Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

Carrera Ingeniería Electrónica

Proyecto integrador para la obtención del título de grado de ingeniero electrónico

---

SISTEMA DE CONTROL DE NAVEGACIÓN APLICADO A VEHÍCULO AUTONÓMO

Alumnos

Armesto, Maximiliano David y Borgnino, Leandro Ezequiel

Director

Dr. Ing. Hueda, Mario R.

Co-Director

Dr. Ing. Lizarraga, Mariano

Córdoba, República Argentina

Septiembre / 2018

Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

Escuela de Ingeniería Electrónica

El Tribunal Evaluador reunido en este acto y luego de haber aprobado la Solicitud de Aprobación de Tema y efectuado las distintas instancias de correcciones del Informe del Proyecto Integrador para la obtención del Título de Grado "Ingeniero Electrónico" y cumpliendo con el Reglamento correspondiente, declaran el Informe Final de los estudiantes **Armesto, Maximiliano David** y **Borgnino, Leandro Ezequiel** como "aceptado sin correcciones" y la defensa oral Aprobada. Por lo tanto, luego de haber tenido en cuenta los aspectos de evaluación que indica el Reglamento, el Proyecto Integrador se considera Aprobado.

Se firma el Acta de Examen correspondiente y se distribuyen los ejemplares impresos.

Firma y aclaración del Tribunal Evaluador

Fecha:

Dedicatoria

Para nuestras familias...

Agradecimientos

A nuestros padres, madres y hermanos, por su incondicional apoyo a lo largo de toda la carrera.

A Fundación Tarpuy, junto a todo su personal, por brindarnos la oportunidad de desarrollar el presente proyecto. Especialmente a Oscar Agazzi, Mario Hueda y Pablo Gianni por ser nuestros mentores en el camino recorrido.

A nuestros amigos y colegas, tanto del ámbito académico como del ámbito laboral, quienes nos han acompañado en estos años de estudio.

A la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de Córdoba, por la oportunidad de realizar esta carrera de grado.

Resumen

El desarrollo de sistemas de navegación autónoma y su aplicación en vehículos no tripulados es un área en auge en la actualidad. Este desarrollo ha ido de la mano con el marcado crecimiento en el área de inteligencia artificial, procesamiento de grandes volúmenes de datos y las herramientas con gran capacidad de cómputo disponibles. Nuestro país no se ha quedado atrás, y desde hace algunos años ya han comenzado a desarrollarse estas áreas en distintos sectores científicos e industriales. Para lograr la autonomía, un vehículo debe ser capaz de interrelacionarse con el entorno en 4 aspectos fundamentales: la localización, a través de la cual el mismo tiene noción de su ubicación dentro del espacio; la percepción, a través de la cual puede reconocer los objetos que lo rodean, y con los que deberá interactuar; el planeamiento de ruta, que calculará e indicará la trayectoria a recorrer entre dos puntos; y el control del vehículo en sí mismo, que consiste en el control de los actuadores para lograr el desplazamiento. El propósito de este trabajo fue el desarrollo de un sistema a escala que cumpla con cada uno de estos aspectos y logre imitar el comportamiento de un vehículo autónomo real. El sistema fue dividido en dos etapas, de alto y bajo nivel, la primera encargada del procesamiento de datos y cálculo de la trayectoria, la segunda encargada de la ejecución de los movimientos para desplazarse en el espacio y del relevamiento de la información obtenida de los sensores. Se diseñó un protocolo de comunicación para asegurar una comunicación estable entre ambas etapas. Los resultados obtenidos demuestran que a través del control de los 4 aspectos fundamentales mencionados, un vehículo puede desplazarse en forma autónoma dentro de un espacio. El desarrollo del presente trabajo demuestra la capacidad técnica de las personas formadas en las universidades de nuestro país, es una prueba de que estamos a la altura para el desarrollo y la implementación de las tecnologías de punta que se encuentran en auge a nivel mundial.

Área Temática y Asignaturas

El Proyecto Integrador se enmarca dentro de las áreas temáticas de Control y Digitales. A su vez las materias involucradas con una mayor incidencia al trabajo son Electrónica Digital III, Sistemas de Control II e Informática Avanzada.

Palabras Clave

vehículo autónomo, navegación autónoma, sistema de control para navegación autónoma, sistemas inteligentes de navegación, inteligencia artificial en vehículos autónomos.

Abstract

Nowadays, the development of autonomous navigation systems and its application on unmanned vehicles are part of a growing area. This development has gone together with the strong growth on the artificial intelligence area, the big data processing and the great computing capabilities available. Our country has not been left behind, and these areas have begun to develop in different scientific-industrial sectors. In order to achieve the autonomy, a vehicle must be able to interrelate with the environment in 4 fundamental aspects: the location, whereby it has a notion of its position within the space; the perception, whereby it can recognize the objects that surround it and which it must interact with; the route planning, which will calculate and indicate the path to go between two points; and the control of the vehicle itself, which consists in controlling the actuators to achieve displacement. The purpose of this work was the development of a scale system that fulfills each of these aspects and manages to imitate the behavior of a real autonomous vehicle. The system was divided into two stages, one of high level and another one of low level, the first one is in charge of the data processing and the trajectory calculation, and the second one is in charge of the execution of the movements to move in space and the survey of the information obtained from the sensors. A communication protocol was designed to ensure a stable communication between both stages. The results obtained show that through the control of the 4 fundamental aspects mentioned before, a vehicle can move autonomously within a space. The development of this work demonstrates the technical capacity of the people trained in the universities of our country. This is a proof that we are up to the task of developing and implementing the latest technologies that are booming worldwide.

Key Words

autonomous car, autonomous navigation, control system for autonomous navigation, intelligent navigation systems, artificial intelligence in autonomous cars.

Resumo

O desenvolvimento de sistemas de navegação autônoma e sua aplicação em veículos não tripulados é uma área em crescimento atualmente. Este desenvolvimento foi acompanhado pelo crescimento acentuado na área de inteligência artificial, processamento de grandes volumes de dados e ferramentas com grande capacidade computacional disponível. Nosso país não foi deixado para trás e, por alguns anos, essas áreas já começaram a se desenvolver em diferentes setores científicos e industriais. Para obter autonomia, um veículo deve ser capaz de se inter-relacionar com o ambiente em 4 aspectos fundamentais: a localização, através da qual tem noção de sua posição no espaço; a percepção, através da qual ela pode reconhecer os objetos que o cercam, com os quais ela deve interagir; planejamento de rotas, que irá calcular e indicar o caminho para viajar entre dois pontos; e o controle do próprio veículo, que consiste em controlar os atuadores para obter o deslocamento. O objetivo deste trabalho foi o desenvolvimento de um sistema de escala que atende a cada um desses aspectos e consegue imitar o comportamento de um verdadeiro veículo autônomo. O sistema foi dividido em duas fases, superior e inferior, a primeira carga de processamento de dados e o cálculo da trajetória; segundo encarregado da execução de movimentos para movimentar no espaço e no levantamento de informação obtida a partir dos sensores. Um protocolo de comunicação foi projetado para garantir uma comunicação estável entre os dois estágios. Os resultados obtidos mostram que através do controle dos 4 aspectos fundamentais mencionados, um veículo pode se mover de forma autônoma dentro de um espaço. O desenvolvimento deste trabalho demonstra a capacidade técnica das pessoas treinadas nas universidades do nosso país, é a prova de que estamos à altura da tarefa de desenvolver e implementar as mais recentes tecnologias que estão crescendo em todo o mundo.

Palavras Chave

veículo autônomo, navegação autônoma, sistema de controle para navegação autônoma, sistemas de navegação inteligente, inteligência artificial em veículos autônomos.

Índice de cuadros

Índice de figuras

Índice general

Capítulo 1

Introducción

1.1. Antecedentes

No se encontraron Proyectos Integradores que busquen cumplir con los mismos objetivos del presente, es decir el desarrollo y la implementación del sistema de navegación. Sí se han encontrado trabajos y papers que tratan temáticas similares tales como:

- "Sistema de Posicionamiento para Vehículos Autónomos"; V. Milanés, J.E. Naranjo, C. Gonzales, J. Alonso, R. García, T. de Pedro; Instituto de Automática Industrial; Madrid, España.[1]
- "Sistema de Navegación Autónoma de un Vehículo usando Visión Robótica y control difuso en LABView"; Ramírez Cortés J.M., Gómez Gil P., Martínez Carballido J. , López Larios F.; Ingeniería Investigación y Tecnología; México.[2]
- Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID"; Pan Zhao, Jiajia Chen, Yan Song, Xiang Tao, Tiejuan Xu, Tao Mei; International Journal of Advanced Robotics Systems; China.[3]

A nivel mundial, se pueden encontrar productos que se están utilizando tanto a nivel industrial como doméstico basados en sistemas de navegación autónoma. Algunos ejemplos:

- Robots aspiradoras para el hogar.
- Cortadoras de césped autónomas.

[1] https://www.researchgate.net/profile/Vicente_Milanes/publication/43551567_Sistema_de_Posicionamiento_para_Vehiculos_Autonomos/links/0f3175347da6c321f8000000/Sistema-de-Posicionamiento-para-Vehiculos-Autonomos.pdf

[2] http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-77432011000200002

[3] http://journals.sagepub.com/doi/full/10.5772/51314

- Automóviles con sistema de estacionamiento autónomo, detección y seguimiento de carril, sistemas de frenado automático, conducción autónoma.
- Robots inteligentes en las industrias automotrices. Los cuales cumplen funciones tales como llevar piezas de un punto a otro sin asistencia alguna.

1.2. Relevancia del trabajo

Se trata de un proyecto muy abarcativo ya que implica la aplicación de conceptos vistos en materias a lo largo de toda la carrera. Materias como Electrónica Digital III, Electrónica Analógica II, Electrónica Industrial, Sistemas de Control II, Informática Avanzada, Robótica y Animatrónica. Esto es muy importante ya que da la posibilidad de demostrar los conocimientos adquiridos por los alumnos y englobar muchas áreas de la electrónica en un mismo trabajo.
A su vez, en la actualidad no existen grandes desarrollos dentro del país y nos parece muy interesante incursionar en el área y poder tener la posibilidad de realizar los primeros trabajos dentro de la misma.

1.3. Motivación

El desarrollo de sistemas basados en algoritmos de Inteligencia Artificial y su aplicación en navegación autónoma son áreas en auge a nivel mundial en la actualidad. Las mismas se encuentran en un proceso de rápido crecimiento y lo más atractivo aún, todavía no se conoce su límite.
Las incursiones que se han realizado en estos campos dentro de nuestro país son muy recientes, el recurso humano aún se encuentra en pleno proceso de capacitación, son pocas las aplicaciones existentes a nivel industrial y particularmente no existen grandes desarrollos relacionados con la navegación autónoma.
Todo lo antes dicho constituye la motivación que llevó al desarrollo del presente Proyecto Integrador, buscando iniciar con él la incursión como profesionales dentro del área mencionada.

1.4. Objetivo General

Desarrollar un sistema de navegación autónomo para ser aplicado en diversos vehículos con diferentes tipos de aplicaciones. Realizar el diseño y construcción de los componentes de hardware y software necesarios para lograr el sistema deseado y aplicarlo en un auto a escala.

1.5. Objetivos Específicos

- Evaluar, seleccionar y adquirir la estructura, microcontrolador, microprocesador y sensores de proximidad adecuados para la implementación del sistema de navegación autónoma.
- Adaptar el sistema de adquisición de la distancia recorrida utilizado en la Práctica Profesional Supervisada (PPS) para medir la distancia recorrida y la rotación del volante de la nueva estructura.
- Adaptar el software del microcontrolador desarrollado en la PPS para controlar el movimiento de la nueva estructura y adquirir los datos de los diferentes sensores.
- Evaluar, diseñar y construir placas de potencia necesarias para controlar los motores del vehículo. Desarrollar los algoritmos de control para los motores.
- Desarrollar el software con el cual el microprocesador hará todo el procesamiento de alto nivel.
- Diseñar e implementar un protocolo de comunicación entre el microcontrolador y microprocesador para lograr una comunicación libre de errores.

1.6. Breve descripción del proyecto

Diseño e implementación, tanto de hardware como de software, de un sistema para navegación autónoma de vehículos. Este sistema será aplicado a un automóvil a escala. El proyecto es continuación de nuestra Práctica Profesional Supervisada cuyo diagrama en bloques se muestra a continuación:

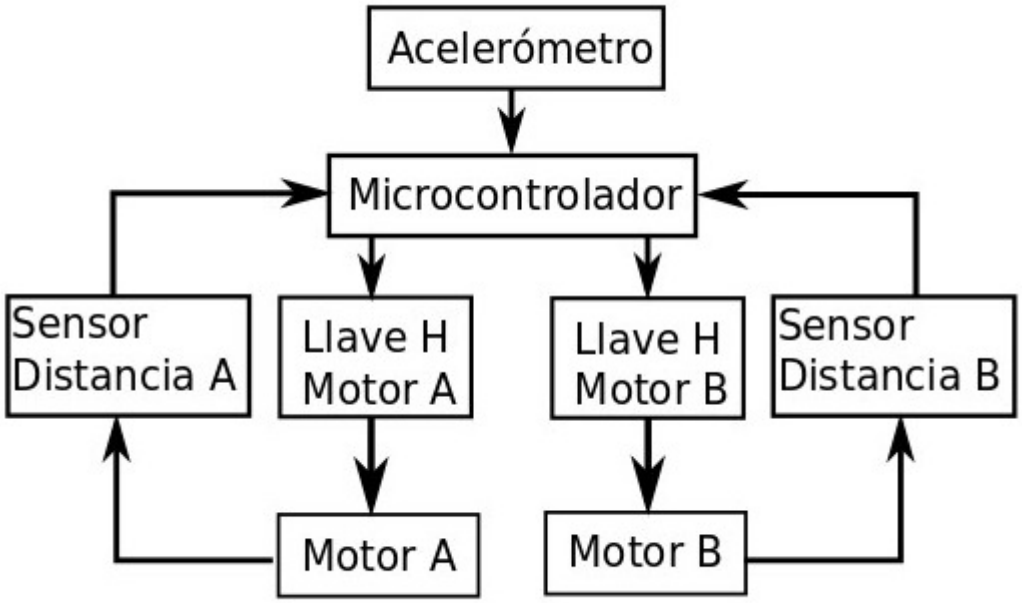


Figura 1.1: PPS - Diagrama general del sistema

Como se puede observar, el sistema de control se centró en un microcontrolador (específicamente en la PPS se utilizó un Arduino MEGA) el cual le enviaba señales de PWM a las llaves H que controlaban la potencia y el sentido de los motores. A su vez, las ruedas del vehículo poseían discos con imanes, y, enfrentados a los discos sensores de efecto Hall que enviaban pulsos al Arduino a partir de los cuales se calculaba la distancia recorrida por el vehículo.
El acelerómetro utilizado fue el MPU-6050, que cuenta con giróscopo, y del cual se extraía el dato del giro en Z para poder hacer rotaciones precisas. La estructura fue un prototipo de hierro soldado, con una plancha de acrílico en su parte superior para poder apoyar todos los componentes electrónicos, este prototipo fue realizado por los estudiantes. Se implementó para este sistema, un software que le indicaba al vehículo una cierta distancia a recorrer con un determinado sentido (por ejemplo 30cm. hacia adelante) y ciertas rotaciones a realizar hacia un lado u otro (por ejemplo 90° hacia la derecha).
Cabe destacar que las rotaciones eran siempre de 90° y se realizaban en el lugar con el sistema de un tanque de guerra, es decir, haciendo girar ambos motores en sentido contrario (las ruedas no doblan hacia un lado u otro).

Se presenta a continuación el diagrama en bloques que representa lo que se desarrolló en el presente Proyecto Integrador:

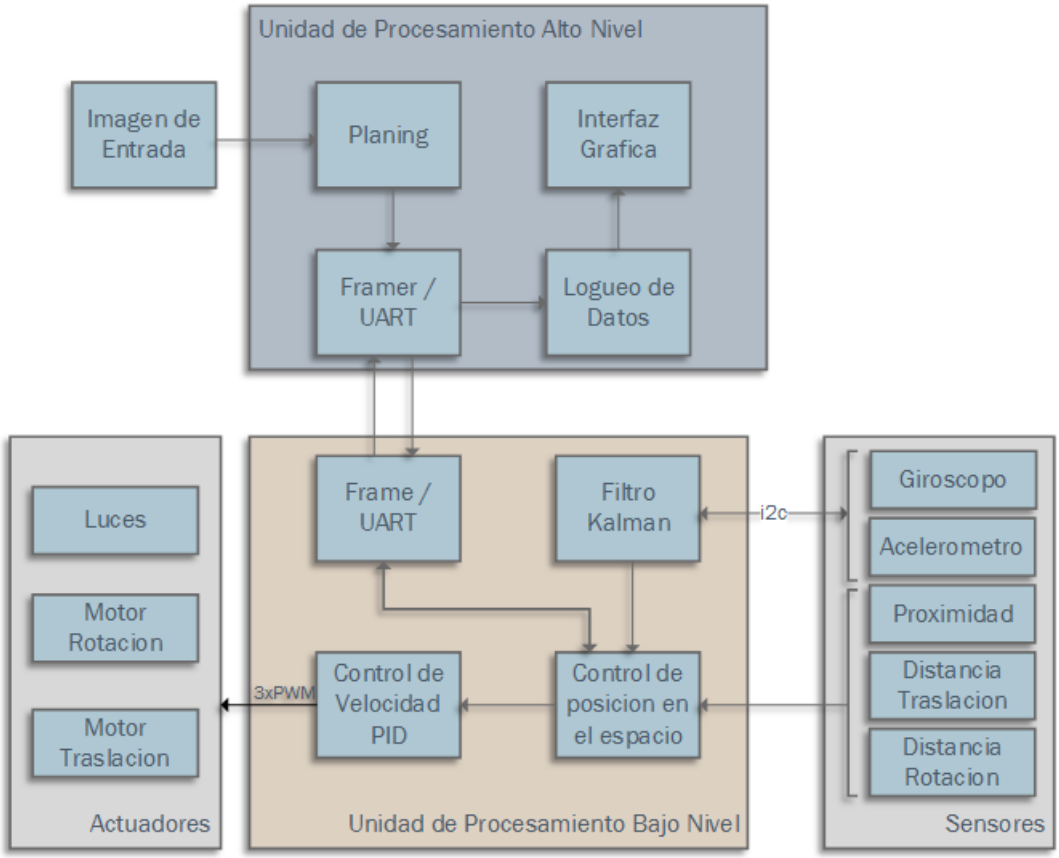


Figura 1.2: Diagrama general del sistema

La estructura realizada en la PPS fue una estructura rápidamente diseñada para poder aplicar el sistema desarrollado, y la misma poseía falencias como rozamientos de las ruedas con la estructura, alto peso del vehículo lo cual dificultaba el movimiento, limitación para hacer trayectorias suaves, mala estética. Para la realización del Proyecto Integrador se decidió comprar un automóvil a escala para evitar todos los problemas anteriormente nombrados. La nueva estructura es más liviana, no posee problemas mecánicos que afecten la electrónica y la forma de giro del vehículo es a través de un motor colocado en el volante para realizar el giro de las ruedas. De esta estructura se extrajo toda la parte electrónica para poder implementar la propia, lo único que se reutilizó fueron los motores. Como los nuevos motores poseían características diferentes que los utilizados en la PPS se rediseñaron y volvieron a construir las llave H para poder cumplir con los nuevos requerimientos. Una de las llave H controla el motor de avance y la otra el motor de giro horizontal de las ruedas delanteras. Los sensores de

distancia utilizan la misma tecnología que se utilizó en la PPS readaptados a la nueva estructura, por un lado se mide la distancia recorrida por el vehículo y por otro el ángulo que rota el volante. El microcontrolador se encarga de tomar los datos de todos los sensores y controlar los motores del vehículo. La velocidad del motor de avance puede ser fijada gracias a un control PID y la rotación del volante cuenta con un control proporcional. Todo el procesamiento de datos para lograr la autonomía del vehículo se realiza en el microprocesador. El mismo toma un mapa con punto de inicio, punto de llegada y obstáculos y calcula la trayectoria más corta entre estos dos puntos esquivando los obstáculos. Una vez calculada la trayectoria se determinan las acciones necesarias a realizar por el vehículo para poder recorrer la misma. Estas acciones luego se envían al microcontrolador para que las ejecute. Microprocesador y microcontrolador se comunican a través de comunicación serie con un protocolo de desarrollo propio que nos permite que la misma sea estable. Se incorporaron sensores de ultrasonido para realizar detección de objetos en forma dinámica y agregarlos al mapa.

1.7. Metodología

La metodología de trabajo utilizada para el desarrollo del proyecto fue Kanban, estableciendo desde un principio todas las tareas que debían ser realizadas para la conclusión del mismo y dividiéndolas en diferentes estados (Pendientes, Trabajo en proceso, Bloqueantes, Completadas).

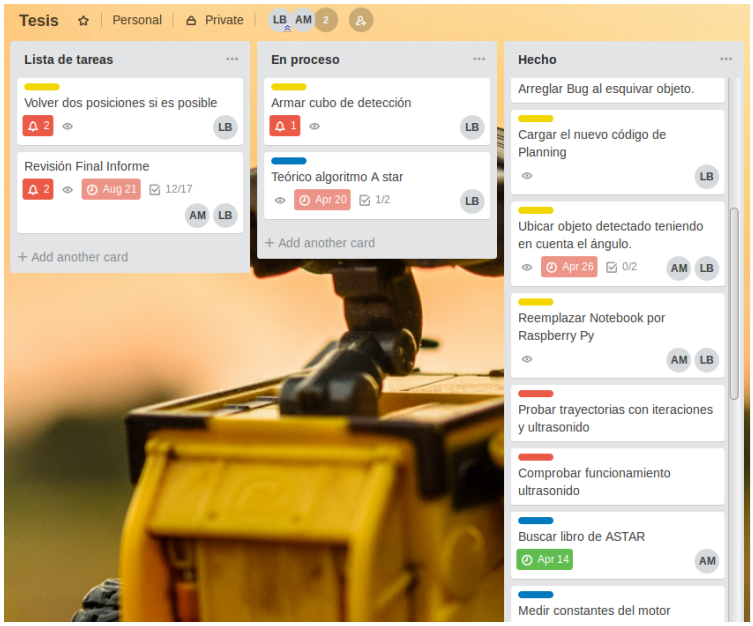


Figura 1.3: Método Kanban utilizado

Lugar previsto de realización: El desarrollo de proyecto integrador se llevó a cabo en la Fundación Tarpuy ubicada en calle Romagosa 518 – B° Colinas de Velez Sarsfield.

Requerimiento de Instrumental y equipos:

- Instrumental de medición: Multímetro Digital, Osciloscopio, Analizador Lógico.
- Generador de funciones.
- Microprocesador y Microcontrolador.
- PC para programación y testeo del sistema.
- Diversos sensores y componentes electrónicos necesarios para la realización del sistema (acelerómetro, giróscopo, encoders, componentes electrónicos en general).
- Estructura de vehículo a escala.

Inversión estimativa por parte de los alumnos: $10.000

Apoyo Económico externo a la Facultad: se recibieron aportes económicos por parte de la Fundación Tarpuy-Fundación Fulgor.

Capítulo 2

Marco Teórico

En este capítulo se verán los puntos teóricos claves para el desarrollo del presente proyecto integrador. Primero se introduce el tema de vehículos autónomos con el objetivo que el lector comprenda dónde se enmarca el trabajo dentro del gran universo de autonomía aplicada a esta área. Luego se introducirán conceptos básicos de inteligencia artificial para poder comprender cómo se aplica ésta en la búsqueda de rutas óptimas para la traslación de vehículos no tripulados. Para poder desplazarse, cualquier vehículo debe contar con un sistema que cumpla tal fin, es por ésto que se detalla también el sistema de desplazamiento clásico utilizado por los automóviles comerciales para que se pueda comprender la complejidad del mismo y como se trabaja con él. Para lograr la autonomía es necesario que el vehículo conozca el entorno donde se encuentra y su posición dentro del mismo, al final del capítulo se detallan diferentes metodologías para lograr tal fin.

2.1. Vehículos Autónomos

2.1.1. Definición

Para establecer qué es un vehículo autónomo primero debe definirse los conceptos de autonomía y conducción autónoma.

- Autonomía: Según la Real Academia Española: *'Condición de quien, para ciertas cosas, no depende de nadie.'*. Si aplicamos esta definición a robots móviles, podemos decir que son los cuales tienen la capacidad de operar sin interacción humana, a través de inteligencia artificial.
- Conducción autónoma: Cualquier vehículo con una serie de funcionalidades que lo hacen capaz de seguir una trayectoria, acelerar y frenar

con intervención parcial o sin intervención alguna del conductor se considera que posee conducción autónoma.

Teniendo en cuenta las definiciones anteriores podemos definir como vehículo autónomo a un robot móvil capaz de conducir autónomamente desde un punto A hacia un punto B sin interacción de un conductor humano. Este vehículo es capaz de moverse por el espacio debido a la capacidad de determinar a través de sensores el estado del entorno, interpretar los datos de estos sensores a través de una unidad de procesamiento y tomar decisiones para cumplir su tarea.

2.1.2. Clasificación

Cuando se habla de vehículos autónomos, se debe establecer el tipo o nivel de autonomía que tiene el sistema particular. La SAE (Society of Automotive Engineers), formalmente Sociedad de Ingenieros de Automoción, es la organización enfocada en la movilidad de los profesionales en la ingeniería aeroespacial, automoción, y todas las industrias comerciales especializadas en la construcción de los vehículos.[12] Esta sociedad se encargó de realizar una clasificación de los vehículos autónomos según sus funcionalidades en niveles crecientes de autonomía del 0 al 5.

- Nivel 0:
  Los vehículos que no tienen ningún tipo de control autónomo comprenden este nivel. El manejo puede ser asistido por sensores que alertan sobre posibles problemas en la conducción. Por lo tanto, el usuario debe realizar todas las tareas correspondientes al manejo del vehículo en cuestión.

- Nivel 1:
  En este nivel el conductor humano controla las tareas críticas de manejo pero puede tener asistencias tecnológicas mínimas. El vehículo puede tener, por ejemplo, un sistema que ayude a la aceleración o desaceleración del vehículo en ciertas circunstancias.
  Todos los autos modernos son mínimo nivel uno, ya que desde 2012 es obligatorio que posean control automático de estabilidad.

- Nivel 2:
  En este nivel el vehículo puede tomar control de la dirección y de la aceleración/desaceleración al mismo tiempo en situaciones particulares. Si bien los vehículos que comprenden este nivel toman control en

ciertos modos de manejo o tareas, el conductor humano tiene el control del mismo todo el tiempo. Un ejemplo de coche semiautónomo es el Mercedes-Benz Clase E. Está a la venta desde marzo de 2016 y destaca por el denominado Drive Pilot que es capaz de evitar la salida de la calzada sin la necesidad de que existan líneas de carril.

- Nivel 3:
  En este nivel de autonomía el vehículo puede controlar todos los aspectos relacionados al manejo en un entorno controlado como una autopista. El conductor humano debe estar en el vehículo para monitorear y controlar el funcionamiento en rutas o escenarios con situaciones inesperadas. En este nivel podría encontrarse el sistema Autopilot de Tesla en el Model S. Es un sistema que está desactivado por defecto en el coche y que debe ser conectado voluntariamente por el conductor. Realiza constantes comprobaciones para asegurarse de que el conductor permanece atento y con las manos en el volante avisando mediante alertas sonoras y luminosas si no detecta las manos en él.

- Nivel 4:
  Este nivel de autonomía lo conforman los vehículos en los cuales no es necesaria interacción con el conductor si el sistema tiene información del entorno. Estos aún vehículos tienen la infraestructura necesaria para poder ser conducidos de forma manual por un usuario. Algunas empresas muy grandes ya han comenzado a hacer sus primeras pruebas serias para ver cuál es el funcionamiento real de los coches autónomos. Google es un ejemplo de conducción autónoma; desde mayo de 2012 tiene licencia para probar coches autónomos en algunos estados de Estados Unidos. Volvo y Uber también se han aliado para crear la primera flota de taxis autónomos del mundo.

- Nivel 5:
  En este nivel el vehículo es completamente autónomo. El usuario sólo puede controlar el destino del vehículo y no puede intervenir en el control del mismo en la trayectoria realizada. Los vehículos de este nivel no existen en la actualidad ya que implica que no exista infraestructura alguna para la conducción manual.

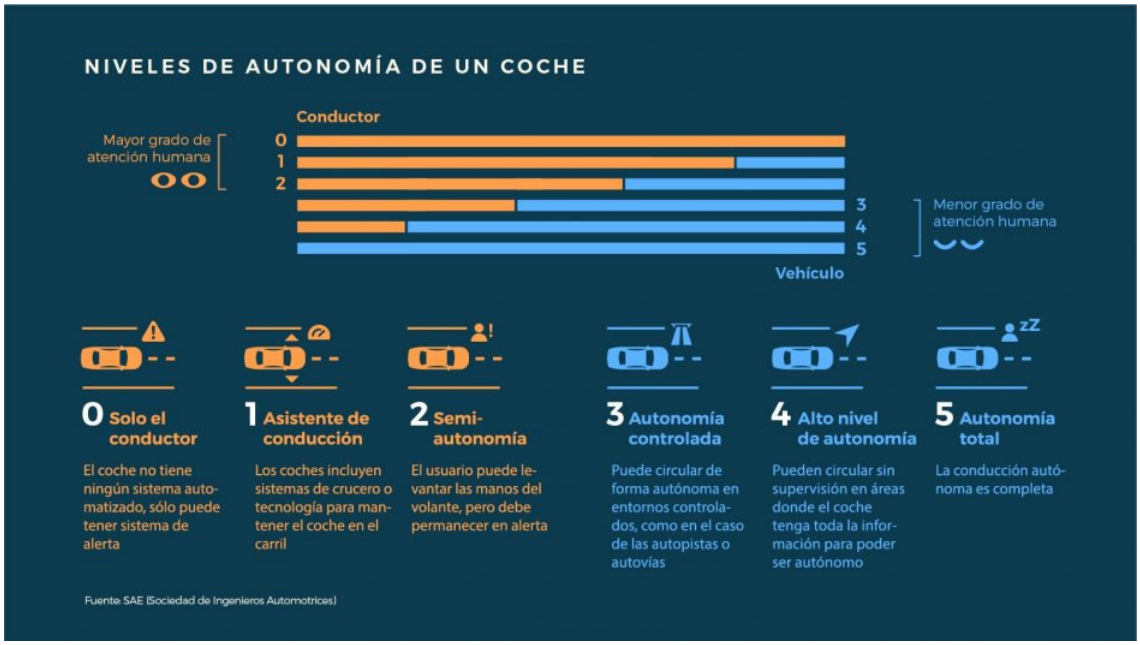


Figura 2.1: Clasificación Vehículo Autónomo SAE[14]

2.2. Inteligencia Artificial

2.2.1. Introducción

Para realizar algoritmos que son juzgados como inteligentes debe conocerse primero la definición de inteligencia. La misma puede definirse de manera sencilla como una serie de funcionalidades que tiene la mente que a un ente dado permite aprender, entender, razonar, tomar decisiones y formarse una idea determinada de la realidad. Una definición más acorde a un algoritmo podría ser la habilidad de tomar decisiones acertadas dada una serie de entradas y una variedad de acciones posibles.
El poder tener una computadora que modele el mundo lo suficientemente bien como para exhibir inteligencia ha sido el foco de investigación en Inteligencia Artificial (IA) desde hace más de medio siglo. La dificultad en poder representar toda la información acerca del mundo de forma utilizable por una computadora, ha llevado a los investigadores a recurrir a algoritmos de aprendizaje para capturar mucha de esta información. Durante mucho tiempo ha habido áreas de estudio complicadas de abordar cómo entender imágenes o lenguaje. Una idea que ha rondado a los investigadores desde hace tiempo es la de descomponer los problemas en subproblemas a diferentes niveles de abstracción. En visión podemos pensar en extraer pequeñas variaciones geométricas que nos representan nuestra imagen en forma abstracta, variaciones que tal vez no son entendibles para el humano, pero que

sirven como información para ser procesada por nuestros algoritmos. Por ejemplo se podrían colocar capas de detección de bordes, de colores o de formas las cuales extrajeran estas características de nuestra imagen para ser luego ingresadas a nuestros algoritmos de IA.
El gran avance de los diferentes algoritmos de IA respecto de los algoritmos tradicionales se basa en que el sistema puede ser entrenado, o incluso puede entrenarse a sí mismo, para encontrar coherencia en los datos de entrada y producir una salida en base a los mismos, de la misma forma que un cerebro humano consigue aprender en función de estímulos y conocimiento previo. Los algoritmos de IA permiten extraer conocimiento de enormes volúmenes de datos, ya sean grandes repositorios de imágenes, vídeos, textos, conversaciones.
Se lograron generar algoritmos que pueden agregar sonidos a una película muda, detectar objetos en una imagen, clasificarlos, convertir imágenes en blanco y negro en imágenes a color, traducir textos puros, detectar textos en imágenes, traducir audios de un idioma hacia otro y hasta increíbles tareas como lo es realizar complejos diagnósticos médicos a través del procesamiento de imágenes.

2.2.2. Clasificación

Según las tareas de aprendizaje realizadas podemos clasificar a los algoritmos de inteligencia artificial como:

- Clasificación : Los datos son objetos caracterizados por atributos que pertenecen a diferentes clases (etiquetas discretas). La meta es inducir un modelo para poder predecir una clase dados los valores de los atributos. Se usan por ejemplo, árboles de decisión, reglas, SVM, etc.
- Regresión: Las clases son continuas. La meta es inducir un modelo para poder predecir el valor de la clase dados los valores de los atributos. Se usan por ejemplo, árboles de regresión, regresión lineal, redes neuronales, LWR, etc.
- Segmentación : Separación de los datos en subgrupos o clases interesantes. Las clases pueden ser exhaustivas y mutuamente exclusivas o jerárquicas. Se usan algoritmos de clustering, SOM (self organization maps), EM (expectation maximization), k means, etc.
- Aprendizaje de dependencias: El valor de un elemento puede usarse para predecir el valor de otro. La dependencia puede ser probabilística,

puede definir una red de dependencias o puede ser funcional (leyes físicas). Se pueden utilizar redes Bayesianas, redes causales y reglas de asociación.

- Detección de desviaciones: Detección de desviaciones, casos extremos o anomalías: Detectar los cambios más significativos en los datos con respecto a valores pasados o normales. Sirve para filtrar grandes volúmenes de datos que son menos probables de ser interesantes. El problema está en determinar cuando una desviación es significativa para ser de interés.
- Mejor acción Aprendizaje de la mejor acción a tomar a partir de experiencia: Esto involucra búsqueda y exploración del ambiente. Esto está relacionado principalmente con aprendizaje por refuerzo, pero también con técnicas como aprendizaje de macro-operadores,chunking y EBL.
- Optimización : Existen una gran cantidad de algoritmos de búsqueda tanto determinística como aleatoria, individual como poblacional, local como global, que se utilizan principalmente para resolver algún problema de optimización. Aquí podemos incluir a los algoritmos genéticos, recocido simulado, colonia de hormigas,técnicas de búsqueda local, enjambres, etc.

2.3. Búsqueda de rutas óptimas en sistemas de navegación autónoma

2.3.1. Introducción

La navegación consiste en una serie de métodos cuya conjugación constituya un sistema capaz de establecer el curso de un vehículo desde un punto de inicio hasta un punto final a través de un entorno dinámico que puede contener obstáculos de distinta índole. Esto se logra mediante la generación de diferentes rutas o trayectorias cuyo objetivo es alcanzar el destino de forma óptima teniendo en cuenta las dimensiones del vehículo y las limitaciones físicas que se establecen en el movimiento del mismo para dejar de considerarlo como un punto en un espacio teórico y tomarlo como un objeto que tiene unas medidas específicas y ocupa un volumen en el espacio de navegación. Para llevar esto a cabo es importante el estudio de la percepción del entorno, la planificación de la ruta, la generación de la trayectoria y el correcto recorrido del mismo; aspectos que se deben tener

en cuenta en el desarrollo del proyecto y aunque son analizados independientemente se relacionan entre sí ya que demuestran continuidad, de allí la importancia de implementar su solución en el orden respectivo.
Las cuatro etapas que componen la navegación pueden definirse de la siguiente manera:

- Percepción del mundo: Mediante el uso de sensores externos, creación de un mapa o modelo del entorno donde se desarrollará la tarea de navegación. En el caso particular del proyecto la percepción inicial del mundo estará dada por un mapa aproximado inicial y luego mediante los sensores el mismo será modificado de ser necesario.

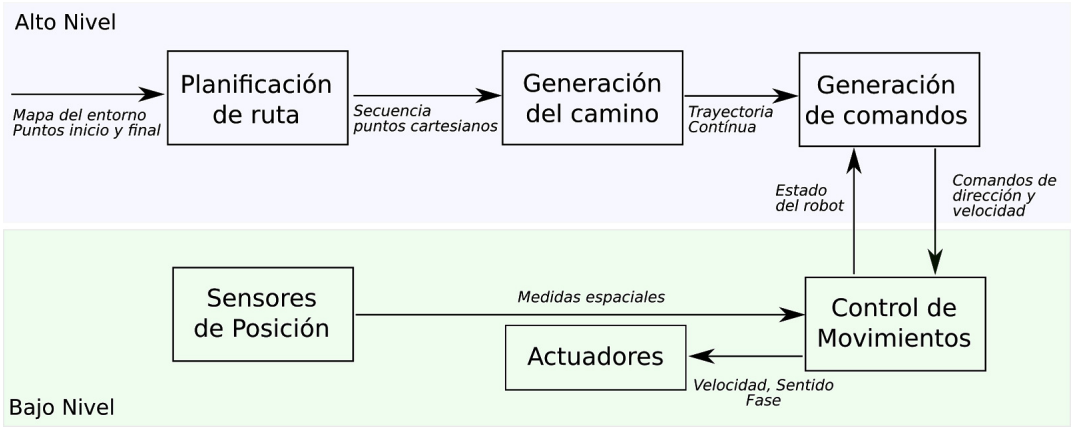
- Planificación de la ruta: Crea una secuencia ordenada de objetivos o submetas que deben ser alcanzadas por el vehículo. Esta secuencia se calcula utilizando el modelo o mapa de entorno, la descripción de la tarea que debe realizar y algún tipo de procedimiento estratégico.

- Generación del camino: En primer lugar define una función continua que interpola la secuencia de objetivos construida por el planificador. Posteriormente procede a la discretización de la misma a fin de generar el camino.

- Seguimiento del camino: Efectúa el desplazamiento del vehículo, según el camino generado mediante el adecuado control de los actuadores del vehículo.[16]

En la figura siguiente se puede observar las fases de la navegación divididas en alto nivel, el cual podría ser un microprocesador y bajo nivel, el cual podría ser un microcontrolador:

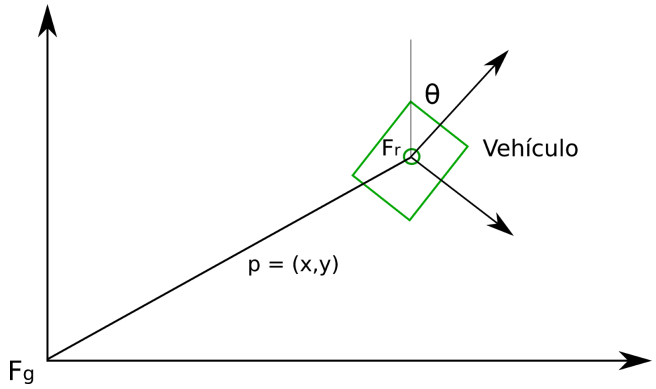
Figura 2.2: Diagrama de bloques planning



2.3.2. Desarrollo matemático del problema de planificación de trayectorias en un vehículo

Se define una configuración q de un robot como un vector cuyas componentes proporcionan información completa sobre el estado actual del mismo. Un vehículo es un objeto rígido al cual se le puede asociar un sistema de coordenadas móvil. La localización del vehículo en un determinado instante de tiempo queda definido por la relación existente entre el sistema de coordenadas global *Fg* en virtud del cual está definido todo el entorno de trabajo y su sistema de coordenadas locales asociado *Fr*.

Figura 2.3: Modelo cartesiano del robot



El estado q del robot viene dado por un vector compuesto por la posición p y la fase θ en un instante de tiempo dado.

$$q = (p, \theta) = (x, y, \theta) \tag{2.1}$$

Se denomina espacio de configuraciones C del robot R a todas las configuraciones q que puede tomar el robot en su entorno de trabajo. El subconjunto C ocupado por el robot R cuando este se encuentra en q , se denota por *R(q)* . Si el robot se modela de forma circular con radio r, *R(q)* se define como:

$$R\,(q) = \{q_i \in C / \, \|q, q_i\| \leqslant r\} \tag{2.2}$$

Si consideramos al vehículo como un punto ideal, la expresión nos quedaría:

$$R\,(q) = \{q_i\} \tag{2.3}$$

En el entorno de trabajo definido, podemos definir un conjunto B que comprende objetos rígidos representados como puntos que se encuentran arbitrariamente distribuidos en el espacio de configuraciones C.

$$B = \{b_1, b_2, b_3, ..., b_n\} \tag{2.4}$$

El conjunto de configuraciones del espacio C ocupadas por un obstáculo se define por $b_i(q)$, de tal forma que el subconjunto de configuraciones de C, que especifican el espacio libre de obstáculos viene dado por:

$$C_1 = \left\{ q \in C/R(q) \cap (\bigcup_{i=1}^{q} b_i(q)) = \oslash \right\} \tag{2.5}$$

Siendo q en el conjunto C_l los puntos en los cuales la intersección del subconjunto que puede ocupar el robot con la unión del subconjunto de obstáculos B da como resultado un conjunto vacío. El objetivo de la planificación es del de buscar una sucesión de puntos q tal que el primero de ellos sea el punto inicial q_i y el último de la sucesión de puntos sea q_f. Todos los puntos intermedios necesariamente deben pertenecer al subconjunto de configuraciones de espacios libres. Por lo que una ruta Q_r que conecta el primer punto q_i con el final q_f está dada por:

$$Q_r = \{q_i, q_j, q_{j+1}, ..., q_f/qj \in C_l\} \tag{2.6}$$

La especificación de este conjunto implica la construcción de una función ruta definida como:

$$\tau : [0, 1]x \rightarrow C_l \tag{2.7}$$

Siendo:

$$\tau(0) = q_i \tau(1) = q_f \tag{2.8}$$

La función ruta a su vez debe ser continua para tener consistencia con el movimiento del vehículo, esto se ve reflejado en la siguiente expresión:

$$\lim_{s \rightarrow s_0} \|\tau(s), \tau(s_0)\| = 0 \tag{2.9}$$

A continuación, se presenta un estudio de diferentes algoritmos para la generación de trayectorias, existiendo diferentes métodos que ya han sido implementados y son eficientes en la planificación de caminos, como por ejemplo, el método de descomposición por celdas o el algoritmo dijkstra. siendo utilizado en este proyecto una variación del método de búsqueda A Star (A*) debido a sus ventajas en esta aplicación específica frente a los algoritmos previamente nombrados.

2.3.3. Clasificación de algoritmos de path planning

- Determinísticos
  - Algoritmos bug
  - Métodos de Wavefront
  - Campos Potenciales Artificiales
  - Basados en Grafos
    - Grafos de visibilidad
    - Diagramas de Voronoi
    - Modelado del espacio libre
    - Descomposición en celdas
- Probabilísticos o aleatorios
  - Planeador aleatorio de trayectorias
  - Mapas probabilísticos
  - Árboles de exploración rápida
  - Basados en optimización
    - Algoritmos genéticos
    - Colonia de hormigas
    - Enjambre de partículas
    - Quimiotaxis bacteriana

Debido a que por definición el proyecto cuenta con información suficiente, a través de un mapa o por información obtenida por medio de GPS, además cuando es detectado un obstáculo dinámicamente puede modificarse el mapa consecuentemente sin perder exactitud en la representación del entorno por el cual se está trasladando, los métodos determinísticos son los adecuados para esta implementación. Los métodos probabilísticos son algoritmos más complejos computacionalmente, mayormente utilizados en métodos de localización y mapeo simultáneo, es decir, cuando uno debe desplazarse por un espacio pero se desconoce el estado inicial del sistema por lo que debe establecerse en qué punto del espacio se encuentra y a su vez trazar trayectorias para alcanzar el destino. Los métodos determinísticos son menos complejos computacionalmente y se obtienen resultados exactos y no sujetos a ciertas probabilidades de error, siempre y cuando el sistema tenga la información necesaria para poder implementarlos.

Dentro de los algoritmos de trazado de trayectorias determinísticos el algoritmo bug no ha sido considerado porque si bien matemáticamente se demuestra su convergencia y, además, si existe un camino válido entre el punto de inicio y el final lo encontrará, este algoritmo no asegura que ese camino sea el más óptimo. El método de campos potenciales artificiales no es óptimo para aplicar en este proyecto particular ya que está basado en técnicas reactivas de navegación y es realmente útiles en entornos desconocidos. Puede ser considerado el mapa que tenemos como entrada al sistema como un grafo y realizar el análisis en base a esto. Por estas razones, serán desarrolladas tres métodos de trazado de trayectorias basadas en grafos: descomposición de celdas, el algoritmo de Dijkstra y el algoritmo A-star.

2.3.4. Algoritmos de búsqueda en grafos

Un grafo, representa un conjunto de vértices unidos en una red a través de aristas. Si dos vértices están unidos, al viajar de uno a otro se consideraría sucesor el vértice de destino, y predecesor el vértice origen. Además, normalmente existirá un coste vinculado al desplazamiento entre vértices. A continuación, una definición más formal de un grafo:

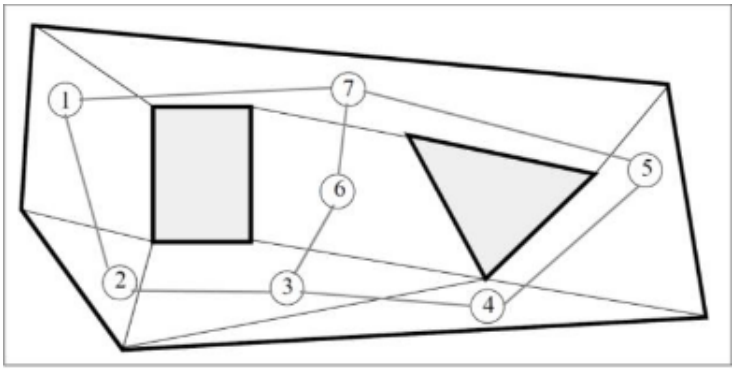
Un grafo G es un conjunto no vacío de vértices V y un conjunto no vacío de aristas A extraído de la colección de subconjuntos de dos elementos de V. Una arista de G es un subconjunto E compuesto por {a,b} con $a, b \in V, a \neq b$

Un algoritmo de búsqueda tratará de encontrar un camino óptimo entre dos vértices como por ejemplo un camino que minimice el coste de desplazamiento, o el número de pasos a realizar. La principal diferencia entre los algoritmos es la información que guardan acerca del grafo. Algunos de ellos no guardan información alguna, simplemente expanden la búsqueda desde el vértice inicial, hasta que se llega al vértice final, otros guardan el coste de viajar desde el origen hasta ese vértice, o incluso una estimación de lo prometedor que es un vértice para conducir el camino a su objetivo. La expansión de la búsqueda se realiza en forma de árbol. Partiendo del vértice inicial, se extenderá la búsqueda a sus vértices vecinos, de cada uno de estos vértices vecinos, a sus respectivos vértices vecinos, y así hasta que uno de los vértices a los que se expande la búsqueda es el vértice objetivo. En esta página se desarrollará un algoritmo de búsqueda lo suficientemente general para trabajar en la mayoría de los grafos, y que da paso a otros métodos de búsqueda más complejos.

2.3.5. Algoritmo de Descomposición de celdas

Este método se fundamenta en una descomposición en celdas del espacio libre. Por lo tanto, la búsqueda de una ruta desde una postura inicial hasta otra final, consiste en encontrar una sucesión de celdas que no presente discontinuidades, tal que la primera de ellas contenga el punto de inicio y la última el punto final . Este método no encuentra una serie de segmentos que modele la ruta, sino una sucesión de celdas; por ello, se hace necesario un segundo paso de construcción de un grafo de conectividad, encargado de definir la ruta . La primera etapa implica construir unas celdas con determinada forma geométrica tal que resulte fácil de calcular un camino entre dos configuraciones distintas pertenecientes a la celda, y la comprobación para averiguar si dos celdas son adyacentes debe disfrutar de la mayor simpleza posible. Aparte de estas características, la descomposición global del espacio libre implica que no deben existir solapamientos entre celdas y que la unión de todas ellas corresponde exactamente al espacio libre. El grafo de conectividad es un grafo no dirigido, y su construcción está asociada a la descomposición en celdas efectuada en el paso anterior, de tal forma, que los vértices van a ser cada una de las celdas, existiendo un arco entre dos celdas si y sólo si son adyacentes.

Figura 2.4: Descomposición de Celdas[15]



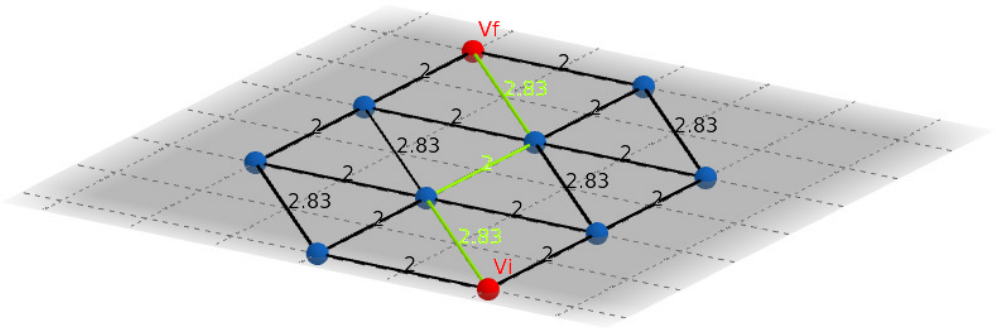
Una vez especificado el grafo de conectividad, sólo queda emplear un algoritmo de búsqueda en grafos para la detección de la celda que contiene la postura a la cual se desea llegar, tomando como partida la que contiene la postura inicial. Los distintos métodos basados en este principio, se distinguen por la forma en la cual realizan la descomposición en celdas y cómo se construye el grafo de conectividad. El método más sencillo de descomposición del espacio libre del entorno en celdas resulta el denominado descomposición trapezoidal. Este método se basa en la construcción de segmentos rectilíneos paralelos al eje Y del sistema global a partir de

los vértices de cada uno de los elementos del entorno. El final del segmento queda delimitado por el primer corte de la línea con un elemento del entorno.[15]

2.3.6. Algoritmo de Dijkstra

El Algoritmo de Dijkstra permite encontrar el camino más corto entre dos vértices de un grafo. Se tiene un grafo G = (V,E) como un conjunto de vértices V donde $V = \{v_0, v_1, ..., v_{n-1}\}$ y un conjunto de aristas E que unen los vértices, a los cuales se les asocia un peso. El objetivo del algoritmo es encontrar una serie de vértices que comiencen en el vértice inicial v_i y concluyan en el vértice final v_f con la menor distancia entre ellos.

Figura 2.5: Algoritmo Dijkstra



El método descrito en pseudo código es el siguiente:

1. Se inicializan tres variables:
   a) Un arreglo de distancias llamado D desde el vértice de inicio v_i a cada uno de los vértices del grafo, siendo el valor para el vértice de inicio cero y para los demás vértices infinito. Esto es sólo al comienzo, después se calcularán las distancias desde el vértice utilizado hacia los demás contiguos.
   b) Una cola Q de todos los vértices del grafo. En el comienzo del algoritmo tendrá todos los vértices, cuando finalice Q estará vacía.
   c) Un set vacío S, que indicará los vértices que han sido evaluados. Al final de la ejecución del algoritmo, este set tendrá todos los vértices del grafo.
2. El algoritmo itera de la siguiente manera hasta que Q esté vacía:

a) Mientras Q no esté vacía, se toma un vértice de la cola que no esté en S y que tenga el valor de distancia menor respecto al vértice que se está tomando en cuenta.

b) Se añade el vértice al set S, debido a que ya se ha tenido en cuenta.

c) Se actualizan los valores de distancias para cada vértice adyacente al nuevo vértice considerado teniendo en cuenta lo siguiente:

1) Si la distancia hasta el vértice anterior más el peso de la arista entre el vértice anterior y el actual es menor a la distancia en el vértice actual (recordando que se ha inicializado en infinito) se actualiza el vector D con el valor.
2) Si no se cumple esto, no se actualiza el valor de distancias. Siempre uno de los vértices adyacentes va a tener una distancia menor a infinito y a su vez se va a actualizar con el que su arista tenga menor peso.

La complejidad computacional del algoritmo de Dijkstra se puede calcular contando las operaciones realizadas:

- El algoritmo consiste en n-1 iteraciones, como máximo. En cada iteración, se añade un vértice al set S.
- En cada iteración, se identifica el vértice con la menor distancia entre los que no están en S. El número de estas operaciones está acotado por n-1.
- Además, se realizan una suma y una comparación para actualizar la distancia de cada uno de los vértices que no están en S.

Es decir, en cada iteración se realizan a lo sumo $2(n-1)$ operaciones. El algoritmo de Dijkstra realiza $O(n^2)$ operaciones (sumas y comparaciones) para determinar la longitud del camino más corto entre dos vértices de un grafo.

2.3.7. Algoritmo A Star

A star (A*) es un algoritmo que se clasifica dentro de algoritmos de búsqueda basados en grafos, particularmente grafos de visibilidad, y su función es encontrar la trayectoria en un espacio euclídeo de dos dimensiones eficiente para conectar un vértice de inicio con un vértice final . Es utilizado en una gran cantidad de proyectos diferentes por su baja complejidad computacional y precisión en los resultados. Peter Hart, Nils Nilsson

and Bertram Raphael of SRI (Stanford Research Institute) escribieron por primera vez acerca de este algoritmo. Es una extensión del Algoritmo Dijkstra con la ventaja de utilizar funciones heurísticas de costo que permiten que el algoritmo sea más óptimo en su ejecución y los resultados sean más precisos. El algoritmo Dijkstra puede verse como un algoritmo A star donde h(x) siendo x cualquier vértice del grafo vale cero, por lo que, el desarrollo previo del algoritmo Dijkstra es comprendido por el algoritmo A star.

2.3.7.1. Desarrollo

El algoritmo A star trabaja realizando un árbol de caminos con el coste más bajo desde el vértice de inicio hasta el vértice objetivo. Lo que diferencia a este algoritmo y lo hace más óptimo para la mayoría de las búsquedas es que para cada vértice utiliza una función que indica un estimado del costo total del camino utilizando el vértice siguiente. Además, la diferencia respecto de algoritmos que se guían exclusivamente por funciones heurísticas es que el algoritmo A* tiene en cuenta dos factores: el costo estimado heurístico de los vértices y el costo real del recorrido. Esto es una ventaja importante ya que las funciones heurísticas son sólo una estimación, por lo que, pueden no ser correctas y desembocar en caminos no óptimos. La expansión de los caminos a través de los vértices más óptima se basa en la función de evaluación:

$$f(n) = g(n) + h(n) \tag{2.10}$$

Donde:

- $f(n)$ =Costo estimado total del camino a través del vértice n
- $g(n)$ =Costo real para alcanza el vértice
- $h(n)$ =Costo estimado desde el vértice n hasta el vértice objetivo. Esta es la parte heurística de la función, una aproximación del costo que podría tener el camino

A* mantiene dos estructuras de datos auxiliares, que podemos denominar abiertos, implementado como una cola de prioridad (ordenada por el valor $f(n)$ de cada vértice), y cerrados, donde se guarda la información de los vértices que ya han sido visitados. En cada paso del algoritmo, se expande el vértice que esté primero en abiertos, y en caso de que no sea un vértice objetivo, calcula la $f(n)$ de todos los vértices contiguos, los inserta

en abiertos, y pasa el vértice evaluado a cerrados. El algoritmo es una combinación entre búsquedas del tipo primero en anchura (BFS) con primero en profundidad (DFS): mientras que $h(n)$ tiende a primero en profundidad, $g(n)$ tiende a primero en anchura. De este modo, se cambia de camino de búsqueda cada vez que existen vértices más prometedores.

Figura 2.6: Ejemplo A Star aplicado a una matriz

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

2.3.7.2. Pseudocódigo A Star

El algoritmo, desarrollado en pseudo-código queda planteado de la siguiente manera:

1. Los datos de entrada del algoritmo son: el grafo con vértices con obstáculos y vértices navegables, el vértice de inicio (v_i), el vértice de destino (v_f), la función heurística para cada vértice y la función costo para cada vértice.

2. Se utilizan dos listas:

   a) Una lista llamada Vértices abiertos (VA) que contiene los vértices que han sido considerados en la búsqueda pero que aún no se ha expandido en sus vértices contiguos. Puede verse como una lista de tareas pendientes. Esta lista contiene la posición (x,y), el valor que toma la función heurística y la función costo de cada vértice en ella.

   b) Otra lista llamada Vértices Cerrados (VC) que contiene los vértices que han sido considerados y que se han expandido sus vértices contiguos.

3. Inicialmente se toma el vértice de inicio (v_i) en la lista VA utilizando como valor f el valor de la función heurística $h(v_i)$.

4. Luego se itera hasta que la lista VA deje de estar vacía, es decir, que no existan más "tareas pendientes". Dentro de estas iteraciones se realizan las siguientes acciones:

   a) Se toma de la lista VA el vértice con el menor costo estimado dado por la suma del costo de la arista más el valor que toma la función heurística valuada en ese vértice, calculados anteriormente.

   b) Si el vértice que hemos seleccionado es el vértice de destino v_f, la solución ha sido encontrada y se termina el algoritmo.

   c) Si no es el vértice de destino se evalúan los vértices adyacentes al mismo. Para cada vértice adyacente que no esté en la lista VC (es decir, que ya se haya evaluado, así como sus vértices adyacentes) se calcula el costo estimado total y se lo agrega a la lista de VA.

   d) Luego se calcula en costo estimativo comprendido por el costo real del vértice actual más la distancia entre el vértice de menor costo seleccionado anteriormente para cada uno de ellos. Si este costo estimativo calculado es mayor o igual que el costo real del vértice adyacente evaluado, este no es el mejor candidato para realizar una trayectoria óptima.

   e) Si el costo estimativo cumple los requisitos anteriores, este vértice es almacenado para luego extraer la trayectoria y se vuelve a iterar hasta encontrar el vértice de destino.

2.3.7.3. Complejidad del algoritmo, evaluación del impacto de la función heurística

La complejidad del algoritmo está asociada con la función heurística, en el peor de los casos con una heurística muy pobre, la complejidad del algoritmo es exponencial $O(n^2)$, igual que el algoritmo Dijkstra. Mientras que utilizando una función heurística precisa, la complejidad del algoritmo tiende a ser lineal. Por estos motivos es importante hacer una elección acertada de la heurística: una buena aproximación de h hará al algoritmo mucho más eficiente.
El criterio para determinar si una función heurística es aceptable o no, es dado por la admisibilidad de la misma.Se define como admisible a una función heurística si nunca sobrestima el verdadero coste de llegar hasta un vértice de destino, es decir, es admisible si la estimación de costos es

optimista y no toma valores superiores al real. Un ejemplo de una función heurística admisible la distancia en línea recta de cada vértice con el origen.

$$h(x, y) = \sqrt{(v_{xj} - v_{xi})^2 + (v_{yj} - v_{yi})^2} \tag{2.11}$$

En la figura podemos ver la representación de esta función considerando el vértice (0,0) como el de inicio.

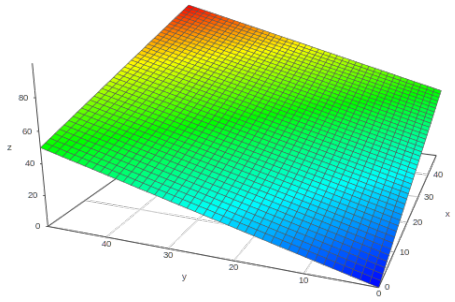


Figura 2.7: Heurística Admisible

Un ejemplo antagónico sería el de una función heurística exponencial, dada por la siguiente fórmula:

$$h(x, y) = (v_{xj} - v_{xi})^2 + (v_{yj} - v_{yi})^2 \tag{2.12}$$

Una representación de esta función puede observarse en la siguiente figura:

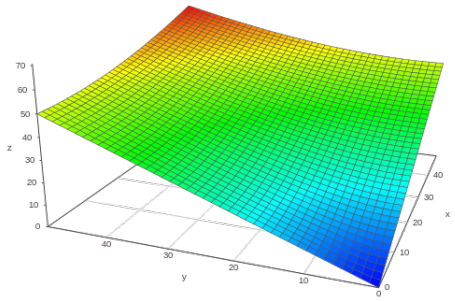


Figura 2.8: Heurística Inadmisible

Una definición matemática de la influencia de la heurística en la complejidad sería:
Sea la función $h*(v)$ el coste mínimo real de una solución que pase por el vértice v y sea la función $\delta = \frac{h*-h}{h*}$, la función δ será el error relativo de $h()$ respecto a $h*(v)$. Si $\delta = 0$ entonces la función heurística es perfecta En estas condiciones, el número de vértices que expandirá el algoritmo dependerá del error relativo de la función heurística. Si tenemos error cero el algoritmo localiza la solución de forma inmediata y si el error es 1 (función heurística nula) entonces el algoritmo estará en las mismas condiciones que en una búsqueda ciega con coste uniforme. La complejidad del algoritmo en tiempo y espacio será del orden de $O(j^{\delta d})$, donde j es el número promedio de descendientes en cada vértice y d es la profundidad necesaria para llegar al destino en la solución óptima.

2.3.7.4. Algoritmo de suavizado de trayectoria

El algoritmo A star nos provee una serie de vértices conectados cuyo nodo de inicio y de final son la posición inicial del vehículo y el destino objetivo que se desea, respectivamente. Este modelo de trayectoria no es adecuado para una implementación real de un vehículo en movimiento, ya que, es un modelo discreto y para poder seguir el trayecto el robot debería rotar de a pasos de 45° para alcanzar el objetivo. En la siguiente figura se observa un ejemplo gráfico de la salida del sistema luego del algoritmo A star, siendo la línea azul la trayectoria y los puntos rojos los vértices que representan obstáculos:

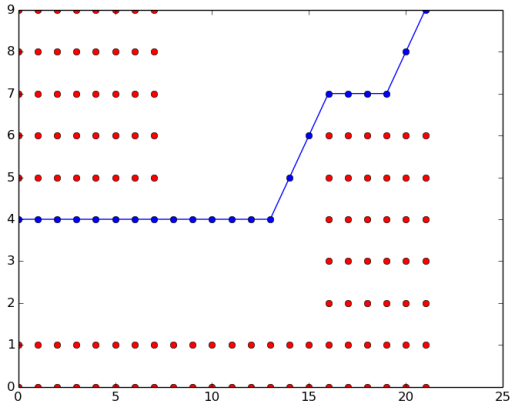


Figura 2.9: Trayectoria A star

Debido a que el modelo de movimiento del robot dificultará en demasía el desplazamiento en trayectos de este tipo es necesaria una función que suavice la trayectoria, manteniendo los postulados de realizar el camino más óptimo sin producirse colisiones.La idea general del algoritmo de suavizado es la de una suma ponderada de diferentes restricciones que deseamos en la trayectoria. Una restricción es la distancia entre un vértice y sus vértices contiguos en la trayectoria. Esta restricción va a tender a forzar a la trayectoria a ser una recta que conecta el punto de inicio con el punto final deseado. Esta restricción puede verse en la figura donde el punto X_1 es trasladado de su posición original calculado con el algoritmo A star hacia un punto intermedio entre sus dos vértices contiguos. Esto puede representarse como una función iterativa de la siguiente forma:

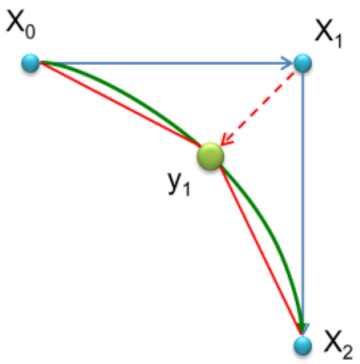


Figura 2.10: Suavizado

$$y_1 = y_1 - \alpha * ((x_0 - y_1) + (x_2 - y_1)) \tag{2.13}$$

$$y_1 = y_1 - \alpha * (x_0 + x_2 - 2 * y_1) \tag{2.14}$$

Donde α es la ganancia que se le da a esta restricción en particular pudiendo tomar valores de cero a uno. Si esta ponderación es cero la trayectoria quedará como la original, y si es uno trazará líneas rectas (en el caso de la figura trazará una línea de 45°) entre el punto de inicio y el punto final. Otra restricción que es útil agregar es la distancia entre el punto analizado y el punto de origen, la misma forzará a la trayectoria a mantener su forma original. Por si sola esta restricción no tendría inferencia en el trayecto original, pero combinada con la anterior restricción nos da mayor sensibilidad para establecer cuán agresiva será la función de suavizado evitando que sea solo una línea recta entre dos puntos. La función que define esta restricción es:

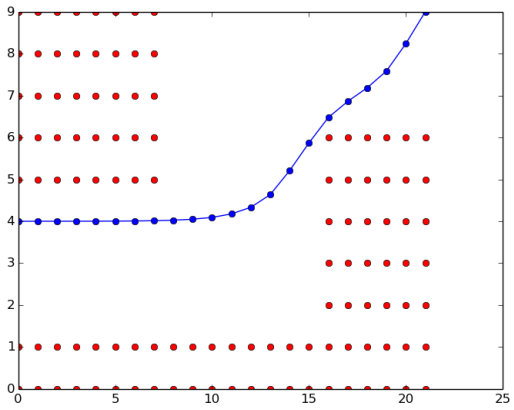
$$y_1 = y_1 - \beta * (x_1 - y_1) \tag{2.15}$$

β = ganancia de la restricción

Por lo que el algoritmo itera hasta que se alcance una tolerancia máxima de diferencia entre el punto original y el punto suavizado. Esta tolerancia depende de la magnitud de las distancias que estamos considerando en el grafo y los requerimientos del diseño. En la siguiente figura podemos

observar la aplicación del algoritmo de suavizado al problema planteado inicialmente en el desarrollo:

Figura 2.11: Trayectoria Filtrada



2.4. Robótica

2.4.1. Definición

La robótica es una ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia. La robótica conjuga múltiples disciplinas de la ciencia como: el álgebra, los autómatas programables, las máquinas de estados, la mecánica, la electrónica y la informática. Esto hace que el estudio de la robótica sea complejo, puesto que se debe tener destreza en diversos campos de la ingeniería.

