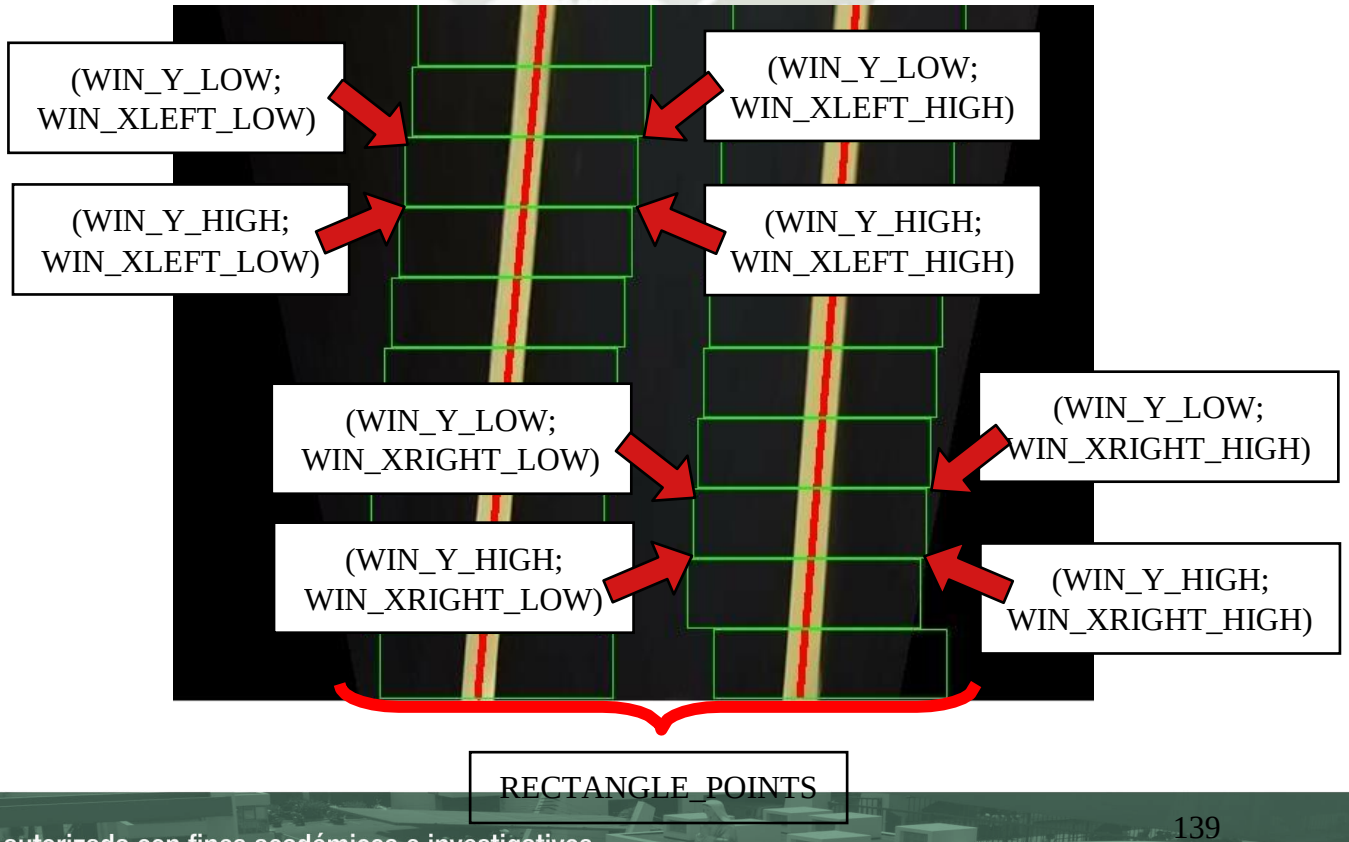
```
left_base = locs(1) %Del vector locs, toma el primer
                    valor


%En esta parte, se explicará el código para reconocer
líneas de carril, aun cuando sólo se llegue a observar
una sola línea de carril, para evitar errores en estos
casos, ejemplo de una sola línea de carril abajo

if any (locs > 100 + left_base) %Esta condición testea si
existen picos después del que estaremos para la izquierda,
a una distancia mínima de 100

   right_base = locs(locs - left_base > 100); %Del vector
   locs, toma todos los valores cuya diferencia entre left
   base y el valor sea mayor a 100

   right_base = right_base(end); %Toma el ultimo valor de
                                 los valores anteriores
   correccion=0; %Como existe valor, no se corrige


else %Cuando no existe valor…
   right_base = left_base; %Se toma la línea derecha
                igual a la izquierda
   correccion=0.0005; %Existe valor de corrección

end
```



```
nwindows = 10 %Será el número de divisiones de toda la
               ventana

window_hight = h/nwindows %Es la cantidad de pixeles que
                           comprende cada división

leftx_current = left_base %Toma el valor del primer pico
                           hallado
rightx_current = right_base %Toma el valor del otro pico

margin = 80 %Será el ancho de los rectángulos desde las
            "bases" a ambos lados
minpix = 40

left_lane_index = [] %Se genera una variable que contendrá
todos los valores de coordenadas de los puntos de la línea
de carril izquierdo
right_lane_index = [] %Se genera una variable que
contendrá todos los valores de coordenadas de los puntos
de la línea de carril derecho

[nonzeroy, nonzerox] = find(binary_image) %Halla todos
los valores 1 (color blanco) de la imagen

rectangle_points = zeros(2 * nwindows,4) %Los valores de
los rectángulos que enmarcarán las líneas de carril
```

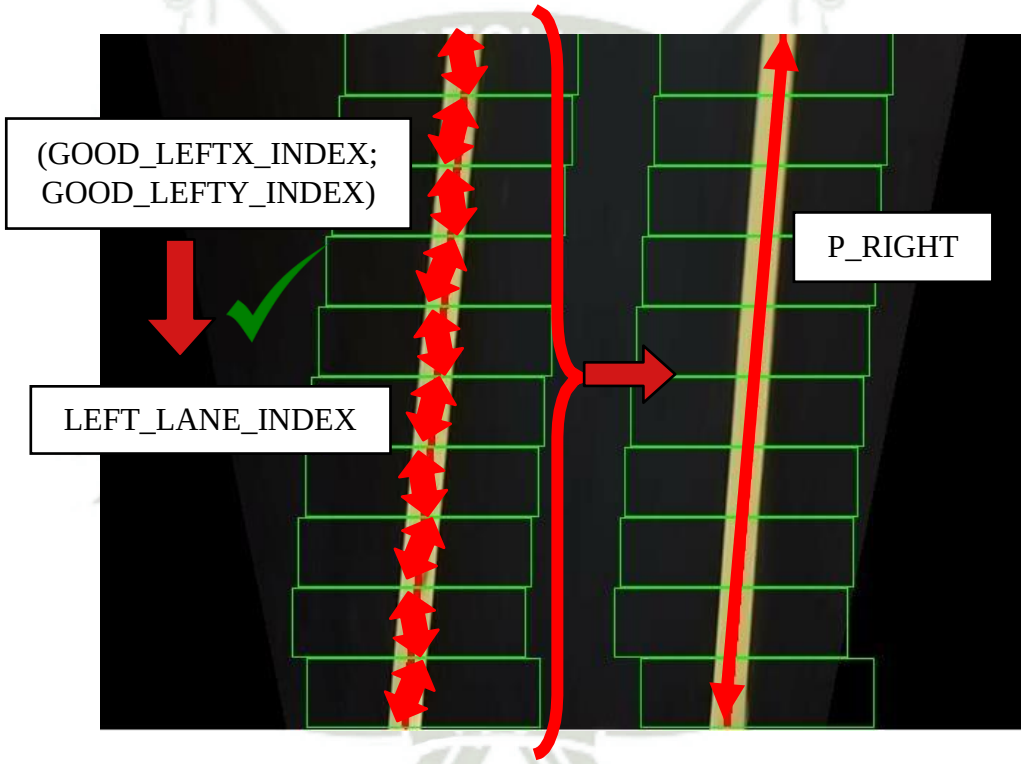


```
for i = 0: nwindows-1 %Bucle para encontrar las líneas de
                      carril

win_y_low = h - (i+1) *window_hight; %Altura superior de
los rectangulos
win_y_high = h - i*window_hight; %Altura inferior de los
rectángulos


win_xleft_low = leftx_current - margin; %Valor X izquierda
de la linea de carril izquierda
win_xleft_high = leftx_current + margin; %Valor X derecha
de la linea de carril izquierda


win_xright_low = rightx_current - margin; %Valor X
izquierda de la linea de carril derecha
win_xright_high = rightx_current + margin; %Valor X
derecha de la linea de carril derecha


rectangle_points(i+1,:) = [win_xleft_low win_y_low
2*margin window_hight]; %Los puntos de rectángulo que
enmarcan el carril izquierdo
rectangle_points(nwindows + 1 + i,:) = [win_xright_low
win_y_low 2*margin window_hight]; %Los puntos de
rectángulo que enmarcan el carril derecho
```



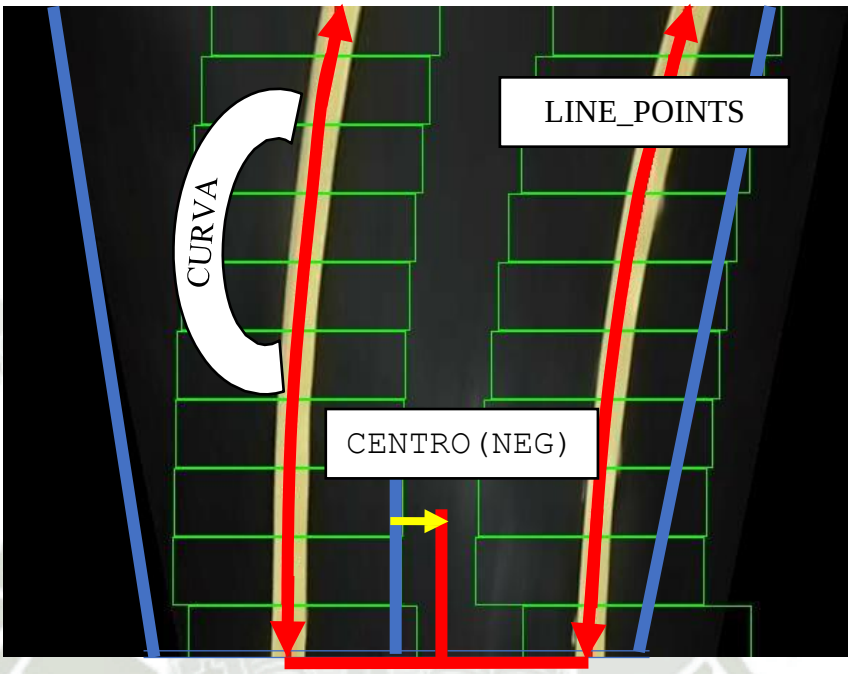
```
%Evalúa todas las coordenadas x que se encuentran en el
rectángulo del carril izquierdo
good_leftx_indexs = nonzerox((nonzeroy >= win_y_low) &
(nonzeroy < win_y_high) & (nonzerox >= win_xleft_low) &
(nonzerox < win_xleft_high));


%Evalúa todas las coordenadas y que se encuentran en el
rectángulo del carril izquierdo
good_lefty_indexs = nonzeroy((nonzeroy >= win_y_low) &
(nonzeroy < win_y_high) & (nonzerox >= win_xleft_low) &
(nonzerox < win_xleft_high));


%Evalúa todas las coordenadas x que se encuentran en el
rectángulo del carril derecho
good_rightx_indexs = nonzerox((nonzeroy >= win_y_low) &
(nonzeroy < win_y_high) & (nonzerox >= win_xright_low) &
(nonzerox < win_xright_high));


%Evalúa todas las coordenadas y que se encuentran en el
rectángulo del carril derecho
good_righty_indexs = nonzeroy((nonzeroy >= win_y_low) &
(nonzeroy < win_y_high) & (nonzerox >= win_xright_low) &
(nonzerox < win_xright_high));


%Concatenar cada resultado anterior dentro de la matriz
de línea decarril derecha o izquierda
left_lane_index = [left_lane_index; [good_leftx_indexs
good_lefty_indexs]];
right_lane_index = [right_lane_index; [good_rightx_indexs
good_righty_indexs]];


%Evalúa si se llega a un valor mínimo de valores, luego
determina el promedio de todos los valores que cumplen
las condiciones de estar dentro del rectángulo, para el
carril izquierdo y derecho

        if(numel(good_leftx_indexs) > minpix)
            leftx_current = mean(good_leftx_indexs);
        end
        if(numel(good_rightx_indexs) > minpix)
            rightx_current = mean(good_rightx_indexs);
        end
end
```

```
%Hace regresión cuadrática de los valores hallados en
línea izquierda
p_left                                                  =
polyfit(left_lane_index(:,2),left_lane_index(:,1),2);


%Hace regresión cuadrática de los valores hallados en
línea derecha
p_right                                                 =
polyfit(right_lane_index(:,2),right_lane_index(:,1),2);

y = 0:1:h;
```




```
%Iniciamos con las condiciones para saber si la regresión
debe ser lineal o cuadrática, de acuerdo con el valor del
coeficiente de grado 2, luego hallamos las salidas de
puntos de curva, vértices de rectángulos, pendiente =0 y
curva de acuerdo con el coeficiente de grado 2


if ([(p_left (1)>=0.00015) || (p_right (1)>=0.00015)] ||
[(p_left (1)<=-0.00015) || (p_right (1)<=-0.00015)])
xleft = polyval(p_left,y);
xright = polyval(p_right,y);

left_points = [xleft; y];
left_points = left_points(:)';
```

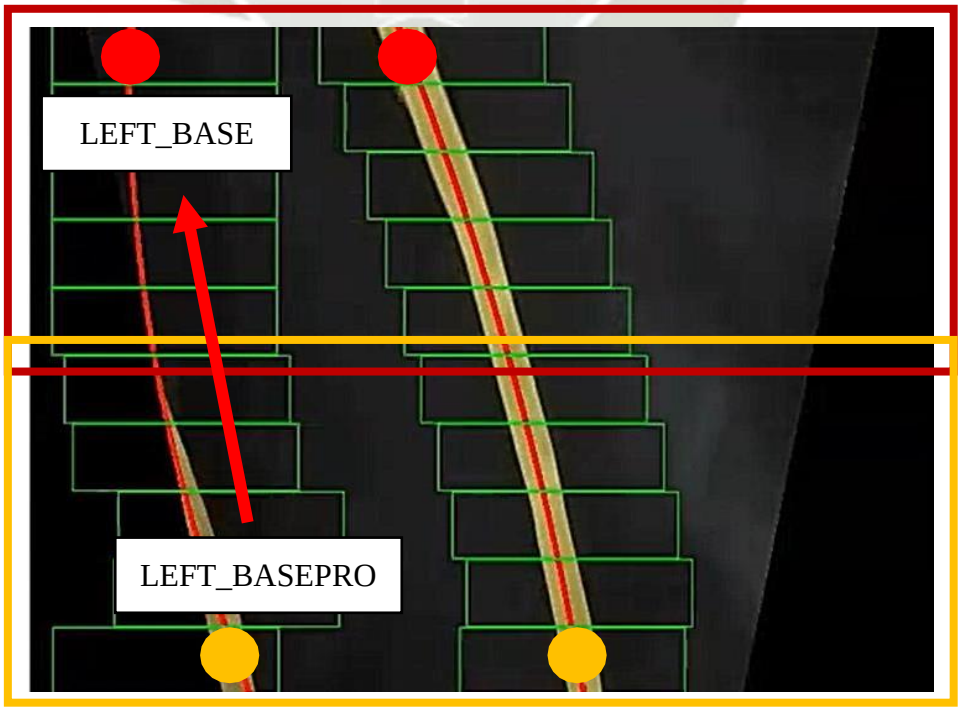
```
right_points = [xright; y];
right_points = right_points(:)';

line_points = [left_points;right_points];
centro = mid_w - (xleft(end)+xright(end))/2;
pendiente=0;
```



