ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

Especialidad Electrónica

CONTROL DE UN VEHÍCULO AUTÓNOMO EN UN ENTORNO LIMITADO

Autor: Julio Labora Gómez

Director: Juan Luis Zamora Macho

Madrid

Agosto 2018

This dissertation was typeset with LaTeX and compiled in TeXmaker. The font families used are Bitstream Charter, Utopia, Bookman, and Computer Modern. Unless otherwise noted, all figures were created by the author using Microsoft PowerPoint® and MATLAB®.

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. JULIO LABORA GÓMEZ

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: CONTROL DE UN VEHÍCULO AUTÓNOMO EN UN ENTORNO LIMITADO, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

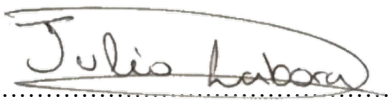
6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 30 de Agosto de 2018

ACEPTA



Fdo……

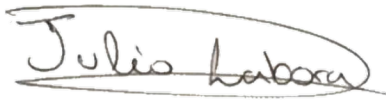
Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

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

Control de un vehículo autónomo en un entorno limitado

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico ……2017/2018…. 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.: Julio Labora Gómez Fecha: 30/ 8/ 2018



Autorizada la entrega del proyecto

EL DIRECTOR DE PROYECTO

Fdo.: Juan Luis Zamora Macho Fecha: 30/ 8/ 2018

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

Especialidad Electrónica

CONTROL DE UN VEHÍCULO AUTÓNOMO EN UN ENTORNO LIMITADO

Autor: Julio Labora Gómez

Director: Juan Luis Zamora Macho

Madrid

Agosto 2018

This dissertation was typeset with LaTeX and compiled in TeXmaker. The font families used are Bitstream Charter, Utopia, Bookman, and Computer Modern. Unless otherwise noted, all figures were created by the author using Microsoft PowerPoint® and MATLAB®.

CONTROL DE UN VEHÍCULO AUTÓNOMO EN UN ENTORNO LIMITADO

Autor: Julio Labora Gómez
Director: Juan Luis Zamora Macho
Entidad colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN

1. Introducción

El crecimiento del uso de la tecnología en todos los ámbitos durante los últimos años ha llevado a las grandes compañías a automatizar cada vez más procesos: desde complejos brazos robóticos que operan en las cadenas de montaje hasta máquinas de refrescos. Y la tendencia demuestra que durante los próximos años la tecnología desarrollará aun más funciones en nuestras vidas: con la reciente popularidad del Internet of Things y el *machine learning*, entre otros, no habrá que esperar mucho antes de vernos rodeados de máquinas capaces de realizar casi cualquier tarea. En esta línea, este proyecto pretende explorar dos campos que son susceptibles de ser automatizados en pocos años: los coches autónomos y los robots motorizados que operan en interiores, emulando la tendencia de grandes compañías, como son *Google*, con su popular vehículo autónomo *Google Car*; o *Amazon*, que emplea almacenes totalmente automatizados a través del uso de pequeños robots que transportan la mercancía. Por tanto, este proyecto tiene como objetivo el desarrollo de un coche autónomo capaz de orientarse en recintos cerrados y navegar por ellos.

2. Metodología

Así pues, para conseguir este objetivo los pasos a seguir son:

1. Elección de los elementos hardware necesarios para cumplir las tareas de navegación autónoma.
2. Diseño y elaboración de la estructura del vehículo.
3. Desarrollo del software de control de los sensores y actuadores del vehículo.
4. Desarrollo del software de interpretación que permite integrar todos los elementos.
5. Pruebas del vehículo en conjunto y optimización del software.

3. Descripción general del hardware

En la figura 3.1 se muestra una visión esquemática de los diferentes elementos integrados en el vehículo y las conexiones de comunicación (líneas moradas) y alimentación (líneas rojas). Estos son:

- una *Raspberry Pi 3B*, que cumple la función de ordenador de a bordo.
- un *Arduino Nano*, que actúa de interfaz de transmisión entre el LIDAR y la Raspberry.
- un LIDAR modelo *RPLIDAR A2M4*, que toma medidas del entorno del vehículo y constituye el núcleo del proyecto por su complejidad.
- una IMU modelo *MPU_9250*, que se comunica mediante SPI con la Raspberry y mide la aceleración y velocidad angular del coche en todo momento.
- dos drivers modelo *MD25*, que controlan los motores.

- 4 motores modelo *EMG30*, que mueven el vehículo e integran sendos encoders capaces de medir la velocidad del vehículo.
- una batería LiPo de tres celdas de alta descarga, que proporciona una tensión de 12 voltios y alimenta todos los elementos del vehículo directa o indirectamente.
- un conversor de tensión que convierte de 5 voltios a 3,3 voltios permitiendo así la comunicación por I2C entre la Raspberry y los drivers.
- un regulador de tensión que convierte los 12 voltios que proporciona la batería a 5 voltios.

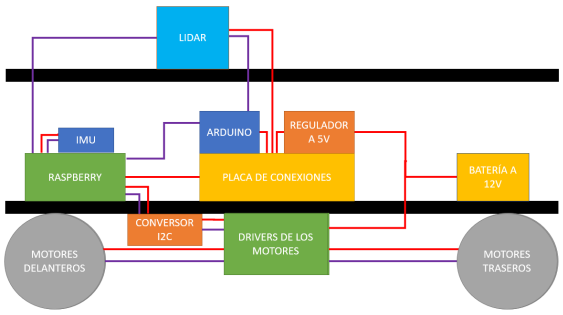


Figura 3.1. Esquema de la integración de los elementos hardware en el vehículo.

4. Descripción general del software

El software de control del vehículo tiene 3 partes fundamentales: el control, el filtro extendido de Kalman (EKF) y la máquina de estados global.

4.1. Control

El control consiste en dos grupos de PID's en cascada que funcionan en paralelo. En el nivel superior se tiene un PID de posición que calcula la referencia del módulo de la velocidad y del ángulo de giro, cada una de ellas para un nuevo PID de velocidad y ángulo respectivamente. El ángulo de giro es a su vez referencia de un control PID de velocidad de giro. Como resultado del PID de velocidad se obtiene el mando de tensión común de los motores, mientras que del control de velocidad de giro se obtiene el mando de tensión diferencial. Desacoplando estos mandos para convertirlos a tensiones del lado izquierdo y derecho, se aplican dichas tensiones a los motores. El vehículo es la planta y la sensorización es la que cierra los lazos por medio de un EKF.

4.2. EKF

El EKF empleado integra información de las medidas de los sensores arriba mencionados y mediante métodos estocásticos estima las variables de estado en todo instante. En este proyecto se ha utilizado un modelo cinemático de las ecuaciones, que consiste en recibir toda la información de la sensorización e ignorar las ecuaciones dinámica, pues se considera que en la determinación de las fuerzas que aparecen en estas existe mucha incertidumbre.

4.3. Máquina de estados

La máquina de estados es la que toma las decisiones de más alto nivel en el vehículo. Es la que determina la transición entre las diferentes estados y la encargada de detener el vehículo en caso de que exista algún problema. Los estados que implementa en este proyecto son solo 4:

1. Inicialización
2. Calibración
3. Operación Normal
4. Fallo

Las transiciones son en este caso secuenciales entre el primero, segundo y tercero condicionado con un contador, excepto en el caso de detectar

una baja tensión de la batería o un error en la comunicación con un sensor. En este caso se entra en el estado de fallo.

5. Integración del LIDAR en el vehículo

5.1. Comunicación con el LIDAR y obtención de medidas

La comunicación con el LIDAR ha requerido desarrollar un driver de este sensor para Simulink. Para ello ha sido necesario llevar a cabo un proceso de ingeniería inversa para conocer todos los detalles del protocolo de comunicación y así poder reproducirlo.

Tras adaptarse a las necesidades del protocolo, el siguiente paso ha sido convertir la información enviada por el LIDAR al ordenador de a bordo en pares ángulo-distancia que dan información real acerca del entorno. Esta información se envía utilizando un protocolo especial para optimizar la cantidad de información comunicada y maximizar la velocidad de medición, de forma las medidas retransmitidas dentro de un mismo mensaje se relacionan a través de diferencias angulares (figura 5.1).

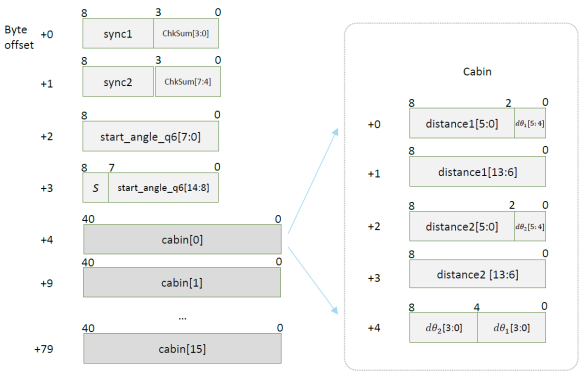


Figura 5.1. Protocolo de RPLIDAR A2 en modo Express Scan.

Tras implementar este driver, se ha conseguido obtener medidas en diferentes entornos, de forma que se puede mapear cualquier recinto cerrado con precisión. Un ejemplo aparece representado en la figura 5.2.

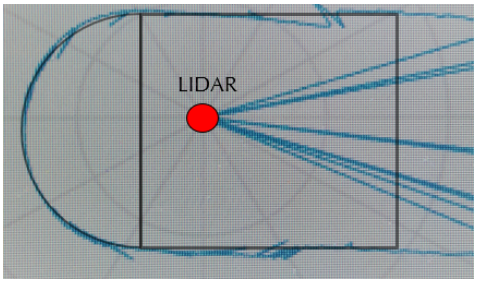


Figura 5.2. Comparación de un entorno con el mapa elaborado por el LIDAR.

5.2. Integración de las medidas en el software

Debido a la elevada velocidad de muestreo del LIDAR, su integración con el resto del software de interpretación requiere un preprocesado independiente de sus medidas, pues integrarlo directamente implica demasiada carga computacional. Este preprocesado consiste en calcular las variables del vehículo (posición y orientación en el mapa) utilizando solamente las medidas de este sensor a través de una regresión por mínimos cuadrados, e integrando directamente estos resultados en el EKF.

En una primera versión, se intentó utilizar las medidas del LIDAR considerando que el tiempo empleado por el sensor en dar una vuelta era suficientemente pequeño como para que el error asociado al movimiento del vehículo en ese tiempo fuera despreciable. Sin embargo, se comprobó que dicho error afectaba mucho a la estimación.

Para realizar dicha corrección se ha estimado el estado del coche en cada instante mientras el LIDAR toma las medidas y se han referido esas medidas a un único sistema de referencia, de forma que todas las medidas se asemejan a las que se hubieran tomado desde un punto concreto con el vehículo parado. Para referirlas a un punto

concreto de la trayectoria se ha utilizado la información del EKF para estimar las variaciones de posición y orientación entre esos dos puntos. Los resultados de este método sí que ofrecen una precisión suficientemente elevada.

Se evalúan gráficamente ambos métodos para el tiempo que tarda el LIDAR en dar una vuelta completa (144 milisegundos) en las figuras 5.3 y 5.4:

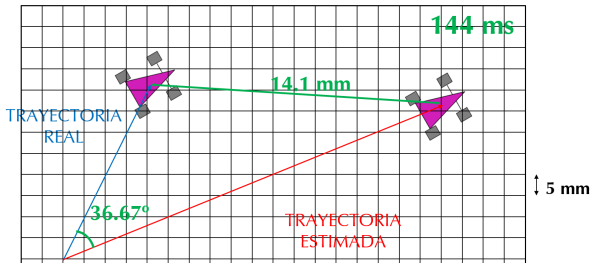


Figura 5.3. Estimación sin corrección del error.

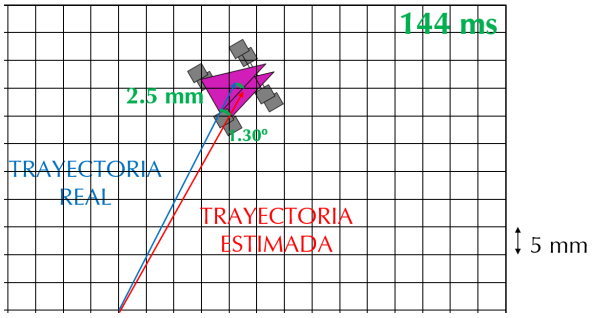


Figura 5.4. Estimación con corrección dinámica del error.

6. Conclusiones

La parte fundamental de este proyecto ha sido todo lo relacionado con el LIDAR. Primeramente, conseguir generar el PWM para que gire y enviar los comandos de la forma exacta que el LIDAR requiere para recibir la información de las medidas, para lo que ha sido necesario comunicar y coordinar el LIDAR tanto con la Raspberry como con un Arduino adicional debido a las dificultades encontradas. Una vez conseguido el control del sensor, ha sido necesario desarrollar un algoritmo especial para preprocesar sus medidas, pues integrarlas directamente en el EKF era demasiado costoso computacionalmente. Para ello ha sido necesario desarrollar las ecuaciones que relacionan la evolución de las medidas del LIDAR en el tiempo con el movimiento del vehículo y utilizar dichas ecuaciones para estimar la posición y orientación del vehículo solamente con las medidas del LIDAR empleando el método de mínimos cuadrados. Para mejorar la precisión de esta estimación se ha utilizado la información del EKF para corregir dinámicamente estas medidas, de forma que ambos procesos se retroalimentan mutuamente y se consigue así una mayor calidad en las estimaciones. El implementar este método tan creativo de coordinar ambas estimaciones ha constituido un auténtico reto.

Palabras clave: Navegación Autónoma · Vehículo autónomo · LIDAR · Filtro extendido de Kalman · Mapeado · Coche robótico · Raspberry Pi · MATLAB · Simulink

CONTROL OF AN AUTONOMOUS VEHICLE IN AN INDOOR ENVIRONMENT

Author: Julio Labora Gómez

Director: Juan Luis Zamora Macho

Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

1. Introduction

The recent technological developments had motivated the biggest companies to automate more and more processes: the use of robotic arms in production lines, the development of autonomous vehicles or even the use of robotic vacuum clearers. And the trend warns us that even more processes are due to be automated in the incoming years. The development of Machine Learning technics and the growing use of different devices interconnected with Internet of Things suggests that we will be shortly surrounded by smart devices capable of performing our daily tasks.

The purpose of this senior thesis is to do some research in two topics which are due to be automated in a few years: autonomous cars and moving indoor robots. Some examples of research in these topics are the popular *Google Car* or the completely automatic warehouses used by *Amazon* to store their products.

Therefore, the objective of this thesis is to design and build a vehicle capable of orienting and navigating autonomously in indoor environments.

2. Methodology

The tasks necessary to fulfill the goals of the project are the following:

1. Choice of the devices necessary to satisfy the requirements of the autonomous navigation task.
2. Design and assembly of the vehicle structure.
3. Development of the software required to monitor and control all the sensors and actuators of the vehicle.
4. Development of the data processing necessary to get all the elements to work together.
5. Vehicle and control testing and optimization.

3. Hardware overview

The figure 3.1 shows an schematic of the different elements mounted on the vehicle. It also shows the communication interfaces (purple lines) and the power distribution (red lines). The elements mounted are the following:

- *Raspberry Pi 3B*, which is the on-board computer.
- *Arduino Nano*, which acts as a communication interface between the LIDAR and the Raspberry.
- LIDAR model *RPLIDAR A2M4*, which maps the environment and is the core of the thesis research.
- IMU model *MPU_9250*, which is connected via SPI with the Raspberry and measures the acceleration and angular speeds of the vehicle.
- 2 motor drivers model *MD25*, which control the motors and connect via I2C with the Raspberry.
- 4 motors model *EMG30*, which allow the vehicle to move and have 4 encoders which measure the vehicle speed.

- LiPo high discharge 3-cell battery, which power the rest of the elements with 12 volts.
- voltage regulator to enable the I2C communication between the Raspberry and the drives, which have different voltage levels.
- voltage regulator which converts the 12 volts from the battery to 5 volts required to power other elements of the vehicle.

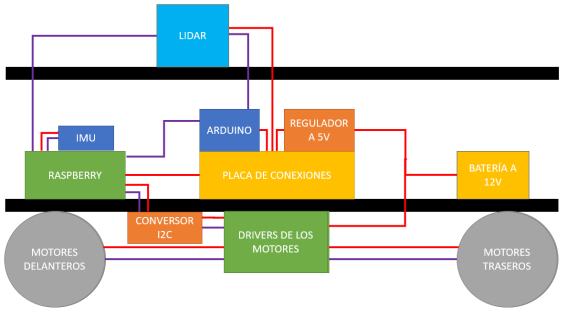


Figure 3.1. Vehicle devices interconnection schematics.

4. Software Overview

The software of the vehicle has 3 main parts: control algorithm, Extended Kalman Filter (EKF) and the state machine.

4.1. Control

The control algorithm consists of two cascade PID's that work in parallel. The higher level has a position PID that calculates the speed modul reference and the angle reference. The angle reference is then processesed by a PID which provides an angular rate reference for a angle rate PID control. Then the angle rate control and the speed control calculate the differential and common voltage for the motors. These voltages need to be decoupled to provide the voltage levels to the left and right motors.The vehicle constitutes the plant of the control and the sensors with the EKF close the control loop.

4.2. EKF

The integrated EKF uses the information provided by the sensors and, with stochastic calculations estimates the state variables which determin the movements of the car. In this thesis a kinematic approach has been used to model the equations, as the dynamical equations of the movements are considered to be inaccurate and result in too much error.

4.3. State Machine

The state machine is in charge of making the decisions which determin the actions of the vehicle in the highest level. It is the responsible of initializing the vehicle, calibrating the sensors, and in case of failure in any part of the system is the one which stops the vehicle. There are 4 states in this state machine:

1. Initialization
2. Calibration
3. Regular operation
4. Failure

In this state machine the state transitions are sequential between states 1 to 3. These transitions are conditioned by a timer, except the transitions regarding the failure state. In any of the other states, if the communication with a sensor fails or the battery voltage drops too much the car enters the failure state, where it stops and shuts down.

5. Integration of the LIDAR

5.1. Communication with the LIDAR and measurements obtention

The communications with the LIDAR have required the development of a driver of the sensor for Simulink. For that purpose, a process of

reverse engineering has been made to know the details of this protocol.

After developing the driver which enables the communication with the LIDAR it has been necessary to transform the data into distance and angle values, which is what Simulink needs to perform the corresponding calculations. The obtention of these values is non-trivial, as the protocol is designed to be very efficient but not simple. An overview of the messages and its relation with the values mentioned is shown in figure 5.1.

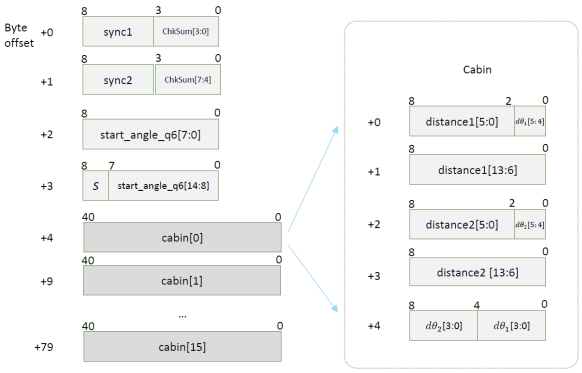


Figure 5.1. RPLIDAR A2 protocol in Express Scan mode.

Thanks to the implementation of this driver it has been possible to obtain the corresponding measurements, which are very accurate. An example is shown in figure 5.2.

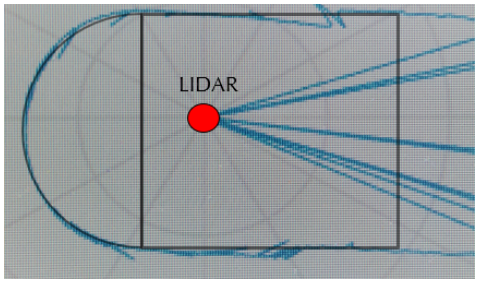


Figure 5.2. Evaluation of the measurement accuracy of the LIDAR.

5.2. Incorporation of the LIDAR measurements in the EKF

Due to the high sample rate of the LIDAR, its incorporation to the interpretation software needs a preprocessing in order to maintein the computational load at reasonable levels. This preprocessing consists of the independent calculation of the state variables (2D position and orientation) using only the LIDAR measurements and estimating the state variables using a least squares regresion.

At first, it was believed that the time taken by the LIDAR to generate a complete map (144 miliseconds) was negligible, but the results showed that it was not, so the final estimation was highly inaccurate.

In order to improve the estimation accuracy, the algorithm implemented refers all the measurments of the map to the coordinate system of one single point in the way of the car, as if all of them would have been made in that location by an static car. To calculate these changes of coordinate system the estimations of the state variables of the EKF have been used. The results obtained with this estimation are quite accurate.

Both cases are compared in figures 5.3 and 5.4.

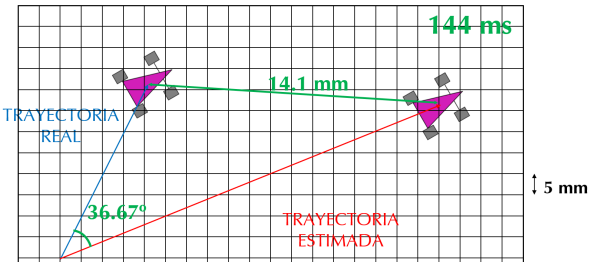


Figure 5.3. Estimation without error compensation.

6. Conclusions

The key part of this thesis has been the complete integration of the LIDAR in other systems with MATLAB and Simulink, not only in hard-

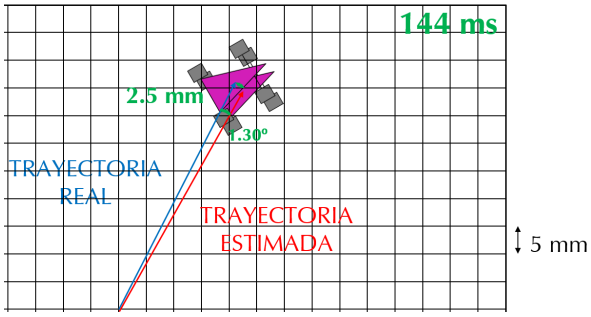


Figure 5.4. Estimation with dynamic error compensation.

ware but also in software. First of all it has been necessary to fulfill all the protocol and physical requierments to use the LIDAR, using an additional device to act as an interface between the Raspberry and it, the Arduino. Once this has been achieved, the integration in the software has also required some creativity in order to reduce the computational load of the calculations performed. The preprocessing algorithm used to estimate the state variables using a least squares estimation, and improving this estimation with a dynamic correction of the LIDAR measurements has been a real challenge.

