GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

NAVEGACIÓN AUTÓNOMA DE UN VEHÍCULO TERRESTRE BASADA EN UN LIDAR 2D

Autor: Javier Andersen Muñoz

Director: Juan Luis Zamora Macho

Madrid

Agosto 2022

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Navegación autónoma de un vehículo terrestre basada en un LIDAR 2D

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2021/22 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Javier Andersen Muñoz Fecha: 27/08/2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Juan Luis Zamora Macho Fecha: 30/ 08/ 2022

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

NAVEGACIÓN AUTÓNOMA DE UN VEHÍCULO TERRESTRE BASADA EN UN LIDAR 2D

Autor: Javier Andersen Muñoz

Director: Juan Luis Zamora Macho

Madrid

Agosto 2022

Agradecimientos

A todos mis amigos, tanto a los que he conocido durante la carrera como los que conocía de antes, por el apoyo que me han dado en todo el camino. Especial agradecimiento a Juan Luis Zamora, por siempre estar dispuesto a ayudar en todo momento y a resolver cualquier duda que surgiera durante el proyecto.

NAVEGACIÓN AUTÓNOMA DE UN VEHÍCULO TERRESTRE BASADA EN UN LIDAR 2D

Autor: Andersen Muñoz, Javier.
Director: Zamora Macho, Juan Luis.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

El objetivo de este proyecto es el desarrollo de un control de seguimiento de pared de un vehículo terrestre a partir de las medidas obtenidas por un sensor LiDAR 2D. Para ello se dispone de un vehículo de tracción diferencial equipado con diversos sensores, entre ellos un LiDAR, así como de dos Raspberry Pi modelo 3B+ para el procesamiento de los datos del mismo y desarrollo de la algoritmia necesaria para implantar el control.

Palabras clave: Navegación autónoma, LiDAR, Python, Matlab, Simulink, Raspberry Pi, ROS 2.

1. Introducción

Desde la automatización de los primeros procesos, el desarrollo y progreso de los mismos no ha dejado de aumentar en los años sucesivos. La llegada de los autómatas programables en la década de 1960 [1] propició la mejora de los procesos industriales en multitud de áreas. Su creación, así como mejoras futuras dio paso a la llegada de los microprocesadores al mercado, introduciendo Intel el 4004 en 1971 [2]. La introducción al mercado de estos dispositivos supuso una revolución para la época ya que abrió las puertas a una amplia gama de aplicaciones, destacando los procesos de automatización [3], así como la mejora del nivel de confort y experiencia de los usuarios en ciertas labores cotidianas.

Entre las áreas que avanzaron en el camino de la automatización, cabe destacar el mundo del automóvil, buscando desde finales del siglo XX el objetivo de lograr introducir al mercado vehículos autónomos. En el marco de este contexto, se introduce este proyecto, buscando la automatización de un vehículo terrestre de tracción diferencial en un entorno controlado de interior mediante el uso de sensores, centrándose el proyecto en el uso del LiDAR 2D para lograr dicho objetivo.

2. Definición del proyecto

Para la realización del proyecto se pueden marcar una serie de objetivos claros.

- Desarrollar el algoritmo necesario para la obtención de las medidas de ángulo y distancia del sensor LiDAR 2D modelo A2M8 de SLAMTEC.
- Gestionar la transmisión de información entre las dos Raspberry Pi modelo 3B+ mediante protocolo TCP/IP por cable Ethernet, para su posterior tratamiento y procesado.
- Desarrollar las técnicas de control requeridas para lograr un control de seguimiento de pared robusto.

3. Descripción del hardware y software empleado

En cuanto al hardware empleado para el proyecto, destaca el uso de un vehículo terrestre de tracción diferencial. Dicho vehículo viene equipado con diversos sensores, entre ellos una IMU (Inertial Measurement Unit), dos motores de corriente continua, pulsadores, luces LED para indicar el estado en que se encuentra el vehículo, así como un sensor RPLIDAR A2M8 del distribuidor SLAMTEC. Cuenta además con dos Raspberry Pi modelo 3B+, una de ellas encargada de gestionar la toma de muestras del LiDAR y el envío de los datos a la otra Raspberry, teniendo esta última como función principal el procesado de la información extraída por el sensor. En la Ilustración 1 se muestra el vehículo empleado durante el proyecto.

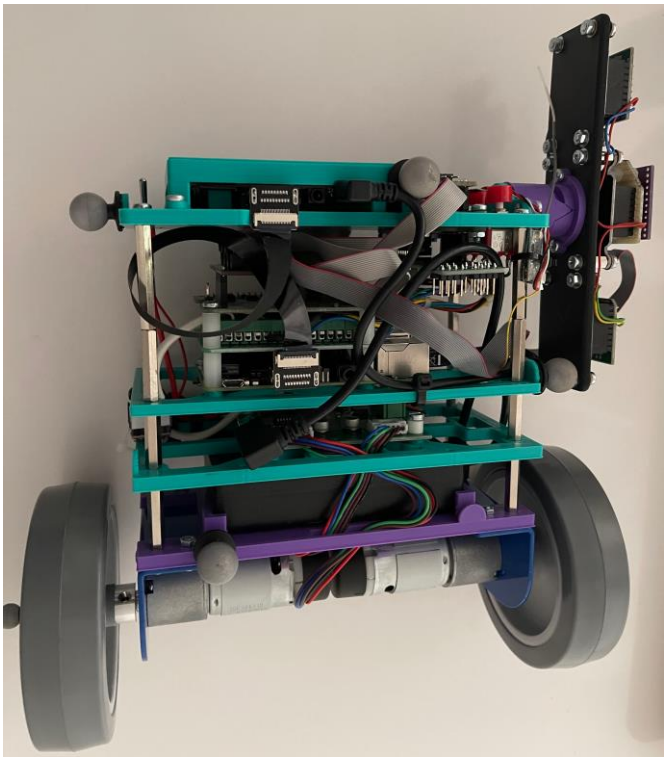


Ilustración 1. Vehículo de tracción diferencial empleado en el proyecto.

Por su parte, el software empleado puede dividirse en dos grupos fundamentales, el empleado para la obtención de medidas y el usado para su posterior procesado y desarrollo del control de seguimiento de pared.

Para la primera parte, es decir, la toma de medidas del LiDAR 2D, se ha empleado el lenguaje de programación Python, debido a la existencia de librerías compatibles con el modelo de LiDAR empleado en el proyecto. Mediante el código desarrollado, es posible activar el motor del sensor y comenzar la toma de muestras. Además, es el encargado de realizar la conversión de los datos a un formato compatible para su transmisión por protocolo TCP/IP, entrando esto último en las labores realizadas por este script.

Para el procesado de la información, así como el desarrollo del algoritmo de control de seguimiento de pared, se ha empleado Matlab-Simulink. Mediante el uso de ambos, es

posible coordinar las diferentes tareas del vehículo de tracción diferencial, así como modificar diferentes parámetros del sistema para lograr un control robusto y eficaz.

4. Resultados

Los resultados obtenidos en las diferentes pruebas realizadas durante el proyecto se pueden dividir en dos grupos.

En primer lugar, los resultados relativos a la calidad de las medidas del LiDAR. Se optó por realizar un mapeado del entorno con el vehículo estático. Los resultados obtenidos se presentan en la Ilustración 2.

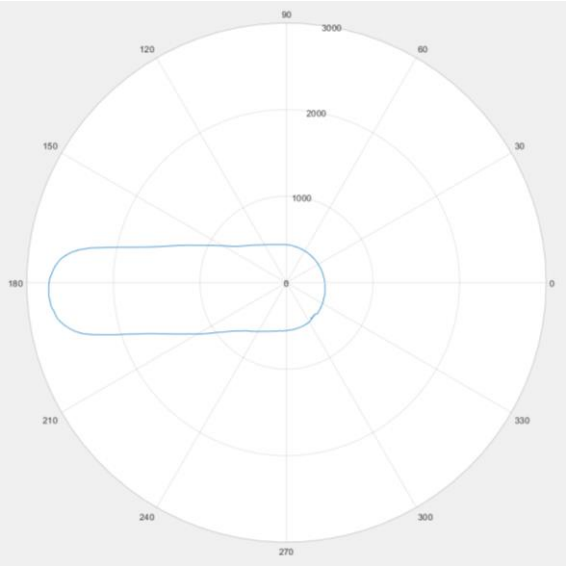


Ilustración 2. Mapa del entorno del vehículo.

Por último, se engloban los resultados obtenidos de la monitorización del control de seguimiento de pared, mostrando la progresión de diferentes variables de interés relativas al control. Los resultados se presentan en la Ilustración 3.

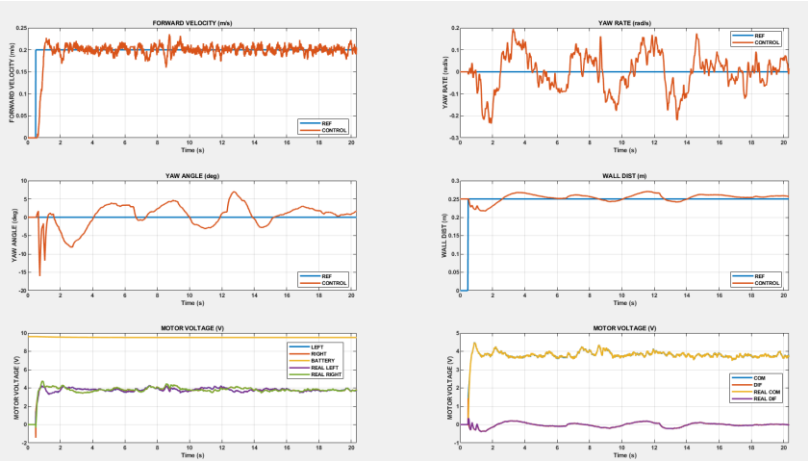


Ilustración 3. Ensayo de control de seguimiento de pared.

5. Conclusiones

Tras la realización de los diferentes ensayos, así como selección de los parámetros adecuados para las labores de control, se ha conseguido cumplir con los objetivos propuestos para este proyecto.

El control de seguimiento de pared implementado, es robusto, permitiendo su replicación en diversos entornos controlados y pudiendo modificar la referencia de distancia a la pared cercana.

En el caso de proyectos futuros, se aconseja el uso de ROS 2 (Robotic Operative System) para el desarrollo del mismo, permitiendo el acceso a diversas funcionalidades desarrolladas por la comunidad, así como una fácil implementación de las mismas. Se pueden explorar otras vías de navegación autónoma como mapeado del entorno mediante el algoritmo SLAM (Simultaneous Localization and Mapping).

6. Referencias

[1] J. Balcells y J. L. Romeral, Autómatas Programables, Barcelona: Marcombo, 1997.

[2] W. Aspray, «The Intel 4004 microprocessor: what constituted invention?,» *IEEE Annals of the History of Computing,* vol. 19, nº 3, pp. 4-15, 1997.

[3] T. Sauter, S. Soucek, W. Kastner y D. Dietrich, «he Evolution of Factory and Building Automation,» *IEEE Industrial Electronics Magazine,* vol. 5, nº 3, pp. 35-48, 2011.

AUTONOMOUS NAVIGATION OF A LAND VEHICLE BASED ON A 2D LIDAR

Author: Andersen Muñoz, Javier.
Supervisor: Zamora Macho, Juan Luis.
Collaborating Entity: ICAI – Universidad Pontificia Comillas.

ABSTRACT

The objective of this project is the development of a wall tracking control of a land vehicle from the measurements obtained by a 2D LiDAR sensor. To do this, a differential traction vehicle equipped with various sensors is available, including a LiDAR, as well as two Raspberry Pi model 3B+ for data processing and development of the necessary algorithms to implement the control.

Keywords: Autonomous navigation, LiDAR, Python, Matlab, Simulink, Raspberry Pi, ROS 2.

1. Introduction

Since the automation of the first processes, their development and progress has continued to increase in successive years. The arrival of programmable controllers in the 1960s led to the improvement of industrial processes in many areas. Its creation, as well as future improvements, gave way to the arrival of microprocessors on the market, introducing Intel the model 4004 in 1971. The introduction of these devices to the market was a revolution for the time since it opened the doors to a wide range of applications, highlighting the automation processes, as well as the improvement of the level of comfort and experience of the users in certain daily tasks.

Among the areas that advanced on the path of automation, it is worth highlighting the world of the automobile, seeking since the end of the 20th century the objective of introducing autonomous vehicles to the market. Within this context, this project is introduced, seeking the automation of a differential traction land vehicle in a controlled indoor environment through the use of sensors, focusing the project on the use of 2D LiDAR to achieve this objective.

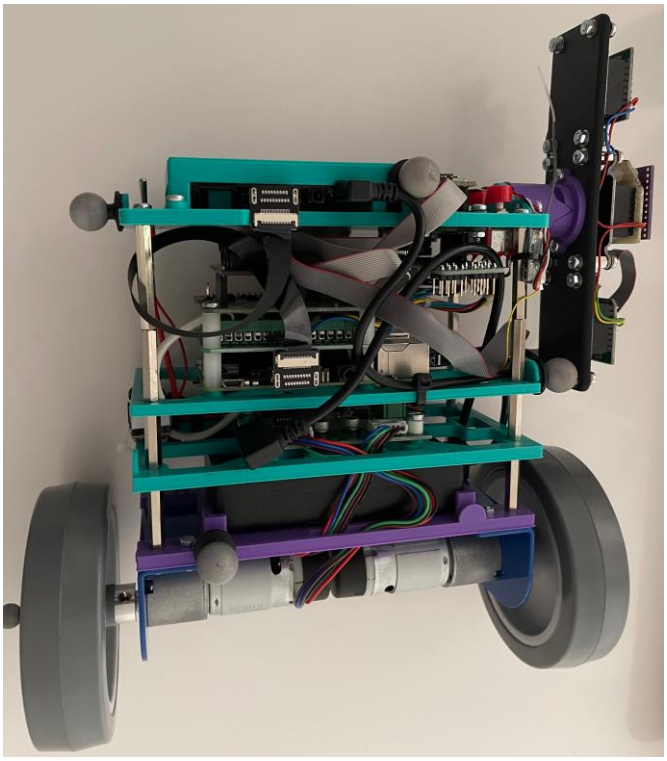
2. Definition of the project

To carry out the project, a series of clear objectives can be set.

- Develop the necessary algorithm to obtain the angle and distance measurements of the 2D LiDAR sensor model A2M8 from SLAMTEC.
- Manage the transmission of information between the two Raspberry Pi model 3B+ via TCP/IP protocol via Ethernet cable, for subsequent treatment and processing.
- Develop the control techniques required to achieve robust wall following control.

3. Description of the hardware and software used

Regarding the hardware used for the project, the use of a differential traction land vehicle stands out. This vehicle is equipped with various sensors, including an IMU (Inertial Measurement Unit), two DC motors, push buttons, LED lights to indicate the status of the vehicle, as well as an RPLIDAR A2M8 sensor from the SLAMTEC distributor. It also has two Raspberry Pi model 3B+, one of them in charge of managing the LiDAR sampling and sending the data to the other Raspberry, the latter having as its main function the processing of the information extracted by the sensor. Illustration 1 shows the vehicle used during the project.



Ilustration 1. Differential traction vehicle used in the project.

On one hand, the software used can be divided into two fundamental groups, the one used to obtain measurements and the one used for its subsequent processing and development of the wall monitoring control.

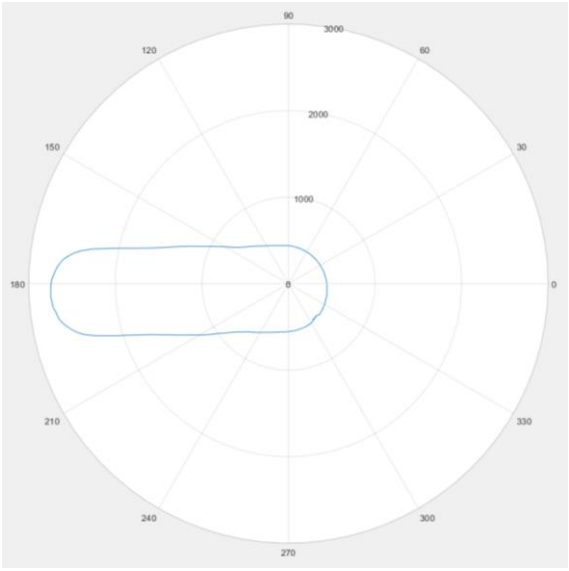
For the first part, that is, the taking of 2D LiDAR measurements, Python programming language has been used, due to the existence of libraries compatible with the LiDAR model used in the project. Using the developed code, it is possible to activate the sensor motor and start sampling. In addition, it is in charge of converting the data to a compatible format for its transmission by TCP/IP protocol, carried out also by this script.