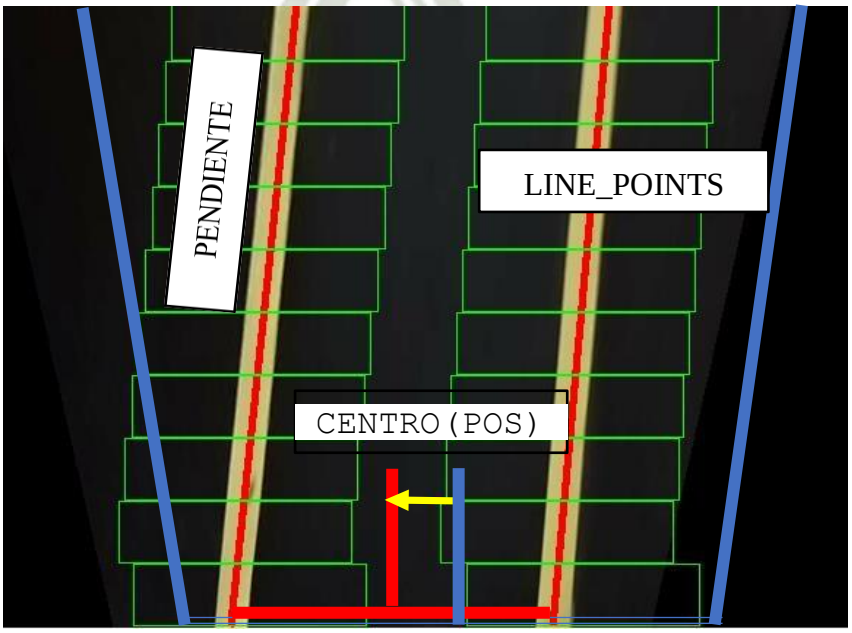

```
%Ahora se explicará la asignación de la salida "curva"
cuando nos encontramos en tramo curvo

%En el primer caso, cuando los signos de las líneas de
curva son iguales, entonces la salida de curva solo sería
el promedio de los 2 valores de curva.
En caso contrario se analiza si la línea de la izquierda
va "hacia la derecha" o "hacia la izquierda", y de acuerdo
con ello se toma el valor de curva correcto.
Esta parte corrige cuando las señales de curva se
contradicen


    if (sign(p_left(1)) == sign(p_right(1)))
    curva=((p_left (1)+ p_right
    (1))/2)+correccion*sign(p_left(1));

    else
        basepro = sum(binary_image(1:150,:));
        [~,locs] = findpeaks(basepro);
        left_basepro = locs(1);
```

```
    sig1=sign(p_left (1));

  if (left_basepro-left_base)>0
       if sig1>0
        curva=p_left (1);
       else
        curva=p_right (1);
       end
  else
       if 0>sig1
        curva=p_left (1);
       else
        curva=p_right (1);
       end
   end
```

```
%En el caso que solo se reconozca una sola línea de carril
curvo, se le agrega un valor de error adicional, para que
ayude a recuperar el centro de carril

    if correccion==0
        centro = mid_w - (xleft(end)+xright(end))/2;
    else
        centro =-70*sign(p_left(1));
    end
```



```
%En caso el coeficiente de grado 2 no cumpla el valor
mínimo para ser función cuadrática, volvemos a hacer
regresión, solo que ahora linealizamos la regresión y
luego obtenemos las salidas

else
p_left =
polyfit(left_lane_index(:,2),left_lane_index(:,1),1);
p_right =
polyfit(right_lane_index(:,2),right_lane_index(:,1),1);
xleft = polyval(p_left,y);
xright = polyval(p_right,y);
left_points = [xleft; y];
left_points = left_points(:)';
right_points = [xright; y];
right_points = right_points(:)';
line_points = [left_points;right_points];

%De la misma forma que en el tramo curvo, en tramo recto
también se puede perder alguna línea de carril recto, por
lo que también se duplicará el valor del error para
recuperar el centro del carril

if correccion==0.0005
    centro = 2*(mid_w - (xleft(end)+xright(end))/2);
else
    centro = mid_w - (xleft(end)+xright(end))/2;
end
curva=0;
pendiente=(p_left (1)+ p_right (1))/2;
end
```



ANEXO C:

CÓDIGO PARA UTILIZAR AL ARDUINO

Para mayor información respecto al acondicionamiento del Arduino con Matlab, se puede observar el siguiente video (Arduino como esclavo)

https://www.youtube.com/watch?v=1LIRKfYU1Ug

Las librerías que funcionarán con el código presentado, se pueden descargar de la siguiente dirección Web:

https://la.mathworks.com/matlabcentral/fileexchange/32374-legacy-matlab-and-simulink-support-for-arduino?s_cid=srchtitle

CÓDIGO IDE

```
/* Analog and Digital Input and Output Server for MATLAB     */
/* Giampiero Campa, Copyright 2012 The MathWorks, Inc        */

/* This file is meant to be used with the MATLAB arduino IO
   package, however, it can be used from the IDE environment
   (or any other serial terminal) by typing commands like:

   0e0 : assigns digital pin #4 (e) as input
   0f1 : assigns digital pin #5 (f) as output
   0n1 : assigns digital pin #13 (n) as output

   1c : reads digital pin #2 (c)
   1e  : reads digital pin #4 (e)
   2n0 : sets digital pin #13 (n) low
   2n1 : sets digital pin #13 (n) high
   2f1 : sets digital pin #5 (f) high
   2f0 : sets digital pin #5 (f) low
   4j2 : sets digital pin #9 (j) to 50=ascii(2) over 255
   4jz : sets digital pin #9 (j) to 122=ascii(z) over 255
```

```
  3a : reads analog pin #0 (a)
  3f  : reads analog pin #5 (f)

  5j    : reads status (attached/detached) of servo on pin #9
  5k    : reads status (attached/detached) of servo on pin #10
  6j1  : attaches servo on pin #9
  8jz : moves servo on pin #9 of 122 degrees (122=ascii(z))
  7j    : reads angle of servo on pin #9
  6j0  : detaches servo on pin #9

  E0cd : attaches encoder #0 (0) on pins 2 (c) and 3 (d)
  E1st : attaches encoder #1 on pins 18 (s) and 19 (t)
  E2vu : attaches encoder #2 on pins 21 (v) and 20 (u)
  G0    : gets 0 position of encoder #0
  I0u : sets debounce delay to 20 (2ms) for encoder #0
  H1    : resets position of encoder #1
  F2    : detaches encoder #2

  R0    : sets analog reference to DEFAULT
  R1    : sets analog reference to INTERNAL
  R2    : sets analog reference to EXTERNAL

  X3    : roundtrip example case returning the input (ascii(3))
  99    : returns script type (0 adio.pde ... 3 motor.pde ) */

#include <Servo.h>

/* define internal for the MEGA as 1.1V (as for the 328) */
#if defined(_AVR_ATmega1280_) || defined(_AVR_ATmega2560_)
#define INTERNAL INTERNAL1V1
#endif

/* define encoder structure                              */
typedef struct { int pinA; int pinB; int pos; int del;} Encoder;
```

```
volatile Encoder Enc[3] = {{0,0,0,0}, {0,0,0,0}, {0,0,0,0}};

/* create servo vector                                  */
Servo servo[70];

void setup() {
 /* initialize serial                                   */
 Serial.begin(115200);
}


void loop() {

 /* variables declaration and initialization            */

 static int   s  = -1;    /* state                      */
 static int  pin = 13;    /* generic pin number         */
 static int  enc = 0;     /* generic encoder number     */

 int  val =  0;          /* generic value read from serial */
 int  agv =  0;          /* generic analog value       */
 int  dgv = 0;           /* generic digital value      */


 /* The following instruction constantly checks if anything
    is available on the serial port. Nothing gets executed in
    the loop if nothing is available to be read, but as soon
    as anything becomes available, then the part coded after
    the if statement (that is the real stuff) gets executed */

 if (Serial.available() >0) {

   /* whatever is available from the serial is read here    */
   val = Serial.read();
```

```
/* This part basically implements a state machine that
   reads the serial port and makes just one transition
   to a new state, depending on both the previous state
   and the command that is read from the serial port.
   Some commands need additional inputs from the serial
   port, so they need 2 or 3 state transitions (each one
   happening as soon as anything new is available from
   the serial port) to be fully executed. After a command
   is fully executed the state returns to its initial
   value s=-1                                    */

switch (s) {


  /* s=-1 means NOTHING RECEIVED YET ******************* */
  case -1:

  /* calculate next state                          */
  if (val>47 && val<90) {
  /* the first received value indicates the mode
      49 is ascii for 1, ... 90 is ascii for Z
      s=0 is change-pin mode;
      s=10 is DI;  s=20 is DO;  s=30 is AI;  s=40 is AO;
      s=50 is servo status; s=60 is aervo attach/detach;
      s=70 is servo read;  s=80 is servo write;
      s=90 is query script type (1 basic, 2 motor);
      s=210 is encoder attach; s=220 is encoder detach;
      s=230 is get encoder position; s=240 is encoder reset;
      s=250 is set encoder debounce delay;
      s=340 is change analog reference;
      s=400 example echo returning the input argument;
                                                  */
    s=10*(val-48);
```

```
}

/* the following statements are needed to handle
   unexpected first values coming from the serial (if
   the value is unrecognized then it defaults to s=-1) */
if ((s>90 && s<210) || (s>250 && s!=340 && s!=400)) {
  s=-1;
}

/* the break statements gets out of the switch-case, so
/* we go back and wait for new serial data          */
break; /* s=-1 (initial state) taken care of        */



/* s=0 or 1 means CHANGE PIN MODE                    */

case 0:
/* the second received value indicates the pin
   from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
if (val>98 && val<167) {
  pin=val-97;              /* calculate pin         */
  s=1; /* next we will need to get 0 or 1 from serial  */
}
else {
  s=-1; /* if value is not a pin then return to -1     */
}
break; /* s=0 taken care of                          */



case 1:
/* the third received value indicates the value 0 or 1 */
if (val>47 && val<50) {
  /* set pin mode                                    */
```

```
  if (val==48) {
    pinMode(pin,INPUT);
  }
  else {
    pinMode(pin,OUTPUT);
  }
}
s=-1;  /* we are done with CHANGE PIN so go to -1      */
break; /* s=1 taken care of                            */



/* s=10 means DIGITAL INPUT ************************** */

case 10:
/* the second received value indicates the pin
   from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
if (val>98 && val<167) {
  pin=val-97;              /* calculate pin          */
  dgv=digitalRead(pin);    /* perform Digital Input */
  Serial.println(dgv);     /* send value via serial  */
}
s=-1;  /* we are done with DI so next state is -1      */
break; /* s=10 taken care of                           */



/* s=20 or 21 means DIGITAL OUTPUT ******************* */

case 20:
/* the second received value indicates the pin
   from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
if (val>98 && val<167) {
  pin=val-97;              /* calculate pin          */
```

```
  s=21; /* next we will need to get 0 or 1 from serial */
 }
 else {
  s=-1; /* if value is not a pin then return to -1     */
 }
 break; /* s=20 taken care of                          */

 case 21:
 /* the third received value indicates the value 0 or 1 */
 if (val>47 && val<50) {
  dgv=val-48;              /* calculate value       */
digitalWrite(pin,dgv);     /* perform Digital Output */
 }
 s=-1;  /* we are done with DO so next state is -1      */
 break; /* s=21 taken care of                          */



 /* s=30 means ANALOG INPUT ************************** */

 case 30:
 /* the second received value indicates the pin
    from abs('a')=97, pin 0, to abs('p')=112, pin 15    */
 if (val>96 && val<113) {
  pin=val-97;              /* calculate pin         */
  agv=analogRead(pin);       /* perform Analog Input  */
Serial.println(agv);       /* send value via serial  */
 }
 s=-1;  /* we are done with AI so next state is -1      */
 break; /* s=30 taken care of                          */



 /* s=40 or 41 means ANALOG OUTPUT ******************** */
```

```
case 40:
/* the second received value indicates the pin
   from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
if (val>98 && val<167) {
  pin=val-97;              /* calculate pin          */
  s=41; /* next we will need to get value from serial  */
}
else {
  s=-1; /* if value is not a pin then return to -1     */
}
break; /* s=40 taken care of                          */


case 41:
/* the third received value indicates the analog value */
analogWrite(pin,val);       /* perform Analog Output  */
s=-1;  /* we are done with AO so next state is -1      */
break; /* s=41 taken care of                          */



/* s=50 means SERVO STATUS (ATTACHED/DETACHED) ******* */

case 50:
/* the second value indicates the servo attachment pin
   from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
if (val>98 && val<167) {
  pin=val-97;              /* calculate pin          */
  dgv=servo[pin].attached();          /* read status */
  Serial.println(dgv);       /* send value via serial  */
}
s=-1; /* we are done with servo status so return to -1*/
break; /* s=50 taken care of                          */
```

```
/* s=60 or 61 means SERVO ATTACH/DETACH ********** */

case 60:
/* the second value indicates the servo attachment pin
   from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
if (val>98 && val<167) {
  pin=val-97;               /* calculate pin         */
  s=61; /* next we will need to get 0 or 1 from serial */
}
else {
  s=-1; /* if value is not a servo then return to -1  */
}
break; /* s=60 taken care of                          */


case 61:
/* the third received value indicates the value 0 or 1
   0 for detach and 1 for attach                       */
if (val>47 && val<50) {
  dgv=val-48;               /* calculate value       */
  if (dgv) servo[pin].attach(pin);     /* attach servo */
  else servo[pin].detach();           /* detach servo */
}
s=-1; /* we are done with servo attach/detach so -1 */
break; /* s=61 taken care of                          */



/* s=70 means SERVO READ ****************************** */

case 70:
```

```
 /* the second value indicates the servo attachment pin
    from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
 if (val>98 && val<167) {
   pin=val-97;              /* calculate pin         */
   agv=servo[pin].read();   /* read value            */
Serial.println(agv);       /* send value via serial  */
 }
 s=-1; /* we are done with servo read so go to -1 next */
 break; /* s=70 taken care of                        */



 /* s=80 or 81 means SERVO WRITE ********************* */

 case 80:
 /* the second value indicates the servo attachment pin
    from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
 if (val>98 && val<167) {
   pin=val-97;              /* calculate pin         */
   s=81; /* next we will need to get value from serial  */
 }
 else {
   s=-1; /* if value is not a servo then return to -1  */
 }
 break; /* s=80 taken care of                        */



 case 81:
 /* the third received value indicates the servo angle  */
 servo[pin].write(val);              /* write value */
 s=-1; /* we are done with servo write so go to -1 next*/
 break; /* s=81 taken care of                        */
```

```
/* s=90 means Query Script Type:
   (0 adio, 1 adioenc, 2 adiosrv, 3 motor)            */

case 90:
if (val==57) {
  /* if string sent is 99  send script type via serial */
  Serial.println(2);
}
s=-1;  /* we are done with this so next state is -1    */
break; /* s=90 taken care of                          */



/* s=210 to 212 means ENCODER ATTACH ****************** */

case 210:
/* the second value indicates the encoder number:
   either 0, 1 or 2                                   */
if (val>47 && val<51) {
  enc=val-48;        /* calculate encoder number       */
  s=211;  /* next we need the first attachment pin     */
}
else {
  s=-1; /* if value is not an encoder then return to -1*/
}
break; /* s=210 taken care of                         */



case 211:
/* the third received value indicates the first pin
   from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
if (val>98 && val<167) {
  pin=val-97;              /* calculate pin          */
```

```
  Enc[enc].pinA=pin;        /* set pin A             */
  s=212;  /* next we need the second attachment pin    */
}
else {
  s=-1; /* if value is not a servo then return to -1  */
}
break; /* s=211 taken care of                         */


case 212:
/* the fourth received value indicates the second pin
   from abs('c')=99, pin 2, to abs('¦')=166, pin 69    */
if (val>98 && val<167) {
  pin=val-97;             /* calculate pin          */
  Enc[enc].pinB=pin;        /* set pin B            */

  /* set encoder pins as inputs                      */
  pinMode(Enc[enc].pinA, INPUT);
  pinMode(Enc[enc].pinB, INPUT);

  /* turn on pullup resistors                       */
  digitalWrite(Enc[enc].pinA, HIGH);
  digitalWrite(Enc[enc].pinB, HIGH);

  /* attach interrupts                             */
  switch(enc) {
    case 0:
      attachInterrupt(getIntNum(Enc[0].pinA), isrPinAEn0, CHANGE);
      attachInterrupt(getIntNum(Enc[0].pinB), isrPinBEn0, CHANGE);
      break;
    case 1:
      attachInterrupt(getIntNum(Enc[1].pinA), isrPinAEn1, CHANGE);
      attachInterrupt(getIntNum(Enc[1].pinB), isrPinBEn1, CHANGE);
      break;
```

```
    case 2:
      attachInterrupt(getIntNum(Enc[2].pinA), isrPinAEn2, CHANGE);
      attachInterrupt(getIntNum(Enc[2].pinB), isrPinBEn2, CHANGE);
      break;
    }

}
s=-1; /* we are done with encoder attach so -1        */
break; /* s=212 taken care of                        */


/* s=220 means ENCODER DETACH  ********************** */

case 220:
/* the second value indicates the encoder number:
   either 0, 1 or 2                                   */
if (val>47 && val<51) {
  enc=val-48;       /* calculate encoder number      */
  /* detach interrupts */
  detachInterrupt(getIntNum(Enc[enc].pinA));
  detachInterrupt(getIntNum(Enc[enc].pinB));
}
s=-1;  /* we are done with encoder detach so -1       */
break; /* s=220 taken care of                        */


/* s=230 means GET ENCODER POSITION ****************** */

case 230:
/* the second value indicates the encoder number:
   either 0, 1 or 2                                   */
if (val>47 && val<51) {
  enc=val-48;       /* calculate encoder number      */
  /* send the value back                             */
```

```
  Serial.println(Enc[enc].pos);
}
s=-1;  /* we are done with encoder detach so -1        */
break; /* s=230 taken care of                          */


/* s=240 means RESET ENCODER POSITION ***************** */

case 240:
/* the second value indicates the encoder number:
   either 0, 1 or 2                                    */
if (val>47 && val<51) {
  enc=val-48;       /* calculate encoder number       */
  /* reset position                                    */
  Enc[enc].pos=0;
}
s=-1;  /* we are done with encoder detach so -1        */
break; /* s=240 taken care of                          */


/* s=250 and 251 mean SET ENCODER DEBOUNCE DELAY ***** */

case 250:
/* the second value indicates the encoder number:
   either 0, 1 or 2                                    */
if (val>47 && val<51) {
  enc=val-48;       /* calculate encoder number       */
  s=251;  /* next we need the first attachment pin     */
}
else {
  s=-1; /* if value is not an encoder then return to -1*/
}
break; /* s=250 taken care of                          */
```

```
      case 251:
      /* the third received value indicates the debounce
         delay value in units of approximately 0.1 ms each
         from abs('a')=97, 0 units, to abs('¦')=166, 69 units*/
      if (val>96 && val<167) {
        Enc[enc].del=val-97;       /* set debounce delay     */
      }
      s=-1;  /* we are done with this so next state is -1    */
      break; /* s=251 taken care of                          */



      /* s=340 or 341 means ANALOG REFERENCE ********************* */

      case 340:
      /* the second received value indicates the reference,
         which is encoded as is 0,1,2 for DEFAULT, INTERNAL
         and EXTERNAL, respectively. Note that this function
         is ignored for boards not featuring AVR or PIC32    */

#if defined(_AVR_) || defined(_PIC32MX_)

      switch (val) {

        case 48:
        analogReference(DEFAULT);
        break;

        case 49:
        analogReference(INTERNAL);
        break;

        case 50:
```

```
  analogReference(EXTERNAL);
  break;

  default:              /* unrecognized, no action  */
  break;
 }

#endif

 s=-1;  /* we are done with this so next state is -1    */
 break; /* s=341 taken care of                          */



 /* s=400 roundtrip example function (returns the input)*/

 case 400:
 /* the second value (val) can really be anything here  */

 /* This is an auxiliary function that returns the ASCII
    value of its first argument. It is provided as an
    example for people that want to add their own code  */

 /* your own code goes here instead of the serial print */
 Serial.println(val);

 s=-1;  /* we are done with the aux function so -1      */
 break; /* s=400 taken care of                          */



 /* ******* UNRECOGNIZED STATE, go back to s=-1 ******* */

 default:
```

```
      /* we should never get here but if we do it means we
         are in an unexpected state so whatever is the second
         received value we get out of here and back to s=-1  */

      s=-1;  /* go back to the initial state, break unneeded */



    } /* end switch on state s                          */

  } /* end if serial available                         */

} /* end loop statement                                */



/* auxiliary function to handle encoder attachment          */
int getIntNum(int pin) {
/* returns the interrupt number for a given interrupt pin
   see http://arduino.cc/it/Reference/AttachInterrupt       */
switch(pin) {
  case 2:
    return 0;
  case 3:
    return 1;
  case 21:
    return 2;
  case 20:
    return 3;
  case 19:
    return 4;
  case 18:
    return 5;
```

```
  default:
    return -1;
  }
}


/* auxiliary debouncing function                    */
void debounce(int del) {
  int k;
  for (k=0;k<del;k++) {
    /* can't use delay in the ISR so need to waste some time
       perfoming operations, this uses roughly 0.1ms on uno  */
    k = k +0.0 +0.0 -0.0 +3.0 -3.0;
  }
}


/* Interrupt Service Routine: change on pin A for Encoder 0 */
void isrPinAEn0(){

  /* read pin B right away                              */
  int drB = digitalRead(Enc[0].pinB);

  /* possibly wait before reading pin A, then read it        */
  debounce(Enc[0].del);
  int drA = digitalRead(Enc[0].pinA);

  /* this updates the counter                           */
  if (drA == HIGH) {  /* low->high on A? */

    if (drB == LOW) {  /* check pin B */
    Enc[0].pos++;  /* going clockwise: increment         */
    } else {
    Enc[0].pos--;  /* going counterclockwise: decrement  */
```

```
  }

 } else {                    /* must be high to low on A */

  if (drB == HIGH) { /* check pin B */
  Enc[0].pos++;  /* going clockwise: increment         */
  } else {
  Enc[0].pos--;  /* going counterclockwise: decrement  */
  }

 } /* end counter update                                */

} /* end ISR pin A Encoder 0                            */



/* Interrupt Service Routine: change on pin B for Encoder 0 */
void isrPinBEn0(){

 /* read pin A right away                               */
 int drA = digitalRead(Enc[0].pinA);

 /* possibly wait before reading pin B, then read it    */
 debounce(Enc[0].del);
 int drB = digitalRead(Enc[0].pinB);

 /* this updates the counter                            */
 if (drB == HIGH) {  /* low->high on B? */

  if (drA == HIGH) { /* check pin A */
  Enc[0].pos++;  /* going clockwise: increment         */
  } else {
  Enc[0].pos--;  /* going counterclockwise: decrement  */
  }
```

```
  } else {                    /* must be high to low on B */

    if (drA == LOW) {  /* check pin A */
    Enc[0].pos++;  /* going clockwise: increment         */
    } else {
    Enc[0].pos--;  /* going counterclockwise: decrement  */
    }

  } /* end counter update */

} /* end ISR pin B Encoder 0  */


/* Interrupt Service Routine: change on pin A for Encoder 1 */
void isrPinAEn1(){

  /* read pin B right away                              */
  int drB = digitalRead(Enc[1].pinB);

  /* possibly wait before reading pin A, then read it        */
  debounce(Enc[1].del);
  int drA = digitalRead(Enc[1].pinA);

  /* this updates the counter                           */
  if (drA == HIGH) {  /* low->high on A? */

    if (drB == LOW) {  /* check pin B */
    Enc[1].pos++;  /* going clockwise: increment         */
    } else {
    Enc[1].pos--;  /* going counterclockwise: decrement  */
    }

  } else { /* must be high to low on A                    */
```

```
  if (drB == HIGH) { /* check pin B */
   Enc[1].pos++;  /* going clockwise: increment         */
   } else {
   Enc[1].pos--;  /* going counterclockwise: decrement  */
   }

 } /* end counter update                              */

} /* end ISR pin A Encoder 1                          */


/* Interrupt Service Routine: change on pin B for Encoder 1 */
void isrPinBEn1(){

 /* read pin A right away                            */
 int drA = digitalRead(Enc[1].pinA);

 /* possibly wait before reading pin B, then read it  */
 debounce(Enc[1].del);
 int drB = digitalRead(Enc[1].pinB);

 /* this updates the counter                         */
 if (drB == HIGH) {  /* low->high on B? */

  if (drA == HIGH) { /* check pin A */
   Enc[1].pos++;  /* going clockwise: increment         */
   } else {
   Enc[1].pos--;  /* going counterclockwise: decrement  */
   }

 } else { /* must be high to low on B                   */

  if (drA == LOW) {  /* check pin A */
```

```
    Enc[1].pos++;  /* going clockwise: increment         */
    } else {
    Enc[1].pos--;  /* going counterclockwise: decrement  */
    }

  } /* end counter update                               */

} /* end ISR pin B Encoder 1                           */


/* Interrupt Service Routine: change on pin A for Encoder 2 */
void isrPinAEn2(){

  /* read pin B right away                             */
  int drB = digitalRead(Enc[2].pinB);

  /* possibly wait before reading pin A, then read it  */
  debounce(Enc[2].del);
  int drA = digitalRead(Enc[2].pinA);

  /* this updates the counter                          */
  if (drA == HIGH) {  /* low->high on A? */

    if (drB == LOW) {  /* check pin B */
    Enc[2].pos++;  /* going clockwise: increment         */
    } else {
    Enc[2].pos--;  /* going counterclockwise: decrement  */
    }

  } else { /* must be high to low on A                   */

    if (drB == HIGH) { /* check pin B */
    Enc[2].pos++;  /* going clockwise: increment         */
    } else {
```

```
  Enc[2].pos--;  /* going counterclockwise: decrement  */
  }

 } /* end counter update                          */

} /* end ISR pin A Encoder 2                      */


/* Interrupt Service Routine: change on pin B for Encoder 2 */
void isrPinBEn2(){

 /* read pin A right away                         */
 int drA = digitalRead(Enc[2].pinA);

 /* possibly wait before reading pin B, then read it      */
 debounce(Enc[2].del);
 int drB = digitalRead(Enc[2].pinB);

 /* this updates the counter                      */
 if (drB == HIGH) {  /* low->high on B? */

  if (drA == HIGH) { /* check pin A */
  Enc[2].pos++;  /* going clockwise: increment       */
  } else {
  Enc[2].pos--;  /* going counterclockwise: decrement  */
  }

 } else { /* must be high to low on B                 */

  if (drA == LOW) {  /* check pin A */
  Enc[2].pos++;  /* going clockwise: increment       */
  } else {
  Enc[2].pos--;  /* going counterclockwise: decrement  */
  }
```

```
  } /* end counter update                    */

} /* end ISR pin B Encoder 2                  */
```