Key words: Autonomous Navigation · Autonomous vehicle · LIDAR · Extended Kalman filter · Mapping · Car-like robot · Raspberry Pi · MATLAB · Simulink

A mis padres

Agradecimientos

En primer lugar, mi más sincera gratitud a mi director de proyecto, Juan Luis Zamora, por su ayuda y apoyo moral durante todo el proyecto y por rescatarme con sus planes Z cuando me quedaba sin ideas. Gracias de verdad.

En segundo lugar, quisiera agradecer a Jaime Boal su asistencia con la plantilla de LaTeX; y a Antonio y Jose, que siempre han estado dispuestos a ayudar con cualquier problema.

En tercer lugar, agradecer a todos los compañeros que me han ayudado durante los meses de proyecto: a Fer, por su inestimable asistencia con LaTeX y Raspberry; a Carlos, por ayudarme con el Arduino; a Víctor y Luis, por estar dispuestos a ayudarme a resolver unas ecuaciones intratables en la pizarra; a Inés y Javi, por ayudarme con, bueno, con todo tipo de cosas desde el puerto serie hasta la redacción de la memoria y mil más; y a todos ellos y los compañeros de laboratorio que no he nombrado por hacer que los meses de proyecto, aunque duros, hayan sido de los más divertidos que recuerdo.

Por último, agradecer a mi familia y a todos aquellos amigos que no he nombrado, la familia que se elige; todo el apoyo durante estos últimos cuatro años: desde que entré en la universidad y empecé un proyecto ilusionante; hasta el día de hoy, en que puedo decir que gracias a todos ellos soy Ingeniero.

Índice general

Índice de figuras

Índice de tablas

Siglas

CFRP Carbon Fiber Reinforced Polymer

EKF Extended Kalman Filter

GPIO General Purpose Input Output

I2C Inter-Integrated Circuit

IMU Inertial Measurement Unit

IoT Internet of Things

LIDAR Laser Imaging Detection And Ranging

LiPo Lithium Polymer

MIT Massachusetts Institute of Technology

NHTSA National Highway Traffic Safety Administration

PCB Printed Circuit Board

PID Proportional Integral Differential

PWM Pulse Width Modulation

RPM Revolutions Per Minute

SLAM Simultaneous Location and Mapping

SPI Serial Peripheral Interface

TCP Transmission Control Protocol

UART Universal Asynchronous Receiver Transmitter

UAV Unmanned Aerial Vehicle

UDP User Datagram Protocol

USB Universal Serial Bus

WiFi Wireless Fidelity

1 Introducción

1.1. Introducción

El objetivo de este proyecto es investigar sobre las aplicaciones y la implementación de pequeños vehículos autónomos que deben realizar una serie de tareas en entornos cerrados o limitados. En lo referente al estado del arte, la principal referencia serán los vehículos autónomos, aunque el diseño también estará basado en otros dispositivos como los UAV's.

En lo referente al vehículo autónomo la realidad es que para que este elemento de ciencia ficción esté tan cerca de hacerse realidad ha sido necesaria mucha investigación sobre cómo emular la conducción humana, principalmente su control del entorno y su capacidad de toma de decisiones.

Para emular estas características los vehículos autónomos deben tener dos partes fundamentales que los diferencian de los vehículos convencionales: en primer lugar una red de sensores capaz de monitorizar perfectamente todos los parámetros del entorno desde la velocidad y disposición de coches alrededor del propio vehículo hasta el estado de la carretera, todos ellos integrados y coordinados a través de un estimador que calculará todas estas variables en función de los datos que reciba de los sensores; y en segundo lugar un control del vehículo capaz de interpretar todos estos datos, tomar las decisiones asociadas a la conducción y enviar las instrucciones a los actuadores del vehículo para que efectúen los cambios necesarios.

Pero para hacer todo esto posible se ha investigado durante los últimos años toda esta tecnología gradualmente. Es decir, los coches han ido incorporando poco a poco ayudas al conductor como pueden ser asistencia de aparcamiento, control de limpiaparabrisas automático, reducción automática de velocidad en caso de obstáculo y muchas más. Sabiendo esto, surge inevitablemente una pregunta: ¿a partir de qué punto es considerado un coche autónomo? Para responder a esta pregunta la Autoridad de Tráfico de Estados Unidos (National Highway Traffic Safety Administration, NHTSA por sus siglas en inglés) hace una clasificación del 0 al 4 de los diferentes niveles de automatización que un vehículo puede tener [1].

[1] Información obtenida de la página web oficial de la agencia NHTSA. Página web: www.nhtsa.gov/staticfiles/rulemaking/pdf/Automated_Vehicles_Policy.pdf. Última consulta: 29 de enero de 2018

Nivel 0: No Automatización

Los vehículos que no poseen ningún control sobre los controles críticos del vehículo (freno, aceleración y dirección principalmente). En este grupo se incluirían aquellos vehículos que son capaces de avisar al conductor respecto de estas cuestiones críticas pero que no efectúan ningún cambio por sí solos.

Nivel 1: Automatización de Funciones Específicas

Este grupo incluiría a aquellos vehículos que son capaces de controlar algunos parámetros del vehículo bajo ciertas circunstancias y dar facilidades al conductor realizando algunas tareas de forma semiautomática. Ejemplos de esto serían el control de crucero o los controles de estabilidad.

Nivel 2: Automatización de Funciones de Forma Combinada

La principal diferencia entre este nivel y el anterior no está en las funciones que se controlan sino en una integración conjunta de los controles de las mismas, de forma que en las circunstancias propicias para el funcionamiento del control el conductor no tiene necesidad de controlar físicamente el vehículo (puede desentenderse del control de los pedales y el volante simultáneamente). Aun así, en este nivel el conductor sigue siendo responsable de controlar el entorno y debe ser capaz de reaccionar rápidamente en caso de necesidad.

Nivel 3: Conducción Autónoma Limitada

En este nivel el vehículo es capaz de controlar perfectamente todas las funciones del vehículo en condiciones favorables y es capaz de avisar con suficiente tiempo de antelación al conductor si es necesario que retome el control del vehículo en caso de existir una complicación que impide la conducción autónoma (cambio en el tiempo, entrada en una zona con carretera en mal estado serían algunos ejemplos). Todo esto se traduce en que deja de ser necesaria la constante monitorización del entorno por parte del conductor, pudiendo este dedicarse a otras actividades mientras conduce.

Nivel 4: Conducción Autónoma Total

El vehículo está preparado para llevar a cabo todas las funciones relacionadas con las conducción sin necesidad de intervención del conductor en ningún momento. Está pensado para requerir únicamente que se le proporcionen los datos asociados a la ruta. Esto incluye también la posibilidad de vehículos no tripulados.

En el caso del pequeño vehículo autónomo es evidente que se debe tomar como referencia el nivel 4, pues no va a estar tripulado. Este es precisamente el nivel que está a punto de alcanzarse y acerca del cual más se está investigando, mejorando no solo la tecnología de los sensores y su fiabilidad (la parte asociada al hardware, que está muy cerca de ser suficiente para los vehículos autónomos del futuro), sino especialmente el software. Esto es, aquellos algoritmos capaces de interpretar toda la información y traducirla en datos con los que el control del vehículo pueda trabajar. La investigación de este trabajo se dedicará, por tanto, principalmente al software.

1.1.1. Motivación

Para todo el sector emergente de dispositivos del hogar que poseen componentes electrónicos y de comunicación para hacer sus tareas de manera más eficiente (Internet of Things) existe la

necesidad de controlar aquellos dispositivos que necesitan moverse por la vivienda para realizar su trabajo, puesto que para que por ejemplo un aspirador funcione sin intervención humana es obvio que además de algún medio con el que comunicarle cuándo debe limpiar la casa es necesario un sistema que le permita efectuar esta tarea y además de forma eficiente. Por ello este tipo de controles serían muy útiles en lo que respecta a automatizar ciertas tareas del hogar y con algunas modificaciones sería aplicable a grandes almacenes o fábricas industriales donde automatizar el transporte de materiales o productos reduciría no solamente los costes sino también los tiempos.

Por estas razones este programa podría ser parte imprescindible de algunos productos innovadores que aumentarían la eficiencia tanto en el hogar como en la industria. Es por todo ello que se ha decidido desarrollar este proyecto.

1.1.2. Objetivos

El objetivo del proyecto es desarrollar el control para que un pequeño vehículo autónomo sea capaz de moverse y operar en un entorno limitado.

1.1.2.1. Diseño del Vehículo e Implantación Física de los Sensores

Se perseguirá un diseño que permita que los cambios que hubiera que realizar a lo largo del proceso sean sencillos y por otra parte que la forma del coche sea relativamente apropiada para su movimiento en entornos reducidos.

1.1.2.2. Diseño del EKF

Se tratará de optimizar la información que el filtro proporciona a la unidad de procesamiento del vehículo para que el control del entorno sea el mejor posible.

1.1.2.3. Desarrollo del algoritmo de control

Se implementará uno de los algoritmos de control descritos anteriormente, y se modificará para que la precisión del control en conjunto con el EKF permitan al vehículo operar con normalidad.

1.1.3. Metodología de Trabajo

Tareas

Para desarrollar el proyecto se definen las tareas que siguen:

1. Definición de objetivos y entrega del anexo B.
2. Desarrollo del software de control del LIDAR.
3. Desarrollo del software de control de los demás sensores.
4. Desarrollo del software de control de los actuadores.
5. Diseño y montaje del vehículo.
6. Diseño y optimización del filtro extendido de Kalman.
7. Desarrollo e implantación de un control para el vehículo.

8. Ensayo del conjunto control-EKF y modificación del tipo de control en caso de ser necesario.
9. Redacción de la memoria del proyecto.

1.1.4. Recursos a Emplear

- Ordenador personal con MATLAB 2018a y Simulink, en el que se han instalado además las librerías de Raspberry de Simulink.
- Múltiples herramientas empleadas habitualmente en este tipo de proyectos, como soldadores, taladros, polímetros, osciloscopio, etc.
- Estructura física del vehículo.
- El conjunto de sensores y actuadores del vehículo, que en su versión final consiste en: 4 motores, dos drivers para controlarlos, un LIDAR, una IMU, un ordenador de a bordo, una batería y varios conversores de diferentes niveles de tensión.

1.1.5. Estructura de la memoria

Este documento se divide en 5 capítulos: un primer capítulo de introducción (1), un capítulo donde se analiza el estado del arte (2), un capítulo donde se explica el hardware del proyecto (3), un capítulo en el que se explica el software (4), un capítulo de resultados (5), y un capítulo donde se hace una conclusión final del proyecto y se desarrollan algunas ideas acerca de posibles futuros desarrollos del proyecto (6).

2

Estado del Arte

En la actualidad la industria del vehículo autónomo se encuentra en un punto intermedio entre los niveles 3 y 4 definidos por la NHTSA. Esto es en realidad a nivel usuario, pues las compañías tecnológicas que trabajan en el proyecto de un vehículo autónomo ya durante los últimos meses han exhibido prototipos que son, en principio, plenamente autónomos. Como es lógico, para el propósito de esta investigación serán esos modelos los que se tomen como referencia, junto a algunos pequeño vehículos autónomos que ya operan en el mercado: desde robots-aspiradora capaces de moverse por una vivienda haciendo recorridos eficientes para minimizar tiempo y energía, hasta los vehículos que la multinacional Amazon emplea para gestionar algunos de sus almacenes, o incluso aquellos dispositivos con aplicaciones militares.

Para evaluar el estado del arte se analizarán por separado el hardware, en el que se incluye la estructura del vehículo; y el software, en el que se analiza el EKF en el campo de la robótica en general y se analizan distintos tipos de control para el vehículo. Finalmente, también se explicarán en detalle algunos vehículos similares al estudiado en este proyecto.

2.1. Hardware

2.1.1. Estructura del vehículo

En lo referente a la estructura del vehículo, existe gran variedad de opciones tanto en los materiales, la cinemática, la dinámica, etc. Por ello se explicarán algunas posibilidades y sus ventajas e inconvenientes de cara a los robots en general y los vehículos robóticos en particular, pero sin hacer un análisis exhaustivo.

2.1.1.1. Materiales

En la elección de materiales casi cualquier decisión es válida siempre y cuando la rigidez y aguante del material no sean un factor crítico. Los más empleados en este tipo de aplicaciones son: plástico reforzado con fibra de vidrio y fibra de carbono y CFRP's. Sin embargo, es esta última la que se ha impuesto durante los últimos años por sus magníficas propiedades.

Figura 2.1. Brazo robótico Jaco de Kinova, construido en fibra de carbono. Fuente: http://www.adaptado.es/brazo-robotico-jaco-de-kinova/

2.1.1.1.1. Fibra de carbono

Sus principales ventajas mecánicas son su elevado ratio resistencia-peso y su rigidez. La fibra de carbono y sus derivados es lo más empleado en los casos donde la dinámica es crucial. Este es el caso de muchos brazos robóticos 2.1, que deben ejecutar tareas precisas sin que su masa sea excesivamente alta, pues esto supondría un consumo energético más elevado. Sin embargo, existe aquí un *trade-off* entre la masa y la frecuencia fundamental, pues esta última reduce la precisión del robot [1]. También es tremendamente beneficioso el uso de estos materiales en los casos en los que el peso es realmente un factor crítico, como cuando la interacción de las dinámicas del robot con el entorno es muy sensible a este parámetro, como es el caso de un robot capaz de saltar sobre la superficie del agua [2]. Ejemplo de ello es el caso de algunos robots submarinos y anfibios, en los cuales un peso elevado limitaría o anularía su operatividad en el medio acuático [3]. Aunque este material presenta numerosas ventajas, su elevado precio puede ser un inconveniente.

2.1.1.1.2. Fibra de vidrio reforzada

Sus principales características son su relación entre resistencia y peso moderadamente buena, su ligereza y su baja tecnología de fabricación. Además, actúa como aislante y es notablemente más barata que la fibra de carbono. Por ello, su uso en la industria está muy extendido, desde la aeronáutica a la medicina, pasando por la electrónica, donde su uso en la fabricación de PCB's es muy conocido.

2.1.1.2. Cinemática

La relevancia de la cinemática en los coches-robot es que determina la capacidad del vehículo de seguir una trayectoria con precisión. En este ámbito destacan dos estructuras que permiten diferentes posibilidades de giro a los vehículos: sistema de giro de Ackermann y el sistema de giro del doble pivote. Aunque ambas opciones son válidas, la tendencia sugiere que el sistema de giro de Ackermann es capaz de seguir con más precisión las referencias de posición, si

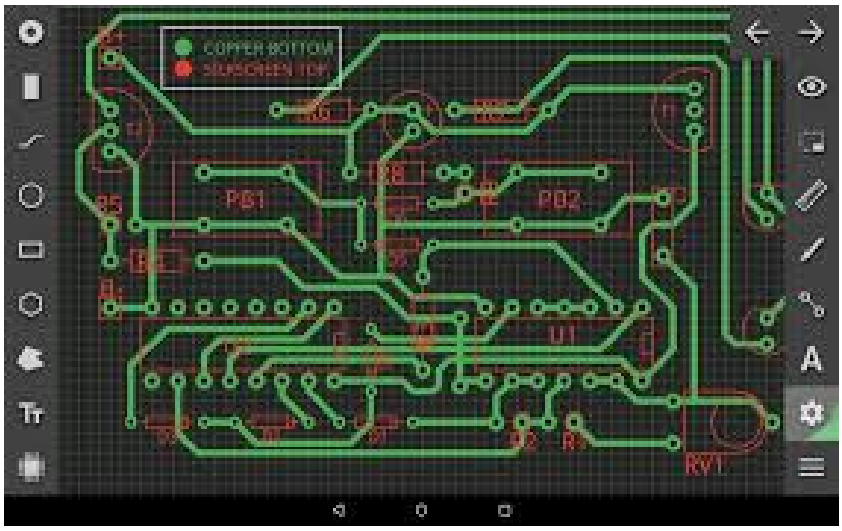


Figura 2.2. Ejemplo de PCB. Fuente: APKPure.com

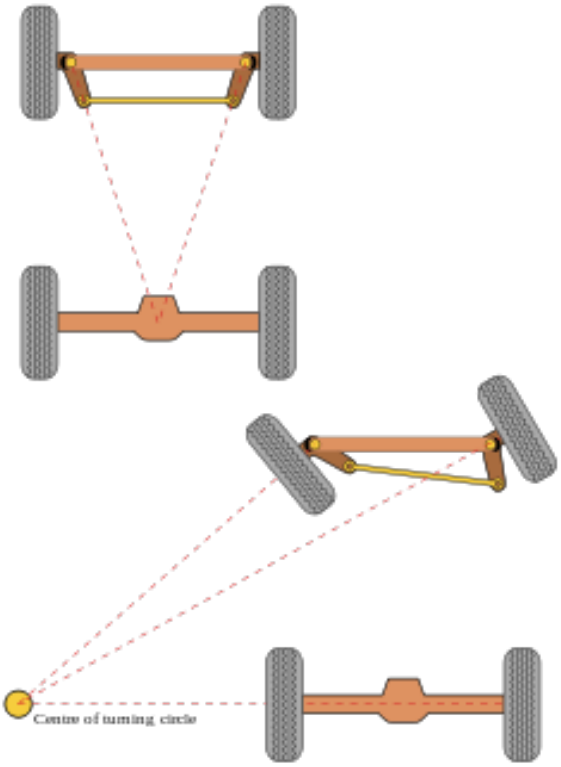


Figura 2.3. Geometría de giro de Ackermann. Fuente: es.wikipedia.org/wiki/Geometría_de_Ackermann

bien el modelado de la estructura complica los cálculos de control, pues se introducen muchas no-linealidades [4, 5].

2.1.2. Actuadores

Los actuadores en el ámbito de este proyecto se limitan a los motores, y entre ellos los más empleados con los motores de corriente continua. Existen muchos modelos en el mercado que ofrecen diferentes prestaciones tales como variados niveles de tensión de operación, distinto tamaño y velocidad de giro, entre otros. La inmensa mayoría operan entre 12 y 24 voltios, aunque también son relativamente frecuentes los modelos que operan a menos tensión. Existe

también variedad en cuanto a la presencia o no de escobillas. Aquellos que emplean escobillas para la conmutación de la corriente de las bobinas del motor suelen ser menos duraderos pero más baratos. Adicionalmente, poseen una buena relación entre potencia y tamaño. Debido a su menor precio y a que su menor durabilidad no suele ser crítica en el ámbito de la robótica, y especialmente a que son apropiados para aplicaciones donde el tamaño sí es un factor relevante; son los más utilizados en la robótica y la electrónica. La alternativa son los motores sin escobillas (comúnmente llamados *brushless*), que requieren de un controlador del motor para realizar la conmutación de las bobinas electrónicamente [6, 7].

2.1.3. Sensores

2.1.3.1. Acelerómetros y giroscopios

Estos dos tipos de sensores se emplean en los vehículos de casi todo tipo y no son para nada innovadores, pero son necesarios para conocer la posición y orientación del coche, además de su velocidad y aceleración. En vehículos de gran tamaño que habitualmente tienen muchos grados de libertad se suelen situar distintos acelerómetros en múltiples partes del vehículo para tener la mayor cantidad de información posible sobre la aceleración de todo el vehículo. En el caso de la electrónica es más común la presencia de un número reducido de estos sensores [8, 9].

2.1.3.2. Magnetómetros

Estos sensores reaccionan a la presencia de un campo magnético, ya sea artificial como cuando se acerca un imán (forma de sensorización muy empleada hoy en día en dispositivos electrónicos); o natural, como el campo magnético terrestre originado por la magnetosfera. En el ámbito de la electrónica en espacios abiertos son muy utilizados los magnetómetros con el fin de conocer la orientación exacta del vehículo. Para ello se miden las variaciones del campo magnético medido y conocida una orientación inicial respecto de la Tierra, se calcula la variación respecto de dicha orientación. Desgraciadamente, estos sensores no funcionan bien en entornos cerrados debido a la elevada interferencia electromagnética que producen los dispositivos eléctricos de los edificios en relación con los niveles de variación medidos por el sensor [8, 10].

2.1.3.3. Encoders

También llamados codificadores rotatorios, estos sensores son capaces de medir el giro de las ruedas del vehículo, típicamente mediante la utilización de sensores de efecto Hall, y conocido el radio de la rueda es posible determinar el valor de las variables asociadas al movimiento lineal (posición, velocidad y aceleración). Aunque los hay de más tipos, la esta explicación se limita a aquellos basados en variaciones de campo electromagnético, pues son los que más interés tienen en este proyecto. Su funcionamiento es el que sigue: cada encoder consiste en uno o más imanes situados en una rueda y dos sensores de efecto Hall que miden la variación del campo magnético producida por el giro de los imanes en la misma, de forma que cada vez que un imán pasa cerca de un sensor se crea una diferencia de tensión. Son necesarios dos sensores para poder determinar ”instantáneamente” el sentido de giro. Sin embargo, aun cuando la resolución del encoder es elevada suele ser necesario el uso de filtros especiales para que la medida sea lo más limpia posible [10].

Figura 2.4. LIDAR HDL-64E. Fuente: Velodyne

2.1.3.4. LIDAR

Este dispositivo emite ondas electromagnéticas en el espectro infrarrojo y midiendo el tiempo de retorno de dichas ondas es capaz de determinar la distancia a un punto. De esta forma, mediante la emisión de varios haces es posible emplear la información tanto con fines de orientación como de mapeo, y en última instancia para hacer *SLAM*, requiriéndose diferentes tipos de software de interpretación en cada caso [11,12]. Adicionalmente, los mapas pueden ser tanto bidimensionales como tridimensionales, dependiendo del LIDAR utilizado y de posibles servos para modificar su orientación en el espacio. Estos sensores son ampliamente utilizados en los prototipos de vehículos autónomos debido a su elevada precisión [13, 14], aunque su coste puede llegar a ser muy elevado en el caso de los modelos más precisos. Ejemplo de ello es el penúltimo modelo de Velodyne, el HDL-64, cuyo precio asciende a 75.000 $. Del último modelo no se ha divulgado el precio, pero cabe esperar que sea varios miles de dólares más alto como poco.

2.1.3.5. Sónares

El funcionamiento de estos sensores es similar al LIDAR, pero en este caso las ondas emitidas y posteriormente interpretadas son ondas mecánicas de ultrasonido [15]. La principal ventaja de este tipo de sensor es que es extremadamente preciso a bajas velocidades y con objetos cercanos, por lo que es ideal para el proyecto en cuestión. Este tipo de sensores son los empleados actualmente para el aparcamiento asistido.