For the processing of the information, as well as the development of the wall tracking control algorithm, Matlab-Simulink has been used. By using both, it is possible to coordinate the different tasks of the differential drive vehicle, as well as modify different parameters of the system to achieve robust and effective control.

4. Results

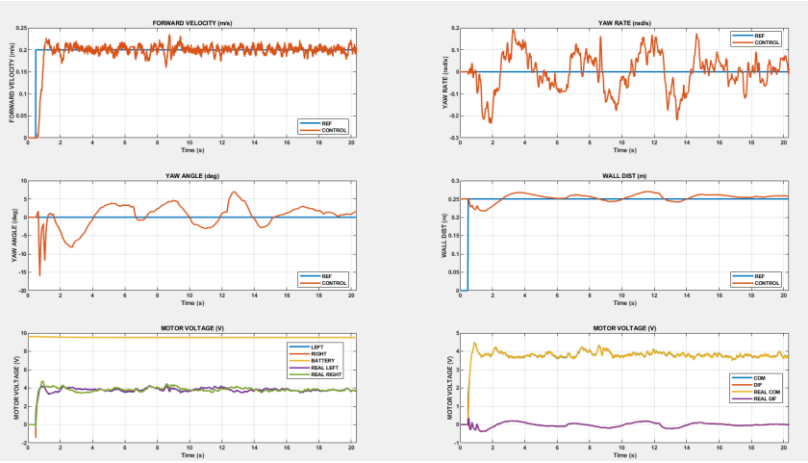
The results obtained in the different tests carried out during the project can be divided into two groups.

In the first place, the results related to the quality of the LiDAR measurements. It was decided to map the environment with the static vehicle. The results obtained are presented in Illustration 2.



Ilustration 2. Vehicle environment map.

Finally, the results obtained from the monitoring of the wall tracking control are included, showing the progression of different variables of interest related to the control. The results are presented in Illustration 3.



Ilustration 3. Wall tracking control test.

5. Conclusions

After carrying out the different tests, as well as selecting the appropriate parameters for the control tasks, it has been possible to meet the objectives proposed for this project.

The implemented wall tracking control is robust, allowing its replication in various controlled environments and being able to modify the distance reference to the nearby wall.

In the case of future projects, the use of ROS 2 (Robotic Operative System) is recommended for its development, allowing access to various features developed by the community, as well as easy implementation of them. Other autonomous navigation paths can be explored as environment mapping using the SLAM (Simultaneous Localization and Mapping) algorithm.

6. References

[1] J. Balcells y J. L. Romeral, Autómatas Programables, Barcelona: Marcombo, 1997.

[2] W. Aspray, «The Intel 4004 microprocessor: what constituted invention?,» *IEEE Annals of the History of Computing,* vol. 19, nº 3, pp. 4-15, 1997.

[3] T. Sauter, S. Soucek, W. Kastner y D. Dietrich, «he Evolution of Factory and Building Automation,» *IEEE Industrial Electronics Magazine,* vol. 5, nº 3, pp. 35-48, 2011.

Índice de la memoria

Índice de figuras

Índice de ecuaciones

Capítulo 1. INTRODUCCIÓN

1.1 INTRODUCCIÓN

El desarrollo tecnológico durante los últimos siglos, ha permitido avances en distintas disciplinas, entre ellas, una de las más beneficiadas, ha sido el sector de los vehículos. El progreso en dicha disciplina, así como el acercamiento de los mismos a la población, ha motivado una serie de necesidades, así como objetivos en el progreso científico. Es aquí donde se engloba el campo de la navegación autónoma y enfoque de este trabajo.

Se entiende por navegación, el proceso por el cual un agente es capaz de trasladarse desde un punto inicial hasta otro punto final, sirviéndose para ello de cierta información, como variables propias del agente (vehículo en este caso), posición, velocidad, aceleración, entre otras, y aquellos parámetros propios del entorno ya sea la presencia de obstáculos, diferentes rutas… [1]

La automatización de este proceso requiere de diferentes medios, entre ellos sensores, que permitan al vehículo recopilar información sobre sus propias variables de estado y sobre el entorno en que se encuentra para lograr el objetivo deseado. Este trabajo basará sus esfuerzos en la implantación y posterior verificación del sistema de control de navegación de un vehículo de tracción diferencial mediante el uso de un LiDAR 2D. Dicho vehículo operará en un ambiente de interiores, tratándose de un ambiente controlado, facilitando su posicionamiento.

En el contexto de la navegación autónoma de los coches, la Society of Automotive Engeneers establece que se pueden distinguir una serie de niveles, del 0 al 5, que marcan el nivel de automatización alcanzado por el vehículo. [5]

- Nivel 0: Sin automatización.

En este nivel de automatización, el conductor se encuentra al mando de los controles primarios del vehículo y se requiere la atención del mismo para controlar el entorno y los controles del vehículo. Además, se incluye en esta categoría aquellos vehículos que, aun contando con ciertas ayudas al conductor, no tengan control sobre la aceleración, frenado o dirección del vehículo.

- Nivel 1: Asistencia a la conducción.

En este nivel de automatización, existen ciertas funciones en el vehículo como el anti-lock braking (ABS), que impide el bloqueo de las ruedas durante la frenada, permitiendo al conductor un mayor control del vehículo. El conductor debe mantener el control casi total de las funciones del vehículo, aunque existe alguna asistencia al mismo, mediante el sistema avanzado de asistencia al conductor (SAAC), permitiendo el control de dirección, frenado o aceleración, aunque nunca de forma simultánea.

- Nivel 2: Automatización parcial.

Se mantiene, al igual que en el nivel anterior, el sistema avanzado de asistencia al conductor, no obstante, en este caso, sí permite el control simultáneo del sistema de control y el de aceleración o freno. La principal distinción con el nivel 1 de automatización es que, a partir de este nivel 2, el vehículo es capaz de circular por sí solo, sin requerir que el conductor tenga sus manos en el volante o sus pies en los pedales. No obstante, debe permanecer atento a las inmediaciones y al entorno por el que circula.

- Nivel 3: Automatización condicional.

En este punto, el vehículo cuenta con un sistema de conducción autónoma (SCA) que le permite tomar el control de todas las funciones de conducción del mismo, permitiendo desempeñar la conducción autónoma. Existen ciertas limitaciones a la misma ya que ante determinadas circunstancias el sistema puede requerir de la intervención del conductor debido a fallos en el sistema de conducción autónoma, errores de conducción de otros vehículos en la vía o situaciones excepcionales en el entorno, entre otros. Es por ello que se requiere de la atención del conductor ante estos posibles imprevistos.

- Nivel 4: Automatización elevada.

Este estado de automatización comparte la mayoría de características con el descrito en el nivel 3. Incorpora un sistema de conducción autónoma, siendo capaz el vehículo de desenvolver las funciones de conducción de forma satisfactoria. La principal diferencia con el punto anterior es que el vehículo es capaz de llevar a cabo esas tareas incluso si el conductor no responde de forma apropiada ante una petición de intervención por parte del sistema de conducción.

- Nivel 5: Automatización completa.

Como indica el nombre, en este nivel, el vehículo es capaz de llevar a cabo la totalidad de las labores de conducción, sin requerir en ningún momento de la asistencia humana para su desarrollo. Es por ello que los ocupantes en el vehículo son simples pasajeros que no intervienen en la conducción del mismo.

1.2 MOTIVACIÓN

La motivación fundamental de la realización de este proyecto se basa en la creciente automatización de las diferentes actividades cotidianas y, en especial, en el sector automovilístico. El campo de la robótica está sentando las bases en un gran número de áreas permitiendo lograr unos estándares de vida más acomodados para la mayor parte de la población en su día a día.

En especial, el campo de aplicación de este proyecto, centrado en la automoción y navegación de vehículos autónomos, tiene como principal razón de ser el facilitar la experiencia a los usuarios, así como automatizar labores de transporte, tanto en medios controlados como puede ser un almacén, como medios más caóticos como una carretera. Por su parte, las aplicaciones en este campo permitirán en un futuro y, con su desarrollo, la reducción de los accidentes, al tratarse la mayoría debido a errores de tipo humano.

1.3 OBJETIVOS

El objetivo final del proyecto es lograr que el vehículo siga una pared manteniendo una distancia de referencia a ella, sirviéndose de las medidas obtenidas por el LiDAR. Este objetivo principal puede dividirse en diversos objetivos para alcanzar dicho resultado.

- Desarrollar el código que permita obtener de forma fiable las medidas del LiDAR para su uso en la navegación autónoma.
- Lograr transmitir dichas medidas mediante protocolo de comunicación TCP/IP mediante cable Ethernet entre dos Raspberry Pi modelo 3B+.
- Desarrollar la algoritmia y técnicas de control necesarias para lograr un seguimiento de pared robusto.

1.4 RECURSOS

Para el desarrollo de este proyecto se cuenta con diversos recursos, pudiendo clasificarse en hardware y software fundamentalmente.

Entre los recursos hardware empleados, destaca el vehículo de tracción diferencial junto con los sensores que emplea. Entre ellos cabe destacar la IMU y el LiDAR 2D modelo A2M8 de la marca SLAMTEC. El vehículo cuenta con otros sensores que serán presentados en el capítulo 3, pero que no presentan relativa importancia en el desarrollo de este proyecto. Cuenta además con pulsadores que permiten cambiar entre los diferentes estados del vehículo, por ejemplo, de toma de medidas y calibración de los sensores a activar el motor para comenzar el movimiento, así como luces LED que permiten determinar el estado en que se encuentra el vehículo.

Se emplean además dos ordenadores Raspberry Pi modelo 3B+. Uno de ellos es el encargado de gestionar la toma de datos por parte del LiDAR así como su transmisión por protocolo TCP/IP mediante cable Ethernet a la otra Raspberry. Esta segunda, recibe dicha información y se encarga, entre otras labores, del procesamiento de los datos recibidos mediante los elementos de software pertinentes.

En cuanto al software empleado, destaca fundamentalmente el uso de Python, empleado para desarrollar el algoritmo de toma de medidas, así como Matlab y Simulink, cuyo uso ha sido de importancia para el procesado de datos y desarrollo de la algoritmia y técnicas de control como el filtro extendido de Kalman, necesario para el control de seguimiento de pared.

Capítulo 2. ESTADO DEL ARTE

Para entender la situación actual en que se encuentran los vehículos autónomos, resulta importante conocer el recorrido que han tenido desde su aparición hasta la actualidad para poder conocer la evolución de las tecnologías empleadas para lograr dicha función.

En la década de 1920 aparecieron los vehículos denominados como 'phantom auto' [3] que eran capaces de realizar labores de conducción sin la necesidad de que una persona estuviera a los mandos del vehículo. Este es un primer punto de partida para la navegación sin vehículos sin conductor, aunque no se puede categorizar como conducción autónoma ya que no era el vehículo ni un algoritmo el encargado de la toma de decisiones, si no un individuo que, mediante sistema de radiocomunicaciones y viajando en un vehículo contiguo al primero, tomaba los controles del vehículo.

Años más tarde, se presentaría en 1939, en la Exposición Universal celebrada en Nueva York, una serie de propuestas sobre cómo se imaginaba el futuro [4]. Se trataba de una iniciativa patrocinada por General Motors, bajo el nombre de Futurama y presentaba las ideas futuristas de los participantes. Entre estas propuestas, destaca la de Norman Bel Geddes, diseñador industrial y teatral estadounidense, que proponía el diseño de vías que permitieran mediante campos electromagnéticos generados por elementos incrustados en las vías de transporte, guiar la circulación de vehículos. No obstante, dicha idea fue posteriormente desechada y se centraron los esfuerzos en el desarrollo de vehículos autónomos que fueran capaces de circular por sí mismos, independientemente de si la vía estaba diseñada para ello.

No fue hasta la década de 1980 cuando empezaron a surgir los primeros proyectos que se pueden englobar en la categoría de conducción autónoma. Uno de ellos fue el proyecto NavLab 1 de la Carnegie Mellon University [5]. Dicho proyecto consistía en una furgoneta equipada con un sistema de tecnología similar al moderno LiDAR, cámaras y un gran número de ordenadores ubicados en el interior de la misma, encargados de la algoritmia

necesaria para el control de movimiento de la furgoneta, así como de otras funciones, permitiendo seguir determinadas rutas.

Figura 1. Fotografía del vehículo NavLab I

En esa misma década surgió el proyecto Autonomous Land Vehicle (ALV) [6] [7]. Permitía la navegación autónoma, aunque de forma limitada. Su funcionamiento se basa en el uso de un sistema de cámaras con las que el vehículo capta el entorno por el cual está circulando. A partir de esta información de entrada (combinada con datos de otros sensores) y con una serie de algoritmos que permitían llevar a cabo una segmentación del entorno basada en colores por medio de los valores en la escala RGB de cada pixel, se podría detectar obstáculos, así como los elementos fundamentales del entorno, ya sea la carretera, caminos o la acera. No obstante, existían ciertas desventajas como la imposibilidad de realizar correctamente la segmentación en entornos con gran contraste de sombras o la dificultad del algoritmo de delimitar los límites de la vía de circulación cuando los colores de esta resultaban similares a los del exterior de la vía. A todo esto, cabe además sumar las limitaciones computacionales de la época.

Figura 2. Fotografía del vehículo Alvin

En los años sucesivos se siguió experimentando para mejorar la fórmula de la conducción autónoma, sirviéndose de diversos sensores y de los avances en las técnicas de algoritmia y computación de la época. Fue en 2002 cuando DARPA (Defense Advanced Research Projects Agency), agencia estadounidense que dirige proyectos para el desarrollo de innovaciones tecnológicas con uso militar y cuyos primeros modelos sirvieron para el desarrollo del ALV y del NatLab, propuso un desafío para incentivar y poner a prueba el desarrollo de la navegación autónoma [8]. Su objetivo final era lograr que, en 2015, un tercio de los vehículos terrestres en el ámbito militar, fueran autónomos. El desafío consistía en que un vehículo autónomo fuera capaz de atravesar una ruta de 240 km a través del desierto de Mojave (Estados Unidos). Se ofreció una recompensa económica de un millón de dólares para el ganador del evento.

La celebración del desafío DARPA tuvo lugar el 13 de marzo de 2004 y, ninguno de los vehículos participantes logró atravesar la línea de meta. El vehículo que logró recorrer la mayor distancia fue el del equipo de la Carnegie Mellon University, universidad desarrolladora del vehículo NavLab 1, logrando una distancia de 11.78 km. El resultado de esta competición resultó una mala noticia para el mundo de la conducción autónoma.

A pesar de los resultados de esta primera competición, se celebraros dos competiciones sucesivas en 2005 y 2007. La primera de ellas tuvo lugar en un entorno de caminos de montaña, logrando la mayoría de participantes superar la distancia alcanzada por el mejor vehículo de la competición anterior. Los vehículos en su recorrido llevaron a cabo maniobras con cierto grado de complejidad como pasar por zonas estrechas o tomar curvas pronunciadas. Al final, 5 de los vehículos lograron llegar a la meta, tras recorrer 212 km. Este logro permitió observar los avances alcanzados en el sector desde el inicio del concepto de vehículo autónomo y sembrar las bases para su posterior desarrollo en el futuro.

En 2007 se celebró una tercera competición, esta vez en entorno urbano. La dificultad aumentó respecto a las ocasiones anteriores ya que se debía hacer el recorrido en un tiempo inferior a 6 horas, con el añadido de respetar todas las medidas de tráfico de California (donde se celebró la competición). Este evento logró un mayor acercamiento a la imagen actual de vehículo autónomo al enfrentar a los mismos a entornos urbanos, en los que los obstáculos son cambiantes y existen una gran cantidad de factores, algunos de ellos poco predecibles, como la actitud de los demás conductores o las reacciones de peatones.

El progreso tecnológico y los avances científicos en los años siguientes, permitieron mejorar los sistemas de navegación autónoma, encontrándose la meta de vehículos con nivel 5 de automatización, más cercana. Para lograr alcanzar esta meta, se deben centrar los esfuerzos en una serie de elementos fundamentales.

2.1 PERCEPCIÓN DEL ENTORNO

Para el desarrollo de los sistemas necesarios para que un vehículo pueda navegar de forma autónoma, es necesario, en primer momento, que éste sea capaz de obtener información del entorno. Esta labor se lleva a cabo a partir de diferentes sensores. Se presentan a continuación algunos de los sensores más empleados en el campo de la navegación autónoma [9].

2.1.1 CÁMARAS

Numerosos vehículos autónomos cuentan con un sistema de cámaras entre otros sensores para obtener información de los alrededores. La información obtenida por las cámaras se procesa posteriormente mediante diversos algoritmos y técnicas de visión por computación combinadas con el uso de inteligencia artificial que permiten, entre otras cosas, clasificar y reconocer los diversos objetos presentes en la vía. Los algoritmos empleados pueden ser muy diversos, desde segmentación basada en colores como en el caso del ya mencionado proyecto ALV, hasta técnicas de clustering avanzadas.