Un robot es un manipulador reprogramable y multifuncional concebido para transportar materiales, piezas, herramientas o sistemas especializados; con movimientos variados y programados, con la finalidad de ejecutar tareas diversas dentro de un entorno de trabajo. En 1921 Karel Capek en su obra R.U.R (Los Robots Universales de Rossum) introdujo el término "robot", que en checo es asociada con la palabra "robota" que quiere decir trabajo forzado.[10]

Los componentes esenciales de un robot son:

- Sensores
  Brindan información del entorno al robot. Proporcionan información

de variables físicas del entorno a través de señales eléctricas que pueden ser digitales o analógicas. Es importante realizar un tratamiento adecuado de estos datos para que la unidad de procesamiento pueda tomar decisiones acertadas. Ejemplos de sensores pueden ser:

- Sensor Acelerómetro
- Termómetro electrónico
- Sensor de Proximidad

- Actuadores
  Son el medio por el cual el robot puede influir sobre el entorno. En general, los actuadores son los encargados de generar el movimiento de los diferentes mecanismos o elementos que conforman el robot y son accionados por la unidad de procesamiento de manera directa o indirecta. En la mayoría de los casos los actuadores no son controlados directamente por la unidad de procesamiento sino que existe una etapa de potencia que adecua la señal proveniente de la parte de control. Ejemplos de actuadores:

  - Motores CC
  - Llaves electrónicas
  - Resistencias Calefactores

- Unidad de Procesamiento
  Es la encargada de conjugar sensores y actuadores. Teniendo como referencia datos del entorno a través de sensores, toma decisiones en base a esto y genera señales de control hacia los actuadores con objetivo de cumplir una tarea específica.
  Ejemplos de unidades de procesamiento:

  - Microcontroladores
  - FPGA
  - Microprocesador
  - Circuitos analógicos/digitales

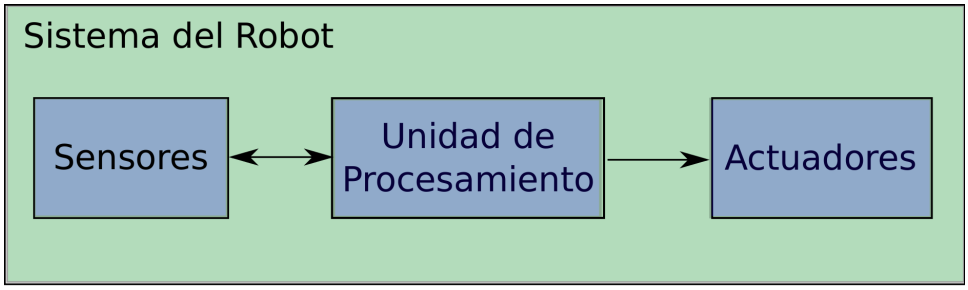


Figura 2.12: Diagrama general Robot

2.4.2. Clasificación

De acuerdo a su grado de autonomía los robots se pueden clasificar como teleoperados, de funcionamiento repetitivo y autónomos o inteligentes.

- Robots teleoperados
  Se denominan robots teleoperados a los cuales las tareas de percepción, planificación y ejecución depende de un ser humano. Existe una realimentación sensorial del entorno, que puede tomar diferentes formas e informa el estado del robot a través de diferentes variables como pueden ser: velocidad, fuerza, aceleración, estado del vehículo. Los sistemas más avanzados de robots teleoperados necesitan del usuario solo la acción de decisión, mientras que manejan todo lo que es control a bajo nivel de forma autónoma.

- Robots de funcionamiento repetitivo
  Son la mayor parte de los que se emplean en cadenas de producción industrial. Trabajan normalmente en tareas predecibles e invariantes, con una limitada percepción del entorno. Son precisos, de alta repetibilidad y relativamente rápidos; incrementan la productividad ahorrando al hombre trabajos repetitivos y, eventualmente, muy penosos o incluso peligrosos.

- Robots autónomos o inteligentes
  Son los más evolucionados desde el punto de vista del procesamiento de información. Son máquinas capaces de percibir, modelar el entorno, planificar y actuar para alcanzar objetivos sin la intervención, o con una intervención mínima de supervisores humanos. Pueden trabajar en entornos poco estructurados y dinámicos, realizando acciones en respuesta a contingencias variadas en dicho entorno. Durante las últimas décadas se han realizado importantes esfuerzos en la aplicación de técnicas de inteligencia artificial. Se han empleado métodos simbólicos

de tratamiento de la información basados en modelos geométricos del entorno.
De esta forma, se resuelven problemas basados en un modelo previo del entorno cuyas soluciones sólo son válidas si el modelo corresponde exactamente a la realidad. La técnica obvia de reducir esta incertidumbre consiste en incrementar la información de que se dispone de dicho entorno mediante realimentación sensorial. Existen métodos que permiten intercalar la formulación y ejecución de planes con la captación de la información necesaria para asegurar que el modelo que se utiliza para la planificación sea lo suficientemente fiable. Las limitaciones vienen impuestas por el sistema de percepción y por la propia arquitectura del sistema de información y control del robot.
Desde el punto de vista de la planificación, existen diferentes arquitecturas diseñadas teniendo en cuenta especificaciones sobre el tiempo que tiene el sistema para responder y la disponibilidad de información potencialmente interesante. La solución se sitúa normalmente entre dos extremos, en uno de los cuales está la planificación puramente estratégica. En este caso, se supone que la situación en la que va a ejecutarse el plan puede ser predicha de forma suficientemente precisa durante la planificación. En el otro extremo se sitúa la planificación puramente reactiva en la que se supone que el entorno es incierto, buscándose la mayor flexibilidad posible para reaccionar en cualquier instante lo suficientemente rápido a las discrepancias entre el modelo actual y la realidad observada en el entorno.[1]

De acuerdo a la arquitectura del robot en cuestión, puede clasificarse en una de las siguientes clases:

- Robots poliarticulados
  Constituyen un grupo que incluye robots de muy diversa forma y configuración cuya característica común es la de ser básicamente sedentarios (aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados) y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas y con un número limitado de grados de libertad. En este grupo se encuentran los manipuladores, los robots industriales, y se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o reducir el espacio ocupado en el suelo.

- Robots móviles
  Son robots con grandes capacidades de desplazamiento, basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores. Estos robots, en entornos industriales aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia

Los robots móviles pueden clasificarse según su sistema de locomoción teniendo en cuenta el medio de movimiento y configuración. Los medios de movimiento más utilizados son: por ruedas, por patas o por orugas. El movimiento por ruedas tiene amplias ventajas respecto a los otros dos debido a su eficiencia en relación a energía sobre superficies lisas y firmes, además, no causan desgaste en la superficie por la cual se desarrolla su traslado y requieren un número menor de partes, habitualmente menos complejas en comparación con los robots de patas y orugas.
Dentro de los robots cuyo medio de locomoción son ruedas pueden encontrarse diferentes tipos de configuración dependiendo el movimiento que pueden realizar las mismas, la ubicación de los actuadores y ejes. Las configuraciones más importantes son:

1. Ackerman: esta configuración tiene cuatro ruedas de las cuales las dos ruedas traseras están fijas a través de un eje transversal y tienen actuadores que sólo le permiten avanzar y/o retroceder. A su vez cuentan con dos ruedas en la parte delantera fijas a un eje que tiene la posibilidad de rotarlas para poder describir un giro sobre el eje Z.

2. Triciclo clásico: esta configuración tiene tres ruedas de las cuales las dos traseras están fijas a un eje sin actuadores y la restante, ubicada al frente, tiene la posibilidad de avanzar o retroceder y girar sobre el eje Z para describir un giro en la trayectoria.

3. Tracción diferencial: cuenta con dos ruedas accionadas por actuadores independientes por lo que el sincronismo entre el movimiento de las mismas le permite al robot avanzar,retroceder o rotar sobre el eje Z.

4. Skid steer: en este tipo de configuraciones se dispone de varias ruedas en cada lado del vehículo que actúan de forma simultánea. El movi-

miento es el resultado de combinar las velocidades de las ruedas de la izquierda con las de la derecha

5. Tracción síncrona: cuenta con ruedas cuyo actuador es único y pueden ser rotadas de manera síncrona para cambiar la trayectoria.

6. Tracción omnidireccional: cada una de las ruedas tiene un actuador propio y además pueden girar independientemente para describir un giro en el plano x e y. Cada una de ellas proporciona al robot una fuerza normal al eje del motor y paralela a la superficie sobre la cual se desplaza. La suma de ellas permite la traslación y rotación de la estructura. Por lo general, presentan una configuración mecánica de tres o cuatro ruedas

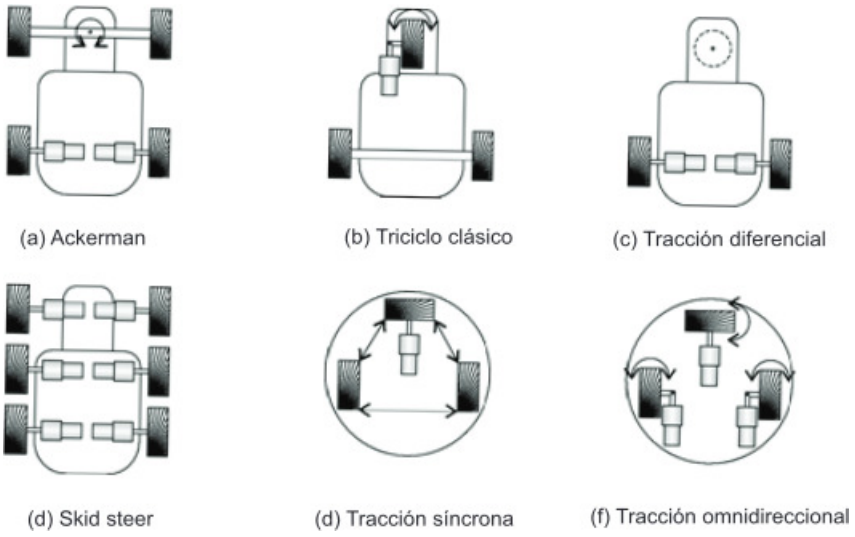


Figura 2.13: Configuraciones de movimiento [1]

2.4.3. Modelado del movimiento de un robot

En esta sección se realizará el estudio cinemático del movimiento de un robot de cuatro ruedas basado en un sistema de dirección Ackerman con ejes de tracción y dirección independientes, idéntico al sistema empleado por autos comerciales.

2.4.3.1. Sistema de dirección Ackerman

El objetivo de un sistema de dirección es permitir al vehículo mantener o cambiar el curso de la trayectoria que está realizando. Se busca que el sistema sea óptimo en cualquier situación.
Cuando un vehículo de cuatro ruedas cambia el curso de la trayectoria

tomando una curva, cada rueda toma la dirección que le marca su eje. Si el centro de giro de unas ruedas es distinto al de otras se produce un deslizamiento indeseado en las mismas que produce desgaste y se producen fuerzas laterales indeseadas provocadas por este fenómeno.
La geometría Ackerman busca que las cuatro ruedas tengan el mismo centro de giro, es decir, que líneas perpendiculares a cada rueda se intersecten en un mismo punto para lograr el mínimo deslizamiento lateral parásito de las mismas.

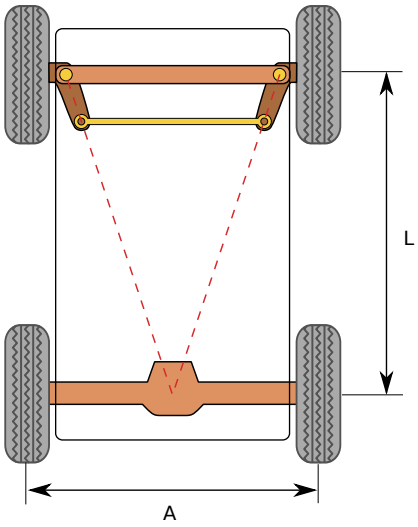


Figura 2.14: Geometría Ackerman[11]

Los ejes de las ruedas traseras siempre están en la misma línea: la prolongación del eje trasero.
Los ejes de las ruedas delanteras giran un ángulo distinto entre ellos, en concreto la rueda interior gira más grados que la exterior, es decir, que adquiere divergencia al girar el volante. Si el cruce de los ejes delanteros se produce justo en la prolongación del eje trasero, las cuatro ruedas giran alrededor del mismo punto. Esto es lo que se conoce como Geometría de Ackerman.

Para simplificar el análisis matemático puede definirse una rueda imaginaria entre las ruedas sobre el eje delantero y definir el centro instantáneo de curvatura respecto a la misma.

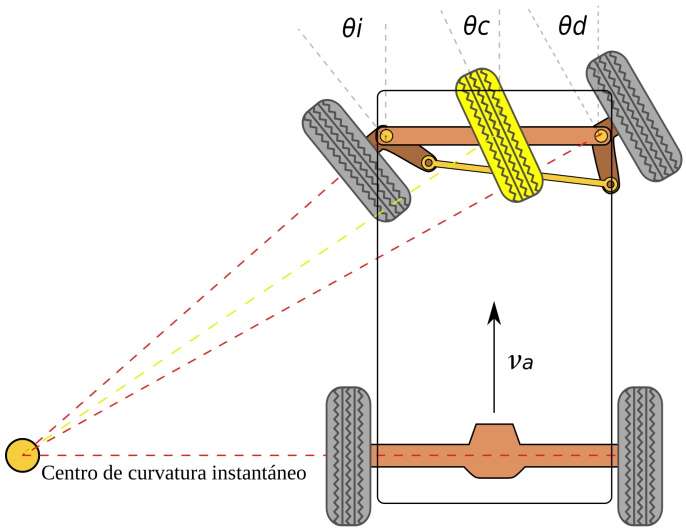


Figura 2.15: Centro de curvatura Ackerman[11]

Los datos recibidos por un robot basado en un sistema de movimiento Ackerman son la velocidad de traslación del vehículo v_a y el ángulo a girar θ_c. Si se denomina la entrada al sistema u_a, puede ser representada en la ecuación (2.16).

$$u_a = \begin{pmatrix} v_a \\ \theta_c \end{pmatrix} \tag{2.16}$$

Teniendo como dato el largo del vehículo L y la entrada θ_c se puede obtener el radio de curvatura r_c en función de la rueda imaginaria a través de trigonometría.

$$r_c = L * \arctan(90^\circ - \theta_c) \tag{2.17}$$

Esta ecuación es válida si θ_c adquiere valores entre $-\frac{\pi}{2}$ y $\frac{\pi}{2}$.
Para calcular cuántos grados debe girar cada una de las ruedas reales consideramos la distancia entre los puntos de giro de cada una de ellas con la rueda central imaginaria considerando dónde se encuentra el centro de curvatura instantáneo. Es decir, como muestra la figura 2.15 la rueda izquierda estará más cerca del centro de curvatura instantáneo por lo que debemos restarle al radio de curvatura r_c la distancia entre su punto de giro y el punto de giro del centro del eje delantero que fue representada por la rueda imaginaria. Utilizando nuevamente trigonometría podemos obtener los ángulos que deberán ser rotadas las ruedas para conseguir un giro con un ángulo total θ_c.

$$\theta_i = \arctan(\frac{L}{r_c - \frac{A}{2}}) \tag{2.18}$$

$$\theta_d = \arctan(\frac{L}{r_c + \frac{A}{2}}) \tag{2.19}$$

2.5. Posicionamiento Relativo del Vehículo. Sensores.

Un coche autónomo es un vehículo capaz de imitar las capacidades humanas de manejo y control, percibiendo el medio que le rodea y desplazándose en consecuencia. Es decir, un coche en el que ningún ocupante del vehículo ha de preocuparse por conducir; sólo se ha de introducir la dirección de destino y despreocuparse por el resto.
Para lograr esto, es necesario contar con diversos sensores que provean al vehículo información de su entorno y de cómo se encuentra ubicado dentro del mismo. Concretamente se necesitan sensores para la detección de obstáculos, unidades de medidas inerciales para la medición de velocidades angulares y componentes de fuerzas gravitacionales en cada eje, sistemas de odometría para la medición de distancia recorrida y para el cómputo de la velocidad de traslación, otros sistemas de posicionamiento global como tecnología GPS. Gracias a estos sistemas el coche es capaz de 'ver' lo que pasa a su alrededor y actuar en consecuencia. Los vehículos autónomos generalmente son capaces de recorrer carreteras previamente programadas y requieren una reproducción cartográfica del terreno, si una ruta no está recogida por el sistema se puede dar el caso que no pueda avanzar de forma coherente y normal.

2.5.1. Unidad de Medidas Inerciales

2.5.1.1. Acelerómetro

Un acelerómetro lineal es un sensor inercial que mide la diferencia total entre la aceleración traslacional y la componente de la aceleración gravitatoria a lo largo de su eje de acción; típicamente tiene como unidad de medida [g], donde $1\text{g} = 9,8m/s^2$. Un acelerómetro incluye una masa suspendida mediante un muelle sobre un marco fijo, donde una fuerza externa producida por una aceleración lineal ocasiona un desplazamiento relativo entre la masa de prueba y el marco, creando al mismo tiempo cambios de tensión sobre el muelle de suspensión. Tanto el desplazamiento relativo como la tensión que sufre el muelle de suspensión al deformarse pueden ser usados para medir una aceleración externa.[61]

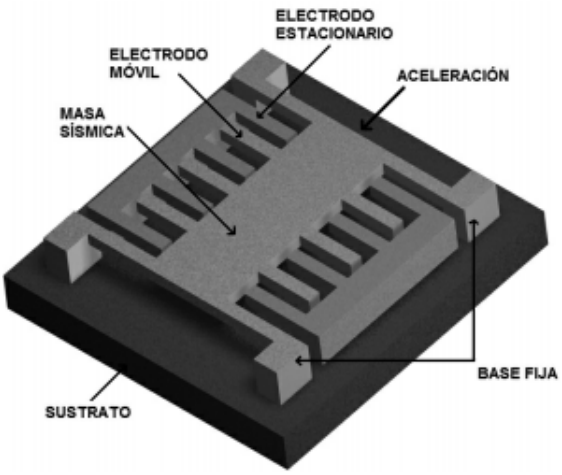


Figura 2.16: Acelerómetro [13]

Los acelerómetros típicamente están especificados por su sensibilidad, máximo rango de operación, resolución, frecuencia de respuesta, no linealidad a máxima escala, offset y supervivencia de choque.
La información que nos brinda el acelerómetro son las fuerzas gravitacionales en cada eje. Si el sensor se encuentra paralelo a la superficie obtendremos 1g en el eje z y 0g en los ejes x e y. Luego, a medida que la plataforma se vaya inclinando hacia un lado o hacia otro esta componente de fuerza de gravedad irá cambiando la magnitud sobre cada eje del sensor.
El objetivo es encontrar los ángulos entre los vectores para determinar la inclinación de la plataforma. Podemos definir estos ángulos como θ_x y θ_y, teniendo en cuenta las fuerzas en cada eje a_x,a_y y a_z y calcularlos a través de la tangente. Vale la pena aclarar que el cociente es entre el módulo de una de las fuerzas y el módulo de las restantes dos.

$$\theta_x = \tan^{-1}\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \tag{2.20}$$

$$\theta_y = \tan^{-1}\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \tag{2.21}$$

2.5.1.2. Giróscopo

Es un sensor inercial que mide velocidad de rotación sobre su propio eje (velocidad angular). El giroscopio logra medir este movimiento rotacional mediante dos principios básicos: vibratorios y ópticos. La resolución, la deriva, la salida en estado estacionario y el factor de escala son características que definen el desempeño del sensor. La resolución del sensor está definida

por el ruido blanco y está expresada en términos de la desviación estándar o velocidad de rotación equivalente por la raíz cuadrada del ancho de banda. °/s / Hz o °/ / Hz , el denominado "angle random walk" puede ser usado también con este fin [°/]. La pequeña variación en la medida del sensor en estado estacionario (sin rotación) define la deriva a corto o largo plazo y está expresada en °/s o °/.
Si se desea obtener posición angular (orientación) con respecto a un ángulo inicial conocido es necesario realizar un proceso de integración en el tiempo de la velocidad angular dada por el sensor. Hay que tomar en cuenta la variación de ángulo en intervalos cortos de tiempo como se muestra en la expresión.

$$d\theta = \omega * dt \tag{2.22}$$

Siendo:

$$d\theta = \text{ángulo medido}$$
$$\omega = \text{velocidad angular medida}$$
$$dt = \text{tiempo de muestreo}$$

Este valor calculado es el ángulo en una variación de tiempo dado, para tener como información el ángulo total de giro alcanzado por el sensor sobre un eje debido a un movimiento debe integrarse estos datos en el tiempo.

$$\theta_k = \theta_{k-1} + \omega_k * dt \tag{2.23}$$

Donde:

$$\theta_k = \text{ángulo total actual}$$
$$\theta_{k-1} = \text{ángulo total anterior}$$
$$\omega_k = \text{velocidad angular actual}$$
$$dt = \text{período de muestreo}$$

Puede observarse que de este tipo de cálculos se desprenden dos inconvenientes principales:

- Deriva: debido a que el giróscopo tiene una deriva propia, acentuándose a causa del método acumulativo utilizado para calcular el ángulo, provoca que la desviación del valor real en la medida del ángulo vaya incrementándose a lo largo del tiempo.

- Estado inicial: los datos que obtengamos a partir del giróscopo son relativos, ya que a falta de conocimiento del estado inicial no puede establecerse una posición absoluta sino una relación con la posición inicial.

2.5.1.3. Fusión Sensorial

La mayoría de los IMU parten de una combinación de acelerómetro y giroscopio, siendo este el IMU más habitual. El motivo es que ambos dispositivos combinan muy bien y compensan las limitaciones del otro.

- Los acelerómetros no tienen deriva (drift) a medio o largo plazo, ya que realizan la medición medida absoluta del ángulo que forma el sensor con la dirección vertical, marcada por la gravedad. Sin embargo, se ven influenciados por los movimientos del sensor y el ruido por lo que no son fiables a corto plazo.
- Los giroscopios funcionan muy bien para movimientos cortos o bruscos, pero al usar giroscopios de vibración que realmente miden la velocidad angular, y obtienen el ángulo por integración respecto al tiempo, acumulan los errores y el ruido en la medición, por lo que a medio o largo plazo tienen deriva (drift).

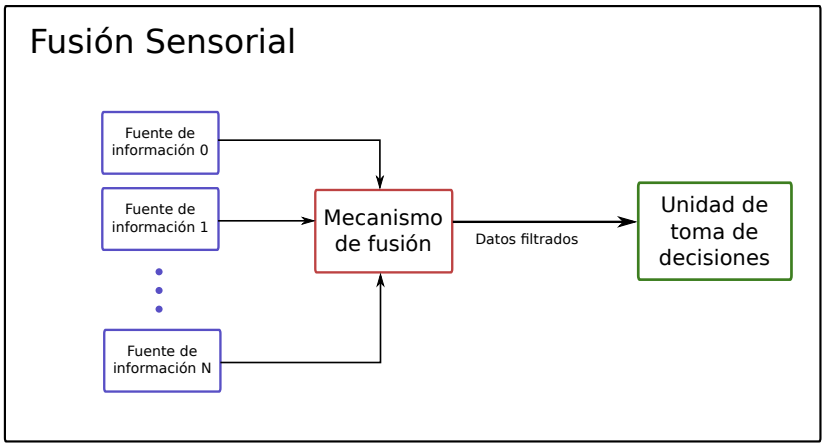


Figura 2.17: Fusión Sensorial

Por tanto combinar las mediciones de ambos dispositivos permite a las IMU obtener mediciones de la orientación más precisas que la de un acelerómetro y un giroscopio por separado.
La aplicación de un filtro Kalman utilizando datos provenientes de una IMU (Unidad de Medidas Inerciales) comprendida por un acelerómetro y

un giróscopo explota la fusión sensorial utilizando un algoritmo predictivo/correctivo, permitiendo reducir el ruido blanco generado por el acelerómetro y predecir la posición real de la plataforma, tomando como referencia sólo un estado anterior (junto con los datos actuales del giróscopo y del acelerómetro).

2.5.1.4. Filtro Kalman

El filtro de Kalman es un algoritmo que sirve para poder identificar el estado oculto (no medible) de un sistema dinámico lineal que sirve cuando el sistema está sometido a ruido blanco aditivo. El filtro de Kalman es capaz de escoger la ganancia K de realimentación del error de forma óptima cuando se conocen las varianzas de los ruidos que afectan al sistema. Ya que el Filtro de Kalman es un algoritmo recursivo, este puede correr en tiempo real usando únicamente las mediciones de entrada actuales, el estado calculado previamente y su matriz de incertidumbre, no requiere ninguna otra información adicional.
El filtro de Kalman tiene numerosas aplicaciones en tecnología. Una aplicación común es la guía, navegación y control de vehículos, especialmente naves espaciales. Además el filtro es ampliamente usado en campos como procesamiento de señales y econometría.
En primer lugar, es importante recordar cuáles son las principales variables de interés en el caso de una plataforma móvil. Se busca estimar la inclinación (x, y), la orientación (z), la velocidad angular (w_x, w_y, w_z), la posición (x, y, z), la velocidad lineal (v_x, v_y, v_z), y la aceleración (a_x, a_y, a_z) de una cierta plataforma.

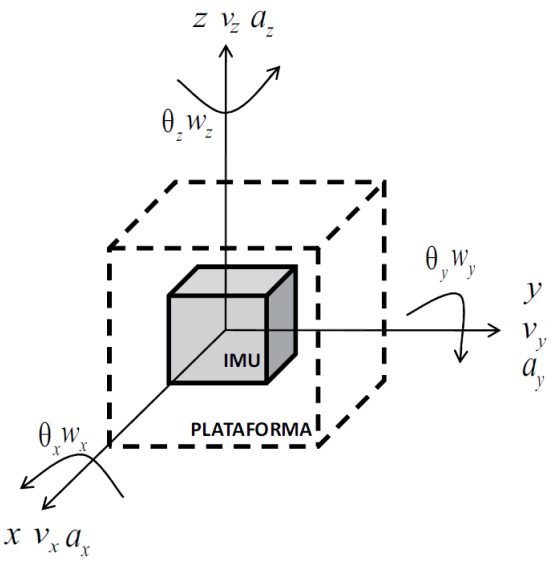


Figura 2.18: Variables de interés IMU

El propósito del filtro Kalman es utilizar mediciones que son adquiridas a lo largo del tiempo y afectadas por variaciones aleatorias (ruido) junto con el conocimiento del comportamiento del sistema, para producir estimaciones que tiendan a estar más cerca del valor real del proceso en cuestión. Todas las mediciones y cálculos basados en modelos son aproximaciones en cierto grado; datos ruidosos provenientes de los sensores, aproximaciones en las ecuaciones que describen cómo cambia el sistema, y otras perturbaciones externas que no son consideradas, introducen cierta incertidumbre en las inferencias realizadas sobre los estados del mismo.
Pueden obtenerse los ángulos de inclinación a partir de las señales del acelerómetro de tres ejes. Adicionalmente, pueden obtenerse las tasas de cambio de los ángulos de Euler a partir de las velocidades angulares de los giróscopos.

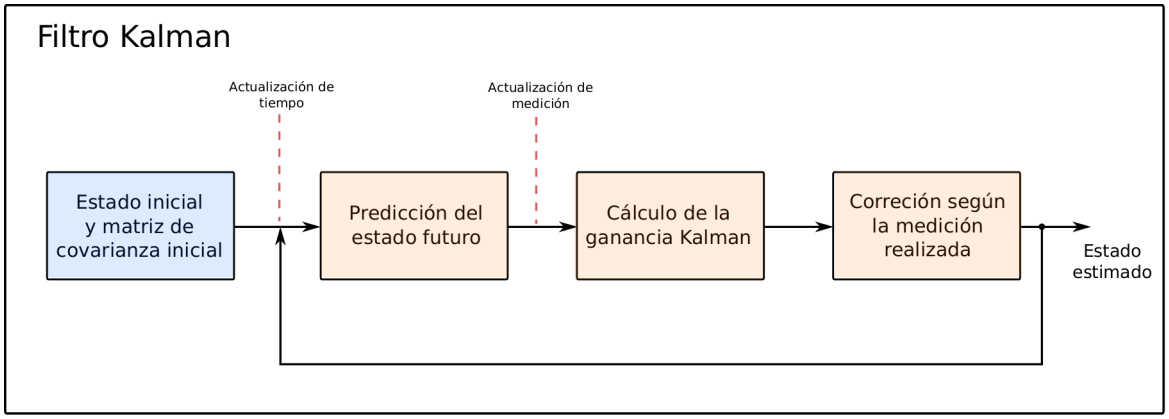


Figura 2.19: Fases de Filtro Kalman

Podemos dividir el grupo de ecuaciones a utilizar según la fase del algoritmo:

1. Ecuaciones de predicción del estado futuro:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k + w_{k-1} \tag{2.24}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2.25}$$

2. Ecuación del cálculo de la ganancia Kalman:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.26}$$

3. Ecuaciones de corrección:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k + H\hat{x}_k^-) \tag{2.27}$$

$$P_k = (I - K_k H)P_k^- \tag{2.28}$$

Donde:
k = instantes de tiempo 0,1,2, ... ,n.
$\hat{x}_k^-$ = estados estimado para el instante k.
$\hat{x}_{k-1}$ = estado en el instante k - 1.
A = matriz de realimentación.
B = matriz de entrada.
w_{k-1} = ruido inherente al proceso.
P_k^- = matriz de covarianza estimada en el instante k.
P_{k-1} = matriz de covarianza en el instante k - 1.
Q = matriz de covarianza de la perturbación del proceso.
R = matriz de covarianza relacionada a la observación.
K_k = ganancia de Kalman.
z_k = medida tomada por el observador.

Podemos observar que la ecuación 2.24 contiene las variables de estado a ser estimadas y representa el modelo del comportamiento del sistema. La ecuación 2.25 representa la estimación de la matriz de covarianza. La ecuación 2.26 representa el cálculo de ganancia kalman con el fin de minimizar la covarianza para la nueva estimación del estado en 2.28. La ecuación 2.27 representa la corrección del estado estimado $\hat{x}_k^-$ en base a la ganancia K y a la innovación en la medida $(z_k - H\hat{x}_k^-)$ dando como resultado un estado corregido $\hat{x}_k$.
La representación de la medición de las variables de estado por parte de

un observador externo,siendo v_k el ruido presente en la medición y H la variable que define la observabilidad de los estados.

$$z_k = H\hat{x}_k^- + v_k \tag{2.29}$$

Si consideramos la ecuación del cálculo del ángulo en el giróscopo 2.23 como modelo del sistema más la presencia del drift del giróscopo (w_k), utilizando las medidas del acelerómetro como como observaciones del sistema y siendo el ángulo θ la variable de estado a corregir:

$$\theta_k^- = A * \theta_{k-1} + B * \omega_k * dt + w_{k-1} * dt \tag{2.30}$$

$$z_k = H * \theta_k^- + v_k \tag{2.31}$$

donde:
A = 1
B = 1
dt = período de muestreo [s]
w_k = velocidad angular medida por el giróscopo [°/s]
θ_k^- = ángulo estimado [° o rad]
z_k = ángulo medido por el acelerómetro [° o rad]
w_{k-1} = deriva del giróscopo [°/s o rad/s]

La ecuación 2.30 representa el estado θ solo en una dimensión, debido a que en el diseño se requiere el ángulo sobre el eje x y sobre el eje y es necesaria el desarrollo matricial con las dos variables:

$$\begin{pmatrix} \theta_{xk}^- \\ \theta_{yk}^- \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_{xk-1}^- \\ \theta_{yk-1}^- \end{pmatrix} + \begin{pmatrix} dt & 0 \\ 0 & dt \end{pmatrix} \omega_k + W_{k-1} \tag{2.32}$$

$$\begin{pmatrix} z_{xk} \\ z_{yk} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_{xk}^- \\ \theta_{yk}^- \end{pmatrix} + V_k \tag{2.33}$$

Siendo R y Q funciones del ruido en la salida en el acelerómetro y la deriva en el giroscopio respectivamente y manteniendo su valor constante a lo largo del tiempo. Nótese además que, en el caso del sistema propuesto las matrices de covarianzas R y Q son diagonales debido a que se asume que no existe correlación alguna entre las mediciones realizadas para los ángulos.

$$R = \begin{pmatrix} v^2 & 0 \\ 0 & v^2 \end{pmatrix} \tag{2.34}$$

$$Q = \begin{pmatrix} (deriva * dt)^2 & 0 \\ 0 & (deriva * dt)^2 \end{pmatrix} \tag{2.35}$$

2.5.2. Sensores de proximidad

2.5.2.1. Introducción

Los sensores de distancia son dispositivos capaces de detectar la presencia de objetos cercanos sin la necesidad de entrar en contacto físico con ellos. Estos están pensados para tomar la medida de distancia lineal o desplazamiento lineal de una forma automatizada, ya que proporcionan una señal eléctrica proporcional a la variación física, en este caso de distancia. Pueden tener un rango de medida muy variable que va a depender del sensor empleado, existen modelos que miden pocas micras y otros capaces de medir cientos de metros.
Existen muchos tipos de sensores de proximidad, entre ellos se encuentran sensores inductivos, de efecto hall, capacitivos, ultrasónicos y ópticos. [52]

2.5.2.2. Sensores ópticos

Estos sensores están diseñados para proveer una señal de salida proporcional a la distancia a la que se encuentra el objeto, independientemente de su forma, color, material, entre otros. Existen diversas técnicas para diseñar este tipo de transductores, siendo los más utilizados triangulación y medición de tiempo de vuelo.
Comercialmente se pueden encontrar varios tipos, dependiendo si son para grandes y/o pequeñas distancias. En casi todos ellos los componentes semiconductores estándar son utilizados como emisores y receptores, llamados también detectores. El emisor convierte la energía eléctrica en luz, y la luz recibida es transformada nuevamente en energía eléctrica por el receptor.
En el emisor suelen utilizarse haces de luz roja, visible, o infrarroja, invisible, siendo la fuente de luz un diodo láser o un transistor emisor de luz. La señal lumínica se propaga por el medio hasta que alcanza algún objeto, logrando su detección con gran precisión.

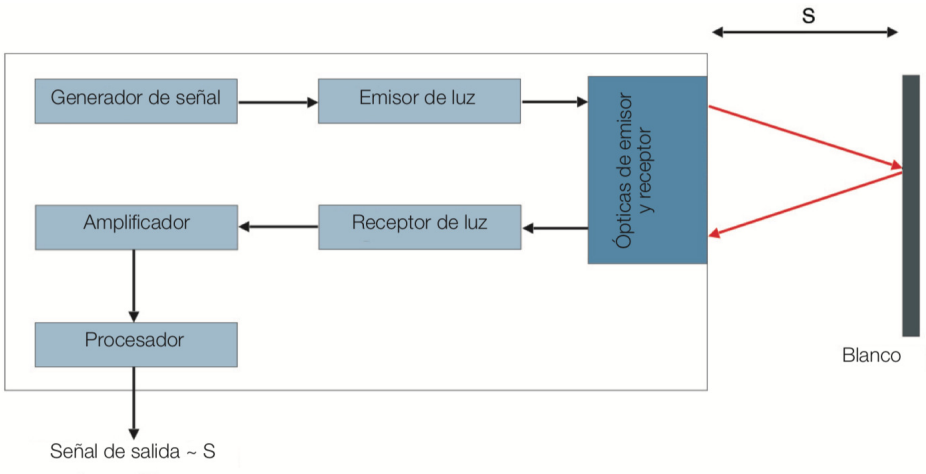


Figura 2.20: Elementos esenciales para el funcionamiento de un sensor óptico[53]

Cuando el haz de luz se encuentra con un objeto, el sensor detecta la presencia del mismo, conmutando y activando salidas de tipo transistor de colector abierto PNP, NPN o bipolares, o usando salidas de relé electromecánico o de estado sólido.
Los fotodiodos son utilizados muchas veces como receptores de luz en los sensores binarios. Cuando la luz impacta en el diodo, una corriente inversa provee una señal fotoeléctrica. La corriente se genera debido a cambios en la capa de barrera por la generación de pares electrones-agujeros.
Estos receptores cuentan con un detector sensible de posición que es un diodo de efecto lateral con una extendida superficie sensible a la luz. El rayo de luz reflejada en esta superficie genera una corriente total I, la cual es dividida en dos corrientes parciales I_1 e I_2 como muestra la figura 2.21. El cociente de esta corriente parcial es determinado por la ubicación X del punto de incidencia.
Asumiendo que la superficie activa tiene una longitud total L, entonces:

$$\frac{I_1 - I_2}{I_1 + I_2} = \frac{L - 2X}{L} \tag{2.36}$$

Podemos determinar la posición del punto de incidencia a través de la medición parcial de la corriente. El cociente de la ecuación 2.36 significa que el resultado no depende de la intensidad de la luz, ni tampoco de la reflexión de la superficie del objeto. En cambio la distribución espacial determina esa relación. El efecto lateral del diodo reacciona en el punto donde incide la luz y puede ser usada como un elemento de sensado para objetos por técnica de triangulación.

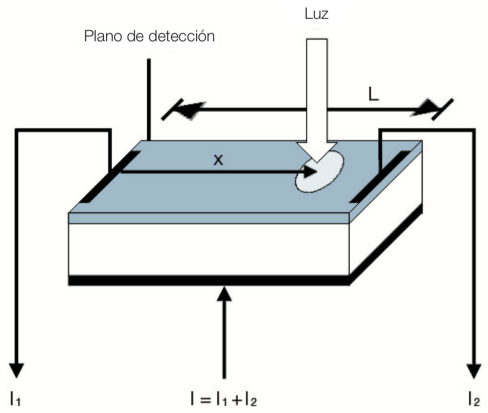


Figura 2.21: Estructura esquemática de un elemento detector sensible de posición[53]

La técnica de medición de tiempo de vuelo en cambio, consiste en enviar el haz de luz y medir el tiempo que demora en reflejarse y ser captado por el receptor del dispositivo. Debido a que la velocidad de la luz es conocida y constante, midiendo el tiempo que el haz tarda en regresar se puede calcular la distancia a la que se encuentra el objeto de interés. Esta técnica es aplicada para la medición con sensores ultrasónicos y se describe con mayor detalle en la sección 2.5.2.3.
En cuanto a los tipos de montaje, se clasifican en barrera, reflex y reflexión directa, tal como se puede ver en la figura 2.22.

- Barrera: se colocan dos módulos, un emisor y un receptor, alineados uno en frente del otro, a una distancia máxima de 200m.
- Reflexión: el emisor y receptor se encuentran en el mismo compartimiento. El haz de luz se refleja sobre un elemento reflector que no es el objeto en sí mismo. Detecta una distancia máxima de 10m.
- Reflexión directa: el emisor y receptor se ubican en el mismo módulo y detectan cualquier objeto próximo al mismo. La reflexión se produce directamente sobre el objeto que se está detectando.[53]

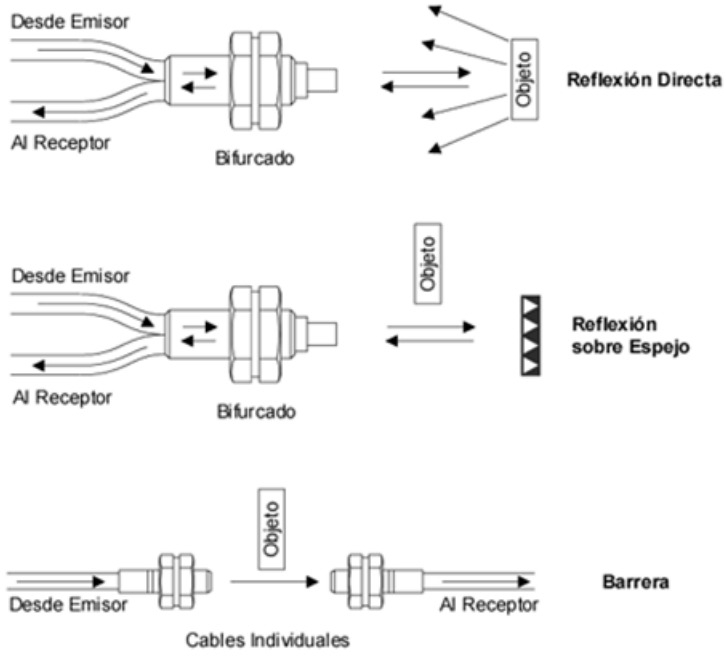


Figura 2.22: Montajes de detectores ópticos[54]

2.5.2.3. Sensores de ultrasonido

Este tipo de sensores utiliza la onda de ultrasonido, que es energía sónica que está por encima del espectro audible, entre los 20 y 60 KHz. Este tipo de ondas requiere un medio físico para que pueda producirse su propagación. Pueden ser enfocadas, reflejadas y refractadas en diversas sustancias, y su energía se atenúa cuando atraviesa un medio.

La reflexión de la onda es debida a la diferencia de impedancias acústicas entre el medio y el objeto. El tiempo de espera entre el envío de la onda ultrasónica hasta su recepción se denomina tiempo de eco, y es utilizado para determinar la distancia al objeto.

Los sensores de ultrasonido utilizan un transductor ultrasónico capaz de crear la señal acústica que viajará por el espacio en busca de objetos. También cuentan con un transductor receptor que recoge la señal ultrasónica del ambiente. Este último puede ser el mismo que el transmisor como en el caso de dispositivos bidireccionales. Es decir, se tiene un receptor que emite un pulso de ultrasonido, el cual rebota con un determinado objeto, y la reflexión del pulso sonoro es detectado por un receptor de ultrasonidos, como se muestra en la figura 2.23.

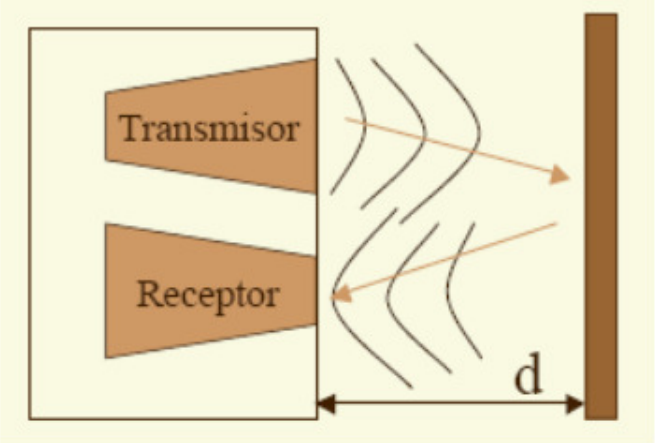


Figura 2.23: Diagramas de emisión y recepción de onda sonora[55]

La onda estacionaria que recoge el receptor es la onda resultante de todas las reflexiones presentes en el ambiente, y su forma de onda depende de la frecuencia ultrasónica (F_s) y la velocidad de propagación en el medio (c):

$$\lambda = \frac{c}{2 * F_s} \tag{2.37}$$

El divisor 2 se coloca debido a que la onda estacionaria es creada por la interacción de ondas directas y reflejadas. Si nadie ni nada se cruza delante de la onda de ultrasonido la onda estacionaria permanece constante, pero si algo se cruza se produce una nueva ruta de reflexión alterando la onda estacionaria en el reflector.

La distancia a la que se encuentra el objeto, se obtiene midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco producida por la reflexión de la onda sonora, mediante la siguiente fórmula:

$$d = \frac{1}{2} V.t \tag{2.38}$$

En donde V es la velocidad del sonido en el aire y t es el tiempo transcurrido entre la emisión y recepción del pulso.
Este sensor al no necesitar el contacto físico con el objeto ofrece la posibilidad de detectar objetos frágiles, como pintura fresca, además detecta cualquier material, independientemente del color, al mismo alcance, sin ajuste ni factor de corrección.
A pesar que posean un funcionamiento, a simple vista, muy sencillo, existen factores que influyen de una forma determinante en las medidas. Con

lo que es necesario conocer las distintas fuentes de incertidumbre, para poder tratar las medidas de forma adecuada minimizando el efecto de estas. Algunos de los factores que alteran la medición son:

- El campo de actuación del pulso que se emite desde un transductor de ultrasonido tiene forma cónica. El eco que se recibe en respuesta de la reflexión del sonido, indica la presencia del objeto más cercano que se encuentra en el cono acústico sin indicar la localización angular del mismo.
- La cantidad de energía acústica reflejada por el obstáculo depende en gran medida de la estructura de su superficie. Para obtener una reflexión altamente difusa del obstáculo, el tamaño de las irregularidades sobre la superficie reflectora debe ser comparable a la longitud de onda de la onda de ultrasonido incidente.
- En los sensores de ultrasonido de bajo costo se utiliza el mismo transductor como emisor y receptor. Tras la emisión del ultrasonido se espera un determinado tiempo a que las vibraciones en el sensor desaparezcan y esté preparado para recibir el eco producido por el obstáculo. Esto implica que existe una distancia mínima, d, proporcional al tiempo de relajación del transductor, a partir de la cual el sensor mide con precisión. Todos los objetos que estén por debajo de dicha distancia, son interpretados como que están a una distancia igual a mínima. Podemos observar este fenómeno en la figura 2.24.

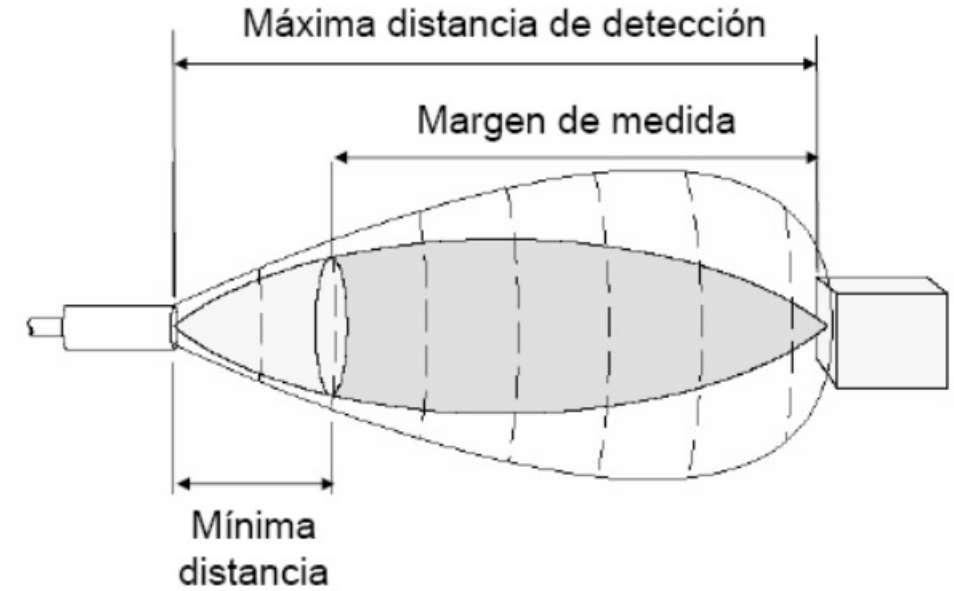


Figura 2.24: Distancia mínima en sensores de bajo costo[55]

- Los factores ambientales tienen una gran repercusión sobre las medidas: las ondas de ultrasonido se mueven por un medio material que es el aire. La temperatura afecta en la capacidad de detección, ya que la densidad del aire depende de la temperatura, con lo que influye sobre la velocidad de propagación de la onda sonora según la siguiente ecuación:

$$V_s = V_{so}\sqrt{1 + \frac{T}{273}} \tag{2.39}$$

  Siendo V_{so} la velocidad de propagación de la onda a 0°C, y T la temperatura absoluta del medio en grados Kelvin.

- Otro error importante y común son los conocidos como falsos ecos. Estos se pueden producir por diversas razones, como por ejemplo que la onda emitida por el transductor se refleje varias veces en distintas superficies antes de que vuelva a incidir en el receptor. Esta situación conocida como reflexiones múltiples, implica que el sensor evidencia algún obstáculo a una distancia proporcional al tiempo transcurrido en el viaje de la onda, es decir una distancia mucho más grande a la que en realidad se encuentra el obstáculo más cercano. Otra fuente común de falsos ecos se conoce como crosstalk, y se da cuando se emplea un cinturón de ultrasonidos donde una serie de sensores están trabajando al mismo tiempo. En este caso puede ocurrir que un sensor emita un pulso y sea recibido por otro sensor que estuviese esperando el eco del pulso que él había enviado con anterioridad (o viceversa).

- Las ondas de ultrasonido obedecen a las leyes de reflexión de las ondas, por lo que una onda de ultrasonido tiene el mismo ángulo de incidencia y reflexión respecto a la normal a la superficie. Esto implica que si la orientación relativa de la superficie reflectora con respecto al eje del sensor de ultrasonido es mayor que un cierto umbral, el sensor nunca reciba el pulso de sonido que emitió.[55]

2.5.2.4. Cálculo de proximidad a través de visión estéreo

Para medir distancia por medio de cámaras, se usa la denominada visión estéreo o estereoscópica. Esta constituye un procedimiento para la obtención de la forma de los objetos en una escena que se determina por medio de la distancia de los objetos en relación con un sistema de referencia.

En la visión estereoscópica artificial se utilizan dos o más cámaras separadas entre sí una cierta distancia relativa con las que se obtienen las correspondientes imágenes.
Estas cámaras captan dos o más imágenes de una misma escena, cada una es tomada desde una posición de las cámaras ligeramente diferente a las anteriores y debido a que las imágenes se presentan ligeramente desplazadas entre sí, se puede obtener la distancia a la que se encuentra un objeto.
Los ejes ópticos de las cámaras son mutuamente paralelos (Z_I y Z_D en la figura 2.25), encontrándose separadas una distancia que se denomina línea base (b). Las cámaras deben tener sus ejes ópticos perpendiculares a la línea base y sus líneas de exploración o epipolares paralelas a la línea base. Estas últimas líneas son líneas que unen las imágenes izquierda y derecha de un mismo punto en la escena.
El desplazamiento entre los ejes ópticos de las dos cámaras es horizontal, por lo que las imágenes de un punto determinado de la escena captado por ambas cámaras difiere solo en la componente horizontal.
En la figura 2.25 se puede observar la geometría de un par de cámaras estéreo, representadas por sus modelos puntuales con sus planos de imagen (I_I e I_D) reflejados sobre sus centros de proyección O_I y O_D, respectivamente.

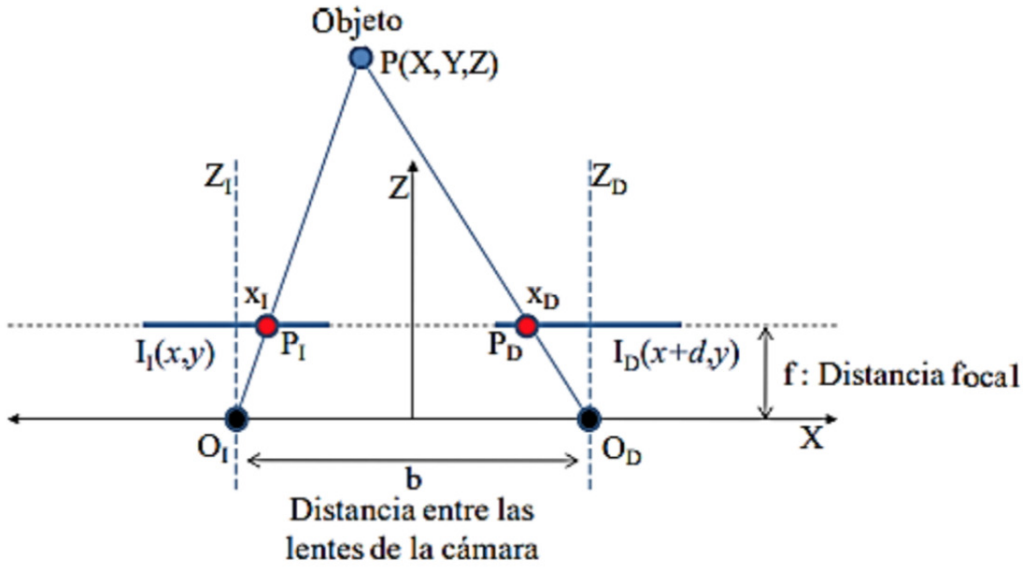


Figura 2.25: Representación de la proyección estéreo. Geometría de dos cámaras estéreo con ejes ópticos paralelos desde una perspectiva superior[57]

El origen del sistema de coordenadas de referencia se encuentra en O, siendo la longitud focal efectiva de cada cámara f, y la línea de base b. Los ejes de coordenadas del mundo X, Y, Z se sitúan entre los ejes de ambas

cámaras. Como consecuencia de la geometría de la imagen se obtiene la denominada restricción epipolar, que ayuda a limitar el espacio de búsqueda de correspondencias, de manera que en el sistema de ejes paralelos convencional todos los planos epipolares originan líneas horizontales al cortarse con los planos de las imágenes.
En un sistema con la geometría anterior se obtiene un valor de disparidad d, para cada par de puntos emparejados $P(x_I, y_I)$ y $P(x_D, y_D)$ dado por $d = x_I - x_D$.
Teniendo en cuenta la geometría del sistema, para la obtención de la distancia es necesario considerar una relación geométrica de semejanza de triángulos, en donde las coordenadas del punto de la escena $P(X, Y, Z)$ se puedan deducir fácilmente.
Observando la figura 2.25 y teniendo en cuenta la expresión de la ecuación 2.40, se puede demostrar que cuando se utiliza esta geometría, la profundidad Z, es inversamente proporcional a la disparidad de la imagen; y para una profundidad dada, cuanto mayor es b, mayor será d.

$$\left.\begin{array}{c} O_I : \frac{\frac{b}{2}+X}{Z} = \frac{x_I}{f} \\ O_D : -\frac{\frac{b}{2}-X}{Z} = \frac{x_D}{f} \end{array}\right\} \Rightarrow \left.\begin{array}{c} x_I = \frac{f}{Z}(X + \frac{b}{2}) \\ x_D = \frac{f}{Z}(X - \frac{b}{2}) \end{array}\right\} \Rightarrow d = x_I - x_D = \frac{fb}{Z} \Rightarrow Z = \frac{fb}{d} \tag{2.40}$$

Existen dos métodos para establecer la correspondencia a partir de dos imágenes estereoscópicas, una es basada en el área y la otra basada en las características.
Las técnicas basadas en características restringen la búsqueda a un conjunto disperso de características. Emplean propiedades simbólicas y numéricas de las características, obtenidas por descriptores. Éstos se encargan de procesar y extraer las características de una imagen.
Los métodos basados en área, comparan ventanas de la imagen de dimensión fija, y el criterio de semejanza es una medida de la correspondencia entre las ventanas de las dos imágenes. El elemento correspondiente queda determinado por la ventana que maximiza el criterio de semejanza dentro de la región de búsqueda. [57]

2.5.3. Odometría

El odómetro es una rueda que al girar sobre una superficie, convierte el número de revoluciones obtenidas en distancia, pudiendo ser leído directamente sobre un contador o una pantalla digital. Es decir es un instrumento que se utiliza para medir la distancia recorrida entre dos puntos.

Es un instrumento fácil de usar y que genera un resultado rápido, pero es limitado con lo que se suele utilizar para el chequeo de distancias realizadas por otros métodos.
Una forma de clasificar los odómetros es en odómetros mecánicos y digitales.

2.5.3.1. Odómetros Mecánicos

Los odómetros mecánicos están formados por una serie de engranajes que se encuentran conectados entre sí y logran una reducción de hasta 2690:1 por medio de engranajes en espiral o 'de gusano' y engranajes normales.
Constan de una rueda que al girar se conecta en un extremo con un engranaje en espiral, el cual impulsa a otro engranaje. Este, a su vez, mueve otros espirales de gusano los cuales se conectan con un engranaje normal, y así sucesivamente hasta que el último engranaje en espiral mueve el número en la rueda que se puede ver en la parte frontal del dispositivo. Esa rueda es otro engranaje, que está conectado con ruedas de los números que representan unidades, decenas, centenas, etc. de la distancia recorrida.
Para que el mecanismo funcione de manera adecuada, el sistema debe estar calibrado cuidadosamente en su fabricación.

2.5.3.2. Odómetros Digitales

Los odómetros digitales están formados por un captador eléctrico, que suele ser un imán que cada vuelta de la rueda acciona un contacto magnético (contacto 'Reed') o un semiconductor de efecto 'Hall', los cuales envían un impulso eléctrico a través de un cable, al contador electrónico. Este dato puede ser enviado a una unidad de procesamiento, como lo puede ser un microcontrolador, para poder ser procesada.
Si en un disco se colocan n de imanes concéntricos con los campos intercalados, el sensor de efecto hall se accionará cada vez que se detecte un cambio en el campo magnético y enviará una señal eléctrica a la unidad de procesamiento. Conociendo la cantidad n de imanes y el diámetro del disco se puede calcular la distancia recorrida de un pulso a otro. La cantidad de imanes que posea el disco junto con el diámetro del mismo darán la precisión del odómetro. A diferencia de los mecánicos, este tipo de instrumentos necesitan alimentación eléctrica permanente, pues la marcación desaparece al quedar sin alimentación.
Los automóviles que utilizan odómetros digitales emplean un sistema simi-

lar, una rueda metálica dentada instalada a la salida de la transmisión y un sensor magnético, que recibe un impulso cada vez que pasa uno de los dientes metálicos de la rueda. Con esto es posible determinar la distancia que recorrió el vehículo, una vez que se haya calibrado el dispositivo (conociendo el diámetro de la rueda y la cantidad de dientes metálicos de la misma).
Otra técnica utilizada, pero de menor precisión, es la implementación de un sensor infrarrojo para el cómputo de distancia recorrida. Se tendrá un diodo emisor de luz infrarroja enfrentado a un fototransistor, el cual tiene conectado a su base un cristal fotosensible que regula la corriente de colector y hará las veces de receptor. La técnica consiste en colocar un disco con pequeños orificios concéntricos a la altura del emisor y receptor, de está forma cada vez que un orificio quedé enfrentado a ellos la luz llegará hasta el receptor y se enviará un pulso a la unidad de procesamiento. La precisión de este sensor dependerá de la cantidad de orificios y diámetro del disco. No es un método muy utilizado debido a su pérdida de rendimiento al exponer el sensor a la luz natural. [58]

Capítulo 3

Marco Metodológico

3.1. Sistema global

En este documento se describe el desarrollo teórico y práctico de un vehículo autónomo nivel 4 a escala con un sistema de locomoción Ackerman. Para lograrlo, podemos dividir las tareas generales que debe realizar el vehículo según la figura 3.1.

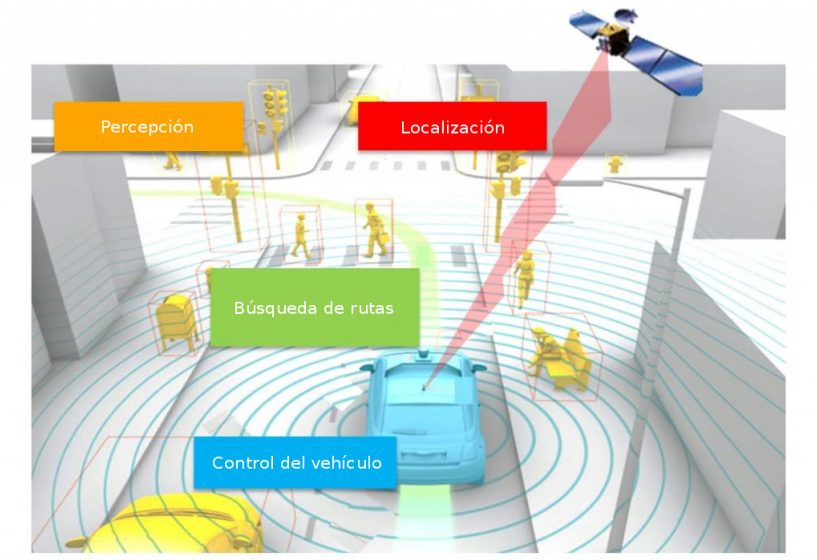


Figura 3.1: Funcionalidades generales del proyecto

- Localización: Esta funcionalidad comprende la ubicación espacial absoluta del vehículo a través de una representación del entorno con obstáculos estáticos fijos. Esto puede darse a través de localización GPS, obteniéndose un mapa estimado del entorno, los caminos transitables y la ubicación absoluta del vehículo en un instante dado. Cabe aclarar que cuando se habla de 'caminos transitables' no se refiere a la trayectoria hasta el objetivo sino los espacios donde no existen obstáculos para desplazarse, por ejemplo, las calles que componen una

ciudad.
El problema de la localización GPS es que tiene un error de hasta diez metros, complicándose su utilización en proyectos que son una representación a escala de un vehículo comercial y que se desplaza por espacios más pequeños y no por una ciudad realmente. Ante este problema, la localización se realizará a través de un mapa específico para la aplicación representado en una imagen con un punto de inicio, un punto de final y los obstáculos estáticos que existen en el entorno.

- Búsqueda de rutas: A través de la localización se obtiene la posición inicial, los obstáculos estáticos del entorno por el cual se va a desplazar el vehículo y la posición final o deseada que será alcanzada. Con estos datos iniciales y a través de algoritmos de inteligencia artificial[2.3] se busca obtener la trayectoria más óptima para desplazarse desde el punto inicial hasta el punto final evitando colisiones con los obstáculos descritos en la localización. Se utiliza una variación del algoritmo A star optimizada con respecto a las dimensiones del vehículo y su método de locomoción Ackerman.

- Control del vehículo: Una vez obtenida la trayectoria a realizar por el vehículo deben controlarse todas las variables relacionadas al movimiento para que el desplazamiento sea preciso. Las variables de movimiento obtenidas a través de diferentes sensores son: velocidades lineales y angulares, distancia recorrida, aceleración, inclinación del vehículo en el plano XY y rotación sobre cada eje (x,y,z). En concordancia con estos datos y la trayectoria que se desea realizar se deben tomar decisiones para controlar los actuadores del vehículo, motores de traslación y rotación para llegar al objetivo final del camino de manera correcta. Además de alcanzar el objetivo, en la tarea de control del vehículo debe mantenerse una velocidad constante (velocidad crucero), configurable por el usuario, a través de un algoritmo de control PID.

- Percepción: Cuando se está llevando a cabo la ruta hacia la posición objetivo, pueden aparecer obstáculos dinámicos no contemplados por el mapa inicial que no permitan realizar la trayectoria planificada o que produzcan una colisión del vehículo, por este motivo, el vehículo debe contar con sensores que le den una idea del entorno para poder tomar decisiones en base a estos datos.
  La percepción en este proyecto se implementa a través de tres sensores

de proximidad: dos ubicados al frente y uno atrás del vehículo. Esta disposición permite detectar objetos de tamaño considerable al frente o atrás del vehículo, cuando ocurre esto, el mapa original es actualizado, se realiza una búsqueda de rutas nuevamente con el punto inicial como estado actual del vehículo y se tiene en cuenta el nuevo obstáculo detectado por el sistema de percepción.

El sistema de control de todas estas tareas se dividió en dos, una unidad de procesamiento de alto nivel y otra de bajo nivel. La división del sistema en dos unidades de procesamiento se realizó porque, como se puede observar, las tareas pueden dividirse fácilmente en dos tipos: una que necesita una gran capacidad de procesamiento para aplicar algoritmos complejos, una interacción gráfica con el usuario, gran disponibilidad de memoria y otro tipo que no requiere tanta capacidad de procesamiento pero si la disponibilidad de periféricos y puertos que permitan la comunicación con sensores, el control de actuadores y con capacidad de operación en tiempo real. En el primer tipo de tareas que son la localización y la búsqueda de rutas es óptimo utilizar un microprocesador brinde la posibilidad de acceder a través de wifi desde cualquier computadora que el usuario disponga. En el segundo tipo de tareas que son el control del vehículo y la percepción, un microcontrolador tiene las características ideales para llevar a cabo el trabajo necesario.

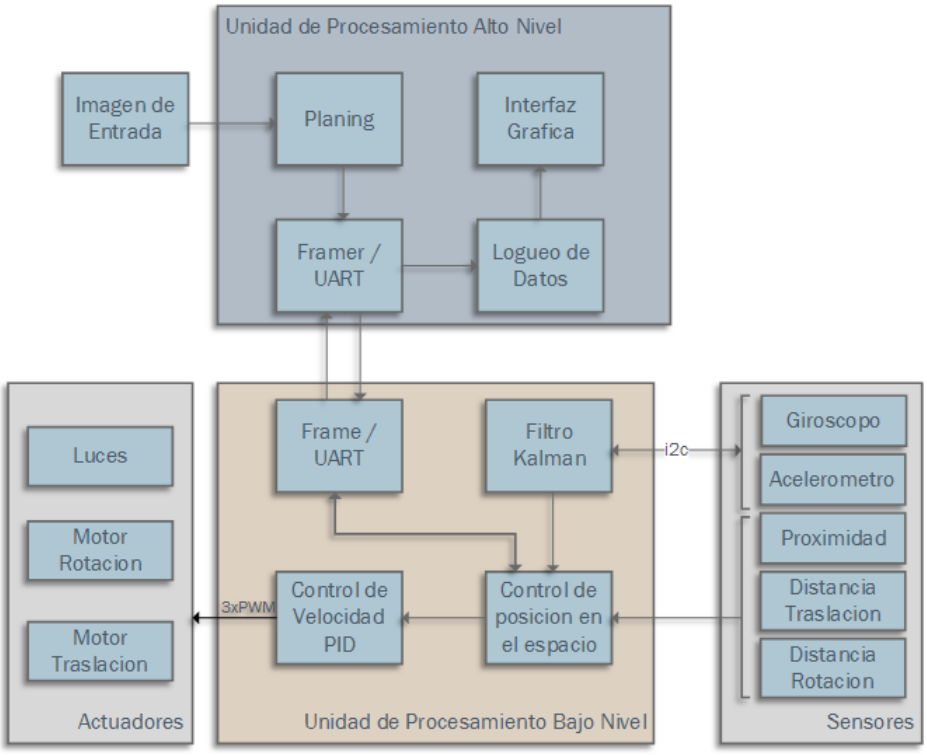


Figura 3.2: Diagrama general del sistema

- Unidad de procesamiento de alto nivel: Es la encargada de realizar procesamiento de imágenes para tomar los datos necesarios del mapa, trazar a través de algoritmos de inteligencia artificial una trayectoria óptima, guardar los datos de la navegación en una base de datos, informar al usuario a través de una interfaz gráfica en tiempo real la evolución de las variables más importantes del vehículo y realizar una comunicación full duplex con una trama de datos específica para el diseño con la unidad de procesamiento de bajo nivel por la cual enviar órdenes de movimiento, configuraciones y recibir señales de estados del vehículo.
- Unidad de procesamiento de bajo nivel: Debe tomar datos, establecer comunicaciones y realizar el procesamiento de la información de todos los sensores a través de diferentes tipos de filtros. Además, debe tomar decisiones para controlar los actuadores del sistema en base a las órdenes recibidas por la unidad de procesamiento de alto nivel y la información del entorno obtenida.