2.1.3.6. Radares

Los radares emiten ondas electromagnéticas para identificar objetos de forma similar al sónar. La principal ventaja es que el alcance de los radares es mucho mayor, de ahí su conocida aplicación en barcos y aviones. Su principal inconveniente es que no solo la precisión no es la de un sónar a cortas distancias, sino que además las *imágenes* proporcionadas por un radar son bidimensionales y en ciertas aplicaciones la altura puede ser un factor clave que haría del radar una herramienta no demasiado útil por sí sola. [16, 17] Sin embargo, como se ve después en el apartado de software, es ventajoso tener sensores redundantes como pueden parecer un sónar

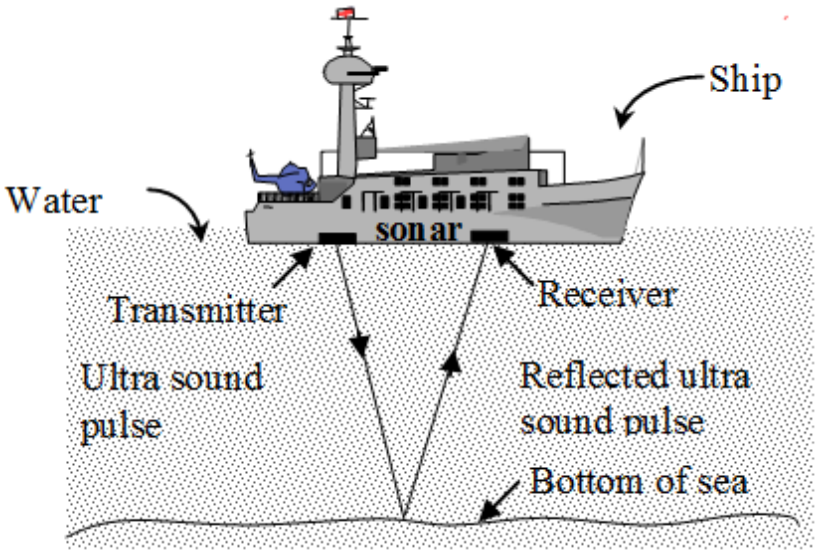


Figura 2.5. Funcionamiento de un sónar. Fuente: Knowledge Universe Online

y un radar, pues a la hora de implementar el software esto proporcionará mayor fiabilidad y precisión a los resultados finales.

2.1.3.7. Sensores de proximidad basados en infrarrojos

Este tipo de sensor funciona de manera similar a un LIDAR, pero utiliza un número reducido de rayos infrarrojos (típicamente uno solo) y tiene como único objetivo medir una distancia más que hacer mapeados o emplearse en la orientación. Para este proyecto en concreto sería un complemento ideal para el LIDAR, pues este tipo de dispositivo tiene una precisión muy elevada en cortas distancias (en el rango de los 15 ó 20 centímetros), que es justamente el rango donde el LIDAR utilizado no puede operar. Algunos usos típicos son la detección de pared en vehículos electrónicos o la asistencia de aparcamiento en vehículos motorizados. Además, son ampliamente utilizados en aplicaciones militares para medir distancias y marcar objetivos [18].

2.1.3.8. Sensores de flujo óptico

Estos sensores capturan imágenes del entorno a una velocidad elevada y son capaces de identificar singularidades en dichas imágenes, de forma que comparando una imagen con otra tomada anteriormente identifican el grupo de píxeles correspondiente a una singularidad, y viendo el desplazamiento de estos píxeles en las imágenes son capaces de calcular la velocidad de movimiento del vehículo.

2.1.4. Software

Como se ha dicho anteriormente la barrera tecnológica que separa al usuario del coche autónomo se debe principalmente al software. En lo referente a este actualmente se trabaja con gran cantidad de aproximaciones sobre como romper esta barrera. Las más disruptivas se centran en las recientes mejoras de los algoritmos de *machine learning* y las redes neuronales de aprendizaje [19, 20]. Existen además diferentes aproximaciones acerca de como tener en cuenta multitud de sensores y aunque únicamente se explica el EKF, hay algunas alternativas que cuestionan no el algoritmo matemático de este filtro (que indudablemente tiene multitud de alternativas válidas), sino el principio de interpretación de grupos de medidas desde un

punto de vista mucho más abstracto. Básicamente, se plantea la fase de interpretación de las medidas en que se deben fusionar las medidas de los sensores para llegar al estadio final de la toma de una decisión [11]. Sin embargo, de cara a una aplicación más sencilla como se persigue en este caso, se analizarán alternativas más sencillas de implementar, sin perder de vista aquellas opciones más complejas que podrían desarrollarse en versiones posteriores.

2.1.4.1. Filtro Extendido de Kalman

También conocido como EKF, el algoritmo de este filtro es capaz de integrar las medidas de todos los sensores y proporcionar al control central del vehículo una información mucho más fiable y precisa de lo que es capaz cualquiera de los sensores individualmente. Para ello emplea la redundancia de las medidas de diferentes sensores y un algoritmo especial que se basa en métodos estocásticos que es capaz de minimizar el error asociado a las medidas conjuntas de todos los sensores traduciéndolas a una medida obtenida ponderando todas ellas en función de su fiabilidad individual y conjunta. Este algoritmo es especialmente útil pues proporciona una característica poco intuitiva al vehículo y es la de ser más preciso en su actuación que cualquiera de los sensores de los que dispone individualmente. Esto introduce la idea de que no es necesario emplear sensores de gran precisión si es posible integrar más sensores menos precisos con un EKF correctamente diseñado [21, 22]. Existen variaciones de este filtro que proporcionan ventajas respecto al filtro extendido. Algunas de ella proporcionan mejores resultados para la navegación de vehículos [23] o consiguen reducir la carga computacional en la fusión de medidas de sensores, de forma que esta sea lineal a la cantidad de medidas en vez de crecer con un ratio polinómico de mayor grado como en el caso del filtro extendido [24].

2.1.4.2. Control del Vehículo

En lo referente al control del vehículo aunque en las aplicaciones relacionadas con los vehículos autónomos de carretera se emplean algoritmos muy sofisticados basados en algunas de las tecnologías en auge que se han mencionado anteriormente, la realidad es que cuando no es requerido ese nivel de fiabilidad y precisión los controles más empleados son el PID, el control por realimentación de estado [25, 26] y el control predictivo [27], en orden creciente de eficiencia y dificultad de implementación.

2.2. Vehículos similares

En este campo algunos de los vehículos más destacados están diseñados y probados en el MIT, tanto en el caso de navegación en interiores como de navegación en exteriores. En el caso de la navegación en interiores, hay una clase muy popular cuyo objetivo es utilizar unos componentes concretos para elaborar y programar un vehículo capaz de orientarse dentro de unas instalaciones específicas del MIT. Los sensores utilizados en este caso son:

Figura 2.6. Ejemplo de vehículo para navegación en interiores del MIT.

- Sensor de flujo óptico modelo *PX4FLOW*.
- LIDAR 2D modelo *Hokuyo UST-10LX*.
- Cámara embarcada modelo *Point Grey Firefly MV*.
- IMU modelo *Razor 9DOF*.

El objetivo de los estudiantes es diseñar ellos el algoritmo de control del vehículo que permita integrar estos sensores para permitir la navegación autónoma.

Por otra parte, es interesante también analizar la sensorización de un vehículo similar pero pensado para operar en exteriores.

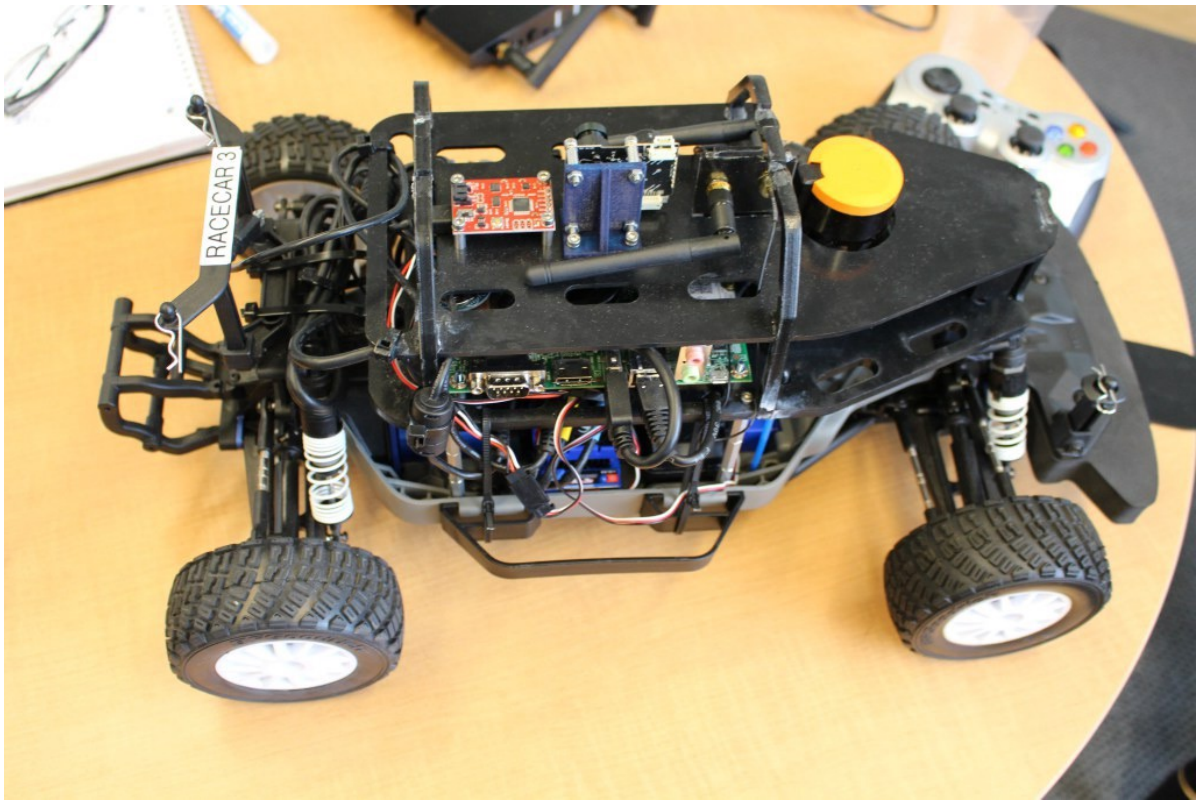


Figura 2.7. Ejemplo de vehículo para navegación en exteriores del MIT.

En este caso, los tipos de sensores utilizados son:

- Encoders en las ruedas.
- LIDAR 2D .
- Magnetómetros.
- Acelerómetros.
- Giróscopos.

Como se ve, los tipos de sensores utilizados son bastante similares si bien hay algunas diferencias relacionadas con el entorno de trabajo. Por supuesto, incluso para una cierta aplicación existen multitud de alternativas válidas al elegir los tipos de sensores en función de como se combinan entre ellos y las medidas que aportan.

3

Hardware

3.1. Dispositivos que incorpora el vehículo

El vehículo integra los siguientes dispositivos:

- una *Raspberry Pi 3B*, que cumple la función de ordenador de a bordo.
- un *Arduino Nano*, que actúa de interfaz de transmisión entre el LIDAR y la Raspberry.
- un LIDAR modelo *RPLIDAR A2M4*, que toma medidas del entorno del vehículo.
- una IMU modelo *MPU_9250*, que mide la aceleración y velocidad angular del coche en todo momento.
- dos drivers modelo *MD25*, que controlan los motores.
- 4 motores modelo *EMG30*, que mueven el vehículo e integran sendos encoders capaces de medir la velocidad del vehículo.
- una batería LiPo de tres celdas de alta descarga, que proporciona una tensión de 12 voltios y alimenta todos los elementos del vehículo directa o indirectamente.
- un conversor de tensión que convierte de 5 voltios a 3,3 voltios permitiendo así la comunicación por I2C entre la Raspberry y los drivers.
- un regulador de tensión que convierte los 12 voltios que proporciona la batería a 5 voltios para así poder alimentar el LIDAR, el Arduino y la Raspberry (que es la que alimenta la IMU).

Se muestra a continuación un esquema de como están conectados todos estos dispositivos. En dicho esquema las conexiones en rojo identifican la transmisión de tensión, mientras que las conexiones en morado identifican la transmisión de datos.

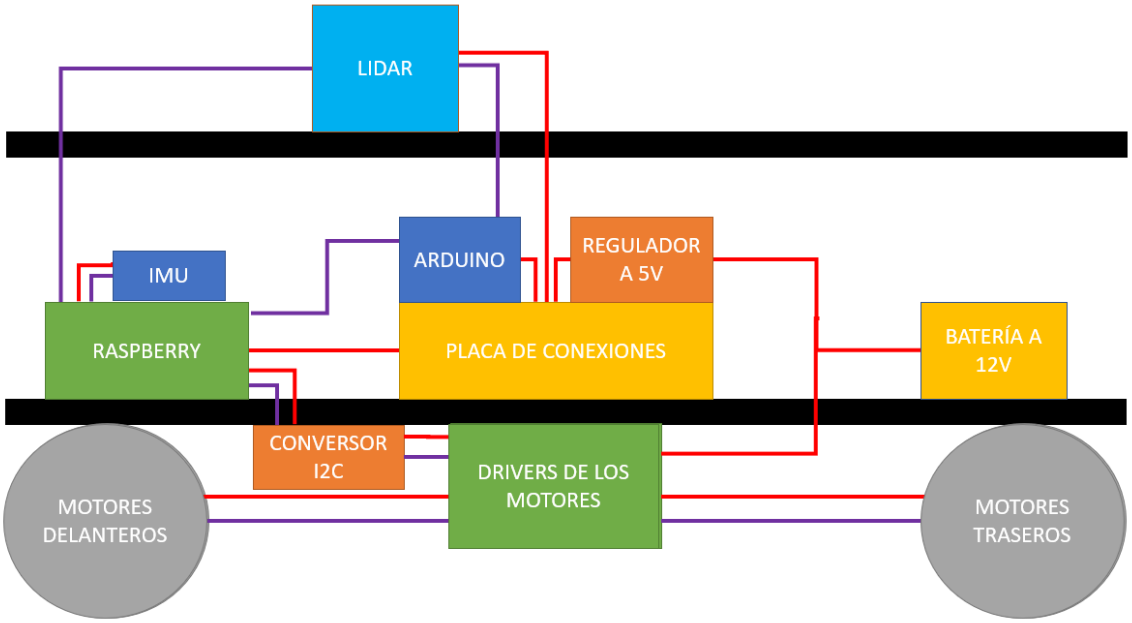


Figura 3.1. Esquema del conexionado del vehículo.

3.2. LIDAR

3.2.1. Función

Desde el inicio del proyecto se sabía que la integración del LIDAR era la parte más ambiciosa del proyecto y que su control supondría un auténtico reto. Por ello, el LIDAR es la parte más importante de este proyecto y por esa razón se explican su funcionamiento y la toma de decisiones acerca de como controlarlo de forma muy detallada. Su función dentro del vehículo es proporcionar la distancia a todos los elementos de alrededor de forma precisa y fiable, y aunque el vehículo posee otros sensores que están integrados en un EKF del que se hablará más adelante, el LIDAR es el que proporciona la mayor cantidad de información.

Figura 3.2. RPLIDAR A2. Fuente:ROSComponents

3.2.2. Justificación de su uso

El LIDAR es el sensor más potente de que dispone el coche y posiblemente el más potente del mercado que pueda ser empleado sin requerir un complejo software de interpretación (como es el caso de cámaras embarcadas en el vehículo). La creación de un control para este dispositivo ha sido desde el inicio el núcleo del proyecto, por lo que en realidad es el único elemento que no requiere de una justificación.

3.2.3. Conexionado y características

Debido a la complejidad del LIDAR y a algunas limitaciones tanto del software de MATLAB y Simulink como de la Raspberry Pi 3B, para poder controlar el LIDAR de forma robusta ha sido necesario utilizar un Arduino Nano como interfaz de transmisión. El LIDAR emplea únicamente 5 pines, que son: alimentación a 5 voltios, conexión a masa, recepción de la UART, transmisión de la UART y el pin de PWM. La generación del PWM y la transmisión son tareas encomendadas al Arduino, mientras que la recepción está conectada a la Raspberry. Esto se debe a que por ser la Raspberry la unidad central de procesamiento del vehículo, es esta la que procesará las medidas y las integrará en el EKF haciendo los cálculos pertinentes. La alimentación y la conexión a masa son proporcionadas por un regulador que convierte los 12 voltios de la batería en los 5 voltios que requiere para alimentarse.

3.2.4. Funcionamiento

El funcionamiento del LIDAR consiste en una comunicación por UART con una unidad inteligente capaz de enviar los comandos que requiere para funcionar, y de recibir las respuestas de este e interpretarlas. El modelo empleado tiene dos modos de funcionamiento que difieren en la frecuencia de muestreo del entorno. A pesar de su mayor complejidad en lo referente a la comunicación, en este proyecto se ha tomado la decisión ambiciosa de emplear el modo que configura el LIDAR para maximizar dicha frecuencia de muestreo, puesto que aunque la diferencia no es significativa para un proyecto con estos objetivos (las dos frecuencias de muestreo son de 2 y 4 kHz, por lo que ambas son relativamente altas), sí que podría suponer una ventaja en un proyecto posterior que pretenda que el vehículo desarrolle tareas más complejas. Para poder retransmitir las medidas escaneadas por la UART, el LIDAR debe encontrarse en un estado de rotación estable, para lo cual además de la alimentación precisa el PWM antes mencionado.

3.2.4.1. Opciones de control valoradas

Para controlar el LIDAR existía una decisión inicial que tomar: este sensor dispone de un dispositivo capaz de hacer de interfaz entre un ordenador y el LIDAR diseñado por los desarrolladores. Estos proporcionan además un software capaz de controlar el LIDAR con una secuencia de comandos ya predefinida. Inicialmente se exploró la posibilidad de emplear este dispositivo y el software de los diseñadores para provocar que el LIDAR entrara en el modo de escaneo y simplemente derivar un cable de la UART para ”espiar” las medidas por un pin de la Raspberry. Sin embargo, el objetivo de este proyecto es más ambicioso, pues gran parte del beneficio de este proyecto era disponer de un control fiable y robusto de un sensor tan potente y versátil como el LIDAR, de forma que en proyectos de años posteriores pueda ser utilizado sin ningún problema.

Para ello era necesario que el usuario fuera capaz de controlar el LIDAR en todo momento, por lo que el uso de un software predefinido y opaco al usuario era insuficiente. A causa de

esto se prescindió del software pero se pensó que sería conveniente utilizar el dispositivo de los desarrolladores como interfaz, no solo por las dificultades que entraña generar el PWM para el LIDAR, que se explicarán más adelante en mayor detalle; sino también porque la conexión por USB del dispositivo es más robusta que el conexionado de cables por pines de la Raspberry y además alimenta tanto el dispositivo como el LIDAR. Esto último era quizá la mayor ventaja debido a la excesiva cantidad de dispositivos que incorpora el vehículo en su versión final y las dificultades que ha conllevado alimentarlos todos apropiadamente.

Así pues, se decidió enviar los datos a través del puerto USB al dispositivo y que fuera este el que transmitiera al LIDAR. Siguiendo las instrucciones de los desarrolladores no se consiguió que el LIDAR fuera capaz de escanear y se probó a utilizar un espía del puerto serie vía software que monitorizara el tráfico de paquetes durante el funcionamiento del programa proporcionado por los desarrolladores para tratar de determinar cual era el protocolo apropiado que se debía configurar en el control de Simulink.

Al hacer esto se descubrió que a pesar de que la información del fabricante no lo indicaba, la forma de interactuar empleando el dispositivo de los desarrolladores como interfaz y de forma directa con el LIDAR no era exactamente la misma, si bien eran relativamente similares. Existían dos diferencias clave: la primera es que si se empleaba el dispositivo como interfaz de transmisión era necesario enviar comandos adicionales, entre ellos uno que ordena al LIDAR que comience a girar (lo cual es imprescindible para que transmita medidas por la UART) y que no aparece en la información del fabricante. La segunda es que para aquellos comandos que requieren además del envío de una información adicional no es posible enviar una secuencia ininterrumpida de bytes con toda la información del comando, sino que debe enviarse la instrucción fragmentada en varios paquetes de distinto tamaño, y debido a que esto no es posible con el bloque de Simulink de comunicación por USB de la Raspberry es por lo que finalmente se optó por descartar también esta opción.

Tras demostrarse las dificultades de emplear estos dos métodos para controlar el LIDAR, se exploró la posibilidad de utilizar dispositivos adicionales que actuaran de esclavos de la Raspberry o que simplemente aportaran la señal de PWM que esta no es capaz de generar. Entre estos cabe destacar un oscilador integrado en un circuito capaz de generar una señal de la frecuencia apropiada. Sin embargo, debido a que los problemas referentes al LIDAR no se limitaban a la generación del PWM, la decisión final fue el empleo de un Arduino Nano, que es la alternativa que se explica en detalle en esta memoria.

3.2.4.2. Protocolo de Comunicación por UART

En primer lugar, es necesario enviar al LIDAR el PWM puesto que este solo es capaz de escanear si se encuentra en un estado de rotación estable. Después, comienza la transmisión. Lo referente a la transmisión es bastante sencillo: se envía una secuencia de comandos, el último de los cuales es el comando Express Scan, y en consecuencia el LIDAR envía su respuesta a cada uno de ellos de forma consecutiva, siendo la última respuesta el Response Descriptor del comando Express Scan y acto seguido el LIDAR comienza a enviar los paquetes de medidas de forma continuada. Si todo funciona correctamente, ya no es necesario transmitir ningún comando más, pues el LIDAR permanece enviando medidas todo el tiempo.

Lo que se refiere a la recepción e interpretación de las medidas es, sin duda alguna, mucho más complejo que la transmisión. Tras entrar en el estado de escaneo, el LIDAR envía de forma continuada paquetes de 84 bytes, que contiene 32 medidas cada uno. Como ya se ha mencionado la frecuencia de muestreo es de 4 kHz, por lo que haciendo un sencillo cálculo

se observa que se envía uno de estos paquetes de medidas cada 8 milisegundos, con lo que se hace imprescindible que el tiempo de muestreo del puerto serie de la Raspberry sea igual o inferior.

3.2.4.3. Estructura de los Paquetes de Medidas

Se procede ahora a explicar la estructura de los paquetes y a justificar con ello la secuencia de comandos enviada por el Arduino, que aunque aparenta ser absurda simplifica notablemente la traducción de las medidas.