CONTROL PID EN SIMULACIÓN 3D

I. OBJETIVO GENERAL

- Apreciar el funcionamiento del control PID en una simulación
- Comprender algunos aspectos relacionados a visión artificial
- Comparar el control PID con el control difuso

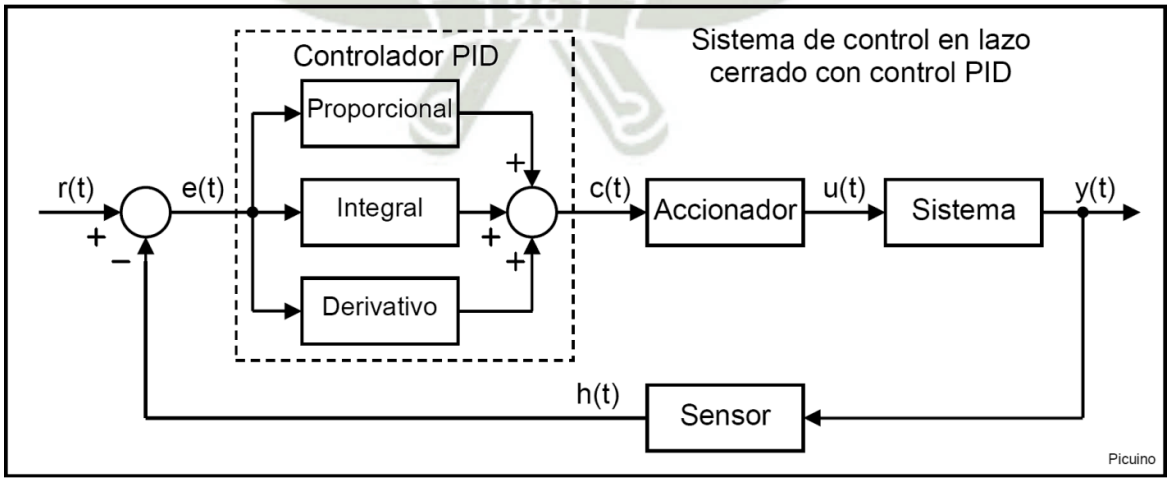
II. MATERIALES Y EQUIPOS

- Computador
- Software Matlab.
- Toolbox Lane_Keeping_Assist_System
- Toolbox Simulink 3D animation

III. MARCO TEÓRICO

CONTROL PID

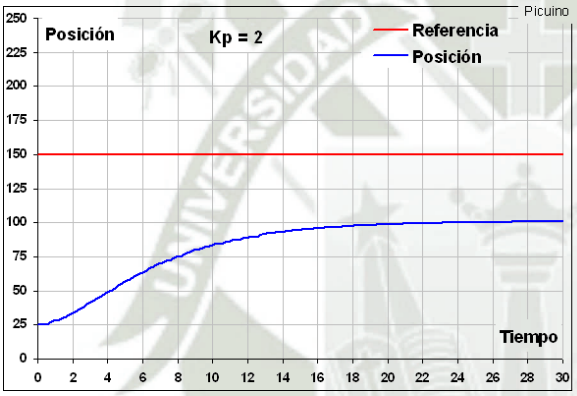
De acuerdo con Katsuhiko, Ogata (2010) los controladores PID permiten controlar un sistema en lazo cerrado para que alcance el "set point" o salida deseada. Este controlador PID está compuesto de tres métodos de control que proporcionan una acción Proporcional, Integral y Derivativa, de donde se deriva el nombre del controlador

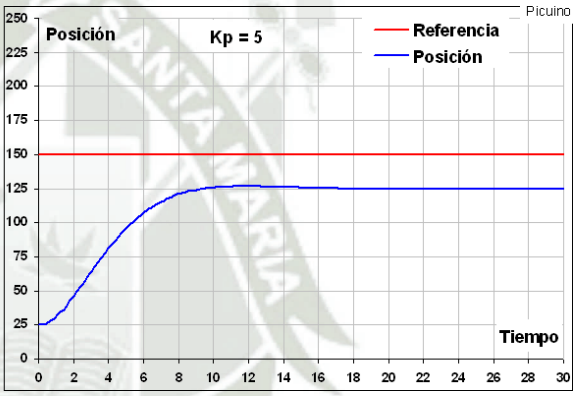


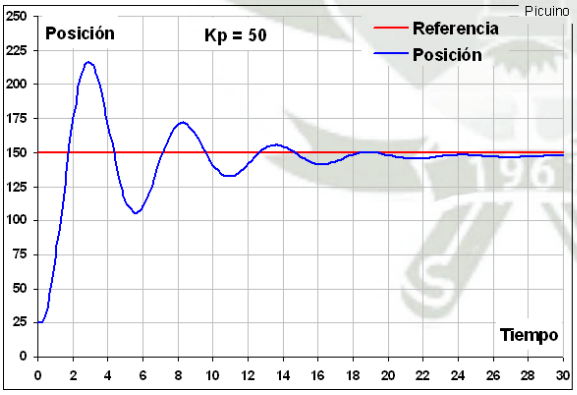
Acción de control Proporcional

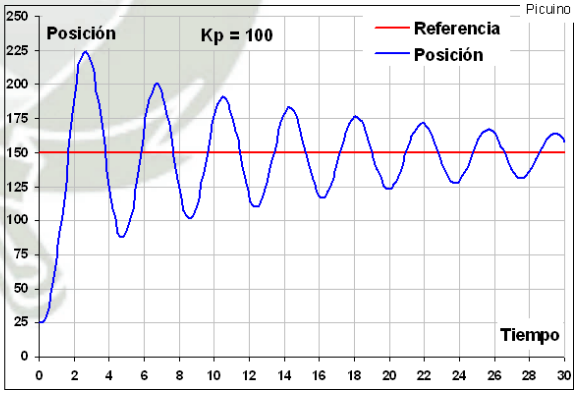
Esta acción de control es proporcional a la señal de error e(t), lo que significa que se multiplica la señal de error por una constante Kp, de modo que se pueda minimizar el error. Mientras se va aumentando el valor Kp se tienen los siguientes efectos, que son visualizados en las gráficas inferiores:

1. Aumenta la velocidad de respuesta del sistema, se llega antes al set point deseado.
2. Disminuye el error del sistema en estado estacionario, con Kp adecuados, aunque no siempre se consigue eliminar el error con la acción proporcional.
3. Aumenta la inestabilidad del sistema, con valores muy altos de Kp.







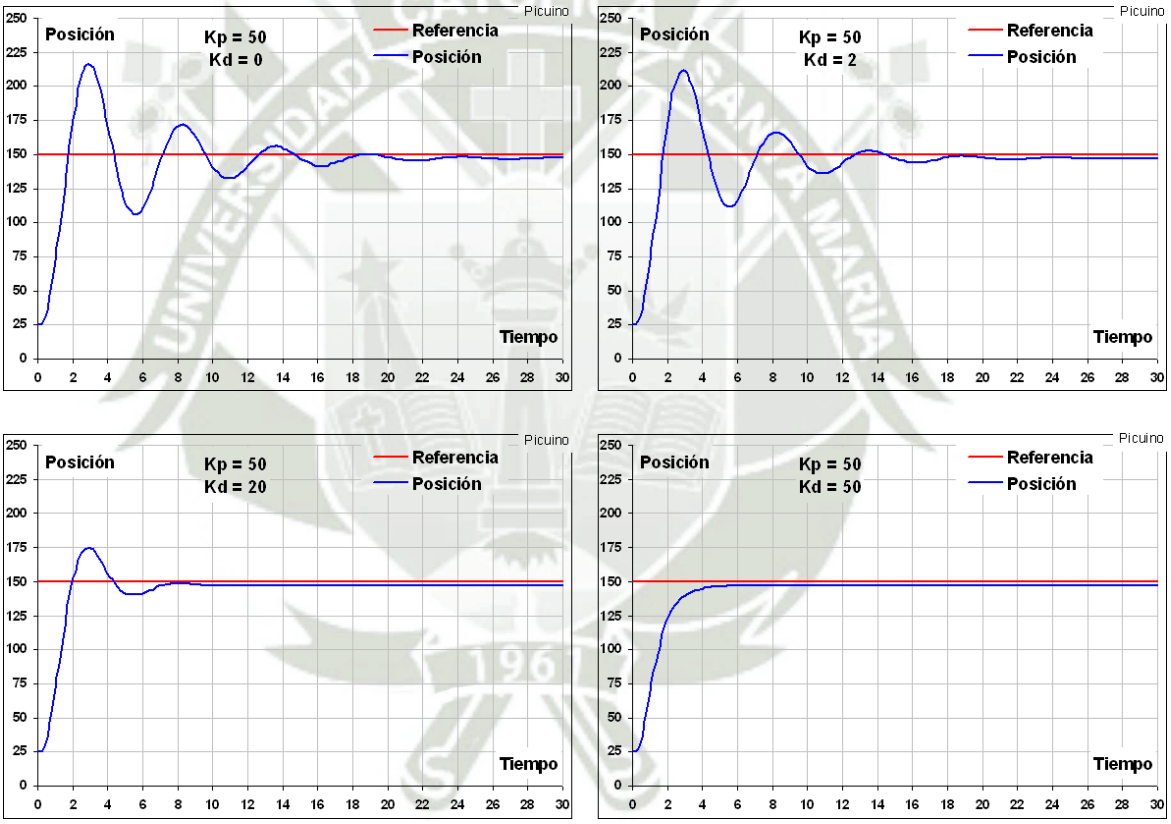


Se observó, por lo tanto, que a medida que el Kp aumenta, el sistema consigue estabilidad a mayor velocidad, pasando de estabilizarse en 20 segundos (Kp=2) a 12-14 segundos (Kp=50), así mismo, se consigue eliminar el error entre Kp=5 y Kp=50 y finalmente con un Kp=100 el sistema se empieza a hacer inestable.

Acción de control Derivativa

Esta acción de control es proporcional a la derivada de la señal de error e(t), es decir se controlará en función a la diferencia de error que haya en el sistema. Mientras se va aumentando el valor Kd se tienen los siguientes efectos, que son visualizados en las gráficas:

1. Aumenta la estabilidad del sistema controlado, sin embargo, con señales muy oscilantes o con frecuencia alta, podría tener más dificultades.
2. Disminuye la velocidad del sistema, sin embargo, no es efecto muy grande.
3. El error en estado estacionario permanecerá igual, por lo que tampoco es una acción definitiva.