Se estipuló que a partir de 2022, los coches deberán incluir de serie una cámara trasera, aunque en este caso, el motivo es facilitar aparcamientos y reducir accidentes, más que emplearse como sensor para dotar de autonomía al vehículo.

El empleo de cámaras viene ligado con ciertos problemas. En primer lugar, el manejo de archivos de video como método de procesamiento de información, necesita unos requerimientos de hardware que van aumentando con el número de cámaras que se empleen en el sistema. Añadido a esto, cabe destacar que en entornos con escasez de luz estos sensores presentarán una elevada dificultad para percibir las inmediaciones del entorno.

Tesla, por ejemplo, anunció que sus vehículos desde mayo de 2021 prescindirán de radares como método principal de autonomía para ser sustituidos por un sistema de cámaras [10].

2.1.2 RADARES

Los radares (su nombre deriva del inglés radio detection and ranging) son otros de los sensores empleados como método de interacción de los diferentes vehículos con el entorno, permitiendo determinar distancias a partir de la emisión y recepción de pulsos de ondas electromagnéticas [11].

A modo de resumen, según la forma empleada para determinar distancias, es posible distinguir dos tipos de radares. En primer lugar, aquellos que basan su funcionamiento en determinar el tiempo que tarda la onda electromagnética emitida en alcanzar el receptor, generalmente localizado al lado del emisor. La onda es emitida y recorre una determinada distancia hasta el objeto a detectar. Una vez alcanza dicho objeto, la onda es reflejada y recorre cierta distancia (por simplificación se considera la misma) hasta alcanzar el receptor.

$$d = \frac{c * t}{2}$$

Ecuación 1. Cálculo de la distancia en un radar.

Mediante la ecuación 1 es posible determinar la distancia *d*, donde *t* es el tiempo que emplea la onda desde su emisión hasta su recepción y *c* es la velocidad de la luz, que en el vacío tiene un valor de 299.792,458 km/s.

La otra manera de determinar la distancia a los objetos del entorno por medio de radares de onda continua, basados en la modulación en frecuencia. Su funcionamiento está basado en la emisión de ondas electromagnéticas a las que se las va modificando la frecuencia. De forma análoga a los radares basados en la medición del tiempo, la onda es reflejada por los objetos del entorno y regresa al receptor. En este caso y, a diferencia de los anteriores radares, se mide la variación de frecuencia experimentada por la onda y, teniendo en consideración determinados parámetros y sirviéndose del efector Doppler, es posible determinar la distancia hasta el objeto.

2.1.3 SENSORES POR ULTRASONIDO

Los sensores por ultrasonido tienen un funcionamiento similar a los radares. A diferencia de trabajar con pulsos de ondas electromagnéticos, se sirven de ultrasonidos, como su propio nombre indica. Estos son emitidos por el emisor y, tras alcanzar algún elemento del entorno, son reflejados al receptor.

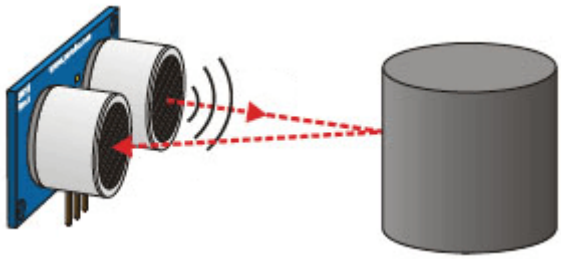


Figura 3. Funcionamiento de un sensor por ultrasonidos.

La ecuación empleada para la obtención de la distancia al objeto es idéntica a la utilizada en los radares basados en medición del tiempo con la única excepción de que en lugar de emplear como velocidad la de la luz, se emplea la del sonido, que en la atmósfera terrestre es de 343.2 m/s.

Su uso es limitado debido a su escaso rango de operación ya que permiten detectar objetos a escasos metros. No obstante, su uso ha sido extendido desde hace décadas, empleándose principalmente para labores de asistencia al aparcamiento

2.1.4 LIDAR

El sensor objeto de estudio de este trabajo. Se presenta a continuación un breve resumen de su funcionamiento y aplicaciones actuales ya que se tratará con más profundidad en el capítulo 3.1.5.

El LiDAR (Light Detection and Ranging o Laser Imaging Detection and Ranging) es un sensor que permite determinar la distancia a un cuerpo a partir del tiempo que tarda el haz laser emitido por el emisor, en llegar al receptor [12]. Existen LiDAR que permanecen estáticos y funcionan de manera similar a los ya descritos radares. Por otra parte, el LiDAR empleado en este proyecto es rotativo, permitiendo obtener una imagen del entorno que le rodea en cualquier dirección. Mientras el sensor rota, emite haces laser que, al regresar, permiten determinar la lejanía de los cuerpos en un rango de 360º.

Los usos de este sensor son muy variados, empleándose en campos tan diversos como la topografía, la física atmosférica, la geología y la navegación autónoma, entre otros. En la actualidad empresas como Google, Uber o Ford centran el progreso de la navegación autónoma en el empleo del LiDAR como forma de detectar el entorno. A pesar de la precisión de las medidas obtenidas y requerir de una menor carga computacional que otros sistemas como las cámaras, presenta una serie de inconvenientes en su uso para la navegación autónoma. Entre ellos destacar su elevado precio y la dificultad de implementarlo de forma estética en los vehículos. Frente a este último punto Volkswagen ha presentado un modelo de coche que integra estos sensores de forma que interfieran lo menos posible con el diseño del vehículo [13]. Inconvenientes como estos han llevado a empresas como Tesla o Audi a prescindir del sensor LiDAR y centrarse en el uso de cámaras para dotar a sus vehículos de autonomía.

2.1.5 IMU

La IMU (Inertial Measurement Unit) o por su traducción al castellano Unidad de Medición Inercial es un dispositivo que permite obtener determinadas propiedades como la aceleración o velocidad angular del vehículo [14]. Consiste en una serie de sensores entre los que se destaca los acelerómetros, giroscopios y, en algunos casos, también incluyen magnetómetros.

Los acelerómetros son un tipo de sensores que, como su nombre indica, permiten determinar la aceleración lineal a lo largo de un eje a la que está sometida un cuerpo. Una vez obtenida la medida de la aceleración, mediante integración, es posible obtener medidas relativas de posición. El funcionamiento de los mismos es diverso debido a la amplia gama existente. Existen acelerómetros capacitivos, en los que la deformación de elementos capacitivos debido a la aceleración da lugar a variaciones de la medida de capacitancia de los mismos, permitiendo determinar así la medida. Otros como los piezoeléctricos aprovechan la estructura cristalina de determinados materiales para medir la aceleración lineal a partir de os cambios en el comportamiento eléctrico de los mismos.

Los giroscopios por su parte, permiten determinar la velocidad angular de los cuerpos. De manera análoga a los acelerómetros, a partir de esta medida, es posible obtener la posición angular del cuerpo mediante integración. Son empleados en numerosas aplicaciones, desde navegación autónoma hasta asistencia a personas de avanzada edad para evitar caídas [15].

Las IMU tradicionales constan de 3 acelerómetros y 3 giroscopios que permiten determinar sus medidas correspondientes a lo largo de 3 ejes ortogonales. No obstante, modelos con mayor grado de sofisticación, conocidos como IMU de 9 ejes incluyen, además de los sensores mencionados, 3 magnetómetros.

Los magnetómetros son sensores capaces de determinar la intensidad de campo magnético en una región del espacio. Su incorporación en las IMU permite obtener una estimación más precisa de la posición del cuerpo. El proceso de integración de las medidas de acelerómetros y giroscopios es susceptible a acumulación de errores ya que al integrarse dan lugar a un crecimiento lineal e incluso cuadrático de los mismos. El uso de magnetómetros permite reducir la incertidumbre de dichas medidas, logrando una reducción del error asociado.

2.1.6 ENCODERS

Se trata de un tipo de sensor que permite convertir diferentes formas de movimiento en señales o impulsos eléctricos que, una vez interpretados, brindan información relativa a la velocidad o posición, tanto lineal como angular, del cuerpo en cuestión.

Existen una gran variedad de tipos de encoders así como de clasificaciones de los mismos. Los encoders ópticos son los más empleados en la industria y su funcionamiento está basado en 3 partes principales. Un emisor de luz, un disco rotativo con diferentes zonas, algunas bloquean el paso de la luz mientras que otras permiten su paso, y un receptor.

A medida que el disco va rotando, el emisor recibe pulsos de luz que son posteriormente tratados por diferentes programas para determinar la posición angular.

Otros tipos de encoders [16] también empleados son los encoders lineales, los de cuadratura o los encoders incrementales, midiendo este último los incrementos en la medida de una magnitud a partir de un valor de referencia.

2.2 SISTEMA DE LOCALIZACIÓN

Una vez obtenida información del entorno en que se encuentra el vehículo, es necesario que éste se localice en el entorno en que se encuentra. Existen diferentes maneras de lograr este objetivo. Se distinguen dos grandes grupos, los sistemas de localización absolutos y relativos.

2.2.1 SISTEMAS DE LOCALIZACIÓN ABSOLUTOS

Los sistemas de localización o posicionamiento absolutos son aquellos que permiten determinar la posición de un cuerpo, con unos ciertos márgenes de incertidumbre, sin necesidad de conocer las características del entorno en que se encuentra. Se engloban en esta categoría sistemas como el GPS o GLONASS [17], que permiten, por ejemplo, determinar la ubicación de un vehículo, sin necesidad de recopilar información de los sensores a bordo del mismo.

El sistema GPS (Global Positioning System) o su análogo ruso GLONASS, son sistemas de localización por satélite que han logrado alcanzar un gran desarrollo a lo largo de los años desde su creación, estando presente la tecnología en la mayoría de los vehículos modernos, permitiendo posicionarnos espacialmente y, a partir de los algoritmos necesarios, determinar la ruta más eficiente hasta nuestro destino.

2.2.2 SISTEMAS DE LOCALIZACIÓN RELATIVOS

Los sistemas de localización relativos, permiten por su parte determinar la ubicación del vehículo relativa al entorno en que se encuentra. Es aquí donde entran en juego los sensores descritos anteriormente. Mediante diversas técnicas y algoritmos, los datos de los sensores son procesados y combinados entre sí para obtener la estimación más precisa posible de la posición del vehículo en el entorno en que circula.

2.2.2.1 SLAM

El método SLAM (Simultaneous Localization and Mapping) permite, como su propio nombre indica, realizar un mapa del entorno en que se encuentra el vehículo y, al mismo tiempo estimar la ubicación del mismo en el entorno [18]. El método SLAM es muy empleado en proyectos relacionados con el LiDAR, pudiendo encontrarse implementado en vehículos autónomos, incluso aspiradoras inteligentes.

En primer lugar, es necesario disponer de un sensor que permita determinar la posición de los obstáculos en el entorno en que se halla el vehículo. Además, se debe disponer de información relativa a la odometría del vehículo. Mediante odometría, es posible determinar la posición relativa del vehículo a partir de medidas de las ruedas. Para esta labor se emplean los conocidos como encoders, ya explicados en el capítulo 2.1.6.

A través de la información proveniente de la odometría, se determina la posición relativa en que se encuentra el vehículo, comparada con la posición en la que comenzó el mapeo. Se conocen además las medidas del LiDAR para cada una de esas posiciones. A partir de esta información es posible recrear un mapa del entorno. No obstante, tanto las medidas provenientes del LiDAR como las medidas de los sensores encargados de recopilar la

odometría, presentan un cierto ruido que impide la exactitud de las mismas. Para paliar este problema la solución pasa por un método basado en grafos, por el cual se van uniendo los distintos puntos por los que ha circulado el vehículo, creando un bucle cerrado. Mediante diferentes herramientas matemáticas cómo el método de mínimos cuadrados, se optimiza la posible ruta real, es decir, obviando los errores en odometría, para así obtener un mapa que refleje de forma más fiel la realidad. Existen diferentes tipos de mapas de ocupación, algunos de ellos binarios, en los que una zona está ocupada por un obstáculo o permanece libre de ellos, o mapas probabilísticos, en los que se asocia del 0 al 1 una probabilidad de que una región del espacio esté ocupada.

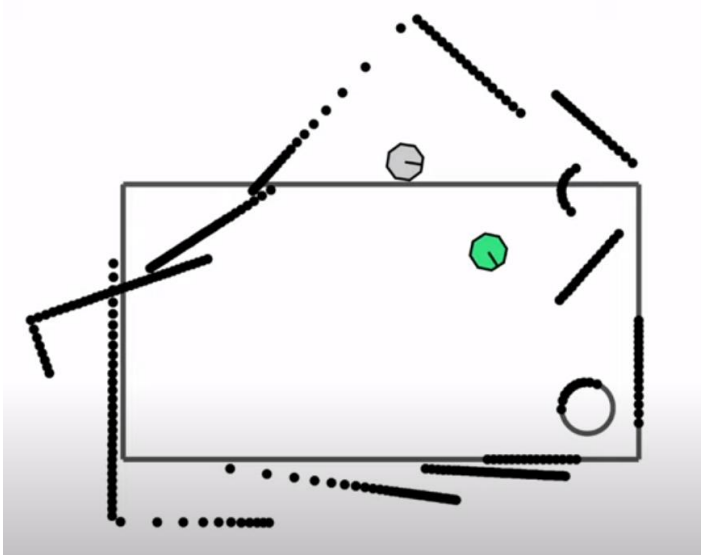


Figura 4. Mapa creado mediante el método SLAM.

En la figura 4 se muestra en verde la posición inicial en la que se comenzó el escaneo frente la posición final, mostrada en gris, que debería coincidir con la inicial en este caso, pero no lo hace debido a errores en estimación de la posición relativa por ruido en las medidas de odometría. Dichas desviaciones de la realidad dan lugar a que el mapa generado, mostrado como una nube de puntos negros, difiera del mapa original.

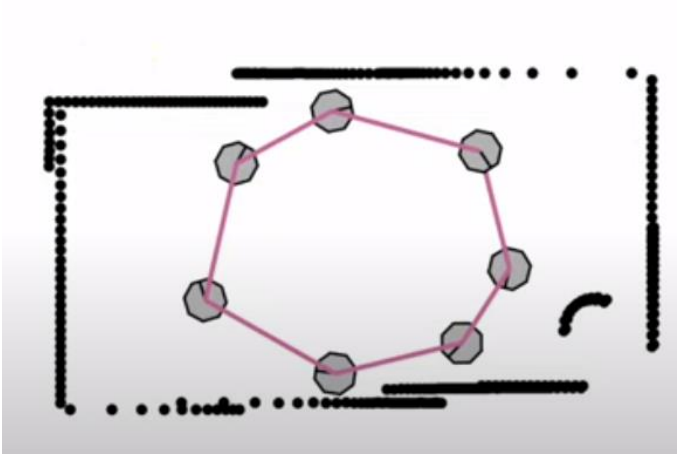


Figura 5. Mapa una vez aplicado la optimización.

La figura 5 muestra el resultado del mapa generado una vez aplicada la optimización basada en teoría de grafos y sus restricciones [19].

Los resultados obtenidos al emplear un LiDAR 2D son limitados ya únicamente es posible obtener las medidas del mismo plano en el que se encuentre ubicado. Existen alternativas como método SLAM empleando cámaras, aunque debido a la gran cantidad de datos que se pueden obtener con el sensor LiDAR, es preferible el uso de este último.

2.2.2.2 Filtro de Kalman

Otro de los métodos empleados para estimar la posición de un vehículo autónomo en el entorno en que se encuentra es el conocido como filtro de Kalman. Los filtros de Kalman son observadores de estado basados en métodos estadísticos [20]. Se trata de algoritmos que permiten determinar el estado de una variable de interés, sea por ejemplo la posición del vehículo, a partir de medidas a las que se tiene acceso directo, sensores en este caso. La variable que pretendemos estimar no puede ser medida directamente debido a limitaciones en los instrumentos de medida o la imposibilidad de los mismos de realizar mediciones de la misma. Es por ello que mediante relaciones matemáticas entre variables que sí son accesibles para ser medidas, se puede definir un modelo matemático del sistema de estudio.

Las medidas de los sensores, como se mencionó en apartados anteriores, presentan ruido, dando lugar a incertidumbres en las medidas. Los filtros de Kalman permiten, mediante su

algoritmia, paliar parte de los efectos del ruido de los sensores, permitiendo obtener una estimación más precisa que las medidas de los propios sensores, sujetas a incertidumbre.

El empleo de filtros de Kalman está muy extendido en aplicaciones de navegación inteligente y en otros campos muy diversos como economía o procesamiento de señales.

Existen diversas variaciones de los filtros de Kalman, así como diversas clasificaciones de los mismos en función a diversos criterios. El caso de estudio de este trabajo se trata de un sistema en el que el modelado requiere de ecuaciones no lineales. A este tipo de filtros de Kalman en el que intervienen relaciones no lineales de las variables se les conoce como filtros extendidos de Kalman.