Esta estructura consiste en un grupo inicial de 4 bytes en el que se integran un byte de sincronización y otro de *checksum* en los dos primeros bytes, conteniendo los otros dos bytes un bit que indica el comienzo de la transmisión y 15 bits que determinan el angulo de referencia para el resto de medidas del paquete . Dicho ángulo es siempre cero en el primer paquete de medidas. Los 80 bytes que siguen se agrupan en 16 grupos de 5 bytes, conteniendo 2 medidas cada grupo, que hace el total de las 32 medidas. Debido a la necesidad de eliminar la información redundante para optimizar todo lo posible la comunicación, los paquetes no envían medidas del ángulo del LIDAR en cada momento sino diferencias angulares empleando el ángulo de referencia mencionado. Cada grupo de 5 bytes incluye los campos de dos diferencias angulares y dos distancias, estructurados de la forma que se explica en la 3.3. Esta estructura, a pesar de ser necesaria para poder muestrear a 4 kHz, dificulta notablemente la traducción de la información a medidas de ángulo y distancia. Esto se debe especialmente a que el cálculo de los ángulos del paquete recibido en el instante *t* requiere no solo la referencia angular de ese paquete sino también la del paquete recibido en el instante *t+1*. Esto que podría parecer ilógico se explica en la información del fabricante y proviene de la fórmula $\theta_k = \omega_i + \frac{AngleDiff(\omega_i, \omega_{i+1})}{32} \cdot k - d\theta_k$, donde la función *AngleDiff* no es otra cosa que la diferencia entre omega en el instante *t+1* y *t* en módulo 360 grados. Esto implicaría un retardo en la traducción de las medidas de un tiempo de muestreo completo en circunstancias normales, pero ajustando ingeniosamente los comandos enviados por el Arduino se consigue artificialmente un desfase de 80 bytes de forma que los últimos 4 bytes recibidos en cada tiempo de muestreo corresponden al paquete siguiente. Este ajuste permite simplificar el algoritmo de interpretación, pues solo es necesario almacenar 3 bytes entre un tiempo de muestreo y otro (uno de los 4 bytes era un byte de sincronización, que se emplea solo para verificar la correcta comunicación) en vez de 83.

3.3. Raspberry

3.3.1. Función

Es la unidad central de procesamiento y actúa de maestro de los demás dispositivos inteligentes del vehículo. La Raspberry es a efectos prácticos un ordenador con un sistema operativo en base Linux. El sistema operativo empleado para este proyecto es una modificación de Raspbian que permite la interacción con MATLAB y Simulink, y es proporcionado por Mathworks.

3.3.2. Justificación de su uso

La justificación del empleo de la Raspberry se basa en 4 razones, principalmente:

- Dispone de una capacidad de procesamiento bastante elevada manteniendo un volumen y peso razonables

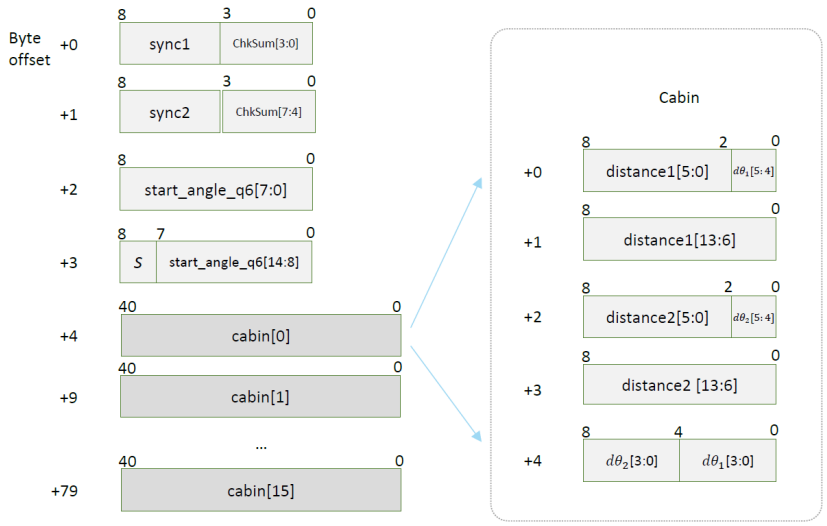


Figura 3.3. Protocolo de RPLIDAR A2 en modo Express Scan. Fuente: SLAMTEC

Figura 3.4. Raspberry Pi 3B. Fuente: www.raspberrypi.org

- Tiene un sistema operativo en tiempo real y permite la convivencia simultánea de procesos de muy distintas velocidades
- Su uso está muy extendido y ello simplifica notablemente algunas cuestiones, pues no solo MATLAB y Simulink incorporan librerías específicas que permiten simplificar la interacción con la Raspberry, sino que además es sencillo encontrar información acerca de como resolver cualquier problema que pueda surgir
- Es un dispositivo muy económico para todas las prestaciones que ofrece. Por todo ello, ha reemplazado a los microprocesadores de la familia STM32F4 que se han empleado en proyectos de años anteriores.

3.3.3. Conexionado

Dispone de múltiples interfaces de comunicación, entre las que destacan: UART, SPI, I2C, Ethernet, Bluetooth, Wi-Fi y USB. Dispone además de pines de propósito general, así como de alimentación a distintos niveles y conexión a masa, que también se han empleado para interactuar con los demás elementos del vehículo. Debido a la cantidad y diversidad

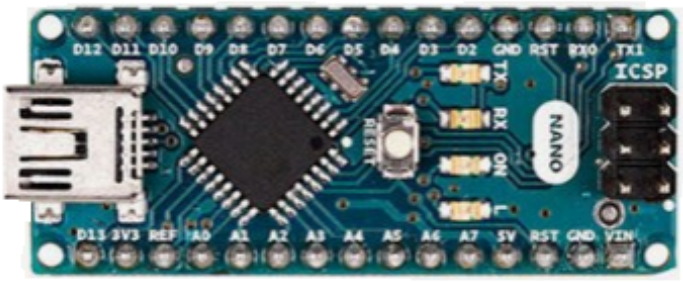


Figura 3.5. Arduino Nano. Fuente: www.arduino.cc

de dispositivos valorados, a lo largo del proyecto se han utilizado todas ellas excepto la comunicación vía Bluetooth, aunque en la versión final ni el Ethernet ni los puertos USB se emplean. Los demás sí se utilizan para interconectar todo el sistema. La comunicación con el conjunto Arduino-LIDAR se ha implementado utilizando la UART para recibir los datos del LIDAR y dos GPIO para controlar al Arduino. Para el control de la IMU se ha utilizado una conexión por SPI y para el control de los drivers de los motores se ha empleado una comunicación por I2C. La comunicación por Wi-Fi es empleada por el protocolo de comunicación Mavlink, que permite enviar y recibir información desde una estación base (habitualmente un ordenador).

3.4. Arduino Nano

3.4.1. Función

El Arduino actúa de interfaz de transmisión entre la Raspberry y el LIDAR, de forma que es este el que el envía todas las órdenes al sensor. Además, genera el PWM necesario para el giro del motor.

3.4.2. Justificación de su uso

El Arduino se ha empleado porque era un dispositivo que resolvía varios de los problemas de comunicación con el LIDAR simultáneamente, además de ser muy sencillo e intuitivo de programar.

3.4.3. Conexionado

Aunque el Arduino es un dispositivo muy potente incluso en su versión Nano, lo cierto es que su uso en este proyecto mínimo, y a diferencia del caso de la Raspberry, se emplean muy pocas de todas las funcionalidades de que dispone. En cuanto a la conexión con el LIDAR, se emplean un pin de propósito general para la generación del PWM y el pin de transmisión por UART para el envío de los comandos.

3.4.4. Funcionamiento

La tarea del Arduino es ejecutar un código que se divide en dos partes: una parte de setup que se ejecuta cada vez que se inicia y un código principal que se ejecuta de forma continuada.

Para controlar el Arduino mediante la Raspberry, esta es capaz de enviar al Arduino dos señales distintas por sendos pines. La primera de ellas comunica al Arduino que debe generar el

PWM para que el LIDAR comience a girar. Debido a que el LIDAR necesita estar en un estado de rotación estable para empezar la transmisión de medidas, esta señal se envía al Arduino al inicializar el programa y se comprueba posteriormente que los bytes recibidos en la Raspberry son correctos. Durante el normal funcionamiento del vehículo, la única tarea del Arduino después del setup es la generación del PWM. Sin embargo, en caso de que falle la sincronización entre el LIDAR y la Raspberry, esta última detiene la señal que activa el PWM y, tras un tiempo prudencial para que el LIDAR frene totalmente y evitar dañar el motor, envía una señal de reset al Arduino, de forma que el proceso de setup comienza de nuevo. Debido a las características del protocolo de comunicación con el LIDAR y al software de interpretación de las medidas, la coordinación entre los tres procesos es fundamental.

3.5. Drivers de los Motores

3.5.1. Función

El propósito de los drivers es doble: permiten controlar los motores del vehículo y además proporcionan la información de los encoders a la unidad central de procesamiento.

3.5.2. Justificación de su uso

Para la elección de los drivers de los motores se han valorado dos opciones: el modelo MD25, que es el empleado en el diseño final; y el modelo CHEETAH, que se descartó por las razones que se exponen a continuación.

3.5.2.1. Modelo CHEETAH 1.0

Este driver tiene dos inconvenientes que dificultan integrarlo en el vehículo: el primero es que cada driver es capaz de controlar un único motor, por lo que como mínimo serían necesarios dos drivers para controlar el coche, y en caso de que en un futuro se pretendiera controlar las 4 ruedas de forma independiente, habría que añadir dos drivers más, lo cual resulta prácticamente inviable por cuestiones de comunicación. Esto nos lleva al segundo y más grave problema de este driver: el CHEETAH 1.0 se puede comunicar únicamente por SPI, y un único driver tiene dos esclavos distintos: el encoder y el puente en H; ambos con comunicaciones independientes. Debido a que la comunicación SPI de la Raspberry soporta solo dos esclavos esto habría implicado utilizar un pin adicional al chip enable para elegir el dispositivo dentro de cada driver, además de impedir que ningún otro dispositivo pudiera funcionar mediante SPI, como es el caso de la IMU en la versión final. Esto no habría sido crítico para este proyecto, pues habría sido posible emplear dos drivers CHEETAH e implementar la comunicación de la IMU mediante I2C, pero para futuros desarrollos dificultaría mucho añadir nuevo hardware al vehículo, pues la ausencia de SPI limita bastante las opciones de comunicación.

3.5.2.2. Modelo MD25

A diferencia del otro modelo, este driver es capaz de controlar dos motores independientes simultáneamente, por lo que aunque finalmente se han empleado dos dispositivos, habría sido posible utilizar solamente uno para los cuatro motores. Además, la comunicación con este modelo se lleva a cabo mediante I2C, por lo que no se agota ninguna opción de comunicación para otros dispositivos.

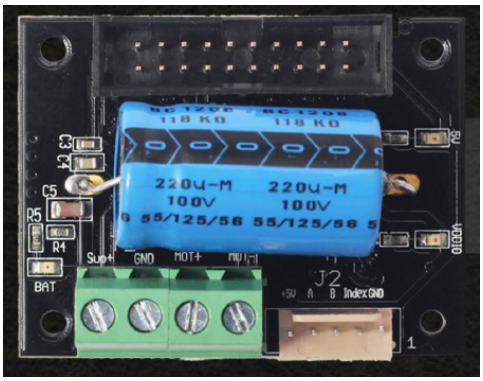


Figura 3.6. Driver CHEETAH 1.0. Fuente: Phi Robotics

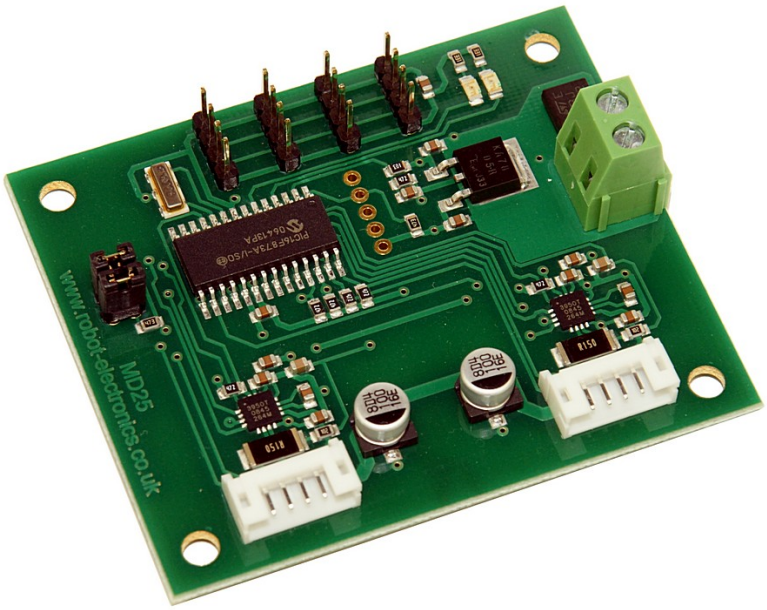


Figura 3.7. Driver MD25. Fuente: SuperRobotica.com

3.5.3. Conexionado

Las conexiones de ambos drivers se limitan a la alimentación, la conexión con los motores y la comunicación por I2C con la Raspberry. La alimentación de ambos es proporcionada por una batería de 12 voltios (no se requiere regulador en este caso). La conexión con los motores es sencilla y directa, pues estos disponen de un conector macho de 6 pines, que se conectan a una conexión hembra idéntica en los drivers. Para la comunicación con I2C ha sido necesaria la utilización de un pequeño regulador que convierte los 5 voltios lógicos del driver a los 3,3 voltios que utiliza la Raspberry para trabajar.

3.5.4. Funcionamiento

Los drivers controlan los 4 motores mediante un PWM. Modificando el factor de servicio, estos son capaces de regular la velocidad de los motores en el total del rango de operación de estos, entre 0 y 170 revoluciones por minuto en ambos sentidos. Para el valor del PWM utilizan un byte, por lo que la resolución es de 1,33 revoluciones por minuto aproximadamente. Además, estos drivers permiten limitar la aceleración de los motores durante la puesta en marcha del vehículo escribiendo en registros concretos durante la inicialización, de forma que se satura el mando durante el transitorio inicial sin la intervención de la Raspberry durante dicho transitorio.

Para la medida de la velocidad del vehículo a través de los encoders se utiliza un registro específico donde se incrementa el número de pulsos detectados por cada encoder, de forma que la lectura de este registro periódicamente permitiría calcular el desplazamiento del vehículo con una resolución de menos de una vuelta entera de la rueda (que equivale a una distancia de 3,14 centímetros), sin embargo, para ello sería necesario resetear el contador periódicamente para que en caso de leve deslizamiento de las ruedas no se acumule error permanentemente en la integración discreta. Es por ello que se ha decidido emplear las medidas de los encoders únicamente para calcular la velocidad promedio del vehículo entre dos mediciones del registro. Considerando que dicho registro se puede leer cada 8 milisegundos y que la velocidad del vehículo no es demasiado alta, dicho valor se puede considerar constante entre dos lecturas sin cometer un error apreciable.

3.6. Motores y Encoders

3.6.1. Función

Los motores cumplen dos funciones: la más obvia es que convierten la energía de la batería en movimiento. La otra función es medir la velocidad del vehículo a través de los encoders y enviar la información a los drivers.

3.6.2. Justificación de su uso

Para la elección de los motores se han valorado tres opciones: motores de Lego idénticos a los utilizados en el laboratorio de regulación, motores modelo CHIHAI MOTOR 6V 210RPM y los empleados finalmente, motores modelo EMG30.

3.6.2.1. Motores de Lego

Esta opción se valoró en la fase inicial del proyecto debido a que su control era muy sencillo, pues dicho control ya estaba diseñado y más que probado en las asignaturas de automática. Estos motores se descartaron debido a que la carcasa estaba hecha de plástico y se creyó que no sería lo bastante robusto para una versión avanzada del coche. A causa de esto se decidió buscar otra alternativa.

3.6.2.2. Motores CHIHAI MOTOR 6V 210RPM

Estos fueron los motores que se pensó que serían el sustituto definitivo a los motores de Lego: la estructura era metálica y robusta, y los encoders tenían una resolución muy alta. El único inconveniente que tenían en apariencia es que el nivel de tensión es de 6 voltios en lugar de los 12 nominales del driver, por lo que los valores de PWM debían saturarse al 50 % y esto disminuiría a resolución a la mitad. Debido a que el vehículo no está diseñado para alcanzar grandes velocidades, ni la resolución en la tensión de los motores era un factor crítico se aceptaron ambos inconvenientes y se procedió a probar el diseño. Sin embargo, al integrar los motores y los drivers y comprobar el funcionamiento de estos se observó que aun cuando el driver debía enviar un PWM constante, el ancho de pulso que los motores recibían variaba sensiblemente a causa de que el motor no estaba preparada para los 12 voltios en el nivel alto del PWM y esto producía vibraciones en el vehículo. Debido a esto se buscó una alternativa diferente.

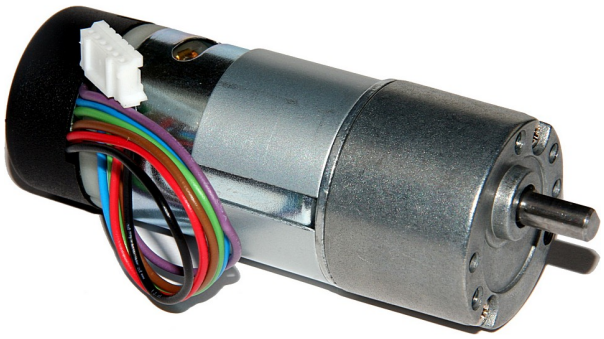


Figura 3.8. Motor EMG30. Fuente: SuperRobotica.com

3.6.2.3. Motores EMG30

Tras los problemas encontrados en la búsqueda de unos motores apropiados para el vehículo se tomó la decisión de utilizar los motores recomendados por el fabricante del driver empleado. Estos tienen la misma tensión nominal que el driver, por lo que se mantiene toda la resolución en el mando de los motores.

3.6.3. Conexionado

Los motores tienen conectores de 6 pines que se conectan directamente a los drivers. Estos 6 pines consisten en: alimentación y conexión a masa de los sensores de efecto Hall, la medida de los dos sensores integrados en el motor, dos pines para la señal de PWM del motor.

3.6.4. Funcionamiento

Los motores tienen el funcionamiento habitual de los pequeños motores de continua: convierten la energía eléctrica en energía mecánica que se transmite a la rueda y la hace girar. El funcionamiento de los encoders es, sin duda, mucho más interesante y por ello aparece explicado en el estado del arte de esta memoria.

3.7. IMU

3.7.1. Función

El propósito de la IMU es medir la aceleración y orientación del vehículo en todo momento, de forma que se disponga de información del vehículo, que junto a la proporcionada por el resto de sensores se integra en el EKF.

3.7.2. Justificación de su uso

La IMU es un sensor muy utilizado en aplicaciones similares a esta, y por ello se integra una en el vehículo. El modelo valorado inicialmente se ha empleado en proyectos en años anteriores y tiene muy buen resultado por lo que la única disyuntiva ha sido si debía integrarse el modelo MPU6250 o el modelo MPU9250, cuyo conexionado es idéntico y cuya única diferencia reside en que el segundo incorpora además un magnetómetro. A pesar de que este solamente funciona

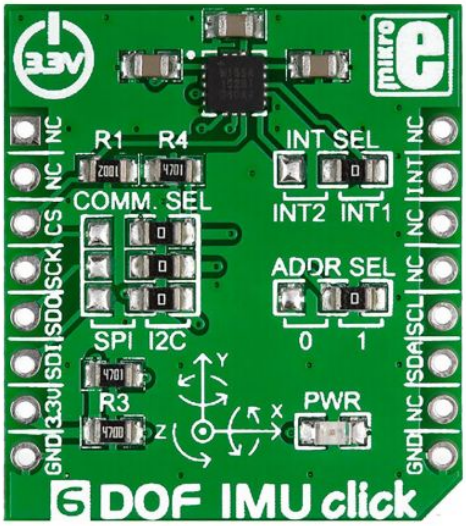


Figura 3.9. IMU MPU9250. Fuente: RS Components

correctamente en exteriores y el LIDAR que se está empleando solo lo hace en interiores, se ha creído que dada la mínima diferencia de precio, sería conveniente integrar el modelo que dispone de magnetómetro, puesto que sustituyendo el LIDAR por uno diseñado para operar en exteriores, la estructura y el hardware del coche estarían perfectamente capacitados para funcionar en exteriores.

3.7.3. Conexionado

El vehículo incorpora dos *Hats* para Raspberry que tienen como finalidad hacer más sencillas y robustas las conexiones. Uno de estos *Hats* está pensado especialmente para esclavos de la Raspberry conectados por SPI o I2C, y por ello la IMU encaja perfectamente en ese *Hat*. Dicha conexión aúna la comunicación completa por SPI, la alimentación y la conexión a masa.

3.7.4. Funcionamiento

La IMU está dotada de 3 grupos de sensores: 3 acelerómetros, 3 giróscopos y 3 magnetómetros, que permanecen inutilizados. En los 3 casos cada sensor se corresponde con el valor en un eje, de forma que las medidas están desacopladas. De esta forma la IMU es capaz de medir las aceleraciones y las velocidades angulares en los 3 ejes, aunque debido a que el movimiento del vehículo está restringido al plano horizontal en su versión actual, solo 3 de estas medidas se utilizan: aceleración en los ejes X e Y, y velocidad angular en el plano XY.

3.8. Batería LiPo de 3 celdas de alta descarga

3.8.1. Función

El propósito de la batería es simple y llanamente alimentar todo el vehículo.

3.8.2. Justificación de su uso

Existen dos cualidades que justificar: la tensión, que depende directamente del número de celdas; y el amperaje. Los drivers operan a 12 voltios, y debido a que los demás dispositivos

Figura 3.10. Batería LiPo de 3 celdas de alta descarga

operan a menos tensión, 3 celdas es lo mínimo imprescindible para el funcionamiento del vehículo y más tensión implicaría añadir un regulador para alimentar los drivers, por lo que la decisión es clara. En lo relativo al amperaje, la disponibilidad de baterías fue la que justificó que se empleara una de alta descarga (60 amperios), cuando el coche bajo ninguna circunstancia podría consumir más de 10 amperios en su estado actual.

3.8.3. Conexionado

La salida de la batería se deriva directamente a dos tomas de alimentación, una que alimenta los drivers y otra el regulador que reduce la tensión a 5 voltios, y alimenta el resto de dispositivos. Los drivers lado tienen una tolerancia elevada en la tensión de alimentación, por lo que no se hace necesario ningún regulador con relación 1:1 y la conexión es directa.

3.9. Reguladores de tensión

3.9.1. Función

Los reguladores presentes en el vehículo son necesarios para proporcionar a cada elemento la tensión correcta.

3.9.2. Funcionamiento

Utilizando conversores de tensión tipo *chopper* y adicionalmente sistemas de protección del nivel de tensión, estos reguladores son capaces de proporcionar un nivel de tensión relativamente estable a la salida aún cuando la entrada no es demasiado estable (realmente con la batería utilizada la variación no es demasiado elevada, pero si que podría alcanzar los 2 voltios). Esto es especialmente relevante para alimentar la Raspberry y el LIDAR.

3.9.3. Conexionado

La entrada es la tensión de la batería y la salida son simplemente una tensión a 5 voltios con su tierra correspondiente, que coincide con la de la batería, de forma que la conexión a masa sigue siendo común a la del resto del vehículo.

4 Software

4.1. Estructura del código de Simulink

El código de Simulink está estructurado en bloques en diferentes niveles de jerarquía. En el nivel superior hay dos bloques: el bloque de hardware y el bloque de control.