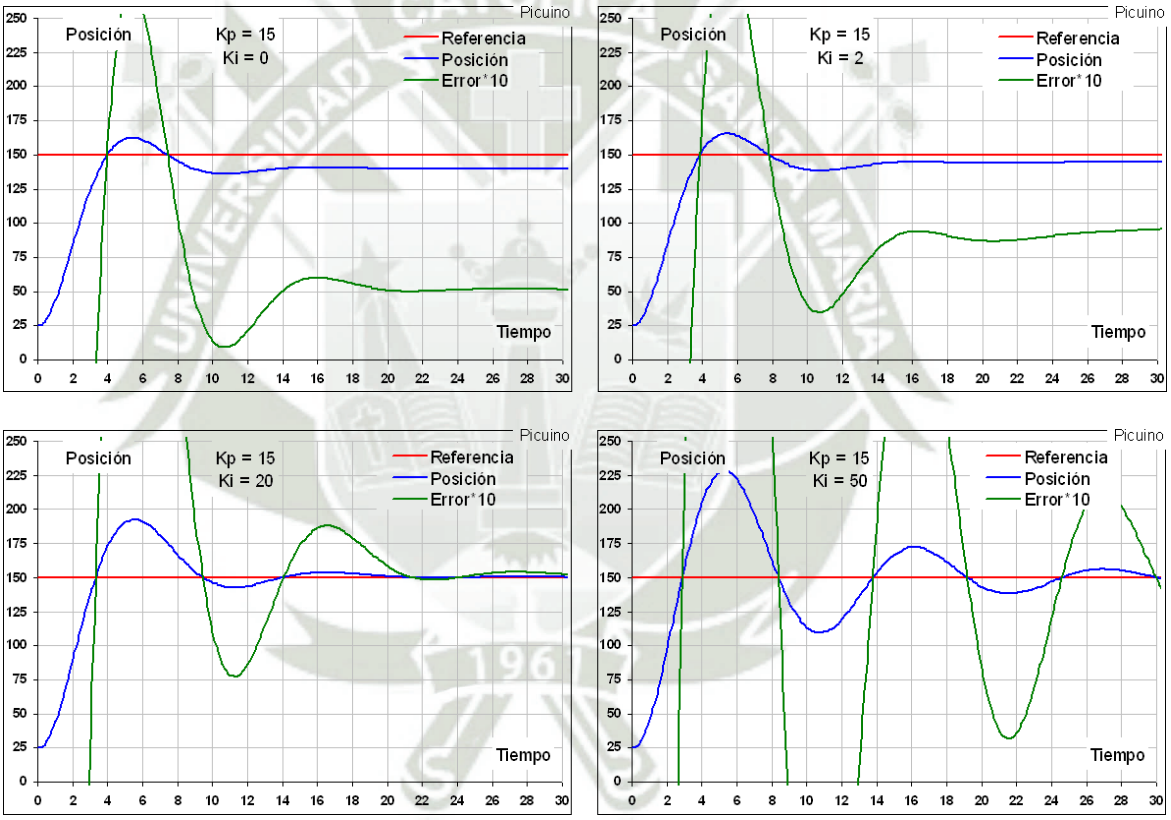


Se observa que conforme se aumenta el Kd, las oscilaciones del sistema van disminuyendo, hasta el punto en que desaparecen con un Kd=50, a su vez, la velocidad con que llegan al valor deseado se disminuye, comparando Kd=0 donde se llega al valor antes de los 2 segundos, mientras que con Kd=50 para llegar al valor demora 2 segundos o más, esto con el fin de "amortiguar" las oscilaciones, además se observa que a pesar de tener diferentes valores de Kd, el error en estado estacionario es el mismo en todos los casos, presentando un pequeño error que no se llega a corregir

Acción de control Integral

Esta acción calcula la integral de la señal de error e(t), es decir, trabaja con la suma o acumulación de errores, por lo que a medida que pasa el tiempo la acción integral es cada vez más grande. Mientras se va aumentando el valor Ki se tienen los siguientes efectos, que son visualizados en las gráficas inferiores:

1. Disminuye el error del sistema en estado estacionario.
2. Aumenta la inestabilidad del sistema, por lo mismo que depende de la suma de errores, creando una especie de inercia en el sistema.
3. Aumenta la velocidad del sistema, aunque no es un efecto muy grande.



De las gráficas anteriores, la señal de color verde has sido desplazada y agrandada para poder observar de mejor manera la variación de la señal de error, se observa que, en todos los casos, esta señal se estabiliza, indicando que el error llega a ser cero, así mismo se comprueban los puntos expuestos anteriormente

Ecuación del controlador

La ecuación del control PID es la siguiente:

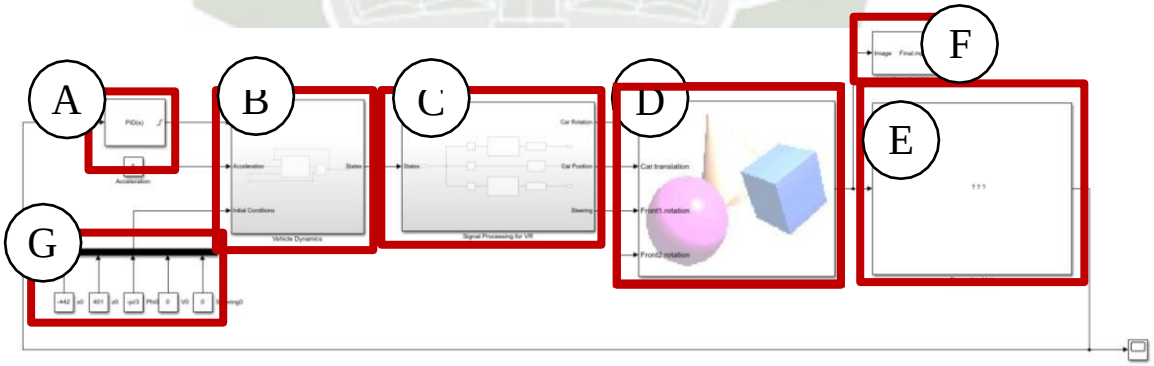
$$c(t) = Kp \times e(t) + Ki \times \int e(t)dt + Kd \times \frac{\partial e(t)}{\partial t}$$

Para:

- c(t) = señal de control
- e(t) = señal de error
- Kp, Ki, Kd = parámetros del controlador PID

SISTEMA SIMULINK DE MANTENIMINETO DE CARRIL

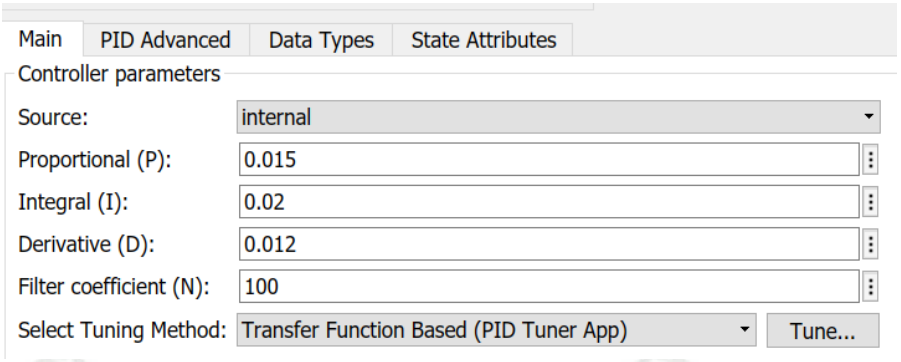
Es necesario conocer algunos aspectos del programa que se va a manejar a continuación. El sistema que se presenta a continuación tendrá la función de simular de acuerdo a las condiciones que se le indiquen inicialmente, así como con los parámetros PID que se desee, para ver el funcionamiento de este tipo de controladores a través de una simulación



Cada una de las partes del bloque elaborado será explicada a continuación

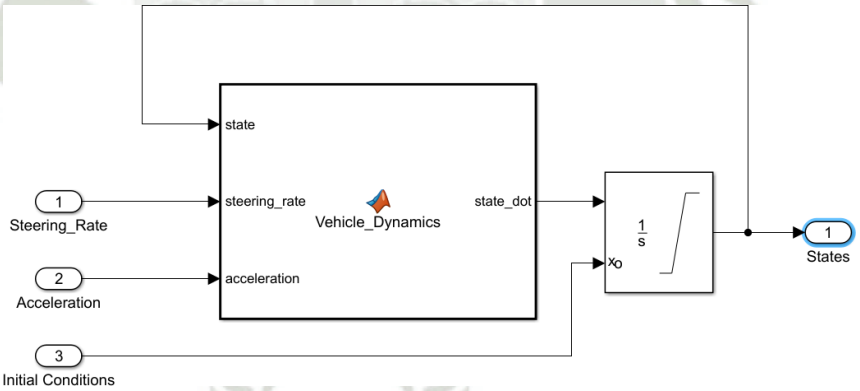
A) BLOQUE PID

Los parámetros utilizados para el control del ángulo de giro de las llantas



B) BLOQUE DE LA DINÁMICA DE VEHÍCULO

Se introducen dentro del bloque de dinámica el ángulo de giro, una aceleración y condiciones iniciales, a la salida del bloque de "dinámicas del vehículo" se obtiene la derivada de estados, por lo que se integra utilizando las condiciones iniciales mencionadas anteriormente, estos resultados se llevarán a un bloque siguiente



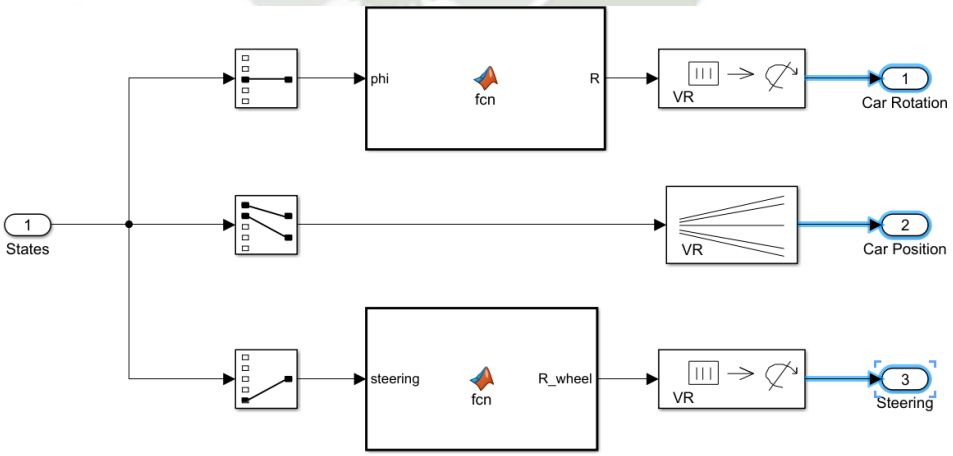
El código dentro del bloque de dinámicas del vehículo se muestra en la parte inferior. Se tienen como entradas las variables mencionadas anteriormente, y se considera que el vehículo tiene una longitud de 4. Para la simulación del vehículo se necesitan algunos parámetros, los cuales son:

- Phi: Orientación del vehículo completo (no de las ruedas)
- V: Velocidad del vehículo
- Steering: Ángulo de giro de las ruedas

```
function state_dot = Vehicle_Dynamics(state,steering_rate,
acceleration)

    L = 4;

    phi = state(3);
    v = state(4);
    steering = state(5);

    x_dot = -v * cos (phi + steering);
    z_dot = v * sin (phi + steering);
    phi_dot = (v/L) * sin(steering);
    v_dot = acceleration;
    steering_dot = steering_rate;
    state_dot = [x_dot;z_dot;phi_dot;v_dot;steering_dot];

end
```

C) BLOQUE DE SEÑALES DE ENTRADA A LA SIMULACIÓN

En la fila superior se llevan los valores de la orientación del vehículo a la simulación, en el medio el desplazamiento a través de la simulación, finalmente en la fila inferior se tiene la señal para las ruedas (este será el ángulo de giro que las ruedas deben tener para corregir la marcha del vehículo)



El código del bloque superior corresponde al ángulo phi, que es el ángulo de orientación del vehículo dentro de la simulación, se usan matrices de rotación, el ángulo de rotación es alrededor del eje Y, la matriz de rotación es parte del código

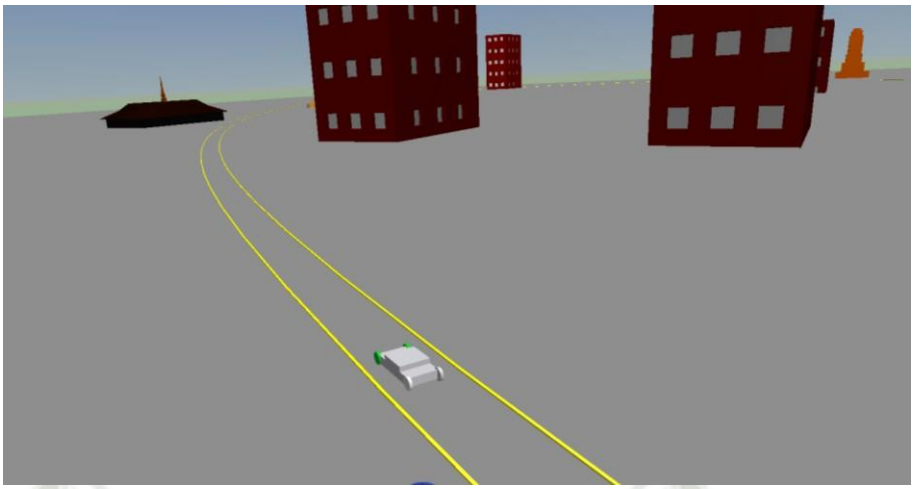
```
function R = fcn(phi)

R = [cos(phi) 0 sin(phi); 0 1 0; -sin(phi) 0 cos(phi)];

end
```

En la fila inferior se tiene el bloque correspondiente al ángulo "steering" que es el ángulo de giro de las llantas.

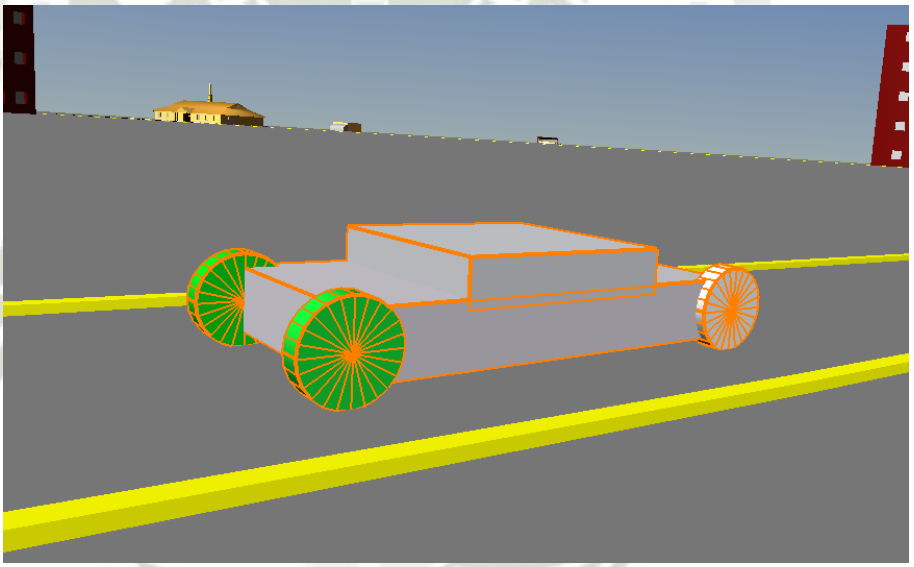
```
function R_wheel = fcn(steering)

theta_x = pi/2;

R1 = [1 0 0; 0 cos(theta_x) -sin(theta_x); 0 sin(theta_x)
cos(theta_x)];
steering = 2 * steering;
R2 = [cos(steering) 0 sin(steering); 0 1 0; -sin(steering)
0 cos(steering)];

R_wheel = R2 * R1;

end
```

D) BLOQUE SIMULACIÓN

Aquí se define el entorno en el cual se desplazará el vehículo, son ayuda del simulador de realidad virtual de Matlab. En este entorno se debe definir la posición y orientación que tendrá el vehículo, las cuales irán variando conforme a lo que indique el controlador difuso



El detalle del carro es el siguiente:

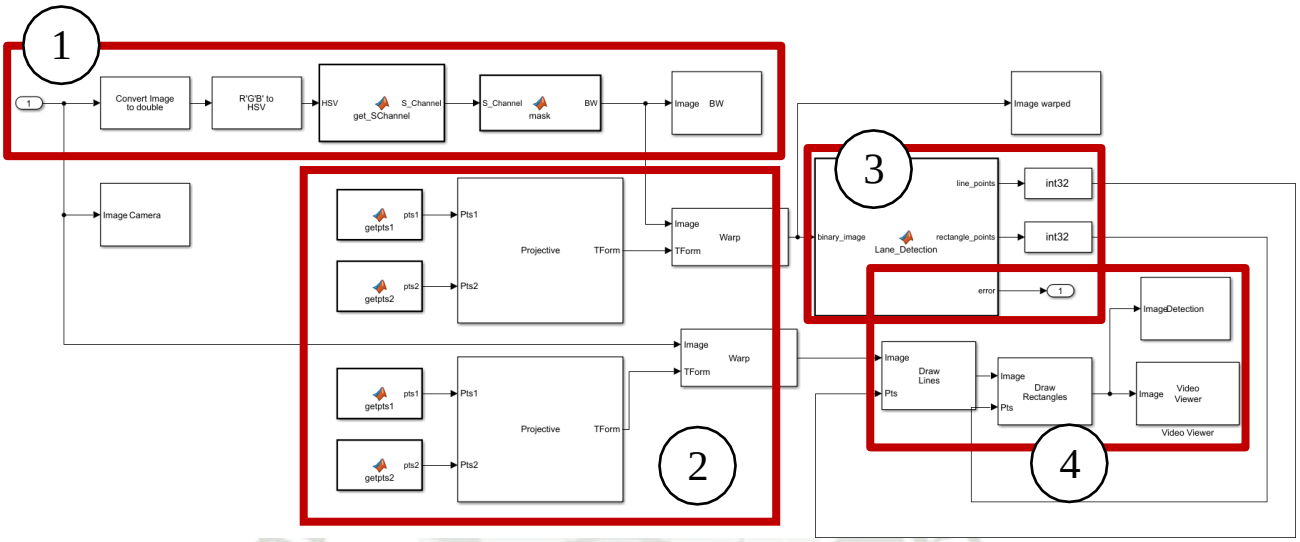


Donde las dimensiones del auto son las siguientes:

- Largo: 4m
- Ancho: 2m
- Altura: 1m
- Diámetro de rueda: 0.4m
- Ancho de rueda: 0.2m

E) BLOQUE PROCESAMIENTO

Aquí se definen las operaciones necesarias para hacer el control de la dirección del vehículo, como se trabaja en simulación, es necesario determinar otras variables adicionales a solo la detección de las líneas, como también son la orientación y avance del vehículo, esto se explicará a continuación:



E.1. Preprocesamiento

En esta primera parte, se recibe la señal que envía la cámara, se la convierte en una matriz con variables “double”, se convierten los colores de RGB a HSV, se filtran los colores para leer solo amarillo y blanco y se obtienen imágenes a blanco y negro al final

E.2. Acondicionamiento de imagen

Se realizan las trasformaciones geométricas necesarias para poder ingresar estas imágenes en el código de identificación de líneas, en este apartado se realiza la eliminación de perspectiva de las líneas de carril y se puede recortar la imagen para evitar errores de lectura

E.3. Procesamiento de la imagen

Aquí se define la transformada de Hough para identificación de líneas y curvas y se obtienen los datos de diferencia del centro del vehículo con el centro de la carretera

E.4. Salidas

Finalmente, los datos obtenidos aquí se mostrarán en una salida de video de la carretera, encerrando en rectángulos las líneas de carril

F) BLOQUE VIDEO

Este bloque tiene la función de guardar lo que muestra la simulación, debido a que el tiempo de procesamiento tarda bastante, para finalmente poder reproducirlo a la velocidad verdadera cuando termine de ejecutarse el código.