En el capítulo 5.2.2 se procederá a explicar con mayor grado de detalle el filtro de Kalman concreto empleado para este proyecto.

2.2.3 PLANIFICACIÓN DE RUTAS

Una vez que el vehículo es capaz de percibir el entorno que le rodea por medio de sensores y de interpretar las medidas de los mismos para, mediante diversos algoritmos, estimar su posición en el entorno en cuestión, éste debe ser capaz de alcanzar una meta final de la manera más eficiente posible.

Es para cumplir esta misión que surgen los algoritmos de planificación de rutas. Estos algoritmos tienen como función determinar la ruta más óptima entre las posiciones de inicio y meta. Algún ejemplo de algoritmo de planificación de rutas es el algoritmo A* [21], versión optimizada del algoritmo de Dijkstra, basado en la teoría de grafos. Para determinar el camino que optimiza la ruta a seguir, se otorgan diferentes pesos a las rutas desde el origen a cada nodo n (llamemos al coste $C_1(n)$), así como desde cada nodo n hasta el nodo objetivo ($C_2(n)$). Se busca, mediante sucesivas iteraciones, el camino que permita minimizar la función objetivo

$$C_1(n) + C_2(n)$$

*Ecuación 2. Función objetivo del algoritmo A**

Una vez realizada la optimización, el camino obtenido garantiza ser el que minimice la función objetivo y, por tanto, el más eficiente según los criterios seleccionados para la misma.

Existen infinidad de algoritmos de planificación de rutas y cada vez más optimizados y específicos para labores concretas. Sus ámbitos de aplicación son muy diversos, siendo empleados en elementos de domótica como aspiradoras inteligentes, hasta en los servicios de Google Maps, permitiendo estos últimos trazar la ruta más óptima entre dos puntos geográficos, teniendo en cuenta las peculiaridades del entorno como carreteras o caminos secundarios, así como determinando las diferentes opciones de transporte y su combinación para alcanzar el destino final en el menor tiempo posible.

Capítulo 3. HARDWARE

Para este proyecto se ha empleado el vehículo de tracción diferencial usado en diversas asignaturas del laboratorio de la Universidad Pontificia Comillas, ICAI.

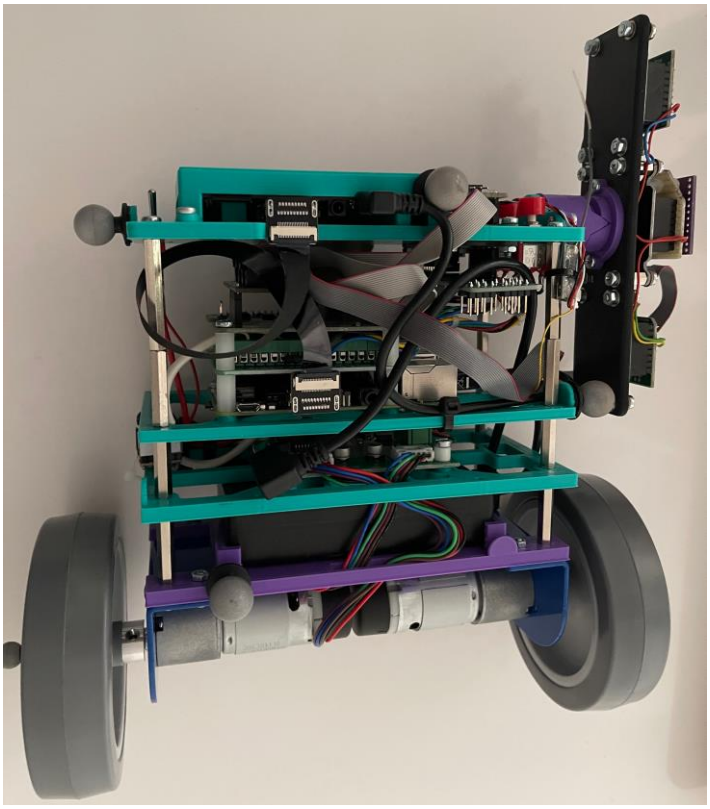


Figura 6. Vehículo de tracción diferencial empleado en el proyecto.

El vehículo, presentado en la figura 6, cuenta con una serie de componentes de hardware que se presentan a continuación.

3.1 MOTORES DE CORRIENTE CONTINUA

Para que el vehículo disponga de tracción diferencial, son necesarios dos motores. De esta manera, al aplicar tensión común a ambos, el vehículo podrá avanzar linealmente, mientras que, si se aplica tensión diferencial, es decir, más tensión a un motor que al restante, el vehículo es capaz de rotar sobre su eje.

Los motores empleados en el proyecto son motores EMG30. Dichos motores tienen una tensión de trabajo de 12V y vienen equipados con encoders magnéticos de efecto Hall [22], permitiendo tener una estimación de ciertos parámetros del vehículo como su posición o velocidad, tanto lineal como angular. Presentan una relación de reducción 30:1. Se recomienda su uso en aplicaciones de robótica sencillas o incluso con carga mediana de complejidad. Además, presenta ciertas ventajas como la presencia de un condensador que permite reducir el ruido.

Sus especificaciones son las siguientes:

- Tensión nominal: 12V
- Torque: 1.5Kg/cm
- Velocidad nominal: 170rpm
- Corriente nominal: 530mA
- Corriente sin carga: 150 mA
- Corriente de parada del motor: 2.5A
- Potencia: 4.22W
- Longitud: 86.6 mm
- Diámetro del motor: 30 mm
- Diámetro del eje: 5 mm

El control de los motores se realiza a través de drivers desarrollados en proyectos anteriores. Su conexionado se realiza a partir de 6 cables que terminan en un conector JST.

Figura 7. Motor de corriente continua EMG30.

3.2 RASPBERRY PI 3B+

Para este proyecto se han empleado dos Raspberry Pi modelo 3B+ diferentes, cada una con sus propias funcionalidades, que serán explicadas posteriormente en este mismo apartado.

Las Raspberry Pi son empleadas en gran cantidad de proyectos de robótica y automatización debido a su accesibilidad al público general, ya sea por precio o por la relativa sencillez de uso. Se trata de single-board computers (SBCs) o por su traducción al español, ordenadores monoplaca. Fueron desarrolladas en Reino Unido por la Raspberry Pi Foundation en colaboración con Broadcom, desarrollador de semiconductores y software, con el objetivo de enseñar los fundamentos básicos de computación a niños en diferentes escuelas.

Desde la aparición de la primera versión, la Raspberry Pi modelo B, en febrero de 2012, se han desarrollado diferentes generaciones de estos SBCs, existiendo en la actualidad un amplio catálogo de ofertas entre los que elegir. La elección de la Raspberry para un proyecto atiende a diferentes criterios, especialmente a que las especificaciones de la misma permitan

el desarrollo del proyecto en cuestión, así como a motivos de carácter monetario, ya que el precio varía entre las diferentes generaciones.

En el caso de la Raspberry empleada para este proyecto, la Raspberry Pi modelo 3B+, tiene un precio de mercado que ronda los 45€, dependiendo del portal de compras seleccionado.

Figura 8. Raspberry Pi model 3B+.

Las principales características y especificaciones [23] de este modelo de Raspberry Pi se presentan a continuación.

- Procesador BCM2837B0 64-bit ARM Cortex-A53 Quad Core SoC (System-on-a-Chip) de 1.4 GHz

- Cuenta con una memoria RAM de 1GB tipo LPDDR2 SDRAM
- Cuenta con salida de audio y video a través de conector de 4 polos, HDMI, Camera Serial Interface (CSI) o sistema LCD sin procesamiento.
- Presenta 4 puertos tipo USB con hasta 1.2A de salida.
- 40 pines de puerto GPIO (General Purpose Input/Output), que permiten la monitorización de sensores, así como mandar señales a los diferentes actuadores conectados a los pines.
- Cuenta con almacenamiento de tarjetas microSD, a partir de las cuales es posible instalar el sistema operativo a gusto del usuario, además de servir de fuente de almacenamiento.
- LAN inalámbrica 2.4 GHz y 5 GHz.
- Conexión Ethernet mediante puerto USB 2.0 con un rendimiento máximo de 300Mbps.
- Existen además ciertos periféricos de bajo nivel, entre ellos pines de conexión a tierra, a +3.3V y +5V, así como un periférico que permite la comunicación por protocolo UART (Universal Asynchronous Receiver/Transmitter), entre otros.

Debido a estas especificaciones, unido a su asequible precio, se ha seleccionado este modelo para la realización del proyecto.

Como se mencionó en el inicio de este apartado, el vehículo cuenta con 2 Raspberry Pi modelo 3B+ diferentes. La primera de ellas es la Raspberry de control, encargada de diversas tareas. Esta Raspberry recibe la información de los diferentes sensores presentes en el vehículo (IMU, encoders, LiDAR…) y se encarga de realizar los cálculos pertinentes, mediante el filtro extendido de Kalman y otros procedimientos, para estimar la posición del vehículo, velocidad y otros parámetros de navegación. Además, es la encargada de las labores de control, permitiendo implementar controles de velocidad de avance lineal, de velocidad de giro, así como un control de seguimiento de pared. Existen

además otras funcionalidades que desempeña, aunque estas quedan fuera del ámbito de este proyecto.

La otra Raspberry es la llamada Raspberry de navegación. Su función principal es recopilar información del sensor empleado para la navegación, en este caso del LiDAR, aunque en otros proyectos se emplean otros sensores como una cámara Ultra Wide Band. La información recopilada por el sensor es procesada en esta tarjeta y, transmitida a la Raspberry de control mediante protocolo TCP/IP a través de cable Ethernet. El algoritmo empleado para el tratamiento de la información del LiDAR será explicado en profundidad en el capítulo 5.1.

3.3 PULSADORES Y LED

El vehículo cuenta con 3 pulsadores que permiten controlar el estado en que se encuentra el mismo durante la ejecución de las diferentes tareas. Controla además la gestión de los diferentes drivers y permite la monitorización del estado del vehículo al tener este incluidos un conjunto de LED RGB, con los que tener información visual sobre el estado y evolución del vehículo.

3.4 RPLIDAR A2M8 SLAMTEC

Se trata del elemento principal de navegación y base de estudio de este proyecto. El modelo RPLIDAR A2M8 ha sido el empleado para la toma de medidas empleadas para la posterior navegación autónoma del vehículo.

Este modelo de LiDAR ha sido desarrollado por la compañía SLAMTEC con sede en Shanghai, China.

Figura 9. RPLIDAR A2M8 SLAMTEC.

Previo a exponer las especificaciones de este modelo y sus características, es necesario exponer el método de funcionamiento de este sensor.

El LiDAR (Light Detection and Ranging) consta de dos partes fundamentales, el emisor y el receptor. El emisor es el encargado de emitir un haz de luz láser en la dirección en que éste se encuentre apuntando. El receptor, por su parte, es el encargado de detectar el haz láser una vez ha sido reflejado por un cuerpo en su trayectoria. A partir del tiempo empleado por uno de los haces en ser emitido, reflejado y posteriormente detectado por el receptor, es posible determinar la distancia a un cuerpo concreto mediante la relación mostrada en la ecuación 1.

En este proyecto el LiDAR empleado es de tipo rotativo, por lo que, mediante un motor, va rotando 360 grados emitiendo haces láser, pudiendo determinar la distancia a los diferentes cuerpos que lo rodean. Al tratarse de un LiDAR 2D existe la limitación de únicamente poder

detectar aquellos cuerpos que se encuentren en el mismo plano que el emisor y receptor, por lo que la información obtenida por este proceso queda supeditada a la complejidad del entorno. Mediante encoders presentes en el motor que permite rotar al LiDAR, es posible determinar el ángulo que ha rotado en cada instante, pudiendo así determinar una dupla distancia-ángulo, obteniendo un mapa del entorno en coordenadas polares.

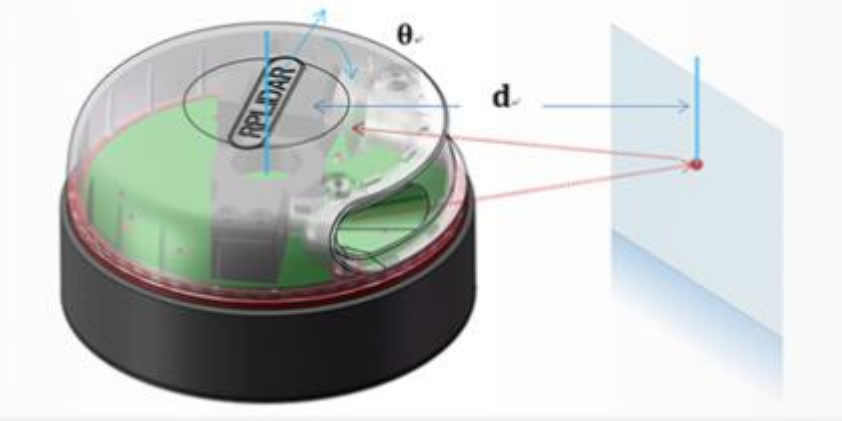


Figura 10. Obtención de las medidas del LiDAR.

Una vez se dispone de una nube de puntos en coordenadas polares es posible transformar dichos puntos a coordenadas cartesianas mediante las transformaciones matemáticas pertinentes.

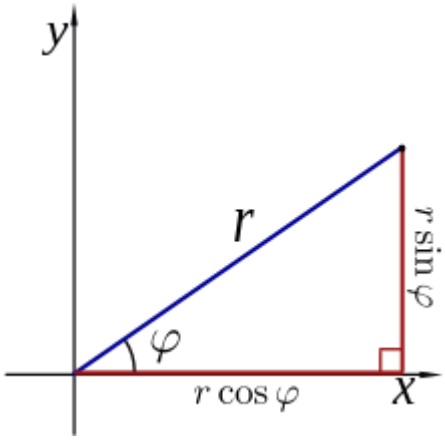


Figura 11. Paso de coordenadas polares a cartesianas.

En la figura 11, se muestra la transformación de coordenadas polares a cartesianas, donde *r* denota la distancia desde el origen, LiDAR en este caso) hasta el punto en cuestión y φ denota el ángulo rotado con respecto al ángulo de referencia.

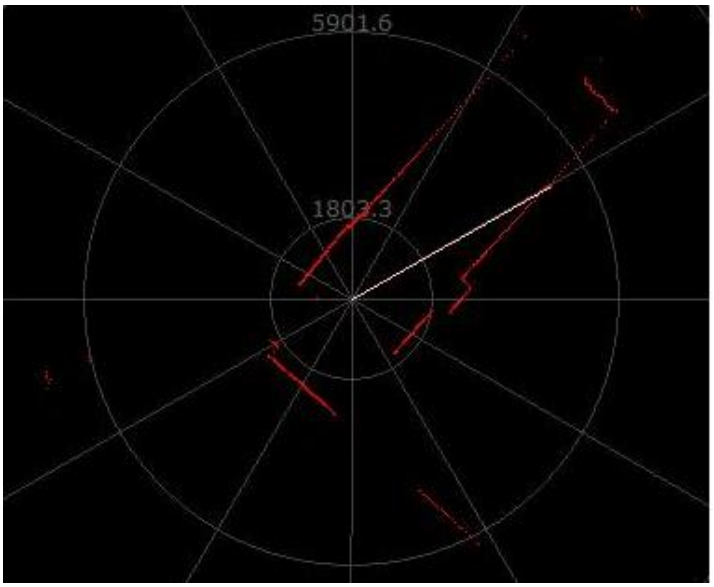


Figura 12. Mapa obtenido mediante el software framegrabber.

En la figura 12, se muestra un mapa realizado mediante sucesivos escaneos del entorno, representado mediante el programa framegrabber, compatible con el modelo RPLIDAR A2M8 de SLAMTEC.

En cuanto a las especificaciones de este producto, podemos distinguir entre aquellas relativas a la calidad de las medidas obtenidas y las referidas al haz láser emitido. En cuanto a las especificaciones relativas a las medidas, el fabricante proporciona la siguiente información [24].

- El rango de distancias que es capaz de medir este sensor se encuentra acotado entre los 0.15 y 8 metros. Este rango de distancias puede variar ya que está basado en cuerpos blancos con un 70% de reflectividad (parte de la radiación que es reflejada por una superficie).
- Es capaz de medir ángulos en un rango de 0 a 360 grados.
- En los rangos típicos de funcionamiento presenta una resolución en la medida de distancias menor al 1% del valor de la misma.
- En cuanto a la resolución angular presentada por este sensor, varía entre los 0.45 y 1.35 grados, encontrándose el valor típico en 0.9 grados, a una frecuencia de escaneo de 10 Hz.
- La frecuencia de toma de medidas puede variar entre un mínimo de 2000 Hz, hasta un valor máximo de 4100 Hz, siendo el valor de funcionamiento típico 4000 Hz, obteniéndose así una medida cada 0.25 ms.