Bloque de hardware

- Sensores
- Actuadores
- Comunicaciones

Bloque de control

- Algoritmo de control
- Máquina de estados global
- Codificación y descodificación de las comunicaciones

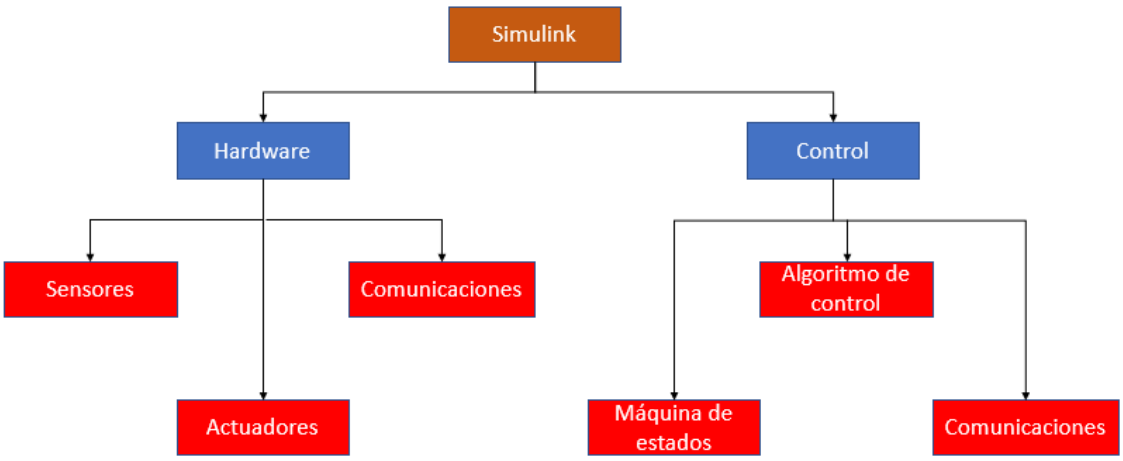


Figura 4.1. Esquema de la jerarquía del código de Simulink.

4.1.1. Bloque de hardware

4.1.1.1. Sensores

En este bloque se encuentran los controles de la IMU y el LIDAR. En el caso de la IMU se produce una inicialización de los registros SPI que se requieren para el uso de los giróscopos y acelerómetros (se ignoran los correspondientes a los magnetómetros) y se leen los datos brutos de la IMU para después traducir esa información bruta en medidas escaladas a las unidades apropiadas. El caso del LIDAR es más complejo, pues el software de control del Simulink interactúa directamente con el Arduino Nano que transmite al LIDAR y con la transmisión del LIDAR. Es por ello que este proceso requiere de una máquina de estados propia situada en la Raspberry para coordinar correctamente todo el proceso. Esta máquina de estados consta en realidad de solamente 3 estados: un estado de inicialización en el que se da la orden al Arduino de transmitir el PWM digital para que gire el LIDAR y se permanece en dicho estado hasta que el LIDAR comience a transmitir. Como medida preventiva en este estado hay un contador que llegado a cierto número (en este caso es 50 períodos de muestreo, es decir, unos 400 milisegundos, aunque esta es una cifra relativamente arbitraria que podría probablemente optimizarse) transmite la señal de resetear el Arduino y resetea el contador. En caso de que no exista ningún problema, en este tiempo el Arduino transmitiría los mensajes pertinentes de forma que el LIDAR respondería a la Raspberry con los Response Descriptor de control y tras verificarlos la máquina de estados pasaría al estado siguiente, el de operación normal. En este estado, se procesa toda la información transmitida por el LIDAR y se convierte a pares distancia-ángulo. En el propio protocolo de comunicación del LIDAR existen grupos de bits de control como se explica en la sección de 3 y en este estado se emplean para verificar que la comunicación es correcta. En caso de no serlo se pasa al último estado, el cual se resetea el Arduino directamente y también el contador de reset y se vuelve al estado de inicialización. Se explica a continuación como opera el código del Arduino y su interacción con la Raspberry y el LIDAR.

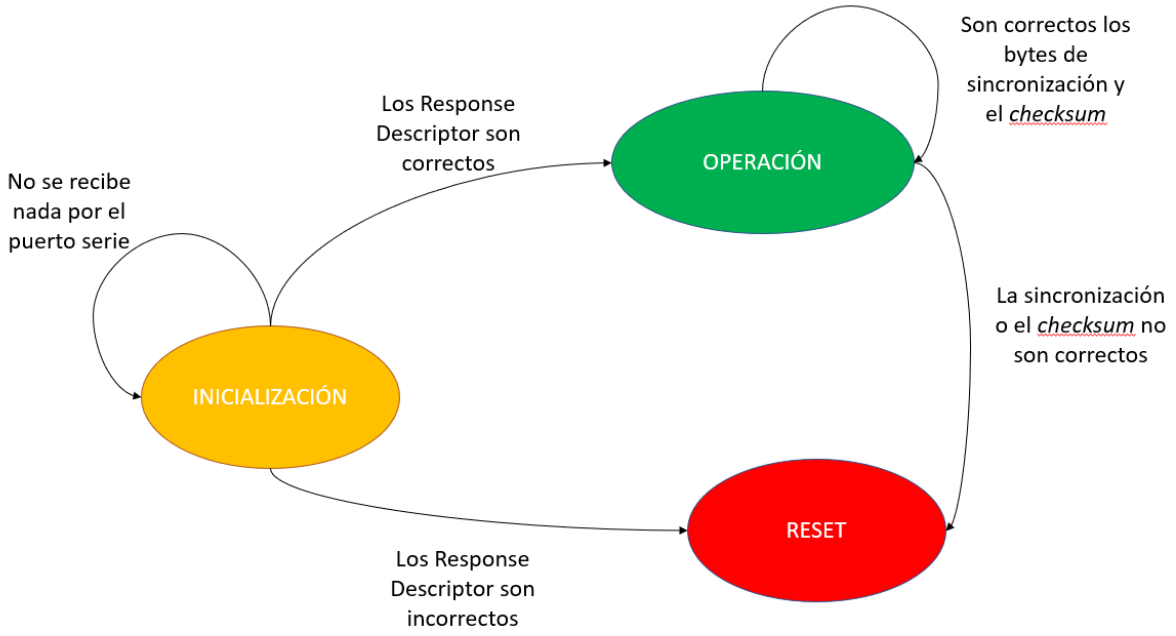


Figura 4.2. Máquina de estados del control del LIDAR.

Explicación del código de setup

El código de setup que ejecuta consiste en una secuencia de 11 comandos que tiene como fin generar el desfase artificial de 80 bytes en la comunicación entre el LIDAR y la Raspberry. Este desfase se genera enviando primeramente un comando al que le corresponde un Response Descriptor de 10 bytes y después enviando 10 veces seguidas el mismo comando, el de Express Scan, al que el LIDAR responde con un Response Descriptor de 7 bytes y acto seguido con los paquetes de medidas. Debido a que los comandos se envían de forma ininterrumpida el LIDAR solo entra en el estado de escaneo tras el último comando y responde a los anteriores únicamente con el Response Descriptor que le corresponde (haciendo una sencilla cuenta se comprueba que $10 \cdot 1 + 7 \cdot 10 = 80$ bytes de desfase.

Explicación del código principal

Una vez que se ejecuta este código de setup la misión del Arduino es la de generar el PWM digital por uno de sus pines, y detener dicho PWM en el caso de que la Raspberry envíe la señal de que la comunicación no está siendo correcta. Para generar el PWM simplemente se han empleado funciones de retardo de Arduino para ajustar tanto el periodo como el factor de servicio. Debido a la inexactitud del reloj interno del Arduino ha sido necesario ajustar el período teórico del PWM para que sea de 42 microsegundos (en vez de los 40 que son necesarios para conseguir una frecuencia de 25 kHz) para que la frecuencia medida con el osciloscopio sea la correcta (25 kHz +- 2 %). Para simplificar el código y debido a que no es crítico, el factor de servicio se ha ajustado a 24 de los 42 microsegundos, con lo que se consigue un 57,14 % de factor de servicio aproximadamente (cercano al 60 % de operación nominal del LIDAR, como se indica en la información del fabricante).

4.1.1.2. Actuadores

En este caso hay un único bloque de actuadores que consiste en dos drivers que controlan cada uno los dos motores de un lado del coche. En ambos casos el funcionamiento es el mismo. Primero se inicializan los registros I2C empleados y en el funcionamiento normal se llevan a cabo tres tareas: lectura de la tensión de la batería, lectura y procesado de la información de los encoders y envío del PWM de los motores. Es interesante en este apartado el procesado de la información de los encoders, para el cual se ha implementado un filtro de media móvil de rango variable que mide el registro de la cuenta y empleando medidas anteriores además de los parámetros pertinentes del motor y el vehículo calcula una serie de variables de interés para el control como pueden ser la velocidad lineal del vehículo, su aceleración o la velocidad angular del motor [28,29]. Posteriormente, se emplean las velocidades medidas en las cuatro ruedas del vehículo para calcular el módulo y sentido de la velocidad del cuerpo en la dirección de avance del vehículo, pues es esta información la que requiere el EKF. En este apartado es interesante destacar la amplia variedad de opciones de filtrado de la medida de los encoders, como emplear ponderaciones exponenciales con un cierto factor de olvido o métodos más avanzados utilizando microprocesadores específicamente para esta tarea, pues uno de los grandes inconvenientes del uso de encoders es que la resolución depende del tiempo de muestreo y cuanto menor sea este mayores son los problemas asociados a la discretización [30].

4.1.1.3. Comunicaciones

Para las comunicaciones externas se ha empleado el protocolo MAVLINK mediante UDP, de forma que el vehículo es capaz de recibir y enviar información a una estación base, típicamente

un PC. De esta forma la estación base envía la referencia al vehículo, ya sea de posición, velocidad o cualquier otra y este es capaz de enviarle cualquiera de las variables de operación para monitorizarlas desde la estación. Las principales ventajas de emplear este método es que la comunicación es *full-duplex*, de forma que es posible transmitir y recibir al mismo tiempo; y que es mucho más rápida que por ejemplo la comunicación por TCP, manteniendo unas prestaciones de robustez de la comunicación casi idénticas en el ámbito de uso del proyecto.

4.1.2. Bloque de Control

Este bloque se divide en tres grandes partes: la codificación y descodificación de la comunicación, el algoritmo de control y la máquina de estados global.

4.1.2.1. Comunicaciones

La codificación y descodificación de la información transmitida por UDP se realiza empleando patrones de inicio de mensaje con número de identificación del tipo de mensaje, de forma que de antemano deben estar definidos de la misma manera los tipos y campos de cada posible mensaje. Hay además bytes de *checksum* al final de cada transmisión para garantizar que el mensaje no está corrupto.

4.1.2.2. Algoritmo de Control

El algoritmo de control consiste en un triple PID en cascada en paralelo con un doble PID en cascada que controla mediante un PID cada una de las variables de estado del vehículo (despreciando la inductancia de los motores), que son: velocidad y posición lineales (solo en los ejes X e Y, puesto que el vehículo no está pensado para circuitos con rampas) y velocidad y posición angulares (en este caso solo la guiñada o rotación respecto del eje Z, por la misma razón). El orden de la anidación en niveles del control en cascada es: en el lazo más interno la velocidad angular, el lazo inmediatamente exterior corresponde a la posición angular, y por último la posición del vehículo en los dos ejes; y por otro lado se controla internamente el módulo de la velocidad lineal del vehículo, estando en el lazo exterior el control de posición.

Propagación en los distintos niveles del control

El lazo exterior, el de la posición en los dos ejes calcula la el objetivo de velocidad aplicando un PID conocidas la posición actual y la referencia, y propaga las velocidades al siguiente nivel. En este nivel se tiene en uno de los controles las posición angular y en otro el módulo de la velocidad. En el primer caso se emplea la tangente de ambas velocidades para calcular la posición angular deseada y con un PID se calcula la velocidad angular requerida. Después esta es transmitida al último nivel como referencia y se emplea para calcular el mando de tensión diferencial de los motores. En el caso del módulo de la velocidad, se calcula la media de las velocidades en ambos ejes para calcular dicho módulo y tras aplicar un control nuevamente se obtiene el mando de tensión común de los motores.

4.1.2.3. Máquina de estados global

La máquina de estados del vehículo tiene 4 estados: un estado de inicialización, un estado de calibración de los sensores, un estado de operación normal, y un estado de fallo 4.4.

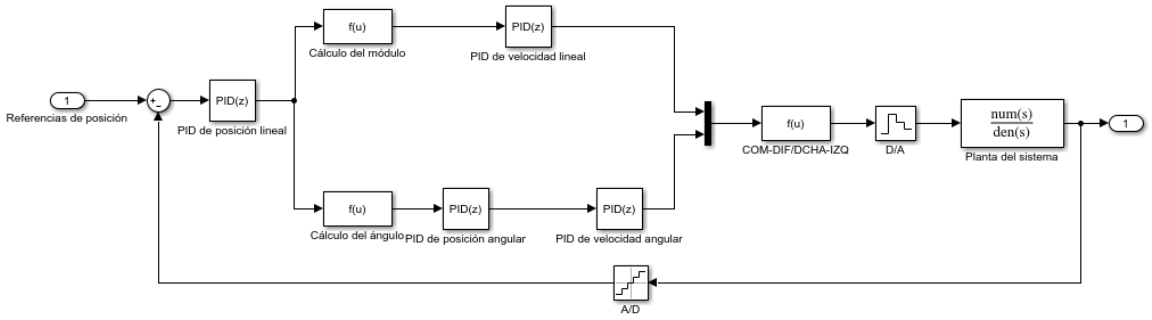


Figura 4.3. Diagrama de bloques del control implementado.

Inicialización

Es un estado necesario en el cual se pretende que todos los elementos alcancen el régimen de trabajo. Dura 3 segundos.

Calibración

Durante 30 segundos se toman medidas de los acelerómetros, los giróscopos y el LIDAR. Con las medidas de los dos primeros se calcula una media y se considera dicho valor el *offset* del sensor (exceptuando la aceleración en el eje Z en el cual se compara contra la gravedad), de forma que posteriormente se resta este valor en las medidas. En el caso del LIDAR se hace un mapeo inicial del entorno que será el utilizado en el siguiente modo. Para toda esta fase es imprescindible que el vehículo se encuentre en posición horizontal por lo explicado anteriormente. En este estado la Raspberry ignora cualquier referencia que se le envíe desde la estación base, de forma que no actúa sobre los motores.

Operación

Después de la calibración, la Raspberry es capaz de recibir instrucciones desde la estación base y actuar en consecuencia de forma que el vehículo sigue las referencias.

Fallo

Se entra en este estado si se corta la comunicación con algún sensor o si la tensión de la batería baja demasiado. En él se paraliza el vehículo completamente.

4.2. EKF

El EKF integrado en este proyecto emplea una serie de variables de estado, que son aquellas que se estiman en todo momento; y las medidas de los sensores correspondientes, que utiliza en dichas estimaciones. Para ello se relacionan esas variables de estado con unas entradas artificiales asociadas a los giróscopos y acelerómetros y con el valor de las variables de estado en el instante anterior. Con esta información el EKF calcula las medidas esperadas por cada uno de los sensores integrados y posteriormente las compara con las medidas reales considerando un cierto ruido de medida que es *gaussiano*, de media nula y varianza asociada a la precisión del sensor, esto es, más reducida cuanto más fiable se considera al sensor concreto. En función de métodos estocásticos a lo largo del tiempo consigue reducir el error asociado al ruido del modelo y de los sensores. Las ecuaciones asociadas a los cálculos de las variables de estado son:

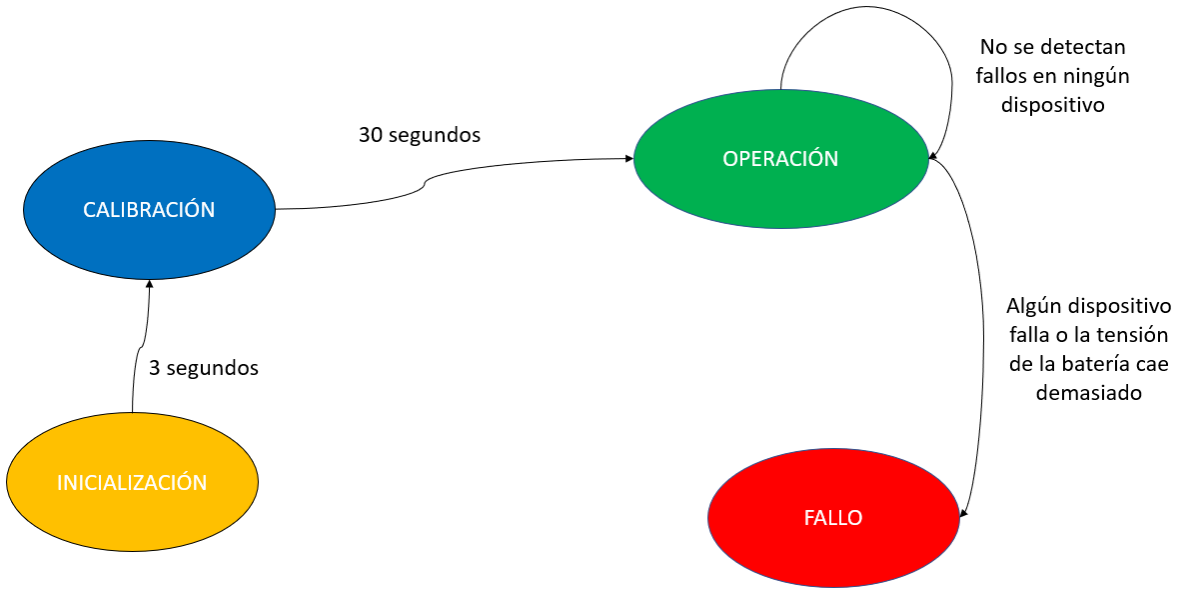


Figura 4.4. Máquina de estados global.

$$x[k] = f(x[k-1], u[k], w[k]) \tag{4.1}$$

$$z[k] = h(x[k], v[k]) \tag{4.2}$$

Donde x representa el vector de variables de estado, u el vector de entradas y z el vector de salidas. v y w representan los ruidos de los sensores y el modelo.

De esta manera, lo único que en realidad hace falta para integrar un sensor nuevo en el EKF es ser capaz de relacionarlo con las variables de estado estimadas. En este caso las variables de estado son la posición angular y lineal del vehículo, su velocidad lineal, y unas variables artificiales que son los sesgos de los acelerómetros y giróscopos. Debido a que se utilizan como entradas, la información de los acelerómetros y giróscopos se utiliza en los cálculos del EKF, si bien luego no se contrastan directamente sus medidas con las variables de estado en este proyecto. Por ello, los únicos sensores que se relacionan con las variables de estado son los encoders y el LIDAR. La ecuación de los encoders es muy sencilla y es la que sigue:

$$|v| = \sqrt{(v_x)^2 + (v_y)^2} \tag{4.3}$$

El caso del LIDAR es más complejo. Las ecuaciones que relacionan al LIDAR con las variables de estado requieren además de los valores anteriores de las medidas del entorno y de cálculos geométricos que no son tan intuitivos como en el caso de los encoders.

La geometría asociada a los cálculos del LIDAR se muestra en la figura 4.5, de donde se desarrollan las ecuaciones que siguen:

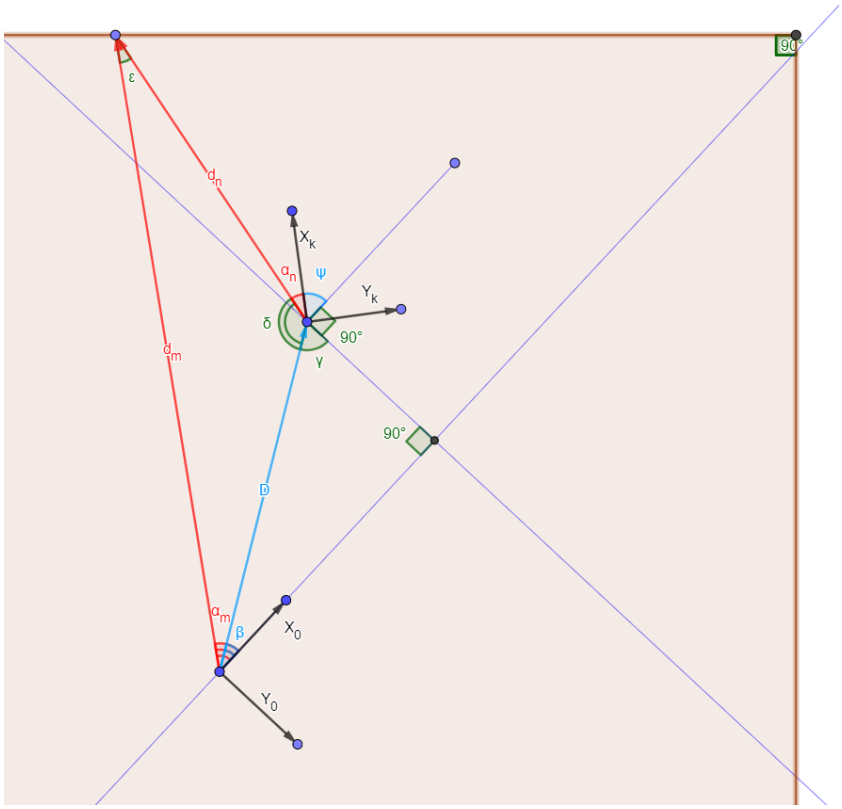


Figura 4.5. Geometría de los cálculos del LIDAR en el EKF. Imagen realizada con Geogebra.

En la imagen 4.5 se representan en rojo las variables asociadas a las medidas del LIDAR en los instantes 0 y k, en azul claro las variables asociadas al estado del vehículo en el instante k (posición en dos ejes en coordenadas polares para simplificar las ecuaciones y ángulo de guiñada), en verde varios ángulos auxiliares empleados en los cálculos y en azul oscuro rectas auxiliares. En negro se muestran los ejes del vehículo en el instante 0 (referencia respecto a la Tierra) y en el instante k (en ambos casos el eje Z apunta hacia el suelo).