G) BLOQUE CONDICIONES INICIALES

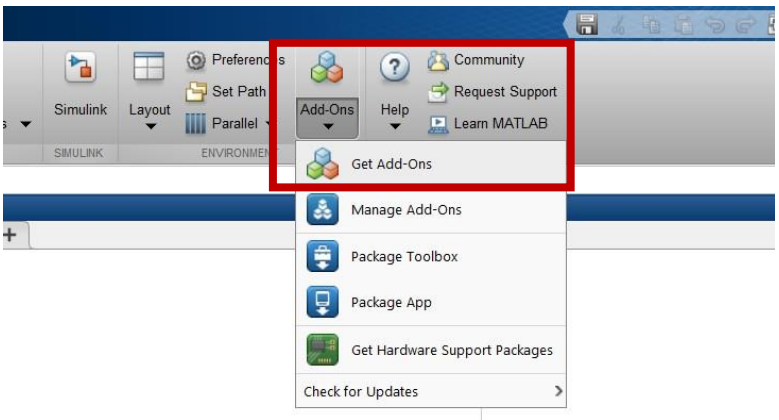
En esta sección se pueden definir las condiciones iniciales a las cuales deseamos poner el vehículo, de modo que se pueda observar una gran perturbación inicial y se vea la efectividad del control PID en las gráficas

IV. PROCEDIMIENTO

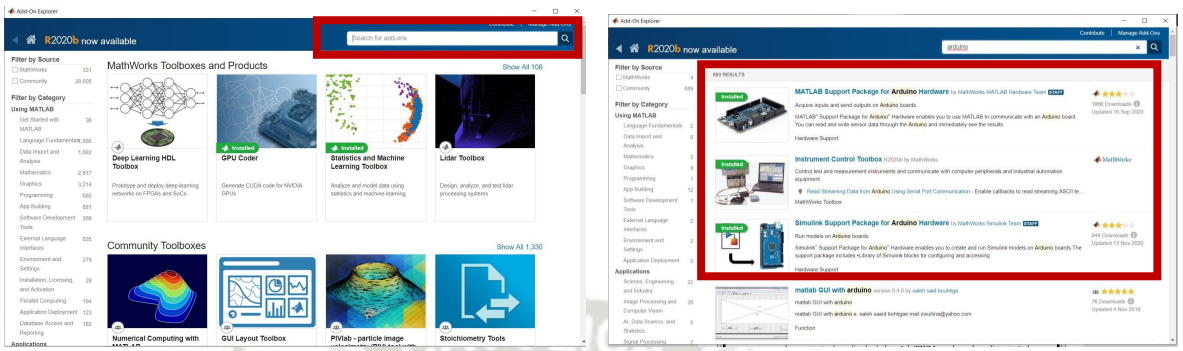
IV.1. INSTALACIÓN DE TOOLBOX

El primer paso que debemos seguir será el de instalar el toolbox Lane_Keeping_Assist_System y el toolbox Simulink 3D animation (que en algunos casos está instalado por defecto, en caso no lo esté, se deberá hacerlo, y si ya está instalado solo instalar el primer toolbox)

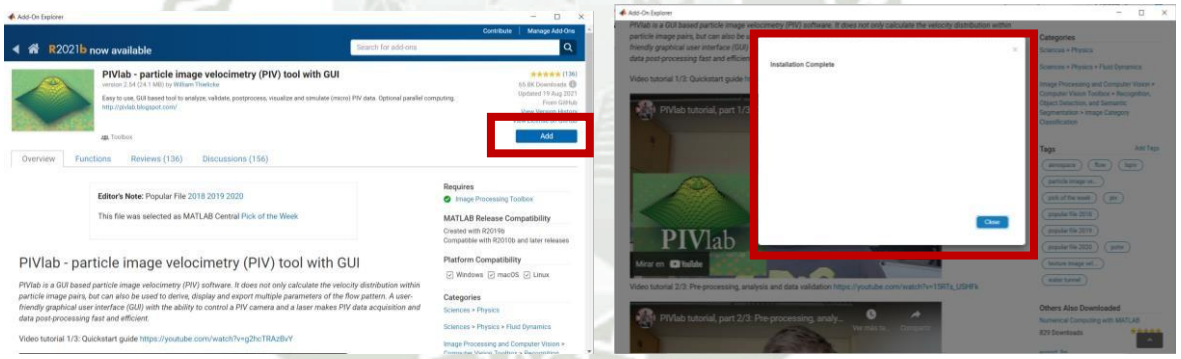
- Se entra a la parte de ADD ONS, luego clic en Get Add-Ons



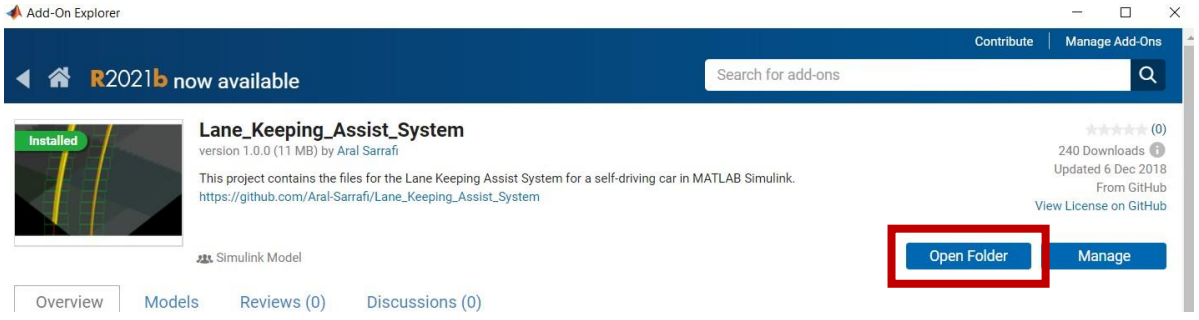
- Usamos los paquetes descargados desde la plataforma de Matlab en internet, en el buscador colocamos el nombre del toolbox que necesitamos, obtendremos a continuación el resultado de la búsqueda realizada



- Finalmente, seleccionamos el toolbox que necesitamos y luego hacemos clic en la opción que nos aparezca, “Add” o “Install”, luego dependiendo del toolbox que se instale se deberán de dar algunos permisos, solo clic en “Aceptar” y siguiente.

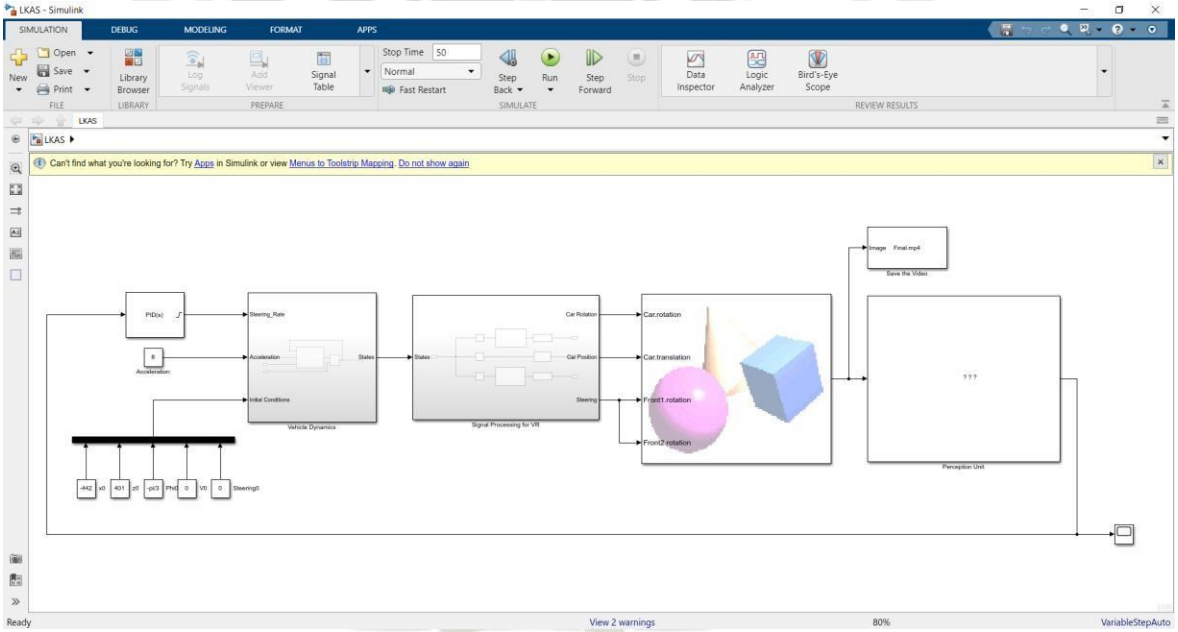


Una vez concluida la parte anterior, debemos hace clic en “Open Folder” en el toolbox Lane Keeping Assist System



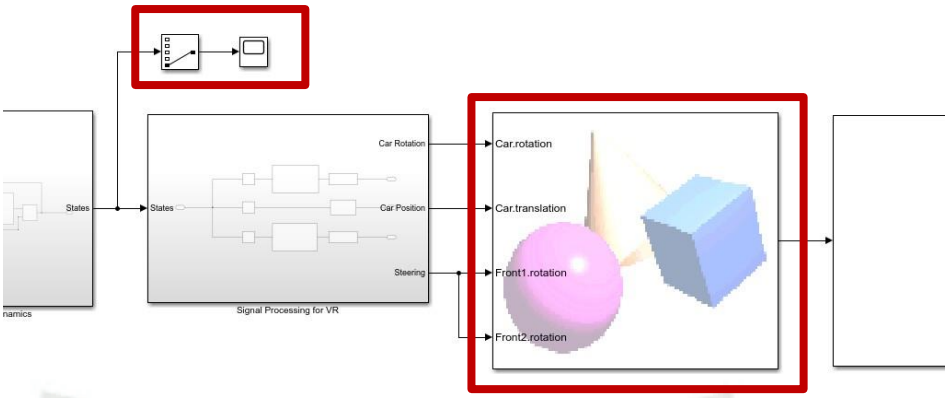
Seguidamente en la ventana “Current Folder” deberían aparecernos las siguientes opciones de la imagen, deberemos hacer clic en “LKAS.slx” que abrirá el Simulink

A continuación, debería de abrirse una ventana similar a la siguiente

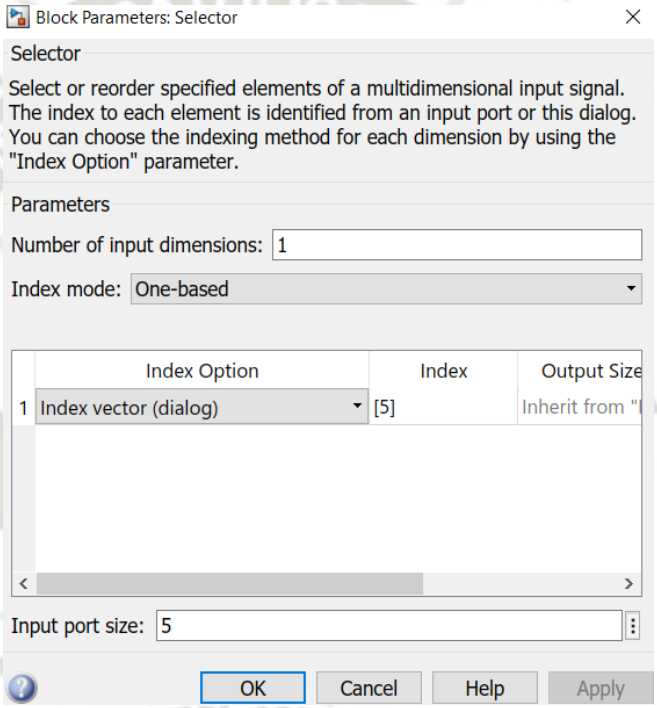


Cada una de las partes que la constituyen fue explicada en la parte del Marco Teórico, ahora debemos de realizar un pequeño cambio para poder observar los parámetros que requerimos

Primero debemos eliminar el bloque “Save the video”, al hacer esto no tendremos referencia de cómo va el carro al momento de iniciar la simulación, así que para poder ver “en vivo” como se mueve el vehículo solo debemos dar doble clic al bloque de simulación 3D, Después deberemos colocar un selector y un scope a la línea de estados, como se ve en la imagen recortada



El selector se debe de configurar de la siguiente manera



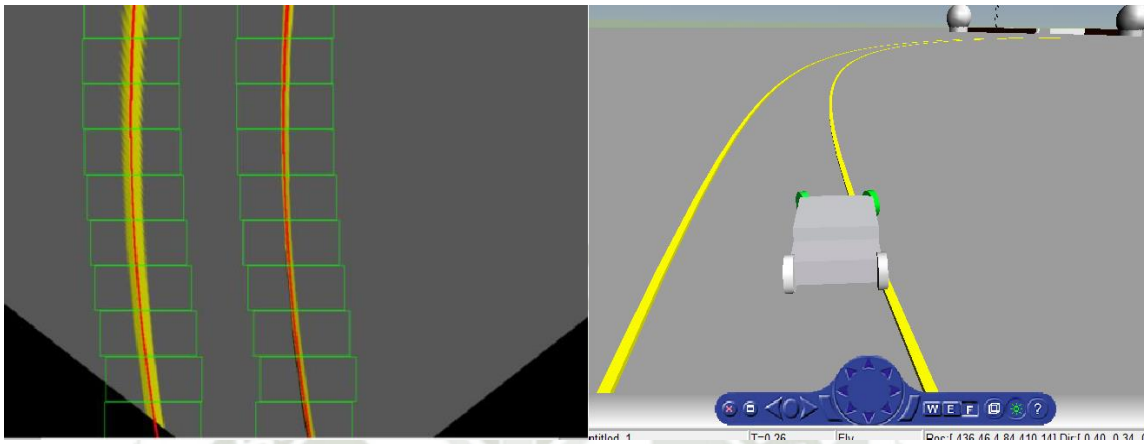
Una vez configurado todo, estamos listos para experimentar

IV.2. FUNCIONAMEINTO DEL SISTEMA

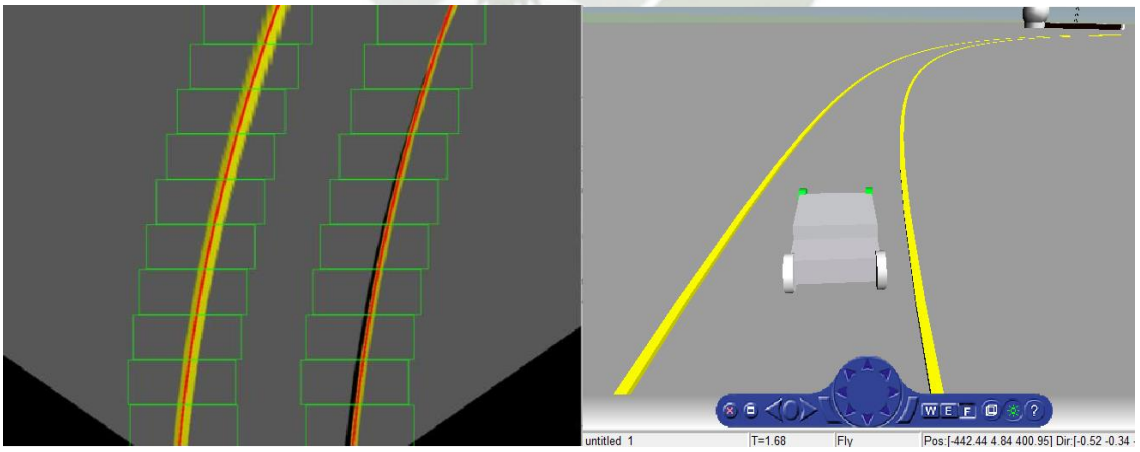
Para el funcionamiento del sistema solo es necesario darle play a la simulación luego de haber efectuado los cambios sugeridos

Es en esta parte que se abrirá una ventana que muestra una vista “vuelo de pájaro” de las líneas de carril, esto en tiempo real conforme a la simulación, estas líneas de carril serán procesadas por los bloques explicados anteriormente, se debería de poder apreciar lo siguiente.

En el primer par de imágenes se observa la posición inicial del vehículo, como se presenta un error de posición grande, entonces el giro de las llantas es grande también, así mismo se observa que en la visión “a vuelo de pájaro” de la izquierda se observan las líneas de carril a un lado de la cámara.



Entonces lo que busca el controlador es conseguir posicionar al vehículo en el centro de la carretera como se observa en el par de imágenes inferior, a su vez, que como el error es menor, entonces el giro que realizan las ruedas también es más pequeño, y en este caso, se observa que las líneas de carril se encuentran centradas en la cámara.