3.2. Unidad de Procesamiento alto nivel

3.2.1. Elección del microprocesador

De acuerdo a las necesidades planteadas en 3.1 se investigaron una serie de microprocesadores de diferentes fabricantes que puedan cumplirlas. Debido a la necesidad de una buena relación entre capacidad de procesamiento y consumo energético dado que la fuente de energía será provista por un banco de baterías se optó por un microprocesador ARM del tipo RISC, conocidos por su buen desempeño en el desarrollo de sistemas embebidos.
En cuanto a la plataforma que contiene el microprocesador y que brinda facilidades de conexión, además de agregar una considerable cantidad de periféricos a utilizar en el proyecto se hizo el análisis de dos opciones que por sus especificaciones de procesamiento, precio y disponibilidad en el país cumplían los estándares requeridos. Las mismas son Raspberry Pi y Cubieboard.

3.2.1.1. Cubieboard

Cubieboard es una motherboard, creada bajo el concepto de SOC, creada en Shenzhen, Guangdong, China. La primeras versiones fueron vendidas

internacionalmente en Septiembre de 2012, y la primera versión se comenzó a vender en Octubre de 2012. Puede ejecutar Android 4 ICS, Ubuntu 12.04 , Archlinux ARM o un servidor básico Debian a través de la distribución Cubian.
Especificaciones de Hardware:

1. ARM Cortex A8, ARM Mali400 MP1 cumple con OpenGL ES 2.0
2. 1GB DDR3 @480MHz
3. 4GB NAND flash interna, más de 32GB en slot SD, mas de 2T en disco 2.5 SATA
4. Ethernet 1x 10/100, soporte wifi usb
5. Entrada 5VDC 2A,2x USB 2.0 HOST, 1x mini USB 2.0 OTG, 1x micro sd
6. 1x HDMI 1080P salida de display
7. Interfaz extendida de 96 pins, incluyendo I2C, SPI, RGB/LVDS, CSI/TS, FM-IN, ADC, CVBS, VGA, SPDIF-OUT, R-TP, y mas

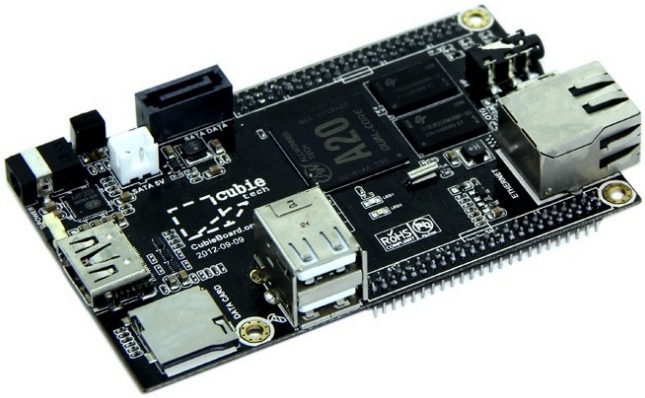


Figura 3.3: Plataforma Cubieboard[31]

3.2.1.2. Raspberry Pi 3

Raspberry Pi es un ordenador de placa reducida (SBC) de bajo costo desarrollado en el Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.
El diseño incluye un System-on-a-chip Broadcom BCM2835, que contiene

un procesador central (CPU)ARM1176JZF-S a 700 MHz (el firmware incluye unos modos "Turbo" para que el usuario pueda hacerle overclock de hasta 1 GHz sin perder la garantía), un procesador gráfico (GPU) VideoCore IV, y 512 MB de memoria RAM. El diseño no incluye un disco duro o una unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación o carcasa.
La fundación da soporte para las descargas de las distribuciones para arquitectura ARM, Raspbian (derivada de Debian), RISC OS 5, Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora); y promueve principalmente el aprendizaje del lenguaje de programación Python, y otros lenguajes como Tiny BASIC, C y Perl.
Raspberry Pi usa mayoritariamente sistemas operativos basados en el núcleo Linux. Raspbian, una distribución derivada de Debian que está optimizada para el hardware de Raspberry Pi, se lanzó durante julio de 2012 y es la distribución elegida para el proyecto.

Figura 3.4: Plataforma Raspberry Pi 3[32]

La placa Raspberry Pi puede comunicarse con dispositivos externos mediante el conector GPIO incorporado. En dicho conector se integran pines de alimentación (+5V y +3.3V), masa, y entradas/salidas capaces de implementar diferentes protocolos como UART, SPI y i^2c.
Dentro del modelo Raspberry Pi 3 existen dos versiones (A y B) con diferentes especificaciones técnicas de hardware representadas en la siguiente tabla:

	Modelo A	Modelo B
SOC	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB)	
CPU	ARM1176JZF-S a 700 MHz (familia ARM11)	
GPU	Broadcom VideoCore IV, , OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia),1080p30 H.264/MPEG-4 AVC	
Memoria (SDRAM)	256 MB (compartidos con la GPU)	512 MB (compartidos con la GPU)
Puertos USB 2.0	2	4
Entradas de vídeo	Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la RPF	
Salidas de vídeo	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4),Interfaz DSI para panel LCD	
Salidas de audio	Conector de 3.5 mm, HDMI	
Almacenamiento integrado	SD / MMC / ranura para SDIO	
Conectividad de red	Ninguna	10/100 Ethernet (RJ-45) via hub USB5
Periféricos de bajo nivel	8 x GPIO, SPI, I^2C, UART	

Haciendo un análisis de las dos plataformas se decidió que la plataforma Raspberry Pi resultaba más útil al proyecto, los puntos más importantes que influyeron en esta determinación fueron:

- Documentación y soporte: La documentación de la plataforma Raspberry Pi, tanto provista por el fabricante como la información que se dispone en internet, es mucho más amplia que la que ofrece cubieboard.
- Precio y disponibilidad: El precio de la plataforma elegida es casi dos veces menor y se puede conseguir fácilmente en nuestro país.
- Compatibilidad y sistema operativo: El sistema operativo utilizado por la plataforma Raspberry Pi es una variación de debian que tiene compatibilidad con una gran cantidad de software.

3.2.2. Sistema general de la Unidad

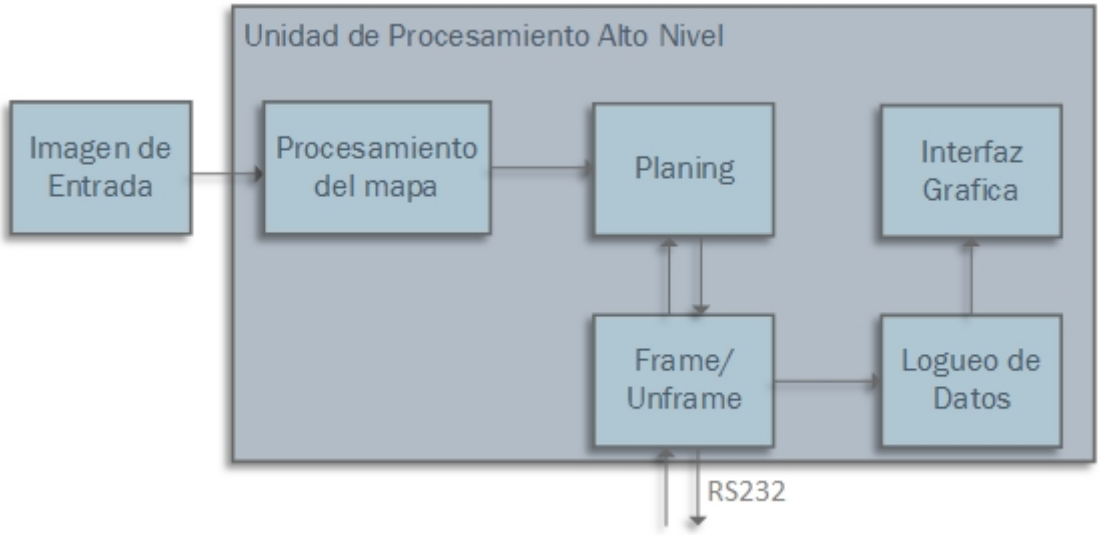


Figura 3.5: Resumen sistema general

En la figura 3.5 se pueden observar las tareas más importantes que debe realizar la unidad de procesamiento de alto nivel.

- Procesamiento del Mapa: Teniendo como entrada el mapa estático del entorno en el que se va a desplazar el vehículo se filtra el mismo y se obtiene la información de los espacios transitables, el punto de inicio y el punto de final.
- Planning o trazado de trayectoria: A través de una modificación de un algoritmo A star de búsqueda heurística informada en grafos se encuentra una ruta que conecta el punto de inicio y de final de manera óptima evitando colisiones. Se generan vectores de movimientos para poder ser aplicado posteriormente un algoritmo de cross track error.
- Comunicación UART/Trama de datos: Se envían y reciben datos a través de una comunicación RS232 serie. Para que la comunicación sea más robusta es diseñada una trama de datos que permite evitar posibles interferencias y determinar si existen errores en los datos recibidos. Los comandos de movimientos son enviados a la unidad de procesamiento de bajo nivel luego de ser encapsulados en la trama y a su vez, en paralelo, se deben recibir datos de las variables de estado del vehículo.
- Logueo de datos: La información de los sensores del vehículo, la trayectoria realizada, el tiempo de ejecución de la ruta y otras variables

de interés son almacenadas en una base de datos para su posterior procesamiento. Estos datos serán de utilidad para realizar mejoras en el diseño y tener información más detallada y precisa del desempeño del vehículo.

- Interfaz gráfica de monitoreo: Se pueden observar el estado del vehículo a través de sus variables más importantes en un gráfico 3D desarrollado en OpenGL.

Todas estas tareas se desarrollan con códigos propios realizados particularmente para el proyecto en cuestión en lenguaje de programación Python en un sistema operativo Debian sobre la plataforma Raspberry Pi.
Para que el sistema funcione correctamente algunas de estar tareas deben ejecutarse simultáneamente. Por ejemplo, el diseño debe soportar enviar y recibir datos al mismo tiempo para diferentes tareas como pueden ser el gráfico y logueo de datos y los comandos de movimientos del vehículo realizando la trayectoria hasta su destino final.
Para soportar este tipo de comportamiento 'paralelo' se utiliza multiprocesamiento en Python a través de hilos de ejecución. Los hilos permiten ejecutar múltiples operaciones de forma concurrente en el mismo espacio de proceso. El proceso principal es el encargado de realizar el procesamiento del mapa y la búsqueda de rutas y a su vez existen tres hilos encargados de: enviar datos, recibir datos y graficar en 3D el estado del vehículo.

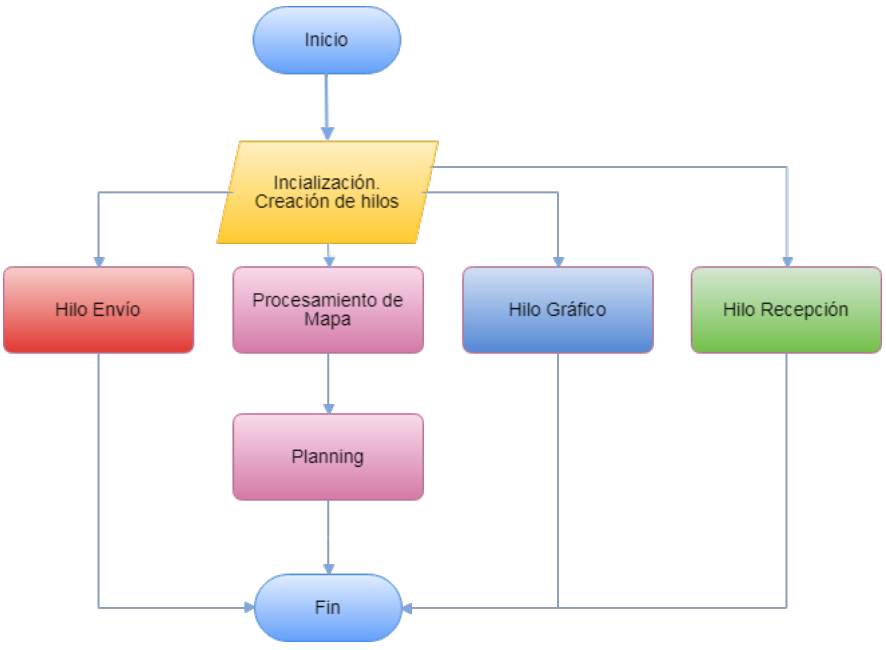


Figura 3.6: Esquema de hilos

Si bien los hilos se ejecutan independientes deben compartir información, por ejemplo, el hilo gráfico debe enviar datos a través del hilo de envío de información requiriendo información de la unidad de procesamiento de bajo nivel como pueden ser datos del acelerómetro y luego al recibirlos por medio del hilo de recepción debe procesarlos y actualizar el gráfico en 3D. Para lograr este tipo de comportamiento se utiliza un planificador (scheduler) que se encarga de registrar todas las clases con un identificador y un método asociado a ellas.
Cada clase del sistema hereda los métodos necesarios para registrarse en el scheduler, asociar un método a su identificador y también el acceso a una FIFO que se encargará de transmitir los datos a través del hilo de envío.
Tanto al enviar como al recibir datos se utilizan los índices del registro de la clase dentro del planificador en la trama de datos. Al recibir datos, se discrimina la clase a la que va dirigida la información y a través del planificador se ejecuta el método asociado a la misma.

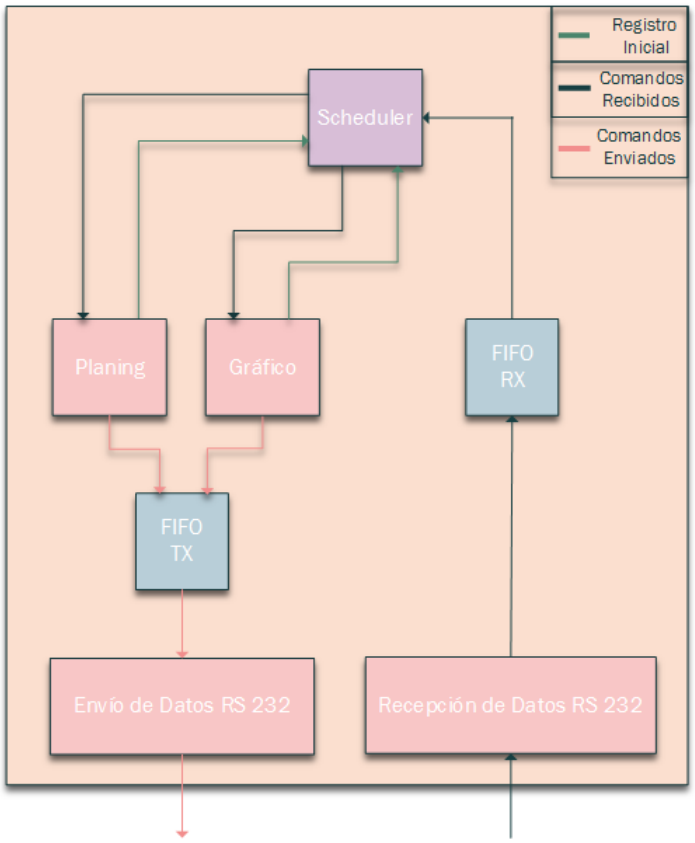


Figura 3.7: Relación entre hilos

3.2.3. Sistema de procesamiento de Mapas

La entrada a la unidad de procesamiento de alto nivel es un mapa, como primer paso debe ser procesado para determinar los espacios transitables,

los obstáculos, el punto de inicio y el punto objetivo o final del vehículo.

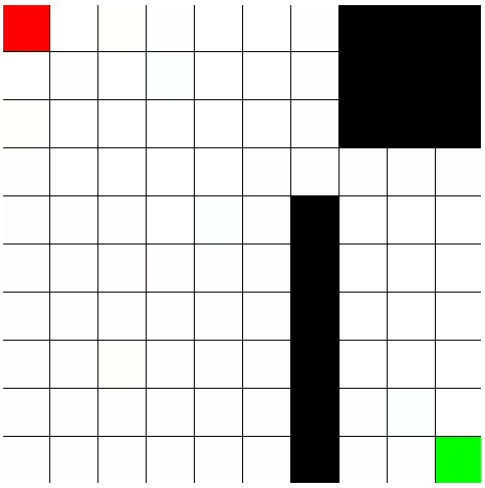


Figura 3.8: Mapa de ejemplo

La imagen de entrada es del tipo RGB donde los píxeles blancos representan espacios transitables, los píxeles negros representan espacios donde hay obstáculos, un píxel verde representa el punto de inicio del vehículo y el punto rojo el punto objetivo del mismo.
En este paso el resultado esperado es una matriz binaria de dos dimensiones MxN, siendo M el largo y N el ancho del espacio en el que se va a desplazar el vehículo. En esta matriz se busca que sean representados cada punto (x, y) con un cero si el punto en cuestión es transitable o con un uno si existe un obstáculo en él, además de obtenerse los puntos (x_i, y_i) y (x_f, y_f) correspondientes al punto de inicio y punto final, respectivamente. Para lograrlo, se utiliza el módulo misc de la librería scipy que permite transformar una imagen en una matriz de MxNx3 dimensiones donde cada píxel es representado por la intensidad de luz de sus tres canales de colores primarios.
El análisis de cada píxel consiste en evaluar la intensidad de cada canal para determinar el color del mismo. Por ejemplo, si deseamos detectar el color verde en un píxel particular se esperaría que la componente en el canal G de la imagen sea alto y que las componentes en el canal R y B sean bajas. Los valores de cada canal dependiendo el color se puede observar en la siguiente tabla:

	Intensidad R	Intensidad G	Intensidad B
Blanco	Alta	Alta	Alta
Negro	Baja	Baja	Baja
Verde	Baja	Alta	Baja
Rojo	Alta	Baja	Baja

La implementación en Python de este procesamiento se ilustra en el siguiente código:

```
# Lectura de la imagen
self.m_imagen = misc.imread(imagen)
self.m_plan = np.zeros((len(self.m_imagen),len(self.m_imagen[0])))
# Se recorre cada pixel evaluando los valores de cada canal RGB
# para determinar si es blanco,negro,rojo o verde respectivamente
for i in range(len(self.m_imagen)):
  for j in range(len(self.m_imagen[0])):
    if (self.m_imagen[i][j][0] > RGB_THR) and \
       (self.m_imagen[i][j][1] > RGB_THR) and \
       (self.m_imagen[i][j][2] > RGB_THR):
       self.m_plan[i][j] = 0
    elif (self.m_imagen[i][j][0] < RGB_THR) and  \
         (self.m_imagen[i][j][1] < RGB_THR) and  \
         (self.m_imagen[i][j][2] < RGB_THR):
       self.m_plan[i][j] = 1
    elif (self.m_imagen[i][j][0] > RGB_THR) and \
         (self.m_imagen[i][j][1] < RGB_THR) and \
         (self.m_imagen[i][j][2] < RGB_THR):
      self.goal = [i,j]
    elif (self.m_imagen[i][j][0] < RGB_THR) and \
         (self.m_imagen[i][j][1] > RGB_THR) and \
         (self.m_imagen[i][j][2] < RGB_THR):
       self.init = [i,j]
```

De acuerdo a la combinación de intensidades en cada canal RGB podemos transformar la matriz de MxNx3 en la matriz objetivo MxN y los puntos de interés.

3.2.4. Sistema de trazado de trayectoria

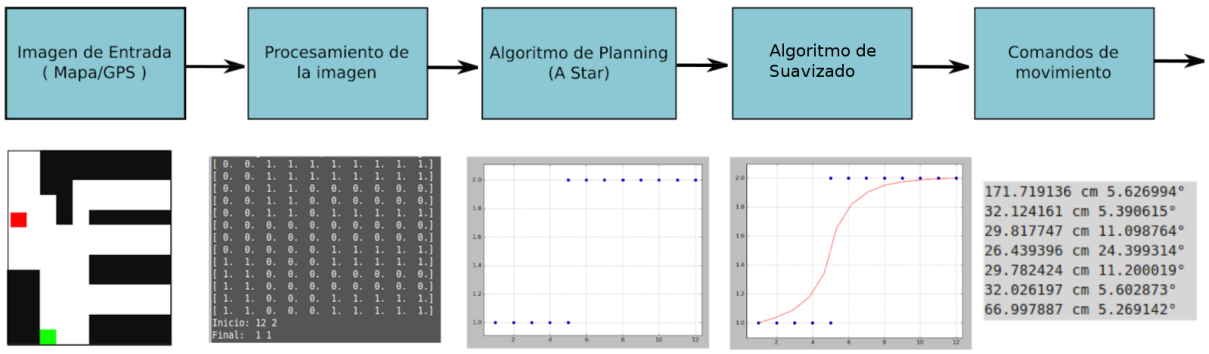


Figura 3.9: Flujo de trazado de trayectoria

Luego que la imagen del mapa de entrada es procesada y es obtenida una matriz MxN, siendo M y N las dimensiones del entorno donde se va a desplazar el vehículo, se tiene información sobre los espacios transitables y los espacios con obstáculos. Además de este dato son obtenidos el punto de inicio y el punto objetivo que pertenecen a la matriz.
Con esta información podemos considerar cada uno de los elementos de la matriz como nodos o vértices y aristas a las conexiones entre cada nodo y sus elementos contiguos, ya que el vehículo puede movilizarse sin problemas a cada uno de ellos.
El objetivo de esta etapa es lograr un conjunto de instrucciones de movimientos que sean enviadas a la unidad de procesamiento de bajo nivel para que sean ejecutadas a través de un algoritmo de cross track error.
Para lograrlo se utiliza una variación del algoritmo A star desarrollado en 2.3 que permite obtener una trayectoria óptima en términos de distancia, maniobras y por ende, energía para el desplazamiento del vehículo desde el punto de inicio hasta el punto final evitando colisiones.
Se realizó una clase que tiene todas los métodos y variables relacionadas al planeamiento de rutas óptimas a través de la aplicación del algoritmo A star y posterior suavizado de la trayectoria por medio de mínimos cuadrados. Esta clase toma como entrada una imagen y provee comandos de movimientos en formato módulo y fase para facilitar la aplicación de un algoritmo del tipo cross track error por la unidad de procesamiento de bajo nivel.
El primer paso es establecer la función heurística que utilizaremos en el algoritmo como estimador. Cabe recordar que según lo desarrollado en 2.3.7.3 la función heurística debe ser admisible. La heurística elegida como

función fue la distancia euclidiana entre cada elemento de la matriz y el origen del movimiento definida en un dominio $\Re^2$ como:

$$d_e(x, y) = \sqrt{(x_n - x_i)^2 + (y_n - y_i)^2} \tag{3.1}$$

Esta heurística puede verse de manera gráfica en la imagen 2.7. La implementación de la función A star que determina los puntos que debe recorrer el vehículo para llegar al destino de manera óptima se representa en la figura 3.10.

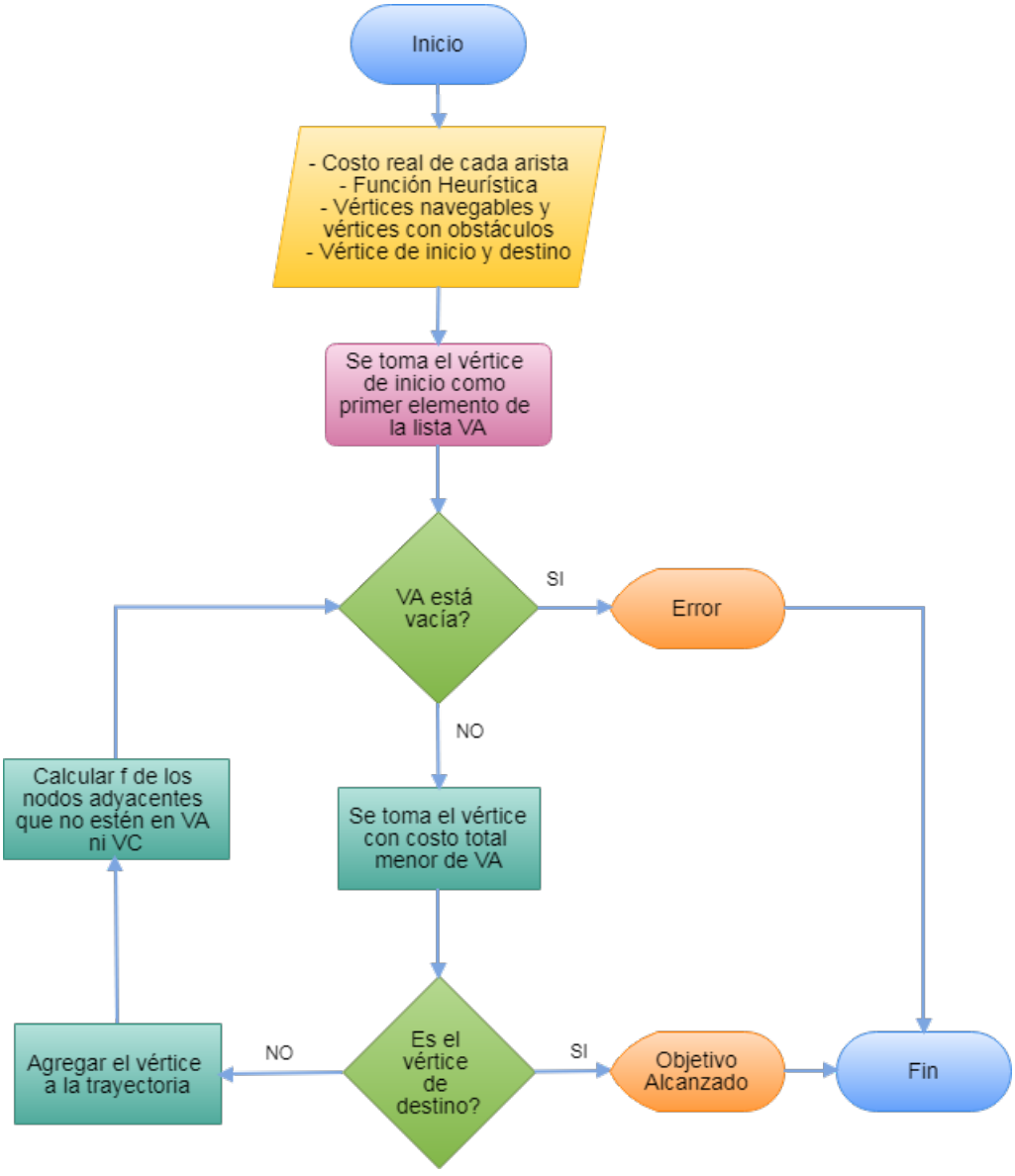


Figura 3.10: Diagrama de flujo A Star

Luego, se obtiene la secuencia de movimientos a través de la diferencia entre puntos consecutivos en la trayectoria estableciendo la longitud del

movimiento definida como paso de píxel dependiendo la escala del mapa y el ángulo de inclinación que debe describir el vehículo para alcanzar el próximo punto. En esta etapa los movimientos pueden adquirir ángulos de inclinación de $\pm k * \frac{\pi}{4}$ siendo k = 0,1,2...N.
Debido a a la diseño del sistema de locomoción del vehículo es necesario, para que el desplazamiento del mismo sea más fluido, suavizar esa trayectoria intentando que los ángulos de inclinación sean menores. Esto se logra a través del algoritmo de suavizado desarrollado en 2.3.7.4. La implementación práctica de la función de suavizado toma como argumentos las ganancias de las restricciones, una tolerancia de corte de las iteraciones y la trayectoria original del algoritmo A star.
La implementación en cuestión puede verse en el siguiente código en lenguaje Python:

```
def smooth(self, weight_data = 0.5, weight_smooth = 0.25, \
                                    tolerance = 0.000001):
  # Copia de self.path
  newpath = [[0 for row in range(len(self.path[0]))] \
                for col in range(len(self.path))]
  for i in range(len(self.path)):
    for j in range(len(self.path[0])):
      newpath[i][j] = self.path[i][j]
      # Minimizar (xi-yi)^2 (yi yi+1)^2
      # Condicion de corte = tolerance
      change = tolerance
      while change >= tolerance:
        change = 0.0
        for i in range(1,len(self.path)-1): # Primer y ultimo punto
                                            #                 impolutos
          for j in range(len(self.path[0])):
            aux = newpath[i][j]
            newpath[i][j] += weight_data * (self.path[i][j] \
                                            - newpath[i][j])
            newpath[i][j] += weight_smooth * ( newpath[i-1][j] \
                                               + newpath[i+1][j] \
                                               - (2.0 * newpath[i][j]))
            change += abs(aux - newpath[i][j])
      self.spath = newpath
```

En esta etapa se tiene una trayectoria, cuyos puntos no están limitados a números enteros dentro de la matriz debido al algoritmo de suavizado, en el plano (x, y) de movimiento del vehículo. La conexión entre estos puntos son los movimientos que deberá realizar el robot para alcanzar el destino o punto final.
Estos movimientos pueden representarse como vectores con módulo y fase en el plano (x,y) de desplazamiento tomando como el sistema cartesiano de referencia el estado inicial del vehículo. El hecho de enviar vectores con módulo y fase facilita la aplicación de algoritmos de cross track error en

una unidad de bajo nivel.
Para calcular el módulo del vector utilizamos la fórmula de distancia euclídea entre dos puntos en $\Re^2$ desarrollada en 3.1 y la fase del vector, a través de la siguiente fórmula:

$$\Theta_i = \arctan(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}) \tag{3.2}$$

```
while (i < (len(path_soft)-1)):
  vector = np.array( [path_soft[i+1][0] - path_soft[i][0], \
                      path_soft[i+1][1] - path_soft[i][1] ] )
   vector_module = vector_module + np.sqrt(np.dot(vector, vector))
   vector_phase  = vector_phase + np.arctan(vector[1]/vector[0])
   if ( (abs(vector_phase) > PHASE_LIMIT)  or \
            vector_module > MODULE_LIMIT  ):
      datos = [chr(107)] + [chr(int(vector_module))] \
            + [chr(VELOCIDAD)] + [chr(1)] + \
              [chr(abs(int(vector_phase*RAD_TO_DEG)))]
      self.send(datos)
```

Luego de haberse obtenido el comando de movimiento se envía el mismo a la unidad de procesamiento de bajo nivel además de información sobre la velocidad a la que se desea realizar la traslación. Si la fase o el módulo a enviar son valores muy pequeños son acumulados para enviarse en la siguiente iteración del algoritmo para evitar movimientos cortos que atentarían contra la fluidez del desplazamiento del vehículo.

3.2.5. Sistema de detección de objetos dinámicos del entorno

El sistema desarrollado hasta este punto tiene la capacidad de trazar un recorrido a través de un mapa inicial para poder alcanzar un punto en el plano requerido por el usuario. El problema evidente es que no tiene noción de lo que sucede en el entorno por el cual se está desplazando. Por ejemplo, si existe algún objeto en movimiento o si se agrega un elemento que no estaba en el mapa estático inicial, el vehículo no lo va a considerar y ,por ende, no lo va a eludir provocándose una colisión.
A través de sensores de proximidad, si un objeto obstruye el camino del vehículo, puede ser detectado y actuar en consecuencia. Esto se logra a través de la comunicación entre la unidad de bajo nivel, que procesa la señales de los sensores de proximidad, y la unidad de alto nivel, que trazará una nueva trayectoria debido a un nuevo obstáculo en el plano de movimiento. En el siguiente diagrama de flujo puede observarse la metodología a seguir si se detecta un obstáculo no previsto con anterioridad:

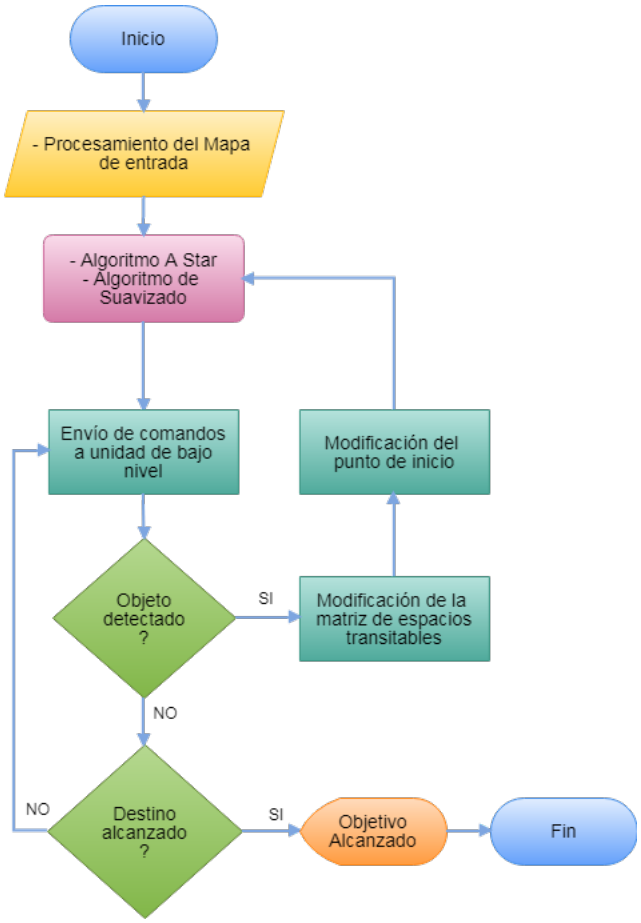


Figura 3.11: Algoritmo de detección

Cuando la unidad de procesamiento de bajo nivel no puede efectuar un comando de movimiento recibido por la unidad de procesamiento de alto nivel debido a la presencia de un objeto no existente en el trazado de la trayectoria, el vehículo vuelve a la posición que se encontraba antes de efectuar el último comando de movimiento y envía una trama de alarma a la unidad de procesamiento de alto nivel con la fase y la distancia que tenía el vehículo al momento de interrumpir su movimiento y el sensor que detectó el objeto en cuestión.
En base a los datos obtenidos, la unidad de alto nivel agrega los obstáculos en la matriz que representa el mapa según las dimensiones del vehículo y establece el punto de inicio como la posición en la que se encuentra actualmente el robot. Luego, se ejecuta el algoritmo A star con estos nuevos datos para trazar la nueva trayectoria.

3.2.6. Sistema de monitoreo de datos

El sistema de monitoreo de datos consiste en la adquisición de datos de las variables dinámicas que caracterizan el vehículo para establecer el desempeño real del funcionamiento del mismo a través de esta información. Debido a que la unidad de procesamiento de bajo nivel tiene acceso a los sensores, la unidad de procesamiento de alto nivel debe requerir estos datos a través de la comunicación entre las dos para que se le envíe una trama de datos con las variables de interés.
Estas variables son:

- Velocidad angular de cada eje
- Velocidad traslacional o lineal del vehículo
- Inclinación del vehículo
- Distancia de detección de cada sensor de proximidad
- Distancia total recorrida por el vehículo

Estas variables pueden ser utilizadas para visualizar la evolución de las mismas en tiempo real del vehículo o para ser almacenadas y luego, al finalizar la trayectoria, utilizarlas para determinar el desempeño del vehículo y realizar optimizaciones.
En el diseño existe un hilo cuyo método asociado se encarga de requerir datos de los sensores a través de la unidad de procesamiento de bajo nivel y graficar la evolución de las variables en tiempo real del vehículo. Esta clase realiza una representación 3D de la plataforma y grafica la inclinación en tiempo real mientras realiza el recorrido. Este gráfico 3D se realiza en python mediante la librería de diseño OpenGL y Pygame. La representación en cuestión se muestra en las figuras a continuación.

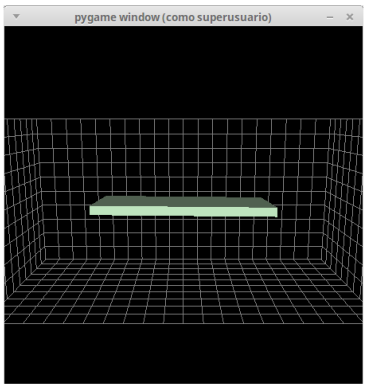


Figura 3.12: Gráfico 3D OpenGL

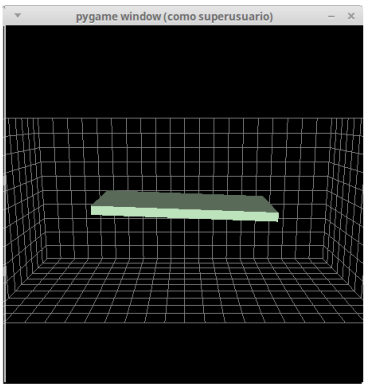


Figura 3.13: Gráfico 3D OpenGL

Teniendo este hilo desarrollado en el diseño, se puede añadir fácilmente los gráficos en tiempo real que se desee dependiendo de la información que necesite el usuario.
A su vez, existe otra clase que se encarga de escribir archivos con todas las variables mencionadas anteriormente que se guardarán en el disco rígido de la unidad de procesamiento de alto nivel con la intención de poder realizar un post-procesamiento de la información, siendo de gran utilidad para obtener los resultados empíricos de la trayectoria y además para realizar optimizaciones en el diseño.

3.2.7. Simulador

Para realizar pruebas del sistema en general y de los algoritmos en particular sin la necesidad de tener todo el hardware funcionando se desarrolló un simulador que modela la unidad de procesamiento de bajo nivel tanto sea la lectura de sus sensores como el sistema de comunicación asociada a la transmisión de estos datos. Al finalizar todos los comandos de movimiento se realiza un gráfico con la ruta original y las rutas trazadas al haber encontrado un obstáculo dinámico. A su vez, son enviados datos de todos los otros sensores lo que permite comprobar el funcionamiento de la parte gráfica y el almacenamiento de datos.

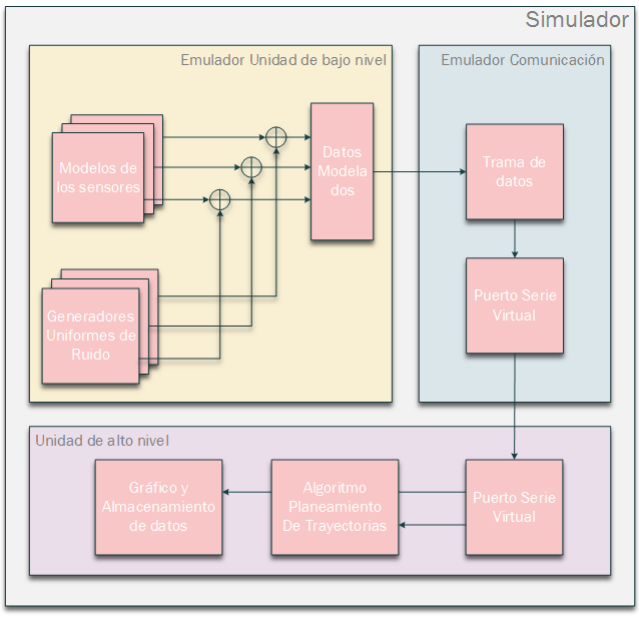


Figura 3.14: Esquema general del simulador

Para lograrlo se realizaron dos clases:

- Emulador Unidad de bajo nivel: La misma tiene métodos que al ser requeridos datos de los sensores genera información real a través de modelos afectados por generadores de ruidos uniformes. Esto hace que, por ejemplo, el sistema detecte en momentos aleatorios un objeto en la trayectoria que está virtualmente recorriendo, forzándolo a actuar en consecuencia.
- Emulador comunicación: Se encarga de armar la trama de datos con la información de los sensores, crear un puerto virtual serie de manera de emular de manera real la parte de la comunicación serial entre los dispositivos

En las siguientes imágenes podemos ver los resultados de dos casos diferentes del simulador en los cuales se tienen distintos mapas. En momentos aleatorios del movimiento se detectan objetos no previstos para la trayectoria y se traza una ruta alternativa representada en la líneas de diferentes colores.

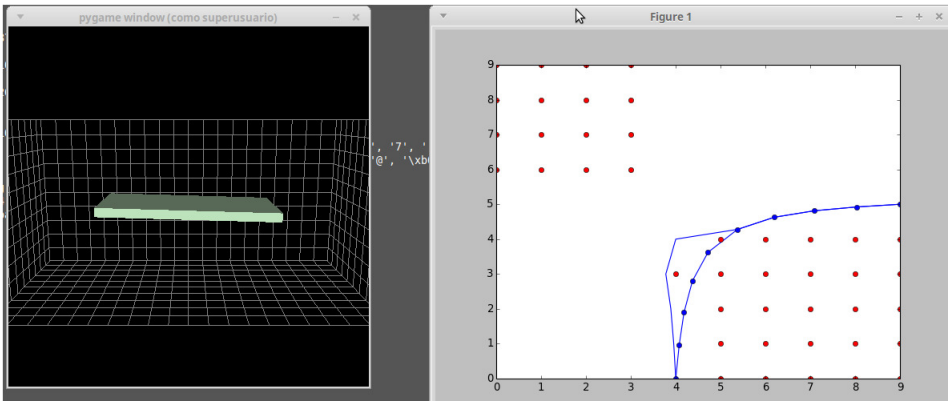


Figura 3.15: Simulador caso A

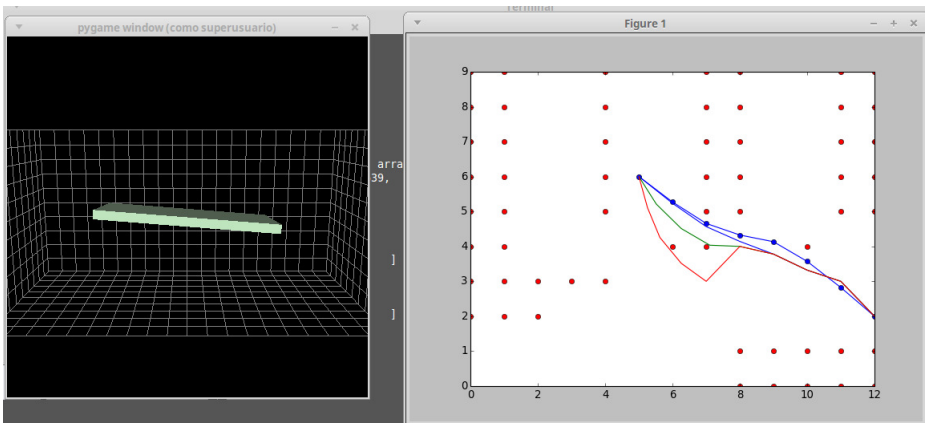


Figura 3.16: Simulador caso B

Se simuló el procesamiento de mapas y el trazado de trayectorias en una situación real para demostrar la escalabilidad de los algoritmos de alto nivel. Se tiene un mapa que proviene de GMaps y se desea ir desde el punto definido por las calles Ayacucho y Pedro Vella hasta el punto final definido por la Fundación Tarpuy.

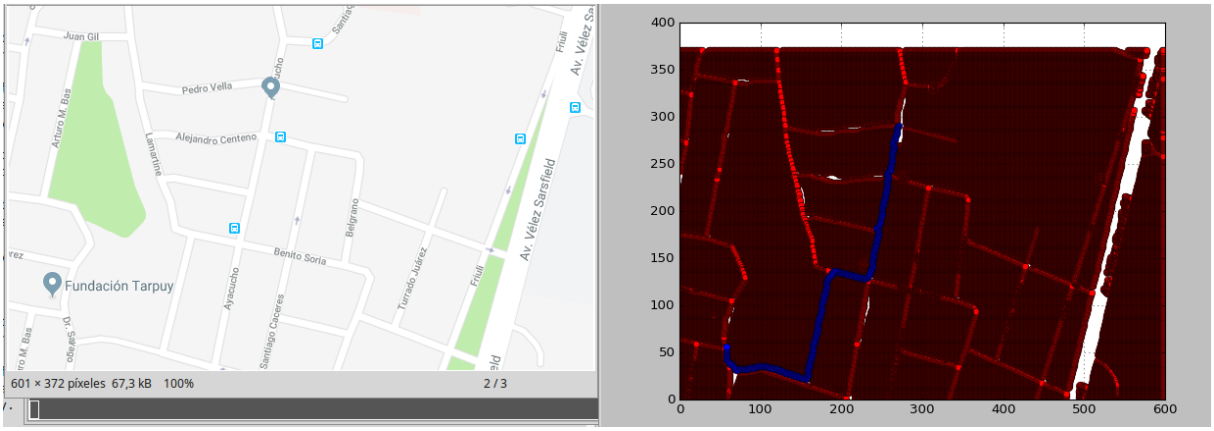


Figura 3.17: Simulación trayectoria Real

3.3. Unidad de Procesamiento bajo nivel

3.3.1. Elección del microcontrolador

De acuerdo a lo desarrollado en 3.1 la unidad de bajo nivel estará encargada de controlar los actuadores para la ejecución de los movimientos, calcular la compensación PID, compensar la orientación del vehículo, tomar los datos de los sensores, comunicarse a través de una comunicación full-duplex con el microprocesador. La ejecución de estas acciones requieren de diversas características dentro del microcontrolador, el mismo debe contar con varias interrupciones de timer para el manejo de diferentes sensores, interrupciones externas, comunicación i2c, comunicación serie, contar con pines tanto digitales como analógicos, la posibilidad de generación de señales PWM. Los principales microcontroladores desarrollados en A.1.3 cubren estas necesidades. Otro requerimiento es la velocidad de procesamiento, necesaria para el tiempo de procesamiento de sensores tales como los de proximidad que necesitan dar una respuesta rápida para evitar colisiones. Esta velocidad también es requerida para poder atender en forma eficiente todas las tareas que debe realizar el microcontrolador a la hora de encargarse de todo el control de bajo nivel. Cómo se expone en la sección anexa A.1, los microcontroladores de ATMEL poseen una gran cantidad de registros lo que disminuye la dependencia del resto de la memoria, mejora

la velocidad y disminuye las necesidades de almacenamiento de datos. Casi todas las instrucciones se ejecutan en 1 ó 2 ciclos de reloj contra 5 a 10 ciclos de reloj para los chips PIC de Microchip, lo que los hace relativamente rápidos dentro de los microcontroladores de 8 bits. Además, la arquitectura AVR es más fácil de programar a nivel de lenguaje ensamblador y más fácil de optimizar con un compilador.
Por su parte los microcontroladores PIC, ofrecen una mayor libertad a la hora de diseñar nuestros algoritmos y sistemas de control y se pueden adquirir por un precio inferior. Aplican muy bien a la hora de hacer desarrollos a nivel industrial debido a lo expuesto anteriormente.

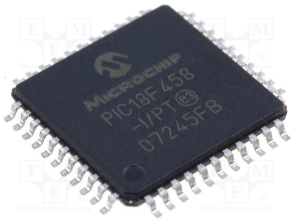


Figura 3.18: PIC 18f458[24]

A la mayor velocidad de procesamiento con la que cuentan los chips de Atmel, se le debe sumar que existe una gran cantidad de periféricos en el mercado, de fácil y rápida adquisición, compatibles con su arquitectura. Debido a estas razones, y que la diferencia de costos no influía significativamente en la aplicación de este trabajo, se decidió utilizar un microcontrolador Atmel para el desarrollo.
Una vez seleccionado el microcontrolador se procedió a la selección de la plataforma que lo contiene. La más popular y que posee una comunidad de desarrollo más grande dentro del marco de los microcontroladores Atmel es la plataforma Arduino. La cual además de una gran comunidad que brinda soporte a la hora del desarrollo, brinda facilidades de conexión, permite conectar gran cantidad de periféricos, posee un costo accesible y se adquiere fácilmente en el mercado local. Estas son las razones por las que se eligió esta plataforma. Dentro de la familia Arduino además se cuenta con varias versiones, donde las principales son: Arduino Uno, Mega, Ethernet, Due, Leonardo, Leonardo ETH, Micro, Mini, Nano y Yun. Dentro de esta familia, y teniendo en cuenta que el procesamiento más pesado se realiza en la parte de alto nivel, los que más se ajustaban a las necesidades del proyecto eran Arduino Uno y Mega que cuentan con una gama intermedia en cuanto a sus capacidades.

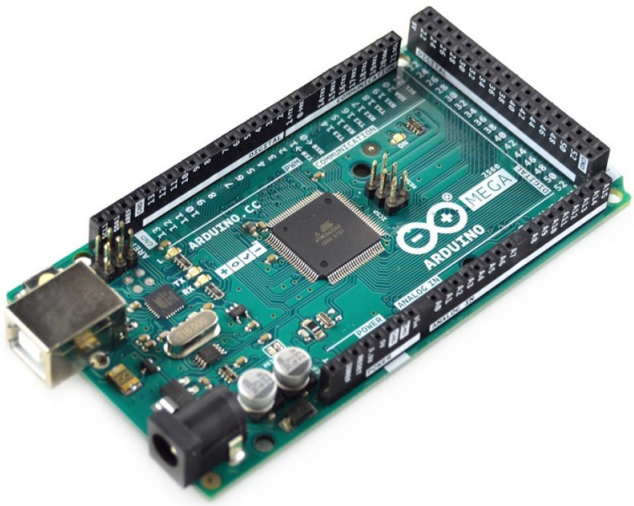


Figura 3.19: Arduino Mega 2560[25]

En la tabla 3.1 puede verse la comparación entre las dos plataformas anteriormente mencionadas:

Características	Arduino Uno	Arduino Mega 2560
Microcontrolador (MCU)	ATmega328P	ATmega2560
Voltaje de operación del MCU	5V	5V
Voltaje de alimentación típico de la placa	7V - 12V	7V - 12V
Pines I/O Digitales	14(incluidos PWM)	54(incluidos PWM)
Salidas PWM	6	15
Pines de entradas analógicas	6	16
Máxima corriente CD por I/O pin	20mA	20mA
Máxima corriente CD para pines de 3.3V	50mA	50mA
Memoria Flash del MCU	32KB	256KB
SRAM del MCU	2KB	8KB
EEPROM del MCU	1KB	4KB
Velocidad del Reloj	16MHz	16MHz
Interrupciones por Timer	3	6
Interrupciones Externas	2	6

Cuadro 3.1: Comparación Arduino Uno - Arduino Mega

Como se puede observar en la tabla anterior el Arduino Mega ofrece una posibilidad mayor de conexión con periféricos, ya que brinda un mayor número de interrupciones y de pines. A esto se le suma mayores capacidades de memoria y mayor velocidad de procesamiento debido a la misma. La

plataforma Mega es más utilizada a la hora de aplicaciones que requieren procesamiento más complejo y una conexión un un mayor número de periféricos. Es por esto que fue la elegida para el desarrollo del proyecto.

3.3.2. Sistema general

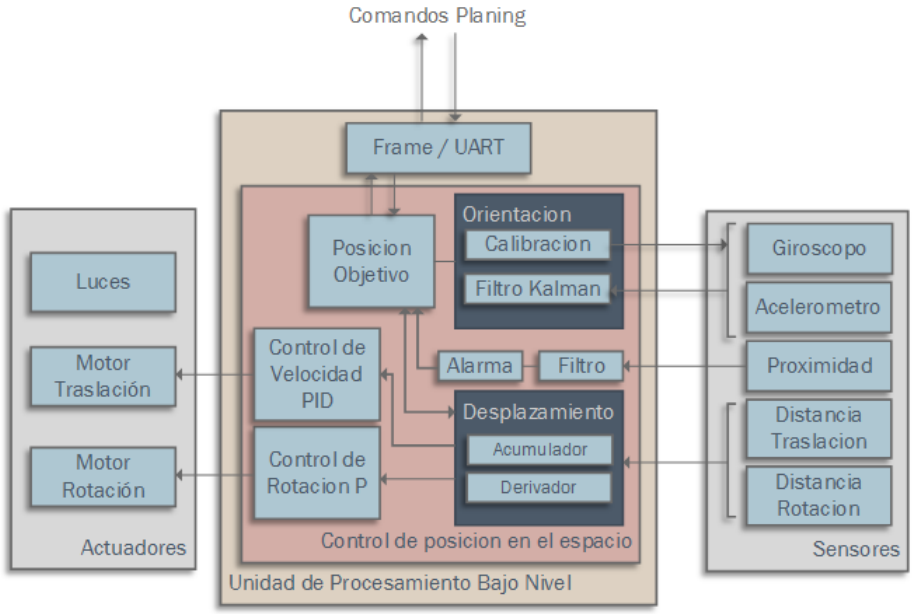


Figura 3.20: Unidad de procesamiento de bajo nivel

En la figura 3.20 se pueden observar los bloques principales que abarca la unidad de procesamiento de bajo nivel.

- Comunicación UART/Trama de datos: se envían y reciben datos a través de una comunicación RS232 serie. Para que la comunicación sea más robusta se trabaja con una trama de datos que permite evitar posibles interferencias y filtrar datos erróneos en la recepción. Se reciben comandos de movimiento por parte de la unidad de alto nivel y a su vez, se le envían datos de las variables de estado del vehículo. Para enviar los datos se utilizan funciones de armado de tramas, y para recibirlos funciones de procesamiento de trama y extracción de la información.

- Control de posición en el espacio: una vez procesada la trama, se obtienen los comandos de movimiento que el vehículo debe realizar, estos comandos incluyen la distancia que se debe avanzar, el grado absoluto final que se debe adquirir y la velocidad del movimiento. Se toman datos de los sensores que informan sobre la orientación y

el desplazamiento del vehículo y permiten conocer como accionar los actuadores para que el vehículo desempeñe el movimiento contenido en el comando. Además se toma el dato de proximidad de objetos para poder informarle a la unidad de alto nivel ante objetos no contenidos en el mapa inicial.

- Actuadores: los actuadores consisten en dos motores, la bocina del vehículo y las luces. Tanto las bocinas como las luces se accionarán ante la detección de un nuevo objeto en el espacio de movimiento. Por su parte los motores, se encargarán uno de la rotación del volante del vehículo y el otro de la traslación del mismo. Estos motores son accionados desde el microcontrolador a través de un control proporcional y uno proporcional-derivativo-integral respectivamente.

- Sensores: los sensores proveen al microcontrolador los datos necesarios para conocer la posición relativa y absoluta del vehículo. Por un lado se utiliza un sensor MPU-6050 de 6 ejes que brinda información acerca de la velocidad angular y de la fuerza gravitacional sobre cada eje y permite conocer los ángulos que el vehículo rota y su inclinación. El dato de las distancias recorridas y ángulos rotados por el volante se obtiene de odómetros digitales y los datos de proximidad de tres sensores de ultrasonido distribuidos a lo largo del vehículo.

Todas estas tareas se desarrollan con códigos propios realizados particularmente para este proyecto en el lenguaje C++ y compilados sobre la plataforma Arduino Mega 2560. El código desarrollado en el microcontrolador puede observarse en C
Estás funciones son implementadas en un script principal encargado de administrar todas las acciones, y en cuatro librerías propias encargadas de realizar las funcionalidades específicas. Las únicas librerías internas de Arduino utilizadas son las que controlan los Timer 1 y Timer 3.
El código implementado a bajo nivel se vale de las interrupciones por Timer para poder simular una ejecución paralela. De esta forma, mientras en el script principal ejecuta un ciclo infinito que contiene todos los estados posibles del proceso, las interrupciones se encargan de recibir y enviar datos, de tomar los valores de los sensores, y de activar las banderas que marcan en que estado esta la ejecución. Las librerías desarrolladas en forma propia son:

- Comunicación Uart: se encarga de enviar y recibir datos por medio del puerto serie.

- Interrupciones: se encarga de ejecutar las acciones correspondientes a las interrupciones de Timer 1, Timer 3 e interrupciones externas.
- MPU6050: se encarga de calibrar el sensor correspondiente, y de obtener los datos de inclinación y grados rotados sobre el eje z.
- Movimientos: se encarga de accionar los actuadores, implementar el controlador PID y filtrar los datos de proximidad obtenidos del sensor de ultrasonido.

3.3.3. Estructura y Actuadores

La implementación de todo el sistema se hizo sobre un vehículo eléctrico a escala de 105cm de largo, 60cm de ancho y 55cm de largo que contaba con placas de control todo/nada para el motor de traslación y el de rotación del volante. De esta plataforma solo se utilizó la estructura, todo el sistema mecánico y los motores eléctricos. Todas las placas electrónicas fueron extraídas para poder incorporar las de desarrollo propio, en las secciones subsiguientes se hará un recorrido por cada uno de los componentes que fueron incorporados al vehículo. La estructura cuenta con cuatro ruedas, dos ejes, las ruedas delanteras con posibilidad de rotación todo estructurado según la configuración Ackerman descrita en 2.4. En la figura 3.21 se presenta la estructura en proceso de incorporación de los diferentes componentes:

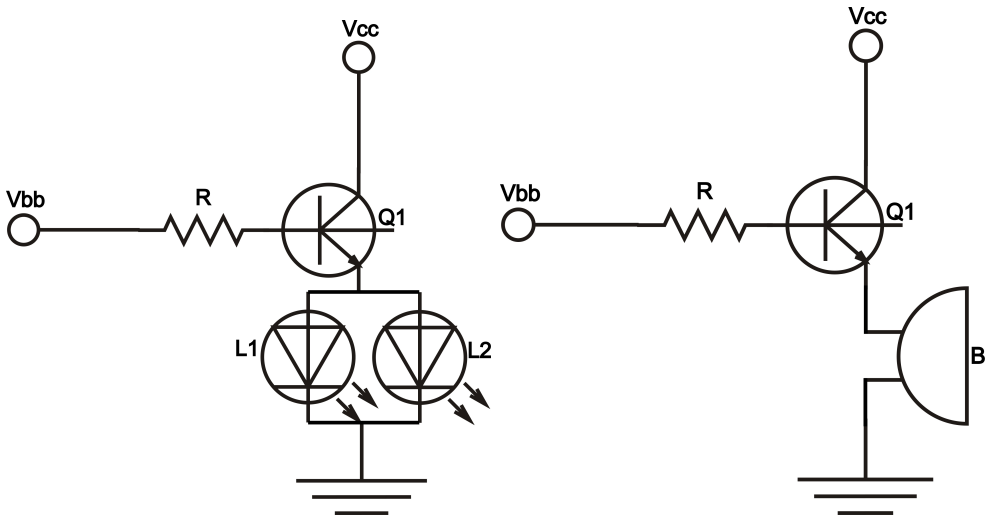
(a) Colocación placas (b) Conexionado baterías

Figura 3.21: Estructura utilizada para montar el sistema de control

Los actuadores, son aquellos que en base a las acciones de control provenientes del microcontrolador sean capaces de permitir el movimiento de la estructura o ejecutar acciones deseadas. Para este desarrollo se incorporaron todos actuadores eléctricos, los cuales se describen a continuación:

- Motores eléctricos: se utilizaron los dos motores eléctricos con los que contaba la estructura, debido a ya están incorporados a la misma de manera óptima. Se trata de dos motores de corriente continua modelo RS550-6V controlador por armadura, estos motores no poseen hoja de datos disponible por lo que se procedió a hacer el modelado de los mismos (sección 3.3.8.1). La corriente y tensión nominal de los mismos son 5A y 6V respectivamente en funcionamiento normal. Estos datos fueron extraídos de la carcasa de los mismos y midiendo la corriente consumida en régimen. Estos actuadores están dispuestos uno en el eje trasero del vehículo, el cual permite los movimientos de traslación y el otro en el eje del volante, el cual permite los movimientos de rotación. Son controlados mediante señales PWM (descritas en A.5.3) para poder regular su velocidad. Entre la salida el microcontrolador donde se genera esta señal PWM y cada motor se encuentra la etapa de potencia, dos puente H, los cuales además de amplificar la potencia de entrada a los motores permiten controlar los sentidos de giro. Estos puentes H son descritos en la sección 3.5.1.
- Bocina: incorporada originalmente a la estructura, era accionada por un botón ubicado en el volante y alimentada por dos pilas doble AA de 1.5V conectadas en paralelo. Se reemplazó el botón por un transistor bipolar NPN para abrir y cerrar el circuito en el momento deseado. Este actuador es accionado al detectarse un objeto en frente del vehículo.
- Luces: la estructura también cuenta con dos luces led en las ópticas que originalmente se accionaban siempre que el vehículo se movía hacia adelante. Estas luces fueron desconectadas del cableado original y se accionan en el mismo momento que la bocina.

El circuito utilizado como llave electrónica para controlar tanto la bocina como las luces se muestra en la figura 3.22:



(a) Circuito Luces (b) Circuito Bocina

Figura 3.22: Circuitos de apertura-cierre para bocina y luces

Para la implementación de los circuitos anteriores se utilizó un transistor bipolar NPN BC548, el cual para una tensión $V_{cc} = 3V$ y una corriente de colector $I_c = 40mA$ posee las siguientes características:

Característica	Valor Típico
V_{CESat}	0.2V
V_{BESat}	0.7V
h_{fe}	150

Cuadro 3.2: Características transistor bipolar BC548

Teniendo en cuenta los datos de la tabla anterior y conociendo que la salida en alto de un pin digital Arduino es de 5V se necesitó una resistencia $R = 5{,}6K\Omega$ para tener una corriente de $I_c = 40mA$ circulando por ambos actuadores. El cálculo se realizó resolviendo las mayas de entrada y salida de los circuitos anteriores. Cabe aclarar que en caso de las luces, al tener dos ópticas y haber sido conectado los leds en paralelo, por cada uno circulan 20mA. Los circuitos de la figura 3.22 fueron implementados en dos placas electrónicas distintas, una para cada actuador.

3.3.4. Sensores

3.3.4.1. Unidad de medidas inerciales

El requerimiento de la unidad de medidas inerciales para este proyecto en particular es que pueda obtener información sobre las velocidades angulares de cada eje cartesiano y las fuerzas gravitacionales relacionadas a

ellas. De acuerdo a esto, se buscaron sensores acelerómetro/giróscopo que pueden proveer la información necesaria.
El criterio de selección del sensor en particular se basó en el precio, la disponibilidad en el país, la facilidad de utilización y los niveles de ruido de los datos que provee.

Sensor	Costo	Disponibilidad	Precisión	Interfaz	Documentación
MPU-6050	Bajo	Alta	Media	Digital	Excelente
EY88	Alto	Media	Media	Digital	Buena
Adxl335	Bajo	Media	Alta	Analógica	Buena

Cuadro 3.3: Posibles sensores unidad de medidas inerciales

En base a la investigación se seleccionó el sensor MPU6050 ya que cumplía con todos los requerimientos, era sencillo adquirirlo en nuestro país y la documentación era muy clara para el tipo de desarrollo que se iba a realizar.

MPU-6050

El sensor MPU-6050 es un sensor digital integrado que combina acelerómetro de 3 ejes, giróscopo de 3 ejes y un procesador digital de movimiento (DMP). Utiliza el protocolo de comunicación i^2c para la comunicación de la información ya sea escritura de registros de comunicación como lectura de datos de los sensores.
Este modelo cuenta con tres conversores analógico-digital para digitalizar los datos provenientes del giróscopo y otros tres relacionados a los datos provenientes de cada eje del acelerómetro.
La versatilidad de este sensor se la da la configuración de diferentes variables de la adquisición de los datos a través de registros de configuración. Esto permite tener precisión tanto en movimientos rápidos como lentos. Puede ser configurado el rango máximo de la medición del giróscopo (± 250, ± 500, ± 1000, ± 2000 $^\circ$/seg) y del acelerómetro ($\pm 2g$, $\pm 4g$, $\pm 8g$, $\pm 16g$). Además puede configurarse el tiempo de muestreo y puede activarse un filtro pasa bajos para intentar evitar ruido de baja frecuencia.

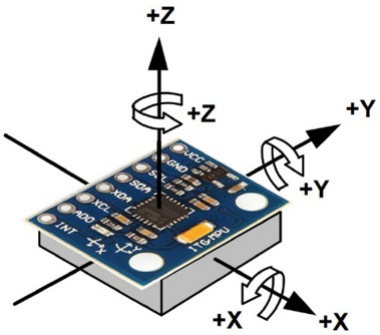


Figura 3.23: Sensor acelerómetro/giróscopo MPU6050[59]

La conexión con el microcontrolador es sencilla a través de los dos puertos (SCL y SDA) específicos del protocolo i^2c. La alimentación de 5v necesaria para el funcionamiento del sensor proviene del regulador interno de la plataforma Arduino y debido a que el consumo de corriente del sensor es baja (3.5 mA) no afecta el funcionamiento del microcontrolador.

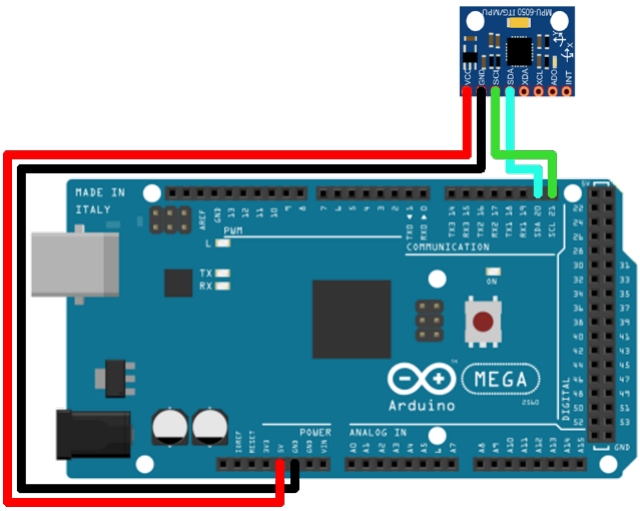


Figura 3.24: Conexión MPU6050[60]

Desarrollo de librería de comunicación y calibración

Se desarrolló una librería en C en orden de soportar la comunicación i^2c entre el microcontrolador y el sensor MPU-6050 para la lectura/escritura de los registros de este último.
La librería cumple las siguientes tareas:

- Debe tener funciones sencillas que representen modos de funcionamiento y que puedan ser llamadas para realizar la escritura de los registros de configuración.

- Debido a que, en general, la información del sensor que se desea leer se encuentra en registros de 16 bits y además deben ser procesadas según la configuración del rango máximo, es necesario tener funciones para el requerimiento de datos que al ser llamadas devuelvan la información ya procesada para ser utilizada.
- La librería debe contar con una rutina de calibración que puede llamarse al iniciarse el sistema para evitar desviaciones iniciales y que los datos del sensores sean más fidedignos.

Un ejemplo de una función de la librería que toma datos de velocidad angular sobre el eje Z es la siguiente:

```
float obtener_z_gyro(float *datos, float *offset)
{
  float z_gyro = 0;
  Wire.beginTransmission(MPU6050_ADRESS);
  Wire.write(0x47);                    // starting with register 0x3B (
   ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU6050_ADRESS, 2, true); // request a total of 14
   registers
  datos[6] = Wire.read() << 8 | Wire.read();
  if (datos[6] >= 0x8000)
    datos[6] = -((65535 - datos[6]) + 1);
  datos[6] = (datos[6] / 131) - offset[2];
  z_gyro = datos[6];
  return z_gyro;
}
```

3.3.4.2. Sensores de proximidad

Se presenta a continuación una tabla comparativa entre los posibles sensores de proximidad que se pueden utilizar para la implementación de la detección de objetos, estos sensores son los vistos en la sección 2.5.2.