Se muestran a continuación las ecuaciones que relacionan las variables representadas con las variables de estado del LIDAR para su implementación en el EKF. Los superíndices se corresponden con referencias temporales y los subíndices con referencias de posición en una lista ordenada de valores en cada instante. Aunque tanto el dibujo como la explicación de las ecuaciones hacen referencia a desplazamientos medidos respecto al instante inicial también sería posible actualizar el mapa a cada instante de muestreo de forma que se comparan los instantes *k* y *k+1*. Además, aunque se representa la habitación como un rectángulo es válido cualquier forma geométrica siempre y cuando el recinto resultante sea convexo (esto es, que desde cualquier punto pueden verse el entorno íntegro).

$$\widehat{D}^{k,0} = \sqrt{(\widehat{x}^{k,0})^2 + (\widehat{y}^{k,0})^2} \tag{4.4}$$

$$\widehat{\beta}^{k,0} = arctg\left(\frac{\widehat{y}^{k,0}}{\widehat{x}^{k,0}}\right) \tag{4.5}$$

Cambio a polares de la posición en los dos ejes.

$$\gamma_n^k = 180 - \widehat{\beta}^{k,0} - 90 \tag{4.6}$$

$$\delta_n^k = 360 - \gamma_n^k - 90 - \widehat{\psi}^{k,0} - \alpha_n^k \tag{4.7}$$

$$\epsilon_n^k = 180 - \left(\alpha_m^0 - \widehat{\beta}^{k,0}\right) - \delta_n^k \tag{4.8}$$

Cálculos y definiciones de ángulos.

$$\frac{d_m^0}{sin\left(\delta_n^k\right)} = \frac{d_n^k}{sin\left(\alpha_m^0 - \widehat{\beta}^{k,0}\right)} = \frac{\widehat{D}^{k,0}}{sin\left(\epsilon_n^k\right)} \tag{4.9}$$

Cálculos geométricos de distancias con el teorema del seno.

$$d_m^0 = f_{mapa}(\alpha_m^0) \tag{4.10}$$

Relación entre un ángulo dado y su distancia en el mapa elaborado en la calibración.

Como se ha explicado, el objetivo es obtener de este conjunto de ecuaciones el valor esperado de la medida del LIDAR en el instante *k* en función de las variables de estado y otros datos conocidos. Para un cierto ángulo que sí es conocido, pues es a partir del cuál se calcula la distancia se pretende conocer entonces d_n^k. Tras el cambio a coordenadas polares restan 6 ecuaciones distintas. Se observa que hay únicamente 6 incógnitas, a saber: $\gamma_n^k, \delta_n^k, \epsilon_n^k, d_m^0, \alpha_m^0, d_n^k$. Por tanto, el sistema tiene solución, como dicta la intuición. El problema es que dicha solución no es explícita, es más, implica a la función f_{mapa}, que es una función artificial creada dándole continuidad a una lista de pares ángulo-distancia obtenida en el mapa inicial. Esto complica notablemente resolver estas ecuaciones y obtener una expresión de la forma $d_n^k = f\left(\widehat{x}^{k,0}, \widehat{y}^{k,0}, \widehat{\psi}^{k,0}, \alpha_n^k\right)$. Además, esta expresión luego tiene que derivarse, pues se necesita el jacobiano para los cálculos del EKF y aunque es posible darle a este mapa carácter C^1 o C^n si así se quisiera mediante derivadas discretas, estas expresiones se tornan inmanejables.

Y esto no es todo. Existe otro problema igualmente insalvable con el hardware de que se dispone. En los cálculos internos del EKF, en cada tiempo de muestreo se debe calcular la inversa de una matriz de número de filas igual al número de medidas que se desea estimar, lo cual se vuelve imposible aunque solo se hicieran los cálculos para el LIDAR y se obviaran otros sensores. Esto se debe a que el LIDAR proporciona cada tiempo de muestreo (esto es, 8 milisegundos) 32 medidas, y esto constituye demasiada carga computacional. Las soluciones más sencillas implican eliminar bastante información asociada a las medidas y como esto claramente no es lo deseable, se plantea una solución alternativa.

4.2.1. Implantación del LIDAR en el EKF

La solución planteada para la integración del LIDAR consiste en realizar un preprocesado de las medidas del LIDAR, de forma que la salida de este bloque sería la estimación directa de las variables de posición y orientación y el ajuste en el EKF sería trivial. Para poder convertir estas medidas en las variables mencionadas primero es necesario una lista ordenada de valores del mapa como la que proporciona el mapeo inicial. Con este método, es la resolución de esta lista la que determina la carga computacional, por lo que se puede optimizar el tamaño en función de la potencia computacional disponible. Para realizar dicha optimización únicamente sería necesario interpolar los valores del mapa hasta reducir la longitud de la lista al valor deseado.

Este preprocesado se realiza en dos fases: en primer lugar se crea una lista ordenada de valores que cubra todo el mapa. Dichos valores pueden provenir del mapa inicial o bien puede actualizarse cada cierto tiempo. La segunda fase consiste en utilizar las medidas de cada instante del LIDAR para estimar por mínimos cuadrados las variables de estado del vehículo. Para ello se emplean las ecuaciones 4.4-4.10, solo que en este caso las incógnitas son las variables que determinan el desplazamiento y rotación del vehículo, puesto que ahora sí que se dispone de dos mapas distintos. Expresando las variables en forma compleja, la operación de mínimos cuadrados se realiza sobre la siguiente ecuación:

$$\begin{bmatrix} \overrightarrow{r_{e_1}} \\ ... \\ \overrightarrow{r_{e_n}} \end{bmatrix} = \begin{bmatrix} \overrightarrow{r_{b_1}} + \overrightarrow{r_{lidar}} & 1 \\ ... & ... \\ \overrightarrow{r_{b_n}} + \overrightarrow{r_{lidar}} & 1 \end{bmatrix} \begin{bmatrix} e^{j\psi} \\ x + jy \end{bmatrix} \tag{4.11}$$

Como se ha dicho, la ecuación 4.11 no es más que una forma alternativa pero equivalente de expresar las ecuaciones 4.4-4.10 en forma vectorial y de manera más compacta. Volviendo a la figura 4.5, los vectores $\overrightarrow{r_b}$ y $\overrightarrow{r_e}$ representan los vectores entre el coche y un punto concreto del entorno, en el primer caso empleando el sistema de referencia de la Tierra, que coincide con el del vehículo en el instante inicial; y en el segundo caso con el del vehículo en un cierto instante *k*. Estos se corresponden con los vectores rojos de distancia-ángulo de la figura. En esta representación se ha considerado el caso más genérico en el que el LIDAR no se haya situado sobre el centro de masas del vehículo. Esto se ha obviado en la imagen para que sea más fácil de entender, pero puesto que el vector que relaciona la posición del LIDAR con el centro de masas del vehículo es conocida de antemano no complica en ningún caso los cálculos. Los datos que se pretenden estimar son esta vez el vector desplazamiento (expresado en este caso en forma cartesiana en vez de polar) y el ángulo de guiñada, de forma que la operación matricial representa una traslación+rotación del sistema de referencia de la Tierra. Cabe destacar que, teóricamente, para garantizar una mayor exactitud en la estimación sería necesario añadir una restricción no-lineal para garantizar la unitariedad del vector de rotación $e^{j\psi}$.

$$cos(\psi)^2 + sin(\psi)^2 = 1 \tag{4.12}$$

Sin embargo, esto aumentaría de manera no despreciable la carga computacional de la estimación, por lo que dependiendo de la resolución del mapa esto no solo resultaría innecesario sino que sería desaconsejable. Con el fin de reducir la carga computacional, es posible hacer la estimación por mínimos cuadrados recursiva. Expresado de esta forma, el criterio de mínimos cuadrados es [31]:

$$\underset{\widehat{\theta}(t)}{min} J(t) = \sum_{i=1}^{t} \left[y(i) - \widehat{y}(i|\widehat{\theta}(t)) \right]^2 \tag{4.13}$$

Aunque existen métodos de eficiencia mejorada para atacar problemas concretos, como el algoritmo extendido de mínimos cuadrados recursivos y algunos otros que se explican en [31–33], lo cierto es que estos no resuelven de manera satisfactoria el problema planteado, pues dada la precisión del LIDAR ni siquiera los métodos específicos para tratar no-linealidades como el planteado en [34] podrían mejorar suficientemente la precisión como para que sea rentable utilizar carga computacional adicional.

A pesar de que este método simplifica notablemente los cálculos, falta aun un problema por resolver: el LIDAR no realiza un mapa completo instantáneamente, sino que necesita de varios tiempos de muestreo para ello (a la velocidad de rotación con que se usa, requiere exactamente de 18 tiempos de muestreo, esto es, $8 \cdot 18 = 144 milisegundos$). Es aquí cuando cobra utilidad la ventaja de la Raspberry mencionada en el capítulo 3, que permite la convivencia de procesos de distintos tiempos de muestreo. Considerando esto, es posible implementar este método utilizando simultáneamente las estimaciones del bloque de preprocesado y del EKF. Es incluso posible que ambos bloques se realimenten mutuamente, de forma que la estimación del EKF cada tiempo de muestreo corrija las medidas del LIDAR tomadas en dicho tiempo de muestreo para que transcurridos los 144 milisegundos en los que opera el bloque de preprocesado se minimicen las inconsistencias del mapa asociadas al movimiento del coche durante esos 144 milisegundos. A pesar de la baja velocidad del vehículo (aproximadamente de 300 mm/s), durante este tiempo se produce un desplazamiento de aproximadamente 40 milímetros, que puede resultar un error moderado en este caso; pero que implicaría un error muy grande en caso de que se implantara este algoritmo en vehículos más rápidos. Algo similar ocurriría con la variación del ángulo de guiñada. Para corregir este error es necesario que el EKF también se emplee para corregir las medidas del LIDAR. Para realizar esta corrección es necesario que cada período de muestreo se realice la corrección de la ecuación 4.14, en la cual se utilizan los incrementos de las variables de estado durante el tiempo que tarda el LIDAR en dar una vuelta completa para estimar cómo sería el mapa resultante si se tomase desde una cierta posición con el vehículo estático. Y es con este mapa estimado, que posteriormente se realiza la estimación por mínimos cuadrados. Así pues, con esta corrección se mejora la precisión práctica del LIDAR a la hora de mapear, pero lo cierto es que con esta corrección se incurre en el mismo error que la ha motivado, con la salvedad de que en este caso el error cometido es mucho menor porque el tiempo de desplazamiento del vehículo es de solo 8 milisegundos. Durante este periodo el máximo desplazamiento es de 2,4 milímetros y la máxima rotación de 0,96 grados, por lo que dada la precisión manejada (el LIDAR mide con resolución de un milímetro) este error sí que debería resultar despreciable para las estimaciones. No obstante, de no ser así, corregir esta variación es bastante sencillo pues simplemente habría que interpolar linealmente las variables de estado durante cada medida del LIDAR (esto es, cada 0,25 milisegundos).

$$\left[\widehat{\overrightarrow{r_{b_i}}} \right] = \left[\overrightarrow{r_{b_i}} + \overrightarrow{r_{lidar}} \quad 1 \right] \begin{bmatrix} e^{j\Delta\psi_i} \\ \Delta x_i + j\Delta y_i \end{bmatrix} \tag{4.14}$$

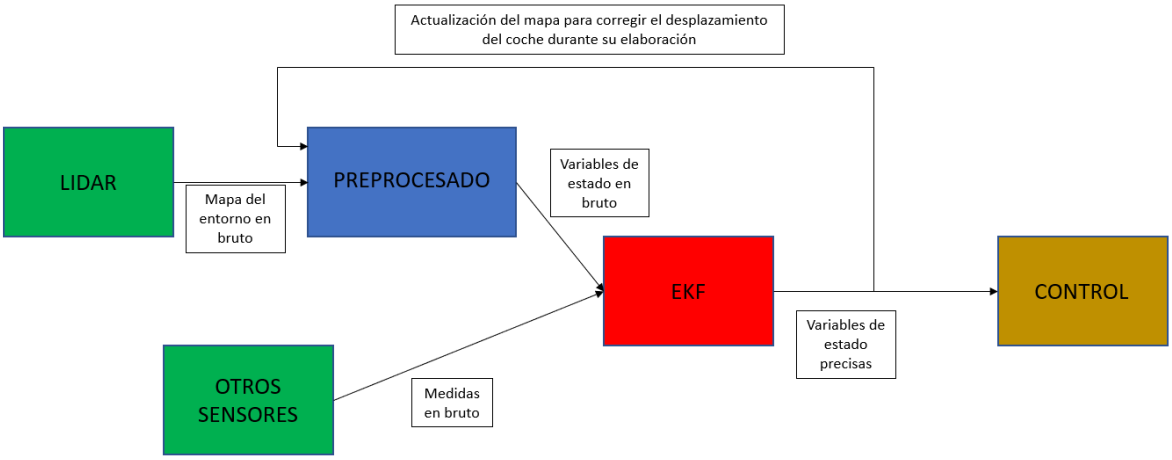


Figura 4.6. Esquema de funcionamiento del algoritmo de preprocesado en conjunto con el EKF.

5

Resultados

La mayoría de resultados de este proyecto consisten en la integración satisfactoria de todos los elementos mencionados en los capítulos 3 y 4, tal y como se describe en dichos capítulos; y estos resultados no se cuantifican más allá del hecho de que el funcionamiento es correcto. Sin embargo, sí existen resultados que requieren de un análisis más complejo, y son estos los que se explican en este apartado. Estos son las mediciones de los encoders y el LIDAR, haciendo especial hincapié en las segundas; y los resultados del bloque de preprocesado que permite su integración en el EKF.

5.1. Mediciones de los sensores

5.1.1. Mediciones de los encoders

Se muestra en primer lugar un ejemplo de las medidas de velocidad instantánea de los encoders sin aplicar ningún tipo de filtro y a velocidad constante. Aunque se aprecia una tendencia más o menos constante es claro que la señal tiene mucho ruido, por lo que se le aplican dos tipos distintos de filtros: filtros de media móvil y filtros de mediana móvil [30]. Se evalúan ambos tipos y sus diferencias para distintos órdenes de los filtros.

Primeramente se muestran comparativas de filtros de órdenes 5 (5.1), 10 (5.2) y 20 (5.3) tanto de la media (rojo) como de la mediana (verde) con la señal original (negro). En dichas comparaciones se observa como los filtros de la media proporcionan señales más puntiagudas, en general más similares que las señales resultado del filtro de la mediana. Estas son totalmente escalonadas, puesto que los valores de dicha señal se corresponden con valores de la señal original y, por tanto, está sujeta a la misma discretización. Por otra parte, se aprecia como dichos escalones amortiguan los picos, especialmente los más pronunciados como puede ser el del transitorio inicial.

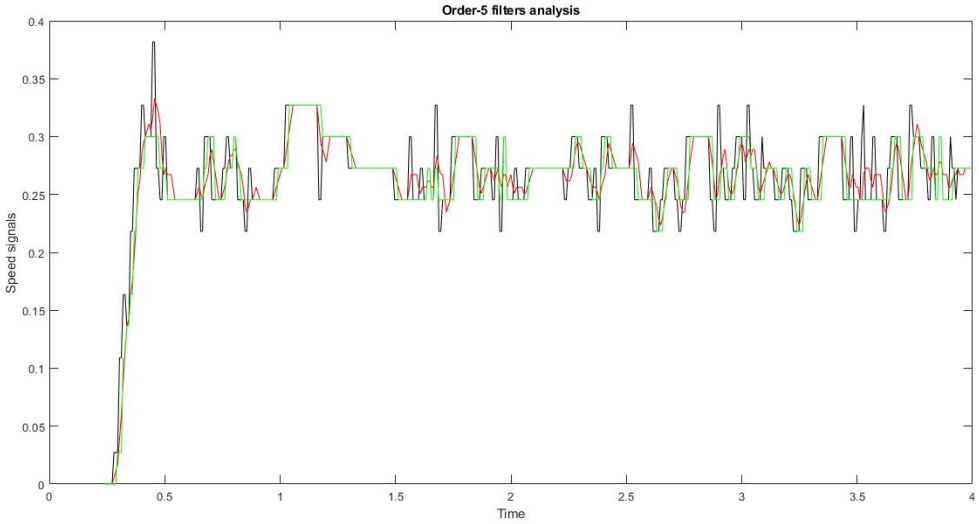


Figura 5.1. Comparación de la señal original con las señales filtradas. Orden de los filtros: 5.

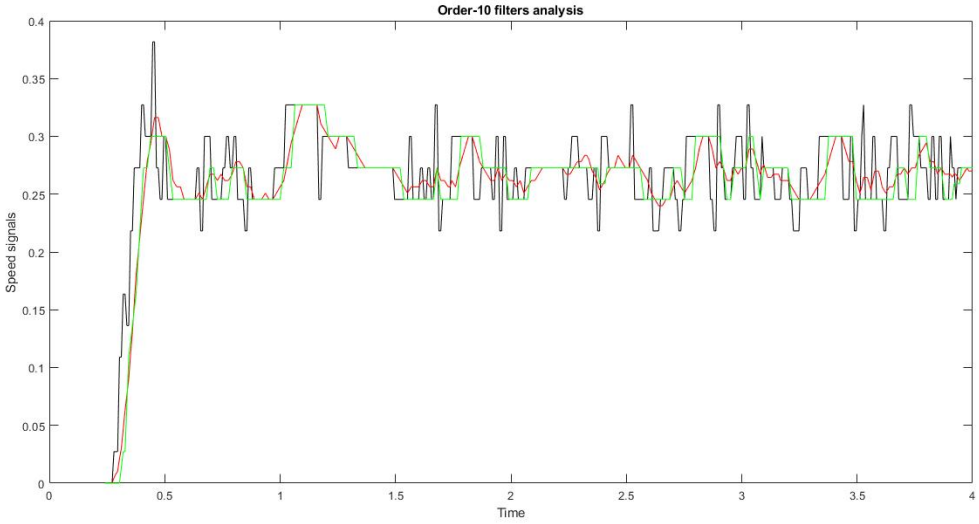


Figura 5.2. Comparación de la señal original con las señales filtradas. Orden de los filtros: 10.

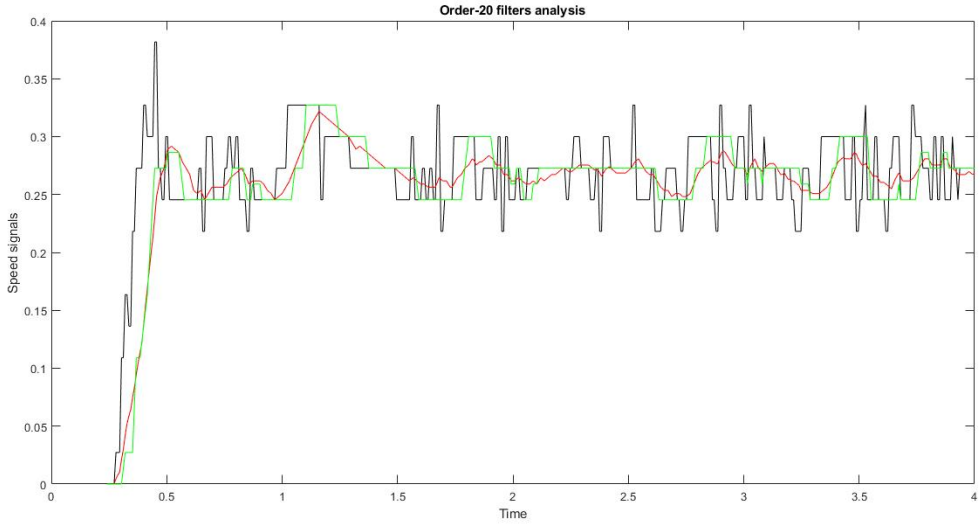


Figura 5.3. Comparación de la señal original con las señales filtradas. Orden de los filtros: 20.

Comparando ahora las señales filtradas con el mismo filtro pero de distinto orden se observa claramente como según aumenta el orden mayor es el amortiguamiento en los transitorios pero más lentos son estos. Observando por ejemplo la imagen 5.4, se ve como la señal resultante de aplicar el filtro de orden 20 tiene un retraso notable respecto a las otras dos. Este fenómeno es más visible en el transitorio inicial, donde queda claro qué señal corresponde a qué filtro observando el retardo y amortiguamiento de cada una. Se ve además, como al aumentar el orden la señal parece menos ruidosa y mantiene valores más constantes. Si se observa la imagen 5.5 en el intervalo entre los segundos 2 y 2,5 este fenómeno es bastante claro.

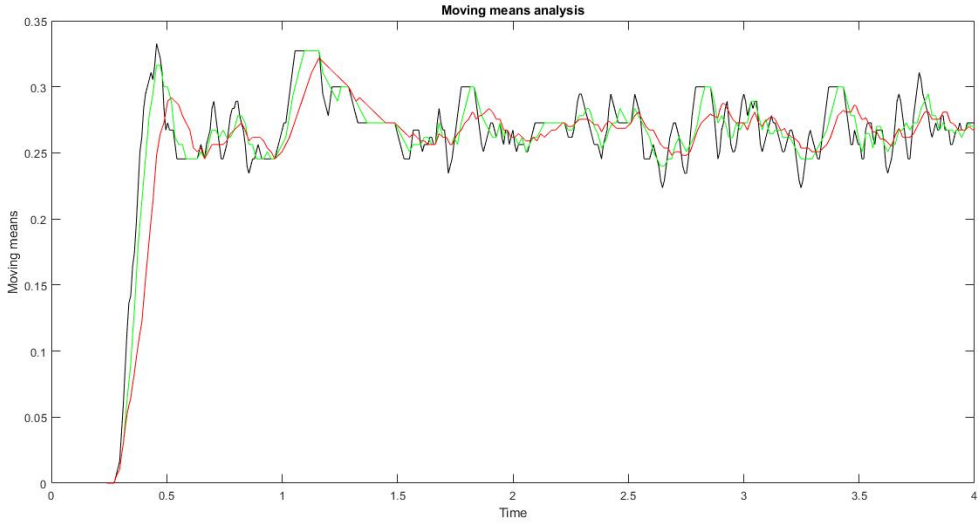


Figura 5.4. Comparación de filtros de la media de distintos órdenes.

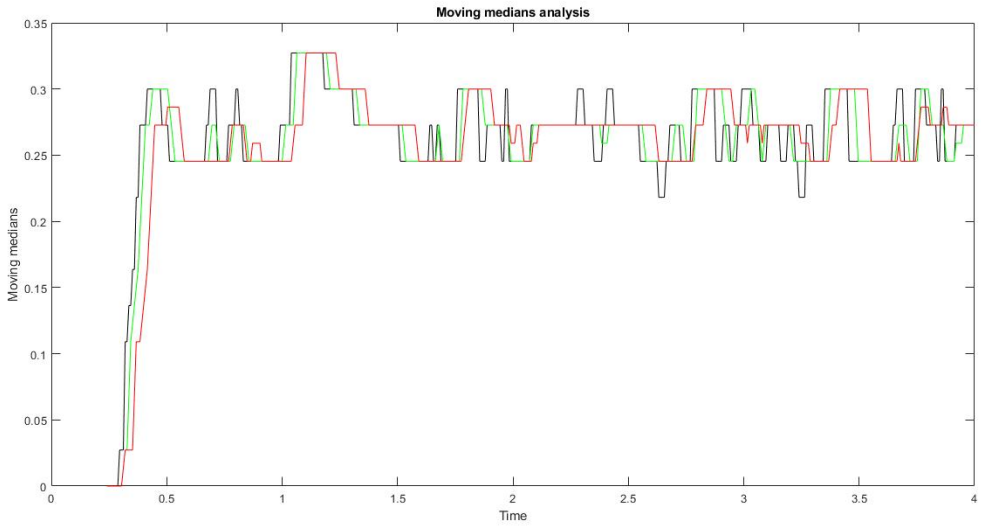


Figura 5.5. Comparación de filtros de la mediana de distintos órdenes.