La información proporcionada por SLAMTEC acerca del láser empleado para las mediciones del RPLIDAR A2M8 se presenta a continuación.

- La longitud de onda de la luz empleada se encuentra en el rango de la radiación infrarroja, con unos valores que varían entre 775 nm y 795 nm.
- La potencia del haz láser puede llegar a los 5 mW, situándose en 3 mW en el caso de funcionamiento típico.

- En cuanto a estándares de seguridad se refiere, este modelo de LiDAR obtiene una clasificación de clase I marcada por la FDA (Food and Drug Administration), considerándose por tanto un producto seguro sin ningún tipo de posible efecto adverso sobre el organismo humano.

El LiDAR requiere un suministro de tensión de 5 V en su rango de funcionamiento nominal, con un rizado en la tensión de 20 mV. Para el arranque del motor presente en el sensor la corriente requerida puede tomar un valor máximo de 1.5 A, mientras que la corriente nominal requerida es de 450 mA.

Para alimentar el LiDAR, así como para las labores de transmisión de información, éste presenta 5 cables que se unen en un conector macho tipo XH2.54-5P.

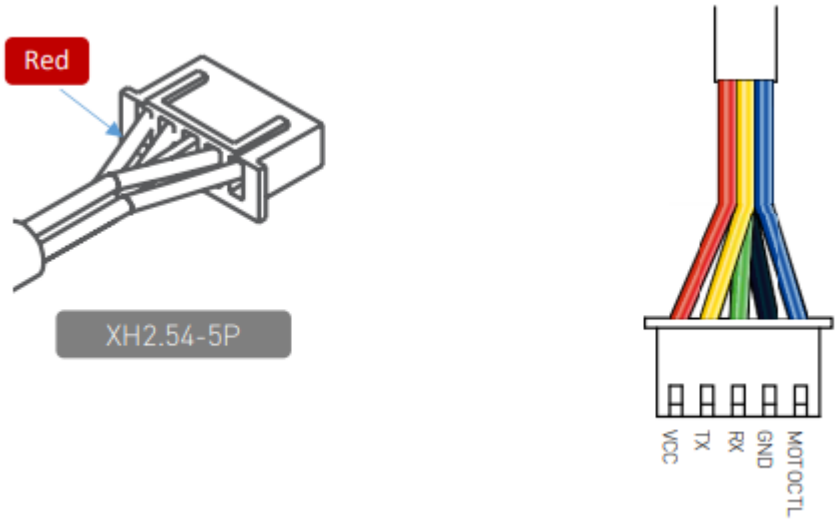


Figura 13. Conexionado del RPLIDAR A2M8 SLAMTEC.

Cada uno de los cables presenta una funcionalidad diferente, a saber:

- Rojo: Es el cable de alimentación, debe estar conectado a una fuente de tensión continua de valor 5 V.
- Amarillo: Se trata de un puerto de salida en serie, a través del cual se puede transmitir la información obtenida en los sucesivos escaneos. El protocolo empleado para la transmisión de la información aparece explicado con mayor nivel de detalles en las especificaciones del producto proporcionadas por el fabricante.
- Verde: Es el puerto de entrada serie del sensor. A través de él, es posible enviar información, en el formato apropiado, para activar el motor, comenzar los escaneos y demás funciones desempeñadas por el LiDAR.
- Negro: Es el cable de conexión a tierra.
- Azul: Mediante este cable es posible controlar el motor a través de señales PWM (Pulse-Width Modulation)

La toma de medidas y su posterior transmisión mediante las Raspberry Pi modelo 3B+ presentes en el vehículo, se han realizado mediante un driver desarrollado en Python con ayuda de la librería rplidar. Más información al respecto se presenta en el 5.1.

Capítulo 4. SOFTWARE

Para la toma de medidas con el sensor RPLIDAR A2M8, así como para establecer la comunicación entre las Raspberry Pi 3B+ y procesar posteriormente la información extraída para finalmente permitir al vehículo navegar de forma autónoma mediante un control de seguimiento de pared, se han empleado diferentes recursos de software que han permitido el desarrollo del proceso descrito.

Se procede en este apartado a presentar la información necesaria sobre:

- Matlab y Simulink
- Python
- Ubuntu 20.04.4 LTS

4.1 MATLAB Y SIMULINK

Matlab, que es una abreviatura de Matrix Laboratory, es un lenguaje de programación interpretado que vio la luz en 1984 a manos de su creador Cleve Moler [25]. Se trata de un lenguaje de programación basado en el uso de matrices para la gran mayoría de cálculos, que se concibió como una forma de acercar, de manera sencilla, ciertas ramas matemáticas al mundo de la programación.

Su creador, Cleve Moler, fue profesor de matemáticas y ciencia de computadores en diferentes universidades como la Universidad de Michigan o la de Stanford, llegando su periodo docente a alcanzar los 20 años. Durante su carrera como profesor en la universidad de Nuevo México, impartía las asignaturas de Álgebra Lineal y Análisis Numérico. Fue en esa época donde comenzó el desarrollo de lo que hoy en día conocemos como Matlab.

Por aquel entonces, los alumnos se servían del lenguaje de programación Fortran [26], empleado principalmente en aplicaciones de computación numérica, para realizar sus

cálculos y tareas universitarias. En particular las librerías empleadas principalmente en las aplicaciones de cálculo mediante Fortran eran EISPACK y LINPACK, siendo el propio Cleve Moler uno de los autores de las mismas. La complejidad del proceso empleado por Fortran llevó a Cleve a comenzar como pasatiempo, la elaboración de un nuevo lenguaje de programación, basado en Fortran, que permitiera un uso más sencillo e intuitivo de los comandos fundamentales empleados por los alumnos en sus programas. El tipo de dato fundamental de este nuevo programa eran las matrices, de donde surgió el nombre de Laboratorio de Matrices o, por su abreviatura, Matlab.

A pesar de la sencillez inicial del lenguaje, contando con un número reducido de funcionalidades, el propio Cleve iba añadiendo nuevas a medida que avanzaban los años, basándose en necesidades propias y las mismas sugerencias de los alumnos.

A pesar de no tener conocimiento alguno sobre teoría de control, reconoció el potencial de aplicación de Matlab, basado en matrices, para resolver los problemas propios de este ámbito. Es así como este lenguaje comenzó a ser empleado por gran cantidad de alumnos en asignaturas basadas en el control y la automatización.

En los años siguientes Matlab fue recibiendo nuevas funcionalidades, presentando a día de hoy un gran número de herramientas y Toolboxes, paquetes desarrollados por personas o entidades que ofrecen funciones para resolver determinados problemas específicos. El ámbito de aplicación de Matlab es muy extendido empleándose en aplicaciones de control, automatización, visión por computación y un largo etcétera.

	MATLAB	Fortran Code Segment
S1	`n=floor(11.5);`	`n=floor(10.5);`
S2	`for i=1:n`	`DO i = 1, n`

S3	`  x=1+2*i;`	`  x=(1+(2*i));`
		`  IF((.NOT.ALLOCATED(A)))THEN`
		`  ALLOCATE(A((1+(2*n))));`
		`  END IF`

S4	`  A(x)=i;`	`  A(x)=i;`
S5	`end`	`END DO`
S6	`n=fix(n/2);`	`n=fix(n/2);` `ArrayBoundsChecking(A,[n+1]);`
S7	`A(n+1)=n;`	`A(n+1)=n;`

Figura 14. Comparativa entre código Matlab y Fortran.

Una de las herramientas fundamentales de Matlab es Simulink. Se trata de un entorno de programación gráfico óptimo para labores de modelado, análisis de sistemas dinámicos de diferente naturaleza y tareas de simulación. Se basa en diagramas de bloques y es muy empleado en aplicaciones de control, automatización o procesamiento de señales digitales, entre otras. Además, es compatible con Matlab, permitiendo interpretar programas escritos en lenguaje Matlab así como escribir en ese lenguaje en los propios bloques de Simulink para ejecutar diversas funcionalidades.

Cabe destacar además la compatibilidad de Matlab con otros lenguajes de programación a través de su API, estando disponible en Python, C++, Java, Fortran, entre otros.

Para el desarrollo de este proyecto se ha optado por el uso de Matlab y Simulink debido a las funcionalidades que ofrecen.

4.2 PYTHON

Se trata de un lenguaje de programación de alto nivel, es decir, los algoritmos y funciones se expresan de tal manera que resulta comprensible para la capacidad cognitiva humana, en lugar de asemejarse al código máquina, interpretados directamente por los circuitos programables [27]. Es además un lenguaje interpretado, por lo que las instrucciones se ejecutan según están escritas en el propio programa, sin necesidad de tener que ser compiladas previamente, como ocurre en otros lenguajes de programación. Es compatible con diferentes paradigmas de programación como el clásico paradigma procedural, en el que el programa se ejecuta línea a línea en el orden en que está escrito, así como programación orientada a objetos o programación funcional.

Este lenguaje de programación fue concebido a finales de la década de los 80 por Guido van Rossum, programador de origen holandés, viendo la luz finalmente en 1991 con la versión Python 0.9.0. Cabe destacar de esta primera versión, la existencia de herencia entre clases, así como diferentes tipos de variables, incluyendo entre ellos las listas o los diccionarios.

Python evoluciona a lo largo de los años con constantes actualizaciones y cambios de versión, pasando a Python 2.0 en el año 2000 y a la versión Python 3.0 en el año 2008.

Para este proyecto Python se ha empleado para el proceso de toma de datos del RPLIDAR A2M8 y su posterior transmisión a la Raspberry Pi de control. Para ello se han empleado determinadas librerías con funciones implementadas que han permitido agilizar el proceso de desarrollo.

4.3 UBUNTU 20.04.4 LTS

Para poder usar la Raspberry Pi, es necesario instalar en ella una imagen. A modo de resumen, una imagen es una copia de un sistema operativo, incluyendo aquellos elementos de software necesarios para la correcta ejecución del mismo.

El software de la Raspberry está basado en Linux, sistema operativo basado en el software libre y de código abierto. Aunque el término Linux adquiere este significado en las aplicaciones del día a día, realmente hace referencia al núcleo (kernel), es decir, la parte que permite la comunicación segura entre el software del dispositivo y los diferentes elementos del hardware. El sistema operativo en su conjunto es una unión de elementos tanto de Linux como de GNU (GNU's Not Unix) y otros elementos de terceros.

Fue en 1983 cuando Richard Stallman, físico y programador con origen en Manhattan (Estados Unidos) inició el proyecto GNU [28], siendo el principal objetivo la popularización de sistemas basados en el software libre. Durante los años siguientes al inicio del proyecto, se fundó la Free Software Foundation acompañado de la redacción de un documento, la Licencia Pública General de GNU, que garantiza a los usuarios que emplean elementos de software libre, la posibilidad de modificarlo, compartirlo y estudiarlo sin restricción alguna.

Ante la situación de que en 1991 el núcleo del proyecto comenzado por Stallman no se encontraba desarrollado, Linus Torvalds, ingeniero de software, comenzó el desarrollo de su propio núcleo. La iniciativa se dio debido a la inconformidad del usuario con el sistema de licencias empleado por el sistema operativo MINIX. Fue así como, basándose en el sistema MINIX, Linus Torvalds desarrolló el núcleo o kernel Linux, alcanzando rápidamente una gran popularidad y extendiendo su uso de manera acelerada.

La imagen empleada en el proyecto de navegación autónoma es Ubuntu 20.04.4 LTS. Ubuntu es una distribución de Linux, cuyo desarrollo está basado en Debian y cumple con los valores de creación de Linux de software libre y abierto. El término LTS se refiere a Long Term Support, haciendo referencia a que se trata de una versión que recibirá constantes actualizaciones tras un periodo de 6 meses y durante los 5 años próximos a su lanzamiento.

Capítulo 5. DESARROLLO DEL PROYECTO

Para presentar el desarrollo del proyecto, se procede a dividir el mismo en diferentes secciones explicando los distintos procesos y funcionalidades en cada una de ellas.

5.1 OBTENCIÓN DE MEDIDAS DEL LIDAR

La primera parte del proyecto consiste en la obtención de medidas mediante el sensor LiDAR. Esas medidas serán posteriormente empleadas para, mediante los cálculos y técnicas de algorítmica necesaria, lograr que el vehículo sea capaz de realizar un control de seguimiento de pared. Para la obtención de las medidas se ha optado por emplear Python para el desarrollo del código.

Se ha empleado la librería rplidar, que ha sido desarrollada por el usuario de github conocido por el nombre de SkoltechRobotics. Dicha librería permite la comunicación con los modelos A1 y A2 de RPLIDAR mediante la clase RPLidar. Dicha clase presenta una serie de métodos que permiten, entre otras cosas, activar el motor del LiDAR y desactivarlo, determinar información relativa al correcto funcionamiento del LiDAR así como obtener las medidas del LiDAR en los sucesivos escaneos realizados.

El código empleado para la realización del proyecto se presenta completo en el Anexo II, sin embargo, se procede en este apartado a analizar el código dividido en diferentes bloques.

```
from rplidar import RPLidar
import time
import numpy as np
import struct
import socket
```

Esta primera parte del código permite importar las librerías que se van a emplear a lo largo del código. La librería rplidar permite realizar las operaciones necesarias para la toma de

medidas del LiDAR. Por su parte, el resto de librerías son empleadas para otras funciones que se describirán a lo largo del código.

```
PORT = 7000
SERVER = '192.168.0.123'
ADDR = (SERVER,PORT)
```

La transmisión de datos entre la Raspberry de navegación y la Raspberry de control se realiza mediante protocolo TCP/IP a través de cable Ethernet. El funcionamiento de dicho protocolo se explica con mayor claridad en el Anexo I. Es necesario, para la transmisión de datos, que exista un servidor y un cliente. En este caso hará las veces de servidor la Raspberry de control, cuya IP en comunicación Ethernet se ha establecido en 192.168.0.123, mientras que la IP de la Raspberry de navegación es 192.168.0.124. La comunicación se establece a través del puerto 7000.

```
#Desconectamos LIDAR en caso de que esté encendido
lidar = RPLidar('/dev/ttyUSB0')

lidar.stop()
lidar.stop_motor()
lidar.disconnect()
```

Comenzando por el main, se empieza desconectando el LiDAR. Esto se realiza por si se ejecuta el programa cuando el LiDAR se encuentra con el motor activado, provocando futuros errores a la hora de la toma de medidas.

```
#Conectar a socket
client = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connected = False

while connected==False:
    try:
        time.sleep(3000)
        b = client.connect(ADDR)
        if b==None:
            connected = True

    except TimeoutError:
        print('Not connected yet...')
    except ConnectionRefusedError:
        print('Not connected yet...')
```

Para realizar la transmisión de datos por protocolo TCP/IP es necesario conectarse al servidor (Raspberry de control). Se crea por tanto una variable client y se trata de conectar al servidor previamente definido por su dirección IP y su puerto. Esta parte del código trata de conectarse al servidor cada 3 segundos. En caso de que el servidor no esté activo en ese momento o que exista un error de conexión al mismo, se repite el proceso hasta que sea posible la conexión.

```
#Mandar un 1 para activar
recibido_1 = False

while recibido_1 == False:
    aceptar = client.recv(8)
    if int.from_bytes(aceptar,byteorder='big')==1:
        recibido_1=True
```

Una vez conectado el cliente al servidor, se espera a recibir un byte con valor 1 por parte del servidor. Hasta que no se reciba el mismo, el LiDAR no comenzará a realizar su escaneo.

```
time.sleep(0.5)

#Encender LiDAR
lidar = RPLidar('/dev/ttyUSB0')

#Información del LiDAR
info = lidar.get_info()
print(info)

#Estado del LiDAR
health = lidar.get_health()
print(health)

#Bucle funcionamiento
while True:

    get_measurements(lidar,client)
    lidar.stop()
    lidar.stop_motor()
    lidar.disconnect()
    time.sleep(0.01)
    lidar = RPLidar('/dev/ttyUSB0')


lidar.stop()
lidar.stop_motor()
lidar.disconnect()
```

Tras la autorización por parte del servidor del inicio de la toma de medidas, se procede a conectar con el puerto en que se encuentra conectado el LiDAR (/dev/ttyUSB0), mostrando por pantalla información relativa al sensor, así como el estado del mismo. Esta información es opcional, pero resulta de utilidad mostrarla por pantalla en caso de disponer de ella, para determinar que todo funciona correctamente.