Tecnología	Complejidad de implementación	Costo	Eficiencia
Sensor ultrasonido	Media	Bajo	Baja
Sensor óptico	Media	Alto	Media
Visión estéreo	Alta	Medio	Alta

Cuadro 3.4: Posibles tecnologías para detección de obstáculos

Elección del sensor

Debido a que la luz viaja por el aire a una velocidad mucho mayor a la del sonido (300.000 km/s contra 343 metros por segundo), la respuesta

de los sensores ópticos es mucho más rápida. Esta rapidez los hace mucho más eficientes y permitirá que el vehículo se desplace a una velocidad mayor y que el error debido al desplazamiento del vehículo sea menor. En cuanto a la utilización de cámaras para visión estero, supera a los otros dos sensores en eficacia debido a que no sólo se detecta que hay un objeto delante del sensor, sino que también se puede determinar en que posición se encuentra en relación al vehículo, la forma del objeto, su tamaño y de esta manera tomar acciones más óptimas en el trazado de la ruta. Los sensores de ultrasonido funcionan de manera similar a los ópticos, solo proveen la información de si hay o no un objeto en frente de ellos, pero son menos eficaces debido a lo expuesto anteriormente con respecto a la velocidad del sonido en el medio.
El procesamiento a través de cámara requiere de una implementación más compleja que escapa a los objetivos del presente trabajo. La implementación necesaria para este trabajo además, no requiere de una velocidad de respuesta muy alta, debido a que las velocidades que puede adquirir el vehículo son bajas (hasta 5 m/s). Esto sumado al bajo costo del sensor llevó a la elección de la tecnología ultrasónica para poder detectar proximidad de objetos.
Dentro de los sensores de ultrasonido se evaluó los que vienen originalmente en los automóviles actuales, pero para poder procesar el dato que devuelven los mismos hacía falta la adquisición de una placa especial para su pre-procesamiento y eso dificultaba el proceso. Por este motivo se decidió adquirir un sensor de ultrasonido convencional.
Recordando lo abordado en la sección 2.5.2.3, para el cálculo de la distancia hasta un objeto es necesario enviar un pulso a través del sensor, el pulso rebota en el objeto y vuelve hacia el sensor, la distancia será proporcional al tiempo que tarda el pulso en volver. Este envío y recepción del pulso se hace a través de pines del ultrasonido, Trig y Echo respectivamente. Comercialmente se consiguen sensores de 4 pines (alimentación, gnd, trig y echo) y de 5 pines (agregan un pin llamado out que se puede utilizar como trig y echo de manera simultánea). Debido a detalles que se especificarán en el siguiente apartado, para la presente implementación es conveniente utilizar sensores de 4 pines y manejar el Trig y Echo de manera independiente, por lo que el sensor que finalmente se adquirió fue el HC-SR04 de 4 pines.

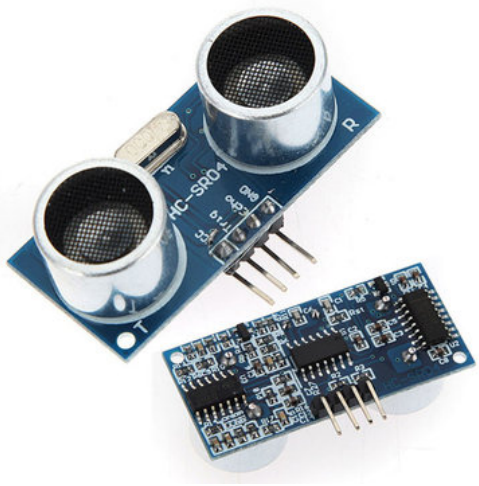


Figura 3.25: Sensor ultrasónico HC-SR04[56]

Funcionamiento

En la sección 2.5.2.3 del capítulo anterior se ha visto el funcionamiento de este tipo de sensores. Básicamente el sensor posee un transductor interno capaz de convertir la señal eléctrica en una onda sonora, la cual viaja en busca de un objeto, y al rebotar en el mismo retorna hacia el sensor y es captada por un transductor receptor, la distancia hasta el objeto será proporcional al tiempo que la onda tarda en regresar. Para que el sensor funcione, se debe enviar, un pulso a través de su pin Trig y esperar su regreso por el pin Echo. Como se mencionó anteriormente pueden utilizarse dispositivos que poseen estos dos en el mismo pin. Para su control, es necesario cambiar constantemente la configuración del pin en cuestión, de 'INPUT' a 'OUTPUT'. Debido a la cantidad de tareas que debe realizar el microcontrolador se prefirió no sobrecargarlo con esta última y utilizar pines independientes.

Figura 3.26: Colocación de los sensores en la estructura

Para mejorar la eficiencia del sistema de detección de objetos del vehículo se utilizaron tres sensores HC-SR04, dos para la parte delantera y uno para la trasera. De esta forma se puede determinar al menos, si el objeto se encuentra del lado izquierdo o derecho del vehículo y con esta información optimizar la trayectoria para esquivar el mismo. Al momento de esquivar un obstáculo el vehículo deberá volver hacia atrás para ganar espacio, en este movimiento hacia atrás en caso de aparecer un objeto no contemplado en el mapa, el mismo podrá detectarse a través del sensor que se ha incorporado en la parte trasera. Dado este caso, se considerará que el vehículo ha quedado encerrado por obstáculos y se esperará a que se libere el camino para continuar el desplazamiento.
El sensor elegido, internamente, coloca en alto el pin Echo cuando se manda el pulso por el pin Trig, y coloca en bajo el Echo cuando el pulso vuelve. De esta forma, midiendo el tiempo de ancho de pulso de pin Echo, se puede calcular la medida deseada. La velocidad del sonido, es igual a la distancia recorrida por la onda sonora, sobre el tiempo que tarda en recorrerla. Por lo tanto con el dato del tiempo y siendo la velocidad del sonido constante, $v = 343{,}2m/s$, se puede calcular la distancia al obstáculo de la siguiente manera:

$$d = \frac{343{,}2m/s * t}{2} \tag{3.3}$$

El tiempo t representa la distancia en que el pulso tarda en volver. Se divide por dos debido a que el pulso va hasta el objeto y vuelve, por lo tanto nos interesa la mitad del tiempo transcurrido para nuestro cálculo.

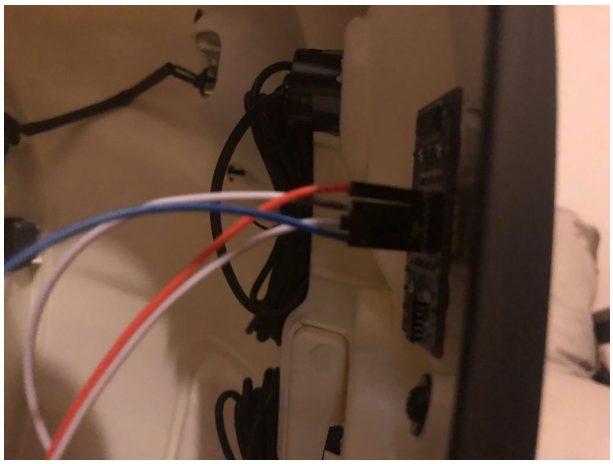


Figura 3.27: Conexionado de los sensores

Implementación en Arduino

En el caso del sistema desarrollado, se necesitan tomar datos de las distan-

cias varias veces por segundo, debido a que no se quiere colisionar contra los obstáculos y por lo tanto se desea detectarlos a tiempo. Claro está, que la velocidad máxima que el vehículo puede adquirir esta limitada por las muestras que se toman por segundo para evitar la colisión. Para poder tener una buena precisión en la medición del tiempo en que tarda en volver el pulso, se toma al mismo en micro-segundos, además se desea tener el dato de la distancia en centímetros. Teniendo en cuenta estas dos consideraciones y transformando las unidades del dato de la velocidad del sonido, la ecuación para el cálculo de la distancia en el microcontrolador queda como sigue:

$$d = \frac{t}{58{,}0} \tag{3.4}$$

El sensor elegido cuenta con 4 pines, dos correspondientes a la alimentación, el pin Trig y el pin Echo. Los pines de alimentación se conectarán a pines de 5V y GND de la placa Arduino. La hoja de datos del mismo se puede observar en B. Como se desean conectar muchos sensores, por practicidad, se diseñó una placa en donde ingresan 5V y GND provenientes del Arduino y son sacados por múltiples pines de salida, de esta forma se pueden alimentar todos los sensores del sistema desde esta placa. El pin de Trig se conecta a un pin digital del microcontrolador configurado como OUTPUT, es a través del mismo por donde se envía el pulso que sale a buscar el objeto. En el caso del pin Echo, como se mencionó en el apartado anterior cambiará su estado al volver el pulso enviado luego de rebotar contra un obstáculo, es decir que no se conoce el momento en que este pin cambiará de estado. Es por esto que generalmente, se conecta el Echo a un pin del microcontrolador con interrupción externa de manera que cada vez que el pulso vuelve salte una interrupción y se calcule la distancia. Desarrollando el sistema de bajo nivel y poniéndolo en marcha se detectaron dos problemas con esta metodología para controlar el sensor:

- El número de sensores a conectar está limitado por el número de interrupciones externas con que cuenta el microcontrolador. En este caso puntual, el Arduino Mega cuenta con 6 interrupciones de este tipo, dos de las cuales son utilizadas para los odómetros digitales, con lo cual alcanza para conectar los 3 sensores de ultrasonido, pero si a futuro se quisieran agregar más para mejorar la precisión de la detección sería una limitante.
- Si el microcontrolador realiza una gran cantidad de tareas, como es el caso, utilizar tantas interrupciones puede causar que el sistema se

tilde y esto afecte, por lo tanto, a la estabilidad del mismo.

Debido a estos dos problemas se diseñó otro método para la toma de datos con los sensores de ultrasonido. El método consiste en utilizar una interrupción por Timer, para lo cual se utilizó la interrupción por Timer 3 del microcontrolador, para ver el estado de un pin digital configurado como INPUT al que está conectado el Echo del sensor. Debido a que el ancho de los pulsos que se miden en el Echo dan el valor de la distancia a un objeto determinado, consultar el estado de este pin en un período constante de tiempo genera un error en la medición. Se seleccionó el valor del período de la interrupción por Timer 3 de manera tal que no introduzca un error significativo, y que a su vez no sea lo suficientemente rápido como para afectar el funcionamiento de las otras rutinas que se ejecutan en el microcontrolador. El período elegido fue entonces $T = 200\mu s$ el cual puede generar un error máximo de 3.44cm por medición, calculado en base a la ecuación 3.4. Con esta metodología se consulta el valor de distancia medido por cada sensor en forma secuencial. Cabe destacar que de esta manera se puede utilizar un gran número de sensores de ultrasonido utilizando una única interrupción por Timer.
En la figura 3.28 se presenta un diagrama de flujo que describe todo el proceso de medición, filtrado y generación de alarma ante un objeto detectado para avisar al microprocesador y que el mismo ejecute la acción necesaria. Como se puede observar el flujo de funcionamiento de estos sensores es a primera vista sencillo, pero se deben tener en cuenta varios aspectos. Todo comienza cuando se envía el pulso a través del pin de Trig, en ese momento a su vez el pin de Echo se pondrá en alto y se comenzará a contar el tiempo en micro-segundos. Cuando el Echo se ponga en bajo significará que el pulso ha vuelto y calculando el tiempo transcurrido se puede calcular la distancia hasta el objeto más próximo. Si la distancia medida es menor a 40cm se considera que el objeto está lo suficientemente cerca y se coloca un 1 en una cola llamada 'cola de detección', de lo contrario se coloca un 0. La cola tiene una profundidad de 3 elementos y cada vez que entra uno nuevo el más viejo sale. Esta cola será evaluada y si dos de sus tres elementos son 1 se considera que hay un objeto cercano al vehículo y se activa una alarma de objeto detectado. La cola de detección hace las veces de filtro de los sensores de ultrasonido, se tiene una cola para cada uno, en caso de haber ruido el la medición (valores atípicos) el mismo será filtrado a través de la evaluación de esta cola. A su vez, la profundidad de la cola es una relación de compromiso entre la precisión del filtro y el tiempo que se tarda en detectar un objeto.

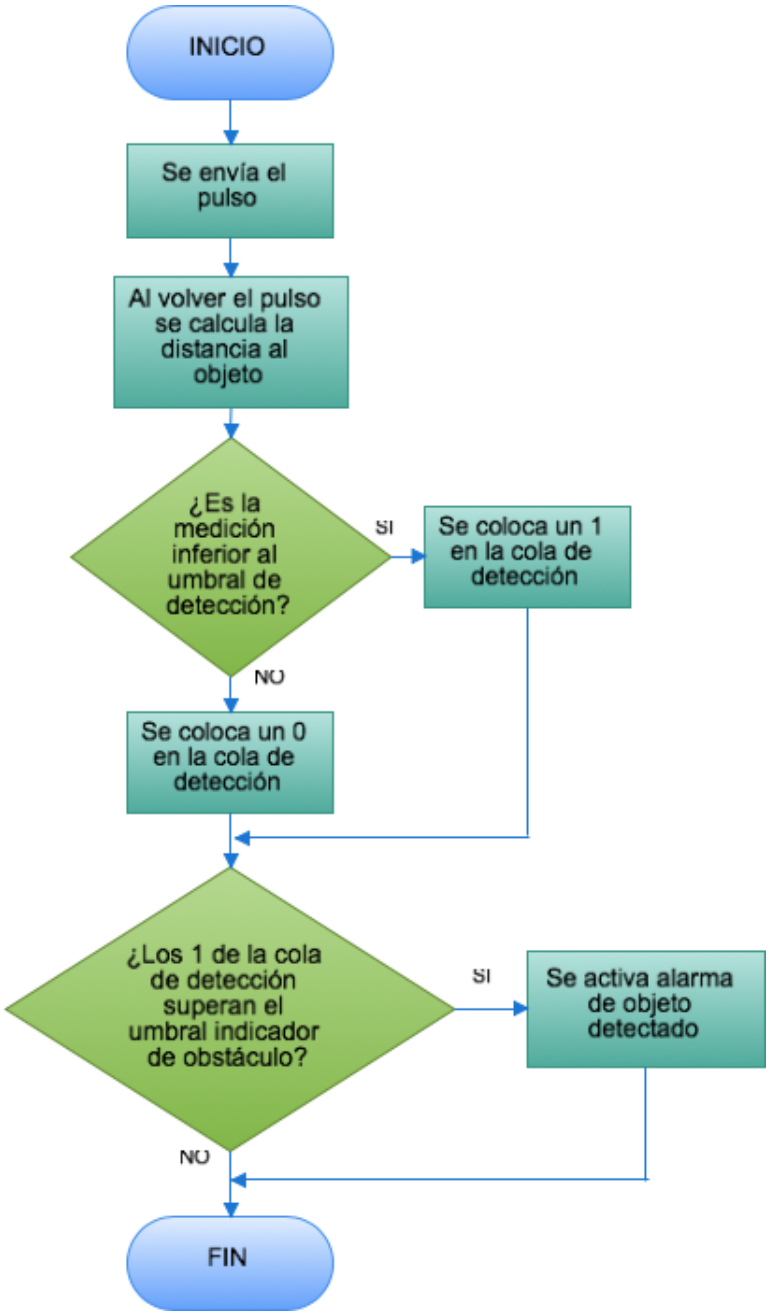


Figura 3.28: Diagrama de Flujo - Sensores de proximidad

Se probó con colas de 2, 3 y 4 elementos obteniendo los siguientes resultados:

Profundidad	Precisión	Tiempo de detección
2 elementos	Mala	Muy bueno
3 elementos	Muy buena	Muy bueno
4 elementos	Muy buena	Malo

Cuadro 3.5: Análisis de la profundidad de la cola de detección

De la tabla anterior, se toma como precisión mala cuando el sistema no

filtra bien el ruido y por lo tanto se detectan obstáculos que no existen. Un tiempo de detección malo se considera cuando el vehículo no llega a frenar y colisiona contra el obstáculo. Con una profundidad de 3 se logró un buen filtrado del ruido y una tiempo de respuesta suficiente como para evitar la colisión.
Este proceso anteriormente explicado, se repite de manera secuencial para todos los sensores. Los Trig de los sensores no pueden enviarse uno detrás de otro, se debe esperar que el pulso de uno de los sensores vuelva para poder enviar el del otro. Esto es para evitar el fenómeno de crosstalk que se produce cuando el pulso de un sensor entra en el Echo del otro generando un falso Echo y una medición errónea (sección 2.5.2.3). La interrupción de Timer 0 del microcontrolador está configurada para interrumpir la ejecución normal del programa cada 25ms, dentro de la misma se ejecuta el envío del pulso a través del pin Trig. Esto significa que cada 25ms se envía un pulso de Trig, pero considerando que la acción se ejecuta de manera secuencial para los 3 sensores, cada sensor envía su pulso cada 75ms. Se debe tener en cuenta, que según la hoja de datos del sensor, para un correcto funcionamiento la duración del pulso enviado debe ser como mínimo de $10\mu s$. A su vez, debido al filtro de la cola de detección, se toma una medición valida cada 3, y por lo tanto el período real por sensor con el que se detecta un objeto es de 225ms. Esto significa que si el vehículo recorre 40cm (umbral de detección de objeto cercano) en menos de 225ms se producirá una colisión, y por lo tanto la velocidad máxima que el mismo puede adquirir es $v = 177{,}7cm/s = 1{,}77m/s$ lo cual es suficiente para los objetivos del presente trabajo.

3.3.4.3. Medición de distancia recorrida y rotación del volante

Para poder realizar correctamente las traslaciones, se debe tener conocimiento de la distancia que el vehículo recorre en tiempo real. Este dato sumado al ángulo absoluto en que se encuentra la estructura permite conocer en que punto de la trayectoria se encuentra y actuar conforme para alcanzar el objetivo. Para obtener esta medición existen diversos sensores de los que ya se ha hablado en la sección 2.5.3 del capítulo anterior, se pueden utilizar odómetros, GPS, hacer el cálculo en base a los datos que brinda el acelerómetro o incluso hacer un cálculo combinando las mediciones de todos los anteriores.
A su vez, para que el vehículo gire deben rotarse las ruedas delanteras a través del volante, se debió diseñar por lo tanto, una metodología para medir el ángulo recorrido por el mismo.

Elección del sensor

Un módulo GPS con una buena precisión posee un error de 30cm. Esto montado en un vehículo autónomo real que recorre las calles de una ciudad, y que combina este dato con el de otros sensores es una excelente precisión. Pero a la escala que trabaja el presente proyecto integrador 30 cm de error en la medición de distancia recorrida es mucho. A esto se le suma el costo elevado de este módulo.
Los sensores acelerómetros brindan luego del procesamiento correspondiente, explicado en la sección 3.3.4.1 el dato de la aceleración en los 3 ejes, integrando este dato dos veces se puede obtener la distancia recorrida por el vehículo en un período determinado de tiempo. Debido a que la integral involucra sumar todos los puntos de una función dentro de determinados límites, el error se acumula en esta sumatoria y se hace muy grande. Esto provoca la necesidad de tener muy buenos sistemas de filtrado de los datos y sensores con bajo nivel de error. Y aún así la medición debería ser complementada con datos provenientes de otro sensor para poder tener datos más certeros. Esto sumado al costo elevado de un sensor acelerómetro de alta precisión, llevó a descartar el mismo como método para obtener el dato en cuestión.
Finalmente, los más óptimos para la presente aplicación son los sensores odómetros, cuyo funcionamiento se encuentra descrito en la sección 2.5.3. Los hay de muy buena precisión y los costos irán de bajos a elevados dependiendo de la misma. Además, existe la posibilidad de la fabricación propia de los mismos obteniendo muy buenos resultados a precios muy accesibles. Dentro de este tipo de sensores los hay analógicos y digitales, pero debido a que los datos son procesados dentro de un microcontrolador, la utilización del sensor analógico requeriría el uso del conversor analógico-digital del mismo, lo cual se prefirió evitar para no agregar más complejidad a las acciones realizadas por el Arduino. Esto sumado a que este sensor también puede utilizarse para medir la rotación del volante llevo a que se diseñe y fabrique un odómetro digital para la medición de las distancias recorridas.

Construcción y funcionamiento

Como se mencionó en la sección 2.5.3 un odómetro digital está básicamente conformado por un captador eléctrico, generalmente un imán, que acciona un contacto magnético o semiconductor de efecto Hall. A su vez este último al ser accionado envía un pulso eléctrico que será leído por el microcontrolador.

Para la construcción de este sistema de medición se utilizó un sensor de efecto hall convencional, el FS177, de fácil y rápida adquisición en el mercado. Para su conexionado se tomo como guía el circuito de prueba de su hoja de datos:

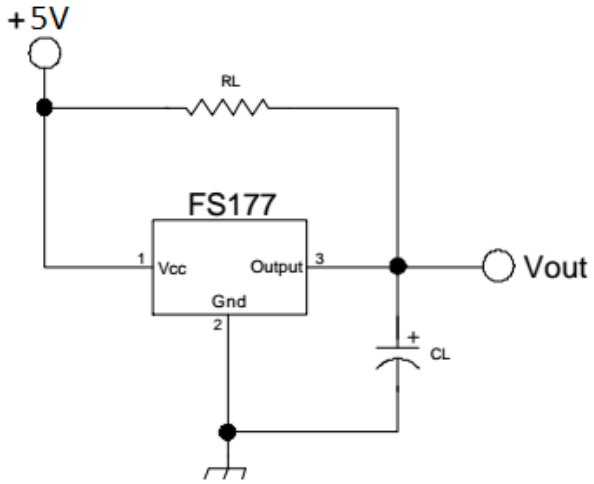
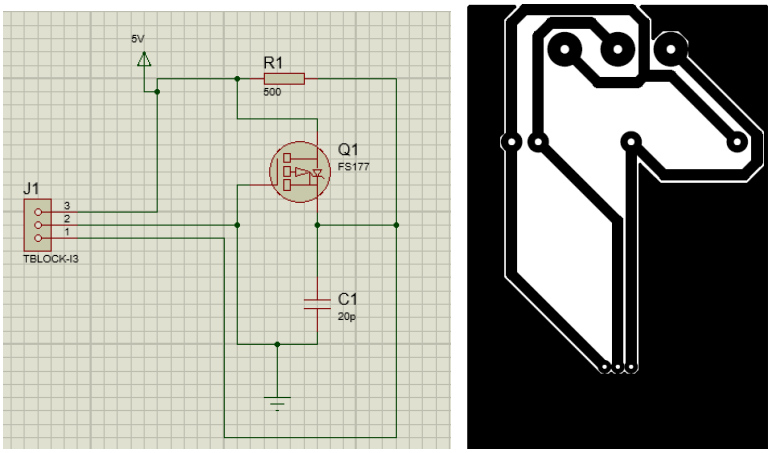


Figura 3.29: Conexionado sensor de efecto hall

En la hoja de datos el circuito se encuentra conectado a una tensión positiva de 12V, pero en este caso fue conectado a una tensión de 5v extraída del microcontrolador. Dicha tensión esta dentro de los rangos permitidos para este sensor. Además, se extrajo de la hoja de datos el valor de los componentes para el anterior circuito utilizado en las pruebas realizadas por el fabricante, véase sección anexa B. Se tomo entonces $RL = 500\Omega$ y $CL = 20pF$. La hoja de datos del sensor puede encontrarse en el anexo B. Se presentan a continuación el esquemático y PCB de la placa diseñada para la implementación del anterior circuito:



(a) Esquemático (b) PCB

Figura 3.30: Implementación circuito sensor de efecto hall

En base a la figura 3.30 se confeccionaron dos placas, una para medir la traslación del vehículo y la otra para medir la rotación del volante. Como accionadores de los contactos magnéticos se utilizaron imanes de neodimio de 3mm de diámetro por 3mm de espesor, se eligió este tipo de imanes por la gran potencia de su campo magnético. Para medir la distancia recorrida se colocaron 36 imanes concéntricos a lo largo del perímetro de la rueda trasera a la que está conectado el motor mientras que para medir rotación se colocaron 11 imanes sobre la parte inferior del volante, equiespaciados y simétricos al centro del mismo. Además todos estos imanes deben colocarse con el campo intercalado. Ambas placas se colocaron de manera tal que los sensores de efecto hall queden enfrentados a los imanes. El sensor tiene una placa interna que cambia de posición en base al campo magnético al que esta expuesta, de esta manera al girar la rueda o el volante, los imanes con campos intercalados irán pasando por el frente del sensor y generando un cambio en la posición de esta placa interna, a su vez este cambio de posición genera un cambio de estado en el pin de salida del FS177.

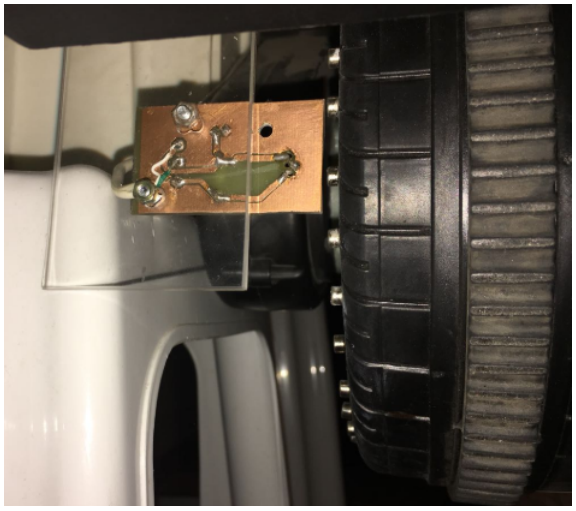


Figura 3.31: Odómetro digital para medición de distancia de avance

Debido a que la salida del sensor de efecto hall cambia de estado cada vez que un imán pasa en frente a él, conociendo la distancia que significa el paso de un imán a otro, se puede conocer la distancia recorrida por el vehículo en base al conteo de estos pasos. Como se han colocado 36 imanes a lo largo de toda la rueda, considerando que la misma posee 20 cm de diámetro y por lo tanto 62.8 cm de perímetro, se tiene un paso de 1.74 cm. Esto quiere decir que el odómetro digital diseñado posee un error máximo de 1.74 cm para la medición de la traslación del vehículo, lo cual es un error aceptable considerando las dimensiones del mismo y los espacios por los que se desplaza.

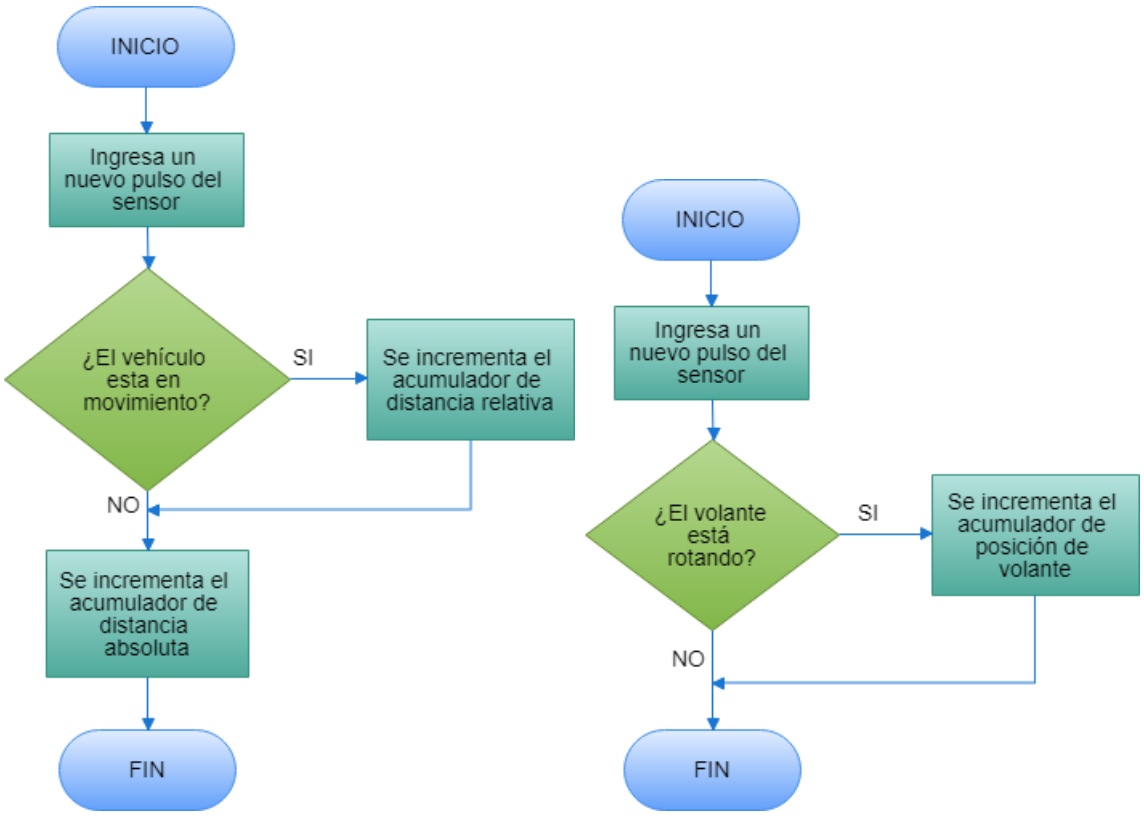
Para la medición de la rotación no interesa tener una unidad de medición, ya que lo único que se busca es colocar el volante en una determinada posición. Es por eso que no se tomó ninguna unidad de medida y se trabajo directamente con pulsos. Se colocó un imán central, donde las ruedas quedan derechas, y 5 imanes para cada uno de los lados. El error de este odómetro particular no interesa demasiado ya que sólo se busca que el volante pueda colocarse en 5 posiciones, giro total para ambos lados, un giro intermedio para ambos lados y centrado.

Implementación en Arduino

La implementación en el caso de este sensor es muy sencilla. Debido a que la velocidad del vehículo se ajusta a la que el microprocesador le exige a la parte de bajo nivel, no se conoce en que momento el estado del pin de salida del sensor de efecto hall cambiará. Es por esto que lo más conveniente en este caso fue conectar los pines de salida de los sensores a pines digitales del microcontrolador, configurados como INPUT, que poseen interrupción externa. Además la interrupción externa se configuró por flanco de subida y bajada, es decir cada vez que el valor del pin cambia de un estado a otro se producirá la interrupción. De esta manera se puede detectar el paso de cada uno de los sensores colocados en la rueda y volante. Los pines digitales utilizados fueron el 2 y 3, para medición de distancia recorrida y de rotación respectivamente.

Cómo se puede observar en los diagramas de la figura 3.32 cada vez que se produce una interrupción se considera que ha entrado un nuevo pulso por parte del sensor. En el caso *a* se incrementan dos distancias. Si el vehículo está en movimiento se incrementa una distancia relativa al movimiento que se está ejecutando en ese momento, recordando que el sistema ejecuta una trayectoria completa a través de la ejecución de múltiples movimientos. Esta variable será reseteada al inicio de un nuevo movimiento. La distancia absoluta en cambio, se incrementará siempre que se produzca la introducción y no se resetea hasta terminar la trayectoria, de esta manera lleva la cuenta de la distancia total recorrida. Ambas variables llevan la cuenta en cm y se incrementan en pasos de 1.74 cm de acuerdo a cálculos realizados anteriormente. En el caso *b*, si el volante del vehículo está rotando y se produce la interrupción se incrementa una variable que marca la posición del volante, en caso de no estar rotando el volante el incremento no se produce. Esta variable acumula en pulsos, los cuales indican en tiempo real cuanto ha rotado el volante. Cada vez que el volante vuelve al centro, la posición del mismo se resetea a 0. Que los incrementos se den solo si el volante

está rotando o sólo si el vehículo está en movimiento tiene una razón, en teoría si estas condiciones no se dieran las interrupciones directamente no se producirían, pero como en la práctica se pueden tener casos en donde se infiltra ruido que genera las interrupciones se utilizan estas condiciones para filtrarlo.



(a) Distancia recorrida (b) Rotación volante

Figura 3.32: Implementación de medición de distancia con sensores de efecto hall

3.3.5. Sistema de traslación

En esta sección se muestra la lógica utilizada en los algoritmos para que el vehículo pueda desplazarse en el espacio. Puntualmente se tratará el desplazamiento el linea recta, el sistema de iteración para lograr ángulos mayores y el ajuste para que el motor alcance las velocidades deseadas.
En la figura 3.33 se muestra el funcionamiento de la implementación en el microcontrolador de las funciones para el accionamiento del motor de traslación.

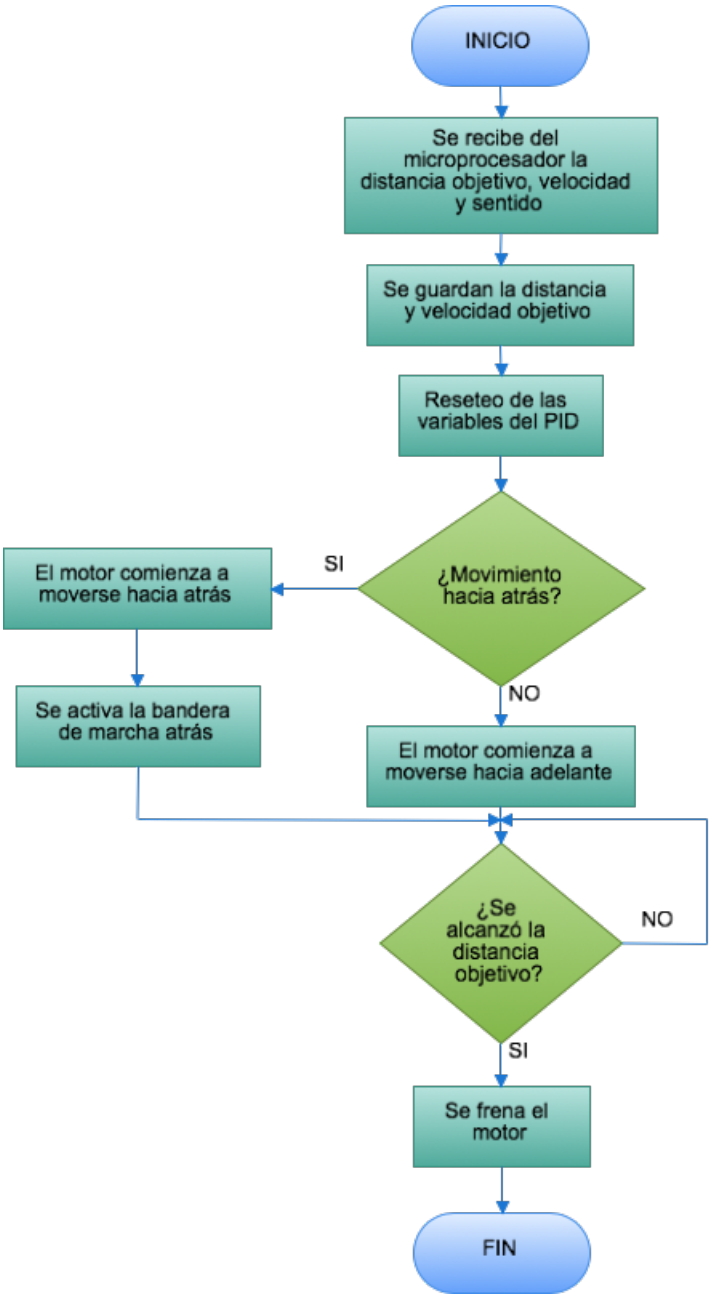


Figura 3.33: Accionamiento de motor de traslación

En el microcontrolador Arduino, se tiene un bucle infinito que se ejecuta constantemente, dentro de este bucle se tienen diferentes bloques condicionales que se ejecutarán dependiendo en la etapa del proceso del movimiento en la que el vehículo se encuentre. Dentro de este bucle el microcontrolador revisa la cola de la comunicación serie en busca de nuevos datos, cuando los mismos están disponibles se decodifican y se revisa el tipo de acción que el microprocesador esta pidiendo ejecutar. En caso que el elemento de índice uno del arreglo recibido sea la letra k se procederá a iniciar un nuevo movimiento descrito en el diagrama anterior. La función de giro será explicada detalladamente en el siguiente apartado, en este caso sólo se hará

foco en el movimiento del motor de traslación. Primero se extrae de los datos recibidos la distancia objetivo a recorrer en el movimiento actual, a qué velocidad y en qué sentido. Luego, se procede a resetear las variables del sistema de control PID para poder controlar un nuevo movimiento. Seguidamente, comienza el movimiento del motor en el sentido deseado, en caso que el sentido sea marcha atrás se pondrá en alto una bandera que activa el bloque de detección de objetos traseros. El código desarrollado en el microcontrolador puede observarse en C. El motor se pondrá en marcha a través del envío de una señal de PWM al mismo. El estado del motor depende del ciclo de trabajo del pulso enviado y se comporta según la siguiente tabla:

Ciclo de Trabajo(%)	Estado Motor
0 - 49	Motor marcha adelante
50	5v Motor detenido
51 - 100	Motor marcha atrás

Cuadro 3.6: Ciclos de trabajo PWM y respuesta del motor

A su vez, la velocidad del motor será controlada a través de un PID que será explicado más adelante. Dentro del Timer 0 del microcontrolador, configurado para producir interrupciones cada 25 ms, se controlará la distancia recorrida acumulada por el odómetro digital y una vez que se alcance la distancia objetivo se frena el motor y el movimiento finaliza. Las variables que acumulan distancia son reseteadas en este punto.
Cuando el movimiento además de incluir una traslación incluye una rotación, puede darse el caso en que se llegue al objetivo de avance en línea recta, pero no se hayan alcanzado los grados de rotación deseados. En ese caso se sigue el procedimiento descrito en la figura 3.34.

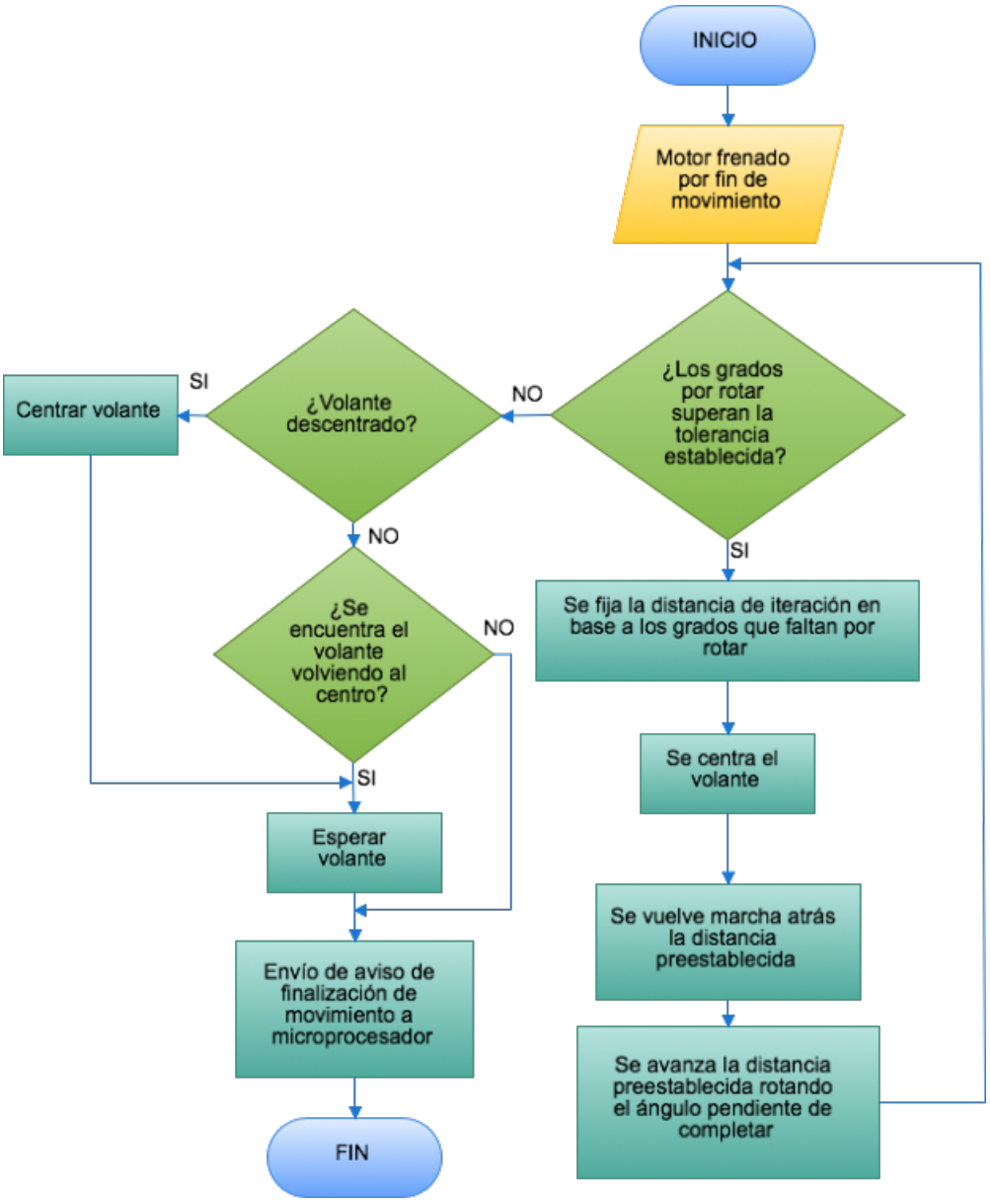


Figura 3.34: Sistema de iteración para alcanzar rotación objetivo

Al finalizar el movimiento, se revisa si el vehículo estaba rotando y si completo los grados que debía rotar o si quedan grados pendientes. En caso de haber grados pendientes, si se trata de más de 10 °, se ejecutará un procedimiento para completar el movimiento, en caso de ser menos se considera que los grados por rotar no son significativos y se ejecutarán en el siguiente movimiento. Para completar el movimiento, primero se fija una distancia de traslación sobre la cual se iterará, en caso de faltar más de 20 ° por completar la misma será de 40 cm y en caso contrario de 30 cm. Esto es porque en caso de ángulos grandes se necesita más margen de distancia para completarlos. El paso siguiente será centrar el volante, ya que si quedo una rotación incompleta el volante estará cruzado. Luego

se vuelve marcha atrás la distancia de iteración, esto es para poder tener más espacio para poder girar, imitando lo que hace un auto real cuando se queda sin radio de giro. Finalmente se avanzará la distancia de iteración pero ahora rotando el ángulo pendiente de completar. Puede darse el caso en que en una de estas maniobras no se complete el ángulo deseado, en ese caso el vehículo itera hacia adelante y atrás hasta completar el movimiento en cuestión. Una vez que se logra completar los ángulos a rotar objetivos del movimiento, se verifica que el volante esté centrado, en caso negativo se lo centra, se espera que llegue al centro y se avisa al microprocesador que el movimiento ha sido realizado con éxito. Todo esto se ejecuta revisando el estado de los sensores (distancia recorrida y ángulo rotado) dentro de la interrupción de 25 ms y con una máquina de estado que se encuentra el el bucle principal que va avanzando de acuerdo a lo explicado anteriormente. Como ya se ha mencionado la velocidad del motor es regulada por un controlador PID de la siguiente manera:

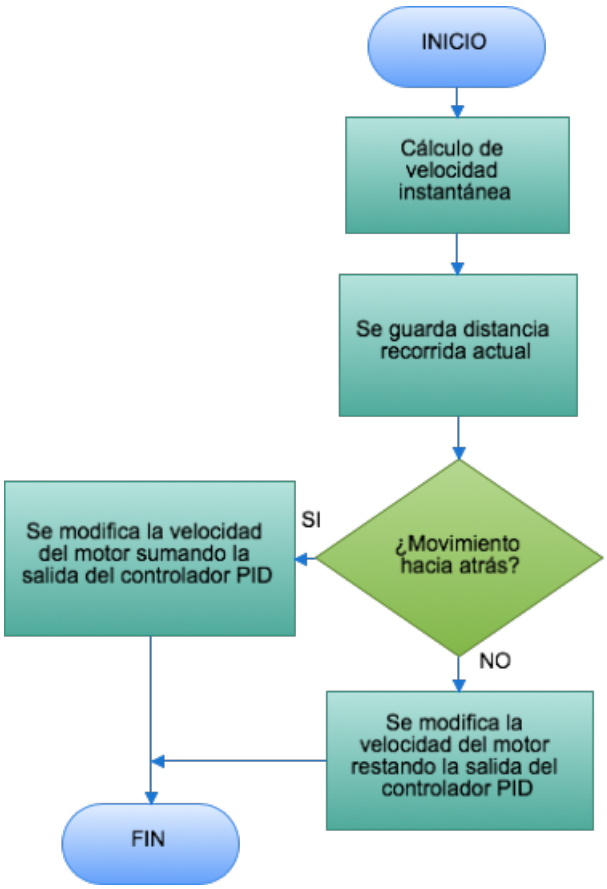


Figura 3.35: Control de la velocidad del motor con PID

Para la regulación de la velocidad, dentro del bucle principal del microcontrolador se tiene un bloque que se ejecuta cada 500 ms, en el cuál se calcula la velocidad instantánea en base al dato de distancia recorrida

actual y de la distancia recorrida de la ejecución anterior del bloque. Luego se evalúa en que sentido está girando el motor, en caso de ir hacia atrás se calcula la salida del controlador PID teniendo como entrada la velocidad instantánea y se suma esta salida al ancho de pulso actual de la señal PWM que va al motor. En caso de estar girando el motor hacia adelante el procedimiento es el mismo pero el valor de la salida del controlador se le resta al ancho del pulso.
Este bloque, como ya se mencionó, se ejecuta cada 500 ms, mientras no se haya alcanzado la velocidad objetivo el controlador producirá valores positivos lo cual generará en ambos casos un aumento en la velocidad de giro del motor. En caso de superarse la velocidad de referencia, el controlador producirá valores negativos generando un decremento en la velocidad del mismo.

3.3.6. Sistema de orientación

Dentro de esta sección se explica detalladamente el funcionamiento de los sistemas de orientación angular del vehículo en el espacio. Además se explican detalladamente las funciones encargadas de ejecutar las rotaciones del mismo.
El sistema toma constantemente la posición angular absoluta del vehículo en el espacio. El microprocesador envía la posición absoluta en donde se desea que el vehículo se encuentre, al finalizar el movimiento en curso, de esta manera se puede posicionar al vehículo en el lugar deseado de una manera absoluta, evitando acumular errores de movimientos rotativos relativos a las posiciones anteriores. Esto además permite, que si se incurrió en un error de posición angular en un movimiento, el error se corrija en el movimiento siguiente. Son ejemplos de esto:

- Si el vehículo en un movimiento debe colocarse en el espacio a $-15°$ pero debido a una perturbación externa terminó a $-20°$, el sistema de alto nivel no se enterará de este error y por lo tanto continuará la trayectoria como si nada hubiera pasado. Si el siguiente movimiento requiere un posicionamiento angular en $-40°$, el vehículo buscará posicionarse en ese punto sin importar el error en el que incurrió anteriormente. Esto no podría ser así si simplemente los giros se hicieran una determinada cantidad de grados sin importar la posición absoluta de la estructura, con estas rotaciones relativas al movimiento actual los errores se acumularían hasta el fin de la trayectoria.

- Si el vehículo se mueve en línea recta, por ejemplo alrededor del ángulo

0°, y debido a una perturbación externa se desvía de esta trayectoria recta, en el siguiente movimiento que envíe el microprocesador se podrá corregir este error en la trayectoria y continuar en línea recta.

Para poder tomar la posición angular absoluta, se establece como el 0° la posición inicial del vehículo y se trabaja dentro de un plano ±180° como muestra la figura 3.36.

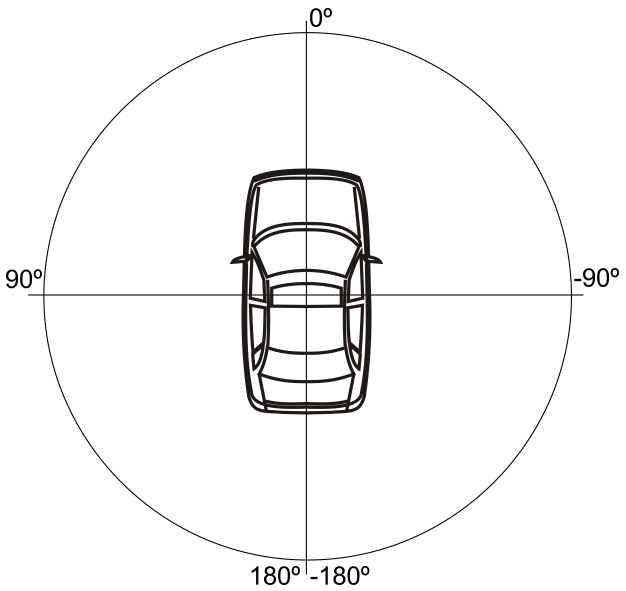


Figura 3.36: Plano de posición absoluta del vehículo

En la figura 3.37 se presenta un diagrama de flujo donde se detalla el proceso que el sistema sigue para efectuar un giro.En el bloque del bucle infinito del microcontrolador donde se reciben los movimientos que envía la parte de alto nivel, se recibe la posición angular absoluta deseada. Con este dato y la posición angular absoluta actual del vehículo se calculan los grados y el sentido en que el vehículo debe rotar para alcanzar el objetivo. Para evitar rotar ángulos muy pequeños, que pueden entorpecer los movimientos solo se ejecuta la rotación si los grados a rotar superan un umbral mínimo de 5°. En caso de no superar el umbral estos grados no se pierden, gracias al sistema de posicionamiento absoluto, en el próximo movimiento se compensarán. Poniendo un ejemplo, si el vehículo se encuentra en 25° y se le indica girar hasta 24°, el vehículo descartará esta acción, pero en el movimiento siguiente si se le pidiera que se coloque a 45° la estructura rotará un total de 25°, los 4 descartados y los 20 que hubiera tenido que rotar a continuación. Siguiendo con el proceso de rotación, en caso de superar el umbral, se girará el volante en el sentido deseado y debido a que el vehículo mientras se ejecuta esta acción se encuentra desplazándose, el mismo comenzará a rotar en la dirección que el volante ha girado. Dentro

de la interrupción por Timer de 25 ms se controla si se ha llegado a rotar los grados objetivo, en caso que así sea se procede a enderezar el volante.

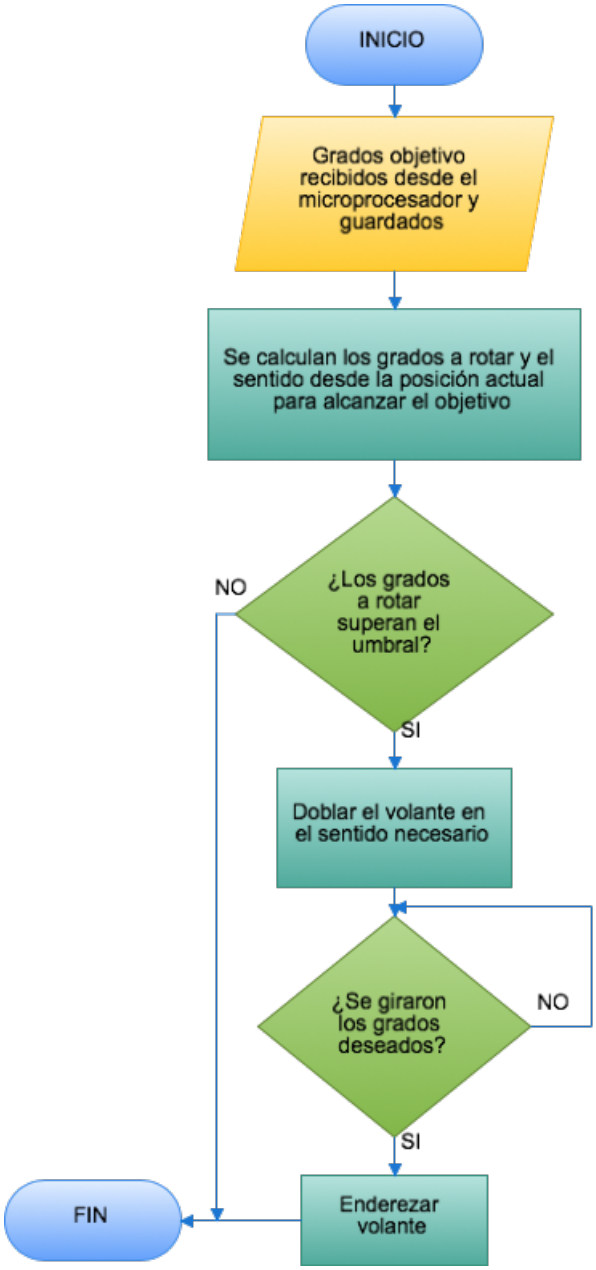


Figura 3.37: Implementación de la función de giro

A continuación se detalla el procedimiento que seguido para controlar la rotación del volante:

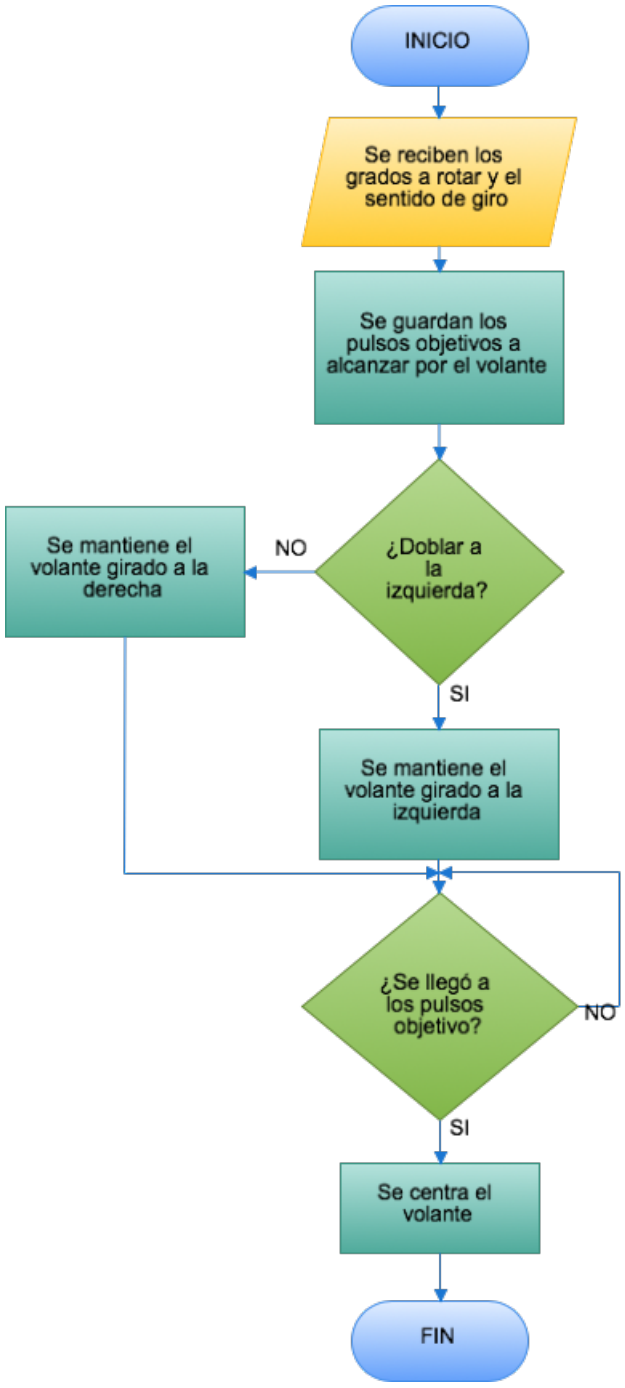


Figura 3.38: Diagrama de flujo del control del volante

La función encargada de mover el volante recibe los grados a rotar y el sentido de giro. Los grados a rotar vienen expresados como cantidad de pulsos a rotar y son guardados como pulsos objetivo. Se evalúa si se debe girar hacia la izquierda o derecha, y dependiendo de esta evaluación se comienza a girar el volante en un sentido u otro. El volante está conectado a un motor de corriente continua idéntico al utilizado para realizar las traslaciones. La forma de accionar este motor es la misma, a través de señales PWM, sólo que en este caso poseen ciclos de trabajo fijos de 25 %

y 75 % dependiendo el sentido de giro. Una vez que este motor comienza a girar, dentro de la interrupción de Timer de 25 ms se controla el valor de pulsos acumulados del odómetro digital encargado de este motor, cuando se llega al número de pulsos objetivo el ciclo de trabajo de la señal PWM se pone al 50 % y el motor se detiene. la cantidad de pulsos recorridos por el volante dependerá de la cantidad de grados que el vehículo debe rotar, si se superan los 10° el volante rotará hasta el tope (5 pulsos), de lo contrario se considera un giro leve y el volante rotará solo hasta el tercer pulso. Una vez que el vehículo llega a la posición angular objetivo, se debe centrar el volante, esto se realiza volviendo la misma cantidad de pulsos que se ha avanzado. Para que todo esto funcione, cada vez que el volante va comienza a girar, se resetea la variable que acumula los pulsos. Algo para resaltar es que al inicio de la trayectoria, antes de realizar cualquier movimiento, la posición del volante no es conocida, es por eso que se acciona el motor una cantidad fija de segundos para asegurar que el volante ha llegado a tope, y luego se lo vuelve 5 pulsos que es la distancia que hay hasta el centro, esta es la llamada función de centrado de volante que puede además, ejecutarse en cualquier momento posterior de la trayectoria.
Para conocer la posición absoluta en que se encuentra el vehículo en tiempo real, es necesario acumular los grados que el mismo va rotando a lo largo de toda la trayectoria. Este dato es extraído del sensor MPU6050 visto en la sección 3.3.4.1. Cómo se vio en esta sección, la librería realizada para extraer los datos del mismo, provee el dato de la velocidad angular en el eje z para un determinado período de tiempo. Conociendo la velocidad angular ω del vehículo se puede conocer su desplazamiento $\theta - \theta_0$ angular entre los instantes t_0 y t a través de la siguiente ecuación:

$$\theta - \theta_0 = \int_{t_0}^{t} \omega dt \tag{3.5}$$

El producto $\omega\ dt$ representa el desplazamiento angular del vehículo en el intervalo dt. En este caso puntal, el dato de la velocidad angular se toma cada 25 ms dentro del Timer 0 del microcontrolador, por lo tanto este será el dt utilizado para el cálculo. El desplazamiento total es la suma de los infinitos desplazamientos angulares infinitesimales en el intervalo dt.
En la siguiente figura se muestran las acciones ejecutadas para obtener la posición angular absoluta:

Figura 3.39: Diagrama de flujo cálculo posición angular absoluta

Cómo se mencionó anteriormente primero se obtiene el dato de la velocidad angular del último período de tiempo, dato entregado por el acelerómetro. Luego se calculan en base a la ecuación 3.5 los grados girados sobre el eje z para el último período de tiempo. Se evalúa si este dato supera un escalón de ruido igual a 0.05, en caso afirmativo se acumula el valor en la variable que guarda la posición absoluta del vehículo, en caso contrario se descarta el valor. Este último procedimiento cumple la función de filtro pasa bajos en la toma de este dato. El valor de posición angular absoluta, debido a la acumulación constante, tiende a valores grandes, es por esto que periódicamente se re-mapea el valor de este ángulo para enviarlo al plano $\pm 180^{\circ}$ donde trabaja el vehículo durante el desplazamiento.

3.3.7. Sistema de detección de obstáculos dinámicos

Se explica a continuación el sistema de bajo nivel utilizado para, en base a los datos filtrados de los sensores de ultrasonido y en actuando en conjunto con el sistema de alto nivel, esquivar objetos en forma dinámica. Sin este sistema el vehículo colisionaría contra todos los obstáculos no contenidos en el mapa original.
En el siguiente diagrama se puede observar las acciones tomadas ante un obstáculo detectado en la parte delantera del vehículo:

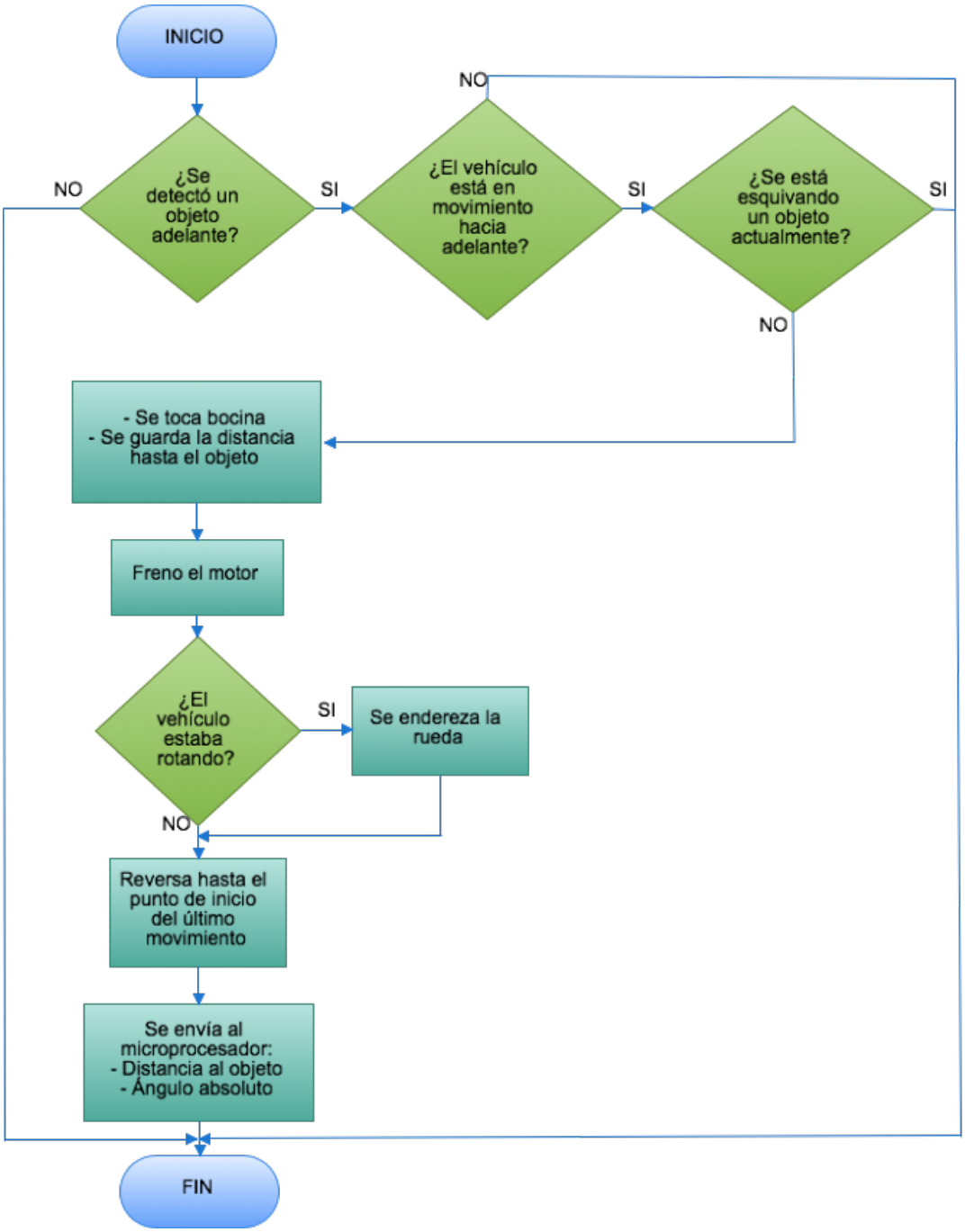


Figura 3.40: Detección dinámica de objetos delanteros

El sistema que controla los sensores de ultrasonido cómo ya se ha visto en la sección 3.3.4.2 filtra los datos medidos y levanta una bandera que indica que se ha detectado un objeto. Cómo se tienen 3 sensores de ultrasonido, se cuenta con 3 banderas de objeto detectado. Dentro de la interrupción de Timer 0, donde se realizan la mayoría de las acciones periódicas, se controla cada 25 ms si alguno de los dos sensores delanteros ha detectado un objeto, en caso afirmativo se consulta si el vehículo se está moviendo hacia adelante, ya que de otra manera no sería un problema para la trayectoria. Además se evalúa si no se está en ese momento en proceso de esquivar otro objeto, ya que de ser así se debe aguardar a terminar este proceso para esquivar un objeto nuevo, en este caso no hay riesgo de colisión ya que lo que realiza la parte de bajo nivel para evitar el obstáculo siempre es marcha atrás. En caso de darse todas estas condiciones el sistema toca bocina y enciende las luces para para anunciar que se ha cruzado con un objeto inesperado, guarda la distancia que hay hasta el objeto y frena el motor. Luego, se procede a hacer reversa hasta llegar al punto de inicio del movimiento que estaba en curso, en caso de tener el vehículo las ruedas rotadas se enderezan las mismas antes de ejecutar esta acción. Una vez que se llegó al punto deseado, se avisa al microprocesador que se ha detectado un obstáculo y se le envían la distancia hasta el objeto y el ángulo absoluto en que se encuentra el vehículo, el sistema de alto nivel utilizará estos datos para agregar el nuevo obstáculo al mapa y recalcular la trayectoria. Estos pasos antes descritos se realizan dentro de una máquina de estados en el bucle principal del microcontrolador.
El caso de la detección trasera es mucho más simple, ya que sólo se utiliza para evitar una colisión mientras se hace reversa, pero no se utiliza para agregar nuevos objetos al mapa. Esto genera que todo sea resuelto en el microcontrolador.

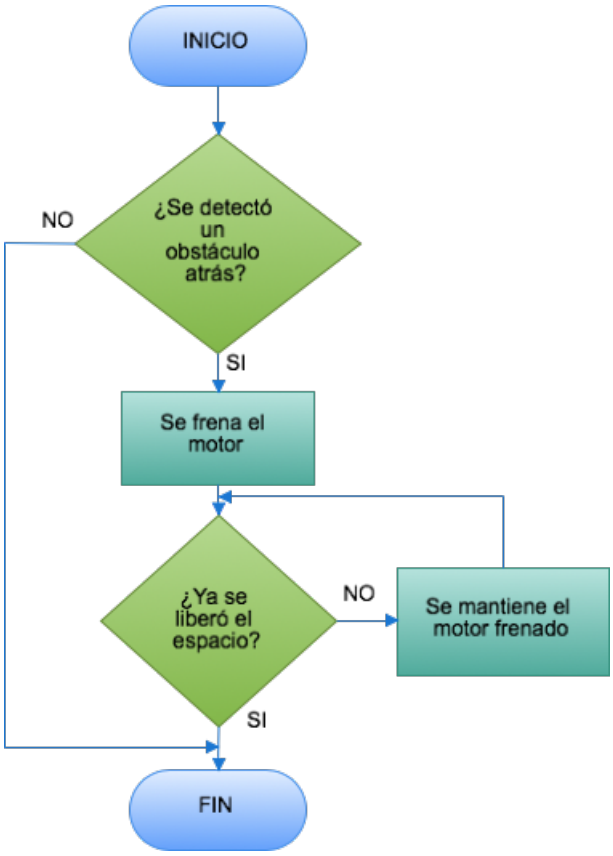


Figura 3.41: Detección dinámica de objetos traseros

Cómo se puede observar en el diagrama de la figura 3.41, dentro del Timer de 25 ms se controla si se ha detectado un obstáculo en la parte trasera, esta acción se ejecuta únicamente si la bandera que indica marcha atrás está levantada, en caso de darse estas condiciones, se frena el motor y se consulta periódicamente si el obstáculo ya se ha ido, el motor se volverá a accionar y se continuará con la tarea que se estaba haciendo sólo si el obstáculo desaparece. Este modo de operación es debido a que las marchas atrás se hacen siempre sobre el recorrido que el vehículo ya ha realizado, donde no había obstáculos, por lo tanto se presupone que en caso de haber un obstáculo, el mismo es pasajero.