Adicionalmente a este análisis, hay que considerar el hecho de que el filtro de la mediana es más costoso computacionalmente, pues calcular una media es más sencillo que ordenar una serie de valores [30]. Aun así, se ha creído que para este proyecto el filtro de la mediana da mejores resultados, pues ni el problema de la discretización ni el de la carga computacional adicional son excesivamente graves (los algoritmos de ordenación más eficientes son O(n log(n)) [35], que para n=10 no es un problema), y este da mejores resultados en la amortiguación de los grandes transitorios. Para no tener demasiado retardo en la señal se ha elegido el filtro de orden 10, pues el retardo es asumible (menos de 100 milisegundos), a cambio de una señal mucho más limpia y manejable.

5.1.2. Mediciones del LIDAR

Se muestra primeramente la precisión del LIDAR (obtenida de la información del fabricante). Viendo lo bajo del porcentaje de error (menos de un 0,2 %, imagen 5.6), no parece necesario probar a medir dicho error, entre otras cosas porque con distancias relativamente pequeñas ni siquiera llega al milímetro de resolución que tiene la medida, y aun con distancias grandes esto nunca va a ser una fuente de error significativa.

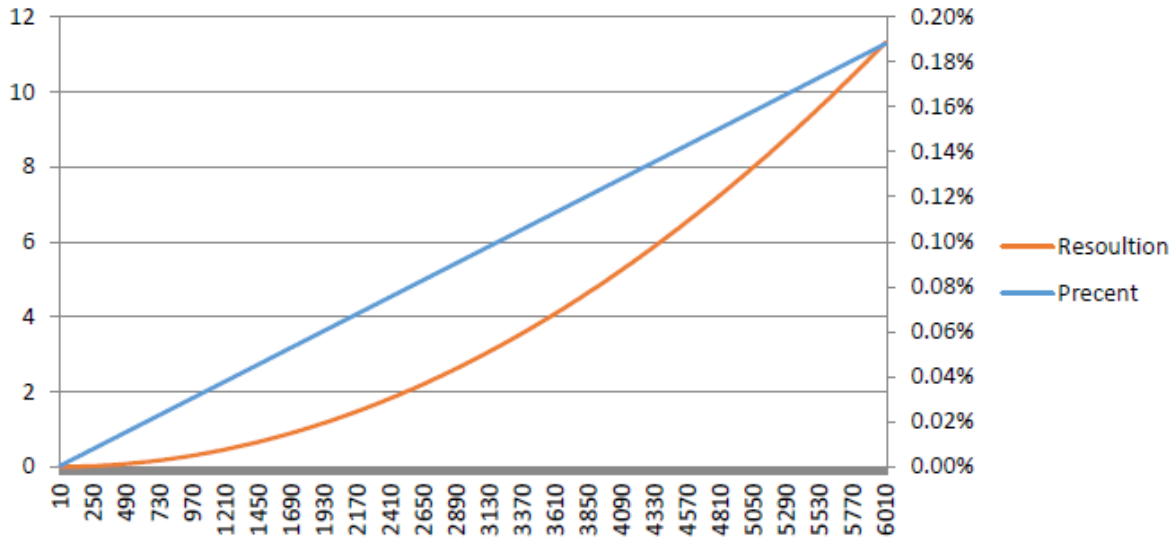
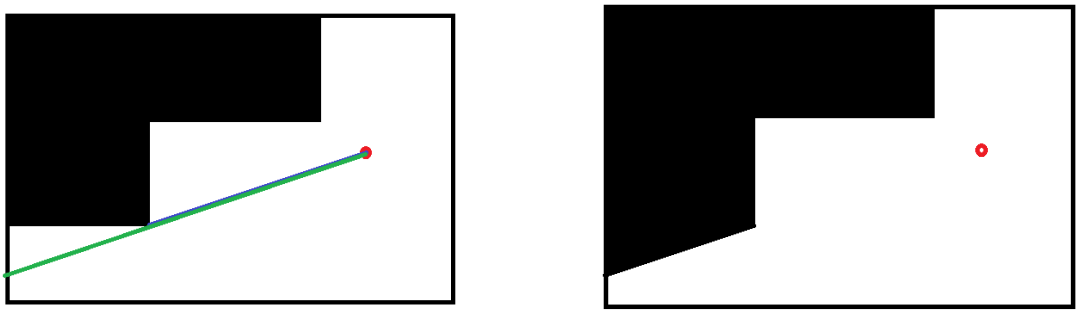


Figura 5.6

Lo que sí es interesante es que dichos valores de error están medidos para objetos blancos, lisos y con una reflectividad concreta (del 70 %). Además, si el entorno no es convexo la imagen se linealiza hasta hacerse convexa (no se puede representar lo que no se puede ver), de forma que, por ejemplo, las esquinas interiores que no se ven directamente no parecen tales. Esto se ejemplifica en las imágenes 5.7a y 5.7b.



(a) Mapeado del LIDAR en el mapa real

(b) Mapa visto por el LIDAR

Figura 5.7

Se analiza en primer lugar un mapa realizado midiendo una semicircunferencia de material similar al empleado para caracterizar la resolución del LIDAR. Este mapa, mostrado en la imagen 5.8 (la imagen se ha recortado y ampliado para que se aprecie mejor la coincidencia entre el mapa y la realidad), coincide con bastante exactitud con el original teniendo en cuenta únicamente a la semicircunferencia y las paredes tangentes a esta, pues en la parte de la derecha no había ninguna pared y por ello las medidas son incorrectas o corresponden a valores muy altos irreales. Esta imagen refleja la idea en la que se ha insistido en todo el documento de que el LIDAR es un sensor muy potente y preciso, puesto que aun eligiendo un contorno más

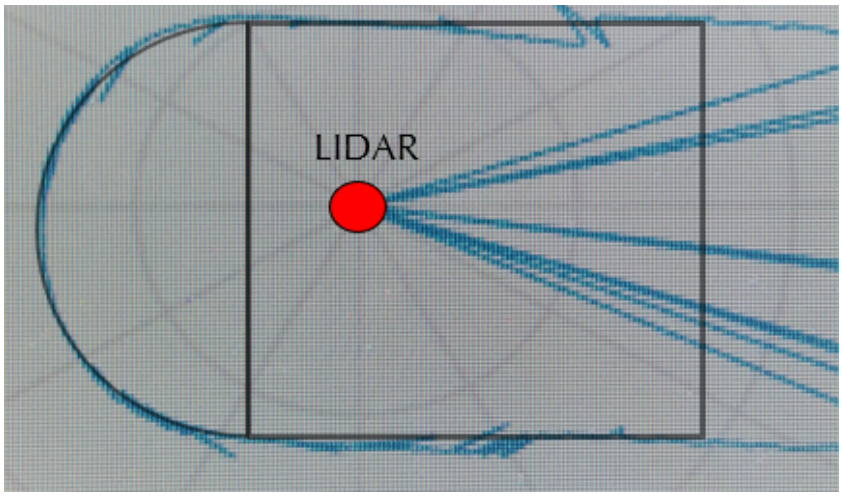


Figura 5.8. Mapeado de un entorno con una semicircunferencia.

complejo que una simple habitación rectangular el mapa elaborado es de mucha calidad. Es por esto que se ha creído más interesante poner a prueba este sensor utilizando objetos cotidianos presentes en una vivienda.

A continuación, se muestran varios mapeos del LIDAR cuando el entorno es menos ”ideal” que una pared blanca y lisa. En la imagen 5.9 cabe destacar en primer lugar que a pesar de que las distancias son pequeñas la medida parece muy ruidosa, lo cual impacta teniendo en cuenta que el error teórico en ese intervalo de distancias es inferior a un milímetro. Olvidando por el momento la ”pared” situada a 0 grados, las otras tres ”paredes” son en este caso dos paredes rugosas color salmón que hacen esquina y una lámina ondulada de plástico color morado. En las partes redondeadas (en negro las paredes y en azul la lámina de plástico) se observa como, cuando la pared es aproximadamente perpendicular al haz del LIDAR, el ruido asociado a las irregularidades de la pared se magnifica (aun cuando en este intervalo las distancias son menores). Observando ahora la cuarta ”pared”, que sin duda parece la más deformada de todas, se ven varios puntos cuya medida es 0 (lo que se corresponde con un error de medida). Esto es más curioso todavía, si se tiene en cuenta que esta ”pared” está compuesta íntegramente de cartón relativamente liso. ¿Dónde está la trampa entonces? Pues que en realidad dicha pared no es ni mucho menos una pared continua y lisa, sino que está formada por cajas de cartón que hacen esquinas exteriores e interiores y huecos de muy pequeño tamaño, pero suficiente como para corresponderse con una medida del LIDAR. Un ejemplo claro de esto se puede ver en las medidas correspondientes con el entorno de 0 grados de la imagen, donde se ven varias medidas erróneas y una discontinuidad en la pared correspondiente a una linealización como la explica en el ejemplo anterior.

Después de analizar diferentes rugosidades, se cuestiona ahora la precisión del LIDAR cuando se modifica la reflectividad. Se sustituye la lámina ondulada de plástico por una lámina de cobre 5.10. Aunque esta superficie es perfectamente lisa se observa como el error, aunque menor, sigue siendo elevado. Esto se debe a que las altas reflectividades perjudican la triangulación del LIDAR. Es de esperar que dicho fenómeno sea más grave todavía ante superficies metálicas pulidas.

Por último, se muestra un ejemplo de el mismo recinto que en el primer caso (aunque debido a las medidas de mayor distancia está escalado) con una pequeña modificación en la figura 5.11. Observando la imagen la intuición dicta que lo que se ha hecho es un ”agujero” en la pared, pero

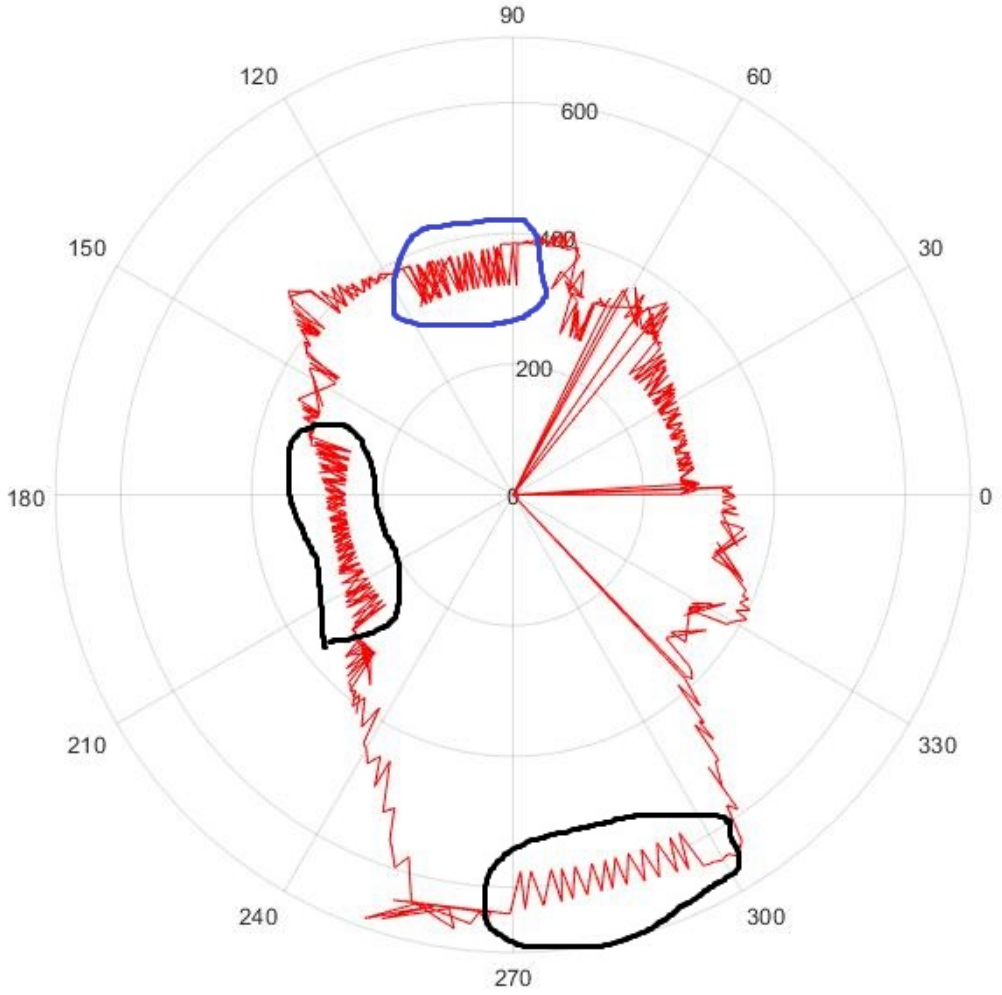


Figura 5.9. Mapeado de un entorno irregular.

en realidad lo que se ha hecho es situar un objeto entre el LIDAR y la pared correspondiente. La trampa en este caso está en el objeto: un espejo. Así, reflejando casi íntegramente los haces de luz, las superficies especulares imposibilitan la triangulación del LIDAR.

5.2. Bloque de preprocesado

Se muestran resultados simulados del bloque de preprocesado descrito en el capítulo 4. Para generar los mapas necesarios para realizar el preprocesado se ha tomado como entorno una circunferencia de radio 800 milímetros y se ha añadido a dicho mapa un ruido uniforme que acota superiormente el error del LIDAR tanto en lo referente a distancia como en lo referente al ángulo (los datos del error máximo del sensor se han tomado directamente de la información del fabricante). El valor de mando común empleado es del 100 % y el del mando diferencial del 25 %. La dirección de desplazamiento y el sentido de giro se han elegido arbitrariamente (en este caso, sentido de giro positivo y dirección de desplazamiento de 30 grados). Con estos valores los datos de posición y orientación finales del vehículo son:

- Desplazamiento: 4,288 milímetros

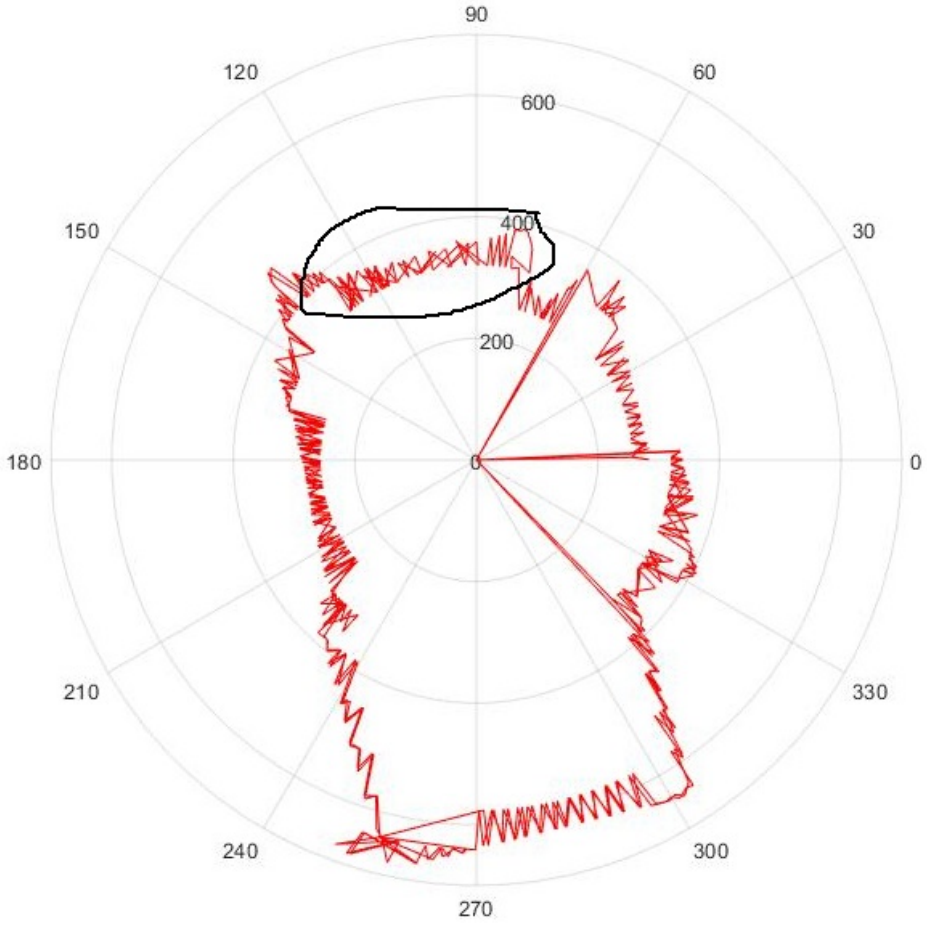


Figura 5.10. Ejemplo de los efectos de la alta reflectividad.

- Ángulo del desplazamiento: 30 grados
- Ángulo de guiñada: 4,285 grados

A continuación, se muestran varios casos donde se comparan estos valores teóricos con los estimados por el bloque de preprocesado (tablas 5.1-5.6) y se calcula el error en la estimación.

5.2.1. Sin corrección del error

Primeramente se muestran los resultados del mapa y la estimación cuando se ignora el error asociado al desplazamiento y giro del vehículo. Se observa en la imagen 5.12 cómo debido al desplazamiento el ”círculo” ni siquiera parece cerrarse. De hecho, observándolo cuidadosamente es posible distinguir la dirección y el sentido de giro del vehículo, aun cuando la variación temporal es de solo 144 milisegundos. En la tabla 5.1 se comprueba como el error cometido en el desplazamiento lineal es muy grande (más del 100 % tanto en módulo como en ángulo), lo que hace imprescindible una corrección con el EKF. Es interesante también observar que el error en la estimación de la guiñada es nulo, mientras que en el módulo sí que existe cierto error, aunque no demasiado significativo.

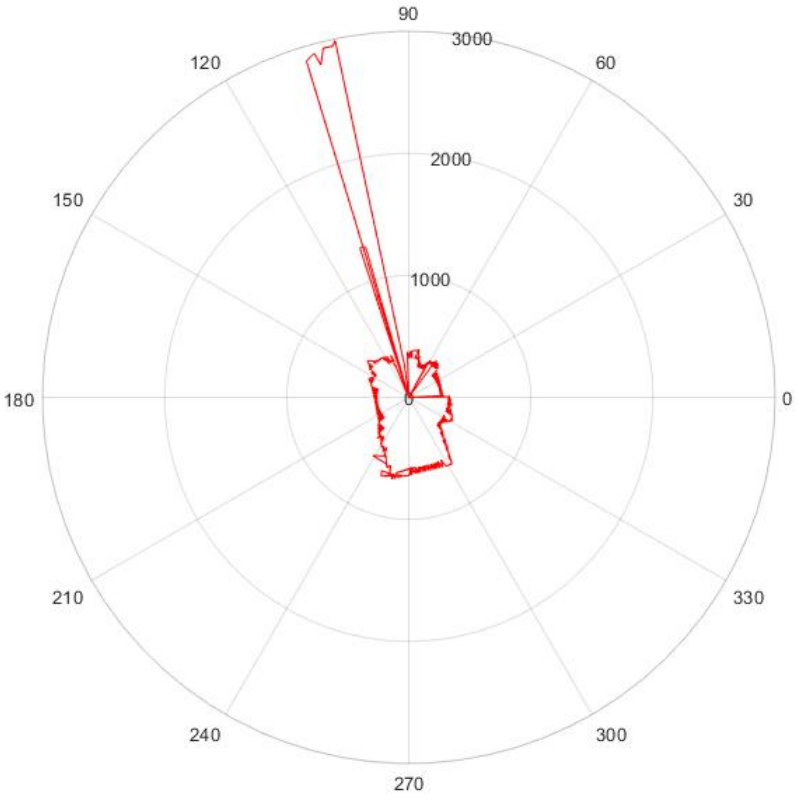


Figura 5.11. Ejemplo del efecto de la reflectividad total.

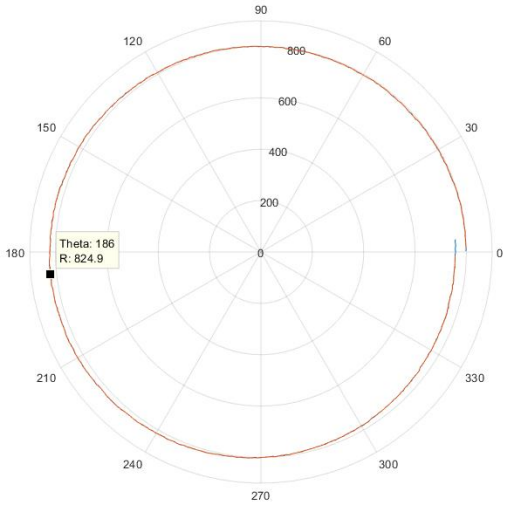


Figura 5.12. Sin corrección del error. Resolución del mapa: 1 grado.

	Valor real	Valor estimado	Error(%)
Módulo del desplazamiento (mm)	4,288	9,307	117,05
Ángulo del desplazamiento (grados)	30	67,67	125,57
Ángulo de rotación (grados)	4,285	4,285	0
Módulo del vector rotación	1	0,9721	2,79

Tabla 5.1. Sin corrección del error. Resolución del mapa: 1 grado.

5.2.2. Aplicando corrección al error

Tras demostrar que es imprescindible corregir los mapas del LIDAR considerando el desplazamiento del vehículo, se muestra el funcionamiento del algoritmo modelando el error asociado a la corrección del desplazamiento del vehículo como un ruido uniforme de cota superior el 10 % de los desplazamientos lineal y angular respectivamente. La corrección se realiza uniformemente sobre todas las medidas de un periodo de muestreo. Se muestran imágenes de los mapas medidos, además de los valores de las variables de estado originales usados para generarlos y su estimación con el bloque de preprocesado. El algoritmo se evalúa para distintos valores de resolución de los mapas, que como ya se mencionó es el valor que habría que modificar en función de la capacidad de computación disponible para los cálculos.

Se muestra primero el caso más favorable considerado, en el cual los cálculos se realizarían con una resolución de 1 grado. En principio este valor no puede reducirse mucho más, pues el propio LIDAR toma cada vuelta 576 muestras, lo que supone que como mínimo los cálculos deberían realizarse con mapas de 0,625 grados de resolución. En cualquier caso, los resultados mostrados para esta resolución son bastante prometedores, pues los errores se mantienen muy bajos y de no acumularse darían al vehículo una precisión de milímetros como mínimo.

	Valor real	Valor estimado	Error(%)
Módulo del desplazamiento (mm)	4,288	4,284	0,09
Ángulo del desplazamiento (grados)	30	28,710	4,30
Ángulo de rotación (grados)	4,285	4,285	0,00
Módulo del vector rotación	1	1,000	0,00

Tabla 5.2. Aplicando corrección al error. Resolución del mapa: 1 grado.