Una vez realizadas las inicializaciones pertinentes, se procede a comenzar con la toma de muestras, llevada a cabo mediante la función get_measurements. En caso de que exista un error durante la toma de muestras, el programa saldrá de la función, desconectará el LiDAR durante un breve periodo de 0.5 ms para posteriormente reconectar y volver a ejecutar la función.

```
#Funcion medidas

def get_measurements(lidar,client):
    try:
        print('BEGIN SCAN...')
        measurements = np.full(40,-1.0) #angle-measurement (20 of each)
        i = 0
        for item in lidar.iter_measurments(max_buf_meas=1000):

            if i==0: #Inicio medidas
                start=time.time()
            if item[3]!=0: #Si la medida es válida la añadimos, si no, -1.0
                measurements[i] = item[2]
                measurements[i+1] = item[3]/1000  #Medida en metros

            i+=2

            if time.time()-start>=0.01 or i>=39:
                if time.time()-start<0.01:
                    time.sleep(0.01-(time.time()-start)) #wait till 10 ms
                #print('time: ',time.time()-start)

                a=bytearray(struct.pack('>fffffffffffffffffffffffffffffffffffffff',
                measurements[0],measurements[1],measurements[2],measurements[3],
                measurements[4],measurements[5],measurements[6],
                measurements[7],measurements[8],measurements[9],
                measurements[10],measurements[11],measurements[12],
                measurements[13],measurements[14],measurements[15],
                measurements[16],measurements[17],measurements[18],
                measurements[19],measurements[20],measurements[21],
                measurements[22],measurements[23],measurements[24],
                measurements[25],measurements[26],measurements[27],
                measurements[28],measurements[29],measurements[30],
                measurements[31],measurements[32],measurements[33],
```

```
                measurements[34],measurements[35],measurements[36],
                measurements[37],measurements[38],measurements[39])).hex()

                msg = bytes.fromhex(a)

                client.send(msg)
                measurements = np.full(40,-1.0)
                i=0

    except:
        #Error
        return
```

Se define la función encargada de la toma de medidas. La función recibe como argumento el objeto lidar, que permite acceder a los métodos de la librería rplidar. Cada una de las medidas del objeto lidar tiene 3 datos de información. La distancia en milímetros al punto medido, el tiempo empleado desde que el haz de luz es emitido hasta que llega al receptor, y una tercera medida que indica la calidad del haz de luz recibido. Si esta última no alcanza un cierto umbral, la medida será considerada no válida y el valor de la distancia aparecerá automáticamente como 0.

Para organizar la información obtenida, se comienza creando un vector de longitud 40 rellenando todas sus posiciones con el valor -1.0. Esto permitirá filtrar las medidas inválidas, cuyos valores de ángulo y distancia se mantendrán en -1.0. Se ha seleccionado el valor de 40 ya que se busca enviar información correspondiente a 20 medidas cada 10 milisegundos, es por ello que se requiere de un vector que puede albergar 40 medidas, ya que cada punto requiere de un ángulo y distancia para su correcta caracterización.

A continuación, se iteran las medidas obtenidas por el LiDAR mediante el método lidar_iter_measurements(). Como parámetro se ha establecido un número máximo de medidas en buffer de 1000. Es por eso que se inicializa la variable start, que guarda información sobre el tiempo en que se comenzó la toma de medidas.

Como se ha mencionado, en caso de que la medida sea válida, se procede a guardar el ángulo y la medida correspondiente a ese ángulo en el vector de medidas. Este proceso se repite hasta que el vector de medidas haya sido rellenado (incluyendo los valores no válidos) o hasta que el tiempo transcurrido desde el inicio de la toma de medidas sea superior a 10

milisegundos. Además, en el primer caso, si se ha rellanado el vector, pero el tiempo empleado para ello ha sido menor que los 10 ms necesarios, se realiza una pausa, mediante la librería time, hasta alcanzar ese tiempo.

Una vez está listo el vector con los datos, ya sea por haber rellenado los huecos correspondientes a datos válidos o por haber completado el tiempo de 10 ms, se procede a enviarlo mediante protocolo TCP/IP a través de cable Ethernet. Para ello, es necesario, en primer lugar, encriptarlo. Se emplea la librería struct que mediante el método pack, permite convertir los datos aportados en la representación mediante cadena de caracteres deseada. Por ello, se pasan como argumentos al método, las 40 medidas obtenidas y la representación de las mismas. Al tratarse de medidas tipo float se emplea la letra f para determinar que se trata de ese tipo de dato numérico.

El resultado de este proceso es recibido como argumento por la función bytearray(), que convierte la cadena de caracteres recibida en una cadena de datos expresado en hexadecimal.

Por último, mediante la función bytes_from_hex() se obtiene la cadena de bytes a partir de la representación hexadecimal de los mismos. Este tipo de variable ya puede ser enviada mediante protocolo TCP/IP. Es así como mediante la variable client, se realiza el envío de la información.

Se ha incluido además gestión de excepciones. Durante la toma de muestras, al realizarse operaciones dentro del bucle, así como esperar ciertos periodos de tiempo hasta completar los 10 ms, existen ocasiones en las que se desincroniza la toma de medidas con las medidas actuales del LiDAR, dando lugar a errores. Esto es debido a que la librería empleada, se sirve de un flag para determinar cuando comienza un nuevo escaneo del sensor. En caso de que ocurra esto, se ha incluido el comando return para salir de la función.

5.2 PROCESAMIENTO DE LAS MEDIDAS

Una vez las medidas del LiDAR han sido obtenidas y enviadas por protocolo TCP/IP a la Raspberry de control, es necesario llevar a cabo el procesamiento de las mismas con el objetivo de poder llevar a cabo el control de seguimiento de pared. El procesamiento y monitorización será llevado a cabo mediante código de Matlab y Simulink.

Previo al procesamiento, es necesario configurar ciertos parámetros para el correcto funcionamiento del control. Uno de los parámetros de importancia es la referencia de distancia a la pared. Dicho valor determinará la distancia que el control tratará de mantener respecto a la pared, controlando para ello los mandos empleados en el control. En el archivo de Matlab CONFIG_CONTROL.m es posible determinar el tipo de referencia que se aplicará a la distancia a la pared, ya sea una referencia constante, una señal cuadrada, pulso, además de otras opciones. Para el desarrollo de este proyecto se ha optado por emplear como referencia una señal de valor constante de 25 cm. El valor por defecto de la referencia constante es de 10 cm, sin embargo, es posible determinar un valor personalizado como en este caso. En este mismo fichero es posible configurar otros parámetros relativos a los controles disponibles así cómo al filtro extendido de Kalman.

Otro parámetro que es necesario determinar es la posición del LiDAR relativa al marco de referencia del vehículo (cuyo centro de coordenadas se localiza en el centro del eje de los motores de continua). Las medidas obtenidas por el proceso descrito en el apartado anterior se refieren al sistema de referencia del propio LiDAR, sin embargo, es necesario realizar una traslación de las medidas obtenidas al sistema del vehículo. Es en el archivo MODEL_IDENT.m donde es posible determinar dichos valores. En el apartado destinado al LiDAR 2D, existen dos variables que permiten determinar el punto en que se encuentra el LiDAR a partir de sus dos coordenadas en el sistema cartesiano, relativas al sistema de referencia del vehículo. Tras realizar medidas sobre el sistema LiDAR – vehículo, se determinó que la distancia del centro del LiDAR se encontraba en el punto (10.5,-3.5) cm respecto a la referencia del vehículo.

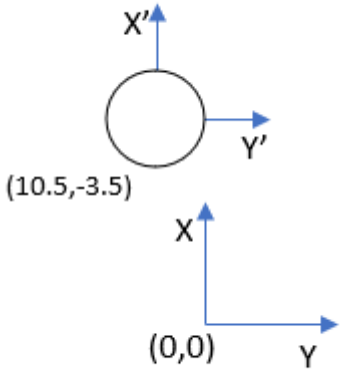


Figura 15. Sistema de referencia LiDAR-vehículo (distancias en centímetros)

Una vez se ha realizado la traslación de las medidas al sistema de referencia del LiDAR, es posible realizar los procedimientos necesarios para el control de seguimiento de pared.

5.2.1 MÍNIMOS CUADRADOS

Previo a emplear el filtro extendido de Kalman para las estimaciones pertinentes que servirán para el control de seguimiento de pared, se ha optado por realizar una estimación de la distancia del vehículo a la pared por medio del método de mínimos cuadrados.

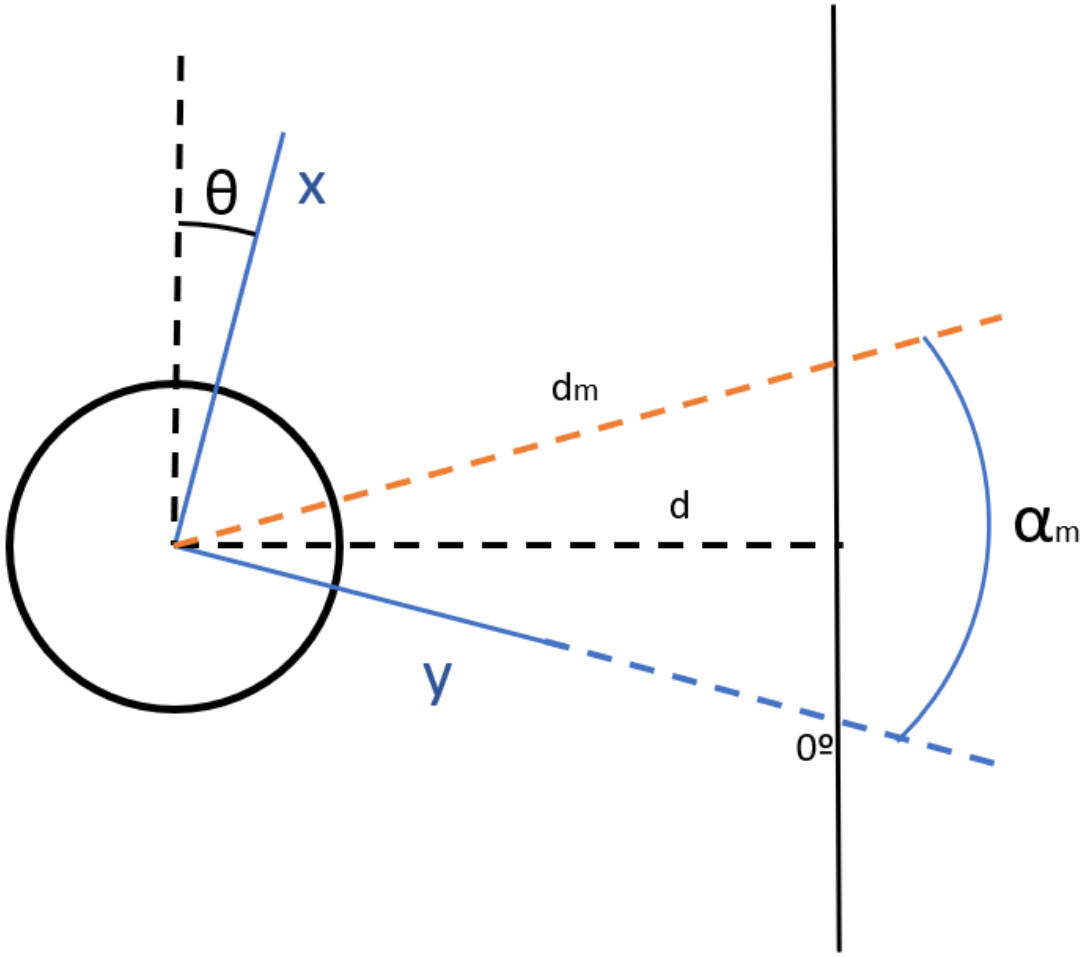


Figura 16. Esquema para el algoritmo de mínimos cuadrados.

En la figura 16, se presenta el esquema empleado para el algoritmo de mínimos cuadrados. En él se presenta el LiDAR, representado por un círculo como simplificación, junto con su sistema de coordenadas, cuya orientación coincide con el sistema de coordenadas del vehículo. Cabe destacar que las medidas del LiDAR son trasladadas como se mencionó anteriormente al sistema de referencia del vehículo.

Se establece como referencia de ángulo 0º con la pared a la intersección entre la pared y la prolongación del eje y. Para el algoritmo de mínimos cuadrados, se recogen aquellas medidas situadas entre -45º y 45º referidas a la referencia 0º de la pared que cumplan además que su distancia a la misma se encuentre en el rango de 0.1 a 0.4 metros. En el esquema mencionado, el ángulo de la medida aparece reflejado por la variable α_m, que toma valores positivos en sentido antihorario y negativos en sentido horario. Se denota además como *d* a la distancia del centro del LiDAR a la pared, como θ, al ángulo que forma el mismo con la pared y d_m denota la distancia medida por el LiDAR para un ángulo α_m.

A partir de las variables descritas, es posible obtener una expresión para la distancia a la pared *d*.

$$d = d_m * \cos(\alpha_m - \theta)$$

Ecuación 3.

Sabiendo además la relación del coseno de la resta de ángulos, podemos manipular la expresión anterior.

$$d = d_m * (cos\alpha_m * cos\theta + sen\alpha_m * sen\theta) = d_m * cos\alpha_m * cos\theta + d_m * sen\alpha_m * sen\theta$$

Ecuación 4.

Si se despeja la variable que describe la distancia a la pared *d*, se obtiene la expresión que se emplea en el algoritmo de mínimos cuadrados.

$$1 = d_m * cos\alpha_m * \frac{cos\theta}{d} + d_m * sen\alpha_m * \frac{sen\theta}{d}$$

Ecuación 5.

Se obtiene por tanto una expresión que depende de las medidas del LiDAR, α_m y d_m, así como de los parámetros $\frac{cos\theta}{d}$ y $\frac{sen\theta}{d}$, que serán los valores a estimar por el método de mínimos cuadrados.

Una vez deducida la ecuación a emplear, es posible expresarla de forma matricial para facilitar el proceso de resolución.

$$1 = [d_m * \ cos\alpha_m \quad d_m * \ sen\alpha_m] * \begin{bmatrix} \frac{cos\theta}{d} \\ \frac{sen\theta}{d} \end{bmatrix}$$

Ecuación 6.

La ecuación 6, trata una simplificación del problema ya que presenta una única observación. No obstante, en el caso de estudio se ejecuta el algoritmo de mínimos cuadrados cuando existen un número de medidas superior a 5 en el rango -45º a 45º, así como una diferencia entre el ángulo máximo y mínimo medidos de al menos 22.5º. Dichas restricciones permiten obtener una estimación más robusta de la distancia a la pared.

En el caso generalizado, se tiene una matriz de unos de tamaño N*1 a la que denotaremos como 1_N, siendo N el número de observaciones en el rango mencionado. Se tiene además la matriz equivalente al caso de una observación, construida a partir de las distancias y ángulos medidos para cada medida dentro del rango de estudio del LiDAR, denotada como A. Por último, la matriz de parámetros a estimar por el algoritmo de mínimos cuadrados permanece igual que en el caso de una observación, y la denotaremos como B. De tal manera, la formulación matricial para el caso general se presenta en la ecuación 7, empleando la notación descrita en este mismo párrafo.

$$1_N = A * B$$

Ecuación 7.

Expresado de esta manera, es posible resolver el problema de mínimos cuadrados que otorgará la estimación de la distancia del vehículo a la pared.

5.2.2 FILTRO EXTENDIDO DE KALMAN

El filtro de Kalman fue creado en 1960 por el ingeniero eléctrico, matemático e inventor Rudolf Kalman. Mediante este algoritmo es posible determinar el estado de una variable a la que no se tiene acceso directo para realizar mediciones, a partir de una predicción determinada por el valor de otras variables del sistema. Dicho sistema debe ser un sistema lineal en caso de tratarse de un filtro de Kalman. Si, por el contrario, y como es el caso de estudio de este proyecto, el sistema es no lineal, se emplea el algoritmo conocido como filtro extendido de Kalman (EKF).

Se trata de un algoritmo recursivo, que se basa en criterios estadísticos de incertidumbre para el cálculo de las estimaciones de las variables. Para su funcionamiento, además de una matriz de incertidumbre de las diferentes mediciones, se requiere las propias mediciones, así como el valor de las mismas en el periodo de muestreo anterior. Los filtros de Kalman son muy empleados en aplicaciones de navegación, así como navegación autónoma de vehículos.

Tanto en el filtro de Kalman como en el filtro extendido de Kalman, existe una matriz de ganancias que debe ser ajustada en cada periodo de muestreo en el caso del filtro extendido de Kalman, mientras que, en el filtro de Kalman, para sistemas lineales, la matriz de ganancias permanece constante durante todo el proceso. Este último algoritmo presenta una estructura idéntica a la de un observador de estado de orden completo.

A modo de resumen y sin entrar en profundidad en deducciones, las ecuaciones diferenciales empleadas para el modelo no lineal del filtro de Kalman se presentan a continuación.

$$\frac{dw}{dt} = -\frac{1}{T_m}w + \frac{K_m}{T_m}u_d$$

Ecuación 8.

$$\frac{d\alpha}{dt} = w + w_{pared}$$

Ecuación 9.

$$\frac{dd_n}{dt} = -(v - wy_A)sen\alpha - wx_A cos\alpha$$

Ecuación 10.