3.3.8. Sistema de control de velocidad crucero

Para controlar la velocidad se desarrolló un controlador PID (proporcional, integral, derivativo). En primer lugar se diseñó el modelo del motor en forma empírica, debido a que no se contaba con los datos necesarios del mismo para realizar el cálculo analítico. En segundo lugar, se procedió al diseño del controlador en base al modelo de motor calculado y a la especificaciones de diseño establecidas.

3.3.8.1. El modelo empírico del motor

Para obtener el modelo empírico del motor de corriente continua, se varió la entrada al mismo mientras se registraba simultáneamente, con ayuda de un osciloscopio, la respuesta de velocidad obtenida y la entrada aplicada.
El tipo de entrada aplicada recomendable para excitar el sistema motor adecuadamente es una señal no determinística; por lo que se generó una señal de pulsos de amplitud constante con duración en alto y bajo variables, simulando de esta forma una entrada no determinística. Para tal fin, se conectó al motor directamente a la batería de 6V utilizada para su alimentación, y se generaron los pulsos a través de una llave electrónica manual. Los componentes utilizados en el experimento se colocaron en base al siguiente esquema:

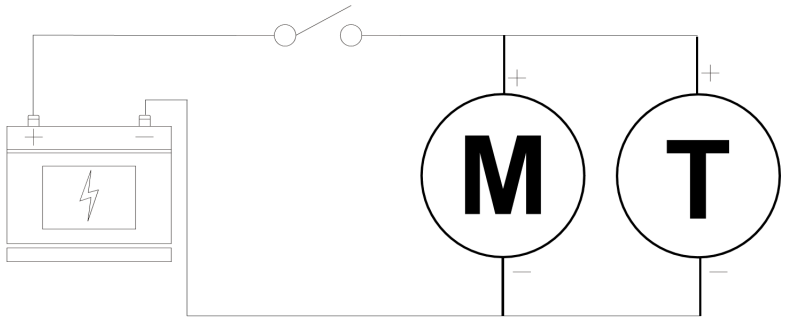


Figura 3.42: Bloques para el cálculo del modelo experimental del motor

Cómo se mencionó anteriormente, se conectó directamente la batería de 6V a la entrada del motor, con una llave electrónica o switch manual de por medio. Cómo se puede observar el la figura 3.42 se colocó un motor cuyo eje es solidario al del motor principal y que hace las veces de tacómetro, permitiendo medir en sus bornes la salida necesaria para el cálculo de la función de transferencia del sistema en cuestión. En la figura 3.43 puede observarse cómo se colocó dicho instrumento:

Figura 3.43: Tacómetro para medir la salida del motor principal

Para el experimento se utilizó una computadora con la herramienta MatLab y la biblioteca de identificación de sistemas **systemIdentification** instalados. El motor que se deseaba medir, el motor que cumple la función de tacómetro, una batería de 6v y un osciloscopio marca 'GW Instek'.
Usando como referencia la figura 3.42 se siguieron estos pasos:

1. Se ajustaron las propiedades del osciloscopio como muestra la tabla 3.7.
2. Se colocó el canal 1 del osciloscopio para medir la entrada al motor, una punta a masa y la otra a la salida del switch.
3. Se colocó el canal 2 para medir la salida del motor en los bornes del motor que se utilizó como tacómetro.
4. Se encendió el osciloscopio y se simuló la señal de excitación pseudo-aleatoria abriendo y cerrando el interruptor varias veces para producir una señal con amplitud constante y duraciones en alto y bajos variables; tal como muestra la figura 3.44. Una vez que se contaba con la figura deseada se le dio stop al osciloscopio para captar las señales en la ventana de tiempo correcta, la duración total de la medición fue de 10 segundos.
5. Los datos capturados con el osciloscopio se guardaron, en formato CSV, en una memoria SD.

Función	Propiedades
Ajustes Canal 1	5V por división
Ajustes Canal 2	5v por división
Barrido Horizontal	1 segundo por división

Cuadro 3.7: Ajustes del osciloscopio para el experimento

Cómo se puede notar en la figura 3.44 se cuenta con una entrada bastante ruidosa, ruido que también se ve reflejado en la salida. No se ha trabajado en la disminución de este ruido debido a que la herramienta MatLab lo maneja muy bien a la hora de calcular un modelo representativo, como se verá más adelante en esta sección.

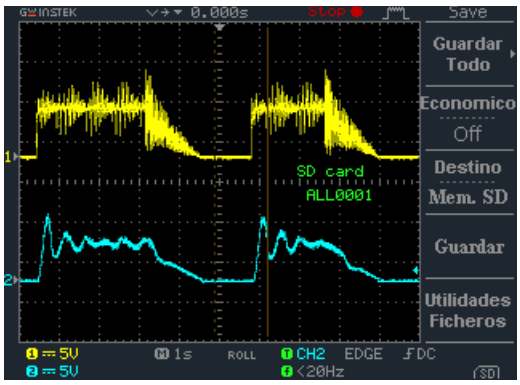


Figura 3.44: Excitación y respuesta del motor de corriente continua

Procesamiento de los datos obtenidos

Los datos que se guardaron desde el osciloscopio en la tarjeta SD son la imagen que se muestra en la figura 3.44 y dos archivos con formato .CSV que contienen la configuración y todos los datos de cada canal. De estos datos los que interesan son el tiempo de muestro, cada punto de entrada y cada punto de salida. Con esto se armó un único .CSV de tres columnas: **Tiempo**, **Entrada** y **Salida**, el cual sirve para ser procesado directamente por la herramienta MatLab.

Este archivo se importó en la herramienta MatLab y sus columnas fueron guardadas en variables, las cuales se utilizaron con la librería **systemIdentification** para poder generar el modelo del motor. Cómo se puede observar en la figura 3.44 se contaba con dos escalones, uno fue utilizado por la librería para identificar el modelo y el otro para validarlo. Para esto se le debe indicar a la librería que datos utilizar para cada fin y que modelo se quiere aproximar. En este caso por la simplicidad del modelo que se infiere de los resultados teóricos obtenidos, detallados en la sección anexa A.3.1 se estimaron tres modelos: uno de primer orden simple, uno de primer orden con retardo y el último de segundo orden con polos reales.

Validación del modelo empírico

Una vez hechas todas las configuraciones anteriores, se procedió a correr la herramienta y generar los modelos. La salida de los mismos se puede comparar en la figura 3.45.

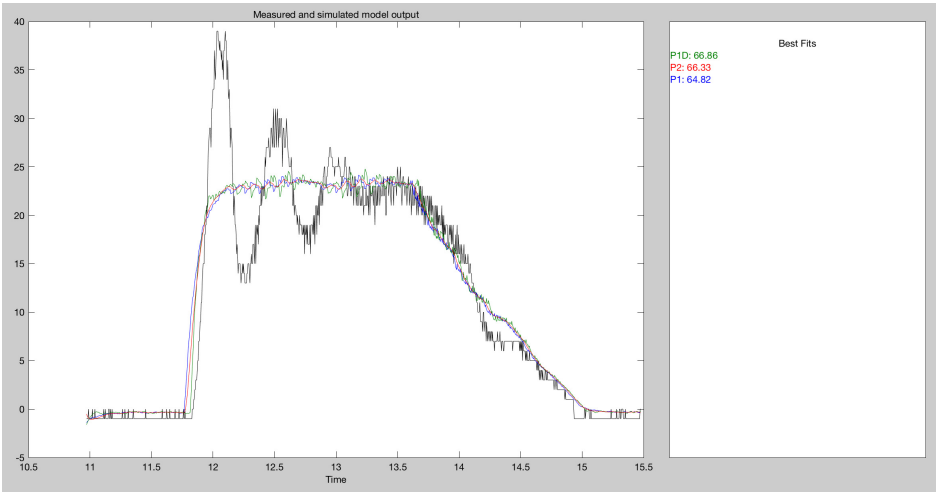


Figura 3.45: Comparación de los resultados de la identificación

En la figura anterior puede observarse como los modelos tienen la pinta estar 'promediando' la función real de respuesta del motor, esto es deseable, ya que están filtrando el ruido que se tenía en la entrada. Como puede observarse en la figura 3.45 los modelos son muy similares entre sí.

Selección del mejor modelo empírico

Cómo se dijo anteriormente, las aproximaciones obtenidas son todas aceptables. Debido a esto, el criterio de selección del modelo a utilizar fue en base a la alta simplicidad del mismo, por lo que el de primer orden sin retardo fue el escogido.

Una vez elegido, se exportó al espacio de trabajo de MatLab el modelo seleccionado en donde se pudo ver la función de transferencia del mismo, la cual está representada por la siguiente ecuación:

$$ft\_motor = \frac{9{,}468}{(s + 12{,}37)} \tag{3.6}$$

Una vez obtenida la función de transferencia se procedió a simular el modelo obtenido con los datos experimentales y comparar la salida real contra la salida simulada.

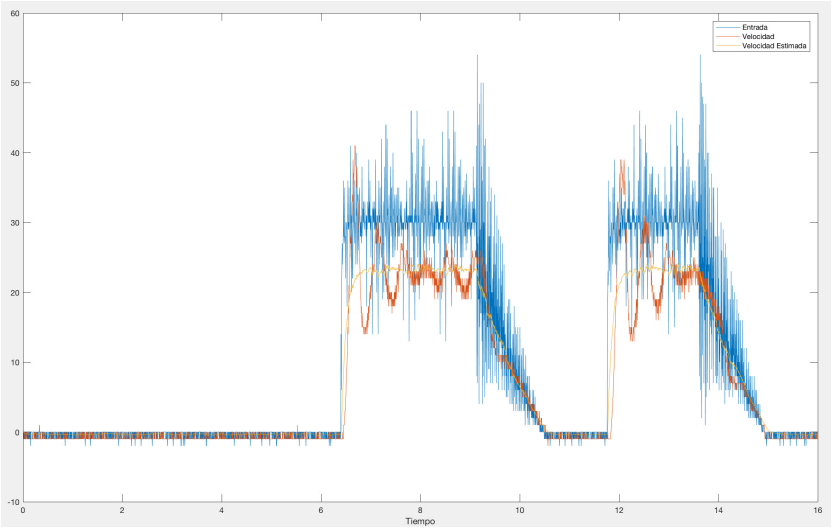


Figura 3.46: Verificación de la respuesta del modelo seleccionado

En la figura 3.46 podemos ver la entrada al sistema, la salida real y la simulada en un mismo gráfico. Puede observarse que la salida simulada representa muy bien la real y por lo tanto el modelo es útil para el cálculo del sistema de control.
El análisis teórico, detallado en la sección anexa A.3, muestra la posibilidad de que el modelo del motor tenga orden dos, con polo reales. Finalmente luego del experimento, se encontró que existe solamente un polo dominante y que el modelo no posee un retardo apreciable. Es por estas razones por las que finalmente se eligió la representación experimental de orden 1.

3.3.8.2. Selección de la estructura de control y la arquitectura del controlador

En este caso, se procedió a la elección de una estructura de control tradicional de un solo lazo, con un solo grado de libertad y el controlador en el camino directo.
Como se mencionó en la sección A.4 cuando no se tiene mucho conocimiento de la planta, el controlador PID es el más adecuado ya que permite el ajuste de los tres parámetros según las necesidades que el modelo presente. Además como se mencionó en esta misma sección el controlador PID suele ser un buen compromiso entre rapidez y error. Cómo se puede observar en la tabla 3.9 una de las especificaciones de diseño planteadas es error de estado estacionario cero para entrada escalón. Si se observa el modelo de motor obtenido, se tiene un sistema de primer orden tipo cero (no posee polos en el origen) y para lograr este error en estado estacionario se necesita un sistema de tipo 1 (un polo en el origen). Esta fue otra razón más por la cual se necesitaba un integrador en nuestro controlador, ya que agrega

un polo en el origen transformando al sistema en tipo 1. En la figura 3.47 puede observarse el lazo de control completo del sistema.

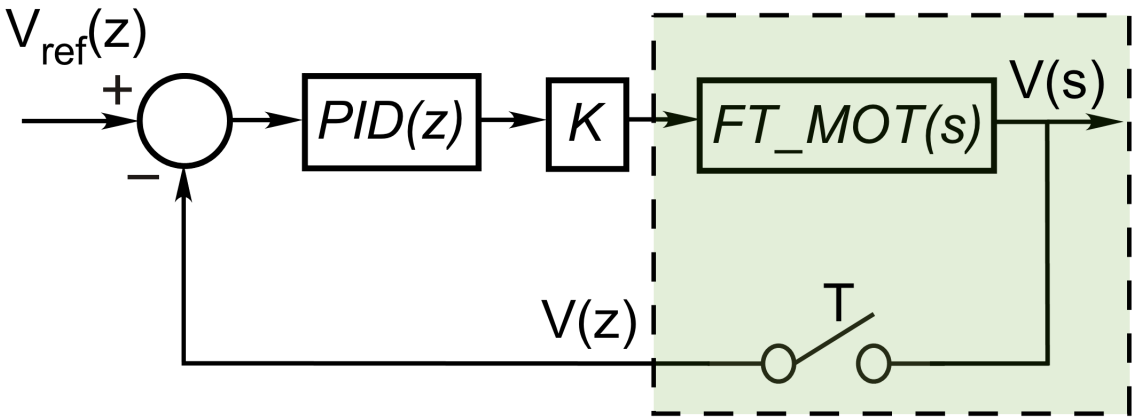


Figura 3.47: Estructura del lazo de control

Cómo puede observarse en la figura anterior, el controlador implementado es digital, y por lo tanto al interactuar con la planta hay un ida y vuelta entre tiempo continuo y tiempo discreto constante. La velocidad de traslación es calculada dentro del microcontrolador en base a los pulsos enviados por un odómetro digital, este calculo se realiza con un tiempo de muestreo $T = 500ms$, el cual es el tiempo de muestreo del sistema discreto. Se ha escogido este valor para T debido a la velocidad con que el odómetro digital entrega los datos al microcontrolador, con un T menor se correría el riesgo de tomar la velocidad antes de haber recibido un nuevo dato de distancia recorrida. Dentro del microcontrolador se compara esta velocidad con una velocidad de referencia programable expresada en metros por segundo. A partir de esta comparación se calcula la señal de error que ingresa al controlador digital. Este nos dará el cambio de velocidad que debe aplicarse a la planta. Pero debido a que los motores son controlados con una señal de PWM, debemos afectar la salida del controlador por una constante K para transformar el cambio de velocidad en un cambio del ancho de pulso de la señal PWM. Si se junta esta constante K con el modelo del motor de corriente continua, la función de transferencia de la planta total queda como sigue:

$$ft\_planta = K * \frac{9{,}468}{(s + 12{,}37)} \tag{3.7}$$

Para el cálculo de la constante K se procedió a probar el vehículo con diferentes anchos de pulso como entrada al motor midiendo la velocidad alcanzada por el mismo. De esta forma se pudo evaluar la relación entre

el incremento en el ancho de pulso y el incremento en la velocidad. Los resultados obtenidos se muestran en la siguiente tabla:

Ancho de pulso (P)	Velocidad (V)	ΔP	ΔV	K ($\Delta P/\Delta V$)
10	40 cm/s	-	-	-
18	31 cm/s	8	9	0.88
25	24 cm/s	7	7	1.0
35	15 cm/s	10	9	1.11

Cuadro 3.8: Medición de incrementos de anchos de pulso y velocidades

Cómo se puede observar en el experimento anterior el valor de K, cómo el incremento del ancho de pulso sobre el incremento de velocidad, oscila alrededor de 1, con lo cual se decidió tomar $K = 1$ lo cual hace directa la transformación entre incrementos.
Para el cálculo del controlador fue necesario obtener el modelo de la planta en forma discreta, por lo que se procedió a tomar la transformada Z del modelo con la herramienta MatLab, con un período de muestreo $T = 500ms$, obteniendo:

$$ft\_planta\_discreto = K * \frac{0{,}91656}{(z - 0{,}00206)} \tag{3.8}$$

Se configuró la herramienta MatLab para que realice la discretización del sistema con retentor de orden cero (ZOH), el cual ya está incluido en el anterior modelo.

3.3.8.3. Diseño en el lugar de las raíces y especificaciones de diseño

Las especificaciones de diseño que se han establecido se muestran en la siguiente tabla:

Condición	Tiempo de Subida	Sobre-impulso	Error de estado estacionario
Valor	$\leqslant$ 2s	0 %	0 %

Cuadro 3.9: Condiciones establecidas para el control

El sistema recibirá como entrada una velocidad de referencia constante mientras el vehículo esta en funcionamiento es por eso que se buscó un error de estado estacionario para entrada escalón igual a cero. Esto se logró, como ya se ha mencionado a través del integrador que introduce un polo en el origen. Lo que se busca en todos los aspectos del sistema es lograr que

el vehículo realice movimientos suaves, además, no se necesitan tiempos de reacción muy rápidos debido a que se trata de un vehículo en donde hablar en el orden de los segundos es suficiente, es por esto que se eligió no tener sobreimpulso y que el tiempo de subida, considerado desde el 10 % al 90 % del valor final de referencia, sea como máximo de 2 segundos.
A continuación se muestra el lugar de raíces del sistema, previo a ser compensado:

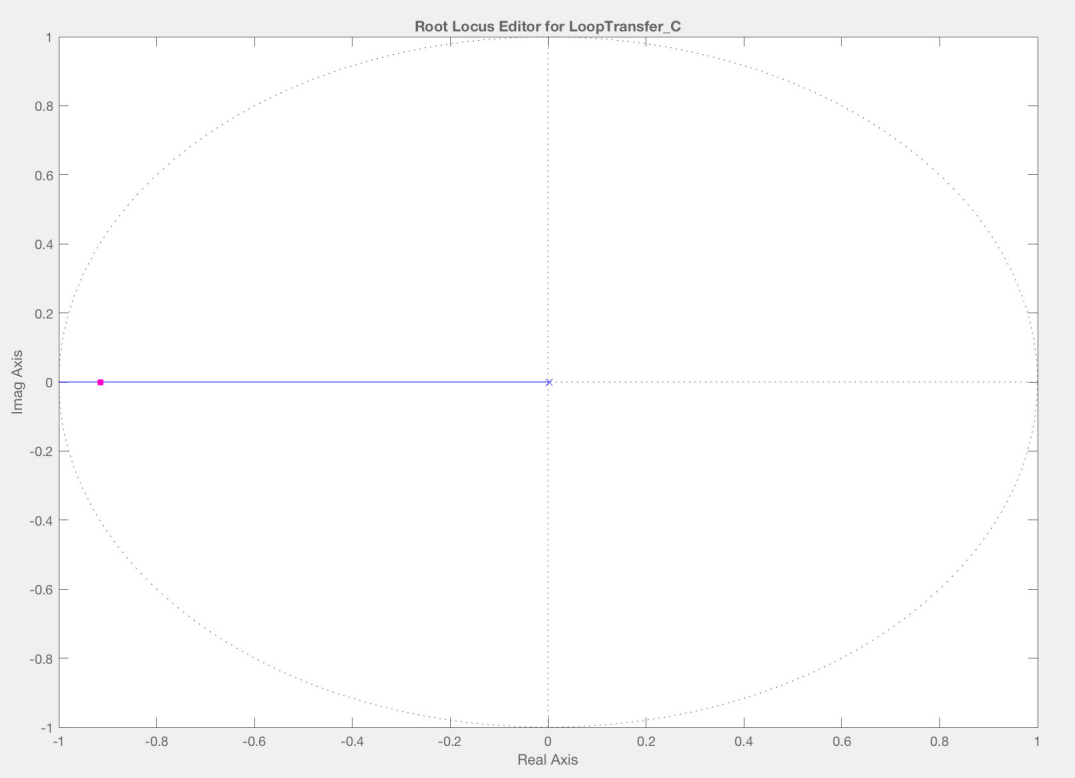


Figura 3.48: Lugar de raíces para la planta

Cálculo del compensador utilizando Diseño Asistido por Computador

Para el cálculo del compensador se utilizó el entorno gráfico de MatLab para el diseño de sistemas SISO (una entrada, una salida del inglés single input single output) en el lugar de raíces. Esta herramienta permitió modificar los parámetros del compensador hasta lograr la respuesta del sistema deseada.
En base a la ecuación A.45 de la sección A.4.3 se puede observar que el compensador PID posee un polo en el origen, otro en uno (en el límite de estabilidad) y dos ceros que fueron utilizados como parámetros de configuración. Uno de los ceros se colocó cancelando el polo de la función de transferencia del sistema no compensado, el cual se encontraba cercano al origen. El otro cero se fue ajustando hasta acercarse a la respuesta deseada,

del lado de los reales negativos se obtenían respuestas con sobrepaso, o que generaban salidas inestables, por lo que se lo colocó del lado de los reales positivos en el lugar en donde se conseguía el mejor tiempo de subida. Aún así, este tiempo no alcanzaba a cumplir los requerimientos, por lo que se ajustó la ganancia del compensador hasta lograr un tiempo de 1.5 segundos.
Finalmente la función de transferencia del compensador obtenida es la siguiente:

$$pid = \frac{0{,}77163(z - 0{,}2924)(z - 0{,}00206)}{z(z-1)} \tag{3.9}$$

Usando la herramienta MatLab para expresar la ecuación en términos de K_p, K_i y K_d:

$$pid = K_p + K_i * \frac{T * z}{z-1} + K_d * \frac{z-1}{T * z} \tag{3.10}$$

Con $K_p = 0{,}226$, $K_i = 1{,}09$, $K_d = 0{,}000232$ y $T = 0{,}5s$.
La ecuación anterior sigue la forma de la ecuación A.40 de la sección A.4.3, que consiste en la función de transferencia del PID denominada 'posicional' y que es una de las más utilizadas.

El lugar de raíces del sistema compensado quedó como sigue:

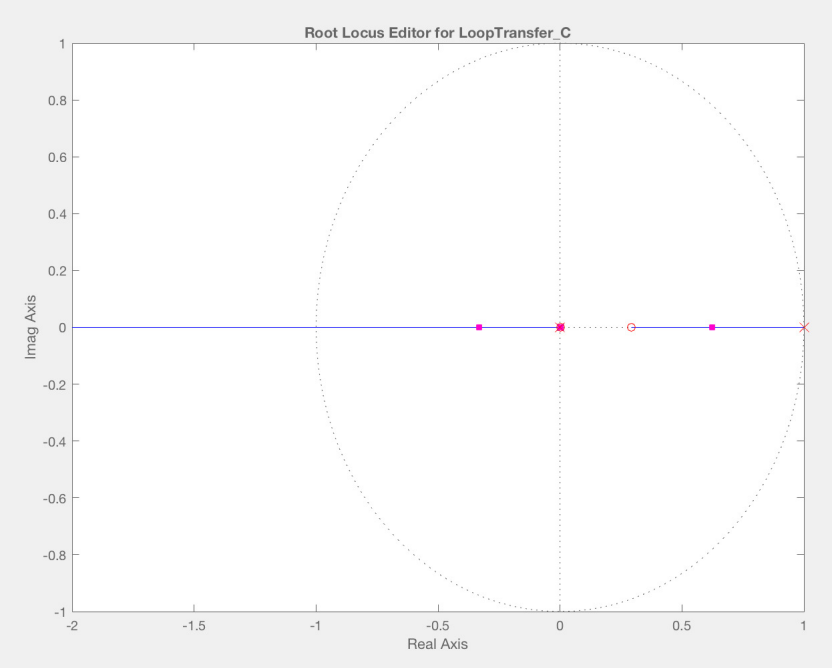


Figura 3.49: Lugar de las raíces del sistema con regulador PID aplicado

Validación del diseño obtenido

Para la validación del diseño realizado, se procede a una simulación de la

respuesta del sistema en lazo cerrado ante una entrada en forma de escalón unitario.
Se busca que la respuesta de la figura 3.50 cumpla con los requerimientos de diseño, cómo se puede observar se tiene un tiempo de subida de 1.5s, 0 % de sobrepaso, y el error en estado estacionario para esta entrada es cero, con lo cual se ha cumplido con los requerimientos que se buscaban.

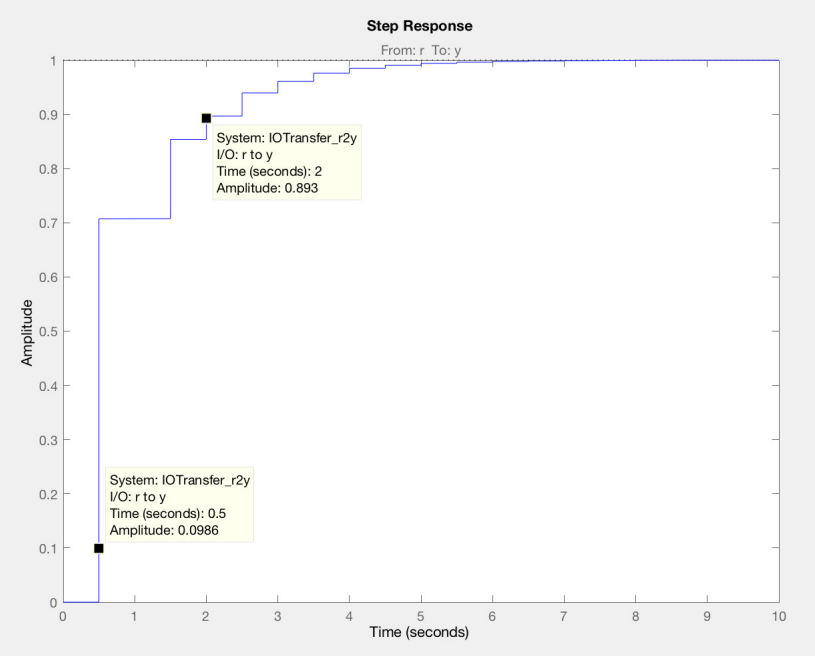


Figura 3.50: Simulación de la respuesta de velocidad del sistema compensado

3.3.8.4. Implementación del controlador PID en microcontrolador Arduino

Luego del diseño del controlador PID se procedió a su implementación en el microcontrolador.
Se presenta a continuación el diagrama de flujo correspondiente a dicha implementación:

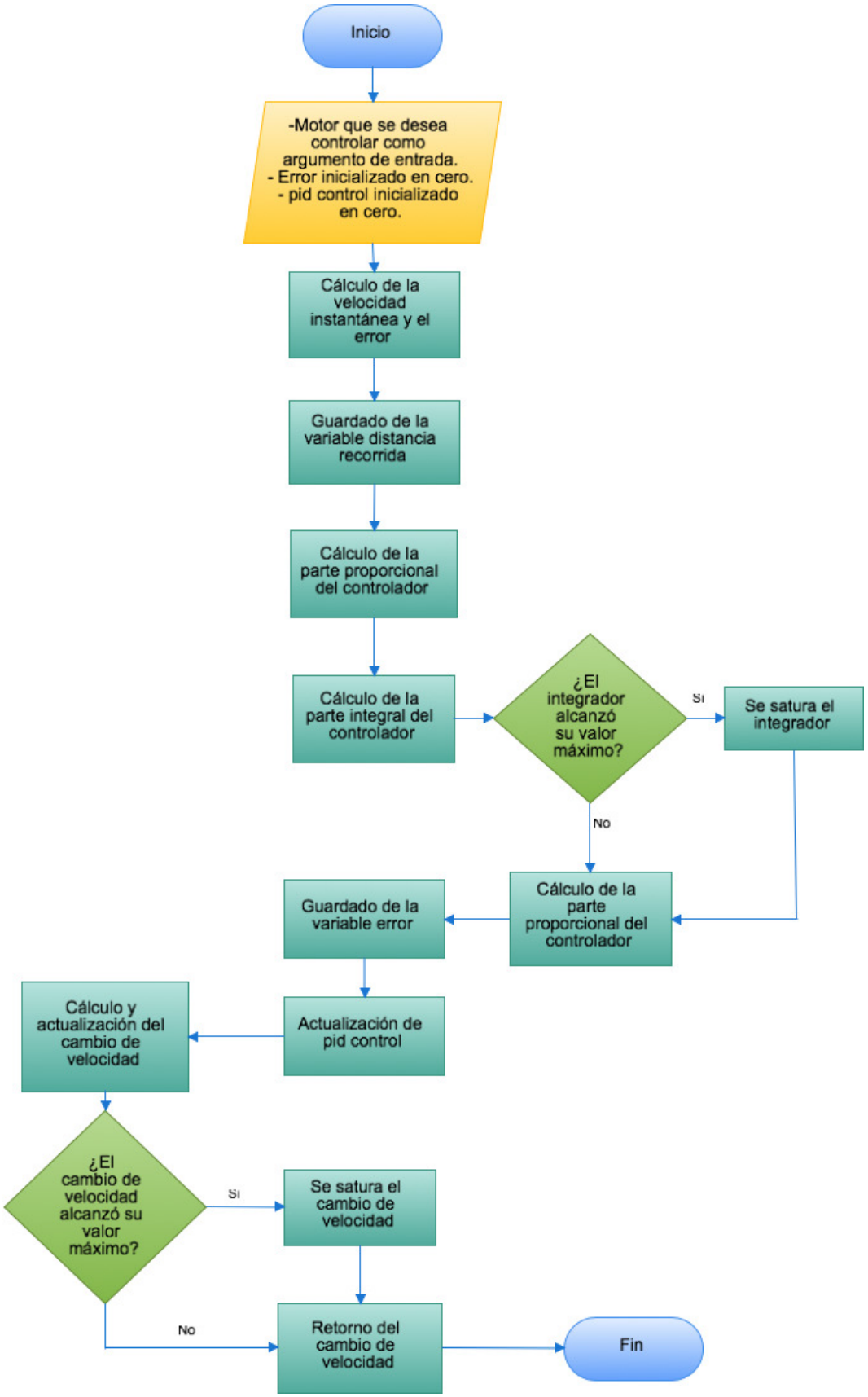


Figura 3.51: Diagrama de Flujo de la implementación del controlador PID

Cómo se puede observar en la figura 3.51 la función encargada del con-

trolador toma como argumento el motor a controlar, esto es para que, si fuera necesario, el sistema pudiera aplicar el control a más de un motor. Al inicio, se inicializan las variables de error y del controlador PID en 0.
Luego se procede a calcular la velocidad instantánea en base al dato de distancia recorrida que provee el odómetro digital y a la distancia recorrida del intervalo de tiempo anterior que se guarda en cada iteración, la velocidad resultará de la relación entre la diferencia de distancias y el intervalo o tiempo de muestro del sistema. Posteriormente a este calculo se deberá guardar la distancia recorrida que será de utilidad en la iteración siguiente. La velocidad instantánea se compara con la de referencia para el cálculo del error del lazo, el cual nos servirá para calcular cada una de las componentes del controlador.
Para la componente proporcional se afecta directamente el error por la constante proporcional obtenida, para la integral se deberá acumular el error afectado por la constante integral obtenida y por el tiempo de muestreo para cada iteración, finalmente para la parte derivativa se deberá restar el error menos el error de la iteración previa multiplicarlo por la constante derivativa y dividirlo por el tiempo de muestreo. El error luego de estos cálculos se guarda en una variable para poder usarlo en el derivador de la iteración siguiente. Este algoritmo se diseñó en base a la ecuación A.37, desarrollada en la sección anexa A.4. Si se compara esta ecuación, en el dominio temporal, con las ecuaciones A.37 y A.37, en el plano z, se puede notar que al antitransformar para volver al dominio del tiempo las constantes del controlador se mantienen y por lo tanto se pueden usar las calculadas en la sección anterior.
Luego del cálculo de la componente integral se debe controlar si la misma ha llegado a un valor máximo de 30.0, valor seleccionado luego de la evaluación del comportamiento del integrador con el vehículo en funcionamiento, y en caso afirmativo saturar esta variable. A esto último se lo llama rutina anti-wind-up por saturación del término integral. El wind-up se presenta en controladores con acción integral cuando el actuador se satura produciendo una no linealidad al sistema, y no correspondiendo la acción que se aplica al proceso con la salida del controlador. Es cuando el sistema tarda mucho en reaccionar ante una perturbación después que se produjo la saturación del actuador. La componente integral eventualmente conducirá la salida del controlador a un límite de saturación, y por eso debe ser previamente saturada a un valor máximo.
Una vez calculadas todas las componentes se suman y se obtiene el valor correspondiente al controlador propiamente dicho, este valor está calcula-

do en base a la comparación de velocidades y deberá ser afectado por una constante K para obtener el valor de cambio por el que se debe afectar el ancho de pulso del PWM que ingresa al motor y determina su velocidad. Este valor del se satura a un valor máximo para evitar llegar a los límites inferior y superior de ancho de pulso en donde el sistema no se comporta de una manera deseada.
Las variables acumulativas como lo es la componente integral, deberán ser reiniciadas a cero cuando el vehículo finaliza un movimiento, esta acción se realiza en la sección correspondiente al fin de las traslaciones.
Para finalizar esta sección se presenta el código correspondiente a la implementación:

```
double pid_controller(int motor)
{
  double error = 0;
  double valor_instantaneo = 0;
  double pid_contr = 0;

  velocidad_temp = ((distancia_temp[motor] - distancia_temp_d[motor])*0.01)
   /DELTA_T;
  error = velocidad_ref - velocidad_temp;
  distancia_temp_d[motor] = distancia_temp[motor];

  p_controller[motor]  = kp * error;
  i_controller[motor] += ki * error * DELTA_T;

  if(i_controller[motor] >= SATURACION_INTEGRADOR)
    i_controller[motor] = SATURACION_INTEGRADOR;

  d_controller[motor]  = (kd * (error - prev_error[motor])) / DELTA_T;
  prev_error[motor]    = error;
  pid_contr       = p_controller[motor] + i_controller[motor] + d_controller
   [motor];
  movimiento[motor]    += pid_contr;
  if(movimiento[motor] > 110.0)
      movimiento[motor] = 110.0;

  return movimiento[motor];
}
```

3.4. Comunicación entre unidades de procesamiento

La comunicación entre las unidades de procesamiento es un tema importante en este diseño. Problemas en la transmisión generarían consecuencias catastróficas en el vehículo pudiendo llegar a provocar accidentes en la circulación hasta su destino final. Por esto, es necesario un sistema de transmisión de datos que sea confiable y que nos permita identificar errores si los hubiese.

En el proyecto se utilizan los módulos UART de la unidad de procesamiento de alto y bajo nivel en un protocolo RS232 con niveles de voltaje TTL. En este caso se utilizarán paquetes de datos de un byte como se establece en el estándar del protocolo RS232. Si bien el protocolo de por sí tiene un método de chequeo de integridad de la información a través del bit de paridad, se desarrolló una trama de datos específica para el proyecto que hace más robusta la comunicación.
Otra funcionalidad que se desea alcanzar con esta trama de datos es la de identificar con direcciones la unidad de bajo nivel a la que se envían los datos y el hilo en la unidad de procesamiento de alto nivel que está enviando los datos. Esto permite que las respuestas por parte del microcontrolador tengan una identificación y puedan ser trasladadas al método del hilo particular que va a utilizar la información recibida.

3.4.1. Trama de datos

La trama de datos desarrollada, como cualquier paquete de datos en protocolos, consta de cabecera, datos y cola.

- Cabecera: Contiene un byte de valor conocido que indica el inicio de la transmisión entre dispositivos (Inicio), dos bytes que indican la cantidad de paquetes de datos que se van a enviar(MSB y LSB), un byte que representa la dirección del dispositivo que envía los datos(DE) y un byte con la dirección del dispositivo hacia el cual va dirigida la información(DR).
- Datos: En estos paquetes se encuentra la información útil de la trama de datos. Pueden ser comandos de movimiento, rutinas de calibración, requerimiento de información de los sensores, configuración de parámetros.
- Cola: Consiste en un byte con un valor conocido que permite determinar el final de la trama de datos y realizar el chequeo de la integridad de los datos recibidos.

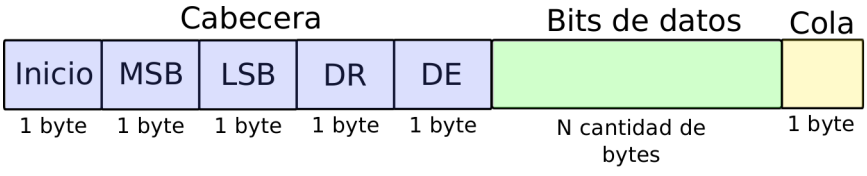


Figura 3.52: Protocolo de comunicación

La implementación del código que codifica y decodifica la información útil se realizó tanto en Python para la Raspberry como en C para el microcontrolador ATMEL. Por cuestiones de simplicidad y claridad se mostrará el código de cada uno desarrollado en Python.El código desarrollado en el microcontrolador puede observarse en C
La codificación de la información es bastante simple y consiste en realizar un arreglo de datos consistente con el protocolo desarrollado.

```
def send(self,data):
  device = 10 # Direccion del microcontrolador
  header = 160 # Secuencia de inicio
  iheader = 64 # Secuencia de stop
  data_len = len(data) # Cantidad de datos a enviar
  uid = data.pop(data_len - 1) # Direccion del hilo que envia datos
  size_l = data_len & 0xff # 8 bits MSB cantidad de datos
  size_h = (data_len & 0xff00)>>8 # 8 bits LSB cantidad de datos
  lista = [header,size_h,size_l,device,uid]+[data]+[iheader] # Trama
  for i in range(len(lista)):
    self.__ser.write(str(lista[i])) # Escribe trama en puerto serie
```

La decodificación de la información puede verse como una máquina de estados finitos que va avanzando según los datos de entrada que el puerto serie que va recibiendo. Por ejemplo, para pasar del estado cero al estado uno necesariamente tiene que llegar la secuencia de inicio que es un valor conocido, sino se mantendrá en estado cero.

```
def receive(self,method):

  # Se bloquea  el metodo  hasta que entran  datos o  vence el
  # timeout si fue seteado.
  if (self.__ser.inWaiting() > 0):
    self.__in_data = self.__in_data + [ord(n) for n in in_data_1]

  # Si no estamos procesando una trama se busca el Inicio
  if self.__flag_in_frame==False:
    indx = [i for i,x in enumerate(self.__in_data) if (x&0xE0)==160]
    if len(indx)>0:
      self.__in_data = self.__in_data[indx[0]:len(self.__in_data)]
      self.__flag_in_frame = True
    else:
      self.__in_data       = []

  # Si encontramos un MIT y no tenemos la longitud de la trama
  # aguardamos hasta tener al menos  2 octetos para extraer el
  # parametro de longitud.
  if self.__flag_in_frame==True and self.__flag_flen==False \
                                and len(self.__in_data)>=2:
      if self.__in_data[0]&0x10:
        self.__flen          =    self.__in_data[1]<<8 + \
                                  self.__in_data[2]
      else:
        self.__flen          =    self.__in_data[0]&0x0F
```

```
27          self.__flag_flen      =    True
28
29      # Una vez con  sincronismo de trama y con la  longitud de la
30      # misma se  espera el resto  de la trama  y se coloca  en la
31      # cola del usuario
32      if self.__flag_in_frame==True and self.__flag_flen==True \
33                      and len(self.__in_data)>= (self.__flen+6):
34        self.datos = self.__in_data[5:(self.__flen+5)]
35        self.uid = self.__in_data[3]
36        self.device = self.__in_data[4]
37        self.__in_data       = self.__in_data[(self.__flen+6):]
38        self.__flag_in_frame = False
39        self.__flag_flen     = False
40        self.__flen          = 0
41        method(self.uid,self.datos) # Se llama al methodo del hilo
42                                    # asociado en el Scheduler
43                                    # a la direccion obtenida
```

3.5. Hardware

3.5.1. Llave H: Placas de Potencia

Para el control de los dos motores de corriente continua que tiene el proyecto se requiere una etapa de potencia. Este etapa de potencia, en cada uno de ellos, debe permitir el cambio de sentido en la dirección de rotación de los motores y el control de la velocidad de giro a través de una modulación por ancho de pulsos. Además de estos requerimientos debe soportar la tensión y corriente que los motores necesitan para funcionar correctamente.
Para cumplir los requerimientos se decidió diseñar un circuito tipo chopper E o inversor con MOSFETs como elementos conmutadores y con el circuito integrado IR2110 como driver de los MOSFETs. La hoja de datos del circuito integrado puede encontrase en el anexo B
El circuito en cuestión puede observarse en la siguiente figura:

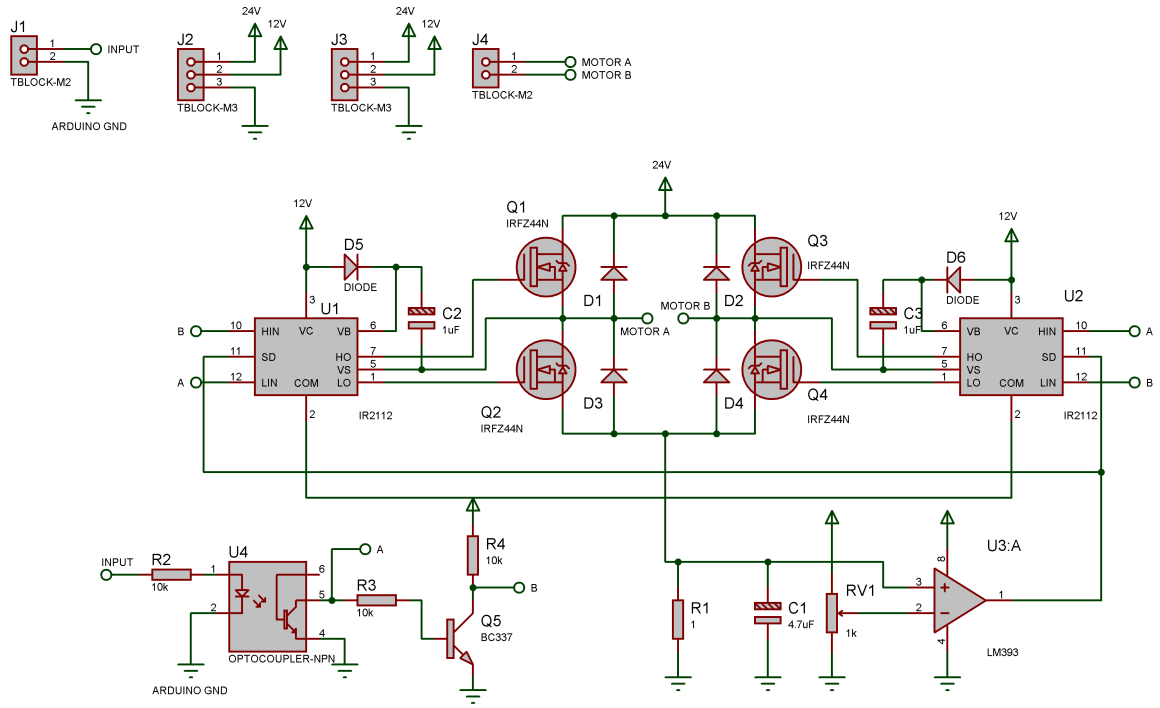


Figura 3.53: Circuito esquemático Puente H

Puede observarse que la señal PWM que proviene del microcontrolador es optoacoplada (U4) para separar el circuito de control con el de potencia como una medida de protección y además para evitar que ruido eléctrico generado por los actuadores perturbe la fuente de alimentación de control utilizada por el microcontrolador. La salida es invertida para obtener el complemento de la señal modulada por ancho de pulsos y así poder comandar los conmutadores de la misma rama con estas dos señales (por ejemplo Q1 y Q2) evitando que los dos se activen en el mismo momento.
El circuito de comando de gate de los MOSFETs es un integrado IR2110. El mismo es un driver Boostrap desarrollado en A.5.2 que permite controlar dos MOSFET de una misma rama en un puente H. La idea de este tipo de circuitos es reducir al mínimo la potencia que se disipa en el MOSFET cuando se pasa de corte a conducción.
Además, tiene un sistema de apagado que puede utilizarse para detener el motor en el caso de que exista una circulación de corriente elevada no prevista para el diseño. Para sensar la corriente que circula por el actuador se coloca un resistor de bajo valor (para que su influencia en el circuito sea despreciable) entre la parte baja del puente H y masa. La caída de tensión que existirá sobre el resistor, por Ley de Ohm, será proporcional a la corriente que circula y es utilizada como una entrada de un comparador LM393. La otra entrada a esta comparador será una tensión de referencia que puede ser ajustada según la corriente máxima que se desea que circule

por el circuito antes de que se active el sistema de apagado del integrado IR2110.

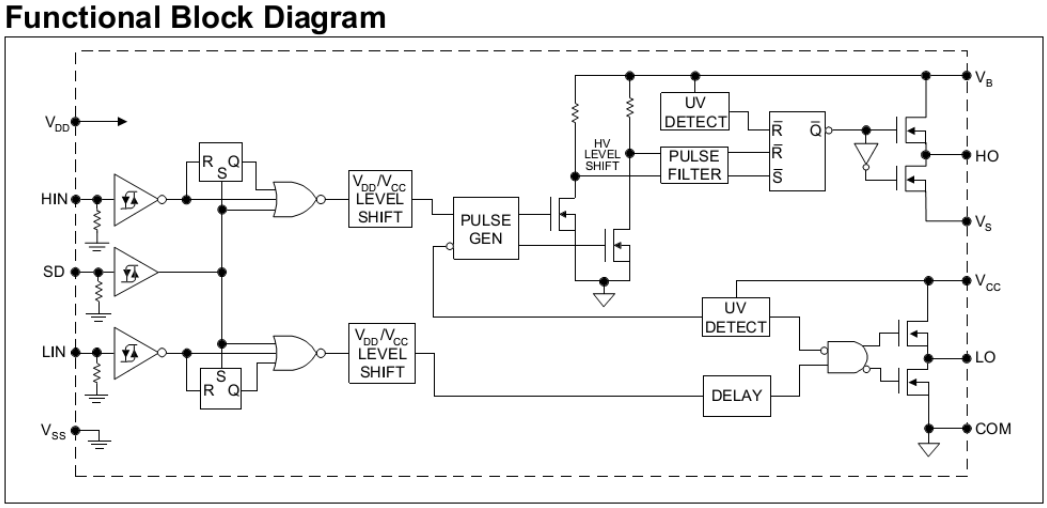


Figura 3.54: Funcionalidad IR2110

Se realizó un primer prototipo del circuito con el software de diseño PCB Proteus y se elaboró la parte práctica de la placa en la Fundación. El objetivo era comprobar el correcto funcionamiento del mismo utilizando la primer versión del vehículo autónomo cuyos actuadores eran motores de 24 Volts y una corriente estimada de 2 Amperes. Debido a estos requerimientos se seleccionó el MOSFET cuyas especificaciones más importantes son: gran capacidad de corriente (ID = 49A), tensión de operación dentro del rango necesario(VDSS = 55V) y una resistencia intrínseca de conducción baja (RDS(on) = 17.5mω). El layout del diseño puede verse en la figura 3.55.

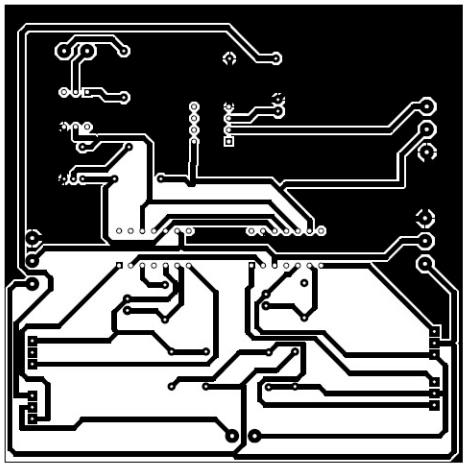


Figura 3.55: Layout Primer Prototipo

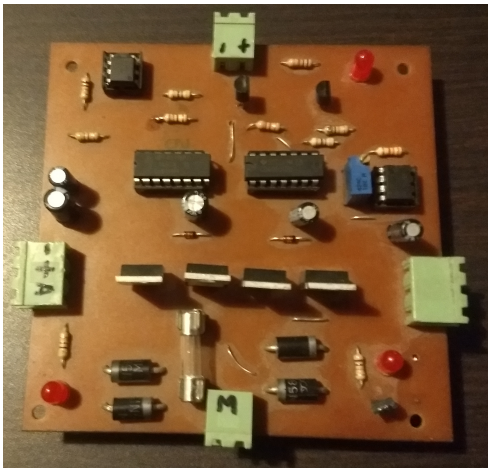


Figura 3.56: Implementación

El principal problema de este prototipo era la baja frecuencia de la modulación por ancho de pulsos que permitía el optoacoplador y la dificultad

de invertir la señal con transistores y su fuerte dependencia a las fluctuaciones de la resistencia de base para no entrar en la parte lineal de la curva del transistor. Por estos motivos, se decidió cambiar este parte del circuito por un inversor integrado 4049 que cumplía la misma funcionalidad y era más robusto. Además, ya se contaba con el segundo prototipo del vehículo autónomo que tenía como actuadores motores DC de 6 Volts y una corriente de aproximadamente 4 Amperes por lo que se reemplazaron los anteriores MOSFET por el MOSFET IRFP250 que tiene mayor capacidad de corriente. Las especificaciones del MOSFET pueden encontrarse en el anexo B. También se agregaron varios capacitores de desacople para evitar ruido en puntos sensibles del circuito.
Debido a que se tenía una certeza mayor que el circuito iba a funcionar debido a las pruebas del primer prototipo, se diseño el nuevo modelo con dos capas de cobre y se envió a realizar la placa a un especialista para que sea más confiable el diseño y con el fin de reducir espacios en el vehículo.

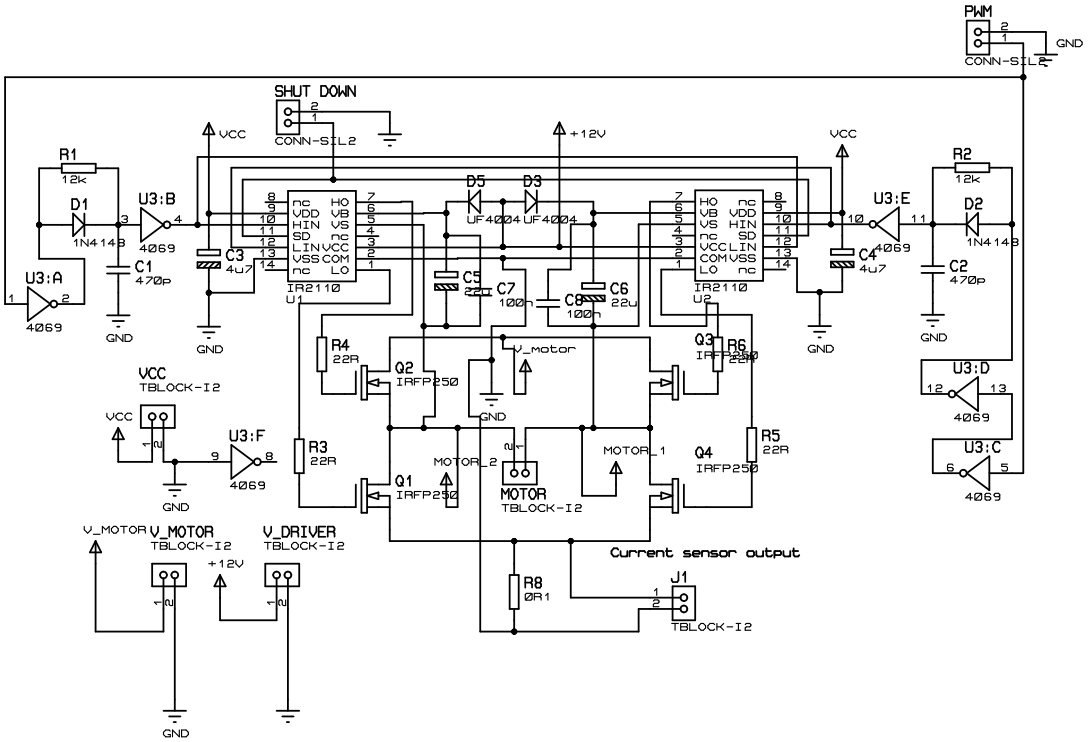


Figura 3.57: Esquemático Final

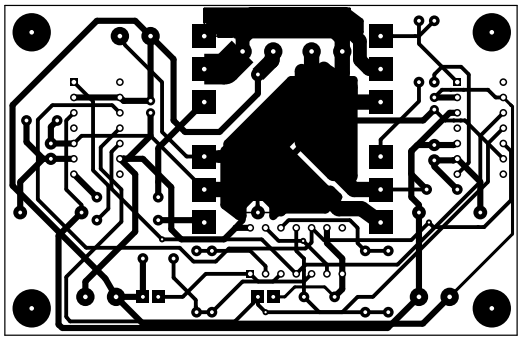


Figura 3.58: Layout Puente H

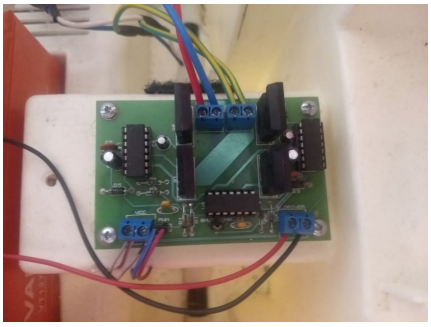


Figura 3.59: Vista superior de la placa

3.5.2. Adaptación de niveles de voltaje

Debido a los requerimientos de niveles de voltaje y corriente necesarios para alimentar el dispositivo Raspberry Pi, partiendo de la fuente de tensión de 12v que representa la batería del diseño, se diseñó un circuito electrónico capaz de suplir estas necesidades.
Los requerimientos según la hoja de datos del dispositivo son una fuente de alimentación estable de 5 v que pueda proveer 2.5 amperes de corriente. Debido a que los reguladores de tensión integrados soportan una corriente máxima de 1 amperes se diseñó un circuito específico utilizando un regulador lineal 7805 con transistores que permitan este nivel de circulación de corriente.
La resistencia R_1 proporciona la corriente de polarización para el regulador en conjunto con la corriente de base de Q_2. Si esta resistencia no se incluye la regulación se perderá para bajas corrientes de salida. El valor de la misma debe ser lo suficiente bajo como para no afectar el funcionamiento del regulador en operación normal, sin embargo, cuando la corriente máxima es requerida, la caída de tensión en la base del transistor lo hace conducir proporcionando una corriente adicional la cuál incrementa la corriente de salida máxima.

$$0 < R_1 \leqslant \frac{V_{BE1}}{I_{REG}} \tag{3.11}$$

La placa se diseñó con el software Proteus+Ares, se realizó la implementación en una placa de 5x5 cm y fue fabricada localmente por los desarrolladores del proyecto.

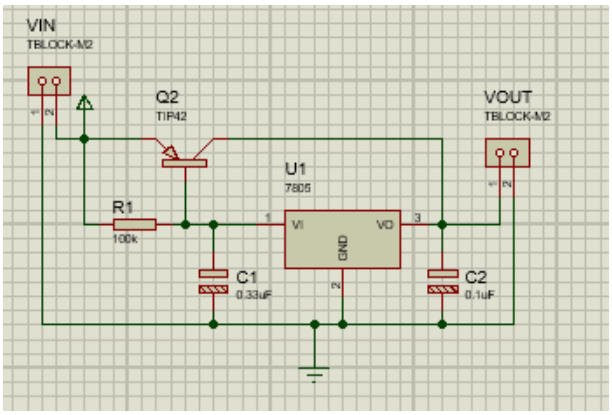


Figura 3.60: Esquemático Final

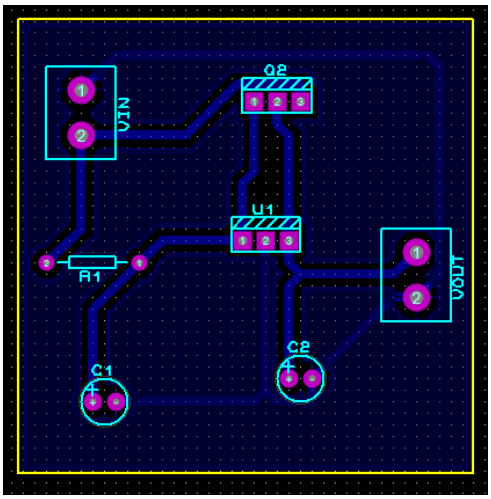


Figura 3.61: PCB Final

Resultados

3.6. Metodología de Verificación

La metodología de verificación establecida consta de múltiples pruebas del vehículo en diferentes tipos de situaciones reales en entornos variables. Se probaron dos mapas estáticos diferentes y luego, en cada uno de ellos se agregaron obstáculos dinámicos para comprobar el correcto funcionamiento del sistema de detección de objetos.
A través del sistema de monitoreo de datos del vehículo se adquiere información de todos los sensores en tiempo real mientras transcurren las pruebas y son almacenados en la ROM. Esto nos permite realizar un post-procesamiento de los datos para determinar el funcionamiento real con respecto a los comandos enviados por la unidad de procesamiento de alto nivel.
Las variables que van a ser adquiridas en tiempo real para el análisis serán:

- Velocidad lineal
- Velocidad angular de cada eje cartesiano
- Aceleración en cada eje cartesiano
- Orientación en el plano XY del vehículo
- Distancia detectada por cada sensor de proximidad

La tasa de muestreo de estas variables en las pruebas es de 100ms, lo que consideramos un tiempo prudente para obtener información fiable sin sobrecargar el sistema de comunicación con un gran volumen de transmisión de datos. Además, se filmarán todas las pruebas y se realizarán mediciones en el lugar para comprobar que la medida de los sensores se corresponde con lo que ocurre en la realidad.
En resumen, se realizarán comparaciones entre los comandos que envía la unidad de alto nivel y los datos de los sensores en tiempo real para establecer el error entre ellos y luego se determinará la desviación entre los sensores y el estado real del vehículo en el espacio de pruebas.

3.7. Pruebas

3.7.1. Mapa 1: Trayectoria sin obstáculos dinámicos

3.7.1.1. Mapa estático

La prueba se realizó en las instalaciones de la Fundación Tarpuy y el mapa que se le fue provisto a la unidad de procesamiento de alto nivel fue el siguiente:

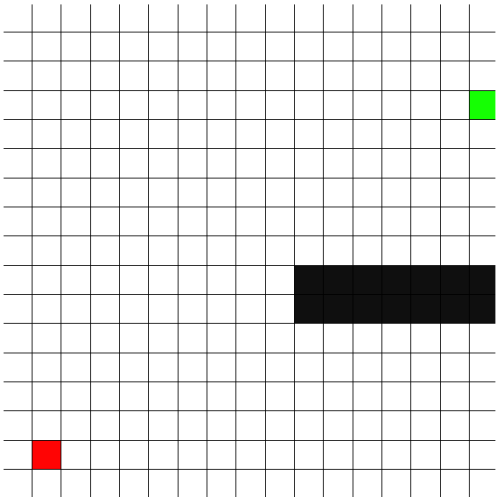


Figura 3.62: Mapa estático de la primer prueba

Puede observarse que el punto de inicio está indicado por el cuadrado verde en la esquina inferior derecha y el punto a alcanzar representado por un cuadrado rojo en la esquina superior derecha. Cada píxel representa un paso de 30 cm, se tomó esta referencia en base a las baldosas del lugar y así hacer más sencilla las medidas teniendo un patrón. Los píxeles blancos representan espacios transitables sin obstáculos y los píxeles negros representan espacios no transitables con obstáculos.El vídeo de la prueba se puede observar en `https://youtu.be/Gb66Qe2L1Ew`.

Figura 3.63: Espacio de prueba Fundación Tarpuy

3.7.1.2. Trayectoria trazada por la Unidad de Alto Nivel

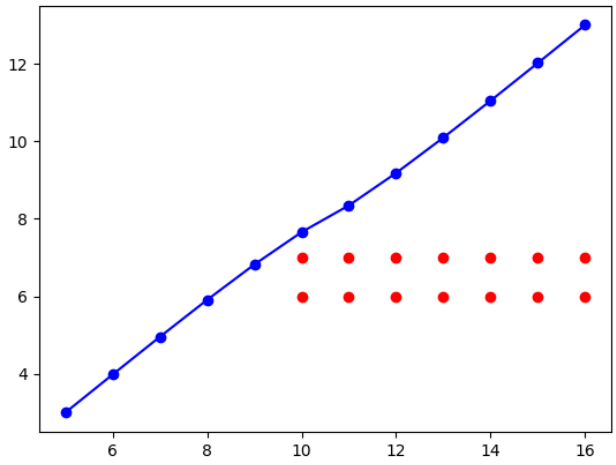


Figura 3.64: Trayectoria estática primer prueba

Observamos en el gráfico que se traza una trayectoria óptima que evita los obstáculos estáticos del mapa. Cada punto de la trayectoria azul representan las etapas que debe pasar el vehículo para llegar a su objetivo final. La conexión entre ellas es tomada como un vector representado con módulo y fase siendo el módulo la distancia que debe recorrer y la fase la orientación que debe tener el vehículo al realizar la trayectoria, esto se traduce a comandos de movimiento que envía la unidad de alto nivel a la

unidad de bajo nivel que utilizará un algoritmo similar a cross track error seguir las órdenes recibidas.
Los comandos que la unidad de alto nivel le envía a la unidad de bajo nivel son los siguientes:

1. Avanzar 36.421465 centimetros a 44.449761 grados
2. Avanzar 36.273329 centimetros a 44.210726 grados
3. Avanzar 35.916236 centimetros a 43.622034 grados
4. Avanzar 35.207930 centimetros a 42.398755 grados
5. Avanzar 33.887202 centimetros a 39.892303 grados
6. Avanzar 31.564901 centimetros a 34.543118 grados
7. Avanzar 33.887202 centimetros a 39.892305 grados
8. Avanzar 35.207932 centimetros a 42.398758 grados
9. Avanzar 35.916238 centimetros a 43.622038 grados
10. Avanzar 36.273333 centimetros a 44.210731 grados
11. Avanzar 36.421469 centimetros a 44.449768 grados

3.7.1.3. Comparación entre referencia y datos de los sensores

El gráfico 3.65 representa una comparación entre la orientación en el plano XY que envía la unidad de procesamiento de alto nivel tomada como referencia y representada por la línea roja y los datos obtenidos del sensor acelerómetro/giróscopo representados por la línea azul. En el eje y tenemos la rotación del vehículo en grados sobre el plano ortogonal del espacio donde se desenvuelve el vehículo que forman XY respecto a Z y en el eje X tenemos el tiempo en el que se desarrolla la prueba.
Puede observarse que en un comienzo el vehículo tiene una inclinación de cero grados tomada como referencia y la unidad de procesamiento de alto nivel envía un comando indicando que debe moverse 36 centímetros con una fase de 44 grados. Para lograrlo el vehículo realiza una serie de iteraciones ya que debido a las limitaciones mecánicas no puede realizar ese cambio de orientación en una distancia tan reducida. Esta iteración consiste en lo que haría un conductor real de un vehículo: girar el volante lo máximo posible hacia el lado que se quiere doblar, avanzar y si no se consigue el resultado esperado centrar el volante, retroceder y empezar desde

el comienzo. Este comportamiento explica el hecho que se observe en un principio aumentos de fase que representan cuando el vehículo avanza con el volante totalmente girado. Luego cuando el vehículo centra el volante y retrocede se observan valores de fase constantes.

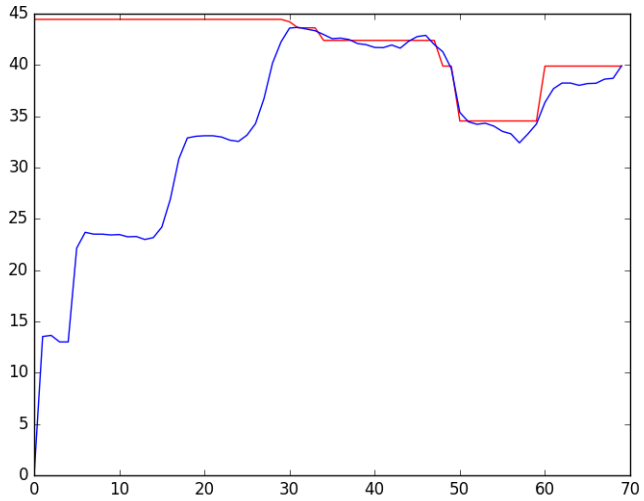


Figura 3.65: Comparación orientación plano XY

Los resultados son los esperados en cuanto a comportamiento y el error de fase es menor a ± 5 grados.
En lo referido a la velocidad lineal en el gráfico 3.66 podemos observar la referencia fijada en 30 cm/s y la velocidad obtenida por los sensores de efecto Hall. La velocidad requerida se alcanza pero al ser múltiples movimientos cortos se observa una disminución y posterior incremento en la velocidad. Debido a esto se realizará un estudio más detallado en la prueba número 3 de la evolución de la velocidad en el tiempo con una trayectoria recta sostenida en el tiempo.

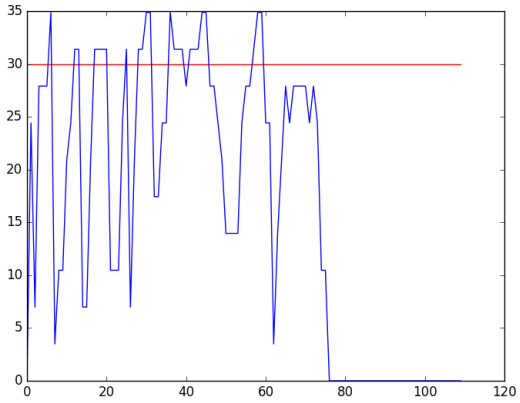


Figura 3.66: Comparación velocidad lineal

3.7.2. Mapa 1: Trayectoria con obstáculos dinámicos

Esta prueba tiene como mapa de entrada el mismo que la prueba anterior pero para comprobar el funcionamiento de la detección de obstáculos y posterior trazado de nueva trayectoria se agrego un objeto al espacio de pruebas que no existía en el mapa de entrada.El vídeo de la prueba se puede observar en `https://youtu.be/n80C4kRtEdc`.

3.7.2.1. Trayectoria trazada por la Unidad de Alto Nivel

Podemos observar en las siguientes figuras, a la izquierda la trayectoria si el vehículo no hubiera detectado obstáculos no contemplados en el mapa y a la derecha el mapa actualizado y la nueva trayectoria trazada para evitarlo.