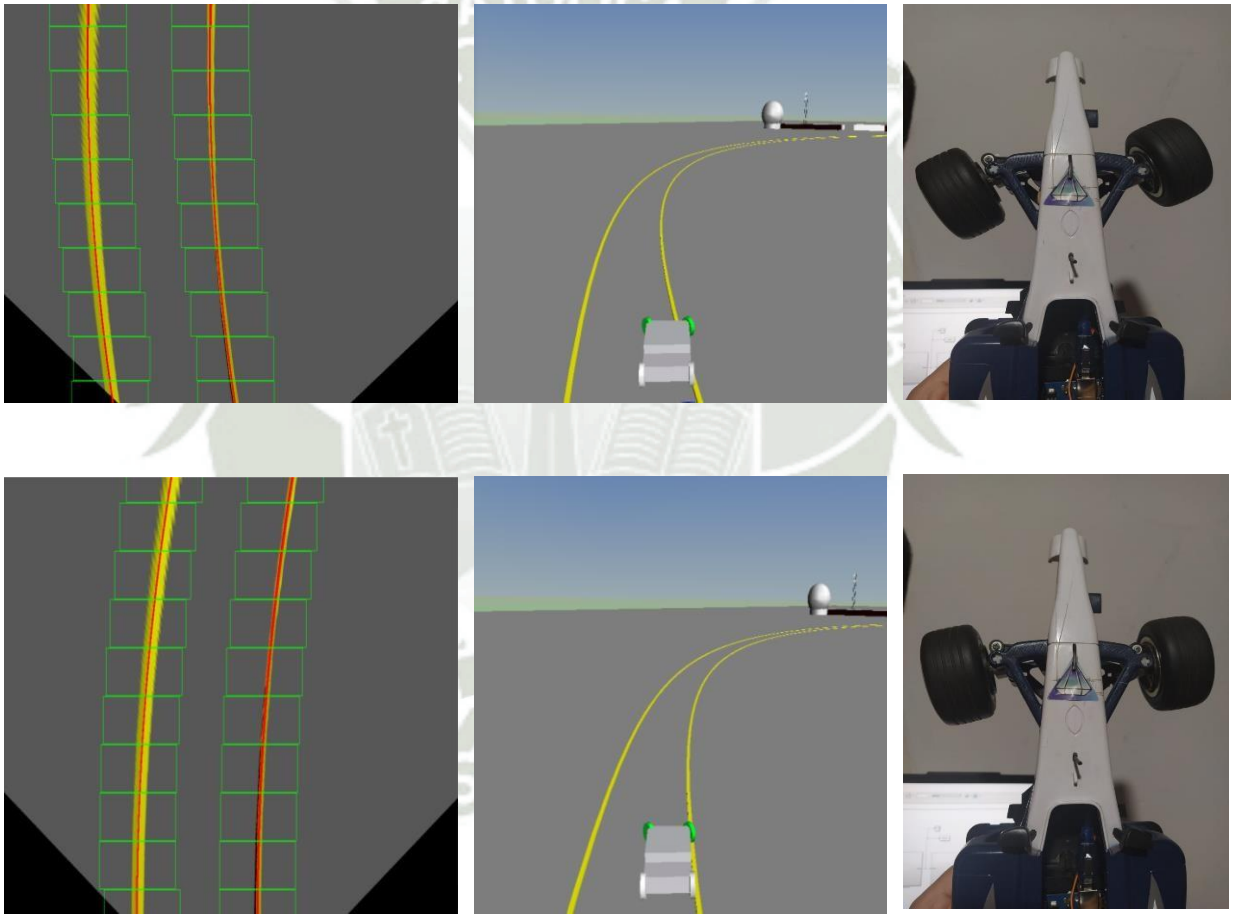


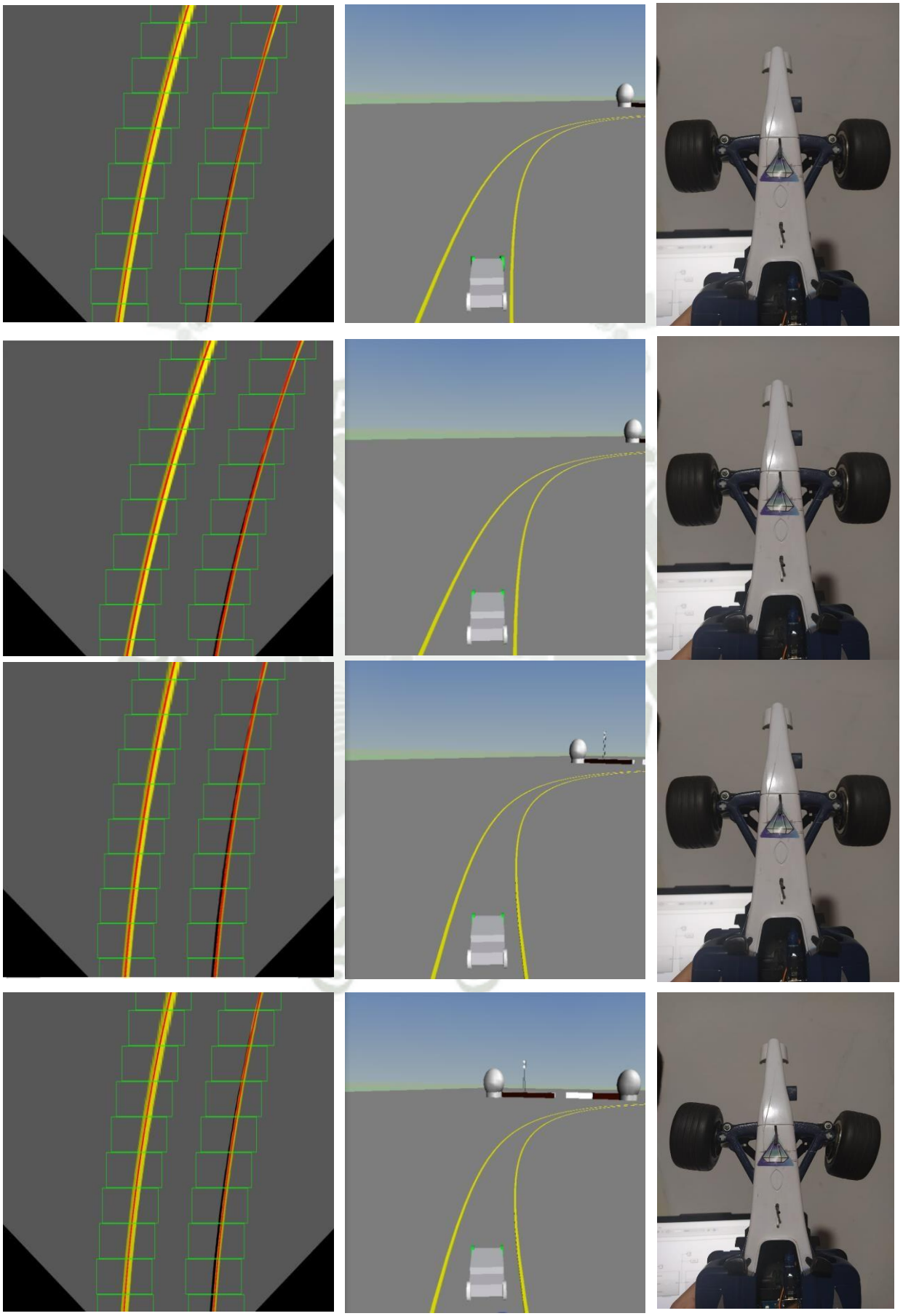
De este modo, el vehículo se centrará en la carretera, a continuación, se presentará una secuencia de imágenes, junto con una aplicación en físico donde se observa que la señal de giro de las llantas es conducida a un modelo de vehículo que tiene conectado un servomotor que permite a ver girar las ruedas del carro conforme a la simulación,

así mismo, se deja un link donde se puede observar el video grabado de esta experimentación:

https://www.youtube.com/watch?v=FuWzVxd0zAY

La simulación inicia con un error en posición de 1.20m, de acuerdo con el diagrama de error que se analiza posteriormente, en este primer estado el ángulo de giro de las ruedas es el máximo que se permite en la simulación, es decir que tiene un valor de 0.3142, y conforme se va acercando el vehículo al centro del carril el ángulo de giro de las ruedas va disminuyendo como se observa en la siguiente cronología:

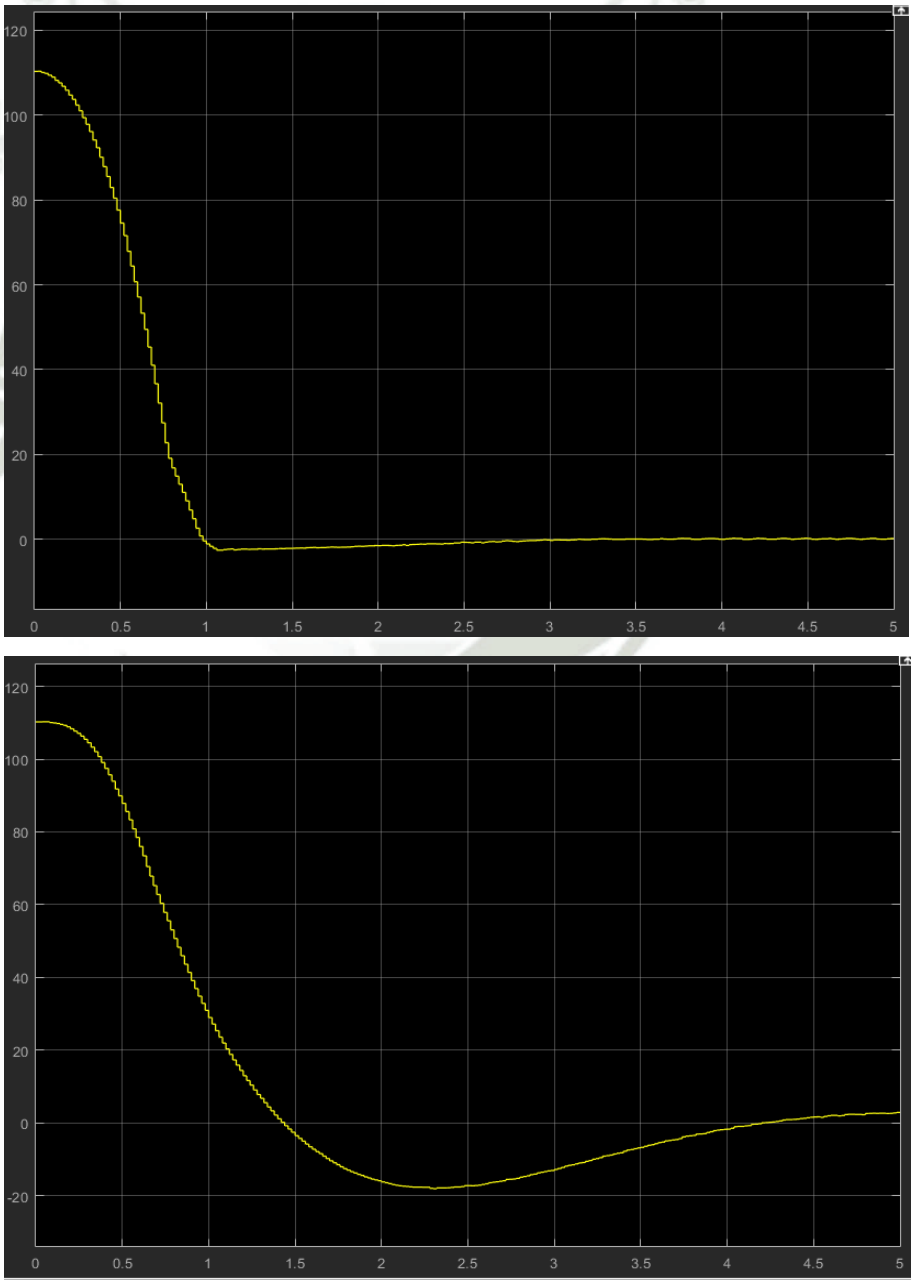




Por último, las ruedas quedarán con un leve giro hacia la derecha para poder cubrir la otra curva que tiene que enfrentar, y avanzando a su vez a lo largo del trayecto que tiene que recorrer. A continuación, se realizará una comparación entre los resultados con un controlador PID convencional y un controlador PD difuso

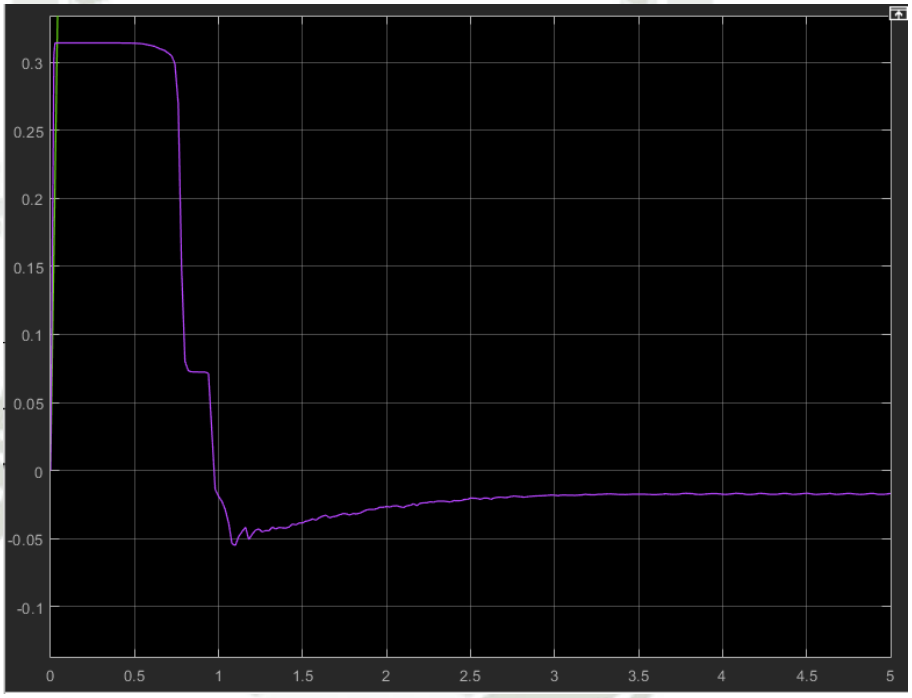
IV.3. COMPARACIÓN CONTROL PID CONVENCIONAL Y CONTROL PD DIFUSO

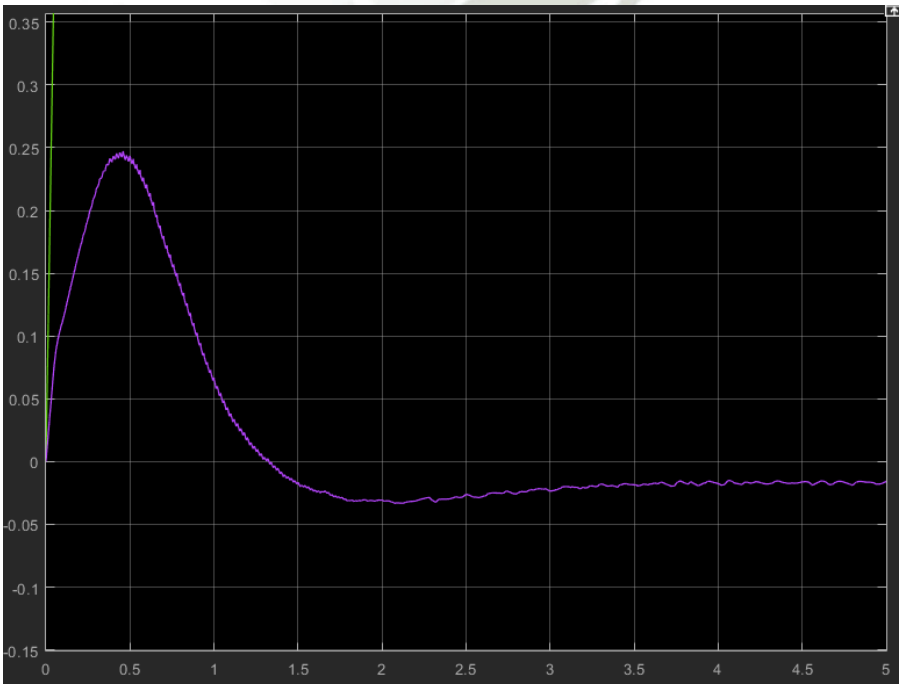
Empezaremos comparando los errores generados por los controladores, en el primer caso la gráfica del error en el controlador PD difuso es la figura superior y la gráfica del control PID convencional es la inferior:



Se observa que la gráfica de error del control difuso demora menos en estabilizar, así como también corrige de manera más rápida el error, sin embargo, la gráfica PID convencional es más suave en su comportamiento.

Lo mencionado antes se observa claramente en las gráficas de ángulos mostradas a continuación, donde la gráfica superior representa a los ángulos brindados por el control difuso y al inferior al controlador PID convencional





Y evidentemente se observa que el control PID convencional tiene un comportamiento más suave al eliminar al error, sin embargo, el control difuso tiene una respuesta más rápida y presenta un sobre impulso menor en el error.