Donde w denota la velocidad angular del vehículo, d_n la distancia mínima a la pared desde el LiDAR, T_m es la constante de tiempo de la función de transferencia que relaciona la tensión diferencial de los motores con la velocidad angular del vehículo, K_m es la ganancia de esa misma función de transferencia. Por su parte v denota la velocidad de avance del vehículo y las coordenadas x_A e y_A hacen referencia a la ubicación del LiDAR en el marco de referencia del vehículo autónomo. Por otra parte, α hace referencia al ángulo que forma el vehículo con la pared, tomando como referencia del vehículo su eje longitudinal. El término w_{pared} es una perturbación que puede explicarse con un pulso.

$$w_{pared} = \frac{\pi}{T_p} = \frac{\pi}{\frac{\pi * r_{curva}}{v}} = \frac{v}{r_{curva}}$$

Ecuación 11.

Quedan así definidas las ecuaciones no lineales empleadas por el filtro extendido de Kalman encargado de controlar el seguimiento de pared por parte del vehículo.

Existen, además, otros filtros de Kalman empleados en el control del vehículo. Un EKF encargado del control de la velocidad de avance y dos filtros de Kalman lineales encargados del control de la velocidad angular.

5.2.3 EJECUCIÓN Y MONITORIZACIÓN

Para la ejecución y puesta en marcha del proyecto, una vez instalado el firmware con la configuración pertinente en el vehículo, se debe emplear el fichero Simulink PC_CONTROL_STATION.slx. Cuando el programa haya cargado y se encuentre en estado de ejecución, se creará un servidor con la IP descrita en el capítulo 5.1. La Raspberry de navegación, que en este punto y, tras ejecutar el script de Python para la obtención de medidas del LiDAR, validará la conexión al servidor. Tras una primera calibración de los sensores del vehículo, una vez se presione uno de los botones del mismo, se enviará una señal para activar el motor del LiDAR y comenzar la toma de muestras, así como el avance y control de seguimiento de pared del vehículo.

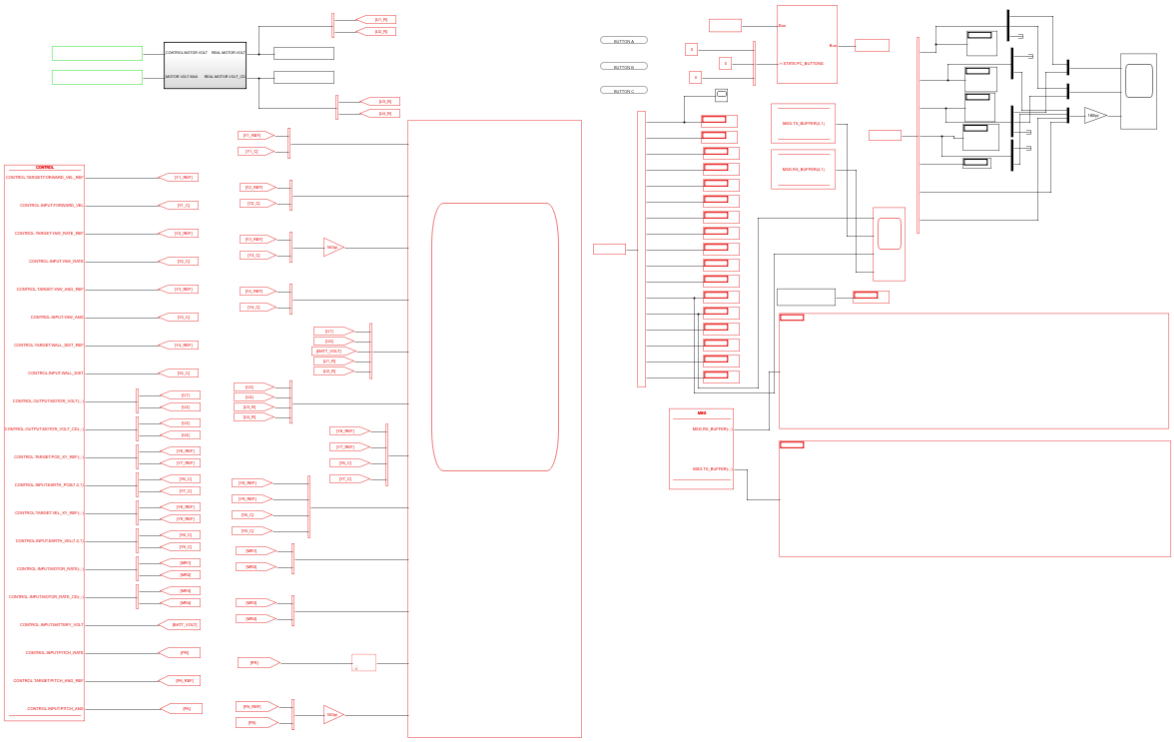


Figura 17. Diagrama de bloques Simulink encargado de la monitorización de las variables.

En la figura 17, se muestra la parte del archivo Simulink destinada a la monitorización de diferentes señales durante el proceso de seguimiento de pared.

Capítulo 6. ANÁLISIS DE RESULTADOS

Durante el desarrollo del proyecto se han realizado diversas pruebas para comprobar el correcto funcionamiento de los diferentes algoritmos y del código desarrollado.

Las primeras pruebas trataron de determinar que el código Python encargado de la toma de medidas del LiDAR funcionaba de forma correcta. Recalcar que dicho código enviaba a la Raspberry de control del vehículo, un vector de 40 posiciones, relleno con los diferentes ángulos acompañados de su medida. En caso de que la medida fuese no válida, el valor tanto de medida como de ángulo correspondiente a la misma permanecería con valor -1.0.

A partir de esta información, se decidió tratar de realizar un mapa del entorno. Para ello se inhabilitaron los motores del vehículo, así como otras funcionalidades mediante el código de Simulink pertinente para evitar que el vehículo se moviese y así, permaneciese estático para el mapeo del entorno. Durante un periodo de 10 segundos, se recogieron las medidas del LiDAR, rellenando un vector de distancias de longitud 360, al tratarse de un mapa con una resolución de 1º. Se realizaba una aproximación del ángulo obtenido al entero más cercano para lograr la resolución mencionada. Además, en caso de existir diversas medidas referidas al mismo ángulo de rotación del LiDAR, se realizaba una media ponderada de las mismas para una mayor precisión en la medida.

La prueba se realizó en el circuito de prueba presente en el laboratorio de universidad ICAI. Los resultados obtenidos se presentan a continuación en la figura 19.

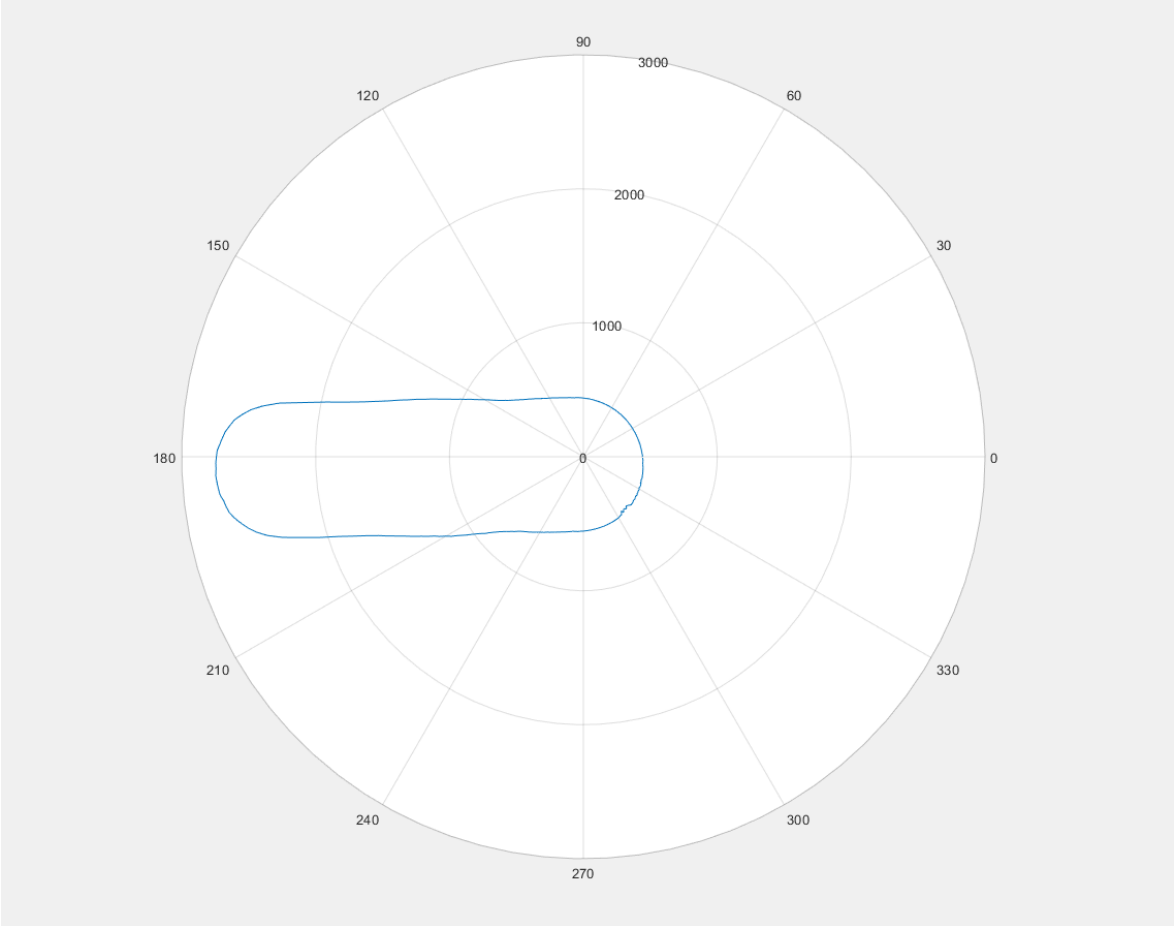


Figura 18. Mapa del circuito del laboratorio de ICAI.

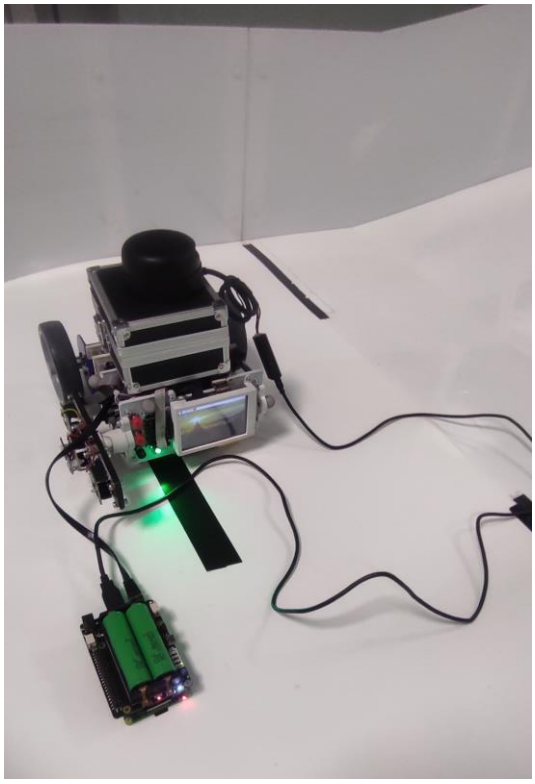


Figura 19. Vehículo en el circuito para el mapeado.

En la figura 18, se observa el mapa creado a partir de las medidas del LiDAR, con una precisión de 1º en las mismas. Se trata de un primer acercamiento al objetivo final del proyecto, ya que permite determinar la precisión de las medidas obtenidas. El mapa resultante presenta una precisión casi perfecta, reflejando la pista de pruebas en la que fue ubicado para la misma. El rango de medidas comprendido cerca de los 300º presenta una ligera perturbación. Esta es debida a que, durante la prueba, era necesario presionar uno de los pulsadores para iniciar las medidas del LiDAR. Por tanto, durante el primer escaneo, se detectó la mano con la que fue pulsado el pulsador como un objeto a una distancia más próxima que la pared. Al promediar el resto de medidas de ese intervalo angular con la obtenida, el resultado fue una ligera perturbación en la distancia de ese mismo rango.

Una vez comprobado el correcto funcionamiento de la toma de medidas del sensor, se procede a realizar ensayos para el seguimiento de pared. El procedimiento de dichos ensayos ha sido descrito en el capítulo 5, realizando cambios en la distancia de seguimiento de pared por defecto, la cual fue cambiada de 10 cm a 25 cm y el peso asignado a cada incertidumbre en el filtro extendido de Kalman, proceso que se realizó de forma manual tratando de adaptarlos según los resultados de ensayos previos. Tras sucesivos cambios en los parámetros de interés para el control de seguimiento de pared, se consiguió un intento exitoso. La monitorización de dicho intento aparece reflejada en la figura 19.

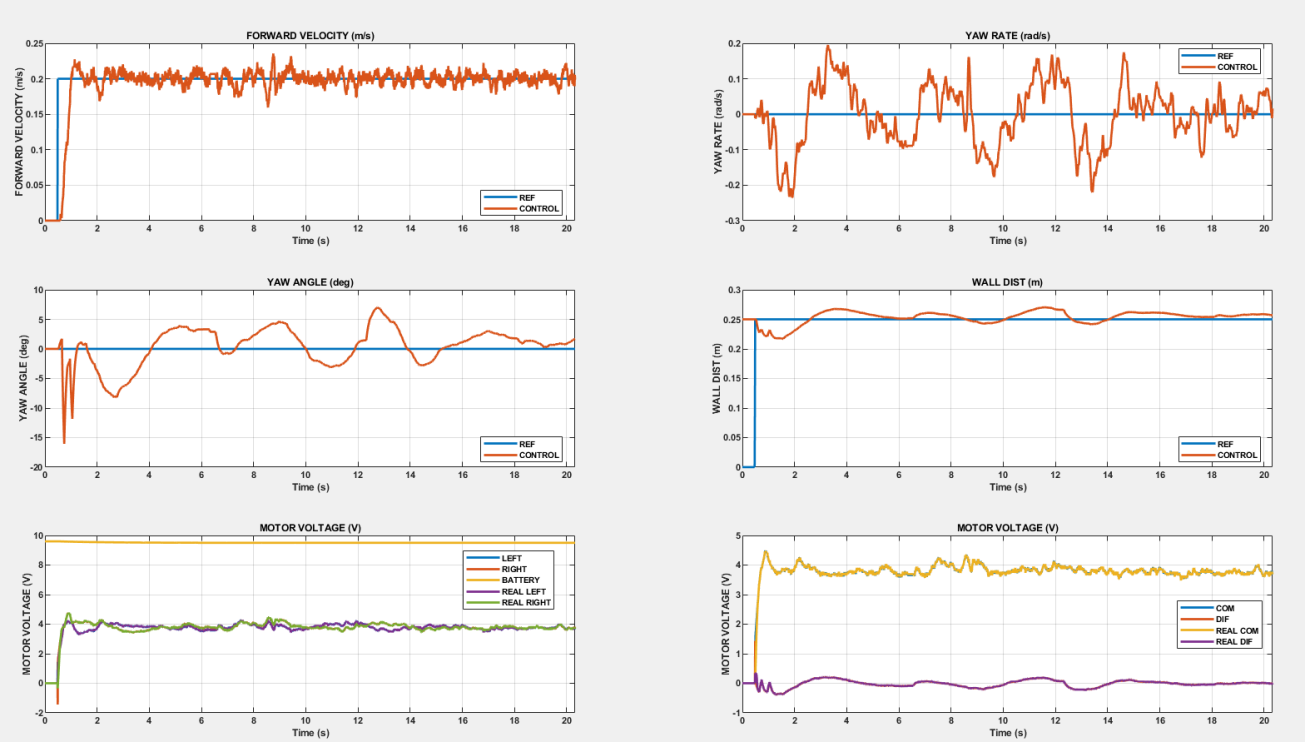


Figura 20. Resultados del control de seguimiento de pared.

Se han decidido incluir aquellas variables que aporten la mayor cantidad de información sobre el control a evaluar. En cuanto a la velocidad de avance, se estableció una referencia constante de 0.2 m/s. El control de velocidad está regido por un filtro extendido de Kalman encargado únicamente de ello. A pesar de existir ligeras variaciones, debido a la corrección

de la distancia del vehículo a la referencia de la pared, el valor permanece en su mayoría constante.