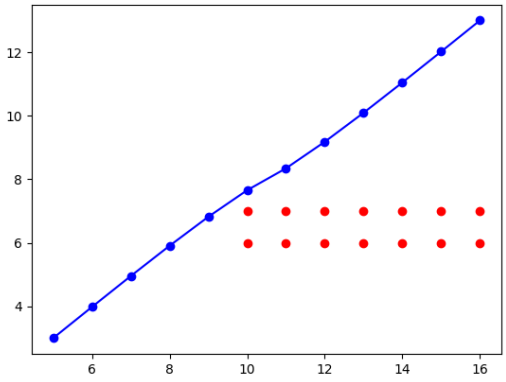


Figura 3.67: Mapa sin obstáculos dinámicos

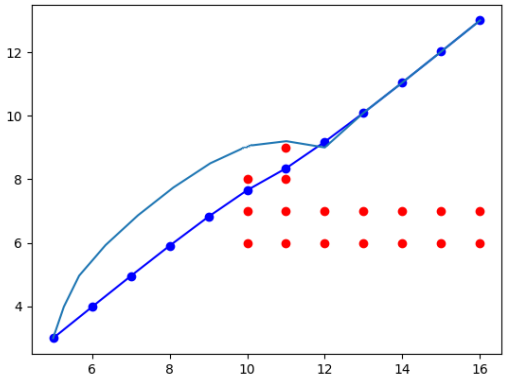


Figura 3.68: Mapa con obstáculos

Los comandos que la unidad de alto nivel le envía a la unidad de bajo nivel son los siguientes:

1. Avanzar 36.421465 centimetros a 44.449761 grados
2. Avanzar 36.273329 centimetros a 44.210726 grados
3. Avanzar 35.916236 centimetros a 43.622034 grados
4. Avanzar 35.207930 centimetros a 42.398755 grados
5. Avanzar 33.887202 centimetros a 39.892303 grados

6. Objeto detectado
7. Avanzar 81.251782 centimetros a 26.495239 grados
8. Avanzar 31.759921 centimetros a 38.699862 grados
9. Avanzar 32.868919 centimetros a 43.837029 grados
10. Avanzar 32.535084 centimetros a 48.286307 grados
11. Avanzar 30.706236 centimetros a 54.743429 grados
12. Avanzar 27.451309 centimetros a 68.066182 grados
13. Avanzar 26.611300 centimetros a 74.351892 grados

3.7.2.2. Comparación entre referencia y datos de los sensores

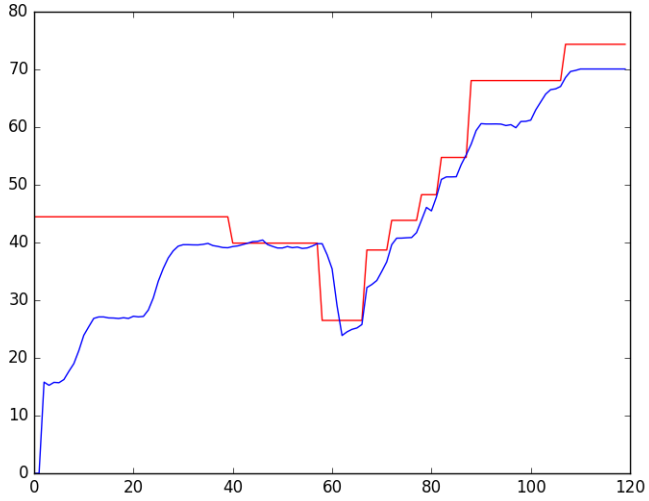


Figura 3.69: Comparación orientación plano XY

En el gráfico 3.69 puede observarse el mismo comportamiento que el obtenido en la prueba anterior, con la diferencia de que existen mayor cantidad de cambios de fase en el momento que el objeto es esquivado.
El siguiente esquema representa el valor de los sensores delanteros de proximidad en metros (trazo rojo y azul) con respecto al tiempo y la distancia de umbral que se utiliza para determinar que un objeto está lo suficientemente cerca como para representar una amenaza en la ejecución de la trayectoria. Puede notarse el punto dónde el sensor detecta un objeto cercano y procede a trazar una nueva ruta.

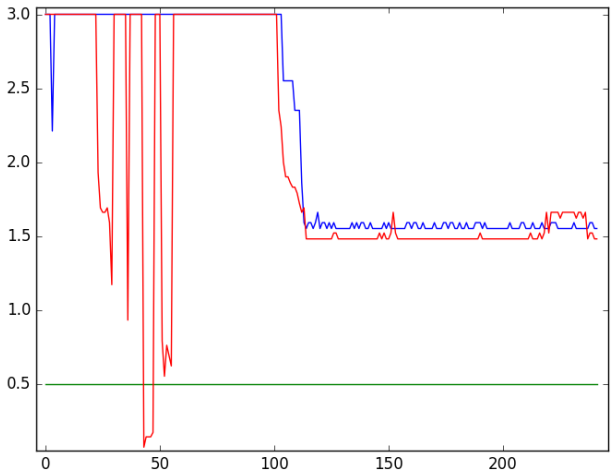


Figura 3.70: Sensores de proximidad en el tiempo

3.7.3. Mapa 2: Trayectoria sin obstáculos dinámicos

En esta prueba se utiliza un mapa de entrada diferente donde el punto final es diferente a la primer prueba, lo que provocará un trazado de trayectoria totalmente diferente con un grado de complejidad superior y se espera que los resultados tengan la misma tendencia que en la primer prueba. El vídeo de la prueba se puede observar en `https://youtu.be/QDA4yqfHBUo`.

3.7.3.1. Mapa estático

La prueba se realizó en las instalaciones de la Fundación Tarpuy y el mapa que se le fue provisto a la unidad de procesamiento de alto nivel fue el siguiente:

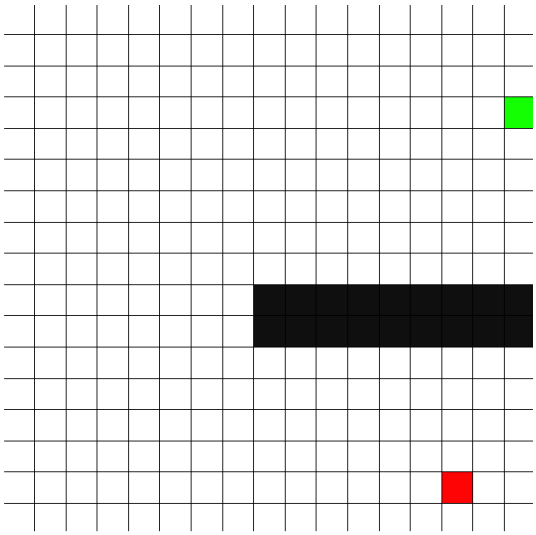


Figura 3.71: Mapa estático segunda prueba

3.7.3.2. Trayectoria trazada por la Unidad de Alto Nivel

En la figura 3.72 puede observarse la trayectoria realizada por la unidad de alto nivel. La ruta debe rodear los obstáculos en el centro del mapa para alcanzar el punto objetivo de manera óptima evitando colisiones. También puede notarse que existen cambios de fase muy abruptos en la trayectoria siendo beneficioso para probar el desempeño del vehículo real en situaciones críticas como esta.

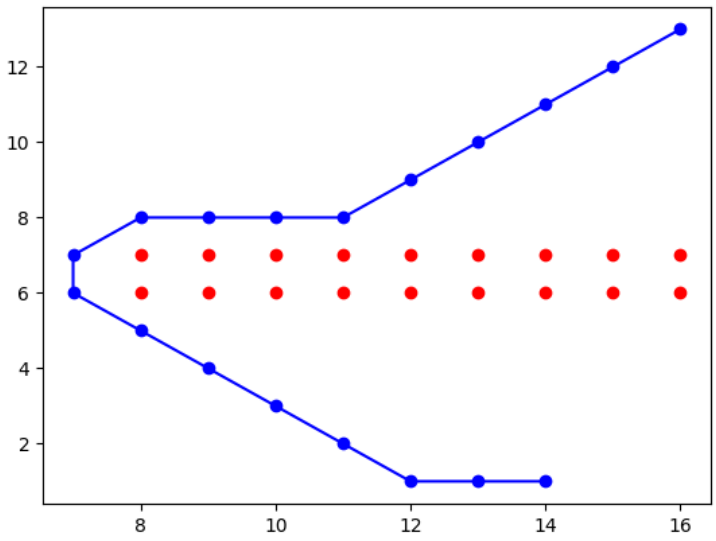


Figura 3.72: Trayectoria Prueba 2

Los comandos que la unidad de alto nivel le envía a la unidad de bajo nivel son los siguientes:

1. Avanzar 36.057362 centimetros a 44.133875 grados
2. Avanzar 161.901797 centimetros a 25.375068 grados
3. Avanzar 68.048290 centimetros a 55.816583 grados
4. Avanzar 21.644232 centimetros a 90.000000 grados
5. Avanzar 26.323355 centimetros a 117.943388 grados
6. Avanzar 30.251056 centimetros a 128.460374 grados
7. Avanzar 32.071421 centimetros a 133.845675 grados
8. Avanzar 32.114183 centimetros a 138.349327 grados
9. Avanzar 30.482342 centimetros a 144.827181 grados
10. Avanzar 27.331246 centimetros a 158.181344 grados
11. Avanzar 26.531976 centimetros a 164.454315 grados

3.7.3.3. Comparación entre referencia y datos de los sensores

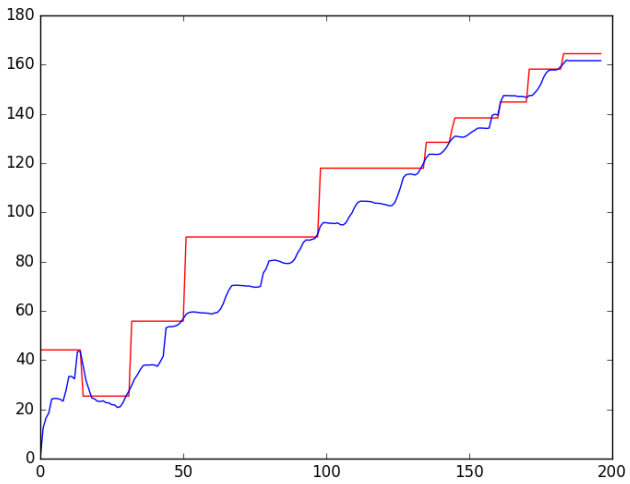


Figura 3.73: Comparación orientación plano XY

En la figura 3.73 que representa la orientación en el plano XY en grados del vehículo respecto al tiempo puede observarse la tendencia que se detectaron en las pruebas anteriores: errores de fase menores a ± 5 grados lo cual es aceptable para es diseño. Además, puede notarse la dificultad que representa para el vehículo los cambios de fase abruptos debido a las limitaciones mecánicas del sistema de rotación del eje delantero.

El gráfico 3.74 representa la distancia que de detección los sensores de proximidad en el tiempo y puede observarse que si bien se fue acercan a obstáculos en la trayectoria, ninguno estaba lo suficientemente cercano como para representar una amenaza en la ejecución de la ruta hacia el objetivo final.

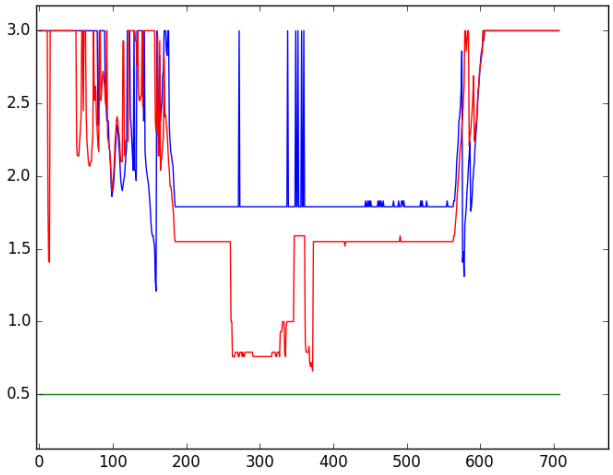


Figura 3.74: Sensores de proximidad en el tiempo

3.7.4. Mapa 2: Trayectoria con obstáculos dinámicos

Esta prueba tiene como mapa de entrada el mismo que la prueba anterior pero para comprobar el funcionamiento de la detección de obstáculos y posterior trazado de nueva trayectoria se agrego un objeto al espacio de pruebas que no existía en el mapa de entrada. El vídeo de la prueba se puede observar en `https://youtu.be/pvQ6C1bSdP0`.

3.7.4.1. Trayectoria trazada por la Unidad de Alto Nivel

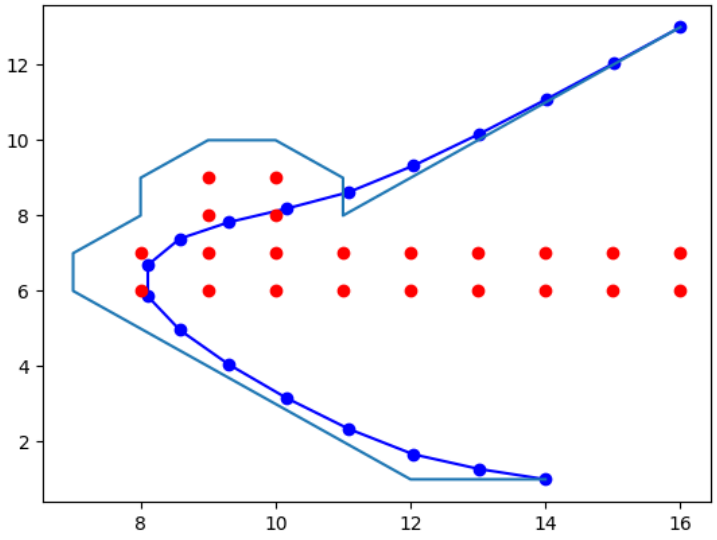


Figura 3.75: Trayectoria obstáculos Prueba 2

Puede notarse en el gráfico 3.75 con el trazo azul la trayectoria que traza al inicio de la ejecución de la ruta, la misma idéntica a la de la prueba anterior. Luego en un punto aleatorio de la ruta es detectado un obstáculo no contemplado por el mapa de entrada. La unidad de procesamiento de bajo nivel notifica a la unidad de procesamiento de alto nivel de este percance y se procede a trazar una nueva ruta (trazo cian en el gráfico) contemplando este nuevo obstáculo en la ruta. Los comandos que la unidad de alto nivel le envía a la unidad de bajo nivel son los siguientes:

1. Avanzar 36.057362 centimetros a 44.133875 grados
2. Avanzar 161.901797 centimetros a 25.375068 grados

3. Objeto Detectado
4. Avanzar 19.185701 centimetros a -51.716782 grados
5. Avanzar 19.897473 centimetros a -31.415545 grados
6. Avanzar 18.237754 centimetros a 3.190946 grados
7. Avanzar 20.590699 centimetros a 38.566915 grados
8. Avanzar 21.369183 centimetros a 62.864600 grados

9. Avanzar 47.357064 centimetros a 96.015118 grados
10. Avanzar 27.932764 centimetros a 119.258087 grados
11. Avanzar 31.141236 centimetros a 128.787051 grados
12. Avanzar 32.546680 centimetros a 133.932144 grados
13. Avanzar 32.367149 centimetros a 138.360251 grados
14. Avanzar 30.618582 centimetros a 144.804516 grados
15. Avanzar 27.406324 centimetros a 158.129034 grados
16. Avanzar 26.582303 centimetros a 164.404414 grados

3.7.4.2. Comparación entre referencia y datos de los sensores

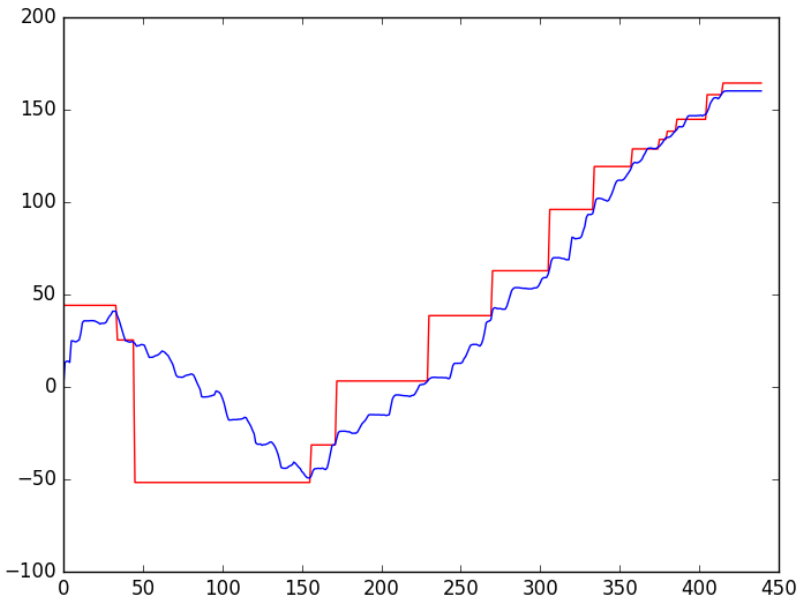


Figura 3.76: Comparación orientación plano XY

En el gráfico 3.76 vuelve a notarse la dificultad que representa para el vehículo los cambios de fase abruptos como lo indica el segmento de pasos 50-150 que a una tasa de muestreo de 100 ms representa diez segundos de demora para alcanzar la fase de referencia esperada. Esto es debido a las limitaciones mecánicas del sistema de rotación del eje delantero. Esto puede mejorarse ajustando las ganancias del algoritmo de suavizado de trayectoria pero el problema no puede eliminarse a menos que se cambie

la estructura física del vehículo.
En el gráfico 3.77 puede notarse que cerca de la muestra 50 el límite de cercanía es superado, representando el obstáculo que se detecta en la trayectoria, que pone en peligro la ejecución de la ruta esperada. En base a esto se traza la nueva trayectoria evitándolo como se observa en el primer gráfico de la prueba realizada.

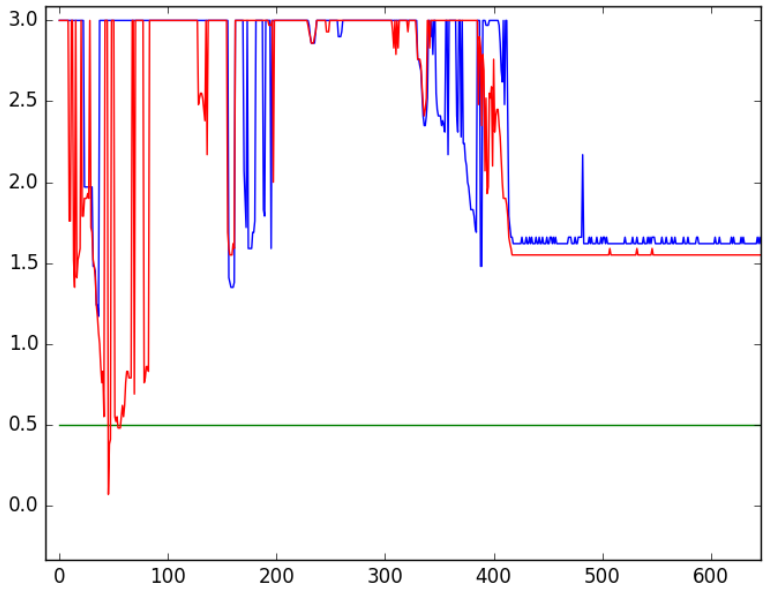


Figura 3.77: Sensores de proximidad en el tiempo

3.7.5. Evolución de la velocidad con controlador PID

En las trayectorias mostradas anteriormente, debido a que están compuestas de movimientos muy cortos, y los desplazamientos involucran procesos de iteración, la velocidad varía mucho y no puede apreciarse la acción del controlador PID implementado. Por esta razón se procedió a hacer está tercer prueba en donde el vehículo se desplaza sobre una línea recta a lo largo de 270cm a una velocidad constante de 35cm/s, siendo la velocidad máxima del vehículo de 40cm/s, se considera que la velocidad de prueba utilizada es lo suficientemente alta para nuestra estructura como para ver como se comporta el sistema de control. Las constantes del controlador utilizadas para las pruebas fueron las calculadas en 3.3.8 y se muestran en la siguiente tabla:

Constante	Valor
K_p	0.226
K_p	1.09
K_p	0.000232
T	$500ms$

Cuadro 3.10: Constantes utilizadas para el controlador pid

En la figura 3.78 se muestra la respuesta de velocidad del sistema compensado con el vehículo desarrollando la trayectoria en cuestión.

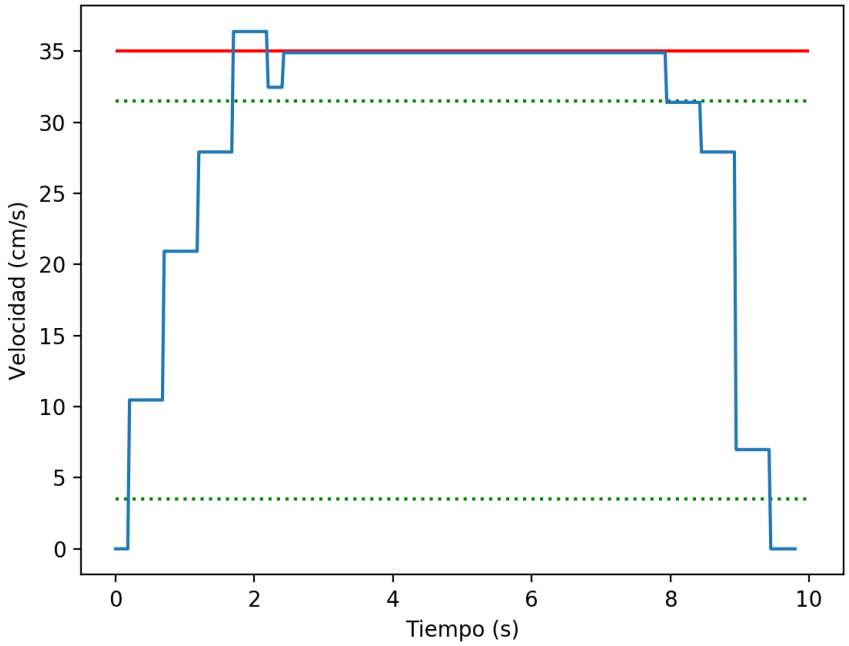


Figura 3.78: Respuesta de velocidad del vehículo en trayectoria recta

Muchas veces los resultados prácticos no reflejan lo que se ha visto u obtenido en la teoría, particularmente se esperaba tener que efectuar ajustes empíricos a las constantes del controlador para lograr un comportamiento deseado. Comparando la figura anterior con la figura 3.50, obtenida a partir de la simulación en Matlab del sistema, puede verse que si bien las respuestas no son iguales, poseen un comportamiento similar, y los resultados obtenidos se acercan mucho a los requerimientos establecidos en un principio. La simulación en Matlab posee un crecimiento abrupto al inicio para después del primer segundo crecer de manera más progresiva, por otro lado la respuesta real posee un crecimiento más parejo hasta lograr el establecimiento, pero el crecimiento es más abrupto que en la simulación, y probablemente debido a esto puede observarse un sobrepasamiento

y una oscilación antes del transitorio. Este sobreimpulso posiblemente este dado por la acción del integrador, que acumula el error y puede tardar un tiempo restablecerse una vez alcanzada la velocidad deseada. Las línea roja de la figura muestra la velocidad de referencia deseada, y las líneas verdes punteadas cuando se alcanza el 10 % y 90 % de la velocidad en transitorio, observando esto se puede notar que el tiempo de subida de la respuesta real es de 1.54 segundos. El sobrepasamiento que se presenta es del 4.27 % y el error en tiempo de establecimiento es de 0.11 %. Finalmente, se puede observar que el tiempo de subida si bien difiere del simulado cumple con el requerimiento planteado, el sobreimpulso buscado era del 0 % y si bien no se cumplió, se logró un porcentaje bajo, en cuanto al error en establecimiento si bien es de 0.11 % se lo considera aceptable para la aplicación realizada sumando a esto que se mantiene constante a lo largo de todo el transitorio.

Conclusiones

La implementación de sistemas de navegación autónoma para lograr vehículos no tripulados al servicio de distintas áreas: sectores público, privado, militar, industrial, es una temática en auge en la actualidad. No son desarrollos ya tratados, investigados y maduros, ni son desarrollos que vendrán a futuro, son desarrollos y aplicaciones actuales que se están dando en el presente a nivel mundial y en donde este es el momento justo para entrar en el rubro y no quedarse atrás. La ejecución del presente trabajo, salvando las distancias y escalas, es una prueba de las capacidades técnicas que posee la gente formada en nuestra Universidad Nacional de Córdoba, de que sí se puede, de que desde nuestra Córdoba y nuestra Argentina podemos generar y desarrollar un gran polo tecnológico que implemente tecnologías de punta en sus invenciones. Esperamos además que no sólo sea una prueba, sino también una fuente de motivación para aquellas personas que buscan incursionar en el área.

Luego de la realización del presente trabajo se logró transformar la estructura de movimientos básicos desarrollada en la PPS, en un sistema de navegación autónoma aplicado a una estructura mecánicamente estable y optimizada. El desarrollo del proyecto implicó lograr efectividad en las áreas fundamentales que deben tenerse en cuenta a la hora del desarrollo de un vehículo autónomo: localización, búsqueda de rutas, percepción, control del vehículo. Para alcanzar esta efectividad se debió diseñar y fabricar las placas para el control de potencia, gran parte de los sensores, montar todo el sistema de sensores y actuadores sobre una estructura estable, desarrollar el software necesario para efectuar todo el procesamiento de los datos y la toma y ejecución de decisiones tanto a alto como a bajo nivel, diseñar e implementar un protocolo de comunicación seguro para el envío y recepción de datos entre el microcontrolador y microprocesador. A través de la realización de lo expuesto anteriormente, se logró cumplir con cada uno de los objetivos planteados al inicio del presente proyecto. Esto sumado a los buenos resultados obtenidos convierte al proyecto en un desarrollo exitoso. Aún así, esperamos que todo el trabajo realizado no quede simplemente

en un desarrollo exitoso, sino que se espera que sirva de base para que otros estudiantes lo utilicen como plataforma para implementar sus propios desarrollos, optimicen el sistema, realicen mejoras, y porque no, que sirva de base para otros desarrollos de nivel superior tanto en el ámbito científico como en el industrial. Gracias a la plataforma desarrollada, futuras personas interesadas en el área podrán realizar al menos las primeras pruebas de sus desarrollos sobre un sistema de navegación ya estable y lo más importante sumamente adaptable a la incorporación de nuevas funcionalidades.

El presente proyecto integrador fue muy enriquecedor debido a que consistió en un viaje a lo largo de toda nuestra carrera, consistió en la implementación de los conocimientos adquiridos desde Taller y Laboratorio realizando un regulador de tensión, pasando por control con el modelado de los motores y el diseño del controlador, analógica II con el diseño de circuitos con transistores, digitales con la programación de sistemas embebidos, hasta llegar al final de la carrera, electrónica industrial con el desarrollo de las placas de potencia. Creemos entonces realmente, que conformó un excelente cierre para nuestras carreras.

En una carrera de grado el proyecto integrador es una puerta, una puerta que le permite al estudiante salir a la vida profesional. Por eso pensamos que no se trataba sólo de una mera investigación e implementación, sino que se trataba también de la profesionalidad dejada en el trabajo realizado. En base a esto se utilizaron herramientas y procesos de la industria para llevar todo el trabajo a cabo, para de esta manera acercarnos aún más a lo que puede ser el desarrollo de un proyecto en nuestra vida profesional. En este orden, se utilizó la metodología ágil de trabajo Kanban para la organización y ejecución de todas las tareas que envolvía el desarrollo. Además se realizó el control de versiones de todo el código tanto bajo como alto nivel a través del uso de repositorios git. Se busco también, que la realización del proyecto envolviera el uso de múltiples herramientas de desarrollo de software, debido a esto se buscó utilizar el lenguaje C++ para bajo nivel, Python para alto nivel, y Matlab para la simulación de los sistemas.

En el transcurso de este proyecto integrador se fueron detectando posibles mejoras a ser realizadas a futuro así cómo también el agregado de nuevas funcionalidades. Esto nos parece una oportunidad interesante para el desarrollo de trabajos futuros por parte de otros estudiantes, sería una satisfacción muy grande poder ver nuestro proyecto ayudando en los proyectos de otras personas. Las posibles mejoras y nuevas funcionalidades consisten en:

- Implementación de visión estéreo para la detección de obstáculos a través del procesamiento de imágenes de manera tal de poder determinar la posición exacta del obstáculo con respecto al vehículo.
- Implementación de sistema de detección de señales de tránsito con redes neuronales convolucionales, procesando las imágenes obtenidas por una cámara. El sistema debería poder actuar en base a la señal detectada.
- Utilización de odómetros o encoders rotacionales de mayor precisión que los utilizados para el cálculo de las distancias recorridas.
- Implementación de tecnología GPS para tener una referencia absoluta de la posición del vehículo.
- Medición de los grados rotados por el volante a través de un acelerómetro.
- Optimización del sistema de iteración para rotar ángulos grandes. Se debería doblar el volante el sentido puesto al hacer marcha atrás para ganar ángulos en ese movimiento y disminuir el número de iteraciones.
- Implementación de filtro de Kalman en reemplazo del filtro pasabajos para los datos obtenidos del acelerómetro.
- Implementación del sistema de navegación autónoma en otras plataformas para la comprobación de su escalabilidad.

Haciendo referencia al último punto. Un objetivo importante de en nuestro desarrollo era que el sistema desarrollado fuera escalable. Esto es, si bien el desarrollo del presente proyecto integrador se hace sobre una estructura determinada, que a futuro el sistema fuera lo suficientemente escalable como para poder ser implementado en otras plataformas diferentes sin hacer cambios significativos sobre el mismo. Sólo como comentario anecdótico, ya se ha probado el sistema en un robot Mindstorm de la marca LEGO, logrando el desplazamiento autónomo del mismo, además se ha implementado el sistema sobre un Dron, logrando que el mismo se desplace de forma autónoma en dos dimensiones, desplazándose a una altura constante. Gracias a esto se pudo comprobar la escalabilidad y estabilidad del sistema desarrollado.
En conclusión, creemos que el presente proyecto integrador fue una excelente manera de consolidar todos los conocimientos vistos a lo largo de nuestra carrera y que junto con prácticas actuales cientifico-industriales

conformó la puerta perfecta que nos lleva a nuestra vida como profesionales. Esperamos que la puerta quede abierta y poder seguir trabajando sobre este proyecto en conjunto con otra gente hasta un día poder verlo, quién dice, montado en la calle sobre un vehículo autónomo.

Bibliografía

[1] Aníbal Ollero Baturone, *Robótica: manipuladores y robots móviles*, Editorial Marcombo, Capítulo 1, **2005**.

[2] Bazaraa, M.S., J.J. Jarvis, *Programación lineal y flujo en redes*, segunda edición, Limusa, México, DF, 2004.

[3] Erik Cuevas, Daniel Zaldivar, Marco Perez Cisneros, *Procesamiento digital de imágenes con MatLAB y SIMULINK*, México **2010**.

[4] GONZALEZ, Rafael; WOODS, Richard., *Digital Imagen Processing.* **Pearson-Prentice Hall. 3ra edición. 2008**.

[5] González, R.C., Wintz, P., *Procesamiento digital de imágenes*, **1996**.

[6] Russell Norving, *Artificial Intelligence. A modern Approach*, 3ra edición, **2009**.

[7] M. Tim Jones, *Artificial Intelligence. A Systems Approach*, Computer Science Series, **2008**.

[8] Upamanyu Madhow, *Fundamentals of Digital Communication*, CAMBRIDGE UNIVERSITY PRESS, University of California, Santa Barbara, **2008**.

Referencias

[9] Filip Hucko, *The development of autonomous vehicles*, **Junio 2017**.

[10] Hernández Lara Diana, Nava Hernández Dulce Tania, Savage Chávez Rodrigo, *ROBOTCODE*, Capítulo 1, http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/4274/Tesis.pdf?sequence=1, **MÉXICO D. F. 2011**.

[11] *Ackermann Illustrations*, https://commons.wikimedia.org/wiki/File:Ackermann.svg, **Marzo 2016**

[12] Sociedad de Ingenieros en Automoción, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, https://www.sae.org/standards/content/j3016_201609/, **Septiembre 2016**.

[13] 5 hertz, *Acelerómetro*, https://www.5hertz.com/index.php?route=tutoriales/tutorial&tutorial_id=2, **Marzo 2015**.

[14] 20 Minutos, *Los seis niveles de clasificación de los coches autónomos*, https://www.20minutos.es/noticia/2825372/0/clasificiacion-coches-autonomos/, **Marzo 2018**.

[15] Slide Player, *Localización y Planificación de Trayectorias*, https://slideplayer.es/slide/8927367/, **Junio 2018**.

[16] Miguel Ángel Fernández Lancha, David Fernández Sanz, Carlos Valmaseda Plasencia, *Planificación de trayectorias para un robot móvil*

[17] Dra. Ana M. Cruz Martín, *Planificación de Trayectorias en Sistemas Multirrobot*, **Málaga, Diciembre de 2004**.

[18] Alicia Beriain Gil, *Matemáticas en un navegador GPS: algoritmos de camino más corto y cálculo de posición*,**2016**.

[19] L. Enrique Sucar (INAOE) , *Métodos de Inteligencia Artificial.*

[20] Zenon Cucho M. Freri Orihuela Q. Rolando Sánchez P. Laureano Rodríguez P, *Microcontroladores*, **2007**.

[21] Pichucho Jorge Aníbal, *MÓDULO PARA VERIFICAR EL FUNCIONAMIENTO DE LOS PROGRAMAS GRABADOS EN EL PIC 16F84A* , Quito, **Marzo 2007**.

[22] Luis Felipe Palma Pozo, Stalin Eduardo Sangopanta Ardila, *Diseño de un PLC con Microcontrolador PIC16F873.*, Pinar del Río, **2011**.

[23] Sherlin Xbot, *Arquitectura de microcontroladores*, `http://sherlin.xbot.es/microcontroladores/introduccion-a-los-microcontroladores/arquitectura-de-microcontroladores`, **Junio 2018**.

[24] TEM Electronic Components, *PIC18F458*, `https://www.tme.eu/en/details/pic18f458-i_pt/8-bit-pic-family/microchip-technology/`, **Junio 2018**.

[25] RobotShop, *Arduino Mega 2560*, `https://www.robotshop.com/de/de/arduino-mega-2560-mikrocontroller-rev3.html`, **Junio 2018**.

[26] *Diagreama de bloques PIC 16F877A*, `https://www.mikroe.com`, **Mayo 2013**.

[27] SCRIBD, *Partes que integran los microprocesadores*, Lozano Larreta Karla Azucena, `https://es.scribd.com/doc/105052981/Partes-Que-Integran-Los-Microprocesadores`, **Julio 2018**.

[28] Partes de, *Partes del Microprocesador*, `http://partesde.com/microprocesador/`, **Julio 2018**.

[29] SCRIBD, *Partes de un Microprocesador*, Fundación Universitaria Cofrem-Unipanamericana, Arquitectura de Hardware, Ing. Aireth Amaya, Villavicencio-meta 2015, `https://es.scribd.com/document/289159794/Partes-Microprocesador`, **Julio 2018**.

[30] Arquitectura de Computadoras, *Comparativa X86/ARM*, Miércoles 14 de enero de 2015, `http://arquitecturadecomputadoresd4.blogspot.com/2015/01/x86arm.html`, **Julio 2018**.

[31] Electronilab, *Cubieboard A20*, https://electronilab.co/tienda/cubieboard-a20/, **Junio 2018**.

[32] Amazon, *Raspberry Pi 3*, https://www.amazon.com/Raspberry-Pi-RASPBERRYPI3-MODB-1GB-Model-Motherboard/dp/B01CD5VC92, **Junio 2018**.

[33] EcuRed Conocimiento para todos, *Motor Eléctrico*, https://www.ecured.cu/Motor_el%C3%A9ctrico, **Mayo 2018**.

[34] El motor eléctrico, *Motor Eléctrico*, http://elmotorelectrico123.blogspot.com.ar/2015/11/partes-del-motor-electrico.html, **Mayo 2018**.

[35] Aficionados a la mecánica, *Coche Eléctrico*, http://www.aficionadosalamecanica.com/coche-electrico_control.htm, **Julio 2018**.

[36] Makers, *Mover motores paso a paso con arduino*, http://diymakers.es/mover-motores-paso-paso-con-arduino/, **Julio 2018**.

[37] Escuela Politécnica Superior, *Diseño y programación de un robot humanoide de bajo coste*, https://rua.ua.es/dspace/bitstream/10045/57291/1/Programacion_de_tareas_de_un_robot_humanoide_de_ba_Hernandez_Ramirez_Oscar.pdf, **Julio 2018**.

[38] Ing. José Carlos López Arenales, *Motores Eléctricos, Proyectos de Ingeniería Mecánica*, http://biblio3.url.edu.gt/Libros/2013/ing/pim/12.pdf, **Mayo 2018**.

[39] Instituto Europeo del Cobre, *Física - Motores eléctricos*, http://www.copperalliance.es/educacion/programas-educativos/fisica-motores-electricos, **Mayo 2018**.

[40] CLR Compañía Levantina de Reductores, *Motores de Corriente Continua y Alterna*, https://clr.es/blog/es/motores-corriente-continua-alterna-seleccion/, **Mayo 2018**.

[41] Bertomeru José, García José, Gardonio Sergio, Gomez Marcelo, Granados Fernando, *Máquinas e Instalaciones Eléctricas, Motor Paso a Paso*, http://www1.frm.utn.edu.ar/mielectricas/docs2/PaP/MOTOR_PaP_FINAL.pdf, **Mayo 2018**.

[42] Francisco A. Candelas Herías, Juan A. Corrales Ramón, *Servomotores*, Publicación Interna 9, 20-09-2007, http://www.aurova.ua.es/previo/dpi2005/docs/publicaciones/pub09-ServoMotores/servos.pdf, **Mayo 2018**.

[43] Eduardo Interiano, **Instituto Tecnológico de Costa Rica**, *Control Digital de Velocidad de un Motor CD*, Cartago, febrero 2015, https://issuu.com/eduardointeriano/docs/controldigitaldeunservodevelocidad.

[44] Lic. Katya Pérez M., Ing Carla Escobar O., *Modelación en Variables de Estado*, Capítulo 8, http://virtual.usalesiana.edu.bo/web/contenido/dossier/22011/700.pdf, **Junio 2018**.

[45] Maximiliano D. Armesto, Leandro E. Borgnino, *Control de velocidad de un motor de corriente continua*, Universidad Nacional de Córdoba, Trabajo Final Sistemas de Control I, **10 de Diciembre de 2014**.

[46] Wikipedia, *Controlador PID*, https://es.wikipedia.org/wiki/Controlador_PID, **Junio 2018**.

[47] Ing. Sergio Laboret, *Análisis en el Dominio de la Transformada Z, Capítulo 3*, Universidad Nacional de Córdoba, Sistemas de Control II, **8 de Junio de 2017**.

[48] José Carlos Villajulca, *Variadores de Velocidad de Motores DC: Fundamentos*, 23 de enero de 2010, https://instrumentacionycontrol.net/variadores-de-velocidad-de-motores-dc-fundamentos/, **Julio 2018**.

[49] Ing. Adrián Agüero, *Drivers para Motores de Corriente Continua*, Universidad Nacional de Córdoba, Cátedra de Electrónica Industrial, **Julio 2018**.

[50] Ing. Sergio Laboret, *MOSFET para conmutación de potencia*, Capítulo 1, Bibliografía Electrónica Industrial, **28 de Noviembre de 2010**.

[51] Maximiliano D. Armesto, Leandro E. Borgnino, *Introducción a los Microcontroladores*, Fundación Tarpuy, Programa Ingenia, **24 de febrero de 2016**.

[52] Wikipedia, *Proximity Sensor*, https://en.wikipedia.org/wiki/Proximity_sensor, **Julio 2018**.

[53] BALLUFF sensors worldwide, *Sensores de desplazamiento y distancia*, http://www.nortecnica.com.ar/pdf/teoria_sens_desp_dist.pdf, **Julio 2018**.

[54] Mescorza, *Detectores Ópticos*, http://www.mescorza.com/neumatica/sensoresweb/sensores/optico1.htm, **Junio 2018**.

[55] Diego Pérez de Diego, *Sensores de Distancia por Ultrasonidos*, http://picmania.garcia-cuervo.net/recursos/redpictutorials/sensores/sensores_de_distancias_con_ultrasonidos.pdf, **Julio 2018**.

[56] Bang good, *HC SR-04*, https://www.banggood.com/th/Wholesale-Geekcreit-Ultrasonic-Module-HC-SR04-Distance-\Measuring-Ranging-Transducer-Sensor-DC-5V-2-450cm-p-40313.html?cur_warehouse=USA, **Junio 2018**.

[57] José Miguel Guerrero Hernández, Gonzalo Pajares Martinsanz, María Guijarro Mata-García, *Técnicas de Procesamiento de Imágenes Estereoscópicas*, Departamento de Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid, http://www.cesfelipesegundo.com/revista/articulos2011/Guerrero,%20J.M.pdf, **Julio 2018**.

[58] Torres Toro Víctor Ricardo, *Diseño y construcción de un odómetro digital*, Ecuador, Quito, Noviembre del 2008, http://bibdigital.epn.edu.ec/bitstream/15000/1290/1/CD-2665.pdf, **Julio 2018**.

[59] Naylamp Mechatronics, *Tutorial MPU6050, Acelerómetro y Giroscopio*, https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html, **Junio 2018**.

[60] Prometec, *Usando el MPU6050*, https://www.prometec.net/usando-el-mpu6050/, **Junio 2018**.

[61] Pozo D. ; Sotomayor N. ; Rosero J.;Morales L., *Medición de Ángulos de Inclinación por Medio de Fusión Sensorial Aplicando Filtro de Kalman.*

[62] Mikael Larsson, *Evaluation of Serial Communication Protocols for Integrated Circuits*, Saab Bofors Dynamics AB, 2da edición **2005**.

[63] Pradosh Priyadarshan, *SERIAL COMMUNICATION BY USING UART*.

[64] EET N 460 "Guillermo Lehmann", *Reguladores Lineales*, **Junio 2011**.

Apéndice A

Complemento Teórico

A.1. Microcontrolador

A.1.1. Arquitectura del Microcontrolador

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su arquitectura las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria principal y unidad de entrada/salida.

- Memoria principal: Es un dispositivo de almacenamiento de información dividido en celdas que se identifican mediante direcciones. Las celdas son todas iguales, del mismo tamaño o número de bits, y contienen tanto datos como instrucciones. Los programas que se realicen serán almacenados temporalmente en la memoria, para después ser ejecutados. Los datos que utilice el programa deberán estar también en la memoria, y los datos que se produzcan se almacenarán también en ella. La instrucción siguiente que debe ejecutar la computadora se determina en base a un registro especial de la CPU denominado contador de programa. Este registro contiene en cada momento la dirección en memoria principal de la siguiente instrucción que debe ser ejecutada.
- Unidad Central de Proceso (CPU): Es la parte encargada del procesamiento de las instrucciones y quien realmente gobierna y ejecuta todos los cálculos. Está compuesta a su vez de dos unidades funcionales, la unidad aritmético-analógica y la unidad de control. Además contiene varios registros para el control del estado del microcontrolador. Uno

de estos registros es el contador de programa.
La unidad aritmético-lógica es la encargada de realizar las operaciones aritméticas y lógicas, entre datos provenientes de la memoria, tales como suma, resta, OR, AND, entre otras.
La unidad de control se encarga de leer las instrucciones máquina almacenadas en la memoria principal, decodificarlas, y generar las señales de control de bajo nivel necesarias para que cada instrucción sea ejecutada.

- Unidad de entrada/salida: Realiza la transferencia de información con las unidades exteriores del microcontrolador, llamadas periféricos. Las líneas de E/S que se adaptan con los periféricos manejan información en paralelo y se agrupan, en general, en conjuntos de ocho. Hay modelos con líneas que soportan la comunicación en serie; otros disponen de conjuntos de líneas que implementan comunicaciones por diversos protocolos, como el I2C, el USB, RS232.

La arquitectura de un microcontrolador permite definir la estructura de su funcionamiento, las dos arquitecturas principales usadas en la fabricación de microcontroladores son: arquitectura de Von Neumann y arquitectura Harvard. Además, estas arquitecturas pueden tener procesadores de tipo CISC o de tipo RISC.

- Arquitectura Harvard: En la arquitectura Harvard existe una memoria específica para instrucciones y datos. En este caso la memoria de programa tiene su bus de direcciones, y su propio bus de datos y su bus de control. Por otra parte, la memoria de datos tiene sus propios buses de direcciones, datos de control, independientes de los buses del programa. La memoria de programa es solo de lectura, mientras que en los datos se puede leer y escribir.

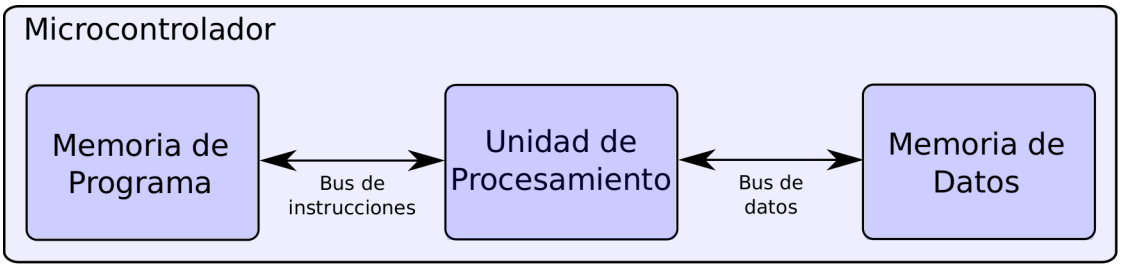


Figura A.1: Arquitectura Harvard

- Arquitectura Von Neumann: En esta arquitectura, los datos y las instrucciones circulan por el mismo bus ya que estos son guardados en la

misma memoria, su principal ventaja es el ahorro de líneas de entrada-salida pero esto supone una disminución en la velocidad con la que se realizan los procesos. ES decir, es imposible manipular simultáneamente datos e instrucciones, debido a la estructura de buses únicos, algo que si es posible con la arquitectura Harvard, que tiene buses separados. Esto confiere a la arquitectura Harvard la ventaja de mayor velocidad de ejecución de los programas.

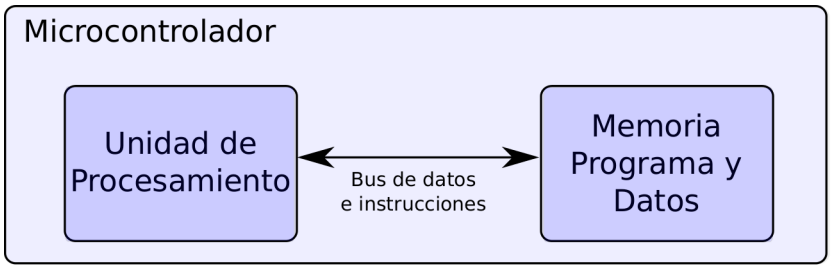


Figura A.2: Arquitectura Von Neumann

A.1.2. Componentes especiales del Microcontrolador

Además de los componentes principales, anteriormente desarrollados, los microcontroladores cuentan con componentes especiales que brindan mayor versatilidad al mismo en caso de ser aplicado en diseños más complejos. Estos componentes especiales, tanto su existencia en la arquitectura como sus características específicas, dependen del modelo de microcontrolador y de su fabricante. Un ejemplo de estos módulos especiales integrados en la arquitectura del microcontrolador puede verse en la figura A.3.

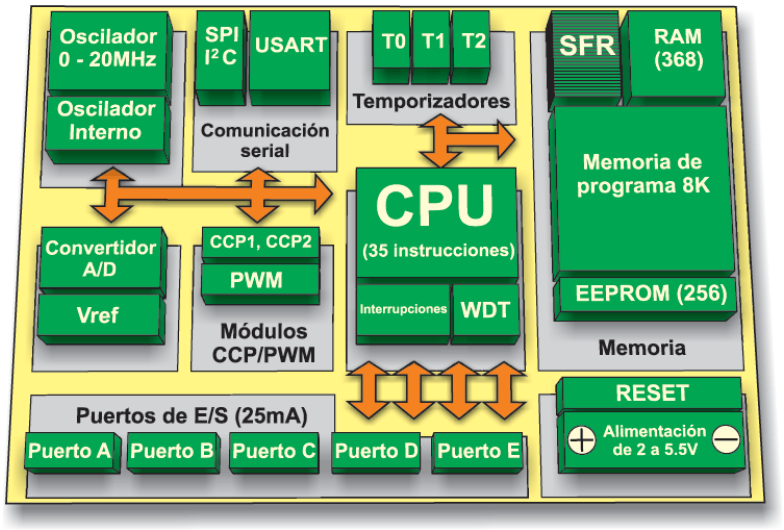


Figura A.3: Ejemplo arquitectura microcontrolador[23]

A continuación se desarrollarán los módulos más importantes:

- Módulo PWM: la modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga. En robótica, es extremadamente útil para controlar la velocidad de motores DC. Algunos microcontroladores permiten esta modulación por hardware a través de registros de configuración para determinar el período y el ciclo de trabajo de la señal modulada.

- Conversor analógico-digital/digital-analógico: el objetivo básico de un ADC es transformar una señal eléctrica analógica en un número digital equivalente. De la misma forma, un DAC transforma un número digital en una señal eléctrica analógica. Esta función exige que los pasos intermedios se realicen de forma óptima para no perder información. Según el tipo de componente y su aplicación existen distintos parámetros que lo caracterizan, éstos pueden ser: la velocidad de conversión, la resolución, los rangos de entrada. Por ejemplo, una mayor cantidad de bit, implica mayor precisión, pero también mayor complejidad. Un incremento en un solo bit permite disponer del doble de precisión (mayor resolución), pero hace más difícil el diseño del circuito, además, la conversión podría volverse más lenta. Dentro de las de aplicaciones de estos sistemas está el manejo de señales de vídeo, audio, los discos compactos, instrumentación y control industrial.

- Interrupciones: se denomina interrupción en un microcontrolador a una función que se ejecuta independientemente del proceso que está realizando en ese momento el microcontrolador, de tal manera, que debe dejar la labor que estaba ejecutando para atender la interrupción solicitante, una vez atendida puede retornar al proceso donde lo dejó. Estas interrupciones pueden dividirse en interrupciones por hardware o por software dependiendo el origen de la misma. Las interrupciones por hardware son asíncronas a la ejecución del microcontrolador, es decir, que pueden ocurrir en cualquier momento, generalmente son originadas por los estados de los puertos de entrada/salida digitales del microcontrolador (producidas por dispositivos externos al microcontrolador, también llamada interrupción Externa).

A.1.3. Principales fabricantes de Microcontroladores

- ATMEL: Esta empresa fabrica los microcontroladores de la familia AVR, esta nueva tecnología proporciona todos los beneficios habituales de arquitectura RISC y memoria flash reprogramable eléctricamente. La característica que los identifica a estos microcontroladores de ATMEL es la memoria flash y EEPROM que incorpora. La firma también produce y vende varios subproductos de la popular familia 8051 con la diferencia de que están basados en la memoria flash. El diseño AVR de ATMEL difiere de los demás microcontroladores de 8 bits por tener mayor cantidad de registros(32) y un conjunto ortogonal de instrucciones (Un set de instrucciones es ortogonal cuando las instrucciones pueden usar se con cualquier registro y modo de direccionamiento, es decir, cualquier instrucción puede utilizar cualquier elemento de la arquitectura como fuente o destino). AVR es mucho más moderna que su competencia. Esto hace que la arquitectura AVR sea más fácil de programar a nivel de lenguaje ensamblador y que sea fácil de optimizar con un compilador. El gran conjunto de registros disminuye la dependencia respecto a la memoria, lo cual mejora la velocidad y disminuye las necesidades de almacenamiento de datos. Además casi todas las instrucciones se ejecutan en 1 ó 2 ciclos de reloj contra 5-10 ciclos de reloj para los chips 8051, 6805, 68HC11 y PIC. Los microcontroladores AVR tienen pipeline con dos etapas (cargar y ejecutar), que les permite ejecutar la mayoría de las instrucciones en un ciclo de reloj, lo que los hace relativamente rápidos entre los microcontroladores de 8-bit.
  Adicionalmente, ATMEL también proporciona en línea el entorno software (AVR estudio) que permite editar, ensamblar y simular el código fuente. Una vez ensamblado y depurado el código fuente del programa, se transferirá el código máquina a la memoria flash del microcontrolador para esto se debe disponer de otro entorno de desarrollo para programar en forma serial o paralelo la memoria flash. [20]

- Microchip: Esta empresa ofrece soluciones para la gama completa de 8-bits, 16-bits y microcontroladores de 32-bits, con una poderosa arquitectura, las tecnologías de memoria flexibles, integrales y fáciles de usar herramientas de desarrollo, documentación técnica completa y posterior diseño de soporte. Fue una de las primeras empresas en utilizar microcontroladores de arquitectura Harvard y la versatilidad de su IDE, pudiéndose programar en lenguajes más amigables que

Assembler como BASIC o C++, hizo que se vuelvan muy populares. Luego, perdieron terreno ya que las otras empresas empezaron a ofrecer las mismas facilidades con arquitecturas más potentes.
Las características más notables de estos microcontroladores son:

- Fácil migración a través de familias de productos.
- Bajo riesgo de desarrollo de productos y mas rápida salida al mercado.
- Reducción del costo total del sistema.
- Servicios de programación de la producción.
- Fácil entorno de programación.

A.2. Microprocesador

A.2.1. Introducción

El microprocesador es el circuito integrado central y más complejo de un sistema informático, es el 'cerebro' de un computador. Conformado por millones de componentes electrónicos. Constituye la unidad central de procesamiento (CPU) de las PC catalogadas como microcomputadores. Se encarga de ejecutar los programas desde el sistema operativo. Ejecuta instrucciones, programa tareas, realiza operaciones.
El primer microprocesador (Intel 4004) se inventó en 1971, era un dispositivo de cálculo de 4 bits, con una velocidad de 108Khz.
Esta unidad central de procesamiento está conformada principalmente por:

- Unidad de Control.
- Unidad aritmético lógica.
- Registros internos.
- Buses Internos.
- Interrupciones.[27]

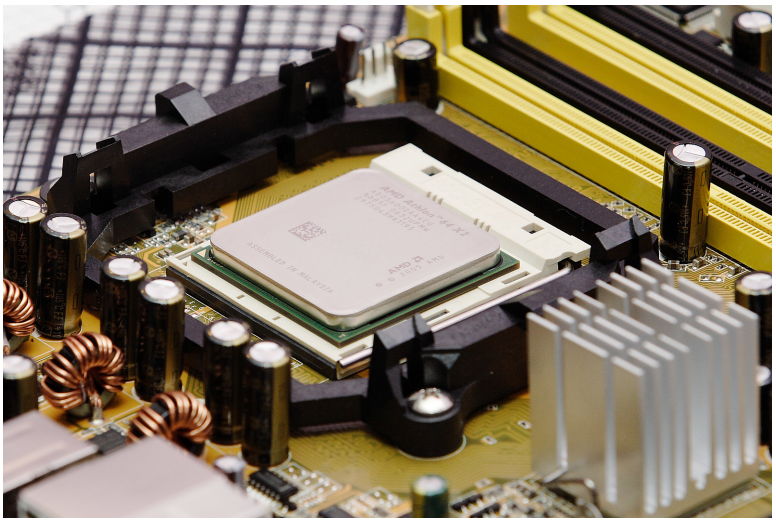


Figura A.4: Microprocesador conectado a su placa base[27]

A.2.2. Partes y complementos de un Microprocesador

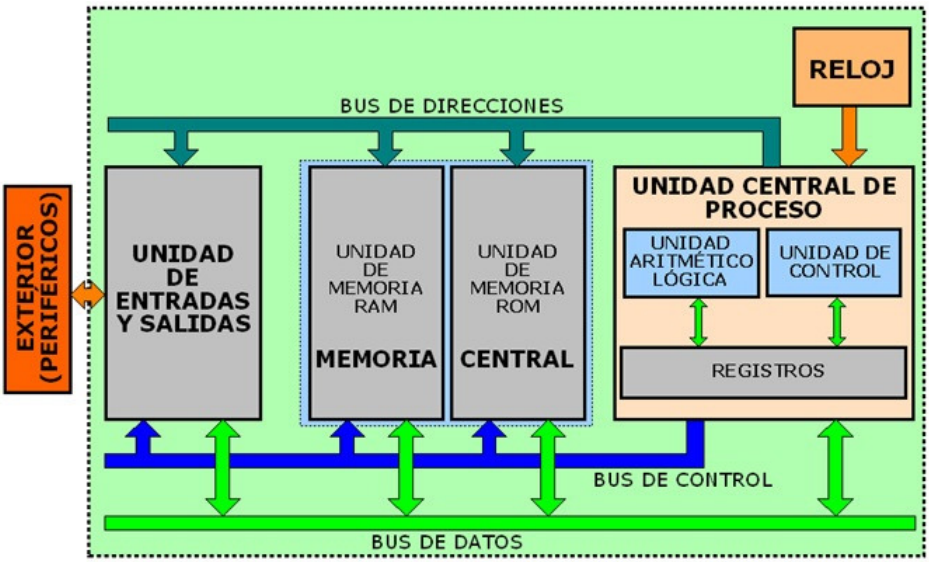


Figura A.5: Arquitectura básica de un microprocesador

- Unidad de control: esta unidad se ocupa de controlar y coordinar el conjunto de operaciones necesarias para realizar el oportuno tratamiento de la información. Su objetivo consiste en extraer de la memoria principal la instrucción a ejecutar. Para ello dispone de un registro, denominado contador de instrucciones, en el que almacena la dirección de la célula que contiene la próxima instrucción a ejecutar, y de un segundo registro, llamado instrucción, en el que deposita la instrucción propiamente dicha.
  Este último está dividido en dos zonas: una contiene el código que

identifica la operación a ejecutar, y la segunda la dirección de la célula en la que está almacenado el operando.
Una vez conocido el código de la operación, la unidad de control ya sabe qué circuitos de la unidad aritmético-lógica deben intervenir, y puede establecer las conexiones eléctricas necesarias a través del secuenciador.
A continuación extrae de la memoria principal los datos necesarios para ejecutar la instrucción en proceso. Para ello simplemente ordena la lectura de la célula cuya dirección se encuentra en la segunda zona del registro de instrucción.
Posteriormente, ordena a la unidad aritmético-lógica que ejecute las oportunas operaciones elementales. El resultado de este tratamiento se deposita en un registro especial de la unidad aritmético-lógica, denominado acumulador. Si la instrucción ha proporcionado nuevos datos, estos son almacenados en la memoria principal.
Por último, incrementa en una unidad el contenido del contador de instrucciones, de tal forma que coincida con la dirección de la próxima instrucción a ejecutar.
También consta de un reloj. El reloj es el oscilador electrónico que hace que el microprocesador vaya de un paso al siguiente al ejecutar las instrucciones (cada instrucción de la máquina ocupa varios ciclos de reloj). La velocidad del reloj se mide en *megaherzios*.

- Unidad aritmético-lógica: la unidad aritmético-lógica (ALU) es el dispositivo encargado de ejecutar las operaciones aritméticas y lógicas, almacenando el resultado en un registro llamado acumulador. Todas estas operaciones las realiza siguiendo las indicaciones dadas por la unidad de control.
  Esta unidad está conectada al mundo exterior a través del bus, canal de señales que une a la ALU con las otras áreas de la unidad central de proceso, y a ésta con dispositivos internos y externos. La ALU puede así recoger los datos de entrada y dar salida a los resultados.

- Interrupciones: una interrupción al igual que en microcontroladores, es una suspensión temporal de la ejecución de un proceso, las interrupciones principalmente son generadas por dispositivos periféricos (externas) o por desbordamientos del timer interno del microprocesador (internas). Para más detalle, puede consultarse la sección A.1.2 del apartado anterior, donde se detalla el funcionamiento de las interrupciones dentro de un microcontrolador.

- Los registros: son básicamente un tipo de memoria pequeña con fines especiales que el micro tiene disponible para algunos usos particulares. Hay varios grupos de registros en cada procesador. Un grupo de registros está diseñado para control del programador y hay otros que son diseñados para ser controlados por el mismo procesador y que la unidad central de procesamientos los utiliza en algunas operaciones.
  Los registros del procesador se emplean para controlar instrucciones en ejecución, manejar direccionamientos de memoria y proporcionar capacidad aritmética. Los registros son espacios físicos dentro del microprocesador con capacidad de 4 bits hasta 64 bits dependiendo del micro que se emplee. Los registros son direccionables por medio de una viñeta, que es una dirección de memoria. Los bits, por conveniencia, se numeran de derecha a izquierda (15, 14, 13, ..., 0).

- La memoria principal: la memoria principal es el dispositivo que conserva durante todo el tiempo de trabajo del microprocesador las instrucciones y los datos necesarios para el desarrollo del proceso.
  Funciona mediante un conjunto de células numeradas (al número que identifica a una célula se le llama dirección). Una vez determinada la dirección de una célula, se puede leer la información que contiene o escribir una nueva información en su interior. Para poder realizar estas operaciones, la memoria dispone de dos registros especiales: el registro de dirección de memoria y el registro de intercambio de datos. El registro de dirección de memoria indica el número de la célula afectada y el registro de intercambio de datos contiene la información leída o la que hay que escribir en la célula en cuestión.
  En ella se almacenan dos tipos de información: el programa o secuencia de instrucciones a ejecutar, y los datos que manejarán dichas instrucciones. Las operaciones que se realizan sobre esta unidad se reducen a dos: lectura y escritura. Evidentemente, las operaciones de escritura destruyen la información almacenada en la célula, al sustituirla por una nueva información. No ocurre así con las de lectura. [29]

A.2.3. Principales arquitecturas de microprocesadores

Las principales arquitecturas en la actualidad son por un lado la x86 (cuyos principales fabricantes son Intel y AMD) y por otro la arquitectura de los microprocesadores ARM. Comparar x86 y ARM es casi como comparar los dos sets de instrucciones principales que nos podemos encontrar, CISC y RISC.

Con x86 nos encontramos el tipo de procesador más usado en equipos de escritorio, el cual utiliza un set de instrucciones de tipo CISC (complex instruction set computing), es decir, que posee soporte para un grupo de instrucciones más complejas, simultáneas y de ejecución más lenta, pero que al fin y al cabo da lugar a códigos menores al poseer menor uso de accesos a memoria (entrada/salida) y la simplificación de la estructura de la programación concurrente.
Por otro lado, los procesadores que lideran el mercado móvil, son los ARM. En este caso utilizan un set de instrucciones del tipo RISC (reduced instruction set computing), es decir, que se centran en la simplificación de instrucciones, buscando siempre la máxima eficiencia por ciclo y organizar mejor las operaciones dentro del núcleo de procesamiento. Las características principales de ambos son:
X86

- Retrocompatibilidad.
- Pocos fabricantes (Intel, AMD, VIA).
- Amplio abanico de instrucciones, de alta complejidad.
- Alto rendimiento.
- Alto uso energético.

ARM

- Amplio abanico de fabricantes, licencias.
- Simpleza, número pequeño de instrucciones.
- Rendimiento aceptable y en crecimiento.
- Bajo uso energético.

[30]

A.3. Modelado en tiempo discreto de Motores CC

Muchas veces es deseable contar con un modelo matemático que represente nuestro motor, su función de transferencia. El mismo nos permitirá modelar el comportamiento del motor en diversas situaciones, pudiendo tener la respuesta que el mismo nos dará frente a distintas excitaciones. En los casos en que deseamos diseñar controles para nuestros sistemas,

contar con la función de transferencia es de fundamental importancia. El modelado podrá realizarse de manera analítica o en forma experimental según los datos con los que se cuente. En el caso que trabajemos en tiempo discreto, el modelado podrá realizarse en tiempo continuo y a través de la transformada Z y con una adecuada tasa de muestreo calcular nuestro modelo en tiempo discreto.[43]

A.3.1. El modelo analítico

El modelo analítico se obtiene a través de la representación de las relaciones entre las diferentes variables físicas del sistema utilizando ecuaciones diferenciales ordinarias de coeficientes constantes. Esto significa que el resultado a obtener es un modelo lineal e invariante en el tiempo.

Las ecuaciones diferenciales del modelo

El siguiente desarrollo corresponde a un motor de corriente continua de imán permanente. Para tener una visión general del motor se utiliza el esquema mostrado en la figura A.6 y se escriben las ecuaciones de malla para el circuito de armadura, la ecuación de la flecha de salida del motor y las relaciones conocidas entre el torque del motor y la corriente de armadura; y la que existe entre la tensión inducida y la velocidad angular del motor.

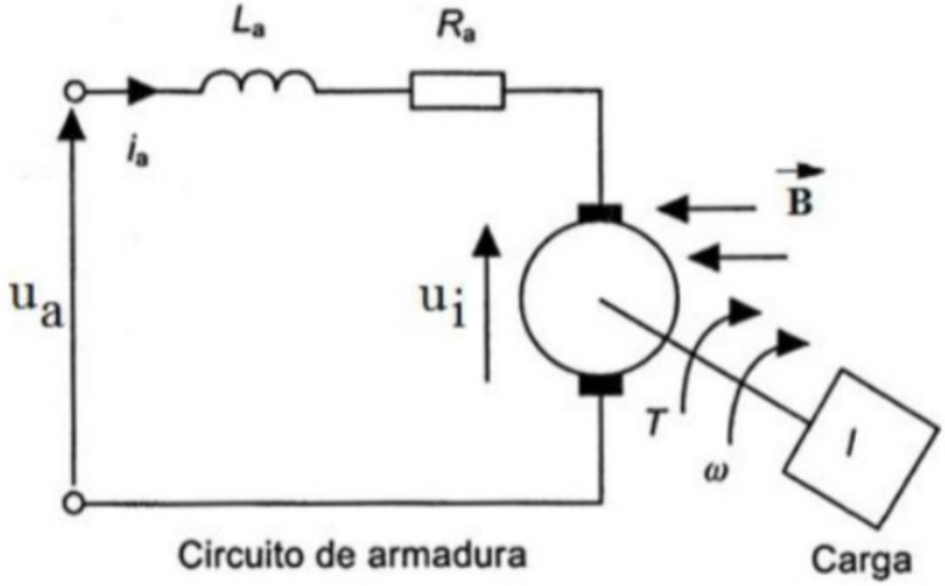


Figura A.6: Esquema del motor CC controlado por armadura

$$u_a = L_a \frac{di_a}{dt} + R_a i_a + u_i \tag{A.1}$$

$$T = J\frac{d\omega}{dt} + B\omega \tag{A.2}$$

$$u_i = K_1\omega(t) \tag{A.3}$$

$$T = K_2 i_a \tag{A.4}$$

Donde:

- u_a es la tensión de armadura.
- L_a es la inductancia de armadura.
- R_a es la resistencia de armadura.
- i_a es la corriente de armadura.
- u_i es la tensión inducida.
- J es la inercia de la flecha de salida del motor.
- B es la fricción del eje del motor.
- T es el torque del motor.
- ω es la velocidad angular del motor.
- $K1$ y $K2$ son constantes del motor, eléctrica y mecánica respectivamente.