II. CUESTIONARIO FINAL

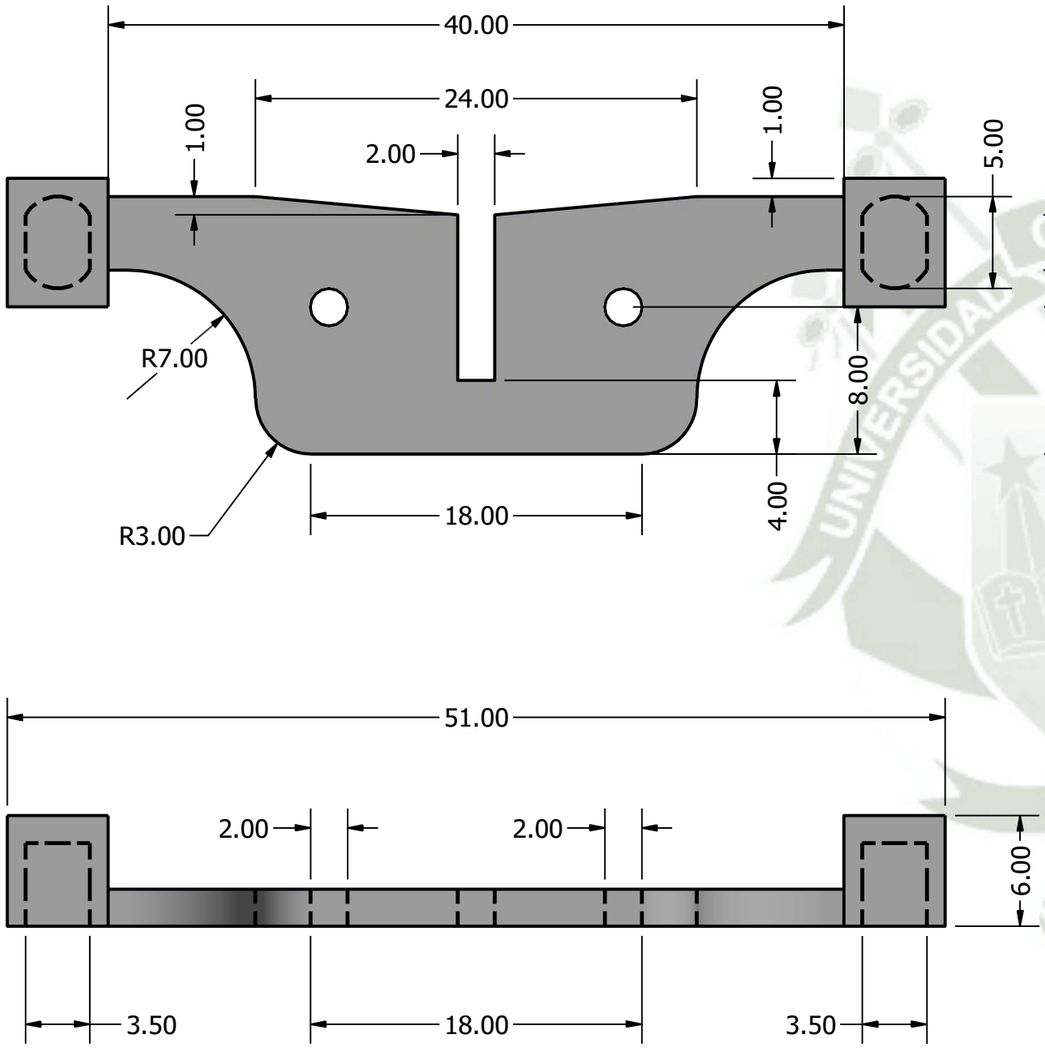
1. ¿Cómo podrías aplicar lo aprendido en la vida real? Mencione 3 ejemplos
2. ¿Qué aplicaciones actuales similares a lo trabajado en clase existen? Mencione 3
3. ¿Qué se puede deducir de la sección IV.3 de comparación entre control PID tradicional y control PD difuso?

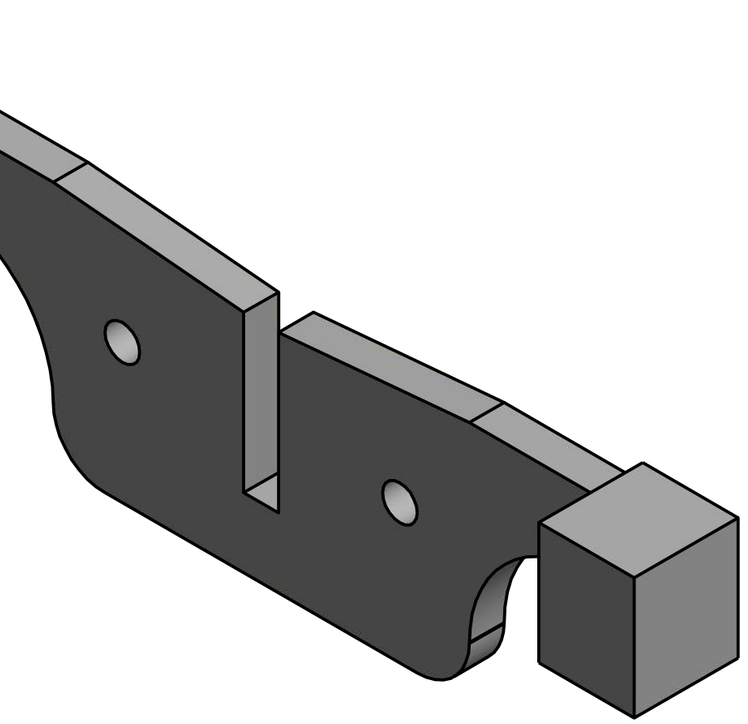
III. REFERENCIAS

Aral Sarrafi (2021). Lane_Keeping_Assist_System (https://github.com/Aral-Sarrafi/Lane_Keeping_Assist_System), GitHub. Retrieved October 29, 2021.

Katsuhiko, Ogata (2010) *Ingeniería de Control Moderna*. Tercera edición. Editorial Prentice Hall. Recuperado de: https://www.u-cursos.cl/usuario/78303fe04da8e4eb340eaee09f1840b2/mi_blog/r/Ingenieria_de_Control_Moderna_Ogata_5a_ed.pdf

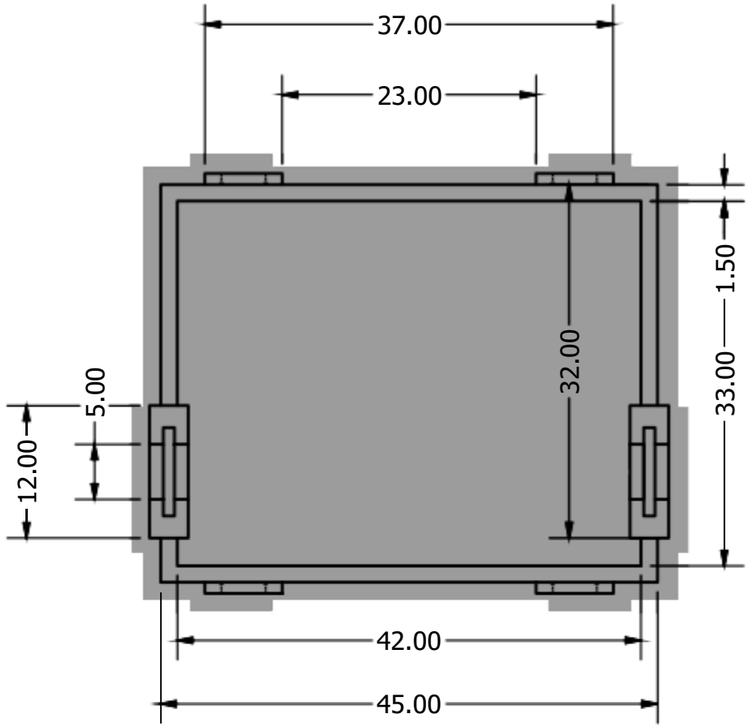
Pardo, Carlos (2021) *PICUINO: Control PID.* Recuperado de: ttps://www.picuino.com/es/arduprog/control-pid.html

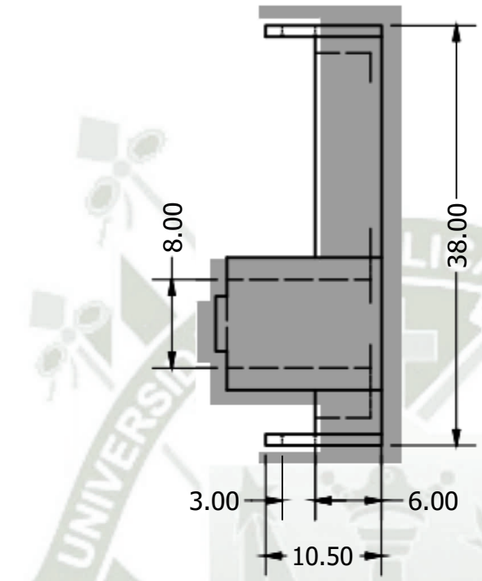


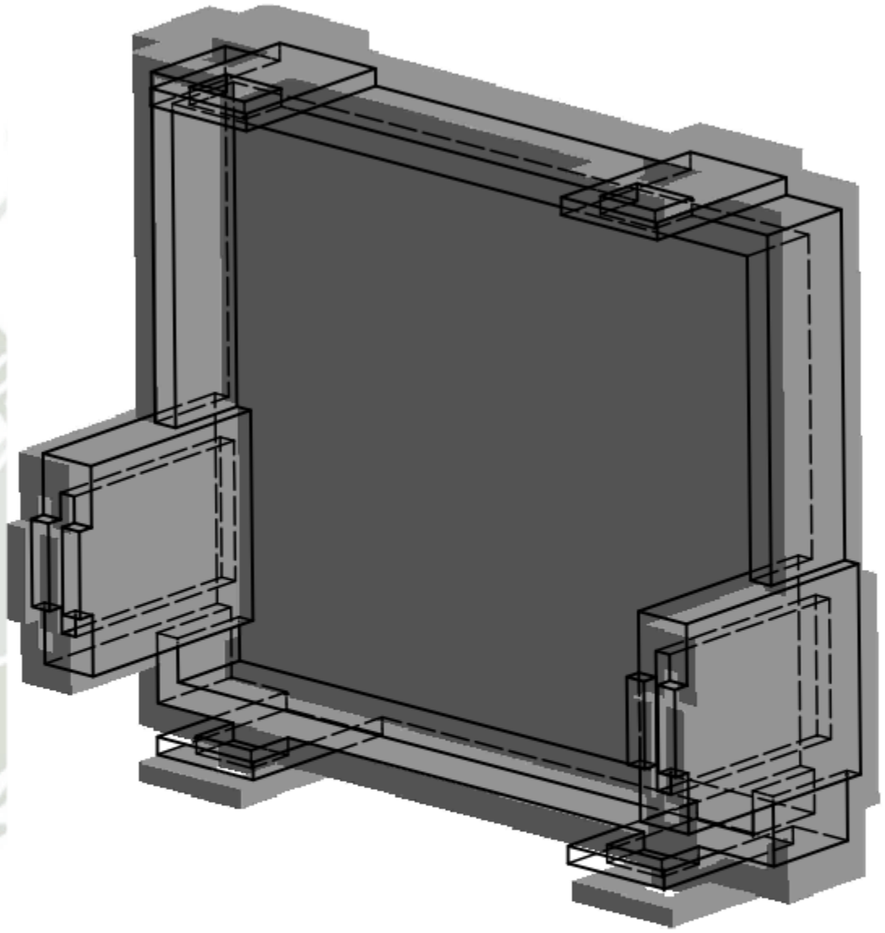


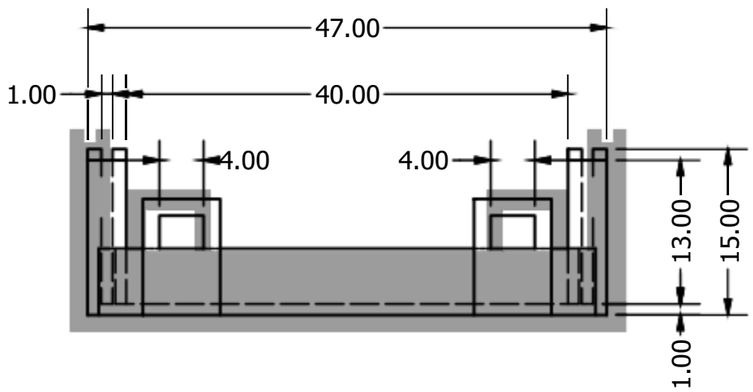
Escala 4:1

N° de Piezas	Denominación	Norma	Material	Marca	Dimensiones
	Nombre	Fecha	Universidad Católica de Santa María		
Elaboró:	André Mayhua Álvarez	30/12/2021	Escuela Profesional de Ingeniería Mecánica, Mecánica - Eléctrica y Mecatrónica		
Revisó:	Marcelo Quispe Ccachuco	10/01/2022			
Escala: 3:1	PLANO PIEZA UNIÓN			Pág: 1	Págs: 6

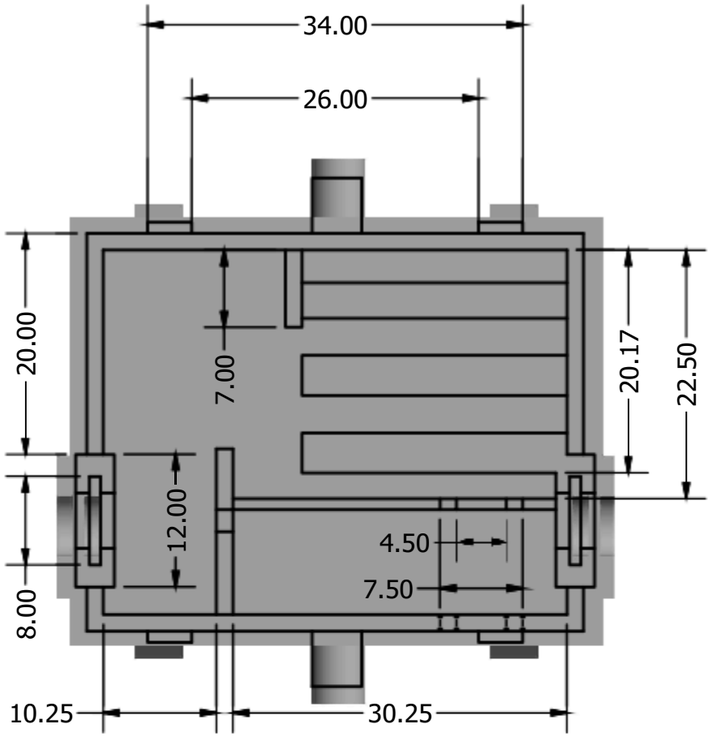


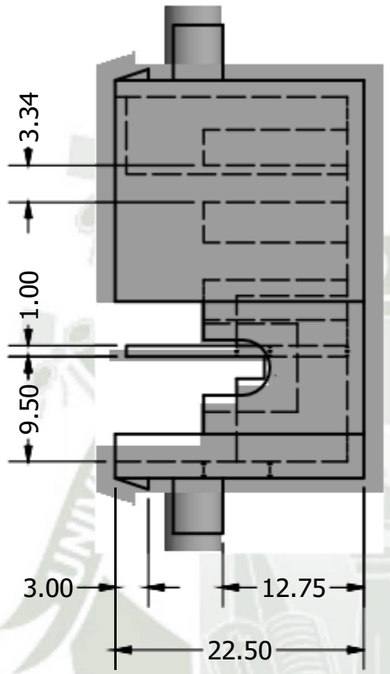


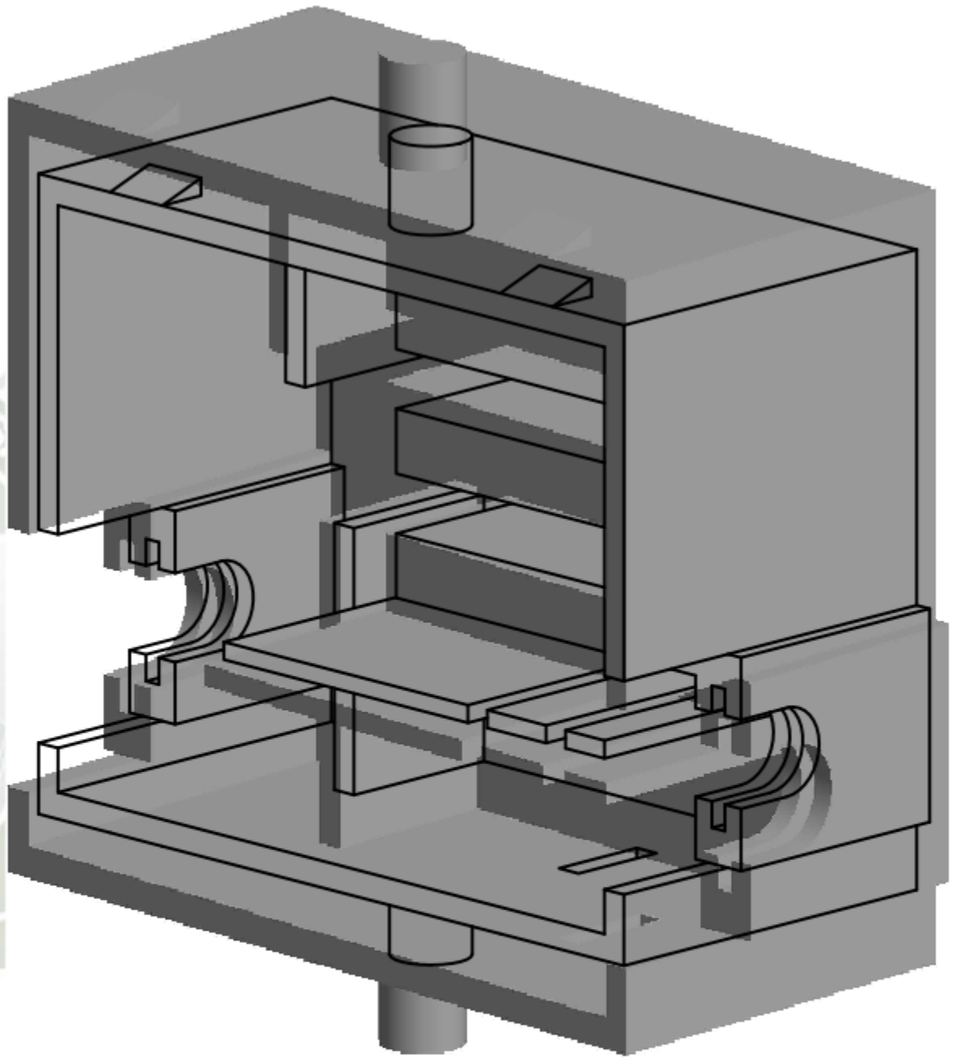


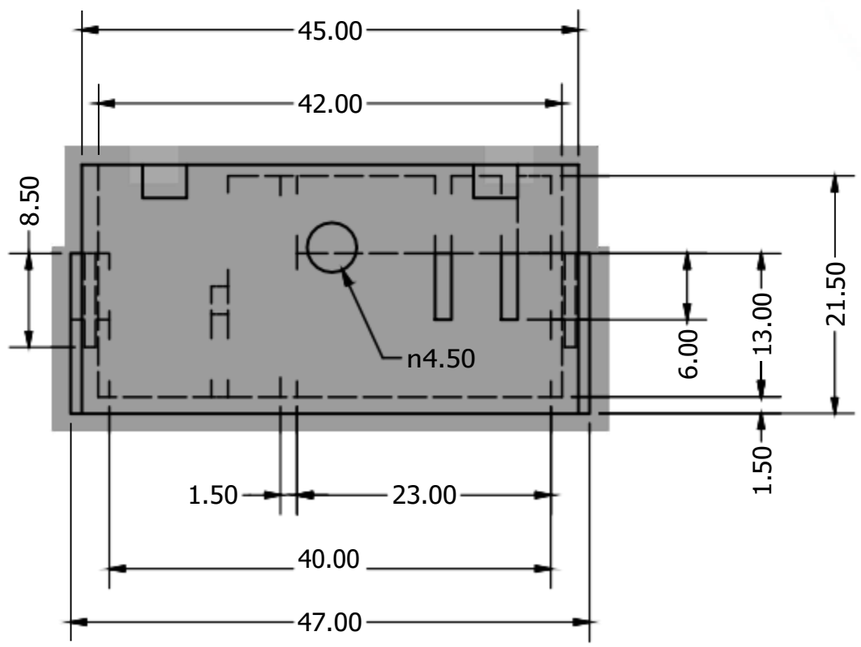


N° de Piezas	Denominación	Norma	Material	Marca	Dimensiones
	Nombre	Fecha	Universidad Católica de Santa María		
Elaboró:	André Mayhua Álvarez	30/12/2021	Escuela Profesional de Ingeniería Mecánica, Mecánica - Eléctrica y Mecatrónica		
Revisó:	Marcelo Quispe Ccachuco	10/01/2022			
Escala:	PLANO PIEZA UNIÓN			Pág:	2
3:1				Págs:	6