A continuación se muestran los valores estimados para las resoluciones de 2 y 5 grados. En estos casos comienza a apreciarse un aumento del error, aunque este se mantiene dentro de lo razonable y pueden resultar interesantes cuando la capacidad de computación sea limitada, pues la estimación por mínimos cuadrados es la parte más ”pesada” del algoritmo de preprocesado y reducir el número de valores entre 2 o entre 5 resultaría muy beneficioso.

	Valor real	Valor estimado	Error(%)
Módulo del desplazamiento (mm)	4,288	4,175	2,64
Ángulo del desplazamiento (grados)	30	28,410	5,30
Ángulo de rotación (grados)	4,285	4,285	0,00
Módulo del vector rotación	1	0,999	0,10

Tabla 5.3. Aplicando corrección al error. Resolución del mapa: 2 grados.

	Valor real	Valor estimado	Error(%)
Módulo del desplazamiento (mm)	4,288	4,610	7,51
Ángulo del desplazamiento (grados)	30	32,290	7,63
Ángulo de rotación (grados)	4,285	4,285	0,00
Módulo del vector rotación	1	0,999	0,10

Tabla 5.4. Aplicando corrección al error. Resolución del mapa: 5 grados.

Por último, los datos para las resoluciones de 10 y 20 grados son menos prometedores: los errores aumentan considerablemente tanto en el módulo como en el ángulo del desplazamiento, aunque sorprendentemente el error sigue siendo nulo para el ángulo de rotación y muy bajo para el módulo del vector de rotación. Esto se debe a que debido a la distribución uniforme del error y a la elevada cantidad de puntos, el promedio de todos los errores se mantiene muy bajo (nulo de hecho si se mide respecto de la precisión de empleada); y en el cálculo de mínimos cuadrados es este valor el que determina el ángulo del vector de rotación.

	Valor real	Valor estimado	Error(%)
Módulo del desplazamiento (mm)	4,288	5,078	18,42
Ángulo del desplazamiento (grados)	30	37,690	25,63
Ángulo de rotación (grados)	4,285	4,285	0,00
Módulo del vector rotación	1	0,999	0,10

Tabla 5.5. Aplicando corrección al error. Resolución del mapa: 10 grados.

	Valor real	Valor estimado	Error(%)
Módulo del desplazamiento (mm)	4,288	6,171	43,91
Ángulo del desplazamiento (grados)	30	21,750	27,50
Ángulo de rotación (grados)	4,285	4,285	0,00
Módulo del vector rotación	1	0,998	0,20

Tabla 5.6. Aplicando corrección al error. Resolución del mapa: 20 grados.

Se analizan ahora los mapas de medidas. En azul se representa el mapa medido por el LIDAR y en rojo el resultado de la interpolación. Es interesante comprobar cómo empleando

una resolución de solo 5 grados (que antes se ha definido como el límite de lo utilizable por el algoritmo) el ojo humano es casi incapaz de distinguir la poligonalidad del contorno interpolado. En el caso donde la resolución es menor sí se aprecia con más claridad el resultado de la interpolación.

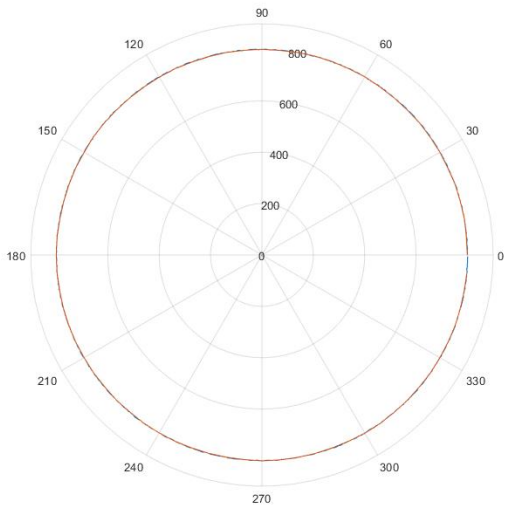


Figura 5.13. Resolución del mapa: 5 grados

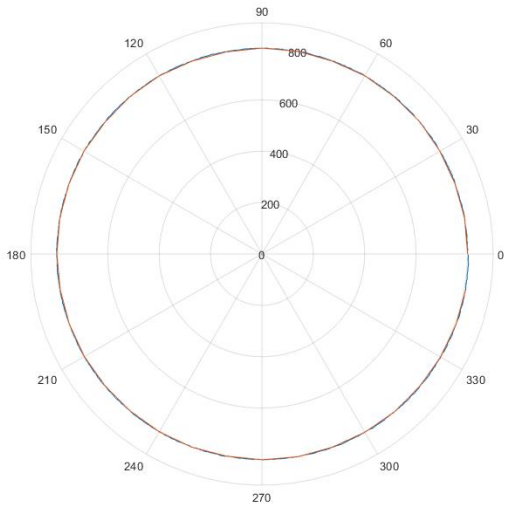


Figura 5.14. Resolución del mapa: 10 grados

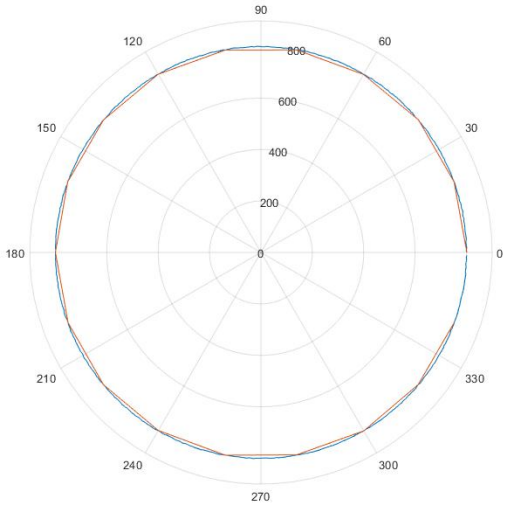


Figura 5.15. Resolución del mapa: 20 grados

6

Conclusiones

6.1. Conclusiones

Durante el desarrollo del proyecto se han cumplido satisfactoriamente los siguientes objetivos:

1. Diseño optimizado del vehículo.
2. Montaje e integración física de todos los elementos hardware del vehículo.
3. Desarrollo del software de control del LIDAR.
    a) Desarrollo de la máquina de estados del conjunto LIDAR-Arduino-Raspberry.
    b) Generación del PWM necesario para hacer girar el LIDAR.
    c) Interpretación de la información transmitida por el LIDAR.
4. Adaptación del control de los drivers y los cálculos de los encoders a vehículos terrestres.
5. Integración del software de control de todos los sensores y actuadores.
6. Desarrollo de la máquina de estados del vehículo.
7. Adaptación del software de comunicación basado en el protocolo MAVLINK implementado en UDP.
8. Implementación de un filtrado para las medidas de los encoders.
9. Integración de las medidas de los encoders en el EKF.
10. Integración de las medidas del LIDAR en el EKF.
    a) Desarrollo de las ecuaciones que definen el movimiento del vehículo y su relación con las medidas del LIDAR.
    b) Desarrollo del algoritmo de preprocesado.
    c) Optimización del algoritmo corrigiendo dinámicamente las medidas del LIDAR.

La parte fundamental de este proyecto ha sido todo lo relacionado con el LIDAR. Primeramente, conseguir generar el PWM para que gire y enviar los comandos de la forma exacta que el LIDAR requiere para recibir la información de las medidas, para lo que ha sido necesario comunicar y coordinar el LIDAR tanto con la Raspberry como con un Arduino adicional debido a las dificultades encontradas. Una vez conseguido el control del sensor, ha sido necesario desarrollar un algoritmo especial para preprocesar sus medidas, pues integrarlas directamente en el EKF era demasiado costoso computacionalmente. Para ello ha sido necesario desarrollar las ecuaciones que relacionan la evolución de las medidas del LIDAR en el tiempo con el movimiento del vehículo y utilizar dichas ecuaciones para estimar la posición y orientación del vehículo solamente con las medidas del LIDAR empleando el método de mínimos cuadrados. Para mejorar la precisión de esta estimación se ha utilizado la información del EKF para corregir dinámicamente estas medidas, de forma que ambos procesos se retroalimentan mutuamente y se consigue así una mayor calidad en las medidas.

Así pues, la principal aportación de este proyecto es la posibilidad de usar el LIDAR e integrarlo en un EKF para cualquier vehículo terrestre o incluso integrarlo en un UAV con algunas modificaciones. Además, hay algunas aportaciones secundarias como la adaptación del control de los drivers a vehículos terrestres y la mejora de las medidas de los encoders para vehículos de cuatro ruedas.

En cuanto a los elementos elegidos para el montaje del vehículo en materia tanto de sensores y actuadores como en lo referente a la estructura es importante destacar que aunque se han probado y desechado diferentes modelos (estas elecciones se hayan debidamente justificadas en el capítulo 3), existen multitud de opciones perfectamente válidas en distintos rangos de presupuesto. Por último, decir que desde el principio se ha buscado diseñar un vehículo con la capacidad de implementar algoritmos de navegación autónoma avanzada, y con la sensorización y capacidad de procesamiento que se ha implementado en la versión final del vehículo, esto es más que posible.

6.2. Futuros Desarrollos

En este apartado se pretende dar algunas ideas y orientaciones acerca de posibles mejoras que se le pueden hacer al vehículo, para lo cuál es necesario hacer primeramente un análisis crítico de la versión final. Las mejoras presentadas a continuación engloban todos los aspectos del vehículo, desde la estructura al hardware embarcado, y por supuesto algunas opciones de software que se pueden implementar para la navegación avanzada sin necesidad de modificar el diseño del vehículo.

6.2.1. Mejoras de la estructura

La estructura es probablemente la parte más sencilla de mejorar del vehículo, pues el diseño y desarrollo de esta no era más que una necesidad de cara a implementar el resto de partes. Las mejoras que se proponen se centran en dos aspectos: el chasis y las ruedas, y la geometría de giro.

Chasis y ruedas

El chasis empleado es bastante rudimentario, pues consiste simplemente en dos placas de fibra de vidrio reforzada separadas por soportes metálicos. Lo ideal sería montar una estructura cerrada, en la que se encuentren todos los componentes excepto el LIDAR (y un hipotético

sensor de proximidad). De esta manera el vehículo podría ser más compacto y pequeño. Esta estructura sería idealmente de un material ligero pero resistente, como podría ser la propia fibra de vidrio o fibra de carbono. En cuanto a las ruedas, estas son bastante grandes, y de conseguir reducir el tamaño de la estructura habría que sustituirlas por unas más pequeñas y preferiblemente con la banda de rodadura de caucho o similar.

Geometría de giro

El modo que tiene de girar el vehículo es mediante la aplicación de tensión diferencial a los motores, lo cual simplifica bastante el diseño del software de control, pero limita a la mitad la velocidad del vehículo, entre otros inconvenientes. Existen varias geometrías más complejas que aportan ventajas al control del vehículo. La geometría de doble pivote o la geometría de Ackermann son quizá las opciones más lógicas, pero existen multitud de configuraciones con mucho potencial como pueden ser aquellas que emplean ruedas suecas u omnidireccionales [36, 37].

6.2.2. Mejoras del hardware

El hardware es la parte más trabajada del proyecto, y aunque con más presupuesto es posible mejorar significativamente cualquier elemento, lo cierto es que en la línea del proyecto solamente se proponen tres mejoras: la inclusión de un sensor de proximidad o sónar para medir distancias cortas y medias al entorno y complementar al LIDAR; la inclusión de una o varias cámaras embarcadas en el vehículo, también como complemento al LIDAR aunque sin duda requieren de un software adicional como se explica en la siguiente sección; y en caso de que el software así lo requiera, una mejora de la capacidad de procesamiento, ya sea incluyendo un microprocesador Intel, o sustituyendo el Arduino Nano por algo más potente y derivando parte de la carga computacional a ese elemento (como podría ser el caso del preprocesado del LIDAR, por ejemplo).

6.2.3. Modificaciones en el software

Se explican en este apartado las 4 mejoras que se consideran más interesantes a nivel de software: la modificación del tipo de control utilizado, el uso de un segundo LIDAR para poder trabajar en 3 dimensiones, la utilización de un filtro de partículas y el empleo de técnicas de *machine learning*.

Modificación del tipo de control

Los controles ideales para un vehículo de estas características son o bien un control predictivo o bien un control adaptativo. Es por ello, que cambiar el algoritmo de control empleado parece una continuación bastante lógica. Evidentemente esto complica bastante el diseño del control, más aun si a esto se le añade una modificación del método de giro, pero estos controles presentan bastantes ventajas y mejorarían notablemente las estimaciones del EKF. Además, en el caso del control predictivo, también se mejoraría el rendimiento de los algoritmos de predicción orientados a la localización del vehículo en entornos más complejos (no convexos), o incluso de tipo laberíntico.

Uso de otro LIDAR

Utilizando un segundo LIDAR es posible estimar y mapear por una parte las tres dimensiones del plano (2 desplazamientos y un ángulo de orientación) ya cubiertas en el proyecto y, por

otra, estimar la altitud y los dos ángulos de orientación restantes, de forma que el vehículo podría desplazarse en entornos tridimensionales con rampas y desniveles y aun así ser capaz de operar con normalidad.

Filtro de partículas

Este es un método de estimación empleado habitualmente en algoritmos de visión por ordenador, que basándose en principios estocásticos estima el estado de un elemento en un determinado instante cuando este varía en el tiempo. Consiste en un tipo de filtro similar en cierto modo al EKF, que empleando algoritmos de Monte Carlo [38] e inferencia estadística Bayesiana consigue predecir el comportamiento de un sistema dinámico.

Machine learning

Esta es posiblemente la mejora más interesante y prometedora de todas las que se proponen, y engloba muchas opciones distintas: redes neuronales, algoritmos genéticos, árboles de decisión, etc [39]. Es por ello que se propone un ejemplo concreto, aunque existan infinitud de alternativas igualmente válidas. Este ejemplo consiste en el empleo de aprendizaje profundo con redes neuronales. Para aportar la información necesaria para el refuerzo se propone la utilización de un sistema de cámaras externas al vehículo. Este tipo de aprendizaje es el que ha implementado *Google* para desarrollar sus algoritmos campeones del mundo en diversos juegos como el Go, el Shogi o el ajedrez.

A

Presupuesto

A.1. Sumas parciales

A.1.1. Estructura del vehículo

Tabla A.1. Sumas parciales: elementos de la estructura del vehículo

Componente	Cantidad	Precio Unitario (€)	Coste Parcial (€)
Placa del chasis	2	11	22
Ruedas	4	10	40
Soportes de las ruedas	4	3,5	14
		TOTAL	76

A.1.2. Componentes hardware

Tabla A.2. Sumas parciales: componentes hardware

Componente	Cantidad	Precio Unitario (€)	Coste Parcial (€)
Raspberry Pi 3B	1	39	39
Arduino Nano	1	20	20
RPLIDAR A2M4	1	450	450
IMU MPU9250	1	11	11
Motor EMG30	4	35	140
Driver MD25	2	50	100
Pi 3 Click Shield	1	12	12
Pi 3 Hat	1	9	9
Regulador a 5 voltios	1	9	9
Conversor de tensión	1	5	5
Batería LiPo 3 celdas 1500 mAh 40C	1	13	13
		TOTAL	808

A.1.3. Software

Tabla A.3. Sumas parciales: programas utilizados

Programa	Cantidad	Precio Unitario (€)	Coste Parcial (€)
MATLAB y Simulink versión estudiante	1	125	125
TeXmaker	1	0*	0*
JabRef	1	0*	0*
PUTTY	1	0*	0*
Device Monitoring Studio	1	0*	0*
		TOTAL	125

**Software gratuito.*

A.1.4. Herramientas y equipos

Tabla A.4. Sumas parciales:herramientas y equipos empleados

Elemento	Cantidad	Precio Unitario (€)	Coste Parcial (€)
Ordenador	1	750	750
Osciloscopio 2 canales	1	450	450
Taladro	1	150	150
Polímetro	1	18	18
Otras herramientas	1	45	45
		TOTAL	1413

A.1.5. Mano de obra directa

Tabla A.5. Sumas parciales: mano de obra directa

Tarea	Horas	Precio Unitario (€/hora)	Coste Parcial (€)
Estado del arte	12	35	420
Integración del hardware	250	65	16250
Montaje del hardware y la estructura	30	20	600
Desarrollo del software	350	65	22750
Documentación	80	20	1600
		TOTAL	41620

A.2. Presupuesto general

Tabla A.6. Presupuesto general

Elemento	Suma Parcial
Estructura del vehículo	76
Componentes hardware	808
Programas software	125
Herramientas y equipos	1413
Mano de obra directa	41620
TOTAL	44042

Bibliografía

[1] H. Yin, F. Yang, and J. Li, "Multi-objective optimization design of a carbon fiber reinforced composite upper arm," 2017.

[2] F. Jiang, J. Zhao, A. K. Kota, N. Xi, M. W. Mutka, and L. Xiao, "A Miniature Water Surface Jumping Robot," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1272 – 1279, 2017.

[3] S. yeol Yoo, B.-H. Jun, H. Shim, P.-M. Lee, and B. Kim, "Finite element analysis of carbon fiber reinforced plastic body frame for seabed robot, Crabster200," 2013.

[4] J. Franch and J. Rodriguez-Fortun, "Control and trajectory generation of an ackerman vehicle by dynamic linearization," 2009.

[5] W. Ren, Q. Gu, D. xin He, and J. Zhao, "A car-like mobile robot design and implementation," 2012.

[6] G. Slemon and A. Straughen, *Electric Machines*. Addison-Wesley, 1980.

[7] J. Gonçalves, J. Lima, P. J. Costa, and A. P. Moreira, "Modeling and simulation of the emg30 geared motor with encoder resorting to simtwo: The official robot factory simulator," July 2013.

[8] M. Stanley, J. Lee, and A. Spanias, *Sensor Analysis for the Internet of Things*. Morgan & Claypool, 2018.

[9] H. Chao and Y. Chen, *Attitude Estimation Using Low-Cost IMUs for Small Unmanned Aerial Vehicles*. IEEE, 2012.

[10] J. R. Brauer, *Hall Effect and Magnetoresistive Sensors*. IEEE, 2006.

[11] Z. Wang and L. Wu, "Automated extraction of building geometric features from raw lidar data," in *2009 IEEE International Geoscience and Remote Sensing Symposium*, vol. 2, pp. II–436–II–439, July 2009.

[12] S. Verghese, "Self-driving cars and lidar," in *2017 Conference on Lasers and Electro-Optics (CLEO)*, pp. 1–1, May 2017.

[13] M. Aldibaja, N. Suganuma, and K. Yoneda, "LIDAR-data accumulation strategy to generate high definition maps for autonomous vehicles," 2017.

[14] J. de Jesús Rico Jimenez, J. J. G. Barbosa, J. B. H. Ramos, F. J. O. Rodríguez, D.-E. H. Garcia, and R. Gonzalez-Barbosa, "Digitalizacion del entorno a partir de un LIDAR HDL-64E," *Nexo*, 2012.

[15] C. Huihai, L. Shuqiang, and Z. Yingsheng, "An obstacle detection algorithm used sequential sonar data for Autonomous Land Vehicle," 2011.

[16] A. K. Maini, *Radar Fundamentals*. Wiley, 2016.

[17] O. Boric-Lubecke, V. M. Lubecke, A. D. Droitcour, B.-K. Park, and A. Singh, *Radar Principles*. IEEE, 2016.

[18] A. K. Maini, *Military Laser Systems*. Wiley, 2016.

[19] D. A. Pomerleau, "ALVINN: An Autonomous Land Vehicle In a Neural Network," *Carnegie Mellon University*, 1989.

[20] N. Patel, A. Choromanska, P. Krishnamurthy, and F. Khorrami, "Sensor modality fusion with CNNs for UGV autonomous driving in indoor environments," 2017.

[21] F. Caron, E. Duflos, and P. Vanheeghe, "Introduction of contextual information in a multisensor EKF for autonomous land vehicle positioning," 2005.

[22] P. J. Hargrave, "A tutorial introduction to kalman filtering," in *IEE Colloquium on Kalman Filters: Introduction, Applications and Future Developments*, pp. 1/1–1/6, Feb 1989.

[23] J. Lacambre, M. Narozny, and M. Duplaquet, "The enriched sigma point kalman filter an adaptation of the unscented kalman filter for navigation applications," in *Proceedings of the 16th International Conference on Information Fusion*, pp. 1813–1818, July 2013.

[24] H. Durrant-Whyte, N. Roy, and P. Abbeel, *A Serial Approach to Handling High-Dimensional Measurements in the Sigma-Point Kalman Filter*. MITP, 2012.

[25] Dutton, Thompson, and Barraclough, *The Art of Control Engineering*. Addison-Wesley, 1997.

[26] N.S.Nise, *Control Systems Engineering*. John Wiley and Sons, 6 ed., 2011.

[27] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," 2017.

[28] R. H. Brown, S. C. Schneider, and M. G. Mulligan, "Analysis of algorithms for velocity estimation from discrete position versus time data," *IEEE Transactions on Industrial Electronics*, vol. 39, pp. 11–19, Feb 1992.

[29] M. Faccio, P. Grande, F. Parasiliti, R. Petrella, and M. Tursini, "An embedded system for position and speed measurement adopting incremental encoders," in *Conference Record of the 2004 IEEE Industry Applications Conference, 2004. 39th IAS Annual Meeting.*, vol. 2, pp. 1192–1199 vol.2, Oct 2004.

[30] A. Romero, A. J. Muñoz-Ramírez, and J. M. G. de Gabriel, "Realimentación de velocidad con encoders de baja resolución en simulink," 2015.

[31] J. Parada Puig and P. A Lischinsky, "Mínimos cuadrados recursivos para el control adaptativo a tiempo real de un péndulo," pp. EC85–EC91, 01 2010.

[32] X. Chen, "Recursive least-squares method with membership functions," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, vol. 3, pp. 1962–1966 vol.3, Aug 2004.

[33] W. Xu and F. Liu, "Recursive algorithm of generalized least squares estimator," in *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, vol. 3, pp. 487–490, Feb 2010.

[34] Q. Chen, Y. Gu, and F. Ding, "Data filtering based recursive least squares estimation algorithm for a class of wiener nonlinear systems," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pp. 1848–1852, June 2014.

[35] S. Gurin, "Algoritmos de ordenación," 2004.

[36] Q. Jia, M. Wang, S. Liu, J. Ge, and C. Gu, "Research and development of mecanum-wheeled omnidirectional mobile robot implemented by multiple control methods," in *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pp. 1–4, Nov 2016.

[37] Y. Liu, H. Li, L. Ding, L. Liu, T. Liu, J. Wang, and H. Gao, "An omnidirectional mobile operating robot based on mecanum wheel," in *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 468–473, Aug 2017.

[38] D. Peña Sánchez de Rivera, *Deducción de distribuciones: el método de Monte Carlo*. Alianza Editorial, 2001.

[39] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.