Se sustituyen la tensión inducida en la ecuación de la malla y el torque en la ecuación de la flecha de salida del motor. Además se reescriben las

ecuaciones resultantes de forma tal que a la derecha se encuentren las derivadas de la velocidad angular y de la corriente de armadura.

$$\frac{d\omega}{dt} = -\frac{B}{J}\omega + \frac{K_2}{J}i_a \tag{A.5}$$

$$\frac{di_a}{dt} = -\frac{K_1}{L_a}\omega - \frac{R_a}{L_a}i_a + \frac{1}{L_a}u_a \tag{A.6}$$

Luego se escriben las ecuaciones en forma matricial para obtener un modelo en variables de estado físicas, con la velocidad angular y la corriente como variables de estado. La salida del sistema es la velocidad angular.[43]

$$\begin{bmatrix} \frac{d\omega}{dt} \\ \frac{di_a}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{B}{J} & \frac{K_2}{J} \\ -\frac{K_1}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \begin{bmatrix} \omega \\ i_a \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L_a} \end{bmatrix} u_a \tag{A.7}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \omega \\ i_a \end{bmatrix} \tag{A.8}$$

Función de transferencia asociada al estado de un sistema

Anteriormente se logró llegar a las ecuaciones en el espacio de estado del sistema, a continuación se mostrará cómo obtener la función de transferencia en base a las mismas teniendo una sola entrada y una sola salida.
Si se considera un sistema cuya función de transferencia se obtiene mediante:

$$\frac{Y(s)}{U(S)} = G(s) \tag{A.9}$$

Este sistema se representa en el espacio de estado mediante las siguientes ecuaciones:

$$\dot{x} = Ax + Bu \tag{A.10}$$

$$y = Cx + Du \tag{A.11}$$

En donde $\dot{x}$ es la derivada de x, x es el vector de estado, u es la entrada e y es la salida. A su vez estos factores están multiplicados por las matrices A, matriz de estado; B, matriz de entrada; C, matriz de salida y D matriz de transmisión directa.
La transformación de Laplace de las ecuaciones anteriores se obtiene mediante:

$$t_0 \neq 0$$

$$sX(S) - x(0) = AX(S) + BU(S) \tag{A.12}$$

$$Y(S) = CX(S) + DU(S) \tag{A.13}$$

Dado que la función de transferencia se definió como el cociente entre la transformada de Laplace de la salida y la de la entrada, cuando las condiciones iniciales son 0, y suponiendo x(0) = 0, se tiene:

$$t_0 = 0$$

$$sX(S) = AX(S) + BU(S) \tag{A.14}$$

$$Y(S) = CX(S) + DU(S) \tag{A.15}$$

Despejando $X(S)$ de A.14:

$$sX(S) - AX(S) = BU(S) \tag{A.16}$$

$$(sI - A)X(S) = BU(S) \tag{A.17}$$

$$X(S) = (sI - A)^{-1}BU(S) \tag{A.18}$$

Sustituyendo A.18 en la ecuación de salida A.15 se obtiene:

$$Y(S) = [C(sI - A)^{-1}B + D]U(S) \tag{A.19}$$

Comparando A.9 con A.19 se tiene:

$$G(S) = C(sI - A)^{-1}B + D \tag{A.20}$$

Esta es la expresión de la función de transferencia en términos de A, B, C y D.[44]

La función de transferencia del modelo

Comparando las ecuaciones A.10 y A.11 con las ecuaciones en espacio de estado del motor, A.7 y A.8, podemos obtener las matrices representativas del sistema:

$$A = \begin{bmatrix} -\frac{B}{J} & \frac{K_2}{J} \\ -\frac{K_1}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} B = \begin{bmatrix} 0 \\ \frac{1}{L_a} \end{bmatrix} C = \begin{bmatrix} 1 & 0 \end{bmatrix} D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Finalmente, reemplazando las matrices representativas del sistema en la ecuación A.20 se encuentra la función de transferencia para el modelo del motor de corriente continua controlado por armadura.[43]

$$G(S) = \begin{bmatrix} 1 & 0 \end{bmatrix} \left(s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} -\frac{B}{J} & \frac{K_a}{J} \\ -\frac{K_1}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ \frac{1}{L_a} \end{bmatrix} \tag{A.21}$$

$$G(S) = \frac{\frac{K_2}{JL_a}}{s^2 + (\frac{BL_a+JR_a}{JL_a})s + (\frac{BR_a+K_1K_2}{JL_a})} \tag{A.22}$$

Análisis del modelo obtenido
Del resultado obtenido para la función de transferencia del motor se puede inferir, ya que el polinomio característico es de orden 2 y todos los elementos existen y son del mismo signo, que la planta tiene dos polos en el semiplano izquierdo y por lo tanto es estable. Depende de los valores particulares para las constantes físicas del motor si los polos son complejos conjugados o si son reales. En el caso de polos reales, uno de ellos puede ser el polo dominante y el otro no; lo que significaría que este último podría simplificarse quedando un modelo de primer orden.[43]

A.3.2. El modelo empírico

Los valores de las constantes del motor pueden ser obtenidos de la hoja de datos proporcionada por el fabricante y algunas constantes pueden medirse en el laboratorio; pero, a veces no se cuenta con todos los valores que se requieren y en este caso se recurre a uno o varios experimentos para el cálculo de las distintas constantes de nuestro motor con las cuales se llega a un modelo empírico del mismo.
Para el cálculo de éstos parámetros se debe colocar una resistencia externa en serie al motor como muestra la figura A.7.[43]

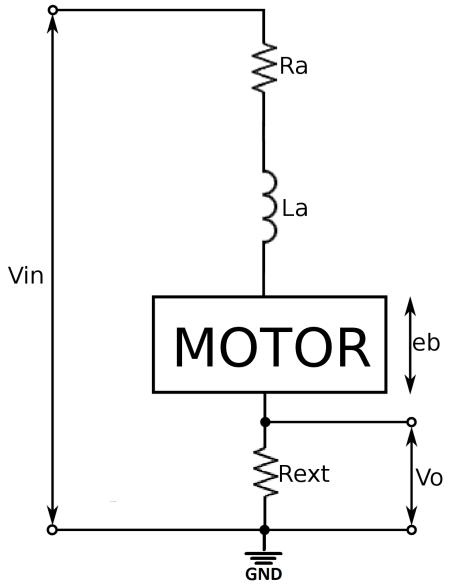


Figura A.7: Circuito del motor con resistencia externa en serie

El procedimiento para la obtención de los mismos se desarrolla a continuación.

Determinación de La y Ra

El procedimiento consiste en introducir un escalón de tensión como entrada y medir a la salida V_o con el motor bloqueado, es decir, $\omega = 0$. Con esto se logra que no haya tensión inducida sobre el mismo y por lo tanto $eb = 0$. La tensión de salida puede expresarse de la siguiente manera:

$$V_o = \frac{V_{in} R_{ext}}{(R_a + R_{ext})} \tag{A.23}$$

Esto implica que la resistencia de interés es:

$$R_a = (\frac{V_{in} R_{ext}}{V_o}) - R_{ext} \tag{A.24}$$

Además si se observa la respuesta transitoria en el osciloscopio se puede calcular la constante de tiempo τ, que indica el tiempo que tarda el sistema en llegar al 63 % de su valor nominal. Está constante se puede calcular de la siguiente manera:

$$\tau = \frac{L_a}{(R_a + R_{ext})} \tag{A.25}$$

Como se puede medir el valor de τ, se puede despejar la inductancia de interés:

$$L_a = \tau(R_a + R_{ext}) \tag{A.26}$$

Determinación de Kb y Kt

Para determinar la constante contra-electromotriz, simplemente se usa el circuito del motor en estado estacionario y se aplican diferentes tensiones de entrada midiendo la velocidad a la salida. La constante puede ser calculada a través de la siguiente ecuación:

$$K_b = \frac{V_{in}}{\omega}[V/(rad/seg)] \tag{A.27}$$

Además, teniendo en cuenta que K_b es numéricamente igual a la constante mecánica K_t, se puede obtener dicha constante considerando las unidades correspondientes (Nm/A).

Determinación de B (constante de amortiguamiento)

Para poder determinar esta constante, se debe trabajar nuevamente con el motor en estado estacionario. Deben aplicarse distintos valores de tensión

a la entrada y medir los correspondientes valores de corriente y velocidad en vacío, I_o y ω_o respectivamente. B puede ser calculada de la siguiente manera:

$$B = \frac{(K_t I_o)}{\omega_o} \tag{A.28}$$

Determinación de Jm

Para obtener el momento de inercia asociado al motor es necesario hacerlo trabajar a una velocidad angular conocida ω_j. Luego se debe desconectar la alimentación del motor y obtener en el osciloscopio una función similar al de la figura A.8, que representa la velocidad en función del tiempo.

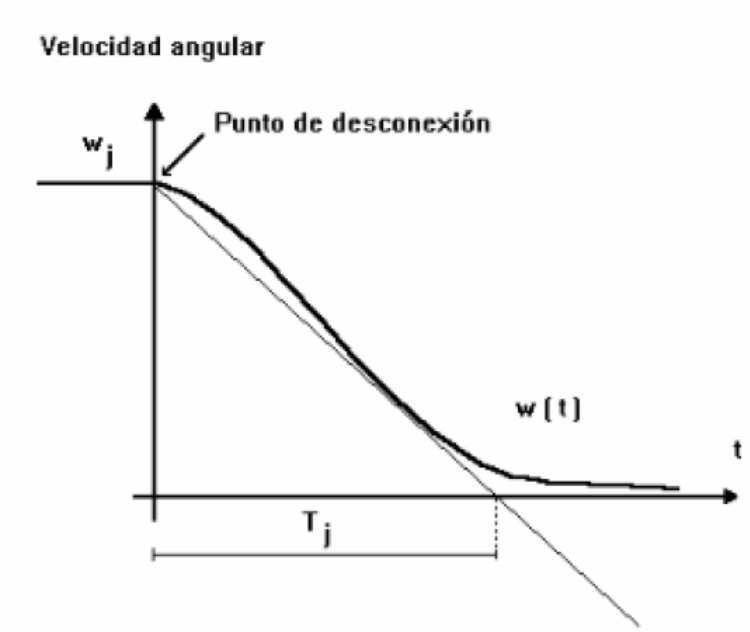


Figura A.8: Velocidad angular vs tiempo[43]

En el punto de desconexión, interactúa con el sistema el momento de frenado (M_b), que produce la disminución de la velocidad. Este parámetro está dado por:

$$M_b = J_m \frac{d\omega}{dt} \tag{A.29}$$

Si se toma el momento de frenado independiente de ω, se puede aproximar la respuesta ideal del sistema con una recta, cuya pendiente es ω_j/T_j. Quedando la ecuación anterior de la siguiente manera:

$$M_b = J_m \frac{\omega_j}{T_j} \tag{A.30}$$

Se puede calcular el momento de frenado a través de las pérdidas ocasionadas por la fricción propia de los componentes inherentes al motor. Esto se realiza a través de una ecuación que surge de la ley de conservación de potencia, dada por:

$$P_R = V_{in} I_a - (I_a^2 R_a) \tag{A.31}$$

Teniendo en cuenta que:

$$P_R = M_b \omega_j \tag{A.32}$$

Reemplazando el momento de frenado en la ecuación A.32 por la expresión de la ecuación A.30, se tiene:

$$P_R = J_m \frac{\omega_j}{T_j} \omega_j \tag{A.33}$$

Si se despeja J_m que es el parámetro que se desea calcular, y se reemplaza P_R por la expresión de la ecuación A.31, finalmente se obtiene la ecuación para el cálculo del momento de inercia asociado al motor:[45]

$$J_m = \frac{P_R T_j}{\omega_j^2} = \frac{[V_{in} I_a - I_a^2 R_a] T_j}{\omega_j^2} \tag{A.34}$$

Obtención del modelo empírico con el software Matlab

En muchas situaciones la disposición física de los componentes imposibilitan el armado de los circuitos anteriores para la medición de cada uno de los parámetros. En estos casos se puede llegar al modelo empírico del motor realizando unas pocas mediciones y utilizando librerías de la herramienta matemática Matlab para la obtención del mismo.

Para obtener el modelo se deben realizar cambios en la entrada del motor, la tensión de armadura, mientras se registran simultáneamente, con la ayuda de un osciloscopio, la respuesta de velocidad obtenida y la entrada aplicada.

El tipo de entrada recomendable para excitar el sistema motor adecuadamente es una señal no determinística; por lo que se debe producir una señal de pulsos de amplitud constante con duración en alto y bajo variables, simulando de esta forma una entrada no determinística.

Luego a través de la librería de identificación de sistemas ident de la herramienta Matlab podrá obtenerse la función de transferencia del motor.[43]

A.4. Control de velocidad Crucero (PID)

A.4.1. Introducción

Un controlador PID (Controlador Proporcional-Integral-Derivativo) es un mecanismo de control por realimentación ampliamente usado en sistemas de control de sistemas. Éste calcula la desviación o error entre un valor medido y un valor deseado.
El algoritmo del control PID consiste de tres parámetros distintos: el proporcional, el integral y el derivativo. El valor proporcional depende del valor actual. El integral depende de los errores pasados y el derivativo es una predicción de los errores futuros. La suma de estas tres acciones es usada para ajustar el proceso por medio de un elemento de control como por ejemplo la potencia suministrada a un motor.
Cuando no se tiene mucho conocimiento de la planta a la que se aplica el control, históricamente se ha considerado que el controlador PID es el más adecuado. Ajustando estas tres variables en el algoritmo de control, el controlador puede proveer una acción de control diseñada para los requerimientos del proceso específico. La respuesta del controlador puede describirse en términos de la respuesta del control ante un error, el grado el cual el controlador sobrepasa el punto de ajuste, y el grado de oscilación del sistema.[46]
El término integral es necesario si se quiere garantizar que el error en régimen permanente sea nulo, es decir si se quiere que la salida del proceso alcance exactamente el valor de la referencia.
El término derivativo requiere que la señal del sensor no sea demasiado ruidosa de lo contrario puede dar problemas. La función de la acción derivativa es mantener el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce y evitando que el error se incremente.
El término proporcional consiste en el producto entre las señal de error y la constante proporcional para lograr que el error en estado estacionario se aproxime a cero.
Las respuestas más rápidas se consiguen con el controlador PD, después con el PID, luego con el P, y la más lenta con el controlador PI. Lo más habitual es la elección del PID completo ya que suele ser un buen compromiso entre rapidez y error.
En esta sección se desarrolla y presenta la función de transferencia de los compensadores PID, se aborda el diseño en el dominio del tiempo de sistemas de control digital utilizando la conocida técnica de lugar geométrico de las raíces, se presentan los compensadores más comunes.[47]

A.4.2. Función de transferencia de un controlador PID digital

La acción de un controlador PID analógico ha sido exitosa en muchos sistemas de control y la mayoría de los lazos de control se pueden manipular mediante un PID digital siempre que el período de muestreo sea lo suficientemente chico.

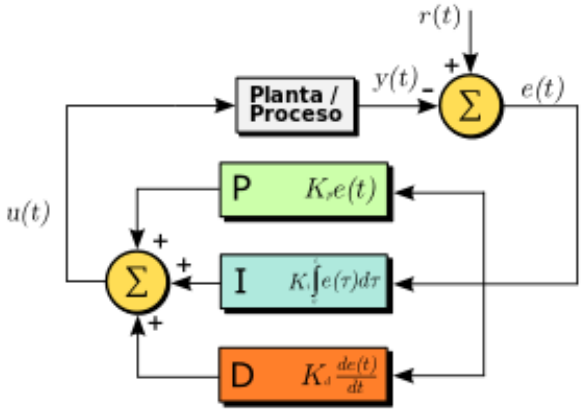


Figura A.9: Controlador PID analógico

La función de transferencia de un PID analógico como el que se muestra en la figura A.9 paralelo está dada por:

$$m(t) = K[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}] \tag{A.35}$$

donde $e(t)$ es la señal error o sea la entrada al controlador en cascada, y $m(t)$ es la variable manipulada, entrada al proceso. K es la ganancia proporcional, T_i es el tiempo de integración o de reajuste (reset), T_d es el tiempo derivativo.
Para obtener la función de transferencia impulso del controlador hay que discretizar esta ecuación para lo cual pueden usarse varios métodos numéricos.
Si se aproxima la integral por medio del método rectangular y la derivada mediante la diferencia entre dos puntos, se obtiene:

$$m(kT) = K\{e(kT) + \frac{T}{T_i}[e(0) + e(T) + ... + e(kT)] + T_d\frac{e(kT) - e((k-1)T)}{T}\} \tag{A.36}$$

$$m(kT) = K\{e(kT) + \frac{T}{T_i}\sum_{n=0}^{k} e(nT) + \frac{T_d}{T}[e(kT) - e((k-1)T)]\} \tag{A.37}$$

Para k que va desde $-\infty$ *a* ∞

Teniendo en cuenta que:

$$Z\{\sum_{n=0}^{k} e(nT)\} = \frac{1}{1-z^{-1}}E(z) \tag{A.38}$$

Se procede a tomar la transformada Z de la ecuación A.37:

$$M(z) = K[1 + \frac{T}{T_i}\frac{1}{1-z^{-1}} + \frac{T_d}{T}(1-z^{-1})]E(z) \tag{A.39}$$

Donde si:

$K = K_P$ ganancia proporcional $\frac{KT}{T_i} = K_I$ ganancia integral $\frac{KT_d}{T} = K_D$ ganancia derivativa

Entonces:

$$M(z) = [K_P + \frac{K_I}{1-z^{-1}} + K_D(1-z^{-1})]E(z) \tag{A.40}$$

A la función de transferencia del PID así determinada, se la denomina 'posicional' y es una forma regularmente usada. Existen otras formas de función de transferencia según los algoritmos de integración y derivación usados.

A.4.3. Función de transferencia en potencias positivas de z

Se multiplica y divide por z para eliminar las potencias negativas:

$$\frac{M(z)}{E(z)} = (K_P + \frac{K_I}{1-z^{-1}} + K_D(1-z^{-1}))\frac{z}{z} = (K_P + \frac{K_I z}{z-1} + K_D\frac{z-1}{z}) \tag{A.41}$$

$$\frac{M(z)}{E(z)} = \frac{K_P z^2 - K_P z + K_I z^2 + 2K_D z^2 - K_D z + K_D}{z(z-1)} \tag{A.42}$$

$$\frac{M(z)}{E(z)} = \frac{z^2(K_P + K_D + K_I) - z(K_P + 2K_D) + K_D}{z(z-1)} \tag{A.43}$$

$$\frac{M(z)}{E(z)} = (K_P + K_D + K_I)\frac{z^2 + z\frac{-(K_P+2K_D)}{K_P+K_D+K_I} + \frac{K_D}{K_P+K_D+K_I}}{z(z-1)} \tag{A.44}$$

De donde puede observarse que la función tiene un polo en el origen, otro en 1 y dos ceros que pueden ser reales o complejos conjugados y son raíces de $z^2(K_P + K_D + K_I) - z(K_P + 2K_D) + K_D = 0$.
Luego el PID se puede poner en forma factorizada como:[47]

$$\frac{M(z)}{E(z)} = K\frac{(z-c_1)(z-c_2)}{z(z-1)} \tag{A.45}$$

A.4.4. Lugar de Raíces

El lugar de las raíces es una técnica que permite conocer las raíces de un polinomio al variar un parámetro (normalmente una ganancia) y es independiente del dominio de transformada que se utilice, es decir las reglas para su construcción son casi las mismas que para tiempo continuo pero contemplando la posibilidad de que haya más ceros que polos, condición que algunas veces puede ocurrir aún para sistemas causales.
La diferencia para el control radica en la relación entre la ubicación de dichas raíces y las características de la respuesta, teniendo en cuenta la correspondencia entre los planos s y z.

Sea el sistema de la figura A.10 con una ganancia positiva y variable $0 < K < \infty$

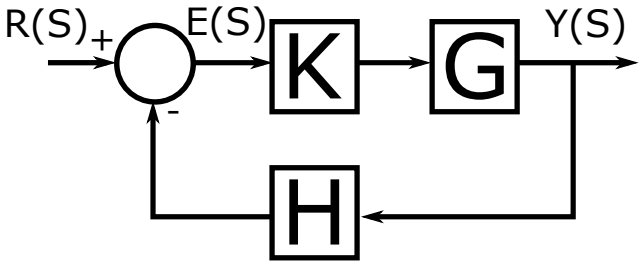


Figura A.10: Sistema de Lazo Cerrado

Su función de transferencia se expresa como sigue:

$$F(z) = \frac{Y}{R} = \frac{KG(z)}{1 + KG(z)} \tag{A.46}$$

Los polos de lazo cerrado son las raíces de la ecuación característica:

$$1 + KGH(z) = 0 \tag{A.47}$$

Lo que implica:

$$GH(z) = -\frac{1}{K} \tag{A.48}$$

Condición de módulo:

$$|GH(z)| = \frac{1}{K} \tag{A.49}$$

Condición de argumento, para valores $k > 0$:

$$\angle GH(z) = (2k + 1)180^{\circ} \tag{A.50}$$

Múltiplo impar de 180°.
Si la función de transferencia de lazo abierto es un cociente de polinomios:

$$F(z)_{LA} = GH(z) = K\frac{N(z)}{D(z)} \tag{A.51}$$

Entonces se puede poner:

$$D(z) + KN(z) = 0 \tag{A.52}$$

Siendo m los ceros finitos de lazo abierto (raíces de N):

$$N(z) = \prod_{i=1}^{m}(z - c_i) \tag{A.53}$$

Y n los polos finitos (raíces de D):

$$D(z) = \prod_{i=1}^{n}(z - p_i) \tag{A.54}$$

Luego las condiciones de módulo y argumento pueden expresarse en término de los polos y ceros como:

$$\frac{\prod_{i=1}^{m}|(z - c_i)|}{\prod_{i=1}^{n}|(z - p_i)|} = \frac{1}{K} \tag{A.55}$$

$$\sum_{i=1}^{m}(z - c_i) - \sum_{i=1}^{n}(z - p_i) = (2k + 1)180^\circ \tag{A.56}$$

Luego el lugar de las raíces es el conjunto de puntos que cumplen la condición de argumento y para una ganancia K se pueden ubicar dichas raíces (polos de lazo cerrado) en puntos fijos que sean parte del LR.
Si bien en tiempo discreto se suele cumplir que $n \geqslant m$, esto no es necesariamente requisito para que el sistema sea realizable, a diferencia de un sistema de tiempo continuo, es decir se debe considerar la posibilidad que haya más ceros que polos.

Resumen de propiedades del lugar de raíces

1. **Puntos de partida** Los puntos de partida $K \to 0$ están en los polos de $GH(z)$, incluye a polos en el infinito, cuando $K = 0 \to D(z) = 0$.

2. **Puntos de llegada** Los puntos de llegada $K \to \infty$ están sobre los ceros de GH(z), incluye a ceros en el infinito. $K = \infty \to N(z) = 0$.

3. **Número de lugares separados** Es igual al número de polos o de ceros, al que sea mayor.

4. **Simetría de los lugares** Los lugares son simétricos con respecto al eje real dado que los polos complejos solo pueden ser de a pares conjugados.

5. **Asíntotas de los lugares de las raíces** Para z grande el lugar de las raíces es asintótico y tiene como asíntotas, lineas rectas, cuyos ángulos están dados por:

$$\theta_k = \frac{(2k+1)\pi}{|n-m|} k = 0, 1, 2, ..., |n-m| \tag{A.57}$$

6. **Centroide** La intersección de las asíntotas en el eje real del plano z está dado por:

$$\sigma = \frac{\sum Rep_i - \sum Rec_i}{n-m} \tag{A.58}$$

7. **Lugares de las raíces sobre el eje real** Existen lugares sobre el eje real, a la izquierda de un número impar de polos y ceros, esto debido a que los argumentos de 2 polos, 2 ceros o un polo y un cero que estén a la derecha se cancelan entre sí, por lo tanto debe quedar al menos un polo o cero a la derecha para que se cumpla la condición de ángulo.

8. **Ángulos de partida y llegada** El ángulo de partida de un polo o de llegada al cero se puede obtener si se toma el punto z_1 sobre el lugar asociado con el polo o cero que esté muy próximo a éste, aplicando la ecuación:

$$GH(z) = (2k+1)180^\circ \tag{A.59}$$

9. **Intersección del lugar de las raíces con el círculo unitario** $|z| = 1$ Los valores de K y de z de la intersección del lugar de raíces con el círculo unitario pueden obtenerse escribiendo el polinomio característico y aplicando un criterio como transformación bilineal o Jury.

10. **Puntos de separación** Los puntos de separación de los lugares de las raíces son los puntos donde se encuentran las raíces repetidas. Se obtiene a partir de la ecuación:

$$\frac{dGH(z)}{dz} = 0 \tag{A.60}$$

    Pero no todas las soluciones a esta ecuación son puntos de separación.

11. **Valores de K sobre los lugares de las raíces** El valor de K en cualquier punto z_1 sobre el lugar de raíces se obtiene a partir de la ecuación de módulo A.55.[47]

A.4.5. Diseño con el Lugar de Raíces

Generalmente no se puede con solo una ganancia variable ubicar arbitrariamente los polos de lazo cerrado, para ello es necesario alterar el lugar introduciendo polos y/o ceros, en general hay varios tipos de controladores o compensadores de tiempo discreto similares a sus equivalentes en tiempo continuo.

- Proporcional (P): consta de una ganancia variable, solo puede aportar al módulo.
- Adelanto: consta de un cero, un polo y una ganancia ajustables con el cero a la derecha. Contribuye con ángulo positivo, proporciona amortiguamiento y mejora la estabilidad.
- Proporcional Derivativo (PD): consta de un polo fijo en el origen, un cero y una ganancia ajustables. Efecto similar al de adelanto.
- Atraso: consta de un cero, un polo y una ganancia ajustables con el polo a la derecha. Proporciona aumento de ganancia estática y mejora el error.
- Proporcional Integrador (PI): consta de un polo, un cero y una ganancia ajustables. Aumenta el tipo de sistema anulando el error permanente, tiene efecto desestabilizante.
- Adelanto atraso: serie de un compensador de adelanto y uno de atraso.
- Proporcional Integrador derivativo (PID): consta de un polo en el origen y dos ceros.

Procedimiento para la compensación

- Determinar la posición deseada de los polos dominantes a lazo cerrado en base a las especificaciones del diseño.
- Graficar el lugar de raíces y verificar si con la ganancia se pueden alcanzar los polos deseados. Si no es posible, calcular la diferencia de ángulo ϕ como: $\phi = 180^{circ}-$ (suma de los ángulos a los polos) + (suma de los ángulos a los ceros). Este ϕ es el que debe proporcionar el compensador.
- El diseño deberá hacerse con la función de transferencia correspondiente al compensador que se desea utilizar. En el caso del compensador PID se utilizará la función correspondiente a la ecuación A.45

- Ubicar el compensador de forma que contribuya al ángulo ϕ.
- Determinar la ganancia de lazo abierto a partir de la condición de módulo.

En caso que el controlador sea PID, entonces hay dos ceros, y existen infinitas soluciones. Lo más habitual es forzar a que los dos ceros sean iguales, con lo que únicamente queda una incógnita que se fija de modo tal que el lugar de raíces pase por el par de polos deseados.
También se puede fijar uno de ellos de forma arbitraria (por ejemplo de un polo del proceso) y elegir el otro para que el lugar de las raíces pase por el par de polos deseados.[47]

A.5. Control de Potencia de Motores CC

A.5.1. Chopper tipo E

A esta configuración también se le conoce como inversor o puente 'H'. Esta estructura, le da la posibilidad de suministrar tensión y corriente positiva y negativa a la carga. Su principal aplicación adicional a la de inversor, es la del control de los campos magnéticos de motores de corriente continua para vehículos eléctricos, este puente permite invertir el sentido de circulación de la corriente en el devanado lo que ocasiona la inversión del sentido de giro del motor.

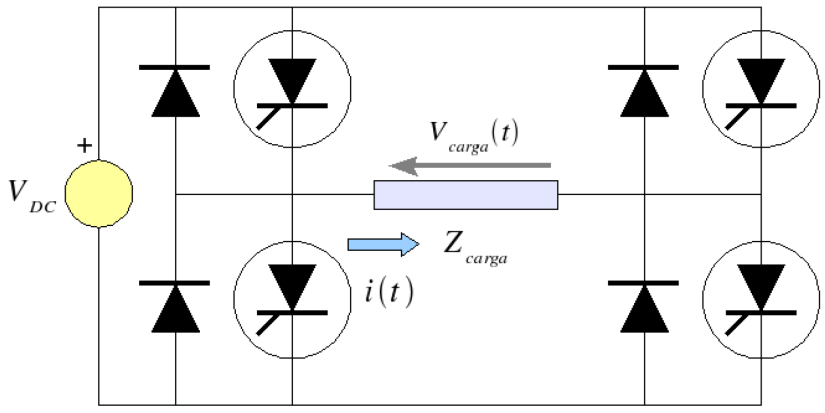


Figura A.11: Chopper tipo E[49]

Principio de funcionamiento con transistores bipolares.
Operación básica
Este puente de transistores y diodos en paralelo es capaz de operar el motor DC en los cuatro modos, es decir, movimiento hacia adelante, movimiento

hacia atrás, frenado hacia adelante y frenado hacia atrás. Los transistores se encienden con señales de control aplicadas a sus bases y referidas a sus emisores.
En el circuito esquemático de la figura A.12 el motor se representa por su resistencia, inductancia y voltaje FEM. Tanto la inductancia como la FEM juegan un papel importante en la conducción de cada transistor o diodo.

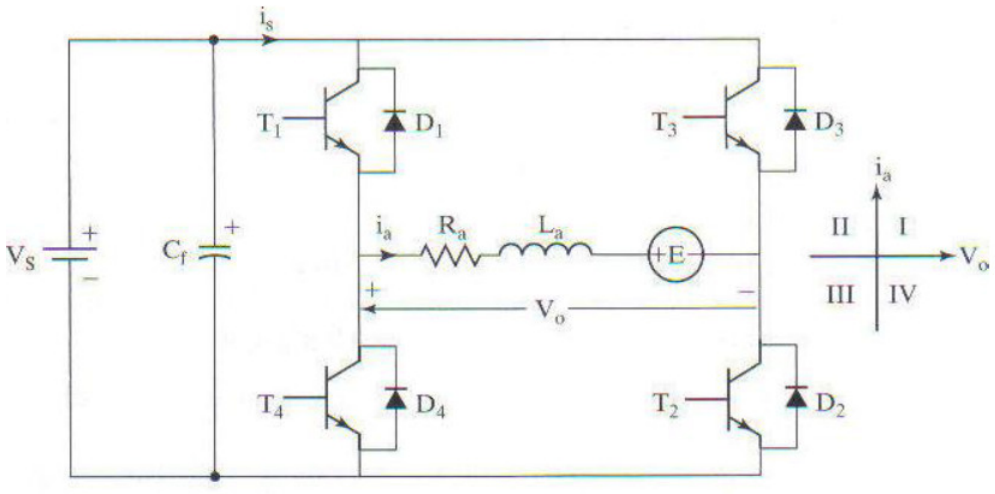


Figura A.12: Circuito Puente H[49]

Operación primer cuadrante

En el período t_{on} donde la corriente en la armadura crece los transistores T1 y T2 se encienden.

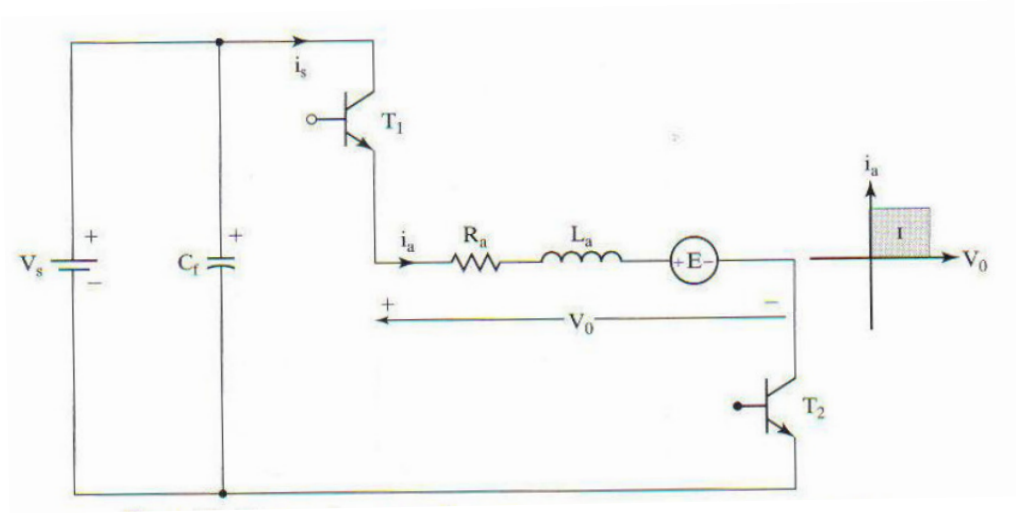


Figura A.13: Operación del primer cuadrante, voltaje y corrientes positivos en la carga[49]

Luego T1 se apaga y se mantiene conduciendo T2. Esto ocasiona que el inductor junto con la FEM hagan conducir el diodo D4, manteniendo el

voltaje de armadura en cero y la corriente circulando en la misma dirección pero decreciente.

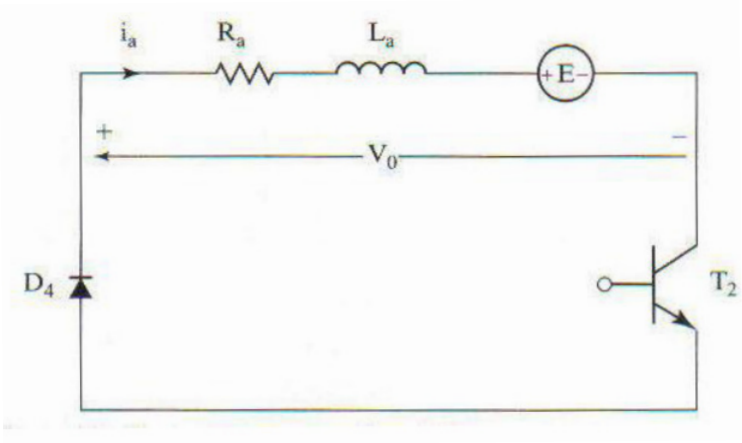


Figura A.14: Operación del primer cuadrante, voltaje y corrientes positivos en la carga[49]

En modo continuo el inductor mantiene circulando la corriente todo el tiempo de t_{off}, por lo que la corriente de armadura no se hace cero, por lo contrario la corriente de la fuente si se hace cero durante este tiempo.

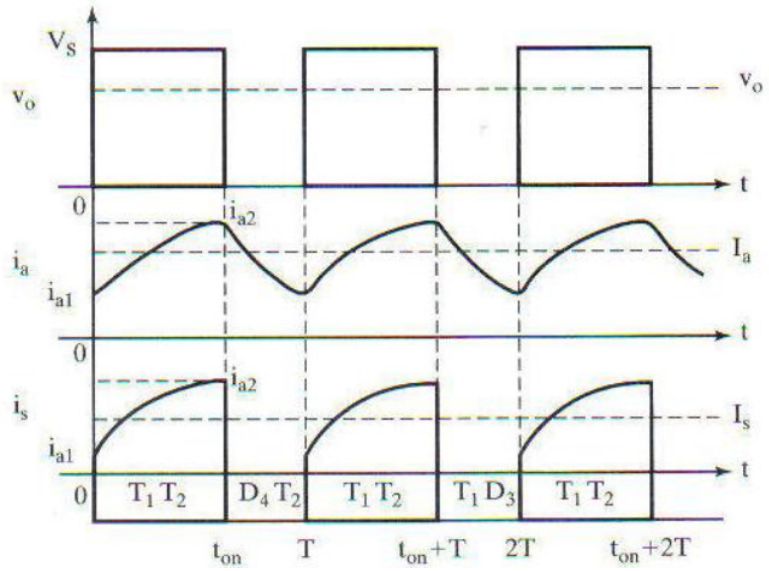


Figura A.15: Gráfico A primer cuadrante[49]

En el modo discontinuo la corriente de armadura se hace cero antes de que comience el próximo t_{on}. Después que la corriente se hace cero el voltaje a través de la armadura se hace igual a la FEM. Este modo se caracteriza por tener mayor rizado en la corriente y el torque y mayor dificultad en los cálculos.

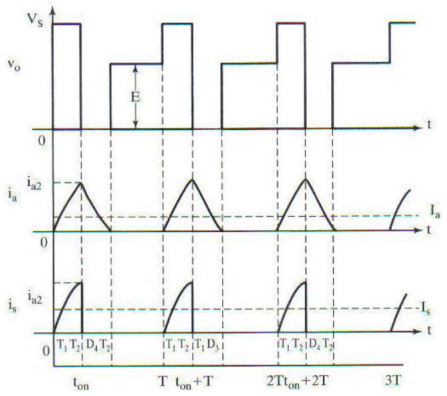


Figura A.16: Gráfico B primer cuadrante[49]

Operación segundo cuadrante

Este modo opera como boost. Para elevar la corriente en el inductor se enciende T2. En este momento el voltaje de armadura es cero. Luego se apaga T2 y el inductor junto a la FEM forzan la corriente a través de D3 y D4 hacia la fuente. Durante este período la corriente en el inductor se reduce.

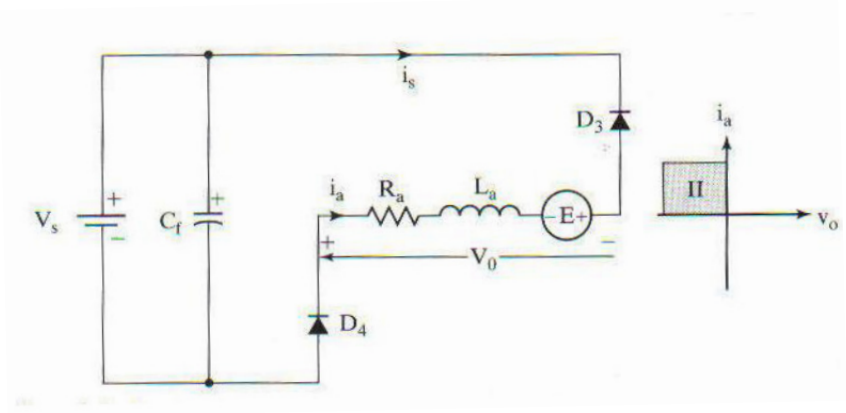


Figura A.17: Operación del segundo cuadrante[49]

La corriente de la fuente es cero durante t_{on} y luego es inversa decreciente durante t_{off}.

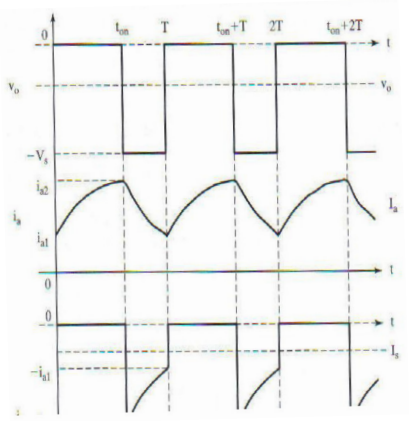


Figura A.18: Operación del segundo cuadrante[49]

Operación tercer cuadrante

Este es el modo en que el motor corre en reversa. Durante t_{on} los transistores T3 y T4 se encienden. El voltaje de armadura es igual al de la fuente y la corriente crece.

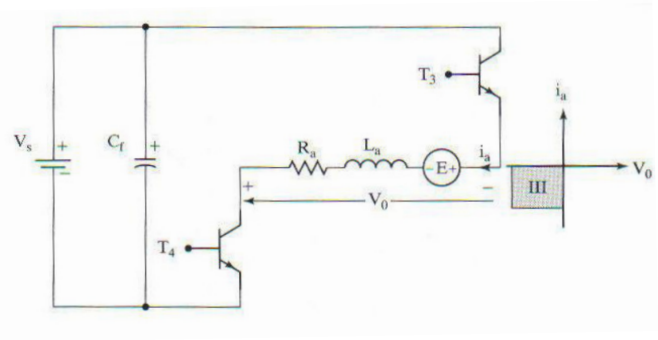


Figura A.19: Operación del tercer cuadrante[49]

Durante t_{off} el transistor T4 se apaga y el inductor fuerza la corriente a través de D1, en este momento la corriente continua circula en la misma dirección pero decreciendo. El voltaje en la armadura es cero. Si la corriente no llega a cero durante este período entonces se está en modo continuo.

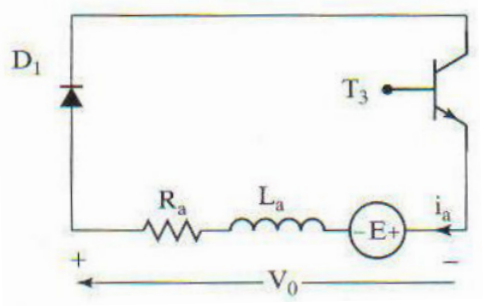


Figura A.20: Operación del tercer cuadrante[49]

El voltaje de armadura cambia entre negativo y cero. La corriente es negativa y nunca sube a cero. La corriente de la fuente es negativa t_{on} y luego es igual a cero.

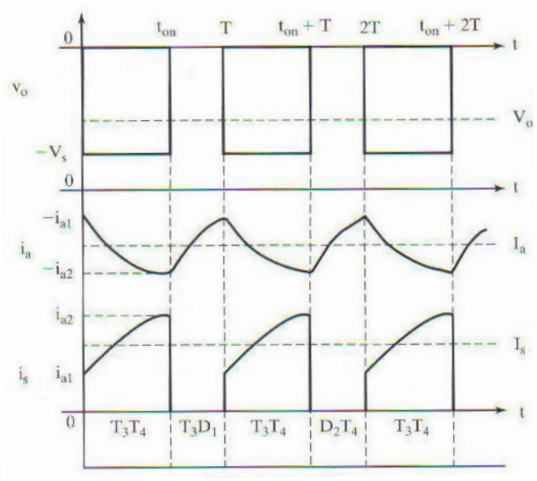


Figura A.21: Gráfico A del tercer cuadrante[49]

El inductor no es capaz de mantener la corriente circulando todo el tiempo para un determinado duty cycle (ciclo de trabajo t_{on}/T) Durante este tiempo que la corriente se hace cero el voltaje de armadura se iguala a la FEM. Los rizados de corriente y torque se acentúan en este modo de operación.

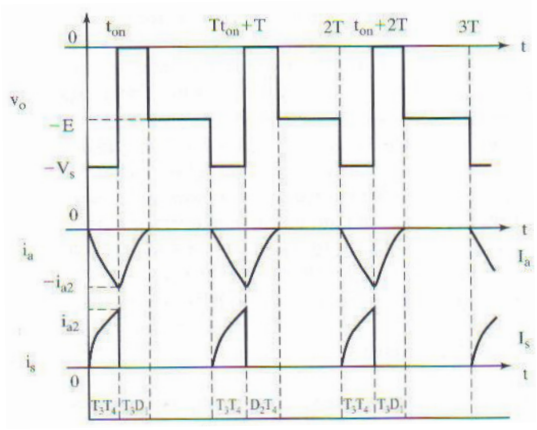


Figura A.22: Gráfico B del tercer cuadrante[49]

Operación cuarto cuadrante

Este modo se utiliza para frenar el motor cuando está en reversa. En este modo opera T4 y D2 para elevar la corriente en el inductor y luego se apaga T4 quedando como único camino D1, D2 para descargar hacia la fuente. En este momento la corriente decrece en valor absoluto acercándose a cero. Si el inductor puede mantener la corriente circulando durante todo t_{off}

entonces se opera en modo continuo.
Generalmente se comienza en modo continuo pero cuando el motor va perdiendo energía cinética la FEM decrece entonces se cae en modo discontinuo.

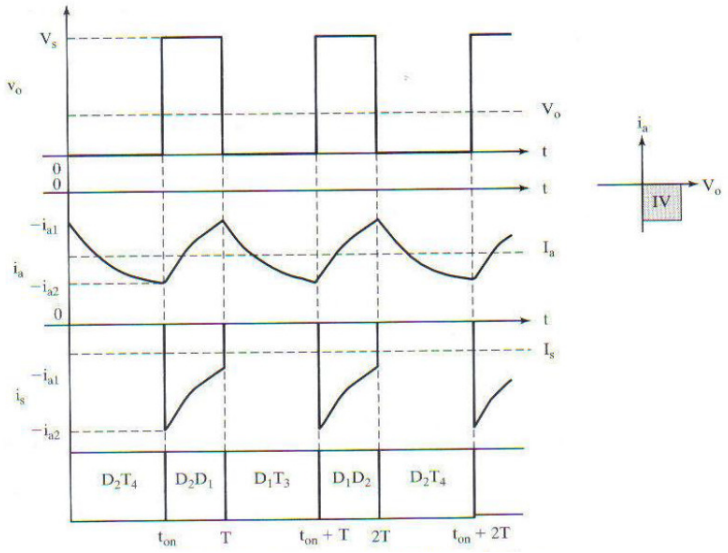


Figura A.23: Gráfico del cuarto cuadrante[49]

Otros dispositivos
Los transistores bipolares pueden ser sustituidos por MOSFET, IGBT e incluso GTO y SCR. Los primeros tres se pueden encender y apagar fácilmente pero el cuarto, el SCR, necesita un sistema especial de conmutación para apagarse. Sin embargo para altas potencias son los únicos que se pueden usar. En general los rangos de potencia van de la siguiente manera:

- Menos de 50KW: BJT, MOSFET.
- Mayor a 50KW: IGBT, GTO SCR, SCR.[49]

A.5.2. Drivers tipo Bootstrap

Un circuito de comando muy utilizado para MOSFETs, es el Bootstrap. Consiste generalmente en un driver tipo totem pole pero comandado por MOSFET. La fuente auxiliar que alimenta al totem pole consiste en un condensador cargado por un diodo desde una fuente referida al 0V de la fuente principal E. La figura A.24 muestra una implementación posible.
Cuando el MOSFET a comandar Q está cortado su source queda al potencial de la referencia, a través del MOSFET 'low side' como en un inversor, o a través de una carga, si se trata de otro tipo de circuito.En ese período el condensador C_{boot} se carga a través del diodo D desde la fuente

U_{cc}, generalmente de 12V a 18V y queda a ese valor. Al aplicar el pulso de prendido a los gates del totem pole, Q_3 se prende y aplica la tensión del condensador C_{boot} al gate de Q, el cual pasa al estado de conducción. EL source de Q sube rápidamente al potencial E al bajar la tensión U_{DS}. El condensador C_{boot} se mantiene cargado a la tensión U_{cc}, manteniendo la tensión de prendido durante el tiempo de conducción. El diodo D queda polarizado en inverso e impide que el condensador se descargue. C_{boot} se descarga solamente para cargar la capacidad de entrada C_{iss} de Q. Su valor se elige entonces uno o dos órdenes de magnitud mayor que dicha capacidad para que la tensión de gate se mantenga en un valor adecuado. Para apagar Q se prende Q_4 que descarga las capacidades de gate en forma usual. EL transistor 'low side' o la carga llevan el source de Q a cero, y C_{boot} repone la carga perdida a través de D.
El problema es entonces cómo llevar la señal de comando desde un circuito de control, normalmente referido a 0V, a los gates del totem pole, cuya tensión de trabajo está referida al source de Q. Para implementar esta función se utiliza un desplazador de nivel o level shifter. El level shifter básico consiste en lo que se llama un MOSFET de muy alta tensión de bloqueo (el valor de E puede llegar a 600V, por ejemplo). En la figura A.24 es el MOSFET Q_5. Cuando la tensión de control U_c es cero, Q_5 prende a través del buffer schimitt trigger inversor y mantiene en un 'cero flotante', coincidente con el potencial de source, a la tensión de entrada del buffer que oficia de driver del totem pole. Como el source de Q está en cero los dos buffers se alimentan de U_{cc}. Si U_c sube al nivel que indica prendido, Q_5 se apaga. La resistencia R_1, conectada entre la entrada del buffer y la tensión U_{cboot} (que en ese momento es aproximadamente U_{cc}), pone un 1 en la entrada del buffer inversor que maneja Q_3 y Q_4. Su salida es un cero que prende el MOSFET canal p Q_3 y por lo tanto Q. El source de Q sube al valor E, por lo tanto la tensión U_{DS} de Q_5, que está cortado, también sube a un valor próximo a E.
Para apagar Q se aplica una tensión $U_C = 0$. Q_5 prende, y mediante R_2, Z_1 y D_1 se aplica un cero en la entrada del buffer 2. El zener o el diodo mantiene la tensión de entrada del buffer dentro de los límites admisibles. Se aplica entonces una señal lógica desde el nivel 0V al nivel E, cumpliéndose la función de 'desplazamiento de nivel'. La salida del buffer 2 sube al valor $Ucboot$, se prende Q_4, Q se apaga y el source de Q vuelve al nivel cero, cumpliéndose un período de prendido y apagado. Este circuito simple permite explicar la función de Bootstrap. En la práctica estos circuitos vienen como circuitos integrados, incluyen una serie de funciones de protección, y

sólo es necesario agregar el diodo D y el condensador C_{boot}. [50]

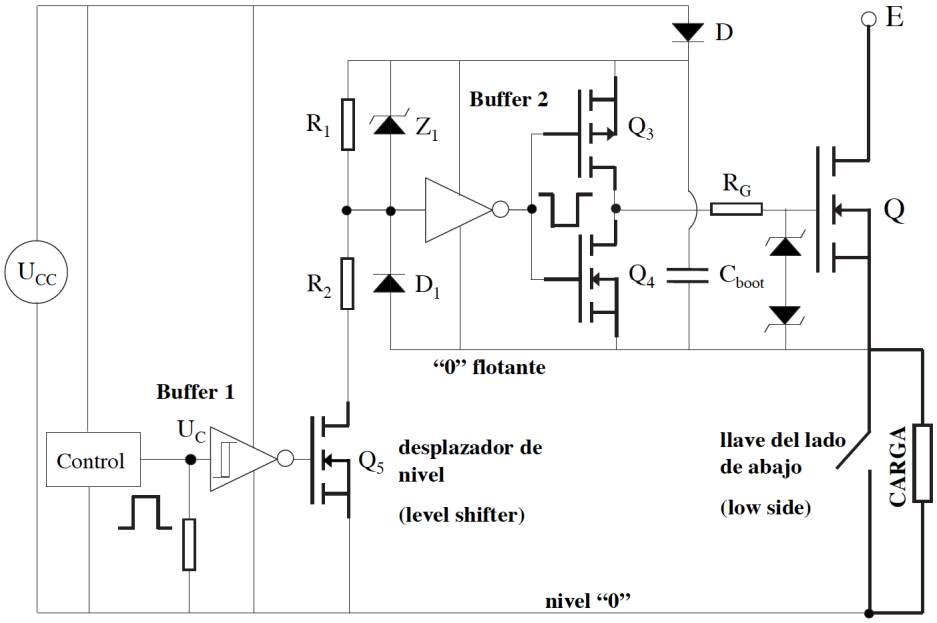


Figura A.24: Circuito de comando tipo Bootstrap

A.5.3. Modulación por ancho de pulsos

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width-modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga. En robótica, es extremadamente útil para controlar la velocidad de los motores de corriente continua, la velocidad del motor será proporcional al ancho de pulso de la señal PWM que lo esté controlando.
El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$D = \frac{\tau}{T} \tag{A.61}$$

D es el ciclo de trabajo.
τ es el tiempo en que la función es positiva (ancho de pulso).
T es el período de la función.

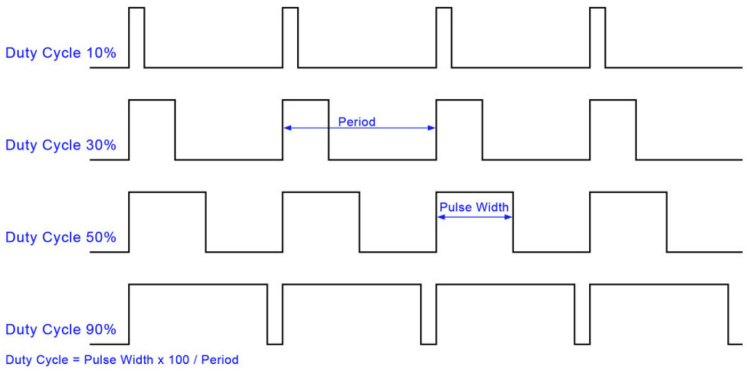


Figura A.25: Señal PWM con diferentes ciclos de trabajo (Duty Cycle)

La construcción típica de un circuito PWM puede llevarse a cabo fácilmente con un LM555. El siguiente circuito se trata de un oscilador de onda cuadrada con modulación de ancho de pulso (PWM).
La configuración del timer 555 es como astable y la variante está en el agregado de dos diodos de conmutación para restringir el semiciclo positivo y negativo en la carga y descarga del capacitor.
El cálculo de frecuencia se realiza con la siguiente fórmula: $f = \frac{1}{(1{,}44 * R * C)}$ donde R está en Ohms y C está en Faraday. Se muestra un ejemplo en la figura A.26 con una resistencia R de 100k Ohms, y un capacitor de 68nF, dando una frecuencia de 102Hz.

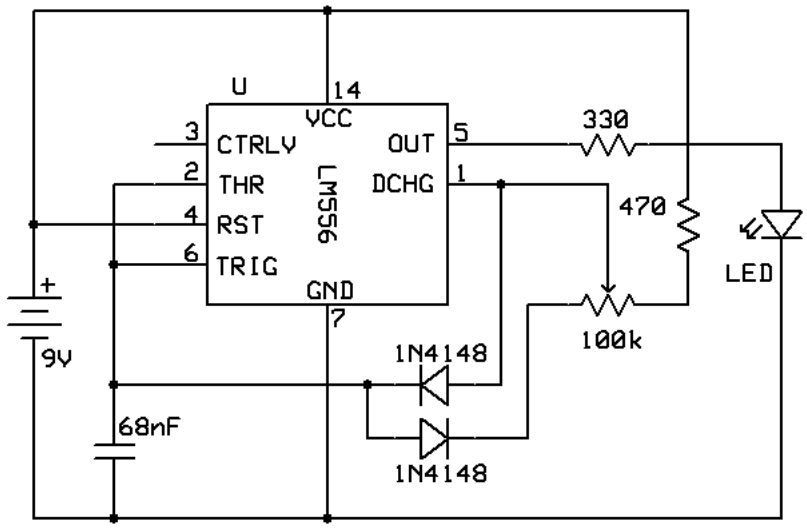


Figura A.26: Circuito PWM con integrado LM555

En la actualidad, la mayoría de los microcontroladores cuentan con módulo PWM, lo que significa no tener la necesidad de utilizar un circuito generador de esta señal. [51]

A.6. Comunicación i^2c

La comunicación i^2c tiene dos características principales: arquitectura de bus verdadero y la utilización de sólo dos líneas de cable para la comunicación. La ventaja de tener un bus verdadero es que se necesitan menos líneas de cables ya que el dispositivo maestro utiliza las direcciones de los dispositivos en modo esclavo para la comunicación en vez de tener una línea de cable para cada uno de ellos. Las dos líneas utilizadas se denominan datos (SDA) y reloj (SCL), ambas están conectadas a través de una resistencia pull-up a la tensión positiva de alimentación (VDD).

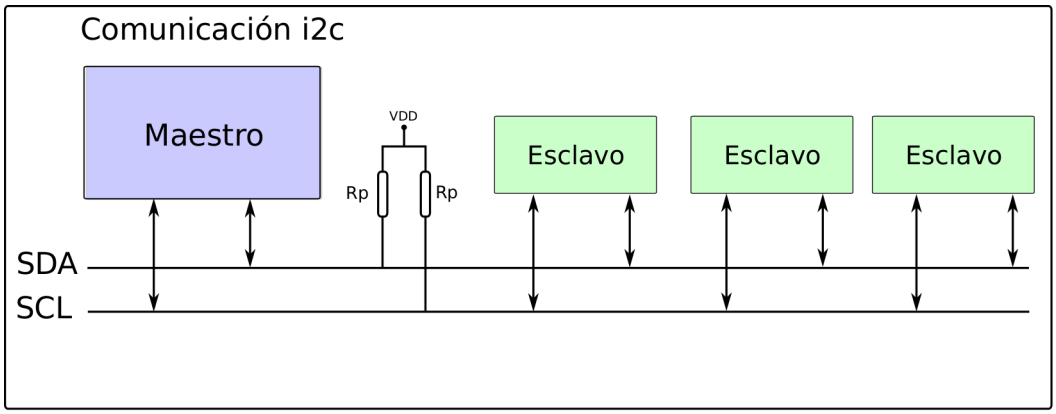


Figura A.27: Comunicación i2c

Este tipo de comunicaciones permite que varios dispositivos en modo maestro estén conectados al mismo bus ya que pueden tener información si el bus está siendo utilizado o está desocupado para empezar la transmisión a través del estado de alta impedancia. En términos eléctricos cuando el bus está libre para utilizarse la línea SDA tiene un uno lógico ya que todas los dispositivos en modo maestro están en modo de alta impedancia.
La comunicación comienza cuando cualquier dispositivo maestro establece una condición de inicio, esto ocurre cuando se pone en estado bajo la línea de datos SDA pero se deja en estado alto la línea de reloj. El primer byte que se transmite luego de la condición de inicio contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar con él (lectura o escritura).
Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits (b0-b6) está presente en el bus, éste contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de reconocimiento (ACK) en bajo le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de

comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos.
Si el bit de lectura/escritura (R/W) fue puesto en esta comunicación a nivel lógico bajo (escritura), el dispositivo maestro envía datos al dispositivo esclavo. Esto se mantiene mientras continúe recibiendo señales de reconocimiento, y el contacto concluye cuando se hayan transmitido todos los datos.
En el caso contrario, cuando el bit de lectura/escritura estaba a nivel lógico alto (lectura), el dispositivo maestro genera pulsos de reloj para que el dispositivo esclavo pueda enviar los datos. Luego de cada byte recibido el dispositivo maestro (quien está recibiendo los datos) genera un pulso de reconocimiento.
El dispositivo maestro puede dejar libre el bus generando una condición de parada (o detención; stop en inglés).Si se desea seguir transmitiendo, el dispositivo maestro puede generar otra condición de inicio en lugar de una condición de parada. Esta nueva condición de inicio se denomina 'inicio reiterado' y se puede emplear para direccionar un dispositivo esclavo diferente o para alterar el estado del bit de lectura/escritura.

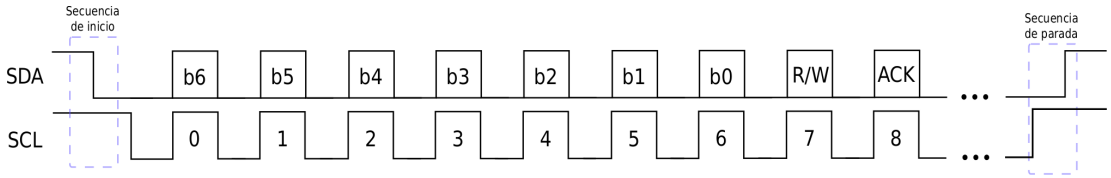


Figura A.28: Ejemplo comunicación i2c

Existen tres estándares de velocidad en los cuales pueden operar los componentes en una comunicación i^2c:

- Standard (100 kbits/s)
- Fast (400 kbits/s)
- Highspeed (3.4 Mbits/s)

Apéndice B

Hojas de datos

Data Sheet No. PD60147 rev.U

International
IOR Rectifier

IR2110(-1-2)(S)PbF/IR2113(-1-2)(S)PbF

HIGH AND LOW SIDE DRIVER

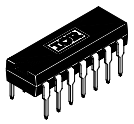
Features

- Floating channel designed for bootstrap operation
  Fully operational to +500V or +600V
  Tolerant to negative transient voltage
  dV/dt immune
- Gate drive supply range from 10 to 20V
- Undervoltage lockout for both channels
- 3.3V logic compatible
  Separate logic supply range from 3.3V to 20V
  Logic and power ground ±5V offset
- CMOS Schmitt-triggered inputs with pull-down
- Cycle by cycle edge-triggered shutdown logic
- Matched propagation delay for both channels
- Outputs in phase with inputs

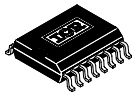
Product Summary

V_{OFFSET} (IR2110)	500V max.
(IR2113)	600V max.
I_O+/-	2A / 2A
V_{OUT}	10 - 20V
$t_{on/off}$ (typ.)	120 & 94 ns
Delay Matching (IR2110)	10 ns max.
(IR2113)	20ns max.

Packages



14-Lead PDIP
IR2110/IR2113

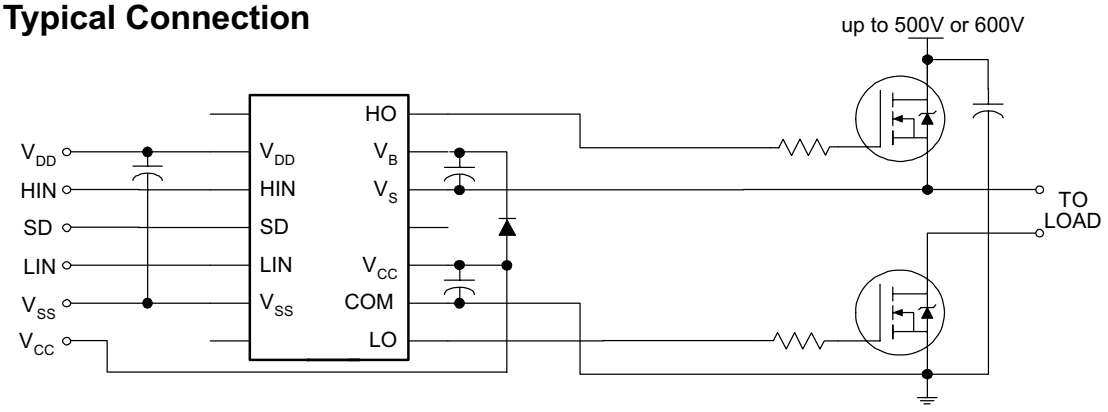


16-Lead SOIC
IR2110S/IR2113S

Description

The IR2110/IR2113 are high voltage, high speed power MOSFET and IGBT drivers with independent high and low side referenced output channels. Proprietary HVIC and latch immune CMOS technologies enable ruggedized monolithic construction. Logic inputs are compatible with standard CMOS or LSTTL output, down to 3.3V logic. The output drivers feature a high pulse current buffer stage designed for minimum driver cross-conduction. Propagation delays are matched to simplify use in high frequency applications. The floating channel can be used to drive an N-channel power MOSFET or IGBT in the high side configuration which operates up to 500 or 600 volts.

Typical Connection



(Refer to Lead Assignments for correct pin configuration). This/These diagram(s) show electrical connections only. Please refer to our Application Notes and DesignTips for proper circuit board layout.

Absolute Maximum Ratings

Absolute maximum ratings indicate sustained limits beyond which damage to the device may occur. All voltage parameters are absolute voltages referenced to COM. The thermal resistance and power dissipation ratings are measured under board mounted and still air conditions. Additional information is shown in Figures 28 through 35.

Symbol	Definition		Min.	Max.	Units
V_B	High side floating supply voltage	(IR2110)	-0.3	525	V
		(IR2113)	-0.3	625	
V_S	High side floating supply offset voltage		V_B - 25	V_B + 0.3	
V_{HO}	High side floating output voltage		V_S - 0.3	V_B + 0.3	
V_{CC}	Low side fixed supply voltage		-0.3	25	
V_{LO}	Low side output voltage		-0.3	V_{CC} + 0.3	
V_{DD}	Logic supply voltage		-0.3	V_{SS} + 25	
V_{SS}	Logic supply offset voltage		V_{CC} - 25	V_{CC} + 0.3	
V_{IN}	Logic input voltage (HIN, LIN & SD)		V_{SS} - 0.3	V_{DD} + 0.3	
dV_s/dt	Allowable offset supply voltage transient (figure 2)		—	50	V/ns
P_D	Package power dissipation @ $T_A \leq +25°C$	(14 lead DIP)	—	1.6	W
		(16 lead SOIC)	—	1.25	
R_{THJA}	Thermal resistance, junction to ambient	(14 lead DIP)	—	75	°C/W
		(16 lead SOIC)	—	100	
T_J	Junction temperature		—	150	°C
T_S	Storage temperature		-55	150	
T_L	Lead temperature (soldering, 10 seconds)		—	300	

Recommended Operating Conditions

The input/output logic timing diagram is shown in figure 1. For proper operation the device should be used within the recommended conditions. The V_S and V_{SS} offset ratings are tested with all supplies biased at 15V differential. Typical ratings at other bias conditions are shown in figures 36 and 37.

Symbol	Definition		Min.	Max.	Units
V_B	High side floating supply absolute voltage		V_S + 10	V_S + 20	V
V_S	High side floating supply offset voltage	(IR2110)	Note 1	500	
		(IR2113)	Note 1	600	
V_{HO}	High side floating output voltage		V_S	V_B	
V_{CC}	Low side fixed supply voltage		10	20	
V_{LO}	Low side output voltage		0	V_{CC}	
V_{DD}	Logic supply voltage		V_{SS} + 3	V_{SS} + 20	
V_{SS}	Logic supply offset voltage		-5 (Note 2)	5	
V_{IN}	Logic input voltage (HIN, LIN & SD)		V_{SS}	V_{DD}	
T_A	Ambient temperature		-40	125	°C

Note 1: Logic operational for V_S of -4 to +500V. Logic state held for V_S of -4V to $-V_{BS}$. (Please refer to the Design Tip DT97-3 for more details).
Note 2: When V_{DD} < 5V, the minimum V_{SS} offset is limited to $-V_{DD}$.

Dynamic Electrical Characteristics

V_{BIAS} (V_{CC}, V_{BS}, V_{DD}) = 15V, C_L = 1000 pF, T_A = 25°C and V_{SS} = COM unless otherwise specified. The dynamic electrical characteristics are measured using the test circuit shown in Figure 3.