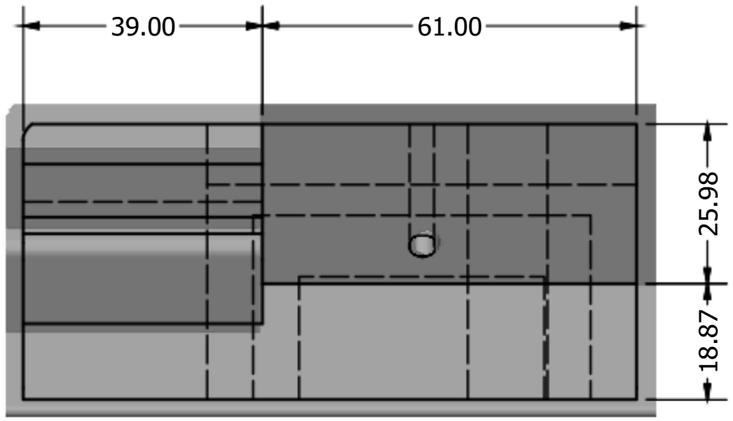


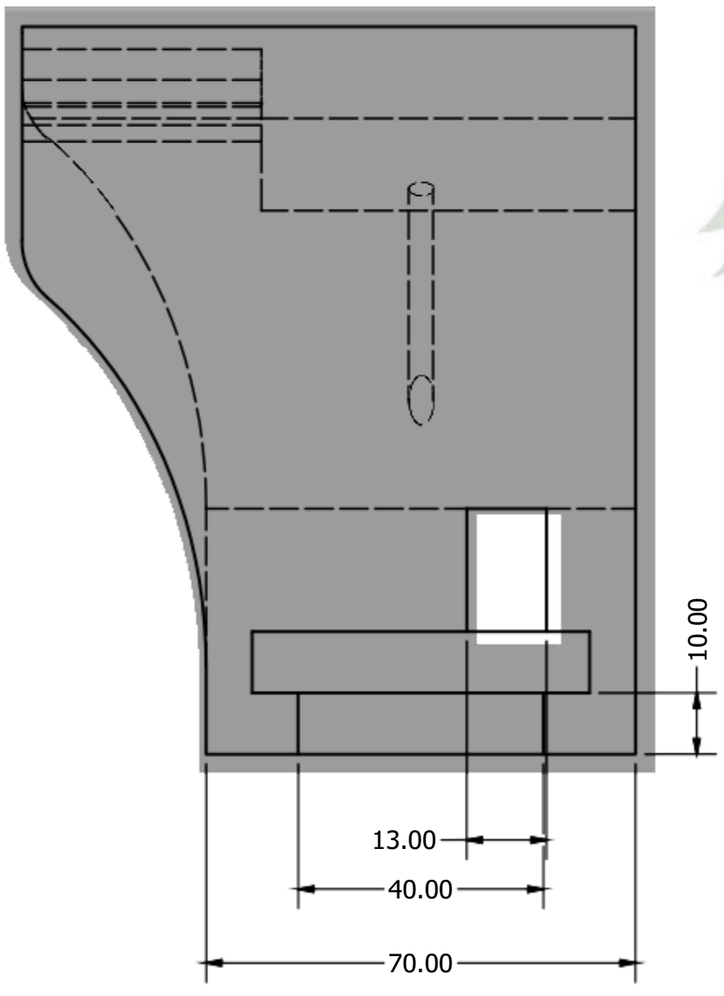


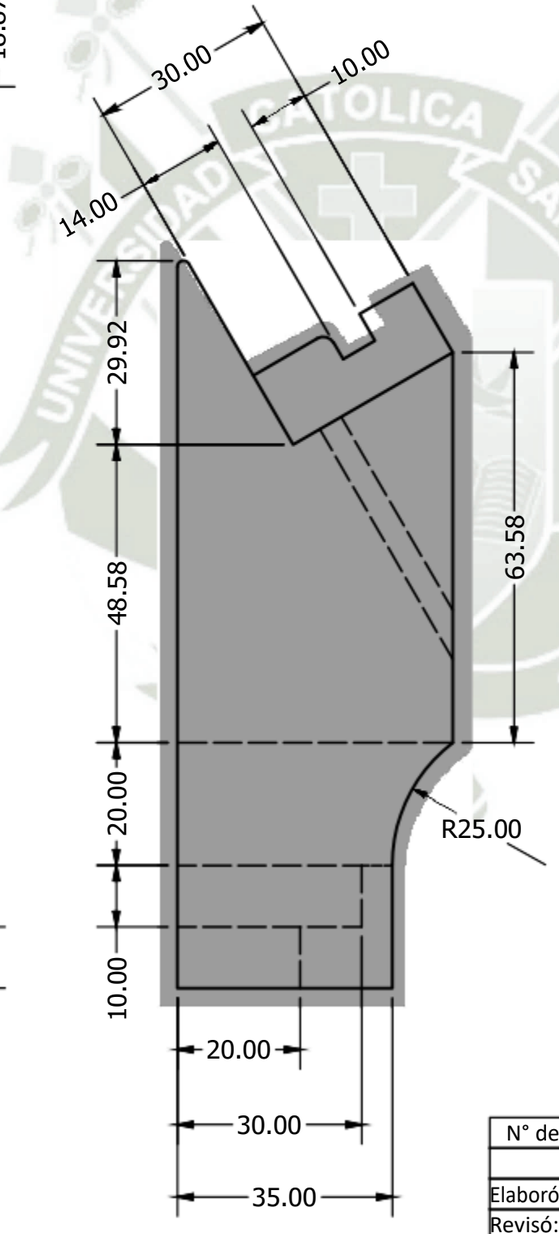


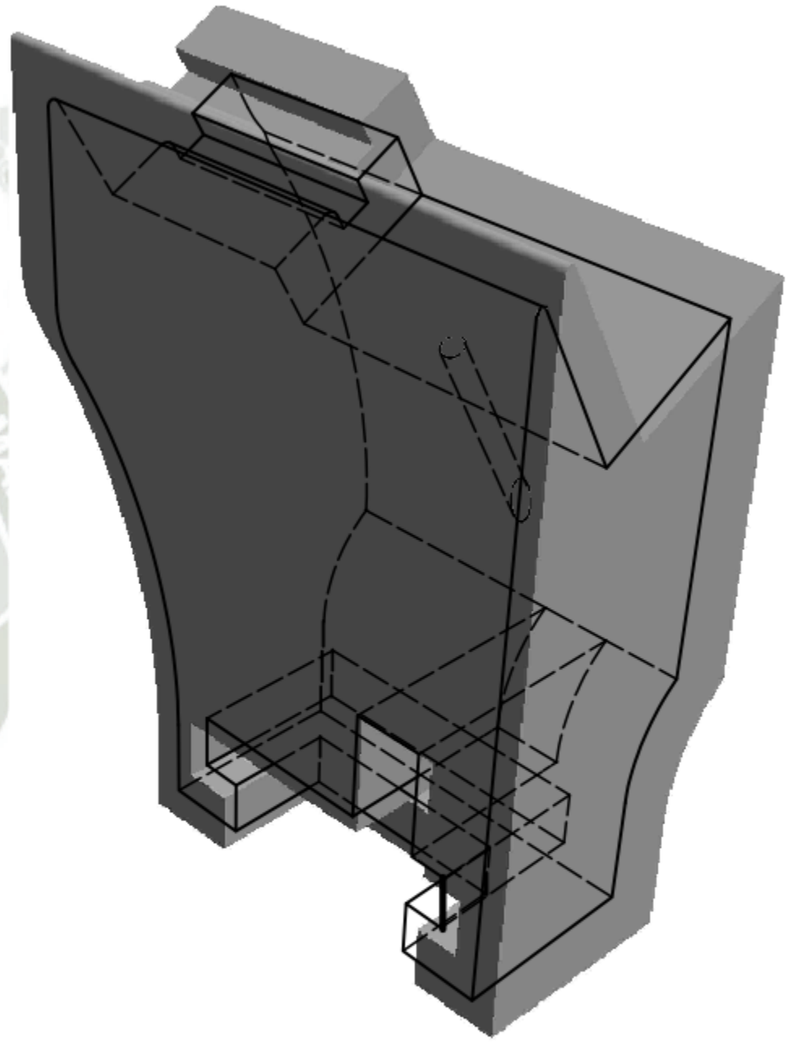


N° de Piezas	Denominación	Norma	Material	Marca	Dimensiones
	Nombre	Fecha	Universidad Católica de Santa María		
Elaboró:	André Mayhua Álvarez	30/12/2021	Escuela Profesional de Ingeniería Mecánica, Mecánica - Eléctrica y Mecatrónica		
Revisó:	Marcelo Quispe Ccachuco	10/01/2022			
Escala:	PLANO PIEZA UNIÓN			Pág:	3
3:1				Págs:	6

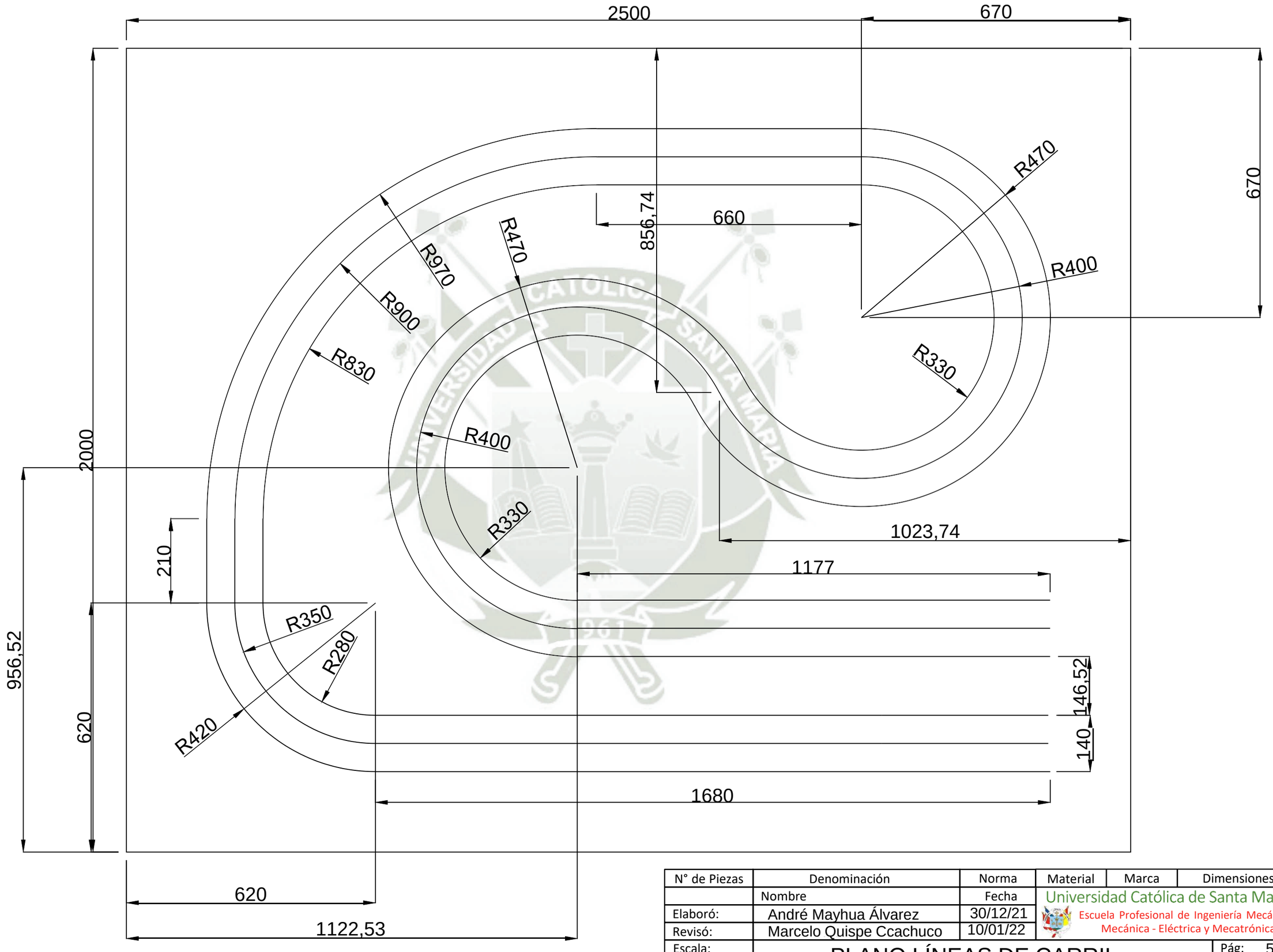








N° de Piezas	Denominación	Norma	Material	Marca	Dimensiones
	Nombre	Fecha	Universidad Católica de Santa María		
Elaboró:	André Mayhua Álvarez	30/12/2021	Escuela Profesional de Ingeniería Mecánica, Mecánica - Eléctrica y Mecatrónica		
Revisó:	Marcelo Quispe Ccachuco	10/01/2022			
Escala: 3:1	PLANO PIEZA UNIÓN				Pág: 4 Págs: 6



N° de Piezas	Denominación	Norma	Material	Marca	Dimensiones
	Nombre	Fecha	Universidad Católica de Santa María		
Elaboró:	André Mayhua Álvarez	30/12/21	Escuela Profesional de Ingeniería Mecánica, Mecánica - Eléctrica y Mecatrónica		
Revisó:	Marcelo Quispe Ccachuco	10/01/22			
Escala:	PLANO LÍNEAS DE CARRIL		Pág: 5		
			Págs: 6		

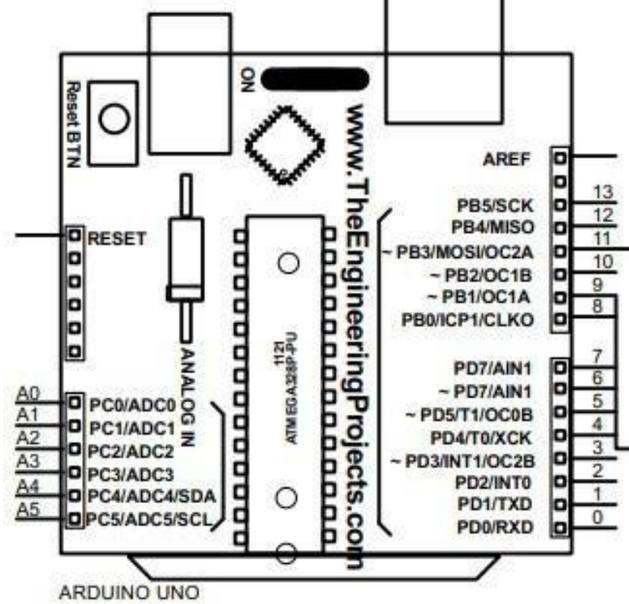
COMPUTADOR

COMUNICACIÓN SERIAL

GENERACIÓN DIRECCIÓN IP

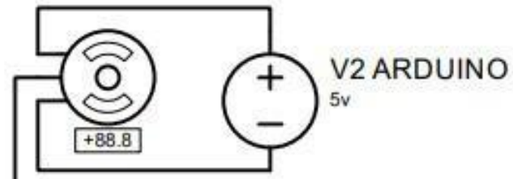
CÁMARA

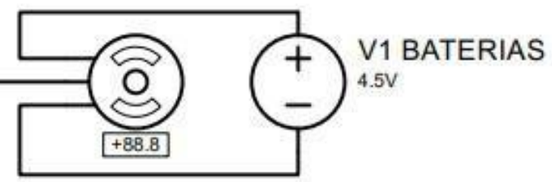
ARD1



ARDUINO

SERVOMOTOR DIRECCIÓN





SERVOMOTOR MODIFICADO

N° de Piezas	Denominación	Norma	Material	Marca	Dimensiones
	Nombre	Fecha	Universidad Católica de Santa María		
Elaboró:	André Mayhua Álvarez	30/12/2021	Escuela Profesional de Ingeniería Mecánica, Mecánica - Eléctrica y Mecatrónica		
Revisó:	.	10/01/2022			
Escala: 3:1	PLANO ELECTRÓNICO				Pág: 6 / Págs: 6