La distancia a la pared presenta en ciertos tramos diferencias en la referencia que alcanzan valores de hasta 2.5 cm, no obstante, el control mediante el trabajo conjunto del algoritmo de mínimos cuadrados y el EKF permite un seguimiento de la referencia que cumple los estándares marcados para el proyecto. Se puede además visualizar en conjunto con la velocidad angular ya que se observa en la gráfica como cuando la distancia a la pared se aleja de la referencia, entra en juego el control y la velocidad angular se ve modificada para lograr paliar los efectos de dichas perturbaciones.

Al tratarse de un seguimiento de una pared recta, el resultado esperado es que la tensión aplicada a los motores sea igual en ambos, al tratarse de un vehículo de tracción diferencial. Los resultados obtenidos están en consonancia con lo esperado, aplicándose una tensión de aproximadamente 4 V en cada uno de los motores. Por su parte, la tensión diferencial ronda los 0 V ya que sólo entra en acción cuando es necesario corregir mediante un giro la distancia a la referencia. Si se observa la gráfica en la que aparece representada la tensión diferencial, se observa que cuando la velocidad angular es positiva, esta también lo es, y de forma análoga cuando presenta valores negativos.

Con todo esto concluye el propósito del proyecto, habiéndose conseguido un seguimiento de pared y teniendo unos resultados que tienen un sentido físico y están en consonancia entre ellos.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

El proyecto logró cumplir con los objetivos principales que se habían marcado.

- Obtención de medidas de ángulo y distancia del LiDAR 2D.
- Gestión de la comunicación por protocolo TCP/IP entre la Raspberry de navegación y la de control.
- Desarrollo de algoritmos necesarios para lograr el control de seguimiento de pared.

El logro de estos objetivos se ha conseguido a partir de los procesos descritos a la largo de este trabajo. En el desarrollo de proyectos futuros relacionados con este tema es posible emplear otras herramientas que permitiesen lograr los mismos resultados de forma más eficiente y sencilla. Una de las opciones estudiadas es el uso de ROS 2.

ROS (Robot Operating System) es un conjunto de librerías de software de uso libre que permiten desarrollar multitud de aplicaciones relacionadas con el campo de la robótica. Se trata de una solución con un gran potencial debido a la comunidad existente dentro de este ámbito. La gran mayoría de aplicaciones sencillas relacionadas con la robótica están ya resueltas y el código empleado por ROS se encuentra disponible en diferentes plataformas de programación como github.

Para proyectos futuros se recomienda el uso de ROS 2, la versión posterior a ROS que trae soluciones y optimizaciones a diferentes problemas que pudiera tener su predecesor. El funcionamiento (tanto de ROS como de ROS 2) se basa en el uso de nodos. A modo de resumen, existen dos tipos fundamentales de nodos, Publisher y Suscriber. Los nodos tipo Publisher son los encargados de, como su nombre indica, publicar diferentes tipos de mensaje. Dentro de ROS existen tipos de mensaje estandarizado, uno de ellos, por ejemplo, es el tipo LaserScan, que contiene información relativa a un sensor láser como puede ser el LiDAR 2D empleado en este proyecto. Dichos mensajes son publicados en un tipo de estructura llamada topic.

Por otra parte, una vez el mensaje ha sido publicado en un topic particular, el nodo tipo Suscriber se encarga de suscribirse a ese topic y poder así consultar la información que en el mismo ha sido publicado. La combinación de diferentes tipos de nodos, mensajes y topics permiten crear proyectos de gran complejidad. Además, al tratarse de una aplicación de software libre, la mayor parte de problemas o necesidades que puedan surgir en el desarrollo de un proyecto, están ya solucionadas por otros usuarios.

Para el caso específico de este proyecto, se han encontrado diversas fuentes de código que permiten solucionar diferentes partes del mismo. Para la obtención de medidas del LiDAR, es posible emplear el código proporcionado en github por el usuario Slamtec. El repositorio empleado es sllidar_ros2 y permite la obtención de un mensaje de tipo LaserScan, con información relativa a las medidas del LiDAR. En pruebas realizadas durante el transcurso del proyecto, se observó que la calidad y cantidad de las medidas obtenidas por este proceso es superior a las obtenidas mediante el script de Python desarrollado para el mismo.

Una vez están disponibles las medidas, es necesario determinar el objetivo a resolver. Existen en github repositorios que permiten realizar seguimientos de pared, control de velocidad o mapeado entre otras funciones. Una aplicación interesante a desarrollar en proyectos futuros podría ser el SLAM (Simultaneous Localization and Mapping). El trabajo desarrollado por el usuario SteveMacenski y publicado en github en el repositorio de nombre slam_toolbox, es un primer paso a seguir para lograr esta funcionalidad. Mediante este código, es posible realizar un mapeado estático del entorno para tenerlo como referencia. Posteriormente, y habiendo guardado ese mapa, se puede proceder a poner en marcha el vehículo para, comparando mediante traslaciones y rotaciones el entorno con el mapa original, determinar la posición del vehículo en el entorno. Además de esta aplicación, existen diversidad de ellas, las cuales pueden ser objeto de estudio de un trabajo futuro en el que se recojan diversas formas de mapeado, así como de navegación autónoma.

Capítulo 8. BIBLIOGRAFÍA

[1] MATLAB, «What Is Autonomous Navigation? | Autonomous Navigation, Part 1,» Youtube, 2020.

[2] A. S. a. W. J. D. Miller, «Situation awareness with different levels of automation» *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 688-693, San Diego, 5-8 Octubre 2014.

[3] D. Rosen y P. Ballou, «Development of the Phantom III remotely operated vehicle» de *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings*, Providence, Septiembre 2000.

[4] R. Marchand, «The Designers Go to the Fair II: Norman Bel Geddes, The General Motors "Futurama," and the Visit to the Factory Transformed,» *MIT Press*, vol. 8, pp. 22-40, 1992.

[5] C.Thorpe, M. H. Hebert, T. Kanade y S. A. Shafer, «ision and navigation for the Carnegie-Mellon Navlab,,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, nº 3, pp. 362-373, 1998.

[6] M. D. e. al, «Autonomous cross-country navigation with the ALV» de *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, 24-29 Abril 1988.

[7] M. A. Turk, D. G. Morgenthaler, K. D. Gremban y M. Marra, «VITS-a vision system for autonomous land vehicle navigation,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, nº 3, pp. 342-361, 1988.

[8] R. B. e. al, «The DARPA grand challenge - development of an autonomous vehicle,» de *IEEE Intelligent Vehicles Symposium, 2004*, 2004.

[9] J. Z. Varghese, «Overview of Autonomous Vehicle Sensors and Systems,» de *Proceedings of the 2015 International Conference on Operations Excellence and Service Engineering*, Florida, USA, 2015.

[10] K. Honkavaara, P. Piot, S. Schreiber y D. Sertore, «Bunch length measurements at the TESLA Test Facility using a streak camer,» de *PACS2001. Proceedings of the 2001 Particle Accelerator Conference*, 2001.

[11] S. M. Patole, M. Torlak, D. Wang y M. Ali, «Automotive radars: A review of signal processing techniques,» *IEEE Signal Processing Magazine*, vol. 34, nº 2, pp. 22-35, 2017.

[12] M. E. Warren, «Automotive LIDAR Technology,» de *2019 Symposium on VLSI Circuits*, 2019.

[13] J. Camós, «motorpasion,» 19 Octubre 2016. [En línea]. Available: https://www.motorpasion.com/volkswagen/marcara-este-volkswagen-i-d-concept-el-inicio-de-una-nueva-era. [Último acceso: 13 Agosto 2022].

[14] N. Ahmad, R. A. R. Ghazilla y N. M. Khairi, «Reviews on Various Inertial Measurement Unit,» *International Journal of Signal Processing Systems*, vol. 1, nº 2, 2013.

[15] A.K.Bourke y G.M.Lyons, «A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor,» *Medical Engineering & Physics*, vol. 30, nº 1, pp. 84-90, 2008.

[16] «Lba industrial,» 7 Marzo 2020. [En línea]. Available: http://www.lbaindustrial.com.mx/que-es-un-encoder/. [Último acceso: 15 Agosto 2022].

[17] J. Wang, C. Rizos y M. e. a. Stewart, «GPS and GLONASS Integration: Modeling and Ambiguity Resolution Issues,» *GPS Solutions*, pp. 55-61, 2001.

[18] G. Grisetti, R. Kümmerle, C. Stachniss y W. Burgard, «A Tutorial on Graph-Based SLAM,» *IEEE Intelligent Transportation Systems Magazine*, vol. 2, nº 4, pp. 31-43, 2010.

[19] MATLAB, «Youtube,» 8 Julio 2020. [En línea]. Available: https://www.youtube.com/watch?v=saVZtgPyyJQ&ab_channel=MATLAB. [Último acceso: 12 Agosto 2022].

[20] Q. Li, R. Li, K. Ji y W. Dai, «Kalman Filter and Its Application,» de *015 8th International Conference on Intelligent Networks and Intelligent Systems*, Tianjin , 1-3 Noviembre 2015.

[21] B. Andrej, K. Martin, B. Peter y F. e. a. Martin, «Path Planning with Modified a Star Algorithm for a Mobile Robot,» *Procedia Engineering*, vol. 96, pp. 56-69, 2014.

[22] «SuperRobotica,» 26 Agosto 2022. [En línea]. Available: http://www.superrobotica.com/S330100.htm. [Último acceso: 27 Agosto 2022].

[23] R. Velasco, «HardZone,» 24 Agosto 2021. [En línea]. Available: https://hardzone.es/reviews/perifericos/analisis-raspberry-pi-3-modelo-b/. [Último acceso: 21 Agosto 2022].

[24] «Slamtec,» 15 Mayo 2017. [En línea]. Available: https://cdn.sparkfun.com/assets/e/a/f/9/8/LD208_SLAMTEC_rplidar_datasheet_A2M8_v1.0_en.pdf. [Último acceso: 12 Agosto 2022].

[25] C. Moler y J. Little, «A history of MATLAB,» *ACM Journals*, vol. 4, nº 81, pp. 1-67, 2020.

[26] J. Backus, «The history of FORTRAN I, II, and III,» *ACM*, vol. 13, nº 8, pp. 165-180, 1978.

[27] M. Nosrati, «Python: An appropriate language for real world programming» de *World Applied Programming*, vol. 1, nº 2 2011.

[28] R. M. Stallman, «Free Software, Free Society», Boston: Free Software Foundation, 2010.

ANEXO I: PROTOCOLO TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) designa un protocolo de transmisión de información dentro de una red. A modo de resumen, se trata de una serie de normas básicas que han de seguir los elementos a enviar o recibir dentro de una red, en la comunicación de los diferentes equipos de la misma.

Este protocolo fue ideado por Robert Kahn y Vinton Cerf en los años 70, implantándose por primera vez en ARPANET, (Advanced Research Projects Agency Network). Para facilitar su desarrollo, el proyecto está dividido en diferentes capas, cada una de las cuales se encarga de realizar una labor determinada y de dividir la información en paquetes. Se presentan a continuación y, de manera breve las capas involucradas.

- Capa de acceso a la red: Se trata del medio físico que permite la comunicación de los ordenadores conectados a una red. Puede tratarse de medios como una red inalámbrica así cómo cable Ethernet entre otros. Se incluye en esta capa, además, aquellos medios técnicos como código, que permiten la conversión de los datos a un formato que pueda ser transmitido.

- Capa de internet: Es la capa encargada del envío de datos de forma segura y rápida. El envío de datos puede verse ralentizado debido a diversos factores como por ejemplo la existencia de gran cantidad de tráfico en la red.

- Capa de transporte: La capa de transporte tiene como objetivo asegurar la correcta emisión y recepción de los paquetes. Dicha labor se realiza mediante paquetes que contienen cierta información que puede ser comprobada por el receptor. La información atiende a los criterios del protocolo establecido y, permite entre otras cosas, determinar si los paquetes se han recibido correctamente o si se ha recibido la totalidad de la información enviada.

- Capa de aplicaciones: Se trata de las interfaces con las que el usuario interactúa y que permiten al mismo acceder a la información recibida o enviar información. Un ejemplo es el correo electrónico.

Para el envío de información siguiendo este protocolo, resulta importante el concepto de IP. Cada elemento dentro de una red tiene asociada una IP dentro de la misma. A la hora de enviar información, este protocolo, se basa además en las IP's de emisor y receptor para la correcta gestión del flujo de información.

ANEXO II: CÓDIGO PYTHON MEDIDAS LIDAR

```
from rplidar import RPLidar
import time
import numpy as np
import struct
import socket

PORT = 7000
SERVER = '192.168.0.123'
ADDR = (SERVER,PORT)


#Funcion medidas

def get_measurements(lidar,client):
    try:
        print('BEGIN SCAN...')
        measurements = np.full(40,-1.0) #angle-measurement (20 of each)
        i = 0
        for item in lidar.iter_measurments(max_buf_meas=1000):

            if i==0: #Inicio medidas
                start=time.time()
            if item[3]!=0: #Si la medida es válida la añadimos, si no, -1.0
                measurements[i] = item[2]
                measurements[i+1] = item[3]/1000  #Medida en metros

            i+=2

            if time.time()-start>=0.01 or i>=39:
                if time.time()-start<0.01:
                    time.sleep(0.01-(time.time()-start)) #wait till 10 ms
                #print('time: ',time.time()-start)

                a=bytearray(struct.pack('>ffffffffffffffffffffffffffffffffffffffff',
                measurements[0],measurements[1],measurements[2],measurements[3],
                measurements[4],measurements[5],measurements[6],
                measurements[7],measurements[8],measurements[9],
                measurements[10],measurements[11],measurements[12],
                measurements[13],measurements[14],measurements[15],
                measurements[16],measurements[17],measurements[18],
                measurements[19],measurements[20],measurements[21],
                measurements[22],measurements[23],measurements[24],
                measurements[25],measurements[26],measurements[27],
                measurements[28],measurements[29],measurements[30],
                measurements[31],measurements[32],measurements[33],
                measurements[34],measurements[35],measurements[36],
                measurements[37],measurements[38],measurements[39])).hex()

                msg = bytes.fromhex(a)

                client.send(msg)
                measurements = np.full(40,-1.0)
                i=0
```

```
    except:
        #Error
        return


### CODIGO ###


#Desconectamos LIDAR en caso de que esté encendido
lidar = RPLidar('/dev/ttyUSB0')

lidar.stop()
lidar.stop_motor()
lidar.disconnect()


#Conectar a socket
client = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connected = False

while connected==False:
    try:
        time.sleep(3000)
        b = client.connect(ADDR)
        if b==None:
            connected = True

    except TimeoutError:
        print('Not connected yet...')
    except ConnectionRefusedError:
        print('Not connected yet...')


#Mandar un 1 para activar
recibido_1 = False

while recibido_1 == False:
    aceptar = client.recv(8)
    if int.from_bytes(aceptar,byteorder='big')==1:
        recibido_1=True


time.sleep(0.5)

#Encender LiDAR
lidar = RPLidar('/dev/ttyUSB0')

#Información del LiDAR
info = lidar.get_info()
print(info)

#Estado del LiDAR
health = lidar.get_health()
print(health)
```

```
#Bucle funcionamiento
while True:

    get_measurements(lidar,client)
    lidar.stop()
    lidar.stop_motor()
    lidar.disconnect()
    time.sleep(0.01)
    lidar = RPLidar('/dev/ttyUSB0')


lidar.stop()
lidar.stop_motor()
lidar.disconnect()
```

ANEXO III: OBJETIVOS DE DESARROLLO SOSTENIBLE

En este apartado se procede a describir la relación de este proyecto con los Objetivos de Desarrollo Sostenible (ODS). Se trata de un conjunto de 17 objetivos, establecidos en 2015 por la Asamblea General de la ONU. Dichos objetivos, entre los que se encuentra el fin de la pobreza, hambre cero o educación de calidad, pretenden conseguir un futuro sostenible con condiciones de vida mejores para toda la población.

El proyecto presentado se enmarca en el ámbito de la navegación autónoma. El desarrollo de futuros proyectos con la tecnología LiDAR dará lugar a mejores y nuevas técnicas de navegación autónoma, optimizando procesos y, pudiendo llegar a utilizar en el futuro sensores que aporten una mayor cantidad de información relativa al entorno, como un LiDAR 3D así como su uso combinado con otros sensores como cámaras.

El lograr en un futuro el desarrollo de vehículos autónomos en fase 5 en la escala de automatización de la navegación hará que se reduzca el número de accidentes al ser la mayoría debidos a acciones del tipo humano. Se logra así cumplir con objetivos marcados como salud y bienestar ya que además de lo mencionado, permitiría un aumento del confort y la calidad de vida de la población.

Las técnicas de navegación perfeccionadas permitirán además elegir la ruta más óptima al destino deseado, así como coordinarse con los demás vehículos para tratar de evitar atascos, reduciendo así el número de emisiones. Es aquí donde se enmarcan aquellos objetivos relacionados con el medio ambiente y la producción y consumo sostenible de los recursos.