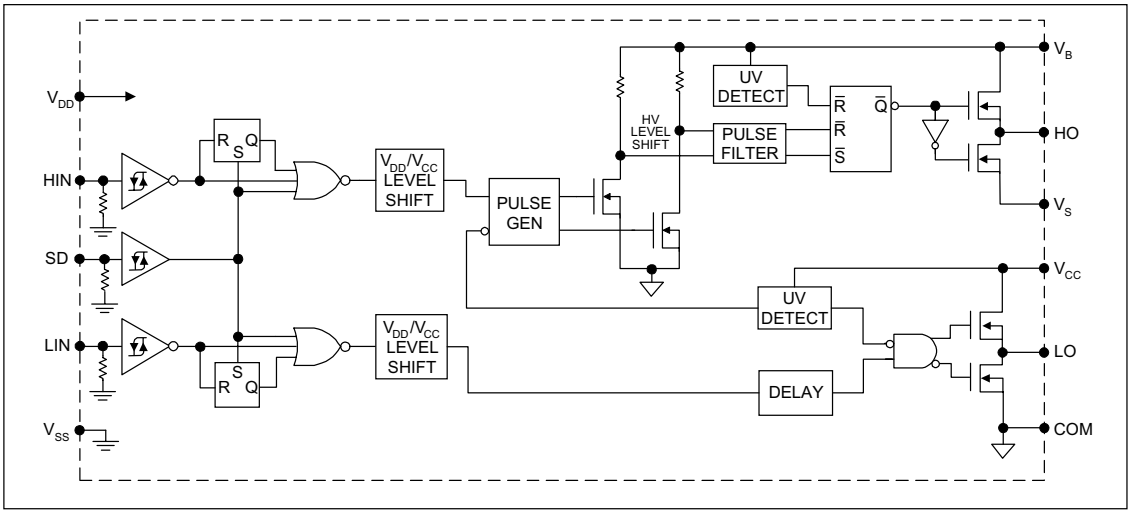
Symbol	Definition	Figure	Min.	Typ.	Max.	Units	Test Conditions
t_{on}	Turn-on propagation delay	7	—	120	150	ns	V_S = 0V
t_{off}	Turn-off propagation delay	8	—	94	125		V_S = 500V/600V
t_{sd}	Shutdown propagation delay	9	—	110	140		V_S = 500V/600V
t_r	Turn-on rise time	10	—	25	35		
t_f	Turn-off fall time	11	—	17	25		
MT	Delay matching, HS & LS (IR2110)	—	—	—	10		
	turn-on/off (IR2113)	—	—	—	20		

Static Electrical Characteristics

V_{BIAS} (V_{CC}, V_{BS}, V_{DD}) = 15V, T_A = 25°C and V_{SS} = COM unless otherwise specified. The V_{IN}, V_{TH} and I_{IN} parameters are referenced to V_{SS} and are applicable to all three logic input leads: HIN, LIN and SD. The V_O and I_O parameters are referenced to COM and are applicable to the respective output leads: HO or LO.

Symbol	Definition	Figure	Min.	Typ.	Max.	Units	Test Conditions
V_{IH}	Logic “1” input voltage	12	9.5	—	—	V	
V_{IL}	Logic “0” input voltage	13	—	—	6.0		
V_{OH}	High level output voltage, V_{BIAS} - V_O	14	—	—	1.2		I_O = 0A
V_{OL}	Low level output voltage, V_O	15	—	—	0.1		I_O = 0A
I_{LK}	Offset supply leakage current	16	—	—	50	µA	V_B=V_S = 500V/600V
I_{QBS}	Quiescent V_{BS} supply current	17	—	125	230		V_{IN} = 0V or V_{DD}
I_{QCC}	Quiescent V_{CC} supply current	18	—	180	340		V_{IN} = 0V or V_{DD}
I_{QDD}	Quiescent V_{DD} supply current	19	—	15	30		V_{IN} = 0V or V_{DD}
I_{IN+}	Logic “1” input bias current	20	—	20	40		V_{IN} = V_{DD}
I_{IN-}	Logic “0” input bias current	21	—	—	1.0		V_{IN} = 0V
V_{BSUV+}	V_{BS} supply undervoltage positive going threshold	22	7.5	8.6	9.7	V	
V_{BSUV-}	V_{BS} supply undervoltage negative going threshold	23	7.0	8.2	9.4		
V_{CCUV+}	V_{CC} supply undervoltage positive going threshold	24	7.4	8.5	9.6		
V_{CCUV-}	V_{CC} supply undervoltage negative going threshold	25	7.0	8.2	9.4		
I_{O+}	Output high short circuit pulsed current	26	2.0	2.5	—	A	V_O = 0V, V_{IN} = V_{DD} PW ≤ 10 µs
I_{O-}	Output low short circuit pulsed current	27	2.0	2.5	—		V_O = 15V, V_{IN} = 0V PW ≤ 10 µs

Functional Block Diagram



Lead Definitions

Symbol	Description
V_{DD}	Logic supply
HIN	Logic input for high side gate driver output (HO), in phase
SD	Logic input for shutdown
LIN	Logic input for low side gate driver output (LO), in phase
V_{SS}	Logic ground
V_B	High side floating supply
HO	High side gate drive output
V_S	High side floating supply return
V_{CC}	Low side supply
LO	Low side gate drive output
COM	Low side return

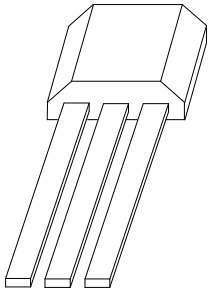
General Description

The FS177 is an integrated Hall effect latched sensor designed for electronic commutation of brush-less DC motor applications. The device includes an on-chip Hall voltage generator for magnetic sensing, a comparator that amplifies the Hall voltage, and a Schmitt trigger to provide switching hysteresis for noise rejection, and open-collector output. An internal bandgap regulator is used to provide temperature compensated supply voltage for internal circuits and allows a wide operating supply range.

A north pole of sufficient strength will turn the output ON. In the absence of a magnetic field, the output is OFF.

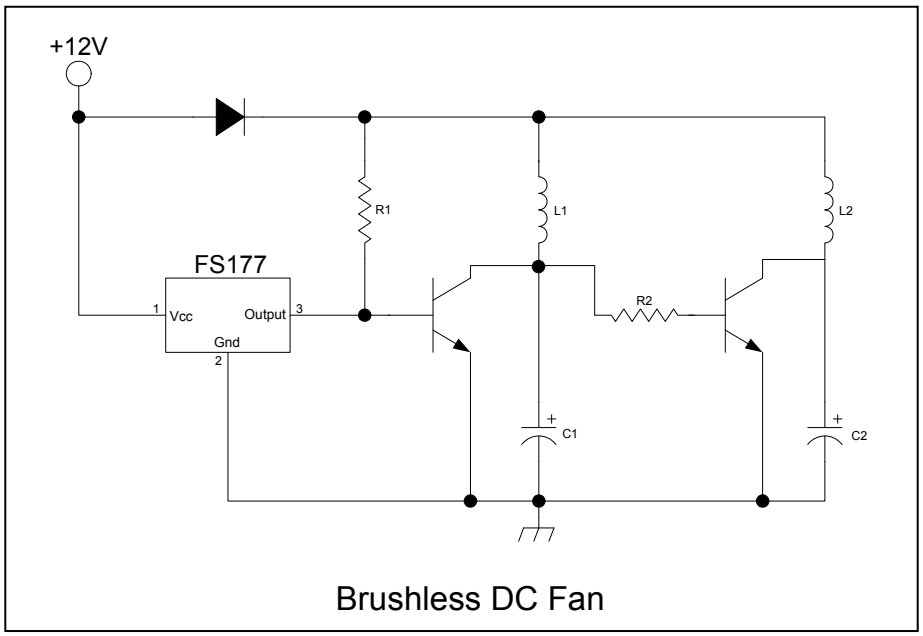
Features

- Wide operating voltage range: 3.0V to 20V
- Maximum output sink current 25mA
- Open-Collector pre-driver
- Reverse polarity protection
- Package : TO-92SP-3



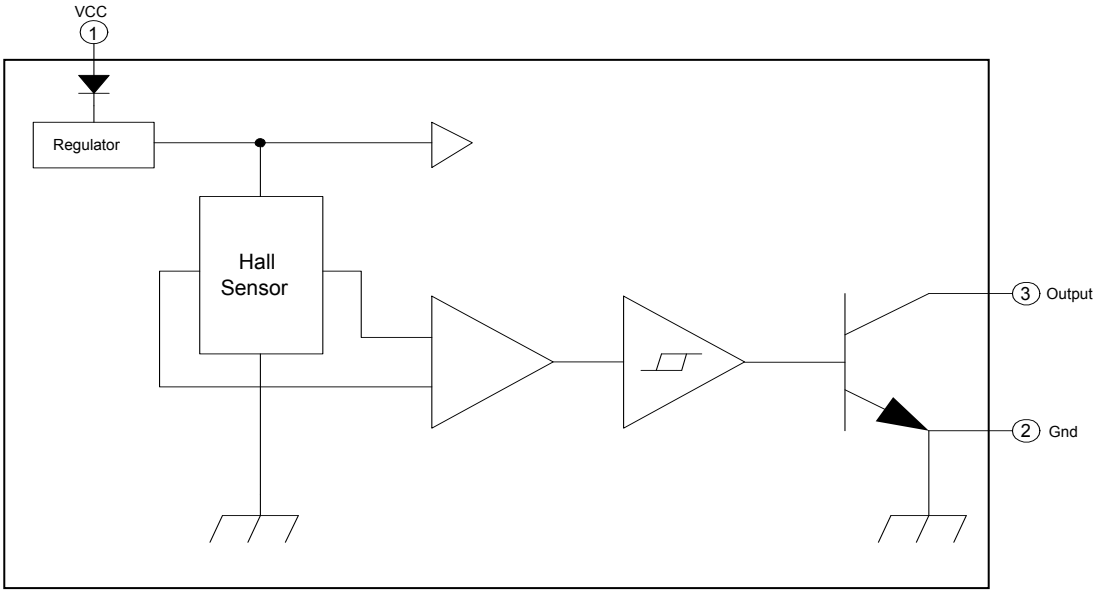
TO-92SP-3

Typical Application Circuit

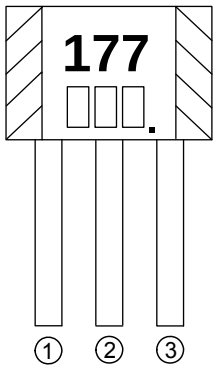


Brushless DC Fan

Functional Block Diagram

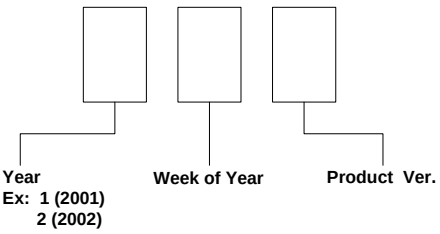


Mark View



Pin Descriptions

NAME	NO.	STATUS	DESCRIPTION
VCC	1	P	IC Power Supply
GND	2	P	IC Ground
Output	3	O	It is low state during the N magnetic field.



Absolute Maximum Ratings (at Ta = 25 °C)

VCC Pin Voltage ---------- 20V

Output OFF Voltage, Vce ---------- 20V

Output ON Current (Io)

Continuous Current ---------- 25mA

Power Dissipation

Ta=25 °C ---------- 400mW

Ta=100 °C ---------- 178mW

Thermal Resistance

Θ_{ja}= ---------- 0.34 °C/mW

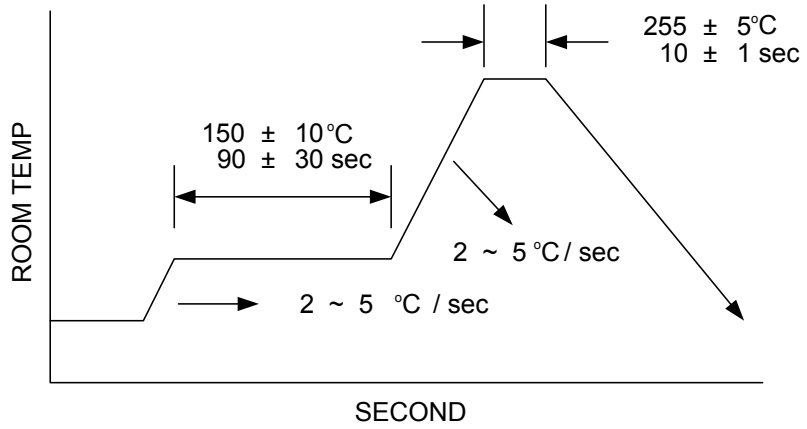
Θ_{jc}= ---------- 0.42 °C/mW

Operating Temperature Range ---------- -20 °C to 100 °C

Storage Temperature Range ---------- -65 °C to 150 °C

Junction Temperature ---------- +160 °C

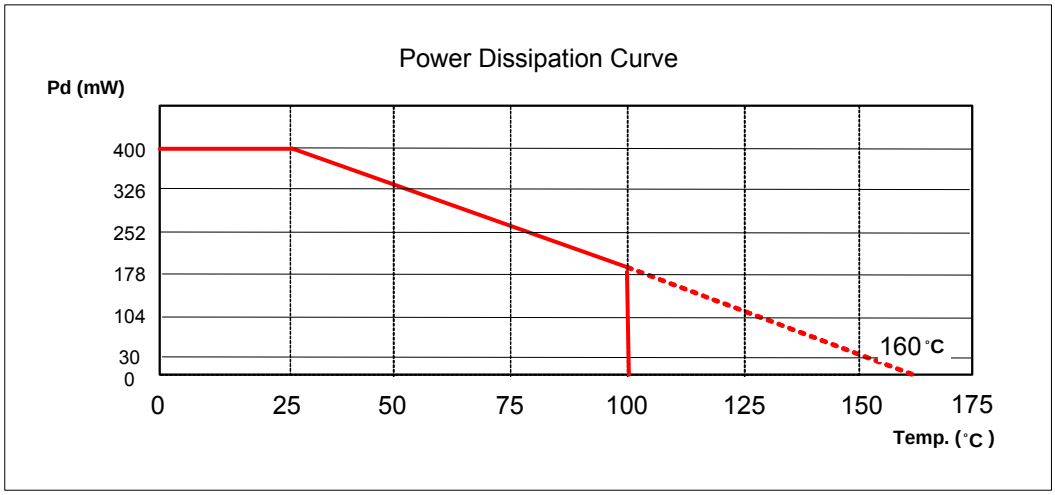
Lead Temperature (Soldering, 10 sec) ---------- +260 °C



Soldering Condition

DC Electrical Characteristics (at Ta = 25°C)

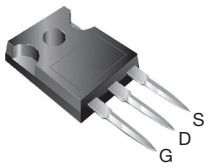
PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Operating Voltage	V_{CC}	No use pin is open (Fig1)	3.0	-	20.0	V
Supply current	I_{CC}	No use pin is open V_{cc} : 3.0V to 20V (Fig1)	-	4.2	10	mA
Output Saturation Voltage	V_{SAT}	Vcc=12V, Io = 20mA	-	165	200	mV
Output Rise time	(t_r)	RL=500ohm, CL=20pF (Fig1)	0.2	-	0.75	uS
Output Fall time	(t_f)	RL=500ohm, CL=20pF (Fig1)	20	-	150	nS
Note: Fig1 The IC output state is under N magnetic field.						

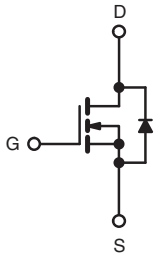


Power MOSFET

PRODUCT SUMMARY		
V_{DS} (V)	200	
$R_{DS(on)}$ (Ω)	V_{GS} = 10 V	0.085
Q_g (Max.) (nC)	140	
Q_{gs} (nC)	28	
Q_{gd} (nC)	74	
Configuration	Single	

TO-247AC





N-Channel MOSFET

FEATURES

- Dynamic dV/dt Rating
- Repetitive Avalanche Rated
- Isolated Central Mounting Hole
- Fast Switching
- Ease of Paralleling
- Simple Drive Requirements
- Compliant to RoHS Directive 2002/95/EC

Available
RoHS* COMPLIANT

DESCRIPTION

Third generation Power MOSFETs from Vishay provide the designer with the best combination of fast switching, ruggedized device design, low on-resistance and cost-effectiveness.

The TO-220AB package is universally preferred for commercial-industrial applications where higher power levels preclude the use of TO-220AB devices. The TO-247AC is similar but superior to the earlier TO-218 package because of its isolated mounting hole. It also provides greater creepage distance between pins to meet the requirements of most safety specifications.

ORDERING INFORMATION	
Package	TO-247AC
Lead (Pb)-free	IRFP250PbF
	SiHFP250-E3
SnPb	IRFP250
	SiHFP250

ABSOLUTE MAXIMUM RATINGS (T_C = 25 °C, unless otherwise noted)

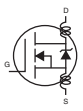
PARAMETER			SYMBOL	LIMIT	UNIT
Drain-Source Voltage			V_{DS}	200	V
Gate-Source Voltage			V_{GS}	± 20	
Continuous Drain Current	V_{GS} at 10 V	T_C = 25 °C	I_D	30	A
		T_C = 100 °C		19	
Pulsed Drain Current[a]			I_{DM}	120	
Linear Derating Factor				1.5	W/°C
Single Pulse Avalanche Energy[b]			E_{AS}	410	mJ
Repetitive Avalanche Current[a]			I_{AR}	30	A
Repetitive Avalanche Energy[a]			E_{AR}	19	mJ
Maximum Power Dissipation	T_C = 25 °C		P_D	190	W
Peak Diode Recovery dV/dt[c]			dV/dt	5.0	V/ns
Operating Junction and Storage Temperature Range			T_J, T_{stg}	- 55 to + 150	°C
Soldering Recommendations (Peak Temperature)	for 10 s			300[d]	
Mounting Torque	6-32 or M3 screw			10	lbf · in
				1.1	N · m

Notes

a. Repetitive rating; pulse width limited by maximum junction temperature (see fig. 11).
b. V_{DD} = 50 V, starting T_J = 25 °C, L = 683 µH, R_g = 25 Ω, I_{AS} = 30 A (see fig. 12).
c. I_{SD} ≤ 30 A, dI/dt ≤ 190 A/µs, V_{DD} ≤ V_{DS}, T_J ≤ 150 °C.
d. 1.6 mm from case.

* Pb containing terminations are not RoHS compliant, exemptions may apply

THERMAL RESISTANCE RATINGS				
PARAMETER	**SYMBOL**	**TYP.**	**MAX.**	**UNIT**
Maximum Junction-to-Ambient	R_{thJA}	-	40	°C/W
Case-to-Sink, Flat, Greased Surface	R_{thCS}	0.24	-	
Maximum Junction-to-Case (Drain)	R_{thJC}	-	0.65	

SPECIFICATIONS (T_J = 25 °C, unless otherwise noted)						
PARAMETER	**SYMBOL**	**TEST CONDITIONS**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
Static						
Drain-Source Breakdown Voltage	V_{DS}	V_{GS} = 0 V, I_D = 250 µA	200	-	-	V
V_{DS} Temperature Coefficient	$\Delta V_{DS}/T_J$	Reference to 25 °C, I_D = 1 mA	-	0.27	-	V/°C
Gate-Source Threshold Voltage	$V_{GS(th)}$	V_{DS} = V_{GS}, I_D = 250 µA	2.0	-	4.0	V
Gate-Source Leakage	I_{GSS}	V_{GS} = ± 20 V	-	-	± 100	nA
Zero Gate Voltage Drain Current	I_{DSS}	V_{DS} = 200 V, V_{GS} = 0 V	-	-	25	µA
		V_{DS} = 160 V, V_{GS} = 0 V, T_J = 125 °C	-	-	250	
Drain-Source On-State Resistance	$R_{DS(on)}$	V_{GS} = 10 V; I_D = 18 A[b]	-	-	0.085	Ω
Forward Transconductance	g_{fs}	V_{DS} = 50 V, I_D = 18 A	12	-	-	S
Dynamic						
Input Capacitance	C_{iss}	V_{GS} = 0 V, V_{DS} = 25 V, f = 1.0 MHz, see fig. 5	-	2800	-	pF
Output Capacitance	C_{oss}		-	780	-	
Reverse Transfer Capacitance	C_{rss}		-	250	-	
Total Gate Charge	Q_g	V_{GS} = 10 V; I_D = 30 A, V_{DS} = 160 V, see fig. 6 and 13[b]	-	-	140	nC
Gate-Source Charge	Q_{gs}		-	-	28	
Gate-Drain Charge	Q_{gd}		-	-	74	
Turn-On Delay Time	$t_{d(on)}$	V_{DD} = 100 V, I_D = 30 A, R_g = 6.2 Ω, R_D = 3.2 Ω, see fig. 10[b]	-	16	-	ns
Rise Time	t_r		-	86	-	
Turn-Off Delay Time	$t_{d(off)}$		-	70	-	
Fall Time	t_f		-	62	-	
Internal Drain Inductance	L_D	Between lead, 6 mm (0.25") from package and center of die contact 	-	5.0	-	nH
Internal Source Inductance	L_S		-	13	-	
Drain-Source Body Diode Characteristics						
Continuous Source-Drain Diode Current	I_S	MOSFET symbol showing the integral reverse p - n junction diode 	-	-	30	A
Pulsed Diode Forward Current[a]	I_{SM}		-	-	120	
Body Diode Voltage	V_{SD}	T_J = 25 °C, I_S = 30 A, V_{GS} = 0 V[b]	-	-	2.0	V
Body Diode Reverse Recovery Time	t_{rr}	T_J = 25 °C, I_F = 30 A, dI/dt = 100 A/µs	-	360	540	ns
Body Diode Reverse Recovery Charge	Q_{rr}		-	4.6	6.9	µC
Forward Turn-On Time	t_{on}	Intrinsic turn-on time is negligible (turn-on is dominated by L_S and L_D)				

Notes

a. Repetitive rating; pulse width limited by maximum junction temperature (see fig. 11).

b. Pulse width ≤ 300 µs; duty cycle ≤ 2 %.

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

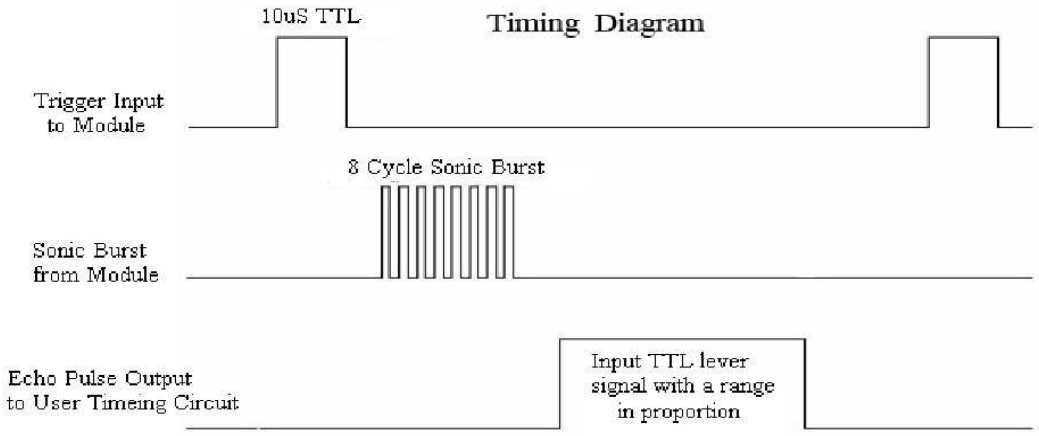
Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Apéndice C

Códigos

C.1. Movimientos Microcontrolador

```
void doblar_volante(int grados, int sentido)
{
  if (!(sentido))
  {
    analogWrite(PWM2, 210);
  }
  else
  {
    analogWrite(PWM2, 45);
  }
  distancia_temp[1] = 0;
  grados_volante_max = grados;
  flag_girar_volante = 1;

}

void centrar_volante()
{
  analogWrite(PWM2,50);
  delay(4000);
  analogWrite(PWM2, 127);
  doblar_volante(GIRO_MOV, 0);
  posicion_volante = VOLANTE_CENTRADO;
}

void mover(unsigned int distancia, double vlc, int sentido)
{
  if (!(sentido))
  {
    analogWrite(PWM1, 127 + vel_inicial);
    flag_back = 1;
  }
  else
  {
    analogWrite(PWM1, 127 - vel_inicial);
  }
  contador_movimiento = 0;
  distancia_max       = distancia;
  velocidad_ref       = vlc;
  sentido_temp        = sentido;
```

```
41   movimiento[0]        = vel_inicial;
42   distancia_temp[0]    = 0;
43   distancia_temp_d[0] = 0;
44   p_controller[0]      = 0;
45   d_controller[0]      = 0;
46   i_controller[0]      = 0;
47   prev_error[0]        = 0;
48   flag_mover           = 1;
49 }
50
51 void girar()
52 {
53   grados_a_rotar = grados_objetivo - valor_giro_total;
54
55   while (fabs(grados_a_rotar) > 180)
56   {
57     if(grados_a_rotar > 0)
58       grados_a_rotar = grados_a_rotar - 360;
59     else
60       grados_a_rotar = grados_a_rotar + 360;
61   }
62
63   if(grados_a_rotar <= 0)
64     sentido_giro = 1;
65   else
66     sentido_giro = 0;
67
68   if(fabs(grados_a_rotar) > GIRO_MIN)
69   {
70     if(fabs(grados_a_rotar) >= LIMITE_GIRO)
71     {
72       doblar_volante(GIRO_LIMITE, sentido_giro);
73       flag_rotacion = 1;
74     }
75     else
76     {
77       doblar_volante(GIRO_LEVE, sentido_giro);
78       flag_rotacion = 1;
79       flag_giro_leve = 1;
80     }
81   }
82 }
83
84 double pid_controller(int motor)
85 {
86   double error = 0;
87   double valor_instantaneo = 0;
88   double pid_contr = 0;
89
90   velocidad_temp = ((distancia_temp[motor] - distancia_temp_d[motor])*0.01)
       /DELTA_T;
91   error = velocidad_ref - velocidad_temp;
92   distancia_temp_d[motor] = distancia_temp[motor];
93
94   p_controller[motor]  = kp * error;
95   i_controller[motor] += ki * error * DELTA_T;
96   d_controller[motor]  = (kd * (error - prev_error[motor])) / DELTA_T;
97   prev_error[motor]    = error;
```

```
98    pid_contr       = p_controller[motor] + i_controller[motor] + d_controller
        [motor];
99    movimiento[motor]     += pid_contr;
100   if(movimiento[motor] > 110.0)
101       movimiento[motor] = 110.0;
102
103   return movimiento[motor];
104 }
```

C.2. Comunicación Microcontrolador

```
1  void send_uart(char* data_send, int respuestaid)
2  {
3    char formlist[150];
4    int data_len    = 0;
5    char caracter   = 0;
6    int header      = 0;
7    int iheader     = 0;
8    int PL          = 0;
9    int mod_header  = 0;
10   int mod_iheader = 0;
11   int size_l      = 0;
12   int size_h      = 0;
13
14   header = 160;
15   iheader = 64;
16   while (caracter != '!')
17   {
18     caracter = data_send[data_len];
19     data_len++;
20   }
21   data_len--;
22   PL = 16;
23   mod_header = header + PL + ((data_len & 0xf0000) >> 16);
24   mod_iheader = iheader;
25   size_l = data_len & 0xff;
26   size_h = (data_len & 0xff00) >> 8;
27   formlist[0] = mod_header;
28   formlist[1] = size_h;
29   formlist[2] = size_l;
30   formlist[3] = respuestaid;
31   formlist[4] = MICRO_ADDR;
32
33   for ( int i = 5 ; i < 5 + data_len ; i++)
34     formlist[i] = data_send[i - 5];
35   formlist[5 + data_len] = mod_iheader;
36
37   for (int i = 0 ; i < data_len + 6; i++)
38     Serial.write(formlist[i]);
39 }
40
41 void receive_uart()
42 {
43   SerRx = Serial.read();
44   if ( ( flag_uart == 0 ) && ( (SerRx & 0xE0) == 160) )
45   {
46     flag_uart++;
```

```
47      data_len_rx = (SerRx & 0x0F);
48    }
49    else if ( flag_uart == 1 )
50    {
51      data_len_rx = data_len_rx + (SerRx << 8);
52      flag_uart++;
53    }
54    else if ( flag_uart == 2 )
55    {
56      data_len_rx = data_len_rx + SerRx;
57      flag_uart++;
58    }
59    else if ( ( SerRx == MICRO_ADDR ) && ( flag_uart == 3 ) )
60    {
61      flag_uart++;
62    }
63    else if ( flag_uart == 4 )
64    {
65      data_rec[0] = SerRx;
66      flag_uart++;
67    }
68    else if ( (flag_uart > 4) && (flag_uart < 5 + data_len_rx) )
69    {
70      data_rec[flag_uart - 4] = SerRx;
71      flag_uart++;
72    }
73    else if ( (SerRx == 64) && ( flag_uart == (5 + data_len_rx) ) )
74    {
75      flag_uart = 0;
76      flag_accion = 1; //Ejecutamos la accion en el loop principal
77    }
78    else flag_uart = 0;
79
80  }
```

C.3. Interrupciones Microcontrolador

```
1  void ISR_Timer3()
2  {
3   if((digitalRead(ULTRA_ECHO)) && (ultr_flag_start == 0)
4       && (flag_ultr_request == 1))
5   {
6    ultr_start_time[0] = micros();
7    ultr_flag_start = 1;
8   }
9   else if ( (ultr_flag_start == 1) && (flag_ultr_request == 1)
10            && (digitalRead(ULTRA_ECHO) == 0))
11  {
12   ultr_distance[0] = (micros()-ultr_start_time[0])/58.0;
13   if(objeto_detectado == 0)
14     filtrar_datos_ultrasonido(0);
15   ultr_flag_start = 0;
16  }
17
18   if((digitalRead(ULTRB_ECHO)) && (ultr_flag_start == 0)
19       && (flag_ultr_request == 2))
20  {
```

```
21   ultr_start_time[1] = micros();
22   ultr_flag_start = 1;
23  }
24  else if ( (ultr_flag_start == 1) && (flag_ultr_request == 2)
25            && (digitalRead(ULTRB_ECHO) == 0))
26  {
27   ultr_distance[1] = (micros()-ultr_start_time[1])/58.0;
28   if(objeto_detectado == 0)
29     filtrar_datos_ultrasonido(1);
30   ultr_flag_start = 0;
31  }
32
33 if((digitalRead(ULTRC_ECHO)) && (ultr_flag_start == 0)
34     && (flag_ultr_request == 0))
35  {
36   ultr_start_time[2] = micros();
37   ultr_flag_start = 1;
38  }
39  else if ( (ultr_flag_start == 1) && (flag_ultr_request == 0)
40            && (digitalRead(ULTRC_ECHO) == 0))
41  {
42   ultr_distance[2] = (micros()-ultr_start_time[2])/58.0;
43   ultr_flag_start = 0;
44  }
45   if(((flag_objeto_detectado[0])||(flag_objeto_detectado[1]))
46      && (objeto_detectado == 0) && (flag_mover)
47      && (flag_back == 0) && (flag_girar_volante == 0))
48   {
49     objeto_detectado = MODO_ULTRASONIDO;
50     distancia_objeto[0] = distancia_objeto[0] + distancia_temp[0];
51     distancia_objeto[1] = distancia_objeto[1] + distancia_temp[0];
52   }
53
54
55 }
56 void ISR_Timer()
57 {
58   if((distancia_temp[1] >= grados_volante_max) && (flag_girar_volante == 1)
      )
59   {
60     analogWrite(PWM2, 127);
61     flag_girar_volante = 0;
62     distancia_temp[1] = 0;
63
64     if(completar_movimiento == 2)
65       completar_movimiento = 3;
66
67     if(enderezar_volante == 2 || esperar_volante == 1)
68       {
69         enderezar_volante = 0;
70         esperar_volante = 0;
71         send_uart('0 !', respuestaid_plan);
72       }
73   }
74
75   grados_por_rotar = grados_objetivo - valor_giro_total;
76
77   if (((fabs(grados_por_rotar) < GIRO_MIN) && (flag_rotacion == 1))
```

```
78            || completar_movimiento == 1 || enderezar_volante == 1)
79    {
80      if(completar_movimiento == 1)
81        completar_movimiento = 2;
82
83      if(enderezar_volante == 1)
84        enderezar_volante = 2;
85
86      if(flag_giro_leve)
87      {
88        doblar_volante(GIRO_LEVE, !sentido_giro);
89        flag_rotacion = 0;
90        flag_giro_leve = 0;
91      }
92      else
93      {
94        doblar_volante(GIRO_MOV, !sentido_giro);
95        flag_rotacion = 0;
96      }
97    }
98
99  contador_pid = (contador_pid + 1) %CONTROL_PERIOD;
100 if(contador_pid == 1)
101   flag_cntrl_vel = 1;
102
103 if ( ((distancia_temp[0] > distancia_max) && (flag_mover == 1))
104       || ((objeto_detectado == 1) && (flag_mover == 1)) )
105   {
106     analogWrite(PWM1, 127);
107
108     flag_mover = 0;
109     flag_cntrl_vel = 0;
110
111     if(((flag_rotacion == 1)
112     && (fabs(grados_objetivo - valor_giro_total) > GIRO_MIN_ITER))
113     || (objeto_detectado == 1))
114     {
115       if(objeto_detectado == 1)
116       {
117         if(flag_rotacion == 1)
118         {
119           completar_movimiento = 1;
120           objeto_detectado = 2;
121           flag_reversa_corta = 0;
122         }
123         else
124         {
125           completar_movimiento = 3;
126           objeto_detectado = 2;
127           flag_reversa_corta = 0;
128         }
129       }
130       else
131       {
132         if(fabs(grados_objetivo - valor_giro_total) <= LIMITE_REVERSA)
133           flag_reversa_corta = 1;
134         else
135           flag_reversa_corta = 0;
```

```
136
137           completar_movimiento = 1;
138         }
139       }
140       else if(flag_rotacion == 1)
141       {
142         enderezar_volante = 1;
143       }
144       else if(completar_movimiento == 0)
145       {
146         if(flag_girar_volante == 1)
147           esperar_volante = 1;
148         else
149           send_uart('0 !', respuestaid_plan);
150       }
151
152       if(completar_movimiento == 4)
153       {
154         if(objeto_detectado == 2)
155         {
156           completar_movimiento = 0;
157           objeto_detectado = 3;
158         }
159         else
160         {
161           completar_movimiento = 5;
162         }
163       }
164       if(flag_back)
165         flag_back = 0;
166       distancia_temp[0] = 0;
167     }
168   contador_ultrasonido = (contador_ultrasonido + 1) %ULTR_PERIOD;
169   flag_ultrasonido = (contador_ultrasonido == 1);
170   flag_timer = 1;
171 }
172
173
174 void ISR_INTE0()
175 {
176   if (flag_mover)
177     distancia_temp[0] = distancia_temp[0] + DIST_PULS;
178   distancia_abs = distancia_abs + DIST_PULS;
179 }
180
181 void ISR_INTE1()
182 {
183   if (flag_girar_volante)
184   {
185     distancia_temp[1] = distancia_temp[1] + PASO_VOLANTE;
186     if(sentido_giro)
187       posicion_volante++;
188     else
189       posicion_volante--;
190   }
191 }
```

Apéndice D

Anexos del Proyecto Integrador

D.1. Solicitud de Aprobación de Tema

Facultad de Ciencias Exactas, Físicas y Naturales
AREA INGENIERÍA
ESCUELA DE ELECTRONICA
C.C. 755 - Correo Central - 5000 - CÓRDOBA
Tel. Directo (0351) 33-4147 int 110
Conmutador: 433-4141 y 33-4152 - Interno 10

Sr. Director de la Escuela de Ingeniería Electrónica
Ing.: RODRIGO BRUNI

Me dirijo a Ud. a fin de solicitar la **aprobación del tema del *Proyecto Integrador*** (***PI***) que propongo a continuación:

TEMA

NOMBRE DEL PROYECTO: SISTEMA DE CONTROL DE NAVEGACIÓN APLICADO A VEHÍCULO AUTÓNOMO

DESCRIPCION: Diseño, desarrollo y aplicación de un sistema de control de navegación autónoma para un vehículo a escala. (Ver descripción detallada en Anexo).

DESARROLLO DE PROTOTIPO: Se desarrollará el prototipo del vehículo autónomo. Se comprará la estructura de un vehículo sobre la cual irá montado el sistema desarrollado en el PI.

AREA TEMATICA DEL PI: Control, Digitales.

ASIGNATURAS: Electrónica Digital III, Sistemas de Control II, Electrónica Industrial, Informática Avanzada.

Director de PI

Nombre: Mario Rafael Hueda
Cargo: Profesor Titular de Señales y Sistemas Lineales
Dirección Personal: Huiliches – Casa 40 – B° Portales del Sur – Córdoba
Dirección Laboral: Av. Velez Sarsfield 1650 - Córdoba
TE: 0351-156812027
E-mail: mario.hueda@unc.edu.ar
Firma del Director / Co Director: .....

Co-Director de PI

Nombre: Enrique Mariano Lizárraga
Cargo: Profesor Titular Cátedra Teoría de Redes
Dirección Personal: Ovidio Lagos 291 3ro. A – Córdoba – CP 5000
Dirección Laboral: Av. Velez Sarsfield 1611 – Córdoba – CP 5000
TE: 0351-5353800 int 29085
E-mail: emlizarraga@unc.edu.ar
Firma del Co-Director: .....

Datos del Estudiante

Nombre y Apellido: Leandro Ezequiel Borgnino.
Matrícula: 36887290
Materias que faltan aprobar: DSP y Gestión de las Organizaciones Industriales.
Dirección: General Paz 911.
Localidad: Rafaela **Provincia**: Santa Fe
E-mail: leo.borgnino@gmail.com
Teléfono 03492-15653350

Firma:.....

Datos del Estudiante

Nombre y Apellido: Maximiliano David Armesto
Matrícula: 37315229
Materias que faltan aprobar: Ninguna
Dirección: Graham Bell 2560
Localidad: Córdoba **Provincia**: Córdoba
E-mail: maxifcefyn@gmail.com
Teléfono 0351-158147894

Firma:.....

Objetivo General: desarrollar un sistema de navegación autónomo para ser aplicado en diversos vehículos con diferentes tipos de aplicaciones. Se llevará a cabo el diseño y construcción de los componentes de hardware y software necesarios para lograr el sistema deseado y se lo aplicará en un auto a escala.

Objetivos Específicos: ver Anexo.

Antecedentes de Proyectos similares: No se encontraron proyectos similares desarrollados dentro de la facultad. Lo más cercano que se encontró es un sistema de detección de objetos, cabe aclarar que nuestro proyecto posee detección dinámica de objetos pero no se realiza con la misma metodología que la encontrada en el siguiente proyecto:

- "Detección de Obstáculos con Imágenes Estereoscópicas para Navegación Centralizada de Unidad Móvil", UNC-FCEFYN.

Duración y Fases de las tareas previstas: ver diagrama de Gantt en Anexo.

Metodología

Lugar previsto de realización: El desarrollo de proyecto integrador se llevará a cabo en la Fundación Tarpuy ubicada en calle Romagosa 518 – B° Colinas de Velez Sarsfield.

Requerimiento de Instrumental y equipos:

- Instrumental de medición: Multímetro Digital, Osciloscopio, Analizador Lógico.
- Generador de funciones.
- Microprocesador y Microcontrolador.
- PC para programación y testeo del sistema.
- Diversos sensores y componentes electrónicos necesarios para la realización del sistema (acelerómetro, giróscopo, encoders, componentes electrónicos en general).
- Estructura de vehículo a escala.

Inversión estimativa prevista por el alumno: $10.000

Apoyo Económico externo a la Facultad: se recibirán aportes económicos por parte de la Fundación Tarpuy-Fundación Fulgor.

Referencias Bibliográficas o de Software:

"Artificial Intelligence: A Systems Approach", M. Tim Jones, Computer Science Series.

"Artificial Intelligence for Robotics", https://classroom.udacity.com/courses/cs373.

"IDE de Arduino", " https://www.arduino.cc/en/Main/Software".

"Sistema Operativo Raspbian", https://www.raspberrypi.org/downloads/.

"Python 2.7", https://www.python.org/downloads/.

Recibido Cátedra PI

.....

Firma

Córdoba, / / .

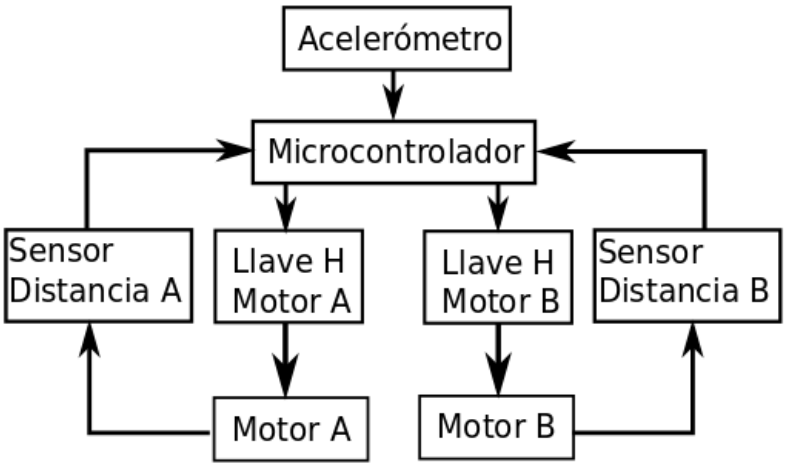
ANEXO

Motivación:

La realización del proyecto se llevará a cabo debido a que no se han encontrado desarrollos significativos de sistemas de navegación autónomos en nuestro país, siendo un área mundialmente en auge y de gran interés por parte de los estudiantes solicitantes.

Descripción Detallada del Proyecto:

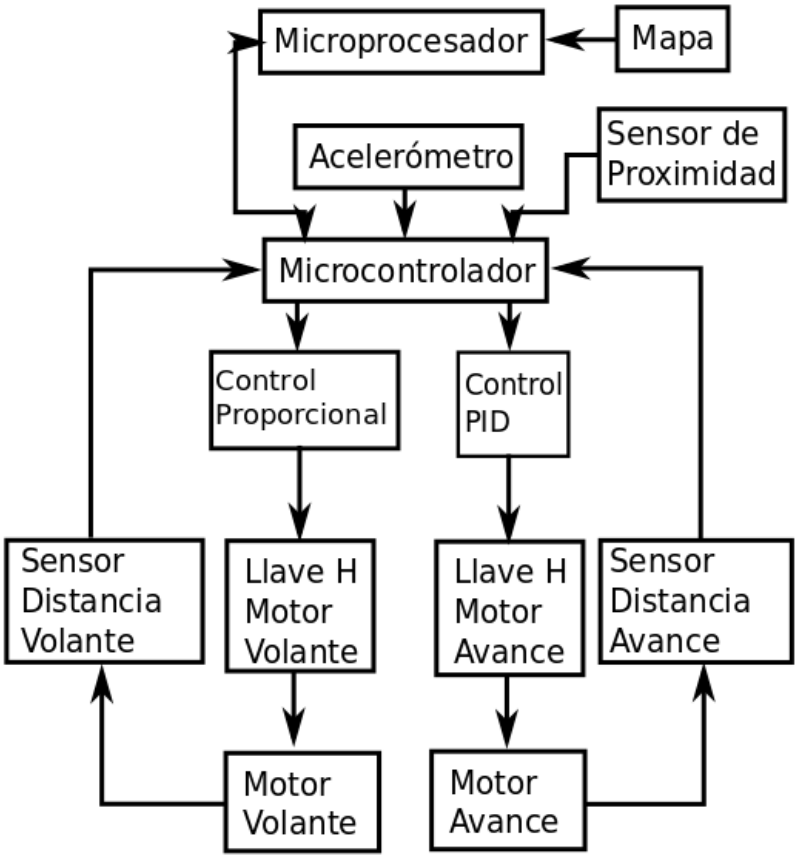
Se diseñará y desarrollará tanto el hardware como el software de un sistema para navegación autónoma de vehículos. Además, este sistema será aplicado a un automóvil a escala. El proyecto es continuación de nuestra PPS cuyo diagrama en bloques se muestra a continuación:



Como se puede observar, el sistema de control se centraba en un microcontrolador (específicamente en la PPS se utilizó un Arduino MEGA) el cual le enviaba señales de PWM a las llaves H que controlaban la potencia y el sentido de los motores. A su vez, las ruedas del vehículo poseían discos con imanes, y, enfrentados a los discos sensores de efecto Hall que enviaban pulsos al Arduino a partir de los cuales se calculaba la distancia recorrida por el vehículo. El acelerómetro utilizado fue el MPU-6050, que cuenta con giróscopo, y del cual se extraía el dato del giro en Z para poder hacer rotaciones precisas. La estructura fue un prototipo de hierro soldado, con una plancha de acrílico en su parte superior para poder apoyar todos los componentes electrónicos, este prototipo fue realizado por los estudiantes. Se implementó para este sistema, un software que le indicaba al vehículo una cierta distancia a recorrer con un determinado sentido (por ejemplo 30cm. hacia adelante) y ciertas rotaciones a realizar hacia un lado u otro (por ejemplo 90° hacia la derecha). Cabe destacar que las rotaciones eran siempre de 90° y se realizaban en el lugar

con el sistema de un tanque de guerra, es decir, haciendo girar ambos motores en sentido contrario (las ruedas no doblan hacia un lado u otro).

Se presenta a continuación el diagrama en bloques que representa lo que se desarrollará en el presente Proyecto Integrador:



La estructura realizada en la PPS fue una estructura rápidamente diseñada para poder aplicar el sistema desarrollado, y la misma poseía falencias como rozamientos de las ruedas con la estructura, alto peso del vehículo lo cual dificultaba el movimiento, limitación para hacer trayectorias suaves, mala estética. Para la realización del Proyecto Integrador se decidió comprar un automóvil a escala para evitar todos los problemas anteriormente nombrados. La nueva estructura deberá ser más liviana, poseer motores propios (cuyas características estén seleccionadas específicamente para la estructura en cuestión), las ruedas delanteras deberán poder doblar (para poder realizar trayectorias suaves). Cambiar los motores implicará evaluar si las llaves H desarrolladas en la PPS se pueden reutilizar y, en caso negativo, rediseñarlas ajustando los cálculos a las características de los nuevos motores. Una de las llave H controlará el motor de avance y la otra el motor de giro horizontal de las ruedas delanteras. Para los sensores de distancia se utilizará la misma tecnología pero habrá que readaptarla a la nueva estructura, y se deberá contar con un

sensor similar que calcule los grados de giro de las ruedas delanteras. El acelerómetro se utilizará del mismo modo que en la PPS con la diferencia que ahora se rotarán ángulos distintos de 90°. El microcontrolador se encargará de tomar los datos de todos los sensores y controlar los motores del vehículo. Se desarrollará un control PID de velocidad para el motor de avance y un control proporcional para la rotación de las ruedas.

Todo el procesamiento de datos para lograr la autonomía del vehículo se hará a través de un microprocesador. El mismo tomará un mapa con punto de inicio, punto de llegada y obstáculos y deberá calcular la trayectoria más corta entre estos dos puntos esquivando los obstáculos. Una vez calculada la trayectoria se determinarán las acciones necesarias a realizar por el vehículo para poder recorrer la misma. Estas acciones serán enviadas al microcontrolador para que las ejecute. El microcontrolador deberá poder comunicarse con el microprocesador para avisar cuando las acciones hayan sido ejecutadas. Para lograr una comunicación estable entre ambos dispositivos se desarrollará un protocolo de comunicación. Finalmente se incorporarán sensores de proximidad para poder detectar objetos en forma dinámica y agregarlos al mapa original en caso que no estuvieran registrados en él.

Objetivos Específicos:

- Evaluar, seleccionar y adquirir la estructura, microcontrolador, microprocesador y sensores de proximidad adecuados para la implementación del sistema de navegación autónoma.
- Adaptar el sistema de sensado de distancia recorrida utilizado en la PPS para medir la distancia recorrida y la rotación del volante de la nueva estructura.
- Adaptar el software del microcontrolador desarrollado en la PPS para controlar el movimiento de la nueva estructura y adquirir los datos de los diferentes sensores.
- Evaluar, diseñar y construir placas de potencia necesarias para controlar los motores del vehículo. Desarrollar los algoritmos de control para los motores.
- Desarrollar el software con el cual el microprocesador hará todo el procesamiento de alto nivel.
- Diseñar e implementar un protocolo de comunicación entre el microcontrolador y microprocesador para lograr una comunicación libre de errores.

Breve descripción general de los materiales que se usarán según el diagrama en bloques presentado:

- Microcontrolador: se utilizará un Arduino Mega. El motivo de selección del mismo es por la gran disponibilidad de sensores compatibles que hay con esta plataforma. Solo se encargará del control a bajo nivel, es decir, órdenes de movimiento a los motores, sensado de datos y el envío de los mismos al microprocesador.
- Microprocesador: se utilizará una Raspberry Py. El motivo de selección es por la gran capacidad de procesamiento que posee. Se encargará de hacer el procesamiento a alto nivel, el cual incluye procesamiento de datos, cálculo de planning, procesamiento de los mapas. Para cumplir estas funcionalidades, el lenguaje más adecuado nos pareció Python y Raspberry Py nos permitía la programación en este lenguaje.
- Acelerómetro: se utilizará un MPU6050 de 6 ejes. El mismo posee acelerómetro y giroscopio. Se seleccionó por contar con giroscopio el cual nos sirve para medir los ángulos rotados por el vehículo. La librería que se utiliza para el sensado del mismo es de desarrollo propio.
- El control proporcional y PID se realizarán desde el Arduino con algoritmos propios.
- Las llaves H son de diseño propio y se mandaron a fabricar a Aries. Las mismas utilizan transistores MOSFET e integrados IR2110.
- La estructura utilizada es un auto Mini Cooper a escala, del cual solo se utiliza la estructura y los motores. Todas las placas originales han sido extraídas y se utilizarán nuestras propias placas y sensores.
- Los motores son los que trae la estructura, son motores de corriente continua 6v 5A, no poseen marca ni hoja de datos.
- Los sensores para medir distancia serán encoders de fabricación propia. Se utilizarán imanes pegados a la rueda con un sensor de efecto hall que enviará al microcontrolador los pulsos correspondientes al movimiento de la rueda. La misma lógica se usa para calcular la posición del motor del volante.
- Para medir proximidad se utilizarán sensores ultrasónicos Jsn Sr04. Son los que vienen en los kit de sensores de estacionamiento para automóviles. Se seleccionó estos sensores por la forma de empotrarlos al vehículo y la buena estética que le da al mismo.

Diagrama de Gantt:

Acciones a Realizar	Tiempo estimado en semanas	Meses / Semanas																								
		1				2				3				4				5				6				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Selección estructura adecuada.	1																									
Selección Microcontrolador. (Evaluar el utilizado en PPS).	1																									
Selección Microprocesador.	1																									
Adaptar el sistema de sensado de distancia recorrida por el vehículo desarrollado en PPS.	3																									
Diseñar y construir el sistema de sensado de rotación para el volante del vehículo.	3																									
Selección sensor acelerómetro y giróscopo e implementación de software (Evaluar lo que se puede reutilizar de PPS).	2																									
Selección sensor de proximidad e implementación de software.	2																									
Estudiar, diseñar y construir placas de potencia necesarias para controlar los motores del vehículo. (Evaluar las desarrolladas en la PPS).	2																									
Adaptar el software de control de movimiento del vehículo desarrollado en la PPS para la nueva estructura.	3																									
Desarrollar un control PID de velocidad para el vehículo.	3																									
Desarrollar un control de rotación proporcional para las ruedas delanteras.	3																									
Diseñar el protocolo de comunicación entre el microcontrolador y microprocesador.	2																									
Analizar diferentes lenguajes de programación para procesamiento de alto nivel y seleccionar el más adecuado para la aplicación.	1																									
Diseñar un algoritmo para procesar el mapa.	1																									
Diseñar e implementar un algoritmo que establezca la trayectoria más corta entre el punto de inicio y el punto de llegada evitando los obstáculos y establezca los movimientos del vehículo necesarios para lograrla.	4																									
Software para la toma de decisiones ante objetos detectados por los sensores de proximidad.	4																									
Documentación	17																									

D.2. Informe de Avances 1

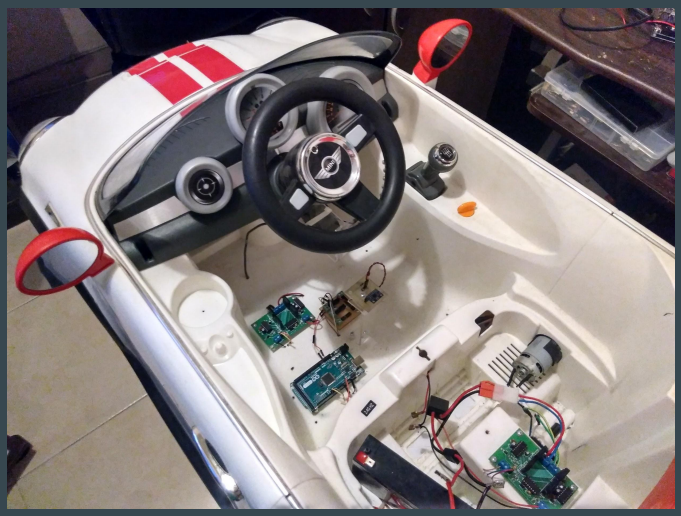
"SISTEMA DE CONTROL DE NAVEGACIÓN APLICADO A VEHÍCULO AUTÓNOMO"

Director: Dr. Mario Hueda
Co-Director: Ing. Mariano Lizárraga.

Estudiantes:
Armesto Maximiliano
Borgnino Leandro

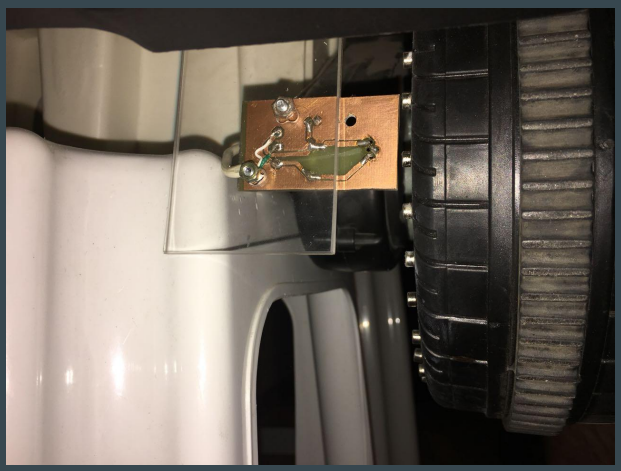
Avances al 30/03/2018

- Se adquirió la nueva estructura. La misma consiste en un vehículo eléctrico a escala para niños.
- El mismo cuenta con un motor para el avance y otro motor para girar el volante.



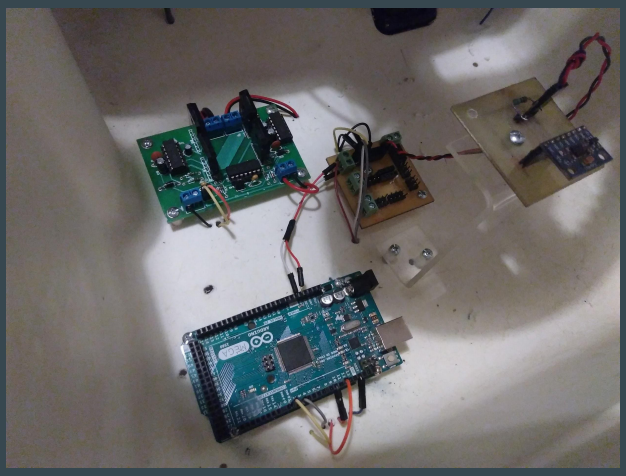
Avances al 30/03/2018

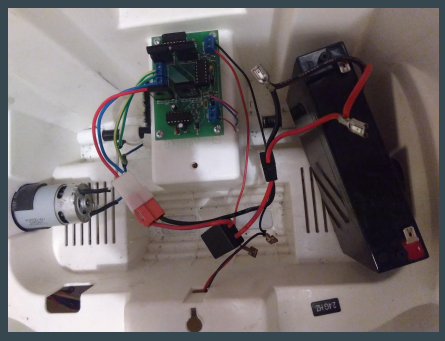
- Se instalaron los sensores de efecto Hall en ambos motores para poder medir avance en centímetros y grados de rotación del volante.
- Se recalcularon las llave H ya que las características de los motores (6V 5A) diferían de los de la estructura anterior (la utilizada en la PPS). Las mismas se mandaron a fabricar.



Avances al 30/03/2018

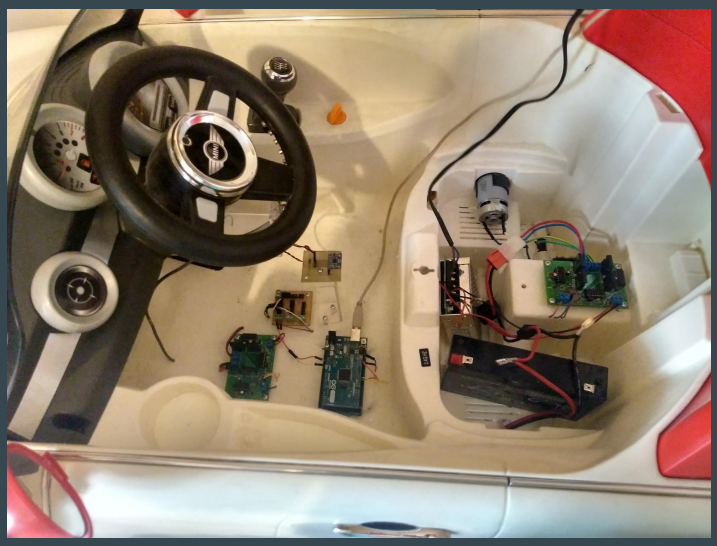
- Se fabricó una placa para alimentar todos los sensores utilizados.
- Se agregó a la estructura el sensor acelerómetro-giróscopo seleccionado.





Avances al 30/03/2018

- Una vez instalado todo el hardware se comenzaron a hacer pruebas individuales de cada uno de los sensores.
- Se crearon las funciones de control de bajo nivel para el vehículo. Las mismas incluyen función de avance, rotación del volante y del vehículo, centrado del vehículo en línea recta.



Avances al 30/03/2018

- Luego de tener todo el hardware funcionando se realizó el fijado de todas las placas y se pasaron los cables por debajo del vehículo.
- Se seleccionaron los sensores de ultrasonido para la detección de objetos y se adaptó la estructura para colocarlos.

Avances al 30/03/2018

- Se probó todo el código en arduino con las nuevas funciones de control de movimiento diseñadas.
- Se diseñó un protocolo de comunicación entre el Arduino y la Raspberry Pi y se comenzaron a enviar los primeros datos.
- Se comenzaron a redactar la introducción y el marco teórico del informe.

D.3. Informe de Avances 2

"SISTEMA DE CONTROL DE NAVEGACIÓN APLICADO A VEHÍCULO AUTÓNOMO"

Director: Dr. Mario Hueda
Co-Director: Dr. Mariano Lizárraga

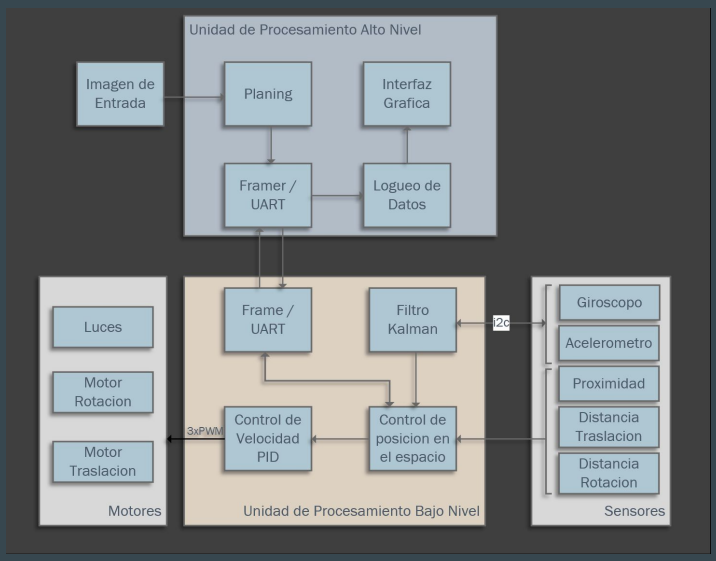
Estudiantes:
Armesto Maximiliano
Borgnino Leandro

Avances al 07/05/2018

- Se desarrolló el código de alto nivel, implementado en el microprocesador Raspberry Pi. El código incluye:

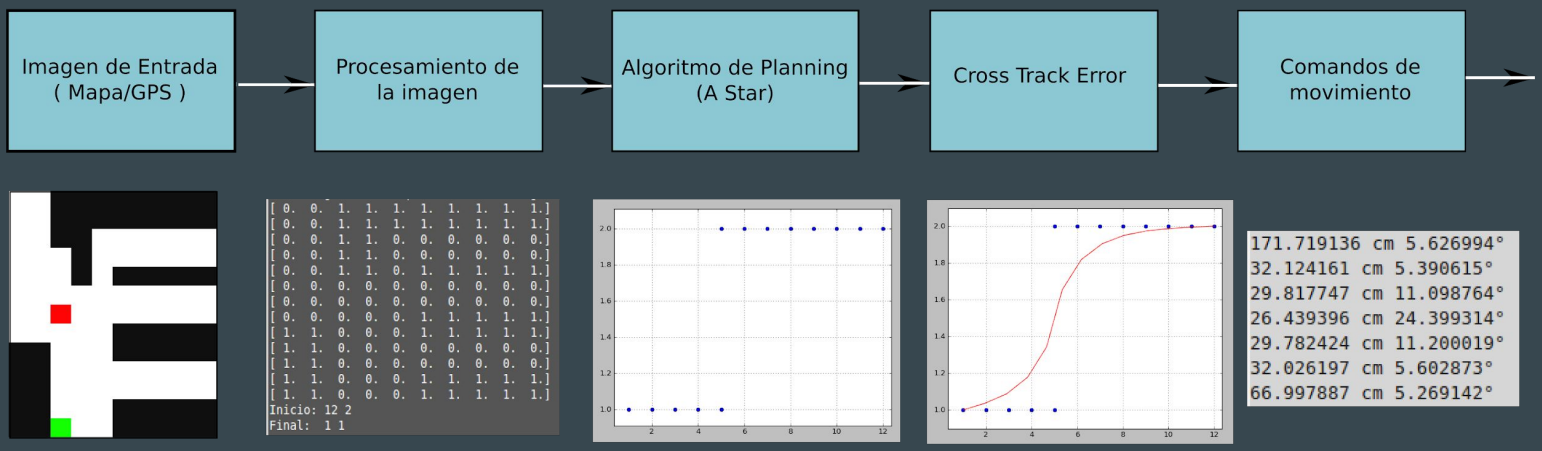
  ➔ Funciones para procesamiento de imágenes (procesamiento del mapa de entrada al sistema).

  ➔ Implementación del algoritmo de Planning A-STAR que calcula la trayectoria más corta entre los puntos de inicio y de fin esquivando los obstáculos, todo esto detallado en el mapa.

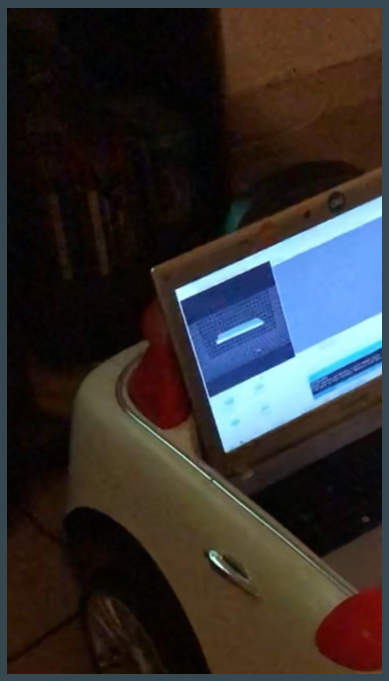


Avances al 07/05/2018

- ➔ Implementación del algoritmo Cross Track Error, el cual se utiliza para suavizar la trayectoria calculada por el algoritmo de Planning, permitiendo que el vehículo se desplace realizando movimientos suaves.
- ➔ Implementación de protocolo de comunicación UART propio para la transferencia de datos con el microcontrolador (Arduino).



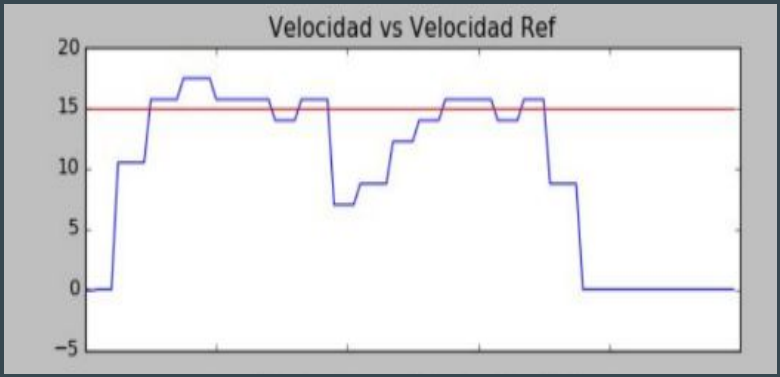
Avances al 07/05/2018



- ➔ Gráfica 3D de la inclinación del vehículo en tiempo Real.
- ➔ Implementación gráfica de Radar para representar en él los datos provenientes de los sensores de proximidad.

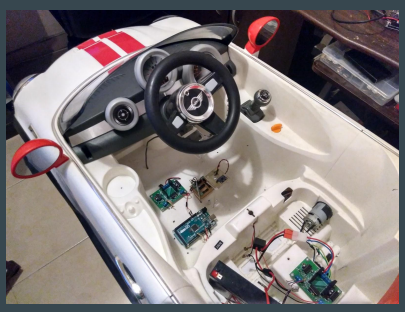
Avances al 07/05/2018

- Se implementó el control de velocidad PID para el vehículo, las constantes Kp, Ki, Kd por el momento se han calculado en forma empírica. Está en proceso el desarrollo teórico.



Avances al 07/05/2018

- Una vez realizado el código de alto nivel se comenzaron las pruebas de todo el sistema funcionando en conjunto, esto es:
  - El microprocesador haciendo los cálculos pertinentes al trazado de la trayectoria y el microcontrolador encargándose de manejar los actuadores y de tomar los datos de los sensores para poder efectuar los desplazamientos.
  - Se testeó todo el sistema con diferentes trayectorias, puntos iniciales y velocidades. Gracias a estas pruebas fueron apareciendo bugs en todo el sistema los cuales se fueron corrigiendo.



Avances al 07/05/2018

- Se loguearon las variables de velocidad y posición del vehículo a lo largo de diferentes recorridos para corroborar el correcto funcionamiento de las funciones de autocentrado del vehículo en línea recta y del control de velocidad PID.
- Se finalizó con la escritura de la introducción del informe y se está escribiendo el Marco Teórico.

D.4. Nota de Aprobación Final del Director

UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

ESCUELA DE INGENIERÍA ELECTRÓNICA

Quien suscribe el Profesor Hueda, Mario R. en su carácter de Director del Proyecto Integrador de los Estudiantes Armesto, Maximiliano David y Borgnino, Leandro Ezequiel, denominado: SISTEMA DE CONTROL DE NAVEGACIÓN APLICADO A VEHÍCULO AUTÓNOMO, considera que el desarrollo del trabajo se ha completado según lo especificado en la Solicitud de Aprobación de Tema y se encuentra en condiciones de tramitar su defensa.

A los efectos de quien corresponda, en fecha ...../...../2018.

Firma y aclaración del Director